MULTISCALE A POSTERIORI ERROR ESTIMATION AND MESH ADAPTIVITY FOR RELIABLE FINITE ELEMENT ANALYSIS

By

AHMED H. ELSHEIKH, M.A.Sc.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfillment of the requirements
for the Degree

Doctor of Philosophy

McMaster University

Your file Votre référence
ISBN: 978-0-494-28138-3
Our file Notre référence
ISBN: 978-0-494-28138-3

MULTISCALE A POSTERIORI ERROR ESTIMATION AND MESH ADAPTIVITY FOR RELIABLE FINITE ELEMENT ANALYSIS

Doctor of Philosophy (2007) McMaster University

(Civil Engineering) Hamilton, Ontario

TITLE: Multiscale a posteriori error estimation and mesh adaptivity for reliable finite element analysis

AUTHOR: Ahmed H. ElSheikh, M.A.Sc. (McMaster University)

SUPERVISORS: Dr. Samir E. Chidiac

Dr. Spencer Smith

NUMBER OF PAGES: 178 pages (i-viii, 1-170)

Abstract

The focus of this thesis is on reliable finite element simulations using mesh adaptivity based on *a posteriori* error estimation. The accuracy of the error estimator is a key step in controlling both the computational error and simulation time. The estimated errors guide the mesh adaptivity algorithm toward a quasi-optimal mesh that conforms with the solution specific features. The simulation time is controlled by minimizing the needed computational resources iteratively through adaptive mesh refinement.

Analysis of existing local a posteriori error estimation techniques is the focus of the first part of this thesis. The Element Residual Method (ERM) is analyzed and numerically tested in comparison to the Zienkiewicz-Zhu (ZZ) error estimator. Steady state flow (diffusion) problems, elasticity problems and advection-diffusion problems are used as numerical test cases. It is shown that the ERM provides better error estimation in comparison to the ZZ error estimator, as the ERM accounts for all terms of the solution residual in the domain interior and on the domain boundary. However, it is observed that the ERM does not produce reliable results for problems solved on very coarse meshes and for problems with points of singularity or boundary layers in the solution. This is attributed to the averaging assumption used for prescribing an artificial boundary condition on the local problems. This assumption is only correct for problems with smooth solution and when the sharp layers and solution specific features are completely resolved by the mesh.

To overcome the limitations of the ERM, a new framework for error estimation based on the variational multiscale method is proposed. The basic idea of evaluating the residual equation locally is coupled with the Variational Multiscale Method (VMS), to design a general framework for local error estimation. The VMS introduces a decomposition of the solution into a resolved components (captured by the mesh)

and an unresolved component (subgrid scales). The VMS decomposition produces a natural variational formulation of the unresolved scale (error). This fine scale variational equation is localized to derive different local error estimation techniques. A new Subdomain Residual Method (SRM) is developed using a partition of unity as the localization operator. The subdomain estimator is flux free and does not introduce artificial boundary conditions to the local problems. It is easy to implement and efficiently provides both upper and lower bound of the error in the energy norm. Numerical results show that the proposed SRM outperforms the ERM and produces very sharp error estimation on coarse meshes.

Acknowledgements

First of all, I am grateful to my supervisors; Dr. Samir Chidiac and Dr. Spencer Smith. They gave me the opportunity to grow as a researcher and treated me more like a research partner than their student. I really appreciate their guidance and encouragement throughout my doctoral study.

I would like to thank the members of my advisory committee; Dr. Dieter Stolle, Dr. Nikolas Provatas and Dr. Alan Wassyng; for their continuous interest in my research and for their constructive criticism on the progress of my thesis.

I gratefully acknowledge the financial support from the Natural Sciences and Engineering Research Council (NSERC) of Canada, the Ontario Graduate Scholarship (OGS) program, the McMaster University Centre for Effective Design of Structures and the Department of Civil Engineering at McMaster University.

Finally and above all, I want to express my love and deepest gratitude to my parents. To them I dedicate this thesis.

This thesis consists of the following papers

Paper I:

ElSheikh, A.H., Chidiac, S.E. and Smith, S., *"Assessment of two a posteriori error estimators for FEM. Part I: Steady-state flow,"* (Submitted)

Paper II:

ElSheikh, A.H., Smith, S. and Chidiac, S.E., *"Assessment of two a posteriori error estimators for FEM. Part II: Elasticity,"* (Submitted)

Paper III:

ElSheikh, A.H., Smith, S. and Chidiac, S.E., *"Numerical investigation of the reliability of a posteriori error estimation for advection diffusion equations,"* Accepted in Communications in Numerical Methods in Engineering.

Paper IV

ElSheikh, A.H., Chidiac, S.E. and Smith, S., *"A posteriori error estimation based on numerical realization of the variational multiscale method,"* (To be submitted)

Thesis related paper (included in the appendix)

ElSheikh, A.H., Smith, S. and Chidiac, S.E., *"Semi-formal design of reliable mesh generation systems,"* Advances in Engineering Software, v.35, n.12, 827-841, 2004.

Contents

Chapter 1
Thesis Summary

1.1 Introduction

The topics discussed in this chapter are essential for understanding, deriving and implementing multiscale *a posteriori* error estimation techniques. These topics are presented for completeness. This chapter starts by stating the motivation for the thesis and objectives, followed by a short introduction to the finite element method. Different *a posteriori* error estimation techniques for the finite element method are presented followed by an introduction to the variational multiscale method and its relation to stabilization techniques for advection-diffusion problems. A brief review of different adaptive mesh refinement techniques employed in the numerical studies is provided. Finally, a summary of the papers included in this thesis is presented followed by the concluding remarks and some suggested ideas for future research.

1.1.1 Context and motivation

Numerical simulation is an integral part of modern engineering design practice and accordingly, the reliability of the numerical simulation plays a key role in the adequacy and competitiveness of the engineering product. Reliable computational methods are those capable of quantifying and controlling the errors in the numerical solution by adapting the available resources. With this definition in mind, the reliability of the numerical simulation contributes both to the quality of the products and to the production cost by shortening the development cycle.

Many engineering and physical processes are modelled using a set of Partial Dif-

ferential Equations (PDEs). These PDEs are discretized over the domain of interest with the aid of numerical methods, such as the Finite Difference Method (FDM), the Finite Element Method (FEM) and the Finite Volume Method (FVM). The resulting system of linear or nonlinear equations can be solved directly or iteratively. The solution of many practical problems may exhibit sharp interfaces, moving fronts and/or points of singularity. These special situations render reliability assessment based on convergence studies using globally refined discretization (meshes) ineffective.

Development of reliable simulation techniques is tightly related to the concepts of verification and validation. According to Roache [1], *verification* corresponds to assessing the quality of the numerical solution by quantifying the errors in the *computational model*, while *validation* corresponds to checking the *mathematical model* and how well it describes the physics of the problem (equations, boundary condition, etc.). Verification of the computational model should precede the validation step, or any comparison with physical experiments. The current work focuses on verification of the computational model, which is a key step in the engineering design process. The validation step is usually addressed by the design engineer or by using model adaptivity techniques [2].

In this thesis the focus falls on developing reliable *a posteriori* error estimation techniques that can be used to quantify the errors in the computational model and to guide a mesh adaptivity iteration. Different classes of partial differential equations are studied to provide the insight and experience necessary to form the kernel of a multiphysics simulation toolbox.

1.1.2 Thesis objectives and scope

In the context of verification and validation, the interest is in the verification of the numerical results. One wants to evaluate the errors due to discretization, with the ultimate goal of providing an error bound on the calculated results. This goal can be achieved through two steps: *i)* error estimation and, *ii)* simulation reliability improvement. Error estimation for a given mathematical model is a way to obtain

an assessment of the accuracy of a specified output provided by a numerical method. Solving a problem reliably means being able to produce a solution that meets a prescribed tolerance by controlling the estimated error (due to discretization) using mesh adaptivity.

The problems dealt with in this thesis are limited to two classes of linear PDEs: diffusion equations and advection-diffusion equations. Error estimation techniques are limited to energy norm error estimates, which are needed to produce quasi-optimal meshes by mesh adaptivity. Moreover, energy norm error estimates play an essential role in error estimation in quantities of interest to obtain bounds for different functional outputs.

1.1.2.1 The main objectives of this thesis are:

- ▷ Evaluate and analyze implicit *a posteriori* error estimation techniques for steady-state flow problems, elasticity problems and advection-diffusion problems.
- ▷ Identify deficiencies with implicit *a posteriori* error estimation techniques.
- ▷ Propose a framework for local error estimation based on a numerical realization of the variational multiscale method. The framework should be a generalization of residual based error estimation techniques and should advance the understanding on how the solution captured by a certain mesh interacts with subgrid scales.
- ▷ Formulate new error estimation techniques based on the proposed framework. These error estimation techniques should be local, inexpensive to compute and minimal with respect to the assumptions being made. The goal for the developed error estimator is to outperform the state-of-the-art local implicit error estimation techniques.
- ▷ Implement and test the proposed local error estimation techniques within a mesh adaptivity algorithm on relevant test problems.

1.2 The finite element method

The finite element method is a well established mathematical method for solving PDEs. The finite element method can be used to solve steady-state elliptic problems as well as transient parabolic and hyperbolic problems. One of the major advantages of the finite element method over other numerical methods includes the ability to handle complex domains. A complete treatment of the subject can be found in many references [3, 4, 5, 6, 7].

To introduce the Galerkin finite element method, a simple elliptic equation is considered. The problem domain Ω is a bounded domain in $\mathbb{R}^2$ with a Lipschitz-continuous boundary $\partial\Omega$ composed of a Dirichlet portion Γ_D and Neumann portion Γ_N, where $\partial\Omega = \Gamma_D \cup \Gamma_N$ and $\Gamma_D \cap \Gamma_N = \varnothing$. The PDE under consideration is written in an abstract format as:

Find u such that:

$$\begin{cases} Lu = f & \text{in } \Omega \\ u = u_D & \text{on } \Gamma_D \\ \frac{\partial u}{\partial \mathbf{n}} = g & \text{on } \Gamma_N \end{cases} \tag{1.1}$$

where $Lu = -\triangle u$ is the linear second order differential operator, $f \in L_2(\Omega)$ is the prescribed loading function and $\mathbf{n}$ is the outward unit normal vector to Γ_N. The boundary data u_D and g are assumed to be sufficiently smooth. Using standard norm notation for Sobolev and Hilbert spaces, as in Reference [4], and equipped with scalar L^2 inner product $(.,.)$, the Galerkin weak formulation of Equation 1.1 can be written as:

$$\begin{cases} \text{Find } u \in U \text{ such that} \\ \mathcal{B}(u,v) = (f,v) + \langle g, v\rangle_{\Gamma_N} \qquad \forall v \in V \end{cases} \tag{1.2}$$

where U is the trial space and V is the test space defined as:

$$U = \{u \in H^1(\Omega) : u|_{\Gamma_D} = u_D\}, \quad V = \{v \in H^1(\Omega) : v|_{\Gamma_D} = 0\} \tag{1.3}$$

and $\mathcal{B}(u,v) = \int_\Omega (\nabla u \cdot \nabla v)$, $(f,v) = \int_\Omega f\ v$ and $\langle g, v\rangle = \int_{\Gamma_N} g\ v$. $H^1(\Omega)$ denotes

the standard Sobolev space. The bilinear $\mathcal{B}$ is bounded; i.e., there exists a constant M such that $|\mathcal{B}(u,v)| \leq M\|u\|_V\|v\|_V$, $\forall u, v \in V$; and V-elliptic; i.e., there exists a constant c_0 such that $\mathcal{B}(v,v) \geq c_0\|v\|_V^2$, $\forall v \in V$.

The functions u and v are approximated by simpler functions u_h and v_h belonging to a finite dimensional space $V_h \subset V$ defined by a mesh size parameter h, corresponding to the domain discretization. The discretized domain contains a set of partitions $\mathcal{T}_h$ of the domain Ω with a set of nodes $\mathcal{N}_h$ and edges $\mathcal{E}_h$ [5]. The partitions are either triangles or convex quadrilaterals with the number of elements defined by the cardinality of the set of partitions $|\mathcal{T}_h|$. Similarly, the number of edges and nodes are $|\mathcal{E}_h|$ and $|\mathcal{N}_h|$. An element in the mesh is denoted by K, an edge is denoted by E and a mesh node (vertex) is denoted by N. The discretized domain Ω_h is defined as

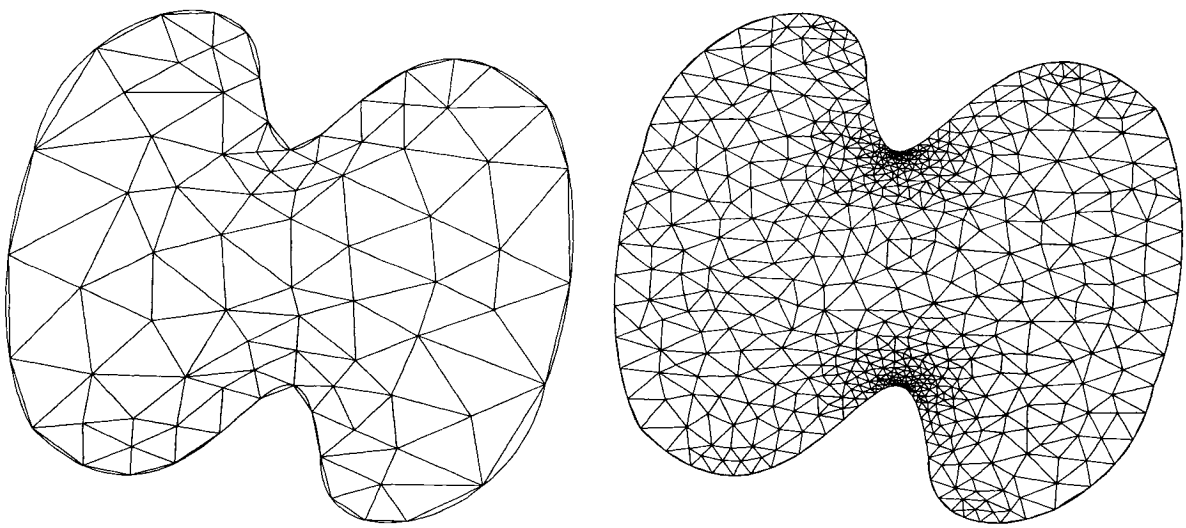
$$\Omega_h = \bigcup_{K \in \mathcal{T}_h} K \tag{1.4}$$

The discretized domain Ω_h is called the geometric interpolation of Ω and does not necessarily coincide with Ω. Figure 1.1 shows a sample triangular discretization (mesh) of a curved domain with different levels of geometric representations. In the rest of the presentation, $\mathcal{T}_h$ is said to be a mesh of the domain Ω neglecting the geometrical error in representing the boundaries. Conforming partitions satisfy the following set of conditions [5]:

1. For all $K_i, K_j \in \mathcal{T}_h$ and $(i \neq j)$, the intersection of the two elements K_i and K_j is either empty, a common edge or a common vertex.

2. For all $K \in \mathcal{T}_h$, there exists a constant γ^* such that $\frac{h_K}{\rho_K} \leq \gamma^*$, where h_K is the diameter of the minimal ball circumscribed around K (for simplicity the longest edge is used) and ρ_K is the diameter of the biggest ball contained in K. This definition is usually referred to as *shape regular discretization.*

Given the domain discretization, a subspace $V_h \subset V$ is defined using the lowest order linear triangular or bilinear rectangular elements. The basis of this space is



(a) (b)

Figure 1.1: Two triangular meshes of a curved domain

the *hat functions* defined by the nodal values $\phi_i(x_j) = \delta_{ij}$. Formally, for a triangular discretization $\mathcal{T}_h$ the approximation space V_h is defined as:

$$V_h = \{v_h \in H^1(\Omega) \quad \wedge \quad \forall K \ \in \mathcal{T}_h, \ v_h|_K \in \mathbb{P}^1(K)\} \tag{1.5}$$

where $\mathbb{P}^1(K)$ is the space of polynomials of degree at most 1 on the triangle K. For a discretization into a set of quadrilaterals the approximation space is defined as:

$$V_h = \{v_h \in H^1(\Omega) \quad \wedge \quad \forall K \ \in \mathcal{T}_h, \ v_h|_K \in \mathbb{Q}^1(K)\} \tag{1.6}$$

where $\mathbb{Q}^1(K)$ is the space of polynomials of complete degree 1 on the mesh element K.

Replacing u and v by their approximations u_h and v_h in Equation 1.2 while using the same space V_h for both the test and trial functions, leads to the Galerkin finite

element formulation:

$$\begin{cases} \text{Find } u_h \in V_h \text{ such that} \\ \mathcal{B}(u_h, v_h) = (f, v_h) + \langle g, v_h \rangle_{\Gamma_N} \qquad \forall v_h \ \in V_h \end{cases} \tag{1.7}$$

Numerically, the approximation space is expressed in terms of the basis functions

$$u_h = \sum_{j=1}^{|\mathcal{N}_h|} c_j \phi_j(x) \qquad v_h = \sum_{j=1}^{|\mathcal{N}_h|} d_j \phi_j(x) \tag{1.8}$$

By back-substitution of u_h and v_h values into the discretized weak formulation, the following system of algebraic equation can be reached

$$\sum_{i=1}^{|\mathcal{N}_h|} \mathcal{B}(\phi_j, \phi_i)\ c_i = (\phi_j, f), \quad j = 1, 2, ..., |\mathcal{N}_h|. \tag{1.9}$$

The coefficients c_i $(i = 1, 2, .., |\mathcal{N}_h|)$ are evaluated by solving the resulting system of equations. The selection of the basis functions ϕ_j as hat functions results in a sparse system of equations with easy to evaluate coefficients.

1.3 A posteriori error estimation techniques

A posteriori error estimates provide a measure to assess the solution accuracy and to drive a mesh adaptation process. *A posteriori* error estimates rely on a sound mathematical theory that has been developed over the last two decades. A review of the subject can be found in [8, 9, 10] and the references therein. Generally speaking, different classifications of error estimators exist. For instance, error estimators can be classified as residual or recovery type. Other classifications include explicit versus implicit, and element based versus patch based.

For the simple boundary value problem defined by Equation 1.1, the error in the

finite element solution u_h is defined as:

$$e = u - u_h \tag{1.10}$$

A priori error estimation for the finite element method of order p shows that the norm of the error $\|e\| = \sqrt{\mathcal{B}(e,e)}$ is related to mesh size by [4]:

$$\|e\| \leq C(u)\ h^p \tag{1.11}$$

where $C(u)$ depends on the unknown solution u. This constant makes a priori estimates impractical to evaluate.

A posteriori error estimates use the residual of the finite element solution $\mathcal{R}_h(u_h)$ to evaluate the error. This residual is defined as

$$\mathcal{R}_h(v) = (f,v) + \langle g, v\rangle_{\Gamma_N} - \mathcal{B}(u_h, v) \qquad \forall v \ \in V \tag{1.12}$$

where it is related to the error e by

$$\mathcal{R}_h(v) = \mathcal{B}(u,v) - \mathcal{B}(u_h,v) = \mathcal{B}(e,v) \qquad \forall v \ \in V \tag{1.13}$$

Putting $v = v_h$ as $v_h \in V_h \subset V$ in the previous equation one gets,

$$\mathcal{R}_h(v_h) = \mathcal{B}(u,v_h) - \mathcal{B}(u_h,v_h) = \left\{(f,v_h) + \langle g, v_h\rangle_{\Gamma_N}\right\} - \left\{(f,v_h) + \langle g, v_h\rangle_{\Gamma_N}\right\} = 0 \tag{1.14}$$

This equation yields the Galerkin orthogonality property of the error where $\mathcal{R}_h(v_h) = \mathcal{B}(e,v_h) = 0$ for all $v_h \ \in V_h$. Equation 1.13 is a global equation of the error e, which can be evaluated by solving a new problem on the same domain Ω as the original problem. To overcome the orthogonality property, the approximation space of the test function v must be larger than V_h used for the initial finite element solution.

This leads to an extrapolation method where the problem is solved again using a globally refined mesh or a higher order approximation.

The main goal of an *a posteriori* error estimator is to divide the global residual problem into a set of local decoupled problems that are approximated explicitly or implicitly. The global error is then evaluated as the sum of the local errors in a certain norm [10].

In Equation 1.13, if the integration domain is divided into a set of elements, then one can rewrite the residual equation as

$$\mathcal{B}(e,v) = \sum_{K \in \mathcal{T}_h} \left\{ \int_K f\, v + \int_{\partial K \cap \Gamma_N} g\, v - \int_K \nabla u_h \cdot \nabla v \right\} \tag{1.15}$$

where ∂K is the boundary of the mesh element K. By integrating by parts and some re-arrangements, one obtains

$$\mathcal{B}(e,v) = \sum_{K \in \mathcal{T}_h} \int_K r\, v + \sum_{E \in \mathcal{E}_h} \int_E R\, v \tag{1.16}$$

where $r = f + \Delta u_h$ is the element residual and R, the edge residual, is defined as:

$$R = \begin{cases} \mathbf{n}_E \cdot \{(\nabla u_h)_K - (\nabla u_h)_{K'}\} & \text{on } E \subseteq (\partial K \cap \partial K') \\ g - \frac{\partial u_h}{\partial \mathbf{n}_E} & \text{on } E \subseteq (\partial K \cap \Gamma_N) \end{cases} \tag{1.17}$$

where K and K' are two adjacent elements sharing an edge E and $\mathbf{n}_E$ denotes a unit outward normal vector to the edge E. The residual equation in the form of Equation 1.16 is the basis for the derivation of different error estimation techniques. In the following subsections, different error estimation techniques will be briefly reviewed. The review will cover explicit, recovery and residual error estimation techniques. Other error estimation techniques based on adjoint problem, error estimation in quantities of interest or error estimation based on the complementary energy are not considered.

1.3.1 Explicit error estimates

Explicit estimators were first introduced by Babuška and Rheinboldt [11] as an efficient method to obtain an estimation of the error. Mathematical manipulation of Equation 1.16 with the aid of Clément interpolation $\Pi_h : V \rightarrow V_h$ are performed to obtain an explicit formula for the error norm. The Clément type interpolation operator $\Pi_h : V \rightarrow V_h$ has the following properties [12]:,

$$\forall v \in V, \qquad \Pi_h v \in V_h \tag{1.18}$$

$$\|v - \Pi_x v\|_{L_2(K)} \leq C\ h_K \|v\|_{H^1(\mathcal{P}(K))} \tag{1.19}$$

$$\|v - \Pi_x v\|_{L_2(E)} \leq C\ h_E^{1/2} \|v\|_{H^1(\mathcal{P}(E))} \tag{1.20}$$

where C is a constant, $\mathcal{P}(K)$ is a patch of elements having a common edge or common vertex with the element $K \in \mathcal{T}_h$, $\mathcal{P}(E)$ is a patch of elements having a common vertex with the edge $E \in \mathcal{E}_h$, and $h_E = |E|$ is the edge length. $L_2(\cdot)$ and $H^1(\cdot)$ are the standard L_2 and Sobolev norms, respectively. From the Galerkin orthogonality and substituting $\Pi_h e$ for v in Equation 1.16, one obtains

$$\mathcal{B}(e, \Pi_h e) = \sum_{K\ \in\ \mathcal{T}_h} \int_K r\ \Pi_h e + \sum_{E\ \in\ \mathcal{E}_h} \int_E R\ \Pi_h e = 0 \tag{1.21}$$

Replacing v by e in Equation 1.16 and then subtracting the previous equation results in

$$\|e\|^2 = \mathcal{B}(e, e) = \sum_{K\ \in\ \mathcal{T}_h} \int_K r\ (e - \Pi_h e) + \sum_{E\ \in\ \mathcal{E}_h} \int_E R\ (e - \Pi_h e) \tag{1.22}$$

Applying the Cauchy-Schwarz inequality (i.e. $(u, v) \leq \|u\|_{L_2} \|v\|_{L_2}$) one obtains

$$\mathcal{B}(e, e) \leq \sum_{K\ \in\ \mathcal{T}_h} \left\{ \|r\|_{L_2(K)}\ \|(e - \Pi_h e)\|_{L_2(K)} \right\} + \sum_{E\ \in\ \mathcal{E}_h} \left\{ \|R\|_{L_2(E)}\ \|(e - \Pi_h e)\|_{L_2(E)} \right\} \tag{1.23}$$

Using the Clément interpolation properties listed above results in

$$\mathcal{B}(e,e) \leq C_1 \sum_{K \in \mathcal{T}_h} \left\{ \|r\|_{L_2(K)} \, h_K \|e\|_{H^1(\mathcal{P}(K))} \right\} + C_2 \sum_{E \in \mathcal{E}_h} \left\{ \|R\|_{L_2(E)} \, h_E^{1/2} \|e\|_{H^1(\mathcal{P}(E))} \right\} \tag{1.24}$$

From the special case of Cauchy-Schwarz inequality $(\sum \alpha_i \beta_i) \leq (\sum \alpha_i^2)^{1/2} (\sum \beta_i^2)^{1/2}$, where α_i denote the entries $\|e\|_{H^1(\mathcal{P}(K))}$ and $\|e\|_{H^1(\mathcal{P}(E))}$ and β_i denote the entries $\|r\|_{L_2(K)} \, h_K$ and $\|R\|_{L_2(E)} \, h_E^{1/2}$, one gets

$$\|e\|^2 = \mathcal{B}(e,e) \leq \|e\|_{H^1(\Omega)} \left\{ C_1 \sum_{K \in \mathcal{T}_h} \|r\|^2_{L_2(K)} \, h_K^2 + C_2 \sum_{E \in \mathcal{E}_h} \|R\|^2_{L_2(E)} \, h_E \right\}^{1/2} \tag{1.25}$$

Squaring both sides and using the coercivity (V-ellipticity) of the bilinear, one gets

$$\|e\|^2 = \mathcal{B}(e,e) \leq \left\{ C_1 \sum_{K \in \mathcal{T}_h} \|r\|^2_{L_2(K)} \, h_K^2 + C_2 \sum_{E \in \mathcal{E}_h} \|R\|^2_{L_2(E)} \, h_E \right\} \tag{1.26}$$

This explicit error estimator is computed locally using the element residual and edge residuals (Equation 1.17).

1.3.2 Recovery error estimates

For many practical problems, the gradient of the finite element solution is of more interest than the solution itself. For example, in elasticity problems stress and strain fields are often more important than the displacement field. However, the gradient of the finite element solution is a discontinuous field. Having a jump in the solution gradient is one known type of element residual at the boundaries that has been treated extensively in the post-processing and visualization of the finite element results. Post-processing can be done explicitly by simple averaging of the discontinuous field to obtain a continuous one or it can be done implicitly by solving a local problem based on surface fitting. Error estimation based on recovery techniques approximates the error as the difference between the smoothed gradient field to the untreated gradient

field resulting from the original finite element solution. The error in the finite element solution using the energy norm is evaluated as

$$\|e\|^2 = \|\nabla u - \nabla u_h\|_{L_2(\Omega)} = \int_\Omega \Big(\nabla u - \nabla u_h\Big)^2 \tag{1.27}$$

However, the true gradient is not known and a recovered gradient $\mathcal{G}_h(u_h)$ using the recovery operator $\mathcal{G}_h : V_h \rightarrow V_h \times V_h$ is assumed to be a good approximation of ∇u. In that case, the estimated error norm is evaluated as

$$\|e\|^2 \approx \eta^2 = \int_\Omega \Big(\mathcal{G}_h(u_h) - \nabla u_h\Big)^2 \tag{1.28}$$

The properties of the operator $\mathcal{G}_h$ are a key factor in the accuracy of the estimator. The recovery operator should satisfy the following three properties: *i)* consistency, *ii)* locality and *iii)* linearity and boundedness [10]. The recovery operator is consistent if

$$\mathcal{G}_h(\Pi_h v) = \Pi_h \nabla v \quad \text{on K} \qquad \forall\, v \in \mathbb{P}^{p+1}(\mathcal{P}(K)) \tag{1.29}$$

where Π_h is a nodal interpolation operator into the finite element space V_h, $\mathcal{P}(K)$ the patch associated with an element K, and u is assumed to belong to the polynomial space of the order $p+1$. The recovery operator $\mathcal{G}_h$ is local if the value of the recovered gradient $\mathcal{G}_h(v)$ at a point $p(x,y) \in K$ depends only on values of ∇v sampled on $\mathcal{P}(K)$. The operator $\mathcal{G}_h$ is bounded if there exists a constant C, independent of the mesh size h, such that

$$\|\mathcal{G}_h(v_h)\|_{L_\infty(K)} \leq C\|\nabla v_h\|_{L_\infty \mathcal{P}(K)} \qquad \forall\, K \in \mathcal{T}_h \qquad \forall\, v_h \in V_h \tag{1.30}$$

The properties of the recovery operator $\mathcal{G}_h(\Pi_h u)$ imply that it is inexpensive to compute and that it provides a good approximation of ∇u if u is a smooth function. Given that the solution itself u is unknown, the recovery operator is applied to u_h

instead of the solution interpolant $\Pi_h u$. In some special cases, ∇u_h converges towards $\nabla(\Pi_h u)$ faster than it does towards ∇u. This property is called the superconvergence phenomenon and it is present if the following condition holds for positive constant $C(u)$ and $\tau \in (0, 1]$ that are independent of the mesh size h

$$\|\nabla u_h - \nabla(\Pi_h u)\|_{L_2(\Omega)} \leq C(u)\ h^{p+\tau} \tag{1.31}$$

where p is the order of the finite element (compare to Equation 1.11). Superconvergence occurs in special cases and a survey of superconvergence results can be found in [13].

The most well known recovery type error estimator was developed by Zienkiewicz and Zhu [14, 15], and it is based on the superconvergent patch recovery. The accuracy of the solution gradient ∇u_h at some points inside the finite elements is exceptionally accurate (superconvergent). These accurate gradients are used to reconstruct (recover) a continuous gradient field $\mathcal{G}_h(u_h)$, which is an accurate approximation to the exact solution gradient ∇u. Midpoints of one-dimensional elements or the barycenters of first order triangular elements are example of these optimal points. Using the solution at these points to feed a local least-squares fitting problem will result in a superconvergent recovered gradient field.

For problems in $\mathbb{R}^2$, a linear approximation of the gradient over a patch of elements has the form $p(x, y) = a_0 + a_1 x + a_2 y$. The local discrete least-squares fitting coefficients (a_i's) that recover the x component of the gradient field are given by:

$$\begin{pmatrix} m & \sum_j x_j & \sum_j y_j \\ \sum_j x_j & \sum_j x_j^2 & \sum_j x_j\, y_j \\ \sum_j y_j & \sum_j x_j\, y_j & \sum_j y_j^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} \sum_j \partial u_h(x_j, y_j)/\partial x \\ \sum_j x_j\ \partial u_h(x_j, y_j)/\partial x \\ \sum_j y_j\ \partial u_h(x_j, y_j)/\partial x \end{pmatrix} \tag{1.32}$$

where $m = \sum_j 1$ is the total number of sampling points. This results from minimizing

the squared sum of the difference between the recovered gradient p and the solution gradient at the superconvergent points $\partial u_h(x_j, y_j)/\partial x$.

1.3.3 Residual error estimates

Residual error estimation techniques involve solving a set of local problems and thus are classified as implicit methods. Error estimation based on residuals is considered to be the most accurate *a posteriori* error estimate because it accounts for all terms in the residual equation and because it is constant free. All residual type estimators are based on localizing Equation 1.13. The domain type of the local problem defines the estimator.

1.3.3.1 Element residual method

In the Element Residual Method (ERM), the residual is localized over each mesh element and local element based problems with Neumann boundary conditions are evaluated. The weak formulation of the ERM starts from Equation 1.13 over an element

$$\mathcal{B}_K(e, v) = (f, v)_K - \mathcal{B}_K(u_h, v) + \langle g_u, v\rangle_{\partial K} \qquad \forall v \in V \tag{1.33}$$

where $\mathcal{B}_K$ is a restriction of the bilinear over an element K, $(.,.)_K$ is a restriction of the L^2 norm over the element K, and g_u is the exact normal flux acting on the element boundary ∂K. On an edge $E \subset \partial K$, the flux is specified as $g_u|_E = \mathbf{n}_E \,.\, (\nabla u)$. As the exact solution is not known, the boundary flux is approximated by averaging the finite element gradients at the element boundaries. The Neumann boundary term $g_{erm}|_E \approx g_u|_E$ is specified as:

$$g_{erm}|_E = \begin{cases} \frac{1}{2}\,\mathbf{n}_E \,.\, \{(\nabla u_h)_K + (\nabla u_h)_{K'}\} & \text{on } E \subseteq (\partial K \cap \partial K') \wedge E \nsubseteq \partial\Omega \\ \mathbf{n}_E \,.\, (\nabla u_h)_K & \text{on } E \subseteq (\partial K \cap \Gamma_D) \\ g & \text{on } E \subseteq (\partial K \cap \Gamma_N) \end{cases} \tag{1.34}$$

The local problems are solved using an enriched space V_b, where second order bubble functions are usually used as the basis functions. Bubble functions are functions with compact support, where each bubble has a value in the interior of an entity but vanishes on the exterior of the entity [10]. For a reference quadrilateral element $\widehat{K}$, with the local coordinates $(-1 \leq \hat{x} \leq 1, -1 \leq \hat{y} \leq 1)$, the edge bubble functions are given by $\chi_1 = 0.5(1-\widehat{x}2)(1-\widehat{y})$; $\chi_2 = 0.5(1+\widehat{x})(1-\widehat{y}2)$; $\chi_3 = 0.5(1-\widehat{x}2)(1+\widehat{y})$; $\chi_4 = 0.5(1-\widehat{x})(1-\widehat{y}2)$; and the element interior bubble function is given by $\chi_5 = (1-\widehat{x}2)(1-\widehat{y}2)$ [10]. Figures 1.2 and 1.3 show a plot of edge bubble functions and element interior bubble function on a reference quadrilateral element, respectively.

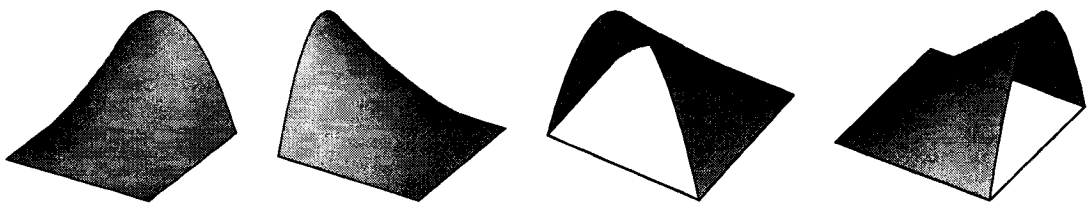


Figure 1.2: Edge bubble functions for a reference quadrilateral element

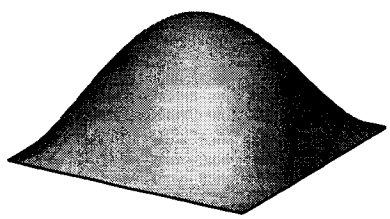
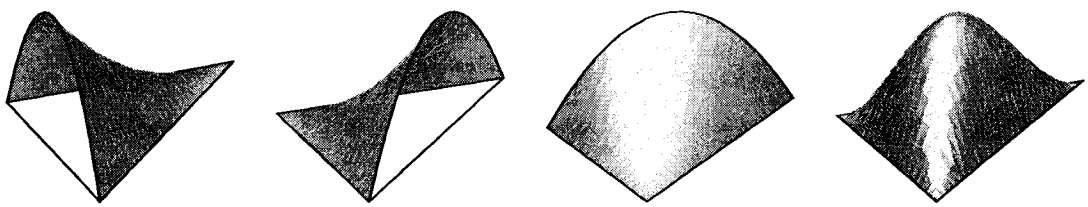


Figure 1.3: Interior bubble function for a reference quadrilateral element

For a reference triangular element $\widehat{K}$, with the barycentric (area) coordinates $\widehat{\lambda_1}, \widehat{\lambda_2}, \widehat{\lambda_3}$, the bubble functions are given by, $\chi_1 = 4\widehat{\lambda_2}\widehat{\lambda_3}$; $\chi_2 = 4\widehat{\lambda_1}\widehat{\lambda_3}$; $\chi_3 = 4\widehat{\lambda_1}\widehat{\lambda_2}$; $\chi_4 = 27\widehat{\lambda_1}\widehat{\lambda_2}\widehat{\lambda_3}$. Figure 1.4 shows a plot of bubble functions for a reference triangular element. The local errors are evaluated over each element and denoted as $\widetilde{e}_K$. The bubble space is defined as $V_b = \text{ span } \{\widehat{\lambda_i}\}$, where λ_i denotes the different bubble functions. The ERM weak formulation with the boundary conditions specified by



(a) (b) (c) (d)

Figure 1.4: Bubble functions for a reference triangular element

Equation 5.23 is:

$$\begin{cases} \text{Find } \widetilde{e}_K \in V_b \text{ such that} \\ \mathcal{B}_K(\widetilde{e}_K, v_b) = (f, v_b)_K - \mathcal{B}_K(u_h, v_b) + \langle g_{erm}, v_b \rangle_{\partial K} \qquad \forall v_b \in V_b \end{cases} \tag{1.35}$$

The local error estimator η_K on an element K is defined as $\eta_K = \left\{\mathcal{B}_K(\widetilde{e}_K, \widetilde{e}_K)\right\}^{1/2}$ and the global error is estimated as the sum of local element contributions as

$$\|e\| \approx \eta = \left\{ \sum_{K \in \mathcal{T}_h} \eta_K^2 \right\}^{1/2} \tag{1.36}$$

The solvability of the local Neumann problem is not guaranteed unless the edge fluxes are in equilibrium. Flux equilibration is also needed for the estimator to be an upper bound of the error (given that local problems are solved exactly). Different techniques for flux equilibration can be found in [16, 17, 18, 19]. Flux equilibration is difficult to implement for three dimensional problems and is rarely applied for practical problems [20, 21].

1.3.3.2 Subdomain residual methods

The idea of approximating the residual equation over a nodal based patch in the Subdomain Residual Method (SRM) goes back to the work of Babuška and Rheinboldt [22]. Recently, four different variations of the SRM have been proposed [23, 24, 25, 26]. All formulations of SRM start from the global residual Equation 1.13 and use a

partition of unity φ_N, which is the first order Lagrangian basis function at node N. A partition of unity has the following property

$$\sum_{N \in \mathcal{N}_h} \varphi_N = 1 \tag{1.37}$$

Inserting the summation value of a partition of unity into the residual equation one gets,

$$\mathcal{B}(e, v) = \mathcal{R}_h(v) = \mathcal{R}_h(v \sum_{N \in \mathcal{N}_h} \varphi_N) \tag{1.38}$$

If the residual is assumed linear then

$$\mathcal{B}(e, v) = \mathcal{R}_h(v \sum_{N \in \mathcal{N}_h} \varphi_N) = \sum_{N \in \mathcal{N}_h} \mathcal{R}_h(v\ \varphi_N) \tag{1.39}$$

Assuming the linearity of the operator $\mathcal{B}$, a set of $|\mathcal{N}_h|$ global problems needs to be solved. The general form of an SRM estimator requires the solution of the following problem for each $N \in \mathcal{N}_h$

$$\begin{cases} \text{Find } e_N \in V \text{ such that} \\ \mathcal{B}(e_N, v) = \mathcal{R}_h(v\ \varphi_N) \qquad \forall v\ \in V \end{cases} \tag{1.40}$$

where, the total error e is the sum of the contributions of errors e_N resulting from each problem. Solving these $|\mathcal{N}_h|$ global problems is as complex as solving the global residual equation.

In the Babuška and Rheinboldt formulation [22], a localization assumption is introduced where the error due to the residual $\mathcal{R}_h(v\ \varphi_N)$ is assumed to vanish outside the patch $\mathcal{P}(N)$ defined as the patch of elements having a common node N. This resulted in transforming the set of problems defined by Equation 1.40 into a set of local problems instead of global ones. The resulting SRM formulation is: for all $N \in \mathcal{N}_h$ solve the following equation subjected to homogenous Dirichlet boundary conditions

[22]:

$$\begin{cases} \text{Find } e_{\mathcal{P}(N)} \in V_{SRM} \text{ such that} \\ \mathcal{B}(e_{\mathcal{P}(N)}, v) = \mathcal{R}_h(v\ \varphi_N) \qquad \forall v\ \in V_{SRM} \end{cases} \tag{1.41}$$

Solving the local problems in an enriched space using the finite element method results in an approximation of the error denoted by $\widetilde{e}_{\mathcal{P}(N)}$. The global error estimator is evaluated as the sum of local patch contributions.

$$\|e\| \approx \eta = \left\{ \sum_{n \in \mathcal{N}_h} \eta_N^2 \right\}^{1/2} \qquad \text{where} \quad \eta_N = \left\{ \mathcal{B}(\widetilde{e}_{\mathcal{P}(N)}, \widetilde{e}_{\mathcal{P}(N)}) \right\}^{1/2} \tag{1.42}$$

Numerically this estimator significantly underestimates the errors due to the artificial homogeneous boundary conditions prescribed on the local problem boundaries.

In the SRM formulation by Prudhomme *et al.* [25], the bilinear $\mathcal{B}$ in Equation 1.40 is replaced by $\mathcal{B}_{SRM1}$ that corresponds to each node N and is defined as

$$\mathcal{B}_{SRM1}(u, v) = \int_{\Omega} \varphi_N (\nabla u \cdot \nabla v) \tag{1.43}$$

While this bilinear definition is artificial, it leads to localizing the global problems defined by Equation 1.40 into a set of local problems because the bilinear vanishes outside the support of φ_N. Zero Neumann boundaries are imposed on each local problem boundary. The resulting SRM formulation is: for all $N \in \mathcal{N}_h$ solve the following equation subjected to homogenous Neumann boundary conditions [25]:

$$\begin{cases} \text{Find } e_{\mathcal{P}(N)} \in V_{SRM1} \text{ such that} \\ \mathcal{B}_{SRM1}(e_{\mathcal{P}(N)}, v) = \displaystyle\int_{\Omega} \varphi_N (\nabla e_{\mathcal{P}(N)} \cdot \nabla v) = \mathcal{R}_h(v\ \varphi_N) \qquad \forall v\ \in V_{SRM1} \end{cases} \tag{1.44}$$

The solution of these problems is defined up to a constant and some condition on the approximation space V_{SRM1} is specified to fix the constant. For an interior node, the

solution space is defined as:

$$V_{SRM1}(\mathcal{P}(N)) = \left\{ v \in C^1(\mathcal{P}(N)) : \int_{\mathcal{P}(N)} v\ \varphi_N = 0 \right\} \tag{1.45}$$

while for a boundary node, the solution space is defined as:

$$V_{SRM1}(\mathcal{P}(N)) = \left\{ v \in C^1(\mathcal{P}(N)) : v = 0 \text{ on } \partial\mathcal{P}(N) \cap \partial\Omega \right\} \tag{1.46}$$

Each patch based problem is approximated using h-refinement (element division) or p-refinement (higher order interpolation) and the solution is denoted $\widetilde{e}_{\mathcal{P}(N)}$. The global estimated error is calculated as

$$\|e\| \approx \eta = \left\{ \sum_{N \in \mathcal{N}_h} \mathcal{B}_{SRM1}(\widetilde{e}_{\mathcal{P}(N)}, \widetilde{e}_{\mathcal{P}(N)}) \right\}^{1/2} \tag{1.47}$$

If the local problems are solved exactly, the estimated error norm η is an upper bound for the exact error. If one denotes by p_h the order of the finite element used to calculate the solution u_h and p_e to be the extra order used to calculate the local errors $\widetilde{e}_{\mathcal{P}(N)}$, the following relaxed upper bound property is obtained [25]

$$\|e\|^2 \leq \sum_{N \in \mathcal{N}_h} \left\{ \mathcal{B}_{SRM1}(\widetilde{e}_{\mathcal{P}(N)}, \widetilde{e}_{\mathcal{P}(N)}) \right\} + 2 \inf_{v \in V^{p_h+p_e}} \|u - v\| \qquad \forall v \in V^{p_h+p_e} \tag{1.48}$$

In the SRM formulation proposed by Morin *et al.* [24], the norm used corresponding to the mesh node N is defined as:

$$\mathcal{B}_{SRM2}(u, v) = \int_{\Omega} \nabla u \cdot \nabla(v\ \varphi_N) \tag{1.49}$$

Morin *et al.* also integrated the right hand side residual term in Equation 1.40 to produce the flux jumps across the sides of the triangulation. A condition for the

approximation space is set such that for an interior node,

$$V_{SRM2}(\mathcal{P}(N)) = \left\{ v \in H^1(\mathcal{P}(N)) : \int_{\mathcal{P}(N)} v\ \varphi_N = 0 \ \text{ and } \int_{\mathcal{P}(N)} |\nabla v|^2 \varphi_N < \infty \right\} \tag{1.50}$$

while for a boundary node the solution space is defined as

$$V_{SRM2}(\mathcal{P}(N)) = \left\{ v \in H^1(\mathcal{P}(N)) : v = 0 \text{ on } \partial\mathcal{P}(N) \cap \partial\Omega \ \text{ and } \int_{\mathcal{P}(N)} |\nabla v|^2 \varphi_N < \infty \right\} \tag{1.51}$$

The special condition $\int_{\mathcal{P}(N)} v\ \varphi_N = 0$ affects the computation of the error field and cannot be achieved by adding a convenient constant [24].

Carstensen *et al.* [23] proposed an SRM estimator based on solving an interface problem over a nodal based patch. For each node $N \in \mathcal{N}_h$ the local problem is defined by

$$\begin{cases} \text{Find } e_{\mathcal{P}(N)} \in V_{SRM3} \text{ such that:} \\ \mathcal{B}_{SRM3}(e_{\mathcal{P}(N)}, v) = \displaystyle\int_{\mathcal{P}(N)} \varphi_N (f + \triangle u_h)\ v - \int_{E \subset \partial K \in \mathcal{P}(N)} \varphi_N\ J|_E\ v \quad \forall v\ \in V_{SRM} \end{cases} \tag{1.52}$$

where the bilinear is defined as in Equation 1.43 and $J|_E$ is the edge residual on an edge $E \subseteq \partial K \cap \partial K'$, defined by

$$J|_E = \mathbf{n}_E \big[(\nabla u_h)_K - (\nabla u_h)_{K'} \big] \tag{1.53}$$

The approximation space of the local problems used in Equation 1.52 has a constraint of the form $\int_{\mathcal{P}(N)} v = 0$. The estimated error norm is defined as

$$\eta^2 = \sum_{N \in \mathcal{N}_h} \int_{\mathcal{P}(N)} \varphi_N \left\{ \nabla e_{\mathcal{P}(N)} \cdot \nabla e_{\mathcal{P}(N)} \right\} \tag{1.54}$$

This error estimator provides an asymptotic upper and lower bound of the exact error

as following,

$$\|e\|^2 \leq \eta^2 \leq 3\ C^2\|e\|^2 \tag{1.55}$$

where the constant C is evaluated by solving an eigenvalue problem over each patch.

More recently another variation of the SRM was proposed by Parés *et al.* [26], in which the calculated errors from different patches are summed over each element before calculating the error norm. The norm of the sum of the errors is used instead of the sum of the norms. The error estimator is based on solving Equation 1.39 using the standard energy norm. After obtaining an approximate solution $\widetilde{e}_{\mathcal{P}(N)}$ over each patch, the error over each element is calculated as

$$\widetilde{e}_K = \sum_{N_i \subset \partial K} \widetilde{e}_{\mathcal{P}(N_i)} \tag{1.56}$$

The estimated error field $\widetilde{e}_K$ is discontinuous and the global estimate $\sum_{K\in\mathcal{T}_h}\|\widetilde{e}_K\|$ is an upper bound of the error norm $\|e\|$, assuming an accurate solution of the local problems is obtained. To ensure the solvability of the problem, the residual term in the right hand side of Equation 1.39 is replaced by $\mathcal{R}_h(\varphi_N\ (v - \Pi_h v))$, where Π_h is a projection operator from V to V_h.

