UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

Multilevel Methods on Unstructured Grids (Ph.D. Thesis)

Susie Go

December 1998

CAM Report 98-54

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90095-1555

UNIVERSITY OF CALIFORNIA

Los Angeles

Multilevel Methods on Unstructured Grids

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in Mathematics

by

Susie Go

1999

© Copyright by

Susie Go

1999

The dissertation of Susie Go is approved.

Christopher Anderson

Bjorn Engquist

Xiaolin Zhong

Tony F. Chan, Committee Chair

University of California, Los Angeles

1999

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to thank the following people: Barry Smith of Argonne National Laboratory for his guidance with the PETSc library, Timothy Barth of NASA Ames for his expertise in unstructured grids and his invaluable help with the computational fluids application of my research, Ludmil Zikatanov of Penn State for all of his support and friendship, and Jun Zou of Chinese University of Hong Kong for all of his technical help. In addition, I would like to thank my committee members for their time and input, and Joe Oliger and RIACS for their major funding of this work. Finally, I would like to express my deepest thanks to my advisor, Tony F. Chan, for his technical, practical, and emotional support during all of my graduate studies.

PUBLICATIONS AND PRESENTATIONS

T. F. Chan, S. Go, and J. Zou. *Boundary treatments of multilevel methods on unstructured meshes.* Technical Report CAM 96–30, Department of Mathematics, University of California at Los Angeles, September 1996.

T. F. Chan, S. Go, and J. Zou. *Multilevel domain decomposition and multigrid methods for unstructured meshes: Algorithms and theory.* In R. Glowinski, J. Périaux, Z.-C. Shi, and O. Widlund, editors, *Domain Decomposition Methods in Science and Engineering*, pages 159–176. John Wiley and Sons, 1997.

T. F. Chan, S. Go, and L. Zikatanov. *Lecture notes on multilevel methods for elliptic problems on unstructured grids.* In *28th Computational Fluid Dynamics*, volume LS 1997-02 of *Lecture Series at the von Karman Institute for Fluid Dynamics*, March 1997.

T. F. Chan, S. Go, and L. Zikatanov. *Multilevel elliptic solvers on unstructured grids.* In M. Hafez, editor, *1997 Computational Fluid Dynamics*, 1997.

ABSTRACT OF THE DISSERTATION

Multilevel Methods on Unstructured Grids

by

Susie Go

Doctor of Philosophy in Mathematics

University of California, Los Angeles, 1999

Professor Tony F. Chan, Chair

The construction and performance of multilevel methods, including multigrid and domain decomposition methods, applied to elliptic problems on general unstructured grids will be studied. The difficulty in using unstructured grids with multilevel methods stems from the need to find a hierarchy of grids, which is not readily available for unstructured grids.

The problems of non-nested spaces and non-matching domain boundaries which arise in the coarsening of unstructured grids for multilevel methods will be discussed, and some analysis and computational approaches will be provided for the construction of unstructured methods which can achieve multigrid convergence rates. In particular, several different interpolants used in the coarse-to-fine transfer

of information between levels are constructed and shown to satisfy the approximation and stability properties which are essential for optimal convergence rates. Numerical results for model elliptic problems are presented and demonstrate that these methods retain optimal convergence rates.

Using the same tools and analysis, applications beyond that of the model elliptic problem are studied. An unstructured multigrid algorithm for solving elliptic eigenvalue problems will be developed by adapting an existing structured multigrid algorithm for the computation of several eigenvectors and eigenvalues.

This unstructured multigrid eigenvalue solver will be used to solve the practical and relevant problem of finding a partition of a graph using a spectral bisection algorithm. When the graph is a finite element mesh, an equivalence between the discrete and continuous Laplacian operator will be shown and used to adapt recently developed multilevel elliptic algorithms for unstructured grids to solving the graph partitioning problem with a true multigrid convergence rate.

CHAPTER 1

Introduction

The development of efficient numerical algorithms specifically designed for parallel computers has emerged as an important area of research as computational resources today continue to improve. Multilevel methods such as domain decomposition and multigrid are particularly well suited for use in a parallel setting and are powerful tools because they are optimal methods in the sense that their convergence speed can often be proven to be independent of the problem size.

Another rapidly-developing tool is the use of unstructured grids for solving large-scale problems on complicated geometries. The primary motivating reasons for using structured grids over unstructured grids are becoming less obvious as computers become faster and have more memory. Cartesian or mapped Cartesian grids are very popular because they are directional, so efficient methods can be used, such as the alternating direction implicit methods (ADI) and fast Fourier transforms (FFT). This was a very compelling reason to use such grids when computer resources were more limited. This structure, however, imposes severe limitations on the types of domains which can be considered. In addition, adaptive refinement cannot be easily done without affecting large portions of the grid, so the

ability to adapt the grids for resolving steep gradients in the solution is a source of difficulty.

Unstructured grids are becoming an increasingly popular alternative because they can provide the flexibility needed to adapt to rapidly changing or dynamic solutions as well as to more realistic and complex geometries [9, 23, 43]. These grids have irregular connectivity and so do not have to adhere to the strict structure of Cartesian-based grids, see Figure 1.1 making adaptation a local problem.

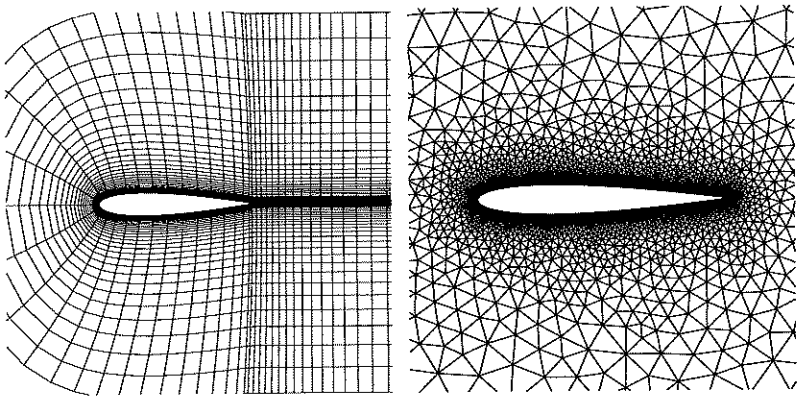


Figure 1.1: A structured grid (left) and unstructured grid (right).

These two powerful tools, multilevel methods and unstructured grids, however, do not lend themselves naturally for use with each other. Multilevel methods require a hierarchical grid structure. For structured grids, the hierarchy can be naturally recovered from the fine grid. Unstructured grids usually do not arise from some general refinement process so no natural coarse grids exist in unstructured meshes. The use of unstructured grids for multilevel methods will usually lead to a hierarchy of non-nested spaces since the coarse elements are not typically unions

of fine elements, so difficulties exist in identifying how to construct coarse grid problems/spaces/boundary conditions. The traditional solvers must to be modified so that their efficiency will not be adversely affected by this lack of structure. They must be redesigned to handle these issues without sacrificing too much in terms of complexity and performance.

The focus of this research is in the construction and analysis of multilevel methods, including multigrid and domain decomposition methods, on unstructured grids which are as efficient as their structured counterparts.

Many different approaches have been developed recently (see for instance [28, 38, 42, 45]). One technique generates a coarse grid hierarchy by using independent grids created by some grid generator (for example, the one which produced the original grid). Another approach uses agglomeration techniques to create a coarse space hierarchy. Still another method uses a graph approach by forming maximal independent sets (MIS) of the boundaries and interiors of the mesh and then retriangulating the resulting vertex set. The advantage of using a maximal independent set approach is that the grids are node-nested and thus efficient methods can be used to create the interpolation and restriction operators needed to transfer information from one level to the other. A disadvantage however, is that for complicated geometries, particularly in three dimensions, special care must be taken to ensure that the coarse grids which are produced are valid and preserve the important geometric features of the fine domain.

In Chapter 2, the model elliptic problem and the standard multigrid (MG) and domain decomposition (DD) algorithms will be introduced. A brief discussion of the necessary properties for optimal convergence rates will be given in this chapter as well. For more details, see [11, 12].

Chapter 3 defines interpolants which can be used to solve second order elliptic problems on unstructured meshes and which are shown to satisfy the stability and accuracy properties needed for optimal convergence. Numerical results on general elliptic problems are given to show that the unstructured methods are as efficient as their structured counterparts.

In the remaining chapters, demonstrations of the general use of the unstructured techniques beyond the model elliptic problem are shown by applying them to different types of problems. In Chapter 4, a well-known structured multigrid eigensolver is adapted for solving eigenvalue problems in the unstructured case by using the interpolants defined in 3 to construct an unstructured multigrid eigensolver.

An application of solving eigenvalue problems arises in one approach to solving the NP-complete graph partitioning problem, which is an important component of parallel computing, particularly in domain decomposition. In Chapter 5, it is shown that when the graph is a standard finite element mesh, the graph Laplacian is spectrally equivalent (up to a diagonal scaling) to the mesh Laplacian. This equivalence can be exploited to adapt recently developed multilevel elliptic algorithms for unstructured grids to solving the graph partitioning problem with a true

multigrid convergence rate and some numerical results using the unstructured elliptic eigenvalue algorithm of the previous chapter will be provided to demonstrate this.

Finally, the software environment which was used for the algorithm design of all the methods will be briefly presented in the appendix.

CHAPTER 2

Preliminaries

2.1 Model elliptic problem

Elliptic problems are one of the most extensively investigated problems in applied mathematics. Their relation to many physical models is well known and the theoretical and numerical results obtained in this area are very useful in practice. As a first approximation to more complicated physical and mathematical models (such as those in computational fluid dynamics), elliptic problems are sometimes the only ones for which rigorous theoretical results are known. The design of numerical methods for such model problems can often be adapted and applied to more complicated situations. Elliptic problems are also important in their own right, for example in computational fluid dynamics in the solution of the pressure equation, implicit time integration schemes, etc.

In this section, we will state the model problems we consider. Our goal is to design effective solvers for the resulting systems of linear equations after discretization. Detailed discussions of the finite element element discretizations that we use

can be found in [49, 20, 6, 32].

Let $\Omega \subset R^d$ be a polygonal ($d = 2$) or polyhedral ($d = 3$) domain. We will consider the following self-adjoint elliptic boundary value problem:

$$\begin{aligned} -\textstyle\sum_{i,j=1}^{d} \frac{\partial}{\partial x_i}(a_{ij}\frac{\partial u}{\partial x_j}) + b\,u = f \quad &\text{in} \quad \Omega, \\ u = 0 \quad &\text{on} \quad \Gamma_D, \\ \textstyle\sum_{i,j=1}^{d} a_{ij}\frac{\partial u}{\partial x_j}\gamma_i = 0 \quad &\text{on} \quad \Gamma_N, \end{aligned} \tag{2.1}$$

where $(a_{ij}(x))$ is symmetric, uniformly positive definite, and $b(x) \geq 0$ in Ω. Γ_D and Γ_N are such that $\bar{\Gamma}_D \cup \bar{\Gamma}_N = \partial\Omega$. $\gamma = (\gamma_1, \gamma_2)$ is the unit outward normal to $\partial\Omega$.

The variational (or Galerkin) formulation of this elliptic problem is: Find $u \in H_0^1(\Omega; \Gamma_D)$ such that

$$a(u, v) = f(v) \text{ for all } v \in H_0^1(\Omega; \Gamma_D), \tag{2.2}$$

where

$$\begin{aligned} a(u, v) &= \int_\Omega (\sum_{i,j=1}^{d} a_{ij}\frac{\partial u}{\partial x_i}\frac{\partial v}{\partial x_j} + b(x)uv)dx, \\ f(v) &= \textstyle\int_\Omega f(x)v dx. \end{aligned} \tag{2.3}$$

Here $H_0^1(\Omega; \Gamma_D)$ denotes the Sobolev space which contains functions which vanish on Γ_D with square integrable first derivatives.

We will use the simplest finite element discretization of the elliptic problem (2.2). Cover Ω with simplicial finite elements (triangles in R^2 and tetrahedra in R^3), $\mathcal{T}^h$. Then the discrete problem can be formulated as follows:

Find $u_h \in V_h$ such that:

$$a(u_h, v_h) = f(v_h) \text{ for all } v_h \in V_h, \tag{2.4}$$

where V_h is the finite dimensional subspace of $H_0^1(\Omega; \Gamma_D)$ consisting of continuous functions linear on each of the simplexes forming the partition.

The values of the discrete solution on the grid nodes are then determined by solving the resulting system of linear equations:

$$Au = f, \tag{2.5}$$

where A is a symmetric and positive definite matrix, f is the right hand side and the nodal values of the discrete solution u_h will be obtained in u after solving the system (2.5). To obtain an accurate enough approximate solution of (2.2), one often has to solve huge discrete problems which are badly conditioned, with condition number growing like $O(h^{-2})$, where h is the characteristic mesh size.

2.2 Iterative methods

Linear iterative methods are methods which solve a linear system of equations $Au = f$ by taking an approximate solution, u^{k-1}, and improving it by adding a correction term to obtain a better approximation, u^k:

$$u^k = u^{k-1} + e^{k-1}, \tag{2.6}$$

where the correction term, e^{k-1}, is defined as the solution to the following error equation

$$Ae^{k-1} = f - Au^{k-1}. \tag{2.7}$$

If equation (2.7) were solved exactly, the iterate, u^k, would be the exact solution to 2.5 and the iteration would stop. Typically, the error equation is only solved approximately, so that

$$e^{k-1} = M^{-1}(f - Au^{k-1})$$

where M^{-1} is an approximate inverse of A. Then the linear iterative method is written as

$$u^k = u^{k-1} + M^{-1}(f - Au^{k-1}), \quad k = 0, 1, 2, \ldots \tag{2.8}$$

M^{-1} is called the iterator. Note that

$$u^k - u = (I - M^{-1}A)^k(u - u^0).$$

Thus as $k \to \infty$, the iterative method converges to the solution for any initial guess, u^0, if and only if

$$\rho(I - M^{-1}A) < 1. \tag{2.9}$$

where $\rho(I - M^{-1}A)$ denotes the spectral radius of $(I - M^{-1}A)$. In developing preconditioners, the approximate inverse M^{-1}, should be to easy to apply and be such that $M^{-1}A$ is small.

Splitting A in the standard way: $A = D - L - U$, where D is the diagonal of A, and $-L$ and $-U$ are the strictly lower and upper triangular parts of A, respectively, an approximate inverse for A given by

$$M^{-1} = \omega$$

yields Richardson's method, while the choice

$$M^{-1} = D^{-1}$$

yields the Jacobi method, and

$$M^{-1} = (D - L)^{-1}$$

gives the Gauss-Seidel method.

The rate of convergence for these simple relaxation schemes depends on the condition number of A. For finite element and finite difference equations such as (2.5), the asymptotic convergence rate for Gauss-Seidel method is of order $1 - O(h^2)$, which makes the method impractical for small mesh sizes h.

These basic methods can be accelerated by Krylov subspace methods, e.g. the preconditioned conjugate gradient (PCG) when A is SPD and the preconditioned GMRES (PGMRES) for general non-symmetric A. The convergence of the PCG method depends on the condition number of the matrix $M^{-1}A$, as seen in the following well-known theorem:

Theorem 2.2.1 *Let A and M be SPD matrices and $\|v\|_A = (Av, v)$. Let u be the solution of the system (2.5). Then for the k-th iterate u^k the following inequality holds*

$$\|u - u^k\|_A \leq 2\left(\frac{\sqrt{\kappa(M^{-1}A)} - 1}{\sqrt{\kappa(M^{-1}A)} + 1}\right)^k \|u - u^0\|_A. \qquad (2.10)$$

Our particular interest will be focused on multilevel methods (such as domain decomposition methods and multigrid methods) used as preconditioners in PCG. The popularity of these methods as preconditioners is based on the fact that they exactly fit in the applications where finite element or finite difference method is used. In other words, the design of such preconditioners uses the properties of finite element spaces which allows precise optimal constructions and theoretical analysis to be done.

2.3 Multilevel methods

For many practical problems, the system of linear equations which arises from finite element or finite difference discretizations becomes very large. A challenge is how to effectively solve such large systems of linear equations, since direct methods face the problem of excessive memory requirements and number of the floating point operations needed. Because of this, many researchers have turned to iterative methods. As parallel computers become more dominant, more attention is being focused on multilevel methods such as multigrid and domain decomposition

methods. These methods are popular because the amount of work required to solve a problem is on the order of the number of unknowns, while the convergence rates are independent of the problem size.

2.3.1 Multigrid methods

In this section, we briefly describe the multigrid methods for solving linear systems of discrete equations. We will consider the case where these systems are obtained via finite element discretization of an elliptic partial differential equation. Detailed discussion on multigrid methods can be found in standard references, e.g. Briggs [7], Bramble [2], Hackbusch [29], and Xu [52, 53].

The idea behind multigrid methods is based on the fact that simple relaxation schemes such as Gauß-Seidel, Jacobi and Richardson possess a good smoothing property; that is, they reduce the highly oscillatory part of the error very well in just a few inexpensive iterations. This part of the error lies in the subspace spanned by the eigenvectors corresponding to large eigenvalues, i.e. the high frequencies. The global error, or the low frequencies unfortunately cannot be corrected well by such iterative schemes and this is where multigrid helps. The low frequencies from fine grid (say original one) are transfered to the coarse grid, where they behave like high frequencies, and are smoothed quickly by a simple relaxation scheme. Recursive application of this idea leads to the multigrid method.

Denote the space which contains the solution u by V_J and assume that the coarse grids are given and with each grid associate a finite dimensional space (like V_J for the fine grid). We denote these spaces by $V_0, \ldots, V_{J-1}$. To unify the notation in this section we define $A_J := A$. We assume that the operators A_k, $k = 0, \ldots J-1$, are given (these operators correspond to different approximations of A on the coarse grids). We also assume that the prolongation operator R_k^T and the smoothing operators S_k are also given. One can consider the action of the smoother on $g \in V_k$ as a fixed number of Gauß-Seidel or Jacobi iterations with right-hand side g and zero initial guess.

