The Pennsylvania State University

The Graduate School

Department of Mathematics

AN ADAPTIVE FINITE ELEMENT CODE FOR ELLIPTIC BOUNDARY VALUE PROBLEMS IN THREE DIMENSIONS WITH APPLICATIONS IN NUMERICAL RELATIVITY

A Thesis in

Mathematics

by

Arup Mukherjee

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 1996

We approve the thesis of Arup Mukherjee.

Date of Signature

Douglas N. Arnold
Distinguished Professor of Mathematics
Thesis Advisor
Chair of Committee

Pablo Laguna
Assistant Professor of Astronomy and Astrophysics

Jie Shen
Assistant Professor of Mathematics

Simon Tavener
Associate Professor of Mathematics

George E. Andrews
Evan Pugh Professor of Mathematics
Chair, Department of Mathematics

Abstract

In this thesis, we develop an algorithm for the efficient solution of elliptic boundary value problems in $\mathbb{R}^3$. In particular, we use linear finite elements on tetrahedral meshes to discretize the problem and a multilevel V-cycle solver to solve the resulting linear system. The nested sequence of meshes used by the multilevel solver are constructed adaptively using an a posteriori error estimator. The values of the error estimator are used in conjunction with a locally adaptive tetrahedral mesh refinement procedure based on bisection of tetrahedra to obtain the sequence of nested meshes. We prove that the repeated bisection of tetrahedra yield only a finite number of similarity classes of tetrahedra, and hence degenerate tetrahedra are not produced while the nested meshes are created. The locally adaptive mesh refinement algorithm is recursive, and we prove that it terminates in a finite number of steps.

As an application, we consider initial data problems in numerical relativity, and show that the adaptive multilevel algorithm presented here may be used for the effective and efficient solution of these.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I wish to thank my advisor Professor Douglas N. Arnold. I could never have successfully completed this work without his guidance. He took the pains to teach me everything I needed for my research.

I also wish to thank the members of my thesis committee, who were always available for help and guidance whenever I needed it. In particular, I wish to thank Professor Pablo Laguna, who introduced me to numerical relativity and was patient with me as I was learning the subject.

Luc Pouly, who visited Penn State as a post doctoral fellow during 1994-95, was closely associated with my research and was of immense help during a crucial period when most of the work in this thesis was accomplished. I wish to thank him for his friendship and support. I also wish to acknowledge the help I received from Professor Eberhard Bänsch, who send a subroutine for implementing the mesh refinement algorithm. I needed and used the accrued knowledge and lessons learnt over the years from all the people who were my teachers in finishing this work. I wish to thank all of them.

Jim Humphreys, who wrote the style file that was used to typeset this document simplified my job significantly. I wish to thank him for his efforts.

I also wish to thank all members of my extended family who always supported me in my work. I am grateful to all my friends who made my stay at Penn State enjoyable. Finally, I wish to recognize the love and support of my wife, Bagisha, whose faith in my abilities was unwavering through many difficult times. I could not have completed this work without her support.

Chapter 1

Introduction

In this thesis, we present a code (AMG3DP1) for the numerical solution of linear elliptic boundary value problems in $\mathbb{R}^3$. AMG3DP1 is an acronym for "Adaptive Multi-Grid in 3-Dimensions using P1-finite elements". The algorithm uses piecewise linear finite elements on tetrahedral meshes to discretize the problem, adaptive mesh refinement to create a sequence of meshes capturing the essential nature of the solution, and a multigrid solver to solve the resulting system of linear equations.

1.1 Motivation

A major motivation for the development of the code comes from numerical relativity. A collaborative effort is under way to construct an observatory capable of detecting gravity waves. The Laser Interferometric Gravity-wave Observatory (LIGO) should be capable of detecting gravity waves emanating from violent cosmological events. In particular, the coalescence of two concentrated spiraling bodies (black holes) is expected to be observable by LIGO. In view of this, there is some urgency to be able to numerically simulate this event. An effective numerical simulation of this event can be used to create a library with which observations from LIGO can be compared. The nature of the gravitational waves emanating from the coalescence of two black holes (binary black hole collisions) are determined by the Einstein equations. These are a system of ten nonlinear coupled second order partial differential equations. The unknowns in the Einstein equations are the ten independent components of the symmetric metric tensor that determines spacetime. In numerical relativity, it is common to view spacetime as a foliation of three dimensional space like hypersurfaces. This decomposition allows the Einstein equations to be decoupled into

a system of four equations that need to be satisfied on each spacelike hypersurface and six other equations that are used in evolution. The problem of determining a symmetric metric satisfying the four equations on an initial space like hypersurface is the initial data problem. Under further simplifying assumptions, the initial data problem reduces to the solution of a semilinear elliptic boundary value problem on a complex geometry in $\mathbb{R}^3$. An efficient and accurate solution of this problem plays an important role in the determination of the nature of gravitational waves emanating from the coalescence of two black holes. In this work, we use Newton's method to reduce the initial data problem to a sequence of linear problems, and employ AMG3DP1 to solve the resulting linear elliptic boundary value problems. A description of the origin of the initial data problem for black hole collisions is provided in Chapter 6.

Under certain symmetry assumptions, the initial data problem for colliding black holes reduces to an elliptic boundary value problem in $\mathbb{R}^2$. In this case, existing codes like PLTMG [5] solve the problem effectively. Unfortunately, we were unable to find a code with similar features that would solve the general initial data problem for colliding black holes and other interesting astrophysical situations. This led to the development of AMG3DP1. The next section describes the main features of AMG3DP1.

1.2 Outline

The algorithm used in AMG3DP1 has the basic structure of a full multigrid and is similar to that used in MGGHAT [29], which solves a linear elliptic boundary value problem in $\mathbb{R}^2$ using finite elements, adaptive mesh refinement and multigrid solvers. We now describe the algorithm for computing the finite element solution to a linear problem using AMG3DP1.

Start with a coarse conforming mesh $\mathcal{T}_0$. Generate meshes $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_M$, each refining the previous, with the number of vertices in $\mathcal{T}_j$ being approximately double

that of $\mathcal{T}_{j-1}$. The approximate solution will be computed in the finite element space V_M associated with the finest mesh $\mathcal{T}_M$ using the nested sequence of finite element spaces in a multigrid V-cycle iteration. The requirement that the number of vertices in the mesh approximately double is made in order to obtain an efficient multigrid solver. In fact, it can be shown under suitable hypotheses that the number of operations needed to solve the linear system is proportional to the dimension of V_M. The multigrid iteration will be discussed in Chapter 4.

In order to obtain an accurate discrete solution with as few unknowns as necessary, the meshes $\mathcal{T}_j$ will be constructed adaptively. Specifically, to obtain $\mathcal{T}_j$ from $\mathcal{T}_{j-1}$, we compute the discrete solution in V_{j-1} and assign an error indicator to each tetrahedron in $\mathcal{T}_{j-1}$ based on the residual of the discrete solution. From these error indicators we determine how much each element of $\mathcal{T}_{j-1}$ needs to be refined. The error indicator and some basic facts about the finite element method will be discussed in Chapter 2.

Once we have computed both the discrete solution in V_{j-1} and an indication of the desired level of refinement for each tetrahedron in $\mathcal{T}_{j-1}$, we need to apply an algorithm to construct a refined mesh $\mathcal{T}_j$. Besides achieving the desired degree of local refinement as closely as possible, our algorithm must ensure that the meshes produced are conforming and that the tetrahedral shapes do not degenerate. The mesh refinement procedure will be discussed in Chapter 3.

Based on the error estimation, mesh refinement, and multigrid algorithms in Chapters 2, 3, and 4, we will give a full algorithmic description of AMG3DP1 in Chapter 5. In this chapter, we will also address the issue of solving a semilinear elliptic boundary value problem using this algorithm, and present some numerical tests to show that AMG3DP1 is an efficient and reliable code.

As we have mentioned before, Chapter 6 deals with the origins and formulation of the initial data problem for numerical relativity using the conformal imaging ap-

proach of York [36]. In this chapter, we also present numerical tests demonstrating the ability of AMG3DP1 to tackle such problems effectively by considering model problems for which the exact solutions are known.

Chapter 7 gives a summary of the work.

Chapter 2

The finite element method and an error estimator

In this chapter, we recall some well known results about the finite element method as applied to a model second order boundary value problem on an open, bounded domain in $\mathbb{R}^3$. The discussion is restricted to piecewise linear finite elements on tetrahedral meshes, but the results generalize to finite elements of higher order and to non-tetrahedral meshes (see for example, Section 2.2 in Ciarlet [15]). The residual error estimator used in adaptively refining the mesh is also described in detail. This was first developed and analyzed for a problem in one dimension by Babuška and Rheinboldt [3, 4]. The derivation given here follows that of Verfürth [33, 34]. The remainder of this chapter is organized as follows. In the first section we describe the general linear boundary value problem that can be solved by AMG3DP1 and give its variational formulation. Section 2 briefly discusses some ideas related to the finite element method. This leads to the formulation of a discrete problem associated with the variational problem in Section 1. Some definitions concerning tetrahedral meshes are also given in this section. These are needed in formulating the discrete problem and in the derivation of the error estimator. Section 3 deals with the derivation of the error estimator. We also show that the error estimator provides an upper bound for the actual error in an appropriate norm.

2.1 The linear boundary value problem

We consider boundary value problems of the following form:

$$-\operatorname{div}\left[\underset{\approx}{\mathcal{A}}(\underset{\sim}{x})\operatorname{grad}\underset{\sim}{} u\right] + b(\underset{\sim}{x})u = f(\underset{\sim}{x}) \text{ in } \Omega, \tag{1}$$

$$u = g_D(\underset{\sim}{x}) \text{ on } \Gamma_D, \tag{2}$$

$$\underset{\approx}{\mathcal{A}}(\underset{\sim}{x})\operatorname{grad}\underset{\sim}{} u \cdot \underset{\sim}{n} + c(\underset{\sim}{x})u = g_N(\underset{\sim}{x}) \text{ on } \Gamma_N. \tag{3}$$

Here $\Omega \subseteq \mathbb{R}^3$ is an open, bounded, polyhedral domain with boundary Γ and outward unit normal $\underset{\sim}{n}$, and $\underset{\sim}{x} \in \Omega$ is an arbitrary point. The boundary of Ω is the union of the closure of two disjoint open sets Γ_D and Γ_N, $\Gamma = \overline{\Gamma}_D \cup \overline{\Gamma}_N$, where the closure is with respect to the boundary. We shall assume that the coefficients a_{ij} of the matrix $\underset{\approx}{\mathcal{A}}$ are piecewise continuously differentiable and the coefficients b and c are piecewise continuous with respect to the finite element meshes introduced below. Although the boundary value problem (1)–(3) allows for more general coefficients, these smoothness assumptions are required for our numerical algorithms. Similarly, we shall assume that f and g_N are piecewise continuous, and g_D is the restriction to Γ_D of a continuous function u_D with square integrable derivatives. It is also assumed that g_D is piecewise linear with respect to the meshes considered. Further, assume that the matrix $\underset{\approx}{\mathcal{A}}$ is symmetric, and that the differential operator in (1) is elliptic; i.e. there exists a constant $\beta > 0$ such that

$$\underset{\sim}{\xi}^t \underset{\approx}{\mathcal{A}} \underset{\sim}{\xi} \geq \beta \underset{\sim}{\xi}^t \underset{\sim}{\xi} \text{ for all } \underset{\sim}{\xi} \in \mathbb{R}^3,\ \underset{\sim}{x} \in \Omega. \tag{4}$$

Let

$$V_0 = \left\{ v \in H^1(\Omega) \mid v = 0 \text{ on } \Gamma_D \right\},$$

and

$$V_D = \left\{ u_D + v \mid v \in V_0 \right\} = \left\{ u \in H^1(\Omega) \mid u = g_D \text{ on } \Gamma_D \right\}.$$

The *weak* formulation of the boundary value problem (1)–(3) is:

Find $\hat{u} \in V_D$ such that

$$a(\hat{u}, v) = l(v) \text{ for all } v \in V_0, \tag{5}$$

where $a(\,\cdot\,,\,\cdot\,) : H^1(\Omega) \times H^1(\Omega) \to \mathbb{R}$, and $l(\cdot) : H^1(\Omega) \to \mathbb{R}$ are defined by

$$a(u,v) = \int_\Omega \left[\left(\underset{\approx}{\mathcal{A}} \underset{\sim}{\operatorname{grad}}\, u \right) \cdot \underset{\sim}{\operatorname{grad}}\, v + buv \right] dx + \int_{\Gamma_N} cuv\, ds, \tag{6}$$

$$l(v) = \int_\Omega fv\, dx + \int_{\Gamma_N} g_N v\, ds. \tag{7}$$

Here we use the usual Sobolev spaces $H^s(\omega)$, with s a nonnegative integer and ω an open subset on $\mathbb{R}^n$. The norm on this space is

$$\|\varphi\|_{s;\omega} = \left(\sum_{|\alpha| \le s} \int_\omega |\partial^\alpha \varphi|^2\, dx \right)^{1/2},$$

where the summation is over all nonnegative n-multi-indices α. When $\omega = \Omega$, the subscript ω in the norms will be dropped. If $b \ge 0$ on Ω, $c \ge 0$ on Γ_N, and $\operatorname{meas}(\Gamma_D) > 0$, then the bilinear form a is coercive on V_0, and it is easy to apply the Lax-Milgram lemma to deduce the existence of a unique solution $\hat{u}$ of (5) (see for example pages 164–167 in Quarteroni and Valli [31]). Existence and uniqueness holds in other cases as well; for example, for the pure Neumann problem ($c \equiv 0, \Gamma_D = \emptyset$), if $b > 0$ on Ω.

2.2 The discrete problem and the finite element method

Let $W_0 \subset V_0$ and $W_D \subset V_D$ be finite dimensional subspaces. The Galerkin method defines an approximation $\tilde{u}$ to $\hat{u}$ as the unique element of W_D for which

$$a(\tilde{u}, v) = l(v) \text{ for all } v \in W_0.$$

The finite element method involves special choices for the subspaces W_0 and W_D. For defining these subspaces, the domain Ω is partitioned by a conforming tetrahedral mesh. By a *tetrahedral mesh* of Ω we mean a set of closed, nondegenerate tetrahedra whose union equals the closure of Ω, and the interiors of the tetrahedra in the set are pairwise disjoint. A mesh $\mathcal{T}$ is said to be *conforming* if each face of each tetrahedron in $\mathcal{T}$ is either a subset of the boundary or a face of another

tetrahedron in $\mathcal{T}$. We shall assume that the mesh conforms to the subdivision of the boundary into its Dirichlet and Neumann parts in the sense that any face contained in Γ belongs entirely to Γ_D or entirely to Γ_N.

We now make some definitions that will be used to characterize different meshes of Ω. For an arbitrary tetrahedron τ, denote by $h(\tau)$ its diameter, by $\rho(\tau)$ the diameter of the largest ball contained in τ, and define by $\sigma(\tau) = h(\tau)/\rho(\tau)$ a *shape constant* of τ. For a mesh $\mathcal{T}$, define the *mesh size* and *shape constant* by

$$h(\mathcal{T}) = \max_{\tau \in \mathcal{T}} h(\tau) \text{ and } \sigma(\mathcal{T}) = \max_{\tau \in \mathcal{T}} \sigma(\tau).$$

A family of meshes $\{\mathcal{T}\}$ is called *shape regular* if the following two conditions are satisfied:

$$\inf_{\mathcal{T}} h(\mathcal{T}) = 0 \text{ and } \sup_{\mathcal{T}} \sigma(\mathcal{T}) < \infty.$$

The definitions of the subspaces W_0 and W_D will require a classification of the vertices (nodes) of tetrahedra appearing in a mesh. To this end, denote by $\mathcal{N}(\tau)$ the vertex set of τ, and let $\mathcal{N}(\mathcal{T}) = \bigcup_{\tau \in \mathcal{T}} \mathcal{N}(\tau)$. The set $\mathcal{N}(\mathcal{T})$ is partitioned into Dirichlet, Neumann, and interior parts given by

$$\mathcal{N}_D(\mathcal{T}) = \left\{\nu \in \mathcal{N}(\mathcal{T}) \mid \nu \subseteq \overline{\Gamma}_D\right\}, \quad \mathcal{N}_N(\mathcal{T}) = \left\{\nu \in \mathcal{N}(\mathcal{T}) \mid \nu \subseteq \Gamma_N\right\},$$
$$\text{and} \quad \mathcal{N}_\Omega(\mathcal{T}) = \mathcal{N}(\mathcal{T}) \setminus \left[\mathcal{N}_D(\mathcal{T}) \cup \mathcal{N}_N(\mathcal{T})\right].$$

Denote by $P_1(\tau)$ the space of all linear functions on a tetrahedron τ, and set for an arbitrary conforming mesh $\mathcal{T}$

$$X(\mathcal{T}) = \left\{v \in H^1(\Omega) \mid v|_\tau \in P_1(\tau) \text{ for every } \tau \in \mathcal{T}\right\},$$
$$V_0(\mathcal{T}) = \left\{v \in X(\mathcal{T}) \mid v(\nu) = 0 \text{ for all } \nu \in \mathcal{N}_D(\mathcal{T})\right\},$$
$$V_D(\mathcal{T}) = \left\{v \in X(\mathcal{T}) \mid v(\nu) = g_D(\nu) \text{ for all } \nu \in \mathcal{N}_D(\mathcal{T})\right\}.$$

Choosing $W_0 = V_0(\mathcal{T})$ and $W_D = V_D(\mathcal{T})$, the *discrete problem* on $\mathcal{T}$ is:

Find $u_\mathcal{T} \in V_D(\mathcal{T})$ such that

$$a(u_\mathcal{T}, v) = l(v) \text{ for all } v \in V_0(\mathcal{T}). \tag{8}$$

The following convergence result is standard (see for example pages 171–174 in Quarteroni and Valli [31]).

Theorem 2.2.1 *Let Ω be a polyhedral, open, bounded subset of $\mathbb{R}^3$ and let $\{\mathcal{T}\}$ be a family of conforming, shape regular meshes of Ω. Suppose that the bilinear form $a(\cdot,\cdot)$ is continuous on $V_D(\mathcal{T}) \times V_0(\mathcal{T})$, V-elliptic on $V_0(\mathcal{T}) \times V_0(\mathcal{T})$, and that the linear functional $l(\cdot)$ is continuous on V_0. Then, the discrete problem has a unique solution $u_{\mathcal{T}} \in V_D(\mathcal{T})$ for every $\mathcal{T}$ in the family, and $u_{\mathcal{T}}$ converges to $\hat{u}$ in $H^1(\Omega)$ as $h(\mathcal{T}) \to 0$. Moreover, there exists a constant C such that if the exact solution $\hat{u} \in H^2(\Omega)$, then*

$$\|\hat{u} - u_{\mathcal{T}}\|_1 \leq C h(\mathcal{T}) \|\hat{u}\|_2.$$

2.3 A residual error estimator

To obtain a prescribed level of accuracy in the discrete solution, the mesh needs to be sufficiently fine. For problems where the solution or its derivatives has singular or nearly singular behavior in certain parts of the domain only, we will use a finer mesh in these regions than in the remaining part of the domain. This decreases the size of the linear system that needs to be solved to obtain a discrete solution to some prescribed accuracy relative to a uniform mesh. For the efficient implementation of an adaptive procedure along these lines, we need to know which portions of the mesh needs to be refined. This may be decided a priori based on the problem and/or the geometry. Another approach is to use an a posteriori error estimator. A posteriori error estimators are quantities that are computed using the discrete solution and information about the mesh, and have been the subject of extensive study (see for example Verfurth [33]). The main characteristics that are desirable in an a posteriori error estimator are the following:

- It should be computable from the discrete solution on the mesh, mesh information, and the data of the problem.

- Its computation should be far less expensive than the computation of the discrete solution.
- It should yield reliable upper and lower bounds to the true error in some norm.

We will use a residual error estimator which assigns a number to each tetrahedron in a mesh. The basic idea is to estimate the residual of the discrete solution $u_{\mathcal{T}}$ in an appropriate norm. This section briefly discusses the derivation of the residual error estimator for each tetrahedron in a mesh based on an approach of Verfürth [33].

Let $\mathcal{T}$ be a conforming mesh of Ω, and $u_{\mathcal{T}}$ denote the finite element (discrete) solution associated with $\mathcal{T}$. Let $R : H^1(\Omega) \to V_0^*$ denote the *residual* mapping defined as follows:

For all $u \in H^1(\Omega)$ and $v \in V_0$,

$$\langle R(u), v\rangle = a(u,v) - l(v) = a(u - \hat{u}, v). \tag{9}$$

Then for any $u \in V_D$ and $v \in V_0$,

$$\|\hat{u} - u\|_1 \leq C \sup_{\substack{v \in V_0 \\ \|v\|_1 = 1}} a(\hat{u} - u, v) \leq C \|R(u)\|_{V_0^*} . \tag{10}$$

Thinking of u as an approximation to the exact solution, we see that the H^1 norm of the error is bounded by the norm of the residual in the dual space. In deriving an expression for the residual error estimator the right hand side of (10) will be approximated by quantities that are computable using the discrete solution and the functions appearing in the boundary value problem (1)–(3). We now make some definitions associated with the faces of a mesh $\mathcal{T}$. For any $\tau \in \mathcal{T}$, denote by $\mathcal{F}(\tau)$ the set of its faces and let $\mathcal{F}(\mathcal{T}) = \bigcup_{\tau \in \mathcal{T}} \mathcal{F}(\tau)$. The face set $\mathcal{F}(\mathcal{T})$ is partitioned into Dirichlet, Neumann, and interior parts given by

$$\mathcal{F}_D(\mathcal{T}) = \left\{\mu \in \mathcal{F}(\mathcal{T}) \mid \mu \subseteq \overline{\Gamma}_D\right\}, \quad \mathcal{F}_N(\mathcal{T}) = \left\{\mu \in \mathcal{F}(\mathcal{T}) \mid \mu \subseteq \overline{\Gamma}_N\right\},$$
$$\text{and} \quad \mathcal{F}_\Omega(\mathcal{T}) = \mathcal{F}(\mathcal{T}) \setminus \left[\mathcal{F}_D(\mathcal{T}) \cup \mathcal{F}_N(\mathcal{T})\right].$$

For any $u \in V_D$ and $v \in V_0$, integration by parts gives

$$\begin{aligned}\langle R(u), v\rangle &= \int_\Omega \left[\underset{\approx}{\mathcal{A}}\underset{\sim}{\operatorname{grad}}\, u \cdot \underset{\sim}{\operatorname{grad}}\, v + (bu - f)v\right] dx + \int_{\Gamma_N} (cu - g_N)\, v\, ds \\ &= \sum_{\tau\in\mathcal{T}} \int_\tau \left[-\operatorname{div}(\underset{\approx}{\mathcal{A}}\underset{\sim}{\operatorname{grad}}\, u) + bu - f\right] v\, dx \\ &\quad + \sum_{\tau\in\mathcal{T}} \int_{\partial\tau} \left(\underset{\approx}{\mathcal{A}}\underset{\sim}{\operatorname{grad}}\, u \cdot \underset{\sim}{n}_\tau\right) v\, ds + \sum_{\mu\in\mathcal{F}_N(\mathcal{T})} \int_\mu (cu - g_N)\, v\, ds.\end{aligned} \tag{11}$$

Here $\underset{\sim}{n}_\tau$ denotes the unit outward normal to the tetrahedron τ. For any face $\mu \in \mathcal{F}_N(\mathcal{T}) \cup \mathcal{F}_\Omega(\mathcal{T})$, let $\underset{\sim}{n}_\mu$ denote the unit normal to μ, where the normal is assumed to be pointing outward with respect to Ω if μ is a Neumann face and is arbitrary for an interior face. Let $\varphi \in L^2(\Omega)$ be piecewise continuous with respect to the mesh $\mathcal{T}$. For any $\mu \in \mathcal{F}_\Omega(\mathcal{T})$ denote by $[\varphi]_\mu$ the *jump* of φ across the face μ in the direction of $\underset{\sim}{n}_\mu$:

$$[\varphi]_\mu(\underset{\sim}{x}) := \lim_{t\to 0+} \varphi(\underset{\sim}{x} + t\underset{\sim}{n}_\mu) - \lim_{t\to 0-} \varphi(\underset{\sim}{x} - t\underset{\sim}{n}_\mu) \text{ for all } \underset{\sim}{x} \in \mu.$$

The discrete problem (8) gives $\langle R(u_\mathcal{T}), v_\mathcal{T}\rangle = 0$ for all $v_\mathcal{T} \in V_0(\mathcal{T})$. Combining this with (11), we obtain:

For any $v \in V_0$ and $v_\mathcal{T} \in V_0(\mathcal{T})$,

$$\begin{aligned}|\langle R(u_\mathcal{T}), v\rangle| &= |\langle R(u_\mathcal{T}), v - v_\mathcal{T}\rangle| \\ &\le \sum_{\tau\in\mathcal{T}} \left\|-\operatorname{div}(\underset{\approx}{\mathcal{A}}\underset{\sim}{\operatorname{grad}}\, u_\mathcal{T}) + bu_\mathcal{T} - f\right\|_{0;\tau} \|v - v_\mathcal{T}\|_{0;\tau} \\ &\quad + \sum_{\mu\in\mathcal{F}_\Omega(\mathcal{T})} \left\|\left[\underset{\approx}{\mathcal{A}}\underset{\sim}{\operatorname{grad}}\, u_\mathcal{T} \cdot \underset{\sim}{n}_\mu\right]_\mu\right\|_{0;\mu} \|v - v_\mathcal{T}\|_{0;\mu} \\ &\quad + \sum_{\mu\in\mathcal{F}_N(\mathcal{T})} \left\|\underset{\approx}{\mathcal{A}}\underset{\sim}{\operatorname{grad}}\, u_\mathcal{T} \cdot \underset{\sim}{n}_\mu + cu_\mathcal{T} - g_N\right\|_{0;\mu} \|v - v_\mathcal{T}\|_{0;\mu}.\end{aligned} \tag{12}$$

Now choose $v_\mathcal{T} = I_\mathcal{T} v$, where $I_\mathcal{T} v$ is the Clément interpolant [16] of v. We recall the approximation properties of this operator. For $\tau \in \mathcal{T}$ and $\mu \in \mathcal{F}(\mathcal{T})$ let

$$\omega_\tau := \bigcup \{\tau' \in \mathcal{T} \mid \tau \cap \tau' \neq \emptyset\} \text{ and } \omega_\mu := \bigcup \{\tau' \in \mathcal{T} \mid \mu \cap \tau' \neq \emptyset\}$$

be the union of tetrahedra meeting τ and μ respectively. Then for all $v \in V_0$

$$\|v - I_{\mathcal{T}} v\|_{0;\tau} \leq c_1 h(\tau) \|v\|_{1;\omega_\tau} \text{ and } \|v - I_{\mathcal{T}} v\|_{0;\mu} \leq c_2 \left[h(\mu)\right]^{1/2} \|v\|_{1;\omega_\mu} ,$$

where c_1 and c_2 depend on the shape constant $\sigma(\mathcal{T})$, and $h(\mu) := \operatorname{diam}(\mu)$. These estimates are proved in [33]. Choosing $v_{\mathcal{T}} = I_{\mathcal{T}} v$ in (12) we have

$$\begin{aligned}
|\langle R(u_{\mathcal{T}}), v\rangle| \leq & \sum_{\tau\in\mathcal{T}} c_1 h(\tau) \left\| -\operatorname{div}(\underset{\approx}{\mathcal{A}} \underset{\sim}{\operatorname{grad}}\, u_{\mathcal{T}}) + b u_{\mathcal{T}} - f \right\|_{0;\tau} \|v\|_{1;\omega_\tau} \\
& + \sum_{\mu\in\mathcal{F}_\Omega(\mathcal{T})} c_2 \left[h(\mu)\right]^{1/2} \left\| \left[\underset{\approx}{\mathcal{A}} \underset{\sim}{\operatorname{grad}}\, u_{\mathcal{T}} \cdot \underset{\sim}{n}_\mu \right]_\mu \right\|_{0;\mu} \|v\|_{1;\omega_\mu} \\
& + \sum_{\mu\in\mathcal{F}_N(\mathcal{T})} c_2 \left[h(\mu)\right]^{1/2} \left\| \underset{\approx}{\mathcal{A}} \underset{\sim}{\operatorname{grad}}\, u_{\mathcal{T}} \cdot \underset{\sim}{n}_\mu + c u_{\mathcal{T}} - g_N \right\|_{0;\mu} \|v\|_{1;\omega_\mu} \\
\leq & \max\{c_1, c_2\} \left(\sum_{\tau\in\mathcal{T}} \left[h(\tau)\right]^2 \left\| -\operatorname{div}(\underset{\approx}{\mathcal{A}} \underset{\sim}{\operatorname{grad}}\, u_{\mathcal{T}}) + b u_{\mathcal{T}} - f \right\|_{0;\tau}^2 \right. \\
& \qquad + \sum_{\mu\in\mathcal{F}_\Omega(\mathcal{T})} h(\mu) \left\| \left[\underset{\approx}{\mathcal{A}} \underset{\sim}{\operatorname{grad}}\, u_h \cdot \underset{\sim}{n}_\mu \right]_\mu \right\|_{0;\mu}^2 \\
& \qquad \left. + \sum_{\mu\in\mathcal{F}_N(\mathcal{T})} h(\mu) \left\| \underset{\approx}{\mathcal{A}} \underset{\sim}{\operatorname{grad}}\, u_{\mathcal{T}} \cdot \underset{\sim}{n}_\mu + c u_{\mathcal{T}} - g_N \right\|_{0;\mu}^2 \right)^{1/2} \\
& \qquad \times \left(\sum_{\tau\in\mathcal{T}} \|v\|_{1;\omega_\tau}^2 + \sum_{\mu\in\mathcal{F}_\Omega(\mathcal{T})\cup\mathcal{F}_N(\mathcal{T})} \|v\|_{1;\omega_\mu}^2 \right)^{1/2} \\
\leq & c_{\mathcal{T}} \|v\|_1 \left(\sum_{\tau\in\mathcal{T}} \left[h(\tau)\right]^2 \left\| -\operatorname{div}(\underset{\approx}{\mathcal{A}} \underset{\sim}{\operatorname{grad}}\, u_{\mathcal{T}}) + b u_{\mathcal{T}} - f \right\|_{0;\tau}^2 \right. \\
& \qquad + \sum_{\mu\in\mathcal{F}_\Omega(\mathcal{T})} h(\mu) \left\| \left[\underset{\approx}{\mathcal{A}} \underset{\sim}{\operatorname{grad}}\, u_{\mathcal{T}} \cdot \underset{\sim}{n}_\mu \right]_\mu \right\|_{0;\mu}^2 \\
& \qquad \left. + \sum_{\mu\in\mathcal{F}_N(\mathcal{T})} h(\mu) \left\| \underset{\approx}{\mathcal{A}} \underset{\sim}{\operatorname{grad}}\, u_{\mathcal{T}} \cdot \underset{\sim}{n}_\mu + c u_{\mathcal{T}} - g_N \right\|_{0;\mu}^2 \right)^{1/2} ,
\end{aligned}$$

for all $v \in V_0$, where $c_{\mathcal{T}}$ depends on the shape constant $\sigma(\mathcal{T})$. Choosing $u = u_{\mathcal{T}}$ in (10) and using the last estimate we get

$$\|\hat{u} - u_{\mathcal{T}}\|_1 \leq C \left(\sum_{\tau\in\mathcal{T}} \left[h(\tau)\right]^2 \left\| -\operatorname{div}(\underset{\approx}{\mathcal{A}} \underset{\sim}{\operatorname{grad}}\, u_{\mathcal{T}}) + b u_{\mathcal{T}} - f \right\|_{0;\tau}^2 \right.$$

$$+ \sum_{\mu \in \mathcal{F}_\Omega(\mathcal{T})} h(\mu) \left\| \left[\underset{\approx}{\mathcal{A}} \underset{\sim}{\mathrm{grad}}\, u_{\mathcal{T}} \cdot \underset{\sim}{n}_\mu \right]_\mu \right\|_{0;\mu}^2$$

$$+ \sum_{\mu \in \mathcal{F}_N(\mathcal{T})} h(\mu) \left\| \underset{\approx}{\mathcal{A}} \underset{\sim}{\mathrm{grad}}\, u_{\mathcal{T}} \cdot \underset{\sim}{n}_\mu + c u_{\mathcal{T}} - g_N \right\|_{0;\mu}^2 \Bigg)^{1/2}, \tag{13}$$

where C is a constant depending on the shape constant of the mesh. The right hand side of the last expression is a candidate for an a posteriori error estimator since it involves only the known data $\underset{\approx}{\mathcal{A}}$, b, f, c and g_N, and the discrete solution $u_{\mathcal{T}}$. We will define the residual error estimator for every tetrahedron in $\mathcal{T}$ based on this. The calculation of the integrals involved may be simplified by using quadrature. Alternatively, the functions involved in the data can be replaced by simpler functions (by projection onto appropriate spaces) and exact integration can be performed. We will follow the latter approach.

Let $\tau \in \mathcal{T}$ and $\mu \in \mathcal{F}(\tau)$ be arbitrary. Denote by x_{bc}^τ the barycenter of τ, and by x_{bc}^μ that μ. For an arbitrary continuous function φ, denote by φ_τ and φ_μ the projections onto the spaces of constant functions $P_0(\tau)$ and $P_0(\mu)$ given by the value φ at x_{bc}^τ and x_{bc}^μ. For an arbitrary matrix $\underset{\approx}{\mathcal{M}} = (m_{ij}) \in \mathbb{R}^{3\times 3}$, let

$$\underset{\sim}{\mathrm{div}} \underset{\approx}{\mathcal{M}} = \begin{bmatrix} \sum_{i=1}^3 \partial_i (m_{i1}) \\ \sum_{i=1}^3 \partial_i (m_{i2}) \\ \sum_{i=1}^3 \partial_i (m_{i3}) \end{bmatrix},$$

where $x_i, 1 \le i \le 3$, are the coordinates in $\mathbb{R}^3$, and ∂_i is the partial derivative with respect to x_i. Since $u_{\mathcal{T}}$ is piecewise linear, $\underset{\sim}{\mathrm{grad}}\, u_{\mathcal{T}}$ is constant on any tetrahedron τ. It follows that for each $\tau \in \mathcal{T}$

$$\mathrm{div} \left(\underset{\approx}{\mathcal{A}} \underset{\sim}{\mathrm{grad}}\, u_{\mathcal{T}} \right) = \underset{\sim}{\mathrm{div}} \underset{\approx}{\mathcal{A}} \cdot \underset{\sim}{\mathrm{grad}}\, u_{\mathcal{T}}.$$

The *residual error estimator* for any $\tau \in \mathcal{T}$ is now defined as

$$\eta_\tau = \left([h(\tau)]^2 \left\| - \left(\underset{\sim}{\mathrm{div}} \underset{\approx}{\mathcal{A}} \right)_\tau \cdot \underset{\sim}{\mathrm{grad}}\, u_{\mathcal{T}} + b_\tau u_{\mathcal{T}} - f_\tau \right\|_{0;\tau}^2 \right.$$

$$(14) \qquad + \frac{1}{2} \sum_{\mu \in \mathcal{F}(\tau) \cap \mathcal{F}_\Omega(\mathcal{T})} h(\mu) \left\| \left[\underset{\approx}{\mathcal{A}}_\mu \underset{\sim}{\operatorname{grad}}\, u_\mathcal{T} \cdot \underset{\sim}{n}_\mu \right]_\mu \right\|_{0;\mu}^2$$

$$+ \sum_{\mu \in \mathcal{F}(\tau) \cap \mathcal{F}_N(\mathcal{T})} h(\mu) \left\| \underset{\approx}{\mathcal{A}}_\mu \underset{\sim}{\operatorname{grad}}\, u_\mathcal{T} \cdot \underset{\sim}{n}_\mu + c_\mu u_\mathcal{T} - (g_N)_\mu \right\|_{0;\mu}^2 \Bigg)^{1/2},$$

where $\left(\underset{\sim}{\operatorname{div}} \underset{\approx}{\mathcal{A}} \right)_\tau$ is the matrix obtained from $\underset{\sim}{\operatorname{div}} \underset{\approx}{\mathcal{A}}$ by replacing each entry by its projection onto $P_0(\tau)$, and $\underset{\approx}{\mathcal{A}}_\mu$ is similarly defined using $\underset{\approx}{\mathcal{A}}$ and $P_0(\mu)$.

Remark 2.3.1 The first term in the expression for η_τ is related to the residual of $u_\mathcal{T}$, the second to the jump of its gradient across interior faces, and the third to the boundary operator. In the simple case where the matrix $\underset{\approx}{\mathcal{A}}$ is constant, the term $\left(\underset{\sim}{\operatorname{div}} \underset{\approx}{\mathcal{A}} \right)_\tau \cdot \underset{\sim}{\operatorname{grad}}\, u_\mathcal{T}$ drops out from the calculation of η_τ.

Observing that each interior face is counted twice in a summation over all tetrahedra in the mesh, we get

$$\begin{aligned} \|\hat{u} - u_\mathcal{T}\|_1 \leq C \Bigg(& \sum_{\tau \in \mathcal{T}} {\eta_\tau}^2 + \sum_{\tau \in \mathcal{T}} \left\| [h(\tau)]^2 \left\{ \left(\underset{\sim}{\operatorname{div}} \underset{\approx}{\mathcal{A}} x \right)_\tau - \left(\underset{\sim}{\operatorname{div}} \underset{\approx}{\mathcal{A}} \right) \right\} \cdot \underset{\sim}{\operatorname{grad}}\, u_\mathcal{T} \right\|_{0;\tau}^2 \\ & + \sum_{\tau \in \mathcal{T}} [h(\tau)]^2 \left\| (b - b_\tau)\, u_\mathcal{T} \right\|_{0;\tau}^2 + \sum_{\tau \in \mathcal{T}} [h(\tau)]^2 \left\| f - f_\tau \right\|_{0;\tau}^2 \\ & + \sum_{\mu \in \mathcal{F}_\Omega(\mathcal{T})} h(\mu) \left\| \left[\left(\underset{\approx}{\mathcal{A}} - \underset{\approx}{\mathcal{A}}_\mu \right) \underset{\sim}{\operatorname{grad}}\, u_\mathcal{T} \cdot \underset{\sim}{n}_\mu \right]_\mu \right\|_{0;\mu}^2 \\ & + \sum_{\mu \in \mathcal{F}_N(\mathcal{T})} h(\mu) \left\| \left(\underset{\approx}{\mathcal{A}} - \underset{\approx}{\mathcal{A}}_\mu \right) \underset{\sim}{\operatorname{grad}}\, u_\mathcal{T} \cdot \underset{\sim}{n}_\mu \right\|_{0;\mu}^2 \\ & + \sum_{\mu \in \mathcal{F}_N(\mathcal{T})} h(\mu) \left\| (c - c_\mu)\, u_\mathcal{T} \right\|_{0;\mu}^2 + \sum_{\mu \in \mathcal{F}_N(\mathcal{T})} h(\mu) \left\| g_N - (g_N)_\mu \right\|_{0;\mu}^2 \Bigg)^{1/2} \end{aligned}$$

from (13) and (14) by using the triangle inequality. The last expression says that the H^1 norm of the error is bounded by the residual on the mesh together with the L^2 norms of some consistency terms that arise due to the projections of the input data onto appropriate subspaces. On the other hand, Verfürth [33] has shown that η_τ is bounded by the H^1 norm of the error on ω_τ and the L^2 norms of some appropriate consistency terms for a general class of elliptic problems. The model

problem of this section is a special case of the general problem considered there and the result holds for the model problem:

$$\eta_\tau \leq C \left(\|\hat{u} - u_{\mathcal{T}}\|^2_{1;\omega_\tau} + \text{ consistency terms } \right)^{1/2} .$$

Note that the residual error estimator satisfies the requirements that were listed for a good a posteriori error estimator.

