# ADAPTIVE METHODS
# FOR FINITE VOLUME APPROXIMATIONS

A Dissertation

by

STANIMIRE ZDRAVKOV TOMOV

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2002

Major Subject: Mathematics

# ADAPTIVE METHODS FOR FINITE VOLUME APPROXIMATIONS

A Dissertation

by

STANIMIRE ZDRAVKOV TOMOV

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Approved as to style and content by:

Raytcho Lazarov
(Chair of Committee)

Joseph Pasciak
(Member)

David Dobson
(Member)

Vivek Sarin
(Member)

William Rundell
(Head of Department)

May 2002

Major Subject: Mathematics

# ABSTRACT

Adaptive Methods
for Finite Volume Approximations. (May 2002)

Stanimire Zdravkov Tomov, B.S., Sofia University Saint Kliment Ohridski, Bulgaria;
M.S., Texas A&M University
Chair of Advisory Committee: Dr. Raytcho Lazarov

In this dissertation we construct, theoretically justify, and test computational methods and tools that (1) yield reliable error control of the finite volume discretization of convection-diffusion-reaction problems in 2-D and 3-D on unstructured grids, and (2) use parallel computational resources efficiently. We achieve balance between obtaining reliable control of the error and efficient use of the available computational resources by an adaptive process of parallel local grid refinement based on a posteriori error analysis.

In our a posteriori error analysis we exploit the ideas known from the finite element method. Namely, we use estimators based on local residuals, local Dirichlet or Neumann problems, and Zienkiewicz-Zhu type estimators. We have constructed such estimators for finite volume approximations and have theoretically justified them by proving, under appropriate assumptions, that they provide both lower and upper bounds for the error. The equivalence of the error and the estimated error is dependent on certain constants. The constants' dependence on the problem's parameters is discussed. The analysis is performed in global $H^1$, $L^2$, and energy norms. Of special interest are problems with large convection. For these problems the standard approximation techniques give oscillating numerical results and mass disbalance. Different up-wind approximations, that give a desired stabilization of the discretization, are discussed and taken into account in the a posteriori error estimators. The results, obtained for steady-state convection-diffusion-reaction problems, are extended to time dependent problems.

The dissertation also carries out extensive numerical testing of the theoretical results. There are two groups of numerical experiments. The first group contains experiments with known exact solutions, for which case the exact error is compared with the numerical results from the error estimators. The second group deals with problems with unknown exact solutions. Also, we have included tests to study the behavior of the error estimators for problems varying from pure diffusion to large convection.

Furthermore, we introduce the parallel local grid refinement algorithms that we implemented into a parallel grid generation tool. Parallel grid generation is essential for the efficient parallel implementation of the adaptive methods and plays an important role in numerical simulations that rely on high performance parallel computers. This part of the dissertation includes element refinement/de-refinement algorithms, techniques to maintain load balance, general issues in developing parallel finite volume (or finite element) code based on domain decomposition, ways to maintain mesh conformity on the different refinement levels, data structures, etc.

In general, we conclude that the ideas from the finite element a posteriori error analysis theory can be successfully used in deriving reliable error estimators for the finite volume method. The efficiency of the error estimators and the parallel grid refinement techniques that we derived is supported by various numerical experiments.

To my parents

# ACKNOWLEDGMENTS

First and most of all I am grateful to my advisor, Prof. Raytcho Lazarov, whose overwhelming help, support, and encouragement were present throughout my graduate studies. His guidance, advice, and encouragement made me work hard and progress in the area that I love. I feel lucky and thankful for the wonderful research environment and opportunities that I enjoyed under his supervision. Apart from the academic area, Dr. Lazarov and his family's friendship helped me not only to easily adjust to the American way of life, but also to feel at home and enjoy my graduate studies and life at Texas A&M University.

I am thankful to the faculty in the numerical analysis group. Their excellence and concern about their students made me work harder in order to meet their high standards and expectations. Prof. Joseph Pasciak has been my model for mathematical rigorousness and perfection since the first numerical analysis class that he taught in my first semester at TAMU. I greatly appreciate all his concern, valuable remarks, and advice throughout my graduate studies. I enjoyed and learned a lot from Prof. James Bramble's special topic classes on multigrid, iterative techniques, finite element method theory, etc. It was always a pleasure to sit in his classes and get inspired from his personality and search for beauty in complicated mathematical proofs.

I thank my committee members for their constructive remarks and suggestions. Also, I greatly appreciate them as instructors and as the professors from whom I learned most. I still consult Prof. David Dobson's excellent notes in functional analysis. I enjoyed and learned a lot in Prof. Vivek Sarin's class "Parallel and Distributed Numerical Algorithms". I thank Prof. Konduru Sivaramakrish from the Department of Accounting for serving on my committee as the Graduate Council

Representative.

I am indebted to the collaborators of the numerical analysis group with whom I worked. And more precisely, to Prof. Joachim Schöberl, whose course on Scientific Computing considerably improved my computational skills. I thank Prof. Carsten Carstensen for his advice and ideas in our joint work. I am especially grateful to Prof. Panayot Vassilevski from the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory (LLNL). He was my supervisor and mentor during three important for my work and education summer internships in the laboratory. I also thank Prof. Charles Tong from LLNL, with whom I worked in Livermore.

I am grateful to the department of Mathematics, especially to Prof. Jay Walton and Prof. Thomas Schlumprecht, who have been the Graduate Advisors during my study, and to Ms. Monique Stewart.

Special thanks to my student colleagues, who shared with me their knowledge and were always ready to help in time of need. I am also thankful to all my friends that made my stay here enjoyable and productive.

I am thankful to my parents and my brothers, who although far away from here, were always with me, encouraged me, believed in me and gave me the strength to pursue my goals.

Special thanks to my fiancee Beverly Wachtel for her attentive editing.

I would like to acknowledge the financial support that I received during my study. I am grateful to the Department of Mathematics and the Institute for Scientific Computing for providing my graduate teaching and research assistantship throughout my study. I am also thankful to the Center for Applied Scientific Computing at LLNL for offering their exceptional facilities and training program.

# TABLE OF CONTENTS

CHAPTER Page

I INTRODUCTION . . . . . . . . . . . . . . . . . . . . . . . . . 1

1.1. Objectives . . . . . . . . . . . . . . . . . . . . . . . . . 2
1.2. Approach . . . . . . . . . . . . . . . . . . . . . . . . . . 2
1.3. Outline . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

II MODEL PROBLEMS AND THEIR DISCRETIZATIONS . . . 12

2.1. Motivation for our study . . . . . . . . . . . . . . . . . . . 12
  2.1.1. Conservation problems . . . . . . . . . . . . . . . . . 13
  2.1.2. Locally conservative approximation methods . . . . . 15
2.2. Model problems . . . . . . . . . . . . . . . . . . . . . . . 17
  2.2.1. Steady state problems . . . . . . . . . . . . . . . . . 17
  2.2.2. Transient problems . . . . . . . . . . . . . . . . . . 20
2.3. Finite volume discretization . . . . . . . . . . . . . . . . . 21
  2.3.1. Discrete spaces and grids definitions . . . . . . . . . 22
  2.3.2. Standard approximation on general grids . . . . . . . 26
  2.3.3. Upwind approximation on general grids . . . . . . . . 27
  2.3.4. Implicit finite volume discretization for transient problems . . . . . . . . . . . . . . . . . . . . . . 28

III ADAPTIVE METHODS . . . . . . . . . . . . . . . . . . . . . 30

3.1. A general adaptive methods framework . . . . . . . . . . . 33
  3.1.1. Adaptive methods for steady-state problems . . . . . 33
  3.1.2. Adaptive methods for transient problems . . . . . . . 34
3.2. A numerical motivation for adaptive grid refinement . . . . 36
3.3. Review of some adaptive finite element methods . . . . . . 38
  3.3.1. A general a posteriori error analysis framework . . . 38
  3.3.2. Residual based (RB) error estimator . . . . . . . . . 41
  3.3.3. Zienkiewicz-Zhu (ZZ) type estimators . . . . . . . . . 42
  3.3.4. Error estimators based on solution of local Dirichlet/Neumann problems . . . . . . . . . . . . . . . . . 43
  3.3.5. Error estimators based on dual problems . . . . . . . 44
  3.3.6. Hierarchical (HB) refinement . . . . . . . . . . . . . 45
  3.3.7. Error estimator based on second derivative (SD) . . 46

CHAPTER Page

3.3.8. Error indicator based on the gradient . . . . . . . . . 47

IV ADAPTIVE FINITE VOLUME METHODS . . . . . . . . . . . 48

4.1. Estimators for standard steady-state approximations . . . . 49
4.1.1. Error representation in the energy norm . . . . . . . 51
4.1.2. Residual type error estimator . . . . . . . . . . . . 54
4.1.2.1. Reliability . . . . . . . . . . . . . . . . . . . 56
4.1.2.2. Efficiency . . . . . . . . . . . . . . . . . . . . 57
4.1.3. Zienkiewicz-Zhu type error estimator . . . . . . . . . 62
4.1.3.1. Reliability . . . . . . . . . . . . . . . . . . . 64
4.1.3.2. Efficiency . . . . . . . . . . . . . . . . . . . . 67
4.1.4. Estimators based on local Dirichlet/Neumann problems 68
4.1.4.1. Definition . . . . . . . . . . . . . . . . . . . . 68
4.1.4.2. Reliability and efficiency . . . . . . . . . . . 69
4.1.5. Estimators based on dual problems . . . . . . . . . . 70
4.2. Estimators for upwind steady-state approximations . . . . . 74
4.3. An error indicator for transient problems . . . . . . . . . . 77

V COMPUTATIONAL ASPECTS OF THE ADAPTIVE METHODS 81

5.1. Adaptive mesh generation . . . . . . . . . . . . . . . . . . . 83
5.1.1. ParaGrid, a parallel adaptive mesh generator . . . . 84
5.1.2. Element refinement algorithms . . . . . . . . . . . . 87
5.1.3. Element de-refinement . . . . . . . . . . . . . . . . . 91
5.2. Mesh partitioning and load balancing . . . . . . . . . . . . 91
5.2.1. Mesh partitioners . . . . . . . . . . . . . . . . . . . . 93
5.2.2. Load balancing for steady-state problems . . . . . . . 94
5.2.3. Load balancing for transient problems . . . . . . . . 96
5.3. Parallel computations . . . . . . . . . . . . . . . . . . . . . 97
5.3.1. Code optimization . . . . . . . . . . . . . . . . . . . 97
5.3.1.1. Locality of reference . . . . . . . . . . . . . . 98
5.3.1.2. Software pipelining . . . . . . . . . . . . . . . 99
5.3.1.3. Performance monitoring . . . . . . . . . . . . 101
5.3.2. Parallel FEM/FVM using OpenMP . . . . . . . . . . 102
5.3.3. Parallel FEM/FVM using MPI . . . . . . . . . . . . 105
5.4. Code organization and data structures . . . . . . . . . . . . 109
5.4.1. The overall code structure . . . . . . . . . . . . . . . 110
5.4.2. Domain decomposition data structures . . . . . . . . 112
5.4.3. Multigrid data structures . . . . . . . . . . . . . . . 113

CHAPTER Page

5.4.4. Internal and external solvers and preconditioners . . 113
5.5. Visualization . . . . . . . . . . . . . . . . . . . . . . . . . 113

VI NUMERICAL RESULTS . . . . . . . . . . . . . . . . . . . . . . 116

6.1. Results with known exact error . . . . . . . . . . . . . . . . 116
6.2. Problems with singular solutions . . . . . . . . . . . . . . . 123
6.3. Application to steady state fluid flow in porous media in 3-D . . . . . . . . . . . . . . . . . . . . . . . . . . . . 131

VII CONCLUSIONS . . . . . . . . . . . . . . . . . . . . . . . . . 139

REFERENCES . . . . . . . . . . . . . . . . . . . . . . . . . . . . 141

VITA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 150

# LIST OF TABLES

TABLE Page

I Numerical results for Example 2, $\varepsilon = 0.01$. . . . . . . . . . . . . . . . . . 119

# LIST OF FIGURES

FIGURE Page

1 Left: Finite element and finite volume partitions in 2-D; Right: Contribution from one element to control volume $V_i$, $\gamma_{ij}$ and $\gamma_{ik}$ in 3-D; Point $q$ is the element's medicenter and internal points for the faces are the medicenters of the faces. . . . . . . . . . . . . . . . 24

2 Control volumes with circumcenters as internal points (Voronoi meshes) and interface $\gamma_{ij}$ of $V_i$ and $V_j$. The right picture shows the segments $\beta_i$ in bold. . . . . . . . . . . . . . . . . . . . . . . . . 25

3 Exact error $e_h$, for solutions $u \in H^{1+4/3-\epsilon}$ (Problem 1), $u \in H^{1+2/3-\epsilon}$ (Problem 2), and $u \in H^{1+1/2-\epsilon}$ (Problem 3), plotted against the degrees of freedom on a log-log scale for uniformly refined grid and locally refined grid based on a posteriori error estimates. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37

4 Distribution of the normal vectors to $\gamma_{ij}$ in one finite element. . . . . 76

5 Crankshaft mesh. The domain is split into $5,830$ elements, with $4,176$ surface elements and $2,140$ vertices. . . . . . . . . . . . . . . . 85

6 Refinement, de-refinement, and maintenance of the refinement edge in 2-D. The refinement edge is kept always between local vertices 0 and 1. When bisecting the new refinement edges are opposite the new vertex `vm` and for uniform refinement they are kept as in the parent. The de-refinement of elements obtained by uniformly refinement is similar to the de-refinement of elements obtained by bisection: `v0:=K0.node[0]`, `v1:=K1.node[1]`, and `v2:=K2.node[2]`. . . . . . . . . . . . . . . . . . . . . . . . . . . 88

7 The 4 different types of shapes obtained in the process of uniform refinement or bisection. Note that bisecting shape $D$ leads to shapes $B$ and $C$. . . . . . . . . . . . . . . . . . . . . . . . . . . 89

FIGURE Page

8 Algorithm for bisection of a tetrahedron. Here `ME` abbreviates marked edge. Note that in the refinement process the vertices are ordered so that the orientation of the elements is preserved. . . . . . 90

9 Crankshaft mesh split in 4 by `pmetis`. The subdomains, starting from left, have correspondingly 1458, 1457 , 1458, and 1457 tetrahedrons. 94

10 Overall code structure : Used Libraries, Developed Libraries and Developed Applications. . . . . . . . . . . . . . . . . . . . . . . . . . 111

11 Visualization classes and their dependency tree . . . . . . . . . . . . 114

12 Exact error, $\eta_Z$, and $\eta_E$ for uniformly refined meshes over different mesh levels for the three problems in Example 1. . . . . . . . . . . . 117

13 Exact error, $\eta_Z$, and $\eta_E$ for locally refined meshes over different mesh levels for the three problems in Example 1. . . . . . . . . . . . 118

14 Reaction-diffusion problem with $\varepsilon = 0.01$; the error (left) and the mesh (right) obtained for level 5 with 2729 grid points. . . . . . . . . 120

15 Reaction-diffusion problem with $\varepsilon = 0.001$; the error (left) and the mesh (right) obtained for level 5 with 2385 grid points by the residual type error estimator (0.3 tolerance). . . . . . . . . . . . . . . 121

16 Reaction-diffusion problem with $\varepsilon = 0.001$; meshes obtained by the error estimator based on local Dirichlet problems (left) with 2013 grid points and by the Zienkiewicz-Zhu type error estimator (right) with 2031 grid points at level 5. . . . . . . . . . . . . . . . . 122

17 Convection-diffusion problem from Example 3 with $\varepsilon = 10^{-3}$; the mesh (left) with 1209 grid points and solution (right) obtained at level 4. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 123

18 Convection-diffusion problem from Example 3 with $\varepsilon = 10^{-5}$; the mesh (left) with 1761 grid points and the solution (right) obtained at level 4. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 124

19 Example 4 : the mesh (left) with 2562 grid points and the level curves (right). . . . . . . . . . . . . . . . . . . . . . . . . . . . . 125

FIGURE Page

20 Computational meshes obtained by local refinement due to $L$-shaped corner singularity (left) and delta function source term (right). Both meshes are for level six with correspondingly 12350 and 11770 nodes. . . . . . . . . . . . . . . . . . . . . . . . . . . 126

21 Example 6 : Poisson equation in a slit domain with solution $u = r^{1/2}\sin(\frac{\theta}{2})$; the mesh (left) with 2064 grid points and the error (right) for level 9. . . . . . . . . . . . . . . . . . . . . . . . . . . 127

22 Convection-diffusion problem in $L$-shaped domain; the mesh on level 5 (left) with 5232 nodes and 10247 triangles; the level curves (right) on the same level. . . . . . . . . . . . . . . . . . . . . . . . 128

23 Convection-diffusion problem in $L$-shaped domain; the mesh on level 4 (left) with 3450 nodes and 6798 triangles; the level curves (right) on the same level. . . . . . . . . . . . . . . . . . . . . . . . 129

24 Convection-diffusion problem in domain with layers; the obtained locally refined mesh after 5 levels of refinement (left) with 6035 nodes and 11892 triangles; the level curves (right) on the same level. 130

25 Homogeneous reservoir; the 3-D mesh on refinement level 6 (top) with 54129 nodes; the concentration level curves (bottom) in the plane $x_2 = 0$ for the case of low biodegradation rate. . . . . . . . . 132

26 Homogeneous reservoir; the mesh (top) in plane $x_2 = 0$ on refinement level 5 (globally with 36427 nodes); the concentration level curves (bottom) in plane $x_2 = 0$ for the case of medium biodegradation rate. . . . . . . . . . . . . . . . . . . . . . . . . . 133

27 Pressure computations in a non-homogeneous reservoir; (top) the locally refined 3-D Mesh on level 2 with 4053 nodes; (bottom) Contour curves of the pressure for level 2. . . . . . . . . . . . . . . . 134

28 Concentration distribution in a non-homogeneous reservoir; (top) the 3-D mesh on refinement level 4 with 39445 nodes; (bottom) contour curves of the concentration for the cross-section $x_2 = 0$; the permeability in the layer is two times smaller than in the rest of the domain. . . . . . . . . . . . . . . . . . . . . . . . . . . 135

FIGURE Page

29 Concentration distribution in a non-homogeneous reservoir; contour curves of the concentration for permeabilities in the layer 5 times (top) and 10 times (bottom) smaller than the permeability in the rest of the domain. . . . . . . . . . . . . . . . . . . . . . . . . 136

30 Pressure computations in a non-homogeneous reservoir with a well; (top) the 3-D Mesh on level 3 with 34236 nodes; (bottom) contour curves of the pressure on level 3. . . . . . . . . . . . . . . . . . 137

31 Concentration distribution in a non-homogeneous reservoir with a well; (top) the 3-D mesh with 67509 nodes in half of the domain obtained after 5 levels of refinement; (bottom) contour curves of the concentration in the plane $x_2 = 0$. . . . . . . . . . . . . . . . . . . 138

# CHAPTER I

# INTRODUCTION

Today's technological advances facilitate and stimulate the research requiring very large scale computations. For example, the numerical discretization of fluid flow and transport in porous media, elastic and plastic deformations, heat and mass transfer, electro-magnetics, may yield linear systems with millions of unknowns, and hence leads to large scale computations. Considering the cost of these large scale computations, it is natural to raise the following two questions.

First, are these computations really necessary. In the case of numerical solutions of partial differential equations (PDEs), this question relates to the problem of estimating and controlling the error due to the discretization. In other words, this is a problem of determining the accuracy and the reliability of the computational method.

The second question is whether the available computational resources are used efficiently. This is related to the fact that very often the solution of a problem of practical importance exhibits local behavior due to discontinuities of the coefficients, localized sources, and boundary data, as well as singularities due to corners, boundary layers, or non-linear behavior. In such cases the overall accuracy of the numerical approximation deteriorates and an obvious remedy that would increase the efficiency would be to refine the computational mesh in the regions of such singular behavior. Problems that arise are: (1) to determine the regions of singular behavior of the solution and (2) to refine the mesh in a balanced manner, so that the overall accuracy is uniform in the whole domain. Furthermore, when discussing large scale computations, one may face hardware problems due to fundamental physical limitations on

---

This dissertation follows the style and format of SIAM Journal of Numerical Analysis both system memory and computer processing speed. The usual remedy for such problems is the use of parallel machines.

## 1.1. Objectives

The above discussion motivates the following three main goals of this dissertation:

- to develop methods for reliable control of the error due to the discretization of the continuous problem,
- to develop methods and strategies for efficient use of the available computational resources, and
- to implement these methods and strategies into an efficient computational methodology for solving convection-diffusion-reaction problems.

## 1.2. Approach

We achieve balance between the first two goals by using adaptive methods with feedback from the computations. That is, we do local grid refinement based on a posteriori error analysis. The a posteriori error analysis provides efficiently computable stopping criterion guaranteeing error control. Further, this analysis is a basis for various grid refinement strategies in case the stopping criterion is not satisfied.

An alternative of a posteriori error control, which we will not discuss in the dissertation, is the asymptotic analysis of the unknown solution and various techniques for obtaining expansions with respect to the parameters of the PDE problem. This information then can be used to construct *a priori* the type of the mesh. For large classes of problems this will lead to Shishkin, Bahvalov, and other meshes (see, for example, the monographs of Miller, O'Riordan, and Shishkin [45], Roos, Stynes, and Tobiska [56], or the survey paper of Roos [55]). This type of technique is very successful for steady-state singularly perturbed problems and leads to excellent results when combined with relevant computational methods. Mostly the grids are piecewise uniform, which can also reduce the storage requirement and the solution cost. However, *a priori* fitted meshes have certain limitations when it comes to embedding the numerical method in a general computational procedure for multidimensional problems. In such cases a posteriori error analysis and adaptive grid refinement are essentially the only practical alternative.

Adaptive methods, and more precisely, adaptive methods for finite element approximations, have been extensively used and studied in the past 25 years with emphasis on both the theoretical and computational aspects of the methods. Concerning the theoretical aspects, the research in the area starts with the pioneering paper of Babuska and Rheinboldt [5] and continues with studies devoted to the so called *Residual Based* method (see the survey paper of Verfürth [69]). In this approach certain local residuals are evaluated and then the a posteriori error indicator is obtained by solving local Dirichlet or Neumann problems, taking the residuals as data [5, 9]. Another approach of the method uses the Galerkin orthogonality, a priori interpolation estimates, and global stability in order to get error estimators in global $L^2$- and $H^1$-norms (see, for example, [26]). Furthermore, solving appropriate dual problems leads to error estimators controlling various kinds of error functionals [11]. Solving finite element problems in an enriched function space (by hierarchical bases) gives rise to the so called *Hierarchical Based* error estimators [9]. There are also error estimators that control the error or its gradient in the maximum norm. Such estimators are based on optimal a priori estimates for the error in maximum norm [28]. Another error indicator, which is widely (and in most cases heuristically) used in many adaptive finite element codes, is the *Zienkiewicz-Zhu* (often called *ZZ*) error estimator [72, 73].

This estimator is based on post-processing of the computed solution gradient in order to get a better approximation, which is later used instead of the exact gradient to bound the energy norm of the error. Some analysis of the method can be found in [54] and the literature cited there.

In this dissertation we study adaptive methods for finite volume approximations. There are few works related to a posteriori error estimates for finite volume methods. In an early work [2], Angermann has studied a balanced a posteriori error estimate for finite volume discretizations for convection-diffusion equations in 2-D on Voronoi meshes. His basic error estimator is derived using the idea of a previous work [3] on finite element method, but contains two new terms, which he has studied. In this dissertation we take a similar path. Namely, the error estimates for the finite volume method are derived using the relation and similarities between the finite volume and finite element methods. Following this approach, estimators based on local residuals, local Dirichlet or Neumann problems, and Zienkiewicz-Zhu type estimators are constructed and theoretically justified by proving, under certain assumptions, that they provide both lower and upper bounds for the error. The equivalence of the error and the estimated error depends on certain constants, whose dependence on the problem's parameters is discussed. The analysis is performed in global $H^1$, $L^2$, and energy norms. The theory of the finite volume methods is still being developed. This raises some difficulties in establishing an independent a posteriori error analysis for finite volume approximations. For instance, optimal order a-priori $L^2$-error estimates with minimal regularity of the solution are not known for the finite volume methods for elliptic equations. Optimal $W^{1,p}$-error estimates for elliptic and parabolic problems have been obtained in the recent studies [30, 44] for $1 < p \leq \infty$.

Our interest in finite volume methods (FVM) is based on their attractiveness for the numerical simulation of various conservation laws. FVM, as the finite element methods, have the following appealing features

- they may be used on general domains;
- they may be used on both structured and unstructured meshes;
- they lead to "good" approximation schemes.

Additionally, since FVMs are based on a "balance" approach (see Chapter II), they feature **local conservation** of numerical fluxes. This property makes them natural to use for modeling when flux and "mass" conservation are important. For example, FV methods have been extensively used in fluid mechanics, heat and mass transfer, etc. Here we use finite volume discretization of steady-state convection-diffusion-reaction problems in 2/3-D on unstructured grids. Such model problems arise, for example, from simulation of fluid flow and transport in porous media. In that case the unknown quantity may represent the concentration of a chemical dissolved and distributed in water due to processes of advection, diffusion, and absorption. The solution of such problems exhibits local behavior, which is due to discontinuity in the boundary data and the coefficients of the differential equations, and/or other local phenomena (for example extraction/injection wells, etc.). Applying adaptive methods to such problems is beneficial and in many cases essential for accurate computations. Of special interest is the case of problems with large convection. Then the standard finite volume approximation techniques give oscillating numerical results and mass disbalance. For such problems we are interested in approximation methods which produce solutions satisfying the maximum principle and are locally conservative. Such schemes are also known as monotone schemes (see, e.g. [36, 56]). A well-known sufficient condition for a scheme to be monotone is that the corresponding stiffness matrix is an $M$–matrix (see [59] p. 182, p. 260 and [56] p. 202). In Chapter II we introduce an upwind approximation for both 2 and 3-D problems that is locally mass conservative and gives the desired stabilization. The upwinding is taken into account in the derived a posteriori error estimators. Other good choices of monotone upwind schemes known from the finite element analysis are Tabata's scheme [64], which is based on upstream weighting approximation of the convection term and produces good computational results on quite general grids, the stream-line upwind Galerkin method (SUPG scheme) of Franca, Frey, and Hughes [33], and the scheme of Xu and Zikatanov [70], which constructs a finite element discretization by an appropriate averaging of the differential equation coefficients on the element edges.

In addition, the results obtained for steady-state convection-diffusion-reaction problems are extended to time-dependent problems. Our approach is similar to the one taken by Eriksson et al. in [26] and [27]. The approximation is done in the following manner. First, we discretize in space using the finite volume method and then discretize in time using the discontinuous Galerkin method. The a posteriori error estimator derived has terms indicating a posteriori the error due to the space discretization (as in the steady-state case) and additional terms measuring the error produced by the time discretization.

An important part of this dissertation, as stated at the beginning, is the practical implementation of the adaptive methods on particular computer architectures. The implementation problems are related to computer science in more than one way. We need special data structures and algorithms, have to resolve management issues, etc. The difficulties arise from the three specific requirements, targeted in our implementation. They are described below.

First, our computations are mainly targeted to 3-D problems. The local refinement procedures in 3-D are significantly more complicated to implement than in 2-D. For 3-D problems we consider tetrahedral meshes and the refinement is done by bisection using the algorithm described by Arnold et al. in [4]. We prefer this approach since it fits our data structures well. Moreover, a repeated application of the algorithm does not lead to mesh degeneration. This algorithm is suitable for parallel adaptive mesh refinement. Also, one can control the mesh growth factor between the refinement levels. More about unstructured mesh generation can be found in [49] and the literature cited there. Alternatives for tetrahedral bisection can be found in the literature cited in [4]. For 2-D problems we use triangles and refine them, depending on how fast we want the mesh size to grow, by uniform splitting into four or bisection. The details are given in Chapter V.

Second, we consider time dependent problems, which in general will require derefinement mesh procedures as well. To derefine the mesh one needs to know the refinement history of the mesh. We keep this history in tree data structure for every element from the starting coarse mesh. This is convenient and requires minimum memory for a derefinement process.

And third, we require the computations to be done in parallel. In order to get efficient parallelization of the adaptive methods we have to address various issues. The first one, and probably the most important, is the selection of the solver for the resulting (from the discretization) algebraic system. Since the finite volume discretizations lead to sparse systems, it is natural to consider iterative methods for their solution. The storage needed to assemble the global matrix (in sparse format) is the same in both direct and iterative solution methods, but the direct methods may produce an arbitrary amount of fill-ins that for 3-D problems might be prohibitively high. Also, since matrix-vector operations can be efficiently parallelized, all iterative solvers can highly benefit the parallelism, while the direct methods need sophisticated techniques to extract limited parallelism. One disadvantage of the iterative methods is that the rate of convergence is problem dependent and may be unacceptably slow. Therefore, another issue that we address, is how to maintain data structures that provide a base for acceleration of the iterative methods by preconditioning techniques. When iterative methods are parallelized on a multiprocessor system the data distribution and the communications are of greatest importance for an efficient execution. Both the data distribution and the communication scheme are usually determined before the execution of the solver by preprocessing. There are many ways to achieve this. It seems that the most popular and efficient way is the use of *Domain Decomposition* data distribution techniques. This is the method of our choice.

Domain decomposition methods use a *divide-and-conquer* concept whose main idea is to split the global problem into sub-problems, solve them concurrently, and somehow merge or combine the local solutions in order to get the global one. This idea translates into the following: first, finding a splitting of the global mesh; second, mapping every subdomain of the splitting to a processor, and performing independent computations on each subdomain; and third, transferring data when necessary. Crucial for the efficient parallel execution of software based on this technique is obviously the quality of the splitting. It should be such that

- there is *load balance* among the processors, and
- the interface between the subdomains is minimal in a certain sense.

In order to have load balance over the processors, the number of elements on each processor should be almost equal, and in order to reduce the communication, the number of nodes on the boundary between the sub-domains should be minimal. *Domain Decomposition* techniques address the question of preconditioning. These are undoubtedly one of the best known and most promising methods, which also take advantage of the parallelism. We exploit the multilevel structure obtained in the process of consecutive local refinements by implementing data structures and procedures utilizing the development of multigrid type preconditioners.

Another main issue concerning the efficient parallelization of adaptive methods is the parallel mesh generation. Parallel grid generation tools play an important role in the scientific research that requires the power of high performance parallel computers. Challenges to be solved are maintaining the mesh conforming between the subdomains on the different refinement levels, maintaining the load balance, etc. We have tested different strategies to maintain the load balance. One is to use a posteriori error estimates as weights in an element based initial splitting of the coarse mesh into sub-domains. The idea is that if the error is balanced over the subdomains, then the local refinements that follow will produce computational mesh with number of tetrahedrons balanced over the subdomains. This strategy does not work for time-dependent problems where we consider element "migration" between the subdomains. The load balance is computed among the elements on the starting mesh with weights being the number of "children" that they have. If a coarse element has to migrate to a neighboring subdomain it is sent along with its children. Such a strategy reduces the number of transfers between the subdomains, reduces the size of the graph to be partitioned (actually it stays the same with only the weights changing), and makes the derefinement easier than if elements from the finest level were to migrate.

## 1.3. Outline

This dissertation has four main components:

1. *Introductory part*;
2. *A posteriori error estimators*;
3. *Parallel mesh refinement*;

4. *Numerical results.*

The *introductory part* consists of two chapters. First, the general form for the problem of interest, convection-diffusion-reaction problem, and its finite volume discretization are introduced in Chapter II. Second, in Chapter III, we give an introduction to adaptive methods. We presents a general adaptive methods framework and also, since we use the ideas from the finite element method, a review of the a posteriori error analysis for the finite element method.

The *a posteriori error estimators* are presented in Chapter IV. The chapter contains the construction and theoretical justification of several a posteriori error estimators for finite volume approximations. Namely, we have estimators based on (1) local residuals, (2) local Dirichlet and Neumann problems, and (3) Zienkiewicz-Zhu type estimators. This chapter also contains the extension of the results obtained for steady state problems to transient problems.

The *parallel mesh refinement* strategies are discussed in Chapter V. The algorithms and the implementation issues concerning the development of parallel grid generation tools are explained. This includes refinement and derefinement algorithms, techniques to maintain load balance, general issues in developing parallel finite element/volume codes based on domain decomposition, ways to maintain mesh conformity on the different refinement levels, data structures, etc.

The *numerical results* are presented in Chapter VI. This part contains various experiments that numerically illustrate the theory that we developed. There are two groups of experiments. The first group contains numerical experiments for model problems with known exact solutions. Here the exact error is compared with the numerical results from the error estimators that we have developed. The second group of experiments is for more realistic problems with unknown exact solution.

Namely, we consider various convection-diffusion-reaction equations with boundary layers and singular solutions due to corners of the boundary or discontinuity of the coefficients. We show the numerical behavior of the error estimators for problems varying from large convection to large diffusion. Also, we consider realistic model problems with applications to steady state fluid flows in 3-D.

Finally, in Chapter VII, we give a summary of the work presented, conclusions, possible extensions and future research plans.

# CHAPTER II

# MODEL PROBLEMS AND THEIR DISCRETIZATIONS

Our main interest is in engineering and physics conservative problems. These are problems of determining the distribution of a certain quantity, having some conservation properties, in time and space. Examples may be problems from fluid mechanics, heat and mass transfer, electro-magnetics, etc. The mathematical model for these problems consists of conservation laws, which usually are convection-diffusion-reaction differential equations, and constitutive relation, used to close the system of unknowns introduced by the conservation laws [23, 31, 46]. In fluid dynamics, for example, conservation and constitutive laws may model the distribution of a contaminant, dissolved and distributed in water in a porous media, due to the processes of convection, diffusion, and reaction.

This chapter starts with a motivation for our interest in conservative problems (Section 2.1). This includes a brief description of the conservation type of problems (Subsection 2.1.1) and locally conservative approximation methods (Subsection 2.1.2). In Section 2.2 we specify the form and the assumptions about the conservative problems that we are interested in. We consider in different subsections correspondingly steady state and transient problems. The last section, Section 2.3, introduces the discretizations used for the numerical solutions of the continuous problems described in the previous section.

## 2.1. Motivation for our study

This section gives a brief motivation for our interest in the finite volume methods. The choice is motivated by the problems described in Subsection 2.1.1. These problems are conservative and it is natural, and in many cases essential that their numerical approximation be locally conservative (see Subsection 2.1.2).

### 2.1.1. Conservation problems

The distribution of quantities in time and space depends on various physical processes and can be modeled by using the underlying physical principles. An important class of physical principles, that is in the basis of many computational models in engineering and physics, expresses the conservation of a certain quantity. Examples of quantities and corresponding conservation principles are

- mass – *mass is neither created nor destroyed*;
- energy – *energy is neither created nor destroyed*;
- momentum – *the rate of change of momentum is proportional to the applied force* (Newton's second law).

The conservation principles give rise to various conservation laws, which usually take the following general integral form

$$\frac{d}{dt}\int\limits_{V} q(x,t)\ dx + \int\limits_{\partial V} \underline{\sigma}(x,t)\cdot\underline{n}\ ds = \int\limits_{V} F(q,x,t)\ dx, \tag{2.1}$$

where $V$ is any fixed sufficiently regular subregion of a domain $\Omega$, $q$ is the quantity being preserved, $x$ and $t$ are correspondingly the space and time variables. The first term on the left represents the rate of increase of $q$ in $V$, the second is the rate at which $q$ is crossing $\partial V$ in outward direction. The vector function $\underline{\sigma}(x,t)$, often called *vector flux*, expresses the mechanism in which $q(x,t)$ is distributed in space. Examples for various mechanisms are given below. The term on the right gives the rate of creation or loss of the substance $q$. The scalar function $F(q,x,t)$, often called source term, according to the problem area may model heat source, chemical reaction, production/injection well, etc.

Functions $q$, $\underline{\sigma}$, and $F$, used in equation (2.1), are usually related trough some scalar or vector unknown. The relations, often called *constitutive laws*, are used to close the system of unknowns introduced by models of global physical principles, like the conservation principles, discussed above. For example, for models of flows in porous media $q$, $\underline{\sigma}$, and $F$ may be related to one of the following two quantities.

1. The pressure $p$. For incompressible flow $q = \rho p$, where $\rho$ is the mass density of the fluid and the constitutive law is the Darcy's law

   $$\underline{\sigma}(x,t) := -D\nabla p.$$

   Here $D$ is $n \times n$ tensor (matrix) representing the permeability of the porous media (see [51]). In the groundwater hydrology literature the normalized tensor $D$ is called hydraulic conductivity.

2. The mass concentration $C$. In this case $q = C$ and

   $$\underline{\sigma}(x,t) := -A\nabla C + \underline{b}C, \quad F = f(x,t) - c(x,t)C.$$

   Here $A$ is $n \times n$ tensor (matrix) representing the diffusion-dispersion distribution mechanism of $C$ in the domain $\Omega$, the convection vector $\underline{b}$ represents the distribution of $C$ due to flow, usually of water, of direction and magnitude given by $\underline{b}$, and $c(x,t) \geq 0$ represents the absorption rate by the media (see [51]).

In our analysis we relate $q$, $\underline{\sigma}$, and $F$ to a scalar function $u(x,t)$. We take $q := u(x,t)$,

$$\underline{\sigma}(x,t) := -A\nabla u + \underline{b}u, \tag{2.2}$$

and

$$F(q,x,t) := f(x,t) - c(x,t)u,$$

where $A$ is a $n \times n$ tensor, $\underline{b}$ is a $n$-dimensional vector function, and $f$ is a scalar function.

### 2.1.2. Locally conservative approximation methods

We are interested in numerical methods that

- may be used on general domains;
- may be used on both structured and unstructured meshes;
- may be used for conservation problems and lead to "good" approximation schemes.