The multigrid method can be viewed as a way of defining a preconditioner, M_J, and can be described in matrix notation by the action of $M_J^{-1}g$. In the simplest case when one pre- and post-smoothing steps are applied, the action of M_k^{-1} is then obtained through the following steps:

Algorithm 2.3.1 *(V-cycle correction scheme multigrid)*

0. If $k = 0$, then $x_0 = A_0^{-1} f_0$.

1. Presmoothing: Apply one transposed smoothing iteration with initial guess, $x_k^0 = 0$.

$$\tilde{x}_k = x_k^0 + S_k^T(f_k - A_k x_k^0) = S_k^T f_k$$

2. Descend to coarse level:

a. Restrict the residual, f^{k-1}

$$f_{k-1} = R_k(f_k - A_k \tilde{x}_k)$$

b. "Solve" $A_{k-1}x_{k-1} = f_{k-1}$ (recursive call to V-cycle MG

c. Interpolate back and correct:

$$\hat{x}_k = \tilde{x}_k + R_k^T x_{k-1}$$

3. Postsmoothing: Apply one smoothing iteration with initial guess $\hat{x}_k$.

$$x_k = \hat{x}_k + S_k(f_k - A_k \hat{x}_{k+1})$$

Note that the above definition is recursive. For some general right-hand side, g, the action of $M_k^{-1}g$ is defined in terms of $M_{k-1}^{-1}g$. Consider now the simplest case: a two-level method (when $J = 1$). Then the correction scheme multigrid method can be written compactly (omitting the index 1) as

$$\begin{aligned} M^{-1}g &= [S + S^T - SAS^T \\ &\quad +(I - SA)R^T A_0^{-1} R(I - AS^T)\big]\, g. \end{aligned}$$

2.3.2 Domain decomposition methods

Domain decomposition (DD) methods are divide-and-conquer methods which take a large problem defined on a physical domain, and appropriately decompose it into many smaller problems defined on subdomains. These smaller subdomain problems can then be solved quickly and independently of each other and their solution suitably combined, usually via an iterative process to obtain the solution to the original problem. Domain decomposition methods fall into two broad categories: overlapping DD (Schwarz methods) and nonoverlapping DD (substructuring or Schur complement methods). Our description here follows that in Chan-Mathew [15]; see also the recently published book by Smith, Bjørstad and Gropp [48]. We will not discuss the nonoverlapping domain decomposition methods here. For a detailed description and investigation of these methods we refer to [15, 48].

2.3.2.1 Overlapping DD

In overlapping DD methods, a set of p overlapping subdomains are formed by taking a set of nonoverlapping subdomains $\{\Omega_i'\}_{i=1}^p$, and extending them to larger subdomains, $\{\Omega_i\}_{i=1}^p$ by some small distance, $\delta > 0$ (see Fig. 2.1). We will assume that $\partial\Omega_i$ does not cut through any element. Corresponding to each subdomain Ω_i, we define a subspace V^i of V^h by

$$V^i = \{v \in V^h; \quad v = 0 \quad \text{on} \quad \Omega \setminus \Omega_i\}.$$

The partitioning induced by such a decomposition amounts to an overlapping block decomposition of the system (2.5). Thus, the overlapping DD methods can be thought of as block iterative solvers, either overlapping block Jacobi or block Gauß-Seidel, depending on whether or not the the most updated iterates are used.

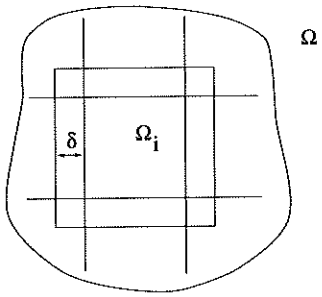


Figure 2.1: Generating a set of overlapping subdomains.

The main ingredients required in all DD methods are:

- *Restriction operators*: Let R_i be the $n_i \times n$ restriction matrix of 1's and 0's which takes a full-length vector in R^n and maps it to a restricted vector in R^{n_i}, where n_i denotes the number of unknowns in subdomain Ω_i. The effect on an n-vector is injection onto the subdomain, Ω_i.

- *Extension operators*: Let R_i^T be the $n \times n_i$ extension matrix, which is defined as the transpose of the restriction matrix, R_i. The effect on an n_i-vector is identity on the subdomain, Ω_i, and zero extension outside the subdomain, i.e. on $\Omega \setminus \Omega_i$.

- *Subdomain operators*: Define the local stiffness matrix on Ω_i to be $A_i =$

$R_i A R_i^T$, where $A_i \in R^{n_i \times n_i}$. Because the restriction and interpolation matrices consist only of 0's and 1's, the local stiffness matrices are simply principal submatrices of A.

- *Subdomain solvers*: Let A_i^{-1} symbolically denote the solver for the restricted operator. These can be either exact or inexact solvers.

The additive Schwarz (block Jacobi) method on p subdomains is given by:

$$u^{k+i/p} = u^{k+(i-1)/p} + R_i^T A_i^{-1} R_i (f - Au^k), \quad i = 1, ..., p.$$

In this form, it is seen that corrections are done simultaneously on p subdomains. Rewriting this as one equation reveals the preconditioned iterative method:

$$u^{k+1} = u^k + M_{as}^{-1}(f - Au^k)$$

where the preconditioner M_{as} is given by:

Additive Schwarz preconditioner.

(block Jacobi on A)

$$M_{as}^{-1} = \sum_{i=1}^{p} R_i^T A_i^{-1} R_i. \tag{2.11}$$

Instead of simultaneous corrections, the corrections can also be done successively, to yield the multiplicative Schwarz (block Gauß-Seidel) method, for

$i = 1, ..., p$:

$$u^{k+i/p} = u^{k+(i-1)/p} + R_i^T A_i^{-1} R_i (f - A u^{k+(i-1)/p}).$$

Because the most currently updated information is used, this method will generally converge faster than additive Schwarz. The drawback is that it is less parallel (but this can be remedied by appropriate coloring of the subdomains).

2.3.2.2 Coarse grid

The domain of dependence for elliptic problems is the entire domain, but because Schwarz methods decompose the problem into smaller, independent problems, information from one subdomain must travel large distances to reach another subdomain. To avoid deterioration of the convergence rates of these methods, some sort of mechanism for the global transfer of data is needed. This is achieved, to some degree, by the overlapping of subdomains in the Schwarz methods. More overlap leads to more coupling between subdomains.

However, this adds redundant work and communications overhead if too much overlap is introduced. Dryja and Widlund [25, 26] showed that the condition number for additive Schwarz (2.11) is given by:

$$\kappa(M_{as}^{-1} A) = O\left(H^{-2}\left(1 + \left(\frac{H}{\delta}\right)^2\right)\right).$$

The condition number is independent of h. For sufficient amount of overlap (choosing $\delta = O(H)$), the condition number is $O(H^{-2})$ and so will increase as H tends

to zero. This means that the method will not be scalable to a large number of processors.

This deterioration can be remedied by introducing a coarse grid to achieve additional global coupling. In addition to the subdomain restriction, interpolation and stiffness matrices used in the one-level Schwarz methods, we need coarse versions of them: $R_H, R_H^T, A_H = R_H A R_H^T$, and A_H^{-1}. Here, R_H and R_H^T will instead be the full weighting restriction and linear interpolation matrices, respectively, which are commonly used in multigrid methods. The two-level additive Schwarz preconditioner can then be written as:

Additive Schwarz preconditioner with coarse grid.

$$M_{asc}^{-1} = R_H^T A_H^{-1} R_H + \sum_{i=1}^{p} R_i^T A_i^{-1} R_i.$$

It can be shown that the condition number for this two-level method is

$$\kappa(M_{asc}^{-1} A) = O(1 + (H/\delta)^2),$$

and the method can be made independent of H, h with sufficient overlap by choosing $\delta = O(H)$.

2.3.2.3 Multilevel Schwarz

Multilevel Schwarz is an extension of two-level Schwarz with L different coarse levels, each level being decomposed into p_l subdomains as previously described. We will denote the i^{th} subdomain on the l^{th} level as: Ω_i^l. Several different variants of multilevel Schwarz can be created, depending on when the most currently updated information is used:

- Fully additive multilevel methods would be additive among subdomains on the same level as well as additive between levels.
- Multilevel methods which are multiplicative among subdomains on the same level, but additive between levels can be viewed as "additive MG".
- Classical V-cycle MG can be viewed as a multilevel Schwarz method which is multiplicative both among subdomains on the same level as well as between levels.

The fully additive multilevel Schwarz preconditioner can be written as:

Fully additive multilevel Schwarz preconditioner.

$$M_{mlas}^{-1} = \sum_{l=1}^{L} \sum_{i=1}^{p_l} (R_i^l)^T (A_i^l)^{-1} (R_i^l).$$

2.4 Introduction to convergence theory

As mentioned in Section 2.2, the estimate of the convergence rate of the PCG requires an estimate of the upper bound of $\kappa(M^{-1}A)$. In particular, estimates on the extreme eigenvalues of $M^{-1}A$ must be obtained. In this section, we first give a general framework for bounding $\kappa(M^{-1}A)$ and then we show how such an analysis can be carried out for the overlapping domain decomposition method. Such an analysis can show (or predict) the convergence rate and in most cases gives a good guess as to how the parameters and approximate operators should be chosen in order to get an optimal iterative method. For a similar approach in analyzing the convergence properties of iterative methods using general subspace splittings for structured meshes, we refer to [52].

We shall adopt a matrix approach for analyzing the domain decomposition methods, in the hope that it is more intuitive and easier to understand. Such a presentation of the analysis can also be found in (see e.g. [48]). More references concerning the theoretical analysis of the domain decomposition methods also can be found there.

Let Ω be a fixed domain in R^d. The norm in Sobolev space $H^k(\Omega)$ is defined to be:

$$\|u\|_k = \left(\sum_{|\alpha| \leq k} \int_\Omega [D^\alpha u(x)]^2 dx \right)^{\frac{1}{2}},$$

and the seminorm in $H^k(\Omega)$ is defined by:

$$|u|_k = \left(\sum_{|\alpha|=k} \int_\Omega [D^\alpha u(x)]^2 dx \right)^{\frac{1}{2}},$$

where $\alpha = (\alpha_1, \ldots, \alpha_d)$ is a multi-index, $D^\alpha = \dfrac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \ldots x_d^{\alpha_d}}$ and $|\alpha| = \alpha_1 + \ldots + \alpha_d$. When the domain (e.g. Ω) needs to be emphasized or clarified, the notation for the seminorm and norm will be $|u|_{k,\Omega}$ and $\|u\|_{k,\Omega}$, respectively.

2.4.1 Subspace correction framework: matrix formulation

Our initial setting in matrix form is as follows: Let Ω be covered by p overlapping subdomains Ω_i, $i = 0, 1, \ldots, p$. Each subdomain Ω_i corresponds to a subspace $V_i \subset R^n$. The subspaces are defined through the restriction operators $R_i \in R^{n_i \times n}$, $i = 0, 1, ..., p$, and we set $V_i = \text{Range}(R_i)$

REMARK. Note here that we will interchangeably use the notation for the coarse grid versions denoted with subscript H in the previous section, with the subscript 0, when convenient. Here, V_0 denotes the coarse space.

We wish to construct a preconditioner for solving the following linear algebra problem

$$Au = f, \quad A \in R^{n \times n} \text{ is SPD.} \tag{2.12}$$

Let us first explain the intuition behind the construction of a preconditioner based on this splitting of R^n. It is natural to take the best approximation to the

solution from each subspace, and then to extend these different approximations to the whole R^n somehow in order to get a global solution. Thus the question is: What is the best correction to the k-th iterate u^k from V_i?

If we measured the error in the A-norm, $\|\cdot\|_A$, then this question can be reformulated as the following minimization problem:

$$\min_{y_i} \|(u^k + R_i^T y_i) - A^{-1} f\|_A. \tag{2.13}$$

The solution is given by:

$$y_i = (R_i A R_i^T)^{-1} R_i (f - Au^k) = A_i^{-1} R_i (f - Au^k). \tag{2.14}$$

The next iterate is then obtained via the equation (note that we correct here only in one subspace V_i)

$$u^{k+1} = u^k + R_i^T A_i^{-1} R_i (f - Au^k) \tag{2.15}$$

Example. The Jacobi iteration (see Section 2.2) corresponds to the splitting $V_i = span\{e_i\}$ where e_i is the i-th unit coordinate vector. The restrictions, R_i, in this case are defined as $R_i v = (v, e_i)e_i$.

Performing these subspace corrections simultaneously gives the additive subspace correction preconditioner:

$$M_{asc}^{-1} = \sum_{i=0}^{p} R_i^T A_i^{-1} R_i.$$

Defining now the projections $P_i \equiv R_i^T A_i^{-1} R_i A$, for $i = 0, \ldots p$, we get

$$M_{asc}^{-1} A = \sum_{i=0}^{p} P_i.$$

As we pointed out earlier, the convergence of the PCG method depends on the condition number of $M^{-1}A$. Thus our goal is to find an upper bound for $\kappa(M^{-1}A)$ which amounts to finding an upper bound for $\lambda_{\max}(M_{asc}^{-1}A)$ and a lower bound for $\lambda_{\min}(M_{asc}^{-1}A)$.

The estimate on the upper bound for $\lambda_{\max}(M_{asc}^{-1}A)$ is easier and it follows directly from the following simple lemmas.

Lemma 2.4.1 *P_i is a projection in $(\cdot,\cdot)_A$, i.e.*

$$AP_i = P_i^T A, \quad P_i^2 = P_i \quad \|P_i\|_A \leq 1.$$

Proof. This follows by direct verification. □

Lemma 2.4.2 *The maximal eigenvalue of the preconditioned matrix satisfies the following inequality:*

$$\lambda_{\max}(M_{asc}^{-1}A) \leq p + 1.$$

Proof. This follows from the simple fact that

$$\rho\left(\sum_{i=0}^{p} P_i\right) \leq \left\|\sum_{i=0}^{p} P_i\right\|_A \leq \sum_{i=0}^{p} \|P_i\|_A = p + 1.$$

□

REMARK. The bound given in the previous lemma can be easily improved to:

$$\lambda_{\max}(M_{asc}^{-1}A) \leq n_c + 1 \equiv c_1,$$

where n_c is the *number of colors to color* Ω_i*'s* in such a way that no two neighboring subdomains are colored the same color.

We next give an estimate on the lower bound for $\lambda_{\min}(M_{asc}^{-1}A)$. This estimate is based on the following *Partition Lemma* which plays a crucial role in the convergence analysis of the domain decomposition methods.

Lemma 2.4.3 ***(Partition Lemma).*** *(Matsokin-Nepomnyaschikh [41], Lions [40], and Dryja-Widlund [24, 25]). Assume that there exists a constant* c_2 *such that*

$$\min_{\substack{u=\sum u_i \\ u_i \in V_i}} \sum_{i=0}^{p} \|u_i\|_A^2 \leq c_2 \|u\|_A^2. \qquad (2.16)$$

then

$$\lambda_{\min}(M_{asc}^{-1}A) \geq \frac{1}{c_2}.$$

The assumption we have made in the partition lemma (equation (2.16)) means that for any given u, a stable decomposition must exist in the sense that the sum of the "energy" of all the pieces u_i lying in V_i is bounded by the global energy norm

of the decomposed vector. This assumption can be viewed as a condition on the V_i's, i.e. the subspaces must not introduce oscillations (high energy components) in u_i.

Combining lemmas 2.4.1– 2.4.3, we get our main theorem:

Theorem 2.4.1 *If assumption (2.16) holds, then for the condition number $\kappa(M_{asc}^{-1}A)$, can be bounded by:*

$$\kappa(M_{asc}^{-1}A) \leq c_1 c_2. \tag{2.17}$$

This result suggests how to construct the decompositions in order to obtain optimal preconditioners. It is immediately seen that we want the constants c_1, c_2 to be independent of the problem parameters such as the number of subdomains, the characteristic mesh sizes h and H, jumps in the coefficients of the underlying PDE, etc. It is also desirable to make c_1 and c_2 as small as possible in order to get a condition number close to 1. But c_1 and c_2 depend on the size of overlaps in the subspaces V_i. More overlap will decrease c_2, but the number of colors c_1 will increase. On the other hand, small overlap will lead to large c_2 and small c_1. Thus the space decompositions have to be made in such a way to ensure that the product $c_1 c_2$ is as small as possible.

2.4.2 Application to two-level overlapping domain decomposition methods

As an example of the application of the above theory, we will present a detailed estimate for the condition number of the two-level Schwarz method.

2.4.2.1 The intuitive idea

As in the previous section, we first present the basic intuitive idea using a simple 1D version of (2.1).

We want a splitting which satisfies the partition assumption (2.16). Take Ω to be a fixed open interval on the real line and cover Ω with p overlapping subdomains Ω_i, $i = 1, \ldots, p$ (see fig 2.2). Consider the partition of unity θ_i corresponding to this covering. By construction the functions θ_i satisfy

$$\sum_{i=1}^{p} \theta_i = 1, \qquad 0 \leq \theta_i \leq 1, \qquad |\theta_i|_{1,\infty} \leq \delta^{-1}, \tag{2.18}$$

where δ is the size of the overlap and $|\theta|_{s,\infty}$ denotes the maximum, norm, i.e. the maximum of the s-th derivative of θ_i. We define $u_i = \theta_i u$, so we have $u = \Sigma_{i=0}^{p} u_i$.