Observing that $\underset{\sim}{\text{grad}}\, u_{\mathcal{T}}$ and $\underset{\sim}{n}_\mu$ are constant on any τ and μ, we get

$$\begin{aligned}
\eta_\tau^2 = [h(\tau)]^2 \left\{ - \left(\underset{\sim}{\text{div}} \underset{\approx}{\mathcal{A}} \right) (x_{bc}^\tau) \cdot \underset{\sim}{\text{grad}}\, u_{\mathcal{T}} + b\,(x_{bc}^\tau)\, u_{\mathcal{T}}\,(x_{bc}^\tau) - f\,(x_{bc}^\tau) \right\}^2 \text{meas}\,(\tau) \\
+ \frac{1}{2} \sum_{\mu \in \mathcal{F}(\tau) \cap \mathcal{F}_\Omega(\mathcal{T})} h(\mu) \left\{ \left[\underset{\approx}{\mathcal{A}}\,(x_{bc}^\mu)\, \underset{\sim}{\text{grad}}\, u_{\mathcal{T}} \cdot \underset{\sim}{n}_\mu \right]_\mu \right\}^2 \text{meas}\,(\mu) \\
+ \sum_{\mu \in \mathcal{F}(\tau) \cap \mathcal{F}_N(\mathcal{T})} h(\mu) \left\{ \underset{\approx}{\mathcal{A}}\,(x_{bc}^\mu)\, \underset{\sim}{\text{grad}}\, u_{\mathcal{T}} \cdot \underset{\sim}{n}_\mu + c\,(x_{bc}^\mu)\, u_{\mathcal{T}}\,(x_{bc}^\mu) - g_N\,(x_{bc}^\mu) \right\}^2 \text{meas}\,(\mu) .
\end{aligned}$$

This expression is used for calculating η_τ in AMG3DP1.

Chapter 3

Bisection of Tetrahedra and adaptive mesh refinement

In this chapter we present an algorithm for producing a nested sequence of conforming, locally adapted tetrahedral meshes starting with a coarse conforming mesh. The algorithm yields a conforming mesh by first subdividing a selected subset of tetrahedra in a given conforming mesh and then continuing to subdivide other tetrahedra as needed until a conforming mesh is obtained. The second stage of the algorithm is called closure refinement. We also ensure that the sequence of nested meshes produced by the algorithm have a bounded shape constant. In describing our mesh refinement algorithm, we will follow a two-part approach. First, we present an algorithm for the subdivision of a single tetrahedron. The second part makes repeated use of this algorithm to produce a conforming mesh from a given conforming mesh and a selected subset of it. Similar approaches for producing adapted, conforming meshes in two and three dimensions have been presented by many authors (see for example [5, 8, 25, 29]).

There are two main approaches for subdividing a single tetrahedron: octasection and bisection. Zhang [38] and Ong [30] have studied the octasection of a single tetrahedron and its use to obtain uniformly refined meshes. Octasection has to be augmented by bisection to obtain meshes that are locally adapted. Bey [11] has recently studied the use of octasection augmented by bisection to produce locally adapted tetrahedral meshes. Bisection is an alternative to octasection for subdividing a single tetrahedron and it is simpler than octasection in the sense that subdividing a tetrahedron using bisection produces one new node, while octasection gives rise to six. The number of new nodes produced is of considerable importance

since all neighboring tetrahedra in the mesh that share an edge containing a new node have to be subdivided to yield a conforming mesh. The implementation of the closure refinement is therefore more complicated if octasection is used for subdividing a single tetrahedron. However, it is harder to ensure that the tetrahedra produced by bisection are nondegenerate. The exclusive use of bisection to produce locally adaptive meshes has also been studied before. The algorithm used in AMG3DP1 is based on a bisection algorithm of Bänsch [8]. However, the description of the algorithm given in this chapter different and easier to follow. A detailed description of algorithm *bisect-tet* for subdividing a single tetrahedron and some related properties are given in Section 1.

Two tetrahedra are *similar* if one can be transformed into the other by translation and uniform scaling. By definition, similar tetrahedra remain similar after any affine transformation.

Remark 3.0.1 Note that since uniform scaling allows for scaling by negative numbers, tetrahedra that are related via "reflection with respect to the origin" are similar. On the other hand, tetrahedra related to each other via reflections relative to lines (or via rotations) are not similar. Thus, the definition of similar used here is different from the usual definition. However, this definition is commonly used in the literature (see for example [11, 24, 26]).

It has been shown by Bänsch [8] that the tetrahedra produced by his bisection algorithm are nondegenerate, but he does not provide a bound for the number of similarity classes of tetrahedra produced. On the other hand, Maubach [26] provides an algorithm for the bisection of n-simplices for any n which has the advantage of being in a compact form, and it has been shown by Arnold and Pouly [1] that the number of similarity classes of n-simplices produced by this algorithm is bounded by $nn!2^{n-2}$. They have also shown that this bound is sharp for $n = 3$. We show in Section 2 that bisect-tet is in a certain sense equivalent to the Maubach algorithm for

$n = 3$, and hence prove that the number of similarity classes of tetrahedra produced by repeatedly applying bisect-tet is at most 36. The Maubach algorithm is only applicable to certain types of tetrahedra. However, we show using the properties of bisect-tet that this does not interfere with our analysis, and that the repeated application of bisect-tet to arbitrary tetrahedra still produce a finite number of similarity classes. The number 36 is an improvement on an upper bound of 168 given by Liu and Joe [24] for an equivalent bisection algorithm based on longest edge bisection.

In Section 3, we present algorithm *local refine* which performs the adaptive mesh refinement process. This algorithm is recursive and we show that it terminates in a finite number of steps for the particular application that we consider. Arguments for the termination of similar recursive algorithms have been provided by Bänsch [8] and Liu and Joe [24]. However, the arguments given in Section 3 are self contained and provide a detailed and clear proof that is different from these.

The bisection algorithms of Bänsch [8], Liu and Joe [24], and Maubach [26] are all similar and equivalent to each other. The algorithm presented in this chapter is equivalent to all of these and we give detailed proofs of the essential features that are needed for the success of such an algorithm.

3.1 Bisection of a single tetrahedron

In this section we describe algorithm bisect-tet for bisecting an arbitrary tetrahedron. The bisection of a tetrahedron entails the introduction of a new vertex at the midpoint of a selected edge of the tetrahedron which we call the *refinement edge*. Two new edges are created by joining the new vertex to the two vertices of the original tetrahedron that do not lie on the refinement edge. The key ingredient in describing bisect-tet is a specification of the refinement edge. For this purpose, define a *marked tetrahedron* to be a tetrahedron for which the refinement edge is specified; some additional information is also assumed to be given. This additional

information is used to specify the refinement edge and additional information for the children produced by bisect-tet. Precisely, we have:

(1) Each marked tetrahedron has a unique refinement edge on which the new vertex is created when the tetrahedron is bisected. The two faces of the tetrahedron that intersect at the refinement edge are its *refinement faces*. The refinement edge is taken as the *marked edge* for each of the two refinement faces.

(2) Each nonrefinement face of a marked tetrahedron also has a marked edge. A tetrahedron is called *planar* if the marked edges of its four faces lie on a plane.

(3) Each marked tetrahedron is assigned a *flag*; the flag will always be unset if the tetrahedron is nonplanar and it may or may not be set if the tetrahedron is planar.

Each marked nonrefinement edge of a marked tetrahedron is either adjacent or opposite to the refinement edge. Thus, there are 3 *types* of marked tetrahedra.

(1) Adjacent-Adjacent (each nonrefinement face has an adjacent marking); these are subclassified into planar and nonplanar. The nonplanar case will be denoted by NP-AA (Non-Planar Adjacent-Adjacent), and the planar case by P (Planar). The planar case is further classified according to whether it is flagged or not. These two subcases will be denoted by P-F (Planar-Flagged) and P-UF (Planar-UnFlagged), and how the two arise will be clear after the description of bisect-tet is complete.

(2) Opposite-Opposite (each nonrefinement face has an opposite marking); this will be denoted by NP-OO. In this case, a pair of opposite edges are marked in the tetrahedron; one as the refinement edge, and the other as the marked edge of the two nonrefinement faces intersecting there.

(3) Adjacent-Opposite (one nonrefinement face has an adjacent marking while the other has an opposite marking); this will be denoted by NP-AO.

Figures 1 and 2 show the four types of tetrahedra. Each marked nonrefinement edge

is indicated by a double line and the refinement edge is indicated as a marked edge for both faces containing it. The tetrahedra in Figure 2 are obtained from their counterparts in Figure 1 by cutting open along the edges incident on the vertex at the peak (the faces of the tetrahedra in Figure 2 are shown as triangles of equal area for convenience).

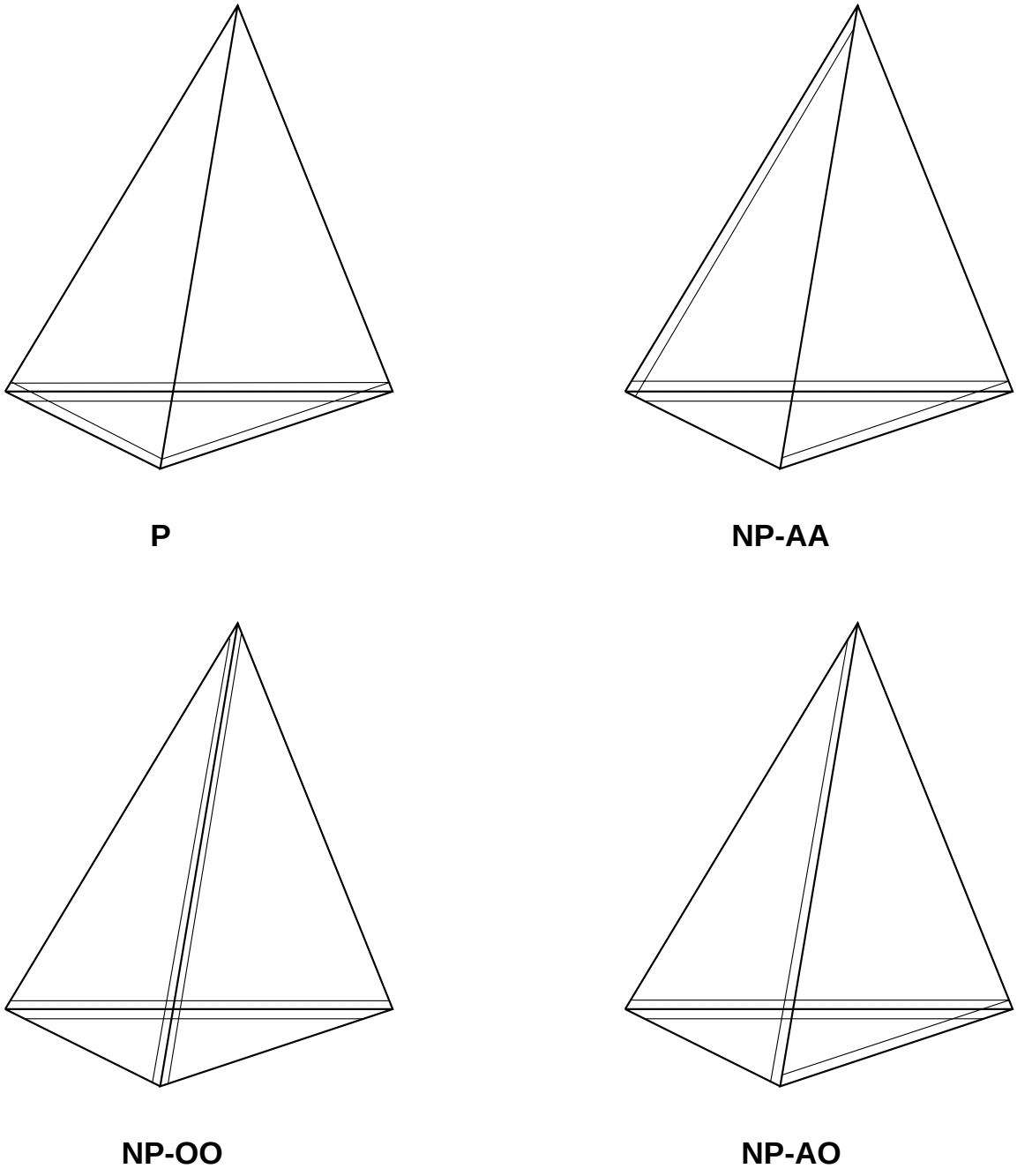

Figure 1. The 4 types of tetrahedra.

Figures showing the typical bisection of a tetrahedron and the bisection rules for all tetrahedra will use the open views of Figure 2, while a figure in Section 3 showing all the tetrahedra created by three bisections of a parent will use the closed view of Figure 1.

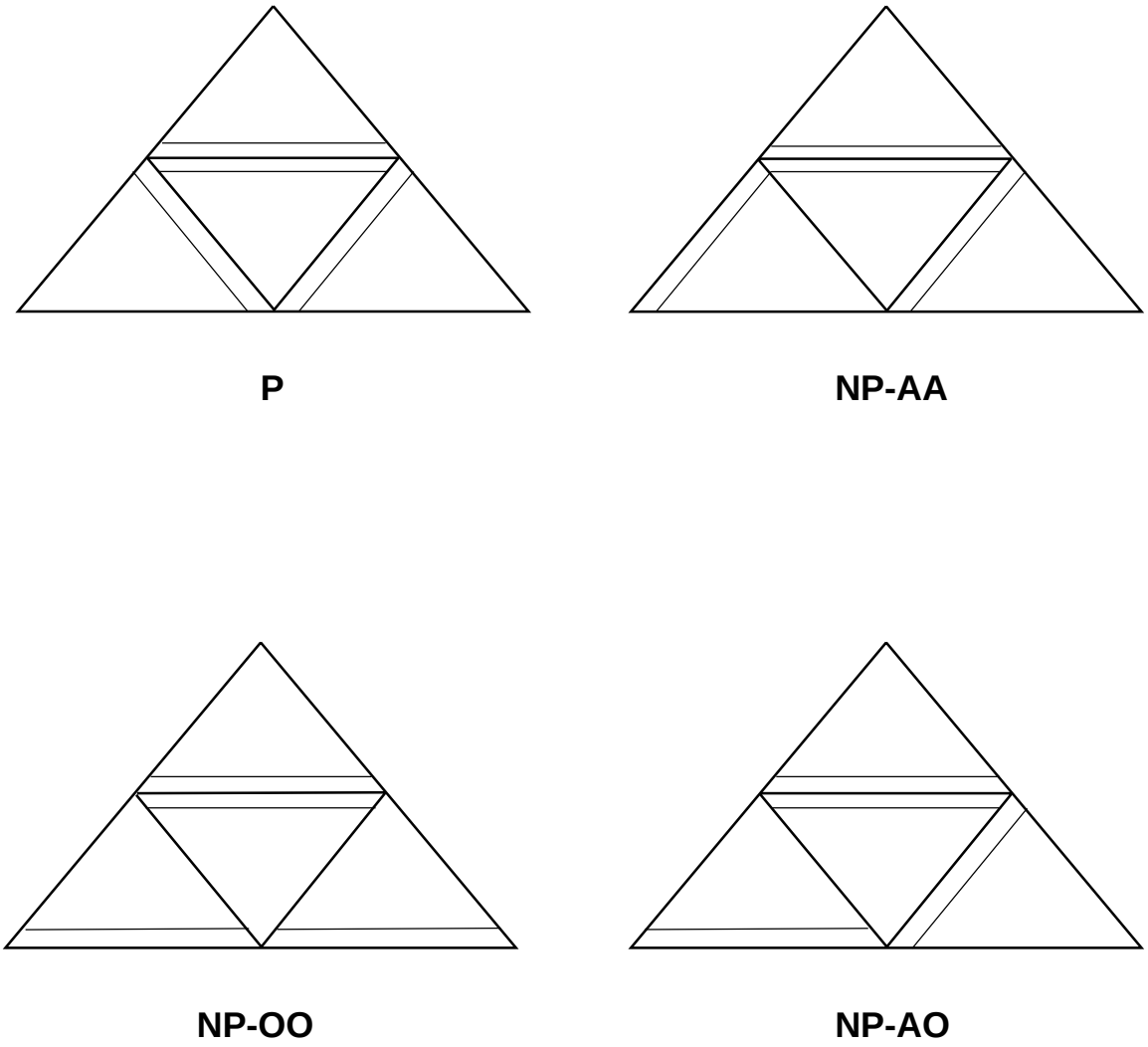

Figure 2. The 4 types of tetrahedra – cut open.

A marked tetrahedron τ will be denoted by the 4-tuple $(\mathcal{N}(\tau), r_\tau, m_\tau, f_\tau)$, where

(1) $\mathcal{N}(\tau) = \{x_0, x_1, x_2, x_3\}$ denotes the vertex set of τ ($x_i \in \mathbb{R}^3, 0 \leq i \leq 3$) as in Chapter 2.

(2) r_τ is a particular edge of τ (representing its refinement edge).

(3) m_τ is an ordered pair of (not necessarily distinct) edges of τ with the property that neither is the refinement edge and each nonrefinement face of τ has exactly one element of m_τ (these are the marked nonrefinement edges of τ).

(4) f_τ is a flag which is always unset ($= 0$) if τ is nonplanar, and it may or may not be set ($= 1$) if τ is planar.

Denote by $\mathcal{T}_M$ the set of all marked tetrahedra. As in Chapter 2, let $\mathcal{F}(\tau)$ denote the face set of a tetrahedron τ. Let a tetrahedron τ be bisected to create children $\tau^{(1)}$ and $\tau^{(2)}$. A face $\mu \in \mathcal{F}\left(\tau^{(i)}\right), i = 1, 2$ will be called an *inherited face* if $\mu \in \mathcal{F}(\tau)$, and a *cut face* if $\mu \subsetneqq \mu'$ for some $\mu' \in \mathcal{F}(\tau)$. The bisection process will also produce a *new face* and four *new edges*. Two of these will be produced via the bisection of an edge of the parent. We will give a detailed classification of the edges produced by the repeated application of bisect-tet in Section 3. Figure 3 shows the inherited (i), cut (c), and new (n) faces in a typical bisection. The new vertex is denoted by n and the new edges are $1n, 2n$ (obtained by the bisection of edge 12 of the parent), and $3n, 4n$.

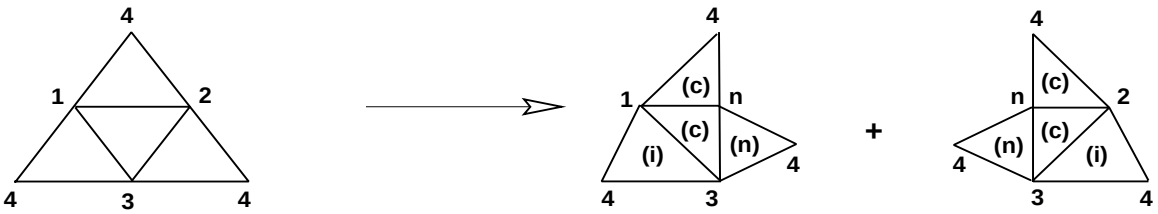

Figure 3. Typical bisection of a tetrahedron.

Given a marked tetrahedron τ (P-F, P-UF, NP-AA, NP-OO, or NP-AO), for a description of algorithm bisect-tet, it suffices to give the marked edges for all faces of $\tau^{(1)}$ and $\tau^{(2)}$, and to specify if the flag is set or unset for a planar $\tau^{(i)}$. Moreover, the marked edges for the bisected tetrahedra should be such that each $\tau^{(i)}$ has a unique refinement edge.

Algorithm 3.1.1

$\left\{\tau^{(1)}, \tau^{(2)}\right\} \longleftarrow$ bisect-tet (τ)

Input: A marked tetrahedron τ.

Output: Marked tetrahedra $\tau^{(1)}$ and $\tau^{(2)}$.

(1) Bisect τ by joining the midpoint of its refinement edge to each of the two vertices not lying on the refinement edge to create children $\tau^{(1)}$ and $\tau^{(2)}$ (these are geometric tetrahedra without any markings and $\mathcal{N}\left(\tau^{(i)}\right)$ is known).

Mark the faces of the children as follows:

(2) Each child has exactly one inherited face. This face inherits its marked edge from the parent, and this marked edge is the refinement edge of the child.

(3) Each child has two cut faces. For each of these cut faces the marked edge is the edge opposite the new vertex with respect to the face.

(4) Each child has one new face (common to both children). The new face is marked the same way for both children. Unless the parent tetrahedron is planar and flagged, the marked edge for the new face is the edge opposite the new vertex with respect to this face. If the parent is P-F, it is the edge connecting the new vertex to the new refinement edge.

(5) The flag is always unset in the children, unless the parent is planar and unflagged.

The bisection for the five cases (P-UF, NP-AA, NP-OO, NP-AO, P-F) together with the markings for the children are shown in Figure 4.

Note that bisect-tet outputs only tetrahedra of types P-F, P-UF, or NP-AA. All of these are such that their marked nonrefinement edges are adjacent to the refinement edge. Thus if $\mathcal{T}_A$ denotes the image set of marked tetrahedra obtained via bisect-tet , $\mathcal{T}_A \subsetneqq \mathcal{T}_M$. Observe also that the repeated application of algorithm bisect-tet to a tetrahedron produces the cycle

$$\text{NP-AA} \longrightarrow \text{P-UF} \longrightarrow \text{P-F} \longrightarrow \text{NP-AA} \longrightarrow \cdots$$

for the types of tetrahedra that are generated (tetrahedra of the type NP-OO and

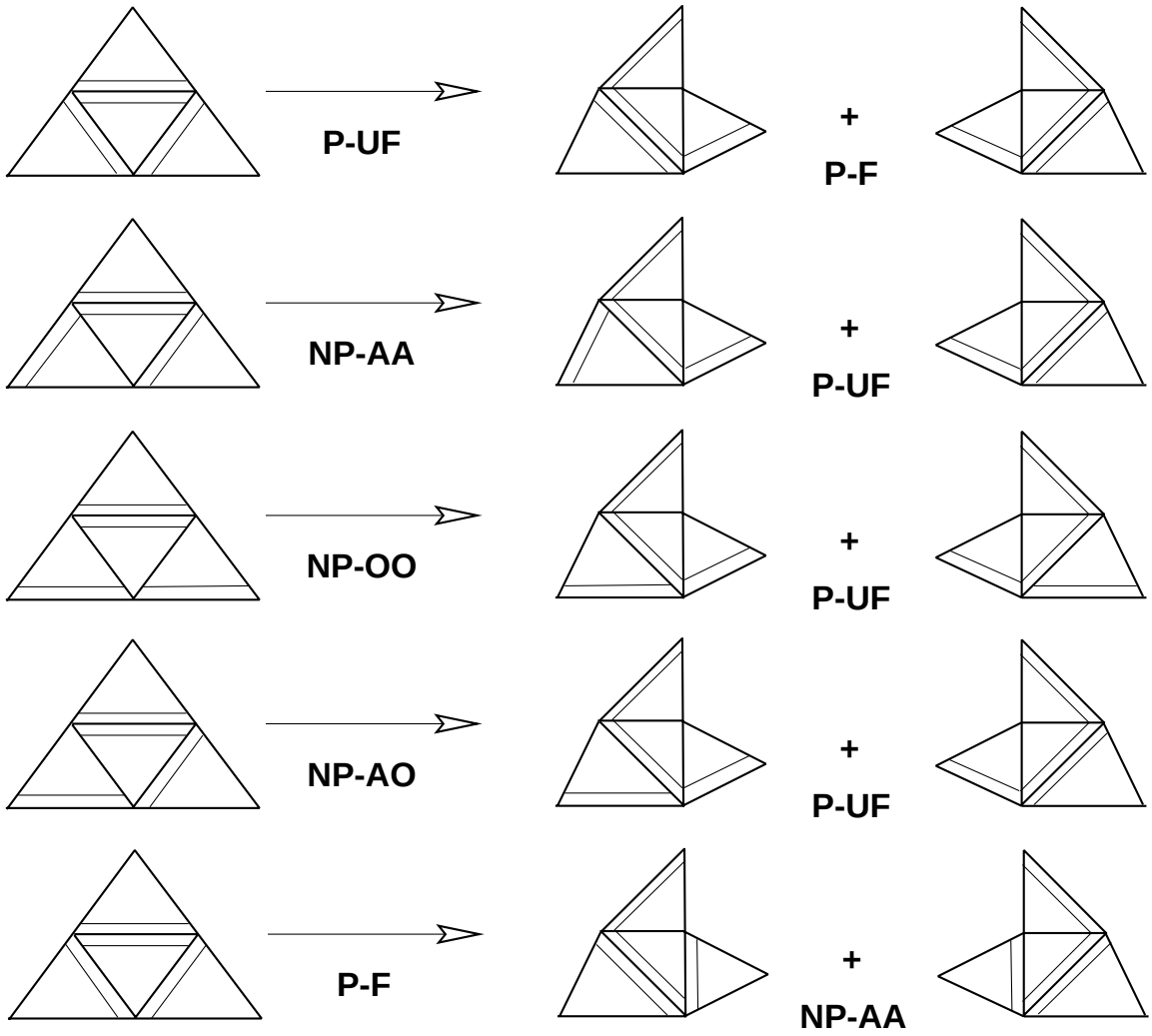

Figure 4. The bisection rules for all the cases.

NP-AO yield P-UF tetrahedra after one bisection). This is equivalent to the cycle given by Bänsch [8]. He calls the tetrahedra *red*, *black*, and *black-black* respectively. The next proposition is easily verified with reference to Figure 4.

Proposition 3.1.2

(1) The marked nonrefinement edges of the parent are the refinement edges of the children.

(2) A planar tetrahedron is flagged if and only if the new vertex created during its bisection is an endpoint of a marked nonrefinement edge for each of its children.

(3) If a tetrahedron is planar and unflagged, then each child is planar with the

plane opposite the new vertex containing all of its marked edges.

3.2 Similarity Classes

Maubach [26] has provided an algorithm for the bisection of an arbitrary n-simplex in $\mathbb{R}^n$. We now present his algorithm, and show that in the special case $n = 3$, it is equivalent to bisect-tet (if we restrict bisect-tet to having only NP-AA, P-UF, and P-F tetrahedra as input). Also, the local mesh refinement algorithm that Maubach provides is more restrictive than the one presented in the next section. The Maubach algorithm takes as input an ordered $(n+1)$-tuple of points in $\mathbb{R}^n$ giving the vertices of the n-simplex and a number $d \in \{1, 2, \ldots, n\}$ determining its refinement edge. Denote a *tagged-n-simplex* by $\mathfrak{t}$, where $\mathfrak{t} = ((x_0, x_1, \ldots, x_n), d)$ is a ordered pair defined as follows:

(1) $(x_0, x_1, \ldots, x_n)$ is an ordered $(n+1)$-tuple of points in $\mathbb{R}^n$ such that the set $\{x_0, x_1, \ldots, x_n\}$ are the vertices of the geometric n-simplex associated with $\mathfrak{t}$.

(2) $d \in \{1, 2, \ldots, n\}$ is a number such that $x_0 x_d$ is the refinement edge of $\mathfrak{t}$.

Algorithm 3.2.1

$$\left\{\mathfrak{t}^{(1)}, \mathfrak{t}^{(2)}\right\} \longleftarrow \text{bisect-n-simplex}\,(\mathfrak{t})$$

Input: A tagged-n-simplex $\mathfrak{t}$.

Output: Tagged-n-simplices $\mathfrak{t}^{(1)}$ and $\mathfrak{t}^{(2)}$.

(1) Set $d^{(i)} = \begin{cases} d-1 \text{ if } d > 1, \\ \\ n \qquad \text{if } d = 1. \end{cases}$

(2) Create new vertex $z = \frac{1}{2}(x_0 + x_d)$.

(3) Create $\mathfrak{t}^{(1)} = \left((x_0, x_1, \ldots, x_{d-1}, z, x_{d+1}, ..., x_n), d^{(1)}\right)$.

(4) Create $\mathfrak{t}^{(2)} = \left((x_1, x_2, \ldots, x_d, z, x_{d+1}, ..., x_n), d^{(2)}\right)$.

To prove that bisect-tet is equivalent to bisect-n-simplex when $n = 3$, we will show that the same tetrahedra (3-simplices) are produced by repeatedly applying bisect-tet (bisect-n-simplex) to an arbitrary tetrahedron (3-simplex). For an effective comparison of the two algorithms we need to associate tagged-3-simplices to

marked tetrahedra and vice versa. To this end, introduce the set $\mathcal{T}_T$ of all tagged-3-simplices, and define a mapping $\mathcal{F} : \mathcal{T}_T \to \mathcal{T}_M$ as follows:

Let $\mathfrak{t} = ((x_0, x_1, x_2, x_3), d) \in \mathcal{T}_T$. Then $\mathcal{F}(\mathfrak{t}) = (\mathcal{N}(\tau), r_\tau, m_\tau, f_\tau)$ with $\mathcal{N}(\tau) = \{x_0, x_1, x_2, x_3\}$ and

(1) If $d = 3$, $r_\tau = x_0x_3$, $m_\tau = (x_0x_2, x_1x_3)$, and $f_\tau = 0$ (τ is NP-AA).

(2) If $d = 2$, $r_\tau = x_0x_2$, $m_\tau = (x_0x_1, x_1x_2)$, and $f_\tau = 0$ (τ is P-UF).

(3) If $d = 1$, $r_\tau = x_0x_1$, $m_\tau = (x_0x_3, x_1x_3)$, and $f_\tau = 1$ (τ is P-F).

The next proposition follows from the definitions.

Proposition 3.2.2 *The following diagram commutes.*

$$\begin{array}{ccc} \mathcal{T}_T & \xrightarrow{\mathcal{F}} & \mathcal{T}_M \\ {\scriptstyle \text{bisect-n-simplex}}\downarrow & & \downarrow{\scriptstyle \text{bisect-tet}} \\ \mathcal{T}_T \times \mathcal{T}_T & \xrightarrow{\mathcal{F}\times\mathcal{F}} & \mathcal{T}_F \times \mathcal{T}_F \end{array}$$

Proof. Suppose $\mathfrak{t} = ((x_0, x_1, x_2, x_3), 3) \in \mathcal{T}_T$, and $\left\{\mathfrak{t}^{(1)}, \mathfrak{t}^{(2)}\right\} \in \mathcal{T}_T \times \mathcal{T}_T$ is produced by bisect-n-simplex$(\mathfrak{t})$. Then $\mathfrak{t}^{(1)} = ((x_0, x_1, x_2, z), 2)$ with $z = (x_0 + x_3)/2$, and consequently $\mathcal{F}\left(\mathfrak{t}^{(1)}\right)$ yields the marked tetrahedron

$$(\{x_0, x_1, x_2, z\}, \{x_0x_2\}, (x_0x_1, x_1x_2), 0).$$

On the other hand, $\mathcal{F}(\mathfrak{t}) = (\{x_0, x_1, x_2, x_3\}, \{x_0x_3\}, (x_0x_2, x_1x_3), 0)$ which is NP-AA, and one of the marked tetrahedra produced by applying bisect-tet to this is $\mathcal{F}\left(\mathfrak{t}^{(1)}\right)$. A similar verification is easily carried out for the other cases. □

With some obvious abuse of notation, we will sometimes refer to the marked tetrahedron $(\mathcal{N}(\tau), r_\tau, m_\tau, f_\tau)$ as the tetrahedron τ. Define *application of bisect-tet m times* to a tetrahedron τ to mean that bisect-tet is applied once to τ, once to both its children, and so on, until we have m such applications producing 2^m tetrahedra. Any tetrahedron obtained via the repeated application of bisect-tet to τ will be called a *descendent* of τ. We assume a similar definition for the repeated application of bisect-n-simplex and the n-simplices produced. Proposition 3.2.2 implies that if

bisect-n-simplex is applied m times to a tagged-3-simplex, the images of the 2^m descendents under the mapping $\mathcal{F}$ are exactly the descendents obtained by applying bisect-tet m times to the image under $\mathcal{F}$ of the parent. Note that descendents are elements of $\mathcal{T}_T$ or $\mathcal{T}_M$. In particular, let $\mathfrak{t} \in \mathcal{T}_T$ be a tagged-3-simplex. Associate to each descendent of $\mathfrak{t}$ obtained via a repeated application of bisect-n-simplex a geometric 3-simplex having the same vertex set as that of the descendent and make a similar association for the descendents of an arbitrary element of $\mathcal{T}_M$ obtained via repeated applications of bisect-tet. Proposition 3.2.2 then implies that the geometric shapes produced by the two algorithms are equivalent in the following sense:

Corollary 3.2.3 *The geometric 3-simplices obtained by applying bisect-n-simplex m times to $\mathfrak{t} \in \mathcal{T}_T$ are exactly the same as those obtained by applying bisect-tet m times to $\mathcal{F}(\mathfrak{t})$.*

We have provided a prescription (function $\mathcal{F}$) for constructing a marked tetrahedron from a tagged-3-simplex. We next present a method for constructing a tagged-3-simplex from a marked tetrahedron. Let $(\{v_0, v_1, v_2, v_3\}, r_\tau, m_\tau, f_\tau) \in \mathcal{T}_A$ be a marked tetrahedron τ and let $((x_0, x_1, x_2, x_3), d) \in \mathcal{T}_T$ be a corresponding tagged-3-simplex.

- If τ is NP-AA with some choice for its refinement and marked nonrefinement edges (say $r_\tau = v_0v_1, m_\tau = (v_0v_2, v_1v_3)$): Set $d = 3$ and choose $x_0 = v_0$ and $x_d = v_1$ (the end points of the refinement edge). Set $x_1 = v_3$ (the vertex determining the marked nonrefinement edge with one vertex as x_d is x_1). Obviously, $x_2 = v_2$. Note that the choice of x_0 and x_d can be reversed. However, given one of the two possible choices for x_0 and x_d the other x_i are specified uniquely based on the marked nonrefinement edges. In particular, the two tagged-3-simplices corresponding to the given marked tetrahedron are $((v_0, v_3, v_2, v_1), 3)$ and $((v_1, v_2, v_3, v_0), 3)$.
- If τ is P-UF with some choice for its refinement and marked nonrefinement

edges (say $r_\tau = v_0v_1, m_\tau = (v_0v_2, v_1v_2)$): Set $d = 2$ and choose $x_0 = v_0$ and $x_d = v_1$ (the end points of the refinement edge). Set $x_1 = v_2$ (the vertex which determines the two marked nonrefinement edges is x_1 and $x_0x_1x_2$ is the plane containing all the marked edges). Note that the choices for x_0 and x_d can be reversed but the choice of x_1 (and hence x_3) is fixed. In particular, the two tagged-3-simplices corresponding to the given marked tetrahedron are $((v_0, v_2, v_1, v_3), 2)$ and $((v_1, v_2, v_0, v_3), 2)$.

- If τ is P-F with some choice for its refinement and marked nonrefinement edges (say $r_\tau = v_0v_1, m_\tau = (v_0v_2, v_1v_2)$): Set $d = 1$ and choose $x_0 = v_0$ and $x_d = v_1$ (the end points of the refinement edge). Set $x_3 = v_2$ (the vertex which determines the two marked nonrefinement edges is x_3 and $x_0x_1x_3$ is the plane containing all the marked edges). Note that the choices for x_0 and x_d can be reversed but the choice of x_3 (and hence x_2) is fixed. In particular, the two tagged-3-simplices corresponding to the given marked tetrahedron are $((v_0, v_1, v_3, v_2), 1)$ and $((v_1, v_0, v_3, v_2), 1)$.

In all the cases listed above, the image under $\mathcal{F}$ of either of the tagged-3-simplices is the given marked tetrahedron. Moreover, Proposition 3.2.2 says that the images under $\mathcal{F}$ of the 2^m descendents of either tagged-3-simplex via m applications of bisect-n-simplex are the 2^m descendents of the original marked tetrahedron obtained via m applications of bisect-tet.

Proposition 3.2.4 *The mapping $\mathcal{F}$ takes $\mathcal{T}_T$ onto $\mathcal{T}_A$ and is 2 to 1.*

Proof. The definition of $\mathcal{F}$ and the method for constructing tagged-3-simplices from marked tetrahedra in $\mathcal{T}_A$ given above imply that $\mathcal{F}$ is onto. Moreover, we

directly verify using the definition of $\mathcal{F}$ that

$$\mathcal{F}\left((x_0, x_1, x_2, x_3), 3\right) = \mathcal{F}\left((x_3, x_2, x_1, x_0), 3\right),$$
$$\mathcal{F}\left((x_0, x_1, x_2, x_3), 2\right) = \mathcal{F}\left((x_2, x_1, x_0, x_3), 2\right),$$
$$\mathcal{F}\left((x_0, x_1, x_2, x_3), 1\right) = \mathcal{F}\left((x_1, x_0, x_2, x_3), 1\right),$$

which implies that $\mathcal{F}$ is at least 2 to 1. Another direct verification gives

$$\mathcal{F}\left((x_0, x_1, x_2, x_3), d\right) = \mathcal{F}\left((x'_0, x'_1, x'_2, x'_3), d'\right)$$

if and only if $d = d'$ and either $x_i = x'_i, 0 \leq i \leq 3$ or they are related by one of the cases given above. Thus, $\mathcal{F}$ is 2 to 1. □

We have thus shown that the algorithms bisect-tet and bisect-n-simplex are equivalent in the sense of Proposition 3.2.2.

Remark 3.2.5 For any given tagged-3-simplex, the case $d = 1$ corresponds to the marked tetrahedron being P-F, $d = 2$ to P-UF, and $d = 3$ to NP-AA. Thus, bisect-n-simplex admits only tagged-3-simplices corresponding to marked tetrahedra in $\mathcal{T}_A$ as input. Even though the types of tetrahedra to which bisect-n-simplex is applicable are restricted, it will not pose a problem in our analysis for similarity classes since tetrahedra of type NP-OO and NP-AO are never produced as children via bisect-tet, and we will treat these cases separately.