By "good" scheme we mean both cheap, i.e., a scheme that is easy to compute and implement for complex problems, and robust, i.e., a scheme that is stable even for particularly difficult problems, such as problems with discontinuous coefficients, nonlinearities, etc.

**Remark II.1** *For sufficiently regular $\underline{\sigma}$, for example $\underline{\sigma} \in (H^1(\Omega))^n$, the integral form (2.1) can be transformed into a differential form. First, we apply the* Divergence *theorem to get*

$$\int\limits_V \left( \frac{\partial}{\partial t} u + \nabla \cdot \underline{\sigma} + cu \right) \, dx = \int\limits_V f \, dx$$

*for any sufficiently regular $V \subset \Omega$. Then, using the last fact and the* localization *theorem (see [57]), we get*

$$\frac{\partial}{\partial t} u(x,t) + \nabla \cdot \underline{\sigma}(x,t) + cu = f(x,t). \tag{2.3}$$

Thus, we often write the conservation laws (2.1) in their differential form (2.3). One of the requirements for the numerical methods is stability for a class of problems that include problems with discontinuous coefficients and nonlinearities. For such problems, the solution $u$ and the coefficients of (2.1) are not smooth enough in order to derive the differential form (2.3) (see Remark II.1) and although we use it, we understand it in its weak sense (2.1), which is the more natural point of view of the conservation laws. Equation (2.1) can be interpreted as a local balance of the quantity $u(x,t)$. It is natural and in many cases essential that the numerical methods for such locally conservative problems also be locally conservative.

The finite volume methods, like the conservation laws that they usually discretize, are build on the same balance approach and hence they are locally conservative. To define the finite volume space discretization one splits the domain $\Omega$ into non-overlapping finite volumes $V_i$ and looks for $u_h(t) = u_h(\cdot, t)$ in a discrete space $S_h$ for each $t$, satisfying

$$\int\limits_{V_i} \frac{\partial}{\partial t} u_h \; dx + \int\limits_{\partial V_i} \underline{\sigma}_h \cdot \underline{n} \; ds + \int\limits_{V_i} c u_h \; dx = \int\limits_{V_i} f \; dx$$

for every $i$. Such definition yields a method that, as stated in the goals, may be used on general domains and on both structured and unstructured meshes. It is easy to compute and implement for complex problems, and as given by the finite volume theory, is stable even for difficult problems such as problems with nonlinearities and discontinuous coefficients.

Different choices for the discretization of $\Omega$ and $u$ give rise to various finite volume methods, which is discussed at the beginning of the finite volume discretization Section 2.3. Section 2.3 also presents a more rigorous definition of the finite volume element method that we use in the dissertation.

## 2.2. Model problems

Our main theoretical a posteriori error analysis results are for steady state problems, and more precisely, steady state convection-diffusion-reaction problems. Their general form is introduced in Subsection 2.2.1. The techniques and the theoretical results obtained for such problems are extended to linear time dependent problems. The general form of the time dependent problems that we consider is introduced in Subsection 2.2.2.

### 2.2.1. Steady state problems

We consider the following convection-diffusion-reaction problem : Find $u = u(x)$ such that

$$\begin{cases} Lu := -\nabla \cdot A\nabla u + \nabla \cdot (\underline{b}u) + cu &= f, \quad \text{in } \Omega, \\ u &= 0, \quad \text{on } \Gamma_D, \\ (-A\nabla u + \underline{b}u) \cdot \underline{n} &= g_N, \quad \text{on } \Gamma_N^{in}, \\ -A\nabla u \cdot \underline{n} &= 0, \quad \text{on } \Gamma_N^{out}. \end{cases} \tag{2.4}$$

Here $\Omega$ is a bounded polygonal domain in $R^n$, $n = 2, 3$ with boundary $\Gamma := \partial\Omega$, $A = A(x)$ is a $n \times n$ symmetric, bounded and uniformly positive definite matrix in $\Omega$, $\underline{n} = \underline{n}(x)$ is the unit vector pointing outward and normal to $\Gamma$, $\underline{b} = \underline{b}(x) = (b_1(x), \ldots, b_n(x))$ is a given vector function, $c = c(x)$ is the given absorption/reaction coefficient, and $f = f(x)$ is a given source function. We have also used the notation $\nabla u$ for the gradient of a scalar function $u$ and $\nabla \cdot \underline{b}$ for the divergence of a vector function $\underline{b}$ in $R^n$. The boundary of $\Omega$, $\Gamma$, is split into Dirichlet $\Gamma_D$ and Neumann $\Gamma_N$ parts. Furthermore, the Neumann boundary is divided into two parts: $\Gamma_N = \Gamma_N^{in} \cup \Gamma_N^{out}$, where $\Gamma_N^{in} = \{x \in \Gamma_N : \underline{n}(x) \cdot \underline{b}(x) < 0\}$ and $\Gamma_N^{out} = \{x \in \Gamma_N : \underline{n}(x) \cdot \underline{b}(x) \geq 0\}$. We assume that $\Gamma_D$ has positive measure.

We use the Hilbert space $H^1_D(\Omega) = \{v \in H^1(\Omega) : \ v|_{\Gamma_D} = 0\}$ and the standard $L^2$- and $H^1$-norms:

$$||u|| = (u,u)^{1/2}, \qquad ||u||_{1,\Omega} := ||u||_1 = \{(u,u) + (\nabla u, \nabla u)\}^{1/2},$$

where $(\cdot,\cdot)$ is the inner product in $L^2(\Omega)$.

We shall use the weak formulation of problem (2.4). First, we introduce the bilinear form $a(\cdot,\cdot)$, defined on $H^1_D(\Omega) \times H^1_D(\Omega)$ as

$$a(u,v) := (A\nabla u - \underline{b}u, \nabla v) + (cu, v) + \int\limits_{\Gamma_N^{out}} \underline{b} \cdot \underline{n}\ u\ v ds. \tag{2.5}$$

Then, we define the linear form $F(\cdot)$ on $H^1_D(\Omega)$ as

$$F(v) := (f, v) - \int\limits_{\Gamma_N^{in}} g_N v ds.$$

The problems that we discuss in the dissertation satisfy the following assumption.

**Assumption II.1** *The coefficients of problem (2.4) are such that the following conditions are satisfied:*

- *the bilinear form $a(\cdot,\cdot)$ is coercive in $H^1_D(\Omega)$, i.e., there is a constant $c_0 > 0$ s.t.*

$$a(u,u) \geq c_0||u||_1^2, \quad \forall u \in H^1_D(\Omega);$$

- *the bilinear form $a(\cdot,\cdot)$ is bounded in $H^1_D(\Omega)$, i.e., there is a constant $c_1 > 0$ s.t.*

$$a(u,v) \leq c_1||u||_1||v||_1, \quad \forall u,\ v \in H^1_D(\Omega);$$

- *the linear form $F(\cdot)$ is bounded in $H^1_D(\Omega)$, i.e., there is a constant $c_2 > 0$ s.t.*

$$F(u) \leq c_2||u||_1.$$

**Remark II.2** *A sufficient condition for the coercivity of the bilinear form is*

$$c(x) + \frac{1}{2}\nabla \cdot \underline{b}(x) \geq 0 \quad \textit{for all} \;\; x \in \Omega.$$

*Indeed, since* $\underline{b} \cdot \underline{n} \geq 0$ *on* $\Gamma_N^{in}$, $\underline{b} \cdot \underline{n} \leq 0$ *on* $\Gamma_N^{out}$, *and using the divergence theorem, we consequently get*

$$\begin{aligned} \int\limits_{\Gamma_N^{out}} \underline{b} \cdot \underline{n}\; u^2 ds &\geq \frac{1}{2} \int\limits_{\Gamma_N^{out}} \underline{b} \cdot \underline{n}\; u^2 ds \geq \frac{1}{2} \int\limits_{\Gamma} \underline{b} \cdot \underline{n}\; u^2 ds \\ &= \frac{1}{2} \int\limits_{\Omega} \nabla \cdot (\underline{b} u^2) dx = \frac{1}{2}((\nabla \cdot \underline{b})\; u, u) + (\underline{b}\; u, \nabla u). \end{aligned}$$

*Using (2.5), the above inequality, the stated sufficient condition, the uniform positive definiteness of the tensor* $A$, *and Poincare's inequality (*$\Gamma_D$ *has positive measure), we get coercivity*

$$a(u,u) \geq (A\nabla u, \nabla u) + (cu + \frac{1}{2}\nabla \cdot \underline{b}\; u, u) \geq C(\nabla u, \nabla u) \geq c_0 ||u||_1^2.$$

Problem (2.4) has the following weak form: Find $u \in H_D^1(\Omega)$ such that

$$a(u,v) = F(v) \;\; \text{for all} \;\; v \in H_D^1(\Omega). \tag{2.6}$$

Its solution is called weak (or generalized) solution of (2.4) in $H_D^1(\Omega)$. Assuming (II.1), Lax-Milgram lemma gives that the weak solution exists and is unique.

In our computations we also use the standard energy norm notation $||u||_a^2 = a(u,u)$.

### 2.2.2. Transient problems

We consider the following transient convection-diffusion-reaction problem : Find $u = u(x,t)$, $x \in \Omega$ and $t \in (0,T)$, $T > 0$ such that

$$\begin{cases} Lu := \frac{\partial}{\partial t}u - \nabla \cdot A\nabla u + \nabla \cdot (\underline{b}u) + cu &= f, \qquad x \in \Omega,\ t \in (0,T), \\ u &= 0, \qquad x \in \Gamma_D,\ t \in (0,T), \\ (-A\nabla u + \underline{b}u) \cdot \underline{n} &= g_N, \qquad x \in \Gamma_N^{in}(t),\ t \in (0,T), \\ -A\nabla u \cdot \underline{n} &= 0, \qquad x \in \Gamma_N^{out}(t),\ t \in (0,T), \\ u(x,0) &= u_0(x), \quad x \in \Omega. \end{cases} \tag{2.7}$$

Here, similarly to the steady-state case, $\Omega$ is a bounded polygonal domain in $R^n$, $n = 2, 3$ with boundary $\Gamma := \partial\Omega$, $A = A(x,t)$ is a $n \times n$ symmetric, bounded and uniformly positive definite matrix in $\Omega \times (0,T)$, $\underline{n}$ is the unit vector pointing outward and normal to $\Gamma$, $\underline{b} = \underline{b}(x,t) = (b_1(x,t), \ldots, b_n(x,t))$ is a given vector function, $c = c(x,t)$ is the given absorption/reaction coefficient, and $f = f(x,t)$ is a given source function. We have also used the notation $\nabla u$ for the space gradient of a scalar function $u$ and $\nabla \cdot \underline{b}$ for the space divergence of a vector function $\underline{b}$ in $R^n$. The boundary of $\Omega$, $\Gamma$, is split into Dirichlet $\Gamma_D$ and Neumann $\Gamma_N$ parts. Furthermore, the Neumann boundary is divided into two parts: $\Gamma_N = \Gamma_N^{in}(t) \cup \Gamma_N^{out}(t)$, where $\Gamma_N^{in}(t) = \{x \in \Gamma_N : \underline{n}(x) \cdot \underline{b}(x,t) < 0\}$ and $\Gamma_N^{out}(t) = \{x \in \Gamma_N : \underline{n}(x) \cdot \underline{b}(x,t) \geq 0\}$. The initial condition, $u_0(x)$, is a given function such that $u_0(x) = 0$ for $x \in \Gamma_D$. We assume that $\Gamma_D$ has positive measure.

To analyze the solvability and to derive the a posteriori error estimators in the subsequent chapters, as in the steady state case, we need the weak formulation of problem (2.7). We define the bilinear form $a(\cdot,\cdot)$ on $H_D^1(\Omega) \times H_D^1(\Omega)$ at time $t \in (0,T)$

as

$$a(u,v) := (A\nabla u - \underline{b}u, \nabla v) + (cu, v) + \int\limits_{\Gamma_N^{out}(t)} \underline{b}\cdot\underline{n}\ u\ vds$$

and the linear form $F(\cdot)$ on $H_D^1(\Omega)$ at time $t \in (0,T)$ as

$$F(v) := (f, v) - \int\limits_{\Gamma_N^{in}(t)} g_N vds.$$

Problem (2.7) has the following weak form: Find $u(x,t)$, $x \in \Omega$ and $t \in (0,T)$ such that

$$\begin{cases} (\dot{u}, v) + a(u,v) = F(v) \quad \text{for all } \ v \in H_D^1(\Omega),\ t \in (0,T) \\ u(x,0) = u_0(x) \quad \text{for all } \ x \in \Omega. \end{cases} \tag{2.8}$$

Here we use the notation $\dot{u} = \frac{\partial}{\partial t}u$.

## 2.3. Finite volume discretization

In this section we derive the finite volume element discretization of the model problems described in the previous section. This is the approximation that we use in our a posteriori error analysis.

There are many particular realizations of the general idea described above that will produce finite volume methods. The most general classification is obtained depending on the choice of : (1) the finite volumes and (2) the discrete space to which the approximate solution belongs. The domain $\Omega$ is meshed and depending on whether the finite volumes are the elements from the original splitting or volumes around the vertices of the original splitting, we have correspondingly *cell-centered* and *vertex-centered* finite volume methods. For the vertex-centered finite volumes, depending on whether the discrete space is piecewise constant over the finite volumes or piecewise linear over the original mesh, we have correspondingly *vertex-centered finite* *volume difference methods* or *vertex-centered finite volume element methods*. The cell-centered finite volumes can lead to *cell-centered finite volume difference methods* or *mixed methods* (when the flux $\underline{\sigma}$ is introduced as an additional unknown).

The stability of the finite volume methods often depends on the way the flux surface integrals are approximated. The most important example is for fluxes with strong convection part. For such cases a straightforward approximation of the flux integrals often yields oscillating solutions. Therefore, it is not surprising that different flux approximations also serve as benchmarks in the classification of the finite volume methods. The most popular approach is to use various upwind schemes (see [41, 46]). The upwinding makes the discrete problem stable but reduces the accuracy. Another approach to approximate the fluxes is to look for solution that is aligned to the streamlines of the convection part. This gives rise to the so called *exponentially fitted finite volume element methods* (see [41, 46]).

### 2.3.1. Discrete spaces and grids definitions

We assume that $\Omega$, a polygonal domain, is partitioned into triangles (in 2-D) or tetrahedra (in 3-D) called finite elements and denoted by $K$. The elements are considered to be closed sets and the splitting is denoted by $\mathcal{T}$. We assume that the partition $\mathcal{T}$ is aligned with $\Gamma_D$, $\Gamma_N^{in}$, $\Gamma_N^{out}$, and with the jumps in the coefficients $A$, $c$, and $f$. Also, we assume that $\mathcal{T}$ is locally quasi-uniform, that is for $K \in \mathcal{T}$, measure$(K) \leq C\rho(K)^n$ with a constant $C$ independent of the partition, where $\rho(K)$ is the radius of the largest ball contained in $K$. In the context of locally refined grids, this means that the neighboring finite elements are of approximately the same size whereas elements that are far away may have different size.

We introduce the set $N_h := \{p : p \text{ a vertex of element } K \in \mathcal{T}\}$ and define $N_h^0$ as the set of all vertices from $N_h$, except those on $\Gamma_D$. For a given vertex $x_i$, we denote by $\Pi(i)$ the index set of all neighbors of $x_i$ in $N_h$, i.e., all vertices that are connected to $x_i$ by an edge.

To derive the finite volume element approximation, we need the so-called dual partitioning of $\Omega$ into finite volumes. For a given finite element partitioning $\mathcal{T}$, we construct a dual mesh $\mathcal{T}^*$ (based upon $\mathcal{T}$), whose elements are called control volumes. In the finite volume methods there are various ways to introduce the control volumes. Almost all approaches can be described in the following general scheme. In each element $K \in \mathcal{T}$, a point $q$ is selected. For the 3-D case (and similarly in 2-D), on each of the four faces $\overline{x_i x_j x_k}$ of $K$ a point $x_{ijk}$ is selected and on each of the six edges $\overline{x_i x_j}$ a point $x_{ij}$ is selected. Then $q$ is connected to the points $x_{ijk}$, and in the corresponding faces, the points $x_{ijk}$ are connected to the points $x_{ij}$ by straight lines (see Figure 1). The control volumes are associated to the vertices $x_i \in N_h$. Control volume associated with vertex $x_i$ is denoted by $V_i$ and defined as the union of the "quarter" elements $K \in \mathcal{T}$, which have $x_i$ as a vertex (see Figure 1). The interface between two control volumes, $V_i$ and $V_j$, is denoted by $\gamma_{ij}$.

In our 3-D computations we use the case when $q$ is the medicenter of the element, $x_{ijk}$ are the medicenters of the corresponding faces, and $x_{ij}$ are the middle of the corresponding edges (as shown on Figure 1).

For the 2-D case, except choosing $q$ to be the medicenter of $K$, we use the construction of the control volumes in which the point $q$ is the circumcenter of the element $K$, i.e., the center of the circumscribed circle of $K$ and $x_{ij}$ are the midpoints of the edges of $K$. This type of control volume forms the so-called Voronoi meshes. Then obviously, $\gamma_{ij}$ are the perpendicular bisectors of the three edges of $K$ (see Figure 2). This construction requires that all finite elements are triangles of acute type, which we shall assume whenever such triangulation is used.

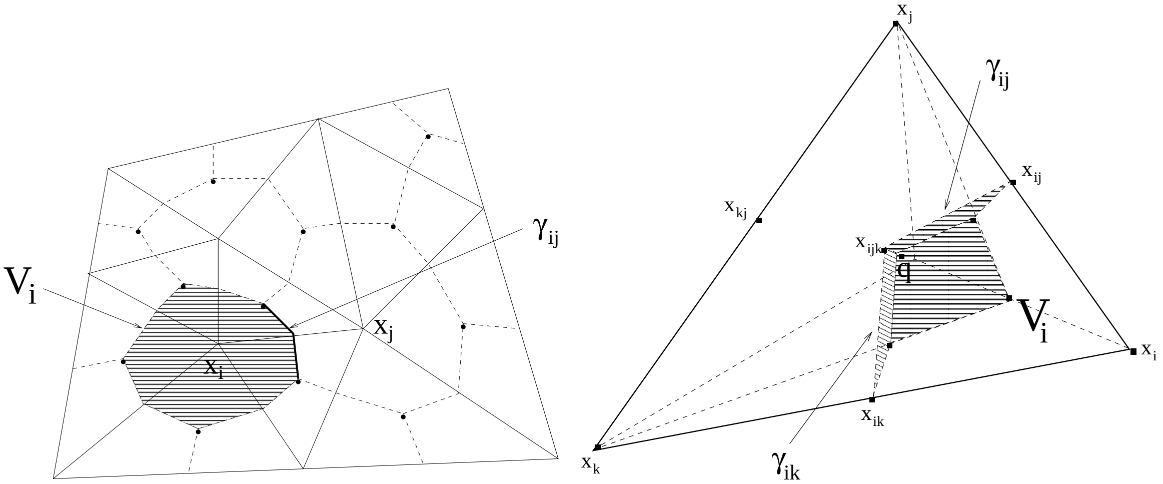


Fig. 1. Left: Finite element and finite volume partitions in 2-D; Right: Contribution from one element to control volume $V_i$, $\gamma_{ij}$ and $\gamma_{ik}$ in 3-D; Point $q$ is the element's medicenter and internal points for the faces are the medicenters of the faces.

We define the linear finite element space $S_h$ as

$$S_h = \{v \in C(\Omega) : \;\; v|_K \quad \text{is linear for all} \;\; K \in \mathcal{T} \;\; \text{and} \; v|_{\Gamma_D} = 0\}$$

and its dual volume element space $S_h^*$ by

$$S_h^* = \{v \in L^2(\Omega) : \;\; v|_V \quad \text{is constant for all} \;\; V \in \mathcal{T}^* \;\; \text{and} \; v|_{\Gamma_D} = 0\}.$$

Obviously, $S_h = \text{span}\{\phi_i(x) : \; x_i \in N_h^0\}$ and $S_h^*$ =span$\{\chi_i(x) : \; x_i \in N_h^0\}$, where $\phi_i$ is the standard nodal linear basis function associated with the node $x_i$ and $\chi_i$ is the characteristic function of the volume $V_i$. Let $I_h : C(\Omega) \rightarrow S_h$ be the interpolation operator and $I_h^* : C(\Omega) \rightarrow S_h^*$ and $P_h^* : C(\Omega) \rightarrow S_h^*$ be the piece-wise constant

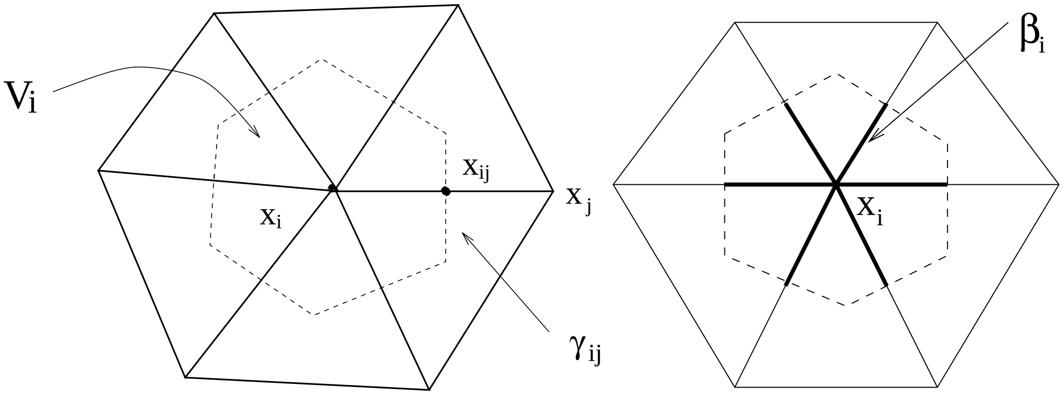


Fig. 2. Control volumes with circumcenters as internal points (Voronoi meshes) and interface $\gamma_{ij}$ of $V_i$ and $V_j$. The right picture shows the segments $\beta_i$ in bold.

interpolation and projection operators, respectively. That is

$$I_h u = \sum_{x_i \in N_h} u(x_i)\phi_i(x),$$

$$I_h^* u = \sum_{x_i \in N_h} u(x_i)\chi_i(x),$$

and

$$P_h^* u = \sum_{x_i \in N_h} \bar{u}_i \chi_i(x).$$

Here, $\bar{u}_i$ is the averaged value of $u$ over the volume $V_i$, i.e.,

$$\bar{u}_i = \frac{1}{|V_i|}\int_{V_i} u\,dx.$$

In fact $I_h$ makes also sense as an interpolation operator from $S_h^*$ to $S_h$, namely $I_h v^* \in S_h$ and $I_h v^*(x_i) = v^*(x_i)$.

### 2.3.2. Standard approximation on general grids

The discrete finite volume element approximation $u_h$ of (2.4) is the solution to the problem: Find $u_h \in S_h$ such that

$$a_h(u_h, v^*) := A(u_h, v^*) + C(u_h, v^*) = F(v^*), \text{ for all } v^* \in S_h^*. \tag{2.9}$$

Here the bilinear form $A(u_h, v^*)$ is defined on $S_h \times S_h^*$ and the linear form $F(v^*)$ on $S_h^*$, and are given by

$$A(u_h, v^*) := \sum_{x_i \in N_h^0} v_i^* \left\{ - \int\limits_{\partial V_i \setminus \Gamma_N} A \nabla u_h \cdot \underline{n} ds + \int\limits_{V_i} c u_h dx \right\} \tag{2.10}$$

and

$$F(v^*) := \sum_{x_i \in N_h^0} v_i^* \left\{ \int\limits_{V_i} f dx - \int\limits_{\partial V_i \cap \Gamma_N^{in}} g_N ds \right\}. \tag{2.11}$$

We use two different approximations for computing $C(u_h, v^*)$. The first one is a straightforward evaluation of $C(u_h, v^*)$:

$$C(u_h, v^*) = \sum_{x_i \in N_h^0} v_i^* \int\limits_{\partial V_i \setminus \Gamma_N^{in}} \underline{b} \cdot \underline{n} u_h ds, \quad u_h \in S_h, \quad v^* \in S_h^*. \tag{2.12}$$

Such approximation can be used for moderate convection fields and dominant diffusion. For large convection (or small diffusion) this approximation gives oscillating numerical results, which we would like to avoid. For such problems we are interested in approximation methods that produce solutions satisfying the maximum principle and are locally conservative. Such schemes are also known as monotone schemes (see, e.g. [36, 56]). A well-known sufficient condition for a scheme to be monotone is that the corresponding stiffness matrix is an $M$–matrix (see [59] p.182, p.260 and [56] p.202).

### 2.3.3. Upwind approximation on general grids

For problems with large convection we introduce upwind approximation that is locally mass conservative and gives the desired stabilization. It is done in the following way. We split the integral over $\partial V_i$ on integrals over $\gamma_{ij} = \partial V_i \cap \partial V_j$ (see Figure 1) and introduce out-flow and inflow parts of the boundary of the volume $V_i$. This splitting can be characterized by the quantities $(\underline{b}{\cdot}\underline{n})_+ = \max(0,\ \underline{b}{\cdot}\underline{n})$ and $(\underline{b}{\cdot}\underline{n})_- = \min(0,\ \underline{b}{\cdot}\underline{n})$, where $\underline{n}$ is the outer unit vector normal to $\partial V_i$. This approximation of the convection form $C(u_h, v^*)$ is denoted by $C^{up}(u_h, v^*)$ and is given by the following expression

$$\begin{aligned} C^{up}(u_h, v^*) = \sum_{x_i \in N_h^0} v_i^* \ \Bigg\{ \sum_{j \in \Pi(i)} \int_{\gamma_{ij}} (\underline{b} \cdot \underline{n})_+ u_h(x_i) + (\underline{b} \cdot \underline{n})_- u_h(x_j) \, ds \\ + \int_{\Gamma_N^{out} \cap \partial V_i} \underline{b} \cdot \underline{n} \, u_h(x_i) \, ds \Bigg\}. \end{aligned} \tag{2.13}$$

The finite volume element approximation $u_h$ of (2.4) becomes the solution to the problem: Find $u_h \in S_h$ such that

$$a_h^{up}(u_h, v^*) := A(u_h, v^*) + C^{up}(u_h, v^*) = F(v^*), \text{ for all } v^* \in S_h^*. \tag{2.14}$$

This is an extension of the classical upwind approximation of the convection term and is closely related to the discontinuous Galerkin approximation (see [38]) or to the Tabata scheme for Galerkin finite element method [64]. It is also related to the scheme on Voronoi meshes derived by Mishev in [47]. A different type of weighted upwind approximation on Voronoi meshes in 2-D has been studied in [2].

Concerning the solvability and the approximation properties of the introduced finite volume element discretizations we will make the following remark.

**Remark II.3** *Let $u$ and $u_h$ be the exact solution of (2.4) and the finite volume el-*

*ement approximation (2.9), respectively. Then the following a priori error estimate can be found in [16, 17, 30, 44]: There is a constant* $C > 0$, *independent of* $h$, *such that*

$$||u - u_h||_1 \leq Ch||u||_2. \tag{2.15}$$

*The estimate follows from an argument similar to Strang's second lemma in the finite element analysis (for upwind approximation, see [56], p. 218). This estimate follows easily from the technique used in the a posteriori error analysis in Chapter IV.*

### 2.3.4. Implicit finite volume discretization for transient problems

First, we discretize in space using the finite volume techniques described in subsections 2.3.2 and 2.3.3. Then, we discretize in time using the discontinuous Galerkin method. The discontinuous Galerkin method is based on finite element discretization in time using discrete space of piecewise-polynomials (see for example [26, 65]). Namely, for a non-negative integer $r$, we look for the discrete solution $u_h(x,t)$ in space

$$S_h^t := \{v(x,t) \ : \ v|_{I_n} = \sum_{i=0}^{r} v_i t^i, \quad v_i \in S_h, n = 1, \dots, M\},$$

where $I_n$, $n = 1, \dots, M$ are the time intervals, on which $(0,T)$ is discretized. We denote the time levels by $t_0, \dots, t_M$, so that $I_n = (t_{n-1}, t_n)$. The test space is

$$S_h^{*,t} := \{v^*(x,t) \ : \ v^*|_{I_n} \in S_h^*\}.$$

The discrete problem is : Find $u_h(x,t) \in S_h^t$ such that

$$\sum_{n=1}^{M} \int\limits_{I_n} ((\dot{u}_h, v^*) + a_h(u_h, v^*)) \ dt + \sum_{n=1}^{M} ([u_h^{n-1}], v^{*,n-1}) = \int\limits_0^T F(v^*) \ dt \tag{2.16}$$

for any $v^* \in S_h^{*,t}$, where the upper indices denote the time level, $[u_h^{n-1}]$ is the jump between the limit from above $u_{h,+}^{n-1} = \lim\limits_{\epsilon \to 0^+} u_h(x, t_{n-1} + \epsilon)$ and the limit from below

$u_{h,-}^{n-1} = \lim_{\epsilon \to 0^-} u_h(x, t_{n-1} + \epsilon)$, i.e., $[u_h^{n-1}] = u_{h,+}^{n-1} - u_{h,-}^{n-1}$. Also, we have denoted $u_{h,-}^0 = u_0$ and $v^{*,n} = v^*|_{I_n}$. Note that formulation (2.16) is equivalent to : Find $u_h(x,t) \in S_h^t$ such that

$$\int\limits_{I_n} ((\dot{u}_h, v^*) + a_h(u_h, v^*))\, dt + ([u_h^{n-1}], v^*) = \int\limits_{I_n} F(v^*)\ dt$$

for any $v^* \in S_h^*$, $n = 1, \ldots, M$.

In this dissertation we consider the case $r = 0$, which yields the following implicit scheme : Find $u_h^n \in S_h$ such that

$$(u_h^n - u_h^{n-1}, v^*) + \triangle t_n\ a_h(u_h, v^*) = \int\limits_{I_n} F(v^*)\ dt \tag{2.17}$$

for any $v^* \in S_h^*$, $n = 1, \ldots, M$, where $u_h^0 = u_0$ and $\triangle t_n = |I_n| = t_n - t_{n-1}$ is the local time step. This approximation is similar to the one that uses backward Euler discretization in time. The difference is that for the Euler case the integrals over $I_n$ are approximated by the mid-point rule.

# CHAPTER III

# ADAPTIVE METHODS

This chapter is an introduction to adaptive methods for partial differential equations. The two main goals of the adaptive methods are

- to get reliable control of the error due to the discretization of the considered PDE, and
- to use the available computational resources efficiently.

First of all, when discussing evaluation of the discretization error, we have to determine the measure in which the error to be evaluated. Possibilities are global $L^2$, $H^1$, or energy norms (see for example [5, 6, 9, 10, 26, 29, 67, 69], etc.). Also, in many cases the error quantity of physical interest may be local, like pointwise error or stress values, mean value over a domain of interest, line integrals of certain directional derivatives over the domain boundary, etc. In such cases the quantity of interest may be defined as a bounded linear functional and the error estimator obtained by using duality techniques [12, 29]. We illustrate this in Subsection 3.3.5.

In general, the class of applications will determine what are the quantities in which we need to control the error. In this dissertation we consider a posteriori error estimators for global $L^2$ norm, $H^1$ semi-norm, and energy norm for convection-diffusion-reaction problems. We denote the global norm of interest by $|||\cdot|||$ and define the exact discretization error $e$ as

$$e := u - u_h,$$

where $u$ is the solution of the weak formulation of the considered problem and $u_h$ the solution of its discretization.

A posteriori error estimators (or indicators) give a computable estimation for the error $e$ in the prescribed norm $|||\cdot|||$. The estimates are based on the discrete solution $u_h$ and the data associated with the PDE problem (coefficients, boundary conditions, and right hand side). The estimated error in the $|||\cdot|||$ norm is denoted by $\rho$ (or sometimes by $|||e_h|||$) and is usually obtained by locally evaluating the error for every element $K \in \mathcal{T}$. The locally computed values for the error over each $K$ are denoted by $\rho_K$ (or $|||e_h|||_K$), so that

$$\rho = \sqrt{\sum_{K \in \mathcal{T}} \rho_K^2}.$$

The a posteriori error indicators are used in the following setting. Suppose we want to find a discrete solution $u_h$ such that

- for a given error tolerance $e_{TOL}$
$$|||e||| \leq e_{TOL};$$
- the available computational resources are used efficiently.

To do so, we need error estimators satisfying

$$|||e||| \leq C|||e_h||| \tag{3.1}$$

and

$$c|||e_h|||_K \leq |||e|||_{\pi(K)}, \tag{3.2}$$

where $c$ and $C$ are known and independent of $h$ constants, $\pi(K)$ is a certain patch of neighboring to $K$ elements. The first inequality states that the error estimation is globally reliable. This means that if a local mesh refinement procedure insures that $C|||e_h||| \leq e_{TOL}$, then $|||e||| \leq e_{TOL}$ is guaranteed. The second inequality gives that the error estimation is locally efficient, which guarantees that the mesh is not over-refined in order to get the desired error control $|||e||| \leq e_{TOL}$. Summing (3.2) over all the elements $K \in \mathcal{T}$ gives global lower bound of the error. In general, the inequalities (3.1) and (3.2) are difficult to get. The estimators that are usually obtained involve additional terms on the right hand sides of the inequalities. These additional terms are of "higher" order under certain regularity requirements, in which case they can be neglected.

A property of the error estimators, relevant to inequalities (3.1) and (3.2), is their *efficiency index* (also called *effectivity index*), defined below.

**Definition III.1** *The efficiency index $\theta$ of an error estimator is defined as*

$$\theta := \frac{|||e_h|||}{|||e|||}.$$

If $c \leq \theta \leq C$ with constants $c$ and $C$ independent of the mesh size $h$, then we say that the error estimator is *equivalent* to the error. The estimator is *asymptotically exact* if $\lim_{h \to 0} \theta = 1$.

The rest of this chapter is organized as follows. First, in Section 3.1, we present an adaptive computational framework that guarantees error control in a global $||| \cdot |||$ norm. We present two generic algorithms:

- adaptive methods for steady state problems, and
- adaptive methods for transient problems.

Next section, Section 3.2, gives a numerical motivation of why and when the use of adaptive methods is important. The last section, Section 3.3, is about the mathematical aspects of the adaptive methods, or more precisely we make a review of the main a posteriori error estimators known from the finite element method. Some of the finite element analysis ideas, as stated in the introduction, are used to derive and theoretically justify a posteriori error estimators for the finite volume method (in Chapter IV).

## 3.1. A general adaptive methods framework

In this section we present the adaptive mesh refinement strategies for both steady state (Subsection 3.1.1) and transient problems (Subsection 3.1.2). Common for these two strategies are the following main components:

- finding efficiently computable stopping criterion guaranteeing error control, and
- mesh modification strategies in case stopping is not satisfied.

The generic steps in the algorithms are subject to a detailed study in Chapters IV and V.

### 3.1.1. Adaptive methods for steady-state problems

Algorithm III.1 describes the general adaptive strategy for steady-state problems, which we have used in our study and implementation.

**Algorithm III.1** *For a given finite element partition* $\mathcal{T}$*, desired error tolerance* $e_{TOL}$*, and norm in which the tolerance to be achieved, say* $|||\cdot|||$*, do the following :*

1. *compute the discrete finite volume approximation* $u_h$*;*
2. *for any finite element* $K \in \mathcal{T}$*, using a posteriori error analysis, compute the error estimate* $\rho_K$*;*
3. *mark those elements* $K$*, for which* $\rho_K \geq e_{TOL}/\sqrt{N}$*, where* $N$ *is the number of elements in* $\mathcal{T}$*;*
4. *if* $\rho > e_{TOL}$ *refine the marked elements;*

5. *additionally refine until a conforming mesh is reached;*
6. *repeat the above process until no elements have been refined.*

The above procedure yields error control and adapted mesh, which are the goals of the adaptive algorithms. The algorithm tries to equilibrate the error among the finite elements, which heuristically yields mesh refinement with overall accuracy uniform in the whole domain.

Variations of the algorithm are possible. For example, in step 3, one can mark elements for which $\rho_K \geq P e_{TOL}/\sqrt{N}$, where $P$ is a certain percent of $N$. Another alternative is always to mark a fixed percent of the elements with highest $\rho_K$. Also, step 5 can be missing for approximations allowing the presence of the so called *slave nodes*.

### 3.1.2. Adaptive methods for transient problems

We give below the general adaptive strategy for transient problems, which we have used in our implementation. The error control is in the global $L^2(\Omega)$ norm.

**Algorithm III.2** *For a given finite element partition* $\mathcal{T}$, *time interval* $(0,T)$, *and error tolerance* $e_{TOL}$, *such that*

$$\max_{t\in[0,T]} ||e(t)|| \leq e_{TOL}$$

*to be achieved, do the following :*

1. *set time step* $k = 0$ *and compute the discrete finite volume approximation* $u_h(t_k)$ *on time step* $t_0 = 0$ *as the* $L^2$ *projection of the initial condition into the discrete solution space using adaptive Algorithm III.1;*
2. *set an initial time step* $\triangle t$, *for example* $\triangle t = (\frac{|\Omega|}{N})^{2/n}$, $n = 2, 3$ *for correspondingly a* 2 *or* 3-$D$ *problem;*

3. *set* $k = k + 1$*;*

4. *compute* $u_h(t_k)$ *on time step* $t_k = t_{k-1} + \triangle t$*;*

5. *for any* $K \in \mathcal{T}$*, using a posteriori error analysis, estimate* $\rho_K$ *(*$||e_h(t_k)||_K$*) by evaluating its space* $\rho_K^S$ *and time* $\rho_K^T$ *components,* $\rho_K = \rho_K^S + \rho_K^T$ *(see the description below);*

6. *mark elements with* $\rho_K^S \geq \frac{e_{TOL}}{2\sqrt{N}}$ *for refinement and elements with* $\rho_K^S \leq \frac{Pe_{TOL}}{\sqrt{N}}$ *for derefinement, where* $N$ *is the number of elements in* $\mathcal{T}$*, and* $P$ *is a certain percent of* $N$*;*

7. *compute* $\rho^S := \sqrt{\sum_{K \in \mathcal{T}} (\rho_K^S)^2}$ *and* $\rho^T := \sqrt{\sum_{K \in \mathcal{T}} (\rho_K^T)^2}$*;*

8. *if* $\rho^S \leq \frac{e_{TOL}}{2}$ *and* $\rho^T \geq \frac{e_{TOL}}{2}$ *set* $\triangle t = \frac{\triangle t}{2}$ *and go to step* 4*;*

9. *if* $\rho^S \geq \frac{e_{TOL}}{2}$ *and* $\rho^T \leq \frac{e_{TOL}}{2}$ *refine the marked elements and go to step* 4*;*

10. *if* $\rho^S \geq \frac{e_{TOL}}{2}$ *and* $\rho^T \geq \frac{e_{TOL}}{2}$ *refine the marked elements, set* $\triangle t = \frac{\triangle t}{2}$ *and go to step* 4*;*

11. *if* $t_k < T$ *derefine the marked elements, set* $\triangle t = 2\triangle t$ *and go to step* 3.