Since A corresponds to a discretization of the second order elliptic operator, $\frac{d}{dx}\alpha(x)\frac{d}{dx}$, it is easy to see that the A-norm and the H^1-seminorm are equivalent in this case: $\|u\|_A \approx \|du/dx\|_0$. Our goal is to bound $\|u_i\|_A$ by $\|u\|_A$. Looking at Fig. 2.2, we see that the function u_i changes from $\|u\|_0$ to 0 over a distance δ and

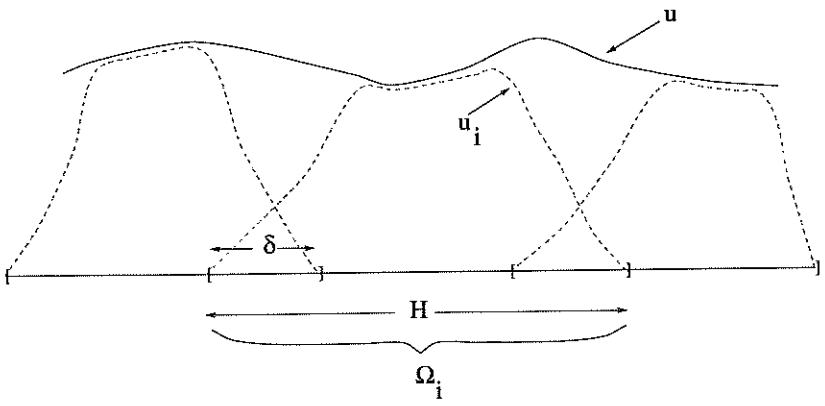


Figure 2.2: No coarse grid.

we get:

$$\|u_i\|_A^2 = \left\| \left(\frac{du}{dx} \right) \right\|_0^2 \leq c \left(\frac{\|u\|_0}{\delta} \right)^2 .$$

We still need to bound $\|u\|_0$ by $\|u\|_A$. But the function, u, satisfying homogeneous Dirichlet boundary conditions, cannot change rapidly over the interval Ω if there is no significant change in the derivative. The well-known Poincaré inequality estimates the norm of the function with the norm of derivatives and its application leads to the following:

$$\left(\frac{\|u\|_0}{\delta} \right)^2 \leq \tilde{c} \left(\frac{\|u\|_A}{\delta} \right)^2 .$$

After summing over all subdomains, we get:

$$\sum_{i=1}^{p} \|u_i\|_A^2 \leq O\left(\frac{1}{\delta^2} \right) \|u\|_A^2 .$$

Therefore,

$$c_2 \;=\; O\left(\frac{1}{\delta^2} \right) = O\left(\left(\frac{H}{\delta} \right)^2 \frac{1}{H^2} \right) .$$

From these inequalities one may conclude that if the overlap is of size $O(H)$, then $\kappa(M^{-1}A) = O(H^{-2})$, which is an improvement over $O(h^{-2})$, but is still unsatisfactory. We can see that the overlapping subdomains alone cannot provide a stable partition of u.

It turns out that this dependence on H can be eliminated by using a global coarse space, V_H, which couples all the subdomains. The idea is to construct a coarse grid approximation u_H to u satisfying the following two important properties:

$$\|u_H\|_A \leq c\|u\|_A \tag{2.19}$$

$$\|u - u_H\|_0 \leq cH\|u\|_A \tag{2.20}$$

Define $w = u - u_H$ and the following partition of u:

$$u_i = \theta_i(u - u_H), \quad u = u_H + \sum_{i=1}^{p} u_i. \tag{2.21}$$

Proceeding as before and taking into account that now the pieces u_i change from 0 to $O(H)$ (not to 1 because of the approximation property (2.23)), we have

$$\|u_i\|_A^2 \leq c\left(\frac{H\|u\|_A}{\delta}\right)^2 \leq c\left(\frac{H}{\delta}\right)^2 \|u\|_A^2.$$

If we make the natural assumption that $\delta = O(H)$, the bound for c_2 now reads

$$c_2 = O\left(\left(\frac{H}{\delta}\right)^2\right) = O(1).$$

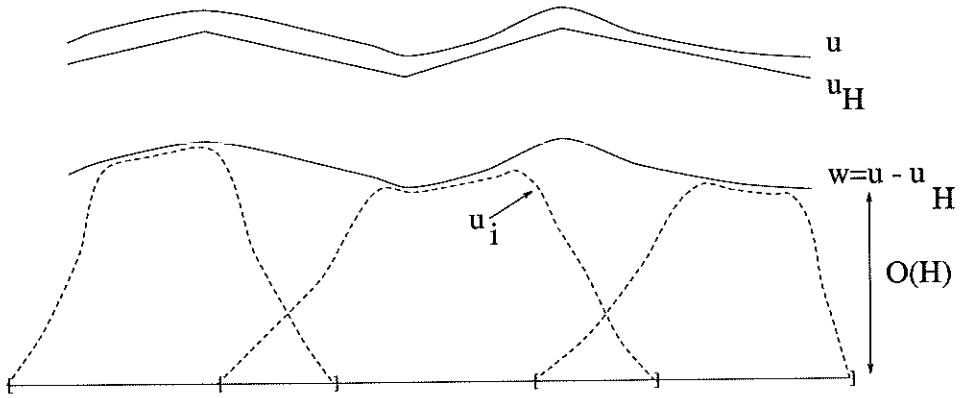


Figure 2.3: With coarse grid.

This estimate shows how the condition number can be improved and actually such a choice of M^{-1} gives a good preconditioner. Thus, we can see that the role of the coarse grid V_H is to make $\|u - u_H\|$ small enough $(O(H))$, so that it can be partitioned in a stable manner.

We would like to comment on the choice of u_H. As it can be seen (compare to section 2.3.2), u_H is a purely theoretical construction and there is no need of its use in the algorithm. One possible choice is $u_H = \mathcal{R}_H u$, where $\mathcal{R}_H$ is a kind of interpolation or projection operator. Of course, $\mathcal{R}_H$ must satisfy properties similar to (2.19) and (2.20) namely

$$|\mathcal{R}_H u|_1 \leq c|u|_1 \quad \text{(stability)} \tag{2.22}$$

$$\|\mathcal{R}_H u - u\|_0 \leq cH|u|_1. \quad \text{(approximation)} \tag{2.23}$$

A natural candidate for such an operator is the nodal value interpolant on the coarse grid, $u_H = I_H u$. A drawback of such a choice is that in $3D$ this interpola-

tion does not satisfy the stability property (2.22). To see this, we take w to be the basis function ψ_i^h associated with the grid node x_i. Then we have

$$\begin{aligned} |w|_1^2 &= \int_{\Delta_h} \left(\tfrac{1}{h}\right)^2 = O(h) \\ |I_H w|_1^2 &= \int_{\Delta_H} \left(\tfrac{1}{H}\right)^2 = O(H). \end{aligned}$$

The last estimate shows that the stability requirement is violated.

If the grid is structured then a good and stable coarse grid approximation to the elements of V_h is the *L_2-projection* Q_H from $V_h \to V_H$ and we can define $\mathcal{R}_H = Q_H$, i.e. $u_H = Q_H u$. It is known that this u_H satisfies the stability and approximation properties (2.22) and (2.23) (see Xu [52] or Dryja and Widlund [24]).

2.4.3 Convergence of multigrid methods

The convergence properties of *multigrid* methods (see Section 2.3.1) depend on many parameters. One can vary the number of smoothing steps, the smoothing operators, the interpolation and restriction operators, coarse grid operators, etc. There are two main approaches in constructing multigrid preconditioners. One of them uses nested subspace splittings of V_h and the other one uses non-nested spaces or specially interpolated bilinear forms (coarse grid matrices). The discussion of the convergence in both these cases is given in [2, 3, 52, 53]. Here we give the simplest convergence result in the case of *nested* spaces and the so-called full elliptic

regularity assumption:

$$\|u\|_2 \leq c\|F\|_0.$$

where u is the solution, F is the right hand side of (2.1).

First, consider the case when the spaces $V_0, \ldots, V_1$ are nested, i.e. $V_0 \subset V_1 \subset \ldots V_{J-1} \subset V_J = V_h$. Again, the stability and approximation properties (2.22) and (2.23) are crucial in the convergence theory. The stability property (2.22) is automatically satisfied when the spaces are nested. From the regularity assumption, the following approximation property follows (see Xu [52, 53]):

There exists a constant, c_1, independent of the mesh parameters (i.e. of the mesh size h) such that

$$\|I - P_{k-1}v\|_A^2 \leq c_1 \frac{1}{\rho(A_k)} \|A_k v\|_2^2 \quad \forall v \in V_k, \tag{2.24}$$

where P_k denotes the elliptic projection defined by $(AP_k u, v) = (Au, v) \ \ \forall v \in V_k$.

The other operator involved in the definition of M_J^{-1} is the smoother and we make the following assumption on it

$$\frac{c_0}{\rho(A_k)}(v, v) \leq (S_{symm,k}v, v) \leq (A_k^{-1}v, v). \tag{2.25}$$

The smoother $S_{symm,k}$ is the symmetric version of S_k and is defined as: $S_{symm,k} := S^T + S - S^T A S$. Inequalities of the type (2.25) are satisfied by the Gauß-Seidel method.

An important thing to mention for the choice of the smoother is that we are

trying to choose smoother which will quickly capture the high frequency components of the error, and we are not going to use it as a solver. For example if the matrix A_J corresponds to the five point finite difference stencil, it can be seen that the Jacobi method (with $\omega = 1$) is not good for a smoother. One must use the damped Jacobi method with $\omega < 1$.

The subspace correction framework presented in the previous section applies to multigrid methods as well. As long as the stability and approximation properties are verified, the following convergence result holds:

Theorem 2.4.2 *Under the assumptions (2.24) and (2.25), the following estimate is true:*

$$\|I - M_J^{-1}A\|_A \leq 1 - \frac{c_0}{c_1}. \tag{2.26}$$

A convergence result similar to Theorem 2.4.2 is also true in the *non-nested* case (see Bramble et.al. [3] or Chan-Zou [18, 19] for unstructured grids).

The construction of interpolation operators for unstructured grids which satisfies the stability property may be an issue and in the next chapter, some possible stable definitions will be discussed. Additional details can also be found in [18] and [19].

CHAPTER 3

Boundary Considerations

Unstructured multilevel methods for solving linear systems like (2.5) require a hierarchy of coarse grids. Grids which are node-nested have the advantage that they can be automatically generated and that efficient methods can be used to create the interpolation and restriction operators needed to transfer information from one level to the other. Disadvantages are that for complicated geometries, particularly in three dimensions, special care must be taken to ensure that the coarse grids which are produced are valid and preserve the important geometric features of the fine domain.

When applying multilevel iterative methods on unstructured meshes, the grid hierarchy can allow general coarse grids which are non-quasiuniform and whose boundaries may be non-matching to the boundary of the fine grid. Care must be applied when constructing intergrid transfer operators for various types of boundary conditions. In this section, we will discuss some possibilities.

3.1 Maximal independent set (MIS) coarsening

A maximal independent set of vertices in a graph is a subset of vertices which is *independent* in the sense that no two vertices in the subset are connected by an edge, and *maximal* if the addition of a vertex results in a dependent subset. An automatic approach to generating node-nested coarse grids is to take a maximal independent set (MIS) of the vertices and call this set, the set of coarse grid nodes, and then retriangulate it [28, 16]. A sequence of coarse grids can thus be created by repeated application of this technique.

A simple technique for finding a MIS of vertices is to first choose a MIS of the boundary vertices by choosing every other boundary vertex and eliminating all its nearest neighbors, and then find a MIS of the interior vertices by selecting a random interior vertex and eliminating all its nearest neighbors, and repeating the process until all vertices are either eliminated or selected. The resulting vertex subset is then retriangulated using for example, the same triangulation routine which generated the original fine grid.

For our implementation a simple approximation to this set based on greedy algorithm can be used. First we find a maximal independent set of boundary points, say $V_\Gamma \subset V$. Then the next front (i.e. the next set in which coarse points will be found) is formed as follows: Let $V_f = \emptyset$, then for any point $v \notin V_\Gamma \cup V_f$, if $Adj(v) \cap V_\Gamma = \emptyset$ we set $V_f = V_f \cup \{v\}$. Then the MIS in V_f is found and is added to V_Γ. This procedure is repeated until all fine grid nodes are explored.

There are algorithms which produce better approximation for the NP-complete problem mentioned above (see Chan and Smith [16], Ciarlet and Lamour [33], [34], Ciarlet, Lamour and Smith [35]). Any of these algorithms can be applied in our case.

3.2 Standard nodal value interpolations

In general, the resulting coarse space V_H will not be a subspace of the fine space V_h since the coarse elements are not typically the unions of some fine elements in unstructured grids. To construct a coarse-to-fine transfer operator, one can use the standard nodal value interpolant associated with the fine space, V_h.

Algorithm 3.2.1 *(Standard nodal value interpolation)*

1. ***For*** *each fine grid node,*

2. *Search through all coarse grid elements until one which contains it is found.*

3. ***If*** *the fine grid node is a coarse grid node, then*

4. *Set the interpolant to be equal to that nodal value,*

5. ***Else***

6. *Set it to be a linear interpolation of the 3 nodal values making up that coarse grid element (see Fig. 3.1).*

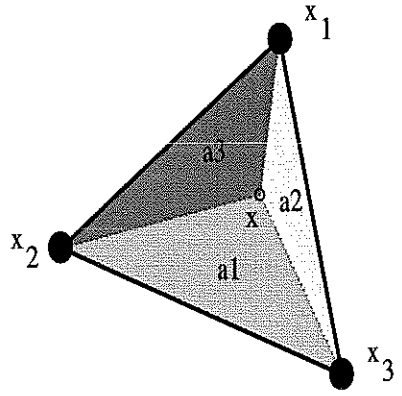


Figure 3.1: Barycentric (natural) coordinates: $\lambda_i(x) = \frac{area(a_i)}{area(T)}$, for $i = 1, 2, 3$, where T is the simplex with vertices x_1, x_2, x_3.

This naive implementation of this routine requires $O(n^2)$ time, but by exploiting the node-nested property of the grids, one can implement this in $O(n)$ time, since only nearest coarse grid elements of a fine node need to be searched.

It has been shown [17] that multigrid methods using these standard linear interpolants satisfies the stability and approximation properties needed for optimal convergence. Thus, the unstructured methods will retain similar convergence rates to their structured counterparts.

The transpose of the standard nodal interpolant results in a weighted restriction operator.

3.3 Interpolations on non-matching boundaries

Notice, however, that the standard nodal value interpolant is only well defined for those fine nodes lying within the coarse domain $\bar{\Omega}_H$, but is undefined for those fine nodes lying outside $\bar{\Omega}_H$. That is, in Step 2 of the standard nodal value interpolation (Algorithm 3.2.1), there is no provision for what to do if all the coarse grid elements have been searched, and none contains the fine grid node. In this section, we give two possible procedures for handling this breakdown.

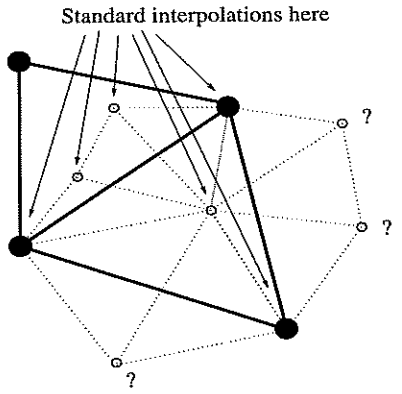


Figure 3.2: Use standard interpolation for fine grid nodes interior to coarse grid. What about fine grid nodes which are not interior to any coarse grid element?

3.3.1 Zero extension interpolations

A simple and natural way to remove this deficiency is to assign those fine node values to zero. This zero extension interpolant works well for Dirichlet boundary conditions [16, 17] but will be neither accurate nor stable for other types of bound-

ary conditions. We shall denote this interpolant as the coarse-to-fine interpolant, $\mathcal{I}_h^0$.

$\mathcal{I}_h^0$: **Zero extension with unmodified coarse boundaries.** Where coarse grid boundary conditions are of Dirichlet type, the standard nodal value interpolants with zero extensions can be accurate enough for interpolating fine grid values outside the coarse grid domain Ω^H (cf. Fig. 3.5a), we refer to [16, 17] for the theoretical and numerical justifications of $\mathcal{I}_h^0$.

$$\mathcal{I}_h^0 v^H(x_j^h) = \begin{cases} v^H(x_j^h) & \text{for } x_j^h \in \Omega \cap \bar{\Omega}^H, \\ 0 & \text{for } x_j^h \in \Omega \setminus \bar{\Omega}^H. \end{cases}$$

Although the interpolant $\mathcal{I}_h^0$ is appropriate to use at Dirichlet boundaries it is not accurate enough, or not accurate at all sometimes, to use at Neumann boundaries, see the numerical results in [17] and Section 3.5.

We provide a simple one-dimensional example to illustrate why better interpolants are needed at non-matching boundaries. This example has a Dirichlet boundary condition at the left boundary point and a homogeneous Neumann boundary condition at the right boundary point. The fine grid function, u, and the coarse grid approximation to it, U_H are shown. For Neumann boundary conditions, the elements from V_h which have to be interpolated are generally not zero at the Neumann part of the boundary. Recall from Section 2.1 that V_h is a subspace of $H_0^1(\Omega, \Gamma_D)$, whose elements are restricted to vanish only on Dirichlet boundary. Using a zero extension correction is the right thing to do at a Dirichlet boundary,

but using a zero extension interpolant at a Neumann boundary will not be accurate enough and will not introduce any correction there. Thus, the only mechanism for removing errors at Neumann boundaries is by the slowly convergent smoother used at the fine level.

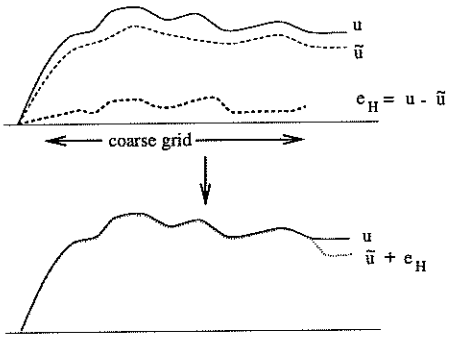


Figure 3.3: Non-matching boundaries: Zero extension interpolation with $\mathcal{I}_h^0$ is not accurate enough.

To achieve better efficiency, we need to modify this intergrid operator to account for the Neumann condition. One way to achieve this is by extending the coarse grid domain to cover any fine grid boundaries of Neumann type and used standard nodal value interpolation. This approach, first proposed and justified in [17], is motivated by the fact that standard nodal value interpolants will still provide a mechanism for coarse grid corrections as long as the coarse grid covers the Neumann boundary part of the fine grid (see Fig. 3.4).