The finiteness of the number of similarity classes of tetrahedra produced by bisect-tet follows from the fact that the number of similarity classes of n-simplices produced by the repeated application of bisect-n-simplex is bounded by $nn!2^{n-2}$. We now state a technical result from [26] that is required in what follows.

Lemma 3.2.6 *Let $\mathfrak{t}'$ be a descendent of the tagged-n-simplex*

$$\mathfrak{t} = \left((0, e_1, e_1 + e_2, \ldots, e_1 + e_2 + \ldots + e_n), d\right)$$

obtained via k applications of bisect-n-simplex, where $\mathcal{B} = \{e_1, e_2, \ldots, e_n\}$ is an arbitrary basis of $\mathbb{R}^n$. Define integers $r \in \{0, \ldots, n-1\}$ and $q \geq 0$ by $n - d + k = r + qn$, let $(x_0, x_1, \ldots, x_n)$ be the ordered vertices of $\mathfrak{t}'$, and define $y_i = x_i - x_0$ for $i = 1, 2, \ldots, n$. Then, there exist two linear mappings π and R from $\mathbb{R}^n$ to $\mathbb{R}^n$ such that $\{\pi(e_1), \pi(e_2), \ldots, \pi(e_n)\}$ is a permutation of the basis vectors of $\mathcal{B}$, and $R(e_i) = \pm e_i$, $1 \leq i \leq n$ (R is a reflection relative to the basis $\mathcal{B}$), with

$$y_i = \left(\frac{1}{2}\right)^q \alpha_i R\pi \sum_{j=1}^{i} e_j \qquad 1 \leq i \leq n,$$

where

$$\alpha_i = \begin{cases} 1 & 1 \leq i \leq n-r, \\ 1/2 & n-r+1 \leq i \leq n. \end{cases}$$

As there are $n!$ possibilities for the permutations π, 2^n possibilities for the reflections R, and exactly n different vectors α_i, a first estimate for an upper bound on number of similarity classes produced by the repeated application of bisect-n-simplex is $nn!2^n$. Recalling that the reflections R and $-R$ give n-simplices in the same similarity class, this estimate improves to $nn!2^{n-1}$. Obviously, this bound is not sharp. For the simple case $n = 2$, the above estimate gives 8 similarity classes while only 4 types of triangles arise in the repeated bisection of an arbitrary triangle. We now give a proof due to Arnold and Pouly [1] showing that the number of similar classes of n-simplices produced is bounded by $nn!2^{n-2}$. This bound is sharp for $n = 2$ since it gives 4 similarity classes in this case. They also show by considering a particular tetrahedron that this bound is sharp for $n = 3$. The idea of the proof is to show that each n-simplex produced is similar to another n-simplex via a translation. For $n = 3$, this estimate improves the upper bound of 168 shown by Liu and Joe [24] for an equivalent algorithm based on longest edge bisection to 36.

Theorem 3.2.7 *The number of similarity classes of n-simplices produced by the repeated application of bisect-n-simplex is bounded by $nn!2^{n-2}$.*

Proof. We will use the notations introduced in Lemma 3.2.6. Observe that q does not play a role in the count for the number of similarity classes of tetrahedra and we will assume $q = 0$ without any loss of generality. Define the mappings $\hat{\pi} : \mathbb{R}^n \to \mathbb{R}^n$ and $\hat{R} : \mathbb{R}^n \to \mathbb{R}^n$ by

$$\hat{\pi}(e_j) = \begin{cases} \pi(e_{n-r+1-j}) & 1 \le j \le n-r, \\ \pi(e_j) & n-r+1 \le j \le n, \end{cases} \tag{15}$$

$$\hat{R}\pi(e_j) = \begin{cases} -R\pi(e_j) & 1 \le j \le n-r, \\ R\pi(e_j) & n-r+1 \le j \le n, \end{cases} \tag{16}$$

for $j \in \{1, 2, \ldots, n\}$. Observe that $\hat{\pi}$ is a permutation and $\hat{R}$ a reflection relative to $\mathcal{B}$. Note that the ordered set $(0, y_1, y_2, \ldots, y_n)$ represents the vertices of the tagged-n-simplex $\mathfrak{t}'$, and denote the ordered set of vertices of another tagged-n-simplex $\hat{\mathfrak{t}}$ by $(0, \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n)$ with $\hat{y}_i = \alpha_i \hat{R}\hat{\pi} \sum_{j=1}^{i} e_i$ for $1 \le i \le n$. Let $\Phi : \mathbb{R}^n \to \mathbb{R}^n$ be the translation defined by

$$\Phi(x) = x - R\pi \sum_{j=1}^{n-r} e_j = x - y_{n-r}. \tag{17}$$

Then $\Phi(0) = -R\pi \sum_{j=1}^{n-r} e_j$ and

$$\Phi(y_i) = \begin{cases} -R\pi \sum_{j=i+1}^{n-r} e_j & 1 \le i < n-r, \\ 0 & i = n-r, \\ -\frac{1}{2}R\pi \sum_{j=1}^{n-r} e_j + \frac{1}{2}R\pi \sum_{j=n-r+1}^{i} e_j & n-r+1 \le i \le n. \end{cases}$$

Using (15), (16), and the definition of $\hat{y}_i$ we get $\Phi(0) = \hat{y}_{n-r}$ and

$$\Phi(y_i) = \begin{cases} \hat{y}_{n-r-i} & 1 \le i < n-r, \\ 0 & i = n-r, \\ \hat{y}_i & n-r+1 \le i \le n. \end{cases}$$

Thus, t' is related to $\hat{t}$ via the translation Φ. The theorem follows from Lemma 3.2.6 and the discussion following it. □

It follows that the number of similarity classes of tetrahedra produced by the repeated application of bisect-tet to arbitrary tetrahedra of type P-F, P-UF, or NP-AA is 36. Since one application of bisect-tet to tetrahedra of the type NP-OO and NP-AO produce children of type P-UF, the repeated bisection of a NP-OO or NP-AO tetrahedron will produce a maximum of 72 similar classes of tetrahedra. This ensures that in an adaptive mesh refinement procedure involving the repeated application of bisect-tet, the meshes produced will have finite shape constants. Such a procedure is presented in the next section, and it is shown that the locally adapted meshes produced are also conforming.

3.3 A locally adaptive mesh refinement procedure

This section describes an algorithm that produces a sequence of locally adapted, nested, conforming meshes. Algorithm local-refine starts with a conforming mesh $\mathcal{T}$ and produces a conforming mesh $\mathcal{T}'$ by first bisecting a selected subset of $\mathcal{T}$ and then performing further closure bisections to ensure a conforming mesh. Algorithm local-refine will be used to create a sequence of nested, locally adapted meshes starting from a coarse conforming mesh $\mathcal{T}_0$. Since local-refine uses bisect-tet, all tetrahedra in the coarse mesh have to be marked. A mesh is said to be *marked* if each tetrahedron in it is marked. A marked, conforming mesh with the property that each face has a unique marked edge will be called a *conformingly-marked mesh.*

Remark 3.3.1 Recall that if a face is shared by two marked tetrahedra in a conforming mesh, it is possible for the face to have distinct marked edges. By definition, a conformingly-marked mesh excludes this possibility.

Given an arbitrary conforming mesh $\mathcal{T}_0$, the following procedure yields a conformingly-marked mesh $\mathcal{T}_0$ (by an obvious abuse of notation we denote both by $\mathcal{T}_0$): Choose the longest edge of each tetrahedron in $\mathcal{T}_0$ as its refinement edge and

mark the longest edge of the nonrefinement faces (small hypothetical perturbation of the coordinates of vertices are performed to break ties). If we further assume that the flag is unset for all tetrahedra in $\mathcal{T}_0$, the procedure gives a conformingly-marked mesh with no flagged tetrahedra. The assumption that the coarse mesh has no flagged tetrahedra will be key to most of the analysis in this section. Let $\mathcal{E}(\tau)$ denote the edge set of a tetrahedron τ, and set $\mathcal{E}(\mathcal{T}) = \bigcup_{\tau \in \mathcal{T}} \mathcal{E}(\tau)$ for an arbitrary mesh $\mathcal{T}$. A tetrahedron $\tau \in \mathcal{T}$ is said to have a *hanging node* on an edge $e \in \mathcal{E}(\tau)$ if both e and its children belong to $\mathcal{E}(\mathcal{T})$ (where the child of an edge is one of the two line segments created by bisecting it). Note that a mesh is nonconforming if and only if it has at least one tetrahedron with a hanging node.

Given a mesh $\mathcal{T}$ and a subset S, local-refine can be broken up into two steps:

(1) Bisect all tetrahedra in S using bisect-tet.

(2) Repeatedly bisect all tetrahedra with hanging nodes in the resulting mesh until a conforming mesh $\mathcal{T}'$ is obtained.

A detailed description of this algorithm requires some terminology and many intermediate algorithms. These algorithms and their properties will be presented here with the ultimate goal of showing that the second step of local-refine has only a finite number of repetitions in the application that we consider.

For a marked tetrahedron τ set $\mathcal{E}_0(\tau) = \mathcal{E}(\tau)$, and denote by $\mathcal{E}_1(\tau)$ the set of 25 line segments defined as follows:

- The 4 line segments obtained by joining the midpoints of each marked edge of τ to the opposite vertex with respect to the face containing the marked edge.
- The 8 line segments obtained by joining the midpoints of each marked edge to those of the other two edges of the face (or faces) of τ for which the edge is marked.
- The line segment obtained by joining the midpoint of the refinement edge of τ to that of the opposite edge with respect to τ.

- The 12 line segments that are the children of the 6 members of $\mathcal{E}(\tau)$.

Let NP-$**$ denote any nonplanar tetrahedron. For any descendent τ' of a tetrahedron τ, define the *bisection level* of τ' to be the number of times bisect-tet is applied to τ to obtain τ'. Let BL (τ') denote the bisection level of τ'. The next result about the edges of descendents created by the repeated application of bisect-tet to an arbitrary tetrahedron is obtained by direct calculation. It can be verified with reference to Figure 5.

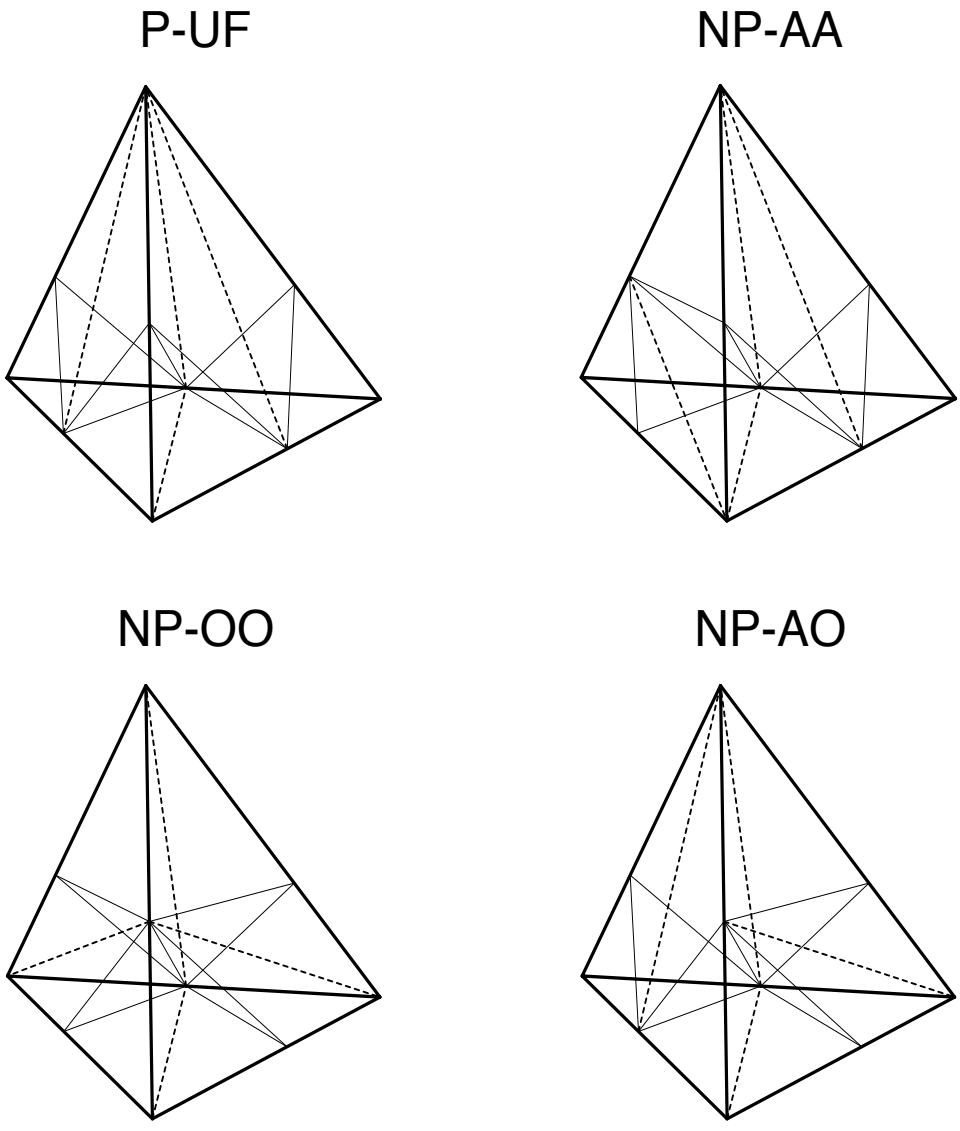

Figure 5. Three applications of bisect-tet to the different tetrahedra.

Lemma 3.3.2 *Let τ be a marked P-UF (NP-**) tetrahedron and τ' a descendent of τ. If $\mathrm{BL}(\tau') \leq 3$ then τ' has edges in $\mathcal{E}_0(\tau) \cup \mathcal{E}_1(\tau)$, and if $\mathrm{BL}(\tau') = 3$, τ' is P-UF (NP-AA) with edges in $\mathcal{E}_1(\tau)$. Moreover, the mesh $\mathcal{T} = \{\tau' | \mathrm{BL}(\tau') = 3\}$ is a conformingly-marked mesh with $\mathcal{E}(\mathcal{T}) = \mathcal{E}_1(\tau)$.*

For $k \geq 2$, recursively define $\mathcal{E}_k(\tau)$ by

$$\mathcal{E}_k(\tau) = \{\cup \mathcal{E}_1(\tau') | \tau' \text{ is a descendent of } \tau \text{ with } \mathrm{BL}(\tau') = 3k-3\}.$$

The next corollary follows from Lemma 3.3.2 and the observation that in 3 applications of bisect-tet to an unflagged tetrahedron τ each face of τ gets subdivided into 4 faces by a line segment joining the midpoint of the marked edge to the opposite vertex, and those joining the midpoint of the marked edge to the midpoints of the nonmarked edges (see Figure 5).

Corollary 3.3.3 *Let τ be a marked P-UF (NP-**) tetrahedron and τ' a descendent of τ. If $\mathrm{BL}(\tau') \leq 3k$ then τ' has edges in $\mathcal{E}_0(\tau) \cup \mathcal{E}_1(\tau) \cdots \cup \mathcal{E}_k(\tau)$, and if $\mathrm{BL}(\tau') = 3k$, τ' is P-UF (NP-AA) with edges in $\mathcal{E}_k(\tau)$. Moreover, the mesh $\mathcal{T} = \{\tau' | \mathrm{BL}(\tau') = 3k\}$ is a conformingly-marked mesh with $\mathcal{E}(\mathcal{T}) = \mathcal{E}_k(\tau)$.*

Let $\mathcal{T}$ be a marked mesh and $S \subseteq \mathcal{T}$. Algorithm *bisect-tets* creates a (possibly) nonconforming mesh $\mathcal{T}'$ from $\mathcal{T}$ by bisecting each tetrahedron in S using bisect-tet.

Algorithm 3.3.4

$\mathcal{T}' \longleftarrow \mathrm{bisect\text{-}tets}(\mathcal{T}, S)$

Input: Marked mesh $\mathcal{T}$ and $S \subseteq \mathcal{T}$.

Output: Marked mesh $\mathcal{T}'$.

(1) Perform $\mathrm{bisect\text{-}tet}(\tau)$ for all $\tau \in S$ to create $\mathcal{T}'$.

Further, denote by *bisect-all* the algorithm that bisects all tetrahedra in a marked mesh: $\mathrm{bisect\text{-}all}(\mathcal{T}) = \mathrm{bisect\text{-}tets}(\mathcal{T}, \mathcal{T})$.

We will now use bisect-all to create a sequence of nested, uniformly refined, conforming meshes beginning with a coarse conforming mesh. This sequence of

meshes has some special properties which will be used later. Let Q_0 be an arbitrary coarse, conformingly-marked mesh with no flagged tetrahedra, and set

$$Q_k = \text{bisect-all}\,(Q_{k-1}) \text{ for } k \geq 1.$$

Set the bisection level of all tetrahedra in Q_0 to be zero, and let

$$\mathcal{E}_k\,(Q_0) = \{\cup \mathcal{E}_k\,(\tau) \,|\, \text{for all } \tau \in Q_0\}\,.$$

The next proposition is a direct consequence of Lemma 3.3.2, Corollary 3.3.3, and the definitions.

Proposition 3.3.5 *The meshes $Q_{3k}, k \geq 1$, are conformingly-marked, have no flagged tetrahedra, consist of tetrahedra with bisection level $3k$, and have $\mathcal{E}_k\,(Q_0)$ as their edge set.*

To clearly classify the edges that appear in the meshes Q_k for arbitrary k, define the *generation* of an edge as $\text{gen}\,(e) = k$ for all $e \in \mathcal{E}_k\,(Q_0)\,(= \mathcal{E}\,(Q_{3k}))$. The generations of all possible edges that appear in the repeated application of bisect-all to Q_0 are given in Table 1 together with the types and bisection levels of tetrahedra that are created.

The next lemma is a consequence of the definition of the generation of an edge and follows from Table 1.

Lemma 3.3.6 *Let τ be a descendent of a tetrahedron in Q_0. If $\text{BL}\,(\tau) < 3(k+1)$, then $\text{gen}\,(e) \leq k+1$ for all $e \in \mathcal{E}\,(\tau)$, and if $\text{BL}\,(\tau) = 3(k+1)$, then τ is unflagged and $\text{gen}\,(e) = k+1$ for all $e \in \mathcal{E}\,(\tau)$.*

Since Q_0 has no flagged tetrahedra, all P-F tetrahedra have bisection level $3k+1$ or $3k+2, k \geq 0$. Moreover, a P-F tetrahedron with bisection level $3k+1$ is the child of a tetrahedron with bisection level $3k$, and a P-F with bisection level $3k+2$ is the parent of a tetrahedron with bisection level $3(k+1)$.

We now present algorithm *refine-to-conformity* which takes a marked nonconforming mesh as input and repeatedly applies bisect-tet to tetrahedra with hanging

Table 1. Edge generation and tetrahedra type.

Bisection Level	Types of Tetrahedra		Generation of Edges	
0	P-UF	NP-**	0	⋆
1	P-F	P-UF	0,1	
2	NP-AA	P-F	0,1	
3	P-UF	NP-AA	1	⋆
4	P-F	P-UF	1,2	
5	NP-AA	P-F	1,2	
6	P-UF	NP-AA	2	⋆
...	...	...	...	
...	...	...	...	
$3k$	P-UF	NP-AA	k	⋆
$3k+1$	P-F	P-UF	$k, k+1$	
$3k+2$	NP-AA	P-F	$k, k+1$	
$3(k+1)$	P-UF	NP-AA	$k+1$	⋆

nodes until a conforming mesh is obtained. This algorithm may not terminate for arbitrary nonconforming meshes, but we will prove that it yields a conforming mesh in a finite number of steps in the particular application that we consider.

Algorithm 3.3.7

$\mathcal{T}' \longleftarrow$ refine-to-conformity $(\mathcal{T})$

Input: Marked mesh $\mathcal{T}$.

Output: Conformingly-marked mesh $\mathcal{T}'$.

(1) Set $S = \{\tau \in \mathcal{T} \mid \tau \text{ has a hanging node}\}$.

(2) If $S \neq \emptyset$,

$\bar{\mathcal{T}} \longleftarrow$ bisect-tets $(\mathcal{T}, S)$

$\mathcal{T}' \longleftarrow$ refine-to-conformity $(\bar{\mathcal{T}})$

(3) Else, $\mathcal{T}' = \mathcal{T}$.

A mesh $\mathcal{T}'$ is said to be a *refinement of* $\mathcal{T}$ if it is any one of the meshes $\widetilde{\mathcal{T}}_k$ with

$\widetilde{\mathcal{T}}_k \longleftarrow$ bisect-tets $\left(\widetilde{\mathcal{T}}_{k-1}, S_{k-1}\right)$ $(k \geq 1)$, with $\widetilde{\mathcal{T}}_0 = \mathcal{T}$ and $S_k \subseteq \widetilde{\mathcal{T}}_k$. The notation $\mathcal{T}' \sqsupset \mathcal{T}$ indicates that $\mathcal{T}'$ is a refinement of $\mathcal{T}$. The next proposition shows that in a particular application of refine-to-conformity to meshes that are refinements of a conformingly-marked mesh with no flagged tetrahedra, the algorithm terminates in a finite number of steps.

Proposition 3.3.8 *Let $\mathcal{T} \sqsupset Q_0$ be a marked mesh such that* $\mathrm{BL}(\tau) \leq 3k$ *for all $\tau \in \mathcal{T}$ (note that if* $\mathrm{BL}(\tau) = 3k$, *τ is unflagged). Then,* refine-to-conformity($\mathcal{T}$) *terminates and* $\mathrm{BL}(\tau) \leq 3k+3$ *for all $\tau \in \mathcal{T}'$.*

Proof. The marked mesh $\bar{\mathcal{T}}$ produced in the first pass through Step 2 has the property that $\mathrm{BL}(\tau) \leq 3k+1$ for all $\tau \in \bar{\mathcal{T}}$. Thus, by Lemma 3.3.6 $\mathrm{gen}(e) \leq k+1$ for all $e \in \mathcal{E}(\bar{\mathcal{T}})$.

Let $\widetilde{\mathcal{T}} \sqsupset \mathcal{T}$ be such that $\mathrm{gen}(e) \leq k+1$ for all $e \in \mathcal{E}(\widetilde{\mathcal{T}})$ (equivalently, $\mathrm{BL}(\tau) \leq 3(k+1)$ for all $\tau \in \widetilde{\mathcal{T}}$). Let $\tau \in \widetilde{\mathcal{T}}$ have a hanging node on $e \in \mathcal{E}(\tau)$. By definition, both e and its children (which have generation strictly greater than e) must be in $\mathcal{E}(\widetilde{\mathcal{T}})$. Thus, $\mathrm{gen}(e) < k+1$ (equivalently, $\mathrm{BL}(\tau) < 3(k+1)$).

Thus, all tetrahedra with hanging nodes in the meshes $\bar{\mathcal{T}}$ produced within refine-to-conformity have bisection level $< 3(k+1)$. We have shown that refine-to-conformity terminates, $\mathrm{BL}(\tau) \leq 3(k+1)$ for all $\tau \in \mathcal{T}'$, and $\mathcal{T}'$ is conforming. The proposition follows if we recall that faces in a mesh are subdivided according to their unique markings. □

Next, we introduce algorithm *local-refine* which yields a conforming mesh by first bisecting a given subset of tetrahedra in a conformingly-marked mesh and then resolving any hanging nodes in the resulting mesh by refine-to-conformity.

Algorithm 3.3.9

$\mathcal{T}' \longleftarrow$ local-refine$(\mathcal{T}, S)$

Input: Conformingly-marked mesh $\mathcal{T}$ and $S \subseteq \mathcal{T}$.

Output: Conformingly-marked mesh $\mathcal{T}'$.

(1) $\bar{\mathcal{T}} \longleftarrow$ bisect-tets $(\mathcal{T}, S)$.

(2) $\mathcal{T}' \longleftarrow$ refine-to-conformity $(\bar{\mathcal{T}})$.

Remark 3.3.10 Note that the subset S may be chosen so as to refine a particular region in the mesh $\mathcal{T}$. Note also that if we choose a marked nonconforming $\mathcal{T}$ and the subset S of tetrahedra with hanging nodes in $\mathcal{T}$ as input to local-refine, refine-to-conformity and local-refine will produce identical outputs.

The next result is a direct consequence of Proposition 3.3.8.

Theorem 3.3.11 *Let $\mathcal{T}_0 = Q_0$ be a conformingly-marked unflagged mesh. For $k \geq 0$, choose $S_k \subseteq \mathcal{T}_k$ arbitrarily and set $\mathcal{T}_{k+1} =$ local-refine $(\mathcal{T}_k, S_k)$. Each call to refine-to-conformity from within local-refine terminates. Moreover, $Q_{3k} \sqsupset \mathcal{T}_k$.*

3.4 Example of a locally adaptive mesh

In this section we show by considering examples that local-refine produces adapted meshes that are quite local. The coarse mesh $\mathcal{T}_0$ is the mesh of six tetrahedra for the unit cube $[0,1]^3$ shown in Figure 6. Note that all six tetrahedra share the edge that is the diagonal of the cube. This is the refinement edge (longest edge) for all $\tau \in \mathcal{T}_0$. The marked edges for tetrahedra in the coarse mesh are determined by the longest edges of its nonrefinement faces (these are the diagonals of the faces of $[0,1]^3$). This choice of refinement and marked edges implies that all tetrahedra in $\mathcal{T}_0$ are NP-AA. This example has been considered by Maubach [26]. Observe that all six tetrahedra in $\mathcal{T}_0$ are similar since they are related to one another by coordinate permutations. In particular, the vertex sets of of the six tetrahedra in $\mathcal{T}_0$ are

$$\{(0,0,0),(1,0,0),(1,1,0),(1,1,1)\}\,, \quad \{(0,0,0),(1,0,0),(1,0,1),(1,1,1)\}\,,$$

$$\{(0,0,0),(0,1,0),(1,1,0),(1,1,1)\}\,, \quad \{(0,0,0),(0,1,0),(0,1,1),(1,1,1)\}\,,$$

$$\{(0,0,0),(0,0,1),(0,1,1),(1,1,1)\}\,, \quad \{(0,0,0),(0,0,1),(1,0,1),(1,1,1)\}\,.$$

The criterion for adaptive local refinement is that any tetrahedron intersecting the

hemisphere

$$\mathcal{H} = \left\{ (x,y,z) \in \mathbb{R}^3 \,|\, (x-0.5)^2 + (y-0.5)^2 + (z-0.5)^2 = \frac{1}{16} \text{ with } x \geq 0.5 \right\}$$

is bisected. In other words, for any mesh $\mathcal{T}$ the set $S \subseteq \mathcal{T}$ in local-refine is $S = \{\tau \in \mathcal{T} \,|\, \tau \cap \mathcal{H} \neq \emptyset\}$. Maubach [26] has shown that for this special choice of coarse mesh tetrahedra the number of similarity classes of tetrahedra produced by repeated bisection is 3. This is verified by our computations using the tetrahedron shape measure introduced by Liu and Joe [24], as well as a more traditional measure involving the ratio of the circumscribed to inscribed sphere for each tetrahedron.

Let $\mathcal{T}_{j+1} = \text{local-refine}\,(\mathcal{T}_j, S)$ be the mesh obtained from $\mathcal{T}_j$ by taking the the subset S of $\mathcal{T}_j$ as defined above. Figures showing various views of $\mathcal{T}_{16}$ are presented in the next two pages. In all figures involving $\mathcal{T}_{16}$, it is referred to as the hemisphere-adapted mesh. The mesh $\mathcal{T}_{16}$ has $25,448$ tetrahedra and $4,864$ nodes. Figure 7 shows a cloud of points representing its nodes, while Figure 8 and Figure 9 show cuts through the mesh along $x = \frac{1}{2}$ and $y = \frac{1}{2}$ respectively. It is clear that local-refine locally adapts the mesh very well in this case.

We will give many more examples of locally adapted meshes produced by local-refine when we consider numerical examples to demonstrate the capabilities of AMG3DP1 at the end of Chapter 5.

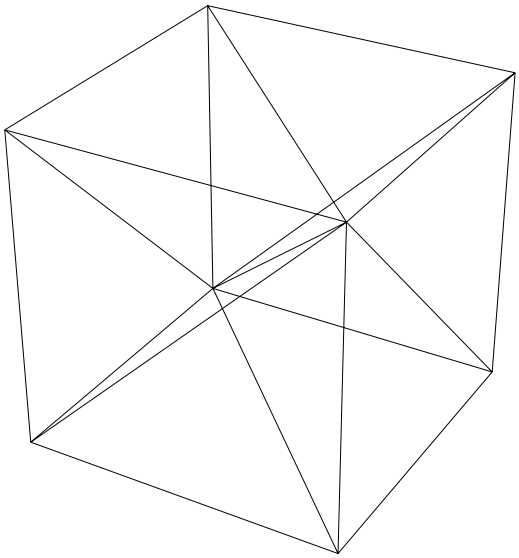

Figure 6. Coarse mesh for $[0,1]^3$: 6 tetrahedra and 8 nodes.

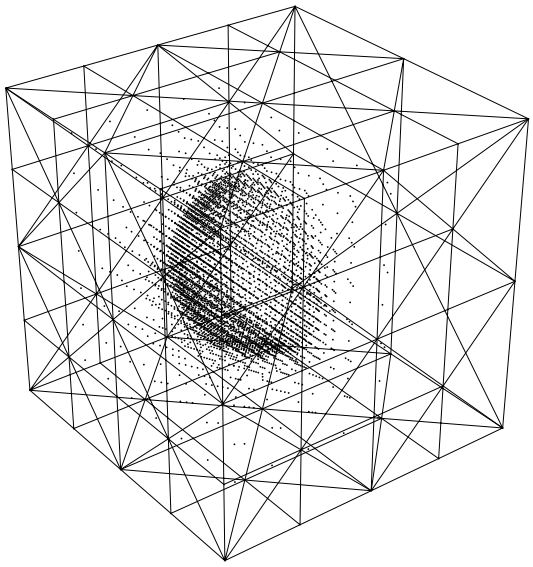

Figure 7. Hemisphere-adapted mesh: the nodes for the adapted mesh.

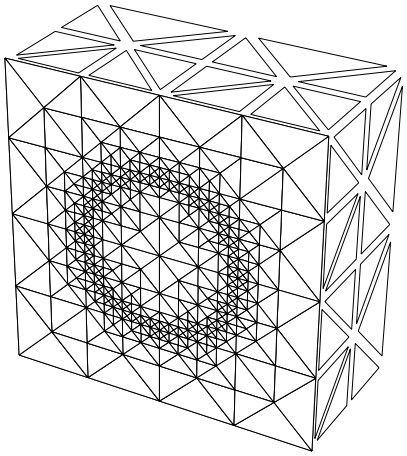

Figure 8. Hemisphere-adapted mesh: the $x = \frac{1}{2}$ plane for the adapted mesh.

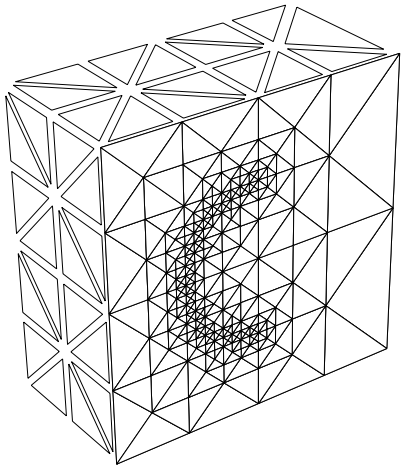

Figure 9. Hemisphere-adapted mesh: the $y = \frac{1}{2}$ plane for the adapted mesh.

Chapter 4

Some aspects of the multilevel solver

Multilevel solvers have established themselves as one of the most efficient methods for solving linear systems arising from the discretization of partial differential equations. In many cases they can be shown to require only $O(N)$ operations to compute a discrete solution involving N unknowns. In this chapter we present a multilevel algorithm suitable for solving linear systems that arise from the finite element discretization of the model boundary value problem of Chapter 2. There has been a lot of work on the convergence theory of multilevel methods for systems of equations arising from finite element discretizations of boundary value problems (for a detailed discussion and references, see for example Bramble [13]). The method presented in this chapter is a V-cycle applied to the sequence of refined meshes that are obtained by the adaptive mesh refinement procedure of Chapter 3. The convergence for this method is shown in Bramble and Pasciak [14].

The V-cycle algorithm has two main components: smoothing on a finer mesh and error correction on a coarser mesh. The smoothing step damps out the oscillatory parts of the error on the finer mesh and the remaining part of the error is transferred to the coarser mesh and corrected there. For an efficient implementation of the algorithm, we introduce 2-level (hierarchical) bases in addition to the usual nodal bases that appear in the implementation of the finite element method. Basis changes between the two will be used in transferring the error from the fine to coarse mesh and the correction from the coarse to the fine mesh (however, the method is not the "hierarchical basis multigrid method" of Bank, Dupont and Yserentant [6]).

In Section 1, we begin by presenting the V-cycle in operator form (based on

the solution of some operator equations in appropriate spaces). Nodal bases for the finite element spaces are then used to derive an equivalent form involving vectors and matrices. Finally, some algebraic manipulations yield a form that is symmetric. The V-cycle algorithm uses a sequence of meshes; the transfer of the error from a finer to a coarser mesh and of the correction from a coarser to a finer are two important steps in the V-cycle. In our implementation, these are achieved via basis changes. Section 2 describes these basis changes (nodal to 2-level and vice versa) and their use to perform the transfer between fine and coarse meshes. Section 3 describes the smoothing step. Moreover, we present a multilevel solution algorithm for the boundary value problem of Chapter 2. This uses the transfer and smoothing algorithms introduced in Sections 2 and 3 and is derived from the symmetric algorithm at the end of Section 1. This is the form of the algorithm that we use in our implementation in AMG3DP1.

4.1 The V-cycle for finite elements

Let $\{\mathcal{T}_k\}_{k=0}^{M}$ be a nested sequence of conforming meshes and denote by

$$\{X_k\}_{k=0}^{M}, \{V_k\}_{k=0}^{M}, \text{ and } \left\{V_k^D\right\}_{k=0}^{M}$$

the spaces analogous to $X(\mathcal{T}), V_0(\mathcal{T})$, and $V_D(\mathcal{T})$ introduced in Chapter 2. The discrete problem on $\mathcal{T}_k$ is:

Find $u_{\mathcal{T}_k} \in V_k^D$ such that $a(u_{\mathcal{T}_k}, v) = l(v)$ for all $v \in V_k$.

Definition 4.1.1 The *nodal basis* for the space X_k will be denoted by $\left\{\phi_i^k\right\}_{i=1}^{N_k}$ and defined as

$$\phi_i^k = \begin{cases} 1 & \text{at node } i \text{ of } \mathcal{T}_k, \\ 0 & \text{at all other nodes of } \mathcal{T}_k, \end{cases}$$

where $N_k := \dim(\mathcal{T}_k)$.

The discrete solution $u_{\mathcal{T}_k}$ can be decomposed as

$$u_{\mathcal{T}_k} = \sum_{i \in \mathcal{N}_D(\mathcal{T}_k)} u_{\mathcal{T}_k}(i)\,\phi_i^k + \sum_{i \in \mathcal{N}_\Omega(\mathcal{T}_k) \cup \mathcal{N}_N(\mathcal{T}_k)} u_{\mathcal{T}_k}(i)\,\phi_i^k =: \hat{z}_k + \hat{u}_k$$

where $\hat{z}_k \in V_k^D$ and $\hat{u}_k \in V_k$. The discrete problem on $\mathcal{T}_k$ is equivalent to:

Find $\hat{u}_k \in V_k$ such that $a(\hat{u}_k, v) = l(v) - a(\hat{z}_k, v)$ for all $v \in V_k$.

Definition 4.1.2 Let $A_k : V_k \to V_k$, and $L_k \in V_k$ be defined by

$$(A_k u, v) = a(u, v) \text{ and } (L_k, v) = l(v) - a(\hat{z}_k, v) \text{ for all } u, v \in V_k.$$

The discrete problem on $\mathcal{T}_k$ is:

(18) Find $\hat{u}_k \in V_k$ such that $A_k \hat{u}_k = L_k$.

Let $Q_k^{k-1} : V_k \to V_{k-1}$ denote the L^2-projection. The adjoint operator $Q_{k-1}^k : V_{k-1} \to V_k$ is then the inclusion. The V-cycle method is an iterative method for the solution of (18), and can be described as:

Start with an initial guess $u_{(k)}^0 \in V_k$, and compute $u_{(k)}^{n+1}$ from $u_{(k)}^n$ by the formula

$$u_{(k)}^{n+1} = u_{(k)}^n - B_k \left(A_k u_{(k)}^n - L_k \right),$$

where the action of the *V-cycle operator* $B_k : V_k \to V_k$ on an arbitrary element of V_k is described by algorithm *operator-*VC_k. This description uses a *smoothing operator* $R_k : V_k \to V_k$, which is assumed to be known and is usually chosen to be easily computable.

Algorithm 4.1.3

$B_k v \longleftarrow$ operator-$VC_k(v)$

Input: $v \in V_k$, R_j for $0 < j \leq k$, and A_j for $0 \leq j \leq k$.

Output: $B_k v \in V_k$.

(0) $B_0 = A_0^{-1}$.

For $k \geq 1$ define B_k in terms of B_{k-1} as:

(1) $x_1 = R_k v$ (pre-smooth).

(2) $x_2 = x_1 - Q_{k-1}^k B_{k-1} Q_k^{k-1} (A_k x_1 - v)$ (coarse level correction).

(3) $B_k v = x_2 - R_k (A_k x_2 - v)$ (post-smooth).

Note that Step 2 of operator-VC_k makes it a recursive algorithm. Also, this step uses the operators A_j, R_j for $j < k$ as it involves the computation of B_j for $j < k$. When the method converges, the sequence $\left\{u^n_{(k)}\right\}$ converges to the solution $\hat{u}_k$ of (18). We next present an equivalent form of the operator-VC_k involving vectors and matrices.

For an arbitrary $v \in V_k$ denote by $\pi_k v$ the *vector of coefficients* in the expansion of v in terms of ϕ^k_i, and by $\mu_k v$ the *vector of inner products* of v with the same basis functions. The two quantities are defined via the equations

$$v = \sum_{i=1}^{N_k} (\pi_k v)_i \, \phi^i_k \text{ and } (\mu_k v)_i = (v, \phi^i_k),$$

where $(\pi_k v)_i$ and $(\mu_k v)_i$ denote the i-th components of $\pi_k v$ and $\mu_k v$ respectively. The two maps $\pi_k : V_k \to \mathbb{R}^{N_k}$ and $\mu_k : V_k \to \mathbb{R}^{N_k}$ are isomorphisms, and

$$(\pi_k v) \cdot (\mu_k w) = (v, w) \text{ for all } v, w \in V_k.$$

Let A_k denote the *stiffness matrix* associated with the basis ϕ^k_i. The symmetric matrix $\mathsf{A}_k \in \mathbb{R}^{N_k \times N_k}$ has entries $a(\phi^k_j, \phi^k_i)$.