1.4 The variational multiscale method

In this section, the two-level Variational MultiScale (VMS) method is presented according to Hughes [27, 28]. The variational multiscale method is a procedure for deriving numerical methods capable of capturing the subgrid scales while solving for the coarse scale degrees of freedom only. The VMS is based on the fundamental concept of the scale decomposition, where the solution space is split into coarse and fine parts. This is formalized as an overlapping sum decomposition of the solution space V into a coarse scale subspace $V_c \subset V$ and fine scale subspace $V_f \subset V$, such that

$$V = V_c \oplus V_f \tag{1.57}$$

The space V_c is the standard finite element space, however the fine scale space V_f is infinite. Introducing this decomposing into Equation 1.2, one obtains the following two variational equations: find $u_c \in V_c$ and $u_f \in V_f$ such that

$$\mathcal{B}(u_c, v_c) + \mathcal{B}(u_f, v_c) = (f, v_c) + \langle g, v_c \rangle_{\Gamma_N} \qquad \forall\ v_c \in V_c \tag{1.58}$$

$$\mathcal{B}(u_c, v_f) + \mathcal{B}(u_f, v_f) = (f, v_f) + \langle g, v_f \rangle_{\Gamma_N} \qquad \forall\ v_f \in V_f \tag{1.59}$$

Neglecting the term $\mathcal{B}(u_f, v_c)$ in Equation 1.58 reduces the equation to the standard finite element formulation, where $u_h \approx u_c$. Equation 1.59 is related to the global residual equation used in deriving different error estimation techniques. In the VMS, it is suggested to approximate the fine scale solution u_f analytically in terms of the coarse scale solution to obtain a more accurate coarse scale variational equation. This procedure can be regarded as a static condensation of the fine scale degrees of freedom.

The fine scale components are driven by the residual of the coarse scale solution as shown in Equation 1.59. This relation between fine and coarse scales is abstracted in the form of an integral operator $\mathcal{S}$ to relate the fine scale components to the residual of the coarse scales, such that [28]:

$$u' = \mathcal{S}(L\overline{u} - f) \tag{1.60}$$

where $L\overline{u} - f$ is the residual of the coarse scale solution and $\mathcal{S}$ is an integral operator depending on the fine scale Green's function of the adjoint problem [27]. Using this formula for the second term in Equation 1.59, one gets after integrating by parts:

$$\mathcal{B}(\overline{u}, \overline{v}) + (\mathcal{S}(L\overline{u} - f), L^*\overline{v}) = (f, \overline{v}) \tag{1.61}$$

where L^* is a linear differential operator. For diffusion dominated problems, the term $(\mathcal{S}(L\overline{u} - f), L^*\overline{v})$ is neglected, while for advection diffusion equations, neglect-

ing the subgrid energy results in physical unrealistic oscillations in the coarse scale solution. For advection diffusion equations with the linear operator L defined as, $L\overline{u} = -\varepsilon \triangle \overline{u} + \mathbf{a} \cdot \nabla \overline{u}$, with $\varepsilon > 0$ as the diffusivity constant and $\mathbf{a}$ as the velocity field, Equation 1.61 can be written as

$$\mathcal{B}(\overline{u}, \overline{v}) + \mathcal{ST}(\overline{u}, \overline{v}) = (f, \overline{v}) \tag{1.62}$$

In which the stabilization operator $\mathcal{ST}$ can have many forms depending on the stabilization method. Common stabilization methods are

$$\mathcal{ST}_1(\overline{u}, \overline{v}) = \sum_{K \in \mathcal{T}_h} \delta_K (-\varepsilon \triangle \overline{u} + \mathbf{a} \cdot \nabla \overline{u} - f \ , \ \mathbf{a} \cdot \nabla \overline{v})_K \tag{1.63}$$

$$\mathcal{ST}_2(\overline{u}, \overline{v}) = \sum_{K \in \mathcal{T}_h} \delta_K (-\varepsilon \triangle \overline{u} + \mathbf{a} \cdot \nabla \overline{u} - f \ , \ \mathbf{a} \cdot \nabla \overline{v} - \varepsilon \triangle \overline{v})_K \tag{1.64}$$

$$\mathcal{ST}_3(\overline{u}, \overline{v}) = \sum_{K \in \mathcal{T}_h} \delta_K (-\varepsilon \triangle \overline{u} + \mathbf{a} \cdot \nabla \overline{u} - f \ , \ \mathbf{a} \cdot \nabla \overline{v} + \varepsilon \triangle \overline{v})_K \tag{1.65}$$

where the subscript K denotes an element of the mesh, δ_K is the stabilization coefficient and the bilinear $(.,.)_K$ is restricted by integration over element K. The term $\mathcal{ST}_1$ corresponds to the Streamline Upwind Petrov Galerkin (SUPG) stabilization method [29], $\mathcal{ST}_2$ corresponds to the Galerkin Least Squares (GSL) stabilization method [30] and $\mathcal{ST}_2$ is referred to as the Unusual Stabilized Finite Element Method (USFEM) [31].

The stabilization parameter δ_K is a very important factor for the convergence of the stabilization method. This parameter controls the proper amount of artificial diffusion in the streamline direction to account for the subgrid energy. For simple problems in $\mathbb{R}^1$, with the help of the exact solution of a reference problem, an optimal value of the parameter δ_K is found according to [29]

$$\delta_K = \frac{h_K}{2\ a} \left(\coth(Pe_K) - \frac{1}{Pe_K} \right) \tag{1.66}$$

where h_K is the nodal spacing and Pe_K is the element Peclet number defined as $Pe_K = (a\ h_K)/(2\ \varepsilon)$. For problems in $\mathbb{R}^N$, a generic definition of the stabilization parameter over a mesh element K is [31]

$$\delta_K = \frac{h_K}{2\|a\|_p}\ \xi(Pe_K) \tag{1.67}$$

$$Pe_K = \frac{m_K\ \|a\|_p\ h_K}{2\varepsilon} \tag{1.68}$$

$$\xi(Pe_K) = \begin{cases} Pe_K & 0 \leq Pe_K < 1 \\ 1 & 1 \leq Pe_K \end{cases} \tag{1.69}$$

$$\|a\|_p = \left(\sum_{i=1}^{N}(a_i)^p\right)^{1/p} \qquad 1 \leq p < \infty \tag{1.70}$$

$$m_K = \min\left\{\frac{1}{3}, 2\ C_K\right\} \tag{1.71}$$

The constant C_K depends on the finite element order and satisfies the inequality

$$C_K \sum_K h_K^2 \|\triangle v\|_{0,K}^2 \leq \|\nabla v\|_0^2 \qquad \forall v \in V_h \tag{1.72}$$

1.5 Mesh adaptivity algorithms

Initial meshes used in the finite element method are usually designed to conform with the problem boundaries and to satisfy certain geometric quality measures. These geometric mesh quality measures are suitable for problems with smooth solutions. In these cases, global mesh refinement can be used to control the errors. For problems with special features, (i.e. points of singularity or sharp layers) smaller nodal spacing is required in parts of the domain where the solution gradient is high. Quasi-optimal meshes, where the nodal spacing is conforming with solution features, can be obtained by local mesh refinement techniques.

Adaptive algorithms are either based on h-adaptivity, where mesh elements are divided without changing the order of interpolation over the element, or r-adaptivity,

where mesh nodes are moved to obtain denser meshes where needed, or p-adaptivity, where higher order basis functions are used over the elements with large errors. The current work is limited to h-adaptivity where the estimated errors in the finite element solution are used to guide a mesh adaptivity process. Algorithm 1 shows a general mesh adaptivity iteration.

Algorithm 1: General framework for adaptive mesh refinement algorithm

Data: initial mesh ($\mathcal{T}_{h0}$), boundary conditions, problem dependent constants, error tolerance in the global energy norm (η_{tol}), maximum number of iteration (i_{max})

Result: estimated error in the global energy norm η, solution u_i

$i \leftarrow 0$

while $i < i_{max}$ **do**

- **Solve** the problem on the mesh $\mathcal{T}_{hi}$ and denote the solution by u_i

$$\mathcal{B}(u_i, v_{hi}) = (f, v_{hi}) + \langle g, v_{hi} \rangle_{\Gamma_N} \qquad \forall v_{hi} \in V_{hi}$$

- **Estimate** the local error η_K in the solution u_i over each element $K \in \mathcal{T}_{hi}$

- **Compute** an estimate of the global error $\eta_{\mathcal{T}_{hi}}$ as:

$$\eta_{\mathcal{T}_{hi}} = \left\{ \sum_{K \in \mathcal{T}_{hi}} {\eta_K}^2 \right\}^{\frac{1}{2}}$$

- **IF** $\eta_{\mathcal{T}_{hi}} < \eta_{tol}$ **BREAK**;

- **Deduce** from mesh $\mathcal{T}_{hi}$ and the local error indicators η_K a new discretization $\mathcal{T}_{hi+1}$

- $i \leftarrow i + 1$

end

The initial step of mesh adaptivity is a marking step, where elements with large errors are flagged for refinement. This step is followed by the actual mesh modification. Different marking strategies exist to equidistribute the error over the problem domain. In mathematical terms, it is desirable to produce a mesh with a solution

u_{i+1} such that:

$$\forall K \in \mathcal{T}_{hi+1}, \qquad {\eta_K}^2 \approx \frac{\eta_{tol}^2}{|\mathcal{T}_{hi+1}|} \quad \text{where } \eta_K \text{ is calculated using } u_{i+1}$$

This objective is usually achieved iteratively by refining a subset of the mesh elements at each iteration. A maximum error marking strategy depends on a positive threshold Θ_r, where the set of marked elements for refinement $\mathcal{M}$ is defined as

$$\mathcal{M} = \left\{ K \in \mathcal{T}_h : \eta_K \geq [\Theta_r * \max_{K \in \mathcal{T}_h} \eta_K] \right\} \tag{1.73}$$

For the case of both adaptive refinement and coarsening, two constants Θ_c, Θ_r such that $0 \leq \Theta_c < \Theta_r \leq 1$ are used to define the coarsening and refinement thresholds, respectively. If few mesh elements have significantly larger errors than the rest of the domain, the maximum error marking strategy might lead to refining only few elements at each iterations. This corresponds to an increased number of adaptivity iterations.

Another marking strategy where a certain percentage of the mesh elements are marked at each iteration can be used. This strategy is referred to as the sorting strategy, where the mesh elements are sorted according to the estimated errors and then all the element exceeding the $(1 - \Theta_r)^{th}$ percentile are marked for refinement. For example, if it is desired to mark 10% of the elements, then the elements corresponding to $\eta_K \geq \eta_{(1-\Theta_r)}$ are marked for refinement, where $\eta_{(1-\Theta_r)}$ is defined as the $(1 - \Theta_r)^{th} = 90^{th}$ percentile.

1.5.1 Mesh modification techniques

Local mesh refinement can produce conforming or non-conforming meshes. In conforming meshes the intersection of two mesh elements is either empty, a common edge or a common vertex. Non-conforming meshes contain hanging nodes that can be either eliminated by further subdivision of the neighboring elements or they can

be constrained during the numerical solution. Figure 1.5 shows the local refinement of triangular mesh starting by element marking (left sub-figure), actual refinement by subdivision (center sub-figure) and then elimination of hanging nodes (right sub-figure).

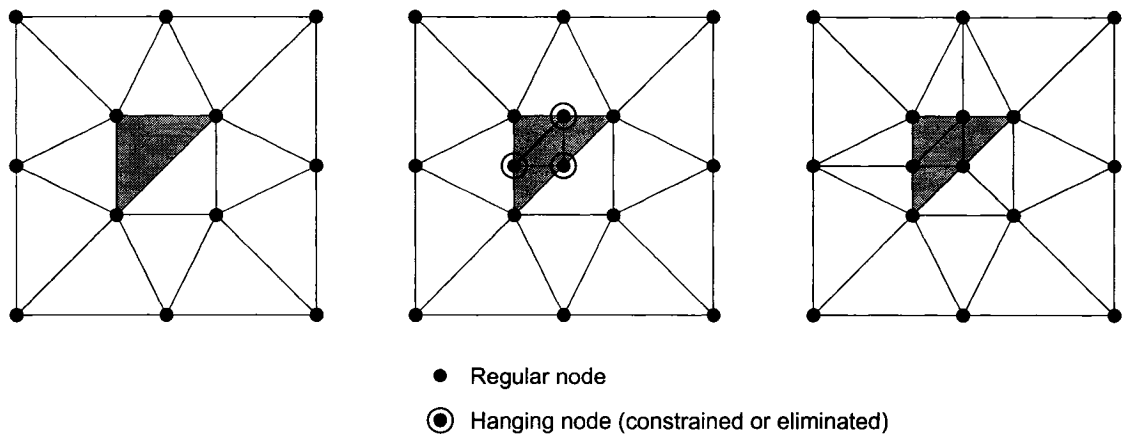


Figure 1.5: Different steps of the mesh refinement algorithm

For triangular meshes elements, there exists a large amount of literature concerning adaptive mesh refinement. Ideally local mesh refinement should not produce mesh elements with very large or very small angles. Generally speaking, a triangle can be divided into four, three or two sub-triangles. Regular refinement (red-refinement) where each triangle is divided into four similar sub-triangles preserves the geometrical properties of the original triangulation but introduces hanging nodes. On the contrary, dividing a triangle into three or two sub-triangles may produce sub-triangles with smaller angles. To avoid that kind of geometrical mesh quality deterioration, Rivara [32] introduced the longest edge bisection algorithm, where triangles are divided only across their longest edge (bisected). Neighboring triangles with hanging nodes are then bisected until there is no remaining hanging nodes. Because each triangle and its descendants are repeatedly bisected across their longest edges, the minimum angle in the adapted mesh is at least one-half the smallest angle in the

original triangulation [33]. Rivara [34, 35] introduced different modifications to the bisection algorithm with the same minimum angle bound.

Another algorithm used for producing conforming meshes is based on regular refinement [36] where all the triangles marked for refinement are divided into four sub-triangle. Then all resulting non-conforming triangular elements with two hanging nodes are divided by regular division. This process is repeated until every triangle has no hanging nodes or only one hanging node. Then a mask division is applied by dividing the set of triangles with one hanging node into two triangles. These elements are deleted before the next refinement iteration. A review of different refinement algorithms, including the newest node bisection algorithm (divide the edge opposite to the newest node), can be found in [37].

The modified longest edge bisection (red-blue-green refinement), as detailed in [8], is employed in parts of this study. The following three patterns of element refinement are defined

Definition 1.5.1. Triangular element refinement of $K \in \mathcal{T}_h$ with edges $E_1, E_2, E_3 \in \mathcal{E}_h$ and $\partial K = E_1 \cup E_2 \cup E_3$ and E_1 the longest edge falls in one of three classes:

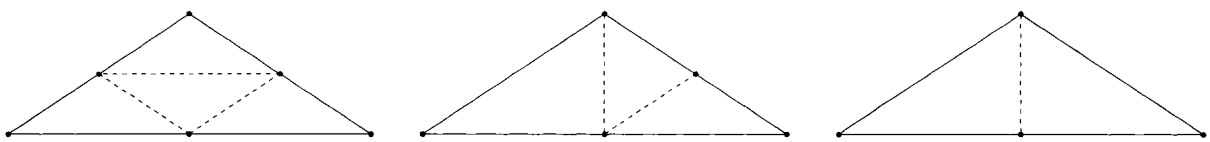
1. A Red-refinement of K is performed by dividing the triangle K into four similar sub-triangles obtained by connecting the midpoints of the edges E_1, E_2, E_3.

2. A Blue-refinement of K is performed by dividing the triangle K into three sub-triangles. First K is bisected along the longest edge E_1 followed by dividing one of the sub-triangles along the edge E_2 or E_3.

3. A Green-refinement of K is performed by dividing the triangle K into two sub-triangles obtained by bisecting the triangle along the longest edge E_1.

Figure 1.6 illustrates the different refinement patterns. All elements marked for refinement are divided using red-refinement followed by conformity enforcement iteration. If an element has two hanging nodes and none of them is on the longest edge,

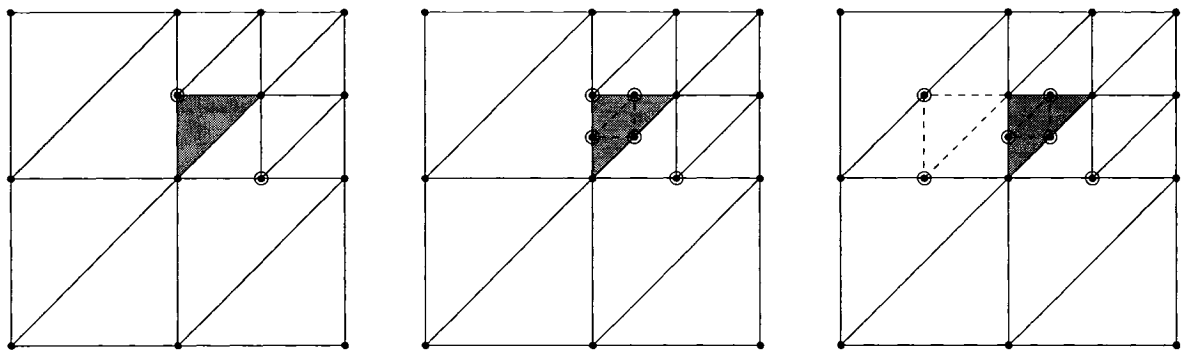
then the element is regularly (red) refined. Other triangles having hanging nodes are either blue refined or green refined.



(a) Red (b) Blue (c) Green

Figure 1.6: Triangle refinement patterns

Regular refinement can be generalized to quadrilateral element adaptivity. The algorithm is slightly modified to allow hanging nodes that are subsequently constrained during the numerical simulation. Only one hanging node is allowed on each edge to produce a one level non-conformal mesh. Figures 1.7 and 1.8 show the sequence of refinement steps from left to right for both triangular and quadrilateral meshes, respectively. The first step is the marking of an element followed by the actual refinement and then additional elements are refined to maintain the level one non-conformity condition.



(a) (b) (c)

Figure 1.7: Triangular mesh refinement maintaining one level non-conformal elements

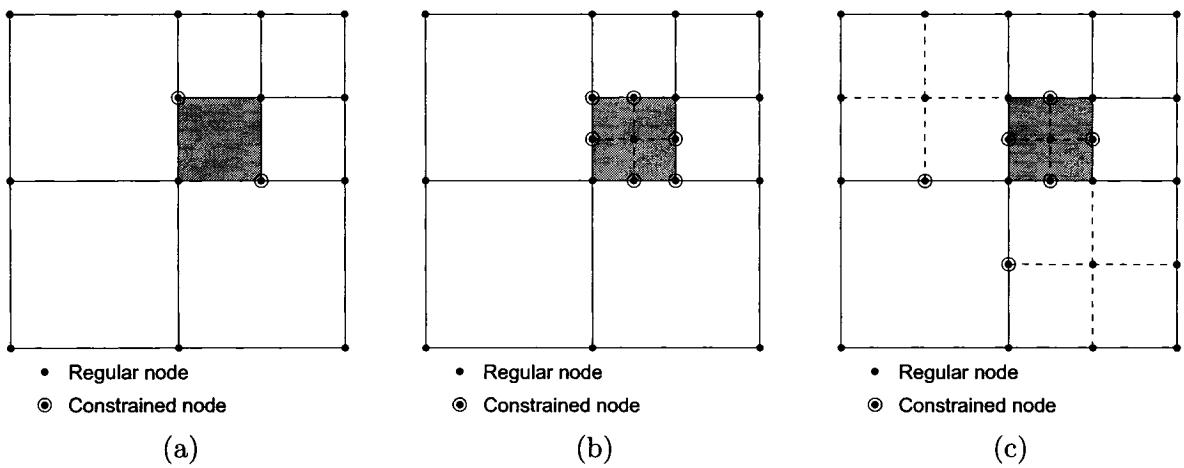


Figure 1.8: Quadrilateral mesh refinement maintaining one level non-conformal elements

1.5.2 Design of a mesh database

The complexity of mesh adaptive techniques and the additional requirements of the error estimation, mesh adaptation and re-solution cycles, make the software development of mesh management tools a complex problem. A new approach to scientific computing based on formal methods was proposed by ElSheikh *et al.* in [38] for the design of mesh generation systems. The proposed method is easily extendable to adaptive finite element systems.

In ElSheikh *et al.* [38], the use of software design documents, semi-formal software specifications and modular decomposition is promoted. Software design documents are a set of separate documents targeting different stages of the software design process. Formal methods are collections of mathematical notations and techniques for describing and analyzing software systems. A semi-formal language is introduced for writing the software design documents. This mathematical way of defining/modeling the software eases the process of analyzing the software correctness. Modular decomposition is the process of dividing the software into a set of modules which are small, easy to understand with as much independence as possible to achieve the flexibility for expected changes. In the proposed framework, a sample design specification

document is summarized for a two dimensional unstructured mesh generation and adaptivity database. The introduced techniques are applicable to a wide class of numerical simulation software systems.

1.6 Summary of papers

1.6.1 Paper I

Assessment of two a posteriori error estimators for FEM. Part I: Steady-state flow, Submitted.

This paper presents a numerical evaluation of two a posteriori error estimation techniques for steady-state flow problems. The Element Residual Method (ERM) and the superconvergent patch recovery technique attributed to Zienkiewicz-Zhu (ZZ) are presented in a unified mathematical framework. The initial finite element solution is obtained using three different meshes representing different classes of structured and unstructured discretizations. The two estimators are tested numerically using four different problems. For problems with smooth solutions, the error estimators are evaluated on a sequence of uniformly refined meshes, while for problems with non-smooth solutions an adaptive mesh refinement iteration is utilized. The quality of the estimators is evaluated by comparting the effectivity index (efficiency index) of the estimated errors. The effectivity index is defined as the ratio of the estimated error to the exact error.

1.6.2 Paper II

Assessment of two a posteriori error estimators for FEM. Part II: Elasticity, Submitted.

This paper forms the second part of a two-part paper on the evaluation of local a posteriori error estimation for FEM. The problem formulation presented in Part I (Paper I) is extended to account for the coupled field problems appearing in elasticity problems. The performance of the two error estimators, ERM and ZZ, is tested nu-

merically using four practical engineering problems, where the ratio of the estimated errors by both methods to the exact error is compared. For problems that possess areas with stress concentration or points of singularity, a mesh adaptation based on the error estimates is used to improve the convergence rates of the finite element solution.

1.6.3 Paper III

Numerical investigation of the reliability of a posteriori error estimation for advection diffusion equations, Accepted in Communications in Numerical Methods in Engineering.

Error estimation for advection-diffusion equations based on ERM is investigated numerically. The finite element formulation for advection-diffusion equations suffers from physically unrealistic oscillations for advection dominant problems. The Streamline Upwind Petrov Galerkin (SUPG) method is used to stabilize the finite element solution and to stabilize the local ERM problems. Error estimation using the stabilized element residual method and using ZZ patch recovery is performed on a number of test problems. In this study, the estimated errors are compared to the exact errors using three different error norms to study the effect of the norm used on the accuracy of the estimated errors. The estimated errors are also used to drive a mesh adaptivity iteration and the adapted meshes are evaluated qualitatively, in terms of conformity to the sharp layers in the solution and in terms of the dependency on the wind direction.

1.6.4 Paper IV

A posteriori error estimation based on numerical realization of the variational multiscale method, Submitted.

In this paper, a general framework for designing different local error estimators is presented. The general framework is based on a numerical realization of the Variational MultiScale method (VMS). The VMS introduces a space decomposition into

resolved components (captured by the mesh) and unresolved components (subgrid scale). Using this decomposition in the finite element variational formulation, a fine scale variational equation is obtained. This equation is localized with a general localization function to derive a generic equation for local error estimation techniques. The properties of the generic error equation are studied theoretically to obtain a practical upper bound and lower bound for the error in the energy norm. The Element Residual Method (ERM) is obtained by using an element based localization function and a new Subdomain Residual Method (SRM) is developed based on using a partition on unity as the localization function. This new SRM formulation is flux-free and straight forward to implement. Numerical evaluation of the proposed SRM estimation is performed on elliptic second order PDEs.

1.7 Conclusions

Significant computational benefits are realized by reliable error estimation and mesh adaptivity. Providing an explicit error calculation for the calculated results and then adapting the mesh to satisfy a certain user-defined tolerance is a very powerful verification method. Adaptive simulation enables the discovery of possible drawbacks in the computational method and is an essential step before and during model evaluation. The creation of quasi-optimal meshes using mesh adaptivity algorithms has significant economical impact because it reduces the necessary computer simulation time. The following subsection presents the main contributions of this thesis followed by a brief list of future research directions.

1.7.1 Main conclusions

▷ Based on the analysis of local a posteriori error estimation for elliptic second order PDEs modeling steady-state flow problems and elasticity problems (Papers I and II), the following conclusions are drawn:

1. The ERM provides a more reliable error estimation with an effectivity

index close to the optimal value of unity in comparison to the ZZ error estimator. As the mesh is refined, higher quality solutions are obtained and the estimated errors using the ERM approach the exact error value.

2. For steady-state flow problems with data oscillation in the load term, or discontinuity in the boundary, the ERM significantly outperforms ZZ in estimating the errors. This deficiency in the ZZ method is attributed to the heuristic nature of the recovery operator, which completely neglects the problem formulation and boundary specification. ERM has the advantage of approximating the residual equation and thus yielding a more accurate error estimation.

3. The unbalanced flux applied on the boundary of the local problems in the ERM reduces the reliability of the method on coarse meshes. However, by combining the ERM with a mesh adaptivity process, one can produce optimal meshes with self-equilibrated local problems. This provides reliable error estimation with an effectivity index close to unity achieved after only few adaptivity iterations.

4. An adaptive algorithm combined with the ERM error estimator can be developed to solve steady-state flow problems with a pre-specified error tolerance in the energy norm.

▷ Based on the investigation of local error estimation techniques for advection diffusion equations (Paper III), the following conclusions are drawn:

1. It was shown that effectivity indices close to unity will only be attained if the sharp layers are completely resolved by the mesh. While this is a significant limitation of error estimation techniques based on solving local problems, these local error estimators still manage to guide the adaptivity process to obtain meshes that resolved the sharp layers.

2. The use of the stability norm instead of the standard weighted energy norm has shown some advantages numerically. In the case of internal layers, measuring the errors in stability norm has resulted in effectivity indices closer to the optimal unit value. Adapted meshes that are guided by the estimated errors using the ERM measured in the stability norm were less dependent on the wind direction.
3. A refinement criteria based on the sorting strategy is recommended for problems with local discontinuities or points of singularity to reduce the number of mesh adaptivity iterations.

▷ A new general framework for error estimation is developed based on the variational multiscale method and different localization techniques (Paper IV). The developed framework is used to derive the element residual methods and a new subdomain residual method. According to this study, the following conclusions are made:

1. The developed framework does not impose any assumptions about the error except for neglecting coarse scale error components. This framework is generic and can be extended to different classes of problems and it can be used to derive different local error estimation techniques.
2. The proposed subdomain error estimator solves local problem over nodal based patches subjected to a zero flux boundary condition. It avoids error locality assumptions or artificial boundary specification. This results in superior numerical performance on coarse meshes where the averaging assumption in the ERM is not valid.
3. The developed subdomain error estimator provides a continuous approximation of the fine scale solution that enables computing a lower bound error estimation. This lower bound error is shown to be very sharp numerically. This continuous error field can also be used visually to show the

quality of the results and the relative magnitude of the errors to the solution values. Moreover, the distribution of the calculated subgrid solutions is a strong indicator of the optimality of the mesh to a certain problem. Optimal meshes have a uniform distribution of the subgrid solution over the problem domain, which correspond to smaller nodal spacing at the sharp features in the solution.

1.7.2 Future research

The following extensions are suggested to the current research for future work:

- ▷ The proposed subdomain error estimator was tested on steady-state flow problems. Extending this error estimator to coupled field problems like elasticity problems would be of great practical interest. Extension to advection-diffusion equations should benefit from the stability of the estimator with enlarging the approximation space. Unlike the ERM, using adaptive solution techniques for solving the local problems might be used as a way to resolve the length scale of the sharp layers.

- ▷ The formulation of the error estimator for three dimensional problems is straight forward. However, the implementation is much more demanding. For three dimensional problems, flux-equilibration for the ERM is particularly complex and the advantages of the flux-free subdomain method should be even more pronounced.

- ▷ For many practical problems the interest is in an integral (functional) of the solution. Examples of output functionals are the displacement at a point, the average stress close to a point or the force integral over a certain boundary of the domain like the drag or lift forces. Energy norm error estimates can be used within the framework of goal-error oriented adaptivity to obtain bounds on these linear output functionals. The energy norm for the primal (original)

problem and the dual problem (corresponding to the output of interest) are weighted together using the parallelogram identity to provide an upper and lower bounds for the error in the desired functional. The ERM was used within the goal oriented framework with some success [39]. It would be of great interest to investigate the performance of the proposed subdomain error estimator instead of the ERM, as it provides uniformly accurate error estimation across the different error scales.

- ▷ Extending the work to nonlinear problems is another research direction. For nonlinear problems, a series of linear problems is solved within an iterative scheme (i.e Newton-Raphson or similar algorithms). Local implicit error estimators can be employed on the linearized equations around the finite element solution.

Bibliography

[1] P. J. Roache, *Verification and Validation in Computational Science and Engineering.* Albuquerque, NM: Hermosa Publishers, 1998.

[2] J. T. Oden and S. Prudhomme, "Estimation of modeling error in computational mechanics," *J. Comput. Phys.*, vol. 182, no. 2, pp. 496–515, 2002.

[3] B. Szabo and I. Babuska, *Finite Element Analysis.* New York: J. Wiley & Sons, 1991.

[4] S. C. Brenner and L. R. Scott, *The mathematical theory of finite element methods*, vol. 15 of *Texts in Applied Mathematics.* New York: Springer-Verlag, second ed., 2002.

[5] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*, vol. 40 of *Classics in Applied Mathematics.* Philadelphia: SIAM, 2002.

[6] K. J. Bathe, *Finite Element Procedures.* Prentice-Hall, 1997.

[7] O. C. Zienkiewicz and R. L. Taylor, *The Finite Element Method, Volume 1: Basic Formulation and Linear Problems.* London: McGraw-Hill, 1989.

[8] R. Verfürth, *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques.* Advances in Numerical Mathematics, Wiley-Teubner, 1996.

[9] I. Babuska and T. Strouboulis, *The Finite Element Method and Its Reliability.* Oxford University Press, 2001.

[10] M. Ainsworth and J. T. Oden, *A Posteriori Error Estimation in Finite Element Analysis.* John Wiley & Sons, 2000.

[11] I. Babuška and W. C. Rheinboldt, "Analysis of optimal finite-element meshes in $\mathbf{R}^1$," *Math. Comp.*, vol. 33, no. 146, pp. 435–463, 1979.

[12] A. Ern and J.-L. Guermond, *Theory and practice of finite elements*, vol. 159 of *Applied Mathematical Sciences.* New York: Springer-Verlag, 2004.

[13] M. Křížek and P. Neittaanmäki, "On superconvergence techniques," *Acta Appl. Math.*, vol. 9, no. 3, pp. 175–198, 1987.

[14] O. C. Zienkiewicz and J. Z. Zhu, "The superconvergent patch recovery and a posteriori error estimates. I. The recovery technique," *Internat. J. Numer. Methods Engrg.*, vol. 33, no. 7, pp. 1331–1364, 1992.

[15] O. C. Zienkiewicz and J. Z. Zhu, "The superconvergent patch recovery and a posteriori error estimates. II. Error estimates and adaptivity," *Internat. J. Numer. Methods Engrg.*, vol. 33, no. 7, pp. 1365–1382, 1992.

[16] P. Ladevèze and D. Leguillon, "Error estimate procedure in the finite element method and applications," *SIAM J. Numer. Anal.*, vol. 20, no. 3, pp. 485–509, 1983.

[17] D. W. Kelly, "The self-equilibration of residuals and complementary a posteriori error estimates in the finite element method," *Internat. J. Numer. Methods Engrg.*, vol. 20, no. 8, pp. 1491–1506, 1984.

[18] R. E. Bank and A. Weiser, "Some a posteriori error estimators for elliptic partial differential equations," *Math. Comput.*, vol. 44, no. 170, pp. 283–301, 1985.

[19] M. Ainsworth and J. T. Oden, "A unified approach to a posteriori error estimation using element residual methods," *Numer. Math.*, vol. 65, no. 1, pp. 23–50, 1993.

[20] V. John, "A numerical study of a posteriori error estimators for convection-diffusion equations," *Comput. Methods Appl. Mech. Engrg.*, vol. 190, no. 5-7, pp. 757–781, 2000.

[21] S. Prudhomme, J. T. Oden, T. Westermann, J. Bass, and M. E. Botkin, "Practical methods for a posteriori error estimation in engineering applications," *Internat. J. Numer. Methods Engrg.*, vol. 56, no. 8, pp. 1193–1224, 2003.

[22] I. Babuška and W. C. Rheinboldt, "Error estimates for adaptive finite element computations," *SIAM J. Numer. Anal.*, vol. 15, no. 4, pp. 736–754, 1978.

[23] C. Carstensen and S. A. Funken, "Fully reliable localized error control in the FEM," *SIAM J. Sci. Comput.*, vol. 21, no. 4, pp. 1465–1484, 1999/00.

[24] P. Morin, R. H. Nochetto, and K. G. Siebert, "Local problems on stars: a posteriori error estimators, convergence, and performance," *Math. Comp.*, vol. 72, no. 243, pp. 1067–1097, 2003.

[25] S. Prudhomme, F. Nobile, L. Chamoin, and J. Oden, "Analysis of a subdomain-based error estimator for finite element approximations of elliptic

problems," *Numer Methods Partial Differential Equations*, vol. 20, no. 2, pp. 165–192, 2004.

[26] N. Parés, P. Díez, and A. Huerta, "Subdomain-based flux-free a posteriori error estimators," *Comput. Methods Appl. Mech. Engrg.*, vol. 195, no. 4-6, pp. 297–323, 2006.

[27] T. J. R. Hughes, "Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods," *Comput. Methods Appl. Mech. Engrg.*, vol. 127, no. 1-4, pp. 387–401, 1995.

[28] T. J. R. Hughes, G. R. Feijóo, L. Mazzei, and J.-B. Quincy, "The variational multiscale method—a paradigm for computational mechanics," *Comput. Methods Appl. Mech. Engrg.*, vol. 166, no. 1-2, pp. 3–24, 1998.

[29] A. N. Brooks and T. J. R. Hughes, "Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations," *Comput. Methods Appl. Mech. Engrg.*, vol. 32, no. 1-3, pp. 199–259, 1982.

[30] T. J. R. Hughes, L. P. Franca, and G. M. Hulbert, "A new finite element formulation for computational fluid dynamics. VIII. The Galerkin/least-squares method for advective-diffusive equations," *Comput. Methods Appl. Mech. Engrg.*, vol. 73, no. 2, pp. 173–189, 1989.

[31] L. P. Franca, S. L. Frey, and T. J. R. Hughes, "Stabilized finite element methods. I. Application to the advective-diffusive model," *Comput. Methods Appl. Mech. Engrg.*, vol. 95, no. 2, pp. 253–276, 1992.

[32] M.-C. Rivara, "Mesh refinement processes based on the generalized bisection of simplices," *SIAM J. Numer. Anal.*, vol. 21, no. 3, pp. 604–613, 1984.

[33] I. G. Rosenberg and F. Stenger, "A lower bound on the angles of triangles constructed by bisecting the longest side," *Math. Comp.*, vol. 29, pp. 390–395, 1975.

[34] M.-C. Rivara, "Algorithms for refining triangular grids suitable for adaptive and multigrid techniques," *Internat. J. Numer. Methods Engrg.*, vol. 20, no. 4, pp. 745–756, 1984.

[35] M.-C. Rivara, "New longest-edge algorithms for the refinement and/or improvement of unstructured triangulations," *Internat. J. Numer. Methods Engrg.*, vol. 40, no. 18, pp. 3313–3324, 1997.

[36] R. E. Bank and A. H. Sherman, "An adaptive, multilevel method for elliptic boundary value problems," *Computing*, vol. 26, no. 2, pp. 91–105, 1981.

[37] W. F. Mitchell, "A comparison of adaptive refinement techniques for elliptic problems," *ACM Transactions on Mathematical Software*, vol. 15, no. 4, pp. 326–347, 1989.

[38] A. H. ElSheikh, S. Smith, and S. E. Chidiac, "Semi-formal design of reliable mesh generation systems," *Adv. Eng. Softw.*, vol. 35, no. 12, pp. 827–841, 2004.

[39] J. T. Oden and S. Prudhomme, "Goal-oriented error estimation and adaptivity for the finite element method," *Comput. Math. Appl.*, vol. 41, no. 5-6, pp. 735–756, 2001.

Chapter 2

Assessment of two a posteriori error estimators for FEM. Part I: Steady-state flow

A.H. ElSheikh, S.E. Chidiac and S. Smith

ABSTRACT

A two-part paper evaluates the application of local a posteriori error estimation to steady-state flow problems (Part I) and elasticity problems (Part II). A posteriori error estimation for the finite element (FE) method is a functional tool for assessing the quality of the finite element results. This paper introduces two error estimators, the Zienkiewicz-Zhu (ZZ) and the Element Residual Method (ERM), within a general mathematical framework. These error estimators are used as drivers for a mesh adaptation process. Four steady-state flow problems using Poisson equation are included to compare the two error estimators and to demonstrate the functional use of a posteriori error estimators for obtaining FE solutions with a pre-specified error tolerance. Of the two methods, the ERM is shown to be more reliable in comparison to the ZZ estimator.

KEY WORDS: Adaptivity, A Posteriori Error Estimation, Element Residual Method, Finite Element Method, Steady-state flow, Zienkiewicz-Zhu patch recovery

2.1 Introduction

The finite element (FE) method is a well-established method for solving Partial Differential Equations (PDEs). The accuracy of the FE solution, which is defined as the amount of agreement between the FE results and the exact solution of the mathematical model primarily depends on the discretization errors. Our goal is to provide a comprehensive numerical evaluation of the Zienkiewicz-Zhu (ZZ) [1, 2] method and the Element Residual Method (ERM) [3] for local a posteriori estimation methods.

The development of a posteriori error estimation techniques for the linear finite elements started in the late 70's and the early 80's [4, 5, 6, 7]. Most of the developed error estimation methods were concerned with global estimation of the error and were generally used as an essential ingredient for mesh adaptivity algorithms. More recently, in the mid 90's, techniques for developing a posteriori error estimation in the so called "quantities of interest" were introduced [8, 9, 10]. Estimating error in the quantities of interests is based on the well known reciprocal theorem used in the construction of influence lines in structural analysis [11]. The quantity of interest is used to formulate a dual problem, which have to be solved globally. The errors in both the original problem (primal) and the dual problem are weighted to produce the error in the quantity of interest. The estimation of the errors locally either by explicit formulation of the residual [8] or based on energy norms [10] is an essential step in estimating the errors in the quantities of interest and the reliability of the local error estimation is therefore a major contributing factor in the final accuracy of the error in the quantities of interest.

This paper presents the variational formulation of steady-state flow problems followed by the underlying theory of a posteriori error estimation based on the error residual equation of the finite element method. The performance of ERM and ZZ error estimators is evaluated numerically using four steady-state flow problems with both smooth and non-smooth solution. The problems are solved using structured and non-structured meshes with different configuration that are uniformly and adaptively

refined.

2.2 Preliminaries

A steady-state flow problem over a bounded domain Ω in $\mathbb{R}^2$ is considered. The boundary $\partial\Omega$ is piecewise smooth and composed of a Dirichlet portion Γ_D and a Neumann portion Γ_N, where $\partial\Omega = \Gamma_D \cup \Gamma_N$. The corresponding PDE can be written in an abstract format as:

Find u such that,

$$\begin{cases} Lu &= f, \quad \text{over} \quad \Omega \\ u &= 0, \quad \text{on} \quad \Gamma_D \\ \partial u/\partial n &= g, \quad \text{on} \quad \Gamma_N \end{cases} \tag{2.1}$$

where L is a linear symmetric second order differential operator of the form $Lu = -\triangle u$ and g is the prescribed values of the normal flux on the Neumann portion of the boundary Γ_N. Using standard norm notation in Hilbert spaces [12], the weak form of the problem can be written as:

$$\text{Find } u \in U \text{ such that:} \quad \mathcal{B}(u,v) = (f,v) + \langle g, v\rangle \qquad \forall v \in V \tag{2.2}$$

in which U is the trial space and V is the test space defined as $V = \{v \in H^1(\Omega) : v = 0 \text{ over } \Gamma_D\}$ and $H^1(\Omega)$ denotes the usual Sobolev space. The bilinear $\mathcal{B}$ is defined on $U \times V$ as $\mathcal{B}(u,v) = \int \nabla u \cdot \nabla v \; d\Omega$. The right side terms in Eq. (2.2) are defined as $(f,v) = \int f \; v \; d\Omega$ and $\langle g, v\rangle = \int g \; v \; d\Gamma_N$. The existence of the solution of Eq. (2.2) is proven by the Lax-Milgram Lemma [13].

2.3 A posteriori error estimation

A posteriori error estimation is based on approximating the residual of the weak formulation [14, 15]. This is usually done by enriching the trial space by employing more basis functions. This can be achieved by refining the initial mesh (h-refinement)

or by increasing the order of the approximation over the finite elements (p-refinement). Using Eq. (2.2), the finite element solution u_h satisfies

$$\mathcal{B}(u_h, v_h) = (f, v_h) + \langle g, v_h \rangle \qquad \forall v_h \in V_h \subset V \tag{2.3}$$

The error in the solution of Eq. (2.3) is $e = u - u_h$, where u is the exact solution and u_h is the finite element solution. Using the value of e in Eq. (2.2) one gets,

$$\mathcal{B}(e, v) = \mathcal{B}(u, v) - \mathcal{B}(u_h, v) = (f, v) + \langle g, v \rangle - \mathcal{B}(u_h, v) \tag{2.4}$$

Dividing the integration domain into a set of N elements, the error equation becomes

$$\mathcal{B}(e, v) = \sum_{l=1}^{N} \left\{ \int_{\Omega_l} f\, v + \int_{\partial\Omega_l \cap \Gamma_N} g\, v - \int_{\Omega_l} \nabla u_h \cdot \nabla v \right\} \tag{2.5}$$

Integrating by parts and rearranging Eq. (2.5) yields

$$\mathcal{B}(e, v) = \sum_{l=1}^{N} \left\{ \int_{\Omega_l} r\, v + \int_{\partial\Omega_l \cap \Gamma_N} R\, v - \int_{\partial\Omega_l - \Gamma_N} \left[\frac{\partial u_h}{\partial n}\right] v \right\} \tag{2.6}$$

where:

$$\begin{aligned} r &= f + \Delta u_h && \text{(Element term)} \\ R &= g - \tfrac{\partial u_h}{\partial n} \text{ on } \partial\Omega_l \cap \Gamma_N && \text{(Boundary term)} \\ \left[\partial u_h / \partial n\right] &= n_k \cdot (\nabla u_h)_k + n_{k'} \cdot (\nabla u_h)_{k'} && \text{(Edge term)} \end{aligned}$$

where the elements k and k' correspond to two elements sharing a common edge. Equation (2.6) is the general equation for evaluating errors in FE method. It contains all the potential sources of discretization error, namely the element interior residual, the inter-element and boundary residual terms.

The goal of a posteriori error estimator is to solve the global residual equation in an efficient way [15, 16]. This is achieved by localizing Eq. (2.6) over an element or

a point based patch. This results in a set of decoupled equations that are simpler to evaluate than the global equation. In the following sections, two different approaches are presented to localize the global error equation, namely the ZZ error estimator and the ERM.

2.4 Recovery Based Error Estimators

Recovery based error estimators use the gradient of the finite element approximation instead of the solution itself. This stems from the fact that the weak formulation imposes C^0 continuity on the unknown u which yields a continuous flow field but the flux field has inter-element jumps. Having a jump in the flux field is one type of element residual at the boundaries, which has been treated extensively in the post-processing of many FE packages by either averaging the gradient to obtain a recovered gradient from the FE solution or by using a least square based approximation to obtain a continuous function derivative.

Recovery type error estimators operates under the premise that smoothed function derivative is more accurate than the untreated gradient resulting from the finite element solution. The difference between the two gradients can be used to approximate the error as following:

$$\|e\|^2 \approx \eta^2 = \int_\Omega |G_h[u_h] - \nabla u_h|^2 \tag{2.7}$$

where $G_h[u_h]$ is the recovered gradient reconstructed from the finite element solution u_h. Substituting $v = e$ in Eq. (2.4) and using the definition of the bilinear $\mathcal{B}$, the error norm is evaluated as:

$$\|e\|^2 = \mathcal{B}(e,e) = \mathcal{B}(u - u_h, u - u_h) = \int_\Omega |\nabla u - \nabla u_h|^2 \tag{2.8}$$

In this case, the recovery operator $G_h[u_h]$ is used to approximate the exact gradient.

The accuracy of this type of error estimator depends completely on the choice of that operator.

The most well known recovery type error estimator is the Zienkiewicz-Zhu error estimator. It is based on gradient recovery by solving local least square problems over discrete patches [1, 2]. The method is relatively easy to incorporate into finite element codes and is independent of the problem formulation. The gradient is sampled at the centroid of the elements because of the superconvergence properties of gradient at these points [17, 18, 19].

2.5 Implicit Residual type Error Estimators

Implicit residual type error estimators, which are based on approximating the residual equation as a set of local problems, accounts for all types of residuals appearing in Eq. (2.6). The domain of the local problem determines the type of a residual error estimator. The residual problem can be solved over one element or over a patch of elements in the element residual method (ERM) and the sub-domain residual methods (SRM) [20, 21], respectively. This paper focuses on the element residual method where a set of local problems with Neumann boundary conditions are solved. These local problems are formulated as:

$$\mathcal{B}_k(e,v) = F_k(v) - \mathcal{B}_k(u_h,v) + \int_{\partial k} \left\langle \frac{\partial u_h}{\partial n_k} \right\rangle v \tag{2.9}$$

where $\mathcal{B}_k$ is a restriction of the bilinear over an element k, $F_k(v)$ the load term defined as $(f,v)_k$ and integrated over the element k, and n_k the normal vector to each edge of the element k. The Neumann boundary terms in Eq. (2.9) are specified as:

$$\left\langle \frac{\partial u_h}{\partial n_k} \right\rangle = \begin{cases} \frac{1}{2}\, n_k \,.\, \{(\nabla u_h)_k + (\nabla u_h)_{k'}\} & \text{on } \partial k \cap \partial k' \\ n_k \,.\, (\nabla u_h)_k & \text{on } \partial k \cap \Gamma_D \\ g & \text{on } \partial k \cap \Gamma_N \end{cases} \tag{2.10}$$

To obtain the exact error, the test space $v \in V_k$ in Eq. (2.9) is chosen such that

$$V_k = \{v \in H^1(k) \ : \ v = 0 \text{ on } \partial k \cap \Gamma_D\} \tag{2.11}$$

Bubble functions are used to approximate both the trial and test space of the local residual problems [15]. Accordingly, edge bubbles are employed to account for the edge residuals (edge flux jump), and element bubbles to account for the local element residual. For a reference triangular element $\widehat{k}$ with functions $\lambda_1, \lambda_2, \lambda_3$, the barycentric (area) coordinates, the edge bubble functions χ_1, χ_2, χ_3 are given by $\chi_1 = 4\ \lambda_2\ \lambda_3$, $\chi_2 = 4\ \lambda_1\ \lambda_3$, $\chi_3 = 4\ \lambda_1\ \lambda_2$, and the interior element bubble function χ_4 is defined as: $\chi_4 = 27\ \lambda_1\ \lambda_2\ \lambda_3$. Establishing a solution for Eq. (2.9) is not guaranteed unless the edge fluxes are in equilibrium. For coarse meshes, the equilibrium condition tends to be violated and a least square method is used to solve this problem. Flux equilibration was not enforced in this study.

2.6 Numerical Examples

The performance of ZZ and ERM error estimators are assessed using four numerical experiments. The Method of Manufactured Solutions (MMS) [22] is used to build the steady-state flow problems. The engineering problems corresponding to steady-state flow are of the form,

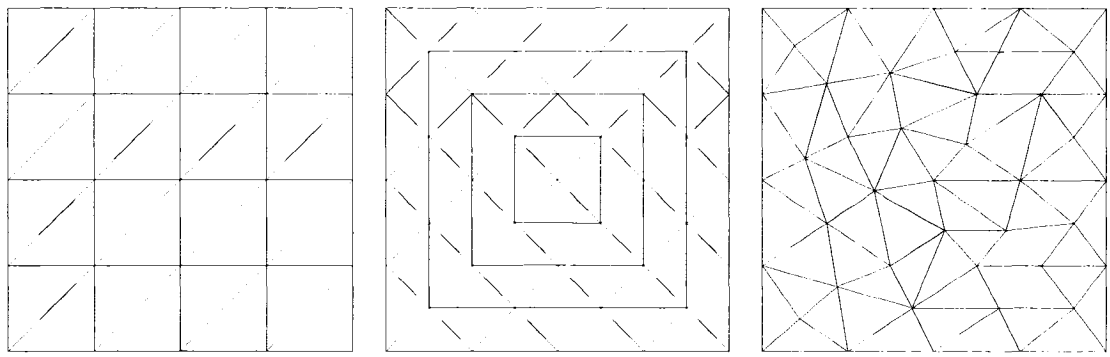
Find u such that:

$$-\Delta u \ = f \qquad \text{on} \quad \Omega \tag{2.12}$$

where the load function f is set such that the equation satisfies the desired exact solution. To simplify the presentation of the results, the quality of the estimates is measured in terms of the efficiency index. The efficiency index (EI) is the ratio of the calculated error using the a posteriori error estimator to the exact error in the energy norm defined as,

$$\|e\| = \left(\int_\Omega |\nabla e|^2 + e^2 \right)^{1/2}$$

The first three problems are defined over the domain $\Omega = (0,1) \times (0,1)$ and three different initial meshes are used. Figure 2.1 plots the three coarse meshes with 25, 41 and 44 degrees of freedom for meshes 1, 2 and 3, respectively.