Let us still denote the modified coarse grid domain by $\bar{\Omega}_H$. Then for all $v^H \in V^H$, the interpolant $\mathcal{I}_h^1$ is defined as:

$\mathcal{I}_h^1$: **Zero extension with modified coarse boundaries.** Modify the orig-

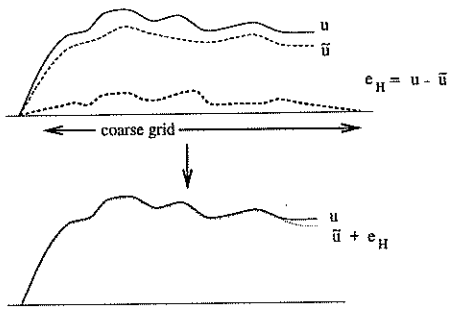


Figure 3.4: Zero extension can be done as long as the coarse grid covers the fine grid ($\mathcal{I}_h^1$) at the Neumann boundary.

inal coarse grid domain Ω^H to make it appropriately larger so that it covers the Neumann boundary part of the fine grid domain (see Fig. 3.5b). Let us still denote the modified coarse grid domain by Ω^H. Then for all $v^H \in V^H$, the interpolant $\mathcal{I}_h^1$ is defined as

$$\mathcal{I}_h^1 v^H(x_j^h) \;=\; \begin{cases} v^H(x_j^h) & \text{for } x_j^h \in \Omega \cap \bar{\Omega}^H, \\ 0 & \text{for } x_j^h \in \Omega \setminus \bar{\Omega}^H. \end{cases}$$

This is a natural extension of v^H by zero outside the Dirichlet boundary part of the coarse grid domain. Similar zero extensions were used in Kornhuber-Yserentant [39] to embed an arbitrarily complicated domain into a square or cube in constructing multilevel methods on nested and quasi-uniform meshes for second order elliptic problems with purely Dirichlet boundary conditions.

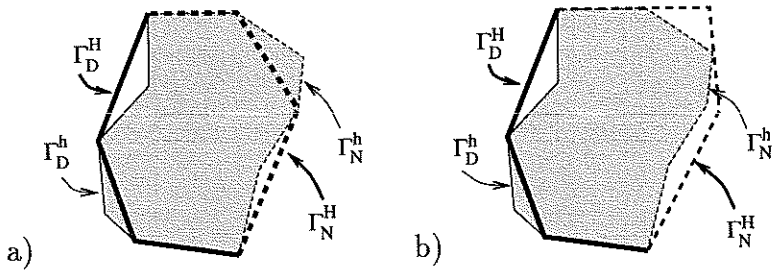


Figure 3.5: **Zero extension interpolants:** a) Unmodified coarse boundaries; b) Coarse boundaries modified to cover the parts where Neumann conditions exist (dashed lines). Thick lines represent coarse grid boundaries.

3.3.1.1 Generating extended coarse grid domains

A coarse grid hierarchy is created by successive coarsenings of a fine grid using a maximal independent set approach. This coarsening method has the nice feature that the coarse grid nodes are a nested subset of the fine grid nodes which can be exploited to save work, as opposed to a completely independent grid hierarchy. The method uses adjacency information to first find the maximal independent set of boundary nodes by eliminating every other boundary node and then find the maximal independent set of the remaining interior nodes. The resulting vertex set is retriangulated using any triangulation algorithm. This method for generating coarse grids often leads to coarse grids whose boundaries do not match those of the fine grid. As we have seen, this is an undesirable outcome which should be avoided. In this section we introduce a boundary adjustment algorithm which extends the boundaries of the resulting coarse grid to ensure that the fine grid domain is always

covered by that of the coarse grid. Note that this choice will not be unique and there are many other ways to extend the coarse domain.

I chose to move vertices around instead of retaining extra vertices to avoid the possibility that the grid might not coarsen much in areas where the domain is convex and also to try to better preserve the maximal independent set. The tradeoff is, the coarsest grid may be quite a great deal larger than the fine grid after several coarsenings. Also, for our purposes, the boundary coarsening algorithm need only be applied to the edges where a mixed Neumann boundary condition occurs. For practical purposes however, we applied the boundary adjustment algorithm to all edges regardless of the boundary condition. This will allow the grids to be reusable, in the event that different boundary conditions for other problems are imposed.

The first step in the boundary coarsening algorithm is to eliminate every other boundary node, keeping track of the eliminated nodes. Next, check to see if an omitted fine node will be exterior to the new coarsened boundary edge, and if so, move a coarse boundary node in such a way that the omitted node will be contained in the coarse domain, otherwise, do nothing. The procedure for moving nodes is as follows:

1. For each coarse boundary node not yet done, label four consecutive fine boundary nodes in order as L, C, R, and N (see Figure 3.6), where node L is a coarse boundary node, node C is the fine node to be eliminated, and node R is generally another coarse boundary node.

2. If node C is interior to the edge $\overline{LR}$, then mark node R as done, and repeat from step (1) with the next coarse node.

3. If node C is exterior to the edge $\overline{LR}$, then either: (a) node R is exterior to edge $\overline{CN}$, or (b) node R is interior to edge $\overline{CN}$.

4. If node R is exterior to edge $\overline{CN}$, then move node R to the intersection point of $\overline{LC}$ and $\overline{RN}$, otherwise move R such that it is now at the point C.

5. Mark the coarse node R as done and repeat from step (1) with the next coarse node until all coarse boundary nodes are done.

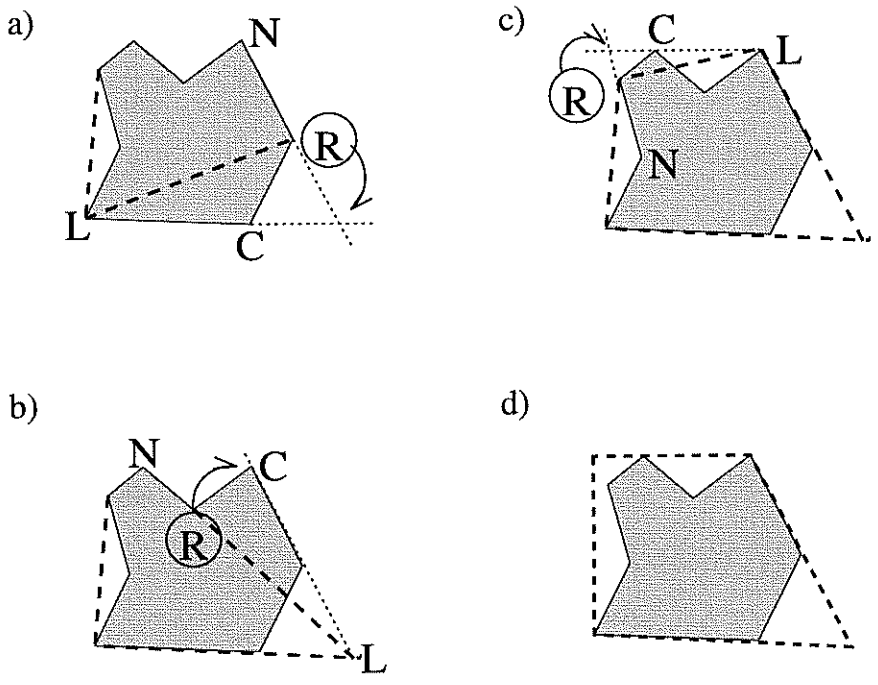


Figure 3.6: Modifying the coarsened the boundaries

We make these choices so that any adjustment of a node will not make things

more problematic as we continue along the boundary. If node C is exterior to edge $\overline{LR}$, then a sufficient condition for it to be contained in the coarse grid domain, is that it lies somewhere on the coarse boundary edge $\overline{LR}$. So it suffices to move node R such that it is somewhere on the line defined by edge $\overline{LC}$. The simplest choice is to move node R to the point at node C. Before moving node R, however, we must remember to treat it as an omitted node also, so if R is interior to edge $\overline{CN}$, the above choice is sufficient. If node R is exterior to edge $\overline{CN}$, we should impose the additional condition that node R be moved such that it also lies on the line defined by edge $\overline{RN}$. The instances where the intersection point of $\overline{LC}$ and $\overline{RN}$ is very far away (for instance, if they are parallel or nearly parallel), or when two adjacent nodes are being eliminated, can be handled specially.

The usual weighted interpolation and restriction matrices with zero extension of exterior nodes can be used, since modifying the boundaries in this way ensures that all fine grid points are interior to the coarse domains and so all fine grid points will be correctly interpolated.

3.3.2 Modified coarse-to-fine interpolations

Although the coarse-to-fine operator $\mathcal{I}_h^1$ works well for mixed boundary conditions and can be used with the standard nodal value interpolants, it requires making modifications to the original coarse grid generated by the automatic MIS

approach to ensure that the coarse domain completely covers the Neumann boundary part of the fine domain. This can be difficult to do for very complicated domains and can often lead to coarse domains which may deviate significantly from the fine domain, leading to poor approximation.

To avoid modifying the original coarse grid, we now consider standard finite element interpolants which provide a definition for interpolating the fine Neumann boundaries which lie exterior to the coarse domain. The idea is as follows: Let us consider a fine grid point, x, which lies outside the coarse grid domain. Find a nearby coarse grid triangle to x (say, τ_H with vertices x_1, x_2, x_3), and *extrapolate* $u(x)$ using the values $u(x_1), u(x_2)$ and $u(x_3)$. Note that such an extrapolation should depend on the type of boundary condition at x.

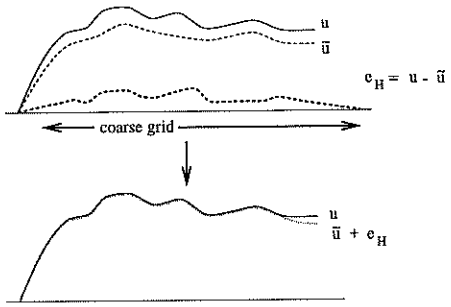


Figure 3.7: More accurate extension with $\mathcal{I}_h^2$ done at the Neumann boundary.

To do so, we first introduce some notation. Let τ_{lr}^H be any coarse boundary element in $\mathcal{T}^H$ made up of the three vertices x_l^H, x_r^H, x_i^H and which has an edge on the boundary $\partial\Omega^H$, denoted by $x_l^H x_r^H$. We use $\Omega(x_l^H, x_r^H)$ to denote the union of all fine elements, if any, which has a non-empty intersection with the unbounded

domain formed by the edge $x_l^H x_r^H$ and two outward normal lines to $x_l^H x_r^H$ at two vertices x_l^H, x_r^H (cf. Fig. 3.8). By including a few more fine elements in some $\Omega(x_l^H, x_r^H)$, if necessary, we may assume that the fine grid part $(\Omega \setminus \Omega^H)$ is included in the union of all $\Omega(x_l^H, x_r^H)$. Moreover, we assume

(H1) *diam* $\Omega(x_l^H, x_r^H) \leq \mu_0$ *diam* τ_{lr}^H,

which implies the measure of $\Omega(x_l^H, x_r^H)$ is bounded by the measure of τ_{lr}^H:

$$|\Omega(x_l^H, x_r^H)| \leq \mu\, |\tau_{lr}^H|,$$

where μ_0 and μ are two positive constants independent of H and h. Without any difficulty, the constant μ_0, and so μ, can be allowed in our subsequent results to depend on the two nodes x_l^H, x_r^H. In this case, μ_0 and μ will enter all the related bounds naturally.

We remark that (**H1**) restricts the size of the fine grid part near the edge $x_l^H x_r^H$ but outside the coarse grid domain Ω^H, that is, each local fine grid part $\Omega(x_l^H, x_r^H)$ is not allowed to be too large compared to its nearest coarse element τ_{lr}^H. This is a reasonable requirement in applications.

Then the standard nodal value interpolant associated with the fine space V^h can be generalized outward to each local fine grid part $\Omega(x_l^H, x_r^H)$ using three given linear functions θ_1, θ_2 and θ_3 which are defined in $\bar{\Omega} \cup \bar{\Omega}^H$ but bounded in $\Omega(x_l^H, x_r^H) \cup \tau_{lr}^H$ and satisfy

$$\theta_1(x) + \theta_2(x) + \theta_3(x) = 1, \quad \forall\, x \in \bar{\Omega} \cup \bar{\Omega}^H. \tag{3.1}$$

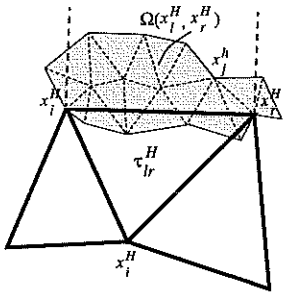


Figure 3.8: Shaded region, $\Omega(x_l^H, x_r^H)$, shows the fine grid part which is not completely covered by the coarse grid domain.

Note that the functions θ_1, θ_2 and θ_3 above are not necessarily non-negative, and though they are element τ_{lr}^H-related, we will not use any index to specify this relation in order to simplify the notation. Then for any coarse function $v^H \in V^H$, we define an operator Θ_h by

$$\Theta_h v^H(x) = \theta_1(x)v^H(x_l^H)+\theta_2(x)v^H(x_r^H)+\theta_3(x)v^H(x_i^H), \quad \forall\, x \in \Omega(x_l^H, x_r^H)\cup\tau_{lr}^H,$$

and assume that

(H2) $\Theta_h v^H = v^H$ on the edge $x_l^H x_r^H$,

which means $\Theta_h v^H$ is indeed an extension of v^H. For convenience, later on we will always regard $\Theta_h v^H$ as a function defined also outside $\Omega(x_l^H, x_r^H)\cup\tau_{lr}^H$ by extending it naturally.

With the above notation, we can introduce the general coarse-to-fine interpolant $\mathcal{I}_h$:

Definition 3.3.1 *For any coarse function v^H in V^H, its image under the coarse-to-fine interpolant $\mathcal{I}_h$ is specified as follows:*

(C1) *For any fine node x_j^h in $\bar{\Omega} \cap \bar{\Omega}^H$,*

$$\mathcal{I}_h v^H(x_j^h) = v^H(x_j^h);$$

(C2) *For any fine node x_j^h in $\Omega(x_l^H, x_r^H) \setminus \bar{\Omega}^H$ with both x_l^H and x_r^H of Neumann nodes,*

$$\mathcal{I}_h v^H(x_j^h) = \Theta_h v^H(x_j^h);$$

(C3) *For any fine node x_j^h in $\Omega(x_l^H, x_r^H) \setminus \bar{\Omega}^H$ with both x_l^H and x_r^H of Dirichlet nodes,*

$$\mathcal{I}_h v^H(x_j^h) = 0;$$

(C4) *For any fine node x_j^h in $\Omega(x_l^H, x_r^H) \setminus \bar{\Omega}^H$ with one of x_l^H and x_r^H the Neumann node and one the Dirichlet node,*

$$\begin{aligned} \mathcal{I}_h v^H(x_j^h) &= 0, \quad \textit{if } x_j^h \textit{ is a fine boundary node of Dirichlet type;} \\ \mathcal{I}_h v^H(x_j^h) &= \Theta_h v^H(x_j^h), \quad \textit{otherwise.} \end{aligned}$$

The following are two concrete examples of interpolants which satisfy the above definition and assumptions. We only give the corresponding forms of Θ_h's required in the definition:

$\mathcal{I}_h^2$: **Nearest edge interpolation.** Define the interpolant at x_j^h by using the nodes of the coarse boundary edge closest to x_j^h (see Fig. 3.9):

$$\mathcal{I}_h^2 v^H(x_j^h) = \lambda(x_j^h) v^H(x_l^H) + (1 - \lambda(x_j^h)) v^H(x_r^H),$$

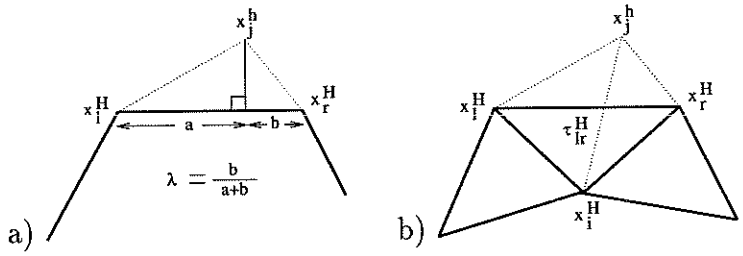


Figure 3.9: **More accurate interpolants:** Fine nodal values outside the coarse domain are interpolated with coarse nodal values on the nearest: a) coarse grid edge; b) coarse element τ_{lr}^H. Thick lines represent coarse grid boundaries or elements, dotted lines show the coarse nodes used in the interpolation.

where x_l^H and x_r^H are the nodes of the coarse boundary edge closest to x_j^h, and λ is the ratio of the lengths of two segments of the edge $x_l^H x_r^H$ cut off by the normal line passing through x_j^h to the edge (see Fig. 3.9). This kind of interpolation was used by Bank-Xu [1] in their construction of a hierarchical basis on an unstructured mesh.

$\mathcal{I}_h^3$: **Nearest element interpolation.** Define the non-zero extension by using barycentric functions (see Fig. 3.9):

$$\mathcal{I}_h^3 v^H(x_j^h) \;=\; \lambda_l(x_j^h)v^H(x_l^H) + \lambda_r(x_j^h)v^H(x_r^H) + \lambda_i(x_j^h)v^H(x_i^H),$$

where $\lambda_l, \lambda_r, \lambda_i$ are three barycentric coordinate functions (also known as area or volume coordinates) corresponding to τ_{lr}^H.

Remark 3.1. Note that the functions λ_l, λ_r and λ_i used in the definition of $\mathcal{I}_h^3$ satisfies $\lambda_l, \lambda_r, \lambda_i \geq 0$ for $x_j^h \in \tau_{lr}^H$, but not so for $x_j^h \notin \tau_{lr}^H$. In the case

as shown in Fig. 3.9b, we have $x_j^h \notin \tau_{lr}^H$, $\lambda_l(x) \geq 0, \lambda_r(x) \geq 0$, but $\lambda_i(x) \leq 0$ and $\lambda_l(x) + \lambda_r(x) + \lambda_i(x) = 1$. The barycentric coordinates may still be defined, provided we consider the area of a simplex to be orientation-dependent, that is, area is > 0 for "right-handed" triangles and are is < 0 for "left-handed" triangles. By **(H1)**, we always have

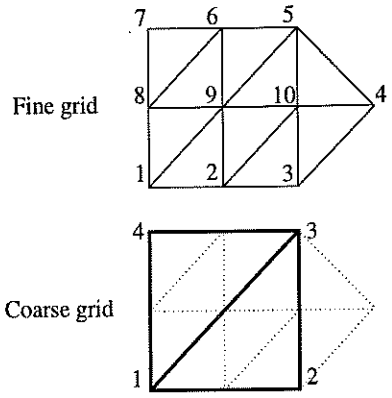
$$|\lambda_l(x)| \leq \mu_1, \quad |\lambda_r(x)| \leq \mu_1, \quad \text{and} \quad |\lambda_i(x)| \leq \mu_1, \quad \forall\, x \in \Omega(x_l^H, x_r^H) \cup \tau_{lr}^H,$$

where μ_1 is a constant independent of h and H but depending only on the constant μ in **(H1)**.