As in the previous case, the above procedure yields error control and heuristically an optimal mesh. The a posteriori error estimators can be computed element by element. They have separate terms, giving indication for the error that is correspondingly due to the space and time discretization. Concerning the space refinement/derefinement, the algorithm tries to equilibrate $\rho_K^S$ among the finite elements, and concerning the time refinement/derefinement, the algorithm tries to equilibrate the total space and time components of the a posteriori error estimate. Such procedure heuristically yields a mesh refinement with overall accuracy uniform in time and space.

Variations of the algorithm, similar to the ones from the steady-state case, are possible.

## 3.2. A numerical motivation for adaptive grid refinement

Here we include a numerical motivation of why and when the use of adaptive methods is important. We consider three elliptic problems in 2-D with exact solutions $r^{4/3} sin\frac{4\theta}{3}$ (Problem 1), $r^{2/3} sin\frac{2\theta}{3}$ (Problem 2), and $r^{1/2} sin\frac{\theta}{2}$ (Problem 3), i.e., the problems have different regularity being correspondingly in $H^{1+4/3-\epsilon} \subset H^2$ (full elliptic regularity), $H^{1+2/3-\epsilon}$, and $H^{1+1/2-\epsilon}$, where $\epsilon > 0$ may be arbitrary small.

We start refining two identical meshes. The first one is refined uniformly and the second is refined locally (using a posteriori error analysis) until certain error tolerance is achieved on the resulting meshes and compare the size of the obtained discrete problems. Figure 3 summarizes the obtained numerical results.

One can see that for problems with full regularity the meshes obtained by uniform and adaptive refinement are identical, so there is no benefit in using local grid refinement. However, for problems with less then full regularity, one can highly benefit from the local refinement. To solve Problem 2 with error less then 0.00122 in $H^1$ norm one needs 8365 degrees of freedom in the adaptive case, compared to 169918 in the uniform refinement case. For more singular solutions, Problem 3, the difference becomes even bigger. To obtain error tolerance of 0.0385 in $H^1$ one needs 2508 degrees of freedom in the adaptive case, compared to again 169918 in the uniform refinement case.

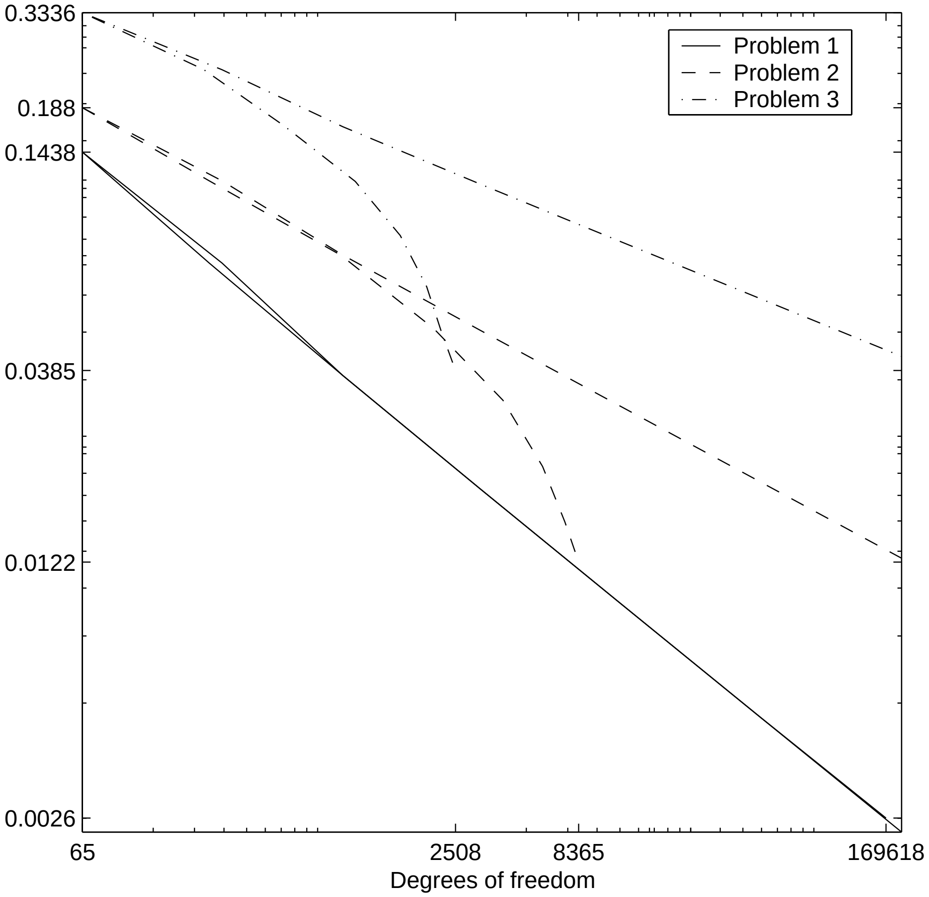


Fig. 3. Exact error $e_h$, for solutions $u \in H^{1+4/3-\epsilon}$ (Problem 1), $u \in H^{1+2/3-\epsilon}$ (Problem 2), and $u \in H^{1+1/2-\epsilon}$ (Problem 3), plotted against the degrees of freedom on a log-log scale for uniformly refined grid and locally refined grid based on a posteriori error estimates.

## 3.3. Review of some adaptive finite element methods

In the context of the finite element method there are two main techniques for the error reduction. One is based on increasing the order of the algebraic polynomials used in the approximation process, the so called "$p$-version" of the finite element method ($p$–refinement), while the other uses polynomials of the same degree, but adaptively refines the grid (by decreasing the mesh size $h$), the so called "$h$–version" adaptive refinement ($h$-refinement). We are concerned with evaluating the error and its reduction by $h$-refinement.

In this section we make a short review of the main a posteriori error analysis techniques for the finite element method. We give the ideas behind seven well known and widely used error indicators based on the $h$-version of the finite element method. Namely, we present estimators based on (1) local residuals, (2) Zienkiewicz-Zhu type averaging/projection, (3) solution of local Dirichlet/Neumann problems, (4) dual problems, (5) hierarchical refinement, (6) second derivatives, and (7) gradient indicator.

In Subsection 3.3.1 we give a general framework for deriving a posteriori error estimators. This framework is a basis for error estimators based on local residuals, dual problems, and solutions of local Dirichlet/Neumann problems, which have been extensively used and studied in the past 25 years (see [5, 6, 9, 10, 12, 26, 29, 67, 69]).

### 3.3.1. A general a posteriori error analysis framework

Here we present a general approach (as given in [29]) for deriving a posteriori error estimates for partial differential equations.

We consider the problem

$$Lu = f, \tag{3.3}$$

where $L : S \rightarrow S'$ is a linear operator on a Hilbert space $S$ with norm $||\cdot||_S$, $S'$ is the dual of $S$ with norm $||\cdot||_{S'}$, and $f \in S'$. We assume that $S \subset L^2 \subset S'$. The Galerkin formulation of (3.3) is : find $u_h \in S_h \subset S$ such that

$$< Lu_h, v_h > \;=\; < f, v_h > \qquad \forall v_h \in S_h, \tag{3.4}$$

where $S_h$ is a finite dimensional subspace of $S_h$ and $< f, v >$, often called duality pairing between $S'$ and $S$, denotes the value of the continuous linear functional $f \in S'$ taken at $v \in S$.

We solve the Galerkin approximation given above, and try to get an estimation for the error $e = u - u_h$ in the global $L^2$ norm $||\cdot||$. Since $L$ is linear on $S$ we get $Lu - Lu_h = Le$ and consequently the Galerkin orthogonality

$$< Le, v_h > \;=\; 0 \quad \forall v_h \in S_h.$$

The derivation of the error estimator has the following main steps:

- define the residual $R := f - Lu_h$ and find $v \in S$ as the solution of the dual problem
  $$< \phi, L^* v > \;=\; (e, \phi) \quad \text{for any } \phi \in S;$$
- use the Galerkin orthogonality to consequently get
  $$\begin{aligned} ||e||^2 &= \;< e, L^* v > \;=\; < Le, v > \;=\; < Le, v - v_h > \\ &= \;< R, v - v_h > \;\leq\; ||R||_{S'} \, ||v - v_h||_S \end{aligned}$$
  for any $v_h \in S_h$;
- choose $v_h \in S_h$ such that
  $$||v - v_h||_S \leq C_i h^\alpha \; ||v||_{H^\alpha},$$

where $C_i$ is some constant, $h$ is the mesh size of the discretization, $0 < \alpha$ and $||\cdot||_{H^\alpha}$ is the norm in Hilbert space $H^\alpha$;

- prove stability estimate for the dual problem

$$||v||_{H^\alpha} \leq C_s||e||,$$

where $C_s$ is some stability constant;

- combine the above to finally get the a posteriori error estimate

$$||e|| \leq C_i C_s h^\alpha ||R||_{S'}.$$

This general idea is implemented in several of the a posteriori error indicators explained below.

**Remark III.1** *Note that although the idea is presented for error estimate in the* $L^2$ *norm one can use it with small modifications for global* $||\cdot||_S$ *norm, or proper "energy" norm relateded to the differential operator* $L$.

In the following subsections we define and give the ideas behind seven a posteriori error estimators. For simplicity we describe these estimators for the Dirichlet problem

$$\begin{cases} -\nabla \cdot A\nabla u &= f \quad \text{in } \Omega \\ \qquad\quad u &= 0 \quad \text{on } \partial\Omega, \end{cases} \tag{3.5}$$

where, as we set before, $\Omega$ is a bounded polygonal domain in $R^n$, $n = 2, 3$, $A = A(x)$ is a $n \times n$ symmetric, bounded and uniformly positive definite matrix in $\Omega$, and $f = f(x)$ is a given source function. The weak formulation of problem (3.5), find $u \in H_0^1(\Omega)$ such that

$$a(u, v) := (A\nabla u, \nabla v) = (f, v) \quad \text{for any} \quad v \in H_0^1(\Omega),$$

is discretized and solved by the Galerkin finite element method.

### 3.3.2. Residual based (RB) error estimator

Estimates of this type were first introduced by Babuska and Rheinboldt [5, 6]. Here we describe the residual method (denoted by *RB Refinement*) as given by Becker et al. in [11, 13] and Verfürth in [68]. This method is based on equilibrating certain residuals. The Galerkin orthogonality $a(e, v_h) = 0$ for all $v_h \in S_h$ and integration by parts yield the identity

$$a(e,v) = \sum_{K\in\mathcal{T}} \left\{ (f + \nabla \cdot A\nabla u_h, v - v_h)_K - \frac{1}{2}(\underline{n} \cdot [A\nabla u_h], v - v_h)_{\partial K} \right\}$$

for any $v \in H_0^1$, where $[A\nabla u_h]$ denotes the jump of $A\nabla u_h$ across the element boundary, $(u, v)_K = \int\limits_K uv\ dx$, $(u, v)_{\partial K} = \int\limits_{\partial K} uv\ ds$, and $v_h \in S_h$ is a suitable approximation of $v$. We apply Hölder's inequality on each element to get

$$a(e,v) \le \sum_{K\in\mathcal{T}} \rho_K\ \omega_K(v), \text{ where} \tag{3.6}$$

$$\begin{aligned} \rho_K &:= h_K\|f + \nabla \cdot A\nabla u_h\|_K + \tfrac{1}{2}h_K^{1/2}\|\underline{n} \cdot [A\nabla u_h]\|_{\partial K}, \\ \omega_K(v) &:= max\left\{ h_K^{-1}\|v - v_h\|_K, h_K^{-1/2}\|v - v_h\|_{\partial K} \right\}. \end{aligned}$$

Next, one should use the local approximation properties of the space $S_h$ to estimate $\omega_K(v)$. Namely, we use the concept of quasi interpolation (see, e.g. [14, 18, 24]) and proceed as follows. Let $\pi(K)$ be a patch of all finite elements that share a vertex with $K$. Using the local $L^2$-projection (see [14], formulas (3.2) and (3.3)), we can find $v_h \in S_h$ such that

$$h_K^{-1}\|v - v_h\|_K + h_K^{-1/2}\|v - v_h\|_{\partial K} + \|\nabla v_h\|_K \le C_i\|\nabla v\|_{\pi(K)}, \tag{3.7}$$

i.e., $\omega_K(v) \leq C_i||\nabla v||_{\pi(K)}$. We take $v = e$ in (3.6), apply the discrete Schwarz' inequality, use the derived for $\omega_K(e)$ bound, and obvious manipulations to finally get the following a posteriori upper bound for the error

$$||e||_1 \leq CC_I \left( \sum_{K \in \mathcal{T}} \rho_K^2 \right)^{1/2}, \tag{3.8}$$

where $C$ depends on the coercivity constant of the form $a(\cdot, \cdot)$ and $C_I = \max\limits_{K \in \mathcal{T}} C_{I,K}$.

### 3.3.3. Zienkiewicz-Zhu (ZZ) type estimators

This type error estimators were introduced by Zienkiewicz and Zhu [72, 75], and are also known as ZZ estimators. They are based on easy to compute local projections or averaging, making them very appealing for implementation in the adaptive mesh refinement software. Their efficiency for various elliptic problems has been numerically confirmed in many practical applications (see, for example, the references in the survey paper [73]). Theoretical study can be found, for example, in [54] and the literature cited therein.

The ZZ indicator uses the diffusive flux $\underline{\sigma} \equiv -A\nabla u$ and its finite element approximation $\underline{\sigma}_h = -A\nabla u_h$. A recovered flux $\underline{\sigma}_h^* \in (S_h)^n$ is computed by smoothing the discontinuous along the element boundaries numerical flux $\underline{\sigma}_h$. The smoothing may be done by nodal averaging of $\underline{\sigma}_h$, i.e., $\underline{\sigma}_h^*$ is computed at a given node by averaging $\underline{\sigma}_h$ from the elements that share the considered node, or $L^2(\Omega)$ projection of $\underline{\sigma}_h$ into $(S_h)^n$. The computation of the global $L^2(\Omega)$ projection is expensive and one often uses "lumping" of the mass matrix to define $\underline{\sigma}_h^*$ at vertices $x_i$ of the mesh as

$$\underline{\sigma}_h^*(x_i) = -\frac{1}{|K_i|} \sum_{K \subset K_i} |K| \; A\nabla u_h|_K,$$

where $K_i$ is the union of elements sharing vertex $x_i$. Heuristically, the continuous at the vertices $\underline{\sigma}_h^*$ is a better approximation to $\underline{\sigma}$ than $\underline{\sigma}_h$ and is used to represent $\underline{\sigma}$, for example in

$$||e||_a^2 = (A\nabla e, \nabla e) = (A^{-1}(\underline{\sigma} - \underline{\sigma}_h), \underline{\sigma} - \underline{\sigma}_h),$$

to get the error indicator $(A^{-1}(\underline{\sigma}_h^* - \underline{\sigma}_h), \underline{\sigma}_h^* - \underline{\sigma}_h)$ in the energy norm.

### 3.3.4. Error estimators based on solution of local Dirichlet/Neumann problems

Estimators based on solution of certain local Dirichlet/Neumann problems are well known and studied (see [5, 10, 67, 69]). They provide, up to higher order terms, local lower and global upper bounds for the error. The proof is usually based on establishing equivalence to the residual type error estimators.

The local Neumann problems are defined for every $K \in \mathcal{T}$ as: find $u_N$ such that

$$\begin{cases} -\nabla \cdot A\nabla u_N &= f + \nabla \cdot A\nabla u_h \quad \text{in } K \\ u_N &= 0 \quad \text{on } \partial K \cap \partial\Omega, \\ \nabla u_N \cdot \underline{n} &= [A\nabla u_h \cdot n] \quad \text{on } \partial K \setminus \partial\Omega. \end{cases} \tag{3.9}$$

The local Dirichlet problems are defined for every internal vertex $x_i$ of the mesh $\mathcal{T}$ as: find $u_D$ such that

$$\begin{cases} -\nabla \cdot A\nabla u_D &= f \quad \text{in } K_i \\ u_D &= u_h \quad \text{on } \partial K_i, \end{cases} \tag{3.10}$$

where $K_i$ is the set of all elements $K \in \mathcal{T}$ sharing vertex $x_i$.

The local Neumann problems are discretized and solved in spaces $S_K$, consisting of the so called “bubble” functions, which are polynomials associated with the edge/face/element midpoints and are 0 at the elements’ vertices. Also, the functions in $S_K$ are 0 on $\partial\Omega$. If we denote the discrete solution by $u_{N,h}$, the error estimator is given by $||\nabla u_{N,h}||_K$ for every $K \in \mathcal{T}$.

The local Dirichlet problems are discretized and solved in spaces $S_i$, consisting of bubble functions on the elements $K \subset K_i$, and 0 on $\partial K_i$. If we denote the discrete solution by $u_{D,h}$, the error estimator is given by $||\nabla u_{D,h} - \nabla u_h||_{K_i}$ for every vertex $x_i$ of the mesh $\mathcal{T}$.

### 3.3.5. Error estimators based on dual problems

The asymptotic exactness of the RB estimator from Subsection 3.3.2 can be lost when applying the Hölder's inequality in (3.6). For example, this is the case for problems with strongly varying coefficients, convection dominated problems, etc. Also, the estimator may not be appropriate for controlling local quantities of the error (point values, line integrals, etc.). Such deficiency is inherited for almost all *Residual Based* error indicators based on local computations and is caused by the fact that they do not control the error propagation or do it partially, of course, for reasons of computational efficiency. The sharpness of the error estimates could be improved and the question is how much one is willing to spend on the computations to find the exact error.

In the residual method based on duality one increases the sharpness of the error estimators (which may be designed for different quantities of the error) by solving dual problems for the quantity of interest and using the obtained solution to compute better weighting factors $\omega_K$ in (3.6). The general idea is as follows (also, see for example [12, 29]). Consider the dual problem: Find $\tilde{e} \in H_0^1(\Omega)$ such that

$$a(v, \tilde{e}) = J(v) \text{ for any } v \in H_0^1(\Omega),$$

where $J(\cdot)$ is a bounded linear error functional defined on $H_0^1(\Omega)$, which combined with the assumptions for the bilinear form (II.1) guarantees exitance and uniqueness of solution $\tilde{e}$ by the Lax-Milgram lemma. For example, if one wants to control the error at point $x$, one chooses $J(v) := \frac{1}{|B_\epsilon|} \int\limits_{B_\epsilon} v dx$, where $B_\epsilon$ is a ball around $x$ of radius $\epsilon$. Substituting $v = e$ and $\epsilon = e_{TOL}$ into the above equation yields $J(e) = e(x) + \mathcal{O}(e_{TOL}^2)$ for $e$ sufficiently smooth. Using the same approach as in (3.6) gives

$$|J(e)| \leq \sum_{K \in \mathcal{T}} \rho_K \omega_K(\tilde{e}).$$

The easiest way to estimate the weights $w_K(\tilde{e})$ is to use one of the a priori error estimates

$$\omega_K(\tilde{e}) \leq C_{I,K} ||\nabla \tilde{e}||_{\pi(K)}, \quad \omega_K(\tilde{e}) \leq C_{I,K} h_K ||\tilde{e}||_{H^2(K)}.$$

The first inequality follow from (3.7). The second inequality is valid if the exact solution is sufficiently regular and to prove it we use the approximation properties of the nodal interpolant.

Improvements, which lead to another increase in the computational cost, deal with techniques for direct evaluation of the residual in (3.6), i.e., the quantity before applying Hölder's inequality (for more details we refer to [11]).

### 3.3.6. Hierarchical (HB) refinement

For the hierarchical bases method [9] the finite element space $S_h$ is enriched by certain hierarchical basis functions. The enriched space is denoted by $\bar{S}_h = S_h \oplus B_h$, where $B_h$ is the span of the additional basis functions. Usually $B_h$ is composed of bubble functions.

The HB error estimator $\bar{e_h} \in \bar{S}_h$ is defined as the solution of

$$a(\bar{e_h}, v) = (f, v) - a(u_h, v) \quad \text{for all } v \in B_h. \tag{3.11}$$

It can be proved, under two assumptions, that $\bar{e_h}$ is equivalent to the error $e_h$ in $|| \cdot ||_a$ up to constants of order one. The assumptions are

1. *Saturation assumption* : denote by $\bar{u}_h \in \bar{S}_h$ the solution of $a(\bar{u}_h, v) = (f, v)$ for any $v \in \bar{S}_h$ and assume

$$||u - \bar{u}_h||_1 \leq \beta ||u - u_h||_1 \quad ,$$

   where $\beta < 1$ is independent of $h$ (which is usually satisfied because of the higher order approximation).

2. *Strengthen Cauchy inequality* : assume that for all $v \in S_h$ and $w \in B_h$

$$a(v, w) \leq \gamma ||v||_1 \; ||w||_1 \; ,$$

   where $\gamma < 1$ is independent of $h$ (which is true for $B_h$ being bubble functions).

**Remark III.2** *Computationally, to simplify the evaluation of the error estimator even further, one can replace the symmetric and positive definite (s.p.d.) bilinear form $a(\cdot,\cdot)$ in (3.11) with another s.p.d. bilinear form $b(\cdot,\cdot)$ that is easier to invert and is "equivalent" to $a(\cdot,\cdot)$. The equivalence is in the sense that there are independent of $h$ positive constants $c_0$ and $c_1$ such that $c_0 \leq \frac{a(v,v)}{b(v,v)} \leq c_1$ for any $v \in B_h$. For more details see ([9]).*

### 3.3.7. Error estimator based on second derivative (SD)

This technique (see [28]) aims to control the gradient of the error in the maximum norm (denoted by SD refinement). The error control is based on an optimal *a priori* estimate of the form

$$||\nabla(u - u^h)||_{\infty,\Omega} \leq C_0 \max_{K \in \mathcal{T}} h_K \sup_{|\alpha|=2,\; x \in K} |D^\alpha u(x)| \; , \tag{3.12}$$

where $D^\alpha$ is the multi–index notation for derivatives of order $|\alpha|$, $C_0$ is a positive constant, assumed to be known. The goal to control the right hand side of (3.12) with a tolerance $\delta$ leads to a choice of $h_K$ such that $C_0 h_K \sup_{|\alpha|=2,\ x\in K} |D^\alpha u(x)| \sim \delta$. The quantity $\sup_{|\alpha|=2,\ \ x\in K} |D^\alpha u(x)|$, denoted as $D^2_H(u^h;K)$, is approximated locally by using local difference quotients of $u^h$ :

$$D^2_H(u^h;K) = \max_{|\alpha|=1,\ K'\in N(K)} \frac{|D^\alpha u^h(P_K) - D^\alpha u^h(P_{K'})|}{|P_K - P_{K'}|} \quad , \tag{3.13}$$

where $N(K)$ is the set of neighboring to $K$ finite elements, $P_K$ is the center of gravity of $K$.

### 3.3.8. Error indicator based on the gradient

This is a very simple and easy to implement error indicator. It is defined locally for every element $K \in \mathcal{T}$ by

$$\rho_K := ||\nabla u_h||_K.$$

It leads to mesh refinement at places where the $L^2$ norm of of the discrete gradient $\nabla u_h$ is "big". It does not give indication for the global error.

# CHAPTER IV

# ADAPTIVE FINITE VOLUME METHODS

The previous chapter contains an overview of the main a posteriori error analysis techniques, which are used in the finite element method. Here, we continue the analysis by extending the existing techniques to the finite volume element method. Again, we achieve the goals of getting reliable error control and efficient use of the available computational resources by local mesh refinement based on a posteriori error analysis. We construct a posteriori error estimators that control the error in global $L^2$, $H^1$, and energy norms, and prove their reliability (3.1) and efficiency (3.2).

Since $u_h \in S_h \subset H^1_D(\Omega)$, the problem of finding the exact error $e = u - u_h$ has the following weak formulation : Find $e \in H^1_D(\Omega)$ such that

$$a(e, v) = F(v) - a(u_h, v) := R(v) \quad \text{for all} \quad v \in H^1_D(\Omega). \tag{4.1}$$

Most of the residual type error estimators solve (4.1) approximately, using an enriched finite dimensional space $S$, $S_h \subset S \subset H^1_D(\Omega)$. Such space is usually obtained by adaptively refining the grid $\mathcal{T}$ (the so called $h$-refinement) or by increasing the order of the algebraic polynomials used in the approximation process (the so called $p$-refinement). Such global solution technique is computationally expensive and is usually replaced by solving the problem locally. We saw different instances of implementing this idea in Section 3.3, while considering the finite element approximations.

Throughout this chapter we use the notations and the assumptions introduced in Chapter II.

We begin our analysis with the standard steady state approximations (Section 4.1). We first derive a finite volume error representation (Subsection 4.1.1). The representation uses residuals as in the finite element case. We consider residual type error estimators (Subsection 4.1.2), Zienkiewicz-Zhu type error estimators (Subsection 4.1.3), estimators based on local Dirichlet/Neumann problems (Subsection 4.1.4), and estimators based on dual problems (Subsection 4.1.5). Furthermore, in Section 4.2, we consider the case of upwind approximations. In Section 4.3 we discuss the error analysis for transient problems.

## 4.1. Estimators for standard steady-state approximations

Here we consider steady state problems, discretized by the standard finite volume method, described in Subsection 2.3.2. First, we derive and analyze error estimators in the energy norm, which in our case is equivalent to the $H^1$ norm. The results for $L^2$ norm are obtained in Subsection 4.1.5 trough duality techniques.

We will use the $\mathcal{T}^*$-piecewise integral mean $\overline{e} = P_h^* e$ of the error $e = u - u_h$. We denote by $\mathcal{E}$ the set of all interior edges/faces in $\mathcal{T}$ respectively in two/three dimensions. For edges $\overline{x_i x_j}$ we denote

$$\beta_{ij} := V_i \cap \overline{x_i x_j}.$$

Also, for $x_i \in N_h^0$, let

$$\beta_i := V_i \cap \mathcal{E}$$

(see Figure 2 for the 2D case). For any edge $E \in \mathcal{E}$ and flux $\underline{\sigma}$ let $[\underline{\sigma}] \cdot \underline{n}$ denote the jump of $\underline{\sigma}$ across $E$ in normal to the edge $E$ direction $\underline{n}$. The orientation of $\underline{n}$ is not important as long as the jump is in the same direction. In general, if $\underline{n}$ is present in boundary integrals, it will denote the outward unit normal vector to that boundary. The orientation will either not matter, which will be the case of computing quantities involving flux jumps, or will be pointing outward and normal to $\Gamma$, which will be the case of computing integrals over $\Gamma$ or part of $\Gamma$.

In Chapter II we introduced the notation $||\cdot|| := ||\cdot||_{L^2(\Omega)}$. Furthermore, $||\cdot||_{L^2(K)}$ and $||\cdot||_{L^2(\gamma)}$ will denote correspondingly the $L^2$ norm over elements $K$ and surfaces $\gamma \subset (\mathcal{E} \cup \Gamma)$. The corresponding inner-products will be denoted by $(\cdot,\cdot)_K$ and $< \cdot,\cdot >_\gamma$. If the integrated quantity is defined element-wise, for example the discontinuous over the elements $\nabla u_h$, the integrals should be considered as taken element-wise.

The meshes that we obtain in the refinement process are locally refined. With every element $K \in \mathcal{T}$, edge $E \in \mathcal{E}$, and volume $V_i \in \mathcal{T}^*$ we associate local, and proportional to their size, mesh size and denote it correspondingly by $h_K$, $h_E$, and $h_i$. Note that since the mesh is locally quasi-uniform (assumption made in Section 2.3), there exist global positive constants $c$ and $C$ such that if $x_i$ and $E$ are vertex and edge of an element $K$, we have the inequalities

$$c h_i \leq h_K \leq C h_i \quad \text{and} \quad c h_E \leq h_K \leq C h_E. \tag{4.2}$$

We introduce a global discontinuous mesh size function $h(x)$, $x \in \Omega$ that takes values $h_K$, $h_E$, and $h_i$ depending on if $x \in K$, $x \in E$, or $x = x_i$. Also, concerning the local mesh sizes, if we have inequality of the form

$$a \leq Cb,$$

with $C$ not depending on the local mesh size function $h$, then we will often write it for simplicity as

$$a \lesssim b.$$

Our analysis of the residual a posteriori error estimates is based on the abstract framework in Section 3.1. The abstract linear operator $L$ and the duality pairing between $S$ and $S'$ in the Galerkin formulation (3.4) are given by the finite volume element formulation (2.9). An explicit formulation of the abstract linear operator $R$

will be given in the next subsection. The equivalent "interpolation type" abstract estimate (the one involving the constant $C_i$) is analyzed in Subsection 4.1.2.1.

### 4.1.1. Error representation in the energy norm

Lemma IV.1 proves an important for our error analysis representation for the energy of the error. The error is given as sums of local residuals over : (1) the elements $K \in \mathcal{T}$, (2) the internal edges $E \in \mathcal{E}$, and (3) the Neumann boundary edges $E \in \Gamma_N$. It is used in the a posteriori error analysis to give local indication on "how far" is the approximation $u_h$ from the exact solution $u$.

**Lemma IV.1** *Let the coefficients of the convection-diffusion-reaction problem (2.4) be such that assumption (II.1) is satisfied. Then the energy norm* $||\cdot||_a$ *of the error* $e = u - u_h$*, where* $u$ *is the solution of (2.6) and* $u_h$ *the solution of (2.9), is represented as*

$$\begin{aligned} ||e||_a^2 = \sum_{K\in\mathcal{T}} \int_K (f - \nabla\cdot\underline{\sigma}_h - cu_h)(e-\overline{e})\ dx - \sum_{E\in\mathcal{E}} \int_E ([\underline{\sigma}_h]\cdot\underline{n})(e-\overline{e})\ ds \\ - \sum_{E\in\Gamma_N^{in}} \int_E (g_N - \underline{\sigma}_h\cdot\underline{n})(e-\overline{e})\ ds - \sum_{E\in\Gamma_N^{out}} \int_E A\nabla u_h\cdot\underline{n}(e-\overline{e})\ ds, \end{aligned} \tag{4.3}$$

*where* $\overline{e}$ *is the* $\mathcal{T}^*$*-piecewise integral mean* $P_h^* e$*.*

*Proof.* We take $v = e \in H_D^1(\Omega)$ in (2.6) and use the definition of $a(\cdot,\cdot)$ in (2.5) to conclude that

$$\begin{aligned} a(e,e) &= a(u,e) - a(u_h,e) \\ &= \sum_{K\in\mathcal{T}} \int_K (f - cu_h)e\ dx + \sum_{K\in\mathcal{T}} \int_K \underline{\sigma}_h\cdot\nabla e\ dx \end{aligned}$$

$$- \sum_{E\in\Gamma_N^{in}} \int_E g_N \; e \; ds - \sum_{E\in\Gamma_N^{out}} \int_E (\underline{b}\cdot\underline{n}) u_h \; e \; ds.$$

We integrate the terms in the second sum on the right hand side by parts on each element $K \in \mathcal{T}$ to get

$$\int_K \underline{\sigma}_h \cdot \nabla e \; dx = \int_{\partial K} (\underline{\sigma}_h \cdot \underline{n}) e \; ds - \int_K e \nabla \cdot \underline{\sigma}_h \; dx.$$

Taking sum over all elements yields the jump contributions $[\underline{\sigma}_h]\cdot\underline{n}$ on $E \in \mathcal{E}$ (see below) and eventually proves

$$\begin{aligned} a(e,e) \; &= \; \sum_{K\in\mathcal{T}} \int_K (f - \nabla\cdot\underline{\sigma}_h - cu_h) e \; dx - \sum_{E\in\mathcal{E}} \int_E ([\underline{\sigma}_h]\cdot\underline{n}) e \; ds \\ & \quad - \sum_{E\in\Gamma_N^{in}} \int_E (g_N - \underline{\sigma}_h \cdot \underline{n}) e \; ds - \sum_{E\in\Gamma_N^{out}} \int_E A\nabla u_h \cdot \underline{n} \; e \; ds. \end{aligned} \tag{4.4}$$

Note that this is the assertion if the argument $e$ is replaced by $(e - \overline{e})$, i.e., it remains to see that the preceding right-hand side vanishes if $e$ is replaced by $\overline{e}$. Using the finite volume formulation (2.9)-(2.12) we get the following equality for each control volume $V_i$

$$\int_{\partial V_i \setminus \Gamma_N} \underline{\sigma}_h \cdot \underline{n} \; ds = \int_{V_i} (f - cu_h) \; dx - \int_{\partial V_i \cap \Gamma_N^{out}} (\underline{b}\cdot\underline{n}) u_h \; ds - \int_{\partial V_i \cap \Gamma_N^{in}} g_N \; ds. \tag{4.5}$$

We modify the integral on the right hand side by applying the Gauß divergence theorem to each non-void $K \cap V_i$, $K \in \mathcal{T}$, to get

$$\int_{\partial V_i \setminus \Gamma_N} \underline{\sigma}_h \cdot \underline{n} \; ds = \sum_{K\in\mathcal{T}} \int_{K\cap V_i} \nabla \cdot \underline{\sigma}_h \; dx + \int_{\beta_i} [\underline{\sigma}_h]\cdot\underline{n} \; ds - \int_{\partial V_i \cap \Gamma_N} \underline{\sigma}_h \cdot \underline{n} \; ds.$$

The difference of the preceding two identities is multiplied by $\overline{e}(x_i)$ and summed over all control volumes. This results in

$$\begin{aligned} 0 &= \sum_{K\in\mathcal{T}} \int_K (f - \nabla\cdot\underline{\sigma}_h - cu_h)\overline{e}\, dx - \sum_{E\in\mathcal{E}} \int_E ([\underline{\sigma}_h]\cdot\underline{n})\overline{e}\, ds \\ &\quad - \sum_{E\in\Gamma_N^{in}} \int_E (g_N - \underline{\sigma}_h\cdot\underline{n})\overline{e}\, ds - \sum_{E\in\Gamma_N^{out}} \int_E A\nabla u_h\cdot\underline{n}\,\overline{e}\, ds. \end{aligned}$$

The combination of the last result with equality (4.4) concludes the proof. □

The quantities

$$\begin{aligned} R_K(x) &:= (f - \nabla\cdot\underline{\sigma}_h - cu_h)\Big|_{x\in K} \\ R_E(x) &:= ([\underline{\sigma}_h]\cdot\underline{n})\Big|_{x\in\mathcal{E}} \\ R_E^{in}(x) &:= (g_N - \underline{\sigma}_h\cdot\underline{n})\Big|_{x\in E}, \quad \text{for } E\in\Gamma_N^{in} \\ R_E^{out}(x) &:= (A\nabla u_h\cdot\underline{n})\Big|_{x\in E}, \quad \text{for } E\in\Gamma_N^{out} \end{aligned} \tag{4.6}$$

have computable $L^2$ norms $||R_K||_{L^2(K)}$, $||R_E||_{L^2(E)}$, $||R_E^{in}||_{L^2(E)}$, and $||R_E^{out}||_{L^2(E)}$, which are used in the definition of the residual type error estimator from Subsection 4.1.2. Using these notations and Lemma IV.1, the linear operator $R$ from (4.1) can be expressed as

$$\begin{aligned} R(v) &= \sum_{K\in\mathcal{T}} (R_K, v)_{L^2(K)} - \sum_{E\in\mathcal{E}} < R_E, v >_{L^2(E)} \\ &\quad - \sum_{E\in\Gamma_N^{in}} < R_E^{in}, v >_{L^2(E)} - \sum_{E\in\Gamma_N^{out}} < R_E^{out}, v >_{L^2(E)} \qquad (4.7) \\ &= a(e, v) \end{aligned}$$

for any $v \in H^1_D(\Omega)$. If we define $R$ as above, then $R$ makes sense for functions in $S^*_h$ as well, and, as we showed in Lemma IV.1, $R(v^*) = 0$ for $v^* \in S^*_h$. In particular $R(\overline{e}) = 0$ and therefore equality (4.3) can be rewritten as $||e||^2_a = R(e - \overline{e})$.

### 4.1.2. Residual type error estimator

The quantities in the following definition are used as indicators for the discretization error coming correspondingly from elements, edges, and the Neumann boundary.