(a) Mesh 1 (b) Mesh 2 (c) Mesh 3

Figure 2.1: Different meshes used for problem P1, P2 and P3

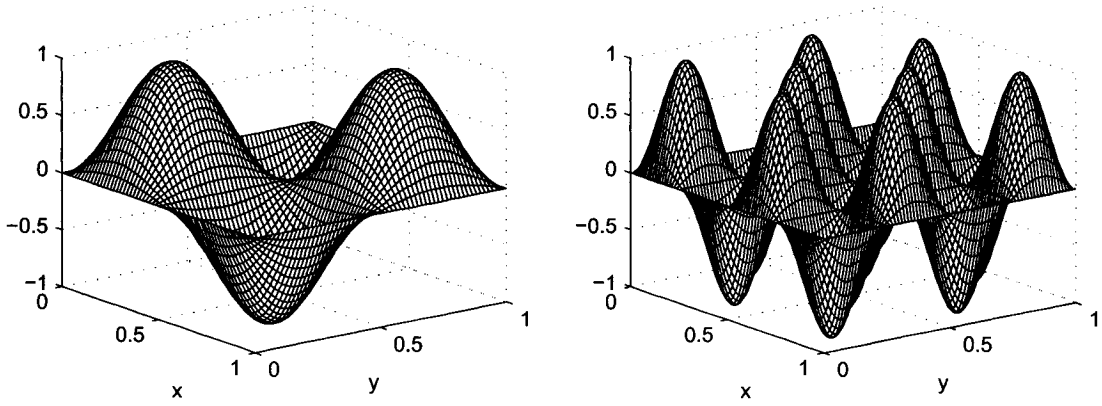
2.6.1 Steady-state flow problem P1

The load function f for the first steady-state flow problem is set such that the exact solution is defined as:

$$u = \sin(a\pi x)\ \sin(a\pi y) \tag{2.13}$$

A homogeneous Dirichlet boundary condition is specified at the boundaries of problem domain. This problem is solved using four different values of a using initial meshes 1, 2 and 3. Figure 2.2 shows a 3D plot of the exact solution for $a = 2$ and $a = 4$. A sequence of uniformly refined meshes are used to study the behavior of both the ZZ and ERM error estimators for different mesh resolutions. The Efficiency Indices of the estimated errors are listed in Tables 2.1, 2.2 and 2.3 for Mesh 1, 2 and 3, respectively. For coarse meshes, the ZZ estimator tends to underestimate the errors while the ERM estimator overestimates the error. This trend increases as a increases, where the mesh is not capable of capturing the solution features. As the mesh is refined, both the

ERM and ZZ estimated error norms are in good agreement with the exact error. The results show that ZZ estimator overestimates the errors when refining Mesh 1 and underestimates the errors when refining Mesh 2 and 3. The efficiency index of the ERM is more consistent for the three different meshes, and as the mesh is refined the ERM efficiency index is asymptomatically exact for all meshes and values of the parameter a.



(a) $a = 2$ (b) $a = 4$

Figure 2.2: Plot of the exact solution of problem P1 with different values for the parameter a

Table 2.1: Efficiency indices for estimated errors using ZZ and ERM for problem P1 mesh 1 - uniform refinement

	$a = 1$		$a = 2$		$a = 3$		$a = 4$	
# nodes	ZZ	ERM	ZZ	ERM	ZZ	ERM	ZZ	ERM
25	0.6487	1.1191	0.4876	1.4684	0.3357	1.6532	0.2558	1.8142
81	0.8738	1.0263	0.7933	1.1234	0.6772	1.2947	0.5501	1.4882
289	0.9990	1.0062	0.9726	1.0266	0.9299	1.0652	0.8732	1.1248
1089	1.0601	1.0015	1.0527	1.0062	1.0405	1.0144	1.0235	1.0266
4225	1.0895	1.0004	1.0876	1.0015	1.0844	1.0034	1.0798	1.0062

Table 2.2: Efficiency indices for estimated errors using ZZ and ERM for problem P1 mesh 2 - uniform refinement

	$a=1$		$a=2$		$a=3$		$a=4$	
# nodes	ZZ	ERM	ZZ	ERM	ZZ	ERM	ZZ	ERM
41	0.6129	1.0805	0.5201	1.2290	0.4595	1.4301	0.8924	0.7434
145	0.7550	1.0248	0.7314	1.0744	0.6871	1.1572	0.6387	1.2551
545	0.8147	1.0062	0.8101	1.0203	0.7959	1.0467	0.7813	1.0790
2113	0.8417	1.0015	0.8409	1.0052	0.8368	1.0123	0.8332	1.0212
8321	0.8543	1.0003	0.8542	1.0013	0.8530	1.0031	0.8522	1.0054

Table 2.3: Efficiency indices for estimated errors using ZZ and ERM for problem P1 mesh 3 - uniform refinement

	$a=1$		$a=2$		$a=3$		$a=4$	
# nodes	ZZ	ERM	ZZ	ERM	ZZ	ERM	ZZ	ERM
44	0.6165	1.1203	0.6060	1.2801	0.5352	1.4590	0.5663	1.4644
157	0.7445	1.0643	0.7387	1.1257	0.7450	1.1799	0.6768	1.2756
593	0.8147	1.0307	0.8222	1.0532	0.8588	1.0675	0.8208	1.1076
2305	0.8505	1.0147	0.8652	1.0229	0.9117	1.0254	0.8876	1.0398
9089	0.8679	1.0072	0.8859	1.0104	0.9349	1.0103	0.9160	1.0157

2.6.2 Steady-state flow problem P2

The load function f for the second steady-state flow problem is set such that the exact solution is

$$u = 0.005\, x^2\, (1-x)^2\, e^{(10x^2)}\, y^2\, (1-y)^2\, e^{(10y)} \tag{2.14}$$

A homogeneous Dirichlet boundary condition is specified at the boundaries of problem domain. Figure 2.3-a shows a 3D plot of Eq. (2.14). Similar to the first problem, a sequence of uniformly refined meshes are used to study the behavior of the two error estimators over three different meshes. The results presented in Table 2.4 confirm the good performance of the ERM in estimating the exact errors independently from

the mesh configuration. The ERM overestimates the errors for coarse meshes and successfully converges to the correct errors on the refined meshes. On the other hand, the results of the ZZ estimator slowly converge to the exact error with mesh refinement. On the most refined meshes, the errors in estimated errors by ERM are 7%, 5% and 5% in comparison to 30%, 28% and 17% for those estimated by ZZ on mesh 1, 2 and 3, respectively.

Table 2.4: Efficiency indices for estimated errors using ZZ and ERM for problem P2 on different meshes - uniform refinement

Mesh 1			Mesh 2			Mesh 3		
# nodes	ZZ	ERM	# nodes	ZZ	ERM	# nodes	ZZ	ERM
25	0.0893	0.8853	41	0.3801	1.2927	44	0.3800	1.4410
81	0.3939	1.2448	145	0.4864	1.1816	157	0.5310	1.2207
289	0.5962	1.0814	545	0.5352	1.2268	593	0.6723	1.1847
1089	0.5912	1.1231	2113	0.6268	1.1371	2305	0.7609	1.1137
4225	0.6909	1.0700	8321	0.7200	1.0537	9089	0.8269	1.0517

2.6.3 Steady-state flow problem P3

Building on the success of tested error estimators, a 2D problem over the domain $\Omega = (0,1) \times (0,1)$ is built such that the exact solution exhibits local features that are hard to capture using a uniform mesh. The load function f in Eq. (2.12) is set such that the exact solution is

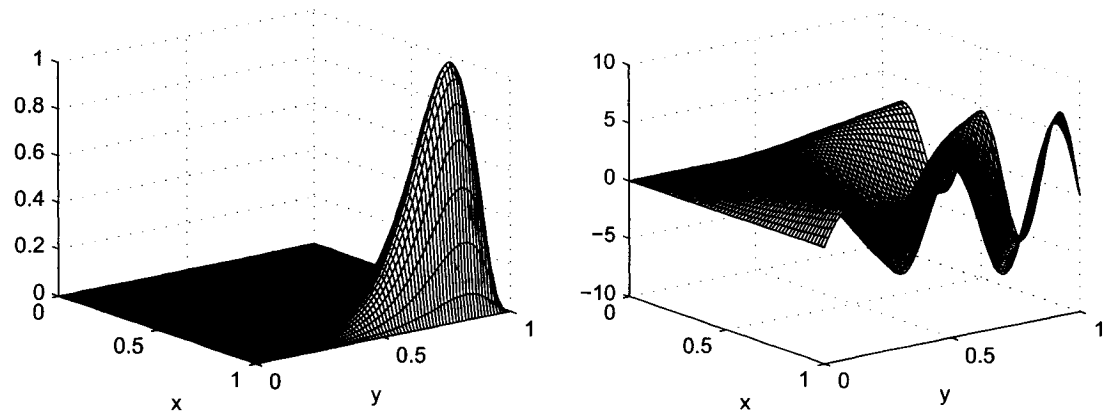
$$u = \sin(5\pi xy)e^{x+y} \tag{2.15}$$

A homogeneous Dirichlet boundary condition is specified at the boundaries with $x = 0$ or $y = 0$ and homogeneous Neumann BC is specified on the rest of the boundary. Figure 2.3-b shows the 3D-plot of the exact solution defined by Eq. (2.15). Table 2.5 lists the efficiency indices for both ZZ and ERM error estimators of the three different meshes. The estimated errors using ERM on coarse meshes is far better than the ones using the ZZ estimator. In some cases, an apparent deterioration of the quality of

estimated errors with global mesh refinement is observed as in the case of mesh 1 using 25 and 81 nodes. The errors in the estimated errors using ERM on mesh 1 with 25 nodes is 5% and 20% for the refined mesh with 81 nodes. At the coarse level, it is apparent that some of the solution features were missed which resulted in the underestimation of the error over parts of the domain. As the mesh is refined, these special features are captured and the estimated errors asymptotically converge to the exact error with further refinement.

Table 2.5: Efficiency indices for estimated errors using ZZ and ERM for problem P3 on different meshes - uniform refinement

Mesh 1			Mesh 2			Mesh 3		
# nodes	ZZ	ERM	# nodes	ZZ	ERM	# nodes	ZZ	ERM
25	0.1413	1.0522	41	0.3856	1.1383	44	0.3650	1.2538
81	0.5153	1.1977	145	0.6635	1.1265	157	0.6594	1.1240
289	0.8352	1.0595	545	0.8371	1.0432	593	0.8327	1.0648
1089	1.0156	1.0096	2113	0.9188	1.0079	2305	0.9396	1.0271
4225	1.0930	1.0010	8321	0.9544	0.9994	9089	0.9908	1.0111



(a) Problem P2 (b) Problem P3

Figure 2.3: Plot of the exact solution of problem P2 and P3

2.6.4 Steady-state flow problem P4

This problem is defined by Laplace equation and solved over the domain Ω, which is a part of a circle centered at $(0,0)$ and defined by $\Omega = \{(r,\theta) : 0 \geq r \geq 1, 0 \geq \theta \geq k\pi/4\}$; where r and θ are the polar coordinates and k is a parameter that defines the size of the domain. The exact solution of this problem possesses local features that are hard to capture using a uniform mesh. The load function f in Eq. (2.12) is set such that the exact solution is

$$u(r,\theta) = r^{2/k}\sin(\frac{2\theta}{k}) \tag{2.16}$$

A homogeneous Dirichlet boundary condition is specified at the boundaries with $\theta = 0$ and Neumann BC are forced on the rest of the boundary.

Table 2.6 shows the efficiency indices for both ZZ and ERM error estimators for the cases, $k = 1,\ 3$ and 5. For the case $k = 1$, the solution is smooth and estimated error on the globally refined mesh is accurate. For the cases $k = 3$ and $k = 5$, the flux of the solution exhibits a singular solution at the point $(0,0)$. In this case, uniform refinement is not capable of adequately capturing the point of singularity. Although, the efficiency index of the ERM still outperforms the ZZ estimator on all the meshes used, it fails to estimate the error with an efficiency index close to one even with large number of DOF (more than 10,000) using uniform refinement. This result shows the limitations of uniform refinement for certain class of problems that possess solution specific features or points of singularity. In the following subsection, the estimated errors are employed to guide a mesh adaptivity process for problems P2, P3 and P4.

2.6.5 Adaptively refined meshes

The objective of adapting the FE mesh using the estimated errors is to equidistribute the errors over all the mesh elements. It is achieved by refining a set of elements with error value more than a pre-specified fraction of the maximum element error in each step (0.4 in the current study).

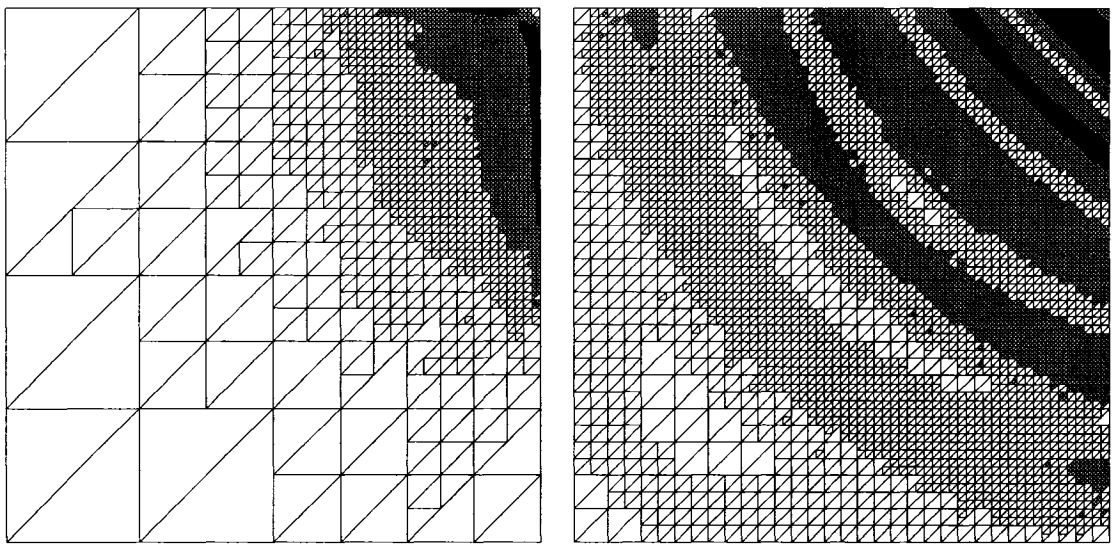
Table 2.6: Efficiency indices for estimated errors using ZZ and ERM for problem P4 on different meshes - uniform refinement

$k=1$			$k=3$			$k=5$		
# nodes	ZZ	ERM	# nodes	ZZ	ERM	# nodes	ZZ	ERM
21	0.6316	1.0336	53	0.5146	0.8334	81	0.5137	0.7210
70	0.8429	1.0772	191	0.5806	0.8126	296	0.5495	0.7309
255	0.9645	1.0481	725	0.6107	0.8106	1131	0.5768	0.7470
973	1.0292	1.0262	2825	0.6250	0.8118	4421	0.5904	0.7585
3801	1.0620	1.0137	11153	0.6323	0.8132	17481	0.5977	0.7658

Figure 2.4 shows the adapted meshes after 16 iterations using ERM as adaptivity driver for problems P2 and P3 starting from mesh 1. The adapted meshes show a smaller nodal spacing at the location of the exponential peak for problem P2 and the wave layers for problem P3. For problem P4, the adapted meshes are plotted in Fig. 2.5 with the parameter k equal to 1, 3 and 5. For $k = 1$, the solution is smooth and the error distribution is uniform over the problem domain. This results a uniformly refined mesh even with local mesh refinement allowed. For the cases of $k = 3$ and 5, larger errors exist close to the point of singularity, which force local refinement to equidistribute the errors especially for relatively coarse meshes.

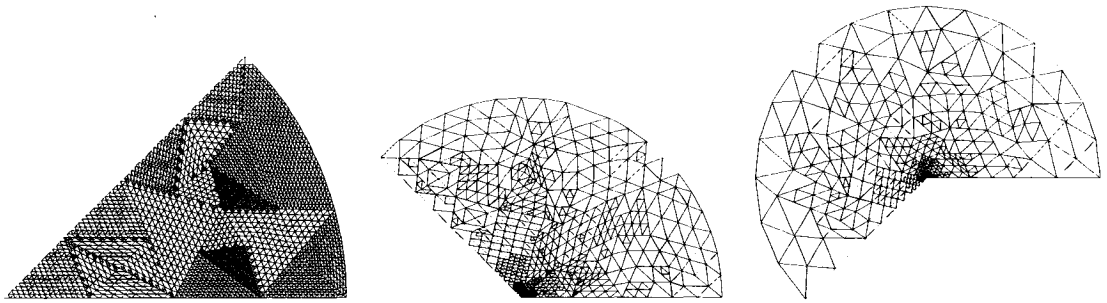
Figure 2.6 shows the efficiency index of the estimated error for problem P2 on the three different meshes using the ERM and ZZ error estimators as adaptivity drivers. For all the six cases, the efficiency index of the estimated errors using the ERM outperforms the ZZ estimator. The oscillation of the ERM efficiency index after the first few iterations is attributed to capturing features of the solution that were not resolved by the second order bubbles. Once the distribution of the errors becomes almost uniform over the domain, the ERM performs as an upper bound of the error.

Figure 2.7 shows similar results for problem P3. The ERM error estimator is clearly shown to yield accurate error estimation, particularly the case of Mesh 1 with



(a) Problem P2 (b) Problem P3

Figure 2.4: Adapted meshes for problem P2 and P3 (after 16 iterations using ERM driven adaptivity on mesh 1)

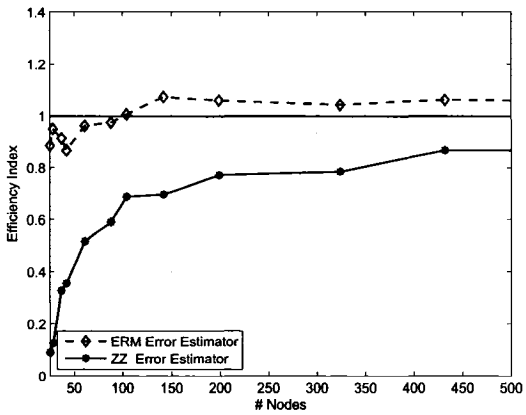


(a) $k = 1$ (b) $k = 3$ (c) $k = 5$

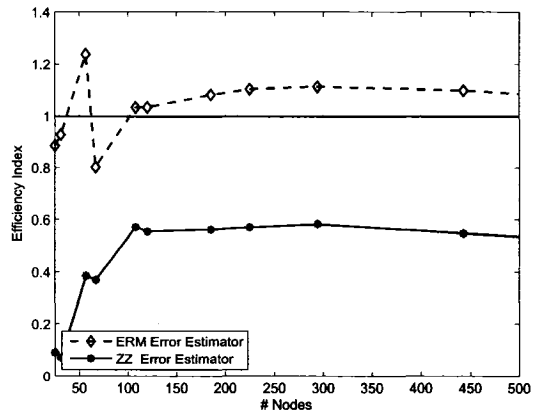
Figure 2.5: Adapted meshes for problem P4 (after 12 iterations using ERM driven adaptivity)

ERM derived adaptivity. Again, the ERM estimator produces better error estimation than the ZZ estimator for the six presented cases for problem P3.

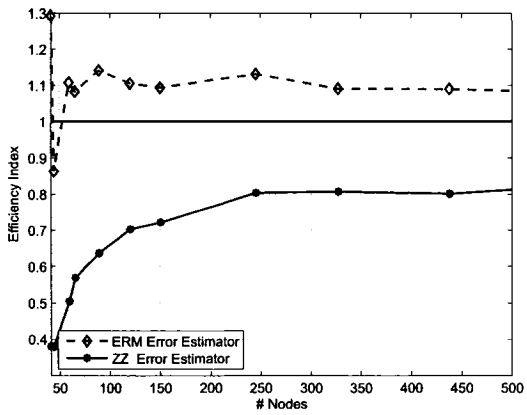
Figure 2.8 presents the global efficiency index for the different variations of problem P4. For the case of $k = 1$, the adaptive mesh refinement produces meshes that capture the solution features. ERM yields accurate error estimation with only 100 nodes. The same conclusion can be drawn for the case $k = 3$, while for the case with $k = 5$, the



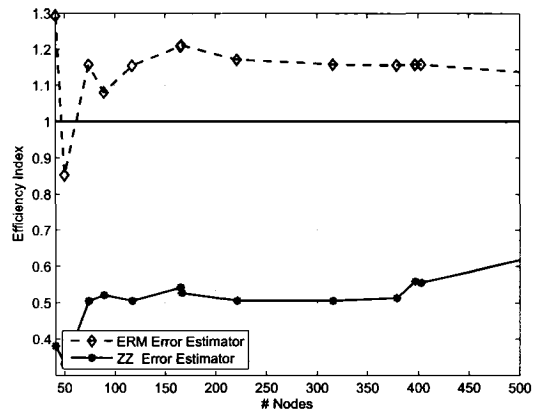
(a) Mesh 1 (ERM Adaptivity)



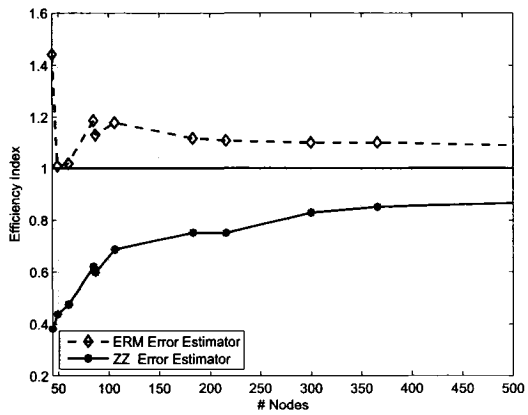
(b) Mesh 1 (ZZ Adaptivity)



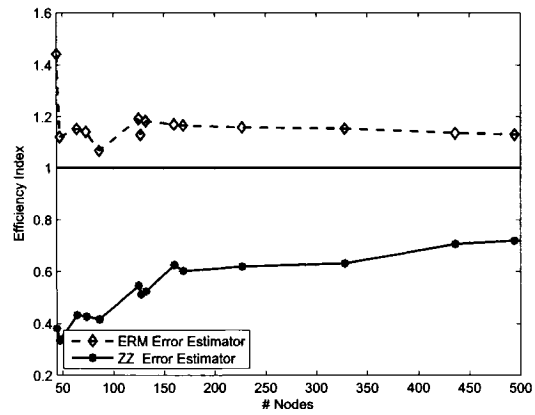
(c) Mesh 2 (ERM Adaptivity)



(d) Mesh 2 (ZZ Adaptivity)

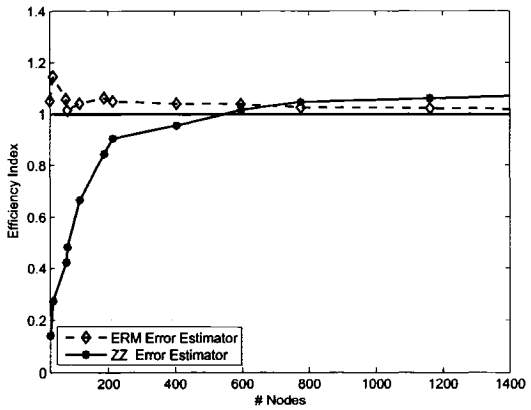


(e) Mesh 3 (ERM Adaptivity)

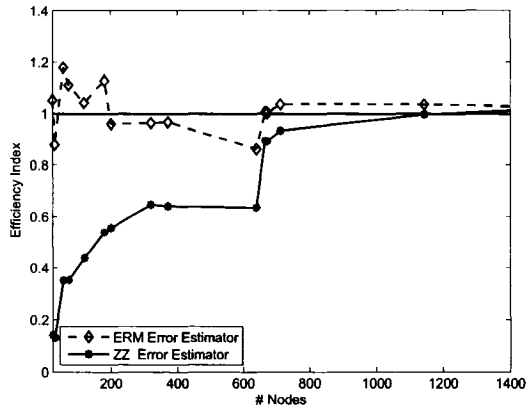


(f) Mesh 3 (ZZ Adaptivity)

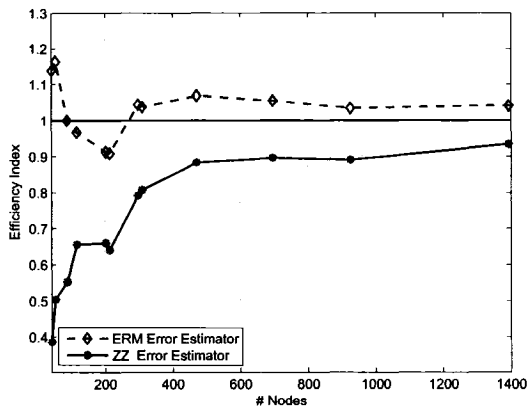
Figure 2.6: Global Efficiency Index of adapted meshes for problem P2



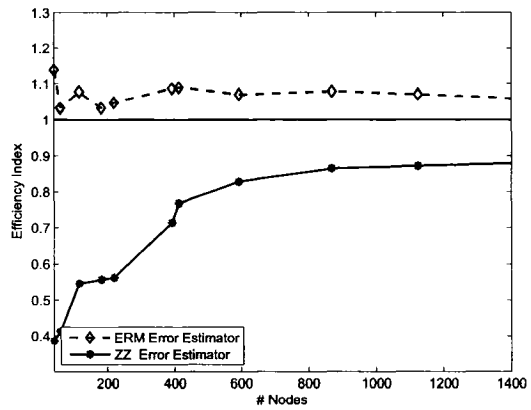
(a) Mesh 1 (ERM Adaptivity)



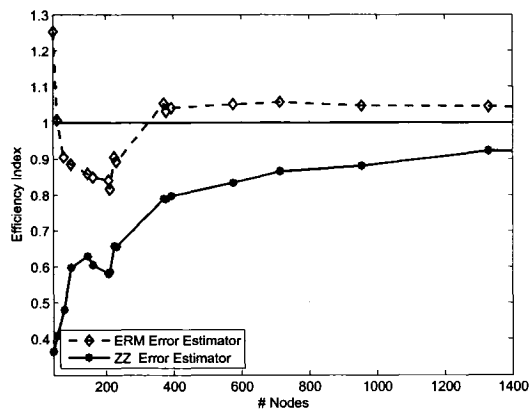
(b) Mesh 1 (ZZ Adaptivity)



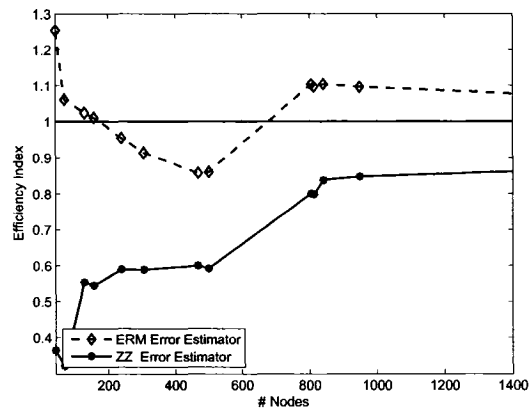
(c) Mesh 2 (ERM Adaptivity)



(d) Mesh 2 (ZZ Adaptivity)



(e) Mesh 3 (ERM Adaptivity)



(f) Mesh 3 (ZZ Adaptivity)

Figure 2.7: Global Efficiency Index of adapted meshes for problem P3

mesh size requirement is almost doubled. Estimated errors by the ERM on meshes with size of 200 nodes acts as an upper-bound and are asymptotically exact with further mesh refinement. For all cases, the efficiency index does not reach one because of the strong singularity of the solution derivative which contributed significantly to the norm used for measuring the errors.

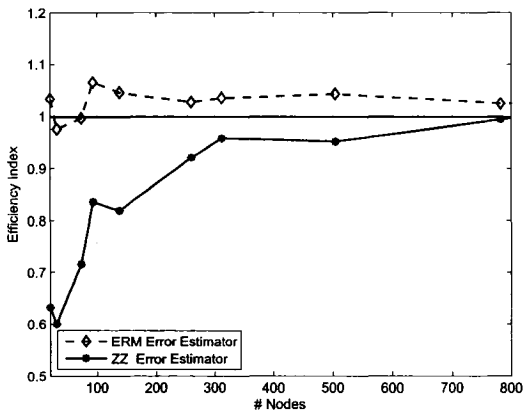
2.7 Conclusions

In this work, the performance of two local error estimation techniques for the finite element method was evaluated. From the results of the numerical experiments, the following conclusions are drawn:

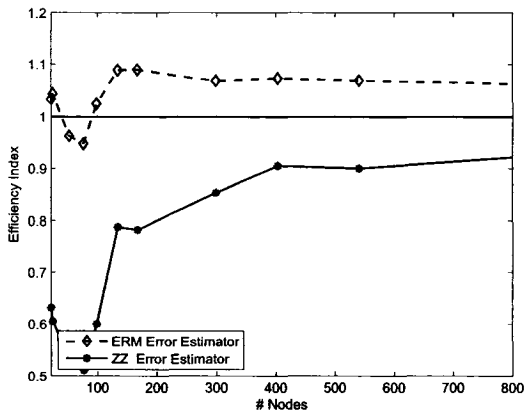
1. The ERM provides a reliable error estimation with an efficiency index close to the optimal value of unity in comparison to the ZZ error estimator. As the mesh is refined, higher quality solutions are obtained and the estimated errors using the ERM approach the exact error value.

2. For steady-state flow problems with data oscillation in the load term or discontinuity in the boundary, the ERM significantly outperforms the ZZ in estimating the errors. This deficiency in the ZZ method is attributed to the heuristic nature of the recovery operator which completely neglects the problem formulation. ERM has the advantage of approximating the residual equation and thus yields a more accurate error estimation.

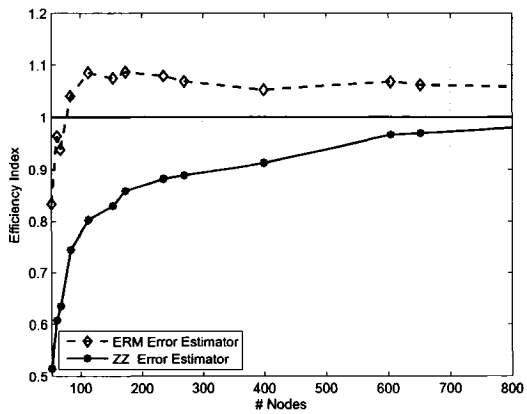
3. The unbalanced flux applied on the boundary of the local problems in the ERM reduces the reliability of the method on coarse meshes. However, by combining the ERM with a mesh adaptivity process, this procedure produces optimal meshes with a self-equilibrated local problems and provides a reliable error estimation with an efficiency index close to unity achieved after few adaptivity iterations.



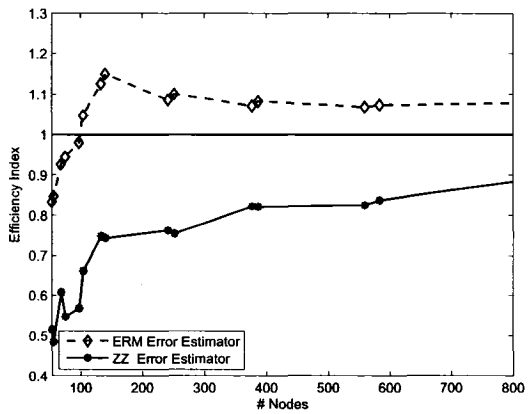
(a) $k = 1$ (ERM Adaptivity)



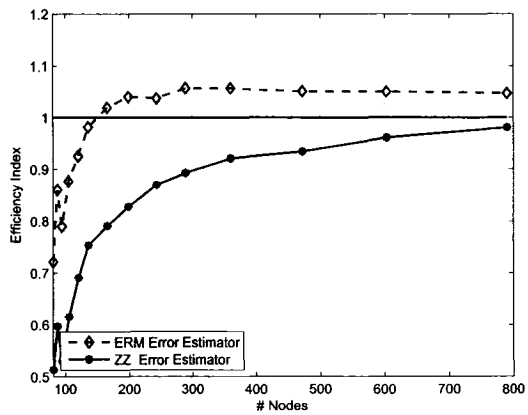
(b) $k = 1$ (ZZ Adaptivity)



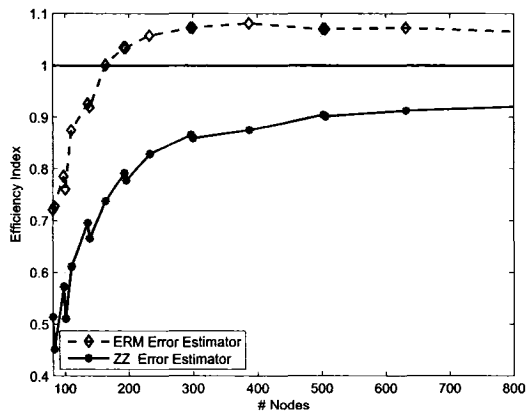
(c) $k = 3$ (ERM Adaptivity)



(d) $k = 3$ (ZZ Adaptivity)



(e) $k = 5$ (ERM Adaptivity)



(f) $k = 5$ (ZZ Adaptivity)

Figure 2.8: Global Efficiency Index of adapted meshes for problem P4

4. An adaptive algorithm combined with the ERM error estimator can be developed to solve steady-state flow problems with a pre-specified error bound

In Part II of this article series, the performance of both error estimators using elasticity problems is further evaluated.

ACKNOWLEDGEMENTS

This research was partially funded through grants from the Natural Science and Engineering Research Council of Canada (NSERC), the Ontario Graduate Scholarship Program (OGS) and the McMaster University's Centre for Effective Design of Structures. The authors acknowledge the use of SHARCNET's computing facilities.

Bibliography

[1] O. C. Zienkiewicz and J. Z. Zhu, "The superconvergent patch recovery and a posteriori error estimates. I. The recovery technique," *Internat. J. Numer. Methods Engrg.*, vol. 33, no. 7, pp. 1331–1364, 1992.

[2] O. C. Zienkiewicz and J. Z. Zhu, "The superconvergent patch recovery and a posteriori error estimates. II. Error estimates and adaptivity," *Internat. J. Numer. Methods Engrg.*, vol. 33, no. 7, pp. 1365–1382, 1992.

[3] M. Ainsworth and J. T. Oden, "A unified approach to a posteriori error estimation using element residual methods," *Numer. Math.*, vol. 65, no. 1, pp. 23–50, 1993.

[4] I. Babuška and W. C. Rheinboldt, "A-posteriori error estimates for the finite element method," *Int. J. Numer. Meth. Eng.*, vol. 12, pp. 1597–1615, 1978.

[5] P. Ladevèze and D. Leguillon, "Error estimate procedure in the finite element method and applications," *SIAM J. Numer. Anal.*, vol. 20, no. 3, pp. 485–509, 1983.

[6] R. E. Bank and A. Weiser, "Some a posteriori error estimators for elliptic partial differential equations," *Math. Comp.*, vol. 44, no. 170, pp. 283–301, 1985.

[7] O. C. Zienkiewicz and J. Z. Zhu, "A simple error estimator and adaptive procedure for practical engineering analysis," *Internat. J. Numer. Methods Engrg.*, vol. 24, no. 2, pp. 337–357, 1987.

[8] R. Becker and R. Rannacher, "A feed-back approach to error control in finite element methods: basic analysis and examples," *East-West J. Numer. Math.*, vol. 4, no. 4, pp. 237–264, 1996.

[9] M. Paraschivoiu, J. Peraire, and A. T. Patera, "A posteriori finite element bounds for linear-functional outputs of elliptic partial differential equations," *Comput. Methods Appl. Mech. Engrg.*, vol. 150, no. 1-4, pp. 289–312, 1997. Symposium on Advances in Computational Mechanics, Vol. 2 (Austin, TX, 1997).

[10] S. Prudhomme and J. T. Oden, "On goal-oriented error estimation for elliptic problems: application to the control of pointwise errors," *Comput. Methods Appl. Mech. Engrg.*, vol. 176, no. 1-4, pp. 313–331, 1999. New advances in computational methods (Cachan, 1997).

[11] H. Steeb, A. Maute, and E. Ramm, "Goal-oriented error estimation in solid mechanics," in *Error-controlled Adaptive Finite Elements in Solid Mechanics* (E. Stein, ed.), pp. 211–261, John Wiley & Sons, Chichester, 2002.

[12] R. A. Adams and J. F. Fournier, *Sobolev Spaces*, vol. 140 of *Pure and Applied Mathematics*. Boston: Academic Press, 2003.

[13] S. C. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods*, vol. 15 of *Texts in Applied Mathematics*. New York: Springer-Verlag, 1994.

[14] I. Babuska and T. Strouboulis, *The Finite Element Method and Its Reliability (Numerical Mathematics and Scientific Computation)*. Oxford University Press, 2001.

[15] M. Ainsworth and J. T. Oden, *A Posteriori Error Estimation in Finite Element Analysis*. John Wiley & Sons, 2000.

[16] R. Verfürth, *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques.* Advances in Numerical Mathematics, Wiley-Teubner, 1996.

[17] I. Babuška, T. Strouboulis, and C. S. Upadhyay, "η%-superconvergence of finite element approximations in the interior of general meshes of triangles," *Comput. Methods Appl. Mech. Engrg.*, vol. 122, no. 3-4, pp. 273–305, 1995.

[18] Z. Zhang and J. Zhu, "Analysis of the superconvergent patch recovery technique and a posteriori error estimator in the finite element method. I," *Comput. Methods Appl. Mech. Engrg.*, vol. 123, no. 1-4, pp. 173–187, 1995.

[19] Z. Zhang and J. Z. Zhu, "Analysis of the superconvergent patch recovery technique and a posteriori error estimator in the finite element method. II," *Comput. Methods Appl. Mech. Engrg.*, vol. 163, no. 1-4, pp. 159–170, 1998.

[20] P. Morin, R. H. Nochetto, and K. G. Siebert, "Local problems on stars: a posteriori error estimators, convergence, and performance," *Math. Comp.*, vol. 72, no. 243, pp. 1067–1097 (electronic), 2003.

[21] S. Prudhomme, F. Nobile, L. Chamoin, and J. Oden, "Analysis of a subdomain-based error estimator for finite element approximations of elliptic problems," *Numer Methods Partial Differential Equations*, vol. 20, no. 2, pp. 165–192, 2004.

[22] P. J. Roache, *Verification and Validation in Computational Science and Engineering.* Albuquerque, NM: Hermosa Publishers, 1998.

Chapter 3

Assessment of two a posteriori error estimators for FEM. Part II: Elasticity

A.H. ElSheikh, S. Smith and S.E. Chidiac

ABSTRACT

This paper forms the second part of a two-part paper on the evaluation of local a posteriori error estimation for FEM. The first part provided an assessment of the error estimators to stead-state flow problems, whereas this second part deals with elasticity problems. Problem formulation presented in Part I is extended to account for the coupled displacement field appearing in elasticity problems. The two error estimators, the element residual method (ERM) and Zienkiewicz-Zhu patch recovery technique (ZZ) are used as drivers for a mesh adaptation process. The results demonstrate the advantages of employing a posteriori error estimators for obtaining finite element solutions with a pre-specified error tolerance. Of the two methods, the ERM is shown to produce adapted meshes that are similar to those adapted by the exact error. Furthermore, the ERM provides higher quality estimates of the error in the global energy norm when compared to the ZZ estimator.

KEY WORDS: Adaptivity, A Posteriori Error Estimation, Element Residual Method, Finite Element Method, Elasticity, Zienkiewicz-Zhu patch recovery

3.1 Introduction

The finite element method is an optimal method for solving elasticity problems appearing in structural mechanics. It is optimal in the sense of minimizing the total potential energy of the system. Over the past decades a large amount of work has been done on error estimation for elasticity problems employing Zienkiewicz-Zhu (ZZ) error estimator [1, 2] and element residual method (ERM) [3, 4, 5]. However, the published results for the two methods were presented in different contexts. The ZZ was expressed using engineering notations and received wide acceptance in the engineering community, while the ERM was mostly presented in a mathematical context using functional analysis notations. To date no comprehensive analysis using both methods on practical problems exist where practitioners can appreciate the merits of the two methods. Moreover, the research community continued focus on the mathematical proofs has curbed the accessibility of error estimation technology by practicing engineers.

This paper begins by presenting the formulation of 2D elasticity problems using Lagrangian finite elements. The theoretical basis of the Zienkiewicz-Zhu (ZZ) error estimator and the Element Residual Method (ERM) is briefly reviewed and concisely formulated so that it can be applied to elasticity problems in a unified way. The performance of the two error estimators is tested numerically using four practical engineering problems. The problems are solved on a sequence of uniformly refined meshes to evaluate the efficiency of the error estimators. For those problems that possess areas with stress concentration or points of singularity, a mesh adaptation based on the error estimates is then applied to improve the convergence rates of the finite element solution.

3.2 Formulation of plane elasticity problems

We consider an elasticity problem defined over a bounded domain Ω in $\mathbb{R}^2$ with a Lipschitz-continuous boundary composed of a Dirichlet portion Γ_D and Neumann

portion Γ_N where, $\partial\Omega = \Gamma_D \cup \Gamma_N$ and $\Gamma_D \cap \Gamma_N = \varnothing$. It is required to find the displacement field $\mathbf{u} = [u_1, u_2]^T$, stress tensor components σ_{ij} and strain tensor components ε_{ij} for $i, j = 1, 2$ at any point $\mathbf{x} = [x_1, x_2]^T$ in the domain. Three equations relates these unknowns, the force equilibrium equation,

$$\sum_{j=1}^{2} \frac{\partial \sigma_{ij}}{\partial x_j} + f_i = 0 \text{ in } \Omega \text{ and } i = 1, 2 \tag{3.1}$$

where, f_i are the body force components of the force field $\mathbf{f} = [f_1, f_2]^T$. The second and third equations are the linear kinematic equation and the material constitutive equation formulated as:

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \qquad \sigma_{ij} = \lambda\left\{\sum_{k=1}^{2} \varepsilon_{kk}\right\}\delta_{ij} + 2\mu\varepsilon_{ij} \tag{3.2}$$

where, δ is the dirac-delta function and λ, μ are the Lamé constants. The boundary conditions $\Gamma_{D,i}$ and $\Gamma_{N,i}$ for $i = 1, 2$ are prescribed as:

$$u_i(\mathbf{x}) = u_{Di}, \qquad \text{on } \Gamma_D \quad \text{and } i = 1, 2 \tag{3.3}$$

$$\sum_{j=1}^{2} \sigma_{ij} n_j = g_i, \qquad \text{on } \Gamma_N \quad \text{and } i = 1, 2 \tag{3.4}$$

Using the previous relations, the Lamé-Navier's equation can be derived as:

$$\mu \sum_{k=1}^{2} \frac{\partial^2 u_i}{\partial x_k^2} + (\lambda + \mu)\frac{\partial}{\partial x_i}\sum_{k=1}^{2} \frac{\partial u_k}{\partial x_k} + f_i = 0 \text{ in } \Omega \quad \text{and } i = 1, 2 \tag{3.5}$$

For plane stress, Lamé constants can be expressed by Young's modulus E and Poisson's ration ν as: $\mu = E/2(1+\nu)$, $\lambda = E\nu/1-\nu^2$ and for plane strain, $\mu = E/2(1+\nu)$, $\lambda = E\nu/(1-2\nu)(1+\nu)$. Using standard norm notation for Sobolev and Hilbert spaces and equipped with scalar L^2 inner product $(.,.)$, the weak formulation

of Eq. (3.5) can be written as:

$$\begin{cases} \text{Find } \mathbf{u} \in U \text{ such that} \\ \mathcal{B}(\mathbf{u}, \mathbf{v}) = (\mathbf{f}, \mathbf{v}) + (\mathbf{g}, \mathbf{v})_{\Gamma_N} \qquad \forall \mathbf{v} \in V \end{cases} \tag{3.6}$$

where U is the displacement trial space and V is the test space and the different terms are defined as:

$$\mathcal{B}(\mathbf{u}, \mathbf{v}) = \int_{\Omega} \sigma(\mathbf{u}) \cdot \varepsilon(\mathbf{v}) \quad , \quad (\mathbf{f}, \mathbf{v}) = \int_{\Omega} \mathbf{f} \cdot \mathbf{v} \quad , \quad (\mathbf{g}, \mathbf{v})_{\Gamma_N} = \int_{\Gamma_N} \mathbf{g} \cdot \mathbf{v} \tag{3.7}$$

where, $\mathbf{f}$ is the load vector and $\mathbf{g}$ the boundary traction vector. For isotropic materials, the constitutive relation is simplified and written in vector format as $\sigma(\mathbf{u}) = \mathbf{C}\ \varepsilon(\mathbf{u})$. In which the stress and stain field is defined as:

$$\sigma = [\ \sigma_{11} \quad \sigma_{12} \quad \sigma_{21} \quad \sigma_{22}\ ]^T, \quad \varepsilon = [\ \varepsilon_{11} \quad \varepsilon_{12} \quad \varepsilon_{21} \quad \varepsilon_{22}\ ]^T \tag{3.8}$$

with the following definition of the material stiffness matrix $\mathbf{C}$,

$$\mathbf{C} = \begin{bmatrix} \lambda + \mu & 0 & 0 & \lambda \\ 0 & \mu & \mu & 0 \\ 0 & \mu & \mu & 0 \\ \lambda & 0 & 0 & \lambda + \mu \end{bmatrix} \tag{3.9}$$

In order to provide a general formulation for any system of two coupled variables, the present formulation was selected instead of the more general form by eliminating the repeated rows and using a 3×3 matrix with the corresponding engineering strain. This allows an easy extension of the theory presented in Part I to elasticity problems.

The strain vector $\varepsilon(\mathbf{u})$, is expressed as:

$$\varepsilon(\mathbf{u}) = \begin{bmatrix} \frac{\partial u_1}{\partial x_1} \\ \frac{\partial u_1}{\partial x_2} \\ \frac{\partial u_2}{\partial x_1} \\ \frac{\partial u_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1} & 0 \\ \frac{\partial}{\partial x_2} & 0 \\ 0 & \frac{\partial}{\partial x_1} \\ 0 & \frac{\partial}{\partial x_2} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \mathbf{Bu} \tag{3.10}$$

where the differential operator $\mathbf{B}$ acts on the displacement field $\mathbf{u}$. Using this definition, the bilinear form of the linear elasticity problem is formulated as:

$$\mathcal{B}(\mathbf{u}, \mathbf{v}) = \int_{\Omega} (\mathbf{Bu})^T \mathbf{C} (\mathbf{Bv}) \tag{3.11}$$

Projecting the solution into the current domain discretization, corresponds to the approximation of $\mathbf{u}$ by $\mathbf{u_h}$ and $\mathbf{v}$ by $\mathbf{v_h}$ and follows directly from the variational formulation presented in Eq. (3.6).

$$\begin{cases} \text{Find } \mathbf{u_h} \in U_h \text{ such that} \\ \mathcal{B}(\mathbf{u_h}, \mathbf{v_h}) = (\mathbf{f}, \mathbf{v_h}) + (\mathbf{g}, \mathbf{v_h})_{\Gamma_N} \qquad \forall \mathbf{v_h} \in V_h \end{cases} \tag{3.12}$$

3.3 A Posteriori Error Estimation

In the following subsections, the formulation of both ZZ Method and the ERM is presented for elasticity problems in $\mathbb{R}^2$. All the notations are consistent with the previous presentation in Part I.

3.3.1 Recovery Based Error Estimators

The finite element solution of elasticity problems using first order Lagrangian elements results in a continuous displacement field with discontinuous stresses across element boundaries. This discontinuity in the stress field is expected because the weak form of the force equilibrium equation is solved instead of the strong (original) equation. The continuity requirements of the stress field is only applied in the

weak sense (under the integral). Zienkiewicz and Zhu [1] proposed a method for recovering a continuous stress field by solving least square problems over a patch of elements. This recovered field is assumed to be more accurate than the discontinuous stress field. Based on that assumption, the error is estimated/calculated using the recovered stress field instead of the exact stress field in the error equations.