3.3.3 An illustrative example

To illustrate the differences among the four different interpolants $(\mathcal{I}_h^0, \mathcal{I}_h^1, \mathcal{I}_h^2, \mathcal{I}_h^3)$, let us consider a simple fine grid with 10 nodes, and a 4-noded MIS coarse grid in which the coarse grid boundary does not match the fine grid boundary. In particular, the MIS coarse grid that is automatically generated will not contain the fine grid node, v_4 (see, e.g. Fig. 3.10). The coarse grids and the corresponding interpolants are shown in Figs. 3.10–3.13 below:

Note that $\mathcal{I}_h^0, \mathcal{I}_h^2$, and $\mathcal{I}_h^3$ are identical except for the way that the value at fine grid node, v_4, is interpolated (row 4). Interpolant $\mathcal{I}_h^0$ results in a zero value extension, interpolant $\mathcal{I}_h^2$ extends linearly in the normal direction with the nearest coarse boundary edge, and interpolant $\mathcal{I}_h^3$ extends by barycentric coordinates of the

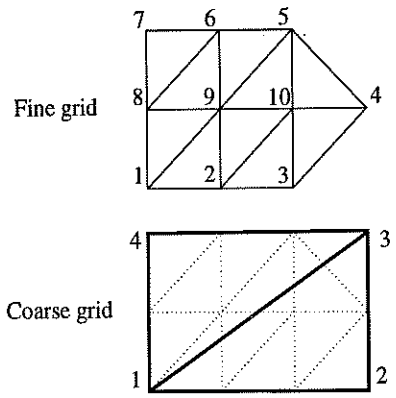


$$\mathcal{I}_h^0 = \begin{bmatrix} 1.0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1.0 \\ 0.5 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0 \end{bmatrix} \Leftarrow$$

Figure 3.10: Fine grid and MIS coarse grid and the corresponding linear interpolant, $\mathcal{I}_h^0$.

nearest coarse element (note here that the entries are not necessarily non-negative, but still sum to one). Because all the coarse boundary nodes are also fine boundary nodes, the definition of the coarse boundary conditions can simply be taken to be the same as that of the corresponding fine boundary nodes.

Interpolant $\mathcal{I}_h^1$ differs in that the coarse grid domain was modified to contain the fine grid domain. As a result, the grids are not element nested and some of the coarse boundary nodes are not nodes from the fine grid. Standard linear interpolation is done everywhere in this case with no special extensions required.



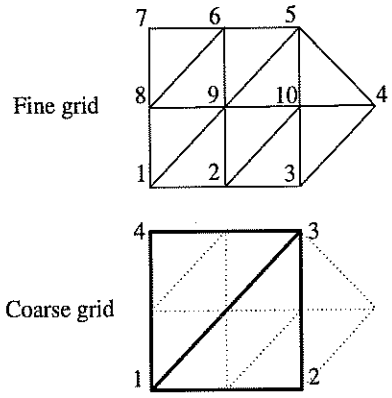
$$\mathcal{I}_h^1 = \begin{bmatrix} 1.0000 & 0 & 0 & 0 \\ 0.6667 & 0.3333 & 0 & 0 \\ 0.3333 & 0.6667 & 0 & 0 \\ 0 & 0.5000 & 0.5000 & 0 \\ 0 & 0 & 0.6667 & 0.3333 \\ 0 & 0 & 0.3333 & 0.6667 \\ 0 & 0 & 0 & 1.0000 \\ 0.5000 & 0 & 0 & 0.5000 \\ 0.5000 & 0 & 0.3333 & 0.1667 \\ 0.3333 & 0.1667 & 0.5000 & 0 \end{bmatrix} \Leftarrow$$

Figure 3.11: Fine grid and modified coarse grid and the corresponding linear interpolant, $\mathcal{I}_h^1$.

However, choosing the proper boundary conditions for the coarse boundary points which are not in the fine grid must be determined somehow (see Def. 3.3.1).

3.4 Stability and approximation of the non-nested interpolation

The convergence theory for overlapping multilevel domain decomposition and multigrid methods require the coarse-to-fine grid transfer operator to possess the local optimal L^2-approximation and local H^1-stability properties as introduced in



$$\mathcal{I}_h^2 = \begin{bmatrix} 1.0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1.0 \\ 0.5 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0 \end{bmatrix} \Leftarrow$$

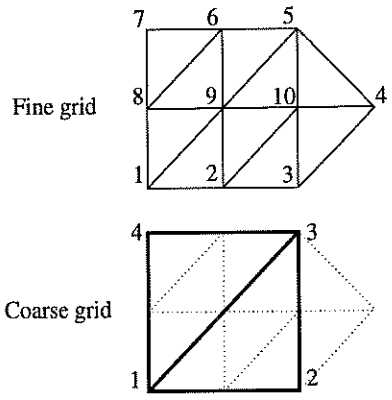
Figure 3.12: Fine grid and MIS coarse grid and the corresponding linear interpolant, $\mathcal{I}_h^2$.

Sec. 2.4. The locality of these properties is essential to the effectiveness of these methods on highly non-quasi-uniform unstructured meshes.

Because the spaces are non-nested (they are *node-nested*, but still *non-nested*, as coarse grid elements are not unions of fine grid elements), in the theory discussed in Section 2.4, the u_H coarse space approximation to u should be defined as:

$$u_H = \mathcal{I}_h \mathcal{R}_H u.$$

Since $\mathcal{R}_H \in V_H \not\subset V_h$, we need to use the interpolation operator $\mathcal{I}_h$ to map $\mathcal{R}_H u$



$$\mathcal{I}_h^3 = \begin{bmatrix} 1.0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ -0.5 & 1.0 & 0.5 & 0 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1.0 \\ 0.5 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0 \end{bmatrix} \Leftarrow$$

Figure 3.13: Fine grid and MIS coarse grid and the corresponding linear interpolant, $\mathcal{I}_h^3$.

back to V_h. The convergence theory now requires both $\mathcal{I}_h$ and $\mathcal{R}_H$ to possess the stability and approximation properties. When the mesh is quasi-uniform, the usual L^2-projection, Q_H, can be used for $\mathcal{R}_H$. But when the mesh is highly non-quasi-uniform, the constant in the approximation property (2.23) can deteriorate if we use Q_H. The trick then is to use a localized version of the L^2-projection, i.e. the so-called Clément's projection. It is known that this projection provides local stable and good approximations. We refer to Clément [22] for its definition and Chan-Zou [18] and Chan-Smith-Zou [17] for its use in domain decomposition

contexts.

For $\mathcal{I}_h$, we can take the coarse-to-fine interpolants introduced in Sec. 3.3 The key step then is proving the stability and approximation properties for $\mathcal{I}_h$. The proof that the non-nested standard interpolation used in the interior is stable and accurate can be found in Cai [8] and Chan-Smith-Zou [17]. The proof for the boundary-specific interpolations can be found in [13].

3.4.1 Two-level convergence theory

We need to introduce some more notation (see § 2.1): for $\tau^h \in \mathcal{T}^h$ and $\tau^H \in \mathcal{T}^H$,

$$\begin{aligned} N(\tau^H) &= \text{union of coarse elements adjacent to } \tau^H, \\ B_k &= \cup_{\tau^H \cap \Omega^k \neq \emptyset} \tau^H, \quad h_k = \max_{\tau^h \subset \Omega^k} h_\tau, \\ S_k &= \cup_{\tau^H \subset B_k} N(\tau^H), \quad H_k = \max_{\tau^H \subset B_k} H_\tau. \end{aligned}$$

Note that B_k is the union of all coarse elements having nonempty intersection with the subdomain Ω^k. We allow each Ω^k to be of quite different size and of quite different shape from other subdomains, but make the following reasonable assumptions:

(A1) Any point $x \in \Omega$ belongs to at most q_0 subdomains of $\{\Omega^k\}_{k=1}^p$ with $q_0 > 0$ an integer.

(A2) $h_k \lesssim H_k$; and $\text{card}\{\tau^H \in \mathcal{T}^H;\ \tau^H \subset B_k\} \leq n_0$ for $1 \leq k \leq p$ with $n_0 > 0$ an integer.

(A3) Any point $x \in \Omega^H$ belongs to at most q_0 subdomains of $\{S_k\}_{k=1}^p$.

The following theorem gives the bound of the condition number $\kappa(MA)$, for the two-level additive Schwarz method (2.3.2.2).

Theorem 3.4.1 *Under the assumptions **(A1) - (A3)**, we have*

$$\kappa(MA) \lesssim \max_{1 \le k \le p} \frac{H_k^2}{\delta_k^2}.$$

Theorem 3.4.1 indicates that an optimal condition number may be expected if the local overlap δ_k is proportional to the local subdomain size H_k.

To prove Theorem 3.4.1, it is essential for any $u \in V^h$ to find a partition $u = \mathcal{I}_h u_H + \sum_{k=1}^p u_k$ with $u_k \in V^k$ $(1 \le k \le p)$ and $u_H \in V^H$ such that $\mathcal{I}_h u_H$ is bounded by u in both L^2-norm and H^1-norm, and preserves the local optimal L^2-norm error approximation to u. This can be done by using the following lemma and the standard partition $\{\theta_i\}_{i=1}^p$ of unity for Ω corresponding to the subdomains $\{\Omega^k\}_{i=1}^p$.

Lemma 3.4.1 *Given any interpolation operator $\mathcal{I}_h$ satisfying Definition 3.3.1, then for any $u^h \in V^h$ there exists $u^H \in V^H$ such that for all $\tau^H \in \mathcal{T}^H$, we have*

$$(l1) \qquad \sum_{\substack{\tau^h \cap \tau^H \neq \emptyset \\ \tau^h \subset \bar{\Omega}^H}} \|u^h - \mathcal{I}_h u^H\|_{0,\tau^h}^2 \le C d^2(\tau^H) |Eu^h|_{1,N(\tau^H)}^2,$$

$$(l2) \qquad \sum_{\substack{\tau^h \cap \tau^H \neq \emptyset \\ \tau^h \subset \bar{\Omega}^H}} |\mathcal{I}_h u^H|_{1,\tau^h} \leq C |Eu^h|_{1,N(\tau^H)},$$

$$(l3) \qquad \sum_{\tau^h \in \Omega(x_l^H, x_r^H)} \|u^h - \mathcal{I}_h u^H\|^2_{0,\tau^h} \leq C d^2(\tau_{lr}^H) \sum_{\tau^H \in N(\tau_{lr}^H)} |Eu^h|^2_{1,N(\tau^H)},$$

$$(l4) \qquad \sum_{\tau^h \in \Omega(x_l^H, x_r^H)} |\mathcal{I}_h u^H|^2_{1,\tau^h} \leq C \sum_{\tau^H \in N(\tau_{lr}^H)} |Eu^h|^2_{1,N(\tau^H)}.$$

REMARK. The inequalities $(l1)$ and $(l2)$ correspond to the parts where the fine grid domain is completely contained in the coarse grid domain. Their proofs can be found in [17, 18]. The last two inequalities $(l3)$ and $(l4)$ correspond to the fine grid parts which are not covered by the coarse grid. Proofs for inequalities $(l3)$ and $(l4)$ can be found in [13]

3.5 Numerical results

In this section, we provide some numerical results of domain decomposition and multigrid methods on unstructured meshes for elliptic problems on various fine grid domains (see Fig. 3.14). The well-known NASA airfoil mesh was provided by T. Barth and D. Jesperson of NASA Ames, and a fine, unstructured square and annulus were generated using Barth's 2-dimensional Delaunay triangulator. All numerical experiments were performed using the Portable, Extensible Toolkit for Scientific Computation (PETSc) [27], running on a Sun SPARC 20. Piecewise linear finite elements were used for the discretizations and the resulting linear system was solved using either multilevel overlapping Schwarz or V-cycle multigrid

as a preconditioner with full GMRES as an outer accelerator.

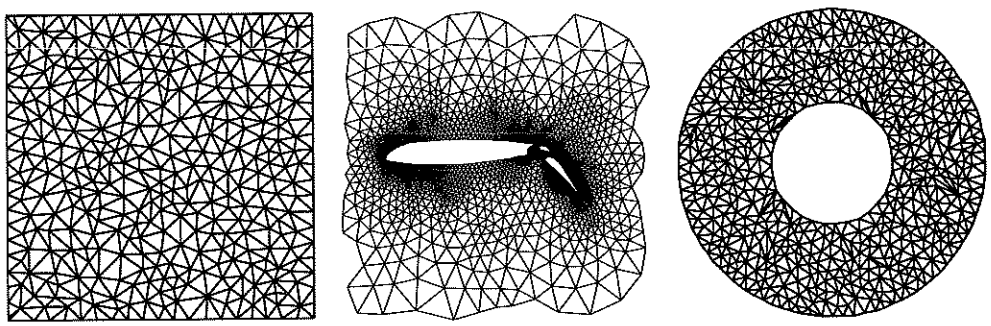


Figure 3.14: Some fine grids: an unstructured square with 385 nodes (left), NASA airfoil with 4253 nodes (center) and an annulus with 610 nodes (right).

Our approach to generating a coarse grid hierarchy is to find a maximal independent set of the boundaries and the interior of the fine grid of the mesh, and then retriangulate the resulting set of vertices (other coarsening algorithms can be used here). This process is then repeated recursively for the desired number of levels. An example coarse grid hierarchy of the airfoil mesh retriangulated with Cavendish's algorithm [10] is shown in Fig. 3.15 where G^2 refers to the first coarsening of the fine grid, G^1 is the coarsening of G^2, and G^0 is the coarsening of the G^1.

We shall present numerical results for Schwarz solvers and multigrid methods. For partitioning, all the domains (except the coarsest) were partitioned using the recursive spectral bisection method [47], with exact solves for both the subdomain problems and the coarse grid problem. To generate overlapping subdomains, we

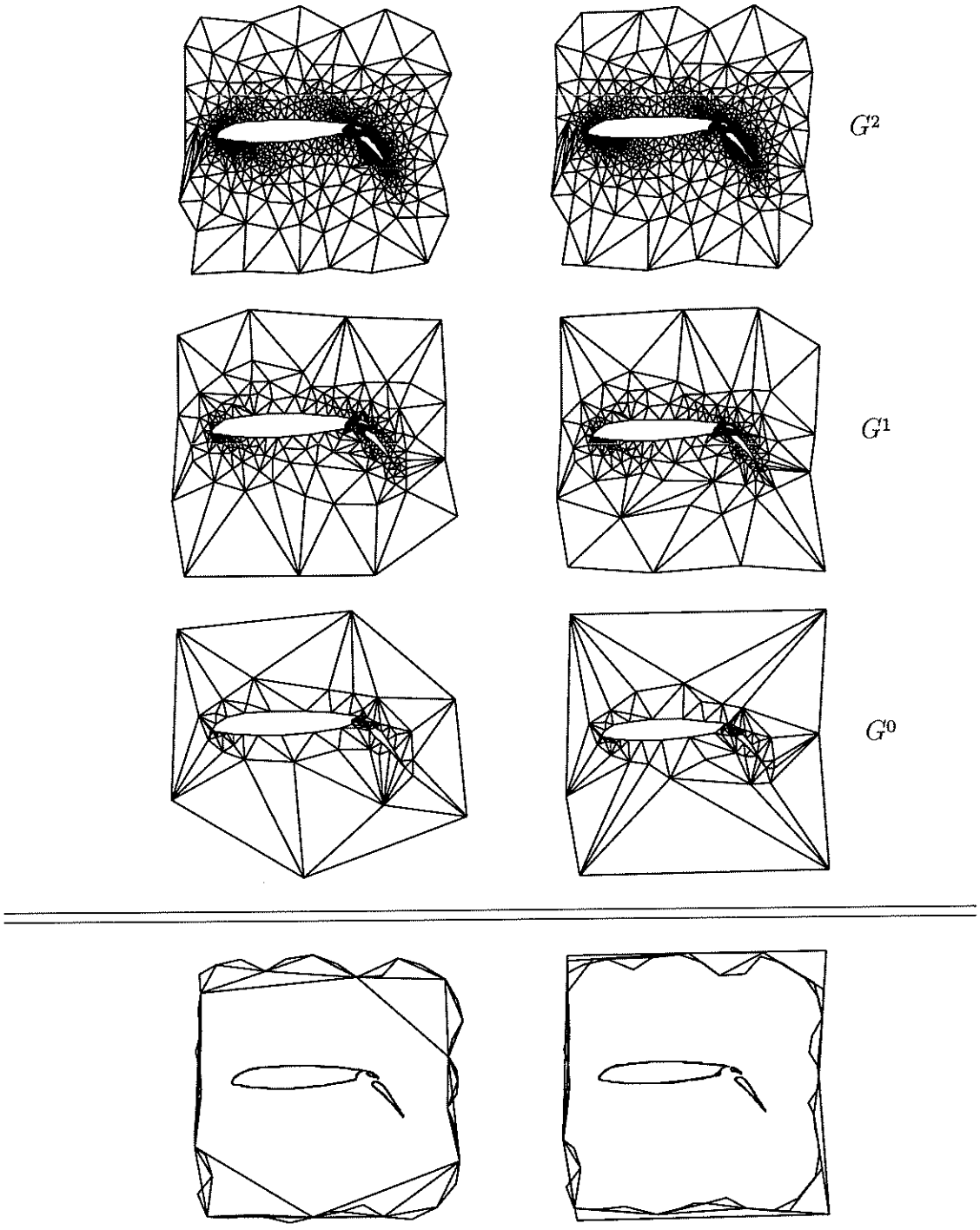


Figure 3.15: *Airfoil* grid hierarchy with unmodified boundaries (left) and modified boundaries (right).

first partition the domain into non-overlapping subdomains and then extend each subdomain by some number of elements.