**Definition IV.1** *We define the error indicators*

$$
\begin{aligned}
\eta_R &:= \left( \sum_{K \in \mathcal{T}} h^2_K ||R_K||^2_{L^2(K)} \right)^{1/2}, \\
\eta_E &:= \left( \sum_{E \in \mathcal{E}} h_E ||R_E||^2_{L^2(E)} \right)^{1/2}, \\
\eta_N &:= \left( \sum_{E \in \Gamma^{in}_N} h_E ||R^{in}_E||^2_{L^2(E)} + \sum_{E \in \Gamma^{out}_N} h_E ||R^{out}_E||^2_{L^2(E)} \right)^{1/2}.
\end{aligned}
$$

The residual type error estimator $\rho$ is defined as the sum of the introduced above error indicators

$$\rho := \left( \eta^2_R + \eta^2_E + \eta^2_N \right)^{1/2}.$$

Equivalently, we can define it element-wise as

$$
\begin{aligned}
\rho^2_K := h^2_K ||R_K||^2_{L^2(K)} + h_K \Bigg( \frac{1}{2} \sum_{E \in (\partial K \cap \mathcal{E})} ||R_E||^2_{L^2(E)} &+ \sum_{E \in (\partial K \cap \Gamma^{in}_N)} ||R^{in}_E||^2_{L^2(E)} \\
&+ \sum_{E \in (\partial K \cap \Gamma^{out}_N)} ||R^{out}_E||^2_{L^2(E)} \Bigg).
\end{aligned}
\tag{4.8}
$$

The following lemma will be used in Subsection 4.1.2.1 to prove that the residual type error estimator, defined above, is reliable.

**Lemma IV.2** *For $e \in H^1_D(\Omega)$, we have the inequality*

$$\sum_{E\in\mathcal{E}} \int\limits_E [\underline{\sigma}_h]\cdot \underline{n}(e-\overline{e})\; ds \le C\; \eta_E\, \|\nabla e\|,$$

*where the constant $C$ is independent of the mesh size $h$.*

*Proof.* A well-established trace inequality (cf., e.g., [15, Theorem 1.6.6] or [20, Theorem 1.4]) and scaling argument prove that there exists a constant $C$, independent of $h$, such that

$$h_E^{1/2}\|v\|_{L^2(E)} \le C\Big(\|v\|_{L^2(K)} + h_E\|\nabla v\|_{L^2(K)}\Big) \tag{4.9}$$

for all $v \in H^1(K)$ and edges $E$ of an element $K \in \mathcal{T}$. Applying this inequality to $v := e - \overline{e}$ on each $K \cap V_i$, where $K \in \mathcal{T}$ and $x_i \in N_h$, leads to

$$\begin{aligned}\int\limits_{\beta_i} [\underline{\sigma}_h]\cdot\underline{n}(e-\overline{e})\, ds &\le \|[\underline{\sigma}_h]\cdot\underline{n}\|_{L^2(\beta_i)}\; \|e-\overline{e}\|_{L^2(\beta_i)} \\ &\lesssim h_i^{1/2}\|[\underline{\sigma}_h]\cdot\underline{n}\|_{L^2(\beta_i)}\Big(h_i^{-1}\|e-\overline{e}\|_{L^2(V_i)} + \|\nabla e\|_{L^2(V_i)}\Big).\end{aligned}$$

If $x_i \in N_h^0$, in which case by definition $\int\limits_{V_i}(e-\overline{e})\,dx = 0$, we use Poincaré's inequality, or if $x_i \in N_h\backslash N_h^0$, in which case $\overline{e} = 0$ on $V_i$ and $e = 0$ on $\partial V_i \cap \Gamma_D$, we use Friedrichs' inequality, to show that

$$h_i^{-1}\|e-\overline{e}\|_{L^2(V_i)} \lesssim \|\nabla e\|_{L^2(V_i)}. \tag{4.10}$$

Substituting the last result into the preceding inequality yields

$$\int\limits_{\beta_i} [\underline{\sigma}_h]\cdot\underline{n}(e-\overline{e})\, ds \lesssim \|h_i^{1/2}[\underline{\sigma}_h]\cdot\underline{n}\|_{L^2(\beta_i)}\; \|\nabla e\|_{L^2(V_i)}$$

for $x_i \in N_h$. A summation over all vertices and (4.2) prove the assertion. □

### 4.1.2.1. Reliability

The following theorem states that the error $e$ is bounded from above from the residual type error estimator. The proof uses the error representation (4.3), the coercivity of the bilinear form $a(\cdot,\cdot)$, and a combination of Cauchy's, trace, Poincaré's, and Friedrichs' inequalities.

**Theorem IV.1** *The residual based error estimator $\rho = (\eta_R^2 + \eta_E^2 + \eta_N^2)^{1/2}$ is reliable, i.e., there exists a constant $C$, independent of $h$, such that*

$$\|e\|_a \le C\rho.$$

*Proof.* Lemma IV.1 is used to represent $\|e\|_a^2$ as

$$\begin{aligned}\|e\|_a^2 &= \sum_{K\in\mathcal{T}}\int_K h_K R_K\ h_K^{-1}(e-\overline{e})\ dx - \sum_{E\in\mathcal{E}}\int_E h_E^{\frac{1}{2}} R_E\ h_E^{-\frac{1}{2}}(e-\overline{e})\ ds \\ &\quad - \sum_{E\in\Gamma_N^{in}}\int_E h_E^{\frac{1}{2}} R_E^{in}\ h_E^{-\frac{1}{2}}(e-\overline{e})\ ds - \sum_{E\in\Gamma_N^{out}}\int_E h_E^{\frac{1}{2}} R_E^{out}\ h_E^{-\frac{1}{2}}(e-\overline{e})\ ds.\end{aligned}$$

We bound the first term using Cauchy's inequality, the second using Lemma IV.2, and the remaining two terms again using Cauchy's inequality. This leads to

$$\|e\|_a^2 \lesssim \eta_R\|h^{-1}(e-\overline{e})\| + \eta_E\|\nabla e\| + \eta_N\|h^{-\frac{1}{2}}(e-\overline{e})\|_{L^2(\Gamma_N)}.$$

Inequality (4.10) is combined with the trace inequality (4.9) to obtain

$$\begin{aligned}\|h^{-\frac{1}{2}}(e-\overline{e})\|_{L^2(\Gamma_N)}^2 + \|h^{-1}(e-\overline{e})\|^2 &\lesssim \sum_{x_i\in N_h}\left(h_i^{-2}\|e-\overline{e}\|_{L^2(V_i)}^2 + \|\nabla e\|_{L^2(V_i)}^2\right) \\ &\lesssim \|\nabla e\|^2\ .\end{aligned}$$

Assumption (II.1) yields $\|\nabla e\| \lesssim \|e\|_a$. This and the preceding two inequalities conclude the proof of the theorem. □

4.1.2.2. Efficiency

The error estimates proved in Theorem IV.1 are sharp. The converse estimate holds even in a more local form (as shown in the proof) then displayed.

**Theorem IV.2** *The residual based error estimator* $\rho = (\eta_R^2 + \eta_E^2 + \eta_N^2)^{1/2}$ *is efficient, i.e., there exists a constant* $C$*, independent of* $h$*, such that*

$$\rho \leq C\ \|e\|_a + h.o.t.$$

**Remark IV.1** *We use linear approximations, therefore a priori, if the considered problem has sufficient regularity,* $||e||_a$ *is of order* $Ch$*, where* $C$ *is an independent of the mesh size* $h$ *constant. Similarly, for sufficient regularity, terms of order* $Ch^\alpha$*,* $\alpha > 1$ *are denoted by h.o.t.*

*Proof.* We will prove that any of the quantities $\eta_R$, $\eta_E$, and $\eta_N$ is bounded by $C\ \|e\|_a + h.o.t.$. First, we will show the bound for $\eta_N$, and more precisely, for the error contribution due to the $\Gamma_N^{in}$ part of the Neumann boundary, i.e., we will prove that

$$\sum_{E \in \Gamma_N^{in}} h_E ||g_N - \underline{\sigma}_h \cdot \underline{n}||^2_{L^2(E)} \leq C||e||_a^2 + h.o.t. \tag{4.11}$$

We consider an element $K \in \mathcal{T}$ that has an edge $E \in \Gamma_N^{in}$. We will use the pair $K$, $E$, as specified, in the rest of the proof. First,

$$h_E^{1/2} \|g_N - \overline{g}\|_{L^2(E)} = \text{ h.o.t.} \quad \text{for} \quad \overline{g} := \frac{1}{|E|} \int\limits_E g_N\, ds,$$

$$h_E^{1/2}\|(\underline{\sigma}_h - \overline{\sigma}_h)\cdot\underline{n}\|_{L^2(E)} = \text{ h.o.t.} \quad \text{for} \quad \overline{\sigma}_h := \frac{1}{|E|}\int\limits_E \underline{\sigma}_h\cdot\underline{n}\,ds.$$

Then

$$\begin{aligned}\|g_N - \underline{\sigma}_h\cdot\underline{n}\|_{L^2(E)} &\leq \|g_N - \overline{g}\|_{L^2(E)} + \|\overline{g} - \overline{\sigma}_h\cdot\underline{n}\|_{L^2(E)} + \|(\overline{\sigma}_h - \underline{\sigma}_h)\cdot\underline{n}\|_{L^2(E)} \\ &\lesssim \|\overline{g} - \overline{\sigma}_h\cdot\underline{n}\|_{L^2(E)} + \ h.o.t.\end{aligned}$$

We will prove that $\|\overline{g}-\overline{\sigma}_h\cdot\underline{n}\|_{L^2(E)} \lesssim h_E^{-1/2}\|\underline{\sigma}-\underline{\sigma}_h\|_{L^2(K_E)}+\text{h.o.t.}$, and then summation over $E\in\Gamma_N^{in}$ will yield (4.11).

Consider an edge-bubble function $b_E\in H^1(\Omega)$, $b_E\geq 0$, $b_E(x)=0$ on $\Omega\setminus K$ and $\partial K\setminus E$, with properties

$$\int\limits_E b_E\,ds = \int\limits_E ds, \quad \|b_E\|_{L^\infty(K)}\lesssim 1, \quad \|\nabla b_E\|_{L^\infty(K)}\lesssim 1/h_E. \tag{4.12}$$

A 2-D example of such bubble is $b_E = 6\phi_1\phi_2$, where $\phi_1$ and $\phi_2$ are the standard linear nodal basis functions associated with the end points of the edge $E$. Let $z\in H^1(K)$ be the harmonic extension of $(\overline{g}-\overline{\sigma}_h\cdot\underline{n})b_E$ from $\partial K$ to $K$. The extension is bounded in $H^1$ [52, Theorem 4.1.1] on a reference element $\hat{K}$ by the $H^{1/2}(\hat{E})$ norm of the extended quantity, and since all norms are equivalent on a finite dimensional space, by its $L^2(\hat{E})$ norm. Therefore, a scaling argument gives

$$h_E^{1/2}\|\nabla z\|_{L^2(K)} + h_E^{-1/2}\|z\|_{L^2(K)} \lesssim \|b_E(\overline{g}-\overline{\sigma}_h\cdot\underline{n})\|_{L^2(E)}. \tag{4.13}$$

We define the linear operator $P_K$ into the space of polynomials of degree 2 on an element $K$ as

$$(b_K\ P_K z, p_h)_{L^2(K)} = (z, p_h)_{L^2(K)}$$

for all polynomials $p_h$ of degree 2, where $b_K \in H^1(\Omega)$, $b_K \geq 0$ is an element-bubble function with properties

$$\text{supp } b_K \subset K, \ \int\limits_K b_K \, ds = \int\limits_K \, ds, \ \|b_K\|_{L^\infty(K)} \lesssim 1, \ \|\nabla b_K\|_{L^\infty(K)} \lesssim 1/h_K.$$

One 2-D example of such bubble is $b_K = 60\phi_1\phi_2\phi_3$, where $\phi_1$, $\phi_2$, and $\phi_3$ are the standard linear nodal basis functions associated with the vertices of the element $K$. Then $\tilde{z} := z - b_K \ P_K z$ by construction has the properties

$$\begin{aligned} &\tilde{z} = (\overline{g} - \underline{\sigma}_h \cdot \underline{n}) b_E \quad \text{on } E, \ \tilde{z} = 0 \quad \text{on } \partial K \backslash E, \\ &(\tilde{z}, \ p_h)_{L^2(K)} = 0 \quad \text{for all polynomials } p_h \text{ of degree 2.} \end{aligned}$$

Inequality (4.13) remains valid for $z$ replaced by $\tilde{z}$, because of the following. Choosing $p_h = P_K z$ in the definition of $P_K$, yields

$$\|b_K^{1/2} P_K z\|^2_{L^2(K)} = (z, P_K z)_{L^2(K)} \lesssim \|z\|_{L^2(K)} \|P_K z\|_{L^2(K)}.$$

We use norm equivalence on finite dimensional spaces on a reference element and scaling to $K$ to get that the quantities $\|b_K P_K z\|_{L^2(K)}$, $\|b_K^{1/2} P_K z\|_{L^2(K)}$, and $\|P_K z\|_{L^2(K)}$ are equivalent up to constants independent of $h$, and therefore $\|b_K P_K z\|_{L^2(K)} \lesssim \|z\|_{L^2(K)}$. We use again the equivalence of norms argument, inverse inequality and the properties of $z$ to get that

$$\begin{aligned} \|\nabla(b_K P_K z)\|_{L^2(K)} &\lesssim \|\nabla b_K\|_{L^2(K)} \|P_K z\|_{L^2(K)} + \|b_K \nabla(P_K z)\|_{L^2(K)} \\ &\lesssim h_E^{-1} \|P_K z\|_{L^2(K)} + h_E^{-1} \|z\|_{L^2(K)} \\ &\lesssim h_E^{-1/2} \|b_E(\overline{g} - \underline{\sigma}_h \cdot \underline{n})\|_{L^2(E)}, \end{aligned}$$

which, combined with the bound for $\|b_K P_K z\|_{L^2(K)}$, finishes the proof that (4.11) is valid for $z = \tilde{z}$.

Now, for any polynomial $p_h$ of degree 2, using the Gauß divergence theorem and the properties of $\tilde{z}$, we get

$$\begin{aligned}\int\limits_E b_E(\overline{g} - \overline{\underline{\sigma}}_h \cdot \underline{n})(\underline{\sigma} - \underline{\sigma}_h) \cdot \underline{n}\, ds &= \int\limits_{\partial K} \tilde{z}(\underline{\sigma} - \underline{\sigma}_h) \cdot \underline{n}\, ds \\ &= \int\limits_K (\underline{\sigma} - \underline{\sigma}_h) \cdot \nabla \tilde{z}\, dx + \int\limits_K \tilde{z}(\nabla \cdot (\underline{\sigma} - \underline{\sigma}_h) - p_h)\, dx \\ &\lesssim \Big(\|\underline{\sigma} - \underline{\sigma}_h\|_{L^2(K)} + h_E \|\nabla \cdot (\underline{\sigma} - \underline{\sigma}_h) - p_h\|_{L^2(K)}\Big) h_E^{-1/2} \|b_E(\overline{g} - \overline{\underline{\sigma}}_h \cdot \underline{n})\|_{L^2(E)}.\end{aligned}$$

Subtracting proper $p_h$ from the second term in the last inequality makes that term *h.o.t.* Namely, we first write the equality (see the basic problem (2.4) definition)

$$\nabla \cdot (\underline{\sigma} - \underline{\sigma}_h) - p_h = cu - f - \nabla \cdot (A \nabla u_h) + u_h \nabla \cdot \underline{b} + \underline{b} \cdot \nabla u_h - p_h. \qquad (4.14)$$

Let $\tilde{f}$, $\widetilde{cu}$, $\widetilde{\nabla \cdot \underline{b}}$, $\tilde{\underline{b}}$, and $\widetilde{\nabla \cdot A}$ are the linear approximations on $K$ of respectively $f$, $cu$, $\nabla \cdot \underline{b}$, $\underline{b}$, and $\nabla \cdot A$. Now, we take the proper $p_h$, write (4.14) in the $L^2(K)$ norm and use triangle's inequality to get

$$\begin{aligned}\|\nabla \cdot (\underline{\sigma} - \underline{\sigma}_h) - p_h\|_{L^2(K)} \;&\leq\; \|f - \tilde{f}\|_{L^2(K)} + \|cu - \widetilde{cu}\|_{L^2(K)} \\ &\quad + \|u_h(\nabla \cdot \underline{b} - \widetilde{\nabla \cdot \underline{b}})\|_{L^2(K)} + \|(\underline{b} - \tilde{\underline{b}}) \cdot \nabla u_h\|_{L^2(K)} \\ &\quad + \|\nabla u_h \cdot (\nabla \cdot A - \widetilde{\nabla \cdot A})\|_{L^2(K)} \\ &\lesssim\; \Big(\|u\|_{H^1(K)} + \|u_h\|_{H^1(K)}\Big)\ \text{h.o.t.} + \|h_K \nabla f\|_{L^2(K)}.\end{aligned}$$

Therefore, (note that $g = \underline{\sigma} \cdot \underline{n}$ on $\Gamma_N^{in}$)

$$\begin{aligned} \|b_E^{1/2}(\overline{g} - \overline{\underline{\sigma}}_h \cdot \underline{n})\|^2_{L^2(E)} &= \int\limits_E \tilde{z}(\overline{g} - g)\, ds + \int\limits_E \tilde{z}(\underline{\sigma} - \underline{\sigma}_h) \cdot \underline{n}\, ds \\ &\quad + \int\limits_E \tilde{z}(\underline{\sigma}_h - \overline{\underline{\sigma}}_h) \cdot \underline{n}\, ds \\ &\lesssim h_E^{-1/2}\Big(\|\underline{\sigma} - \underline{\sigma}_h\|_{L^2(K)} + \text{ h.o.t.}\Big)\ \|(\overline{g} - \overline{\underline{\sigma}}_h \cdot \underline{n}) b_E\|_{L^2(E)} \end{aligned}$$

and so

$$\|b_E^{1/2}(\overline{g} - \overline{\underline{\sigma}}_h \cdot \underline{n})\|_{L^2(E)} \lesssim h_E^{-1/2}\|\underline{\sigma} - \underline{\sigma}_h\|_{L^2(K)} + h.o.t.$$

Using again the equivalence-of-norms estimate (equivalence of norms on finite dimensional spaces on reference element plus scaling)

$$\|\overline{g} - \overline{\underline{\sigma}}_h \cdot \underline{n}\|_{L^2(E)} \lesssim \|b_E^{1/2}(\overline{g} - \overline{\underline{\sigma}}_h \cdot \underline{n})\|_{L^2(E)}$$

finally proves that

$$\begin{aligned} \|g_N - \underline{\sigma}_h \cdot \underline{n}\|_{L^2(E)} &\leq \|g_N - \overline{g}\|_{L^2(E)} + \|\overline{g} - \overline{\underline{\sigma}}_h \cdot \underline{n}\|_{L^2(E)} + \|(\overline{\underline{\sigma}}_h - \underline{\sigma}_h) \cdot \underline{n}\|_{L^2(E)} \\ &\lesssim \|b_E^{1/2}(\overline{g} - \overline{\underline{\sigma}}_h \cdot \underline{n})\|_{L^2(E)} + \ h.o.t. \\ &\lesssim h_E^{-1/2}\|\underline{\sigma} - \underline{\sigma}_h\|_{L^2(K)} + \ h.o.t. \end{aligned}$$

Similarly, $||A\nabla u_h \cdot \underline{n}||_{L^2(E)} \lesssim h_E^{-1/2}\|\underline{\sigma} - \underline{\sigma}_h\|_{L^2(K)} + \ h.o.t.$, for $E \in \Gamma_N^{out}$, which, combined with the result for $E \in \Gamma_N^{in}$, proves that $\eta_N \leq C||e||_a + \ h.o.t.$

A similar technique shows that $\eta_E \leq C||e||_a + \ h.o.t.$

The last inequality, $\eta_R^2 = \sum\limits_{K \in \mathcal{T}} h_K^2 ||R_K||^2_{L^2(K)} \leq C||e||_a^2 + \text{h.o.t.}$, can be proved in the following way. This time we use the average $\bar{R}_K$ of $R_K$ over the element $K$ to get

$$||\bar{R}_K||_{L^2(K)} \leq ||R_K - \bar{R}_K||_{L^2(K)} + ||R_K||_{L^2(K)} = \text{ h.o.t.} + ||R_K||_{L^2(K)}.$$

We take $v = b_K\bar{R}_K$ in (4.7) to get that $(R_K, b_K\bar{R}_K)_{L^2(K)} = a(e, b_K\bar{R}_K)$ and therefore

$$(R_K, b_K\bar{R}_K)_{L^2(K)} = ||b_K^{1/2}R_K||^2_{L^2(K)} - (R_K, b_K(R_K - \bar{R}_K))_{L^2(K)} = a(e, b_K\bar{R}_K)$$

$$\leq ||e||_{a(K)}||b_K\bar{R}_K||_{a(K)} \lesssim ||e||_{a(K)}h_K^{-1}||\bar{R}_K||_{L^2(K)} \lesssim h_K^{-1}||e||_{a(K)}||R_K||_{L^2(K)} + \text{h.o.t.},$$

where we used an inverse inequality and denoted by $||\cdot||_{a(K)}$ the $||\cdot||_a$ norm restricted to $K$. Then we take the term $(R_K, b_K(R_K - \bar{R}_K))_{L^2(K)}$ on the right hand side and consider it as h.o.t., and use that $||b_K^{1/2}R_K||_{L^2(K)} \approx ||R_K||_{L^2(K)}$ to finally get that

$$||R_K||_{L^2(K)} \lesssim h^{-1}||e||_{a(K)} + \text{h.o.t.},$$

which we wanted to prove, and which also concludes the proof of Theorem IV.2. □

### 4.1.3. Zienkiewicz-Zhu type error estimator

We define the following locally computable quantity and use it as an error estimator (see the proofs for the upper and lower bounds correspondingly in Subsections 4.1.3.1 and 4.1.3.2). The estimator is based on a local projection, which is typical for the Zienkiewicz-Zhu type error estimators.

**Definition IV.2** *Let $P_i$ be the $L^2$-projection onto the linear functions on $V_i$. The Zienkiewicz-Zhu type error estimator, denoted by $\eta_Z$, is defined for $A(x)$ smooth over the volumes $V_i$, as*

$$\eta_Z^2 := \sum_{x_i \in N_h} ||\underline{\sigma}_h - P_i\underline{\sigma}_h||^2_{L^2(V_i)}.$$

**Remark IV.2** *If we allow jumps of $A$ in the volumes $V_i$, then we have to change the projection $P_i$. For example, if $V_i = V_i^1 \cup V_i^2$ and $A$ is smooth on $V_i^1$ and $V_i^2$ but has* *jump across their interface, then $P_i$ is defined as the piecewise function*

$$||\underline{\sigma}_h - P_i\underline{\sigma}_h||^2_{L^2(V_i)} = ||\underline{\sigma}_h - P_i^1\underline{\sigma}_h||^2_{L^2(V_i^1)} + ||\underline{\sigma}_h - P_i^2\underline{\sigma}_h||^2_{L^2(V_i^2)},$$

*where $P_i^1$ and $P_i^2$ are correspondingly the $L^2$-projections onto the linear functions on $V_i^1$ and $V_i^2$.*

**Lemma IV.3** *There holds*

$$h_i^{1/2}||[\underline{\sigma}_h]\cdot\underline{n}||_{L^2(\beta_i)} \;\lesssim\; ||\underline{\sigma}_h - P_i\underline{\sigma}_h||_{L^2(V_i)} + h.o.t. \tag{4.15}$$

*for all $x_i \in N_h$. In particular, $\eta_E \lesssim \eta_Z + h.o.t.$ The multiplicative constants behind the notation $\lesssim$ depend on the shape of the elements in $\mathcal{T}$ and the shape of the control volumes in $\mathcal{T}^*$. The h.o.t. depends on the element-wise smoothness of $\underline{\sigma}_h$, and hence on the coefficients $A$ and $\underline{b}$.*

*Proof.* If $\underline{\sigma}_h|_K$ are in finite dimensional spaces for any $K \in \mathcal{T}$, then we prove that h.o.t. = 0 in (4.15), i.e., if we denote the finite dimensional $\underline{\sigma}_h|_K$ as $\overline{\underline{\sigma}}_h$, then we have to prove that

$$h_i^{1/2}||[\overline{\underline{\sigma}}_h]\cdot\underline{n}||_{L^2(\beta_i)} \;\lesssim\; ||\overline{\underline{\sigma}}_h - P_i\overline{\underline{\sigma}}_h||_{L^2(V_i)} \quad \text{for all } x_i \in N_h. \tag{4.16}$$

We prove estimate (4.16) by an equivalence-of-norm argument on finite dimensional spaces. Both sides of (4.16) define semi-norms for the finite dimensional $\overline{\underline{\sigma}}_h$. If the right-hand side vanishes for some $\overline{\underline{\sigma}}_h$, then $\overline{\underline{\sigma}}_h = P_i\overline{\underline{\sigma}}_h$ on $V_i$. Since $P_i\overline{\underline{\sigma}}_h$ is linear on $V_i$, so is $\overline{\underline{\sigma}}_h$. Therefore, the jump $[\overline{\underline{\sigma}}_h]$ is zero on $\beta_i$, i.e. the left-hand side of (4.16) vanishes as well. This proves that the semi-norm on the right-hand side is stronger than the semi-norm on the left-hand side and so proves (4.16). A scaling argument shows that the multiplicative constant behind $\lesssim$ in (4.16) is independent of $h_i$.

The case when $\underline{\sigma}_h|_K$ are not finite dimensional is treated using the averaging techniques from the proof of Theorem IV.2. Namely, we introduce finite dimensional approximations $\overline{\underline{\sigma}}_h$ of $\underline{\sigma}_h$ for any $K \in \mathcal{T}$, and denote

$$||\underline{\sigma}_h - \overline{\underline{\sigma}}_h||_{L^2(V_i)} = \text{h.o.t.} \quad \text{and} \quad ||[\underline{\sigma}_h - \overline{\underline{\sigma}}_h] \cdot \underline{n}||_{L^2(V_i)} = \text{h.o.t.}$$

Then, we consequently get

$$\begin{aligned} h_i^{1/2}||[\underline{\sigma}_h] \cdot \underline{n}||_{L^2(\beta_i)} &\lesssim h_i^{1/2}||[\overline{\underline{\sigma}}_h] \cdot \underline{n}||_{L^2(\beta_i)} + h_i^{1/2}||[\underline{\sigma}_h - \overline{\underline{\sigma}}_h] \cdot \underline{n}||_{L^2(\beta_i)} \\ &\lesssim ||\overline{\underline{\sigma}}_h - P_i\overline{\underline{\sigma}}_h||_{L^2(V_i)} + \text{h.o.t.} \\ &\lesssim ||\overline{\underline{\sigma}}_h - \underline{\sigma}_h||_{L^2(V_i)} + ||\underline{\sigma}_h - P_i\underline{\sigma}_h||_{L^2(V_i)} \\ &\qquad + ||P_i(\underline{\sigma}_h - \overline{\underline{\sigma}}_h)||_{L^2(V_i)} + \text{h.o.t.} \\ &\lesssim ||\underline{\sigma}_h - P_i\underline{\sigma}_h||_{L^2(V_i)} + \text{h.o.t.}, \end{aligned}$$

where we used the boundedness of the projection $P_i$ in $L^2$ and the element-wise approximation properties of the finite dimensional $\overline{\underline{\sigma}}_h|_K$ for any $K \in \mathcal{T}$. This concludes the proof of inequality (4.15). We get $\eta_E \lesssim \eta_Z + \text{h.o.t.}$ by summation of (4.15) over all $x_i \in N_h$. □

### 4.1.3.1. Reliability

The Zienkiewicz-Zhu type error estimator $\eta_Z$ bounds the error from above, which is the result of the following reliability theorem.

**Definition IV.3** *We define a strip $\Omega_D$ around the Dirichlet boundary as*

$$\Omega_D := \cup\{V_i : x_i \in N_h \cap \Gamma_D\}.$$

**Theorem IV.3** *Suppose $f \in H^1(\Omega)$. Then, there holds*

$$\|e\|_a \lesssim \eta_Z + \eta_N + h.o.t., \tag{4.17}$$

*where the h.o.t. depends on the h.o.t. from Lemma IV.3 and on the following expression*

$$\|h^2 \nabla(cu_h)\| + \|hf\|_{L^2(\Omega_D)} + \|h^2 \nabla f\|.$$

**Remark IV.3** *The finite volume scheme at hand is of first order. For $f \in H^1(\Omega)$ we have that $\|f\|_{L^2(\Omega_D)} \lesssim h^{1/2}||f||_{H^1(\Omega)}$, and so*

$$\|h^2 \nabla(cu_h)\| + \|hf\|_{L^2(\Omega_D)} + \|h^2 \nabla f\| = \ h.o.t.$$

*is of higher order.*

*Proof.* To prove the theorem we use again the error representation (4.3) from Lemma IV.1. In Theorem IV.1 we bounded the third and fourth sum of (4.3) by $C\eta_N||\nabla e||$. The second sum of (4.3) was bounded by $C\eta_E||\nabla e||$, and $\eta_E$ was bounded by $C\eta_Z +$ h.o.t. in Lemma IV.3 with constants $C$ independent on the mesh size $h$. Therefore, to prove the present theorem, we need to get the bound

$$\sum_{K \in \mathcal{T}} \int_K (f - \nabla \cdot \underline{\sigma}_h - cu_h)(e - \overline{e}) \; dx \lesssim (\eta_Z + \text{h.o.t.})||\nabla e||.$$

For each $x_i \in N_h^0$ we have the following

$$\begin{aligned} \int_{V_i} (f - \nabla \cdot \underline{\sigma}_h - cu_h)(e - \overline{e}) \, dx &= \int_{V_i} (f - \overline{f})(e - \overline{e}) \, dx \\ &- \int_{V_i} \nabla \cdot (\underline{\sigma}_h - P_i \underline{\sigma}_h)(e - \overline{e}) \, dx - \int_{V_i} (cu_h - \overline{cu_h})(e - \overline{e}) \, dx \end{aligned} \tag{4.18}$$

$$\leq \|e - \overline{e}\|_{L^2(V_i)}(\|f - \overline{f}\|_{L^2(V_i)} + \|\nabla \cdot (\underline{\sigma}_h - P_i \underline{\sigma}_h)\|_{L^2(V_i)} + \|cu_h - \overline{cu_h}\|_{L^2(V_i)}),$$

where $\overline{f}$ and $\overline{cu_h}$ are the integral means over $V_i$ of correspondingly $f$ and $cu_h$. Poincaré's inequality gives

$$\begin{aligned} \|e - \overline{e}\|_{L^2(V_i)} &\lesssim h_i \|\nabla e\|_{L^2(V_i)}, \\ \|f - \overline{f}\|_{L^2(V_i)} &\lesssim h_i \|\nabla f\|_{L^2(V_i)}, \\ \|cu_h - \overline{cu_h}\|_{L^2(V_i)} &\lesssim h_i \|\nabla (cu_h)\|_{L^2(V_i)}. \end{aligned} \tag{4.19}$$

The term $\|\nabla \cdot (\underline{\sigma}_h - P_i \underline{\sigma}_h)\|_{L^2(V_i)}$ is treated by introducing again the finite dimensional element-wise approximation $\overline{\underline{\sigma}}_h$ of $\underline{\sigma}_h$. We use the inverse estimate

$$\|\nabla \cdot (\overline{\underline{\sigma}}_h - P_i \overline{\underline{\sigma}}_h)\|_{L^2(V_i)} \lesssim h_i^{-1} \|\overline{\underline{\sigma}}_h - P_i \overline{\underline{\sigma}}_h\|_{L^2(V_i)}$$

and averaging techniques as before to prove that for element-wise smooth $\underline{\sigma}_h$

$$\|\nabla \cdot (\underline{\sigma}_h - P_i \underline{\sigma}_h)\|_{L^2(V_i)} \lesssim h_i^{-1} \|\underline{\sigma}_h - P_i \underline{\sigma}_h\|_{L^2(V_i)} + \text{h.o.t.} \tag{4.20}$$

The combination of (4.18)-(4.20) shows

$$\begin{aligned} &\int\limits_{V_i} (f + \nabla \cdot \underline{\sigma}_h - cu_h)(e - \overline{e})\, dx \\ &\qquad \lesssim \|\nabla e\|_{L^2(V_i)} \Big( \|h_i^2 \nabla f\|_{L^2(V_i)} + \|h_i^2 \nabla (cu_h)\|_{L^2(V_i)} \\ &\qquad\qquad + \|\underline{\sigma}_h - P_i \underline{\sigma}_h\|_{L^2(V_i)} + \text{h.o.t.} \Big). \end{aligned} \tag{4.21}$$

So far estimate (4.21) holds for $x_i \in N_h^0$. For $x_i \in N_h \cap \Gamma_D$ we replace $\overline{e}$, $\overline{f}$, and $\overline{cu_h}$ by zero and deduce the first and third inequalities of (4.19) from Friedrichs' inequality (notice that $e$ and $cu_h$ vanish on $\Gamma_D \cap V_i$). The inverse estimate (4.20) holds for $x_i \in N_h \cap \Gamma_D$ as well. The aforementioned arguments prove (4.21) with $\|h_i^2 \nabla f\|_{L^2(V_i)}$

replaced by $\|h_i f\|_{L^2(V_i)}$. This shows

$$(f + \nabla \cdot \underline{\sigma}_h - cu_h, e - \overline{e}) \\ \lesssim \Big(\eta_Z + \|h^2\nabla(cu_h)\| + \|h^2\nabla f\| + \|hf\|_{L^2(\Omega_D)} + \text{h.o.t.}\Big)||\nabla e||.$$

The last result, the discussion at the beginning of the theorem and $\|\nabla e\| \lesssim \|e\|_a$, which is the coercivity of the form $a(\cdot,\cdot)$ from Assumption (II.1), conclude the proof of the theorem. □

#### 4.1.3.2. Efficiency

The Zienkiewicz-Zhu type error estimator $\eta_Z$ is also efficient, which is the result of the following theorem.

**Theorem IV.4** *There holds*

$$\eta_Z \leq \|e\|_a + h.o.t.$$

*Proof.* The proof is straightforward, namely, since $P_i$ is a linear $L^2(V_i)$ projector, we have that

$$\|\underline{\sigma}_h - P_i\underline{\sigma}_h\|_{L^2(V_i)} \leq \|\underline{\sigma}_h - P_i\underline{\sigma}\|_{L^2(V_i)}.$$

Adding and subtracting $\underline{\sigma}$ in the right hand side, and applying triangle's inequality gives

$$\|\underline{\sigma}_h - P_i\underline{\sigma}\|_{L^2(V_i)} \leq \|\underline{\sigma}_h - \underline{\sigma}\|_{L^2(V_i)} + \|\underline{\sigma} - P_i\underline{\sigma}\|_{L^2(V_i)} = \|\underline{\sigma}_h - \underline{\sigma}\|_{L^2(V_i)} + \text{h.o.t.},$$

i.e., $\|\underline{\sigma}_h - P_i\underline{\sigma}_h\|_{L^2(V_i)} \leq ||e||_{a(V_i)} + \text{h.o.t.}$, since $\|\underline{\sigma} - P_i\underline{\sigma}\|_{L^2(V_i)} =$ h.o.t. is of higher-order if $\underline{\sigma}$ is smooth. Summation over all $x_i$ concludes the proof of the theorem. □

### 4.1.4. Estimators based on local Dirichlet/Neumann problems

The error estimators in this subsection are obtained by replacing the expensive solution in an enriched finite dimensional space $S$, $S_h \subset S \subset H^1_D(\Omega)$, by solutions of local Dirichlet or Neumann problems. In the finite element setting, $S$ is usually obtained : (1) by using higher order polynomials over the same mesh, or (2) by using polynomials of the same order, but over locally $h$-refined mesh.

The local Dirichlet/Neumann problems are defined using the idea described in Subsection 4.1.4. The continuous problems can be discretized and solved either in finite volume or finite element method setting. In both cases one proves that the discrete local solutions are equivalent to the local residuals from Subsection 4.1.2. However, the proof in the finite volume setting introduces some discrete norms, which we would like to avoid. In Subsection 4.1.4.1, we define the finite volume a posteriori error estimators as solutions of the finite element discretizations of certain local Dirichlet/Neumann problems. In Subsection 4.1.4.2 we discuss the proof of the reliability and the efficiency of the defined estimators.

#### 4.1.4.1. Definition

Recall that $K_i$ denotes the union of finite elements $K \in \mathcal{T}$ sharing a vertex $x_i$. For an edge $E$ and an element $K$, we will use the edge bubble function $b_E$ and the volume bubble function $b_K$, which in 2-D are defined as $b_E = 6\phi_1\phi_2$, where $\phi_1$ and $\phi_2$ are the standard linear nodal basis functions associated with the end points of the edge $E$, and $b_K = 60\phi_1\phi_2\phi_3$, where $\phi_1$, $\phi_2$, and $\phi_3$ are the standard linear nodal basis functions associated with the vertices of $K$. For every element $K$ and vertex $x_i$ we define correspondingly the spaces

$$S_K := \{b_K \text{ and edge bubbles } b_E,\ b_E = 0 \text{ for } E \in \Gamma_D,\ E \text{ is an edge of } K\}$$

and

$$S_i := \{v \in C(\overline{K_i}) \ : \ v|_K \in S_K, \ K \in K_i, \ v = 0 \text{ on } \partial V_i \setminus \Gamma_N\}.$$

We define $u_{N,h} \in S_K$ and $u_{D,h} = u_h + v_h$, $v_h \in S_i$ as the solutions of

$$a(u_{N,h}, \ v) = R(v) \quad \text{for } \forall v \in S_K \tag{4.22}$$

and

$$a(u_{D,h}, \ v) = F(v) \quad \text{for } \forall v \in S_i, \tag{4.23}$$

where $R$ is defined as in (4.7) but with integrals restricted to $K$ and $\partial K$. The local Neumann and Dirichlet error estimators, as given in Subsection 4.1.4, are correspondingly

$$\eta_{LN}^K := ||\nabla u_{N,h}||_{L^2(K)} \quad \text{and} \quad \eta_{LD}^i := ||\nabla u_{D,h} - \nabla u_h||_{L^2(K_i)} \ .$$

### 4.1.4.2. Reliability and efficiency

The estimators $\eta_{LN}^K$ and $\eta_{LD}^i$ bound locally the error $e_h$ from above and below. The proof is by proving equivalence to the local residuals from Subsection 4.1.2.