In general, for coupled problems in $\mathbb{R}^2$, four least square problems need to be solved to recover two continuous derivatives of both components of the field. This is true for elasticity problems as well, but because of the special properties of the coupling matrix $\mathbf{C}$ where the second and the third rows are identical, only three components of the derivative need to be recovered. This is the same reason why the constitutive matrix for plane elasticity can be reduced to a 3×3 matrix with the corresponding reduction of the stresses into the three stress $\sigma_{xx}, \sigma_{yy}, \gamma_{xy}$.

The mathematical formulation of the error takes the following form:

$$\|\mathbf{e}\|^2 = \|\mathbf{u} - \mathbf{u_h}\|^2 = \mathcal{B}(\mathbf{u} - \mathbf{u_h}, \mathbf{u} - \mathbf{u_h}) \approx \int_\Omega (G_h[\mathbf{u_h}] - \mathbf{Bu})^T \mathbf{C} (G_h[\mathbf{u_h}] - \mathbf{Bu}) \quad (3.13)$$

where $G_h[\mathbf{u_h}]$ is the recovered gradient reconstructed from the finite element solution $\mathbf{u_h}$. The global error is then approximated as the sum of local element contributions, as follows:

$$\|e\|^2 = \sum_{k=1}^{N} \left\{ (G_h[\mathbf{u_h}] - \mathbf{Bu})_k^T \mathbf{C}_k (G_h[\mathbf{u_h}] - \mathbf{Bu})_k \right\} \quad (3.14)$$

where N is the total number of elements and the recovery operator $G_h[u_h]$ is based on solving local least square problems.

3.3.2 Implicit Residual type Error Estimators

In the Element Residual Method, errors are approximated by solving the residual equations using higher order basis functions. The global residual equation is localized into a set of element based problems and the total error is calculated as the sum of each element contribution [3]. One advantage of this approach is that local problems will

reflect the properties of the elasticity problem under consideration and is formulated as:

$$\mathcal{B}_k(\mathbf{e},\mathbf{v}) = (f,\mathbf{v}) - \mathcal{B}_k(\mathbf{u_h},\mathbf{v}) + \int_{\partial k} \langle \mathbf{t}_k \rangle \, \mathbf{v} \tag{3.15}$$

where the subscript k indicates the restriction of the integration over an element k, $\mathbf{t}_k$ the average traction on each edge of an element k and is defined as:

$$\langle \mathbf{t}_{ke} \rangle = \begin{cases} \frac{1}{2}\, n_k\, (\mathbf{t}_k + \mathbf{t}'_k) & \text{on } \partial k \cap \partial k' \\ n_k\, \mathbf{t}_k & \text{on } \partial k \cap \Gamma_D \\ \mathbf{g} & \text{on } \partial k \cap \Gamma_N \end{cases} \tag{3.16}$$

in which $\mathbf{t}_g$ is the traction on the Neumann part of the problem boundary. Each average traction $\langle \mathbf{t}_k \rangle$ has two components corresponding to the two displacement components u_x, u_y. After calculating the errors by solving N problems corresponding to N elements, the global error is calculated according to:

$$\|\mathbf{e}\|^2 = \sum_{k=1}^{N} \mathcal{B}_k(\mathbf{e},\mathbf{e}) \Rightarrow \sum_{k=1}^{N} \left\{ (\mathbf{Be})_k^T \mathbf{C}_k (\mathbf{Be})_k \right\} \tag{3.17}$$

3.4 Numerical Examples

Four elasticity problems are solved to evaluate the performance of both ZZ and ERM on a series of uniformly refined meshes. The errors are measured in the global energy norms. The ratio of the estimated error based on ERM or ZZ to the exact error is called the efficiency index of the error estimator.

3.4.1 Elasticity problem E1

This engineering problem describes the stress distribution in an infinite plate with a hole in the middle. Due to the symmetry of both the problem domain and loading configuration only one quarter of the domain needs to be solved. Essential boundary

conditions are applied on the lines of symmetry, while traction boundary conditions representing far field effects are applied on the rest of the boundary using the exact values given by in [6]:

$$\begin{aligned}
\sigma_{xx} &= 1 - \frac{r^2}{R^2}\left(\frac{3}{2}\cos(2\theta) + \cos(4\theta)\right) + \frac{3r^4}{2R^4}\cos(4\theta) \\
\sigma_{yy} &= -\frac{r^2}{R^2}\left(\frac{1}{2}\cos(2\theta) - \cos(4\theta)\right) - \frac{3r^4}{2R^4}\cos(4\theta) \\
\sigma_{xy} &= -\frac{r^2}{R^2}\left(\frac{3}{2}\sin(2\theta) + \sin(4\theta)\right) + \frac{3r^4}{2R^4}\sin(4\theta)
\end{aligned} \tag{3.18}$$

where θ is the angle from a point to the x axis, r the radius of the hole and R the distance from the point of evaluation to the center of the hole. Figure 3.1 shows a schematic plot of the model used as well as the initial mesh.

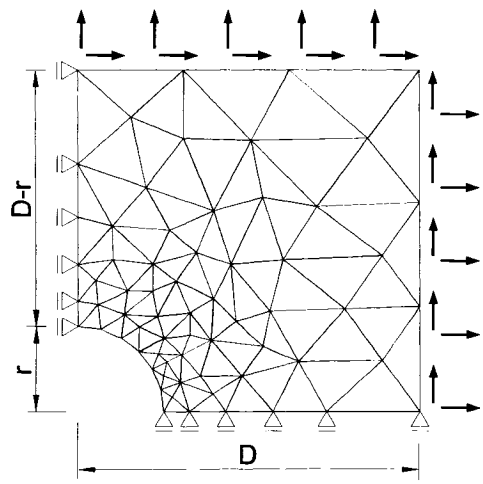
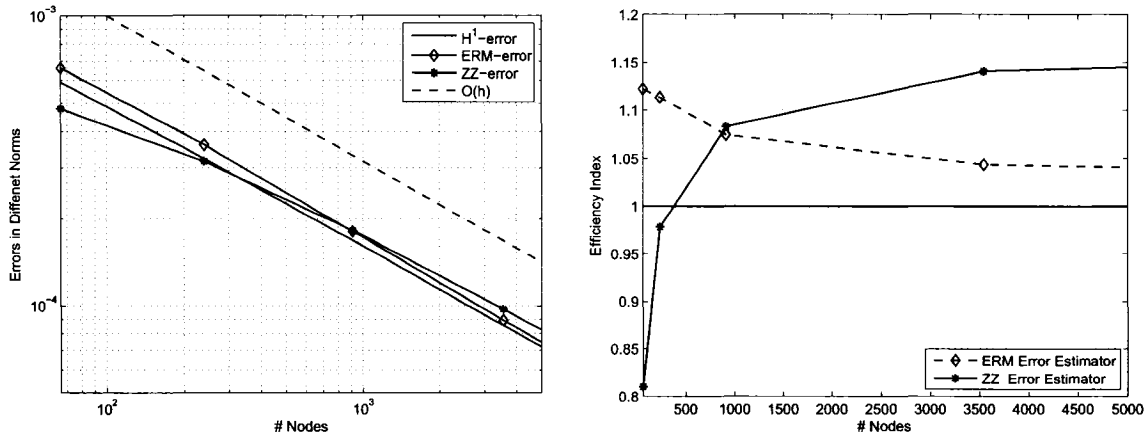


Figure 3.1: Schematic figure and initial mesh for problem E1

The estimated error in the global energy norm using the ERM and ZZ method is shown in Fig. 3.2 along with the the exact errors referred to as H^1 error. Due to the smoothness of the solution, convergence rate of errors in the energy norm is of order $\mathcal{O}(h)$. The ERM method overestimates the errors for coarse meshes while the ZZ method underestimates the error in the global norm. With the global mesh



(a) Errors using uniform adaptivity (b) Efficiency Index of the estimated errors

Figure 3.2: Results on uniformly refined meshes for problem E1

refinement, the errors estimated using ERM gains more accuracy and the efficiency index asymptotically approaches one, while the ZZ goes from underestimating the errors to overestimating the error as the mesh gets more refined. The efficiency index for ZZ is 1.15 even with meshes of 5000 nodes while the corresponding efficiency index for ERM is 1.04.

3.4.2 Elasticity problem E2

An L-Shaped plane elastic body with a unit thickness is analyzed for internal stress. The geometry and boundary conditions of the problem are shown in Fig. 3.3. The plate is loaded using traction forces that satisfies the exact solution according to Reference [6]. The stress components used to load the body along the sides AB, BC, CD and FA are:

$$\begin{aligned} \sigma_x &= \lambda r^{(\lambda-1)}\big((2-Q1(\lambda+1))cos((\lambda-1)\theta)-(\lambda+1)cos((\lambda-3)\theta)\big) \\ \sigma_y &= \lambda r^{(\lambda-1)}\big((2+Q1(\lambda+1))cos((\lambda-1)\theta)+(\lambda-1)cos((\lambda-3)\theta)\big) \\ \tau_{xy} &= \lambda r^{(\lambda-1)}\big((\lambda-1)sin((\lambda-3)\theta)+Q(\lambda+1)sin((\lambda-1)\theta)\big) \end{aligned} \quad (3.19)$$

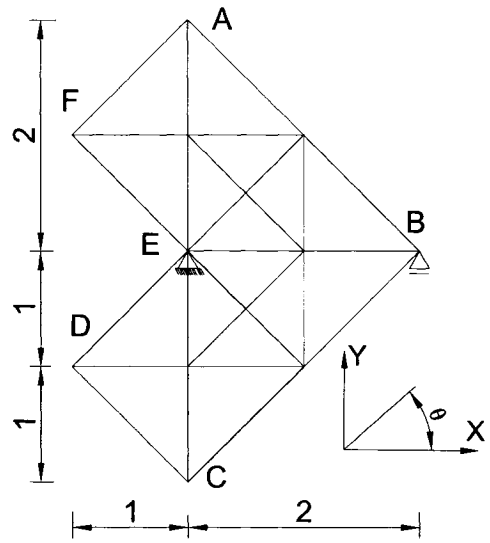
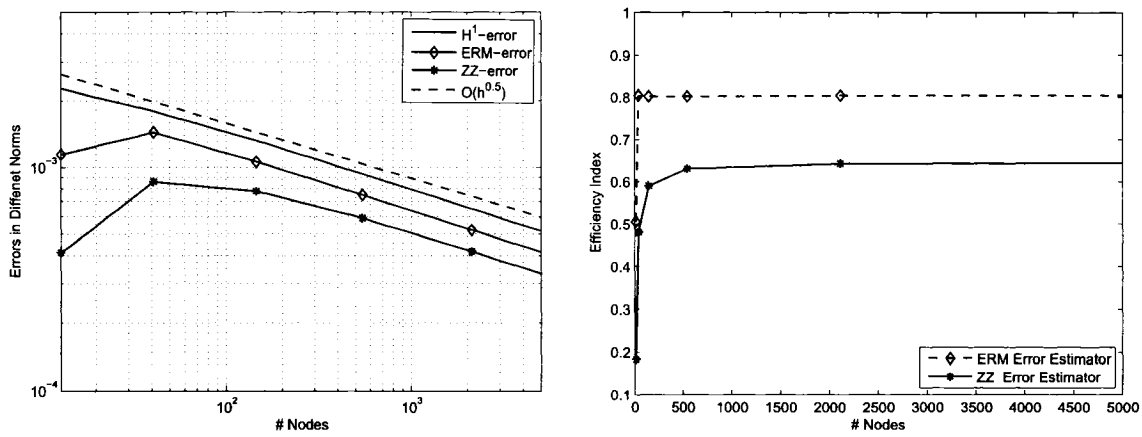


Figure 3.3: Schematic figure and initial mesh for problem E2

where $Q = 0.543075579$ and $\lambda = 0.544483737$ is the solution of $sin(\lambda\theta) + \lambda sin(\theta) = 0$ with $\theta = \frac{3\pi}{2}$. The solution of this problem yields a singular stress component at the reentrance point E corresponding to the position where $r = 0$. The influence of this singularity on the convergence rate is evaluated using uniform mesh refinement.

In Fig. 3.4-a, the exact error in the energy norm is plotted against the number of nodes. A sequence of meshes generated by dividing each element into four elements in a uniform refinement process. This global refinement produces an order of convergence proportional to $\mathcal{O}(h^{0.5})$ due to stress singularity at point E. The estimated errors on uniform meshes are found to be not reliable as illustrated by an efficiency index of 0.85 for the ERM and 0.65 for the ZZ error estimator plotted in Fig. 3.4-b. The refined mesh fails to capture the singularity because the singularity is localized at a length scale much smaller than the globally refined element size. These results demonstrate that it is not feasible nor practical to obtain a globally refined mesh capable of capturing this singularity. Improvements in the order of convergence, due to mesh adaptivity are explained later in this paper.



(a) Errors using uniform adaptivity (b) Efficiency Index of the estimated errors

Figure 3.4: Results on uniformly refined meshes for problem E2

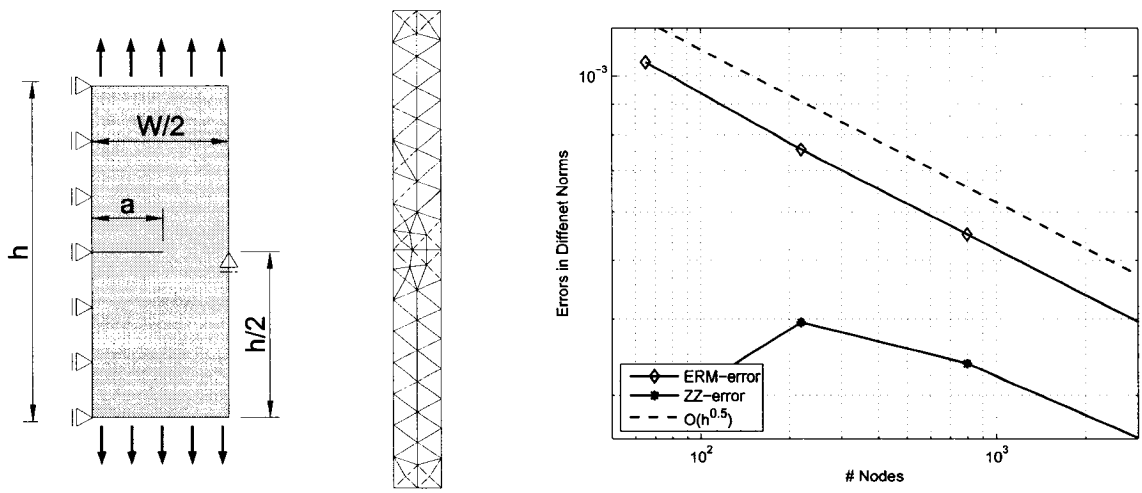
3.4.3 Elasticity problem E3

The third engineering problem models a center-cracked plate. Due to symmetry, only half of the domain is solved using the appropriate boundary conditions, as shown in Fig. 3.5-a. The normalized dimensions used are W/a=5 and h/a=25, where h the height of the plate, W the width of the plate and a the crack length. The crack tip produces a $\frac{1}{\sqrt{r}}$ singularity. The maximum stress near the crack tip is expressed in terms of the stress intensity factor K. For the current loading configuration

$$K_I = \lim_{r \to 0} \sigma_{yy} \sqrt{2\pi r} = \sigma_{yy} \sqrt{\pi a} \tag{3.20}$$

The stress intensity factor is calculated from the finite element results based on an approximation of the displacement gradient near the crack tip using the displacement correlation method [7]. The initial mesh is plotted in Fig. 3.5-b, where no special consideration is made around the crack tip to show how the mesh adaptivity process will capture the singularity.

The calculated error in the global energy norm is shown in Fig. 3.5-c. The convergence rate is also of order $\mathcal{O}(h^{0.5})$ due to stress singularity at the crack tip. The



(a) Schematic figure (b) Initial mesh (c) uniform adaptivity

Figure 3.5: Initial mesh and results on uniformly refined meshes for problem E3

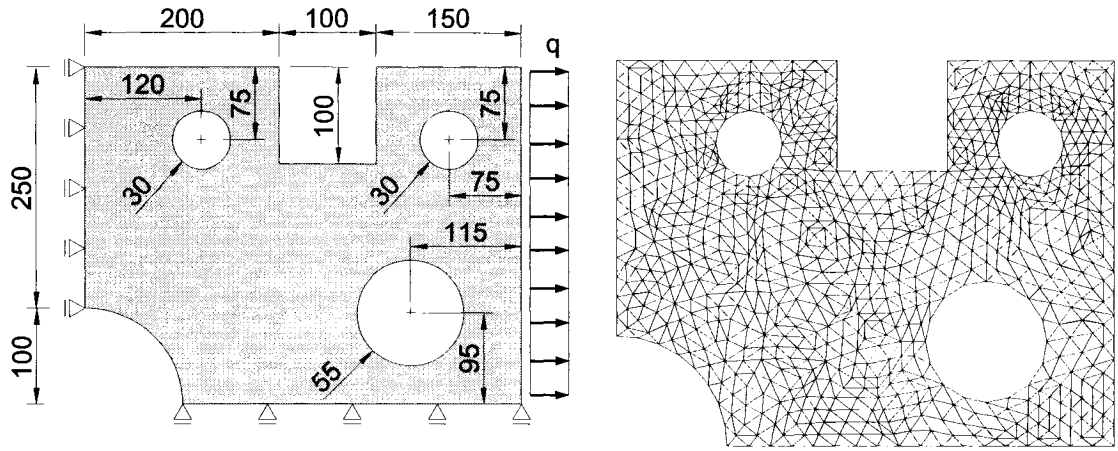
Table 3.1: Stress intensity factor (SIF) for problem E3 on uniformly refined meshes

# nodes	SIF
65	0.85625
219	1.11418
797	1.26810
3033	1.34506
Exact SIF	1.12099

stress intensity factor (SIF) is compiled in Table 3.1. For the initial mesh, SIF is far below the exact result and as the mesh size increases, the SIF is overestimated. Again, the results reveal that uniform mesh refinement fails to capture the correct SIF within a reasonable mesh size. The match of the result for the second globally refined mesh process is regarded as a transition from underestimating the SIF to meshes that overestimate the SIF.

3.4.4 Elasticity problem E4

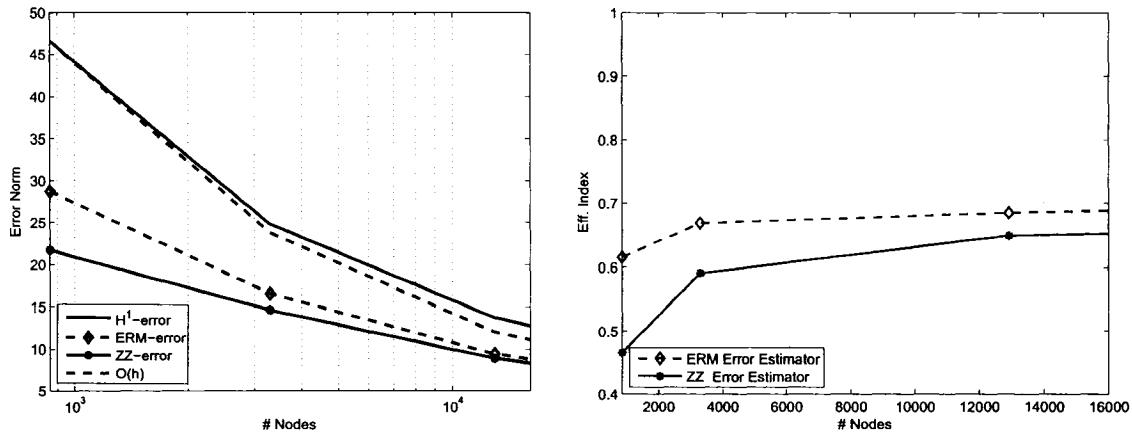
This problem simulates a plate under plane-strain. The geometry is based on similar model in [8] where the lower boundary is restrained in the y-direction and the left boundary is restrained in the x-direction. A traction load is imposed on the right boundary as shown in Fig. 3.6, where all dimensions are in SI units and $q = 100N/m$. The errors are calculated by comparison to a reference solution on a very fine mesh with 74046 nodes.



(a) Problem geometry (b) Initial mesh

Figure 3.6: Problem E4 geometry and initial mesh

In Fig. 3.7-a, the exact error in the energy norm is plotted against the number of nodes for a sequence of uniformly refined meshes. Due to the singularity of the stress at the corners, the errors are underestimated by both error estimators. The efficiency index of the estimated error using ERM performs slightly better than the ZZ method but still underestimates the error by a ratio of 0.7 for the most refined mesh as shown in Fig. 3.7-b.



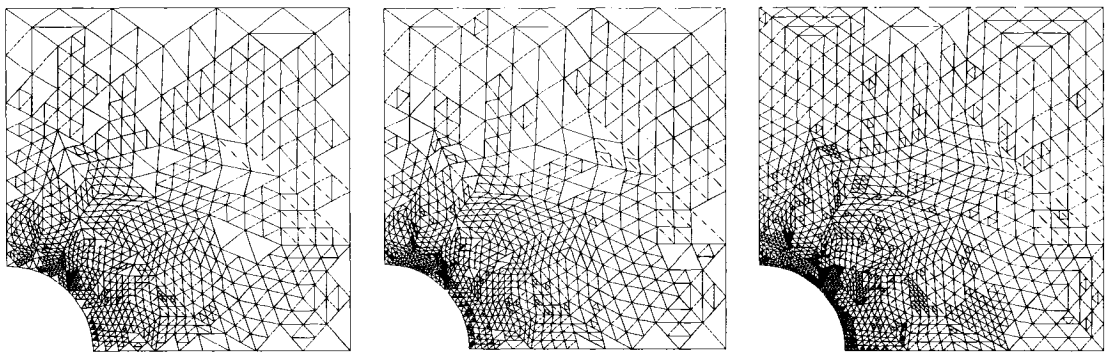
(a) Errors using uniform adaptivity (b) Efficiency Index of the estimated errors

Figure 3.7: Results on uniformly refined meshes for problem E4

3.5 Mesh adaptivity

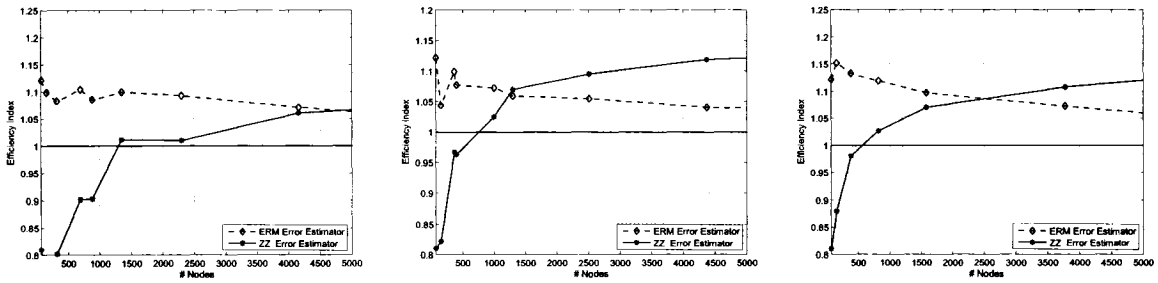
The errors estimated according to the ERM and ZZ provide valuable information that can guide a mesh adaptation process. To optimize the number of nodes or unknowns for a given discretization error, only regions with high local errors are marked for refinement. A set of elements with an error value more than a pre-specified fraction of the maximum element error is marked for refinement. Because of the existence of a singularity in the solution for problems E2 and E3, elements close to the singular points will have relatively large error in comparison to the rest of the domain. To overcome this problem, the refinement fraction was set to 0.2 for all of the four problems to allow a more aggressive refinement than just a few elements that are close to the singular point. This decision results in fewer refinement iterations.

Figure 3.8 shows the adapted meshes for problem E1 using the two error estimators as well as the exact error calculated using the exact analytical solution. The error distribution is concentrated close to the hole where the stress concentration exists. Another reason for the relative large error values around the hole in comparison to the rest of the domain is the approximation of the curved boundary by linear elements. It can be observed that the implemented refinement algorithm corrects the boundary



(a) ZZ error estimator (b) ERM error estimator (c) Exact error

Figure 3.8: Adapted meshes using different adaptivity drivers for problem E1 - After 5 iterations



(a) ZZ error estimator (b) ERM error estimator (c) Exact error

Figure 3.9: Efficiency indices of the estimated errors using different adaptivity drivers for problem E1

representation errors by projecting the newly inserted nodes during mesh refinement to the correct position on the curved boundary. The performance of the two error estimators on adapted meshes is shown in Fig. 3.9. The efficiency indices for the ERM start by overestimating the errors and asymptotically approach the optimal value of one with mesh adaptation. These results do not differ from the results on uniformly refined meshes shown in Fig. 3.2, because of the smoothness of the solution of this problem. In comparison, the efficiency index for ZZ starts by underestimating the error and then overestimates the error with mesh adaptation. The ZZ efficiency index does not converge to the optimal value of one.

Figure 3.10 plots the exact errors in the energy norm on different meshes generated

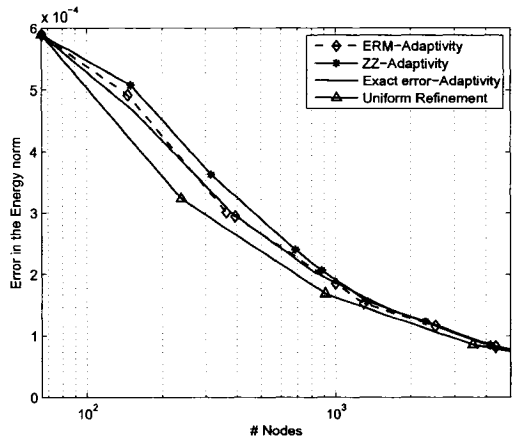
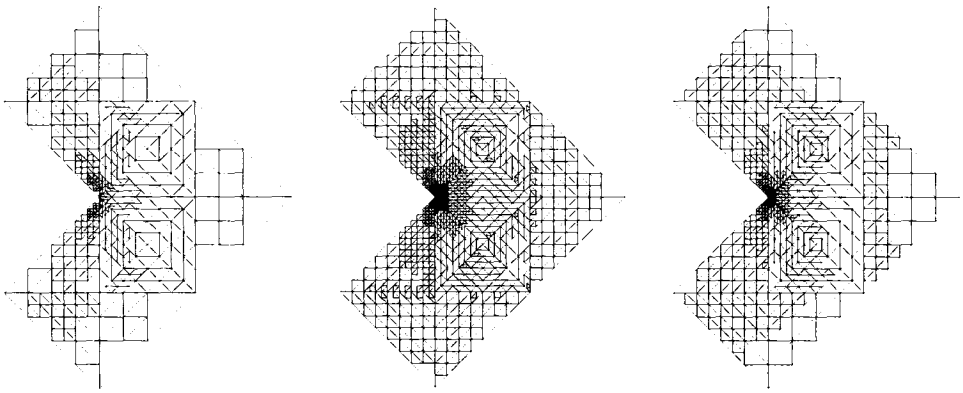


Figure 3.10: Convergence of the global error for problem E1 using different adaptivity drivers

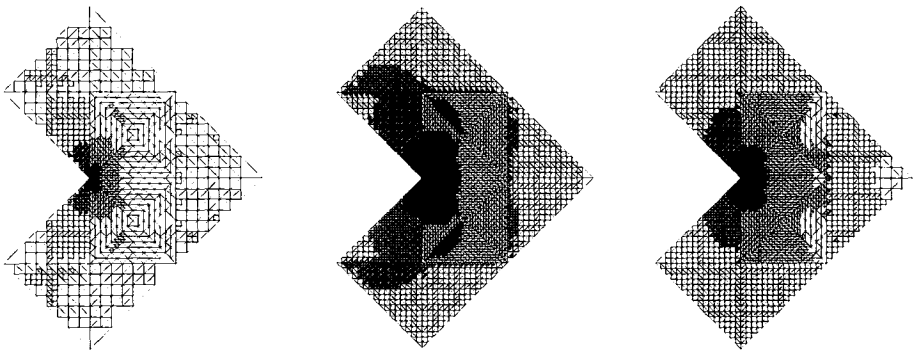
by different refinement criteria. Due to the uniformity of the errors and smoothness of the solution, uniform refinement outperforms adaptive refinement in reducing the errors in the global norm. The results also show no significant difference between ERM and ZZ in comparison to each other, or in comparison to the mesh adapted according to exact errors obtained using the analytical solution.

For problem $E2$, Figures 3.11 and 3.12 show the adapted meshes using different refinement criteria after 10 and 15 iterations, respectively. The singularity of the solution is detected after a few mesh adaptation steps as mesh density gets higher at the re-entrant point. The mesh adapted according to ZZ is less refined over the domain with more refined elements at the singular point. This shows that the errors at elements close to the singular point are much higher than over the rest of the domain and thus most of the elements are not marked for refinement given that the local error is less than 0.2 of the maximum element error. Meshes adapted according to the element residual method are very close to those adapted by the exact errors. This supports the claim that the distribution of errors detected by ERM is similar to that of the exact errors. In regard to the efficiency of the estimated error in the global norm, the ERM method outperforms the ZZ on all the meshes even when the



(a) ZZ error estimator (b) ERM error estimator (c) Exact error

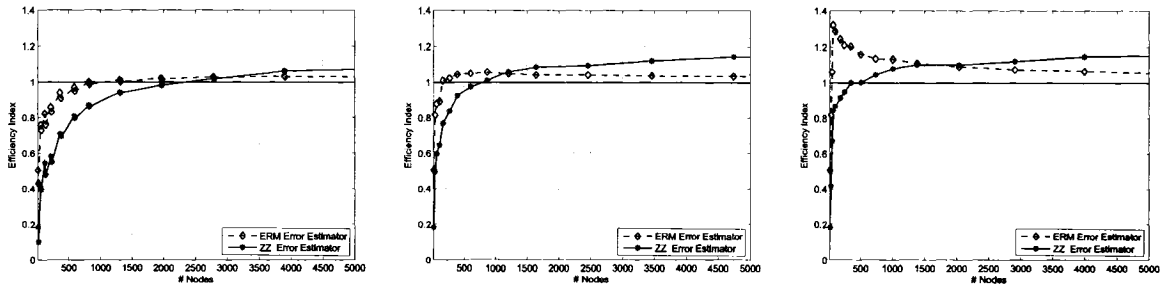
Figure 3.11: Adapted meshes using different adaptivity drivers for problem E2 - After 10 iterations



(a) ZZ error estimator (b) ERM error estimator (c) Exact error

Figure 3.12: Adapted meshes using different adaptivity drivers for problem E2 - After 15 iterations

meshes are generated by the ZZ error estimator, as shown in Figure 3.13. The results show that the ERM method is reliable in estimating the errors on meshes with a few hundred nodes. Using the ERM both for adaptation and estimating the error is shown to outperform the estimation on meshes generated by the exact error. This is attributed to the fact that the calculated errors and the adapted meshes both try to minimize the traction imbalance in local element by element problem, as well as the error in the energy norm. Minimizing the local problem traction imbalance increases the accuracy of the estimated errors.



(a) ZZ error estimator (b) ERM error estimator (c) Exact error

Figure 3.13: Efficiency indices of the estimated errors using different adaptivity drivers for problem E2

Figure 3.14 shows the effect of the mesh adaptation on the reduction of the error in the energy norm. Due to the singularity, all adapted meshes by ERM, ZZ and the exact error, clearly outperform uniform mesh refinement in the error reduction. For coarser meshes, the ERM yields superior results in comparison with ZZ error estimator in reducing the errors.

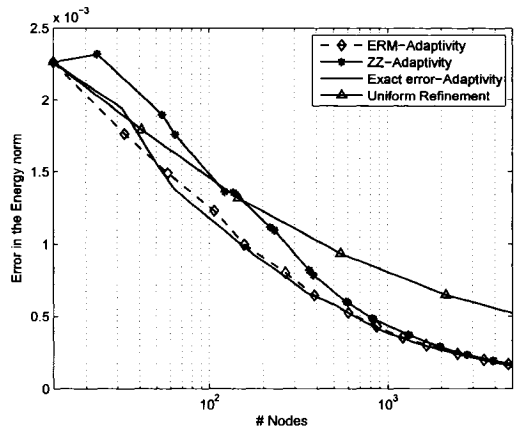
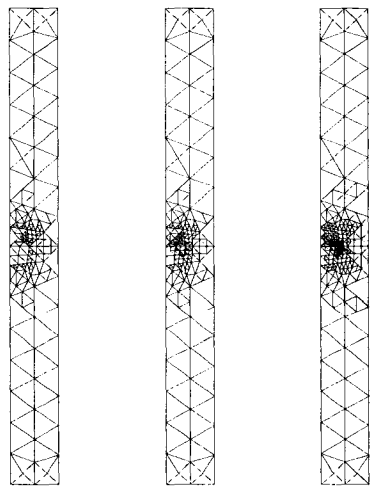


Figure 3.14: Convergence of the global error for problem E2 using different adaptivity drivers

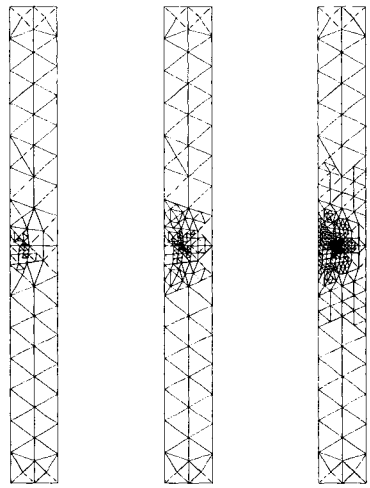
Figures 3.15 and 3.16 show the adapted meshes for problem E3 using ZZ and ERM, respectively. Both error estimates succeeded in capturing the singularity at the crack tip even with the initial coarse mesh. Visual inspection of the adapted

meshes provides qualitative comparison. For this problem, the stress intensity factor needs to be calculated for a quantitative comparison. Figure 3.17 shows the stress intensity factor calculated using the results obtained from uniformly adapted meshes, mesh adapted using ZZ and ERM error estimator and the exact value of the SIF. The ERM based adaptivity manages to predict the SIF with an error of 2%, while SIF computed on the same mesh size produced by uniform adaptivity or ZZ error estimator is found to be significantly larger than the analytical value. Figure 3.18, part a and b shows, the convergence rates of the global error norm. Even with the presence of the singularity, the optimal $\mathcal{O}(h)$ rate of convergence was achieved after a few mesh adaption steps.



(a) 3 iter. (b) 5 iter. (c) 8 iter.

Figure 3.15: Adapted meshes for problem E3 using the ZZ error estimator



(a) 3 iter. (b) 5 iter. (c) 8 iter.

Figure 3.16: Adapted meshes for problem E3 using the ERM error estimator

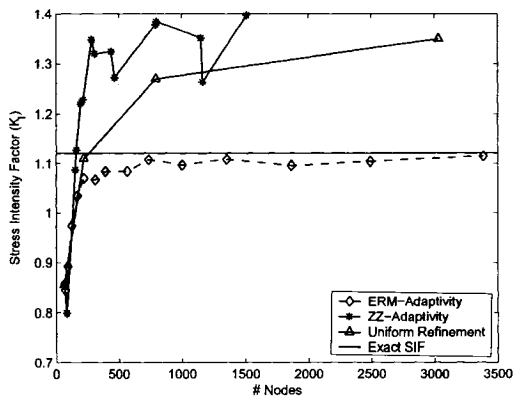
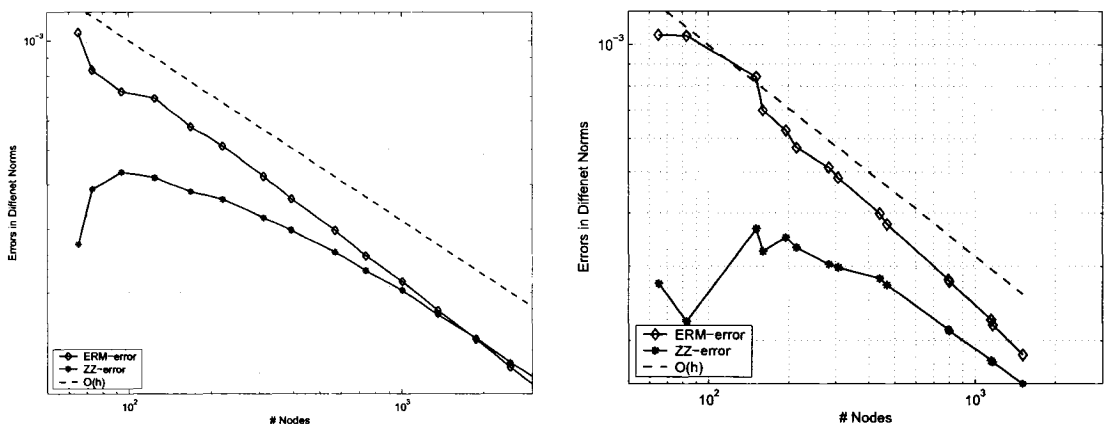


Figure 3.17: Stress Intensity factors using different adaptivity drivers for problem E3



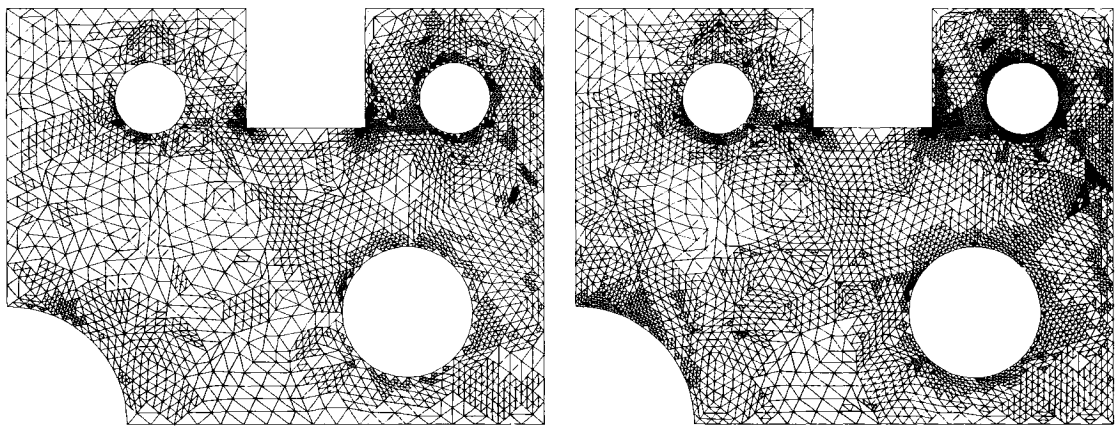
(a) ERM adaptivity (b) ZZ adaptivity

Figure 3.18: Convergence of the global error norm for problem E3

Figure 3.19 shows the adapted meshes for problem E4 using ZZ and ERM, respectively after 8 iterations. Both error estimates succeeded in capturing the corner singularity, but computed exact error on the mesh produced by the ERM method is 65% of the mesh refined by the ZZ estimator. That reduction corresponds to the mesh size of 6069 nodes and 3570 nodes for the ERM produced mesh and the ZZ driven mesh, respectively. Figure 3.20 shows the efficiency index of the estimated errors on ERM and ZZ adapted meshes. Mesh adaptivity improved the efficiency index from the 0.7 corresponding to uniform refinement, to approximately 0.85 for the maximum mesh size of 16000.

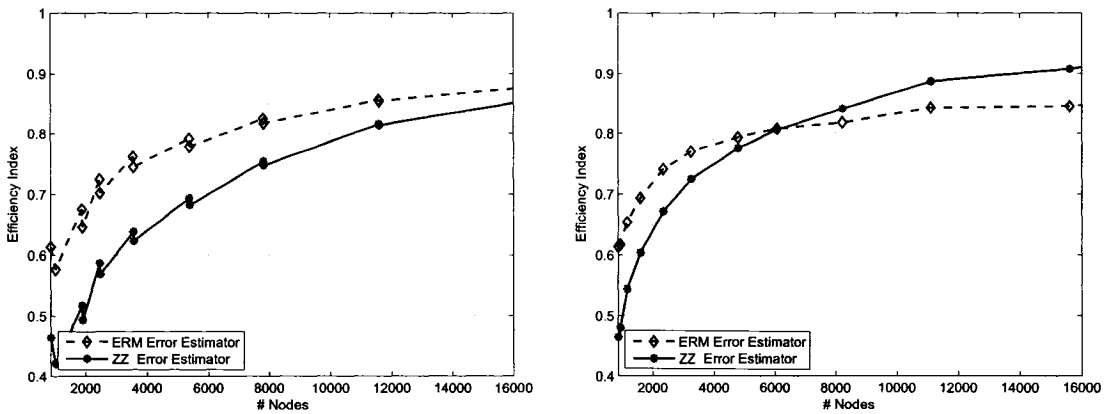
3.6 Conclusions

In this work, the performance of the element residual method and the ZZ error estimator has been investigated for elasticity problems. Two parameters are of great interest when evaluating a posteriori error estimator, the ratio of the estimated errors to the exact error (efficiency index) and the ability of the estimated errors to guide a mesh adaptation process for producing optimal meshes that minimize the global error. On the basis of the results obtained for all the elasticity problems tested, the



(a) ZZ error estimator (b) ERM error estimator

Figure 3.19: Adapted meshes for problem E4 (after 8 iterations)



(a) ZZ error estimator (b) ERM error estimator

Figure 3.20: Efficiency indices of the estimated errors using different adaptivity drivers for problem E4

following conclusions are drawn:

1. Uniform mesh refinement is only adequate for problems with smooth solutions.

2. The produced meshes using ERM adaptivity are very close to those adapted using the exact error. This shows that ERM yields a distribution of errors that is similar to that of the exact error.

3. In general, ERM yields superior error estimation in comparison with ZZ and the difference is most significant for coarser meshes. Also, the ERM converges steadily to the exact error with adaptive mesh refinement, while the ZZ goes from underestimating the errors to over estimating the errors in most of the cases.

4. Solutions based on ERM adaptivity is capable of predicting the SIF with an error of 2%, while solutions based on ZZ adaptivity or uniform mesh refinement yield erroneous results for SIF.

5. An adaptive algorithm combined with the ERM error estimator can be developed to reliably solve engineering problems that possess singularities with a pre-specified error bound.

ACKNOWLEDGEMENTS

This research was partially funded through grants from the Natural Science and Engineering Research Council of Canada (NSERC), the Ontario Graduate Scholarship Program (OGS) and the McMaster University's Centre for Effective Design of Structures. The authors acknowledge the use of SHARCNET's computing facilities.

Bibliography

[1] O. C. Zienkiewicz and J. Z. Zhu, "The superconvergent patch recovery and a posteriori error estimates. I. The recovery technique," *Internat. J. Numer. Methods Engrg.*, vol. 33, no. 7, pp. 1331–1364, 1992.

[2] O. C. Zienkiewicz and J. Z. Zhu, "The superconvergent patch recovery and a posteriori error estimates. II. Error estimates and adaptivity," *Internat. J. Numer. Methods Engrg.*, vol. 33, no. 7, pp. 1365–1382, 1992.

[3] M. Ainsworth and J. T. Oden, "A posteriori error estimation in finite element analysis," *Comp. Meth. Appl. Mech. Engrg.*, vol. 142, pp. 1–88, 1997.

[4] I. Babuska and T. Strouboulis, *The Finite Element Method and Its Reliability (Numerical Mathematics and Scientific Computation).* Oxford University Press, 2001.

[5] R. Verfürth, *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques.* Advances in Numerical Mathematics, Wiley-Teubner, 1996.

[6] B. Szabo and I. Babuska, *Finite Element Analysis.* New York: J. Wiley & Sons, 1991.

[7] M. H. Aliabadi and D. Rook, *Numerical Fracture Mechanics.* Kluwer Academic Publishers, 1991.

[8] G. H. Paulino, I. F. M. Menezes, J. B. Cavalcante Neto, and L. F. Martha, "A methodology for adaptive finite element analysis: Towards an integrated computational environment," *Computational Mechanics*, vol. 23, pp. 361–388, 1999.

Chapter 4

Numerical investigation of the reliability of a posteriori error estimation for advection diffusion equations

A.H. ElSheikh, S. Smith and S.E. Chidiac

ABSTRACT

A numerical investigation of the reliability of a posteriori error estimation for advection diffusion equations is presented. The estimator used is based on the solution of local problems subjected to Neumann boundary conditions. The estimated errors are calculated in a weighted energy norm, a stability norm and an approximate fractional order norm in order to study the effect of the error norm on both the effectivity index of the estimated errors and the mesh adaptivity process. The reported numerical results are in general better than what is available in the literature. The results reveale that the reliability of the estimated errors depends on the relation between the mesh size and the size of local features in the solution. The stability norm is found to have some advantages over the weighted energy norm in terms of producing effectivity indices closer to the optimal unit value, especially for problems with internal sharp layers. Meshes adapted by the ERM measured in the stability norm conform to the sharp layers and are shown to be less dependent on the wind direction.

KEY WORDS: Adaptive Finite Element; A Posteriori Error Estimates; Advection Diffusion Equations; Error Norm

4.1 Introduction

Reliable error estimation for the finite element method has been the focus of intense research during the past decade. A review of the subject can be found in [1, 2, 3] and the references therein. A posteriori error estimators provide a measure to assess the accuracy of the solution and to drive a mesh adaptation process. Error estimation techniques can generally be divided into implicit and explicit methods. Explicit methods use approximate formulas to estimate the residual in the element interior and on the element boundaries [4, 5]. These formulas contain problem dependent constants that can be determined by solving a dual problem over the problem domain [6, 7, 8]. On the other hand, implicit a posteriori error estimation methods are based on solving the residual equation. The residual equation is localized over a single element, as in the element residual method (ERM) [9, 10] or over a patch of elements [4, 11, 12] to avoid solving a global equation.

Theoretical aspects of a posteriori error estimation for advection-diffusion equations can be found in [13, 14, 15]. Numerical studies of the accuracy of local error estimators for this class of problem can be found in [16, 17, 18]. The first study [16] uses different variations of implicit and explicit error estimators. A gradient based error indicator, explicit residual estimator (three variations), Galerkin element residual method (ERM), stabilized element residual method and the Zienkiewicz-Zhu (ZZ) patch recovery were tested [16]. It was concluded that none of the considered local error estimators worked satisfactorily in all test problems. A potential reason for this shortcoming can be attributed to a priori limit on the maximum mesh size (number of nodes), even though the test problems had very sharp features. This decision can result in an insufficient number of elements being available to resolve the sharp layers. In the second study [17], the stabilized ERM estimator, ZZ patch recovery and an explicit residual estimator were evaluated numerically. The results showed that the effectivity index of the estimated error never approaches unity with mesh refinement, except for one of the test problems that used the element residual

method. In the third study [18], an error estimator based on solving local Poisson problems with Neumann boundary conditions was investigated. Again, the reported numerical results showed that the effectivity indices of the estimated errors are far from unity on both uniformly and adaptively refined meshes.

The focus of this paper is not to present a new error estimator for advection-diffusion equations, instead we are interested in the numerical performance error estimation based on solving local Neumann problems. The stabilized element residual method for a posteriori error estimation is presented along with an error estimation technique attributed to Zienkiewicz and Zhu (ZZ)[19, 20]. In order to investigate the quality of the element residual error estimator, the estimated errors are compared to the exact errors. The comparison is carried out using three different error norms to study the effect of the norm used on the accuracy of the estimated errors. The estimated errors are also used to drive a mesh adaptivity iteration and the adapted meshes are evaluated qualitatively, in terms of conformity to the sharp layers in the solution and in terms of dependency on the wind direction. The numerical test problems will be limited to two dimensional problems discretized into first order quadrilateral finite elements. To our knowledge, no such investigation has been carried out using different error norms.

The outline of the paper is as follows: In section 2, the Galerkin finite element formulation for advection diffusion equations is presented. Stability issues of this formulation for advection dominant problems are highlighted and the details of the Streamline Upwind Petrov Galerkin (SUPG) stabilization technique are presented. In section 3, a detailed description of the stabilized element residual method for a posteriori error estimation is presented, along with the three different error norms used for computing the global error norm. In section 4, the details of the numerical test problems are presented, followed by the computational results in section 5. The concluding statements are drawn in section 6.