Remark 4.1.4 Note that these definitions make sense for any basis of X_k. Later, we will introduce the 2-level basis for the spaces X_k and distinguish clearly between the nodal and 2-level representations of vectors and stiffness matrices associated with nodal and 2-level bases.

The next proposition is a direct consequence of the definitions.

Proposition 4.1.5 *The following diagram commutes.*

$$\begin{array}{ccc} V_k & \xrightarrow{A_k} & V_k \\ {\scriptstyle \pi_k}\downarrow & & \downarrow{\scriptstyle \mu_k} \\ \mathbb{R}^{N_k} & \xrightarrow{\mathsf{A}_k} & \mathbb{R}^{N_k} \end{array}$$

Proof. Let $v \in V_k$ be arbitrary. Then, using the definitions and linearity

$$(\mu_k A_k v)_i = \left(A_k v, \phi^k_i\right) = a\left(v, \phi^k_i\right) = a\left(\sum_{j=1}^{N_k} (\pi_k v)_j \, \phi^k_j, \phi^k_i\right)$$

$$= \sum_{j=1}^{N_k} a\left(\phi_i^k, \phi_j^k\right)\left(\pi_k v\right)_j = \left(\mathsf{A}_k\, \pi_k v\right)_i .$$

□

Thus, pre-multiplication of the vector of coefficients of an arbitrary element in V_k by the stiffness matrix yields the vector of inner products of the image of the element under A_k. This, and similar equivalences stated below will be used in deriving the matrix form of the V-cycle algorithm. Let $\mathsf{C}_k^{k-1} : \mathbb{R}^{N_k} \to \mathbb{R}^{N_{k-1}}$ determined by

$$\phi_i^{k-1} = \sum_{j=1}^{N_k} c_{ij}\, \phi_j^k \text{ for } i = 1, 2, \ldots, N_{k-1},$$

be the *transformation matrix* for the bases of X_k and X_{k-1}. It follows from direct computations that the following diagrams commute.

$$\begin{array}{ccc} V_k & \xrightarrow{Q_k^{k-1}} & V_{k-1} \\ {\scriptstyle \mu_k}\downarrow & & \downarrow{\scriptstyle \mu_{k-1}} \\ \mathbb{R}^{N_k} & \xrightarrow{\mathsf{C}_k^{k-1}} & \mathbb{R}^{N_{k-1}} \end{array} \qquad \begin{array}{ccc} V_{k-1} & \xrightarrow{Q_{k-1}^{k}} & V_k \\ {\scriptstyle \pi_{k-1}}\downarrow & & \downarrow{\scriptstyle \pi_k} \\ \mathbb{R}^{N_{k-1}} & \xrightarrow{\mathsf{C}_{k-1}^{k}} & \mathbb{R}^{N_k} \end{array}$$

The matrix C_{k-1}^k is the transpose of the matrix C_k^{k-1}, and the second diagram is a consequence of the first by taking adjoints and observing that $\pi_k = (\mu_k^*)^{-1}$, where μ_k^* is the adjoint of μ_k. The first diagram states that the vector of inner products of the L^2-projection of any element in V_k is the same as the result of pre-multiplying its vector of inner products by the transformation matrix and the second has a similar interpretation. We shall now define matrices $\mathsf{R}_k\,, \mathsf{B}_k \in \mathbb{R}^{N_k \times N_k}$ corresponding to the smoothing operator R_k and the V-cycle operator B_k such that the following diagrams commute.

$$\begin{array}{ccc} V_k & \xrightarrow{R_k} & V_k \\ {\scriptstyle \mu_k}\downarrow & & \downarrow{\scriptstyle \pi_k} \\ \mathbb{R}^{N_k} & \xrightarrow{\mathsf{R}_k} & \mathbb{R}^{N_k} \end{array} \qquad \begin{array}{ccc} V_k & \xrightarrow{B_k} & V_k \\ {\scriptstyle \mu_k}\downarrow & & \downarrow{\scriptstyle \pi_k} \\ \mathbb{R}^{N_k} & \xrightarrow{\mathsf{B}_k} & \mathbb{R}^{N_k} \end{array}$$

These say that pre-multiplication of the vector of inner products of an arbitrary element of V_k by the smoothing (V-cycle) matrix is the same as the vector of coefficients of the element in V_k obtained by the application of the smoothing (V-cycle) operator. The matrices A_k, C_k^{k-1}, C_{k-1}^k, and R_k are related to the V-cycle matrix B_k according to the following result:

Theorem 4.1.6 *Let $v \in V_k$ be given and let x_1, x_2, and $B_k v$ be defined via operator-VC_k. Set $\boldsymbol{v} = \mu_k v$, $\boldsymbol{x}_1 = \pi_k x_1$, and $\boldsymbol{x}_2 = \pi_k x_2$. Then,*

(1) $\boldsymbol{x}_1 = \mathsf{R}_k \boldsymbol{v}$,

(2) $\boldsymbol{x}_2 = \boldsymbol{x}_1 - \mathsf{C}_{k-1}^k \mathsf{B}_{k-1} \mathsf{C}_k^{k-1} (\mathsf{A}_k \boldsymbol{x}_1 - \boldsymbol{v})$,

(3) $\mathsf{B}_k \boldsymbol{v} = \boldsymbol{x}_2 - \mathsf{R}_k (\mathsf{A}_k \boldsymbol{x}_2 - \boldsymbol{v})$.

Proof. The relations follow from the definitions and commutative diagrams above. For example, Step 3 follows from Step 3 of operator-VC_k as follows:
Step 3 of operator-VC_k is $B_k v = x_2 - R_k (A_k x_2 - v)$. Applying π_k to both sides we get $\pi_k B_k v = \boldsymbol{x}_2 - \pi_k R_k (A_k x_2 - v)$. The properties involving R_k and B_k give $\mathsf{B}_k \mu_k v = \boldsymbol{x}_2 - \mathsf{R}_k \mu_k (A_k x_2 - v)$. Distribution of μ_k over the two terms on the right and the properties of the operator A_k give Step 3 of Theorem 4.1.6.

Steps 1 and 2 follow from their counterparts in operator-VC_k by calculations similar to the one given above. □

In view of Theorem 4.1.6 the matrix-vector analogue of operator-VC_k is:

Given $\boldsymbol{u}_{(k)}^0 \in \mathbb{R}^{N_k}$, the sequence $\left\{\boldsymbol{u}_{(k)}^n\right\}$ is obtained by

$$\boldsymbol{u}_{(k)}^{n+1} = \boldsymbol{u}_{(k)}^n - \mathsf{B}_k \left(\mathsf{A}_k \boldsymbol{u}_{(k)}^n - \mathsf{L}_k\right),$$

where $\mathsf{L}_k = \mu_k L_k$, and the action of B_k on an arbitrary $\boldsymbol{v} \in \mathbb{R}^{N_k}$ is described by algorithm *matrix-VC_k*.

Algorithm 4.1.7

$\mathsf{B}_k \boldsymbol{v} \longleftarrow$ matrix-$VC_k (\boldsymbol{v})$

Input: $\boldsymbol{v} \in \mathbb{R}^{N_k}$, R_j for $0 < j \leq k$, and A_j for $0 \leq j \leq k$.

Output: $\mathsf{B}_k\boldsymbol{v} \in \mathbb{R}^{N_k}$.

(0) $\mathsf{B}_0 = \mathsf{A}_0^{-1}$.

For $k \geq 1$ define B_k in terms of B_{k-1} as:

(1) $\boldsymbol{x}_1 = \mathsf{R}_k\boldsymbol{v}$.

(2) $\boldsymbol{x}_2 = \boldsymbol{x}_1 - \mathsf{C}_{k-1}^k\mathsf{B}_{k-1}\mathsf{C}_k^{k-1}\left(\mathsf{A}_k\boldsymbol{x}_1 - \boldsymbol{v}\right)$.

(3) $\mathsf{B}_k\boldsymbol{v} = \boldsymbol{x}_2 - \mathsf{R}_k\left(\mathsf{A}_k\boldsymbol{x}_2 - \boldsymbol{v}\right)$.

We will now derive an equivalent form of matrix-VC_k which is symmetric in the sense that the two smoothing steps have a similar structure. Moreover, the transfer of the error (respectively correction) from the fine to coarse (respectively coarse to fine) is demonstrated clearly.

Proposition 4.1.8 *Set*

$$\boldsymbol{y}_{(k)} = \boldsymbol{u}_{(k)}^n - \mathsf{R}_k\left(\mathsf{A}_k\boldsymbol{u}_{(k)}^n - \mathsf{L}_k\right) \text{ and } \boldsymbol{z}_{(k)} = \boldsymbol{y}_{(k)} - \mathsf{C}_{k-1}^k\mathsf{B}_{k-1}\mathsf{C}_k^{k-1}\left(\mathsf{A}_k\boldsymbol{y}_{(k)} - \mathsf{L}_k\right).$$

Then $\boldsymbol{u}_{(k)}^{n+1} = \boldsymbol{z}_{(k)} - \mathsf{R}_k\left(\mathsf{A}_k\boldsymbol{z}_{(k)} - \mathsf{L}_k\right)$.

Proof. The proof is immediate by the observation that $\boldsymbol{y}_{(k)}$ and $\boldsymbol{z}_{(k)}$ are the quantities obtained in Steps 1 and 2 of matrix-VC_k on taking $\boldsymbol{v} = \mathsf{A}_k\boldsymbol{u}_{(k)}^n - \mathsf{L}_k$ as input. □

Notice that if we view the calculations for $\boldsymbol{y}_{(k)}$, $\boldsymbol{z}_{(k)}$, and $\boldsymbol{u}_{(k)}^{n+1}$ as the three steps that yield one iteration of the V-cycle method, the computations for $\boldsymbol{y}_{(k)}$ and $\boldsymbol{u}_{(k)}^{n+1}$ have the form of a stationary linear iterative method for the solution of $\mathsf{A}_k\boldsymbol{x} = \mathsf{L}_k$. In fact, the operator R_k (and hence the matrix R_k) will be chosen such that these steps are equivalent to performing Gauss-Seidel iterations on $\mathsf{A}_k\boldsymbol{x} = \mathsf{L}_k$.

Remembering that $\mathsf{C}_k^{k-1}\mathsf{C}_{k-1}^k\boldsymbol{y} = \boldsymbol{y}$ for all $\boldsymbol{y} \in \mathbb{R}^{N_{k-1}}$, the calculation of $\boldsymbol{z}_{(k)}$ can be rewritten as

$$\mathsf{C}_k^{k-1}\left(\boldsymbol{z}_{(k)} - \boldsymbol{y}_{(k)}\right) = -\mathsf{B}_{k-1}\mathsf{C}_k^{k-1}\left(\mathsf{A}_k\boldsymbol{y}_{(k)} - \mathsf{L}_k\right).$$

Since the V-cycle is an iterative method, this is equivalent to solving

$$\mathsf{A}_{k-1}\mathsf{C}_k^{k-1}\left(\boldsymbol{z}_{(k)} - \boldsymbol{y}_{(k)}\right) = -\mathsf{C}_k^{k-1}\left(\mathsf{A}_k\boldsymbol{y}_{(k)} - \mathsf{L}_k\right)$$

approximately using the V-cycle. Hence, the computation of $\boldsymbol{z}_{(k)}$ in Proposition 4.1.8 is equivalent to the following two computations:

Find $\hat{\boldsymbol{y}}_{(k-1)} \in \mathbb{R}^{N_{k-1}}$ solving

$$\mathsf{A}_{k-1}\hat{\boldsymbol{y}}_{(k-1)} = \mathsf{A}_{k-1}\mathsf{C}_k^{k-1}\boldsymbol{y}_{(k)} - \mathsf{C}_k^{k-1}\left(\mathsf{A}_k\boldsymbol{y}_{(k)} - \mathsf{L}_k\right),$$

and then $\boldsymbol{z}_{(k)} \in \mathbb{R}^{N_k}$ is given by

$$\boldsymbol{z}_{(k)} = \boldsymbol{y}_{(k)} + \mathsf{C}_{k-1}^{k}\left(\hat{\boldsymbol{y}}_{(k-1)} - \mathsf{C}_k^{k-1}\boldsymbol{y}_{(k)}\right).$$

Remembering that the definition of B_k is recursive in the sense that it uses B_j for $j < k$, Proposition 4.1.8 and the above calculation gives a symmetric form of the V-cycle described in algorithm *symm-VC_k*. Algorithm symm-VC_k takes the nodal stiffness matrix A_k, a right hand side L_k, and a vector $\boldsymbol{u}_{(k)}^n$ as input and outputs the vector $\boldsymbol{u}_{(k)}^{n+1}$ obtained by performing one iteration of the V-cycle to the system $\mathsf{A}_k\boldsymbol{u}_{(k)}^n = \mathsf{L}_k$. Note that the algorithm will call symm-VC_j for $j < k$, and the right hand side vectors for these calls are computed within the algorithm (in particular, it is not equal to L_j when symm-VC_j is called).

Algorithm 4.1.9

$$\boldsymbol{u}_{(k)}^{n+1} \longleftarrow \text{symm-}VC_k\left(\boldsymbol{u}_{(k)}^n, \mathsf{A}_k, \mathsf{L}_k\right)$$

Input: $\boldsymbol{u}_{(k)}^n \in \mathbb{R}^{N_k}$, R_j for $0 < j \leq k$, A_j for $0 \leq j \leq k$, and L_k.

Output: $\boldsymbol{u}_{(k)}^{n+1} \in \mathbb{R}^{N_k}$.

(1) $\boldsymbol{y}_{(k)} = \boldsymbol{u}_{(k)}^n - \mathsf{R}_k\left(\mathsf{A}_k\boldsymbol{u}_{(k)}^n - \mathsf{L}_k\right)$ (pre-smooth using $\boldsymbol{u}_{(k)}^n$ as initial guess).

(2) Let $\mathsf{E}_{k-1} = \mathsf{A}_{k-1}\mathsf{C}_k^{k-1}\boldsymbol{y}_{(k)} - \mathsf{C}_k^{k-1}\left(\mathsf{A}_k\boldsymbol{y}_{(k)} - \mathsf{L}_k\right)$

The coarse mesh problem is $\mathsf{A}_{k-1}\hat{\boldsymbol{y}}_{(k-1)} = \mathsf{E}_{k-1}$.

(3) If $k \neq 1$, set $\boldsymbol{u}_{(k-1)}^0 = \mathsf{C}_k^{k-1}\boldsymbol{y}_{(k)}$,

$\boldsymbol{u}_{(k-1)}^1 \longleftarrow \text{symm-}VC_{k-1}\left(\boldsymbol{u}_{(k-1)}^0, \mathsf{A}_{k-1}, \mathsf{E}_{k-1}\right)$, $\hat{\boldsymbol{y}}_{(k-1)} = \boldsymbol{u}_{(k-1)}^1$.

Else if $k = 1$,

Solve $\mathsf{A}_0\hat{\boldsymbol{y}}_{(0)} = \mathsf{E}_0$ exactly.

(4) $\boldsymbol{z}_{(k)} = \boldsymbol{y}_{(k)} + \mathsf{C}_{k-1}^{k}\left(\hat{\boldsymbol{y}}_{(k-1)} - \mathsf{C}_{k}^{k-1}\boldsymbol{y}_{(k)}\right)$ (add back correction).

(5) $\boldsymbol{u}_{(k)}^{n+1} = \boldsymbol{z}_{(k)} - \mathsf{R}_k\left(\mathsf{A}_k\boldsymbol{z}_{(k)} - \mathsf{L}_k\right)$ (post-smooth using $\boldsymbol{z}_{(k)}$ as initial guess).

The stiffness matrices $\mathsf{A}_j, j < k$, are used in symm-VC_k. We could store all the stiffness matrices associated with $\mathcal{T}_j$ for $j < k$, but this can cause problems in the implementation. The dimensions of the spaces X_j are not known a priori; they are obtained by the adaptive mesh refinement procedure as the computation progresses. To overcome this difficulty, the matrix A_{k-1} will be computed from A_k via a change of basis. To this end, we introduce a 2–level basis for X_k using the meshes $\mathcal{T}_k$ and $\mathcal{T}_{k-1}$.

Remark 4.1.10 The choice of the initial guess $\boldsymbol{u}_{(k-1)}^{0}$ in Step 3 of symm-VC_k is nonstandard in the sense that many V-cycle algorithms start with a zero initial guess for the coarse mesh problem. This is due to the fact that the coarse mesh problem of Step 2 may be written as a linear system for the "correction" $\hat{\boldsymbol{y}}_{(k-1)} - \mathsf{C}_{k}^{k-1}\boldsymbol{y}_{(k)}$, or for the "solution" $\hat{\boldsymbol{y}}_{(k-1)}$ as we have done. Zero is a good initial guess in the former case, while the projection of the smoothed solution $\boldsymbol{y}_{(k)}$ is a good choice in the latter.

4.2 Basis changes and transfer operators

Definition 4.2.1 The 2–*level basis* for X_k is denoted by $\left\{\xi_i^k\right\}_{i=1}^{N_k}$, and defined as

$$\xi_i^k = \begin{cases} \phi_i^{k-1} & \text{if } 1 \le i \le N_{k-1}\ , \\ \phi_i^{k} & \text{if } N_{k-1} < i \le N_k. \end{cases}$$

Here it is assumed that $N_{k-1}+1, \ldots, N_k$, are the nodes (vertices) that appear in $\mathcal{T}_k$ but not in $\mathcal{T}_{k-1}$. These will be called *new nodes for level k*. The 2–level basis defined above is the hierarchical basis based on the two meshes $\mathcal{T}_{k-1}$ and $\mathcal{T}_k$. A hierarchical basis for the space X_k is defined similar to the 2–level basis except that it uses the hierarchy of meshes $\mathcal{T}_0, \ldots, \mathcal{T}_k$. Figure 10 gives an example of nodal and

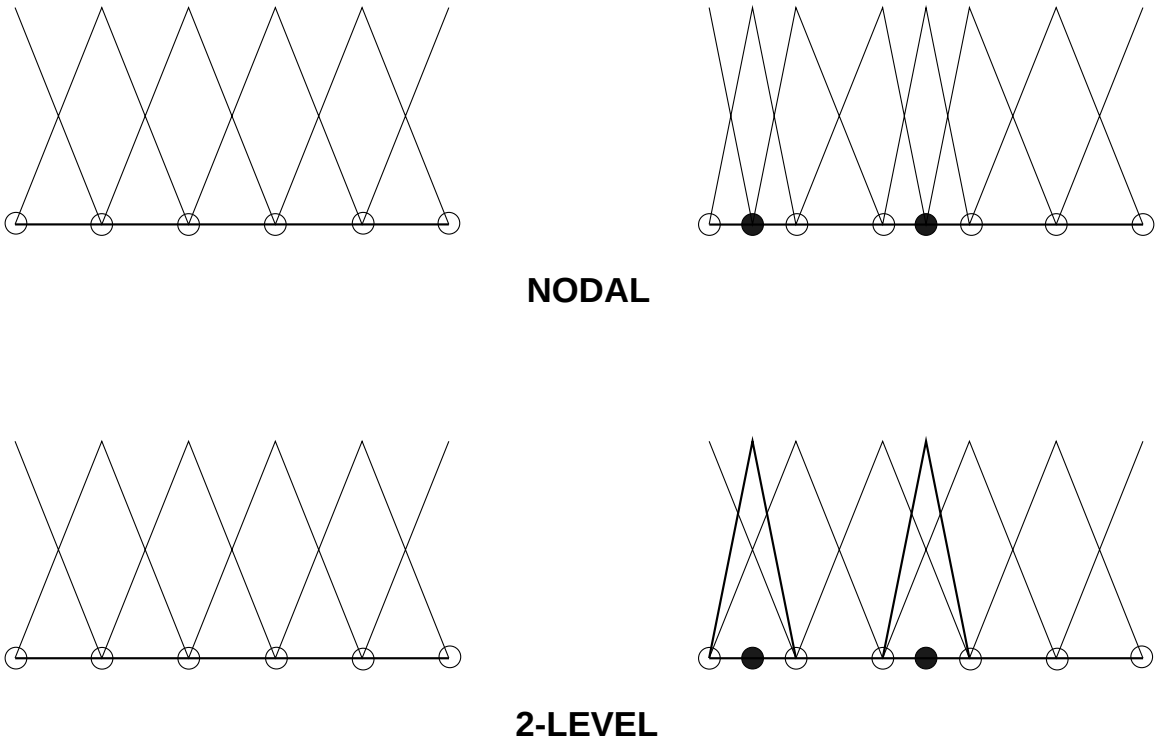

Figure 10. Nodal and 2–level bases in $\mathbb{R}^1$.

2–level bases for two meshes in $\mathbb{R}^1$. The finer mesh is obtained via a refinement of the coarse mesh, and the new nodes for the fine mesh are denoted by a $\bullet$.

Let A_k and $\widetilde{\mathsf{A}}_k$ be the stiffness matrices associated with the nodal and 2–level bases of X_k. Matrices and vectors associated with the 2–level basis will be characterized by putting a $\sim$ above them. The linear system we want to solve is $\mathsf{A}_k \boldsymbol{x} = \mathsf{L}_k$. The use of the nodal stiffness matrix in the formulation of the linear system is standard. Each row of the matrix A_k is associated with a node in $\mathcal{T}_k$. Based on the special numbering of the nodes, the stiffness matrices A_k and $\widetilde{\mathsf{A}}_k$ can be partitioned as

$$\mathsf{A}_k = \begin{bmatrix} \mathsf{A}_k^{(cc)} & \mathsf{A}_k^{(cf)} \\ \mathsf{A}_k^{(fc)} & \mathsf{A}_k^{(ff)} \end{bmatrix} \quad \text{and} \quad \widetilde{\mathsf{A}}_k = \begin{bmatrix} \widetilde{\mathsf{A}}_k^{(cc)} & \widetilde{\mathsf{A}}_k^{(cf)} \\ \widetilde{\mathsf{A}}_k^{(fc)} & \widetilde{\mathsf{A}}_k^{(ff)} \end{bmatrix},$$

where c and f are abbreviations for *coarse* and *fine*. Here fine represents all new nodes at level k and coarse stands for all other nodes. Any vector $\boldsymbol{v} \in \mathbb{R}^{N_k}$ is

represented using the two bases as

$$\boldsymbol{v} = \sum_{i=1}^{N_k} (\boldsymbol{v})_i \, \psi_i^k = \sum_{i=1}^{N_k} \left(\widetilde{\boldsymbol{v}}\right)_i \xi_i^k,$$

and partitioned as

$$\boldsymbol{v} = \begin{bmatrix} \boldsymbol{v}^{(c)} \\ \boldsymbol{v}^{(f)} \end{bmatrix}.$$

The matrix that transforms the vector of coefficients of an element in V_k from its 2–level to its nodal representation is called the 2–level matrix.

Definition 4.2.2 The 2–*level matrix* denoted by H_k (the letter H indicates the connection of the 2–level basis to the hierarchical basis) is defined as

$$\mathsf{H}_k = \begin{bmatrix} \mathsf{I} & \mathsf{O} \\ \mathsf{S} & \mathsf{I} \end{bmatrix},$$

where I denotes the identity matrix of appropriate order as determined by the partition into coarse and fine, and $\mathsf{S} \in \mathbb{R}^{(N_k - N_{k-1}) \times (N_{k-1})}$ has entries s_{ij} with $\mathsf{s}_{ij} = \xi_j^k (x_i, y_i, z_i)$. Here (x_i, y_i, z_i) are the coordinates of the i-th node.

The coefficients s_{ij} represent the values of some of the basis functions in the 2–level basis at the nodes that appear in the finest mesh only ($N_{k-1} < i \leq N_k$). The basis functions involved are some of the nodal basis functions for the mesh $\mathcal{T}_{k-1}$. With reference to Figure 10, $\mathsf{S} \in \mathbb{R}^{2 \times 6}$, and each row of S has exactly two non zero entries. In general we have,

$$\mathsf{s}_{ij} = \xi_j^k (x_i, y_i, z_i) = \phi_j^{k-1} (x_i, y_i, z_i) \text{ for } 1 \leq j \leq N_{k-1}.$$

The transformation matrices introduced earlier can be written as

$$\mathsf{C}_k^{k-1} = [\, \mathsf{I} \quad \mathsf{S}^T \,] \quad \text{and} \quad \mathsf{C}_{k-1}^k = \begin{bmatrix} \mathsf{I} \\ \mathsf{S} \end{bmatrix},$$

and the definition of H_k gives $\widetilde{\mathsf{A}}_k = {\mathsf{H}_k}^T \mathsf{A}_k \mathsf{H}_k$. The nodal system $\mathsf{A}_k \boldsymbol{x} = \mathsf{L}_k$ is the system of equations that are solved for obtaining the finite element solution on V_k,

and the 2–level system $\widetilde{\mathsf{A}}_k\widetilde{\boldsymbol{x}} = \widetilde{\mathsf{L}}_k$ is used in reconstructing the coarse mesh problem on V_{k-1}. Noting that the two systems

$$\mathsf{A}_k\boldsymbol{x} = \mathsf{L}_k \text{ and } \mathsf{H}_k^T\mathsf{A}_k\mathsf{H}_k\mathsf{H}_k^{-1}\boldsymbol{x} = \mathsf{H}_k^T\mathsf{L}_k$$

are equivalent, we have $\widetilde{\boldsymbol{x}} = \mathsf{H}_k^{-1}\boldsymbol{x}$ and $\widetilde{\mathsf{L}}_k = \mathsf{H}_k^T\mathsf{L}_k$. For $\boldsymbol{y}_{(k)} = \begin{bmatrix} \boldsymbol{y}_{(k)}^{(c)} & \boldsymbol{y}_{(k)}^{(f)} \end{bmatrix}^T \in \mathbb{R}^{N_k}$, the equivalence of the two systems $\mathsf{A}_k\boldsymbol{y}_{(k)} = \mathsf{L}_k$ and $\widetilde{\mathsf{A}}_k\widetilde{\boldsymbol{y}}_{(k)} = \widetilde{\mathsf{L}}_k$ leads to the relations

$$\begin{aligned}
\widetilde{\mathsf{A}}_k^{(cc)} &= \mathsf{A}_k^{(cc)} + \mathsf{S}^T\mathsf{A}_k^{(fc)} + \mathsf{A}_k^{(cf)}\mathsf{S} + \mathsf{S}^T\mathsf{A}_k^{(ff)}\mathsf{S}, \qquad \widetilde{\mathsf{A}}_k^{(ff)} = \mathsf{A}_k^{(ff)}, \\
\widetilde{\mathsf{A}}_k^{(fc)} &= \mathsf{A}_k^{(fc)} + \mathsf{A}_k^{(ff)}\mathsf{S}, \qquad \widetilde{\mathsf{A}}_k^{(cf)} = \mathsf{A}_k^{(cf)} + \mathsf{S}^T\mathsf{A}_k^{(ff)}, \\
\widetilde{\boldsymbol{y}}_{(k)}^{(c)} &= \boldsymbol{y}_{(k)}^{(c)}, \qquad \widetilde{\boldsymbol{y}}_{(k)}^{(f)} = \boldsymbol{y}_{(k)}^{(f)} - \mathsf{S}\boldsymbol{y}_{(k)}^{(c)}, \\
\widetilde{\mathsf{L}}_k^{(c)} &= \mathsf{L}_k^{(c)} + \mathsf{S}^T\mathsf{L}_k^{(f)}, \qquad \widetilde{\mathsf{L}}_k^{(f)} = \mathsf{L}_k^{(f)},
\end{aligned} \tag{19}$$

and the nodal stiffness matrices A_k and A_{k-1} satisfy $\mathsf{A}_{k-1} = \mathsf{C}_k^{k-1}\mathsf{A}_k\mathsf{C}_{k-1}^k$. The coarse mesh problem (Step 2 of symm-VC_k) is:

Given $\boldsymbol{y}_{(k)} = \begin{bmatrix} \boldsymbol{y}_{(k)}^{(c)} & \boldsymbol{y}_{(k)}^{(f)} \end{bmatrix}^T \in \mathbb{R}^{N_k}$ computed in Step 1, find $\hat{\boldsymbol{y}}_{(k-1)} \in \mathbb{R}^{N_{k-1}}$ satisfying

$$\mathsf{C}_k^{k-1}\mathsf{A}_k\mathsf{C}_{k-1}^k\left(\hat{\boldsymbol{y}}_{(k-1)} - \boldsymbol{y}_{(k)}^{(c)}\right) = \mathsf{C}_k^{k-1}\left(\mathsf{L}_k - \mathsf{A}_k\boldsymbol{y}_{(k)}\right).$$

The first equality in (19) and the definitions of the transformation matrices give

$$\widetilde{\mathsf{A}}_k^{(cc)} = \begin{bmatrix} \mathsf{I} & \mathsf{S}^T \end{bmatrix}\mathsf{A}_k\begin{bmatrix} \mathsf{I} \\ \mathsf{S} \end{bmatrix} = \mathsf{C}_k^{k-1}\mathsf{A}_k\mathsf{C}_{k-1}^k.$$

Using (19), the coarse mesh problem in Step 2 of symm-VC_k can be rewritten as:

Given $\boldsymbol{y}_{(k)} = \begin{bmatrix} \boldsymbol{y}_{(k)}^{(c)} & \boldsymbol{y}_{(k)}^{(f)} \end{bmatrix}^T \in \mathbb{R}^{N_k}$, find $\hat{\boldsymbol{y}}_{(k-1)} \in \mathbb{R}^{N_{k-1}}$ satisfying

$$\begin{aligned}
\widetilde{\mathsf{A}}_k^{(cc)}\hat{\boldsymbol{y}}_{(k-1)} &= \widetilde{\mathsf{A}}_k^{(cc)}\boldsymbol{y}_{(k)}^{(c)} + \mathsf{C}_k^{k-1}\left(\mathsf{L}_k - \mathsf{A}_k\boldsymbol{y}_{(k)}\right) \\
&= \mathsf{A}_k^{(cc)}\boldsymbol{y}_{(k)}^{(c)} + \mathsf{S}^T\mathsf{A}_k^{(fc)}\boldsymbol{y}_{(k)}^{(c)} + \mathsf{A}_k^{(cf)}\mathsf{S}\,\boldsymbol{y}_{(k)}^{(c)} + \mathsf{S}^T\mathsf{A}_k^{(ff)}\mathsf{S}\boldsymbol{y}_{(k)}^{(c)} \\
&\quad + \mathsf{L}_k^{(c)} + \mathsf{S}^T\mathsf{L}_k^{(f)} - \mathsf{A}_k^{(cc)}\boldsymbol{y}_{(k)}^{(c)} - \mathsf{A}_k^{(cf)}\boldsymbol{y}_{(k)}^{(f)} \\
&\quad - \mathsf{S}^T\mathsf{A}_k^{(fc)}\boldsymbol{y}_{(k)}^{(c)} - \mathsf{S}^T\mathsf{A}_k^{(ff)}\boldsymbol{y}_{(k)}^{(f)}
\end{aligned}$$

$$
\begin{aligned}
&= \mathsf{S}^T \mathsf{L}_k^{(f)} + \mathsf{L}_k^{(c)} - \mathsf{A}_k^{(cf)} \left(\boldsymbol{y}_{(k)}^{(f)} - \mathsf{S}\boldsymbol{y}_{(k)}^{(c)} \right) - \mathsf{S}^T \mathsf{A}_k^{(ff)} \left(\boldsymbol{y}_{(k)}^{(f)} - \mathsf{S}\boldsymbol{y}_{(k)}^{(c)} \right) \\
&= \widetilde{\mathsf{L}}_k^{(c)} - \left(\mathsf{A}_k^{(cf)} + \mathsf{S}^T \mathsf{A}_k^{(ff)} \right) \left(\boldsymbol{y}_{(k)}^{(f)} - \mathsf{S}\boldsymbol{y}_{(k)}^{(c)} \right) \\
&= \widetilde{\mathsf{L}}_k^{(c)} - \widetilde{\mathsf{A}}_k^{(cf)} \, \widetilde{\boldsymbol{y}}_{(k)}^{(f)}.
\end{aligned}
$$

Remembering that the coarse mesh part of the 2–level stiffness matrix is the nodal stiffness matrix of the coarser level, it immediately follows that the construction of the nodal stiffness matrix A_{k-1} from A_k can be realized via a basis change. The process of constructing the coarse mesh problem is called *restriction*. Restriction involves the creation of the problem to solve on $\mathcal{T}_{k-1}$ given the nodal stiffness matrix on $\mathcal{T}_k$, the nodal representation of a right hand side vector E_k, and the vector $\boldsymbol{y}_{(k)}$ created in Step 1 of symm-VC_k (recall that $\mathsf{E}_k = \mathsf{L}_k$ for the fine level mesh $\mathcal{T}_k$ in symm-VC_k).

Algorithm 4.2.3

$$(\mathsf{A}_{k-1}, \mathsf{E}_{k-1}) \longleftarrow \text{restrict } \left(\boldsymbol{y}_{(k)}, \mathsf{A}_k, \mathsf{E}_k \right)$$

Input: A_k, E_k, and $\boldsymbol{y}_{(k)}$ from Step 1 of symm-VC_k.

Output: A_{k-1} and E_{k-1}.

(1) Compute $\widetilde{\mathsf{A}}_k$, $\widetilde{\mathsf{E}}_k$, and $\widetilde{\boldsymbol{y}}_{(k)}$.

(2) Compute $\mathsf{E}_{k-1} = \widetilde{\mathsf{E}}_k^{(c)} - \widetilde{\mathsf{A}}_k^{(cf)} \, \widetilde{\boldsymbol{y}}_1^{(f)}$ and $\mathsf{A}_{k-1} = \widetilde{\mathsf{A}}_k^{(cc)}$.

Coarse mesh problem is $\mathsf{A}_{k-1} \hat{\boldsymbol{y}}_{(k-1)} = \mathsf{E}_{k-1}$.

Note that restrict only creates the coarse mesh problem that needs to be solved on mesh $\mathcal{T}_{k-1}$ after the pre-smoothing step has been performed for the nodal system on $\mathcal{T}_k$. The solution of this problem on $\mathcal{T}_{k-1}$ again involves a V-cycle iteration. Thus, to obtain the solution of the the coarse mesh problem on $\mathcal{T}_{k-1}$, pre-smoothing is performed on it and a new coarse mesh problem is formulated for $\mathcal{T}_{k-2}$ using restrict. This process is recursively repeated (Step 2 of symm-VC_k) till a coarse mesh problem is formulated on $\mathcal{T}_0$. This problem is solved exactly and the correction is added back to the next finer mesh. Post-smoothings are then performed on this

corrected problem before adding back the correction to the next finer mesh. This process is repeated till we get $\boldsymbol{u}_{(k)}^{n+1}$. The process of adding the correction from the coarse mesh (Step 4 of symm-VC_k) is called *prolongation*. This can be summarized as:

Given $\boldsymbol{y}_{(k)} \in \mathbb{R}^{N_k}$, and $\hat{\boldsymbol{y}}_{(k-1)} \in \mathbb{R}^{N_{k-1}}$, compute $\boldsymbol{z}_{(k)} \in \mathbb{R}^{N_k}$ by

$$\boldsymbol{z}_{(k)} = \boldsymbol{y}_{(k)} + \mathsf{C}_{k-1}^{k} \left(\hat{\boldsymbol{y}}_{(k-1)} - \boldsymbol{y}_{(k)}^{(c)} \right).$$

Using the relations (19), we obtain

$$\begin{aligned} \boldsymbol{z}_{(k)} &= \begin{bmatrix} \boldsymbol{y}_{(k)}^{(c)} \\ \boldsymbol{y}_{(k)}^{(f)} \end{bmatrix} + \begin{bmatrix} \mathsf{I} \\ \mathsf{S} \end{bmatrix} \left[\hat{\boldsymbol{y}}_{(k-1)} - \boldsymbol{y}_{(k)}^{(c)} \right] \\ &= \begin{bmatrix} \boldsymbol{y}_{(k)}^{(c)} + \hat{\boldsymbol{y}}_{(k-1)} - \boldsymbol{y}_{(k)}^{(c)} \\ \widetilde{\boldsymbol{y}}_{(k)}^{(f)} + \mathsf{S}\,\widetilde{\boldsymbol{y}}_{(k)}^{(c)} + \mathsf{S}\,\hat{\boldsymbol{y}}_{(k-1)} - \mathsf{S}\,\boldsymbol{y}_{(k)}^{(c)} \end{bmatrix} \\ &= \begin{bmatrix} \hat{\boldsymbol{y}}_{(k-1)} \\ \widetilde{\boldsymbol{y}}_{(k)}^{(f)} + \mathsf{S}\hat{\boldsymbol{y}}_{(k-1)} \end{bmatrix} = \mathsf{H}_k \begin{bmatrix} \hat{\boldsymbol{y}}_{(k-1)} \\ \widetilde{\boldsymbol{y}}_{(k)}^{(f)} \end{bmatrix}. \end{aligned}$$

We will denote by algorithm *prolong* the algorithm that realizes the process of prolongation.

Algorithm 4.2.4

$$\boldsymbol{z}_{(k)} \longleftarrow \text{prolong}\left(\boldsymbol{y}_{(k)}, \hat{\boldsymbol{y}}_{(k-1)}\right)$$

Input: $\boldsymbol{y}_{(k)} \in \mathbb{R}^{N_k}$ and $\hat{\boldsymbol{y}}_{(k-1)} \in \mathbb{R}^{N_{k-1}}$ from Steps 1 and 3 of symm-VC_k.

Output: $\boldsymbol{z}_{(k)} \in \mathbb{R}^{N_k}$.

(1) Compute $\widetilde{\boldsymbol{y}}_{(k)}$.

(2) $\boldsymbol{z}_{(k)} = \mathsf{H}_k \begin{bmatrix} \hat{\boldsymbol{y}}_{(k-1)} \\ \widetilde{\boldsymbol{y}}_{(k)}^{(f)} \end{bmatrix}$.

The algorithms presented in this section assume knowledge of the transformation and 2–level matrices. The basis changes do not interfere with the efficiency of the multilevel algorithm. The matrix S has only two nonzero entries in every row. This is a direct consequence of the fact that we use linear finite elements and the mesh refinement procedure does not introduce any new nodes in the interior of tetrahedra. The restriction and prolongation procedures involve computing the

action of the matrix H_k. This means that the number of operations involved in these procedures is a small multiple of the number of new nodes at the finest level. If we assume $N_k = FN_{k-1}$, for some $F > 1$, the work done for the restriction and prolongation is $O(N_k)$.