For example, consider $\eta_{LN}^K$. The reliability follows from the $|| \cdot ||_a$-stability of $u_{N,h}$ with respect to the right hand side. For local problems, discretized by the finite element method as in Subsection 4.1.4.1, this reads

$$||u_{N,h}||_{a(K)}^2 = R(u_{N,h}) \leq ||R|| \ ||u_{N,h}||_{L^2(K)} \lesssim h_K ||R|| \ ||\nabla u_{N,h}||_{L^2(K)},$$

where the last inequality is true since the discrete $u_{N,h}$ is 0 at the vertices of $K$. Using the explicit formula for $R$ in (4.7) we see that $h_K ||R||$ represents the contributions from $K$ and $\partial K$ to the residual based estimator $\eta_R + \eta_E + \eta_N$.

To prove the efficiency first note that (see the exact error (4.1) representation)

$$R(v) = a(e,v) \leq ||e||_a \; ||v||_a \;\text{ for all }\; v \in H^1_D(\Omega),$$

i.e., for any $v \in H^1_D(\Omega)$, the quantity $\frac{R(v)}{||v||_a}$ gives a lower bound for the error. For problem (4.22) the above remark translates to $||u_{N,h}||_a \geq \frac{R(v)}{||v||_a}$ for any $v \in S_K$. For proper test functions $v$, the quantity $\frac{R(v)}{||v||_a}$ is bounded from below by the residual based error estimator $\rho$ plus h.o.t. For example, for local problems that are discretized by the finite element method as in Subsection 4.1.4.1, we take $v = b_K \bar{R}_K \in S_K$, use averaging techniques as before, inverse inequality, etc., to obtain the bound

$$\begin{aligned} ||u_{N,h}||_a \;&\geq\; \frac{R(v)}{||v||_a} = \frac{||b_K^{1/2}\bar{R}_K||^2 + (R_K - \bar{R}_K, b_K\bar{R}_K)}{||b_K\bar{R}_K||_a} \;\geq\; C\frac{||\bar{R}_K||^2_{L^2(K)} - \text{h.o.t.}}{h_K^{-1}||\bar{R}_K||_{L^2(K)}} \\ &\geq\; Ch_K||\bar{R}_K - R_K + R_K||_{L^2(K)} - \text{h.o.t.} \geq Ch_K||R_K||_{L^2(K)} - \text{h.o.t.} \end{aligned}$$

Similarly, we get that $||u_{N,h}||_a$ is bounded from below from the edge contributions to the residual based error estimator.

Similar techniques work for the estimators based on local Dirichlet problems. The only difference is that in the Neumann problems the discrete solutions are directly compared to the residuals. In the Dirichlet estimators, the residuals are obtained by integration by parts.

### 4.1.5. Estimators based on dual problems

We use dual techniques to get error estimators for different quantities of the error, which, as given in Subsection 3.3.5, are properly defined linear error functionals in $H^1_D(\Omega)$.

In this subsection we will show how to use the duality technique in order to derive an error estimator in the global $L^2(\Omega)$-norm. The technique will follow the idea from the abstract a posteriori error analysis framework from Section 3.1.

**Definition IV.4** *We define the error indicators*

$$\tilde{\eta}_R := \left\{ \sum_{K\in\mathcal{T}} \left( h_K^2 ||R_K - \bar{R}_K||^2_{L^2(K)} + h_K^4 ||R_K||^2_{L^2(K)} \right) \right\}^{1/2},$$

$$\tilde{\eta}_E := \left\{ \sum_{E\in\mathcal{E}} \left( h_E ||R_E - \bar{R}_E||^2_{L^2(E)} + h_E^3 ||R_E||^2_{L^2(E)} \right) \right\}^{1/2},$$

$$\tilde{\eta}_N := \left\{ \sum_{E\in\Gamma_N^{in}} \left( h_E ||R_E^{in} - \bar{R}_E^{in}||^2_{L^2(E)} + h_E^3 ||R_E^{in}||^2_{L^2(E)} \right) \right.$$

$$\left. + \sum_{E\in\Gamma_N^{out}} \left( h_E ||R_E^{out} - \bar{R}_E^{out}||^2_{L^2(E)} + h_E^3 ||R_E^{out}||^2_{L^2(E)} \right) \right\}^{1/2},$$

*where* $\bar{R}_K$*,* $\bar{R}_E$*,* $\bar{R}_E^{in}$*, and* $\bar{R}_E^{out}$ *are the mean values over correspondingly* $K \in \mathcal{T}$*,* $E \in \mathcal{E}$*,* $E \in \Gamma_N^{in}$*, and* $E \in \Gamma_N^{out}$ *of* $R_K$*,* $R_E$*,* $R_E^{in}$*, and* $R_E^{out}$ *defined in (4.6).*

*We define the residual* $L^2$ *a posteriori error estimator* $\tilde{\rho}$ *as*

$$\tilde{\rho} := (\tilde{\eta}_R^2 + \tilde{\eta}_E^2 + \tilde{\eta}_N^2)^{1/2}.$$

Our aim is to show that the estimator $\tilde{\rho}$ is reliable in the $L^2(\Omega)$ norm. The a posteriori $L^2(\Omega)$ error analysis involves the continuous dual problem: Find $\tilde{e} \in H_D^1(\Omega)$ such that

$$a(v, \tilde{e}) = (e, v) \quad \text{for any} \quad v \in H_D^1(\Omega), \tag{4.24}$$

where $e$ is the exact error, defined as before.

**Theorem IV.5** *Let the solution* $\tilde{e}$ *of the dual problem (4.24) be* $H^2(\Omega)$*-regular. If the coefficients of our basic problem (2.4) are sufficiently regular, namely* $R_K$*,* $R_E$*,*

*$R_E^{in}$, and $R_E^{out}$ are correspondingly in $H^1(K)$, $H^{1/2}(\mathcal{E})$, $H^{1/2}(\Gamma_N^{in})$, and $H^{1/2}(\Gamma_N^{out})$, then the residual $L^2$ a posteriori error estimator from Definition IV.4 is reliable, i.e., there exists a constant $C$, independent of the mesh size $h$, such that $||e|| \leq C\tilde{\rho}$.*

*Proof.* For $v = e$ in (4.24), applying techniques similar to the ones for the residual based error estimator from Subsection 4.1.2, we get

$$\begin{aligned} ||e||^2 &= a(e,\tilde{e}) = R(\tilde{e} - e^*) \\ &= \sum_{K\in\mathcal{T}} \int_K R_K(\tilde{e} - e^*)\ dx - \sum_{E\in\mathcal{E}} \int_E R_E(\tilde{e} - e^*)\ ds \\ &\quad - \sum_{E\in\Gamma_N^{in}} \int_E R_E^{in}(\tilde{e} - e^*)\ ds - \sum_{E\in\Gamma_N^{out}} \int_E R_E^{out}(\tilde{e} - e^*)\ ds, \end{aligned} \tag{4.25}$$

where $e^* \in S_h^*$ is arbitrary. To evaluate the right hand side of identity (4.25) we use the nodal interpolation operator $I_h$ and its properties. If $\tilde{e} \in H^2(\Omega)$ the Sobolev inequalities [15, Theorem 4.3.4] guarantee that $I_h\tilde{e}$ is well defined. The properties of the interpolant are well established in the finite element literature (see, for example [15]). Namely, we have that

$$h_K^{-2}||\tilde{e} - I_h\tilde{e}||_{L^2(K)} + h_K^{-1}|\tilde{e} - I_h\tilde{e}|_{H^1(K)} + h_K^{-1.5}||\tilde{e} - I_h\tilde{e}||_{L^2(\partial K)} \leq C_{I,K}|\tilde{e}|_{H^2(K)}. \tag{4.26}$$

Now, in equation (4.25), we replace $\tilde{e} - e^*$ by

$$\tilde{e} - e^* = (\tilde{e} - I_h\tilde{e}) + (I_h\tilde{e} - I_h^* I_h\tilde{e}),$$

where $I_h^*$ is the nodal piecewise-constant interpolation operator, introduced in Section 2.3. We apply Schwartz inequality on the integrals involving $\tilde{e} - I_h\tilde{e}$ and use (4.26) to get the bound for $R(\tilde{e} - I_h\tilde{e})$:

$$
\begin{aligned}
R(\tilde{e} - I_h\tilde{e}) \lesssim \sum_{K\in\mathcal{T}} h_K^2 ||R_K||_{L^2(K)}||\tilde{e}||_{H^2(K)} + \sum_{E\in\mathcal{E}} h_K^{1.5}||R_E||_{L^2(E)}||\tilde{e}||_{H^2(K)} \\
+ \sum_{E\in\Gamma_N^{in}} h_K^{1.5}||R_E^{in}||_{L^2(E)}||\tilde{e}||_{H^2(K)} + \sum_{E\in\Gamma_N^{out}} h_K^{1.5}||R_E^{out}||_{L^2(E)}||\tilde{e}||_{H^2(K)}.
\end{aligned}
\tag{4.27}
$$

For the integrals involving $I_h\tilde{e} - I_h^* I_h\tilde{e}$ we first note that if $K$ is a fixed element in $\mathcal{T}$, then for every vertex $x_i$ of $K$, the quantities $|K \cap V_i|$ are the same. Also, for vertices $x_i$ of a face $E$ we have that the boundary quantities $|E \cap V_i|$ are the same. Therefore, we have the following equalities

$$
\int\limits_K (I_h\tilde{e} - I_h^* I_h\tilde{e})dx = 0, \qquad \int\limits_E (I_h\tilde{e} - I_h^* I_h\tilde{e})ds = 0.
$$

We use the last fact on the integrals involving $I_h\tilde{e} - I_h^* I_h\tilde{e}$ in order to subtract from $R_K$, $R_E$, $R_E^{in}$, and $R_E^{out}$ their mean values $\bar{R}_K$, $\bar{R}_E$, $\bar{R}_E^{in}$, and $\bar{R}_E^{out}$. Then, we use Schwartz inequality and Poincaré's inequality to get a bound for $R(I_h\tilde{e} - I_h^* I_h\tilde{e})$:

$$
\begin{aligned}
& R(I_h\tilde{e} - I_h^* I_h\tilde{e}) \lesssim \\
& \sum_{K\in\mathcal{T}} h_K||R_K - \bar{R}_K||_{L^2(K)}||\tilde{e}||_{H^1(K)} + \sum_{E\in\mathcal{E}} h_K^{1/2}||R_E - \bar{R}_E||_{L^2(E)}||\tilde{e}||_{H^1(K)} \\
& + \sum_{E\in\Gamma_N^{in}} h_K^{1/2}||R_E^{in} - \bar{R}_E^{in}||_{L^2(E)}||\tilde{e}||_{H^1(K)} + \sum_{E\in\Gamma_N^{out}} h_K^{1/2}||R_E^{out} - \bar{R}_E^{out}||_{L^2(E)}||\tilde{e}||_{H^1(K)},
\end{aligned}
\tag{4.28}
$$

where we have used the inequality

$$
\begin{aligned}
||I_h\tilde{e} - I_h^* I_h\tilde{e}||_{L^2(K)} &\lesssim h_K|I_h\tilde{e}|_{H^1(K)} \lesssim h_K|\tilde{e} - I_h\tilde{e}|_{H^1(K)} + h_K|\tilde{e}|_{H^1(K)} \\
&\lesssim h_K^2|\tilde{e}|_{H^2(K)} + h_K|\tilde{e}|_{H^1(K)} \lesssim h_K|\tilde{e}|_{H^1(K)}.
\end{aligned}
$$

Using (4.25), (4.27), (4.28), the stability of the dual problem with respect to the right hand side $||\tilde{e}||_{H^2(\Omega)} \leq C||e||$, and obvious manipulations, we get that the $L^2$ a posteriori error estimator $\tilde{\rho}$ is reliable. Moreover, for the stated in the theorem regularity, we can apply Poincaré's inequality to the terms

$$||R_K - \bar{R}_K||_{L^2(K)},\ ||R_E - \bar{R}_E||_{L^2(E)},\ ||R_E^{in} - \bar{R}_E^{in}||_{L^2(E)},\ \text{and}\ \ ||R_E^{out} - \bar{R}_E^{out}||_{L^2(E)}$$

to get one more power of $h$, which will make the error estimator of second order.

Note that we did not explicitly apply the Poincaré's inequality in the definition of the error estimator in order to make it well defined for problems with less that the stated in the theorem regularity. □

## 4.2. Estimators for upwind steady-state approximations

This is problem (2.14) with convection part determined by (2.13). The upwind approximation of the convection will bring additional error term, and we modify the above argument in the following way. From $a_h^{up}(u_h, v^*) = F(v^*)$ and $a_h(u, v^*) = F(v^*)$ for $v^* \in S_h^*$ we get the orthogonality condition:

$$a_h(u, v^*) - a_h^{up}(u_h, v^*) = 0.$$

Similarly to the residual based error estimator, the estimate for the error in the energy norm now becomes

$$\begin{aligned} c_0||e||_1^2 &\leq a(e,e) = a(e,e) - a_h(u,v^*) + a_h^{up}(u_h,v^*) \\ &= \{a(e,e) - a_h(e,v^*)\} + \{a_h^{up}(u_h,v^*) - a_h(u_h,v^*)\}. \end{aligned}$$

To estimate the term $a(e,e) - a_h(e,v^*) = R(e - v^*)$ we use the residual based error estimator from Subsection 4.1.2. For the second term we use the equality

$$a_h^{up}(u_h, v^*) - a_h(u_h, v^*) = C^{up}(u_h, v^*) - C(u_h, v^*)$$

to get

$$\begin{aligned} C^{up}(u_h, v^*) \quad & -C(u_h, v^*) = \\ & \sum_{x_i \in N_h^0} v_i^* \left\{ \sum_{j \in \Pi(i)} \int_{\gamma_{ij}} ((\underline{b} \cdot \underline{n})_+ u_h(x_i) + (\underline{b} \cdot \underline{n})_- u_h(x_j)) ds \right. \\ & \left. - \int_{\partial V_i \setminus \Gamma_N} \underline{b} \cdot \underline{n} u_h ds + \int_{\partial V_i \cap \Gamma_N^{out}} (\underline{b} \cdot \underline{n}\, u_h(x_i) - \underline{b} \cdot \underline{n}\, u_h)\, ds \right\}. \end{aligned}$$

Now we rewrite this sum of integrals over the volume boundaries $\partial V_i$ as a sum of integrals over $\gamma_{ij}$. First, for each $\gamma_{ij}$ we fix its normal vector $\underline{n}_{ij}$ so that $\underline{b} \cdot \underline{n}_{ij} \geq 0$ and the order of the indexes $ij$ is such that $(x_i - x_j) \cdot \underline{n} \leq 0$. This means that $\underline{n}_{ij}$ is the unit normal vector pointing outside of $V_i$ on $\gamma_{ij}$. This is shown on Figure 4. According to these notations if a finite element $K$ intersects the volume boundaries $\partial V_i$, $\partial V_j$, and $\partial V_k$, then for a given vector-function $\underline{F}$ we have the equality:

$$\begin{aligned} \sum_{l \in \{i,j,k\}} v_l^* \int_{\partial V_l \cap K} \underline{F} \cdot \underline{n} \quad ds &= (v_i^* - v_j^*) \int_{\gamma_{ij}} \underline{F} \cdot \underline{n}_{ij}\, ds \\ & + \quad (v_j^* - v_k^*) \int_{\gamma_{jk}} \underline{F} \cdot \underline{n}_{jk}\, ds + (v_k^* - v_i^*) \int_{\gamma_{ki}} \underline{F} \cdot \underline{n}_{ki}\, ds. \end{aligned}$$

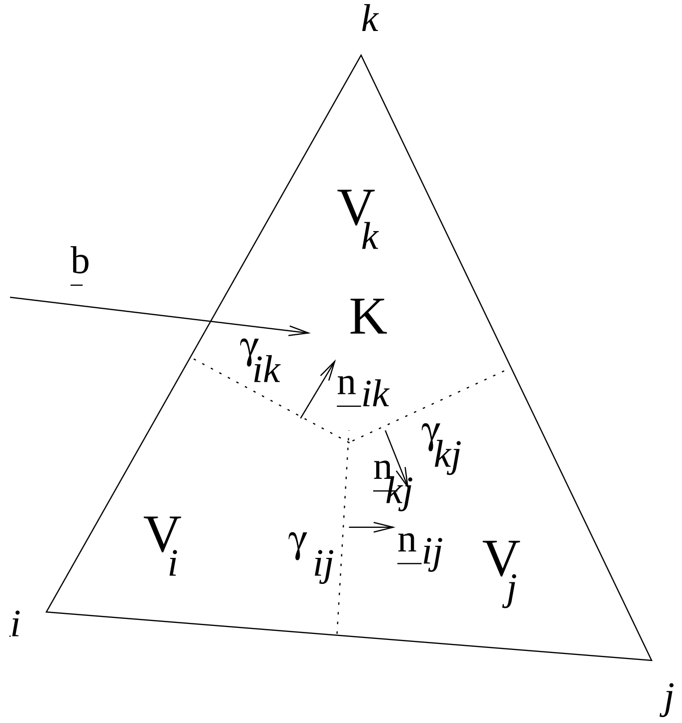


Fig. 4. Distribution of the normal vectors to $\gamma_{ij}$ in one finite element.

As a result, we get

$$\begin{aligned} a_h^{up}(u_h, v^*) \quad - \quad & a_h(u_h, v^*) = C^{up}(u_h, v^*) - C(u_h, v^*) = \\ = \quad & \sum_{K \in \mathcal{T}} \left\{ \sum_{\gamma_{ij} \subset K} (v_i^* - v_j^*) \int_{\gamma_{ij}} \underline{b} \cdot \underline{n}_{ij} (u_h(x_i) - u_h) \, ds \right. \\ & \left. + \sum_{V_i \cap K} v_i^* \int_{\partial V_i \cap \Gamma_N^{out}} \underline{b} \cdot \underline{n} \, (u_h(x_i) - u_h) ds \right\} . \end{aligned}$$

We denote $R_{\gamma_{ij}} \equiv \underline{b} \cdot \underline{n}_{ij}(u_h(x_i) - u_h)|_{\gamma_{ij}}$ and take $[v^*]$ to be the jump of $v^*$ across $\gamma_{ij}$. Then, using Schwartz inequality, the term that involves integral over $\gamma_{ij}$ is bounded by $C||[e-v^*]||_{\gamma_{ij}}||R_{\gamma_{ij}}||_{\gamma_{ij}}$, where $||\cdot||_{\gamma_{ij}}$ denotes the $L^2$-norm on $\gamma_{ij}$.

Further, we introduce the notations

$$\omega_K^\gamma = h_K^{-1/2}\left(\sum_{\gamma_{ij}\subset K}||[e-v^*]||^2_{\gamma_{ij}}\right)^{1/2}, \ ||R_{\gamma_K}|| = \left(\sum_{\gamma_{ij}\subset K}||R_{\gamma_{ij}}||^2_{\gamma_{ij}}\right)^{1/2}.$$

Again, using the local quasi-interpolant and its approximation properties, we get $\sum_{K\in\mathcal{T}}\omega_K^\gamma \le C_I||\nabla e||_\Omega$. Let $\underline{n}^\perp$ be a unit vector normal to $\underline{n}$. For the term involving integration over $\partial V_i \cap \Gamma_N^{out}$, we have $|u_h(x_i)-u_h| \le h_K|\nabla u_h\cdot\underline{n}^\perp|$. Also, taking $v^* = I_h^* v(e)$ as in (3.7), and using Schwartz and trace inequalities, we bound the term involving integration over $\Gamma_N^{out}$ by

$$\sum_{V_i\cap K} v_i^* \int\limits_{\partial V_i\cap\Gamma_N^{out}} \underline{b}\cdot\underline{n}\,(u_h(x_i)-u_h)ds \le Ch_K^{1/2}||\nabla e||_{\pi(K)}||\underline{b}\cdot\underline{n}\,\nabla u_h\cdot\underline{n}^\perp||_{\partial V_i\cap\Gamma_N^{out}}.$$

Combining all these estimates, we get a residual based error estimator with element-wise defined error $\rho_K$ similar to (4.8) with two additional terms due to the upwind approximation:

$$\begin{aligned}\rho_K := h_K||R_K||_{L^2(K)} + h_K^{1/2} & \Bigg( \sum_{E\in\partial K\cap\mathcal{E}}||R_E||_{L^2(E)} + \sum_{E\in\partial K\cap\Gamma_N^{in}}||R_E^{in}||_{L^2(E)} \\ & + \sum_{E\in\partial K\cap\Gamma_N^{out}}||R_E^{out}||_{L^2(E)} \\ & + ||R_{\gamma_K}|| + ||\underline{b}\cdot\underline{n}\,\nabla u_h\cdot\underline{n}^\perp||_{L^2(\partial K\cap\Gamma_N^{out})}\Bigg).\end{aligned}$$

### 4.3. An error indicator for transient problems

Here we extend the finite volume a posteriori error results to the time-dependent problem (2.7). The discretization is as given in Subsection 3.1.2. We solve the discrete problem (2.17), consider the difference between the solutions of (2.7) and

(2.17), and propose an error indicator for it in the global $L^2(\Omega)$-norm for every time interval $n = 1, \dots, M$. The approach is similar to the one taken by Eriksson et al. in [26, 27] and Thomee in [65], and follows the idea from the abstract a posteriori error analysis framework from Section 3.1. Although we show the main steps in proving the reliability of the error indicator, our goal is not to make a rigorous error analysis, but to show how to reduce the finite volume error analysis for transient problems to the finite element analysis that is given in [26, 27, 65].

**Definition IV.5** *We define the $L^2(\Omega)$ error indicator for time level $M$ as*

$$\rho_T := L_M \max_{1 \le n \le M} \Big( \tilde{\rho}_{I_n} + || \frac{h_n^2}{\triangle t_n} [u_h^{n-1}] || + || \frac{h_n}{\triangle t_n} [u_h^{n-1} - \bar{u}_h^{n-1}] || \\ + \triangle t_n ||R||_{I_n} + ||[u_h^{n-1}]|| \Big),$$

*where*

$$L_M := 2 + \max_{1 \le n \le M} \ \max \left( \sqrt{\log \frac{t_n}{\triangle t_n}}, \log \frac{t_n}{\triangle t_n} \right),$$

$|| \cdot ||_{I_n} = \max_{t \in I_n} || \cdot ||$, $\tilde{\rho}_{I_n} = \max_{t \in I_n} \tilde{\rho}(u_h^n)$ *with $\tilde{\rho}$ from Definition IV.4 and $R$ from (4.7).*

The finite volume a posteriori error analysis and its reduction to the finite element method analysis have the following main steps.

First, consider the dual problem : Find $\tilde{e}(x,t)$, $x \in \Omega$ and $t \in (0,T)$ such that

$$\begin{cases} -(\dot{\tilde{e}}, v) + a(v, \tilde{e}) = 0 \quad \text{for all} \ \ v \in H_D^1(\Omega), \ t \in (0,T) \\ \tilde{e}(T,x) = e(T,x) \quad \text{for all} \ \ x \in \Omega. \end{cases} \tag{4.29}$$

We use upper indices to denote the time levels (see the notations in Subsections 2.2.2 and 3.1.2) , i.e., $e^M = e(T,x) = e(t_M,x)$. Second, we multiply the initial condition of

(4.29) by $e^M$, take the $L^2(\Omega)$ norms on both sides, use that

$$\sum_{n=1}^{M} \int\limits_{I_n} -(\dot{\tilde{e}}, e) + a(e, \tilde{e})\ dt = 0,$$

and modify the last equality by integrating $(\dot{\tilde{e}}, e)$ by parts. Note that $e$ is discontinuous between the time levels, so the integration by parts produces the jumps of $e$ between the time levels. We get

$$\begin{aligned}
||e^M||^2 &= (\tilde{e}^M, e^M) \ + \ \sum_{n=1}^{M} \int\limits_{I_n} -(\dot{\tilde{e}}, e) + a(e, \tilde{e})\ dt \\
&= \sum_{n=1}^{M} \left\{ \int\limits_{I_n} (\tilde{e}, \dot{e}) + a(e, \tilde{e})\ dt + ([e^{n-1}], \tilde{e}^{n-1}) \right\} \\
&= \sum_{n=1}^{M} \left\{ \int\limits_{I_n} F(\tilde{e}) - a(u, \tilde{e}) + a(u - u_h, \tilde{e})\ dt - ([u_h^{n-1}], \tilde{e}^{n-1}) \right\} \\
&= \sum_{n=1}^{M} \left\{ \int\limits_{I_n} F(\tilde{e}) - a(u_h, \tilde{e})\ dt - ([u_h^{n-1}], \tilde{e}^{n-1}) \right\},
\end{aligned}$$

where we used that $\dot{e} = \dot{u} - \dot{u}_h = \dot{u}$ on any interval $I_n$, the weak formulation (2.8), and that $[e^{n-1}] = -[u_h^{n-1}]$. Third, using the last result, the approximation property (2.17), i.e.

$$\int\limits_{I_n} F(v^*) - \ a_h(u_h, e^*)\ dt - ([u_h^{n-1}], e^*) = 0 \quad , e^* \in S_h^*,$$

and the technique used in the proof of Lemma IV.1, the exact formula for the error in the $L^2(\Omega)$ norm becomes

$$||e^M||^2 \ = \ \sum_{n=1}^{M} \left\{ \int\limits_{I_n} R(\tilde{e} - e^{*,n})\ dt - ([u_h^{n-1}],\ \tilde{e}^{\ n-1} - e^{*,n-1}) \right\} \ := R^M(\tilde{e} - e^*)$$

for any $e^* \in S_h^{*,t}$. Then, the analysis proceeds as follows. If $\tilde{e}$ is sufficiently regular we take $\tilde{e} - e^*\Big|_{I_n}$ from the above error representation to be

$$\tilde{e} - e^*\Big|_{I_n} = \Big(\tilde{e} - I_h\tilde{e}\Big) + \Big(I_h\tilde{e} - PI_h\tilde{e}\Big) + \Big(PI_h\tilde{e} - I_h^*PI_h\tilde{e}\Big)$$

where

$$Pv\Big|_{I_n} := \frac{1}{|I_n|}\int\limits_{I_n} v\ dt.$$

We use the approximation properties of $I_h$, $I_h^*$, and $P$ to represent $R^M(\tilde{e} - e^*)$ as the computable error indicator $\rho_T$ times quantity of the solution $\tilde{e}$ of the continuous dual problem (4.29), which quantity is bounded by $||e^M||$ by assuming proper regularity of the dual problem. A rigorous proof will involve technical details and complicated regularity analysis, which is not in the scope of this dissertation.

# CHAPTER V

## COMPUTATIONAL ASPECTS OF THE ADAPTIVE METHODS

An important and challenging component of the adaptive methods is their practical computer implementation. The problems in this direction are related to computer science in more than one way. The difficulties come from the three specific requirements for our implementation. First, our computations are mainly targeted to 3-D problems. Second, we consider time dependent problems. And third, we require the computations to be done in parallel. Therefore, our research in the adaptive methods is targeted not only in their mathematical aspects, but also in their computational aspects.

Although there are some ready, stand alone tools that can be used for adaptive computer simulations, they are usually “black boxes” that are difficult to change and adjust to specific problem requirements. This motivated our interest in the development of new tools, and their integration with already existing tools, for large scale parallel adaptive computations. Some of our results in this direction are summarized in the technical report “Tool-box for large scale parallel computations of 3-D fluid flow problems using domain decomposition” [66]. Here we extend the results and describe the tools that we have used, developed, and implemented in a computer system for the parallel adaptive numerical solution of steady and transient convection-diffusion-reaction problems.

Our goal was to create a simulator that uses various tools and that is based on

- discretization techniques utilizing finite element and finite volume discretizations for 2 and 3-D problems;
- efficient preconditioning of the resulting algebraic system;

- error control and adaptive grid refinement;
- parallel implementation on a multiprocessor computer system utilizing the concept of domain decomposition data distribution.

We created a tool, named ParaGrid, which achieves the goal that we described above. We group the computational difficulties in designing and implementing ParaGrid, into the following main components:

- adaptive mesh generation;
- mesh partitioning and load balancing;
- parallel computations;
- data structures;
- visualization.

Parallel grid generation tools play an important role in the scientific research that requires the power of high performance parallel computers. To enable the development of efficient computational technologies such tools may have to generate finer meshes only in some regions of the computational domain. We describe the challenges and the solutions in developing such tools in the "adaptive mesh generation" component (Section 5.1). We explain our mesh generation strategy, and concerning the local refinement, give the algorithms that we have used for element refinement and derefinement. The next main component, "mesh partitioning and load balancing" (Section 5.2), is about the parallel issues in developing parallel grid generation tools. The main concerns here are to (1) find a balanced among the processors partitioning that minimizes in certain sense the interface between the processors, and

(2) keep the above partitioning property during the refinement process. The "parallel computations" component, given in Section 5.3, gives an introduction to writing efficient parallel finite element/volume software in shared (using OpenMP) and distributed (using MPI) memory computational environment. The next section, Section 5.4, gives the overall code organization and the maintained data structures. The section concerns the "efficient preconditioning of the resulting algebraic system". The last section, Section 5.5, is about the visualization of the obtained numerical results. It is essential to have a visualization tool to help in the analysis of the obtained results. In the area of the adaptive methods their availability is highly appreciated even for debugging purposes, where one would like to see if the refinement is in areas of where the solution singularities are. The available visualization tools often can not be modified and tuned for specific applications, which may be needed for 3-D problems, numerical results streaming in parallel from subdomains, or time dependent problems, etc.

## 5.1. Adaptive mesh generation

Finding a "good" mesh is one of the key elements in the development of any efficient computational methodology that is based on finite element or finite volume methods. There are many available stand alone mesh generators. A survey in the field is given by Owen [49]. Despite the large variety, not many of the available software products are directly usable for research purposes in the adaptive methods. For example, not many of the products are in the public domain, some do not support adaptivity, and most of them do not support parallelism. What we need is a mesh generation tool that

- produces quasi-uniform grids;

- supports adaptivity;
- adaptively maintains the mesh in parallel;
- provides decomposition of the domain (from the splitting among the processors) and multilevel (from the adaptive process) data structures suitable for domain decomposition and multigrid/multilevel type preconditioners.

We have developed such parallel adaptive mesh generation tool. The generator is named ParaGrid. Its features are discussed in Subsection 5.1.1.

ParaGrid uses as input a coarse mesh. In 2-D this initial mesh is obtained by the stand alone mesh generator `triangle`. It generates exact Delaunay triangulations, constrained Delaunay triangulations, and quality conforming Delaunay triangulations [61]. Triangle is freely available for non-commercial use from Internet address `http://www-2.cs.cmu.edu/~quake/triangle.html`.

For 3-D domains we obtain the initial mesh by the stand alone tetrahedral mesh generator `NETGEN`. It is based on the advancing front method and is developed by Schöberl [60]. The input is a 3-D domain described by boolean operations (*or*, *and*, *not*) on primitives, such as planes, cylinders, spheres, cones, etc., which allow generation of complicated domains, for example the one on Figure 5. `NETGEN` is freely available for non-commercial use from Internet address `http://www.sfb013.uni-linz.ac.at/~joachim/netgen/`.

Subsection 5.1.2 and Subsection 5.1.3 discuss correspondingly the refinement and derefinement algorithms that we have used in our implementation.

### 5.1.1. ParaGrid, a parallel adaptive mesh generator

We developed a tool, named ParaGrid, for parallel adaptive mesh generation. The development started from a 2-D project assignment of a summer internship at the

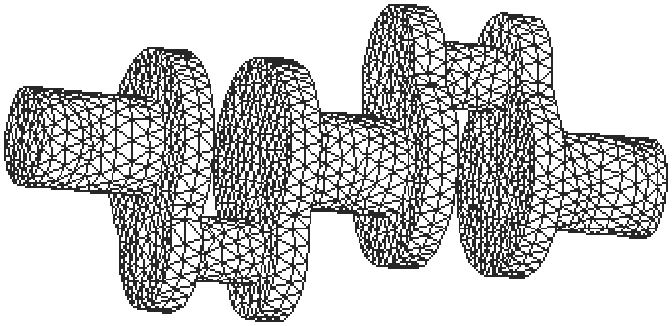


Fig. 5. Crankshaft mesh. The domain is split into 5, 830 elements, with 4, 176 surface elements and 2, 140 vertices.

Center for Applied and Scientific Computing (CASC) at Lawrence Livermore National Laboratory. Then it was further developed for 3-D tetrahedral meshes. ParaGrid is software that takes as input a coarse tetrahedral (or triangular in the 2-D case) mesh, which describes well the domain, splits it using MeTiS, distributes the partitioning among the available processors, and generates in parallel a sequence of meshes. It has its internal solvers and is able to generate various Finite Element/Volume discretizations. The maintained data structures allow ParaGrid to be easily connected to (or used to provide data to) external parallel finite element/volume solvers based on domain decomposition data distribution. It has been successfully used by several researchers in CASC for algorithm testing purposes.

The refinement in ParaGrid is based on the a posteriori error indicators developed in Chapter IV. The estimators can be used in various computational strategies. For example, one may vary the refinement/derefinement element selection strategy, as explained in Algorithms III.1 and III.2. Also, the error estimators can be used to obtain load balance among the processors in parallel computations, as given in Subsection 5.2.2.

The mesh can be maintained globally conforming or left to be non-conforming between the subdomains. The latter possibility is faster and can be used for mortar type approximations. In both cases, every processor keeps an array of faces on the interfaces between the subdomains. In the non-conforming case this allows us to compute for example integrals like $\int\limits_F \phi_i \psi_j dx$, where $F$ is face on the subdomain interface, $\phi_i$ and $\psi_j$ are basis functions associated with the different sides of the interface. In the conforming case the interface faces are ordered in the same way from both sides of the interfaces. This gives the face correspondence from the two sides of the interfaces. It is used for example to make the mesh conforming between the subdomains.

A narrative version of the parallel algorithm that produces conforming between the subdomains mesh is given as follows.

**Algorithm V.1** *This parallel algorithm refines/derefines certain marked elements and then additionally refines in order to produce a mesh that is conforming between the subdomains. The following is run in parallel by every processor:*

- *refine/derefine the locally marked elements;*
- *additionally refine until local conformity is reached;*
- *send packages to the neighboring subdomains for every interface, giving how many times each of the faces in that interface has been bisected;*
- *receive similar information and check if refinement is needed;*
- *if yes, then refine the elements that need refinement and refine additionally, until local conformity is reached;*
- *repeat the above process until no subdomain has performed refinement.*

The data structures that we needed in order to implement Algorithm V.1 are discussed in Section 5.4.

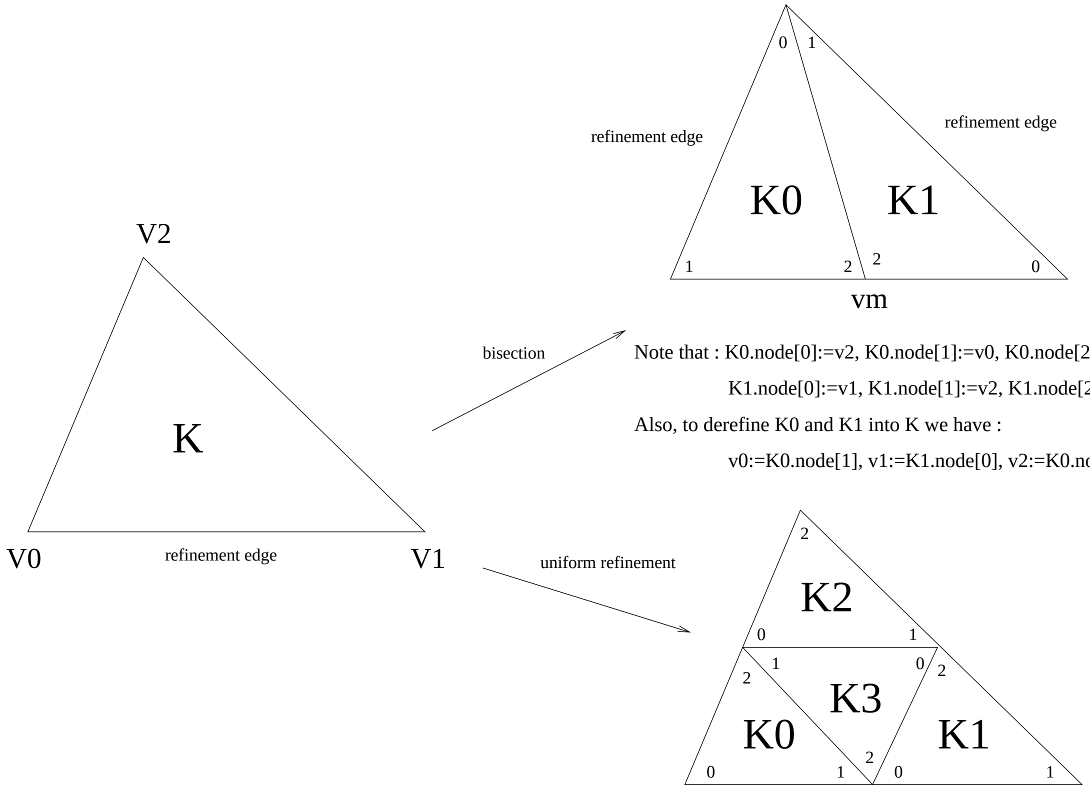
After conformity is reached we use MeTiS to check the load balance. The input is the graph of the starting mesh with nodes being the elements. Every node has weight, which is the number of children that the element has. If an element has to migrate in order to maintain the load balance the element migrates along with its children. A more detailed description is given in Section 5.4.

The finite element refinement is given in Subsection 5.1.2 and the finite element derefinement, in Subsection 5.1.3.

### 5.1.2. Element refinement algorithms

We have implemented two refinement strategies for 2-D problems. The standard one is uniform splitting into 4 and the alternative is element bisection (see Figure 6).