In all the experiments, the initial iterate is set to be zero and the iteration is stopped when the discrete norm of the residual is reduced by a factor of 10^{-5}.

For our first experiment, we use additive Schwarz to solve the Poisson problem on a unit square with homogeneous Dirichlet boundary conditions. Because the fine domain is so simple and Dirichlet boundary conditions are given, non-matching boundaries are not an issue here and no special interpolants are used. We provide these results simply for completeness, as multilevel Schwarz results on unstructured grids have not been previously found in the literature to the authors' knowledge. Table 3.1 shows the number of GMRES iterations to convergence with varying fine grid problem and varying number of levels.

Providing a coarse grid improved convergence, and without it the method is not scalable to the case with a large number of subproblems. Interesting things to notice are that for a fixed number of levels, multilevel Schwarz is mesh-size independent, but that the number of iterations increases with the number of levels for a fixed problem size. This had also been previously observed for structured meshes using a multilevel diagonal scaling method in [48] and is due to the additive nature of the method. Also, increasing the amount of overlap improved convergence, but in practice, a one-element overlap was sufficient.

In our second experiment, we solve a mildly varying coefficient problem on the

airfoil:

$$\frac{\partial}{\partial x}((1+xy)\frac{\partial u}{\partial x}) + \frac{\partial}{\partial y}((\sin(3y))\frac{\partial u}{\partial y}) = (4xy+2)\sin(3y) + 9x^2\cos(6y)$$

with either a purely Dirichlet boundary condition or a mixed boundary condition: Dirichlet for $x \leq 0.2$ and homogeneous Neumann for $x > 0.2$. For this problem, the non-homogeneous Dirichlet condition is $u = 2 + x^2\sin(3y)$. Table 3.16 shows the number of GMRES iterations to convergence using additive multilevel Schwarz with the different boundary treatments.

The deterioration in the method can be observed when Neumann conditions are not properly handled.

In Table 3.17, we show results for the same problem, but solved using a hybrid multiplicative-additive Schwarz (multiplicative between levels but additive among subdomains on the same level). As in the additive case, deterioration of the method occurs when mixed boundary conditions are present. However, we can achieve optimal convergence rates, even with a varying number of levels with the hybrid method. Still further improvement can obtained when using a multiplicative method (both on the subdomains and between levels) and the method behaves much like multigrid (see Tables 3.18–3.2). In fact, this is nothing more than multigrid but with a block smoother. A V-cycle multigrid method with pointwise Gauss-Seidel smoothing and 2 pre- and 2 post-smoothings per level was used to produce the results in Table 3.2.

Table 3.3 shows some multigrid results for the Poisson equation on an annulus.

The forcing function is set to be one and both kinds of boundary conditions were tested. A V-cycle multigrid method with pointwise Gauss-Seidel smoothing and 2 pre- and 2 post-smoothings per level was used. When mixed boundary conditions are present, the deterioration is less pronounced in the multigrid method, but it still exists. It it interesting to note that in our previous multigrid experiments on a quasi-uniform annulus (see [14]), the observed deterioration in the method was much more dramatic than those observed here with the unstructured annulus. We believe that this was due to some extremely poor element aspect ratios on the fine grid in the quasi-uniform case, compounding the effect of the poor approximation on Neumann boundaries.

For partitioning, all the domains (except the coarsest) were partitioned using the recursive spectral bisection method [47], with exact solves for both the subdomain problems and the coarse grid problem. To generate overlapping subdomains, we first partition the domain into nonoverlapping subdomains and then extend each subdomain by some number of elements.

We solve a mildly varying coefficient problem on the airfoil:

$$\frac{\partial}{\partial x}((1+xy)\frac{\partial u}{\partial x}) + \frac{\partial}{\partial y}((\sin(3y))\frac{\partial u}{\partial y}) = F(x,y),$$

where

$$F(x,y) = (4xy+2)\sin(3y) + 9x^2\cos(6y),$$

with either a purely Dirichlet boundary condition or a mixed boundary condition: Dirichlet for $x \leq 0.2$ and homogeneous Neumann for $x > 0.2$. For this problem,

the non-homogeneous Dirichlet condition is $u = 2 + x^2 \sin(3y)$.

Fig. 3.17 shows results when a hybrid 4-level multiplicative-additive Schwarz method is used (multiplicative between levels but additive among subdomains on the same level). As can be seen, deterioration of the method occurs with interpolant $\mathcal{I}_h^0$ when mixed boundary conditions are present. The next figure (Fig. 3.18) shows results for the multiplicative Schwarz method (both on the subdomains and between levels). This method behaves much like multigrid (see Table 3.2). In fact, this is nothing more than standard V-cycle multigrid with a block smoother used as a preconditioner. A V-cycle multigrid method with pointwise Gauss-Seidel smoothing and 2 pre- and 2 post-smoothings per level was used to produce the results in Table 3.2.

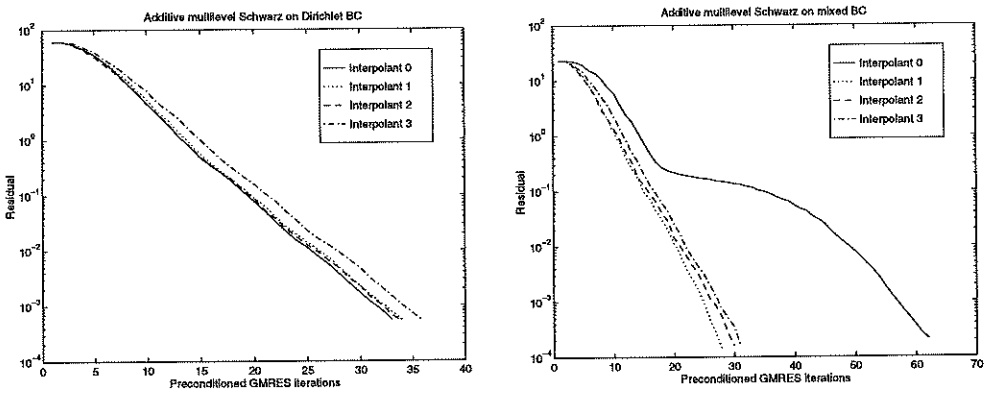


Figure 3.16: **Additive 4-level Schwarz.** Dirichlet boundary conditions (left) or mixed boundary conditions (right) for the elliptic problem with mildly varying coefficients on the *airfoil* grid.

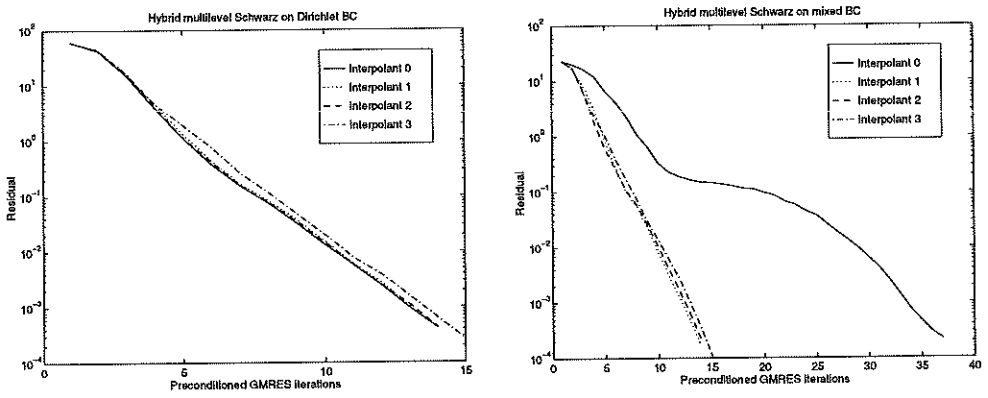


Figure 3.17: **Hybrid multiplicative-additive 4-level Schwarz.** Dirichlet boundary conditions (left) or mixed boundary conditions (right) for the elliptic problem with mildly varying coefficients on the *airfoil* grid.

3.6 Conclusions

When using general unstructured meshes, the coarse grid domain may not necessarily match that of the fine grid. For the parts of the fine grid domain which are not contained in the coarse domain, special treatments must be done to handle different boundary conditions. The transfer operators using linear interpolation with a zero extension is the most natural to implement and is effective for problems with Dirichlet boundary conditions.

For problems where Neumann boundary conditions exist however, zero extension is no longer appropriate and special interpolants should be sought. Our

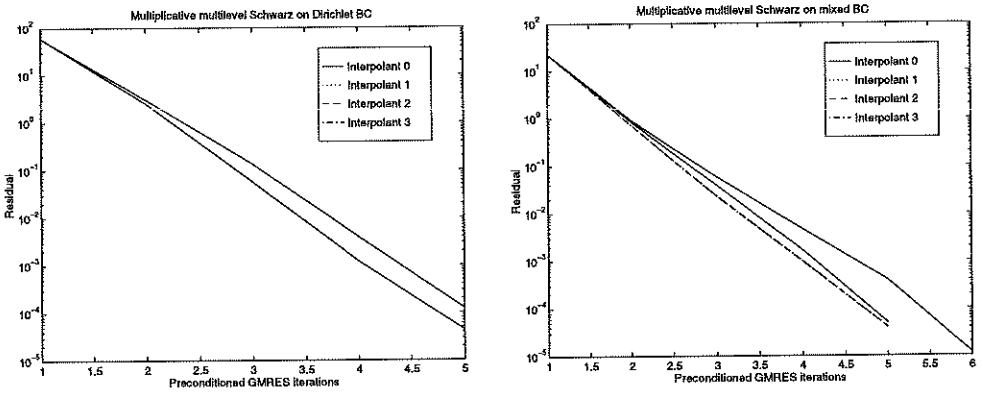


Figure 3.18: **Multiplicative 4-level Schwarz.** Dirichlet boundary conditions (left) or mixed boundary conditions (right) for the elliptic problem with mildly varying coefficients on the *airfoil* grid.

numerical results show the significance of the assumption that when standard interpolations with zero extension are used, the coarse grid must cover the Neumann boundaries of the fine grid problem, otherwise deterioration of the methods occurs. The deterioration is most significant when using additive multilevel methods, but can still be seen for the multiplicative methods. When coupled with highly stretched elements, the deterioration can be very significant, even for multiplicative methods.

Although modifying the coarse grid domains to ensure that this assumption is satisfied is effective, this approach can be problematic to implement for particularly complicated domains or can sometimes generate coarse grid domains which deviate significantly from the fine domain.

An alternative is to modify the interpolants so that non-zero extensions be used on those fine grid boundaries which have Neumann conditions and which are not contained within the coarse grid domain. Since we are using the multilevel methods only as preconditioners, the extension need not be particularly accurate; we used either constant extension with the nearest boundary nodal value or extension using the barycentric functions of the nearest coarse grid element, neither of which are difficult to implement.

Table 3.1: **Additive multilevel Schwarz** iterations for the Poisson problem on a unit square grid. Table shows the number of GMRES iterations to convergence.

Dirichlet boundary conditions

# of levels	# of nodes	# of subdomains	# overlap elements		
			0	1	2
1	6409	256	84	63	50
	1522	64	45	36	27
	385	16	26	19	16
2	1522 385	64 1	19	16	16
	385 102	16 1	19	15	15
	102 29	4 1	17	15	15
3	6409 1522 385	256 64 1	28	24	25
	1522 385 102	64 16 1	32	25	26
	385 102 29	16 4 1	31	26	26
4	6409 1522 385 102	256 64 16 1	43	37	37
	1522 385 102 29	64 16 4 1	42	37	37

Table 3.2: **Multigrid** iterations for the elliptic problem with mildly varying coefficients on the *airfoil*. Tables show the number of GMRES iterations to convergence.

Dirichlet boundary conditions

# of fine grid nodes	MG levels	# of coarse grid nodes	Special Interpolant Used			
			$\mathcal{I}_h^0$	$\mathcal{I}_h^1$	$\mathcal{I}_h^2$	$\mathcal{I}_h^3$
4253	2	1170	4	4	4	4
	3	340	4	4	4	4
	4	101	4	4	4	4

Mixed Dirichlet/Neumann boundary conditions

# of fine grid nodes	MG levels	# of coarse grid nodes	Special Interpolant Used			
			$\mathcal{I}_h^0$	$\mathcal{I}_h^1$	$\mathcal{I}_h^2$	$\mathcal{I}_h^3$
4253	2	1170	6	5	4	4
	3	340	6	4	5	5
	4	101	7	5	5	5

Table 3.3: **Multigrid** iterations for the Poisson problem on an *annulus*. Tables show the number of GMRES iterations to convergence.

Dirichlet boundary conditions

# of fine grid nodes	MG levels	# of coarse grid nodes	Special Interpolant Used			
			$\mathcal{I}_h^0$	$\mathcal{I}_h^1$	$\mathcal{I}_h^2$	$\mathcal{I}_h^3$
2430	2	610	4	4	4	4
	3	160	4	4	4	4
	4	47	4	4	4	4

Mixed Dirichlet/Neumann boundary conditions

# of fine grid nodes	MG levels	# of coarse grid nodes	Special Interpolant Used			
			$\mathcal{I}_h^0$	$\mathcal{I}_h^1$	$\mathcal{I}_h^2$	$\mathcal{I}_h^3$
2430	2	610	6	5	4	4
	3	160	7	5	4	4
	4	47	7	5	4	4

Table 3.4: **Multigrid** iterations for the *annulus* grid with varying fine grid mesh sizes. Results are for Dirichlet / Mixed boundary conditions.

# of fine grid nodes	MG levels	# of coarse grid nodes	Interpolant Used			
			$\mathcal{I}_h^0$	$\mathcal{I}_h^1$	$\mathcal{I}_h^2$	$\mathcal{I}_h^3$
576	2	159	4 / 11	4 / 7	4 / 6	4/ 6
	3	47	4 / 11	4 / 7	4 / 6	4/ 6
	4	15	4 / 11	4 / 7	4 / 6	4/ 6
2176	2	575	5 / 17	5 / 7	5 / 7	5/ 7
	3	159	5 / 17	5 / 7	5 / 7	5/ 7
	4	47	5 / 17	5 / 7	5 / 7	5/ 7
8448	2	2175	5 / 25	5 / 8	5 / 8	5/ 8
	3	574	5 / 25	6 / 8	5 / 8	5/ 8
	4	158	5 / 25	6 / 8	5 / 8	5/ 8

CHAPTER 4

A Multigrid Eigenvalue Solver for Unstructured Grids

In this chapter, the techniques from the previous chapter will be combined with a well-known multigrid eigenvalue solver for structured grids to produce an effective algorithm for solving eigenvalue problems on unstructured grids. The efficiency of the unstructured eigenvalue solver will be shown to be similar to that of the structured solver.

The problem of calculating the eigenvalues of A can be viewed as solving a non-linear problem [44, 50]. Because of this, the correction scheme (CS) multigrid introduced in Section 2.3.1 cannot be used to solve such problems. A variant of multigrid, known as full approximation scheme (FAS) multigrid introduced by Brandt [4], is a version which can be used to solve non-linear problems. We begin with a description for a two-level method where the coarse level is denoted by subscript-H and the fine level is denoted by subscript-h, followed by a general multilevel FAS scheme.

4.1 Two-level FAS multigrid

The difference between FAS and CS multigrid is that the coarse grid variable to be considered is not the error, e^H, but rather a coarse representation of the fine grid variable

$$x_H = R_h^H \tilde{x}_h + e_H.$$

So instead of solving the coarse grid error equation, we solve

$$A_H x_H = A_H R_h^H \tilde{x}_h + r_H$$

where r_H is the coarse residual, $R_h^H(f_h - A_h \tilde{x}_h)$. Define

$$f_H = R_h^H f_h + A_H R_h^H \tilde{x}_h - R_h^H A_h \tilde{x}_h$$

so the coarse problem can be written simply as

$$A_H x_H = f_H.$$

Note that in the case where A is a linear operator, then this coarse grid problem is equivalent to the coarse grid problem in CS multigrid:

$$\begin{aligned} A_H x_H \equiv A_H(R_h^H \tilde{x}_h + e_H) &= f_H \\ A_H R_h^H \tilde{x}_h + A_H e_H &= A_H R_h^H \tilde{x}_h + r_H \\ A_H e_H &= r_H. \end{aligned}$$

The FAS multigrid method is as follows: Given an initial guess, x_h, smooth on the equation

$$A_h x_h = f_h$$

to get a fine grid approximation, $\tilde{x}_h$. Restrict this fine grid approximation to the coarse grid

$$x_H = R_h^H \tilde{x}_h,$$

and calculate f_H

$$f_H = R_h^H f_h + A_H R_h^H \tilde{x}_h - R_h^H A_h \tilde{x}_h.$$

Solve the coarse level problem:

$$A_H x_H = f_H \tag{4.1}$$

either exactly (with direct or iterative methods) or inexactly with initial guess x_H.

Correct the fine grid approximation, $\tilde{x}_h$

$$x_h = \tilde{x}_h + R_H^h (x_H - R_h^H \tilde{x}_h).$$

Note that this is not the same correction as in CS, since the coarse level variable is not a coarse representation of the error, but rather is a coarse representation of the solution. One then further improves on this corrected approximation by performing a few post relaxations on $A_h x_h = f_h$.

4.2 Multilevel FAS multigrid

The FAS method for solving $A_h x_h = f_h$ can be generalized for any number of levels by recursively defining the solution process for the coarse grid problem in

terms of the two-level method described above for as many levels as needed. We will use the same notation as in Section 2.3.1. For clarity, denote the restriction operator from level $k+1$ to k as R_{k+1}^{k}, and the prolongation operator from level k to $k+1$ R_{k}^{k+1}.

Given l levels, define the corresponding problem on each level to be:

$$\begin{aligned} A_l x_l &= f_l \quad \text{finest grid problem} \\ A_{l-1} x_{l-1} &= f_{l-1} \quad \text{next coarser grid problem} \\ &\vdots \\ A_0 x_0 &= f_0 \quad \text{coarsest grid problem} \end{aligned} \tag{4.2}$$

where level l is the finest level and level 0 is the coarsest level and f_k is defined below.