4.3 The smoothing operator

The description of the multilevel solver will be complete once we describe how the smoothing operations are performed. The smoothing will be performed by doing Gauss-Seidel iterations on the system $\mathsf{A}_k\boldsymbol{x} = \mathsf{E}_k$ using an appropriate initial guess. Let A_k be decomposed as

$$\mathsf{A}_k = \mathsf{D}_k + \mathsf{LT}_k + \mathsf{LT}_k^T,$$

where D_k denotes the diagonal part of A_k, and LT_k denotes its lower triangular part. The Gauss-Seidel method for the solution of $\mathsf{A}_k\boldsymbol{x} = \mathsf{E}_k$ is:

Given an initial guess $\boldsymbol{x}_0 \in \mathbb{R}^{N_k}$, compute $\boldsymbol{x}_{n+1}$ form $\boldsymbol{x}_n$ by

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \left(\mathsf{D}_k + \mathsf{LT}_k\right)^{-1}\left(\mathsf{A}_k\boldsymbol{x}_n - \mathsf{L}_k\right). \tag{20}$$

Bramble and Pasciak [14] have shown that in applying multilevel V-cycle to locally adaptive meshes, the smoothing can be performed in only those parts of the domain where the new nodes are being added during the refinement phase.

Definition 4.3.1 Any two nodes $i, j \in \mathcal{N}(\mathcal{T}_k)$ will be called *neighbors* if

$$\text{supp}\left(\phi_i^k\right) \bigcap \text{supp}\left(\phi_j^k\right) \neq \emptyset,$$

where supp(φ) denotes the set of points on which φ is not zero.

In Steps 1 and 5 of Symm-VC_k, we will perform the smoothing on all new nodes for level k, and all neighbors of these that are not new nodes for level k. It is known that if the smoothing is performed only on the new nodes for level k, and the smoothing operator is such that the smoothing step corresponds to a symmetric

Gauss-Seidel method for the linear system, the V-cycle method is the "hierarchical basis multigrid method" of Bank et.al. [6]. It is also known that the hierarchical basis method is not efficient for elliptic boundary value problems in $\mathbb{R}^3$ (see for example Xu [35]). The smoothing used in AMG3DP1 is sufficient to overcome this difficulty while not interfering with the $O(N)$ efficiency of the V-cycle iteration. If we assume $N_k = FN_{k-1}$, for some $F > 1$, the number of operations needed for smoothing on all nodes at level k is $O(N_k)$. Obviously, if the smoothing is done for only some of the nodes at level k, the number of operations needed to perform the smoothing step will be $O(N_k)$. The Gauss-Seidel method (20) can be rewritten as

For $1 \leq i \leq N_k$,

$$(\boldsymbol{x}_{n+1})_i = (\mathsf{L}_k)_i - \frac{1}{a_{ii}} \left(\sum_{j<i} a_{ij} (\boldsymbol{x}_n)_j + \sum_{j>i} a_{ij} (\boldsymbol{x}_{n+1})_j \right).$$

Using this formulation, we can perform the smoothing on any subset of the nodes in $\mathcal{T}_k$. Let algorithm *smooth-GS*$_k$ be the algorithm that performs Gauss-Seidel smoothing on $\mathcal{T}_k$. It takes $\mathsf{A}_k, \mathsf{E}_k, \boldsymbol{x} \in \mathbb{R}^{n_k}$, and a subset S of nodes in $\mathcal{N}(\mathcal{T}_k)$ as input and outputs a vector $\boldsymbol{x}' \in \mathbb{R}^{N_k}$.

Algorithm 4.3.2

$\boldsymbol{x}' \longleftarrow$ smooth-$GS_k(\mathsf{A}_k, \mathsf{E}_k, \boldsymbol{x}, S)$

Input: $\mathsf{A}_k, \mathsf{E}_k, \boldsymbol{x} \in \mathbb{R}^{N_k}$, and $S \subseteq \mathcal{N}(\mathcal{T}_k)$.

Output: $\boldsymbol{x}' \in \mathbb{R}^{N_k}$.

(1) Set $N(S) = \{i \in \mathcal{N}(\mathcal{T}_k) \,|\, i \text{ is a neighbor of some } n \in S\}$.

(2) For $i \in N(S)$,

$$(\boldsymbol{x}')_i = (\mathsf{E}_k)_i - \frac{1}{a_{ii}} \left(\sum_{j<i} a_{ij} (\boldsymbol{x})_j + \sum_{j>i} a_{ij} (\boldsymbol{x}')_j \right).$$

We will now use algorithms restrict, prolong, and smooth-GS_k to rewrite symm-VC_k. The algorithm we present here (algorithm *solve-MG*$_k$) solves the linear system arising from the finite element discretization of the boundary value problem (1)–(3) using the mesh $\mathcal{T}_k$. It involves repetitions of symm-VC_k starting with an initial

guess $\boldsymbol{u}_{(k)}^0$ till some convergence criterion is satisfied. Recall that for any two meshes $\mathcal{T}_k$ and $\mathcal{T}_{k-1}$, the nodes are assumed to be numbered such that the new nodes for level k are the ones with numbers $N_k + 1, \ldots, N_k$. The smoothing on the mesh $\mathcal{T}_k$ will only be carried out for the new nodes for level k.

Algorithm 4.3.3

$$\hat{\boldsymbol{u}}_{(k)} \longleftarrow \text{solve-}MG_k\left(\mathsf{A}_k, \mathsf{E}_k, \boldsymbol{u}_{(k)}^0\right)$$

Input: $\mathsf{A}_k, \mathsf{E}_k,$ and initial guess $\boldsymbol{u}_{(k)}^0 \in \mathbb{R}^{N_k}$.

Output: $\hat{\boldsymbol{u}}_{(k)} \in \mathbb{R}^{N_k}$ approximating $\hat{u}_k \in V_k$.

Set $n = 0$.

(1) If $\left\|\mathsf{A}_k \boldsymbol{u}_{(k)}^n - \mathsf{E}_k\right\| > \text{tol}$,

(2) For $j = k, k-1, \ldots, 1$

$$\boldsymbol{y}_{(j)} \longleftarrow \text{smooth-}GS_j\left(\mathsf{A}_j, \mathsf{E}_j, \boldsymbol{u}_{(j)}^n, \{\nu : N_{j-1} < \nu \leq N_j\}\right)$$

$$(\mathsf{A}_{j-1}, \mathsf{E}_{j-1}) \longleftarrow \text{restrict}\left(\boldsymbol{y}_{(j)}, \mathsf{A}_j, \mathsf{E}_j\right) \text{ and } \boldsymbol{u}_{(j-1)}^n = \mathsf{C}_j^{j-1}\boldsymbol{y}_{(j)}$$

End For.

(3) Solve $\mathsf{A}_0 \hat{\boldsymbol{y}}_{(0)} = \mathsf{E}_0$.

(4) For $j = 1, 2, \ldots, k$

$$\boldsymbol{z}_{(j)} \longleftarrow \text{prolong}\left(\boldsymbol{y}_{(j)}, \hat{\boldsymbol{y}}_{(j-1)}\right)$$

$$\boldsymbol{u}_{(j)}^{n+1} \longleftarrow \text{smooth-}GS_j\left(\mathsf{A}_j, \mathsf{E}_j, \boldsymbol{z}_{(j)}, \{\nu : N_{j-1} < \nu \leq N_j\}\right)$$

End For.

(5) $n \longleftarrow n + 1$ and go to (1).

(6) Else $\hat{\boldsymbol{u}}_{(k)} = \boldsymbol{u}_{(k)}^n$.

This method for the implementation of the V-cycle has been used before. In particular, Bank [5] uses it to implement the hierarchical basis method in PLTMG. The same procedure has also been used by Mitchell [29] for implementing the V-cycle in MGGHAT. This form of the V-cycle does not involve recursive calls to subroutines and is easily implemented using Fortran.

Chapter 5

AMG3DP1 and an application to semilinear problems

The first part of this chapter describes the algorithm used by AMG3DP1 to solve the boundary value problem (1)–(3) of Chapter 2. The algorithm uses the residual error estimator η_τ of Chapter 2, a locally adaptive mesh refinement algorithm similar to local-refine of Chapter 3, and the multilevel algorithm solve-MG_k of Chapter 4. Algorithm *linear-bvp* in this chapter is similar to a full multigrid. Starting with a coarse mesh $\mathcal{T}_0$ it creates a sequence of nested meshes $\mathcal{T}_0, \mathcal{T}_1, \ldots$, each refining the previous with the requirement that the number of nodes in the meshes approximately double at each step. The approximate (finite element) solution is calculated on the current mesh $\mathcal{T}_j$ together with the residual error estimators η_τ for $\tau \in \mathcal{T}_j$. The calculation of the approximate solution involves the solution of a linear system. This is achieved via solve-MG_j. The values of η_τ are used to decide how $\mathcal{T}_j$ is refined to obtain $\mathcal{T}_{j+1}$, and the finite element solution on $\mathcal{T}_j$ is interpolated onto $\mathcal{T}_{j+1}$ to be used as the initial guess for the V-cycle iteration there. Algorithm *adapt-mesh* creates an adapted conforming mesh using a given conforming mesh. Adapt-mesh ensures that the number of nodes in the mesh is approximately doubled, and attempts to equidistribute the error estimator on the finer mesh. The finite element solution, nodal stiffness matrix, and nodal right hand side vector associated with the coarse mesh are also input to adapt-mesh and the corresponding quantities for the fine mesh are output by it. Section 1 describes adapt-mesh and linear-bvp in detail. AMG3DP1 can handle domains with spherical boundaries. The procedure used for this is also outlined briefly at the end of Section 1.

Section 2 describes a Newton multilevel method to solve semilinear boundary value problems. For solving the semilinear boundary value problem on a given mesh, the method requires the solution of a sequence of linear boundary value problems. These are solved using solve-MG_k and the solution to the last problem is used in conjunction with adapt-mesh to create the next finer mesh. Algorithm *semilin-bvp* is an algorithm based on these concepts and is presented in this section.

In Section 3 we present some numerical tests that demonstrate the abilities and also some limitations of AMG3DP1. We also provide an example of a semilinear boundary value problem solved using semilin-bvp.

5.1 An algorithm for linear boundary value problems

Given a marked conforming mesh $\mathcal{T}$, the nodal stiffness matrix $\mathsf{A}_{\mathcal{T}}$, the nodal representation $\mathsf{L}_{\mathcal{T}}$ of the linear form $l(\cdot)$, and the finite element solution $\boldsymbol{u}_{\mathcal{T}}$, adapt-mesh produces a conforming mesh $\mathcal{T}'$ such that the number of nodes in $\mathcal{T}'$ are approximately double that of $\mathcal{T}$ and the tetrahedra in $\mathcal{T}$ with higher values of η_τ are bisected "more" than those with lower values resulting in an approximate equidistribution of the error estimator in $\mathcal{T}'$. Moreover, the updated stiffness matrix $\mathsf{A}_{\mathcal{T}'}$, the updated vector $\mathsf{L}_{\mathcal{T}'}$, and the interpolated solution $\boldsymbol{u}_{\mathcal{T}'}$ are also output by adapt-mesh.

We first present an expression for calculating a number q_τ assigned to each $\tau \in \mathcal{T}$ with the following property:

- The number of nodes in the mesh obtained by bisecting each $\tau \in \mathcal{T}$ q_τ times is approximately twice the number of nodes in $\mathcal{T}$.
- The values of the error estimator is approximately the same for all tetrahedra in the mesh obtained from $\mathcal{T}$ by bisecting each $\tau \in \mathcal{T}$ q_τ times (the error is equidistributed on the new mesh).

We assume that the values η_τ are known for all $\tau \in \mathcal{T}$ and use these to obtain an expression for q_τ. Theorem 2.2.1 says that the H^1 norm of the error due to the

discretization decreases like $h(\mathcal{T})$. Locally adapted meshes may contain tetrahedra with widely varying diameters. The mesh size $h(\mathcal{T})$ defined in Chapter 2 is not useful in characterizing the size of such meshes. For these meshes, it is common to use the number of nodes in the mesh as an indicator of its size. Define a finite element method to be of *order* α if the error due to the discretization decreases like $N^{-\alpha}$. Since we use linear finite elements in AMG3DP1, the method has optimal order 1/3 in the H^1 norm. Assume that the mesh $\mathcal{T}$ has $N = \text{card}(\mathcal{N}(\mathcal{T}))$ nodes and recall that for a uniform mesh $\mathcal{T}$, $N^{-1/3} = O(h(\mathcal{T}))$. In this section the symbol $\approx$ will be used to indicate that two quantities are comparable.

Remark 5.1.1 Most of the discussion is heuristic and is used to design an algorithm that gives approximate node doubling in the sequence of meshes that are created during the solution of a boundary value problem by AMG3DP1. However, algorithm adapt-mesh of this section performs well in practice.

Theorem 2.2.1 together with the discussion and definitions above gives

$$\frac{\|\hat{u} - u_{\mathcal{T}'}\|_1}{\|\hat{u} - u_{\mathcal{T}}\|_1} \approx \frac{(2N)^{-\frac{1}{3}}}{N^{-\frac{1}{3}}} = 2^{-\frac{1}{3}}.$$

If $\eta_\tau^{\mathcal{T}}$ and $\eta_\tau^{\mathcal{T}'}$ are the residual error estimators for an arbitrary tetrahedron τ in $\mathcal{T}$ and $\mathcal{T}'$ respectively, it follows that

$$\frac{\sum_{\tau \in \mathcal{T}'} \left(\eta_\tau^{\mathcal{T}'}\right)^2}{\sum_{\tau \in \mathcal{T}} \left(\eta_\tau^{\mathcal{T}}\right)^2} \approx 2^{-\frac{2}{3}}.$$

Assuming that the error estimators are approximately equidistributed on $\mathcal{T}'$ so that $\eta_\tau^{\mathcal{T}'} = \bar{\eta}^{\mathcal{T}'}$ for all $\tau \in \mathcal{T}'$, we obtain

$$\bar{\eta}^{\mathcal{T}'} = \frac{2^{-\frac{2}{3}}}{2\,\text{card}(\mathcal{T})} \sum_{\tau \in \mathcal{T}} \left(\eta_\tau^{\mathcal{T}}\right)^2. \tag{21}$$

Note that we have assumed that the mesh $\mathcal{T}'$ has $2\,\text{card}(\mathcal{T})$ tetrahedra and it follows from (21) that $\bar{\eta}^{\mathcal{T}'}$ is computable from $\mathcal{T}$ and $u_{\mathcal{T}}$.

Assuming further that the exact solution $\hat{u} \in H^2(\Omega)$, we get the following estimate for the error:

$$\|\hat{u} - u_{\mathcal{T}}\|_{1;\tau}^2 \approx h(\tau)^2 \|\hat{u}\|_{2;\tau}^2 = O\left(h(\tau)^5\right).$$

Recalling that $\|\hat{u} - u_{\mathcal{T}}\|_1^2 \approx \sum_{\tau \in \mathcal{T}} \left(\eta_\tau^{\mathcal{T}}\right)^2$ if we ignore the consistency terms in the computation of η_τ, we get $\left(\eta_\tau^{\mathcal{T}}\right)^2 \approx h(\tau)^5$ by comparing $\eta_\tau^{\mathcal{T}}$ with the last expression. When a tetrahedron τ is bisected into $\tau^{(1)}$ and $\tau^{(2)}$ by a plane passing through the midpoint of an edge and the two vertices of τ that are not on that edge (as is done in bisect-tet), we have $h\left(\tau^{(i)}\right)^3 \approx \text{vol}(\tau^{(i)}) = \frac{1}{2}\text{vol}(\tau) \approx \frac{1}{2}h(\tau)^3$. Assuming that the two values $\eta_{\tau^{(i)}}^{\mathcal{T}'}$ are approximately equal, it follows that

$$\frac{\left(\eta_{\tau^{(i)}}^{\mathcal{T}'}\right)^2}{\left(\eta_\tau^{\mathcal{T}}\right)^2} \approx \frac{h\left(\tau^{(i)}\right)^5}{h(\tau)^5} = 2^{-5/3} \text{ for } i = 1, 2.$$

By a similar argument, we have

$$\frac{\left(\eta_{\tau^{(i)}}^{\mathcal{T}'}\right)^2}{\left(\eta_\tau^{\mathcal{T}}\right)^2} \approx 2^{-5q_\tau/3} \text{ for } 1 \leq i \leq 2^{q_\tau} \tag{22}$$

for any descendent $\tau^{(i)}$ of τ obtained by applying bisect-tet q_τ times. If we assume that $\mathcal{T}'$ is obtained from $\mathcal{T}$ by bisecting each $\tau \in \mathcal{T}$ q_τ times, and the criteria of approximate equidistribution of the error estimator together with approximate node doubling are satisfied for $\mathcal{T}'$, q_τ satisfies (22) with $\eta_{\tau^{(i)}}^{\mathcal{T}} = \bar{\eta}^{\mathcal{T}'}$. Hence, the equation for determining q_τ is

$$2^{-5q_\tau/3}\left(\eta_\tau^{\mathcal{T}}\right)^2 = \left(\bar{\eta}^{\mathcal{T}'}\right)^2 \text{ or } q_\tau = -\frac{\ln\left[\left(\bar{\eta}^{\mathcal{T}'}\right)^2 - \left(\eta_\tau^{\mathcal{T}}\right)^2\right]}{\ln 2^{5/3}}. \tag{23}$$

Algorithm adapt-mesh is based on bisecting subsets of tetrahedra in $\mathcal{T}$ as indicated by q_τ and then ensuring that the resulting mesh is conforming. However, since adapt-mesh also updates the matrices, vectors, and solution during the mesh

refinement process, we will need algorithms parallel to bisect-tet, bisect-tets, and refine-to-conformity which perform these updates together with the inherent mesh refinement features in these algorithms. Some definitions and notations needed for these are presented next. A marked mesh $\mathcal{T}$ is called *number-graded* if there is a number $q_\tau \geq 0$ assigned to every $\tau \in \mathcal{T}$. For an arbitrary number-graded mesh $\mathcal{T}$ with nodal basis $\{\phi_\nu\}_{\nu \in \mathcal{N}(\mathcal{T})}$ associated to the finite element space $X(\mathcal{T})$, denote by $\mathsf{A}_\mathcal{T}$ the nodal stiffness matrix, by $\mathsf{L}_\mathcal{T}$ the vector of inner products associated with the linear form, and by $\boldsymbol{v}_\mathcal{T}$ an arbitrary nodal vector of coefficients.

Algorithm *up-bisect-tet* is an algorithm that bisects a single tetrahedron in a number-graded mesh and updates the stiffness matrix, right hand side, and any vector associated with the mesh.

Algorithm 5.1.2

$$(\mathcal{T}', \boldsymbol{v}_{\mathcal{T}'}, \mathsf{A}_{\mathcal{T}'}, \mathsf{L}_{\mathcal{T}'}) \longleftarrow \text{up-bisect-tet}\,(\mathcal{T}, \tau, \boldsymbol{v}_\mathcal{T}, \mathsf{A}_\mathcal{T}, \mathsf{L}_\mathcal{T})$$

Input: Number-graded mesh $\mathcal{T}$, $\tau \in \mathcal{T}$, nodal vector $\boldsymbol{v}_\mathcal{T}$,
nodal stiffness matrix $\mathsf{A}_\mathcal{T}$, and nodal right hand side $\mathsf{L}_\mathcal{T}$.

Output: Number-graded mesh $\mathcal{T}'$, nodal vector $\boldsymbol{v}_{\mathcal{T}'}$,
nodal stiffness matrix $\mathsf{A}_{\mathcal{T}'}$, and nodal right hand side $\mathsf{L}_{\mathcal{T}'}$.

(1) $\left\{\tau^{(1)}, \tau^{(2)}\right\} \longleftarrow \text{bisect-tet}\,(\tau)$

$\mathcal{T}' = (\mathcal{T} \setminus \{\tau\}) \cup \left\{\tau^{(1)}, \tau^{(2)}\right\}$,

$q_{\tau^{(i)}} = \max(0, q_\tau - 1)$, $i = 1, 2$.

(2) Let $\nu_1\nu_2$ be the refinement edge of τ and $\nu_n = \frac{1}{2}(\nu_1 + \nu_2)$ the new node;

$$(\boldsymbol{v}_{\mathcal{T}'})_\nu = \begin{cases} (\boldsymbol{v}_\mathcal{T})_\nu & \text{if } \nu \in \mathcal{N}(\mathcal{T}), \\ \frac{1}{2}\left[(\boldsymbol{v}_\mathcal{T})_{\nu_1} + (\boldsymbol{v}_\mathcal{T})_{\nu_2}\right] & \text{if } \nu = \nu_n. \end{cases}$$

(3) The rows and columns of the square matrix $\mathsf{A}_\mathcal{T}$ are indexed by $\mathcal{N}(\mathcal{T})$ and those of $\mathsf{A}_{\mathcal{T}'}$ by $\mathcal{N}(\mathcal{T}') = \mathcal{N}(\mathcal{T}) \cup \{\nu_n\}$; $\mathsf{A}_{\mathcal{T}'}$ is derived from $\mathsf{A}_\mathcal{T}$ by setting the (ν_1, ν_2) and (ν_2, ν_1) entries to zero, and adding a row and column corresponding to ν_n with nonzero entries $a(\phi_\nu, \phi_{\nu_n})$ and $a(\phi_{\nu_n}, \phi_\nu)$, $\nu \in \mathcal{N}(\tau) \cup \{\nu_n\}$.

(4) Append $l(\phi_{\nu_n})$ to $\mathsf{L}_{\mathcal{T}}$ to create $\mathsf{L}_{\mathcal{T}'}$.

Next, we describe algorithm *up-bisect-tets* and *up-refine-to-conformity* which have the same mesh refinement features as the corresponding algorithms in Chapter 3, and are based on up-bisect-tet.

Algorithm 5.1.3

$(\mathcal{T}', \boldsymbol{v}_{\mathcal{T}'}, \mathsf{A}_{\mathcal{T}'}, \mathsf{L}_{\mathcal{T}'}) \longleftarrow$ up-bisect-tets $(\mathcal{T}, S, \boldsymbol{v}_{\mathcal{T}}, \mathsf{A}_{\mathcal{T}}, \mathsf{L}_{\mathcal{T}})$

Input: Number-graded mesh $\mathcal{T}$, $S \subseteq \mathcal{T}$, nodal vector $\boldsymbol{v}_{\mathcal{T}}$, nodal stiffness matrix $\mathsf{A}_{\mathcal{T}}$, and nodal right hand side $\mathsf{L}_{\mathcal{T}}$.

Output: Number-graded mesh $\mathcal{T}'$, nodal vector $\boldsymbol{v}_{\mathcal{T}'}$, nodal stiffness matrix $\mathsf{A}_{\mathcal{T}'}$, and nodal right hand side $\mathsf{L}_{\mathcal{T}'}$.

(1) If $S \neq \emptyset$,

Choose $\tau \in S$

$(\bar{\mathcal{T}}, \boldsymbol{v}_{\bar{\mathcal{T}}}, \mathsf{A}_{\bar{\mathcal{T}}}, \mathsf{L}_{\bar{\mathcal{T}}}) \longleftarrow$ up-bisect-tet $(\mathcal{T}, \tau, \boldsymbol{v}_{\mathcal{T}}, \mathsf{A}_{\mathcal{T}}, \mathsf{L}_{\mathcal{T}})$

Set $\bar{S} = S \setminus \{\tau\}$

$(\mathcal{T}', \boldsymbol{v}_{\mathcal{T}'}, \mathsf{A}_{\mathcal{T}'}, \mathsf{L}_{\mathcal{T}'}) \longleftarrow$ up-bisect-tets $(\bar{\mathcal{T}}, \bar{S}, \boldsymbol{v}_{\bar{\mathcal{T}}}, \mathsf{A}_{\bar{\mathcal{T}}}, \mathsf{L}_{\bar{\mathcal{T}}})$.

(2) Else, $\mathcal{T}' = \mathcal{T}, \boldsymbol{v}_{\mathcal{T}'} = \boldsymbol{v}_{\mathcal{T}'}, \mathsf{A}_{\mathcal{T}'} = \mathsf{A}_{\mathcal{T}}, \mathsf{L}_{\mathcal{T}'} = \mathsf{L}_{\mathcal{T}}$.

Algorithm 5.1.4

$(\mathcal{T}', \boldsymbol{v}_{\mathcal{T}'}, \mathsf{A}_{\mathcal{T}'}, \mathsf{L}_{\mathcal{T}'}) \longleftarrow$ up-refine-to-conformity $(\mathcal{T}, \boldsymbol{v}_{\mathcal{T}}, \mathsf{A}_{\mathcal{T}}, \mathsf{L}_{\mathcal{T}})$

Input: Number-graded mesh $\mathcal{T}$, nodal vector $\boldsymbol{v}_{\mathcal{T}}$, nodal stiffness matrix $\mathsf{A}_{\mathcal{T}}$, and nodal right hand side $\mathsf{L}_{\mathcal{T}}$.

Output: Number-graded conforming mesh $\mathcal{T}'$, nodal vector $\boldsymbol{v}_{\mathcal{T}'}$, nodal stiffness matrix $\mathsf{A}_{\mathcal{T}'}$, and nodal right hand side $\mathsf{L}_{\mathcal{T}'}$.

(1) Set $S = \{\tau \in \mathcal{T} | \tau$ has a hanging node$\}$.

(2) If $S \neq \emptyset$,

$(\bar{\mathcal{T}}, \boldsymbol{v}_{\bar{\mathcal{T}}}, \mathsf{A}_{\bar{\mathcal{T}}}, \mathsf{L}_{\bar{\mathcal{T}}}) \longleftarrow$ up-bisect-tets $(\mathcal{T}, S, \boldsymbol{v}_{\mathcal{T}}, \mathsf{A}_{\mathcal{T}}, \mathsf{L}_{\mathcal{T}})$

$(\mathcal{T}', \boldsymbol{v}_{\mathcal{T}'}, \mathsf{A}_{\mathcal{T}'}, \mathsf{L}_{\mathcal{T}'}) \longleftarrow$ up-refine-to-conformity $(\bar{\mathcal{T}}, \boldsymbol{v}_{\bar{\mathcal{T}}}, \mathsf{A}_{\bar{\mathcal{T}}}, \mathsf{L}_{\bar{\mathcal{T}}})$.

(3) Else, $\mathcal{T}' = \mathcal{T}, \boldsymbol{v}_{\mathcal{T}'} = \boldsymbol{v}_{\mathcal{T}'}, \mathsf{A}_{\mathcal{T}'} = \mathsf{A}_{\mathcal{T}}, \mathsf{L}_{\mathcal{T}'} = \mathsf{L}_{\mathcal{T}}$.

Algorithm adapt-mesh takes a number-graded conforming mesh $\mathcal{T}$, the stiffness matrix $\mathsf{A}_{\mathcal{T}}$, the right hand side vector $\mathsf{L}_{\mathcal{T}}$, and the finite element solution $\boldsymbol{u}_{\mathcal{T}}$ as input. It is further assumed that the numbers q_τ for $\tau \in \mathcal{T}$ are calculated using η_τ and (23).

Algorithm 5.1.5

$$(\mathcal{T}', u_{\mathcal{T}'}, \mathsf{A}_{\mathcal{T}'}, \mathsf{L}_{\mathcal{T}'}) \longleftarrow \text{adapt-mesh}\,(\mathcal{T}, \boldsymbol{u}_{\mathcal{T}}, \mathsf{A}_{\mathcal{T}}, \mathsf{L}_{\mathcal{T}})$$

Input: Number-graded conforming mesh $\mathcal{T}$, finite element solution $\boldsymbol{u}_{\mathcal{T}}$, nodal stiffness matrix $\mathsf{A}_{\mathcal{T}}$, and nodal right hand side $\mathsf{L}_{\mathcal{T}}$.

Output: Number-graded conforming mesh $\mathcal{T}'$, interpolated solution $\boldsymbol{u}_{\mathcal{T}'}$, nodal stiffness matrix $\mathsf{A}_{\mathcal{T}'}$, and nodal right hand side $\mathsf{L}_{\mathcal{T}'}$.

(1) Set $S = \{\tau \in \mathcal{T} |$ with $q_\tau > 0\}$.

(2) If $S \neq \emptyset$,

$$(\bar{\mathcal{T}}, \boldsymbol{u}_{\bar{\mathcal{T}}}, \mathsf{A}_{\bar{\mathcal{T}}}, \mathsf{L}_{\bar{\mathcal{T}}}) \longleftarrow \text{up-bisect-tets}\,(\mathcal{T}, S, \boldsymbol{u}_{\mathcal{T}}, \mathsf{A}_{\mathcal{T}}, \mathsf{L}_{\mathcal{T}})$$

$$\left(\tilde{\mathcal{T}}, \boldsymbol{u}_{\tilde{\mathcal{T}}}, \mathsf{A}_{\tilde{\mathcal{T}}}, \mathsf{L}_{\tilde{\mathcal{T}}}\right) \longleftarrow \text{up-refine-to-conformity}\,(\bar{\mathcal{T}}, \boldsymbol{u}_{\bar{\mathcal{T}}}, \mathsf{A}_{\bar{\mathcal{T}}}, \mathsf{L}_{\bar{\mathcal{T}}})$$

$$(\mathcal{T}', u_{\mathcal{T}'}, \mathsf{A}_{\mathcal{T}'}, \mathsf{L}_{\mathcal{T}'}) \longleftarrow \text{adapt-mesh}\left(\tilde{\mathcal{T}}, \boldsymbol{u}_{\tilde{\mathcal{T}}}, \mathsf{A}_{\tilde{\mathcal{T}}}, \mathsf{L}_{\tilde{\mathcal{T}}}\right).$$

(3) Else, $\mathcal{T}' = \mathcal{T}, u_{\mathcal{T}'} = u_{\mathcal{T}'}, \mathsf{A}_{\mathcal{T}'} = \mathsf{A}_{\mathcal{T}}, \mathsf{L}_{\mathcal{T}'} = \mathsf{L}_{\mathcal{T}}$.

We are now ready to describe algorithm linear-bvp for solving the boundary value problem (1)–(3) of Chapter 2. This takes a marked mesh $\mathcal{T}_0$ and an initial guess $\boldsymbol{u}^i_{(0)}$ as input, and it is assumed that $\mathcal{T}_0$ has no flagged tetrahedra. The algorithm cycles between calls to solve-MG_k and adapt-mesh until the finite element solution $\hat{\boldsymbol{u}}_{(k)}$ is close to the exact solution $\hat{u}$ of the boundary value problem as determined by some tolerance parameter.

Algorithm 5.1.6

$$\boldsymbol{u}^f \longleftarrow \text{linear-bvp}\left(\mathcal{T}_0, \boldsymbol{u}^i_{(0)}\right)$$

Input: Conformingly-marked mesh $\mathcal{T}_0$ with no flagged tetrahedra and initial guess $\boldsymbol{u}^i_{(0)}$.

Output: Approximate solution $\boldsymbol{u}^f$ for (1)–(3).

Set $k = 0$, and compute A_0, L_0.

(1) $\hat{\boldsymbol{u}}_{(k)} \longleftarrow$ solve-$MG_k \left(\mathsf{A}_k, \mathsf{L}_k, \boldsymbol{u}^i_{(k)}\right)$.

(3) If "stopping criterion" is met, $\boldsymbol{u}^f = \hat{\boldsymbol{u}}_{(k)}$.

(2) Else, Compute η_τ and q_τ for all $\tau \in \mathcal{T}_k$ using $\hat{\boldsymbol{u}}_k$,

$$\left(\mathcal{T}_{k+1}, \boldsymbol{u}^i_{(k+1)}, \mathsf{A}_{k+1}, \mathsf{L}_{k+1}\right) \longleftarrow \text{adapt-mesh}\left(\mathcal{T}_k, \hat{\boldsymbol{u}}_{(k)}, \mathsf{A}_k, \mathsf{L}_k\right)$$

$k \longleftarrow k+1$ and go to (1).

Note that in view of the assumption that the mesh $\mathcal{T}_0$ has no flagged tetrahedra and the equivalence of adapt-mesh and local-refine in terms of their mesh refinement features as indicated above, the calls to adapt-mesh at Step 2 of linear-bvp terminate. Note also that we use a generic form of the the stopping criterion for linear-bvp in Step 2. In practice, a target number of nodes may be used as a stopping criterion. Recently, Becker et.al. [9] have studied the stopping criterion for multigrid solvers coupled with adaptive techniques for finite elements. The incorporation of such criteria in AMG3DP1 will be a subject of further study. Linear-bvp is a full multigrid algorithm since it uses the interpolated value of the finite element solution from a coarse mesh as the initial guess for the V-cycle iteration on the fine mesh.

AMG3DP1 is set up to be able to handle domains with spherical boundaries (the boundary value problem to be solved for finding initial data for the collision of black holes is posed on such a domain). Assume that the spherical boundaries of the computational domain are approximated by the faces of tetrahedra in the coarse mesh $\mathcal{T}_0$. Let

$$\mathcal{E}\left(\mathcal{T}_k\right) = \mathcal{E}_\Omega\left(\mathcal{T}_k\right) \cup \mathcal{E}_\Gamma\left(\mathcal{T}_k\right)$$

be the decomposition of the edges associated with $\mathcal{T}_k$ into its interior and boundary parts. During the course of creating $\mathcal{T}_{k+1}$ from $\mathcal{T}_k$ using adapt-mesh, any new nodes that are introduced as the midpoint of an edge $e \in \mathcal{E}_\Gamma\left(\mathcal{T}_k\right)$ is projected onto the surface of the corresponding sphere. This ensures that the spherical boundary Γ is

better approximated by the computational domain as linear-bvp progresses.

5.2 An algorithm for semilinear boundary value problems

Consider semilinear elliptic boundary value problems of the following form:

$$\begin{aligned} \mathcal{L}u = -\operatorname{div}\left[\underset{\approx}{\mathcal{A}}(\underset{\sim}{x})\operatorname{grad}_{\sim} u\right] + f(\underset{\sim}{x}, u) = 0 \text{ in } \Omega, \\ u = g_D(\underset{\sim}{x}) \text{ on } \Gamma_D, \\ \underset{\approx}{\mathcal{A}}(\underset{\sim}{x})\operatorname{grad}_{\sim} u \cdot \underset{\sim}{n} + c(\underset{\sim}{x})u = g_N(\underset{\sim}{x}) \text{ on } \Gamma_N. \end{aligned} \tag{24}$$

The assumptions on the domain, boundary, and coefficients $\underset{\approx}{\mathcal{A}}$, g_D, c, and g_N are the same as in Chapter 2 and we assume $\frac{\partial f}{\partial u} \in C^0\left(\bar{\Omega}\right)$. Define $a_{\mathcal{L}}\left(\cdot\,,\cdot\right) : H^1(\Omega) \times H^1(\Omega) \to \mathbb{R}$ by

$$a_{\mathcal{L}}\left(u, v\right) = \int_\Omega \left[\left(\underset{\approx}{\mathcal{A}}\operatorname{grad}_{\sim} u\right) \cdot \operatorname{grad}_{\sim} v + f(\underset{\sim}{x}, u)v\right] dx + \int_{\Gamma_N} \left(cuv - g_N v\right) ds.$$

The weak problem associated with (24) is:

Find $\hat{u} \in V_D$ such that

$$a_{\mathcal{L}}(\hat{u}, v) = 0 \text{ for all } v \in V_0. \tag{25}$$

A multilevel Newton scheme is to generate a sequence $\left\{\hat{u}_{(k)}\right\}$ of approximations to $\hat{u}$. The quantities $\hat{u}_{(k)}$ are approximations to solutions of problems similar to (25) posed on the finite element spaces associated with a nested sequence of conforming meshes of Ω. Considering the sequence of meshes $\{\mathcal{T}_k\}_{k=0}^M$ and the associated spaces introduced in Chapter 4, the scheme for obtaining $\hat{u}_{(k)}$ is the following:

Start with an initial guess $u_{(k)}^0 \in V_k^D$ and compute $u_{(k)}^{j+1} \in V_k^D$ from $u_{(k)}^j \in V_k^D$ such that

$$b_{\mathcal{L}}\left(u_{(k)}^j; u_{(k)}^{j+1} - u_{(k)}^j, v\right) = -a_{\mathcal{L}}\left(u_{(k)}^j, v\right) \text{ for all } v \in V_k, \tag{26}$$

where $b_{\mathcal{L}}(u; \cdot\,, \cdot) : H^1(\Omega) \times H^1(\Omega) \to \mathbb{R}$ is a bilinear form given by

$$b_{\mathcal{L}}(w; u, v) = \int_\Omega \left[\left(\underset{\approx}{\mathcal{A}}\operatorname{grad}_{\sim} u\right) \cdot \operatorname{grad}_{\sim} v + \frac{\partial f}{\partial w}(\underset{\sim}{x}, w)uv\right] dx + \int_{\Gamma_N} cuv\, ds.$$

The sequence $\left\{u^j_{(k)}\right\}$ converges to $\hat{u}_{(k)}$. Note that (26) corresponds to one iteration of Newton's method for solving the semilinear problem on $\mathcal{T}_k$. In particular, (26) is the discrete problem on $\mathcal{T}_k$ associated with the linear boundary value problem

$$
\begin{gathered}
-\operatorname{div}\left(\underset{\approx}{\mathcal{A}}(\underset{\sim}{x})\operatorname{grad}_{\sim} u^{j+1}_{(k)}\right) + \frac{\partial f}{\partial u^j_{(k)}} u^{j+1}_{(k)} = \frac{\partial f}{\partial u^j_{(k)}} u^j_{(k)} - f \text{ in } \Omega, \\
u^{j+1}_{(k)} = g_D(\underset{\sim}{x}) \text{ on } \Gamma_D, \\
\underset{\approx}{\mathcal{A}}(\underset{\sim}{x})\operatorname{grad}_{\sim} u^{j+1}_{(k)} \cdot \underset{\sim}{n} + c(\underset{\sim}{x}) u^{j+1}_{(k)} = g_N(\underset{\sim}{x}) \text{ on } \Gamma_N,
\end{gathered}
$$

where f and $\frac{\partial f}{\partial u^j_{(k)}}$ are evaluated at $(\underset{\sim}{x}, u^j_{(k)})$. Problems like (26) will be solved using the multilevel scheme of Chapter 4 to create the sequence $u^j_{(k)}$ until two successive entries in the sequence are close enough. For a mesh $\mathcal{T}_k$, the last $u^j_{(k)}$ is used in conjunction with the adaptive mesh refinement algorithm described in the previous section (adapt-mesh) to obtain the next mesh $\mathcal{T}_{k+1}$ and the solution is interpolated onto the associated finite element space and used as the initial guess $u^0_{(k+1)}$. Bank and Rose [7] have analyzed a class of multilevel methods for nonlinear finite element equations that include the above scheme as a particular case. A key assumption in their analysis is that each iteration of multilevel method for solving the linear boundary value problem reduces the error by a fixed factor independent of the level k, and this is true for the multilevel solver introduced in Chapter 4. They show that the number of Newton's iterations required is asymptotically equal to one, and that for linear finite elements the optimal order of convergence is 1/3. Algorithm *semilin-bvp* takes a marked conforming mesh with no flagged tetrahedra and an initial guess as input and outputs an approximate solution for the semilinear boundary value problem (24). For describing semilin-bvp, we will need algorithm *adapt-solution* which is similar to adapt-mesh except that it does not have the nodal stiffness matrix and nodal right hand side as either input or output. This actually requires another set of algorithms similar to the ones preceding adapt-mesh in the previous section, but we assume the existence of these without giving them explicit

names. Let $\boldsymbol{u}^j_{(k)}$ be the vector of coefficients associated with $u^j_{(k)}$ in terms of the nodal basis for $\mathcal{T}_k$.