4.2 Linear advection diffusion equations

Advection dominated equations describe a wide variety of physical phenomena of special interest in engineering and applied physics. Fluid dynamics, contaminant transport and many other engineering problems rely on solving this type of equations. We consider the scalar linear advection diffusion problem over a bounded domain Ω in $\mathbb{R}^2$ with a Lipschitz-continuous boundary $\partial\Omega$ composed of a Dirichlet portion Γ_D and Neumann portion Γ_N, where $\partial\Omega = \Gamma_D \cup \Gamma_N$ and $\Gamma_D \cap \Gamma_N = \varnothing$. The PDE under consideration is written in an abstract format as:

$$\begin{cases} Lu &= f & \text{in} \quad \Omega \\ u &= u_D & \text{on} \quad \Gamma_D \\ \varepsilon \frac{\partial u}{\partial \mathbf{n}} &= g & \text{on} \quad \Gamma_N \end{cases} \tag{4.1}$$

where u denotes the unknown concentration, L is a linear second order differential operator of the form $Lu := -\varepsilon \triangle u + \mathbf{a} \cdot \nabla u$, $\varepsilon > 0$ is the diffusivity constant, $\mathbf{a} \in L^\infty(\Omega)^2$ is the velocity field and $f \in L_2(\Omega)$ is the prescribed load function. The boundary data u_D and g are assumed to be sufficiently smooth and $\mathbf{n}$ denotes the outward unit normal vector to Γ_N. Using standard norm notation for Sobolev and Hilbert spaces as in Reference [21] and equipped with scalar L^2 inner product $(.,.)$, the weak formulation of Equation (4.1) can be written as:

$$\begin{cases} \text{Find } u \in U \text{ such that} \\ \mathcal{B}(u,v) = (f,v) + \langle g, v \rangle_{\Gamma_N} \qquad \forall v \in V \end{cases} \tag{4.2}$$

where $U = \{u \in H^1(\Omega) : u|_{\Gamma_D} = u_D\}$ is the trial space, $V = \{v \in H^1(\Omega) : v|_{\Gamma_D} = 0\}$ is the test space. The bilinear $\mathcal{B}$ is defined as $\mathcal{B}(u,v) := \int_\Omega (\varepsilon \nabla u \cdot \nabla v + \mathbf{a} \cdot \nabla u \, v)$, $(f,v) := \int_\Omega f \, v$ is the L^2-inner product and $\langle g, v \rangle_{\Gamma_N} := \int_{\Gamma_N} g \, v$ is the boundary integral term.

4.2.1 Failure of the standard Galerkin method

In the Galerkin finite element method both the solution and the test functions are approximated using a finite dimensional subspace of V. Using the space $V_h \subset V$ defined by a mesh size parameter h and spanned by continuous piecewise polynomials, the Galerkin discrete weak formulation is:

$$\begin{cases} \text{Find } u_h \in V_h \text{ such that} \\ \mathcal{B}(u_h, v_h) = (f, v_h) + \langle g, v_h \rangle_{\Gamma_N} \qquad \forall v_h \ \in V_h \end{cases} \tag{4.3}$$

where u_h is the finite element solution. Existence of a solution for equation (4.3) with an estimated error bound is based on Lax-Milgram Lemma, where the bilinear $\mathcal{B}$ is required to be bounded and V-elliptic. Formally, there should exist two constants, $\alpha > 0$ and $0 < M < \infty$ such that:

$$\alpha \|v\|_V^2 \leq \mathcal{B}(v, v), \qquad \mathcal{B}(w, v) \leq M \|v\|_V \|w\|_V \tag{4.4}$$

If these two conditions are satisfied, the error bound can be written as

$$\|u - u_h\|_V \leq \frac{M}{\alpha} \inf_{v_h \in V_h} \|u - v_h\|_V \tag{4.5}$$

The method is successful if $M/\alpha \approx 1$, which is the case for elliptic equations like the Poisson equation. For advection dominated problems the ratio M/α is proportional to $\|\mathbf{a}\|_\infty/\varepsilon$, which makes the error in the right hand side of Equation (4.5) very large for small values of ε. The ratio of the velocity to the diffusivity is commonly related to the mesh size using the local Peclet number defined as $P_e = \|\mathbf{a}\|_{\infty,K} \ h_K/2 \ \varepsilon$, where h_K is the mesh diameter for an element K. If the value of P_e is greater than 1, oscillations appears in the numerical solution because the data advection is not captured by the mesh length scale.

4.2.2 Petrov-Galerkin methods

In order to eliminate the unphysical oscillations that might appear in the Galerkin finite element solution, a consistent stabilization technique is used. Artificial diffusion terms that vanish for the exact solution are added to the Galerkin weak formulation. Streamline Upwind Petrov Galerkin (SUPG) method is one of the commonly used stabilization techniques. SUPG falls in the class of Petrov-Galerkin methods, where the test and trial functions belongs to different spaces. The SUPG uses a modified test function defined by $w := \delta(\mathbf{a} \cdot \nabla v) + v$, where δ is a stabilization parameter. For first order approximations, the stabilization contributions are only defined inside the element interiors because the modified test function is discontinuous along the element edges. A general stabilized discrete variational formulation is

$$\begin{cases} \text{Find } u_h \in V_h \text{ such that} \\ \mathcal{B}(u_h, v_h) + \sum_K \int_K \delta_K \mathcal{R}(u_h)\mathcal{P}(v_h) = (f, v_h) + \langle g, v_h \rangle_{\Gamma_N} \qquad \forall v_h \in V_h \end{cases} \tag{4.6}$$

where δ_k is a locally defined stabilization parameter and $\mathcal{R}(u_h)$ is the residual of the strong PDE defined as $\mathcal{R}(u_h) := -\varepsilon \triangle u_h + \mathbf{a} \cdot \nabla u_h - f$. For the SUPG method, the term $\mathcal{P}(v_h)$ is defined as $(\mathbf{a} \cdot \nabla v_h)$. The Galerkin least squares (GLS) method follows the general Equation (4.6) with $\mathcal{P}(v_h)$ defined as the residual of the equation without the loading term, $\mathcal{P}(v_h) := -\varepsilon \triangle v_h + \mathbf{a} \cdot \nabla v_h$. For first order finite elements, both the SUPG and GLS yield the same formulation because the second derivative term $-\varepsilon \triangle v_h$ vanishes. A very important factor for the convergence of both the SUPG and GLS methods is the selection of the stabilization parameter δ_K. This parameter controls the right amount of artificial diffusion in the streamline direction. In the current study, a general definition of the stabilization parameter following [22] is used:

$$\delta_K := \frac{h_K\, \omega}{2\, \|\mathbf{a}\|_{\infty,K}}, \quad \text{where} \quad \omega := \begin{cases} \frac{h_K}{2}\left(1 + \frac{1}{P_e}\right) & P_e > 1 \\ 0 & P_e \leq 1 \end{cases} \tag{4.7}$$

4.3 A posteriori error estimation

The error in the finite element solution u_h is defined as $e = u - u_h$, where u is the exact solution. Substituting the value of $u = u_h + e$ in Equation (4.2) results the global residual equation

$$\mathcal{B}(e, v) = (f, v) + \langle g, v \rangle_{\Gamma_N} - \mathcal{B}(u_h, v) \tag{4.8}$$

In the previous equation, if v is selected such that $v \in V_h$, the residual will vanish. This shows that the error field e can be evaluated by solving the global residual equation, but the approximation space has to be larger than V_h used for the finite element solution. This leads to an extrapolation method where the problem is solved again using a globally refined mesh or using a higher order finite elements.

The main goal of practical a posteriori error estimator is to approximate the solution of the global residual Equation (4.8) in a numerically efficient way [1, 3]. Localizing this equation over an element or a point based patch of elements results in a set of decoupled problems, which are much cheaper to evaluate. In this case, the global error is evaluated as the sum of the local contributions. In the following subsection, the details of SUPG stabilized ERM [10] for a posteriori error estimation are presented.

4.3.1 Stabilized element residual method

In this paper, the focus will be on the Element Residual Method (ERM) where a set of local problems with Neumann boundary conditions are solved. Substituting $u = u_h + e$ in the original problem (strong form) and localizing it over each element K, one gets:

$$-\varepsilon \triangle e + \mathbf{a} \cdot \nabla e = f + \varepsilon \triangle u_h - \mathbf{a} \cdot \nabla u_h \quad \text{over each mesh element } K \tag{4.9}$$

The boundary conditions of Equation (4.9) have to be carefully specified. For the element edges intersecting with the problem Neumann boundary the boundary condition is specified as

$$\varepsilon \frac{\partial e}{\partial \mathbf{n}} = g - \varepsilon \frac{\partial u_h}{\partial \mathbf{n}} \tag{4.10}$$

The remaining boundaries of the element are expressed as Neumann boundaries that depend on the exact solution u. For each (non-boundary) edge i of the element K, the boundary condition is:

$$\varepsilon \frac{\partial e}{\partial \mathbf{n}_{Ki}} = \varepsilon \frac{\partial u}{\partial \mathbf{n}_{Ki}} - \varepsilon \frac{\partial u_h}{\partial \mathbf{n}_{Ki}} \tag{4.11}$$

where $\mathbf{n}_{Ki}$ is the outward unit normal vector to the edge i of the element K. As the exact solution is not known, it is postulated that the average flux over the element edges is a good approximation of the exact flux. This is expressed as follows:

$$\frac{\partial u}{\partial \mathbf{n}_{Ki}} \approx \frac{1}{2} \mathbf{n}_{Ki} \cdot \{(\nabla u_h)_K + (\nabla u_h)_{K'}\} \text{ on } \partial K \cap \partial K' \tag{4.12}$$

where the elements K and K' share the (non-boundary) edge i. Equation (4.9) and the corresponding boundary conditions can be solved using the finite element method to calculate an approximation of the error over each element denoted as $\widetilde{e}_b$. The ERM weak formulation is:

$$\begin{cases} \text{Find } \widetilde{e}_b \in V_b \text{ such that} \\ \mathcal{B}_K(\widetilde{e}_b, v_b) = (f, v_b)_K - \mathcal{B}_K(u_h, v_b) + \int\limits_{\partial K} \varepsilon \left\langle \frac{\partial u_h}{\partial \mathbf{n}_{Ki}} \right\rangle v_b \qquad \forall v_b \in V_b \end{cases} \tag{4.13}$$

where $\mathcal{B}_K$ is a restriction of the bilinear over an element K, $(.,.)_K$ is a restriction of the L^2 norm over the element K and V_b is the approximation space for the local

problem. The Neumann boundary term in Equation (4.13) is specified as:

$$\varepsilon \left\langle \frac{\partial u_h}{\partial \mathbf{n}_{Ki}} \right\rangle = \begin{cases} \frac{1}{2}\, \varepsilon\, \mathbf{n}_{Ki} \,.\, \{(\nabla u_h)_K + (\nabla u_h)_{K'}\} & \text{on } \partial K \cap \partial K' = \text{ Edge } i \not\subseteq \partial\Omega \\ \varepsilon\, \mathbf{n}_{Ki} \,.\, (\nabla u_h)_K & \text{on } \partial K \cap \Gamma_D \\ g & \text{on } \partial K \cap \Gamma_N \end{cases} \tag{4.14}$$

Second order bubble functions are used as the basis functions of the approximation space V_b. Bubble functions are functions with compact support, where each bubble has a value in the interior of an entity but vanishes on the exterior of the entity [3]. For a reference quadrilateral element $\widehat{K}$, with the local coordinates $(-1 \leq \hat{x} \leq 1, -1 \leq \hat{y} \leq 1)$, the edge bubble functions are given by $\chi_1 = 0.5(1-\widehat{x}2)(1-\widehat{y})$; $\chi_2 = 0.5(1+\widehat{x})(1-\widehat{y}2)$; $\chi_3 = 0.5(1-\widehat{x}2)(1+\widehat{y})$; $\chi_4 = 0.5(1-\widehat{x})(1-\widehat{y}2)$; and the element interior bubble function is given by $\chi_5 = (1-\widehat{x}2)(1-\widehat{y}2)$.

The solution of the local problems is stabilized using the SUPG method. The formulation of the stabilized element residual method starts from Equation (4.13), by adding the SUPG consistent stabilization term

$$\mathcal{B}_K(\widetilde{e}_b, v_b) + \int_K \delta_K \mathcal{P}(v_b)\mathcal{R}_h(\widetilde{e}_b) = (f, v_b)_K - \mathcal{B}_K(u_h, v_b) + \int_{\partial K} \varepsilon \left\langle \frac{\partial u_h}{\partial \mathbf{n}_{Ki}} \right\rangle v_b \qquad \forall v_b \in V_b \tag{4.15}$$

where $\mathcal{R}_h(\widetilde{e}_b)$ is the residual of Equation (4.9) defined as $\mathcal{R}_h(\widetilde{e}_b) = -\varepsilon \triangle \widetilde{e}_b + \mathbf{a} \cdot \nabla \widetilde{e}_b - f - \varepsilon \triangle u_h + \mathbf{a} \cdot \nabla u_h$ and $\mathcal{P}(v_b) = (\mathbf{a} \cdot \nabla v_b)$. After some rearrangement and the elimination of the second derivative terms, a simplified stabilized ERM formulation as in [16] is obtained:

$$\begin{aligned} \mathcal{B}_K(\widetilde{e}_b, v_b) \;+\; & \int_k \delta_K (\mathbf{a} \cdot \nabla \widetilde{e}_b)\, (\mathbf{a} \cdot \nabla v_b) = (f, v_b)_K + \int_K \delta_K f\, (\mathbf{a} \cdot \nabla v_b) - \mathcal{B}_K(u_h, v_b) \\ - \; & \int_K \delta_K (\mathbf{a} \cdot \nabla u_h)\, (\mathbf{a} \cdot \nabla v_b) + \int_{\partial K} \varepsilon \left\langle \frac{\partial u_h}{\partial \mathbf{n}_{Ki}} \right\rangle v_b \qquad \forall v_b \in V_b \end{aligned} \tag{4.16}$$

Establishing a numerical solution for Equation (4.16) is not guaranteed unless the

edge fluxes are under equilibrium. Ainsworth and Oden [10] proposed a method to equilibrate the edge fluxes while maintaining the consistency condition of the flux across the element boundaries. This flux equilibration method is difficult to implement and is rarely applied in practice [16, 23].

4.3.2 The choice of an error norm

The norm used for measuring errors depends on the problem at hand. While a standard energy norm exists for the elliptic operator, such a standard norm does not exist for the advection diffusion equation. The argument presented by Kunert in [24] is adopted, where different norms reflect different modelling background and research purposes. The available literature [16, 17, 18, 25] on the effectivity of local error estimation techniques was based on using a single norm in each study. In this study, three different error norms are investigated. The first is the weighted energy norm defined as:

$$\|u\|_{en} = \left(\varepsilon\|\nabla u\|^2_{L_2(\Omega)} + \|u\|^2_{L_2(\Omega)}\right)^{1/2} \tag{4.17}$$

This error norm is a special case of the error norm used for reaction diffusion equations with the reaction coefficient set to one. This norm is referred to as the energy norm in the rest of this paper. This error norm was the norm of choice in previous numerical studies [16, 17]. The second error norm considered is the stability norm introduced by Brezzi, *et al.* in [26]. This norm is defined as:

$$\|u\|_{sn} = \left(\varepsilon\|\nabla u\|^2_{L_2(\Omega)} + \sum_K h_K \ \|\mathbf{a}\cdot\nabla u\|^2_{L_2(T)}\right)^{1/2} \tag{4.18}$$

where h_K is the diameter of an element K. This mesh-dependent norm arises in the analysis of the streamline diffusion finite element method, and the analysis of the residual-free bubble method [26]. The third error norm employed is a special case of the natural norm for coercive and non-symmetric operators introduced by Sangalli in

[25], with the reaction term set to zero

$$\|u\|_{fn} = \left(\varepsilon\|u\|^2_{H^1_0(\Omega)} + |\mathbf{a}|\|u\|^2_{1/2}\right)^{1/2} \approx \left(\varepsilon\|u\|^2_{H^1_0(\Omega)} + |\mathbf{a}|\|u\|^2_{L_2(\Omega)}\right)^{1/2} \tag{4.19}$$

The fractional order semi-norm is approximated by an L_2 norm. This norm will be referred to in the rest of this paper as an approximate fractional order norm.

In addition to error estimation by ERM, computational results of an error estimator based on patch recovery techniques attributed to Zienkiewicz and Zhu are included. The ZZ estimator is based on gradient recovery by solving local least square problems over discrete patches [19, 20]. The ZZ error estimator is relatively easy to implement and is independent of the problem formulation. The error calculated by the ZZ patch recovery is measured in the following unweighted energy norm,

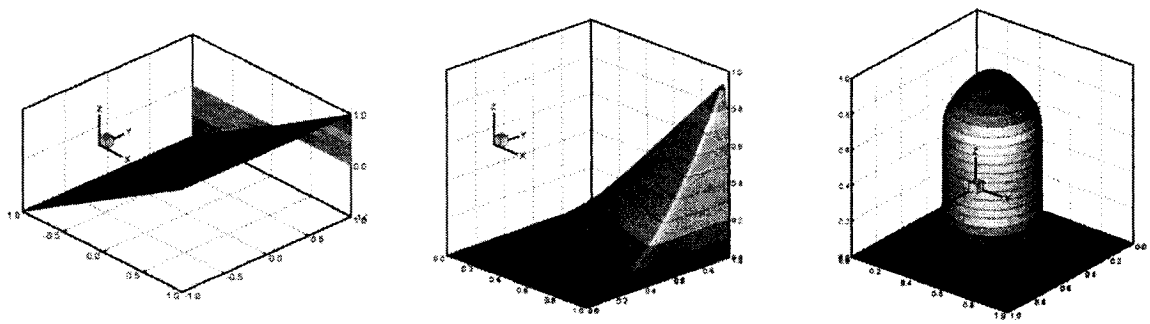
$$\|u\|_{unw} = \left(\|\nabla u\|^2_{L_2(\Omega)}\right)^{1/2} \tag{4.20}$$

4.4 Numerical test problems

Three test problems with analytical solutions are used to evaluate the effectivity indices of the estimated errors and to investigate the effect of using different error norms on mesh adaptivity. The exact solution of these problems exhibit sharp layers on the interior or the boundary of the problem domain. The first problem is adopted from [18], while the second and third problems are similar to problems 5.2 and 5.3 in Reference [16], respectively.

The first problem AD1 is governed by Equation (4.1) and defined over the domain $\Omega = (-1,1) \times (-1,1)$. The problem is assumed to have a constant velocity vector $\mathbf{a} = (0,1)$ and a zero loading function f. The essential boundary condition $\Gamma_D = \partial\Omega$ satisfies the following exact solution:

$$u = x\left(1 - exp\left(\frac{y-1}{\varepsilon}\right)\right)/\left(1 - exp\left(-\frac{2}{\varepsilon}\right)\right) \tag{4.21}$$



(a) AD1 ($\varepsilon = 10^{-4}$) (b) AD2 ($\varepsilon = 10^{-2}$) (c) AD3 ($\varepsilon = 10^{-6}$)

Figure 4.1: Exact solution plot for problems AD1, AD2 and AD3

The second problem AD2 is defined over the domain $\Omega = (0,1)\times(0,1)$ and governed by Equation (4.1) with a constant velocity vector $\mathbf{a} = (2,3)$. The essential boundary condition $\Gamma_D = \partial\Omega$ and the loading function f are chosen such that the exact solution is:

$$u = xy^2 - y^2\, exp\left(\frac{2x-2}{\varepsilon}\right) - x\, exp\left(\frac{3y-3}{\varepsilon}\right) + exp\left(\frac{2x-5+3y}{\varepsilon}\right) \tag{4.22}$$

The third problem AD3 is similar to AD2, except the loading function f and the essential boundary condition are chosen such that the exact solution is:

$$u = 16\,x\,(1-x)\,y\,(1-y)\left(1/2 + \frac{\arctan\left[2\sqrt{\varepsilon^{-1}}\left(1/16 - (x-1/2)^2 - (y-1/2)^2\right)\right]}{\pi}\right) \tag{4.23}$$

Three dimensional plots of the exact solution for problems AD1, AD2 and AD3 are shown in Figure 4.1. The solution for AD1 exhibits a sharp boundary layer at the boundary $y = 1$, while the solution for AD2 exhibits two sharp boundary layers at $y = 1$ or $x = 1$. For problem AD3, the exact solution exhibits a sharp circular internal layer.

Two additional problems, without analytical solutions, adopted from [17] will be used to investigate the effect of using different error norms on the mesh adaptivity

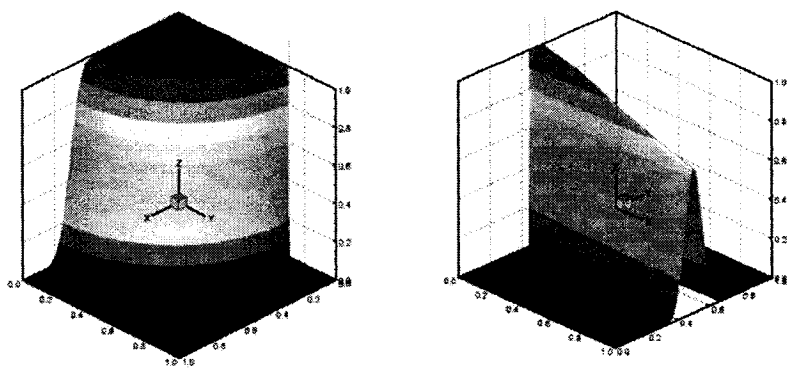
process. Problem AD4 is defined over the square domain $\Omega = (0,1) \times (0,1)$, with a spatially varying velocity vector $\mathbf{a} = (y, -x)$ and zero loading function. Dirichlet boundary condition is prescribed on the inflow part of the boundary defined by $x = 0$ or $y = 1$ such that:

$$u = 1 \text{ for } x = 0 \text{ and } y \leq 0.7 \text{ and } u = 0 \text{ otherwise} \tag{4.24}$$

On the outflow boundary, homogeneous Neumann boundary conditions are prescribed. Problem AD5 is similar to problem AD4 except that the velocity vector is set to $\mathbf{a} = (1, 0)$ and the inflow part of the Dirichlet BC is defined as:

$$u = 1 \text{ for } x = 0 \text{ and } |y - 0.5| \leq 0.05 \text{ and } u = 0 \text{ otherwise} \tag{4.25}$$

Similar to problem AD4, homogeneous Neumann boundary conditions are prescribed at the outflow part of the boundary. A three dimensional plots of the solution for problem AD4 and AD5 using a very fine mesh are shown in Figure 4.2.



(a) AD3 ($\varepsilon = 10^{-3}$) (b) AD4 ($\varepsilon = 10^{-3}$)

Figure 4.2: Exact solution plot for problems AD4 and AD5

The quality of the estimates are measured in terms of the effectivity index. The effectivity index (EI) is the ratio of the estimated error to the exact error, both measured in the same norm. Each test problem has a parameter ε that controls the

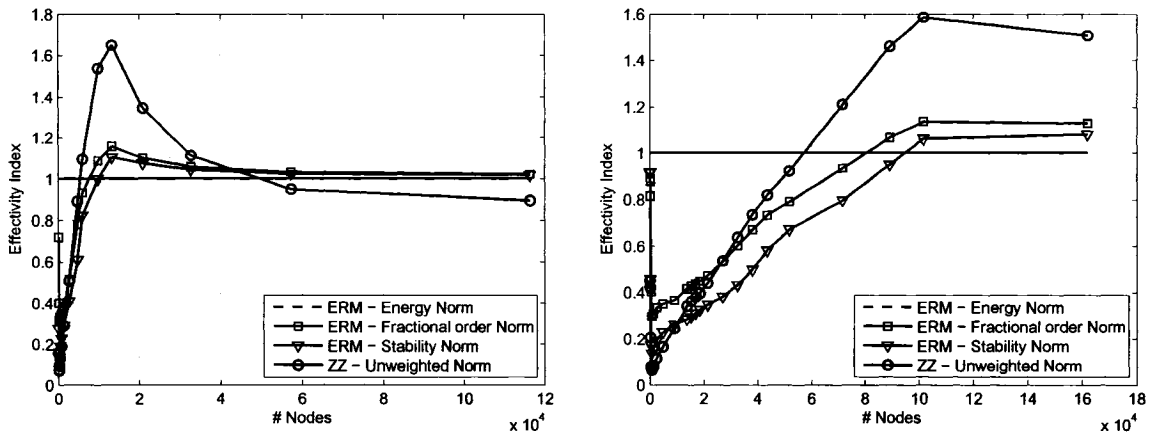
width of the sharp layer. Smaller values of ε imply more advection in the problem and thus more difficulty in both solving the problem and in estimating the errors. The solution scheme with adaptive mesh refinement was implemented within the open-source libMesh finite element library [27].

4.5 Computational results

A posteriori error estimation results are used to guide the mesh adaptivity process. The objective of this adaptation is to equidistribute the errors over the entire problem domain. This objective is achieved by refining, in each adaptivity step, a set of elements that have error values exceeding a pre-specified fraction of the maximum elemental error.

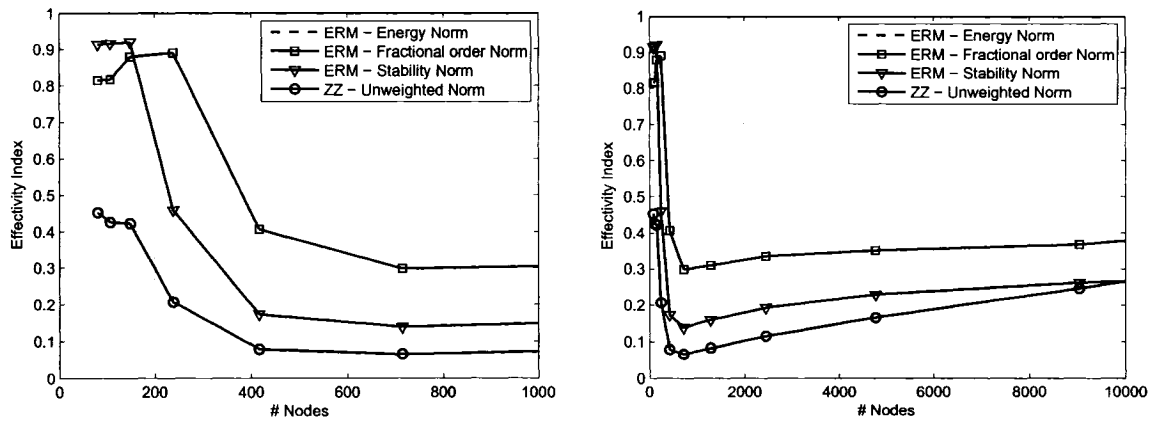
Problem AD1: Figure 4.3 shows the effectivity indices of the estimated errors using different error norms. Two values for the parameter ε are used. In the first case with $\varepsilon = 10^{-3}$, the effectivity index shows an asymptotic behavior that approaches unity as the mesh is refined. The fractional order norm and the energy norm (weighted) produced almost identical effectivity indices and their corresponding curves overlap in Figures 4.3 and 4.4. This is attributed the insignificant differences between the two norms for the case of a unit velocity vector. For coarse meshes, the ERM and ZZ error estimators, regardless of the norm used, tend to underestimate the error. This is attributed to the fact that the solution was not resolved by these meshes. While the SUPG method produces a stable solution, the accuracy of the solution close to the boundary layer is lost. With mesh refinement, the ERM outperforms the ZZ in the accuracy of the estimated errors. In Figure 4.3-(b), the effectivity index of the estimated error is plotted against the mesh size for the case of $\varepsilon = 10^{-4}$. The error estimator is only accurate for mesh sizes larger than 10^5 because this mesh resolves the solution at the boundary layer. The stability norm performs slightly better in estimating the errors on the adapted meshes.

To gain a better understanding of the performance of the error estimator, the



(a) $\varepsilon = 10^{-3}$ (b) $\varepsilon = 10^{-4}$

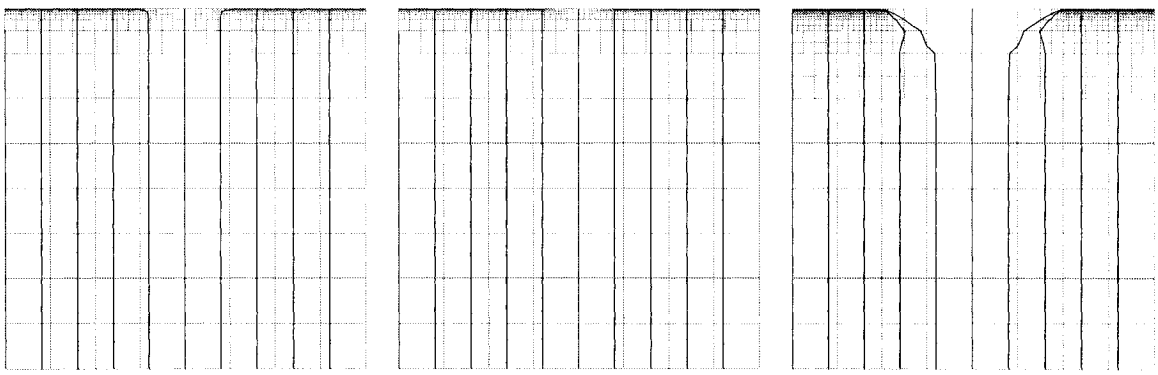
Figure 4.3: Effectivity indices for problem AD1 (Adapted by the energy norm of the exact error)



(a) Up to 10^3 Nodes (b) Up to 10^4 Nodes

Figure 4.4: Zoom in view of the effectivity indices plot for problem AD1 with $\varepsilon = 10^{-4}$

results plotted in Figure 4.3-(b) are plotted again using a different scale in Figure 4.4. In part a, the mesh is very coarse and the boundary layer is localized over a few mesh elements. Thus the estimated errors in the global norm are relatively accurate, but this scenario is not guaranteed to occur in every problem. With mesh refinement, as shown in Figure 4.4-(b), the effectivity index is deteriorating because the solution



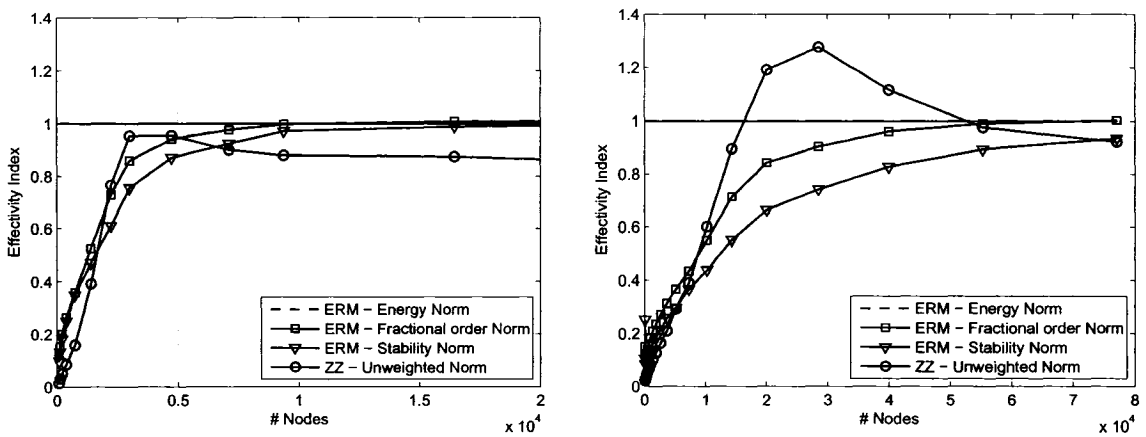
(a) ERM - Energy norm (b) ERM - Stability norm (c) ZZ - Adaptivity

Figure 4.5: Adapted meshes for Problem AD1 after 10 adaptivity iterations ($\varepsilon = 10^{-4}$)

close to the boundary layer is not properly resolved due to the damping effect of the stabilization parameter. The quality of the estimated errors improves with further mesh refinement and the effectivity index approaches the optimal unit value as shown in Figure 4.3-(b). It can be seen that for sharp boundary layers, the estimated errors are only accurate when the mesh resolves these layers. This is to be expected as the ERM uses second order bubble functions to approximate the error, while the exact solution exhibits exponential boundary layers. To obtain accurate error estimation for smaller values of ε much larger mesh sizes with smaller nodal spacing are needed. The previous analysis explains the results of example 2 in Reference [17], where a reduction in the effectivity index with mesh refinement was observed for the case of $\varepsilon = 10^{-10}$. The adapted meshes for problem AD1 after 10 adaptivity iterations ($\varepsilon = 10^{-4}$) are shown in Figure 4.5. Adapted meshes using ERM measured in the stability norm are sharper in the middle of the boundary layer, where the solution value is small. On the other hand, the mesh obtained by ZZ adaptivity showed some dependency on the magnitude of the solution gradient.

Problem AD2: The effectivity indices of the estimated errors are shown in Figure 4.6. Initially, for coarse meshes, the ERM underestimated the errors but with further refinement the effectivity index approached unity. The different norms used

to measure the errors did not produce significantly different results in the effectivity index. It is worth mentioning that the energy norm produced almost the same results are the fractional order norm. The adapted meshes are plotted in Figure 4.7 after



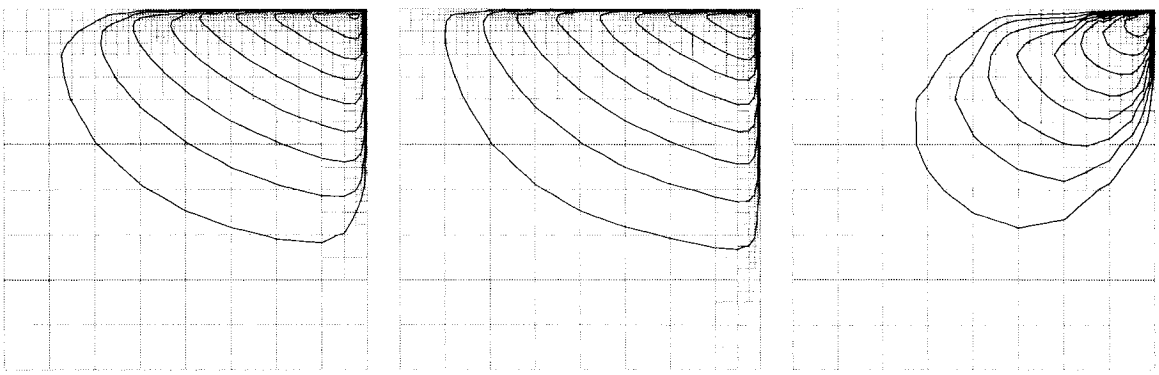
(a) $\varepsilon = 10^{-2}$ (b) $\varepsilon = 10^{-3}$

Figure 4.6: Effectivity indices for problem AD2 (Adapted by ERM measured in the stability norm)

8 adaptivity iterations. Again, the ERM measured in the stability norm produced meshes with an overall better distribution of mesh density that conforms with the solution boundary layers. The ZZ estimator, as for problem AD1, produced a refined mesh that depends on the magnitude of the solution gradient and subsequently resulted in a fine mesh where the two boundary layers intersects.

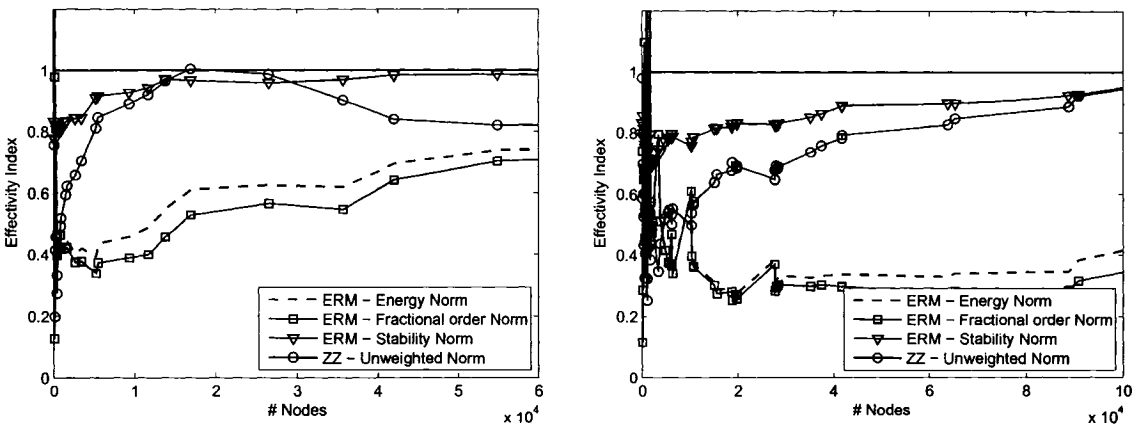
Problem AD3: For this problem, the effectivity indices of the estimated errors are plotted in Figure 4.8. The effectivity index for the stability norm was significantly closer to the optimal unit value for both ε values of 10^{-6} and 10^{-8}. Figure 4.9 shows the adapted meshes using the different adaptivity drivers. The mesh adapted using the ERM measured in the stability norm was symmetric and showed no affect of the wind direction. In contrast to that, the mesh adapted by the ZZ patch recovery method did not reflect the symmetry of the exact solution.

Problem AD4: The adapted meshes are shown in Figure 4.10 and Figure 4.11



(a) ERM - Energy norm (b) ERM - Stability norm (c) ZZ - Adaptivity

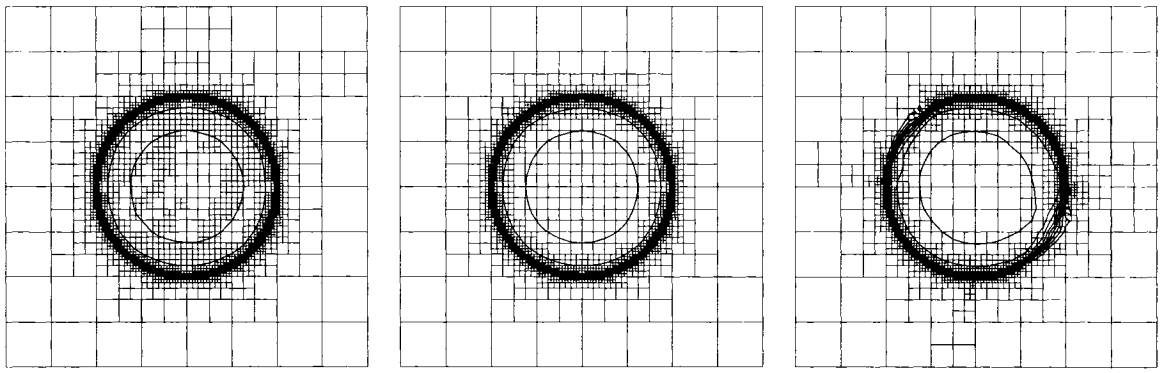
Figure 4.7: Adapted meshes for Problem AD2 after 8 adaptivity iterations ($\varepsilon = 10^{-3}$)



(a) $\varepsilon = 10^{-6}$ (b) $\varepsilon = 10^{-8}$

Figure 4.8: Effectivity indices for problem AD3 (Adapted by ERM measured in the stability norm)

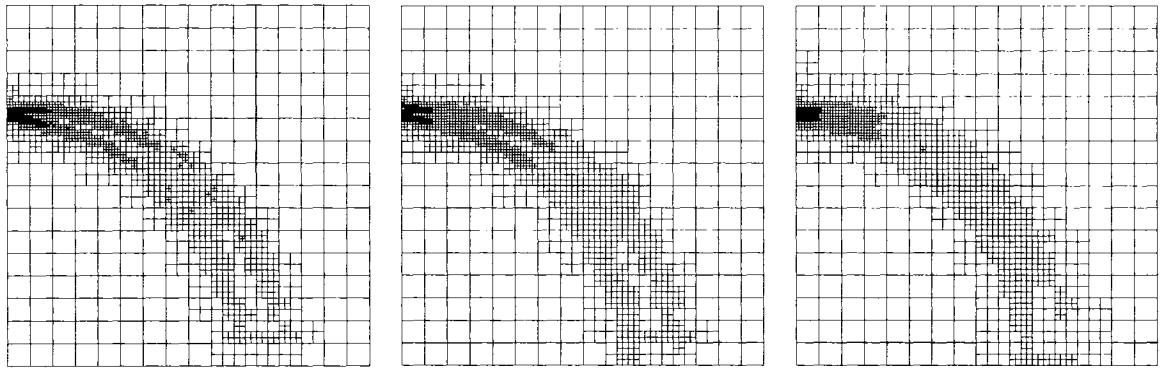
corresponding to $\varepsilon = 10^{-3}$ and $\varepsilon = 10^{-5}$, respectively. Both the meshes adapted by the ERM measured in the energy norm and the stability norm captured the singularity at the inflow boundary for $\varepsilon = 10^{-3}$ as well as around the internal layer for the case of $\varepsilon = 10^{-5}$. Meshes adapted according to the ERM managed to eliminate the oscillation around the internal layer, even without using a shock capturing scheme. However, the mesh adapted by ZZ failed to capture this higher order phenomena at the shock



(a) ERM - Energy norm (b) ERM - Stability norm (c) ZZ - Adaptivity

Figure 4.9: Adapted meshes for Problem AD3 after 18 adaptivity iterations ($\varepsilon = 10^{-6}$)

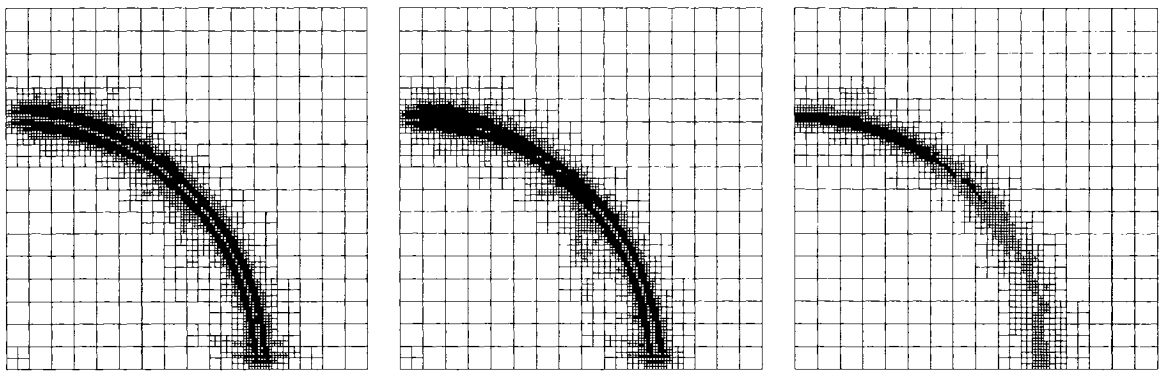
edges.



(a) ERM - Energy norm (b) ERM - Stability norm (c) ZZ - Adaptivity

Figure 4.10: Adapted meshes for Problem AD4 after 7 adaptivity iterations ($\varepsilon = 10^{-3}$)

Problem AD5: The solution for this problem exhibits two discontinuities at the inflow boundary which are reflected in the adapted meshes shown in Figure 4.12. Meshes adapted by ERM measured in the stability norm are slightly better than those produced by ZZ or ERM measured in the energy norm. The magnitude of the calculated errors close to the points of discontinuities are much larger than in the rest of the domain, which resulted in marking few elements for refinement at each



(a) ERM - Energy norm (b) ERM - Stability norm (c) ZZ - Adaptivity

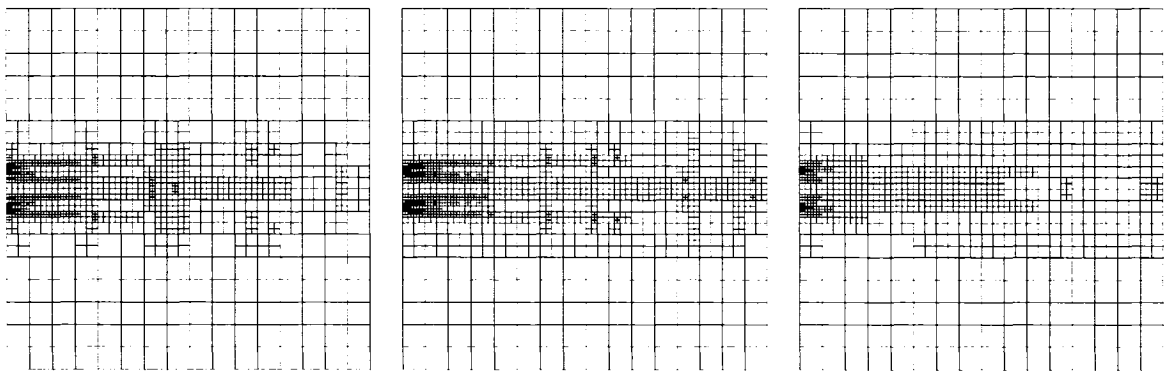
Figure 4.11: Adapted meshes for Problem AD4 after 10 adaptivity iterations ($\varepsilon = 10^{-5}$)

adaptivity iteration. This resulted in an increased number of adaptivity iterations to attain a certain level of error in the global norm.

Another marking scheme referred to as the sorting strategy is used for this problem. The sorting strategy is based on sorting the mesh elements according to the estimated local errors and refining a certain percentage of the mesh elements with the largest error norm. This strategy resulted in fewer mesh adaptivity iterations. Figure 4.13 shows the adapted meshes using the sorting strategy, where 10% of the mesh elements are marked for refinement at each adaptivity iteration. The adapted meshes after 10 adaptivity iterations have a relatively similar distribution inside the problem domain. Using a more aggressive refinement scheme, especially when the starting mesh is very coarse might result in a mesh that is affected by the wind direction as for example 6.2 in Reference [16].

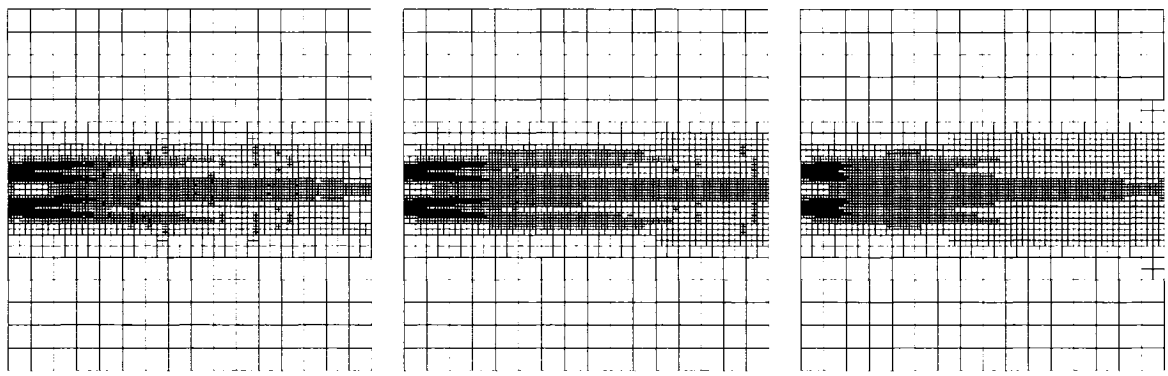
4.6 Conclusions

In this work, error estimation for advection diffusion equations was numerically investigated. Three different error norms were employed to measure the error norm. The errors were estimated by solving local element based problems subjected to Neumann boundary condition and using ZZ patch recovery. The presented results are



(a) ERM - Energy norm (b) ERM - Stability norm (c) ZZ - Adaptivity

Figure 4.12: Adapted meshes for Problem AD5 after 15 adaptivity iterations ($\varepsilon = 10^{-3}$)



(a) ERM - Energy norm (b) ERM - Stability norm (c) ZZ - Adaptivity

Figure 4.13: Adapted meshes for Problem AD5 after 10 adaptivity iterations using the sorting strategy ($\varepsilon = 10^{-3}$)

generally better than what have been reported in the literature with respect to the effectivity indices, because of the use of more adaptivity iterations with a mesh size depending on the local feature of the problem solution. It was shown that effectivity indices close to unity will only be attained if the sharp layers are completely resolved by the mesh. While this is a great limitation of the error estimation techniques based on solving local problems, these error estimators managed to guide the adaptivity process to obtain meshes that were able to resolve the sharp layers.

The use of the stability norm instead of the standard weighted energy norm has

shown some advantages numerically. For problem AD1 and AD2, the adapted meshes using the stability norm were more conforming to the sharp layers in the solution. In the case of internal layers, measuring the errors in stability norm resulted in effectivity indices closer to the optimal unit value. Adapted meshes guided by ERM measured in the stability norm were less dependent on the wind direction. A refinement criteria based on the sorting strategy where a constant percentage of the mesh elements are refined at each iteration was tested for problem AD5. This strategy is recommended for problems with local discontinuities or points of singularity in order to reduce the number of mesh adaptivity iterations.

ACKNOWLEDGEMENTS

This research was partially funded through grants from the Natural Science and Engineering Research Council of Canada (NSERC) and the McMaster University's Centre for Effective Design of Structures. The authors acknowledge the use of SHARCNET's computing facilities.

Bibliography

[1] Verfürth R. *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques.* Wiley-Teubner: Advances in Numerical Mathematics, 1996.

[2] Babuška I, Strouboulis T. *The Finite Element Method and Its Reliability.* The Clarendon Press Oxford University Press: Numerical Mathematics and Scientific Computation, 2001.

[3] Ainsworth M, Oden JT. *A Posteriori Error Estimation in Finite Element Analysis.* Wiley-Interscience [John Wiley & Sons]: Pure and Applied Mathematics, 2000.