Given some initial guess and starting at the finest level, level l, relax a few times to obtain an approximation, $\tilde{x}_l$ and descend to the next coarser level. To descend between two levels, $k+1$ to k for $k = l-1, \ldots, 0$, two operations are performed:

- **restrict** $\tilde{x}_{k+1}$ to level k by

$$x_k = R_{k+1}^{k} \tilde{x}_{k+1}.$$

- **calculate** f_k on level k as defined by

$$f_k = \begin{cases} f_h & \text{if } k = l \\ R_{k+1}^{k} f_{k+1} + A_k R_{k+1}^{k} \tilde{x}_{k+1} - R_{k+1}^{k} A_{k+1} \tilde{x}_{k+1} & \text{if } k < l. \end{cases} \tag{4.3}$$

Repeatedly relax on level k (with initial guess equal to the restriction of $\tilde{x}_{k+1}$) and descend until the coarsest level is reached. At the coarse level, the linear system, $A_0 x_0 = f_0$, may be solved by either direct methods or iteratively with initial guess x^0.

Given a coarse solution, ascend to the next finer level. To ascend between two levels, k to $k+1$ for $k = 0, \ldots, l-1$, perform the following:

- **correct** $\tilde{x}_{k+1}$ by

$$x_{k+1} = \tilde{x}_{k+1} + R_k^{k+1}(x_k - R_{k+1}^k \tilde{x}_{k+1})$$

Repeatedly relax and ascend until the finest grid is reached.

The general algorithm for FAS MG could be written as

Algorithm 4.2.1 *(FAS multigrid)*

0. If $k = 0$, then $x_0 = A_0^{-1} f_0$.

1. Presmoothing: Apply one transposed smoothing iteration with initial guess, x_k^0.

$$\tilde{x}_k = x_k^0 + S_k^T(f_k - A_k x_k^0)$$

2. Descend to coarse level:

a. Restrict the solution for coarse initial guess

$$x_{k-1}^0 = R_k^{k-1} \tilde{x}_k$$

b. Calculate coarse right hand side f^{k-1}

$$f_{k-1} = R_k^{k-1} f_k + A_{k-1} x_{k-1}^0 - R_k^{k-1} A_k \tilde{x}_k$$

3. "Solve" $A_{k-1} x_{k-1} = f_{k-1}$ (recursive call to FAS MG with initial guess $x_{k-1}^0 = x_{k-1}$)

4. Interpolate back and correct:

$$\hat{x}_k = \tilde{x}_k + R_{k-1}^k (x_{k-1} - R_k^{k-1} \tilde{x}_k).$$

5. Postsmoothing: Apply one smoothing iteration with initial guess $\hat{x}_k$.

$$x_k = \hat{x}_k + S_k(f_k - A_k \hat{x}_{k+1})$$

Many variations on the general FAS multigrid method described above can be found by changing the definition of one or more of the following components in the method:

- obtaining an initial guess,
- the definition of A_k,

- the definition of R_k^{k+1},
- the number of pre- and post-smoothings done on each level,
- solution of the coarse problem.

4.3 Choice of coarse operator

The choice for coarse level operators must also be considered. There are two different ways to define a coarse operator. One way is to simply rediscretize the problem, given the coarse grids. Another way is called the Galerkin method, where the coarse operator is defined by $A^H = R_h^H A^h R_H^h$.

The choice of whether to use a coarse problem arising from rediscretizion or from Galerkin coarsening is problem-dependent. There are advantages to each method. Rediscretization can be done automatically, using the same software that was used to discretize the fine level problem. Also, rediscretization approximates the fine level problem well as long as the coarse grids are of good quality.

Alternatively, Galerkin coarsening can be used when purely algebraic methods are used, when the coarse grids are not given. However, Galerkin coarsening usually results in stiffness matrices which are more dense than those arising from rediscretizations (see Fig. 4.1). Because the adjacency matrices inherit all the nearest neighbors from the previous grid, recursive application of Galerkin coarse

operators for many levels can quickly become prohibitive.

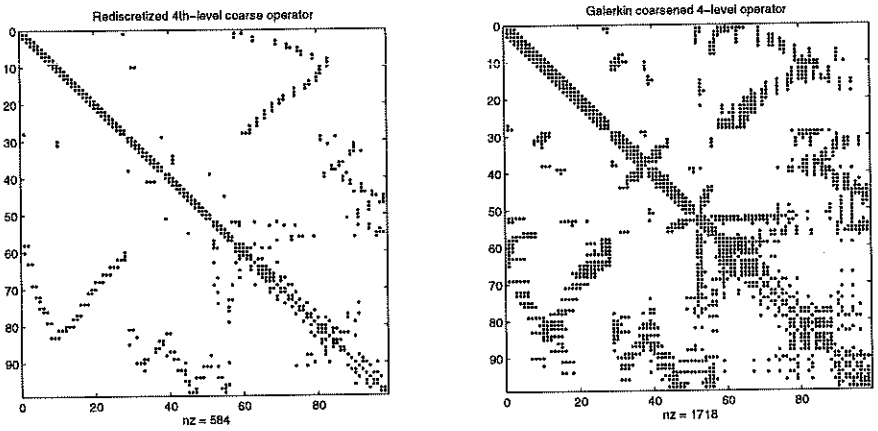


Figure 4.1: Spy plots for 4th-level coarse operators: rediscretized (left) and Galerkin (right).

4.4 Elliptic problems

Consider Poisson's problem

$$\begin{aligned} -\triangle u &= f, \quad \text{in } \Omega \\ u &= 0, \quad \text{on } \Gamma_D \end{aligned} \tag{4.4}$$

Figure 4.2 shows the convergence histories for solving the elliptic pde using the FAS multigrid described in Section 4.2 on various uniform and unstructured grids. The number of unknowns on each fine grid increases by a factor of approximately four; the meshwidths decrease by about a factor of two. Here, we solve equation (4.4) with homogeneous Dirichlet boundaries on a unit square. Three-level

FAS multigrid is used where coarse level problems are calculated by rediscretizing the problem on a coarse mesh. Two pre- and post-symmetric Gauss-Seidel smoothings are performed at each level, and an exact solve is done at the coarsest level. FAS V-cycles are stopped when the norm of the residual has decreased by a factor of 10^{-6}. The reduction factors for each problem are shown in the legend. We see that the method is independent of meshwidth for both the structured and unstructured cases.

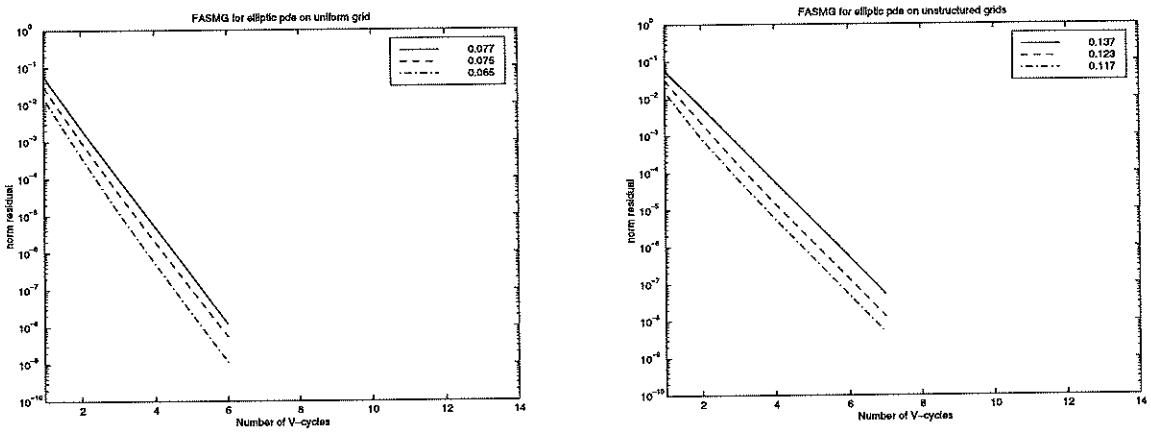


Figure 4.2: FAS MG for solving an elliptic pde on the unit square with uniform (left) and unstructured (right) grids. Uniform grids: Solid line (289 unknowns), dashed line (1089 unknowns), and dash-dotted line (4225 unknowns). Unstructured grids: Solid line (310 unknowns), dashed line (1064 unknowns), and dash-dotted line (4164 unknowns).

4.5 Eigenvalue problems

We will use the FAS multigrid method to calculate the eigenvalues of A, which will be viewed as a non-linear problem, since x and λ are both unknowns. Since the solution of the eigenvalue problem is unique up to a multiplicative constant, a normalization constraint is added so the eigenvalue problem can be written as

$$Ax = \lambda x, \quad \eta(x) = 1 \tag{4.5}$$

where $\eta(x)$ is some normalization functional of the unknown x. Typically, the normalization is chosen such that $\|x\|_2 = 1$.

Brandt-McCormick-Ruge's [5] method for solving differential eigenvalue problems uses a FAS multigrid method with Ritz projection for the calculation of several eigenvector/value pairs. We will adapt their eigensolver for unstructured meshes. The FAS setup is done in the usual manner:

Given l levels, define the corresponding problem on each level to be:

$$\begin{aligned}
(A_l - \lambda I_l)x_l &= f_l, & \eta_l(x_l) &= \sigma_l & &\text{finest grid problem} \\
(A_{l-1} - \lambda I_{l-1})x_{l-1} &= f_{l-1}, & \eta_{l-1}(x_{l-1}) &= \sigma_{l-1} & &\text{next coarser grid problem} \\
&\vdots & & & & \\
(A_0 - \lambda I_0)x_0 &= f_0, & \eta_0(x_0) &= \sigma_0 & &\text{coarsest grid problem}
\end{aligned} \tag{4.6}$$

where I_k is the identity matrix on levels $k = 0, \ldots, l$. f_k is defined by $f_l = 0$ and

for $k < l$,

$$\begin{aligned} f_k &= R^k_{k+1} f_{k+1} + (A_k - \lambda I_k) R^k_{k+1} \tilde{x}_{k+1} - R^k_{k+1}(A_{k+1} - \lambda I_{k+1})\tilde{x}_{k+1} \\ &= R^k_{k+1} f_{k+1} + A_k R^k_{k+1} \tilde{x}_{k+1} - R^k_{k+1} A_{k+1} \tilde{x}_{k+1} \end{aligned}$$

and σ_k may be defined analogously by $\sigma_l = 1$ and for $k < l$,

$$\sigma_k = R^k_{k+1}\sigma_{k+1} + \eta_k(R^k_{k+1}\tilde{x}_{k+1}) - R^k_{k+1}\eta_{k+1}(\tilde{x}_{k+1}) \tag{4.7}$$

Note that in this case, the definition of f_k does not differ from the definition given in equation (4.3). In addition, the definition of σ_k (4.7) can be simplified. This definition is constructed to satisfy $\eta_l(x_l) = 1$, but in fact, the final solution need not be scaled to one. Brandt, et. al. chose a simpler normalization which was easier to enforce while still providing a zero correction at convergence. For further details, see [5].

To find the first eigenvector/value pair given an initial guess for the eigenvector and eigenvalue on the fine level, a standard FAS V-cycle multigrid step is taken keeping λ fixed at all but the coarsest level. At the coarsest level, the non-linear problem $Ax - \lambda x = f$, is solved by repeated relaxation, λ-update, and eigenvector normalization steps. The justification for this is that only the high frequency errors are being damped out at the fine grid levels, so the size of the solution obtained at the fine levels remains relatively constant, thus no normalization is necessary there. The size of the solutions will significantly change only at the coarsest level, so this is where normalization is performed. The solution to the

coarse problem is a coarse representation of the eigenvalue solution of the fine problem, so normalization should be done with respect to the fine level problem.

The eigenvalue, λ, is updated at each iteration by calculating the modified Rayleigh quotient, which is given by:

$$\begin{aligned} \lambda \equiv Ray(A) &= \min_{x \neq 0} \|Ax - \lambda x - f\|_2 \\ &= \frac{x^T(Ax - f)}{x^T x}. \end{aligned} \tag{4.8}$$

To find the next eigenvector/value pair, repeat the FAS V-Cycle MG above, introducing an orthogonalization step in addition to the λ-update and normalization step at the coarse level. Again, since the coarse solution is a coarse representation of the fine level problem, the orthogonalization should also be done with respect to the fine level problem.

When all V-cycles are complete, the resulting eigenvector/value pairs can be further refined by a Ritz step. This should be inexpensive to do if the number of smallest eigenvalues sought is small.

In the next few sections, the following components were used in each method unless otherwise noted:

- initial guess was found exactly on the coarse grid using the LAPACK symmetric eigenvalue routine `dsyev` for the coarse level problem $A_0 x_0 = \lambda x_0$ and then interpolated up through the levels to provide an initial fine level guess,
- 3-level FAS V-cycle multigrid,

- two pre- and post- symmetric Gauss-Seidel smoothings at each level,
- coarse level problems were constructed using Galerkin coarsening ($A_k = R_{k+1}^k A_{k+1} R_k^{k+1}$),
- the coarse level problem, $Ax = \lambda x + g$, was solved iteratively using 20 symmetric Gauss-Seidel followed by orthogonalization/normalization (with respect to the fine grid solution) and λ update using equation (4.8).
- 3 eigenvector/value pairs were found for the Ritz problem.
- grid hierarchy:

Level	Number of unknowns					
	Unstructured grids			Uniform grids		
fine	4164	1061	310	4225	1089	289
interm.	1061	310	86	1089	289	81
coarse	310	86	24	289	81	25

Figure 4.4 shows the convergence histories for solving the elliptic eigenvalue problem

$$
\begin{aligned}
-\triangle u &= \lambda u, \quad \eta(u) = 1 \text{ in } \Omega \\
\frac{\partial u}{\partial n} &= 0 \quad \text{on } \partial\Omega
\end{aligned}
$$

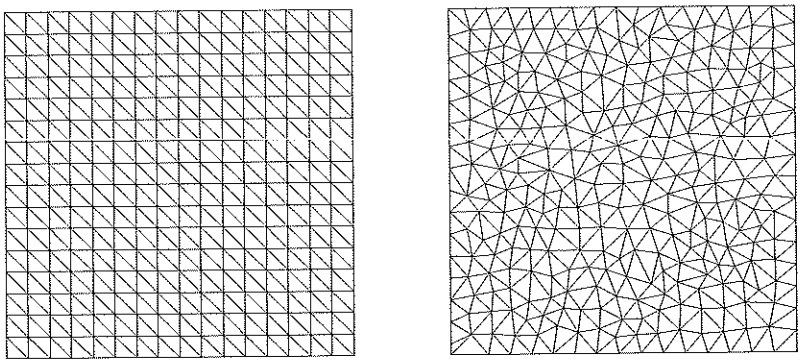


Figure 4.3: A uniform grid (left) and an unstructured grid (right).

on a uniform grid and an unstructured grid, both on the unit square. The method on the uniform grid achieves roughly the same convergence rate as in solving Poisson's equation, but the unstructured grid shows a degradation which does not occur with the uniform grid. This appears to be an anomaly, as other grids do not exhibit the same kind of deterioration.

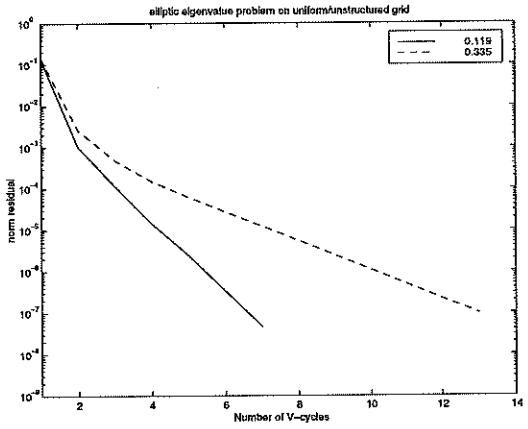


Figure 4.4: FAS MG for solving an elliptic eigenvalue problem on the unit square for uniform and unstructured grids. Solid line is the problem on a uniform grid (289 unknowns), dashed line is on an unstructured grid (310 unknowns).

4.6 Generalized eigenvalue problems

The FAS MG method for solving eigenvalue problems described in the previous section can be used for the generalized problem with some slight modification. For the elliptic eigenvalue problem

$$-\triangle u = \lambda u, \text{ in } \Omega$$

with some boundary conditions, discretization using finite elements leads to solving the generalized eigenvalue problem

$$Ax = \lambda Bx, \tag{4.9}$$

where A is the usual bilinear form and B is typically the mass matrix:

$$B_{i,j} = \int_{\Omega} \phi_i \phi_j \, dx,$$

with finite element basis functions ϕ_i and ϕ_j.

Given l levels, define the corresponding problem on each level to be:

$$\begin{array}{rcl@{\qquad}rcll}
(A_l - \lambda B_l)x_l & = & f_l, & \eta_l(x_l) & = & \sigma_l & \text{finest grid problem} \\
(A_{l-1} - \lambda B_{l-1})x_{l-1} & = & f_{l-1}, & \eta_{l-1}(x_{l-1}) & = & \sigma_{l-1} & \text{next coarser grid problem} \\
& \vdots & & & & & \\
(A_0 - \lambda B_0)x_0 & = & f_0, & \eta_0(x_0) & = & \sigma_0 & \text{coarsest grid problem}
\end{array} \tag{4.10}$$

where A_k is the usual bilinear form and B_k is typically the mass matrix on levels $k = 1, \ldots, l$. f_k is defined by $f_l = 0$ and for $k < l$,

$$f_k = R_{k+1}^k f_{k+1} + (A_k - \lambda B_k) R_{k+1}^k \tilde{x}_{k+1} - R_{k+1}^k (A_{k+1} - \lambda B_{k+1}) \tilde{x}_{k+1}.$$

Note that in this case, the definition for f_k differs slightly from the definition given in equation (4.3 since $B_k R_{k+1}^k \neq R_{k+1}^k B_{k+1}$.

The eigenvectors, x, will be orthogonalized with respect to the B-norm, and the corresponding eigenvalue is given by the modified Rayleigh quotient in the appropriate norm

$$\begin{aligned} Ray(A, B) &= \min_{x \neq 0} \|Ax - \lambda Bx - f\|_B \\ &= \frac{x^T (Ax - f)}{x^T B x}. \end{aligned}$$

where $\|\cdot\|_B$ is defined by $\|z\|_B^2 = z^T B^{-1} z$.