Algorithm 5.2.1

$\boldsymbol{u}^f \longleftarrow$ semilin-bvp $\left(\mathcal{T}_0, \boldsymbol{u}^i_{(0)}\right)$

Input: Marked conforming mesh $\mathcal{T}_0$ with no flagged tetrahedra and initial guess $\boldsymbol{u}^i_{(0)}$.

Output: Approximate solution $\boldsymbol{u}^f$ for (24).

Set $k = 0$, $j = 0$, and $\boldsymbol{u}^j_{(k)} = \boldsymbol{u}^i_{(0)}$.

(1) If $\left\| \boldsymbol{u}^j_{(k)} - \pi_k \hat{u} \right\| > \text{tolerance}_1$,

Create $\mathsf{A}^j_{(k)}$ and $\mathsf{L}^j_{(k)}$ associated with (26)

$\boldsymbol{u}^{j+1}_{(k)} \longleftarrow$ solve-MG_k $\left(\mathsf{A}^j_{(k)}, \mathsf{L}^j_{(k)}, \boldsymbol{u}^j_{(k)}\right)$.

(2) If $\left\| \boldsymbol{u}^{j+1}_{(k)} - \boldsymbol{u}^j_{(k)} \right\| > \text{tolerance}_2$,

$j \longleftarrow j + 1$, go to (1).

(3) Else, compute η_τ and q_τ for all $\tau \in \mathcal{T}_k$ using $\boldsymbol{u}^{j+1}_{(k)}$

$\left(\mathcal{T}_{k+1}, \boldsymbol{u}^0_{(k+1)}\right) \longleftarrow$ adapt-solution $\left(\mathcal{T}_k, \boldsymbol{u}^{j+1}_{(k)}\right)$

$k \longleftarrow k + 1$, $j \longleftarrow 0$, go to (1).

(4) Else, $\boldsymbol{u}^f = \boldsymbol{u}^j_{(k)}$.

Since semilin-bvp involves only one Newton's iteration per level in the asymptotic range, it follows that the number of operations needed to solve a semilinear problem is equal to that for solving a linear problem on the finest mesh together with the operation count leading to the formulation of this problem. Since the solution of a linear problem by solve-MG_k involves $O(N_k)$ operations, as do the steps involving the construction of the stiffness matrix and right hand side for the linear problem, it follows that semilin-bvp on $\mathcal{T}_M$ involves $O(N_M)$ operations. The success of semilin-bvp depends on the choice of a good initial guess. For the applications that we consider, we assume the existence of such an initial guess.

5.3 Numerical examples

In this section we show by considering various numerical examples that linear-bvp converges with optimal order 1/3 in the H^1 norm. We also show optimal convergence rates for a semilinear problem solved using semilin-bvp. All the problems considered in this section are posed on the unit cube $\Omega = [0,1]^3$ and are such that the exact solution is known. Problem 3 has mixed boundary conditions (both Neumann and Dirichlet) while all other problems have Dirichlet boundary conditions only. Problem 2 is taken from a model problem considered by Duval et. al. [20], and Problems 1 and 5 are adapted to three dimensions from a collection of elliptic boundary value problems in $\mathbb{R}^2$ given in [32]. The exact solution for Problem 3 has a boundary layer behavior, while that for Problem 4 has singular derivatives. Problems 1–4 are linear problems and Problem 5 is semilinear. All the problems except Problem 2 are such that the solution or the gradient changes rapidly in some portion of Ω. The adaptive mesh refinement in AMG3DP1 is supposed to capture this behavior and refine the meshes locally in such regions. A more detailed description of each problem is given later.

The coarse mesh $\mathcal{T}_0$ is taken to be a uniform mesh having 96 tetrahedra and 35 nodes for all the problems. For the class of problems considered in this thesis $\|v\|_1$ is equivalent to the energy norm defined by $\|v\|_E^2 = a(v,v)$. This energy norm of the error is used to show convergence rates for the problems considered here. It is also known (with reference to Theorem 2.2.1) that

$$\|\hat{u} - u_{\mathcal{T}}\|_0 \leq C\,[h(\mathcal{T})]^2\,\|\hat{u}\|_2$$

under some additional assumptions on the boundary value problem (1)–(3). Thus, AMG3DP1 has optimal order 2/3 convergence if the L^2 norm of the errors are considered. For each problem in this section, we also report the L^2 norm of the errors. All integrals involved in the computation of errors, as well as those in

computing stiffness matrices and right hand sides, are evaluated using one-point quadrature rules like the ones used in the evaluation of η_τ in Chapter 2.

Numerical experiments suggest that two iterations of the V-cycle (two repetitions of Steps 2–4 of solve-MG_k) are sufficient for its convergence. Increasing the number of iterations does not change the finite element solution by an amount that is significant in the computation of the errors. Thus, we use two iterations of the V-cycle in all of our numerical tests. In other words, for our numerical tests, Step 1 of solve-MG_k is replaced by the statement "For $n = 0, 1$". We use an exact solver using LU-decomposition from LAPACK for the exact solve in Step 3 of solve-MG_k.

As we have mentioned before, a target number of nodes may be used as a stopping criterion for linear-bvp. Since we are interested in studying the convergence rates for the errors, the finite element solution is computed for meshes that are sufficiently fine so as to capture the essential features of the solution. All the calculations are done using a Fortran 77 code using single precision on a DEC Alpha workstation (DEC 3000 model 500 with a 150 MHz DEC Alpha processor). For the numerical tests, we compute the finite element solution using the highest possible number of nodes (this replaces the stopping criteria in Step 2 of linear-bvp and Step 1 of semilin-bvp). The fact that adapt-mesh approximately doubles the number of nodes in each step and that this process is problem dependent introduces a range of values for the maximum number of nodes for which the solution is computed for each problem. This number varies between $40,000$ and $94,000$. The criterion for the number of Newton's iterations in Step 2 of semilin-bvp is chosen quite conservatively. In fact, we choose this to be

$$\left| \left(\boldsymbol{u}_{(k)}^{j+1} \right)_\nu - \left(\boldsymbol{u}_{(k)}^{j} \right)_\nu \right| < 10^{-7} \text{ for all } \nu \in \mathcal{N}\left(\mathcal{T}_k\right).$$

Even this conservative choice requires at most two Newton's iterations per level for Problem 5 (and also for most problems that we have tried using semilin-bvp).

Remark 5.3.1 AMG3DP1 can handle much more complicated domains including domains with spherical boundaries. Elliptic semilinear boundary value problems on domains with spherical boundaries having Robin boundary conditions arise naturally in the course of finding solutions to the initial data problem for black holes. There are some simple situations where the exact solution of such boundary value problems are known, and some other cases where quantities of physical interest (for example, total energy) have been reported in the literature. We will show at the end of the next chapter by considering these problems that AMG3DP1 gives satisfactory error rates for the first class of problems, and that the physically relevant quantities obtained by using AMG3DP1 compare very well with published results. This will demonstrate that AMG3DP1 can handle spherical boundaries and non Dirichlet boundary conditions efficiently.

For Problems 1–5 we report the energy and L^2 norms of the errors obtained using adaptive refinement and compare them with the corresponding values using a uniform refinement of meshes. Given any conformingly-marked mesh $\mathcal{T}$ with no flagged tetrahedra, $\mathcal{T}'$ is said to be an uniform refinement of $\mathcal{T}$ if $\mathcal{T}' \longleftarrow$ bisect-all^3 $(\mathcal{T})$. Recall that $\mathcal{T}'$ will be conforming (by Proposition 3.3.5). The figures showing the errors are plotted on a log-log scale and use the following conventions:

- The x-axis shows the number $N^{-1/3}$ where N is the number of nodes for an arbitrary mesh. Note that this quantity is an indicator of the mesh size. So, we expect to see optimal rates of 1 for the energy error and 2 for the L^2 error.
- The y-axis shows the absolute value of the error.
- We use a "$\circ$" to indicate the energy norm, and "$\times$" to indicate the L^2 norm of the error using adaptive mesh refinement. The corresponding quantities using uniform refinement are indicated by "straight" (respectively "dotted") lines joining the $\circ$'s (respectively $\times$'s).

  The figures for the errors also have lines of slope 1 and 2 drawn so that the

actual convergence rates can be compared with these. Views of the adaptively refined meshes and isovalues of the solution are also shown for each problem. These are shown on the same figure whenever possible, but in some cases separate figures are provided as they give a clearer idea of the details. Also, for some problems we show a cut through the adapted mesh (the plane of the cut is given in the caption for the picture in this situation). All figures and pictures are titled with the problem number and a short description of its content. The pictures of the meshes are generated using the mesh viewing routines in the finite element package MODULEF. The adaptively refined meshes are not shown for the finest mesh for any problem as this does not render a clear image owing to the large number of elements in the mesh. However, the adaptive nature of the meshes are evident from the pictures provided. We next present the problems together with their salient features in detail. A short description of the relevant results are also presented. Since the absolute energy and L^2 norms of the error are reported for the convergence rates, we also report the *relative percent energy error* for each problem. This is the quantity $100 \times (\|\hat{u} - u_{\mathcal{T}}\|_E / \|\hat{u}\|_E)$. The relative percent L^2 error can be defined similarly using the L^2 norm. The values of this are not reported explicitly for each problem, but are approximately $1/10$ that of the relative percent energy error for all the cases.

Problem 1. Choose $\underset{\approx}{\mathcal{A}} = \underset{\approx}{I}$, $b \equiv 0$, $\Gamma_N = \emptyset$, and $g_D = u_{\text{ex}}$ in (1)–(3) where

$$u_{\text{ex}} = \left(x^2 - x\right)\left(y^2 - y\right)\left(z^2 - z\right)\, e^{-\alpha\left[(x-a)^2+(y-b)^2+(z-c)^2\right]}.$$

Thus, we solve $-\Delta u = f$ on Ω where f is chosen such that u_{ex} satisfies the equation. The exact solution is smooth but is strongly peaked at $(a, b, c) \in \mathbb{R}^3$ for large values of α. For our numerical test we use the parameters $(a, b, c) = (0.25, 0.25, 0.25)$ and $\alpha = 100$. The effective local refinement around the point $(0.25, 0.25, 0.25)$ can be clearly seen from Figure 12 which shows a cut through the domain along the plane $x = 1/4$. This is an intermediate adapted mesh having $2,116$ nodes and $11,418$

tetrahedra. Figure 11 shows the absolute errors in the energy and L^2 norms for both adaptive and uniform mesh refinement. It clearly indicates the improvement in the errors provided by adaptive refinement in comparison with uniform refinement. The finest mesh using uniform refinement has $68,705$ nodes and the relative percent energy error is approximately 15.85%, while the maximum number of nodes using adaptive refinement are $62,738$ and the corresponding relative percent energy error is 4.95%.

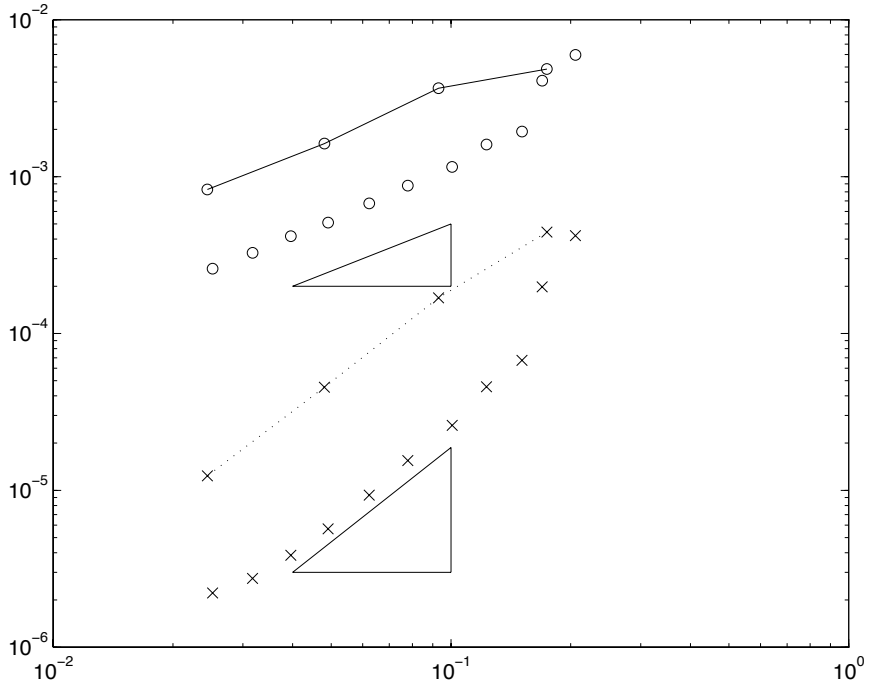

Figure 11. Problem 1: $\alpha = 100$, $(a,b,c) = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4})$; absolute energy and L^2 errors.

Problem 2. Choose $b \equiv 0$, $\Gamma_N = \emptyset$,

$$g_D = u_{\text{ex}} = e^{\lambda x + \mu y + \sigma z}, \text{ and } \underset{\approx}{\mathcal{A}} = \begin{bmatrix} 1+x^2 & 0 & \sin(x) \\ 0 & 1+y^2 & 0 \\ \sin(x) & 0 & 1+z^2 \end{bmatrix}$$

in (1)–(3). Thus, we solve $-\operatorname{div}(\underset{\approx}{\mathcal{A}} \underset{\sim}{\operatorname{grad}}\, u) = f$ on Ω where f is chosen such that u_{ex} satisfies the equation. We take $\lambda = \mu = 3$ and $\sigma = 1$ for our numerical test. Note that unlike Problem 1 u_{ex} does not change rapidly in any particular portion of the domain. Thus, the norms of the error for adaptive and uniform refinement should

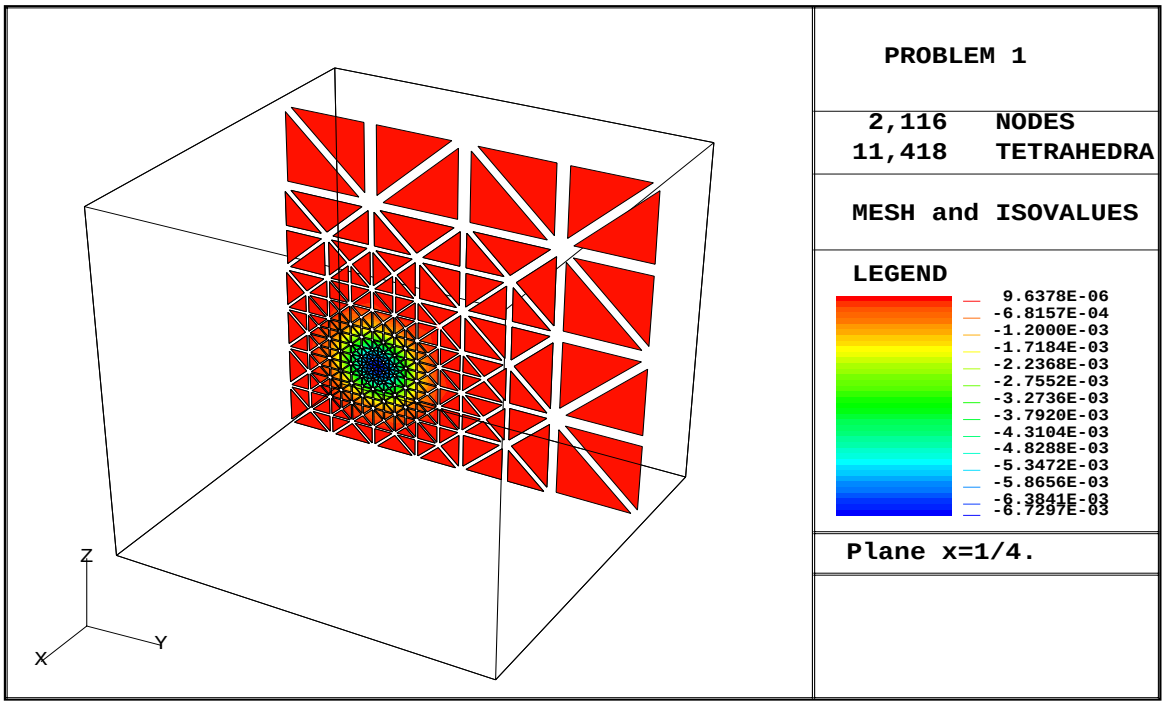

Figure 12. Problem 1: $\alpha = 100$, $(a, b, c) = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4})$.

be close as is indicated in Figure 13 showing the absolute errors in the energy and L^2 norms. The errors show an optimal rate of convergence. The relatively large values of the absolute errors are not a cause of concern as is indicated by the corresponding percentage relative energy errors. For uniform refinement the finest mesh has $68,705$ nodes and a relative percent energy error of 2.85% while the finest mesh using adaptive refinement has $54,956$ nodes and relative percent energy error of 1.86%. Even though there is not much difference between the errors produced by uniform and adaptive refinement in this case, the meshes produced by adaptive refinement capture the behavior of u_{ex} quite well (Figure 14). Note that

$$\underset{\sim}{\text{grad}}\, u_{\text{ex}} = \left(3e^{3x+3y+z}, 3e^{3x+3y+z}, e^{3x+3y+z}\right) \text{ and } |\underset{\sim}{\text{grad}}\, u_{\text{ex}}|^2 = 10e^{3x+3y+z}.$$

Thus, on any plane parallel to the xy plane both $|u_{\text{ex}}|$ and $|\underset{\sim}{\text{grad}}\, u_{\text{ex}}|$ behave like e^{3x+3y}, while for planes parallel to the xz or yz planes their behavior is determined by e^{3x+z} and e^{3y+z} respectively. This qualitative difference in behavior of the exact solution is captured well by the adaptive mesh refinement procedure as seen in

Figure 14 showing the $x = 1$, $y = 1$, and $z = 1$ plane of Ω for the intermediate adapted mesh having $19,448$ tetrahedra and $4,102$ nodes.

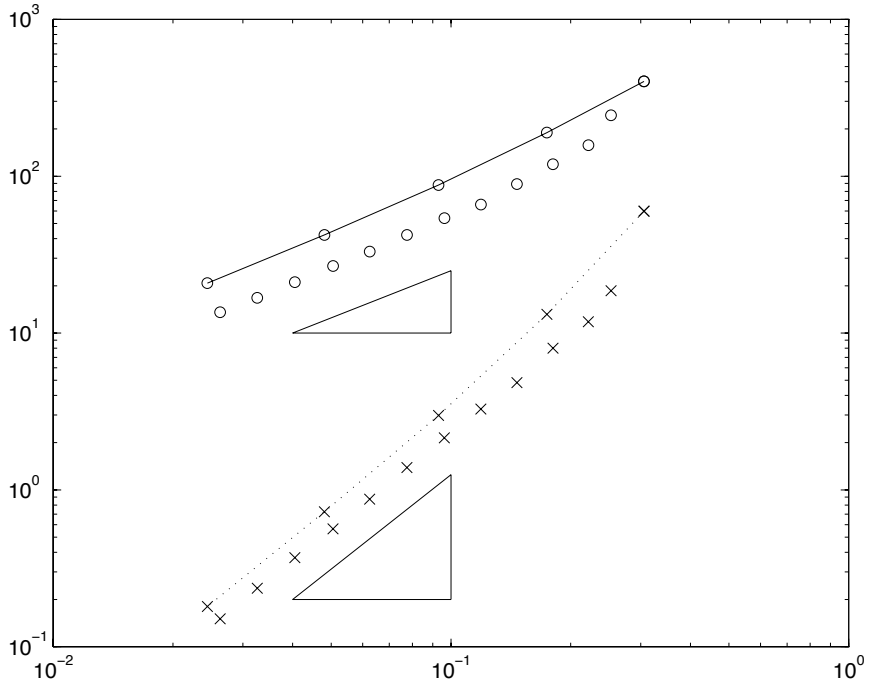

Figure 13. Problem 2: $\lambda = \mu = 3$, $\sigma = 1$; absolute energy and L^2 errors.

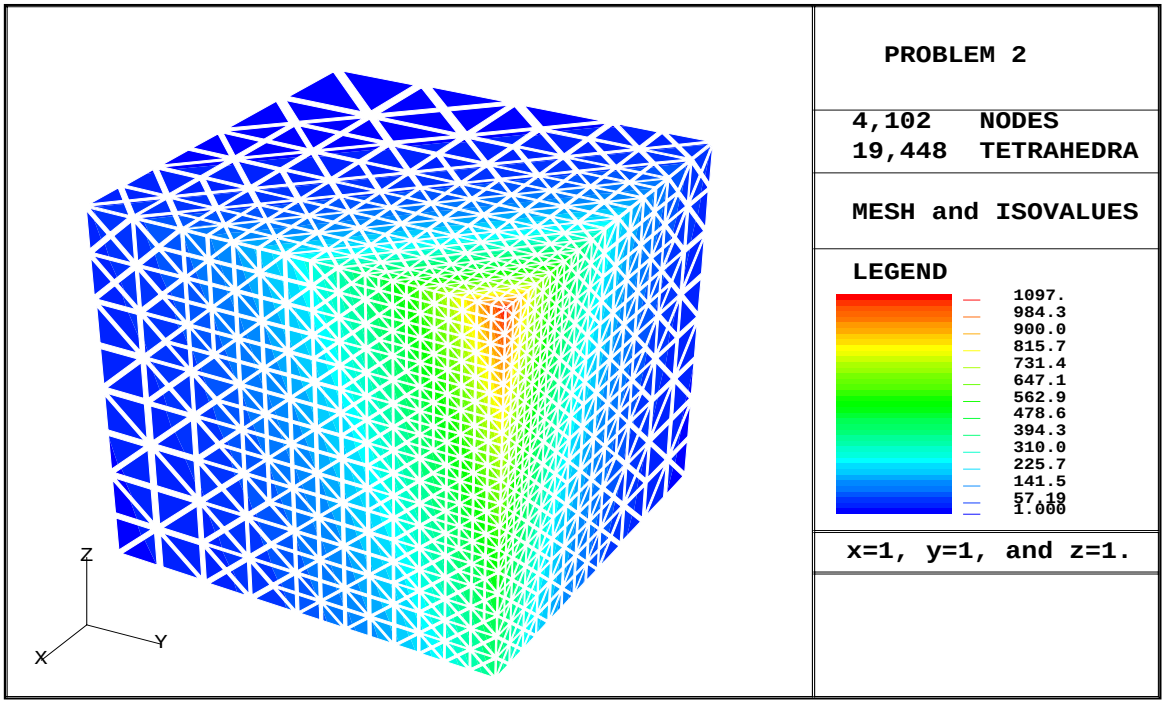

Figure 14. Problem 2: $\lambda = \mu = 3$, $\sigma = 1$; solution.

Problem 3. Choose $\underset{\approx}{\mathcal{A}} = \epsilon^2 \underset{\approx}{I}$, $b = 1$, $\Gamma_N = \{(x,y,z)|y,z = 0 \text{ or } 1\}$, $c = 0$, $g_N = 0$, and $g_D = u_{\text{ex}}$ in (1)–(3) where $u_{\text{ex}} = 1 - e^{-x/\epsilon}$. Thus, we solve $-\epsilon^2 \Delta u + u = f$ on Ω where f is chosen such that u_{ex} satisfies the equation ($f = 1$). We report the absolute errors in the energy and L^2 norms for $\epsilon = 0.05$ in Figure 15. This problem is known to have a boundary layer near $x = 0$. We know that $u_{ex} = 0$ for $x = 0$, and it changes rapidly in the x direction. In particular, for $\epsilon = 0.05$,

$$u_{ex}(x) = 0.99 \times u_{ex}(1) \text{ for } x = 0.23.$$

The adaptive mesh refinement captures this behavior quite well as seen in Figure 16 which shows the isovalues of the solution as well as the mesh on the faces $x = 0$, $y = 1$, and $z = 1$ for an intermediate adapted mesh having $11,303$ nodes and $54,780$ tetrahedra. The local mesh adaptivity in the region having the boundary layer is quite clear. The absolute errors in the energy and L^2 norms for both adaptive and uniform refinement are shown in Figure 15. The finest mesh using uniform refinement has $68,705$ nodes and the corresponding relative percent energy error is 1.87% while the finest mesh using adaptive refinement has $93,792$ nodes and the corresponding relative percent energy error is 0.74% (the relative percent energy error is 1.71% using $11,303$ nodes using adaptive refinement).

Problem 4. Choose $\underset{\approx}{\mathcal{A}} = \underset{\approx}{I}$, $b \equiv 0$, $\Gamma_N = \emptyset$, and $g_D = u_{\text{ex}}$ in (1)–(3) where $u_{\text{ex}} = (x^2+y^2+z^2)^{\alpha/2}$. Thus, we solve $-\Delta u = f$ on Ω where f is chosen such that u_{ex} satisfies the equation. A direct evaluation yields that $u_{\text{ex}} \in H^1 \setminus H^2$ if $-\frac{1}{2} < \alpha < \frac{1}{2}$. We performed tests with $0 < \alpha < \frac{1}{2}$ so that the solution had singular derivatives at $(0,0,0)$. The absolute errors (using both uniform and adaptive refinement) in the energy and L^2 norms for $\alpha = 0.1$ are shown in Figure 17. It is clear that the adaptive procedure shows optimal orders of convergence as well as substantial improvement over uniform refinement. In particular, the finest mesh using uniform refinement has $68,705$ nodes and the corresponding relative percent energy error is 75.36% while the finest mesh using adaptive refinement has $40,633$ nodes and the

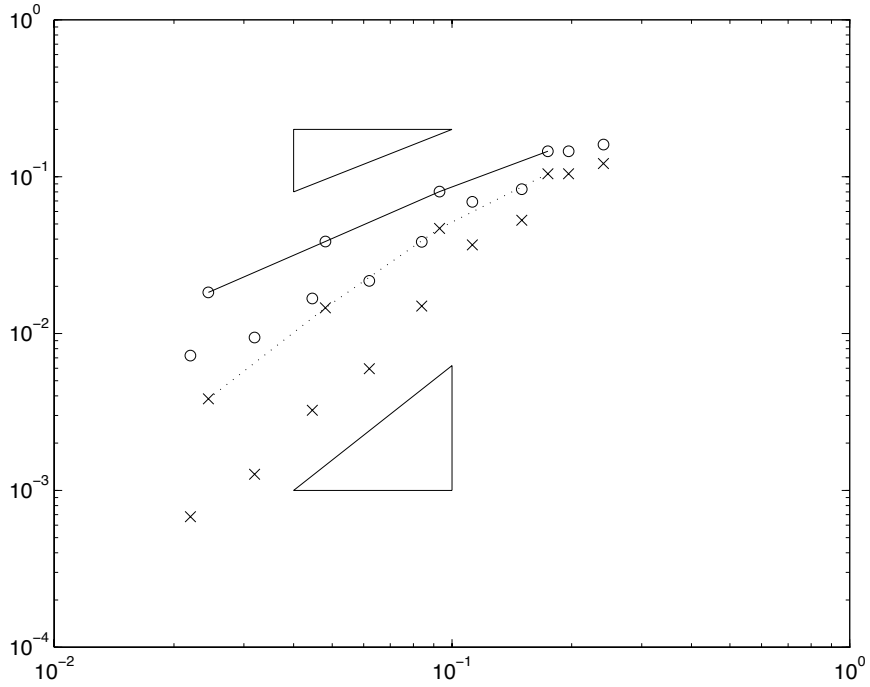

Figure 15. Problem 3: $\epsilon = 0.05$; absolute energy and L^2 errors.

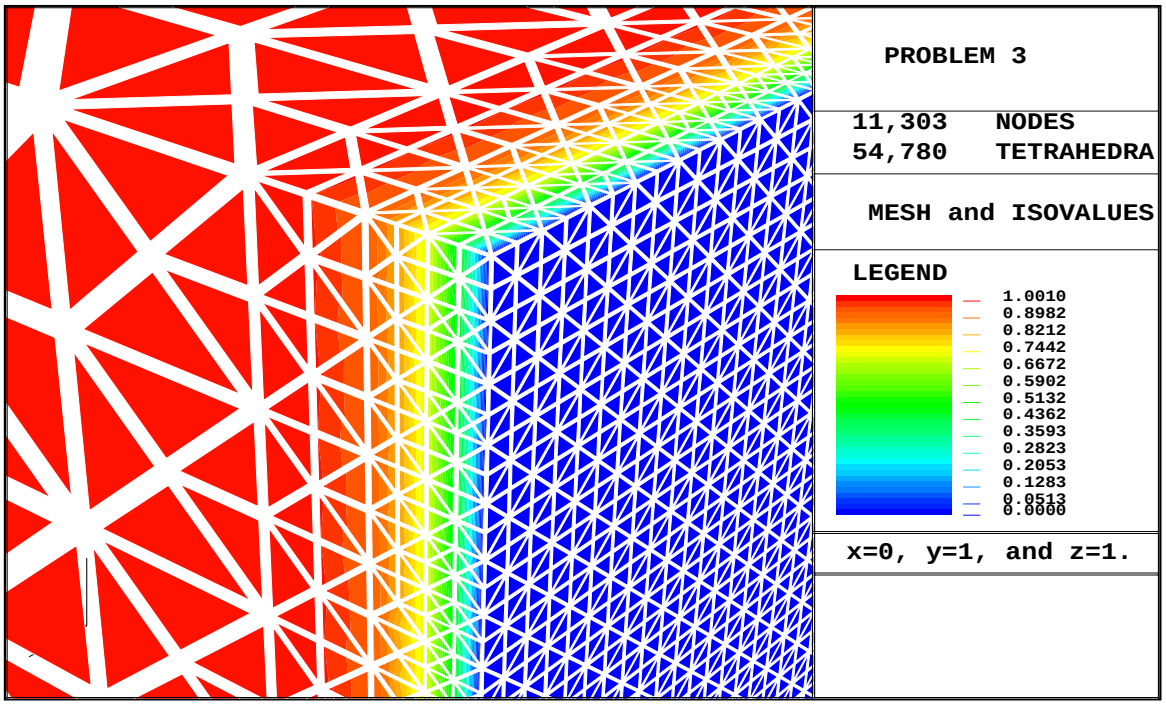

Figure 16. Problem 3: $\epsilon = 0.05$; solution.

corresponding relative percent energy error is 3.59%. The exact solution is smooth on $\Omega \setminus (0,0,0)$ and has a singular derivative at $(0,0,0)$. Figure 18 shows the isovalues

of the solution on the planes $x = 0, y = 0$, and $z = 0$ for an intermediate adapted mesh having $9,870$ tetrahedra and $2,171$ nodes, while Figure 19 shows the mesh on the same faces. The local mesh adaptivity near $(0,0,0)$ is clear.

Remark 5.3.2 It can be verified via a direct computation that $u_{\text{ex}} \in H^s$ for $s < 1.6$ and $\notin H^s$ for $s > 1.6$. The errors obtained using uniform refinement show convergence rates of 0.6 and 1.6 in the energy and L2 norms as expected (the solid and dotted lines in Figure 17).

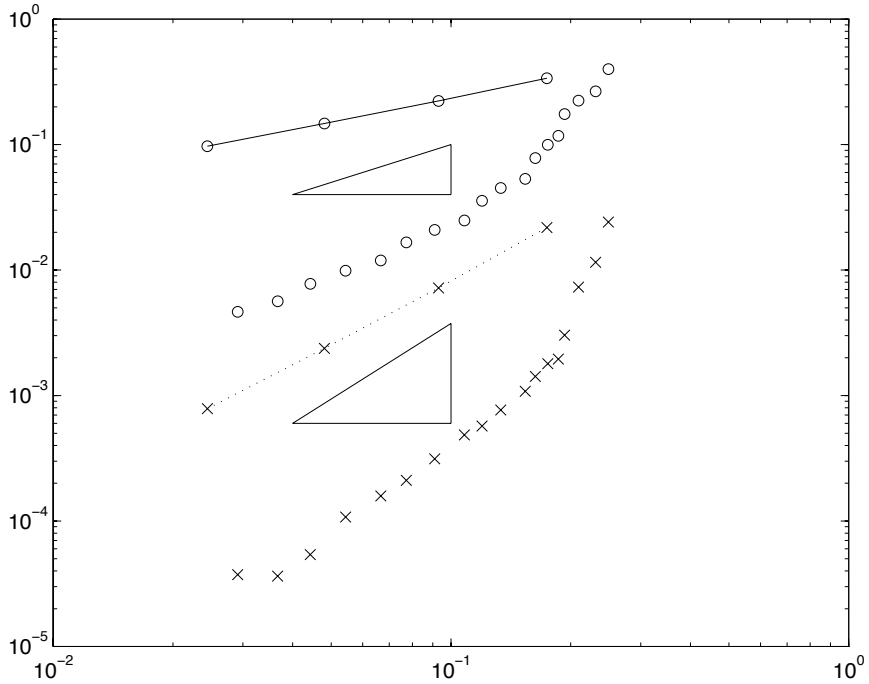

Figure 17. Problem 4: $\alpha = 0.1$; absolute energy and L^2 errors.

Problem 5. Choose $\underset{\approx}{\mathcal{A}} = \underset{\approx}{I}$, $f(\underset{\sim}{x}, u) = u^3 - h$, $\Gamma_N = \emptyset$, and $g_D = u_{\text{ex}} = (xyz)^{\alpha/2}$ in (24) with $\alpha = 20$. The resulting semilinear problem is $-\Delta u + u^3 = h$ and it is solved using semilin-bvp of the previous section. The energy and L^2 errors are reported using the approximate solution of the final linear problem solved on a mesh. Figure 20 shows the absolute energy and L^2 errors. The optimal order of convergence for the method and the improvement provided by adaptive mesh refinement compared to uniform refinement can be clearly seen. The finest mesh

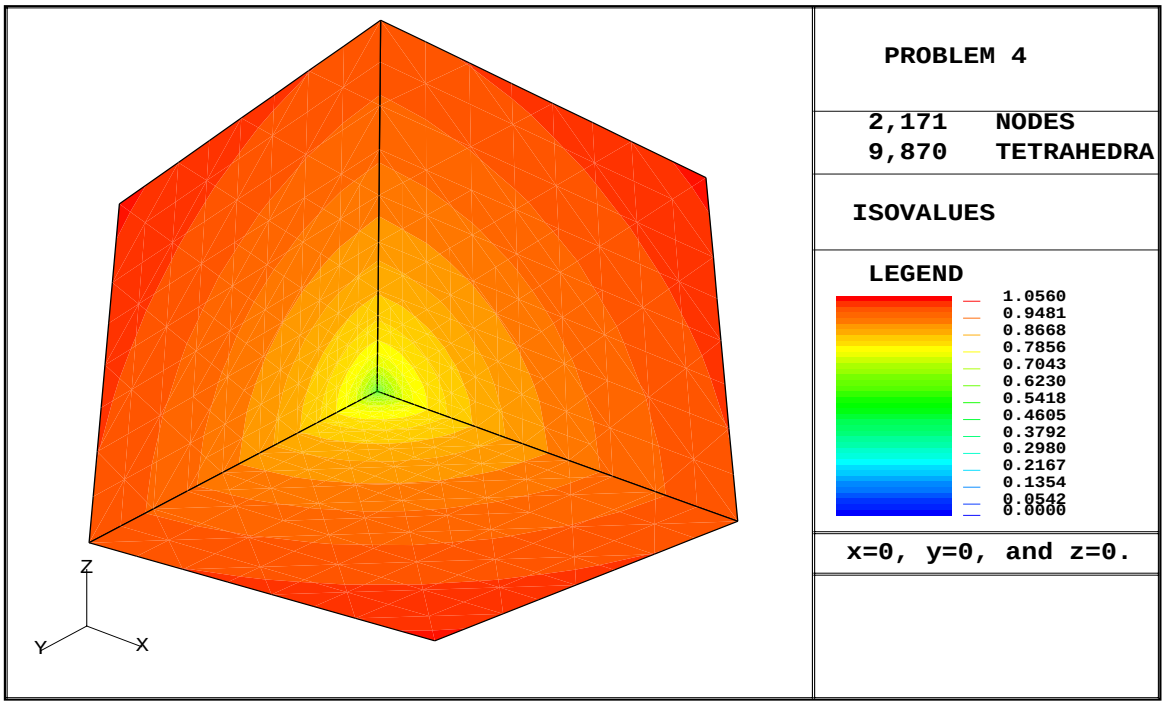

Figure 18. Problem 4: $\alpha = 0.1$; solution.

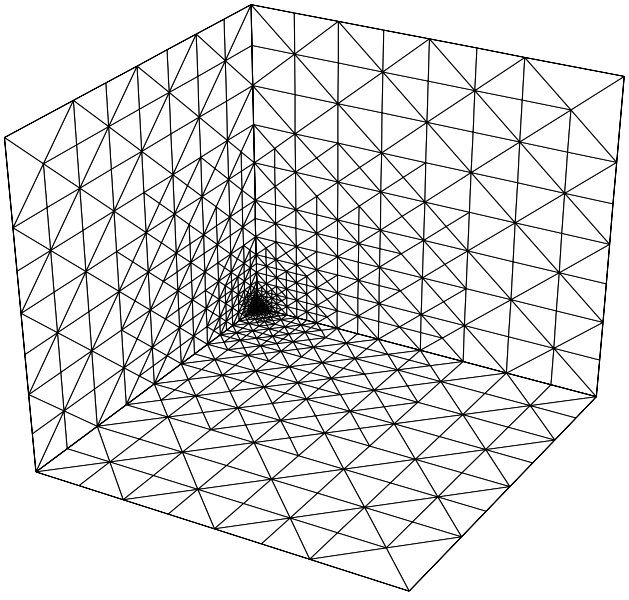

Figure 19. Problem 4: $\alpha = 0.1$; adapted mesh having $9,870$ tetrahedra and $2,171$ nodes showing the $x = 0$, $y = 0$, and $z = 0$ planes.

using uniform refinement has $68,705$ nodes and a corresponding relative percent

energy error of 14.24% while the finest mesh using adaptive refinement has 59,323 nodes and a relative percent energy error of 2.3%. The exact solution and its derivatives are smooth and the large value of α implies that most of the variation of the exact solution occurs in the vicinity of $(1,1,1)$. A view of the mesh on the $x = 1$, $y = 1$, and $z = 1$ faces of Ω for an intermediate adapted mesh having 5,988 tetrahedra and 1,321 nodes clearly shows the local mesh refinement near the corner $(1,1,1)$ (see Figure 21). The back view for the mesh is not shown since it is relatively uninteresting. In fact, it shows a uniform mesh (as is to be expected since the solution and its derivatives are zero on the faces $x = 0$, $y = 0$, and $z = 0$). Using the same intermediate adapted mesh, the isovalues of the solution are shown in Figure 22. The Newton's iteration converged in two steps for all meshes in the asymptotic range and in one step for the two finest meshes considered.