[4] Babuška I, Rheinboldt WC. A-posteriori error estimates for the finite element method. *International Journal for Numerical Methods in Engineering* 1978; **12**:1597–1615.

[5] Eriksson K, Johnson C. An adaptive finite element method for linear elliptic problems. *Mathematics of Computation* 1988; **50**(182):361–383.

[6] Becker R, Rannacher R. An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica* 2001; **10**:1–102.

[7] Becker R, Rannacher R. A feed-back approach to error control in finite element methods: Basic analysis and examples. *East-West Journal of Numerical Mathematics* 1996; **4**(4):237–264.

[8] Eriksson K, Estep D, Hansbo P, Johnson C. Introduction to adaptive methods for differential equations. *Acta Numerica* 1995; **105**:105–158.

[9] Bank RE, Weiser A. Some a posteriori error estimators for elliptic partial differential equations. *Mathematics of Computation* 1985; **44**(170):283-301.

[10] Ainsworth M, Oden JT. A unified approach to a posteriori error estimation using element residual methods. *Numerische Mathematik* 1993; **65**:23–50.

[11] Morin P, Nochetto RH, Siebert KG. Local problems on stars: a posteriori error estimators, convergence, and performance. *Mathematics of Computation* 2003; **72**(243):1067–1097.

[12] Prudhomme S, Nobile F, Chamoin L, Oden JT. Analysis of a Subdomain-based Error Estimator for Finite Element Approximations of Elliptic Problems. *Numerical Methods for Partial Differential Equations* 2004; **20**(2):165–192.

[13] Agarwal AN, Pinsky PM. Stabilized element residual method (SERM): a posteriori error estimation for the advection-diffusion equation. *Journal of Computational and Applied Mathematics* 1996; **74**(1):3–17.

[14] Verfürth R. A posteriori error estimators for convection-diffusion equations. *Numerische Mathematik* 1998; **80**(4):641–663.

[15] Verfürth R. Robust a posteriori error estimates for stationary convection-diffusion equations. *SIAM Journal on Numerical Analysis* 2005; **43**(4):1766–1782.

[16] John V. A numerical study of a posteriori error estimators for convection-diffusion equations. *Computer Methods in Applied Mechanics and Engineering* 2000; **190**(5-7):757–781.

[17] Papastavrou A, Verfürth R. A posteriori error estimators for stationary convection-diffusion problems: a computational comparison. *Computer Methods in Applied Mechanics and Engineering* 2000; **189**(2):449–462.

[18] Kay D, Silvester D. The reliability of local error estimators for convection-diffusion equations. *IMA Journal of Numerical Analysis* 2001; **21**(1):107–122.

[19] Zienkiewicz OC, Zhu, JZ. The superconvergent patch recovery and a posteriori error estimates. I. The recovery technique. *International Journal for Numerical Methods in Engineering* 1992; **33**(7):1331–1364.

[20] Zienkiewicz OC, Zhu, JZ. The superconvergent patch recovery and a posteriori error estimates. II. Error estimates and adaptivity. *International Journal for Numerical Methods in Engineering* 1992; **33**(7):1365–1382.

[21] Brenner SC, and Ridgway SL. *The Mathematical Theory of Finite Element Methods.* Springer-Verlag: Texts in Applied Mathematics, 1994.

[22] Fischer B, Ramage A, Silvester DJ, Wathen AJ. On parameter choice and iterative convergence for stabilised discretisations of advection-diffusion problems. *Computer Methods in Applied Mechanics and Engineering* 1999; **179**(1-2):179–195.

[23] Prudhomme S, Oden JT, Westermann T, Bass J, Botkin ME. Practical methods for a posteriori error estimation in engineering applications. *International Journal for Numerical Methods in Engineering* 2003; **56**(8):1193–1224.

[24] Kunert G. A note on the energy norm for a singularly perturbed model problem. *Computing* 2002; **69**(3):265–272.

[25] Sangalli G. On robust a posteriori estimators for the advection-diffusion-reaction problem. *Technical Report, ICES Report 04-55, The Institute for Computational Engineering and Sciences, The University of Texas at Austin* 2004;

http://www.ticam.utexas.edu/research/reports/2004/0455.pdf [6 September 2006].

[26] Brezzi F, Hughes TJR, Marini LD, Russo A, Süli E. A priori error analysis of residual-free bubbles for advection-diffusion problems. *SIAM Journal on Numerical Analysis* 1999; **36**(6):1933–1948.

[27] LibMesh: A C++ framework for the numerical simulation of partial differential equations on serial and parallel platforms.
http://libmesh.sourceforge.net

Chapter 5

A posteriori error estimation based on numerical realization of the variational multiscale method

A.H. ElSheikh, S.E. Chidiac and S. Smith

ABSTRACT

This paper presents a numerical realization of the variational multiscale method with the objective of providing a reliable and easy to implement local error estimation technique. The variational multiscale framework provides a systematic approach of solution scale decomposition into coarse scales captured by the mesh and fine or subgrid scales. In the proposed work, the coarse scale errors in the finite element solution are neglected in comparison to the fine scale errors. The fine scale variational equation is then localized using a general localization function over an element, or a patch of elements, to develop local error estimation technique. Based on the proposed framework, a consistent formulation of a new subdomain error estimator is derived, without the necessity of introducing an error locality assumption. The new subdomain error estimator is evaluated numerically within a mesh adaptivity algorithm and it is shown to produce very sharp error estimates that outperform the element residual method estimates.

KEY WORDS: Adaptive Finite Element; A Posteriori Error Estimates; Poisson Equation, Variational Multiscale Method

5.1 Introduction

Mathematical modelling of many physical phenomena arising in engineering applications leads to partial differential equations (PDE's). These PDEs are often numerically solved using the finite element method because of its modularity, ease of implementation and its strong theoretical foundation. Given the value of the finite element method, an understanding of potential sources of errors in the solution is critical. One of the major sources of error in the finite element solution is the discretization error. Fortunately, adaptive mesh refinement based on a posteriori error estimates offers an effective method to estimate and control the discretization error.

Reliable error estimation for the finite element method has been the focus of intense research during the past decade [1, 2, 3]. A posteriori error estimation techniques can generally be divided into implicit and explicit methods. Explicit methods use approximate formulas to estimate the residual in the element interior and on the element boundaries [4, 5]. These formulas contain problem dependent constants that can be determined by solving a dual problem over the problem domain [6, 7, 8]. On the other hand, implicit a posteriori error estimation methods are based on approximating the solution of the residual equation. The residual equation is localized over a single element, as in the element residual method (ERM) [9, 10], or over a patch of elements, as in the subdomain residual methods (SRM) [4, 11, 12]. The localization is introduced to avoid having to solve a global equation, as in extrapolation methods.

In this paper, a general framework for error estimation based on a numerical realization of the Variational MultiScale (VMS) method is proposed. In the VMS framework [13], the solution space is decomposed into resolved components (captured by the mesh) and unresolved components (subgrid scale). This decomposition provides a simple way to derive a variational formulation for both resolved and unresolved scales. Different localization techniques can be applied to the fine scale variational formulation to derive different local error estimators. The proposed framework is used to derive a new flux-free subdomain residual method for patch based error es-

timation. This new flux-free estimator has some similarity to the subdomain error estimator presented in [11], but the new estimator is conceptually different and easier to implement. In comparison to existing SRM formulations, no assumptions are introduced regarding the error norm for measuring the errors, as in the SRM formulation presented in [12], or regarding the boundary conditions of the local problems, as in the SRM formulation presented in [4]. The localization effect on the calculation of the unresolved scales is observed to be minimal, as demonstrated in the numerical examples.

This paper starts with a simple problem formulation followed by a brief review of the VMS method. A general framework for local error estimation based on the variational multiscale decomposition is introduced. In addition, some theoretical properties of the estimated errors using the proposed framework are presented. The element residual method is then derived from the general framework. A new flux-free error estimator that is sharp and provides a quasi-upper bound of the exact error in the energy norm is then introduced. The last section demonstrates the reliability of the proposed SRM through numerical examples.

5.2 Problem setup

A linear elliptic problem defined by

$$\begin{cases} Lu = f & \text{in } \Omega \\ u = u_D & \text{on } \Gamma_D \\ \frac{\partial u}{\partial \mathbf{n}} = g & \text{on } \Gamma_N \end{cases} \tag{5.1}$$

is considered, where Ω is a bounded domain in $\mathbb{R}^2$ with a Lipschitz-continuous boundary $\partial\Omega$ composed of Dirichlet portion Γ_D and Neumann portion Γ_N where $\partial\Omega = \Gamma_D \cup \Gamma_N$ and $\Gamma_D \cap \Gamma_N = \varnothing$. $Lu = -\triangle u$ is a linear second order differential operator and $f \in L_2(\Omega)$ is the prescribed loading function. The boundary data u_D and g are assumed to be sufficiently smooth and $\mathbf{n}$ denotes the outward unit normal vector to

Γ_N. Using standard norm notation for Sobolev and Hilbert spaces as in Reference [14] and equipped with scalar L^2 inner product $(.,.)$, the Galerkin weak formulation of Equation (5.1) can be written as

$$\begin{cases} \text{Find } u \in V = H_0^1(\Omega) \text{ such that} \\ \mathcal{B}(u,v) = (f,v) + \langle g, v \rangle_{\Gamma_N} \qquad \forall v \ \in V \end{cases} \tag{5.2}$$

where the bilinear $\mathcal{B}$ is defined as $\mathcal{B}(u,v) = \int_\Omega (\nabla u \cdot \nabla v)$, $(f,v) = \int_\Omega f \ v$ is the L^2-inner product and $\langle g, v \rangle = \int_{\Gamma_N} g \ v$ is the boundary integral term.

To specify a Galerkin finite element formulation for (5.2), a shape regular discretization [15] of the domain Ω into a set of triangular partitions $\mathcal{T}_h$ with a set of nodes $\mathcal{N}_h$ and edges $\mathcal{E}_h$ is introduced. The number of elements is given by the cardinality of the set of partitions $|\mathcal{T}_h|$. Each pair of partitions in $\mathcal{T}_h$ is either disjoint or intersects at a common edge or a common vertex. A finite dimensional space $V_h \subset V$ is defined over the discretization $\mathcal{T}_h$, with a nonuniform size parameter h, called the element diameter and defined by the diameter of the minimal ball circumscribed around the mesh element. Defining V_h using polynomials of degree at most 1 on each element $K \in \mathcal{T}_h$, the first order Galerkin finite element formulation is

$$\begin{cases} \text{Find } u_h \in V_h \text{ such that} \\ \mathcal{B}(u_h, v_h) = (f, v_h) + \langle g, v_h \rangle_{\Gamma_N} \qquad \forall v_h \ \in V_h \end{cases} \tag{5.3}$$

The discretization error e is defined as $e = u - u_h$, where u is the exact solution and u_h is the finite element solution. A standard way to measure the error is through the energy norm defined as $\|u\| = \sqrt{\mathcal{B}(u,u)}$. The residual of the finite element solution u_h is defined as

$$\mathcal{R}_h(v) = (f,v) - \mathcal{B}(u_h, v) + \langle g, v \rangle_{\Gamma_N} = \mathcal{B}(e,v) \qquad \forall v \ \in V \tag{5.4}$$

The Galerkin orthogonality property follows directly as $\mathcal{B}(e, v_h) = \mathcal{R}_h(v_h) = 0, \ \forall v_h \ \in$

V_h. Implicit a posteriori error estimation techniques are based on approximating the solution of Equation (5.4) efficiently [3].

5.3 The variational multiscale method

In this section the two level variational multiscale method according to Hughes [13, 16] is presented. The VMS is based on the fundamental concept of scale decomposition, where the solution is decomposed into coarse scale and fine scale components. The fine scale components are defined as the components neglected after projecting the solution space into a finite dimension space defined by the domain discretization (mesh). For advection dominated equations, the solution tends to oscillate when the fine scale components are neglected. To overcome this oscillation in the solution, a stabilization method is commonly incorporated in the finite element formulation to account for the subgrid energy.

The overlapping sum decomposition of the solution space V into a coarse scale subspace $V_c \subset V$ and fine scale subspace $V_f \subset V$ is formalized as

$$V = V_c \oplus V_f \tag{5.5}$$

A similar decomposition is introduced for the solution test and trial functions $u = u_c + u_f$ and $u = v_c + v_f$, where $u_c, v_c \in V_c$ and $u_f, v_f \in V_f$. Using this space decomposition in the Galerkin weak formulation presented in Equation (5.2) yields the following two scale variational equation: Find $u_c \in V_c$ and $u_f \in V_f$ such that

$$\mathcal{B}(u_c, v_c) + \mathcal{B}(u_f, v_c) = (f, v_c) + \langle g, v_c \rangle_{\Gamma_N} \qquad \forall\, v_c \in V_c \tag{5.6}$$

$$\mathcal{B}(u_c, v_f) + \mathcal{B}(u_f, v_f) = (f, v_f) + \langle g, v_f \rangle_{\Gamma_N} \qquad \forall\, v_f \in V_f \tag{5.7}$$

When the term $\mathcal{B}(u_f, v_c)$ in Equation (5.6) is neglected, the standard finite element formulation is obtained, with u_c corresponding to u_h in Equation (5.3). Equation (5.7) is the global residual equation that can be solved globally to obtain a reference

estimation of the error.

5.4 A general framework for implicit local error estimation

The goal of error estimation techniques is to estimate the error field e by approximating the residual Equation (5.4) without solving a global problem. The error in the Galerkin finite element solution u_h has two components: coarse scale and fine scale errors. Using a mesh dependent projection operator $\Pi_h : V \rightarrow V_h$, the coarse scale error field is defined as $e_c = \Pi_h u - u_h$ and the fine scale error is defined as $e_f = u - \Pi_h u$. It is postulated that fine scale errors dominate the error norm. For problems with small subgrid energy contributions the contribution of ∇e_f is expected to be much larger than ∇e_c and thus dominate the error norm. For problems where the contribution of the subgrid energy is large, neglecting the term $\mathcal{B}(u_f, v_c)$ in the VMS formulation might result in physically unrealistic oscillations in the solution u_c. Stabilization techniques are usually applied to eliminate the oscillations as in the case of advection diffusion equations. Once a stable coarse scale solution is obtained, the coarse scale errors can be neglected in comparison to the fine scale errors. Moreover, the most successful local error estimation technique based on the ERM implicitly neglects the errors at the mesh nodes, where only edge bubbles and element interior bubbles are employed to solve the local element problem [3]. Based on that postulation, a *practical objective* is set for the proposed framework to estimate the fine scale error reliably.

Accordingly, the finite element solution u_h is assumed to be a good approximation of u_c and a special form of the global residual equation can be obtained. Replacing u_c with u_h and u_f with e_f in Equation (5.7) yields,

$$\mathcal{B}(e_f, v_f) = (f, v_f) + \langle g, v_f \rangle_{\Gamma_N} - \mathcal{B}(u_h, v_f) \qquad \forall\, v_f \in V_f \tag{5.8}$$

In the proposed framework, the focus is on solving Equation (5.8) locally. To obtain a consistent formulation, a localization function ψ is applied to the strong equation

corresponding to Equation (5.8). The corresponding global residual equation is derived by multiplying the strong equation by the fine scale test function v_f, and then by applying Green's theorem. At this level, a localization function ψ is introduced as follows:

$$\begin{aligned} (-\Delta(u_h + e_f), v_f) &= (f, v_f) && \Longrightarrow \text{Global Strong Equation} \\ (-\Delta(u_h + e_f), v_f\ \psi) &= (f, v_f\ \psi) && \Longrightarrow \text{Localized Equation using } \psi \end{aligned}$$

where, the coarse scale error in u_h is neglected. Integrating by parts results in

$$\mathcal{B}(e_f, v_f\ \psi) = (f, v_f\ \psi) - \mathcal{B}(u_h, v_f\ \psi) + \langle g, v_f\ \psi \rangle_{\Gamma_N} \qquad \forall\ v_f\ \in V_f \tag{5.9}$$

The localization functions ψ can have different forms. For example, it can have a unit value over the problem domain Ω to obtain the global residual equation. If ψ is defined over one element with a unit value and vanishes everywhere else in the domain, the ERM formulation is obtained. If the localization function is set as a partition of unity then a consistent formulation of the subdomain residual method (SRM) is obtained. After selecting the localization function ψ, other technical problems such as the boundary conditions and the solvability of local problems have to be addressed. In the following subsections, the properties of the proposed framework are presented followed by a consistent derivation of the element residual method and a new subdomain residual method for a posteriori error estimation.

5.4.1 Upper bound property of the estimator

Theorem 5.4.1. *Let e_f be the estimated error by the proposed framework, e the exact error and $\Pi_h e$ the projection of the exact error to the local subspace defined by the finite element discretization, the following inequality holds:*

$$\|e_f\|^2 \geq \|e\|^2 - \|\Pi_h e\|^2 \tag{5.10}$$

Proof

Given the properties of the approximation space decomposition of $V = \Pi_h V \oplus V_f$, one can substitute $v_f = v - \Pi_h v$ in the fine scale residual equation and get

$$B(e_f, v_f) = \mathcal{R}_h(v_f) \Longrightarrow \mathcal{B}(e_f, v - \Pi_h v) = \mathcal{R}_h(v - \Pi_h v) \tag{5.11}$$

Replacing v by the exact error e, as both functions are continuous functions, Equation (5.11) becomes

$$\mathcal{B}(e_f, e - \Pi_h e) = \mathcal{R}_h(e - \Pi_h e) = \mathcal{R}_h(e) - \mathcal{R}_h(\Pi_h e) \tag{5.12}$$

By using the Galerkin orthogonality property, the residual of the nodal projection of the exact error vanishes ($\mathcal{R}_h(\Pi_h e) \equiv \mathcal{R}_h(v_h) = 0$). Substitution of $\mathcal{R}_h(e) = \mathcal{B}(e, e) = \|e\|^2$ in Equation (5.12) yields

$$\mathcal{B}(e_f, e - \Pi_h e) = \|e\|^2 \tag{5.13}$$

Employing this result in the following expansion,

$$\|(e - \Pi_h e) - e_f\|^2 = \|(e - \Pi_h e)\|^2 + \|e_f\|^2 - 2\mathcal{B}(e_f, e - \Pi_h e) \geq 0 \tag{5.14}$$

yields,

$$\|(e - \Pi_h e)\|^2 + \|e_f\|^2 - 2\|e\|^2 \geq 0 \tag{5.15}$$

Rearranging Equation (5.15), one gets

$$\|e_f\|^2 \geq 2\|e\|^2 - \|(e - \Pi_h e)\|^2 \tag{5.16}$$

Expansion of the last term in the right hand side results

$$\|e_f\|^2 \geq 2\|e\|^2 - \Big(\|e\|^2 + \|\Pi_h e\|^2 - 2\mathcal{B}(e, \Pi_h e)\Big) \tag{5.17}$$

The term $\mathcal{B}(e, \Pi_h e)$ vanishes because of the Galerkin orthogonality. This yields,

$$\|e_f\|^2 \geq \|e\|^2 - \|\Pi_h e\|^2 \tag{5.18}$$

The theorem demonstrates that any error estimator that is based on approximating the fine scale error only is a theoretical upper bound for the term $\|e\|^2 - \|\Pi_h e\|^2$. This implies that $\|e_f\|$ is a sharp estimator of the exact error norm $\|e\|$ when the norm of the coarse scale errors $\|\Pi_h e\|^2$ is small. Different localization techniques in Equation (5.9) result in different approximation of e_f denoted as e_m where m refers to the localization method used. The estimated error field e_m might be discontinuous and thus the error norm $\|e_m\|$ might over estimate the exact error norm $\|e_m\|$. The computed error is denoted by $\widetilde{e}_m$ and is practically approximated using a finite dimension space.

5.4.2 Lower bound property of the estimator

Estimation of a lower bound to the error norm can be done using a continuous approximation of the error field [17]. A postprocessing approach using the potentially discontinuous upper bound error field $\widetilde{e}_m$ is used to produce a continuous error field. The quality of the lower bound estimate depends on the quality of the upper bound error estimate as well as the smoothing operator used to derive a continuous error field.

Theorem 5.4.2. *For any continuous field ξ, the following lower bound property holds [17]:*

$$\frac{\mathcal{R}_h(\xi)^2}{\|\xi\|^2} \leq \|e\|^2 \tag{5.19}$$

Proof (according to Ref. [17])
Using the following expansion

$$0 \leq \|e - \lambda\xi\|^2 = \|e\|^2 + \lambda^2\|\xi\|^2 - 2\lambda\mathcal{B}(e, \xi) \tag{5.20}$$

where λ is a constant. For a continuous field ξ, $\mathcal{B}(e,\xi) = \mathcal{R}_h(\xi)$. Using this property in the above expansion one obtains the lower bound property,

$$2\lambda\mathcal{R}_h(\xi) - \lambda^2\|\xi\|^2 \leq \|e\|^2 \tag{5.21}$$

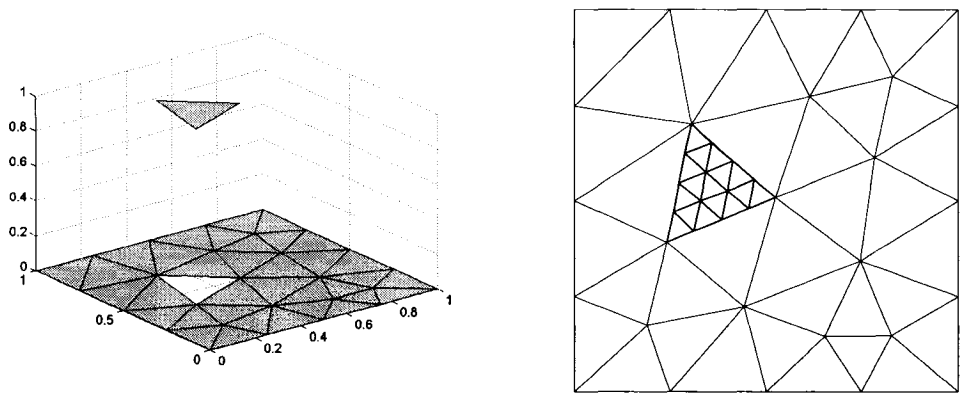
This equation is valid for any value of λ. To obtain a sharp lower bound estimate, an optimal value of λ that maximizes the right hand side is used. Differentiating the left hand side and setting the result to zero, one obtains an optimal value of $\lambda = \mathcal{R}_h(\xi)/\|\xi\|^2$. Back-substitution of the optimal λ value results in the lower bound estimate.

5.5 A variational multiscale element residual method

The localization function ψ is defined over the mesh element $K \in \mathcal{T}_h$ as $\psi_K = 1$ and vanishes over all other mesh elements. This results in a number of decoupled local problems corresponding to the number of mesh elements. The formalization of the method follows Equation (5.9) as

$$\mathcal{B}(e_f, v_f\ \psi_K) = (f, v_f\ \psi_K) + \left\langle \frac{\partial(u_h + e_f)}{\partial \mathbf{n}_E}, v_f\ \psi_K \right\rangle_{E\in\mathcal{E}_h} - \mathcal{B}(u_h, v_f\ \psi_K) \tag{5.22}$$

where $\mathbf{n}_E$ denotes a unit normal vector to the edge E. Figure 5.1 shows a plot of the localization function ψ_K. All the terms in Equation (5.22) are evaluated locally over the element K because of the properties of ψ_K. A local uniform refinement over each element can be used to obtain an approximation space for solving the local problems, as shown in Figure 5.1 (b). The boundary integral term resulting from the integration by parts needs special treatment as it depends on the unknown exact solution $u = u_h + e$. This boundary term is usually approximated by averaging the



(a) Localization function (b) Discretization of the local problem

Figure 5.1: Localization and discretization of the ERM

flux from two adjacent elements as

$$\frac{\partial(u_h+e)}{\partial \mathbf{n}_E} \approx g|_E = \begin{cases} \frac{1}{2}\,\mathbf{n}_E \cdot \{(\nabla u_h)_K + (\nabla u_h)_{K'}\} & \text{on } E \subseteq (\partial K \cap \partial K') \\ g & \text{on } E \subseteq (\partial K \cap \Gamma_N) \end{cases} \tag{5.23}$$

where K and K' are two adjacent elements sharing the edge E. In the current formulation, the fine scale errors at the Dirichlet boundary ($\partial K \cap \Gamma_D$) are applied as an essential boundary condition for the local problem.

The detailed calculation of the upper bound error estimates starts by solving a set of local problems

$$\begin{cases} \text{Find } e_{erm} \in V_f \text{ such that} \\ \mathcal{B}_K(e_{erm}, v_f) = (f, v_f)_K - \mathcal{B}_K(u_h, v_f) + \langle g|_E, v_f \rangle_{E \subset \partial K} \quad \forall v_f \in V_f \end{cases} \tag{5.24}$$

where e_{erm} is the error field approximated by the ERM and the subscript K denotes a restriction of the bilinear and the loading term over an element K. An approximate space V_{fh} is defined by a locally refined discretization of each element and the error evaluated in the space $V_{fh} \subset V_f$ is denoted by $\widetilde{e}_{erm}$. The local error norm η_K is

evaluated as

$$\eta_K = \left\{ \mathcal{B}_K(\widetilde{e}_{erm}, \widetilde{e}_{erm}) \right\}^{1/2} \tag{5.25}$$

and the global error in the energy norm is calculated as the sum of the local contributions as

$$\eta_{erm} = \left\{ \sum_{K \in \mathcal{T}_h} \eta_K^2 \right\}^{1/2} \tag{5.26}$$

The solvability of the local problems is guaranteed iff the boundary fluxes are balanced. Flux equilibration methods have been proposed in [10, 18] to correct the weights used in averaging the edge flux jumps. However, flux equilibration is difficult to implement and rarely applied for practical problems [19]. It should be noted that the estimated error norm is a guaranteed upper bound of the exact error iff the local problems are solved exactly and flux equilibration is applied. The analysis of the asymptotic exactness of the error estimate can be found in [10] and is beyond the scope of the current work.

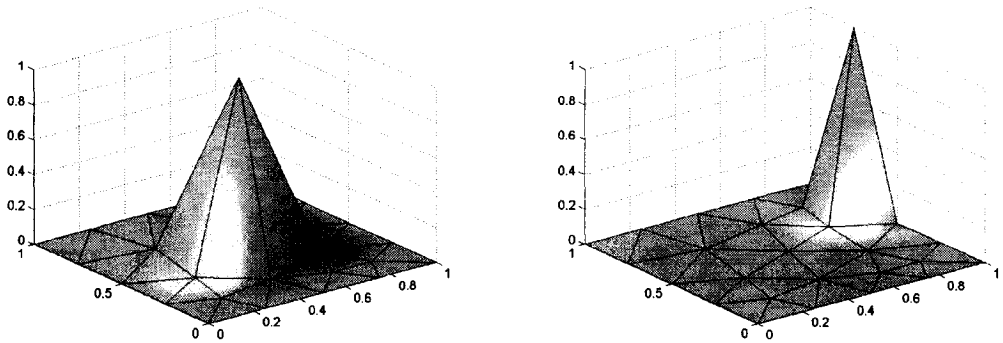
Obtaining a lower bound error estimate from the ERM is based on smoothing the discontinuous error field $\widetilde{e}_{erm}$ by averaging the errors over each mesh edge. This smoothing operation is sufficient to produce a lower bound of the error norm, but the quality of such an estimate is poor. The poor performance is attributed to the introduction of artificial deformation of the error field by averaging the error at the mesh edges, while keeping the error values in the element interior unchanged. Following the work in [19], local problems described by Equation (5.24) and subjected to Dirichlet boundary conditions are solved to obtain an improved continuous error field. The Dirichlet boundary values are the averaged error on each mesh edge. The newly calculated element interior error field and the averaged edge errors are used to calculate a lower bound estimate using Equation (5.19).

5.6 A variational multiscale subdomain residual method

The first subdomain residual method (SRM) was introduced by Babuška and Rheinboldt [20]. The residual was decomposed using a partition of unity and local problems were formulated over nodal based patches with homogeneous essential boundary conditions. The error field was assumed to be local due to the locality of the residual term after the decomposition. This assumption is not valid on coarse meshes, or if the problem has any multiscale features [21]. The performance of that SRM formulation was found to significantly underestimate the errors. Recently, four different variations of the subdomain error estimation techniques were proposed [11, 12, 22, 23]. The major differences between these methods are in the definition of the bilinear used to calculate the error and the boundary conditions of the local problems. Additionally, constraints were specified on the approximation space of the local problems to eliminate zero energy modes. Published numerical results [12, 24] show that subdomain based error estimation techniques are not as reliable as the ERM, with the exception of the work of Parés *et al.* [23] where the estimator is claimed to be as sharp as the ERM.

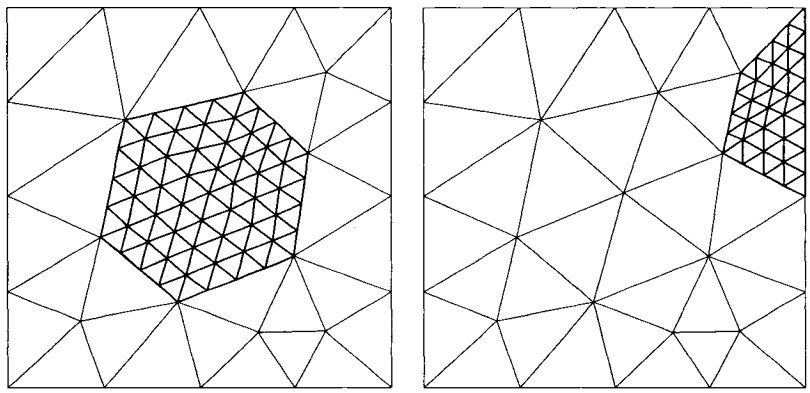
In the current work, the subdomain residual estimate is obtained by defining the localization function ψ in Equation (5.9) as a partition of unity $\psi = \varphi_N$, corresponding to the mesh node $N \in \mathcal{N}_h$. The localization function is applied before the integration by parts to produce a consistent bilinear form in contrast to the bilinear used in other SRM formulations [12, 22]. The SRM is formulated as a number of overlapping local problems defined over a nodal based patches where $\mathcal{P}(N)$ denotes the set of mesh elements containing the node N. Each problem is defined as:
Find $e_{srm} \in V_f$ such that

$$\mathcal{B}_{\mathcal{P}(N)}(e_{srm}, v_f\ \varphi_N) = (f, v_f\ \varphi_N)_{\mathcal{P}(N)} - \mathcal{B}(u_h, v_f\ \varphi_N)_{\mathcal{P}(N)} + \left\langle \frac{\partial(u_h + e)}{\partial \mathbf{n}_E}, v_f\ \varphi_N \right\rangle_{E \subset \mathcal{P}(N)} \qquad \forall v_f \in V_f \qquad (5.27)$$



(a) Internal patch (b) Boundary patch

Figure 5.2: Localization function ψ used in the subdomain error estimator



(a) Internal patch (b) Boundary patch

Figure 5.3: Discretization of the local problem

The integral term over the edges $E \subset \mathcal{P}(N)$ vanishes on the interior edges of the patch because the exact solution $u = (u_h + e)$ is continuous. It also vanishes on the boundary edges because it is multiplied by the partition of unity φ_N. This results in a flux-free consistent formulation for the SRM. The solution space of the local problem is obtained by h-refinement for each patch. Figure 5.2 shows a plot of a partition of unity for internal and boundary nodal based patches and Figure 5.3 shows the corresponding h-refinement mesh for each patch with two levels of local refinement.

The discretized form of the local problems at an internal node is defined as

$$\mathcal{B}_{\mathcal{P}(N)}(\widetilde{e}_{srm}, v_f\ \varphi_N) = (f, v_f\ \varphi_N)_{\mathcal{P}(N)} - \mathcal{B}(u_h, v_f\ \varphi_N)_{\mathcal{P}(N)} \qquad \forall v_f \in V_{fh} \tag{5.28}$$

where $\widetilde{e}_{srm}$ is the error field approximated by the SRM and the subscript $\mathcal{P}(N)$ denotes a restriction over the patch. For the case of boundary nodes the externally specified Neumann conditions are applied on $\Gamma_N \cap \partial\mathcal{P}(N)$ and the exact error in the finite element solution $e_f = u_D - u_h$ is enforced on the Dirichlet portion of the boundary condition $\Gamma_D \cap \partial\mathcal{P}(N)$. For internal patches with a zero flux boundary condition, zero energy modes are eliminated by obtaining a solution in the space $V_f = V - \Pi_h V$ (see remark 3.1 in [22]). By eliminating the constraints on the solution space, the implementation of the proposed estimator becomes much simpler than the other subdomain methods [11, 12].

On triangular meshes, the current estimator evaluates the error field over each element three times corresponding to the three nodal patches. There are alternative methods to compute the error field; the simplest one is to calculate the error norm using the bilinear $\mathcal{B}(\widetilde{e}_{srm}, \widetilde{e}_{srm})$ from each patch and then divide the result by the number of partitions of unity spanning over the element (three for triangular meshes). In that case, the discontinuity of the error field is accounted for implicitly without the need to store a discontinuous error field. If one denotes the error field calculated using the SRM formulation corresponding to the nodal problem N as $\widetilde{e}_{srmN}$, the local error norm is then evaluated as

$$\eta_K = \left\{ \frac{\sum_{N \subset \partial K} \mathcal{B}_K(\widetilde{e}_{srmN}, \widetilde{e}_{srmN})}{\sum_{N \subset \partial K} 1} \right\}^{1/2} \tag{5.29}$$

and the global error in the energy norm is calculated from

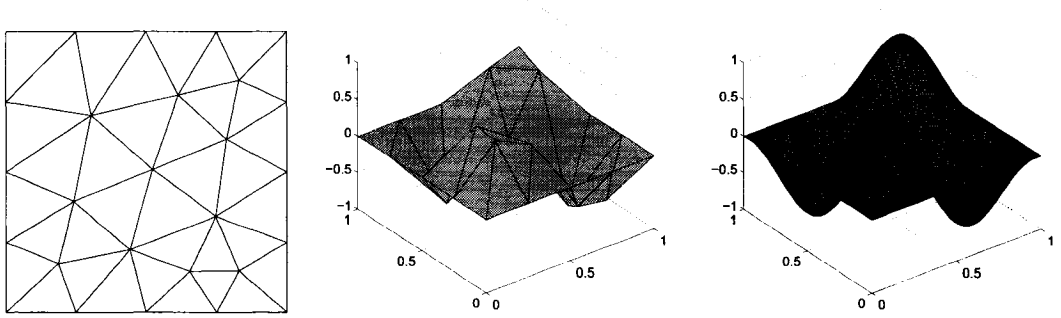
$$\eta_{srm} = \left\{ \sum_{K \in \mathcal{T}_h} \eta_K^2 \right\}^{1/2} \tag{5.30}$$

To calculate a lower bound for the error norm, a continuous error field is evaluated by smoothing the solution from the patch based problems. A partition of unity is used to weigh the error field resulting from each patch solution and the summation of the contributions from each patch results in a continuous field. Since the continuous error field is efficiently evaluated, a lower bound estimate of the error norm can be obtained using Equation (5.19).

5.7 Application to Poisson equations

The proposed ERM and SRM are evaluated numerically for Poisson type equations. Four test problems are formulated using Equation (5.1), where the load function f and the boundary data are set such that the solution satisfies the desired exact solution. For all four test problems, the procedure adopted to calculate the errors is as follows: (i) The upper bound ERM estimate, denoted ERM-up, is calculated using Equation (5.24). The local contributions are collected following Equations (5.25) and (5.26). (ii) The lower bound ERM estimate, denoted ERM-lo, is calculated by first recovering a continuous error field from $\widetilde{e}_{erm}$, and then using the general lower bound formula of Equation (5.19). (iii) The solution of the local problems is done by introducing an h-refinement of the local domain. The symbol RL is used to refer to the refinement level. Each triangle is refined to 16 triangles in the case of $RL = 2$ and 64 triangles in the case of $RL = 3$. (iv) The upper bound SRM estimate, denoted SRM-up, is calculated using Equations (5.28), (5.29) and (5.30). (v) The lower bound SRM estimate, denoted SRM-lo, is calculated using a continuous error field obtained by postprocessing $\widetilde{e}_{srm}$. The continuous error field is plugged into the general lower bound estimate defined by Equation (5.19).

The quality of the upper bound estimates is measured by the effectivity index (EI) which is the ratio of the estimated error norm to the exact error norm. A value close to unity indicates an accurate error estimation. Another index denoted by ρ is used for measuring the quality of both the upper and lower bound error estimates. The



(a) Initial mesh (b) Sol. on initial mesh (c) 4 it. to global refinement

Figure 5.4: Problem P1 - Initial mesh and finite element solution

index ρ is defined as

$$\rho = \left\{ \frac{\text{estimated error}}{\text{exact error}} - 1 \right\} \times 100\%$$

Test problem P1

The first test problem is defined over the domain $\Omega = (0,1) \times (0,1)$ with the loading function f set such that the exact solution is

$$u_{exact} = \sin(2\pi x) \ \sin(2\pi y) \tag{5.31}$$

Figure 5.4 shows the initial mesh and the finite element solution on the initial mesh and on a fine mesh obtained by four iterations of global mesh refinement. The exact solution is smooth and symmetric and the initial mesh is unstructured. The initial mesh only captures the main features of the solution, as shown in Figure 5.4 (b).

Table 5.1 shows the index ρ for the estimated errors using the SRM and the ERM. For the case of $RL = 2$, the subdomain method produces a very high quality upper bound error estimate with $\rho = 1.88\%$, for the coarse mesh. The ERM for the same case overestimates the errors by 28%. On refined meshes, the subdomain method slightly underestimates the exact error by about 1%. The quality of the SRM lower bound estimate contains $3 - 5\%$ error. The quality of the ERM estimator is also

Table 5.1: Problem P1 - Effectivity indices for the estimated errors in terms of the index ρ

# nodes	$RL = 2$				$RL = 3$			
	SRM-up	SRM-lo	ERM-up	ERM-lo	SRM-up	SRM-lo	ERM-up	ERM-lo
27	1.883	-5.335	28.461	-10.988	3.539	-2.708	44.523	-14.006
89	-0.799	-4.289	9.908	-6.718	-0.311	-1.907	18.599	-8.279
321	-0.967	-3.954	3.745	-4.632	-0.858	-1.660	10.691	-4.639
1217	-0.798	-3.796	2.087	-3.970	-0.777	-1.523	8.668	-3.416
4737	-0.676	-3.733	1.627	-3.736	-0.669	-1.464	8.140	-2.987

evident and converges to the exact value with mesh refinement. This is attributed to the solution smoothness, which means that the averaging technique used to calculate the boundary conditions of the local problems is valid.

On coarse meshes, the subdomain estimator is very accurate considering that the finite element solution only represents the main features of the exact solution. On the refined meshes, the observed small deviations of the SRM-up and SRM-lo from the exact error norm are attributed to using a finite dimension space for solving the local problems and to the ratio of the coarse scale error norm defined as $\mathcal{B}(\Pi_h e, \Pi_h e)$ to the total exact error norm $\mathcal{B}(e, e)$. One can study the effects of the approximation space by comparing the effectivity indices using $RL = 2$ and $RL = 3$.

The use of three levels of local refinement ($RL = 3$) results in sharper SRM-up and SRM-lo error estimates. Mesh refinement resultes in a slight enhancement of the quality of the SRM upper bound estimate. For the initial coarse mesh, the estimated error field tends to be discontinuous which yields some over-estimation in the error norm, while for finer meshes, the amount of deviation from the exact error is mostly attributed to neglecting the coarse scale errors. The quality of the lower bound error norm improves with expanding the approximation space because of the increased continuity of the error field.

For the ERM expanding the approximation space corresponds to an increased deviations from the exact errors especially for the upper bound estimate. This is

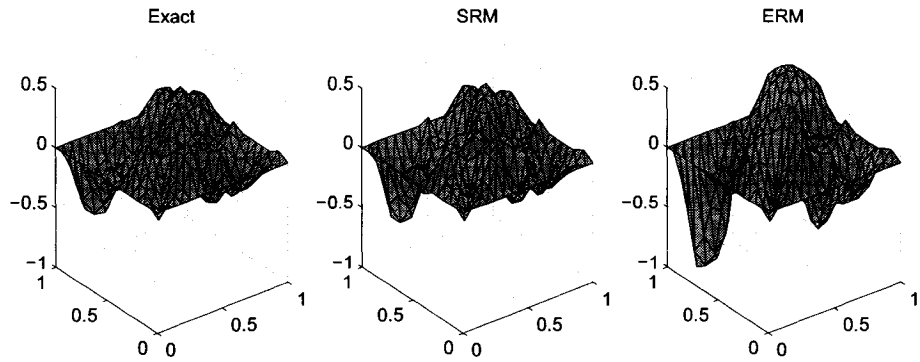


Figure 5.5: Plot of the error field for problem P1 on the initial mesh ($RL = 2$)

attributed to the averaging technique used to approximate the boundary fluxes. The artificial boundary conditions contain spurious modes that are magnified when enlarging the approximation space. The results from this smooth problem provide a strong evidence that the proposed subdomain estimator behaves as expected from the theory and produces very sharp results on coarse meshes in comparison to the ERM.

For a better understanding of the performance of the two estimators, the estimated error fields used to calculate the lower bound error norm are plotted along with the exact errors in Figure 5.5. The subdomain method resulted in an error field that strongly resembles the exact error field in all parts of the domain, while the ERM generally over estimates the errors with major deviation from the exact error over the triangular element with two Dirichlet boundary conditions. This special configuration of the local problem with two prescribed boundary conditions and the third boundary approximated using an averaging technique results in a strongly unequilibrated local problem. For most of the interior mesh elements, the averaging technique results in better approximation because the errors in the boundary term cancels each other, which is not the case for the corner element.

The histogram of the local effectivity indices of the upper bound error estimates is shown in Figure 5.6. The subdomain method produces local effectivity indices in the

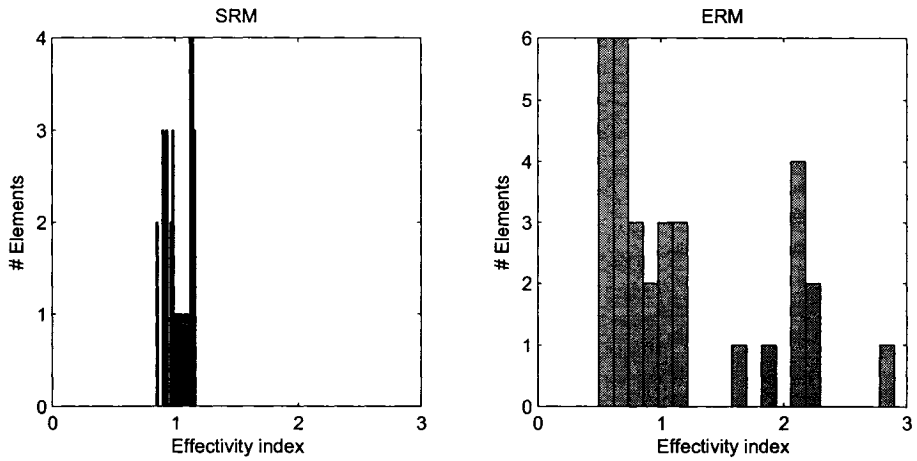


Figure 5.6: Histogram of local effectivity indices for problem P1 using the initial mesh ($RL = 2$)

range of 0.8 to 1.2, which is of very high equality for a coarse mesh. ERM produces local effectivity indices in the range of 0.5 to 3. The large variation in the value of the local effectivity raises question about the quality of the information used for mesh adaptivity based on the ERM.

The histograms of local effectivity indices for the SRM and ERM after two global mesh refinement iterations are plotted in Figure 5.7. For the subdomain method, the local EI is bounded between 0.8 and 1.2 but the distribution has two peaks at the effectivity indices of 0.9 and 1.2. These two peaks are attributed to the local distribution of the coarse scale errors. The distribution for the ERM estimated errors is similar to a normal distribution, with a peak at a local effectivity index of 1. This distribution is favorable as it shows that most elements have an effectivity index close to unity. The good performance of the ERM is again attributed to the smoothness of the problem solution.

The effectivity indices of the local estimated error norms using SRM and ERM are plotted against the magnitude of the error norm using a logarithmic scale in Figure 5.8. The histogram of the exact error norm for each element is also included to show the number of element corresponding to each error norm level. Each element

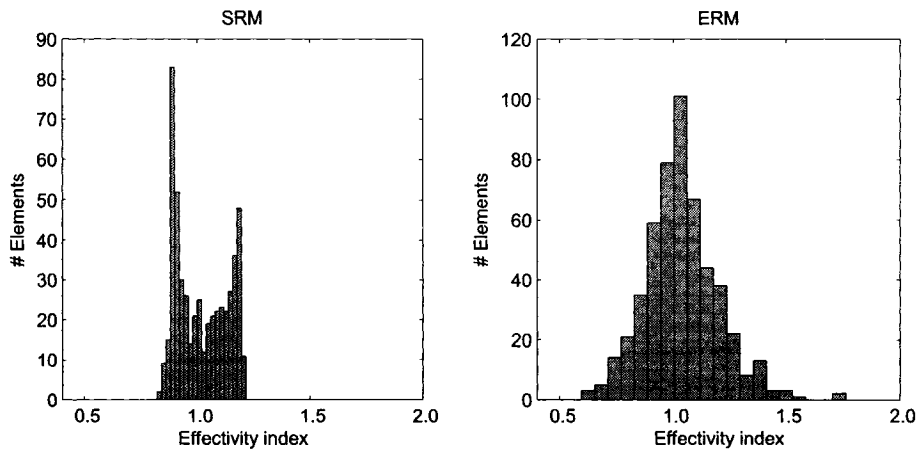


Figure 5.7: Histogram of local effectivity indices for problem P1 after two global mesh refinement iterations ($RL = 2$)

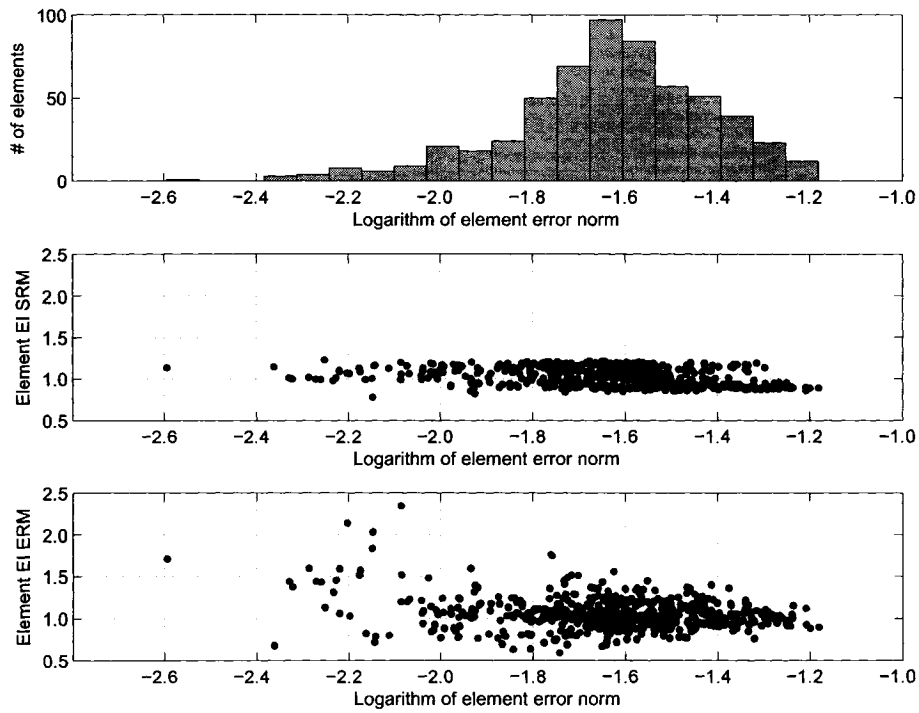


Figure 5.8: Problem P1 - Distribution of local effectivity indices against the exact local error norm. Solution obtained after two global refinement iterations ($RL = 2$)

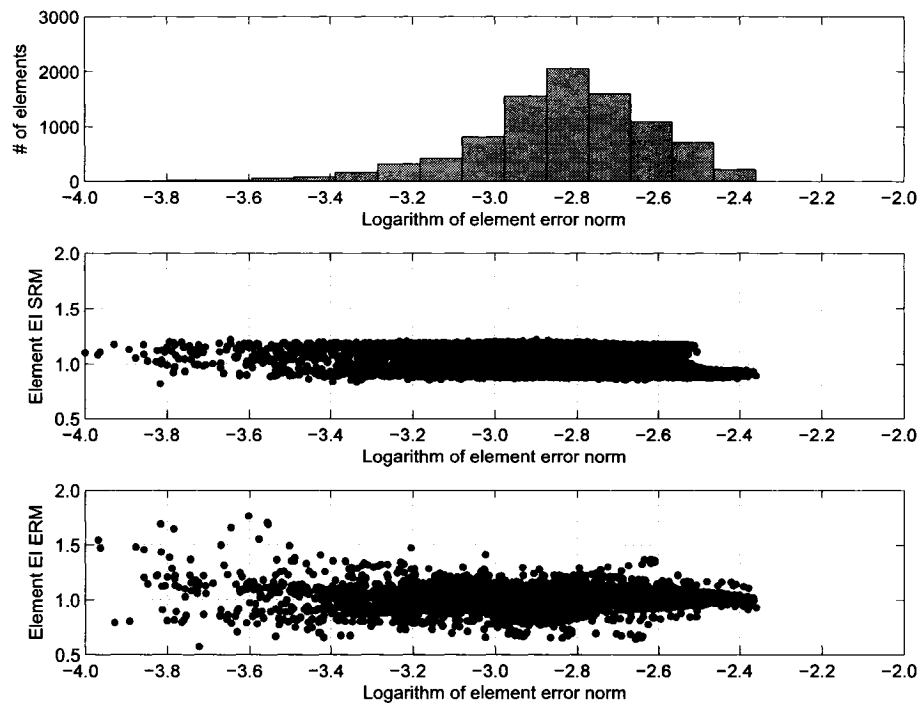


Figure 5.9: Problem P1 - Distribution of local effectivity indices against the exact local error norm. Solution obtained after four global refinement iterations ($RL = 2$)

corresponds to a dot on the graph with the coordinates indicating the exact error for an element against the effectivity index of the element error norm. The SRM local effectivity indices plotted in Figure 5.8 are bounded between 0.8 and 1.2, which indicates the good behavior of the method, independent of the scale of the error norm. On the other hand, ERM produces error norms that are dependent on the error magnitude, with low quality local estimates for elements with small error norms. Figure 5.9 confirms the reliable behavior of the subdomain method in comparison to the ERM on a much finer mesh.