We sort the edges on the starting mesh by length and then select for every element the longest edge as a "refinement edge". Refinement edge is the edge that will be bisected if needed. It is maintained for every element during the adaptive process: (1) if we do bisection, then the refinement edges in the new elements are selected to be opposite to the new vertex, and (2) if we do uniform refinement, then the refinement edges in the new elements are inherited from the parent (see Figure 6). The refinement edge is not explicitly stored. It is maintained between element's vertices 0 and 1. This means that after refinement, the new elements have vertices as shown on Figure 6.



Note that : K0.node[0]:=v2, K0.node[1]:=v0, K0.node[2]:=vm

K1.node[0]:=v1, K1.node[1]:=v2, K1.node[2]:=vm

Also, to derefine K0 and K1 into K we have :

v0:=K0.node[1], v1:=K1.node[0], v2:=K0.node[0]

V2
K
V0
refinement edge
V1
bisection
uniform refinement
refinement edge
refinement edge
K0
K1
0
1
1
2
2
0
vm
K2
K3
K0
K1
2
0
1
1
0
2
2
2
0
1
0
1

Fig. 6. Refinement, de-refinement, and maintenance of the refinement edge in 2-D. The refinement edge is kept always between local vertices 0 and 1. When bisecting the new refinement edges are opposite the new vertex `vm` and for uniform refinement they are kept as in the parent. The de-refinement of elements obtained by uniformly refinement is similar to the de-refinement of elements obtained by bisection: `v0:=K0.node[0]`, `v1:=K1.node[1]`, and `v2:=K2.node[2]`.

The repeated application of the algorithm does not lead to element shape degeneration. For a given element (see Figure 7) one can get at most four types of shapes, which are the original one, denoted by $A$, and also $B$, $C$, and $D$, as shown on the figure.

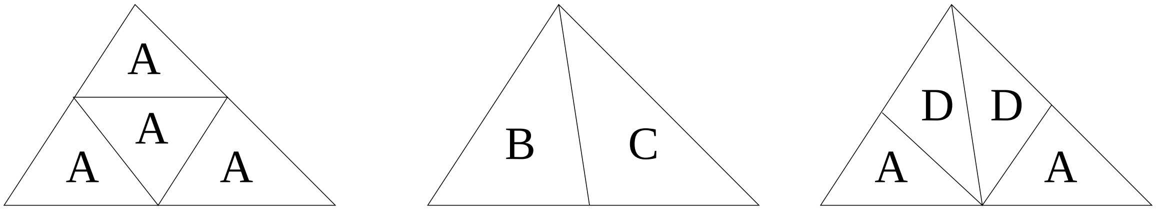
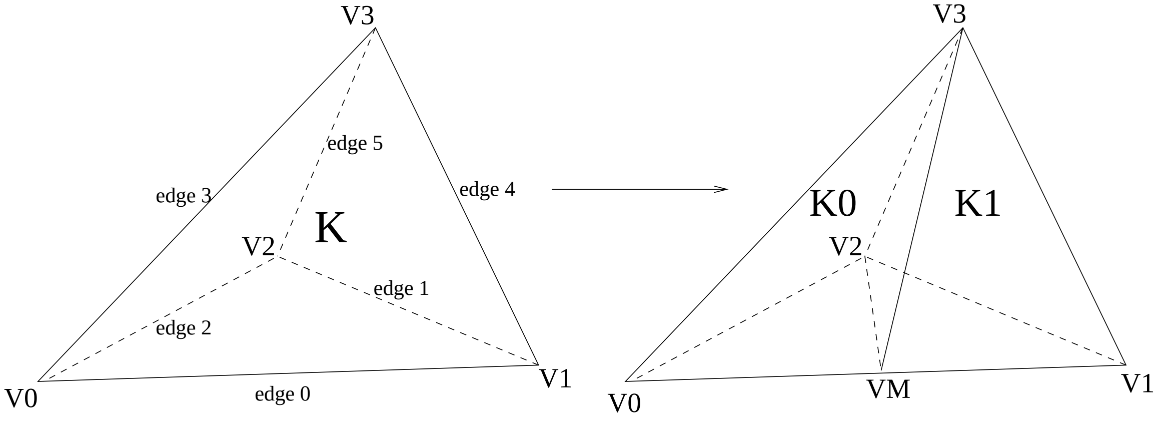


Fig. 7. The 4 different types of shapes obtained in the process of uniform refinement or bisection. Note that bisecting shape $D$ leads to shapes $B$ and $C$.

In the 3-D case we use the bisection algorithm described by Arnold et al. in [4]. The general idea is similar to the one in the 2-D bisection. First, we sort the edges by length on the coarse mesh. Then, we select the longest for every element edge as “refinement edge”. This is the edge that will be bisected if we mark the element for refinement. Additionally, we select in every face the longest edge and denote it as “marked edge” (`ME`). We introduce a local numbering for the edges of an element $K$, which ordering is according to the ordering of the nodes (see Figure 8), or more precisely, edge 0 is between vertices `V0` and `V1`, edge 1 is between `V1` and `V2`, etc.



Left sub-element, K0 :

```
K0.type = (K.type + 1)%3
switch (ME for face V0 V2 V3)
    case 2 :  K0.node = {V0, V2, V3, VM}
              if (K.type == 2) ME = {2, 4}
              else             ME = {2, 1}
    case 3 :  K0.node = {V3, V0, V2, VM}
              if (K.type == 2) ME = {1, 3}
              else             ME = {1, 2}
    case 5 :  K0.node = {V3, V2, VM, V0}
              ME = {3, 4}
```

Right sub-element, K1 :

```
K1.type = (K.type + 1)%3
switch (ME for face V1 V3 V2)
    case 1 :  K1.node = {V2, V1, V3, VM}
              if (K.type == 2) ME = {1, 3}
              else             ME = {1, 2}
    case 4 :  K1.node = {V1, V3, V2, VM}
              if (K.type == 2) ME = {2, 4}
              else             ME = {2, 1}
    case 5 :  K1.node = {V2, V3, VM, V1}
              ME = {3, 4}
```

Fig. 8. Algorithm for bisection of a tetrahedron. Here `ME` abbreviates marked edge. Note that in the refinement process the vertices are ordered so that the orientation of the elements is preserved.

The refinement edge is kept always between vertices `V0` and `V1`, i.e., edge 0. Our refinement implementation does not need the edges or the faces to be explicitly generated. The element has the four integers that give the node indices in specific order and one more integer, which we use bitwise in order to save several flags. We have two flags, `ME[0]` and `ME[1]`, that give the marked edges for faces $\overline{\mathtt{V0V2V3}}$ and $\overline{\mathtt{V1V3V2}}$ (for the other 2 faces edge 0 is the marked one). These flags are integers from 1 to 5, so to store them we need 3 bits for each of them. Two more bits are used for the "element type" flag (see [4]) and the rest, if needed, is used for element attribute. The refinement edges and the marked edges are maintained in the adaptive process according to the algorithm in [4], which is summarized in a pseudocode on Figure 8. The element types on the coarse level are initialized in the following way. If `K.ME={1,2}` then `K.type = 1`, else `K.type = 3` for every $K \in T_h$. The repeated application of the bisection algorithm, that we described above, does not lead to element shape degeneration.

### 5.1.3. Element de-refinement

In 2-D we can derefine two elements, `K0` and `K1`, or four elements, `K0`, `K1`, `K2`, and `K3`, into an element $K$ if they have been obtained by correspondingly bisection or uniform refinement of $K$. In order to have such information we store the history of the refinement process. The implementation uses a tree data structure (see Section 5.4). Also, we have the elements, that have to be derefined, ordered in the way that some previous refinement process produced them (see Subsection 5.1.2). This allows us to determine the vertices and their order in the original element $K$. We show how to do this on Figure 6.

In 3-D we have a similar situation. For example, from element `K0`, since the ordered pair `ME` is unique (see Figure 8), we can determine uniquely nodes `V0`, `V2`, and `V3`. Also, `ME[0]` and `K.type = (K0.type + 2)%3`. The rest is similarly obtained from `K1`.

## 5.2. Mesh partitioning and load balancing

Mesh partitioning and load balancing algorithms are the key ingredients for an efficient parallel solution methodology that is based on domain decomposition methods.

Domain decomposition methods use a *divide-and-conquer* concept, which main idea is to split the global problem into sub-problems, solve them concurrently, and somehow merge or combine the local solutions in order to get the global one. Such idea translates into first, finding a splitting $\{T_{i,h}\}_{i=1}^N$ of the global mesh $T_h$, then mapping every $T_{i,h}$ to a processor, doing independent computations on each $T_{i,h}$, and transferring data when necessary. Crucial for the efficient parallel execution of software based on this technique is obviously the quality of the splitting $\{T_{i,h}\}_{i=1}^N$. It should be such that

- there is *load balance* among the processors, and
- the interface between $\{T_{i,h}\}_{i=1}^N$ is minimal in a certain sense.

In order to have load balance over the processors the number of tetrahedrons in each $T_{i,h}$ should be almost equal. Also, in order to reduce the communication, the number of nodes on the boundary between the sub-domains should be minimal. Although this problem is *NP-complete*, many relatively simple and effective heuristic methods have been devised (see the survey [25]). One way to approach the problem is to model the mesh by a graph, and then using a graph partitioner to partition it into equal parts, with number of edge-cuts minimized. The enormous interest in parallel computing explains the variety of available graph partitioning libraries. In Subsection 5.2.1 we mention some of the most popular and briefly discuss the one that we use. In the next two subsections, Subsection 5.2.2 and Subsection 5.2.3, we explain our strategies to maintain load balance during the adaptive mesh refinement process for correspondingly steady-state and transient problems.

### 5.2.1. Mesh partitioners

A simple "mesh partitioner" search on Internet gives a wide choice of mesh partitioning software. Some of the most popular are:

- MeTiS – developed by George Karypis and Vipin Kumar from the University of Minnesota;
- CHACO – developed by Bruce Hendrickson and Robert Leland from Sandia National Laboratory;
- JOSTLE – developed by Chris Walshaw from the University of Greenwich, London.

All of the mentioned partitioning libraries handle unstructured meshes (graphs) and can be used in parallel to repartition existing partitions. The availability of the latter characteristic is essential in the parallel adaptive mesh refinement since it is used to maintain the load balance during the adaptive process.

In our work we have used MeTiS, a software package for partitioning large irregular graphs/meshes and computing fill reducing orderings of sparse matrices. The algorithms in MeTiS are based on multilevel graph partitioning, where the graph is consecutively coarsen, the coarse graph partitioned and then the computed partitions projected into the fine graph.

We have used the graph partitioning part of MeTiS, where the vertices of the graph are the finite elements and the graph links are the common for the elements edges or feces, correspondingly in 2 or 3-D. MeTiS provides two programs to partition graphs into $k$ equal parts: `pmetis` and `kmetis`. The first one, `pmetis`, is based on multilevel recursive bisection algorithm (see [40]) and is preferable for partitioning graphs into a small number of sub-domains. The second one, `kmetis`, is based on

$k$-way partitioning (see [39]) and is preferred when the graph has to be split into more than eight parts. Our computational observations are that both programs produce well balanced meshes, with `pmetis` being better for small values of $k$, and `kmetis` being slightly better for large values of $k$. Also, `kmetis` often produces disconnected domains, especially for large $k$. In general, `pmetis` produces "smoother" and "better connected" domains.

Figure 9 shows the domain from Figure 5 split into 4 by `pmetis`. The domain is split at places where one intuitively would do it in order to have balance of the elements in the sub-domains, and minimum interface between the sub-domains.

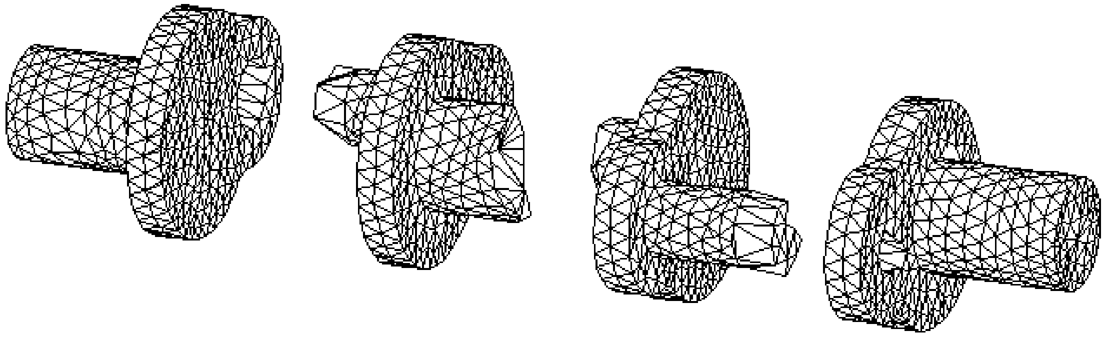


Fig. 9. Crankshaft mesh split in 4 by `pmetis`. The subdomains, starting from left, have correspondingly 1458, 1457 , 1458, and 1457 tetrahedrons.

### 5.2.2. Load balancing for steady-state problems

If we have a fixed partition of the domain, the repeated application of local adaptive refinement will in general lead to load disbalance among the processors. Therefore, we need algorithms that will dynamicly maintain the load balance. We have tested three techniques. The first two are narratively described as follows, and the third one, which is based on element migration between the subdomains, is explained in Subsection 5.2.3. In all the cases we start with a “good” coarse mesh that has been produced by `triangle` for 2-D domains and `NETGEN` for 3-D domains. The coarse mesh is read by all the processors, so at the beginning every processor stores a copy of the initial mesh.

**Strategy V.1** *Every processor solves the problem, produces a posteriori error estimates, and based on them, adaptively refines its copy of the mesh until some error tolerance, for example* 16 *times bigger than the targeted tolerance, is reached. Such refinement is used to locally adapt the mesh to the singularities of the solution. The still coarse mesh is split into subdomains using MeTiS and every sub-domain is “mapped” to a processor. From this point every processor handles its subdomain and all the computations are done in parallel. Uniform parallel refinement follows until the* **desired error** *tolerance is reached.*

The uniform refinement steps in the second part of the strategy guarantee that the load balance will be maintained on the different refinement levels. The benefits of such strategy are that it is easy to implement and there is load balance among the processors. The weakness is that the efficiency is not optimal since a local refinement that is entirely led by a posteriori error analysis would have produced a mesh with less degrees of freedom for the same error tolerance.

**Strategy V.2** *Again, every processor solves the problem on the coarse mesh. The solution is used to compute a posteriori error estimates for every element of the mesh, which estimates are used as weights in an element based splitting of the coarse mesh into sub-domains (using MeTiS). Then, every sub-domain is “mapped” to a* *processor. This is the point where the parallel computations start. Every processor adaptively refines its region using the sequential Algorithm III.1 (plus synchronizing communications with the other processors).*

The weighted graph splitting in Strategy V.2 heuristically insures that the local refinements that follow will produce computational mesh with balanced over the subdomains number of elements. The strategy is easy to implement and produces meshs that are entirely based on a posteriori error analysis. It is still not optimal in the sense that the load balance may not be optimal on the different refinement levels.

A strategy that removes the weaknesses of the first two, but is much harder to implement, involves element migration between the subdomains, and is described below.

### 5.2.3. Load balancing for transient problems

Strategies V.1 and V.2 are feasible for steady state problems and not applicable to transient problems. For transient problems we usually have moving fronts and the load balance varies very dynamically along the adaptive algorithm. For such cases we need to do element "migration" between the subdomains in order to maintain load balance. The strategy that do it is described as follows.

**Strategy V.3** *Split the initial mesh into subdomains using MeTiS and "map" every subdomain to a processor. Every processor starts a parallel version of Algorithm III.2. The parallel modification of the algorithm is that on every time step ParMeTiS is used to check the load balance of the current mesh. This is done by assigning a weight to every element of the coarse mesh, which is the number of "children" that the element has due to the refinement process, and repartitioning the so obtained weighted mesh. If the ownership of coarse elements has to be changed then they move, along with their* *children, from their old to their new subdomain.*

This strategy is used also in the steady-state case, but the load balance is checked on the finest level. If there is need for finest elements migration, then they are moved to their new subdomain. In the time dependent case, the need for derefinement determines that the migration be on the coarse level (along with the children), as described in the strategy above. A strength of such approach is that it reduces the number of communications, since the threshold for a migration to take place is higher.

## 5.3. Parallel computations

This section describes the machinery that we used and developed for writing parallel finite element/volume code. We start with some fundamental scalar code optimization concepts, associated with modern parallel architectures (Subsection 5.3.1). We discuss the importance of the optimization and show how it may help in a finite element/volume code. Subsections 5.3.2 and 5.3.3 give some key concepts in implementing parallel domain decomposition. Also, they may serve as an introduction in writing parallel finite element/volume software under correspondingly shared and distributed memory paradigms.

All illustrations and examples of how to use OpenMP/MPI are for C/C++ applications. When we use machine performance values they are limited to *SGI Origin 2000* system with Mips R10000 processors running at 250MHz, 4MB L2 cache each.

### 5.3.1. Code optimization

There are two main concepts, whose understanding is vital for writing efficient scalar code. These are *Locality of reference* (important in improving memory performance) and *Software pipelining* (important in improving the CPU performance). We will discuss both of them, but the stress will be on the first one, since it's more critical and less machine dependent.

### 5.3.1.1. Locality of reference

The importance of the concept *Locality of reference* (or simply the rule "Keep things that are used together, close together") arises from the *memory hierarchy*, and more precisely from the different times of accessing data on the different levels in this hierarchy. For example, to access the fastest memory for the machine specified above, the registers, it takes 0 CPs (clock period or cycle), the L1 cache 3 CPs, the L2 cache 6 CPs, and the main memory $\approx$ 200 CPs. The mechanism of using the cache memory is the following. When a program refers to data that is not in the cache the CPU requests a load of a *cache line*, which is a block of 128B (the L1 cache line is 32B), i.e., referring an element from array of doubles will bring a block of 16 elements in the L2 cache and 4 into the L1 cache. Due to the *pipelining* architecture (explained below) the CPU can often continue working while accessing data from the main memory, but still multiple successive cache misses can bring the effective work to a halt because of waiting for data. That is why, for example, a simple computation like

```
A[i][j][k] += B[i][j][k] * C[i][j][k]
```

executed in loop order $i, j, k = 0..100$ is more that 10 times faster than the same computation performed in loop order $k, j, i$.

Programs that effectively apply the principle *Locality of reference* are sometimes called *cache friendly*. There are many techniques to develop *cache friendly* programs. One, as we discussed above, is the loop interchange. Other possibilities are : (1) split computations over large datasets into computations over "data blocks" that entirely fit in the cache memory (technique called *blocking*); (2) group frequently used data fields into single object so they tend to stay in the cache; (3) avoid searching linked lists (especially of big objects); (4) use `memalign()` to allocate important objects on the 128B boundaries and so on.

Developing *cache friendly* finite element packages, when iterative solvers are used, is important. Some of the techniques that we have used in our implementation are

- *vertex reordering* – the locality is increased by reordering the vertices. A simple algorithm, which we implemented is *Cuthill-McKee* (see [58]);
- *blocking* – as we mentioned above this is a technique where the data set is split into blocks, so that the blocks fit into the cache. For parallel computations, when applying *divide-and-conquer* technique, the cache used on one processor for solving a local problem is greatly utilized. This technique is another motivation for using domain decomposition data distribution for parallel computations;
- *fusion* – technique where multiple loops are merged into one. We used this technique to incorporate certain vector operations into other loops (in the CG and PCG solvers), which allowed us to re-use certain vectors.

We show how blocking improves the cache memory use in [66]. The other two techniques, in a test problem of size 200000 unknowns, gave a speed improvement of a CG solver from 864 seconds to 742 seconds, which includes the time for reordering.

#### 5.3.1.2. Software pipelining

The other main concept in code optimization, *Software pipelining* (SWP), is important in improving the CPU performance. Such improvement is machine dependent and in R10000 is possible because some of its functional units are pipelined. A pipelined unit is unit partitioned into independent hardware subunits, each specializing a specific phase (to be executed in one CP) of the operation that the unit performs. Thus, a unit partitioned in 4 will finish a pipelined sequence of $n$ operations in $n+3$ CPs, compared to $4n$ CPs for non-pipelined operations. Now, SWP tries to find a valid rearrangement of instructions (from the **innermost loops only**) so that to engage concurrently as many pipelines as possible. SWP is carried out by the compiler and is enabled if compiling options **-r10000 -O3** are used. The use of SWP in computing `A[i][j][k]+=B[i][j][k]*C[i][j][k]`, $i, j, k = 0..100$ increased the CPU performance from 13 Mflops to 1115 Mflops, making it $\approx 100$ times faster.

Some of the cases when SWP looses efficiency are : (1) data dependences occur; (2) long loops; (3) branching (function calls, goto and if-else statements); (4) low iteration counts. Some of the possible techniques to improve SWP, that we have used in our implementation, are

- *inlining* – in general one inlines only small functions, which are called many times within loops, for example, inner-product, vector addition/subtraction functions;
- *splitting/fusing* – splitting wide loops may sometimes enhance prospects for SWP; fusing small loops provides a richer variety of instructions to schedule efficiently with SWP (example for applying the technique in the FE software was given in Subsection 5.3.1.1);
- *outer loop unrolling* – provides richer variety of instructions in the innermost loop (done by increasing the step in the outer loop and explicitly writing the missed iterations in the inner loop); May be combined with *prefetching*, which is, assigning array elements, that are frequently used in inner loops, to local variable (done in outer loop) in order to benefit from register reuse; Prefetching

is a technique that is used very often.

#### 5.3.1.3. Performance monitoring

Performance monitoring tools are very useful in code optimization. Optimization efforts should be spent on the most time consuming routines. A performance routine that we have used is **ssrun**. It may be used to get information on the amounts of time that a program spends on the different routines. For example, to monitor ParaGrid, we compile without optimizations and execute

```
> ssrun -fpcsampx ParaGrid
> prof -lines ParaGrid.fpcsampx.m####
```

where `####` stands for process ID. The output, in the case when no *fusing* or *inlining* (for vector addition and inner-product) is done, provides the following program execution statistics

```
 [index]      secs     %    cum.%   samples  function (dso: file, line)
     [1]    85.497  58.8%   58.8%     85497  Matrix::Action(double*,
     [2]    12.411   8.5%   67.3%     12411  Method::vvadd(int,doubl
     [3]    11.567   8.0%   75.3%     11567  Method::inprod(int,doub
```

To see the values of various hardware performance indices one can use

```
> perfex -a -x -y ParaGrid
```

The output includes statistics for performance indices such as cache line reuse, cache misses, loads/stores, issued instructions, Mflops reached, and so on.

More information about code optimization can be found in [50].

### 5.3.2. Parallel FEM/FVM using OpenMP

OpenMP is a standard agreed upon by major hardware and software vendors. It consists of a portable set of compiler directives, library routines, and environment variables that can be used to specify shared memory parallelism in Fortran and C/C++. Advantages of using OpenMP are that it's simple, portable, and has flexible interface. In Subsection 5.3.1 we stressed on the optimization importance. Its importance for parallel computations is even greater. For example, it's reasonable to expect that the following fragment of Matrix-vector multiplication ($y = Ax$) is scalable.

```
#pragma omp parallel for private(j, end)
for(i=0;i<dimension;i++){
  end  = V[i+1];
  y[i] = 0;
  for(j=V[i]; j < end; j++)
    y[i] += A[j]*v1[VV[j]];
}
```

In practice, tests on machines with Non-uniform memory access (NUMA) architecture show that the parallel execution may be even slower than the consecutive. One of the problems is that the used arrays are not properly distributed among the processors, which is crucial for the performance of a NUMA architecture machine.

The problem about the data distribution for NUMA machines is solved on software [1] level by introducing new pragmas, namely `#pragma distribute` and `#pragma distribute_reshape`. Both pragmas are used to allocate the memory for an array among the local memories of the processors. The first one, called *regular distribu-*

---

[1] On hardware level there is the so called *page migration*, which is automatic reallocation in the main memory of whole pages (16KB)

*tion*, is constrained to distribution in terms of pages (16KB) and is useful only if each processor's portion of the array is substantially larger then the page size. The second one, called *reshaped*, overcomes the page-level constraints for the cost of certain restrictions on the usage of the reshaped arrays. Once distributed, the arrays may be redistributed using `#pragma redistribute`. For more information on the subject see [62], Chapter 5.

Having discussed the crucial issue of data distribution, we concentrate again on the development of parallel FE/FV software. A naive approach to parallelize a sequential FE/FV code, using the discussed until now techniques, is to use the sequential one and add some parallel constructs. The algorithm is as follows. First, parallelize the loops using the OpenMP directives and functions. Second, increase the locality by some vertex reordering technique. Finally, improve the data distribution. Such strategy may look feasible, but the practical implementation shows that the appeal is only theoretical. The problems are :

- *inefficient memory usage* – data distribution can be applied only to statically allocated arrays (fixed size), whereas in an adaptive code the memory allocation is usually dynamic. To overcome this problem one may have to allocate "bigger" arrays and dynamically redistribute them on each refinement level using *block-cyclic* distribution with blocks of size $\left\lceil \frac{\texttt{\# nodes}}{\texttt{\# processors}} \right\rceil$. Also, in an adaptive code one would like to make use of the different refinement levels by defining multilevel preconditioners, which could further worsen the inefficient memory usage;

- *poor scalability* – due to many synchronization points. Also, in Matrix-vector product ($y = Ax$) even in a perfect data distribution one processor still have to access parts of $x$ residing on the local memories of other processors. We also suspect that there is computational overhead using reshaped arrays;

- *programming efforts* – the use of reshaped arrays requires change of the syntax of many functions, more precisely, it requires specific format for passing such arrays as function arguments.

Based on the mentioned problems and our computational experience, the conclusion is that data placement is unlikely to help for direct parallelization of finite element/volume code and redesign of the algorithm is needed in order to get efficient, well scaled software. The choice is to use decomposition of the domain (see Section 5.2) into subdomains. The computational model is similar to the model for shared memory machines (see the MPI Subsection 5.3.3) and uses only a few basic OpenMP directives and functions. For example, here is a fragment of our main function.

```
#include <omp.h>
 . . .
Subdomain **S;
main(){
  int np;                          // # of available processors
  cout << "Insert the number of processors : "; cin >> np;
  cout << endl << "Input the file name : "; cin >> fname;
  Mesh m(fname, ...);              // Initialize mesh, where fname
                                   // is the output from NETGEN.
  int *tr = new int[m.GetNTR()];
  m.DomainSplit(np, tr);           // Split the domain into "np''
                                   // domains using MeTiS.
  S = new PSubdomain[np];
  omp_set_num_threads( np);        // Set the number of threads
  #pragma omp parallel            // Parallel region - used to
  {                               // initialize the sub-domains
    int myrank;
    myrank = omp_get_thread_num(); // Get the thread number
    S[myrank] = new Subdomain(myrank, &m, tr);
  }
  . . .
}
```

Note that in this model we do not use the complicated OpenMP data distribution constructs. Instead we use the so called *first touch* rule, which guarantees that memory is allocated to the processors that are the first to access or touch the data. In the example above processor with thread number `myrank` allocates in its local memory sub-domain with index `myrank`. The communication between the sub-domains is done using the shared memory. For example, in implementing a global action, instead of non-owners sending contributions to owners, like in the distributed memory machines from next subsection, the owners directly take and add the slave contributions. For dot products we use `#pragma omp critical` to add the sub-domains' dot products. The alternative, the OpenMP `reduction`, seems computationally not efficient.

### 5.3.3. Parallel FEM/FVM using MPI

The *Message-Passing Interface*, or MPI, is a complex system of 129 functions which allow distributed memory parallel programming model. In such model the computation comprises one or more processes that communicate by calling library routines to send and receive messages to other processes. Although there are many MPI functions usually 5 or 6 are enough. We will describe these main functions and show how they are used in our code. The examples will be how the data structure was used in a parallel implementation of CG using MPI (in C++). A simple introduction model may be demonstrated with a fragment of our main function.

```
#include <mpi.h>
. . .
int main(int argc, char *argv[]){
  int nprocessors, myrank;
  . . .
  read_input(argc, argv, ... );
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
  MPI_Comm_size(MPI_COMM_WORLD, &nprocessors);

  Mesh m(...);                    // Every processor creates a
                                  // copy of the original mesh
  int *tr = new int[m.GetNTR()];
  m.DomainSplit(nprocessors, tr);// Split the mesh into nprocessors

  Subdomain dd(myrank, &m, tr);  // Initialize sub-domain ‘‘myrank’’
  . . .                           // using the splitting in tr
  MPI_Finalize();
}
```

Here `MPI_Init` is used to initiate the MPI computation, `MPI_Comm_rank` to determine the process's identifier in `myrank`, `MPI_Comm_size` to get the number of processors running the program, and `MPI_Finalize` to terminate the computation. The next function that we use is `MPI_Allreduce`. One of its important uses is to compute dot-products. The only change in the consecutive dot product in this CG routine is that at the end we add the following :

```
  . . .                          // compute res = (x, y)
  #ifdef DOMAIN_DECOMPOSITION
    double total;
    MPI_Allreduce(&res,&total,1,MPI_DOUBLE,MPI_SUM,MPI_COMM_WORLD);
    return total;
  #else
    return res;
  #endif
```

The other vector operations, addition/subtraction and scaling, are as in the consecutive version. The other important MPI functions, sending and receiving, will be demonstrated in the implementation of a parallel global Matrix-vector product ($y = Ax$).

```
void Matrix::Action(double *x, double *y){
  #ifdef DOMAIN_DECOMPOSITION        // The owners update the values
  Subdomain->Update_Slave_Values( x);// of x in the slave nodes
  #endif
  . . .                              // standard  y = A x
  #ifdef DOMAIN_DECOMPOSITION
  Subdomain->Update( y);             // Slave nodes add their
  #endif                             // values to the owners
}
```

Functions `Update_Slave_Values` and `Update` have similar implementation and use the same MPI functions, so we will show only function `Update`. This function updates the values of its argument by adding the contributions from no owners to owners. It also makes the values at the no-owned nodes 0.

```
void Subdomain::Update(double *x){
  int i, j, k, PNum, n = 0, start;
  double Buf[MAX_PACKET], Bdr[6*MAX_PACKET];

  MPI_Request request;
  MPI_Status Status;

  for(i=0; i<NPackets; i++)
    if (Packets[i].Owner != SN){
      start = n;
      for(j=0; j<Packets[i].NPoints; j++){
        Bdr[n++] = x[Packets[i].Vertices[j]];
        x[Pa[i].Ve[j]] = 0.;
      }
      // Send to the owner Packets[i].Owner - the corresponding packet
      // number is Packets[i].Pack_Num_Owner (see the receiving part)
      MPI_Isend(&Bdr[start], Packets[i].NPoints, MPI_DOUBLE,
                Packets[i].Owner, Packets[i].Pack_Num_Owner,
                MPI_COMM_WORLD, &request);
    }
  // The owners receive packets from neighboring subdomains and
  // perform the necessary corrections.
  for(i=0; i<NPackets; i++)
    if (Packets[i].Owner == SN)              // Every owner reads
      for(j=0; j<Packets[i].NSubdom; j++){   // Pa[i].NSubdom packets
        MPI_Recv( Buf, MAX_PACKET, MPI_DOUBLE, MPI_ANY_SOURCE,
                  MPI_ANY_TAG, MPI_COMM_WORLD, &Status);
        PNum = Status.MPI_TAG;
        for(k=0; k< Packets[PNum].NPoints; k++)
          x[Packets[PNum].Vertices[k]] += Buf[k];
      }
}
```

We put in `Bdr` all the information that has to be send in order to use non-blocking send (function `MPI_Isend`). The alternative, using blocking send, is slower since the sending procedure `MPI_Send` does not complete until the whole message is delivered. Moreover, a deadlock is possible in the latter case.

More about the syntax of the used and other MPI functions can be found in [34].

## 5.4. Code organization and data structures

Although there are many useful software engineering hints and directions on how to organize and develop large software applications, the organization and the development remain art and therefore, depend on the developer's taste and experience. We have organized our code in a well accepted and standard for the development of finite element/volume packages way. This is, we separate in different libraries, the (1) linear algebra part, which in general contains different solvers, (2) the mesh generation routines and data structures and (3) the finite element/volume discretization routines.

Specific information about our code structure is given in Subsection 5.4.1. Also, the code design is connected to, and should be adjusted according to, the desired solvers for the resulting from the discretization algebraic system. The problems, that we are interested in, lead to the solution of extremely large scale sparce linear systems. Taking into account the big problem size, it is a necessity that iterative methods are used for their solutions. The storage to assemble the global matrix (in sparce format) is the same for both direct and iterative methods, but the direct methods may produce an arbitrary amount of fill-ins that may be prohibitively high. Also, we are interested in parallel computations and since matrix-vector operations can be efficiently parallelized, all iterative solvers can highly benefit the parallelism, while the direct methods need sophisticated methods to extract limited parallelism. One disadvantage of the iterative methods is that the rate of convergence is problem dependent and may be unacceptably slaw. Therefore, the software should be designed so that there are data structures suitable for the development of preconditioners that would accelerate the iterative methods. Our computational approach provides decomposition of the domain data structures, which are suitable for the development of domain decomposition type preconditioners (see Subsection 5.4.2). They are undoubtedly one of the best known and promising preconditioning methods that also take good advantage of the parallelism. The adaptive process produces multilevel data structures, suitable for the development of multigrid type preconditioners (see Subsection 5.4.3). Finally, in Subsection 5.4.4, we mention the internal and external solvers and preconditioners that we use.

### 5.4.1. The overall code structure

The code is written in C++ and has object oriented structure. The C++ classes that we developed are grouped into the 5 Libraries given on Figure 10.

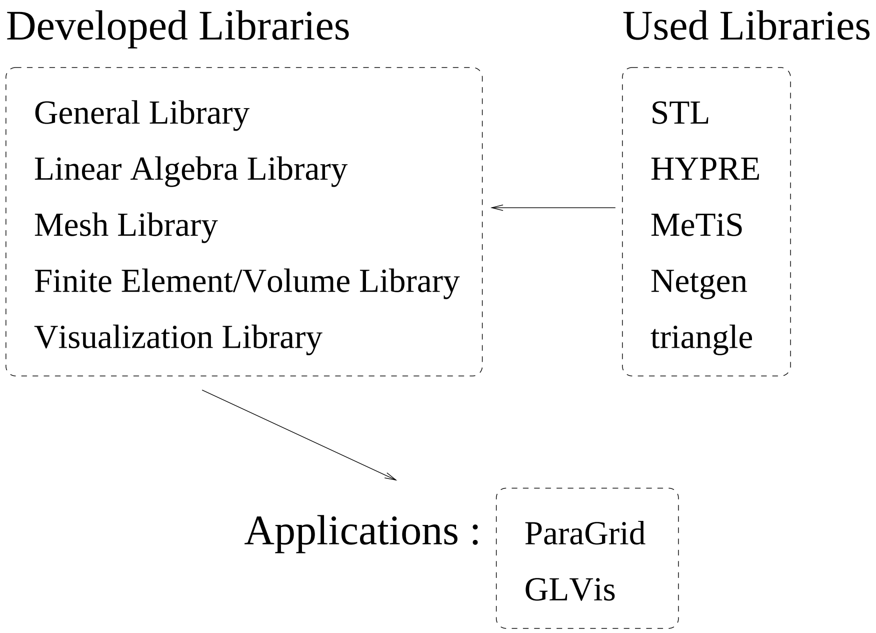

Developed Libraries
Used Libraries
General Library
Linear Algebra Library
Mesh Library
Finite Element/Volume Library
Visualization Library
STL
HYPRE
MeTiS
Netgen
triangle
Applications :
ParaGrid
GLVis

Fig. 10. Overall code structure : Used Libraries, Developed Libraries and Developed Applications.

We have already discussed the functionality of ParaGrid in Subsection 5.1.1. Sockets, which we use to send data for visualization from ParaGrid to GLVis, are defined in the `General Library`. The `General Library` also includes different data structures (classes) such as tables, trees, etc. We have used standard data structures from the `STLibrary` as well : vectors, sets, stacks, etc. The `Linear Algebra Library` contains the internal for ParaGrid iterative solvers. We can easily connect and provide data to external solvers, such as the developed in Lawrence Livermore National Laboratory *HYPRE* preconditioners and solvers Library. The `Mesh Library` has classes that are able to maintain adaptively, and in parallel, the mesh, as given by the mesh algorithms in the previous sections. The library is coupled with stand alone mesh generators, such as `NETGEN` and `triangle`, and the graph partitioner MeTiS.

The `Visualization Library` deals with the visualization (see Section 5.5).

### 5.4.2. Domain decomposition data structures

Our implementation provides data structures that are helpful in implementing non-overlapping domain decomposition type precondiitoners. First, in the `Mesh Library`, we have a class `Mesh` that provides the described in the previous sections mesh functionality on a single processor. `Mesh` is inherited by class `Subdomain`, which provides the parallel functionality. `Subdomain` has data fields that define the connections between the different subdomains. The connections are in terms of `Packets`, which are defined as follows. Every vertex is shared by a group of processors. Vertices, having the same group of processors that share them, are grouped into a packet. The idea of such grouping is to define the different types of communications between the subdomains. The packets have the following data fields :

- array of indices of the vertices belonging to the packet;
- array of the indices of the subdomains sharing the packet and, since every subdomain that share the packet keeps a local copy of it, the address of the local copy;
- index of the subdomain defined as "owner" of the packet.

When we keep the mesh conforming between the subdomains and the degrees of freedom are in the vertices, the packet vertices in the "owner" subdomain are taken as degrees of freedom and the others are considered as slave nodes.

The idea of grouping the vertices into packets is extended to grouping of common interfaces (element edges or faces) into packets. Such grouping extends the data structures to cases when the degrees of freedom are associated with the elements' edges or faces.