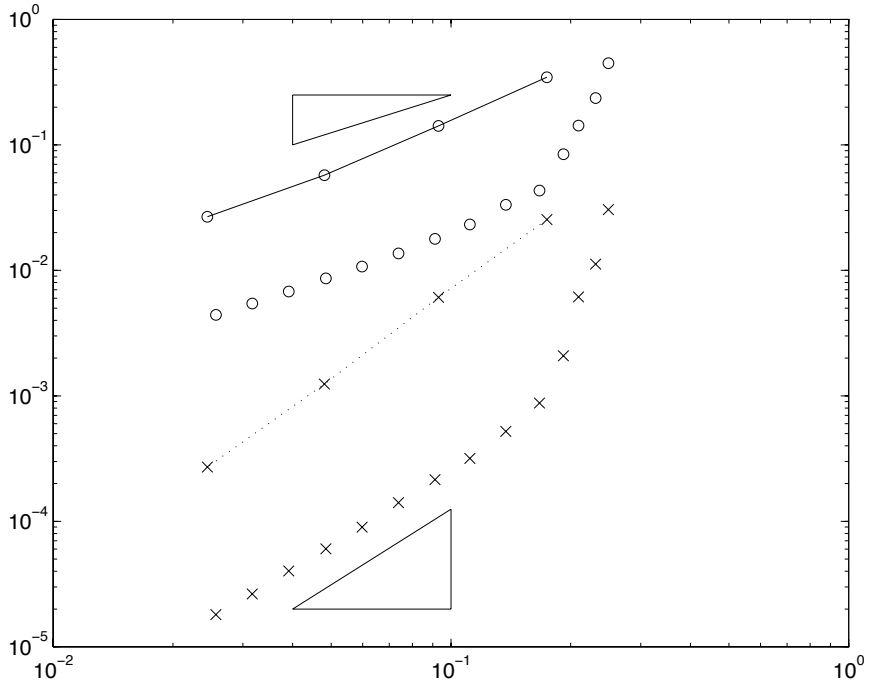

Figure 20. Problem 5: $\alpha = 20$; absolute energy and L^2 errors.

Recall that if the consistency errors introduced due to the approximation of functions appearing in the boundary value problem by their projections while com-

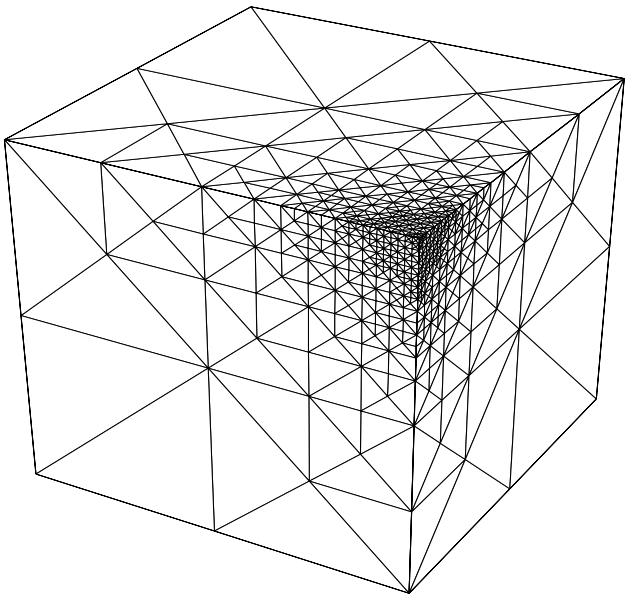

Figure 21. Problem 5: $\alpha = 20$; adapted mesh having $5,988$ tetrahedra and $1,321$ nodes showing the $x = 1$, $y = 1$, and $z = 1$ planes.

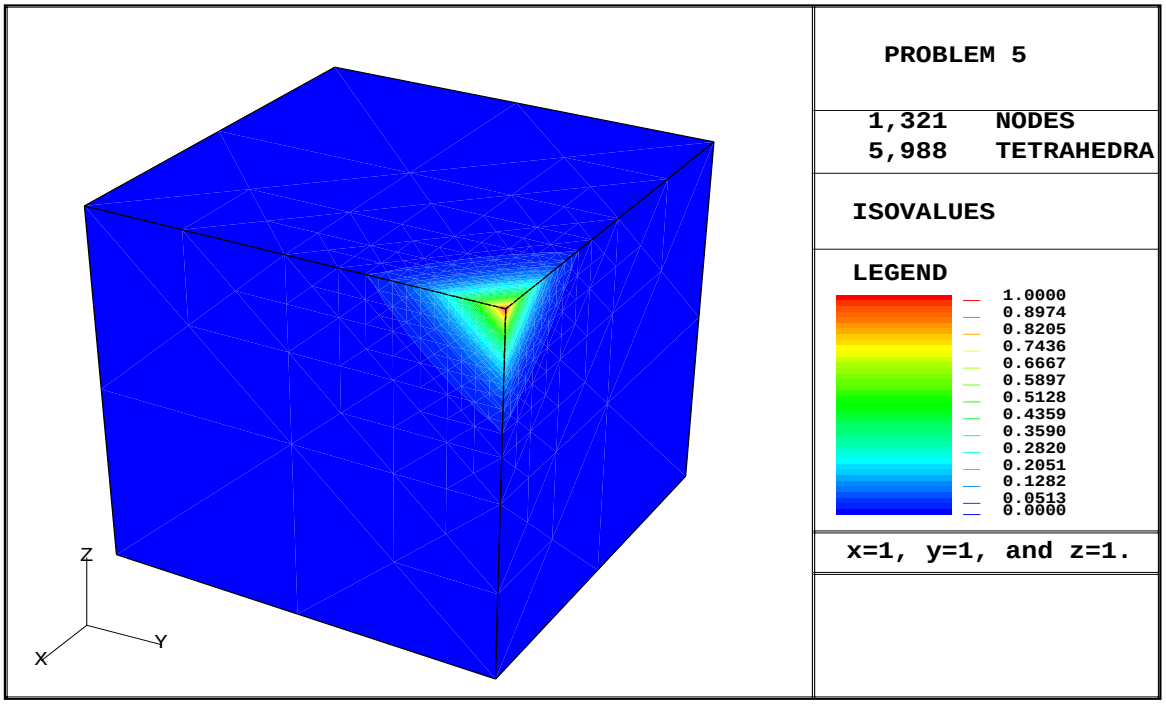

Figure 22. Problem 5: $\alpha = 20$; solution.

puting η_τ are ignored, we have $\|\hat{u} - u_{\mathcal{T}}\|_E^2 \leq C \sum_{\tau \in \mathcal{T}} \eta_\tau^2$. We report the "estimated

error" given by $\left(\sum_{\tau \in \mathcal{T}} \eta_\tau^2\right)^{1/2}$ together with the absolute energy error in Figure 23. The estimated errors are represented by a $*$'s while the absolute energy errors are represented by $\circ$'s (the absolute energy errors are shown for easy comparison of the two quantities). The figure is drawn on a log-log scale with the x axis representing mesh size. Note that the ratio of the two quantities is nearly constant for all the problems. The value of this constant is approximately equal to 10 for all the problems considered here (note that different graphs in the figure have different scales in the y direction). This is an indication that the residual error estimator may be used to devise good stopping criteria for adaptive algorithms. However, this will require much more detailed analysis and a reliable method for estimating the constant C appearing above.

The number of operations needed for the solution of a linear boundary value problem using AMG3DP1 is expected to be proportional to the number of nodes in the finest mesh. We would also expect a similar result for semilin-bvp. We report this in Figure 24. The figure reports the total CPU time in seconds for the solution of Problem 5 using semilin-bvp against the number of nodes on a log-log scale. As is clear from the figure, the operation count for our algorithm is very nearly optimal. A straight line with slope 1 would indicate that the algorithm is $O(N)$ and this line is also shown in the figure for comparison with the actual data. The actual data points are shown by $\circ$'s and connected by line segments. A similar optimal operation count behavior was also observed for the other problems.

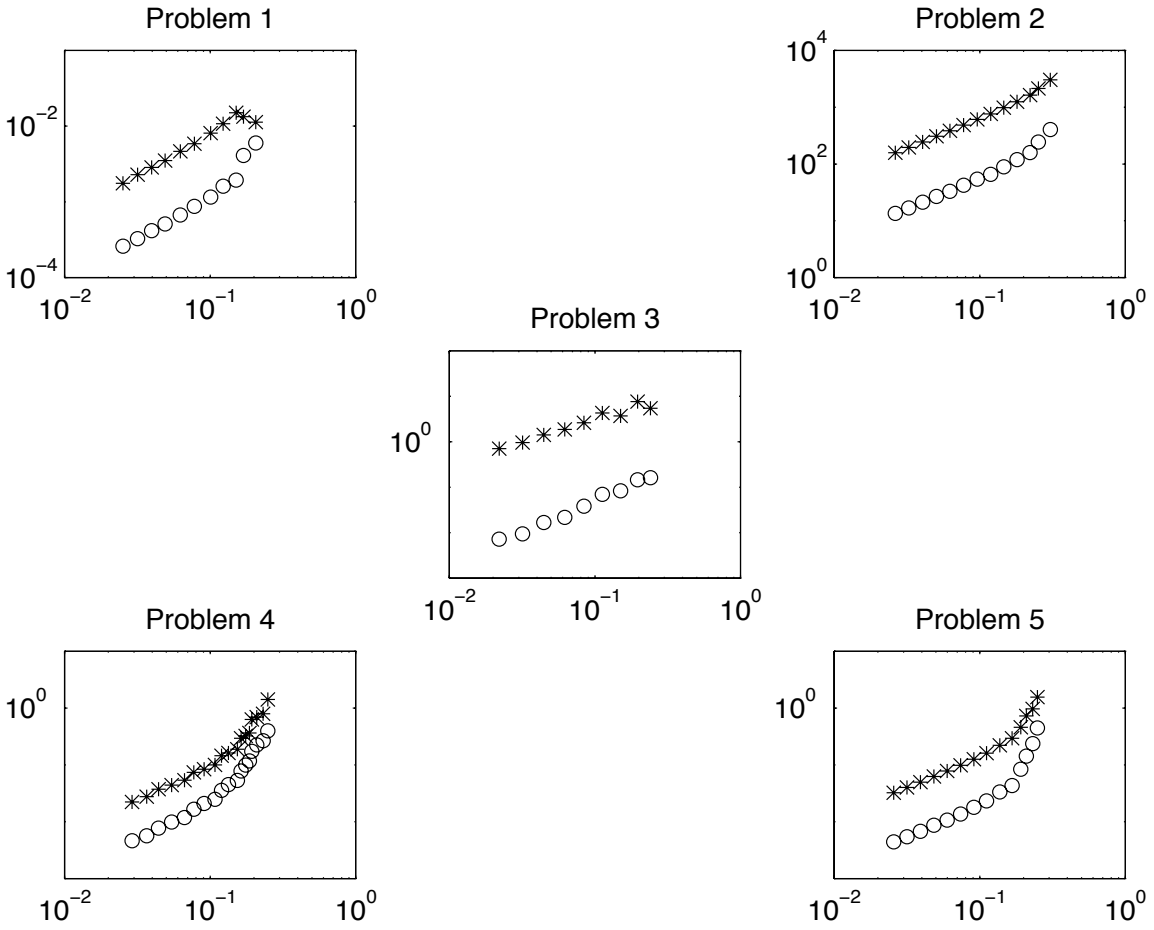

Figure 23. Problems 1–5: absolute energy (∘) and estimated (∗) errors.

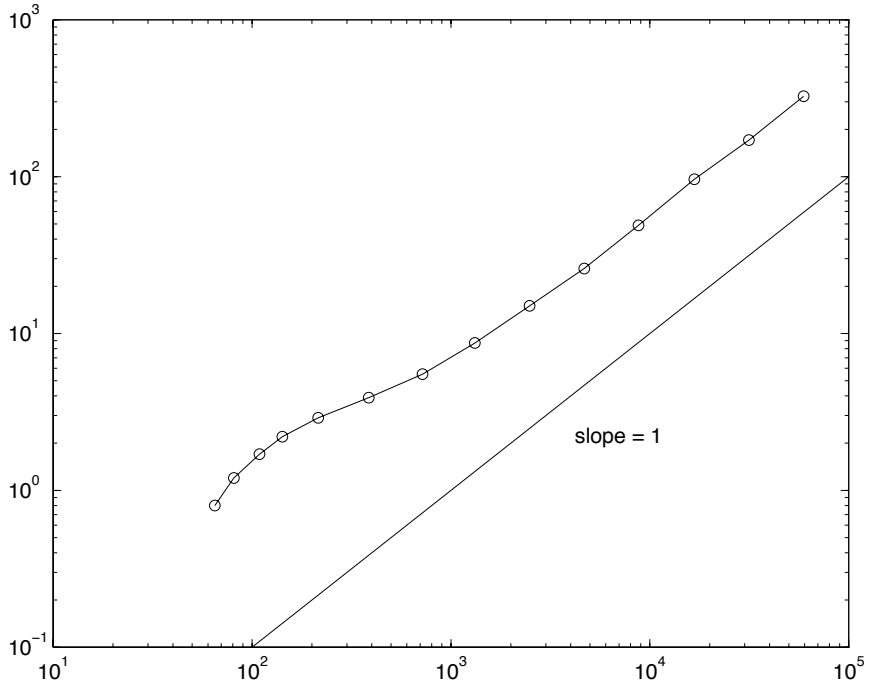

Figure 24. Problem 5: total CPU time in seconds (y axis) and number of nodes (x axis).

Chapter 6

The initial data problem for binary black hole systems

The theory of vacuum general relativity can be mathematically formulated as a system of ten coupled, nonlinear, second order partial differential equations called the Einstein equations. The unknowns of this system are the components $g_{\mu\nu}(0 \leq \mu, \nu \leq 3)$ of a symmetric metric describing spacetime. Each $g_{\mu\nu} = g_{\mu\nu}(x^\mu)$ is a function of the coordinates x^μ of spacetime. The Einstein equations are too complicated to solve except in very particular cases involving many symmetries. In fact, in most realistic astrophysical situations, such as the spiraling coalescence of two black holes, an initial value formulation of the equations is necessary for understanding and computing solutions. The initial value formulation for the Einstein equations involves a foliation of spacetime into space like hypersurfaces parameterized by a time like coordinate. In particular, the ADM (3+1) decomposition [2] for the metric $g_{\mu\nu}$ is:

$$ds^2 = g_{\mu\nu}dx^\mu dx^\nu = -\left(\alpha^2 - \beta_i\beta^i\right)dt^2 + 2\beta_i dx^i dt + \gamma_{ij}dx^i dx^j, \tag{27}$$

where Greek indices are assumed to take values $0, 1, 2, 3$, Roman indices $1, 2, 3$, x^0 is assumed to be the time like direction for spacetime, and the summation convention is used. The induced symmetric metric γ_{ij} is the metric for the space like hypersurfaces and has six independent components, the function α is called the *lapse function* and the vector β^i is the *shift vector*. Four of the ten equations obtained by projecting the Einstein equations on the above foliation does not involve second order time derivatives of γ_{ij} (see for example [28]). These four equations are called

the *constraints*. The six equations which involve second order time derivatives of γ_{ij} are the *evolution* equations. The scheme for computing solutions to the Einstein equations is:

- Determine "appropriate initial data" for one of the space like hypersurfaces (this involves the determination of the initial space like hypersurface).
- Evolve the initial data using the evolution equations.

The lapse and the shift are not dynamical variables for the constraint or evolution equations. They represent the freedom in choosing coordinates in general relativity. In fact, this arbitrariness in the choice of the lapse and the shift is used in facilitating the computation of solutions for the evolution equations. As in many time dependent second order equations, each evolution equation is rewritten as two first order differential equations. This involves using a "time derivative" of the metric γ_{ij} as a dynamical variable together with the metric itself. The dynamical variable used for this purpose is the *extrinsic curvature* K_{ij}. Let $\frac{\partial}{\partial n}$ denote the unit future oriented normal to one of the arbitrary space like hypersurfaces. Then $\frac{\partial}{\partial t} = \alpha \frac{\partial}{\partial n} + \beta^i \partial_i$, and the extrinsic curvature of the slice is defined as

$$K_{ij} = \frac{1}{2\alpha}\left(D_i\beta_j + D_j\beta_i - \frac{\partial}{\partial t}\gamma_{ij} \right),$$

where D_i represents the covariant derivative on the slice associated with γ_{ij}. The metric γ_{ij} is intrinsic to the space like hypersurface and describes its geometry, while the extrinsic curvature is a measure of how the hypersurface is embedded in spacetime. "Appropriate initial data" on an initial space like hypersurface Σ consists of specifying the pair (γ_{ij}, K_{ij}) satisfying the four constraint equations. Thus, we have twelve quantities that satisfy four equations. York [36] has given a conformal decomposition technique that provides a framework for finding (γ_{ij}, K_{ij}) satisfying the constraints. We present the conformal imaging approach to construct initial data for the spiraling coalescence of two black holes in Section 1 (see for

example [12, 17, 22, 23, 36]). The problem reduces to the solution of a semilinear elliptic boundary value problem on a domain with spherical boundaries in $\mathbb{R}^3$.

Cook et.al. [19] have investigated the solution of the elliptic boundary value problem using various numerical schemes. In Section 2, we present an algorithm for solving the initial data problem using semilin-bvp of Chapter 5. We show via numerical results in particular cases for which the exact solution is known, that our algorithm performs well (shows near optimal order of convergence). Even though we consider the initial data problem for the collision of two black holes, our solution method is quite general in the sense that it is applicable to initial data problems arising from a large class of astrophysical situations. We also comment on this at the end of Section 2.

6.1 The initial data problem

The four constraint equations for vacuum general relativity are the Hamiltonian (scalar) constraint

$$R + (\operatorname{tr} K)^2 - K_{ij}K^{ij} = 0, \tag{28}$$

and three momentum (vector) constraints

$$D_j \left(K^{ij} - \gamma^{ij} \operatorname{tr} K\right) = 0. \tag{29}$$

Here R is the Ricci scalar curvature associated with γ_{ij}, $\operatorname{tr} K$ the trace of K_{ij}, γ^{ij} the inverse of the symmetric matrix γ_{ij} such that $\gamma_{ik}\gamma^{kj} = \delta_{ij}$, and indices are raised and lowered using the metric γ_{ij}. Following the approach of York [36] and Cook [17], we will next specify the various choices that fix the topology of the initial hypersurface Σ and determine which components of γ_{ij} and K_{ij} are freely specifiable.

Since we consider the initial data for astrophysical situations like the collision of two black holes, the hypersurface Σ is assumed to be asymptotically flat. A

common method for choosing the topology of Σ is to assume that it consists of two identical asymptotically flat hypersurfaces (sheets) connected by two throats or Einstein-Rosen bridges (cf. Einstein and Rosen [21], Misner [27], Cook [17]). The two throats represent the black holes on Σ, and the two sheets are assumed to be identified via an isometry condition that is satisfied by the metric and the extrinsic curvature. It is known that if the two black holes are represented as spheres having radius a_i and centers $\boldsymbol{C}_i \in \mathbb{R}^3$ for $i = 1, 2$, the two sheets are represented by $\left\{ \boldsymbol{x} \in \mathbb{R}^3 | \, |\boldsymbol{x} - \boldsymbol{C}_i| > a_i, i = 1, 2 \right\}$ as point sets (cf. Kulkarni [22]).

Assume that the physical hypersurface (space) Σ is conformally related to a background space $\bar{\Sigma}$ so that the physical metric $\gamma_{ij} = \psi^4 \bar{\gamma}_{ij}$, where $\psi > 0$ is a conformal factor and $\bar{\gamma}_{ij}$ is the conformal background metric. Further, let

$$K_{ij} = A_{ij} + \frac{1}{3}\gamma_{ij} \operatorname{tr} K = \psi^{-2} \bar{A}_{ij} + \frac{1}{3}\gamma_{ij} \operatorname{tr} K, \tag{30}$$

where $\bar{A}_{ij}$ is the trace free conformal background extrinsic curvature. In terms of the variables ψ, $\bar{\gamma}_{ij}$, $\bar{A}_{ij}$, and $\operatorname{tr} K$, the scalar and momentum constraints can be rewritten as

$$\bar{\Delta}\psi - \frac{1}{8}\psi \bar{R} - \frac{1}{12}\psi^5 (\operatorname{tr} K)^2 + \frac{1}{8}\psi^{-7} \bar{A}_{ij} \bar{A}^{ij} = 0, \tag{31}$$

$$\bar{D}_j \bar{A}^{ij} - \frac{2}{3}\psi^6 \bar{\gamma}^{ij} \bar{D}_j \operatorname{tr} K = 0, \tag{32}$$

where all quantities with bars are assumed to be computed using the background metric $\bar{\gamma}_{ij}$. Since γ_{ij} has six independent components and the conformal factor ψ must satisfy the scalar constraint (31), $\bar{\gamma}_{ij}$ has five independent components. Similarly, the vector constraints (32) determine three of the six independent components of K_{ij}, $\operatorname{tr} K$ accounts for one, and the remaining two are contained in $\bar{A}_{ij}$. Thus, once the quantities $\bar{\gamma}_{ij}$, $\operatorname{tr} K$, and $\bar{A}_{ij}$ are given, equations (31) and (32) may be solved for ψ and the remaining three components of K_{ij} leading to the construction of initial data.

For studying binary black hole systems, a common choice for $\bar{\gamma}_{ij}$ is the flat space metric δ_{ij}. This choice has two advantages:

(1) The scalar constraint simplifies ($\bar{R} = 0$).

(2) The operators $\bar{\Delta}$ and $\bar{D}_i$ become the usual flat space operators.

It is also assumed that the slice Σ is maximally embedded in spacetime, so that $\operatorname{tr} K = 0$. This assumption decouples the scalar constraint (31) from (32) and reduces the vector constraints to $\bar{D}_j \bar{A}^{ij} = 0$.

Bowen and York [12] give closed form solutions of these vector constraints for the case of a single black hole. Kulkarni et. al. [23] provide solutions in the form of formal infinite series for multiple black holes. Cook [18] provides an efficient and accurate method for evaluating the formal series in the case of two spiraling black holes. This procedure requires that the radii a_i, centers $\boldsymbol{C}_i$, linear momenta $\boldsymbol{P}_i$, and spin $\boldsymbol{S}_i$ for the two holes be known.

Assuming that the vector constraints have been solved for $\bar{A}_{ij}$, the scalar constraint (31) is rewritten as a semilinear boundary value problem using inversion symmetry and asymptotic flatness to construct the boundary conditions. Kulkarni [22] has shown (extending the approach of Bowen and York [12] for one black hole) using inversion through a sphere that the initial data for binary black hole systems needs to be specified only on

$$\left\{ \boldsymbol{x} \in \mathbb{R}^3 \middle| \, |\boldsymbol{x} - \boldsymbol{C}_i| \geq a_i, i = 1, 2 \right\}.$$

Under the assumptions mentioned above, the scalar constraint reduces to solving the semilinear problem

$$\bar{\Delta}\psi = -\frac{1}{8}\psi^{-7}\bar{A}_{ij}\bar{A}^{ij} \text{ in } \left\{ \boldsymbol{x} \in \mathbb{R}^3 \middle| \, |\boldsymbol{x} - \boldsymbol{C}_i| > a_i, i = 1, 2 \right\} \tag{33}$$

for ψ. If $\Gamma_i = \left\{ \boldsymbol{x} \in \mathbb{R}^3 \middle| \, |\boldsymbol{x} - \boldsymbol{C}_i| = a_i \right\}$ is the boundary of the i-th black hole, the isometry condition on the physical metric γ_{ij} gives the following as the boundary

condition for ψ:

$$\frac{\partial \psi}{\partial \boldsymbol{n}_i} + \frac{1}{2a_i}\psi = 0 \text{ on } \Gamma_i, \tag{34}$$

where $\boldsymbol{n}_i$ is the unit outward normal to Γ_i with respect to $\boldsymbol{C}_i$. The asymptotic flatness of Σ translates to the condition $\lim_{|\boldsymbol{x}|\to\infty} \psi = 1$. To pose the semilinear boundary value problem on a finite computational domain, a multipole expansion for the conformal factor is truncated to give an approximate boundary condition $\frac{\partial \psi}{\partial r} = \frac{1-\psi}{r}$ at some large value of the radial distance r measured from the origin in comparison to the radii of the black holes and the distance $|\boldsymbol{C}_1 - \boldsymbol{C}_2|$ (see for example York and Piran [37]). If we denote by $B_R = \left\{\boldsymbol{x} \in \mathbb{R}^3 \,|\, |\boldsymbol{x} - \boldsymbol{C}_i| < R\right\}$ a ball with large radius R, by Γ_R the boundary of B_R, and by $B_i = \left\{\boldsymbol{x} \in \mathbb{R}^3 \,|\, |\boldsymbol{x} - \boldsymbol{C}_i| < a_i\right\}$ the ball with radius a_i, the boundary value problem for determining ψ is

$$\bar{\Delta}\psi = -\frac{1}{8}\psi^{-7}\bar{A}_{ij}\bar{A}^{ij} \text{ in } B_R \setminus (B_1 \cup B_2), \tag{35}$$

$$\frac{\partial \psi}{\partial \boldsymbol{n}_i} + \frac{1}{2a_i}\psi = 0 \text{ on } \Gamma_i, i = 1, 2, \tag{36}$$

$$\frac{\partial \psi}{\partial r} + \frac{1}{r}\psi = \frac{1}{r} \text{ on } \Gamma_R. \tag{37}$$

Taking $\Omega = B_R \setminus (B_1 \cup B_2)$, $\Gamma_D = \emptyset$, and $\Gamma_N = \Gamma_R \cup \Gamma_1 \cup \Gamma_2$, (35)–(37) has the form of (24) of Chapter 5. Here we have identified u with ψ and chosen $\underset{\approx}{\mathcal{A}} = \underset{\approx}{I}$ and $f(\underset{\sim}{x}, u) = -\frac{1}{8}H(\underset{\sim}{x})u^{-7}$ where $H(\underset{\sim}{x}) = \bar{A}_{ij}\bar{A}^{ij}$ is a known function. Note that the normals $\boldsymbol{n}_i$ to Γ_i point into the domain Ω, and this interferes with an usual existence and uniqueness analysis for the problem (the coefficient $-1/2a_i$ multiplying ψ in (36) is not positive when the condition is rewritten as $\frac{\partial \psi}{\partial \boldsymbol{m}_i} - \frac{1}{2a_i}\psi = 0$ with $\boldsymbol{m}_i = -\boldsymbol{n}_i$ the unit normal to Γ_i pointing outward with respect to Ω). However, We will proceed under the assumption that (35)–(37) admits a locally unique solution.

Remark 6.1.1 The conformal imaging approach is quite general in the sense that it provides a framework for the solution of the Einstein equations for various physical phenomena. An important first step in this approach is the construction

of initial data for the initial space like hypersurface. This involves the solution of the constraint equations (31) and (32). Different astrophysical situations lead to different choices for the background metric $\bar{\gamma}_{ij}$. However, given any symmetric background metric $\bar{\gamma}_{ij}$, the first term in (31) always has the form $\mathrm{div}(\underset{\approx}{\mathcal{A}}\underset{\sim}{\mathrm{grad}}\,\psi)$, where the matrix $\underset{\approx}{\mathcal{A}}$ is derivable from $\bar{\gamma}_{ij}$ and is symmetric. Thus, if $\mathrm{tr}\,K$ and $H(\underset{\sim}{x})$ are known (equivalently, the extrinsic curvature K_{ij} is known), the scalar constraint reduces to a semilinear equation like (24) of Chapter 5 for any choice of $\bar{\gamma}_{ij}$. If the hypersurface Σ can be replaced by a finite domain in $\mathbb{R}^3$ and appropriate boundary conditions are specified, the resulting boundary value problem can be efficiently solved using semilin-bvp.

6.2 Numerical examples

The conformal imaging procedure outlined in the previous section also applies to the case of one black hole. The function H depends on the linear and angular momenta associated with the black hole. In the simplest possible situation, the function H is zero. This corresponds to time symmetric initial data and the scalar constraint reduces to Laplace's equation. The exact solution is known in this case (this is the Schwarzschild solution). Bowen and York [12] also give a model value for the function H (which is not identically zero) and the corresponding analytic expression for the conformal factor ψ satisfying (35). This solution also satisfies the boundary conditions (36) and (37). Both the Schwarzchild and the model Bowen-York solutions are radial. This model H will be denoted by H_{model} and parameterized by a quantity P representing the linear momentum of the black hole (it actually represents a linear momentum $(0, 0, P)$). When $P = 0$, the conformal factor corresponding to H_{model} reduces to the Schwarzchild solution.

Other model values of H proposed by Bowen and York for one black hole are essentially two dimensional in the sense that the expression for H depends on r and θ if we consider spherical polar coordinates (r, θ, ϕ) for $\mathbb{R}^3$. The expression for

the conformal factor ψ is explicitly known only when $H(= H_{model})$ is independent of θ. Two other expressions for H that we will consider will be denoted by H_{lin} (corresponding to a black hole having linear momentum), and H_{ang} (corresponding to a black hole having angular momentum).

We will perform all computations involving one black hole on a domain $\Omega = B_R \setminus B_1$ with $\boldsymbol{C}_1 = (0,0,0)$, $a_1 = \frac{1}{2}\sqrt{3}$, and $R = 1028a_1$. Other authors have considered these test problems before. In particular, Cook [17] has solved them on a two dimensional domain by exploiting the inherent symmetries in the solution. He uses a finite difference discretization and a multigrid method for solving the resulting equations on a domain where the outer radius R is approximately $22,000$ times the radius of the black hole a_1. He uses a $16 \times 6(= 96)$ grid as the coarse grid and a $1024 \times 384(= 393216)$ grid as the finest grid for his multigrid.

The expression for H_{model} and the corresponding exact expression for ψ_{model} is

$$H_{model} = 6\frac{P^2}{r^4}\left(1 - \frac{a_1^2}{r^2}\right)^2 \quad \text{and} \quad \psi_{model} = \left(1 + \frac{2E}{r} + \frac{6a_1^2}{r^2} + \frac{2a_1^2 E}{r^3} + \frac{a_1^4}{r^4}\right)^{1/4},$$

where $E = (P^2 + 4a_1^2)^{1/2}$ and P is a parameter as mentioned before. The expressions for H_{lin} (respectively H_{ang}) are parameterized by a linear momentum parameter P (respectively J) and are given as

$$H_{lin} = \frac{9P^2}{2r^4}\left[\left(1 - \frac{a_1^2}{r^2}\right)^2 + 2\cos^2\theta\left(1 + 4\frac{a_1^2}{r^2} + \frac{a_1^4}{r^4}\right)\right],$$

$$H_{ang} = 18\frac{J^2}{r^6}\sin^2\theta.$$

Two quantities of interest that are associated with the conformal factor (equivalently, the metric γ_{ij}) are the ADM-energy and ADM-mass defined as

$$E = \frac{1}{16\pi}\int_\Omega H\psi^{-7}\,dv + \frac{1}{4\pi a_1}\int_{\Gamma_1}\psi\,ds \quad \text{and} \quad M = \left(\frac{1}{16\pi}\int_{\Gamma_1}\psi^4\,ds\right)^{1/2}.$$

The ADM-energy represents the the total energy of the initial spacelike slice Σ computed in Cartesian coordinates under the assumption that the outer boundary Γ_R represents a sphere of arbitrarily large radius. The ADM-mass is related to the area of the throat, and is expressed in Cartesian coordinates above (these quantities can be defined for any space like hypersurface if the metric is known, but the expressions given above have been deduced for the special case that we consider by using the scalar constraint and boundary conditions). Note that these quantities are known analytically for H_{model}, and Cook [17] has reported these for a wide range of values of the parameters P and J corresponding to H_{lin} and H_{ang}. These reported values have been used by other authors (see for example Duval et. al. [20]) as reference values for code validation. In Problems 6–8 below, we solve the scalar constraint by taking H to be equal to H_{model} in Problem 6, H_{lin} in Problem 7, and H_{ang} in Problem 8. In all cases report the computed values of E and M scaled by the radius of the black hole (denoted by E_{comp}/a_1 and M_{comp}/a_1) and compare them with the corresponding calculations of Cook for different choices of P/a_1 (for Problem 6 and 7) and J/a_1^2 (for Problem 8). The scaled values of the energy and mass used as reference will be denoted by E_{ref}/a_1 and M_{ref}/a_1 and relative errors will be computed as

$$100 \times \frac{\left| \frac{E_{ref}}{a_1} - \frac{E_{comp}}{a_1} \right|}{\left| \frac{E_{ref}}{a_1} \right|} \quad \text{and} \quad 100 \times \frac{\left| \frac{M_{ref}}{a_1} - \frac{M_{comp}}{a_1} \right|}{\left| \frac{M_{ref}}{a_1} \right|}.$$

For Problem 6 we also report on the absolute energy errors for two choices of the parameter P ($P/a_1 = 0$ representing the Schwarzchild solution, and $P/a_1 = 10$ representing the solution of a general semilinear scalar constraint). We also report the estimated errors in these two cases. Cook reports on the *average relative pointwise error* defined as

$$\|\pi\hat{u} - \boldsymbol{u}_{\mathcal{T}}\|_{\ell_1} = \frac{1}{\mathrm{card}(\mathcal{N}(\mathcal{T}))} \sum_{\nu \in \mathcal{N}(\mathcal{T})} \frac{|(\pi\hat{u})_\nu - (\boldsymbol{u}_{\mathcal{T}})_\nu|}{|(\pi\hat{u})_\nu|}$$

for a mesh $\mathcal{T}$, where $\pi\hat{u}$ is the vector of coefficients associated with the exact solution $\hat{u}$ as defined in Chapter 4, and $\boldsymbol{u}_{\mathcal{T}}$ the finite element solution corresponding to $\mathcal{T}$. For the values of the parameter P mentioned above for Problem 6, we also report the ℓ_1 errors.

The energy and estimated errors for the various meshes are represented using $\circ$ and $*$ respectively as was done for the numerical examples in Chapter 5. The average relative pointwise errors are represented on the same graphs by +'s joined by solid line. The coarse mesh for the one hole domain $\Omega = B_R \setminus B_1$ has $1,440$ tetrahedra and 286 nodes.

Problem 9 deals with the calculation of the conformal factor for the problem of two colliding black holes. Here, we solve the scalar constraint on the domain $\Omega = B_R \setminus (B_1 \cup B_2)$, with $\boldsymbol{C}_1 = (0,0,-b)$, $\boldsymbol{C}_2 = (0,0,b)$, $a_2 = 2a_1$, $a_1 = a$ and $R = 128a$. The linear and angular momenta $\boldsymbol{P}_i$ and $\boldsymbol{S}_i$ associated with the holes are assumed to be known. Thus, the function H is computable using a procedure of Cook [18]. We show that algorithm semilin-bvp is $O(N)$ in this case and show pictures of the resulting mesh and solution.

Problem 10 deals with an initial data problem where the background metric $\bar{\gamma}_{ij}$ is not the flat space metric. The details on this problem together with references are given in the description of the problem.

The problems were all solved using the same parameters as the ones used in the numerical examples in Chapter 5 (two iterations of the V-cycle and the same stopping criterion for the Newton's iteration).

Problem 6. Choose $\underset{\approx}{\mathcal{A}} = \underset{\approx}{I}$, $f(\underset{\sim}{x}, u) = -\frac{1}{8}u^{-7}H_{model}$, $\Gamma_D = \emptyset$, $\Gamma_N = \Gamma_1 \cup \Gamma_R$,

$$c = \begin{cases} \dfrac{-1}{2a_1} \text{ on } \Gamma_1, \\ \dfrac{1}{R} \quad \text{on } \Gamma_R, \end{cases} \quad \text{and} \quad g_N = \begin{cases} 0 \quad \text{on } \Gamma_1, \\ \dfrac{1}{R} \text{ on } \Gamma_R. \end{cases}$$

The computed and reference (analytic in this case) values of the ADM-energy and mass are reported in Table 2 together with the relative errors. We used a maximum

of less than $70,000$ nodes in our finest mesh for all choices of the parameter P, and it is clear from Table 2 that the relative errors are quite good.

Table 2. ADM-energy and ADM-mass for H_{model}.

P/a_1	E_{ref}/a_1	E_{comp}/a_1 (relative error)	M_{ref}/a_1	M_{comp}/a_1 (relative error)
0	2.00000	1.97825 (1.09%)	2.00000	1.96345 (1.83%)
5	5.38516	5.34288 (0.78%)	2.71750	2.66409 (1.96%)
10	10.1980	10.0669 (1.28%)	3.49257	3.42347 (1.98%)
17.5	17.6139	17.2142 (2.27%)	4.42876	4.33908 (2.02%)

Figure 25 reports the absolute energy norm of the error ($\circ$) and the estimated error ($*$) for $P/a_1 = 0$ (linear problem). As is evident from the line with slope 1 on the graph, the convergence rate is optimal. Also, the estimated error over estimates the absolute energy error by a factor of approximately 10 as was the case for the numerical examples of Chapter 5. The figure also shows the average relative pointwise error (+'s joined by line segments). A line indicating a slope of 2 is drawn for comparison. The figure is drawn on a log-log scale and has the mesh size on the x axis. Figure 26 shows the same quantities for $P/a_1 = 10$ (semilinear problem). As is clear from these two figures, the errors and rates are quite comparable for the two problems. For $P/a_1 = 0$ Cook reports an average relative pointwise error of 0.00345 using his finest mesh of $393,216$ grid points, while we get a value 0.00469 using $62,441$ nodes, while for $P/a_1 = 10$ we get an average relative pointwise error of 0.00302 using $59,248$ nodes as compared to 0.00178 reported by Cook for his finest grid.