It should be noted that effectivity indices for 10% of the elements with the lowest exact error norm are not included in the histogram plots, such as Figures 5.6 and 5.7, because these elements do not contribute significantly to the global error norm.

However, for the local effectivity comparisons against the exact error norm as plotted in Figures 5.8 and 5.9, the elements with small exact error norms are included.

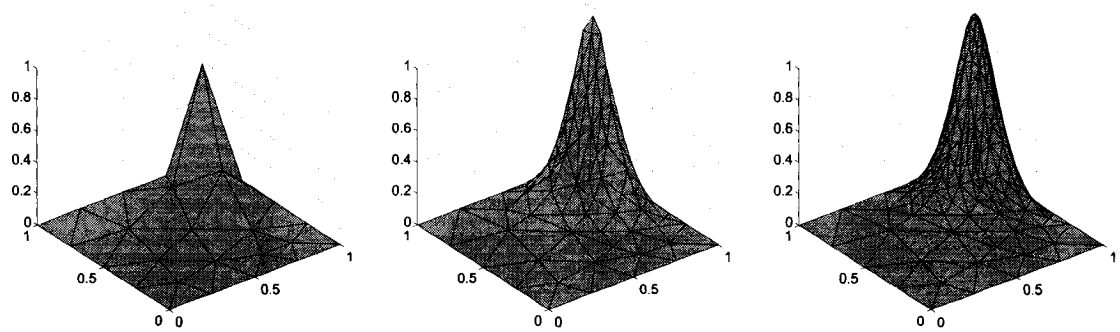
Test problem P2

This problem is defined over the domain $\Omega = (0,1)\times(0,1)$ with the loading function f set such that the exact solution is

$$u_{exact} = 0.005\ x^2\ (1-x)^2\ e^{(10x^2)}\ y^2\ (1-y)^2\ e^{(10y)} \tag{5.32}$$

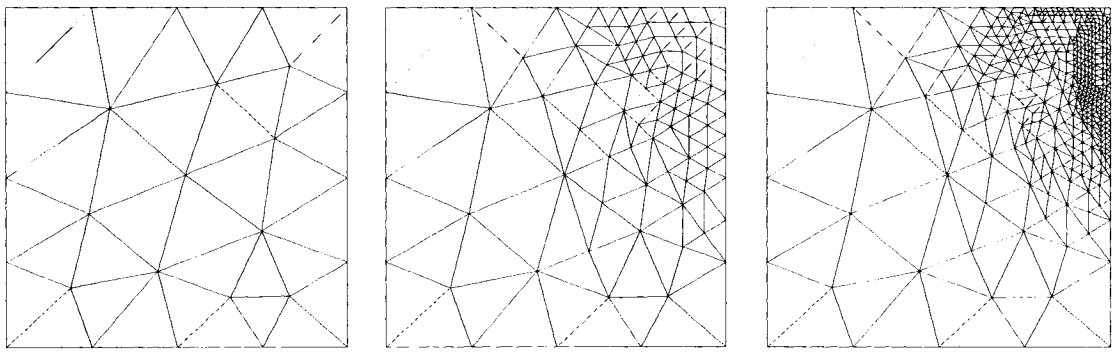
Figure 5.10 shows the finite element solution on the initial mesh and on the adapted meshes after two and four adaptivity iteration. The corresponding meshes are plotted in Figure 5.11. The solution exhibits a localized exponential peak that is hard to capture efficiently using uniform mesh refinement.

It should be noted that the SRM is used to guide the mesh adaptivity for problems P2, P3 and P4. The SRM is selected to drive the adaptivity because of the quality of the estimator, as demonstrated throughout this paper. Adaptive mesh refinement is done by the red-blue-green refinement detailed in [1]. Elements are marked for refinement using the sorting strategy, where mesh elements are sorted according to the estimated error norm, then all the element exceeding the $(70)^{th}$ percentile are marked for refinement. This corresponds to refining at least 30% of the mesh elements at each adaptivity iteration. The global effectivity indices of the estimated errors using SRM and ERM are listed in Table 5.2. For very coarse meshes, the advantage of the SRM over the ERM is evident. The index ρ has a value of 2.5% for the subdomain upper bound estimate and -9.6% for the lower bound estimate. Solving local problems with a larger approximation space ($RL = 3$) resulted in a ρ index equals to 7.5% and -4.3% for the upper and lower bound error estimates, respectively. This result shows that the accuracy of the lower bound increases with enlarging local problems approximation space, while the upper bound error norm might overestimate the exact errors due to the increased discontinuity of the error field. The quality of the estimated



(a) Initial mesh (b) Two adaptive iterations (c) Four adaptive iterations

Figure 5.10: Finite element solution of problem P2 on different meshes



(a) Initial mesh (b) Two adaptive iterations (c) Four adaptive iterations

Figure 5.11: Adapted meshes for problem P2

errors using the SRM improves significantly with adaptive mesh refinement. However, the ERM produces error norms that have a relatively large value of the index ρ.

Figure 5.12 shows a plot of the exact error in the finite element solution on the initial mesh and the corresponding estimated error fields by the SRM and the ERM. The error field approximated by the SRM captures all the features of the exact solution and the error values are almost exact, while the ERM produces results close to the exact error field, but with slight overestimation. The histogram of the local effectivity indices of the estimated errors on the initial mesh are shown in Figure 5.13. The SRM upper bound estimates slightly underestimate the exact error norm on most of mesh elements, while the ERM significantly overestimates the error norm for a few

Table 5.2: Problem P2 - Effectivity indices for the estimated errors in terms of the index ρ

# nodes	$RL = 2$				$RL = 3$			
	SRM-up	SRM-lo	ERM-up	ERM-lo	SRM-up	SRM-lo	ERM-up	ERM-lo
27	2.454	-9.645	15.090	-14.956	7.539	-4.320	28.690	-16.158
49	-0.524	-10.856	19.047	-19.118	2.440	-4.152	31.892	-20.025
96	-2.036	-7.017	13.473	-12.530	0.603	-2.752	25.543	-14.045
194	-2.821	-4.894	14.583	-9.923	-1.265	-2.106	26.935	-12.686
425	-2.751	-5.015	15.161	-11.742	-1.517	-2.577	28.133	-15.912
944	-1.941	-5.117	13.040	-11.608	-1.351	-2.716	25.065	-16.021

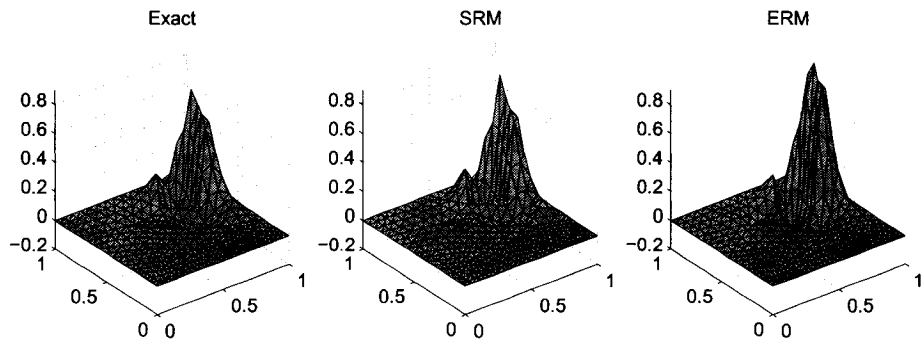


Figure 5.12: Estimated versus exact error field plot for problem P2 solved using the initial mesh ($RL = 2$)

mesh elements. On such a coarse mesh, both error estimators are not expected to produce very reliable results, but the behavior of the ERM is undesirable because it inflates the estimated global error norm artificially due to overestimating the errors over just a few mesh elements.

The histogram of local effectivity indices after two adaptive mesh refinement iterations is plotted in Figure 5.14. The SRM produces upper bound estimates in the range of 0.7 to 1.3 of the exact error norm. The distribution is similar to a normal distribution with the peak slightly below the optimal effectivity index value of one. The ERM produces good error estimation for most of the mesh elements but the estimated error norms for a few mesh elements are almost twice the exact value.

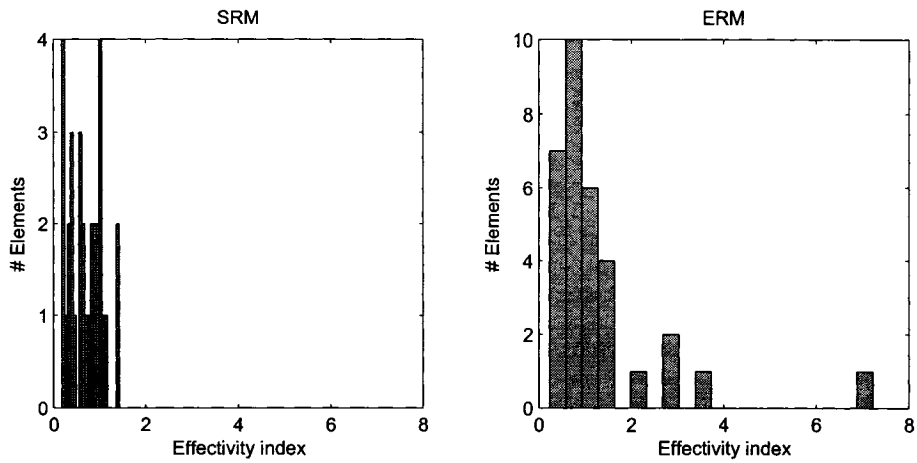


Figure 5.13: Histogram of local effectivity indices for problem P2 using the initial mesh ($RL = 2$)

The quality of the estimated errors are further assessed in relation to the magnitude of the exact error norm. Figure 5.15 shows a plot of the exact error norm on a logarithmic scale against the local effectivity index after two adaptive mesh refinement iterations. The SRM shows superior results in comparison to the ERM with an almost uniform distribution of the local effectivity index across the different error norm scales. The ERM upper bound error estimate tends to be less accurate for elements with a small error norm. Results on adapted meshes after five iterations are shown in Figures 5.16 and 5.17. A narrower histogram of the SRM-up effectivity indices is observed in comparison to those estimated by ERM-up, as shown in Figure 5.16. Moreover, the distribution across the error norm scales is more uniform for the SRM-up than for the ERM-up, as shown in Figure 5.17.

Test problem P3

The third test problem is also defined on the domain $\Omega = (0,1) \times (0,1)$, but in this case more complex boundary conditions are used. The load function f is set such that the exact solution is

$$u = \sin(5\pi xy)\ e^{x+y} \tag{5.33}$$

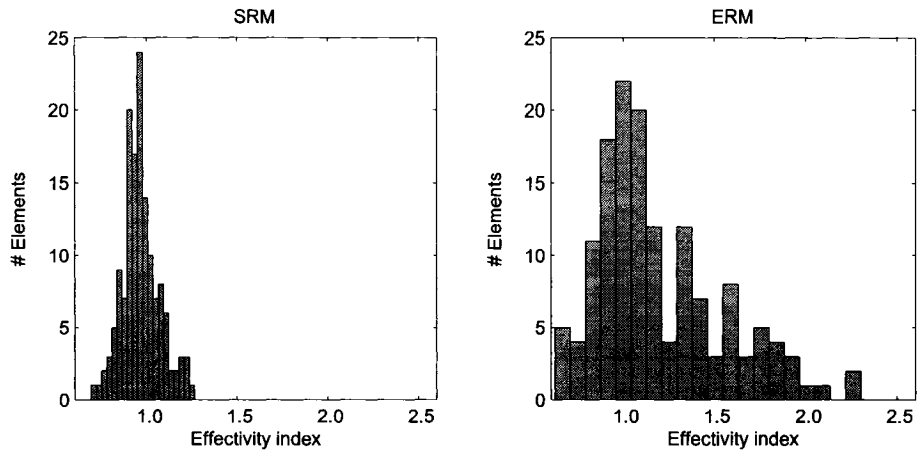


Figure 5.14: Histogram of local effectivity indices for problem P2 after two adaptive mesh refinement iterations ($RL = 2$)

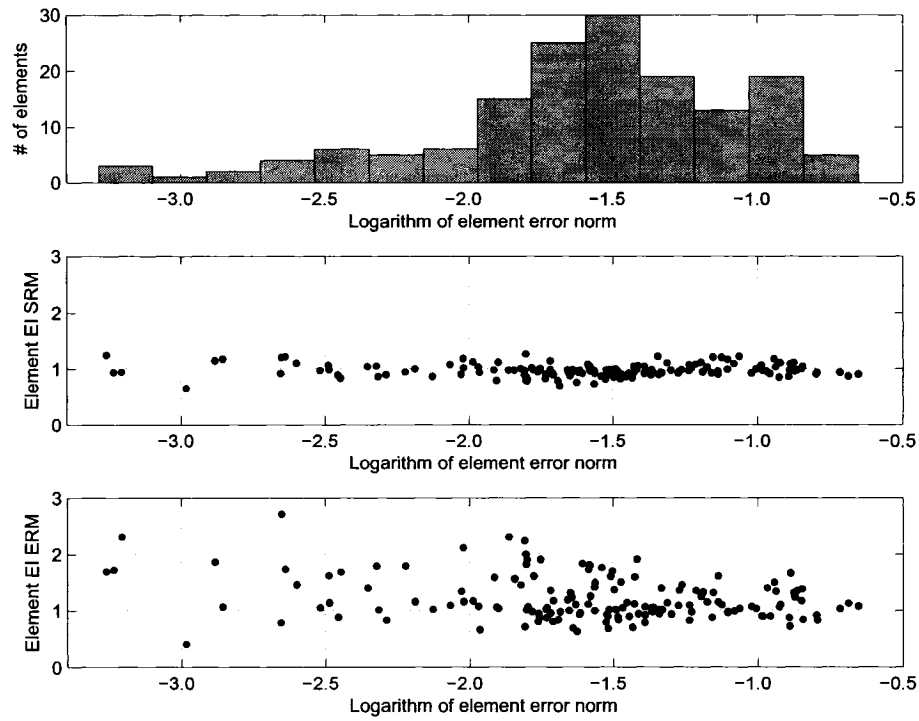


Figure 5.15: Problem P2 - Distribution of local effectivity indices against the exact local error norm. Solution obtained after two adaptive mesh refinement iterations ($RL = 2$)

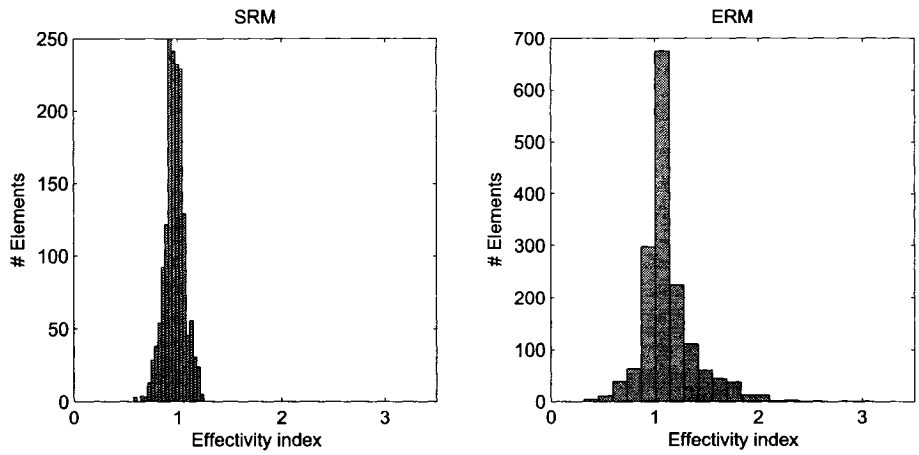


Figure 5.16: Histogram of local effectivity indices for problem P2 after five adaptive mesh refinement iterations ($RL = 2$)

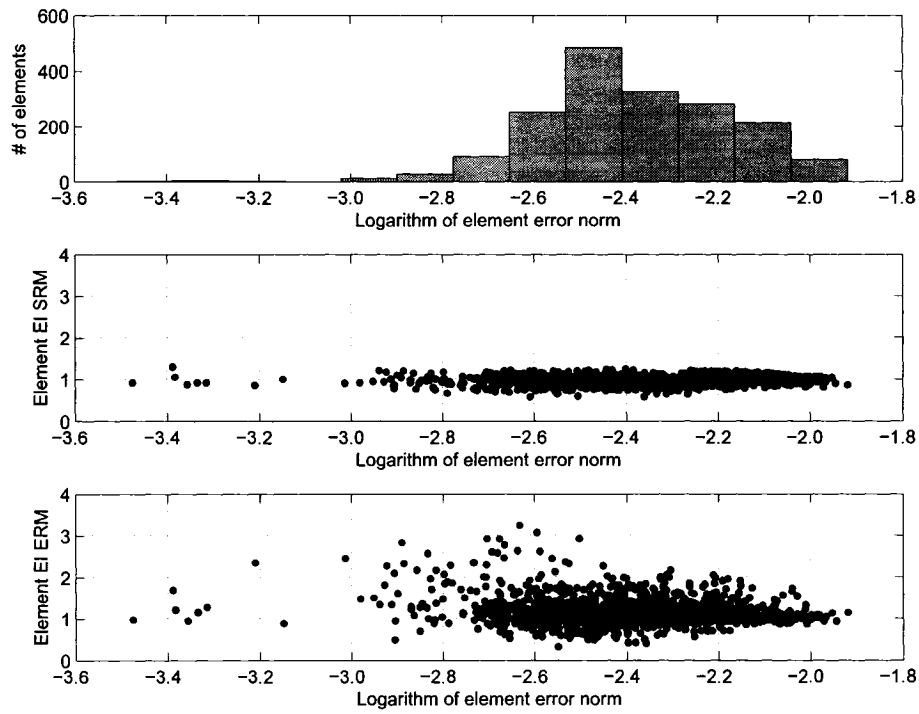
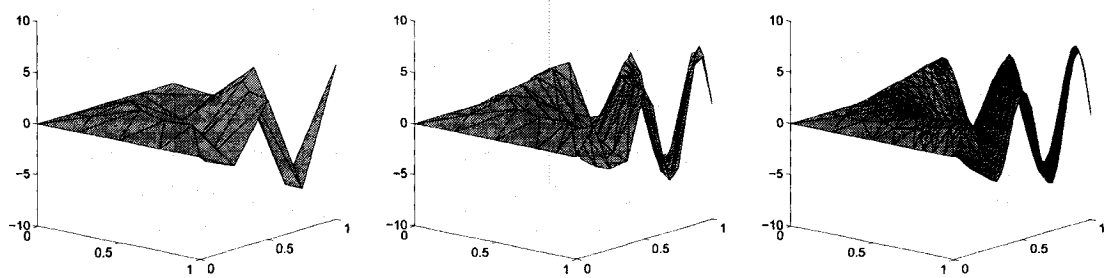
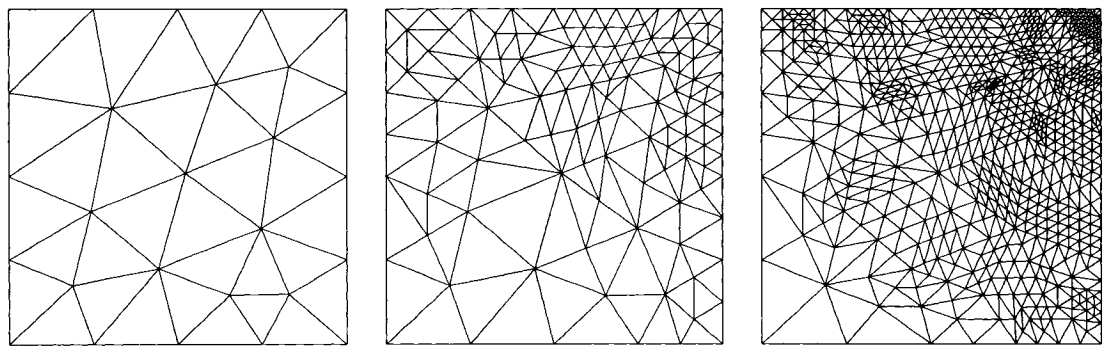


Figure 5.17: Problem P2 - Distribution of local effectivity indices against the exact local error norm. Solution obtained after five adaptive mesh refinement iterations ($RL = 2$)



(a) Initial mesh (b) two adaptive iterations (c) four adaptive iterations

Figure 5.18: Finite element solution of problem P3 on different meshes



(a) Initial mesh (b) two adaptive iterations (c) four adaptive iterations

Figure 5.19: Adapted meshes for problem P3

A homogeneous Dirichlet boundary condition is specified at the boundaries with $x = 0$ or $y = 0$ and a Neumann boundary condition that satisfies the exact solution is specified on the rest of the domain boundary. The specified boundary conditions makes it hard to obtain a correct solution at the corner with $x = 1$ and $y = 1$. Figure 5.18 shows a 3D-plot of different finite element solutions using three different discretization and Figure 5.19 shows the corresponding meshes. Table 5.3 lists the index ρ values for the SRM and the ERM on six different meshes obtained by SRM driven mesh adaptivity iteration. The estimated errors using the SRM on coarse meshes are far better than those produced by the ERM estimator. In some cases, an apparent deterioration of the quality of estimated errors using the SRM is observed with mesh refinement. This deterioration only happens when the estimated error

Table 5.3: Problem P3 - Effectivity indices for the estimated errors in terms of the index ρ

# nodes	$RL = 2$				$RL = 3$			
	SRM-up	SRM-lo	ERM-up	ERM-lo	SRM-up	SRM-lo	ERM-up	ERM-lo
27	6.182	-14.214	37.463	-25.550	11.173	-12.530	56.987	-28.209
51	-1.476	-5.533	28.883	-17.310	-0.141	-3.207	45.910	-21.309
122	-3.429	-7.208	20.353	-17.904	-2.082	-4.514	35.306	-22.494
300	-3.838	-9.189	14.808	-19.183	-2.941	-6.280	30.753	-24.514
697	-4.403	-8.268	15.669	-16.395	-3.328	-5.183	30.467	-20.883
1581	-3.571	-7.210	14.509	-15.385	-2.564	-4.962	28.164	-20.570

norm falls between the exact error norm ($\|e\|^2$) and the theoretical upper bound of the error estimator ($\|e\|^2 - \|\Pi_h e\|^2$). The results from this problem confirm that enlarging the approximation space used to solve the local problems increases the quality of the lower bound SRM estimator and decreases the underestimation of the upper bound whenever it exists. The estimated errors using the ERM are not as sharp as the SRM. Moreover, the ERM requires more refined meshes to produce results that are comparable to the SRM.

A plot of the estimated error field used to calculate the lower bound error norm for the SRM and the ERM is shown in Figure 5.20. The results correspond to the finite element solution on the initial mesh and using two levels of local refinement for error estimation. Although, the magnitude of the coarse scale errors is large, the SRM captures the oscillating nature of the errors without any significant differences from the exact error. The ERM overestimates the error value in a few locations in the domain, most notably close to the corner where $x = 1$ and $y = 1$. The histograms of the local effectivity indices of the estimated errors are plotted in Figure 5.21. The SRM error estimator again produces errors in the range from 0.7 to 1.3, which are excellent results given that the finite element solution used for calculating the residual is produced on coarse mesh and has large coarse scale errors. The ERM produces a relatively overestimated errors on most of the elements and the effectivity index falls

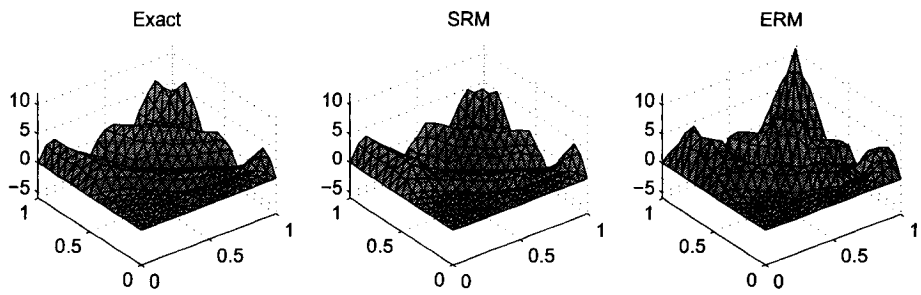


Figure 5.20: Estimated versus exact error field plot for problem P3 using initial mesh ($RL = 2$)

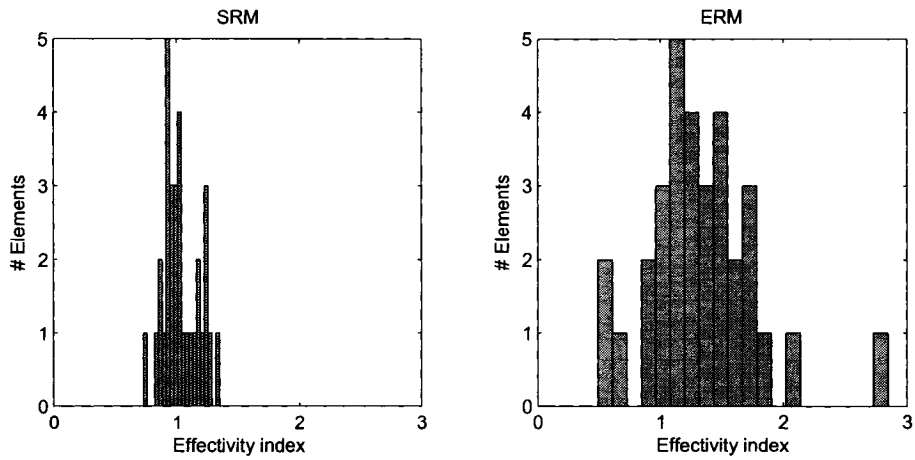


Figure 5.21: Histogram of local effectivity indices for problem P3 on the initial mesh ($RL = 2$)

in the range of 1 to 2. Figure 5.22 shows the histogram of the local effectivity indices for both the SRM and the ERM after two adaptivity iterations. The distribution of the histogram for SRM is almost centered at the optimal unit value while the ERM again produced large effectivity indices for some mesh elements. Figure 5.23 shows the distribution of the local effectivity indices versus the error norm value on a logarithmic scale. The subdomain method is consistent over all error scales and the effectivity index is close to the optimal unit value, while the ERM overestimated the errors in an unpredictable way. Further adaptive mesh refinement improves the

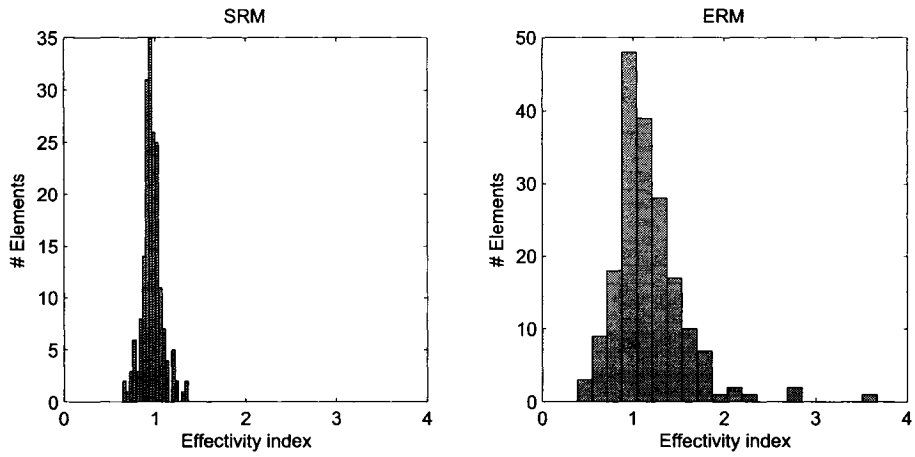


Figure 5.22: Histogram of local effectivity indices for problem P3 after two adaptive mesh refinement iterations ($RL = 2$)

accuracy of both the SRM and ERM but the superior performance of the SRM is still evident as shown in Figures 5.24 and 5.25.

Test problem P4

The fourth test problem is defined over an L-Shaped domain bounded by $(-1,1) \times (-1,1)$ after subtracting the lower right square domain $(0,1) \times (-1,0)$. The exact solution u is defined in terms of the polar coordinates r, θ as following

$$u_{exact} = r^{2/3} \times \sin(2\theta/3) \tag{5.34}$$

Essential boundary conditions satisfying the exact solution are set at the domain boundary and a zero loading function f is prescribed. This solution exhibit a singularity in the solution derivative at the cartesian coordinates $(0,0)$. This singularity is similar to corner stress singularities appearing in elasticity problems. Figure 5.26 shows the initial mesh and the adapted meshes for problem P4 after two and five adaptivity iterations. Table 5.4 lists the effectivity indices of the estimated errors using the SRM and the ERM on different meshes derived by adaptive mesh refine-

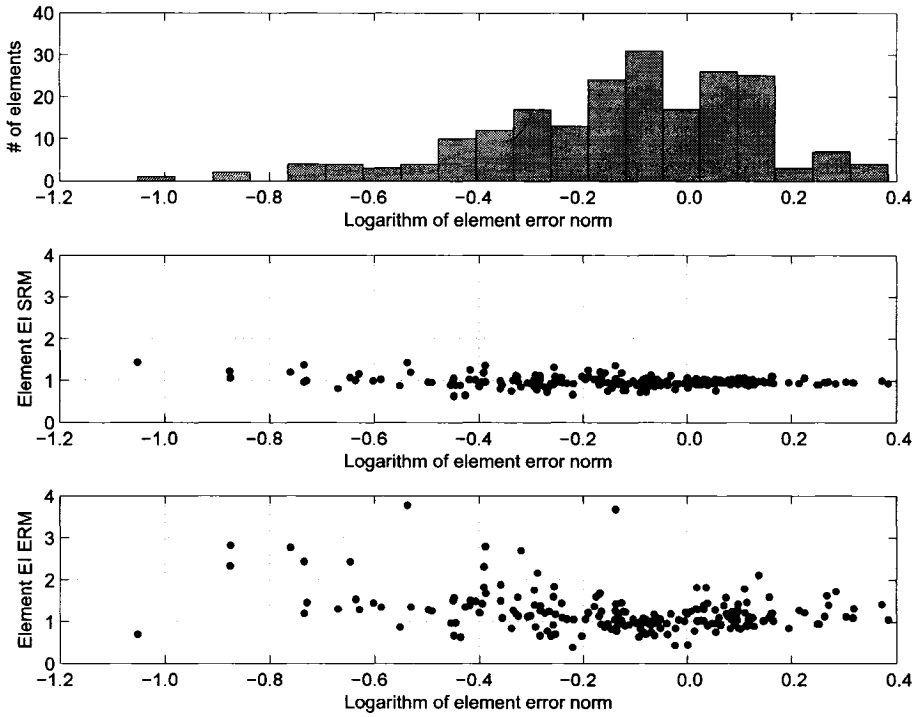


Figure 5.23: Problem P3 - Distribution of local effectivity indices against the exact local error norm. Solution obtained after two adaptive mesh refinement iterations ($RL = 2$)

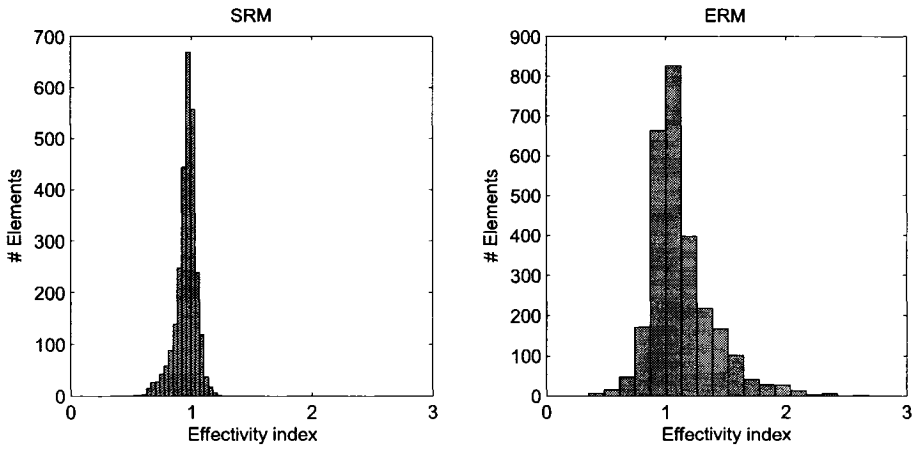


Figure 5.24: Histogram of local effectivity indices for problem P3 after five adaptive mesh refinement iterations ($RL = 2$)

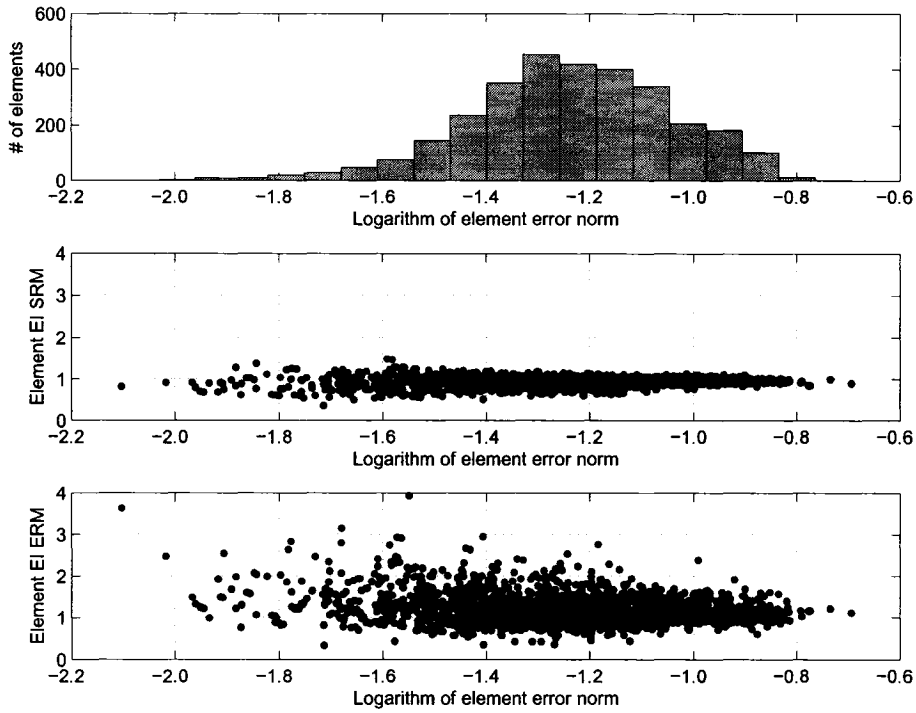
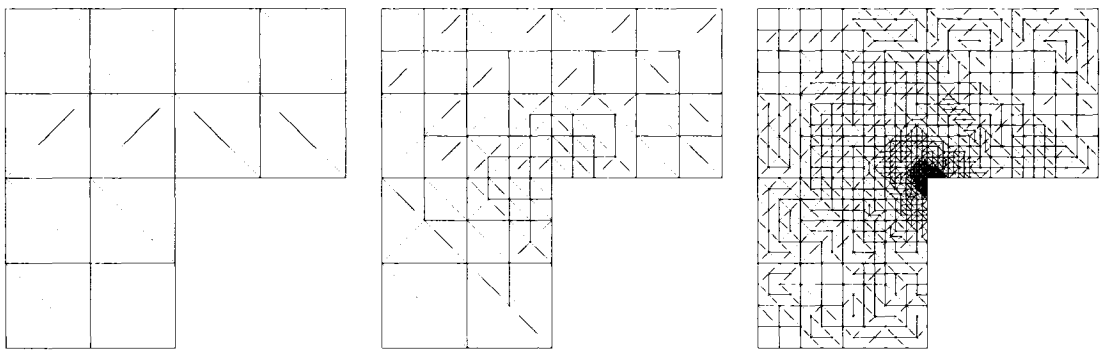


Figure 5.25: Problem P3 - Distribution of local effectivity indices against the exact local error norm. Solution obtained after five adaptive mesh refinement iterations ($RL = 2$)



(a) Initial mesh (b) two adaptive iterations (c) five adaptive iterations

Figure 5.26: Adapted meshes for problem P4

Table 5.4: Problem P4 - Effectivity indices for the estimated errors in terms of the index ρ

# nodes	$RL = 2$				$RL = 3$			
	SRM-up	SRM-lo	ERM-up	ERM-lo	SRM-up	SRM-lo	ERM-up	ERM-lo
21	-16.811	-7.212	6.675	-12.127	-14.405	-2.941	24.122	-14.260
39	-17.056	0.467	8.852	-6.250	-14.842	4.563	26.563	-9.884
74	-15.938	16.187	9.063	7.196	-13.843	20.383	25.983	1.020
159	-15.655	12.392	11.616	2.963	-13.595	16.005	28.273	-3.147
344	-15.084	23.038	13.008	11.098	-13.103	26.742	29.686	3.303
756	-14.498	9.449	13.349	1.123	-12.595	12.890	29.915	-4.464
1595	-13.654	31.104	12.736	19.083	-11.916	35.130	29.048	9.400

ment. The ERM-up estimate overestimates the exact error relative to the SRM-up estimate. The subdomain method tends to underestimate the errors due to the inability of the approximation space for the local patch problems to capture the subgrid energy at the point of singularity. This failure in resolving the singularity results in a consistent underestimated error norm. On the contrary, the ERM overestimated the errors norm even without resolving the high solution gradient close to the point of singularity. This is attributed to the overestimation of the errors in parts of the domain corresponding to small error norms. The upper bound energy norm using the SRM gains accuracy with enlarging the approximation space. It is also observed that lower bound error estimation for this problem is not reliable and in many cases exceeds the value of the upper bound. This undesirable behavior of the lower bound is observed in both the ERM and the SRM.

The histograms of the local effectivity indices for the estimated error after two mesh adaptivity iterations are shown in Figure 5.27. The SRM histogram on such a coarse mesh is bounded between 0.6 and 1.6, while for the ERM the effectivity indices for a few mesh elements exceed 2. This result confirms the pervious analysis of how the ERM-up overestimates the error norm without resolving the subgrid scales. Figure 5.28 shows the distribution of the effectivity index against the logarithm of the

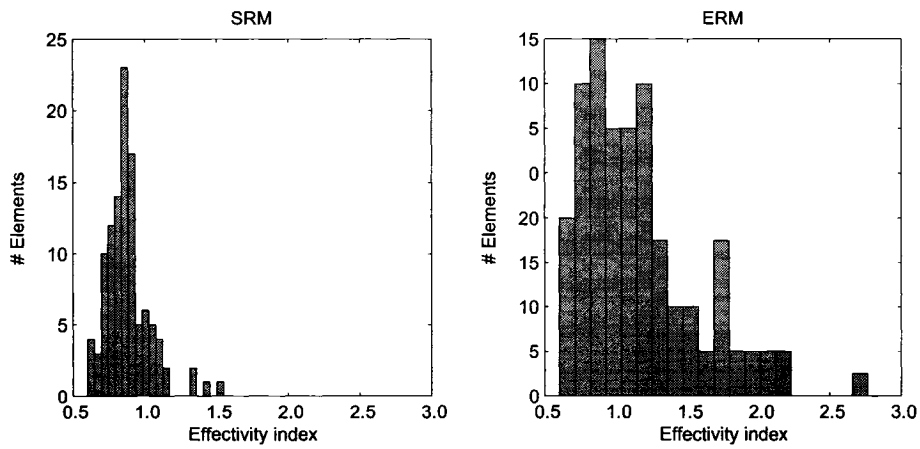


Figure 5.27: Histogram of local effectivity indices for problem P4 after two adaptive mesh refinement iterations ($RL = 2$)

error norm. It is observed that a very few elements, which are close to the point of singularity, have significantly larger (exact) error norms than the rest of the domain. This discrepancy in the distribution of the error norm means that the global error norm is strongly affected by the values of a very few elements close to the point of singularity. For both the SRM and ERM, the effectivity indices for these elements are close to 1. For the remaining mesh elements, the effectivity indices of the estimated errors are more accurately estimated by the SRM than the ERM. Figure 5.29 shows the histogram of the element effectivity indices after eight adaptive mesh refinement iterations. The SRM histogram follows a normal distribution that is bounded for most mesh elements between 0.7 and 1.1, while the ERM histogram is bounded between 0.5 and 2. Figure 5.30 displays the distribution of the element effectivity against the error norm value. The distribution confirms the consistent high quality local error estimation using the SRM in comparison to the ERM.

5.8 Discussion of the results and conclusions

A general framework for a posteriori error estimation based on a numerical realization of the variational multiscale methods is presented. In the proposed framework

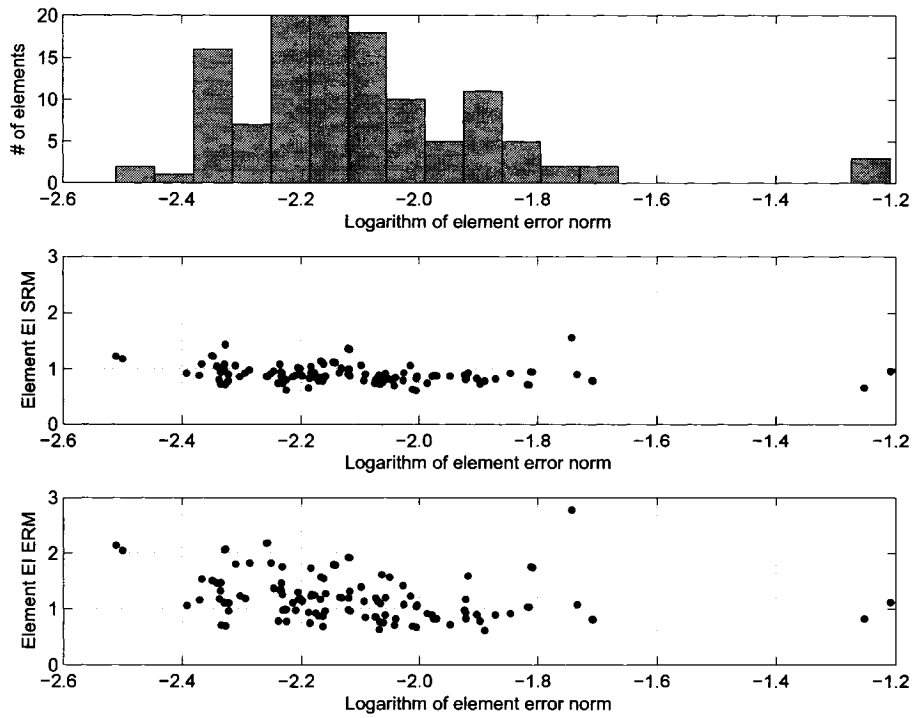


Figure 5.28: Problem P4 - Distribution of local effectivity indices against the exact local error norm. Solution obtained after two adaptive mesh refinement iterations ($RL = 2$)

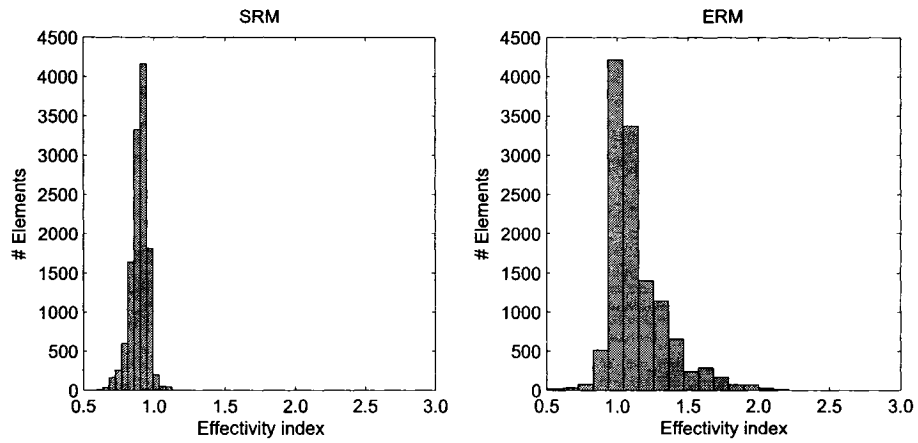


Figure 5.29: Histogram of local effectivity indices for problem P4 after eight adaptive mesh refinement iterations ($RL = 2$)

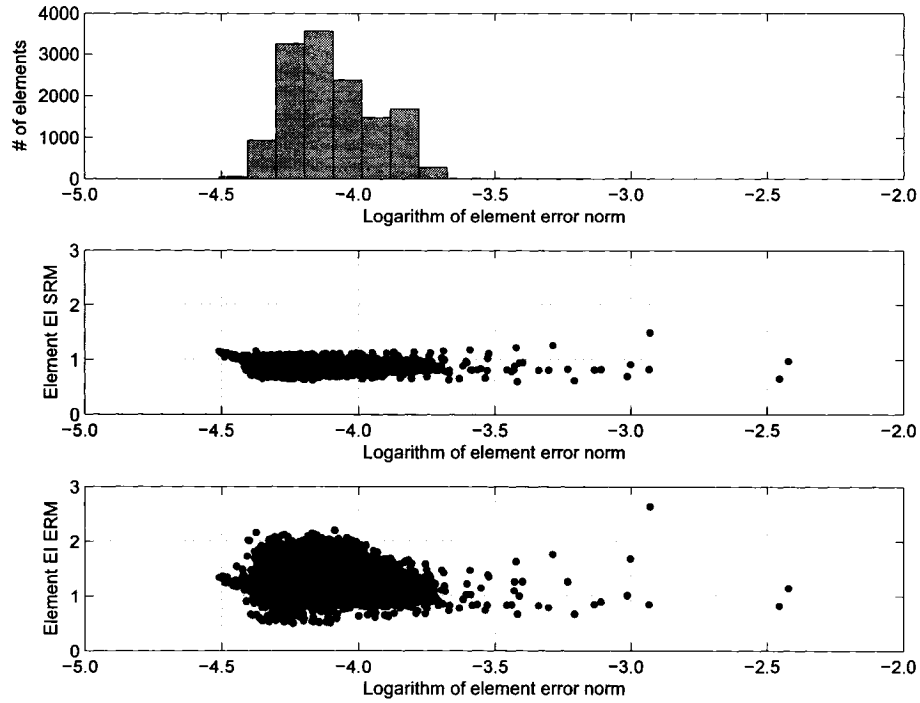


Figure 5.30: Problem P4 - Distribution of local effectivity indices against the exact local error norm. Solution obtained after eight adaptive mesh refinement iterations ($RL = 2$)

different error estimators are derived using different localization functions applied to the fine-scale equation of the VMS. The number of assumptions are kept minimal; for instance, the proposed SRM error estimator formulation has only one assumption related to neglecting the coarse scale error at the mesh nodes. Also, the proposed framework is consistent with the standard element residual method. The specification of the solution space as the fine scale $V_{fh} \subseteq V_f \subset V - \Pi_h V$ is reflected in a weaker upper bound property of the estimator. However, forcing the errors to be zero at the mesh nodes corresponds to balancing the residual with fine scale degrees of freedom only. This may result in a slight overestimation of the estimator over the true exact error. Based on the results of the numerical test problems, the following conclusions regarding the performance of the proposed SRM formulation are drawn:

1. The proposed SRM provides results comparable to the ERM, and in many numerical test problems, is shown to outperform the ERM in terms of the effectivity index both locally and globally.