### 5.4.3. Multigrid data structures

Starting with the coarse mesh we keep the refinement/derefinement history in a tree data structure. The tree data structure can be used to define the multigrid spaces. For the steady state case the spaces are the ones from the different refinement levels. In that case, along the adaptive process, we compute and store the corresponding interpolation and stiffness matrices.

### 5.4.4. Internal and external solvers and preconditioners

We have implemented internal interactive solvers, such as CG, PCG and GMRES, and preconditioners, such as multigrid and hierarchical multigrid. Additionally, we can provide data and use external solvers and preconditioners. The code is connected to the Hypre preconditioners library.

## 5.5. Visualization

The Visualization Library is composed of the classes given on Figure 11. The figure also gives the class dependency tree.

The library and the application GLVis were developed in a group project at Texas A&M University. The scenes that we visualize are build of geometric primitives such as vertices, lines, and polygons. To render these primitives we have used OpenGL, and to open windows and detect interactive user input we have used "The OpenGL Programming Guide Auxiliary Library" (see [48]). With the auxiliary library the input is handled through callback functions for specific events, such as pressed buttons and mouse movements and clicks.

The first two classes in the hierarchy, `VisScene` and `VisSceneScalarData`, are abstract. The rest, `VisSceneSolution`, `VisSceneVector`, `VisSceneSolution3d`, and

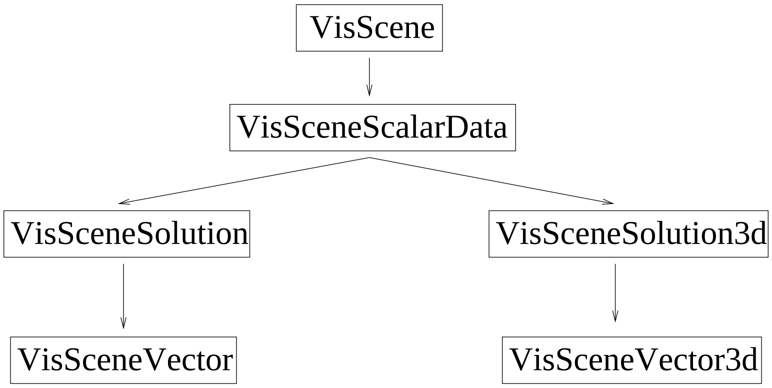


Fig. 11. Visualization classes and their dependency tree

`VisSceneVector3d`, can be used to define objects and visualize them. The objects may be for problems that yield 2-D scalar and vector solutions, and 3-D scalar and vector solutions. Each of the last 4 classes has implementation of a `Draw` function, which is used to render the represented by the class object. `Draw` is called every time the user input is intended to change the visualization scene. For example, the user initializes an event by pressing a button, clicking the mouse, etc. Then a callback function, corresponding to the event, is called. The callback function is programmed to make the necessary changes in the model and calls `Draw`, which visualizes the model in the new setting.

For all classes of problems the user can interactively do scene translations, different types of spinning and rotation, object scaling, change of the light position, etc. One can display/hide colorbar, axes, elements, and mesh. For both 2 and 3-D scalar data we have level curves. For the vector case we have vector field visualization, displacements visualization, etc. For 3-D problems we have volume visualization through moving cutting plane. There are also visualization effects like shading (smooth/flat), perspective (turned on/off), various palettes, different backgrounds, and material properties, etc. Concerning the input we have input from files or sockets. The last feature safes the trouble of going through files, allows the visualizer and the program computing the data to be compiled on different type of machines and still the user to have the feeling that they are connected. Also, one can use the last feature for real time visualization of time dependent problems. One scenario, in which we use the sockets, is the following. GLVis is installed and runs as a server on a desk/lap-top machine with fast visualization. The computations are done on a remote machine, in our case parallel, which sends through sockets data in parallel to the server to be visualized.

# CHAPTER VI

# NUMERICAL RESULTS

The numerical tests that we conducted confirm our theoretical results. Here we include several examples, numerically demonstrating the performance of the discussed error estimators. In Section 6.1 we consider problems with known solutions. The behavior of the different error estimators, that we have developed, is compared with the exact errors. The rest of the sections are for more realistic problems, where the exact solution is unknown. Namely, in Subsection 6.2, we consider various convection-diffusion-reaction equations with boundary layers and singular solutions due to corners of the boundary or discontinuity of the coefficients. We have three examples of singular problems. The first one has singularities due to corners. The second is for a problem with discontinuous right hand side $f$. The last example is for a problem with discontinuity of the matrix $A(x)$. In Subsection 6.3 we consider an example that is related to simulation of steady state fluid flow in porous media in 3-D.

## 6.1. Results with known exact error

**Example 1.** We consider the Dirichlet problem on an $L$-shaped domain. The boundary values are taken such that the exact solutions are the harmonic functions $r^{1/2} sin\frac{\theta}{2}$, $r^{2/3} sin\frac{2\theta}{3}$, and $r^{4/3} sin\frac{4\theta}{3}$. The theory gives that $\eta_E$ and $\eta_Z$ are equivalent to the error, which is also confirmed from the computations. The numerical results are summarized on Figures 3, 12, and 13.

On Figure 3 we plot the error in the energy norm $|| \cdot ||_a$ as a function of the degrees of freedom. We have taken both of the scales to be logarithmic. For every of the 3 problems there are two graphs showing the error reduction on uniform and locally refined mesh. Thus, we compare the efficiency of using local refinement for problems with different regularity. One can see that the more singular the solution is, the more beneficial the application of local refinement is.

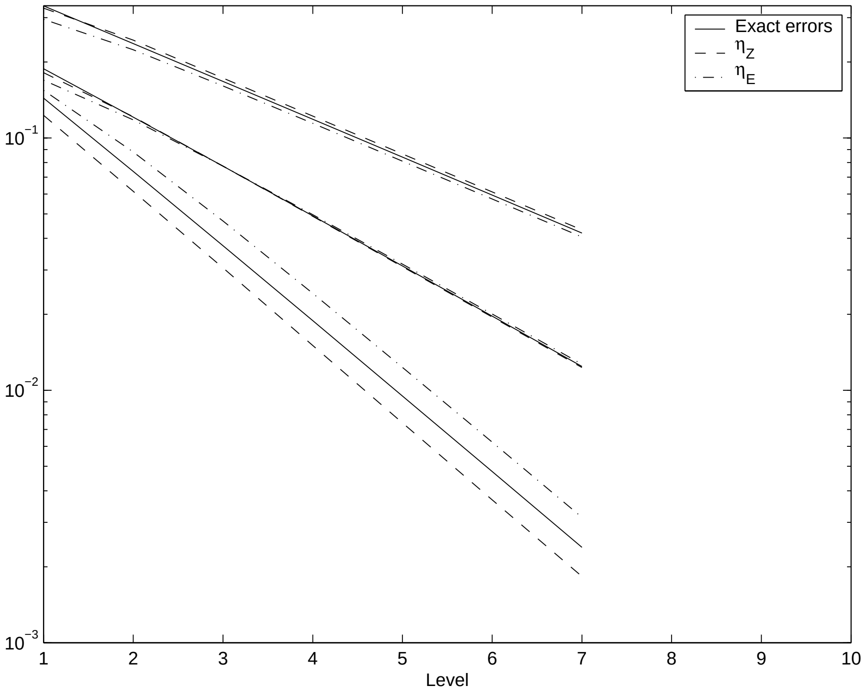


Fig. 12. Exact error, $\eta_Z$, and $\eta_E$ for uniformly refined meshes over different mesh levels for the three problems in Example 1.

Figure 12 gives the exact error (solid line), the a posteriori error estimator $\eta_Z$ (dashed line), and $\eta_E$ (dash-dotted line) for the three problems over the different levels of the mesh. The levels are obtained by uniform refinement (splitting every triangle into 4) and have correspondingly 65, 255, 833, 3201, 12545, 49665, and 197633 nodes. The errors are printed in logarithmic scale in order to demonstrate the linear over the levels error reduction. For exact solutions in $H^{1+1/2-\epsilon}$, $H^{1+2/3-\epsilon}$, and $H^{1+4/3-\epsilon}$ ($\epsilon > 0$), one can see the theoretically expected error reduction over the levels of correspondingly 1/2, 2/3, and 1. One can observe that both $\eta_Z$ and $\eta_E$ are equivalent

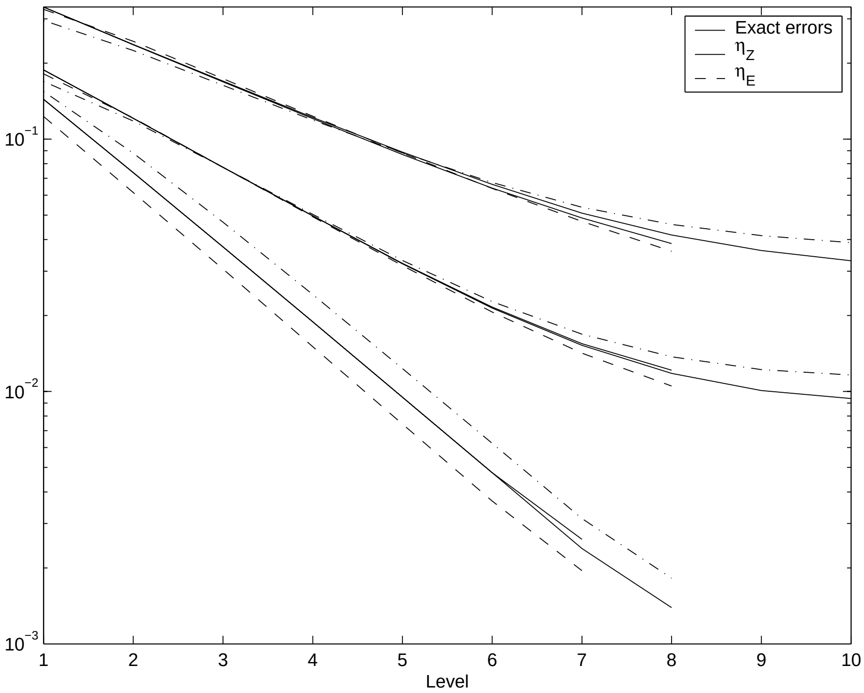


Fig. 13. Exact error, $\eta_Z$, and $\eta_E$ for locally refined meshes over different mesh levels for the three problems in Example 1.

to the exact error, as proved in the theoretical section. The same is true when local refinement is applied (see Figure 13). The error tolerances supplied to the refinement procedures for the three problems are the exact errors on level 7 of the uniformly refined mesh, i.e., the one with 197633 nodes. Refinement based on $\eta_Z$ leads to final meshes of 2508, 8365, and 169618 nodes, correspondingly for Problems 3, 2, and 1. Refinement based on $\eta_E$ leads to final meshes of 1986, 10097, and 581852 nodes correspondingly for Problems 3, 2, and 1 (see also Figure 3). Note the difference in the order of the mesh sizes for uniform refinement and local refinement for the first two problems. Note also that the order of the error reduction over the different levels is the same as in the uniform refinement case. For the third problem we have full elliptic regularity and $\eta_Z/\eta_E$ are supposed to lead to uniform refinement, which is confirmed by the numerical experiment.

**Example 2.** Our next example will demonstrate the local refinement strategies performance on the reaction dominated diffusion problem $A = \varepsilon I$, $\mathbf{b} \equiv 0$, $c = 2$, $\Omega$ is the unit square and $f$ is such that we have the following exact solution:

Table I. Numerical results for Example 2, $\varepsilon = 0.01$.

| level | # nodes | $\|\|e\|\|_{max}$ | $\|\|e\|\|_{L^2}$ | $\|\|e\|\|_{H^1}$ |
|---|---:|---:|---:|---:|
| 0 | 25 | 0.26563 | 0.16796 | 13.12243 |
| 1 | 81 | 0.16181 | 0.12131 | 9.41188 |
| 2 | 289 | 0.05351 | 0.07652 | 6.43875 |
| 3 | 479 | 0.01053 | 0.03558 | 4.16403 |
| 4 | 1010 | 0.00174 | 0.01190 | 2.42569 |
| 5 | 2729 | 0.00043 | 0.00328 | 1.28376 |
| 6 | 8506 | 0.00005 | 0.00088 | 0.65283 |
| *Globally uniform grid* | | | | |
| 5 | 16641 | 0.00041 | 0.00327 | 1.28750 |

$$u(x_1, x_2) = x_1 x_2 (1 - exp(\frac{x_1 - 1}{\varepsilon}))(1 - exp(\frac{x_2 - 1}{\varepsilon})).$$

For $\varepsilon$ small, the solution has a boundary layer near the boundary $x_1 = 1$ and $x_2 = 1$.

The local refinement strategies intuitively are supposed to produce much finer grids in the region where the boundary layer is located. For $\varepsilon = 0.01$ the results are summarized in Table I. Quadrature that preserves cubic polynomials was used in the computation of the discrete $L^2$ and $H^1$ norms and 13-point Gaussian quadrature (degree of precision 7) was used in the computation of the right hand side .

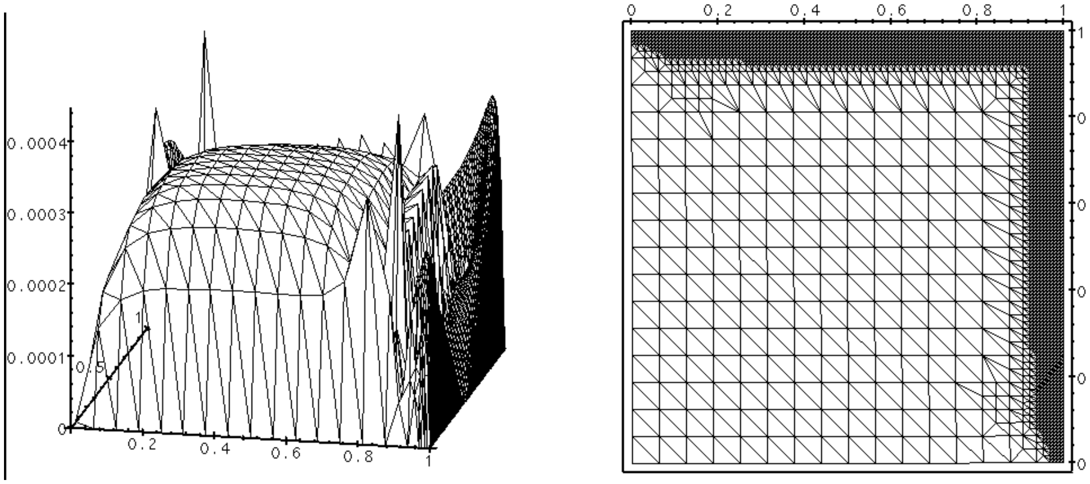


Fig. 14. Reaction-diffusion problem with $\varepsilon = 0.01$; the error (left) and the mesh (right) obtained for level 5 with 2729 grid points.

Figures 14 and 15 give the computational results for reaction-diffusion problem solved using the residual type error estimator with 5 levels of refinement. As expected, the grid is refined in the boundary layer. The error is equilibrated on level 5, as seen on Figure 14 (left). The relative error $||e||/||u||$ on level 6 in discrete maximum, $L^2$, and $H^1$ norms is correspondingly 0.0005, 0.0028, and 0.11. For comparison we have also computed an approximation by using a uniform grid. On level 5 this grid has 16641 nodes and the error in $L^2$-norm is 0.003276, while the error in $H^1$-norm is 1.28.

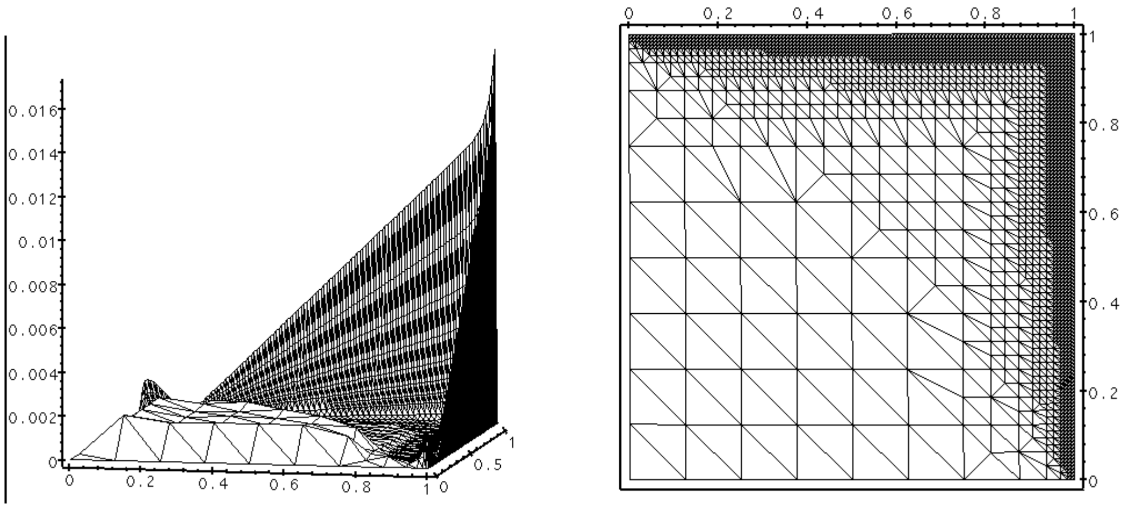


Fig. 15. Reaction-diffusion problem with $\varepsilon = 0.001$; the error (left) and the mesh (right) obtained for level 5 with 2385 grid points by the residual type error estimator (0.3 tolerance).

The accuracy is comparable to the accuracy of the locally refined grid on level 5 with 2729 nodes. Note, that the locally refined grid has an order of magnitude less grid points than the uniform grid.

The other error estimators give similar results. There are small differences in the obtained meshes. However, qualitatively the meshes are refined in the area where the solution has some type of singularity (or very steep gradient) and quantitatively they have almost the same number of nodes. However, the more singular the problem is the larger the error of the finite element approximation is. Comparing Figure 14 and Figure 15 we see that, while the solutions of reaction-diffusion problems with $\epsilon = 0.01$ on a grid with about 2800 nodes has maximal error of 0.05%, the solution for the same problem with $\epsilon = 0.001$ on a grid with about 2400 nodes has a maximal error of about 2%. Similar results were obtained by using other types of refinement. For example, the meshes on level 5, using a posteriori error estimates based on the solution of local Neumann problems and the Zienkiewicz-Zhu type error estimator, are given on Figure 16.

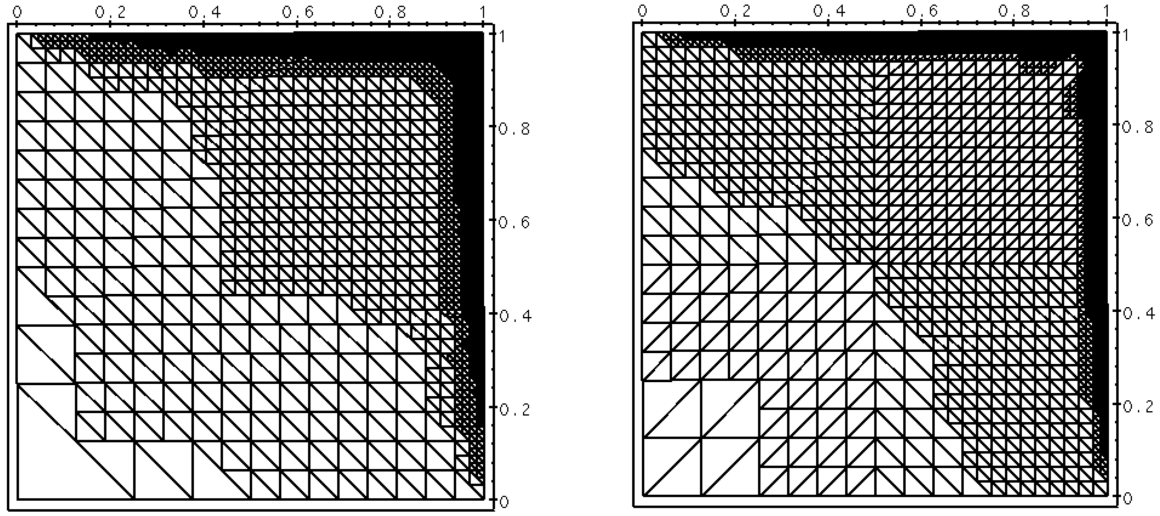


Fig. 16. Reaction-diffusion problem with $\varepsilon = 0.001$; meshes obtained by the error estimator based on local Dirichlet problems (left) with 2013 grid points and by the Zienkiewicz-Zhu type error estimator (right) with 2031 grid points at level 5.

## 6.2. Problems with singular solutions

**Example 3.** Figures 17 and 18 show the computational results obtained for problems with large convection. Namely, we solve the problem (2.4) with Dirichlet data on the whole boundary $\partial\Omega$, $A = \varepsilon I$, $\mathbf{b} \equiv (2, 1)$, $c = 0$, $f = 0$ in a trapezoid. On the left and on the upper edges of $\Omega$ we have zero boundary conditions, while on the rest of the boundary the solution is equal to 1.

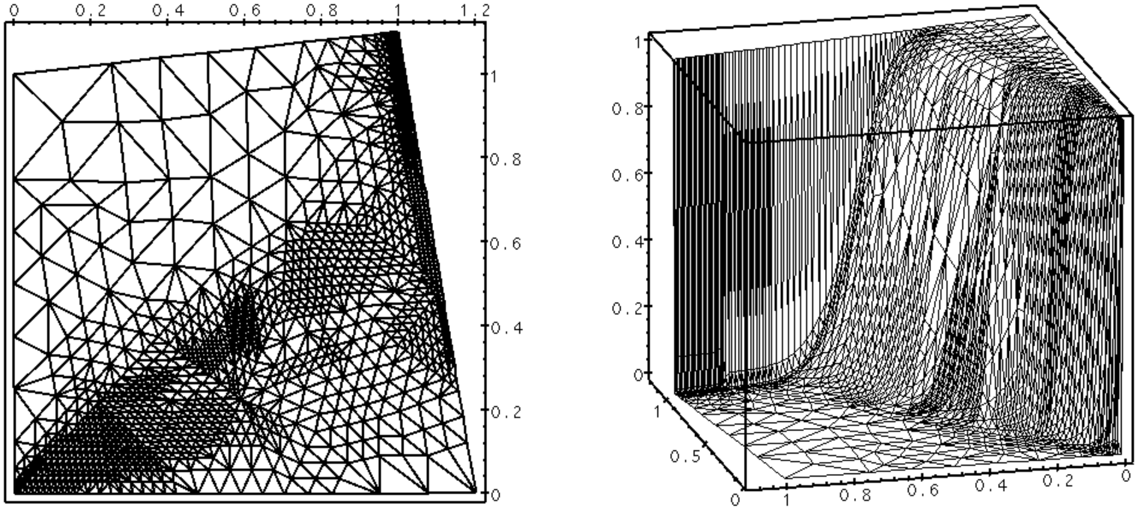


Fig. 17. Convection-diffusion problem from Example 3 with $\varepsilon = 10^{-3}$; the mesh (left) with 1209 grid points and solution (right) obtained at level 4.

The solution develops a boundary layer at the upper half of the right edge of $\Omega$ and an interior layer along the line $x_1 = 2x_2$. Also, there are two corner singularities at the origin and at the upper right corner, due to the discontinuity of the Dirichlet data. Figures 17 and 18 represent the computational meshes and the solutions for $\varepsilon = 10^{-3}$ and $\varepsilon = 10^{-5}$, correspondingly. The computations are obtained using *RB*-

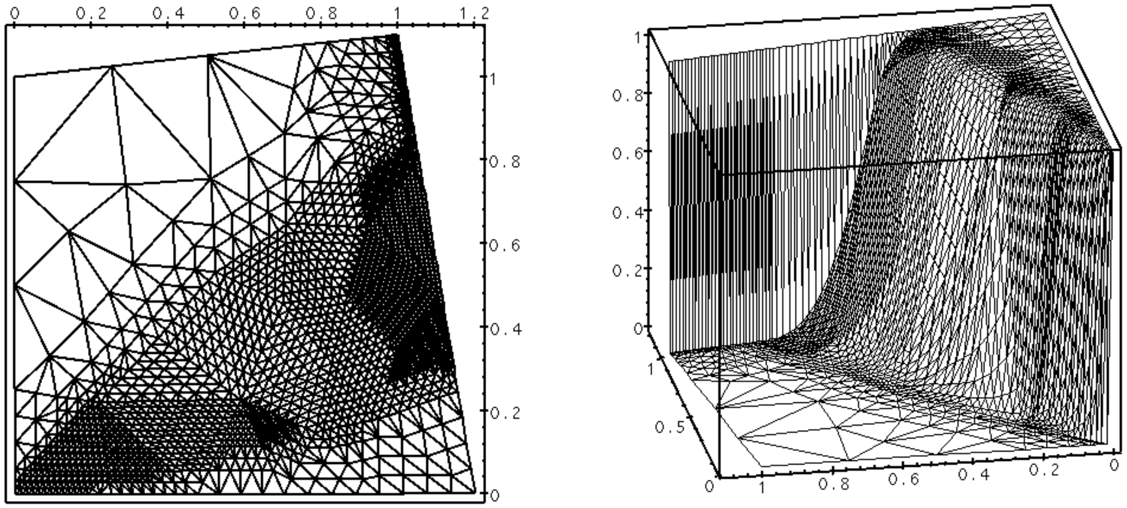


Fig. 18. Convection-diffusion problem from Example 3 with $\varepsilon = 10^{-5}$; the mesh (left) with 1761 grid points and the solution (right) obtained at level 4.

refinement for the case $\varepsilon = 10^{-3}$ and $ZZ$ refinement for $\varepsilon = 10^{-5}$. As seen from the figures, the error estimator produces finer grids in the regions where the solution has a boundary layer or singularity. The computed solutions are monotone, so no oscillations due to the numerical approximation are produced.

**Example 4.** We consider elliptic problems with corner singularities. Our first problem is the model problem with $A = I$, $\mathbf{b} \equiv 0$, $c = 0$, $f = 1$ in a "rose"-shaped domain with homogeneous Dirichlet boundary conditions. Due to the symmetry we can consider only a quarter of the domain (in fact we could have considered only 1/12 of the domain). The isolines of the computed unknown solution and the corre-

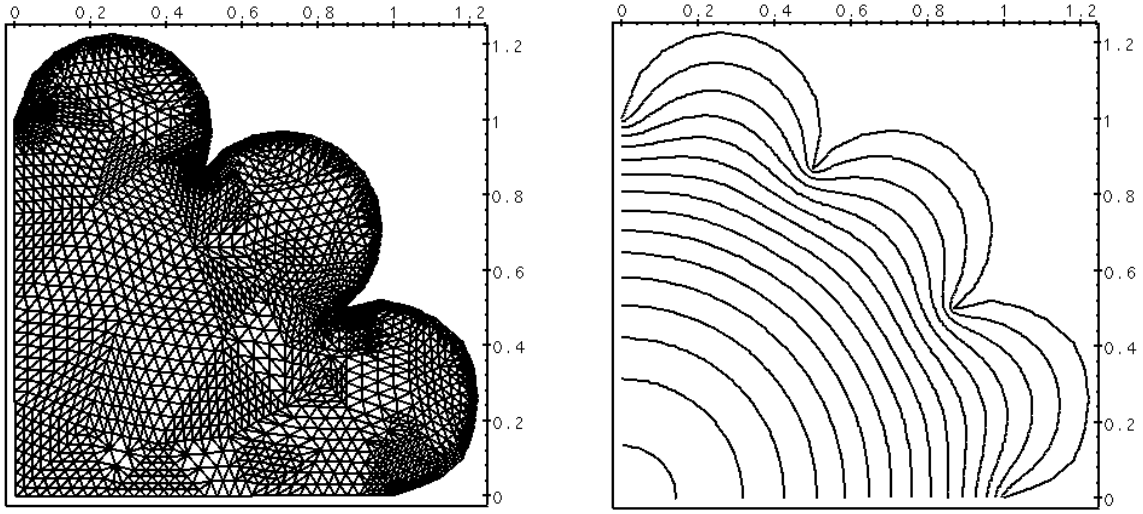


Fig. 19. Example 4 : the mesh (left) with 2562 grid points and the level curves (right).

sponding grid are given on Figure 19. Here we have used residual based refinement. As expected, the grid is refined around the corners since the solution has corner singularities.

**Example 5.** This is another example of corner singularity. We consider two 3-D problems and show on Figure 20 the obtained computational meshes after few levels of refinement. The first test is again for the homogeneous Poisson equation with $f = 1$. We consider an $L$-shaped domain $\Omega$ shown on Figure 20 (left). Here the singularity is due to the non-convex corner at the origin. The second test is for the Poisson equation in the unit cube shown on Figure 20 (right) with Dirichlet boundary condition on $x = 0$, $y = 0$, $z = 0$ and Neumann boundary conditions on the rest of the boundary. The singularity is due to the source term, which is taken to be $\delta$-function concentrated at the corner $(1,1,1)$. In both cases, as expected, the error control strategy yields mesh refinement around the corners, where the singularities are located.

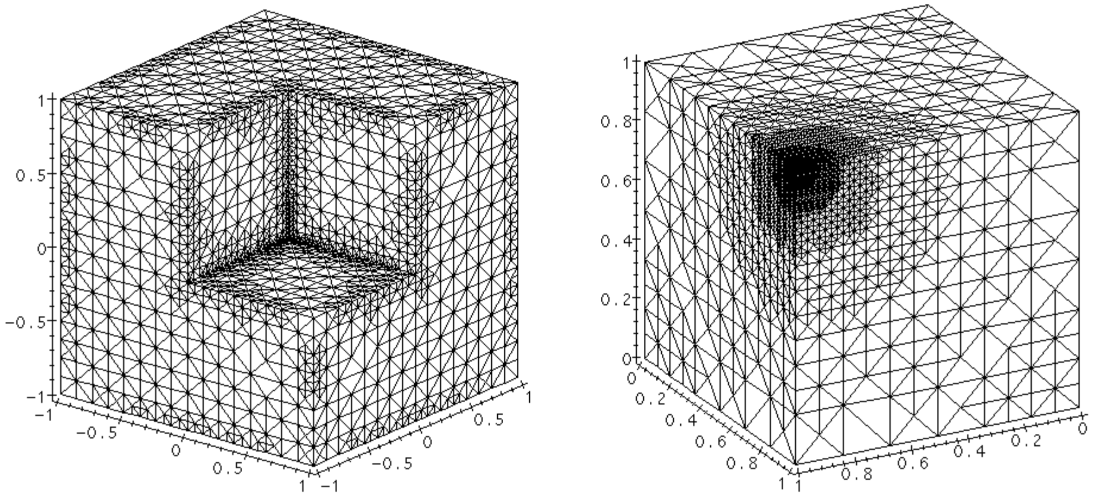


Fig. 20. Computational meshes obtained by local refinement due to $L$-shaped corner singularity (left) and delta function source term (right). Both meshes are for level six with correspondingly 12350 and 11770 nodes.

**Example 6.** Here we consider the Laplace equation in a slit domain. Namely, in the square $\Omega = \{(-1,1) \times (-1,1)\}$, we make a cut along the positive $x_1$-axis. Thus, the boundary now consists of all four sides of $\Omega$ plus the points in the interval $\{(x_1, 0) : 0 < x_1 < 1\}$. Dirichlet boundary conditions are prescribed on the whole boundary of the slit domain so that the exact solution is $u(x_1, x_2) = r^{1/2}\sin(\frac{\theta}{2})$. The singularity at the origin is $O(r^{1/2})$. Figure 21 gives the computational mesh on level 9 obtained using the residual based refinement and the corresponding error. For this level the relative errors in maximum, $L^2$-, and $H^1$-norms are 0.59%, 0.12%, and 1.07%, correspondingly.

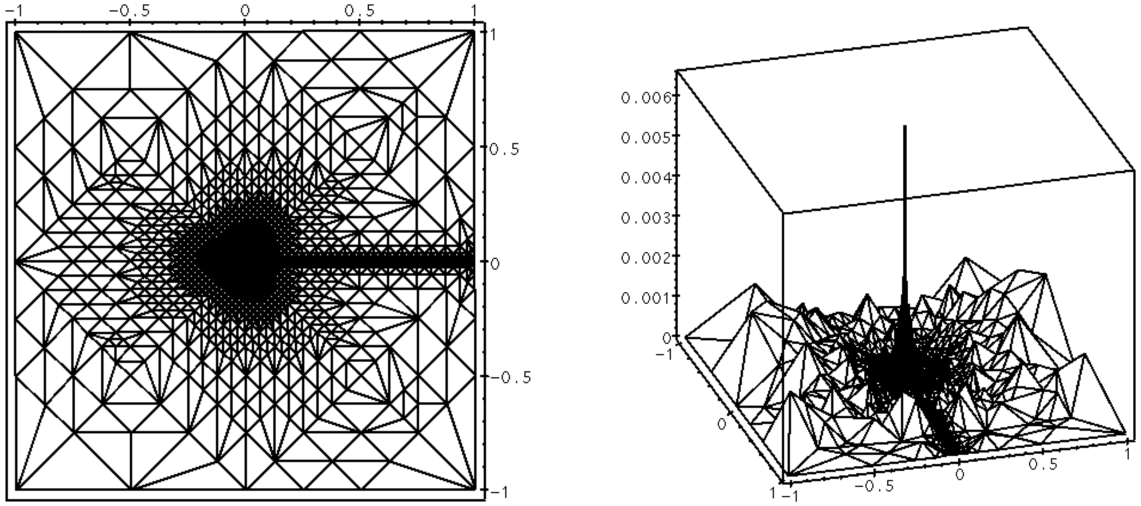


Fig. 21. Example 6 : Poisson equation in a slit domain with solution $u = r^{1/2}\sin(\frac{\theta}{2})$; the mesh (left) with 2064 grid points and the error (right) for level 9.

Both problems have symmetry, which allows to use only half of the domain. In fact, we have intentionally used the whole domain in order to see how the refinement procedure works. We see that the obtained meshes are very symmetric.

**Example 7.** We consider problem (2.4), where $\Omega$ is an $L$-shaped domain given on Figure 22, $\Gamma_D \equiv \Gamma$, $\underline{b} = (1.5, 1)$, $f = 0$, and $A(x) = 0.01\ I$, where $I$ is the identity matrix. The Dirichlet boundary values are 0 on $x = -1$ and $y = 1$, linearly increases from 0 to 1 on $(-1, -1)..(-1, -0.75)$ and 1 on the rest of the boundary. The up-wind scheme gives solution without oscillations. The local error estimators lead to local refinement around the expected boundary and internal layers. On Figure 22 we give the mesh on level 5 (left), which has 5232 nodes and 10247 triangles. On the right are the level curves of the solution computed on the same level.

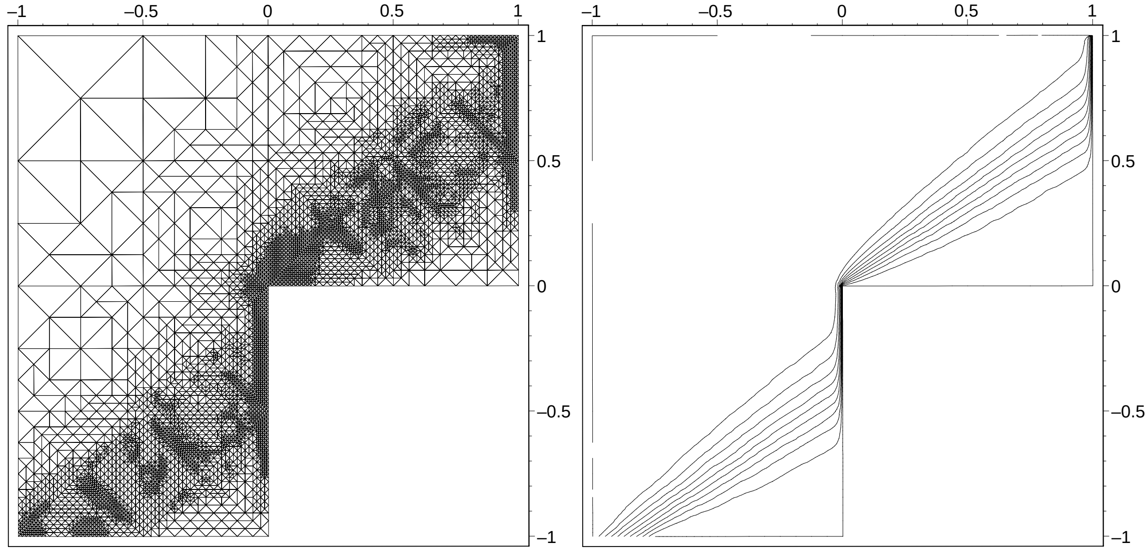


Fig. 22. Convection-diffusion problem in $L$-shaped domain; the mesh on level 5 (left) with 5232 nodes and 10247 triangles; the level curves (right) on the same level.

**Example 8.** Again $\Omega$ is an $L$-shaped domain, $\Gamma_D \equiv \Gamma$, $\underline{b} = (1.5, 1)$, and $A(x) = 0.01\ I$. The Dirichlet boundary value is 0 everywhere. $f$ is 0 everywhere except in the square with lower left and upper right corners $(-0.55, -0.55)$ and $(-0.45, -0.45)$, correspondingly. Figure 23 gives the mesh obtained after 4 levels of refinement. It has 3450 nodes and 6798 triangles. On the right are the level curves of the solution.