The conformal factor corresponding to H_{model} is radial, and this is seen for the computed solution in the two views showing the isovalues and mesh on the plane $x = 0$ for an intermediate adapted mesh having $13,378$ nodes and $73,584$ tetrahedra in Figure 27 and 28 using $P/a_1 = 10$. Figure 27 shows the intersection of the plane

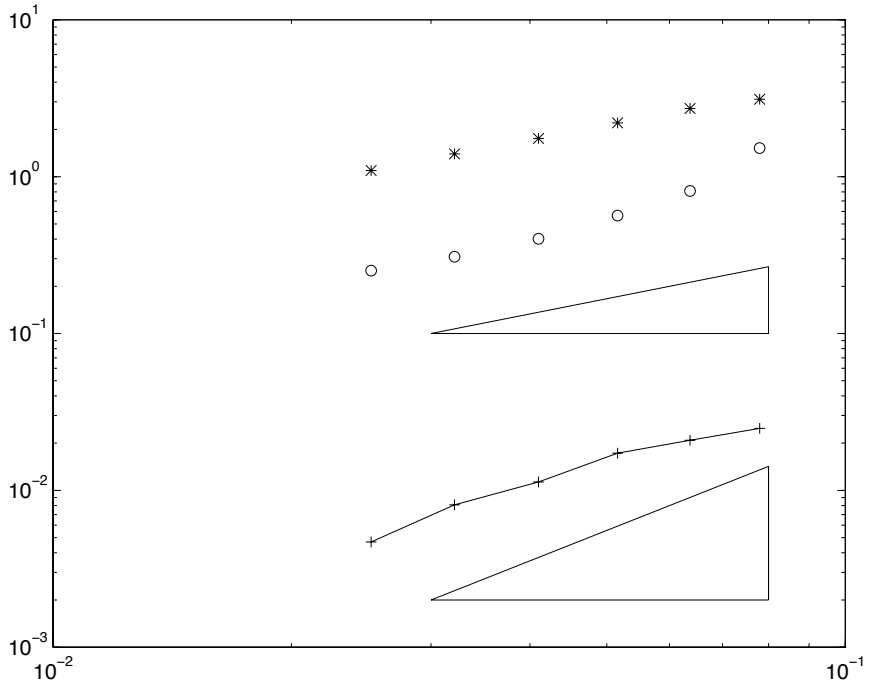

Figure 25. Problem 6: $P/a_1 = 0$; absolute energy, estimated, and ℓ_1 errors.

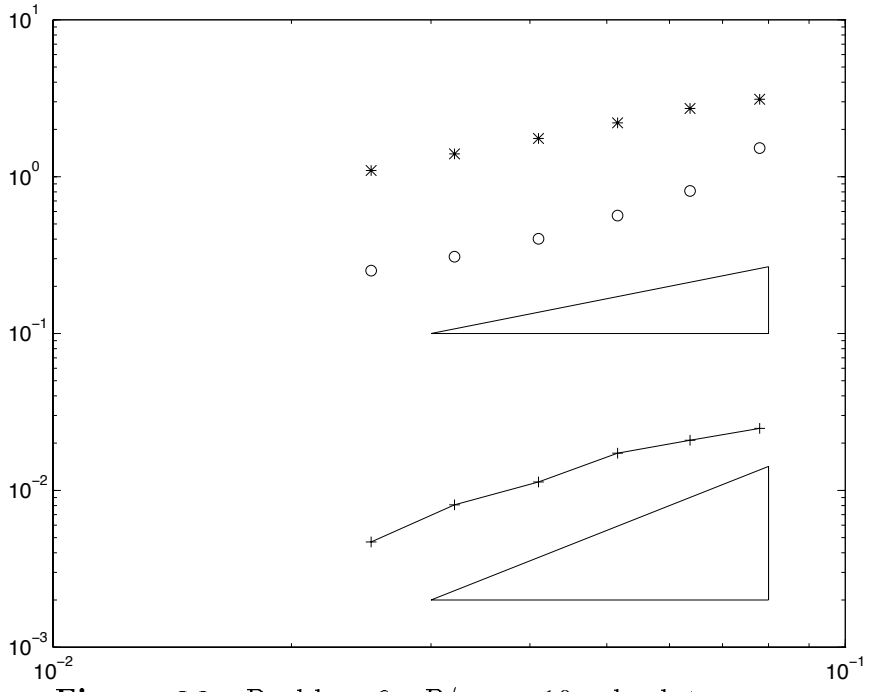

Figure 26. Problem 6: $P/a_1 = 10$; absolute energy, estimated, and ℓ_1 errors.

with the domain Ω, while Figure 28 is a zoom of this plane (near $(0,0,0)$). We also know that the conformal factor has steep gradients near the black hole while it is

flat away from the hole (asymptotic flatness of the initial slice). This behavior is captured quite well by the adaptive mesh refinement.

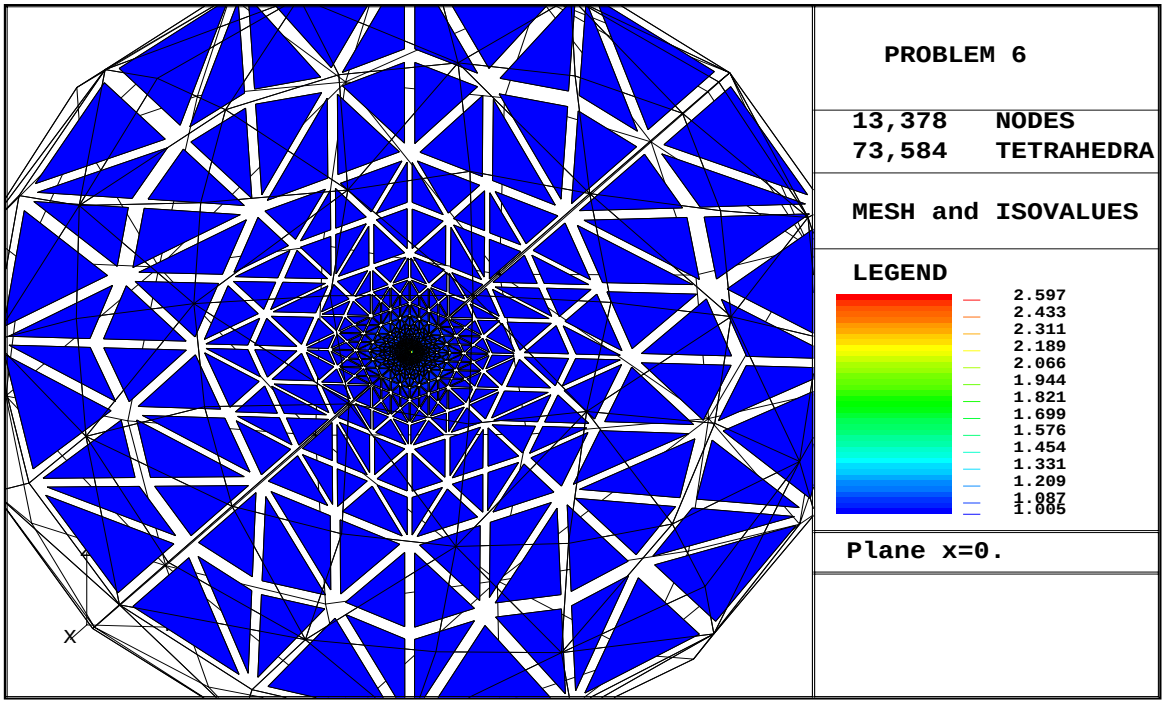

Figure 27. Problem 6: $P/a_1 = 10$; solution.

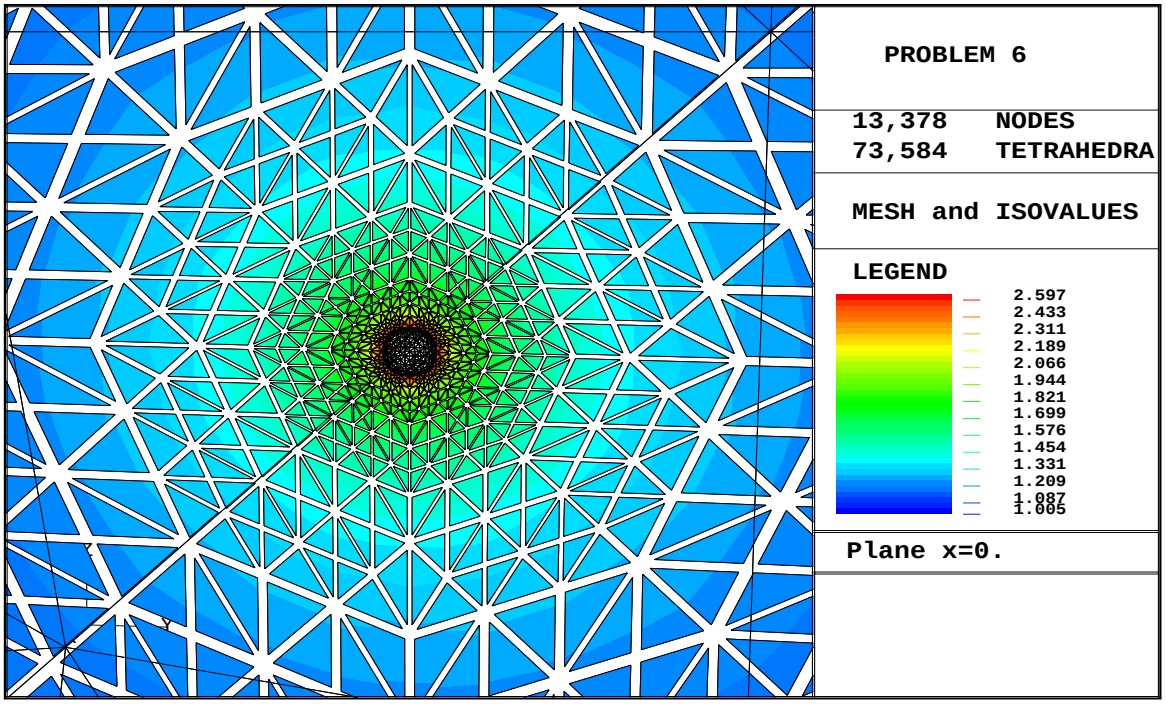

Figure 28. Problem 6: $P/a_1 = 10$; solution (zoom).

Problem 7. Choose $\underset{\approx}{\mathcal{A}} = \underset{\approx}{I}$, $f(\underset{\sim}{x}, u) = -\frac{1}{8}u^{-7}H_{lin}$, $\Gamma_D = \emptyset$, $\Gamma_N = \Gamma_1 \cup \Gamma_R$, and

$$c = \begin{cases} \dfrac{-1}{2a_1} \text{ on } \Gamma_1, \\ \dfrac{1}{R} \quad \text{on } \Gamma_R, \end{cases} \quad \text{and} \quad g_N = \begin{cases} 0 \quad \text{on } \Gamma_1, \\ \dfrac{1}{R} \text{ on } \Gamma_R. \end{cases}$$

The computed and reference (reported by Cook) values of the ADM-energy and mass are reported in Table 3 together with the relative errors. We used a maximum of less than $70,000$ nodes in our finest mesh for all the choices of the parameter P, and it is clear from Table 3 that the relative errors are quite good.

Table 3. ADM-energy and ADM-mass for H_{lin}.

P/a_1	E_{ref}/a_1	E_{comp}/a_1 (relative error)	M_{ref}/a_1	M_{comp}/a_1 (relative error)
1	2.34726	2.34696 (0.01%)	2.11263	2.09120 (1.01%)
5	6.13267	6.15093 (0.30%)	3.06881	3.07487 (0.20%)
10	11.5406	11.4480 (0.80%)	4.04867	4.06908 (0.52%)
17.5	19.8085	19.3742 (2.19%)	5.29517	5.22558 (0.58%)

Problem 8. Choose $\underset{\approx}{\mathcal{A}} = \underset{\approx}{I}$, $f(\underset{\sim}{x}, u) = -\frac{1}{8}u^{-7}H_{ang}$, $\Gamma_D = \emptyset$, $\Gamma_N = \Gamma_1 \cup \Gamma_R$, and

$$c = \begin{cases} \dfrac{-1}{2a_1} \text{ on } \Gamma_1, \\ \dfrac{1}{R} \quad \text{on } \Gamma_R, \end{cases} \quad \text{and} \quad g_N = \begin{cases} 0 \quad \text{on } \Gamma_1, \\ \dfrac{1}{R} \text{ on } \Gamma_R. \end{cases}$$

The computed and reference (reported by Cook) values of the ADM-energy and mass are reported in Table 4 together with the relative errors. We used a maximum of less than $70,000$ nodes in our finest mesh for all the choices of the parameter J, and it is clear from Table 4 that the relative errors are quite good.

Problem 9. Choose $\underset{\approx}{\mathcal{A}} = \underset{\approx}{I}$, $f(\underset{\sim}{x}, u) = -\frac{1}{8}u^{-7}H$, $\Gamma_D = \emptyset$, $\Gamma_N = \Gamma_1 \cup \Gamma_2 \cup \Gamma_R$, and

$$c = \begin{cases} \dfrac{-1}{2a_i} \text{ on } \Gamma_i, \\ \dfrac{1}{R} \quad \text{on } \Gamma_R, \end{cases} \quad \text{and} \quad g_N = \begin{cases} 0 \quad \text{on } \Gamma_i, \\ \dfrac{1}{R} \text{ on } \Gamma_R, \end{cases} \quad \text{for } i = 1, 2.$$

Table 4. ADM-energy and ADM-mass for H_{ang}.

J/a_1^2	E_{ref}/a_1	E_{comp}/a_1 (relative error)	M_{ref}/a_1	M_{comp}/a_1 (relative error)
1	2.04765	2.02800 (0.96%)	2.03275	2.00023 (1.60%)
100	10.4049	10.4498 (0.43%)	8.07053	7.99379 (0.95%)
10000	103.796	104.311 (0.50%)	77.5627	76.6156 (1.22%)

For the computation of the function H we assign zero spin to the two holes ($\boldsymbol{S}_i = 0$), and linear momenta given by $\boldsymbol{P}_1 = (15, 0, 0)$ and $\boldsymbol{P}_2 = (-15, 0, 0)$. The centers of holes are at a distance of $2b$ on the z axis, and the outer radius is $128a$ where a is the radius of the smaller hole. For our numerical example, we choose $a = \sqrt{3}/2$ and $b = \sqrt{12}(\approx 3.46)$. Figure 31 shows the total CPU time taken for the solution of this realistic problem against the number of nodes on a log-log scale. The actual data points are indicated by $\circ$'s joined by line segments. A line of slope 1 is also shown for easy comparison. It is clear from the figure that semilin-bvp shows a $O(N)$ behavior for this problem. It should be noted that more than half the CPU time was spent in computing the extrinsic curvature H using a recursive formula given by Cook [18]. The finest mesh had $63,133$ nodes as compared to 585 for the coarse mesh.

The computed values of the conformal factor have steep gradients near the two black holes, and approach the constant value 1 away from the holes. The adapted mesh near the two holes is seen clearly in a view of the mesh and the isovalues of the solution on the $x = y$ plane for an intermediate adapted mesh having $75,300$ tetrahedra and $13,899$ nodes. Figure 29 shows the solution and isovalues for this mesh, while Figure 30 shows the same plane with a zoom at the origin.

Problem 10. As a last example, we consider the application of our code to solve the initial data problem for a single black hole plus Brill wave spacetime [10]. In this case, the conformal imaging approach still applies but the background metric

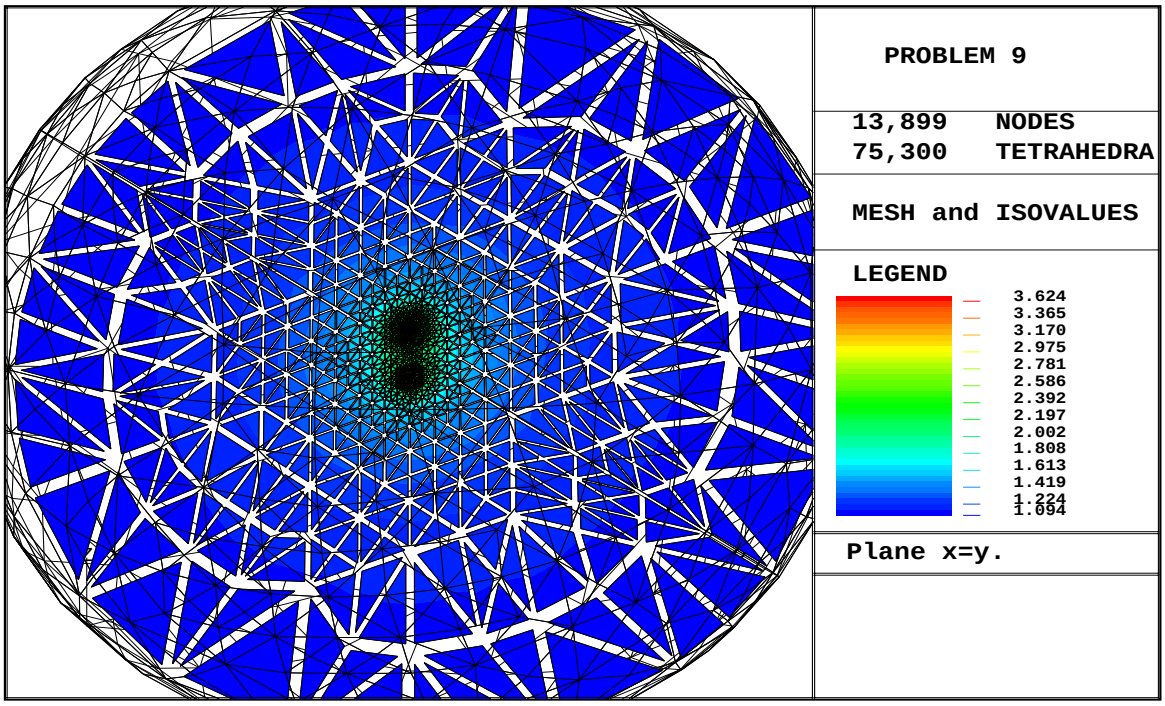

Figure 29. Problem 9: solution.

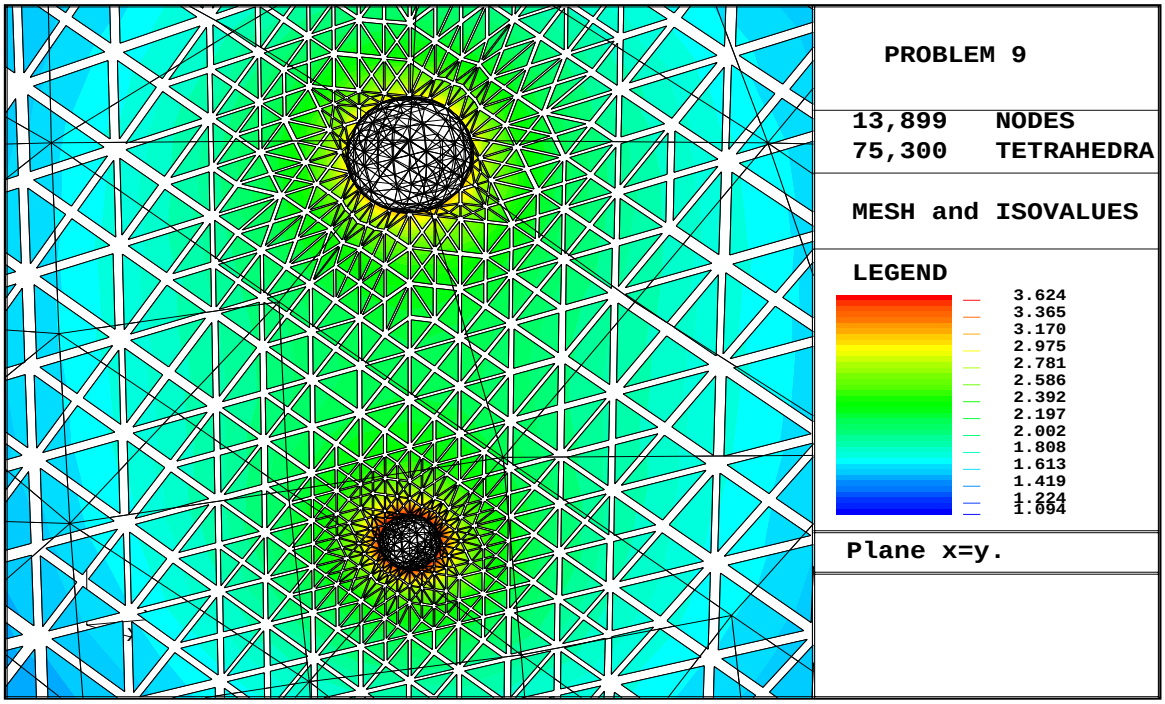

Figure 30. Problem 9: solution with zoom.

$\bar{\gamma}_{ij}$ is not flat. In particular, Bernstein et. al. [10] have studied this problem extensively and specified a choice for the background metric. They use axisymmetry to

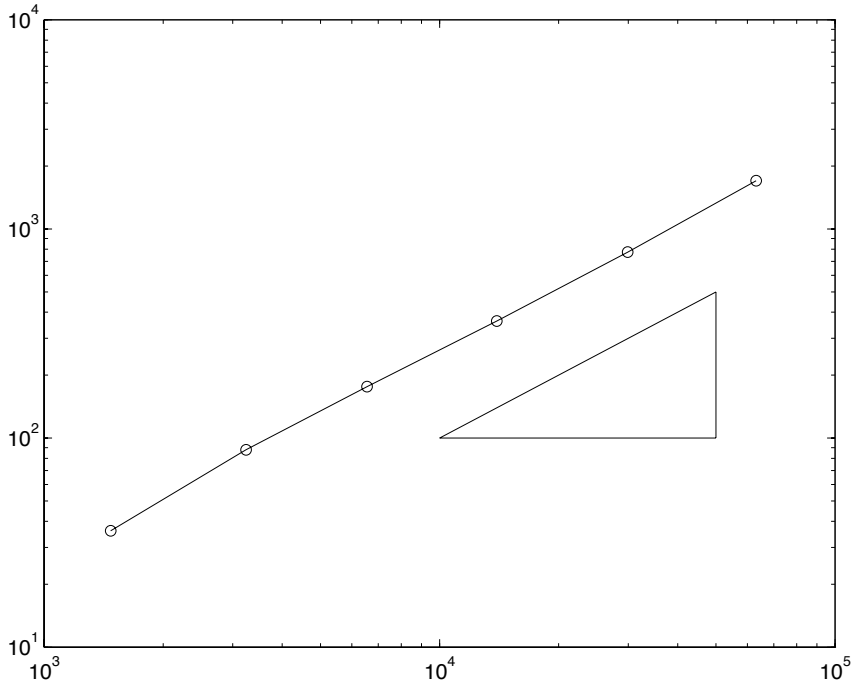

Figure 31. Problem 9: total CPU time in seconds (y axis) and number of nodes (x axis).

pose the corresponding initial data problem as a boundary value problem in $\mathbb{R}^2$, but we will solve the same problem on $\Omega = B_R \backslash B_1$ used for the solution of Problems 6–8. The boundary conditions are the same as the ones used in Problems 6–8.

For this case we choose $K_{ij} = 0$ (the third and fourth terms in (31) vanish). The background metric $\bar{\gamma}_{ij}$ is chosen so that the metric $\underset{\approx}{\gamma} = \{\gamma_{ij}\} (= \psi^4 \bar{\gamma}_{ij})$ is

$$\underset{\approx}{\gamma} = \psi^4 \begin{bmatrix} \frac{x^2 e^{2q} + y^2}{x^2+y^2} & \frac{xy(e^{2q}-1)}{x^2+y^2} & 0 \\ \frac{xy(e^{2q}-1)}{x^2+y^2} & \frac{x^2+y^2 e^{2q}}{x^2+y^2} & 0 \\ 0 & 0 & e^{2q} \end{bmatrix}, \text{ with } q = a \sin^2\theta \left[e^{-(\frac{\eta+b}{w})^2} + e^{-(\frac{\eta-b}{w})^2} \right].$$

Here $\eta = \ln(r/a_1)$, and the function q (hence the metric γ_{ij}) is chosen to be independent of the ϕ variable in spherical polar coordinates (r, θ, ϕ). The parameter set (a, b, w) represents the amplitude, position, and width of the Brill wave being considered. With this choice of $\bar{\gamma}_{ij}$, the scalar constraint (31) reduces to $\bar{\Delta}\psi - \frac{1}{8}\bar{R}\psi = 0$ where $\bar{\Delta}$ is the Laplacian, and $\bar{R}$ the scalar curvature computed using the metric $\bar{\gamma}_{ij}$. A direct computation reduces this to the equation

$$-\operatorname{div}(\underset{\approx}{\mathcal{A}} \underset{\sim}{\operatorname{grad}}\, \psi) + \frac{1}{8}\bar{R} e^{2q} \psi = 0 \tag{38}$$

with

$$\underset{\approx}{\mathcal{A}} = \begin{bmatrix} \frac{x^2+y^2e^{2q}}{x^2+y^2} & \frac{xy(1-e^{2q})}{x^2+y^2} & 0 \\ \frac{xy(1-e^{2q})}{x^2+y^2} & \frac{x^2e^{2q}+y^2}{x^2+y^2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \bar{R} = \frac{-2\left(q_{\theta\theta} + rq_r + r^2q_{rr}\right)}{r^2e^{2q}}.$$

Equation (38) together with the boundary conditions yields a linear boundary value problem of the form (1)–(3). Bernstein et. al. [10] have computed and analyzed solutions to this problem for a wide range of parameter values. We have also calculated solutions for $(a, 0, 1)$ with $-0.3 \leq a \leq 1.3$ using AMG3DP1. These preliminary tests show good agreement with the reported values in [10]. For example, with $a = 0.1$, Bernstein et. al. report the value 0.952 for the ADM-energy scaled with respect to the ADM-energy for the Schwarzchild black hole (this corresponds to $a = 0$ in this context), while we get the value 0.947 (a relative error of 0.52%).

Figures 32 and 33 show the mesh and isovalues of the computed solution for $a = 1$ (the data set $(1, 0, 1)$ for the parameters) for an intermediate adapted mesh having $8,905$ nodes and $46,038$ tetrahedra. Figure 32 shows a cut along the plane $x = 0$ while Figure 33 shows a cut along the plane $z = 0$. The qualitative behavior of the solution is similar to that reported by Bernstein et. al. The solution is radial in the $z = 0$ plane as expected, and the isovalues on the $x = 0$ plane show the Brill wave.

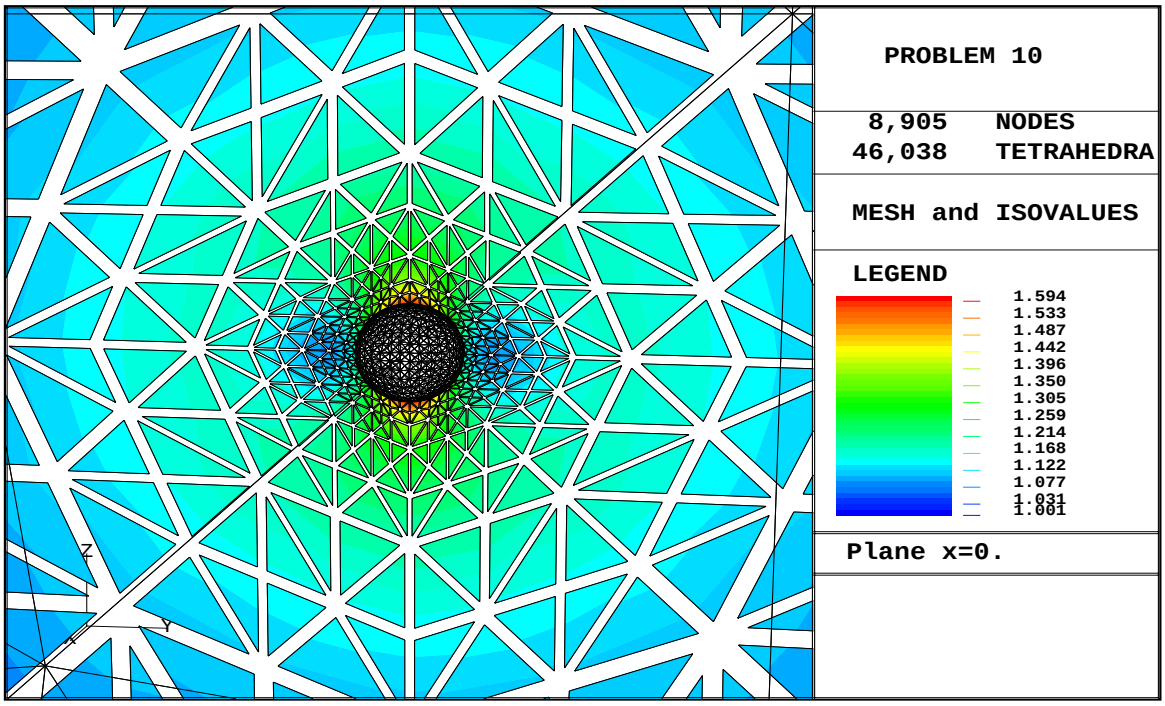

Figure 32. Problem 10: solution ($x = 0$).

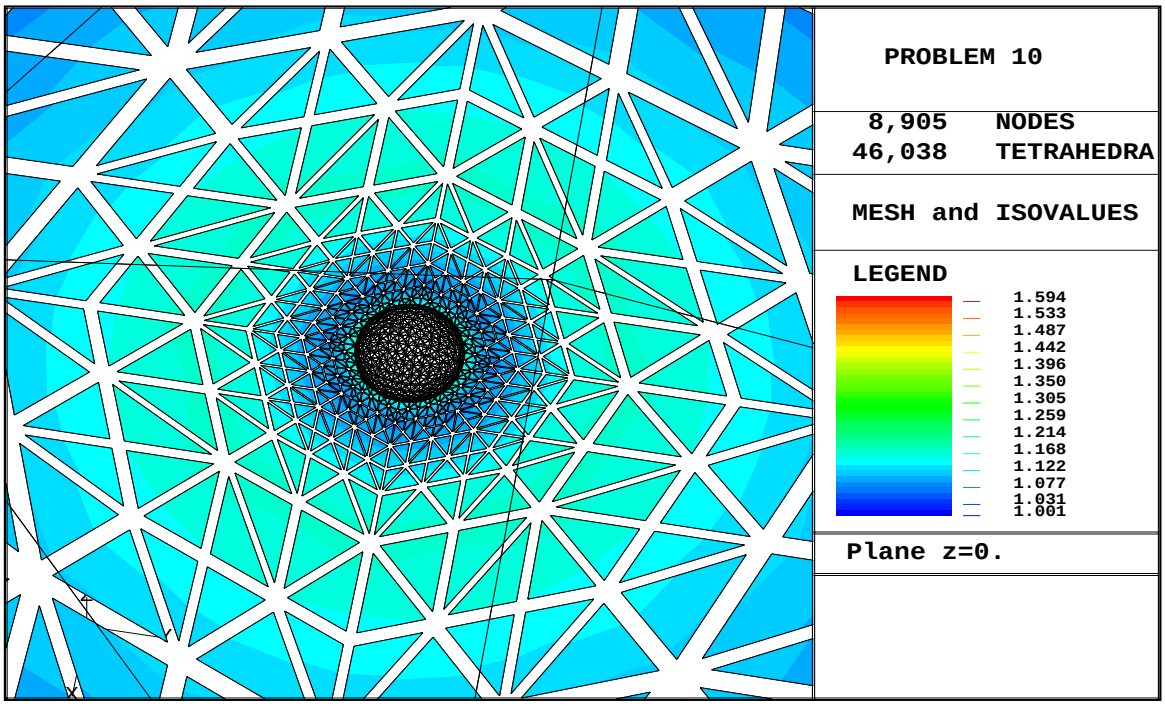

Figure 33. Problem 10: solution ($z = 0$).

Chapter 7

Summary

In this thesis, we have developed an adaptive finite element code for the solution of linear (and semilinear) boundary value problems in $\mathbb{R}^3$. We use tetrahedral meshes and linear finite elements. The solution of linear systems (semilinear problems are reduced to a sequence of linear problems using Newton's iteration) is achieved via a multigrid solver using Gauss Seidel smoothing. Given a mesh and a finite element solution, we compute a posteriori error estimators for each tetrahedron in the mesh, and show in Chapter 2 that the H^1 norm of the error is bounded by the error estimators on the mesh. The error estimators indicate which portions of the mesh need to be refined so as to capture steep gradients in the solution. An algorithm based on heuristic arguments that assigns a number to each tetrahedron in the mesh indicating the number of times the tetrahedron is to be bisected is presented in Chapter 5. This tries to ensure that the nested meshes produced have a geometric increase in the number of nodes. The algorithm performs well in practice. The actual subdivision of tetrahedra is accomplished by an algorithm based on repeated bisection of tetrahedra. We prove in Chapter 3 that the repeated bisection of tetrahedra give rise to a finite number of similarity classes, and hence degenerate shapes are avoided in our nested meshes. We also prove that a recursive algorithm for producing locally adapted meshes terminates in a finite number of steps. We show by considering examples in Chapter 3, 5, and 6, that the meshes produced by these algorithms are locally adapted. The numerical tests in Chapter 5 also show that AMG3DP1 demonstrates optimal rates of convergence for a wide class of boundary value problems.

The motivation for the development of AMG3DP1 comes from initial data problems in numerical relativity. We show, by considering various examples in Chapter 6, that AMG3DP1 effectively and efficiently solves such problems. In particular, by considering the scalar constraint that needs to be solved to produce initial data in the conformal imaging approach, we demonstrate that AMG3DP1 can efficiently solve this problem for arbitrary choices of the background metric. The code is validated by comparing the results with published results for various choices of initial data modeling different astrophysical situations. In most cases, the published results are for "essentially two dimensional" initial data, but we solve them as boundary value problems in three dimensions. Thus, we have shown that AMG3DP1 can be used to generate general three dimensional initial data for a wide range of astrophysical situations of interest.

Bibliography

[1] D. N. Arnold and L. Pouly. On the number of n-simplices created by bisection. Private Communication.

[2] R. Arnowitt, S. Deser, and C. W. Misner. *Gravitation-An introduction to current research*, chapter The dynamics of general relativity, pages 227–265. Wiley, 1962.

[3] I. Babuška and W. C. Rheinboldt. Error estimates for adaptive finite element computations. *SIAM J. Numer. Anal.*, 15:736–754, 1978.

[4] I. Babuška and W. C. Rheinboldt. A posteriori error estimates for the finite element method. *Int. J. Numer. Meth. Engrg.*, pages 1597–1615, 1978.

[5] R. E. Bank. *PLTMG: A software package for solving elliptic partial differential equations Users Guide 6.0.* SIAM, 1990.

[6] R. E. Bank, T. Dupont, and H. Yserentant. The hierarchical basis multigrid method. *Numr. Math.*, 52:427–458, 1988.

[7] R. E. Bank and D. J. Rose. Analysis of a multilevel iterative method for nonlinear finite element equations. *Math. Comp.*, 39(160):453–465, October 1982.

[8] E. Bänsch. Local mesh refinement in 2 and 3 dimensions. *Impact of Comp. in Sci. and Engg.*, 3:181–191, 1991.

[9] R. Becker, C. Johnson, and R. Rannacher. Adaptive error control for multigrid finite element methods. *Computing*, 55:271–288, 1995.

[10] David Bernstein, David Hobill, Edward Seidel, and Larry Smarr. Initial data for black hole plus brill wave spacetime. *Phys. Rev. D*, 50(6):3760–3782, September 1994.

[11] Jürgen Bey. Tetrahedral grid refinement. *Computing*, 55:271–288, 1995.

[12] J. M. Bowen and J. W. York. Time-symmetric initial data for black holes and black-hole collisions. *Phys. Rev. D*, 21:2047–2056, 1980.

[13] J. H. Bramble. *Multigrid methods.* Number 294 in Pitman Research Notes in Mathematics Series. Longman Scientific & Technical, 1993.

[14] J. H. Bramble and J. E. Pasciak. New estimates for multilevel algorithms including the v-cycle. *Math. Comp.*, 60(202):447–471, April 1993.

[15] P. G. Ciarlet. *The finite element method for elliptic problems.* North Holland, 1978.

[16] Ph. Clément. Approximation by finite element functions using local regularization. *RAIRO Modél. Math. Anal. Numér.*, 2:77–84, 1975.

[17] G. B. Cook. *Initial data for the two-body problem of general relativity.* PhD thesis, University of North Carolina at Chapel Hill, 1990.

[18] G. B. Cook. Initial data for axisymmetric black hole collisions. *Phys. Rev. D*, 44(10):2983–3000, November 1991.

[19] G. B. Cook, M. W. Choptuik, M. R. Dubal, S. Klasky, R. A. Matzner, and S. R. Oliveira. Three-dimensional initial data for the collision of two black holes. *Phys. Rev. D*, 47(4):1471–1490, February 1993.

[20] M. R. Duval, S. R. Oliveira, and R. A. Matzner. Solution of elliptic equations in numerical relativity using multiquadrics. In R. d'Inverno, editor, *Approaches to numerical relativity.* Cambridge University Press, 1992.

[21] A. Einstein and N. Rosen. The particle problem in the theory of general relativity. *Phys. Rev.*, 48:73–77, 1935.

[22] A. D. Kulkarni. Time-asymmetric initial data for the n black hole problem in general relativity. *J. Math. Phys.*, 25(4):1028–1034, 1984.

[23] A. D. Kulkarni, L. C. Shepley, and J. W. York. Initial data for n black holes. *Phys. Lett.*, 96A(5):228–230, July 1983.

[24] A. Liu and B. Joe. On the shape of tetrahedra from bisection. *Math. Comp.*, 63(207):141–154, July 1994.

[25] A. Liu and B. Joe. Quality local refinement of tetrahedral meshes based on bisection. *SIAM J. Sci. Comput.*, 16(6):1269–1291, November 1995.

[26] J. M. Maubach. Local bisection refinement for n-simplicial grids generated by reflection. *Siam J.Sci. Comput*, 16(1):210–227, January 1995.

[27] C. W. Misner. The method of images in geometrostatics. *Ann. Phys.*, 24:102–117, 1963.

[28] C. W. Misner, K. S. Thorne, and J. A. Wheeler. *Gravitation.* W. H. Freeman and Company, 1973.

[29] W. F. Mitchell. *Unified multilevel adaptive finite element methods for elliptic problems.* PhD thesis, University of Illinois at Urbana-Champaign, 1988.

[30] M. E. G. Ong. *Hierarchical basis preconditioners for second order elliptic problems in three dimensions.* PhD thesis, University of Washington, 1989.

[31] A. Quarteroni and A. Valli. *Numerical approximation of partial differential equations*, volume 23 of *Springer series in computational mathematics.* Springer-Verlag, 1994.

[32] John R. Rice and Ronald F. Boisvert. *Solving elliptic problems using ELLPACK*, volume 2 of *Springer series in computational mathematics*. Springer-Verlag, 1985.

[33] R. Verfürth. A review of a posteriori error estimation and adaptive mesh-refinement techniques. Technical report, Lecture Notes of a Compact Seminar at the TU Magdeburg, June 2–4 1993.

[34] R. Verfürth. A posteriori error estimates for nonlinear problems. finite element discretizations of elliptic equations. *Math. of Comp.*, 62(206):445–475, 1994.

[35] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM Review*, 34(4):581–613, December 1992.

[36] J. W. York. *Sources of gravitational radiation*, chapter Kinematics and dynamics of general relativity, pages 83–126. Cambridge, 1979.

[37] J. W. York and T. Piran. The initial value problem and beyond. In R. Matzner and L. Shepley, editors, *Spacetime and Geometry: The Alfred Schild Lectures*. University of Texas Press, Austin, 1982.

[38] S. Zhang. *Multi-level iterative techniques*. PhD thesis, The Pennsylvania State University, 1988.