2. The SRM formulation is flux free, and thus does not need flux equilibration. Thus, it increases the reliability of the method on coarse meshes.

3. In some of the numerical test problems, the upper bound estimate using the ERM is found to overestimate the errors while the SRM slightly underestimates the exact error norm. This overestimation is neither guaranteed nor is attributed to a consistent behavior of the ERM, as has been revealed by studying the local effectivity indices. The ERM loses the upper bound property if the boundary fluxes are not equilibrated. Numerical test results have shown that the ERM might significantly overestimate the errors over some mesh elements in an unpredictable manner, which results in an increased value of the global error norm.

4. The SRM provides an effective way to calculate a sharp lower bound estimate of the error norm without the need to solve an auxiliary set of local problems. The continuous error field is shown to accurately represent the error field, if the local problems are solved using adequate discretization.

5. The proposed error estimator is easy to implement without the need to store a discontinuous error field as in the case of the SRM formulation presented in [23]. Moreover, the new estimator does not need to implement special constraints on the solution space of the local problems, as in [11, 12, 22].

6. The presented SRM has a consistent effectivity over the entire problem domain that is independent of the scale of the error. This significantly improve the quality of the adapted meshes.

The proposed framework can be easily extended to other problems, as it depends on only two elements: the variational multiscale method, which is quite general, and on a localization function of choice. This study has also shown that a partition of unity is a good choice of the localization function for first order Lagrangian finite element method.

ACKNOWLEDGEMENTS

This research was partially funded through grants from the Natural Science and Engineering Research Council of Canada (NSERC) and the McMaster University's Centre for Effective Design of Structures.

Bibliography

[1] R. Verfürth, *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques.* Advances in Numerical Mathematics, Wiley-Teubner, 1996.

[2] I. Babuska and T. Strouboulis, *The Finite Element Method and Its Reliability.* Oxford University Press, 2001.

[3] M. Ainsworth and J. T. Oden, *A Posteriori Error Estimation in Finite Element Analysis.* John Wiley & Sons, 2000.

[4] I. Babuska and W. C. Rheinboldt, "A-posteriori error estimates for the finite element method," *Internat. J. Numer. Methods Engrg.*, vol. 12, pp. 1597–1615, 1978.

[5] K. Eriksson and C. Johnson, "An adaptive finite element method for linear elliptic problems," *Math. Comp.*, vol. 50, no. 182, pp. 361–383, 1988.

[6] R. Becker and R. Rannacher, "An optimal control approach to a posteriori error estimation in finite element methods," *Acta Numerica*, vol. 10, pp. 1–102, 2001.

[7] R. Becker and R. Rannacher, "A feed-back approach to error control in finite element methods: Basic analysis and examples.," *East-West J. Numer. Math.*, vol. 4, pp. 237–264, 1996.

[8] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson, "Introduction to adaptive methods for differential equations," *Acta Numerica*, pp. 105–158, 1995.

[9] R. E. Bank and A. Weiser, "Some a posteriori error estimators for elliptic partial differential equations," *Math. Comput.*, vol. 44, no. 170, pp. 283–301, 1985.

[10] M. Ainsworth and J. T. Oden, "A unified approach to a posteriori error estimation using element residual methods," *Numer. Math.*, vol. 65, no. 1, pp. 23–50, 1993.

[11] P. Morin, R. H. Nochetto, and K. G. Siebert, "Local problems on stars: a posteriori error estimators, convergence, and performance," *Math. Comp.*, vol. 72, no. 243, pp. 1067–1097, 2003.

[12] S. Prudhomme, F. Nobile, L. Chamoin, and J. Oden, "Analysis of a subdomain-based error estimator for finite element approximations of elliptic problems," *Numer Methods Partial Differential Equations*, vol. 20, no. 2, pp. 165–192, 2004.

[13] T. J. R. Hughes, "Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods," *Comput. Methods Appl. Mech. Engrg.*, vol. 127, no. 1-4, pp. 387–401, 1995.

[14] S. C. Brenner and L. R. Scott, *The mathematical theory of finite element methods*, vol. 15 of *Texts in Applied Mathematics*. New York: Springer-Verlag, second ed., 2002.

[15] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*, vol. 40 of *Classics in Applied Mathematics*. Philadelphia: SIAM, 2002.

[16] T. J. R. Hughes, G. R. Feijóo, L. Mazzei, and J.-B. Quincy, "The variational multiscale method—a paradigm for computational mechanics," *Comput. Methods Appl. Mech. Engrg.*, vol. 166, no. 1-2, pp. 3–24, 1998.

[17] P. Díez, N. Parés, and A. Huerta, "Recovering lower bounds of the error by postprocessing implicit residual a posteriori error estimates," *Internat. J. Numer. Methods Engrg.*, vol. 56, no. 10, pp. 1465–1488, 2003.

[18] P. Ladevèze and D. Leguillon, "Error estimate procedure in the finite element method and applications," *SIAM J. Numer. Anal.*, vol. 20, no. 3, pp. 485–509, 1983.

[19] S. Prudhomme, J. T. Oden, T. Westermann, J. Bass, and M. E. Botkin, "Practical methods for a posteriori error estimation in engineering applications," *Internat. J. Numer. Methods Engrg.*, vol. 56, no. 8, pp. 1193–1224, 2003.

[20] I. Babuška and W. C. Rheinboldt, "Error estimates for adaptive finite element computations," *SIAM J. Numer. Anal.*, vol. 15, no. 4, pp. 736–754, 1978.

[21] M. Larson and A. Malqvist, "Adaptive variational multiscale methods based on a posteriori error estimates: Energy estimates for elliptic problems," *Comput. Methods Appl. Mech. Engrg.*, vol. preprint.

[22] C. Carstensen and S. A. Funken, "Fully reliable localized error control in the FEM," *SIAM J. Sci. Comput.*, vol. 21, no. 4, pp. 1465–1484, 1999/00.

[23] N. Parés, P. Díez, and A. Huerta, "Subdomain-based flux-free a posteriori error estimators," *Comput. Methods Appl. Mech. Engrg.*, vol. 195, no. 4-6, pp. 297–323, 2006.

[24] H.-W. Choi and M. Paraschivoiu, "Adaptive computations of a posteriori finite element output bounds: a comparison of the hybrid-flux approach and the flux-free approach," *Comput. Methods Appl. Mech. Engrg.*, vol. 193, no. 36-38, pp. 4001–4033, 2004.

Appendix A

This appendix includes the paper titled:

"Semi-formal design of reliable mesh generation systems,"

Published in the Journal, Advances in Engineering Software, volume 35, issue 12, pages 827-841, 2004.

ELSEVIER

Advances in Engineering Software 35 (2004) 827–841

ADVANCES IN ENGINEERING SOFTWARE

www.elsevier.com/locate/advengsoft

Semi-formal design of reliable mesh generation systems

A.H. ElSheikh[a], S. Smith[b], S.E. Chidiac[c,*]

[a]*Department of Civil Engineering, McMaster University, Hamilton, Ont., Canada, L8S 4L7*
[b]*Department of Computing and Software, McMaster University, Hamilton, Ont., Canada, L8S 4L7*
[c]*Department of Civil Engineering, McMaster University, Hamilton, Ont., Canada, L8S 4L7*

Received 28 October 2003; revised 20 May 2004; accepted 10 June 2004

Available online 17 September 2004

Abstract

A reliable mesh generation infrastructure is designed based on software engineering principles. Formal methods, software design documents and clear modular decomposition criteria are introduced to improve the quality of mesh generation software. The design document for a simple 2D mesh generation data structure is presented using a semi-formal specification. The proposed semi-formal documentation system avoids any ambiguity during the software design process and will help in driving the software test cases. Using the proposed software, design techniques result in a consistent software design that is easy to extend and modify.

Keywords: Mesh generation; Software engineering; Modular decomposition; Formal methods; Software quality

1. Introduction

Mesh generation is an essential component in many numerical methods used for physical simulation. The accuracy of the finite element and the finite volume methods heavily depend on the mesh to be used for the discretization process. The requirements of adaptive numerical methods where mesh modification is needed to increase the accuracy of the solution increases the design complexity of the mesh generation toolboxes. Attempts have been made to improve the design of mesh generators [3,16]. These attempts have identified many of the mesh generation software requirements [3]. One of the major drawbacks of these attempts is a high dependency on a specific implementation language, which was C++ in both cases. In the current practice object oriented methods are usually confused with software engineering principles. It should be clarified that object oriented languages facilitate and encourage many software engineering principles such as data abstraction, information hiding, encapsulation, module generalization and template implementation, but all these concept can be implemented by any well-designed imperative language in combination with disciplined programming practices.

Instead of software specification many programs substitue informal descriptions and comments throughout the program code. Visual specification languages like UML [10] can be used effectively for a pictorial representation of architectural concepts, but these cannot be used to specify mathematical operations or pre- and post-conditions and they lack a mathematically rigorous semantics [6]. This informal way of designing and specifying software poses hardships on all the stages of the software development process that follow. The ability to verify and validate the correctness of the system is missing because of the absence of a reference that specifies the correct software behavior. As a consequence of the above point, software reuse, maintainability and extendability are extremely difficult within the current mesh software development practices.

Recent work [5] suggests that software engineering principles can help with these problems. Whereas Ref. [5] takes a breadth approach and considers several stages of the software cycle, the current work will take a more specific perspective by incorporating three major ideas to improve the quality of mesh generation software. These ideas are formal methods, software design documents and a clear modular decomposition criteria for mesh generation software systems. Formal methods are collections of

* Corresponding author.

0965-9978/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2004.06.012

mathematical notations and techniques for describing and analyzing systems [14]. This paper will embrace formal methods as a particular method for increasing software quality and will focus on the process of describing software systems with formal methods. The process of analyzing the software description can be done through the verification process, which can be done deductively, or by testing. Some tools like PVS can be used in the verification [17], but they are hard to use and limited to simple data structures.

Software design documents are a set of separate documents targeting different stages of the software design process. In many cases, these documents are ambiguous or not complete. Specification documents are important for communicating ideas between different parts of the software development team. In this paper, we suggest using a semi-formal language for documenting mesh generation software design so that we can be as specific as necessary.

The ultimate goal of any mesh generation software is to be correct and this correctness is based on analyzing the relation between the computer program p and the specification s. This relation can appear in three different classes of problems. If we are given the specification s and a program p that satisfies these specification needed, then we are dealing with a design problem. If both the specification s and program p are given we can check whether p satisfies s and hence we are dealing with a validation problem. The third case is when we have a program p and we want to extract the specification s of this program. In this case, we are dealing with a reverse engineering problem. The previous three problems may initially look different, but they have many overlapping issues, such as the syntax and semantics to be used and the underlying mathematics of the specification.

The last idea suggested for increasing the quality of the meshing software is the use of a clear modular decomposition criteria. Modular decomposition is the process of dividing a big job into a set of jobs which are small, easy to understand and as independent as possible. The decomposition process may be based on different goals such as, design generality, simplicity, efficiency or the flexibility for certain changes. Identifying the criteria for decomposition rules will result in software code that is consistent with the targeted design.

This paper starts with defining the notation used to specify software components semi-formally. A discussion of the theoretical bases of modelling software systems as state machine is also presented. A simple way for specifying the Module State Machines (MSMs) by both defining the Module Interface Specification (MIS) and the semantics of the transition functions is outlined. The basic rules of modular structure design of software system will be discussed. Finally, a sample design specification document of a 2D unstructured mesh generation data structure followed by the specifications of the Delaunay insertion algorithm is presented. This algorithm will show how to apply the formal methods to this class of problem.

2. Notations

The semi-formal language used throughout this paper is based on simple set notations and first-order logic. This language has atomic types *int*, *bool*, *char*, *string* and *real*. These atomic types can be used in tuples or collections. The syntax used for tuples is (*Type*1, *Type*2,...,*TypeN*) with a semantics of N elements of types *TypeI* where $I = 1, 2, \ldots, N$. Internal fields in the tuple can be referenced using the dot notation. For example, if *TB* is a tuple (var_1: *Type*1, var_2: *Type*2) then the first field can be referenced as $TB.var_1$. Collections of elements are stored in containers, which may be ordered or un-ordered collections with unlimited size. For unordered collections without duplicate elements, the syntax $(Type1)_{set}$ is used for describing a set of *Type*1 which has no limit on the size, in the same sense as an abstract datatype; that is, a set is a mathematical notion independent of any concrete implementation.

One way for defining sets is by constructors that select all the elements of some type that satisfy a given predicate. For example, $S = \{x: int | ODD(x)\}$ is a predicate specification for set S of all odd integers [15]. Ordered collections are described by sequences with $(Type2)_{seq}$ as a syntax for sequences. Sequences are unlimited in size in the same sense as an abstract datatype. Sequences are indexed using conventional array notation: $s = \langle s[0], s[1], \ldots, s[n-1] \rangle$. Adding elements to sequences is done be using the appending symbol ||. Concatenation of two sequences is done using the same symbol. The concatenation can be done from the head or the tail of the sequence only. To specify the size of a sequence, or set, a norm notation is used; for example, $|s|$ is used to show the size of the collection s.

Simple prepositional logic operators will be used throughout our specification language. Propositional variables with binary value of TRUE or FALSE are used, along with simple formula including the boolean operators $\wedge$, $\vee$ and $\neg$. First order quantifiers like $\forall$ and $\exists$ will be used as prefix for formulas especially when dealing with sets and sequences. A comprehensive introduction to prepositional logic can be found in Ref. [8].

3. Modelling of meshing software

Modelling is the process of abstraction of the system while preserving a limited number of original details. In this process, the main properties of the system are highlighted to allow better management of complex systems. Modelling software system relies on the concept of state. The state of software can be abstracted into a set of state variables. The size of this set depends on the level of refinement of the model. These state variables capture information about certain steps in the executions path of the software. This information may be the size and content of some data structure or may be a flag for some condition. The set of

state variables can be called initial, intermediate or final state depending on the point of program execution. The relation between the initial state and the final state is of great importance because it can be used in defining both pre-conditions and post-conditions, which are widely used in the verification process.

A software system is composed of smaller pieces of software called Modules. A module is a self-contained work assignment for a programmer or programming team [12]. A module can be modelled mathematically as a state machine. A simple form of the formalism for this model as a state machine is a tuple (S, s_0, I, O, T, E) [7] where S is the set of states and s_0 is the initial state and $s_o \in S$, I is a set of inputs, O is a set of outputs and T is the transition function T: $S*I \rightarrow S$ and E is the output function (E: $S*I \rightarrow O$). The domain of both T and E are $S*I$ where the * denotes the Cartesian product. This way of description as MSMs [7] can provide an easy mathematical basis for specifying software modules. Comprehension of the state machine in relational form may be tedious and time consuming. However, a simple method for a complete description of the MSM can be done by listing the state variables and specifying the interface of access functions which change the state variables and produce outputs. The state variables definition is done by listing the name and type of each state variable. Access function are defined by listing the name of the function and types of the input and return values of the function. A mathematical description of the semantics of each function also needs to be given. This method of specifying modules is referred to as a MIS.

4. The modular structure

The first step of designing any software system is to decompose the software into a set of simpler problems through what is called the modular decomposition process. The five goals of modular decomposition as highlighted by Parnas [12] are:

(1) Each module should have a simple structure that can be understood by any programmer who is not a member of the development team.
(2) Each module should be self-contained and the coupling between modules should be minimized. This allows changing the implementation of one module without complete knowledge of other modules and without affecting the behavior of other modules.
(3) The module interface should be flexible so that it can accommodate internal changes of the module without any external changes. Interface changes are avoided because this would export the effect of internal module changes into other modules.
(4) Ideally major changes in the software should be done as a set of independent changes to individual modules.
(5) Understanding the functionality of each module should be possible without knowing the internal details of the module design.

The adopted module decomposition criteria are based on the principles of information hiding, design for change and stepwise refinement. According to information hiding principle, details that are likely to change should be the secrets of separate modules [13].

These ideas of Modular Decomposition can be applied easily to the data structures used in the mesh generator. Any data structure which is expected to change under any circumstances should be hidden inside one module. The access to the data inside this module is done through the set of access functions of this module. This is done to reduce the ripple effects when modifying or extending the program. Drawbacks of extensive use of modularization are the reduction of the efficiency of the whole software system and an increase in the development time. The efficiency problem can be reduced with inline access functions, which are allowed by most modern compilers.

Flexible interfaces may be a challenge in the implementation phase, but generic programming through function pointers and templates offers a solution to achieve the needed flexibility. It should be noted that the level of assumed generality should not be applied to every data structure used in the program, based on the trade-off between generality and efficiency. Certain assumptions should be made on some major data structures and a software design decision can be assumed that this data structure will not change. If such decisions are made, a detailed description of the reasons behind it should be appended to the software design documentation.

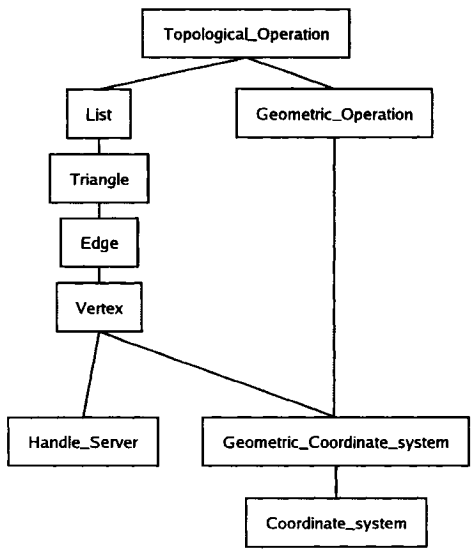


Fig. 1. A hierarchy for the designed meshing data structure.

5. A simple 2D triangular mesh data structure

The purpose of the section is not to design a complete 2D triangular mesh generator, but to demonstrate how the semi-formal specification methodology outlined can be applied to mesh generation software. The simplest modular decomposition can be found by assigning a module to each of the geometrical entities of vertex, edge and triangle. After defining these basic entities, a module for storage of these basic elements should be defined. A software design decision should be made on whether to use the same container structure for the three elements or not. After defining the basic sets of data structures, the algorithms applied on these data structures should be analyzed and divided into modules. Simple geometric operations can be contained in one module. The higher-level algorithms, which are the core of the mesh generation algorithm, should be localized in a set of independent modules because of the possibility of changing the algorithms. It should be noted that modular decomposition is not an easy job to be done in one step, instead a series of steps using stepwise refinement is applied. The previous decomposition can be represented by a uses hierarchy. We say that a module A uses a module B if correct execution of B may be necessary for A to complete its work [11]. Fig. 1 shows a uses hierarchy for the designed mesh generation software. The level of the graph shows the dependency where modules at the bottom use no other modules and considered to be at level 0. Modules at level i are the set of modules which use at least one module of level $i-1$ and do not use any module at level higher than $i-1$.

The goals of our 2D mesh design can be summarized as following:

(1) Having a separate and flexible representation for each mesh entity. For instance, the representation of the vertex, edge or triangle that can be easily modified or extended to accommodate different mesh generation algorithm requirements.
(2) Having a complete separation between the geometry or physical data on the mesh and the topology or

Used External Functions : NONE
Used External Data Types : NONE
External Constants : NONE
Type Definitions : Handle = int
: Handle_server = $(Handle)_{set}$
Exported Constants : MAX_SIZE: int
Exported Functions

Exported Function	Input Type	Output Type	Exception
HS_init	Handle_server	Handle_server	
HS_getHandle	Handle_server	Handle	Server_Is_Full
HS_addHandle	Handle_server, Handle	Handle_server	Server_Is_Full Handle_Exists
HS_delHandle	Handle_server, Handle	Handle_server	Handle_Not_Exist

State Variables: NONE
Function semantics:

Handle_server HS_init (s: Handle_server)
Output: s={}

Handle HS_getHandle (s: Handle_server)
Exception: $|s| \geq$ MAX_SIZE $\Rightarrow$ Server_Is_Full
Output: h: Handle where $h \notin s$

Handle_server HS_addHandle (s: Handle_server, h: Handle)
Exception: $|s| \geq$ MAX_SIZE $\Rightarrow$ Server_Is_Full
$h \in s \Rightarrow$ Handle_Exists
Output: $s \cup \{h\}$

Handle_server HS_delHandle (s: Handle_server, h: Handle)
Exception: $h \notin s \Rightarrow$ Handle_Not_Exist
Output: $s = s - \{h\}$

Fig. 2. Specifications of the Handle Server Module.

connectivity information of the mesh. This is done to ease the extension of the 2D mesh generator into surface meshing.

(3) The mesh generator should be able to work with different coordinate systems.
(4) A flexible data structure to store sets of vertices, edges and triangles, which can be changed based on the meshing algorithm requirements.
(5) The mesh generation can be done by different mesh generation algorithms available in the literature with a minimal amount of local changes.

The first step in our design is to define new datatypes. For example, one new datatype is introduced because of the need for each entity like the vertices, edges and triangles to have a global index or Handle. Manipulating the handle information through adding and deleting elements is not simple because of the dynamic nature of unstructured mesh generation, which allows both refinement and coarsening. Adding and removing entities during mesh generation makes the use of simple indexing infeasible. To hide the information of how to deal with indexing a Handle Server Module is defined to provide us with unique index for each of the vertices, edges and triangles. The access function of this module have a variable of type Handle server within its input parameter to provide the needed flexibility of the module to deal with three different handle servers, one for the vertices and one for edges and one for triangles. Fig. 2 shows the MIS of the Handle Server Module.

Used External Functions : NONE
Used External Data Types : NONE
External Constants : NONE
Type Definitions : CS_type = set of {2DC,2DP}
: Coord_sys = tuple of (size: int, sym: CS_type)
Exported Constants : NONE
Exported Functions :

Exported Function	Input Type	Output Type	Exception
CS_exists	CS_type	bool	
CS_getsize	CS_type	int	Coordsys_Not_Exist
CS_getcoordsys	CS_type	Coord_sys	Coordsys_Not_Exist
CS_getcoordtype	Coord_sys	CS_type	Coordsys_Not_Exist

State Variables: Coord_tbl : (Coord_sys) set:= {(2,2DC),(2,2DP)}
Where 2DC stands for Cartesian coordinate system (x, y) and 2DP stands for polar coordinate system (r, θ).
Function semantics:

bool CS_exists (c: CS_type)
Output: (∃ c: CS_type | (i,c) ∈ Coord_tbl)

int CS_getsize (c: CS_type)
Exception: ¬ (∃ c: CS_type | (i,c) ∈ Coord_tbl)⇒ Coordsys_Not_Exist
Output: i: int where (i,c) ∈ Coord_tbl

Coord_sys CS_getcoordsys (c: CS_type)
Exception: ¬ (∃ c: CS_type | (i,c) ∈ Coord_tbl)⇒ Coordsys_Not_Exist
Output: (i,c): Coord_sys where (i,c) ∈ Coord_tbl

CS_type CS_getcoordtype (cs: Coord_sys)
Exception: ¬ (cs ∈ Coord_tbl) ⇒ Coordsys_Not_Exist
Output: c: CS_type where (i: int, c) = cs

Fig. 3. Specifications of the Coordinate System Module.

For vertices, the handle should be combined with the geometrical data in a tuple to completely define the topology and physical information. The physical information in simple applications is limited to the geometrical data, which can be represented in many different ways. For example, the coordinate system can be Cartesian or polar. To hide the information of the coordinate system we used a Coordinate System Module as shown in Fig. 3. This module is pre-initialized with two coordinate systems, namely 2D Cartesian and 2D Polar system. This module is initialized at compilation time because of the need to define some functions to manipulate each coordinate system. The second layer of defining the geometric data is hidden in the Geometric Coordinate System Module as shown in Fig. 4. This module has the ability to manipulate information based on the specified coordinate system. An extension to this module is made by adding a set of functions for geometrical operations. Due to the large number of these geometrical operations, a separate module is defined for that purpose in Fig. 5.

Combining the handle and coordinate information for vertices is done in the Vertex Module as shown in Fig. 6. The edges can be represented explicitly as an element connecting two vertices or it can be done implicitly as the element separating two triangles. In our case, a two vertex representation is assumed because we want to keep the interface as intuitive as possible. Fig. 7 shows the MIS of the Edge Module. The triangle elements can also be represented in two ways: as three edges or by defining three vertices. It is worth mentioning that the topology or connectivity data is completely independent of whether the mesh is embedded in 2D or in 3D space as a surface mesh. Fig. 8 presents the MIS of the Triangle Module and its access functions.

Used External Functions : CS_type = set of {2DC,2DP}
: Coord_sys = tuple of (size: int, sym: CS_type)
Used External Data Types : NONE
External Constants : NONE
Type Definitions : Geometry = (real) seq
Coordinate = tuple of (cs: Coord_sys, g: Geometry)
Exported Constants : NONE
Exported Functions :

Exported Function	Input Type	Output Type	Exception
GE_getCoord	Coord_sys, Geometry	Coordinate	Coord_Not_Consitent
GE_setCoord	Coordinate, Geometry	Coordinate	Coord_Not_Consitent
GE_getgeom	Coordinate	Geometry	
GE_getCoordsys	Coordinate	Coord_sys	

State Variables: NONE
Function semantics:

Coordinate GE_getCoord (cs: Coord_sys, g: Geometry)
Exception: ¬ (cs.size = | g |) ⇒ Coord_Not_Consistent
Output: (cs,g)

Coordinate GE_setCoord (cd: Coordinate, g: Geometry)
Exception: ¬ (cd.cs.size = | g |) ⇒ Coord_Not_Consistent
Output: (cd.cs, g)

Coord_sys GE_getCoordsys (cd: Coordinate)
Output: cd.cs

Geometry GE_getgeom (cd: Coordinate)
Output: cd.g

Fig. 4. Specifications of the Geometric Coordinate System Module.

Used External Functions : NONE
Used External Data Types : Coord_sys = (size: int, sym: CS_type)
: Geometry = (real) seq
: Coordinate = tuple of (cs: coord_sys, g: Geometry)
External Constants : NONE
Type Definitions : NONE
Exported Constants : NONE
Exported Functions :

Exported Function	Input Type	Output Type	Exception
GO_Compute_dist	Coordinate,Coordinate	real	Coord_Not_Consistent
GO_IS_Linear	Coordinate,Coordinate, Coordinate	bool	Coord_Not_Consistent
GO_IN_Circle	Coordinate,Coordinate, Coordinate,Coordinate	bool	
GO_IN_Triangle	Coordinate,Coordinate, Coordinate,Coordinate	bool	
GO_Check_orientation	Coordinate,Coordinate, Coordinate	bool	
GO_Check_EdgeEncroach	Coordinate,Coordinate, Coordinate	bool	
GO_Check_TriEncroach	Coordinate,Coordinate, Coordinate,Coordinate	bool	
GO_Get_EdgeMid	Coordinate,Coordinate	Coordinate	
GO_GetCircumcenter	Coordinate,Coordinate, Coordinate	Coordinate	

State Variables: NONE
Selected function semantics:

real GO_Compute_dist (c1: Coordinate, c2: Coordinate)
Exception: ¬ (GE_getCoordsys(c1)= GE_getCoordsys (c2)) ⇒ Coord_Not_Consistent
Output: If CS_getcoordtype(GE_getCoordsys(c1))=2DC GE_Compte_dist2DC(c1.g,c2.g)
If CS_getcoordtype(GE_getCoordsys(c1))=2DP GE_Compte_dist2DP(c1.g,c2.g)

Internal function semantics:
real GO_Compute_dist2DC (g1: Geometry, g2: Geometry)
Output: Sqrt((g1[0]-g2[0])^2+(g1[1]-g2[1])^2)

real GO_Compute_dist2DP (g1: Geometry, g2: Geometry)
Output: Sqrt(g1[0]^2+ g2[0]^2-2*g1[0]* g2[0]* cos(g2[1]-g1[1]))

Fig. 5. Excerpts from specifications of the Geometric Operation Module.

The next step is to define the container specifications of each entity. A generic container specification is shown in Fig. 9. A type variable, which may be a vertex, edge or triangle is used in this specification. A specialization of this list or container is done to have the VertexList and the EdgeList and the TriangleList. Finally, a set of some topological operation commonly used by unstructured mesh generation algorithms are bundled in the Topological Operation Module shown in Fig. 10.

6. Specifications of mesh generation algorithms

Mesh generation algorithms can be specified using the developed infrastructure. Mesh generators usually needs two types of relation between mesh entities. These relations can be divided into incidence and adjacency relations. Betri [2] formalized the definition of the *incidence* relation into the relation of a subset. If a mesh entity f is inside of another entity c then f and c

Used External Functions : NONE
Used External Data Types : Coordinate = tuple of (cs: coord_sys, g: Geometry)
: Handle = int
External Constants : NONE
Type Definitions : Vertex = tuple of (hd: Handle, cd: Coordinate)
Exported Constants : NONE
Exported Functions

Exported Function	Input Type	Output Type	Exception
V_createVertex	Handle, Coordinate	Vertex	
V_getHandle	Vertex	Handle	
V_getCoord	Vertex	Coordinate	
V_Compare	Vertex, Vertex	bool	
V_setVertCoord	Vertex, Coordinate	Vertex	

State Variables: NONE
Function semantics:

Vertex V_createVertex (h: Handle, cd: coordinate)
Output: (h,cd)

Handle V_getHandle (v: Vertex)
Output: v.hd

Coordinate V_getCoord (v: Vertex)
Output: v.cd

bool V_Compare (v1: Vertex, v2:Vertex)
Output: v1.hd = v2.hd

Vertex V_setVertCoord (v: Vertex, cd: coordinate)
Output: (v.hd,cd)

Fig. 6. Specifications of the Vertex Module.

are incident. For example, there is an incidence relation between the start vertex of an edge and the edge itself. It is clear that elements of the same topological dimension are never incident, but they may have another type of relation called *adjacency* relation. For example, we can define that two edges are *adjacent* if they share the same vertices. The incident relations are specified in the mesh infrastructure sections, where a downward incidence relation from elements with higher topological dimensions are connected to elements with only 1D less in the topological sense. Thus, triangles are defined in terms of edge and edges are defined in terms of vertices. On the other hand, the adjacency relation was identified as being algorithm dependant. For example, Oct-tree based mesh generators rely on parent/child adjacency relations between entities of the same topological dimension, while in Delaunay triangulation each triangle needs to know the neighboring triangles through the neighbor adjacency relation. Due to this dependency of the adjacency relations on the mesh generation algorithm, these relations are not defined in the mesh infrastructure.

As an example of using the suggested semi-formal documentation and specification style, a key operation of a Delaunay mesh generation algorithm is specified. Delaunay triangulation is one of the most common algorithms for triangular mesh generation. These algorithms are usually done incrementally, where an initial large triangle that geometrically bounds all the domain is defined. Following this, vertices along the boundaries are inserted incrementally. Once all the boundary vertices are inserted, boundary edges are recovered. The recovery is

Used External Functions : NONE
Used External Data Types : Vertex = tuple of (hd: Handle, cd: Coordinate)
: Handle = int
External Constants : : NONE
Type Definitions: : Edge : tuple of (hd: Handle, v0: Vertex, v1: Vertex)
Exported Constants: NONE
Exported Functions:

Exported Function	Input Type	Output Type	Exception
E_createEdge	Handle, Vertex, Vertex	Edge	Edge_Not_Valid
E_getHandle	Edge	Handle	
E_Compare	Edge, Edge	bool	
E_getStart	Edge	Vertex	
E_getEnd	Edge	Vertex	
E_setEdge	Edge, Vertex, Vertex	Edge	Edge_Not_Valid

State Variables: NONE
Function semantics:

Edge E_createEdge (h: Handle, v0: Vertex, v1: Vertex)
Exception: V_Compare(v0,v1) ⇒ Edge_Not_Valid
Output: (h,v0,v1)

Handle E_getHandle (e: Edge)
Output: e.hd

bool E_Compare (e1: Edge, e2: Edge)
Output: e1.hd = e2.hd

Vertex E_getStart (e: Edge)
Output: e.v0

Vertex E_getEnd (e: Edge)
Output: e.v1

Edge E_setEdge (e: Edge, v0: Vertex, v1: Vertex)
Exception: V_Compare(v0,v1)⇒ Edge_Not_Valid
Output: (e.hd,v0,v1)

Fig. 7. Specifications of the Edge Module.

also done by inserting vertices along the missing boundaries. Finally a mesh improvement by refinement is done for all the triangles that do not meet a certain quality measure. A new vertex is inserted at the circumcenter of each triangle that fails the geometrical quality predicate. A complete description of the Delaunay refinement algorithms can be found in Ref. [18]. It is clear from the previous description that vertex insertion is the core step of this algorithm. This insertion should maintain the validity of the Delaunay empty circumcenter property of every triangle in the mesh. Fig. 11 introduces the specifications of neighbor adjacency relation of the mesh edges. This relation is needed for the Delaunay refinement algorithms to identify adjacent triangles. Additional adjacency relations can be defined in the implementation process, but if any redundancy in the stored information is introduced, validity checks should also be added to avoid any inconsistency. A pictorial representation of the Bower/Watson point insertion algorithm [4,19] is shown in Fig. 12. In this

Used External Functions : NONE
Used External Data Types : Handle = int
: Edge = tuple of (hd: Handle, v0: Vertex, v1: Vertex)
External Constants : : NONE
Type Definitions: : Triangle = tuple of (hd: Handle, e0: Edge, e1: Edge, e2: Edge)
Exported Constants: NONE
Exported Functions:

Exported Function	Input Type	Output Type	Exception
T_createTri	Handle, Edge, Edge, Edge	Triangle	Triangle_Not_Valid
T_getHandle	Triangle	Handle	
T_Compare	Triangle, Triangle	bool	
T_getE0	Triangle	Edge	
T_getE1	Triangle	Edge	
T_getE2	Triangle	Edge	
T_isEdge	Triangle, Edge	bool	
T_setTriangle	Triangle, Edge, Edge, Edge	Triangle	Triangle_Not_Valid

State Variables: NONE
Function semantics:

Triangle T_createTri (h: Handle, e0: Edge, e1: Edge, e2:Edge)
Exception: E_Compare(e0,e1) ∨ E_Compare(e1,e2) ∨ E_Compare(e2,e0) ⇒ Triangle_Not_Valid
Output: (h,e0,e1,e2)

Handle T_getHandle (t: Triangle)
Output: t.hd

bool T_Compare (t1: Triangle, t2: Triangle)
Output: t1.hd = t2.hd

Edge T_getE0 (t: Triangle)
Output: t.e0

Edge T_getE1 (t: Triangle)
Output: t.e1

Edge T_getE2 (t: Triangle)
Output: t.e2

bool T_isEdge (t: Triangle, e: Edge)
Output: E_Compare(t.e0,e) ∨ E_Compare(t.e1,e) ∨ E_Compare(t.e2,e)

Triangle T_setTriangle (t: Triangle, e0: Edge, e1: Edge, e2: Edge)
Exception: E_Compare(e0,e1) ∨ E_Compare(e1,e2) ∨ E_Compare(e2,e0) ⇒ Triangle_Not_Valid
Output: (t.hd,e0,e1,e2)

Fig. 8. Specifications of the Triangle Module.

algorithm, whenever a new vertex is inserted all the triangles where the new vertex falls within its circumcircle (encroached) are deleted. The new cavity is then triangulated by connecting the new vertex to the vertices on the boundary of the resulting cavity. Fig. 13 presents the specifications of point insertion as a part of the Bower/Watson algorithm for Delaunay triangulations.

Used External Functions : NONE
Used External Data Types : Handle = int
: Entity = a
where a is a type variable which
is one of three type Vertex, Edge or Triangle.
: List = (Entity) set
External Constants : NONE
Type Definitions: : NONE
Exported Constants: : MAX_SIZE: int
Assumption: : List is initialized before usage.
Exported Functions:

Exported Function	Input Type	Output Type	Exception
L_size	List	int	
L_addObj	Entity, List	List	List_Is_Full, Obj_Exists
L_delObj	Entity, List	List	Obj_Not_Exist
L_clear	List	List	

State Variables: NONE
Function semantics:

int L_size (l: list)
Output: $|l|$

List L_addObj (e: Entity, l: List)
Exception: $|l| \geq$ MAX_SIZE $\Rightarrow$ List_Is_Full
$\exists$ e1: Entity $\in$ l where e1.hd = e.hd $\Rightarrow$ Obj_Exists
Output: $l \cup \{e\}$
List L_delObj (e: Entity , l: List)
Exception: $e \notin l \Rightarrow$ Obj_Not_Exist
Output: $l - \{e\}$

List L_clear (l: List)
Output: $l = \{\}$

Fig. 9. Specifications of the List Module.

7. Extendability and scalability

The extendability of the introduced mesh generation system is granted by our modularization. For example, Oct-tree based meshing algorithms do not share many operations with Delaunay based algorithms, but our meshing system can be extended to Oct-tree algorithms in a straight forward way. Oct-tree mesh generation requires a tree structure to define the adjacency between the mesh entities. This tree structure will be specified as a variation of the adjacency relation module. The mesh generation algorithm can be considered as a variation of the Delaunay insertion algorithm where vertices are inserted incrementally with different criterion to maintain the tree balancing. Once a node is inserted inside a triangle that includes another vertex, that triangle should be divided into a pre-specified number of children followed by a tree-balancing step. Boundary recovery will also depend on inserting new vertices. This demonstrates that to adopt a completely different mesh generation algorithm only two modules need to be changed. These two modules are the adjacency relation module and the mesh generation module.

The scalability of this meshing system is assumed to be similar to the development of matrix analysis libraries BLAS [9] and LAPACK [1]. The BLAS library provides the basic vector and matrix operation on different data

Used External Functions : NONE
Used External Data Types : Handle_server = (Handle) set
: Vertex = (hd: Handle, cd: Coordinate)
: Edge = (hd: Handle, v0: Vertex, v1: Vertex)
: Triangle = (hd: Handle, e0: Edge, e1: Edge, e2: Edge)
: list = (Entity) set
External Constants : NONE
Type Definitions : NONE
Exported Constants : NONE
Selected Exported Functions :

Exported Function	Input Type	Output Type	Exception
TO_splitEdge	Edge		
TO_InsVertOnEdge	Vertex, Edge		Vert_Not_OnEdge

State Variables:
VertexList: list :=(Vertex) set
EdgeList: list :=(Edge) set
TriangleList: list :=(Triangle) set
Vhandle_server: Handle_server
Ehandle_server: Handle_server
Thandle_server: Handle_server

Selected Function semantics:

TO_splitEdge (e: Edge)
Transition:
v1_temp = E_getStart(e: Edge)
v2_temp = E_getEnd(e: Edge)
cd1_temp = V_getCoord(v1_temp)
cd2_temp = V_getCoord(v2_temp)
cd3_temp = GO_GetEdgeMid(cd1_temp, cd2_temp)

v3hd_temp = HS_getHandle(Vhandle_server)
Vhandle_server =HS_addHandle(Vhandle_server, v3hd_temp)
v3_temp = V_createVertex(v3hd_temp,cd3_temp)
VertexList = L_addObj(v3_temp, VertexList)

e1hd_temp = E_getHandle(e)
e1_temp = E_createEdge(e1hd_temp, v1_temp, v3_temp)
e2hd_temp = HS_getHandle(Ehandle_server)
Ehandle_server=HS_addHandle(Ehandle_server, e2hd_temp)
e2_temp = E_createEdge(e2hd_temp, v3_temp, v2_temp)

EdgeList = L_delObj(e,EdgeList)
EdgeList = L_addObj(e1_temp,EdgeList)
EdgeList = L_addObj(e2_temp,EdgeList)

Fig. 10. Specifications of the Topological Operation Module.

types and LAPACK provides high level routines for different problems like the solution of linear equations, singular value decomposition and many other problems. The newly introduced mesh generation infrastructure is similar in concept to the BLAS library and has been divided into a storage scheme, topological and geometrical operation at the lowest level. On top of this different mesh generation algorithms can be developed. These algorithms can interface cleanly with different storage and data access schemes. Finally, the high level mesh based applications, such as the finite element method can utilize the entire infrastructure.

Used External Functions : NONE
Used External Data Types : Handle = int
: Edge = (hd: Handle, v0: Vertex, v1: Vertex)
EAdj_List = (hd0: Handle, hd1: Handle) set
External Constants : NONE
Type Definitions: : NONE
Exported Constants: : NONE
Assumption: : Adj_List is initialized before usage.
Exported Functions:

Exported Function	Input Type	Output Type	Exception
EA_addpair	Edge, Edge, EAdj_List	EAdj_List	Edge_AlreadyIn_Pair
EA_delpair	Edge, EAdj_List	EAdj_List	Edge_Not_In_Pair
EA_IsAdj	Edge, EAdj_List	Bool	
EA_getAdj	Edge, EAdj_List	Edge	Edge_Not_In_Pair
EA_clear	List	List	

State Variables: NONE
Function semantics:

Bool EA_IsAdj (e: Edge, a: EAdj_List)
Output: ((e0.hd, hd:Handle) ∧ (hd:Handle, e0.hd)) ∈ a)
EAaj_List EA_addPair (e0: Edge, e1: Edge, a: EAdj_List)
Exception: (EAdj_IsAdj(e0,a) ∨ EAdj_IsAdj(e1,a))
⇒ Edge_AlreadyIn_Pair
Output: a ∪ (e0.hd, e1.hd) ∪ (e1.hd, e0.hd)
EAdj_List EA_delEdge (e: Edge, a: EAdj_List)
Exception: (e.hd, hd:Handle) ∉ a ∨ (hd:Handle, e.hd) ∉ a ⇒ Edge_Not_In_Pair
Output: a − {(e.hd, hd:Handle),(hd:Handle, e.hd)}

Fig. 11. Specifications of the edge Adjacency List Module.

8. Conclusions

Using the specified 2D mesh generation infrastructure reliable mesh generation software can be developed in a simple way for any mesh generation algorithm. The clear high level description of the basic entities of the mesh and the complete separation between the topological and geometrical information makes it easy to extend and modify this tool. The high level of abstraction of the containers as sets leaves the selection of an efficient representation for storage of mesh entities until a decision about the meshing algorithm is taken. This can be done in the next step of specification refinement, or it can be left for the implementation phase. The specification presented can significantly help in avoiding any ambiguity during the design process of mesh generation software. Writing the design specifications in a formal way, which is intended for humans and eventually for machine

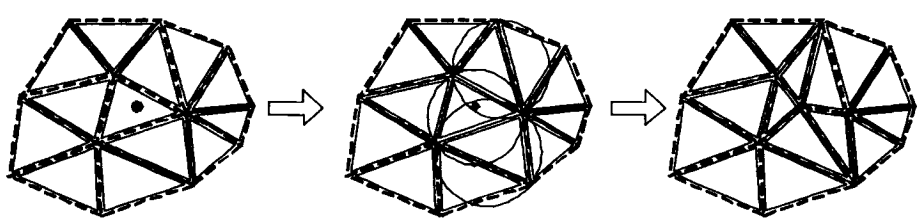


Fig. 12. Bower/Watson point insertion algorithm.

Used External Functions : NONE
Used External Data Types : Handle_server = (Handle) set
: Vertex = (hd: Handle, cd: Coordinate)
: Edge = (hd: Handle, v0: Vertex, v1: Vertex)
: Triangle = (hd: Handle, e0: Edge, e1: Edge, e2: Edge)
: list = (Entity) set
: EAdj_list = (hd0: Handle, hd1: Handle) set
External Constants : NONE
Type Definitions : NONE
Exported Constants : NONE
Selected Exported Functions :

Exported Function	Input Type	Output Type	Exception
D_InsVert	Vertex		

State Variables: VertexList: list :=(Vertex) set; EdgeList: list :=(Edge) set
TriangleList: list :=(Triangle) set
Vhandle_server: Handle_server; Ehandle_server: Handle_server
Thandle_server: Handle_server

Selected Function semantics:

D_InsVert (v: Vertex)
Transition: Del_TriList_temp = { t: Triangle | GO_Check_TriEncroach(
V_getCoord(E_getStart(t.e0)),V_getCoord(E_getStart(t.e1)),
V_getCoord(E_getStart(t.e2)), V_getCoord(v)) }

Del_EdgeList_temp = { e: Edge | T_isEdge(t,e) ∧ t ∈ Del_TriList_temp }

Replaced_EdgeList_temp = { e: Edge | e ∈ Del_EdgeList_temp ∧
EA_getAdj(e) ∉ Del_EdgeList_temp }

for all e ∈ Replaced_EdgeList_temp DO D_CreateTri(v, e: Edge)
EdgeList = EdgeList - { Del_EdgeList_temp - Replaced_EdgeList_temp }
TriangleList = TriangleList - Del_TriList_temp

Internal function semantics:

D_CreateTri (v: Vertex, e: Edge)
Transition: e1hd_temp = E_getHandle(e)
Ehandle_server=HS_addHandle(Ehandle_server, e2hd_temp)
e1_temp = E_createEdge(e1hd_temp, E_getStart(e), v)

e2hd_temp = HS_getHandle(Ehandle_server)
Ehandle_server=HS_addHandle(Ehandle_server, e2hd_temp)
e2_temp = E_createEdge(e2hd_temp, E_getEnd(e) , v)

EdgeList = L_addObj(e1_temp,EdgeList)
EdgeList = L_addObj(e2_temp,EdgeList)

thd_temp = HS_getHandle(Thandle_server)
Thandle_server =HS_addHandle(Thandle_server, thd_temp)
t_temp = T_createTri(thd_temp, e, e1_temp ,e2_temp)
TriangleList = L_addObj(t_temp,TriangleList)

Fig. 13. Excerpts from Delaunay vertex Insertion Module.

verification is considered by the authors as a very reliable method. Identifying exception cases early and defining the proper action to be taken protects the software design from major changes at the testing stages. The complete specification with all exception cases defined will significantly help in driving test cases to check the correctness of the final product as well as for testing each module separately.

References

[1] Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J, Croz JD, Greenbaum A, Hammarling S, Mckenney A, Sorensen D. LAPACK users' guide, 3rd ed. Philadelphia, PA: SIAM; 1999.

[2] Berti G. Generic software components for scientific computing. PhD Thesis. BTU Cottbus, Germany: Faculty of Mathematics, Computer Science, and Natural Science; 2000.

[3] Berti G. GrAL—the grid algorithms library. Lect Notes Comput Sci 2002;2331:745–54.

[4] Bowyer A. Computing dirichlet tessellations. Comput J 1981;24: 162–6.

[5] Chen C-H. A software engineering approach to developing a mesh generator. Master's Thesis. Hamilton, Ont.: McMaster University; 2003.

[6] Glinz M. Problems and deficiencies of UML as a requirements specification language Proceedings of the 10th International Workshop on Software Specification and Design.: IEEE Computer Society; 2000 p. 11.

[7] Hoffman D, Strooper P. Software design, automated testing, and maintenance. A practical approach.: International Thomson Publishing; 1995.

[8] Huth MR, Ryan MD. Logic in computer science: modelling and reasoning about systems. Cambridge: Cambridge University; 2000.

[9] Lawson CL, Hanson RJ, Kincaid DR, Krogh FT. Basic linear algebra subprograms for fortran usage. ACM Trans Math Software, 5 1979; 3(3):308–23.

[10] Object Management Group. OMG unified modeling language specification 2003. Version 1.5.

[11] Parnas DL. On a 'buzzword': hierarchical structure Proceedings of the IFIP 74.: North Holland Publishing Company; 1974 p. 336–339.

[12] Parnas DL, Clement PC, Weiss DM. The modular structure of complex systems International Conference on Software Engineering 1984 p. 408–419.

[13] Parnas DL, Weiss DM, Hoffman D. Software fundamentals: collected papers. In: Parnas DL, editor.. Reading, MA: Addison-Wesley; 2001.

[14] Peled DA. Software reliability methods. Berlin: Springer; 2001.

[15] Piff M. Discrete mathematics: an introduction for software engineers. Cambridge: Cambridge University; 1991.

[16] Remacle J-F, Shephard MS. An algorithm oriented mesh database. Int J Numer Meth Eng 58-2 2003;349–74.

[17] Owre S, Rajan S, Rushby JM, Shankar N, Srivas MK. PVS: combining specification, proof checking, and model checking. In: Alur R, Henzinger TA, editors. Proceedings of the Eighth International Conference on Computer Aided Verification CAV (New Brunswick, NJ, USA/1996), vol. 1102. Berlin: Springer; 1996, p. 411–4.

[18] Shewchuk JR. Delaunay refinement mesh generation. PhD Thesis. Pittsburgh, PA: School of Computer Science, Carnegie Mellon University, May; 1997. Available as Technical Report CMU-CS-97-137.

[19] Watson DF. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. Comput J 1981;24:167–71.