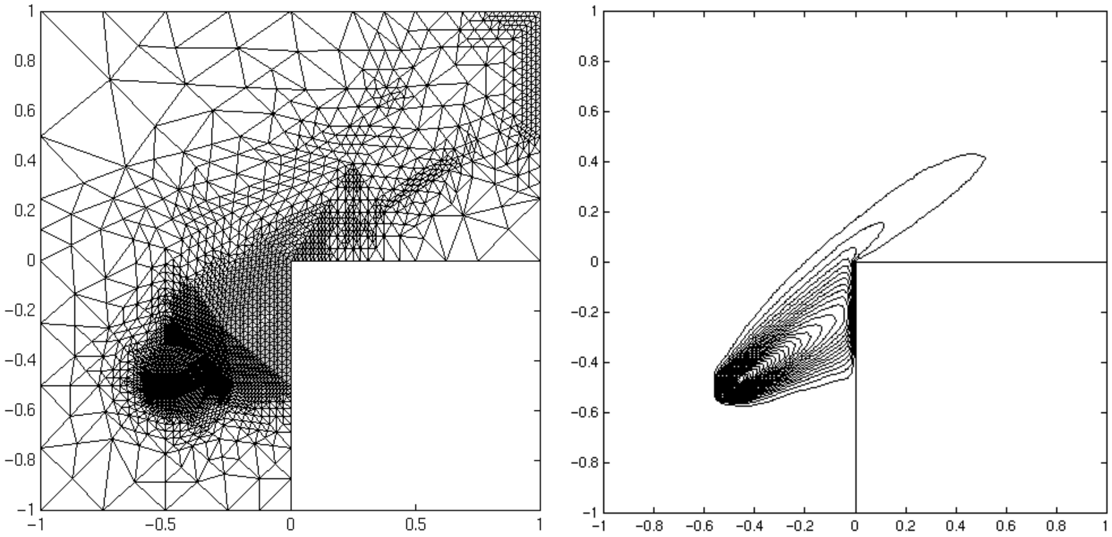


Fig. 23. Convection-diffusion problem in $L$-shaped domain; the mesh on level 4 (left) with 3450 nodes and 6798 triangles; the level curves (right) on the same level.

**Example 9.** We take $\Omega$ to be the domain shown on Figure 24 with one internal layer. In this problem, $\Gamma_D$ is the upper boundary, $\underline{b} = (1, -0.5)$, $f = 0$, and $A(x) = 0.05\ I$ in the layer and $A(x) = 0.01\ I$ in the rest of the domain. The Dirichlet boundary value is 1 for $x < 0.2$ and 0 otherwise. On the Neumann boundary, we take $g_N = 0$. Figure 24 shows the mesh on level 5 (left) with 6035 nodes and 11892 triangles. On the right are the solution level curves.

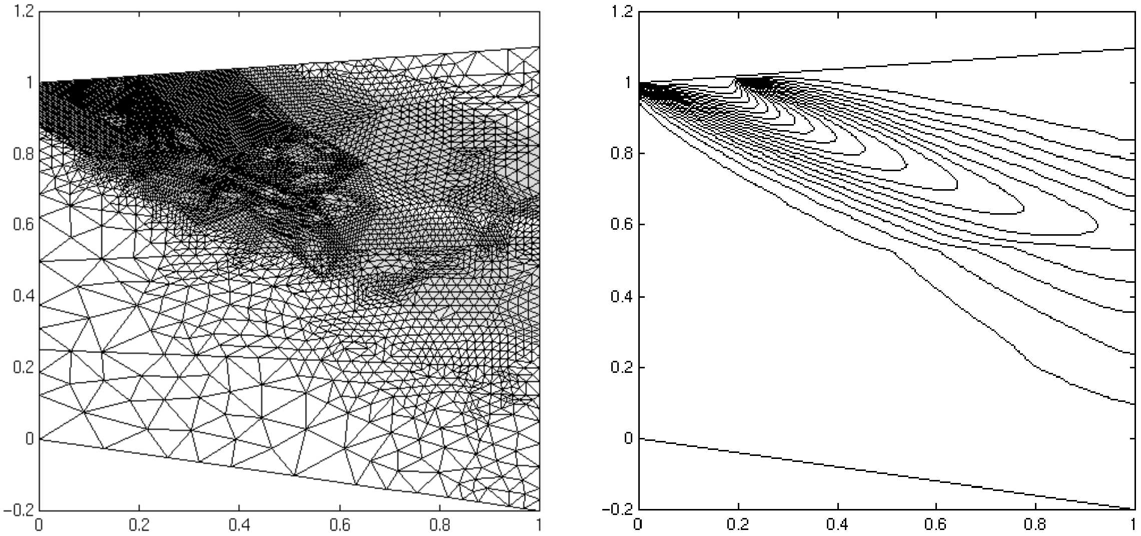


Fig. 24. Convection-diffusion problem in domain with layers; the obtained locally refined mesh after 5 levels of refinement (left) with 6035 nodes and 11892 triangles; the level curves (right) on the same level.

## 6.3. Application to steady state fluid flow in porous media in 3-D

In this section we show the results from applying the developed computational technology to steady-state problems of flow in porous media. The problems that we consider are based on realistic data and the setting is described in below. The mathematical model is based on problems (2.4) and includes wells modeled as line delta functions (see [43]). The singularities, due to jumps in the input data, the wells, domain non-homogeneity and so on, make the application of local grid refinement essential.

A steady state flow, with Darcy velocity $\underline{v}$, measured in *ft/yr*, has been established in a parallelepiped shaped reservoir of size $1000 \times 1000 \times 500$. The problem setting (see below) gives symmetry with respect to the plane $x_2 = 0$, so the equations are solved only in half of the domain, i.e., the parallelepiped $(0, 1000) \times (0, 500) \times (0, 500)$. The transport of a contaminant, in our case benzene dissolved in the water, is described by the convection-diffusion-reaction equation (2.4), where $u$ is the benzene concentration, $\underline{b}$ is the Darcy velocity $\underline{v}$, $A$ is the dispersion-diffusion tensor, and $c$ is the biodegradation rate. We assume that the Darcy velocity $\underline{v}$ is obtained by solving the pressure equation (Laplace equation, subject to various boundary conditions [43]) for fully saturated porous media under appropriate boundary conditions. We consider two cases:

(1) homogeneous reservoir; and

(2) non-homogeneous reservoir.

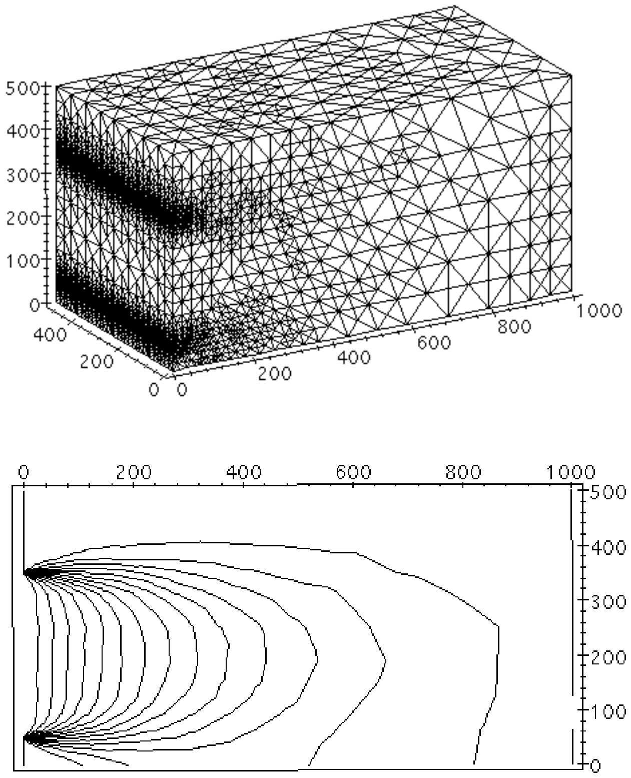


Fig. 25. Homogeneous reservoir; the 3-D mesh on refinement level 6 (top) with 54129 nodes; the concentration level curves (bottom) in the plane $x_2 = 0$ for the case of low biodegradation rate.

**Homogeneous reservoir.** The pressure at the faces $x_1 = 0$ and $x_1 = 1000$ is constant, correspondingly, 1000 and 0 and the permeability tensor is $D = 32I$, $I$ is the identity matrix. On the rest of the boundary a homogeneous Neumann condition is specified. This setting creates a uniform Darcy velocity $\underline{v} = (32, 0, 0)$ *ft/yr*. A steady state leakage on boundary strip $x_1 = 0,\ x_3 = 50..350$ of 30 *mg/l* has been established. The dispersion/convection process causes the dissolved benzene to

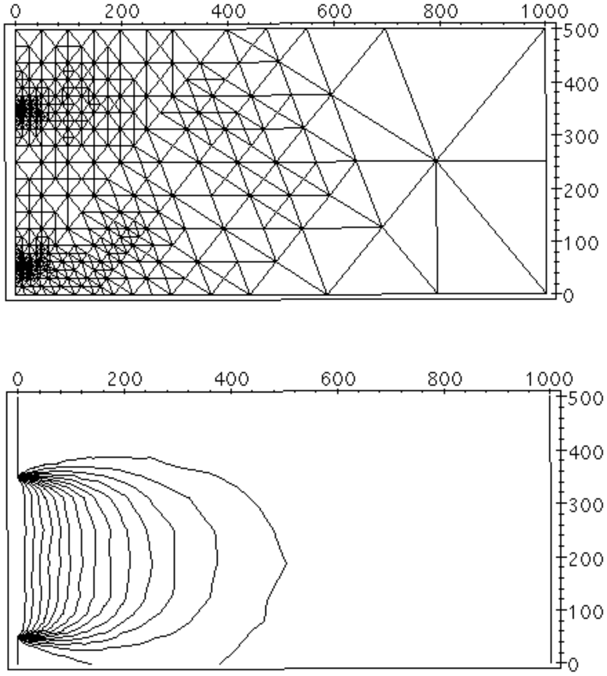


Fig. 26. Homogeneous reservoir; the mesh (top) in plane $x_2 = 0$ on refinement level 5 (globally with 36427 nodes); the concentration level curves (bottom) in plane $x_2 = 0$ for the case of medium biodegradation rate.

disperse in the reservoir. The dispersion tensor has the form

$$K = k_{diff} I + k_t \underline{v}^T \underline{v} / |\underline{v}| + k_l (|\underline{v}|^2 I - \underline{v}^T \underline{v}) / |\underline{v}|,$$

where $k_{diff} = 0.0001$, $k_t = 21$ and $k_l = 2.1$. The biodegradation is transforming the pollutant into a solid substance which is absorbed by the soil. This leads to a decrease in the benzene. Its concentration level curves are shown on Figure 25 for the case of low biodegradation rate $a = 0.1$ *mg/yr* and on Figure 26 for medium biodegradation rate $a = 0.2$ *mg/yr*. We have started with an initial coarse mesh with 52 nodes.

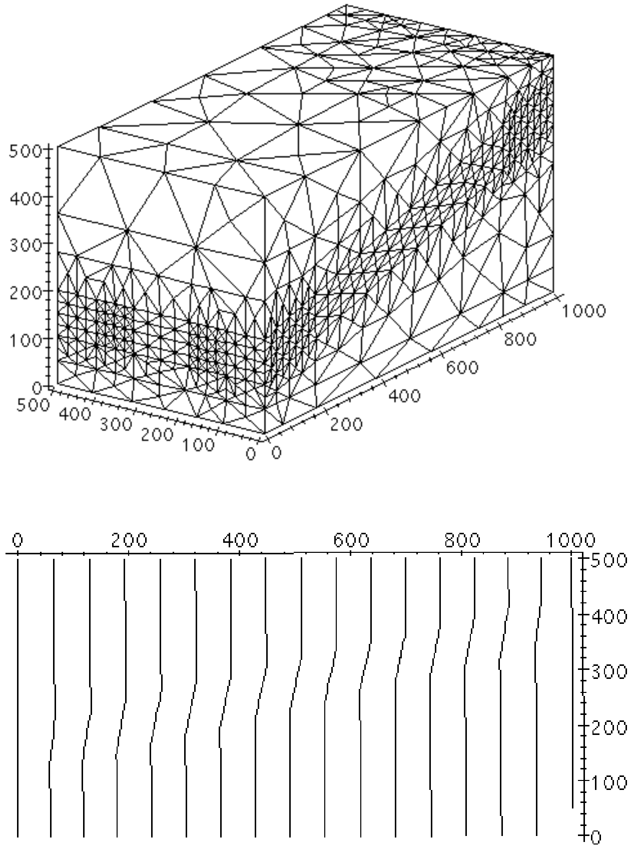

500
400
300
200
100
0
500
400
300
200
100
0
0
200
400
600
800
1000
0
200
400
600
800
1000
500
400
300
200
100
0

Fig. 27. Pressure computations in a non-homogeneous reservoir; (top) the locally refined 3-D Mesh on level 2 with 4053 nodes; (bottom) Contour curves of the pressure for level 2.

**Non-homogeneous reservoir.** Here the problem setting is as above, but a layer is added (see Figures 27, 28, and 29). In the layer strip we take the permeability $D_{layer}$ to be twice smaller than in the rest of the domain, i.e., $D_{layer} = 16I$. In this case, the Darcy velocity is not constant and the error estimators force the grid to be refined around the layer. The obtained grid is shown on Figure 27.

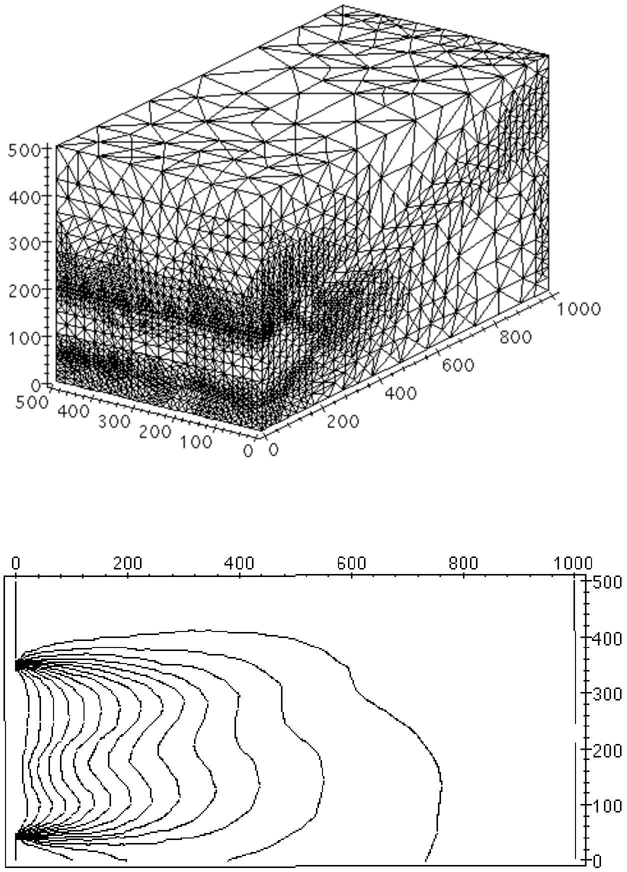


Fig. 28. Concentration distribution in a non-homogeneous reservoir; (top) the 3-D mesh on refinement level 4 with 39445 nodes; (bottom) contour curves of the concentration for the cross-section $x_2 = 0$; the permeability in the layer is two times smaller than in the rest of the domain.

After the pressure is found with prescribed accuracy, we solve the corresponding problem for the concentration. Figure 28 shows the obtained mesh and the isolines for the concentration in the reservoir cross-section $x_2 = 0$ on grid refinement level 4.

Two more experiments varying the permeability tensor are shown on Figure 29. The top one shows the concentration isoline in the reservoir cross-section $x_2 = 0$ when the permeability in the layer is 5-times smaller than the permeability in the rest of the reservoir. The result on the bottom is for 10-times smaller permeability. The initial coarse mesh in both cases has 235 nodes.

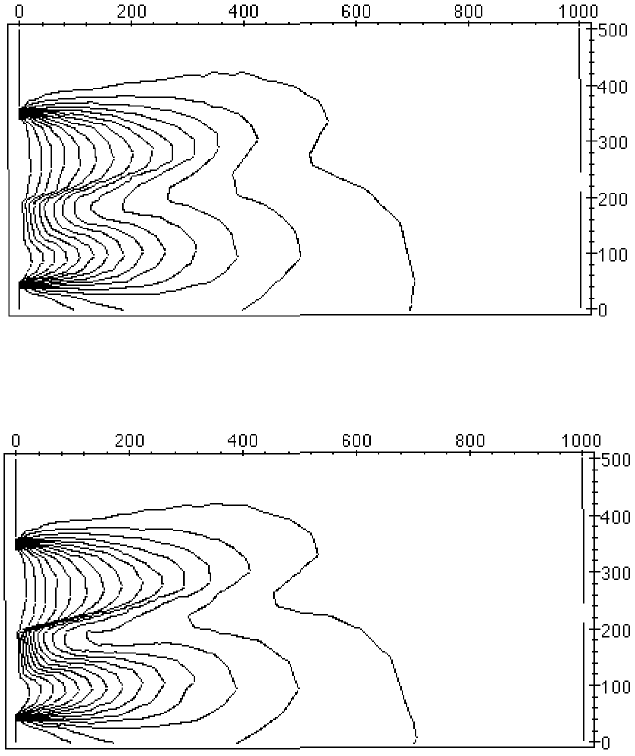


Fig. 29. Concentration distribution in a non-homogeneous reservoir; contour curves of the concentration for permeabilities in the layer 5 times (top) and 10 times (bottom) smaller than the permeability in the rest of the domain.

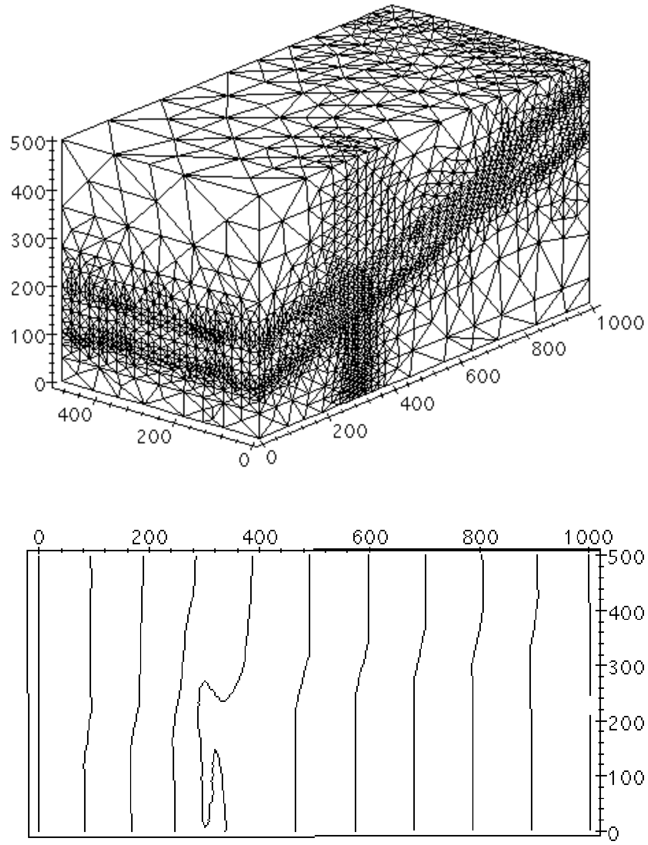


Fig. 30. Pressure computations in a non-homogeneous reservoir with a well; (top) the 3-D Mesh on level 3 with 34236 nodes; (bottom) contour curves of the pressure on level 3.

**Non-homogeneous reservoir with a well.** Finally, we consider a problem with one well using the line delta function well model that is described in [43]. The well has an axis along the segment $x_1 = 250$, $x_2 = 0$, $x_3 = 0..400$ and its production rate is $Q = 200000 \ l/yr$. On Figure 30 we show the adapted mesh and the level curves for the pressure in the reservoir cross-section $x_2 = 0$. On Figure 31 we show the obtained computational mesh and the level curves for the concentration in the reservoir cross-section $x_2 = 0$ on grid refinement level 5.

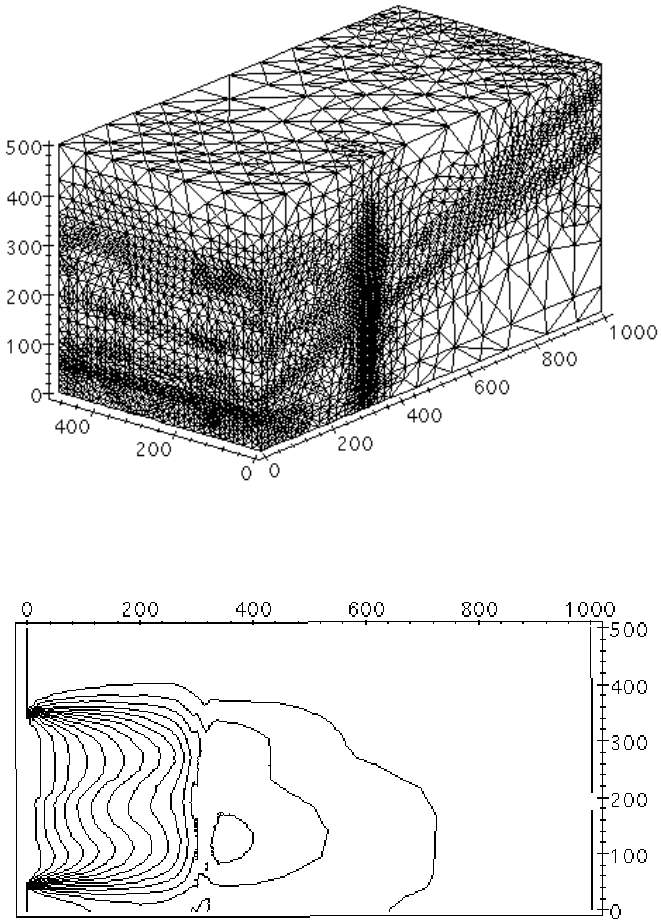


Fig. 31. Concentration distribution in a non-homogeneous reservoir with a well; (top) the 3-D mesh with 67509 nodes in half of the domain obtained after 5 levels of refinement; (bottom) contour curves of the concentration in the plane $x_2 = 0$.

# CHAPTER VII

# CONCLUSIONS

This dissertation represents a research in construction, theoretical study, practical justification, and testing of parallel adaptive finite volume methods for convection-diffusion-reaction equations in 2-D and 3-D on unstructured grids.

Our goal was to develop and implement a posteriori error estimators that guide the adaptive grid refinement process. Furthermore, we wanted the computations to be in parallel.

The idea that we pursued for the development of a posteriori error estimators was to apply the existing ideas in the finite element method to the finite volume approximations. To do so, we first extensively studied, both theoretically and practically, the a posteriori error estimates for the finite element method. A part of this direction of study was represented in the dissertation by a review of the most commonly used finite element a posteriori error analysis techniques, and a general adaptive methods framework that the residual based error estimators follow. Our analysis showed that the approach of comparing and exploiting the finite element ideas was feasible and led to useful constructions. Namely, we developed and theoretically justified a posteriori error estimators based on local residuals, local Dirichlet or Neumann problems, and Zienkiewicz-Zhu averaging/projection techniques. The efficiency and the reliability of the error estimators were supported by the numerical experiments for various model problems. The computations showed the equivalence of the estimated error to the actual error for various model problems. They also showed the capability of the adaptive refinement algorithm to detect the regions of singular behavior of the solution and to resolve these singularities with required accuracy.

We also developed algorithms and strategies for the parallel implementation of adaptive methods for partial differential equations. The algorithms were implemented and used to develop parallel mesh generation. The implementation issues were an important part of the dissertation. We discussed various techniques and issues regarding: (1) adaptive mesh generation; (2) mesh partitioning and load balancing; (3) parallel computations; and (4) visualization. Concerning the parallel computations we focused our research on efficient parallelization techniques of sequential finite element and finite volume methods. Our parallelization approach was based on domain decomposition data distribution.

This dissertation addressed a wide range of issues concerning the adaptive methods and therefore left many open problems and study directions. Some of the results have to be extended, others further studied. For example, our further research plans are to extend the results obtained and their application area to problems of elasticity and electro-magnetics. We have to further study in detail some time-dependent and non-linear problems. Also, a more rigorous analysis of the constants appearing in the reliability and efficiency estimates is needed. Concerning the computer implementation, although we created several general purpose libraries, there are many possibilities for extensions and improvements, mostly related to various applications, "tuning" the code to particular computer architectures, interfaces to external solvers and preconditioners, user friendly interfaces, etc. Moreover, we have to use the code developed in order to further study and develop internal solvers and preconditioners based on domain decomposition.

# REFERENCES

[1] M. Ainsworth and J.T. Oden, *A unified approach to a posteriori error estimators based on element residual methods*, Numer. Math., 65 (1993), pp. 23–50.

[2] L. Angermann, *Balanced a posteriori error estimates for finite-volume type discretizations of convection-dominated elliptic problems*, Computing, 55 (1995), pp. 305–324.

[3] L. Angermann, *An a-posteriori estimation for the solution of an elliptic singularly perturbed problem*, IMA J. Numer. Anal., 12 (1992), pp. 201–215.

[4] D. Arnold, A. Mukherjee, and L. Pouly, *Locally adapted tetrahedral meshes using bisection*, SIAM J. Sci. Comput., 22 (2000), pp. 431–448.

[5] I. Babuska and W. Rheinboldt, *Error estimates for adaptive finite element computations*, SIAM J. Numer. Anal., 15 (1978), pp. 736–754.

[6] I. Babuska and W. Rheinboldt, *A-posteriori error estimates for the finite element method*, Internat. J. Numer. Methods Engrg., 12 (1978), pp. 1597–1615.

[7] I. Babuska and I. Miller, *A feedback finite element method with posteriori error estimation: Part I. The finite element method and some basic properties of the a-posteriori estimators*, Comput. Methods Appl. Mech. Engrg., 61 (1987), pp. 1–40.

[8] R. Bank, *A simple analysis of some a posteriori error estimates*, Appl. Numer. Math., 26 (1998), pp. 153–164.

[9] R. Bank and R. Smith, *A posteriori error estimates based on hierarchical bases*, SIAM J. Numer. Anal., 30 (1993), pp. 921–932.

[10] R. Bank and A. Weiser, *Some a posteriori error estimators for elliptic partial differential equations*, Math. Comp., 44 (1985), pp. 283–301.

[11] 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., 4 (1996), pp. 237–264.

[12] R. Becker and R. Rannacher, *An optimal control approach to a posteriori error estimation in finite element methods*, Acta Numerica, 10 (2001), pp. 1–103.

[13] R. Becker, C. Johnson, and R. Rannacher, *Adaptive error control for multigrid finite element methods*, Computing, 55 (1995), pp. 271–288.

[14] J. Bramble, J. Pasciak, and O. Steinbach, *On the stability of the* $L^2$*-projection on* $H^1(\Omega)$, Math. Comp., 71 (2002), pp. 147–156.

[15] S. Brenner and L. Scott, *The Mathematical Theory of Finite Element Methods*, Springer-Verlag, New York, 1994.

[16] Z. Cai, *On the finite volume element method*, Numer. Math., 58 (1991), pp. 713–735.

[17] Z. Cai, J. Mandel, and S. McCormick, *The finite volume element method for diffusion equations on general triangulations*, SIAM J. Numer. Anal., 38 (1991), pp. 392–402.

[18] C. Carstensen, *Quasi-interpolation and a posteriori error analysis in finite element methods*, Math. Model. Numer. Anal., 33 (1999) pp. 1187–1202.

[19] C. Carstensen and S. Bartels, *Each Averaging Technique Yields Reliable A Posteriori Error Control in FEM on Unstructured Grids Part I: Low Order Con-*

*forming, Nonconforming, and Mixed FEM*, Math. Comp., posted on February 4, 2002, PII S 0025-5718(02)01402-3 (to appear in print).

[20] C. Carstensen and S. Funken, *Constants in Clément-interpolation error and residual based a posteriori estimates in finite element methods*, East-West J. of Numer. Anal., 8 (2000), pp. 153–175.

[21] C. Carstensen and S. Funken, *Fully reliable localized error control in the FEM*, SIAM J. Sci. Comput., 21 (2000), pp. 1465–1484.

[22] C. Carstensen and S. Funken, *A posteriori error control in low-order finite element discretizations of incompressible stationary flow problems*, Math. Comp., 70 (2001), pp. 1353–1381.

[23] A. Chorin and J. Marsden, *A Mathematical Introduction to Fluid Mechanics*, Texts in Appl. Math., 4, Springer-Verlag, New York, 1993.

[24] P. Clément, *Approximation by finite element functions using local regularization*, RAIRO Anal. Numer., 9 (1975), pp. 77–85.

[25] U. Elsner, *Graph Partitioning: A Survey*, Technical Report SFB393/97-27, TU Chemnitz, 1997.

[26] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson, *Computational Differential Equations*, Cambridge University Press, Lund, Sweden, 1996.

[27] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson, *Adaptive Finite Elements*, Springer, Berlin, 1996.

[28] K. Eriksson and C. Johnson, *An adaptive finite element method for linear elliptic problems*, Math. Comp., 50 (1988), pp. 361–382.

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

[30] R. Ewing, R. Lazarov, and Y. Lin, *Finite volume element approximations for nonlocal reactive flows in porous media*, Numer. Meth. PDE's, 16 (2000), pp. 285–311.

[31] R. Eymard, T. Gallouët, and R. Herbin, *Finite Volume Methods*, in Handbook of Numerical Analysis, VII, pp. 713–1020, North-Holland, Amsterdam, 2000.

[32] J. Flaherty, P. Paslow, M. Shephard, and J. Vasilakis (Editors), *Adaptive methods for partial differential equations*, SIAM Proceedings Series, Philadelphia, 1989, pp. 1–265.

[33] L. Franca, S. Frey, and T. Hughes, *Stabilized finite element methods. I: Application to advective-diffusive problems*, Comput. Meth. Appl. Mech. Engrg., 95 (1992), pp. 253–271.

[34] W. Gropp, *Tutorial on MPI: The Message-Passing Interface*, Mathematics and Computer Sci. Division, Argonne National Laboratory, Internet address (accessed on 12/2001): http://www-unix.mcs.anl.gov/mpi/tutorial/index.html.

[35] W. Hackbusch and J. Wappler, *Remarks on a posteriori error estimation for inaccurate finite element solutions*, Computing, 60 (1998), pp. 175–191.

[36] T. Ikeda, *Maximum Principle in Finite Element Models for Convection-Diffusion Phenomena*, Lecture Notes in Numer. Appl. Anal., 4, North-Holland Mathematics Studies, 76, Kinokuniya Book Store Co., Ltd., Tokyo, 1983.

[37] V. John, *A posteriori* $L^2$*-error estimates for the nonconforming* $P_1/P_0$*-finite element discretization of the Stokes equations*, J. Comput. Appl. Math., 96 (1998),

pp. 99–116.

[38] C. Johnson, *Numerical Solution of Partial Differential Equations by the Finite Element Method*, Cambridge University Press, Cambridge, 1987.

[39] G. Karypis and V. Kumar, *Multilevel k-way partitioning scheme for irregular graphs*, Journal of Parallel and Distributed Computing, 48 (1998), pp. 96–129.

[40] G. Karypis and V. Kumar, *A fast and highly quality multilevel scheme for partitioning graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392.

[41] R. Lazarov, I. Mishev, and P. Vassilevski, *Finite volume methods for convection-diffusion problems*, SIAM J. Numer. Anal., 33 (1996), pp. 31–55.

[42] R. Lazarov, J. Pasciak, and S. Tomov, *Error control, local grid refinement and efficient solution algorithms for singularly perturbed problems*, in Analytical and Numerical Methods for Convection-Dominated and Singularly Perturbed Problems (L. Vulkov, J. Miller, and G. Shishkin, Editors), NOVA Science Publ., Inc., New York, 2000, pp. 71–82.

[43] R. Lazarov and S. Tomov, *Adaptive finite volume element method for convection-diffusion-reaction problems in 3-D*, in Scientific Computing and Application (P. Minev, Y. Wong, and Y. Lin, Editors), Advances in Computation: Theory and Practice, NOVA Science Publ., Inc., New York, 7 (2001), pp. 91–106.

[44] R. Li, Z. Chen, and W. Wu, *Generalized Difference Methods for Differential Equations – Numerical Analysis of Finite Volume Methods*, Monogr. and Textb. in Pure and Appl. Math., 226, Marcel Dekker, Inc., New York, 2000.

[45] J. Miller, E. O'Riordan, and G. Shishkin, *Fitted Numerical Methods for Singular Perturbation Problem*, World Scientific, Singapore, 1996.

[46] I. Mishev, *Finite Volume and Finite Element Methods for Nonsymmetric Problems*, Ph.D. Dissertation, Texas A&M University, 1996.

[47] I. Mishev, *Finite volume methods on Voronoi meshes*, Numerical Methods for PDEs, 14 (1998), pp. 193–212.

[48] J. Neider, T. Davis, and M. Woo (OpenGL Architecture Review Board), *OpenGL Programming Guide : The Official Guide to Learning OpenGL, Release* 1, Addison-Wesley Publishing Company, Boston, May 1996.

[49] S. Owen, *A Survey of Unstructured Mesh Generation Technology*, Internet address (accessed on 12/2001):
http://www.andrew.cmu.edu/user/sowen/softsurv.html.

[50] D. Padua and M. Wolfe, *Advanced compiler optimizations for supercomputers*, Communications of ACM (special issue), 29 (1986), pp. 1184–1201.

[51] G. Pinder and W. Gray, *Finite Element Simulation in Surface and Subsurface Hydrology*, Academic Press, Inc., New York, 1977.

[52] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Clarendon Press, Oxford, 1999.

[53] R. Rannacher and R. Scott, *Some optimal error estimates for piecewise linear element approximations*, Math. Comp., 38 (1982), pp. 437–445.

[54] R. Rodriguez, *Some remarks on Zienkiewicz-Zhu Estimator*, Numerical Methods for PDEs, 10 (1994), pp. 625–635.

[55] H. Roos, *Layer-adapted grids for singular perturbation problems*, Z. Angew. Math. Mech. (ZAMM), 78 (1998), pp. 291–309.

[56] H. Roos, M. Stynes, and L. Tobiska, *Numerical Methods for Singularly Perturbed Differential Equations*, Springer-Verlag, Berlin, 1996.

[57] H. Royden, *Real Analysis*, Macmillan Publishing Company, New York, 3 edition, 1988.

[58] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Internet address (accessed on 12/2001):
http://www-users.cs.umn.edu/ saad/books.html.

[59] A. Samarskii, *The Theory of Difference Schemes*, Marcel Dekker, New York, 2001.

[60] J. Schöberl, *NETGEN - An advancing front 2D/3D-mesh generator based on abstract rules*, Comput. Visual. Sci, 1 (1997), pp. 41–52.

[61] J. Shewchuk, *Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator*, First Workshop on Applied Computational Geometry (Philadelphia, Pennsylvania), ACM, May 1996, pp. 124–133.

[62] Silicon Graphics, Inc., *MIPSpro C and C++ Pragmas*, Document Number 007-0701-130, 1999, Internet address (accessed on 12/2001):
http://autarch.loni.ucla.edu/ebt-bin/nph-dweb/dynaweb/SGI_Developer.

[63] M. Slodicka and R. Van Keer, *A nonlinear elliptic equation with non-local boundary condition solved by linearization*, Preprint 1, Department of Mathematics, University of Gent, Belgium, 2000.

[64] M. Tabata, *A finite element approximation corresponding to the upwind finite differencing*, Mem. Numer. Math., 4 (1977), pp. 47–63.

[65] V. Thomee, *Galerkin Finite Element Methods for Parabolic Problems*, Springer, Berlin-Heidelberg, 1997.

[66] S. Tomov, *Tool-box for large scale parallel computations of* 3*-D fluid flow problems using domain decomposition*, Texas A&M University, ISC Technical Report ISC-00-10-MATH, 2000.

[67] R. Verfürth, *A Review of a Posteriori Error Estimators and Adaptive Mesh Refinement Techniques*, Teubner-Wiley, Stuttgart, 1996.

[68] R. Verfürth, *A posteriori error estimators for convection-diffusion equations*, Numer. Math., 80 (1998), pp. 641–663.

[69] R. Verfürth, *A posteriori error estimation and adaptive mesh-refinement techniques*, J. Comp. Appl. Math., 50 (1994), pp. 67–83.

[70] J. Xu and L. Zikatanov, *A monotone finite element scheme for convection-diffusion equations*, Math. Comp., 68 (1999), pp. 1429–1447.

[71] Q. Zhu and Q. Lin, *Superconvergence Theory for Finite Element Methods*, Hunan Scientific Press, Changsha, 1989.

[72] O. Zienkiewicz and J. Zhu, *A simple error estimator and adaptive procedure for practical engineering analysis*, Internat. J. Numer. Methods Engrg., 24 (1987), pp. 337–357.

[73] O. Zienkiewicz and J. Zhu, *Adaptivity and mesh generation*, Internat. J. Numer. Methods Engrg., 32 (1991), pp. 783–810.

[74] O. Zienkiewicz and J. Zhu, *The superconvergence patch recovery and a posteriori error estimates, Part 2: Error estimates and adaptivity*, Internat. J. Numer. Methods Engrg., 33 (1992), pp. 1365–1382.

[75] O. Zienkiewicz, J. Zhu, A. Craig, and M. Ainsworth, *Simple and practical error estimation and adaptivity: h and h-p version procedures*, Adaptive Meth. PDEs, pp. 100–114, SIAM, Philadelphia, 1989.

# VITA

Stanimire Zdravkov Tomov was born in Vidin, Bulgaria on April 18, 1971 to Yanka and Zdravko Damyanov. He studied at Sofia University Saint Kliment Ohridski from September 1989 and graduated with a Bachelor of Science in Computer Science in September 1994. After serving his country in the army from March 1995 to March 1996, he began his graduate studies in numerical analysis and scientific computation at Texas A&M University in the fall of 1996. He spent the summers of 1999, 2000, and 2001 as a summer student at the Center for Applied Scientific Computing at the Lawrence Livermore National Laboratory. The current dissertation on the adaptive methods for finite volume approximations was defended in February 2002.

Until May 2003, Stanimire Tomov will be a Research Associate at the Information Technology Division at Brookhaven National Laboratory. He can be contacted at:

Department of Mathematics
Texas A&M University
College Station, TX 77843-3368
U. S. A.
E-mail address: tomov@math.tamu.edu