Alternatively, we could also consider the matrix B to be some sort of scaling matrix,

$$\tilde{I}_k = (R_k^{k+1})^T R_k^{k+1}.$$

Using the convergence results on the uniform grid as a baseline, the FAS MG method on the unstructured grid achieves similar convergence rates as it does on the uniform grid for both the generalized eigenvalue problem $Ax = \lambda Mx$, and for the scaled eigenvalue problem $Ax = \lambda \tilde{I}^k x$. The degradation of the method on the unstructured grid as seen in Figure 4.4 is no longer there. Figure 4.5 shows the

results for solving each problem on a uniform grid and an unstructured grid of comparable size.

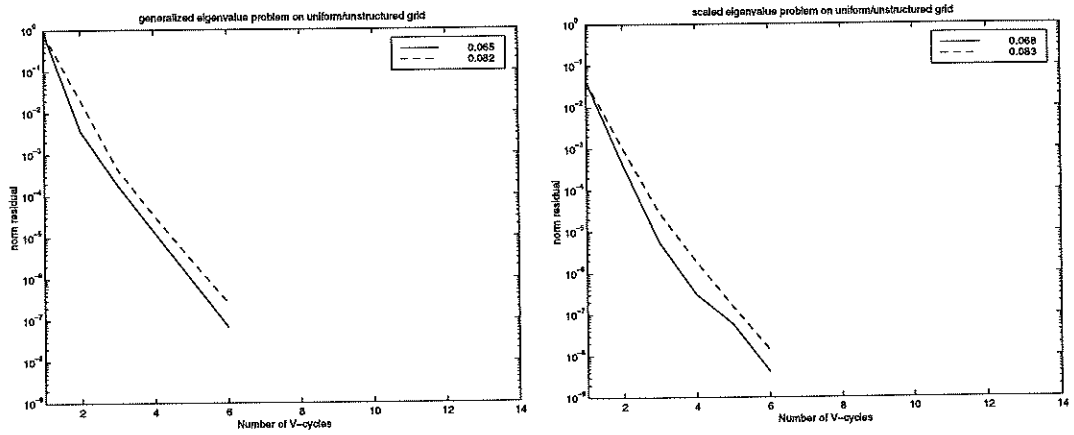


Figure 4.5: FAS MG for solving a generalized elliptic eigenvalue problem (left) and scaled elliptic eigenvalue problem (right) on the unit square. Solid line is the problem on a uniform grid (289 unknowns), dashed line is on an unstructured grid (310 unknowns).

Figure 4.6 shows the convergence histories for solving a generalized elliptic eigenvalue problem using FAS multigrid on unstructured grids with varying meshwidths. Here, we solve equation (4.9) on a unit square. The number of unknowns on each grid increases by a factor of approximately four; the meshwidths decrease by about a factor of two. We see that the method is independent of the meshwidth.

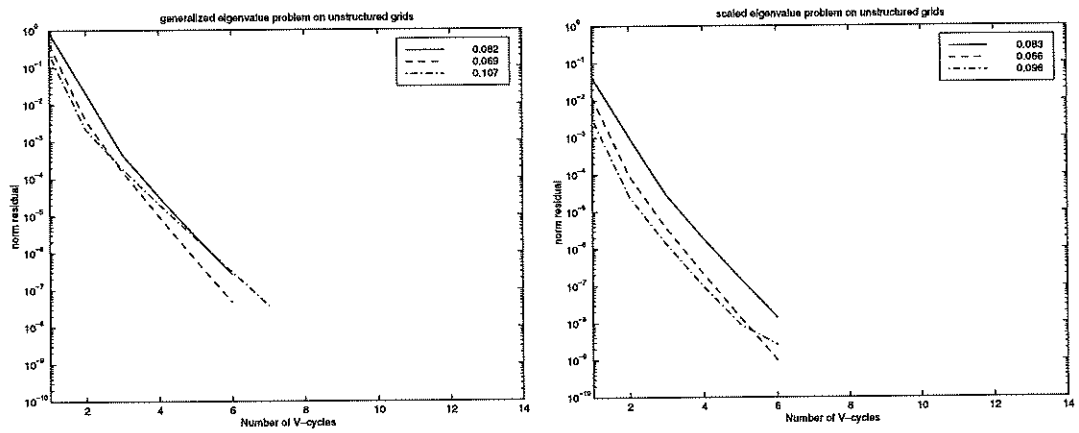


Figure 4.6: FAS MG for solving a generalized elliptic eigenvalue problem (left) and scaled elliptic eigenvalue problem (right) on the unit square with unstructured grids of varying meshwidths. Solid line is the problem with 310 unknowns dashed line has 1064 unknowns, and dash-dotted line has 4164 unknowns.

CHAPTER 5

Multilevel Spectral Partitioning of Unstructured Grids

As distributed-memory parallel computers become more commonplace, there has been a great deal of effort spent to ensure that the performance of numerical algorithms is not sacrificed as a result of parallelism. *Scalability* is the ability of an algorithm to speed up proportionally with the number of added processors. In other words, two processors should be able to get the job done twice as fast. However, as the number of processors increases, there is a point when another factor begins playing an increasingly larger role, that being *communication* among processors.

Two important points to remember are that communication among processors is an expensive operation, and that bottlenecks should be avoided. For problems discretized on a grid, this means that in order to minimize communication, the number of nearest neighbors residing on different processors should be minimized. In other words, the number of edges cut in the grid should be minimized.

The second point is that bottlenecks occur when the amount of work on each processor is not evenly distributed. Assuming that the same amount of work is done at each vertex, then the number of vertices per processor should be uniformly

distributed.

These two issues: minimizing the number of edges cut and maintaining a balanced load, lead to the partitioning problem. The graph partitioning problem is an NP-complete problem and many different approaches have been used for approximating the solution. There is usually a tradeoff between the speed to compute a partition and the quality of the resulting partition. However, one should strive to obtain a quality partition despite its potential costliness to compute because a good partition often leads to better convergence rates of iterative solutions [51]. This is particularly useful if a grid is reused often.

Of the different approaches, the spectral bisection method introduced by [47] yields very high-quality partitions but is expensive to compute. Them main cost of the spectral method is in solving the associated discrete eigenvalue problem. In this chapter, it will be shown that when the graph is a standard finite element mesh, the graph Laplacian is spectrally equivalent (up to a diagonal scaling) to the mesh Laplacian. This equivalence can be exploited to adapt recently developed multilevel elliptic algorithms for unstructured grids to solving the graph partitioning problem with a true multigrid convergence rate. Using the tools from Chapters 3 and 4, an unstructured multigrid spectral partitioner will be developed and some numerical results will be provided to demonstrate that optimal convergence rates are retained.

5.1 Partitioning algorithms

Pothen [46] has a very nice survey of the latest graph partitioning algorithms. Detailed descriptions of many of the partitioning approaches which have been developed can be found in this survey. Some of the most important partitioning algorithms are summarized below.

- **Kernighan-Lin** Given an initial partition of a vertex set, this method swaps the vertices between the two sets to reduce the number of edges between them. Repeated application is done until it is either no longer possible to reduce the number of edges between them, or until a prescribed number of iterations has been reached. The quality of partitions resulting from this method is very dependent on the quality of the initial partition it is given. The Kernighan-Lin algorithm is most frequently used in many partitioners as a post-processing refinement step.

- **greedy** These methods find partitions by first selecting vertices which are approximately at the greatest distance apart in the graph and grows subsets of vertices until the required number of vertices is obtained.

- **coordinate bisection** This method bisects a graph using coordinate information at the vertices. A straight line bisector is found that divides the vertices into two approximately equal sets. These methods make no use of edge information, and so they make partitions which do not minimize the

number of edges cut.

- **inertial bisection** Inertial methods use the coordinates of the vertices of a graph to find a bisection. The vertices of the mesh are considered as discrete points and the center of gravity of the mesh is calculated. Then the axis of minimum angular momentum is found and used as a bisector.

- **spectral bisection** Spectral methods produce very high quality partitions, but they are expensive to calculate. The main cost of the spectral method is in solving for the Fiedler vector (the lowest nontrivial eigenvector of the associated graph Laplacian). More discussion of this method follows in the next section.

- **multilevel bisection** These algorithms make high-quality but expensive partitioners practical to use by performing the costly steps of partitioners on an appropriately smaller problem and then "uncoarsening" the resulting partition back to the original graph.

5.2 The spectral bisection algorithm

Let $G = (V, E)$ be an undirected graph, where $V = \{v_i\}_{i=1}^{n}$ is the set of vertices of the graph, and $E = (v_i, v_j)$ is the set of edges. The Laplacian of the graph is

defined as:

$$L_{ij} = \begin{cases} \deg(v_i) & \text{if } i = j \\ -1 & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

Some characteristics of L:

- $L = D - A$ where D is the diagonal matrix of the degrees of each vertex, and A is the adjacency matrix of the graph.

- The bilinear form associated with L can be written:

$$\begin{aligned} u^T L u &= \sum_i \sum_j u_i L_{ij} u_j \\ &= -2 \sum_{(i,j)\in E} u_i u_j + \sum_i u_i{}^2 \deg(i) \\ &= -2 \sum_{(i,j)\in E} u_i u_j + \sum_{(i,j)\in E} (u_i{}^2 + u_j{}^2) \\ &= \sum_{(i,j)\in E} (u_i - u_j)^2. \end{aligned}$$

- L is symmetric positive semidefinite.

- The smallest value and its corresponding vector is $(0, \vec{1})$.

- The second eigenvector of L is known as the Fiedler vector and the following result is due to Fielder:

  Let P denote the vertices of a graph G whose second eigenvector components are non-positive. The subgraph (P, E_P) is a connected subgraph of G. Similarly for non-negative.

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two connected subgraphs of G such that $V_1 \cup V_2 = V$ and define $x \in R^n$ to be a vector such that

$$x_i = \begin{cases} +1 & \text{if } i \in V_1 \\ -1 & \text{if } i \in V_2. \end{cases}$$

Then a balanced bisection for G occurs for x when $\sum_i x_i = 0$ and the minimum number of edges cut is given by minimizing the bilinear form over all such x, since

$$x^T L x = \sum_{(i,j) \in E} (u_i - u_j)^2 = 4 \text{ number of edges cut}$$

Since the bisection problem is NP-complete, the problem cannot be solved exactly. The spectral bisection method developed by Pothen-Simon-Liou [47] finds a partition of G by relaxing the constraints to allow for x continuous and $\|x\| = n$. Then the partitioning problem becomes one for solving the discrete Laplacian eigenvalue problem, $Lu = \lambda u$, for the Fiedler vector. A bisection of the graph is found by sorting the components of the resulting Fielder vector, and assigning half the vertices to one subgraph and the other half to another subgraph. The method is can be used recursively on the resulting subgraphs until the desired number of 2^p partitions is attained, and is known as the recursive spectral bisection (RSB) method.

5.3 Currently available software

The spectral bisection problem has been extensively investigated lately and because of this, many practical implementations of partitioning algorithms are currently available.

The first spectral bisection partitioner solved the eigenvalue problem using a modified Lanczos algorithm to compute the Fiedler vector but this was too expensive to be really useful. The multilevel spectral bisection partitioner of Barnard and Simon was one of the first practical partitioners available. Recursive application of the contraction step is applied until an appropriately small coarse problem is attained. Once a suitably sized coarse problem is attained, the coarse eigenvalue problem is solved in any of a number of ways. Barnard and Simon's method solves the coarsened eigenvalue problem using a Lanczos algorithm. The solution is then interpolated up and used as an initial guess for the Rayleigh quotient iteration. Since the initial guess is good, the Rayleigh quotient iteration will converge quickly. SYMMLQ was used to solve the symmetric indefinite system in the Rayleigh quotient iteration.

The Chaco [30] implementation was a modification of the multilevel RSB method of Barnard and Simon. The two methods differ mainly in the way the fine grid problem is contracted to a coarse problem. Barnard and Simon's multilevel RSB method used a maximal independent set to generate a smaller coarse problem while Chaco employs a maximal matching of edges to coarsen the prob-

lem. In addition, the Chaco software also allowed for easy experimentation with many different parameters to customize the partitioner for different situations.

The Metis software by Karypis and Kumar [36] provides a very fast partitioner with many options for customization. Though it is a multilevel method, it is not spectral in the sense that it does not solve the spectral partitioning problem on the fine graph. Metis also uses maximal matching of edges to generate a coarse problem, with many different variants provided.

Parallel and dynamic variants of many of these popular partitioners are also becoming available.

Chaco and Metis allow for many choices and combinations of eigensolvers at the coarse level, which includes implementations of RSB. Most implementations follow up the partition with a Kernighan-Lin local refinement which greatly improves the partition. For further details, see the user guides [31, 37].

5.4 Spectral equivalence of the graph and grid Laplacian

In this section, we show that for standard finite element meshes, the discrete Laplacian is spectrally equivalent to the continuous Laplacian. This equivalence can then be used to show that multilevel elliptic eigenvalue solver can be used to solve the graph partitioning problem with true multigrid convergence rates.

Let L be the Laplacian of the graph of G as defined in (5.1). Let A be the

stiffness matrix arising from the finite element discretization of the Laplacian.

Theorem 5.4.1 *The graph Laplacian L is equivalent to the stiffness matrix A in the sense that there exist some number c_1, c_2 independent of mesh size, such that*

$$c_1 w^T A w \leq w^T L w \leq w^T A w, \quad \forall w \in R^p \tag{5.2}$$

Proof. Let K be any triangle ($n = 2$) or tetrahedron ($n = 3$) in T^h. let $\hat{K}$ be the reference element with $n+1$ vertices $\hat{x}_i$, for $i = 1, \ldots, n+1$.

Let $F : \hat{K} \to K$ be the affine mapping defined by $F(\hat{x}) = B\hat{x} + b$ such that the vertices $\hat{x}_i, i = 1, \ldots, n+1$ of the element $\hat{K}$ are mapped to the vertices $x_i, i = 1, \ldots, n+1$ of the element K. Let u be any linear function on K, with nodal values u_i at x_i. Clearly, $\hat{u}$ has the same values at its vertices as u. Moreover, we have

$$\|\nabla u\|^2_{L^2(K)} \leq C|\det(B)|\|B^{-1}\|^2 \|\nabla \hat{u}\|^2_{L^2(\hat{K})} \tag{5.3}$$

where $\|\cdot\|$ is the spectral norm.

By the definition of the matrix norm and the shape regularity assumption, we can derive the following bounds [21]

$$\|B^{-1}\| \leq h_K^{-1}, \quad \|B\| \leq C h_K, \tag{5.4}$$

and

$$|B| \leq \text{meas}(K) \leq C h_K^n, \quad |B^{-1}| \leq C h_K^{-n}. \tag{5.5}$$

Then from (5.3), it follows that

$$\|\nabla u\|^2_{L^2(K)} \leq C h_K^{n-2} \|\nabla \hat{u}\|^2_{L^2(\hat{K})}. \tag{5.6}$$

Now consider the finite dimensional quotient space $P_1(\hat{K})/R$. It is easy to check that

$$|||\hat{u}||| \stackrel{\text{def}}{=} \left(\sum_{\hat{x}_i, \hat{x}_j \in \hat{K}} (\hat{u}(\hat{x}_i) - \hat{u}(\hat{x}_j))^2 \right)^{1/2}$$

is a norm on $P_1(\hat{K})/R$. We also know that $\|\nabla \hat{u}\|^2_{L^2(\hat{K})}$ is a norm in $P_1(\hat{K})/R$. Therefore, they are equivalent and from (5.6), this implies

$$h_K^{n-2} \|\nabla u\|^2_{L^2(K)} \leq C \|\nabla \hat{u}\|^2_{L^2(\hat{K})} \simeq C \sum_{x_i, x_j \in K} (u(x_i) - u(x_j))^2 \tag{5.7}$$

since $\hat{u}(\hat{x}_i) = u(x_i)$.

Similarly as (5.3), we have

$$\|\nabla \hat{u}\|^2_{L^2(\hat{K})} \leq C |B^{-1}| \|B\|^2 \|\nabla u\|^2_{L^2(K)} \leq C h_K^{n-2} \|\nabla u\|^2_{L^2(K)}, \tag{5.8}$$

combining this with (5.7) gives

$$h_K^{n-2} \|\nabla u\|^2_{L^2(K)} \leq \sum_{x_i, x_j \in K} (u(x_i) - u(x_j))^2 \leq h_K^{n-2} \|\nabla u\|^2_{L^2(K)}. \tag{5.9}$$

From (5.9), we obtain

$$\sum_{K \in \mathcal{T}^h} \int_K h_K^{n-2} |\nabla u|^2 \simeq \sum_{i,j \in E} (u_i - u_j)^2 \tag{5.10}$$

where each edge is just counted once.

But note that $u^T L u = \sum_{(i,j) \in E} (u_i - u_j)^2$, hence we have proved the equivalence.

□

5.5 FAS MG algorithm for the spectral partitioning problem

The spectral equivalence of the graph Laplacian and the mesh Laplacian motivates us to use the same tools for solving the discrete eigenvalue problem of spectral partitioning as we use for the elliptic eigenproblem. Although the solution to the spectral partitioning problem is a discrete Laplacian eigenvalue problem, we can still use the FAS multigrid eigensolver described in Chapter 4 for solving the partitioning problem. Because of the equivalence of the continuous and discrete problems, the linear interpolants which arise from a multigrid solver on a continuous Laplacian can be used for the discrete problem. In this way, the actual fine-level spectral partition is being solved instead of some coarsened problem which gets projected up to the fine level.

The FAS MG partitioner, then, uses all the same machinery as the elliptic eigenvalue solver: the same interpolations and restrictions, the same coarse grid solution process. Some general issues:

1. Galerkin coarsening makes more sense for solving the graph Laplacian, because it is a discrete problem based on the adjacencies of the fine graph, which likely has nothing to do with those of rediscretized coarse graphs. However, as we mentioned earlier, Galerkin coarsening generally results in matrices which are denser than those which arise from rediscretizations.

2. The partitioning problem only requires the second eigenvector. So the FAS

MG method is used to solve the discrete problem, but modified so as to seek the second eigenvector, since the first eigenvector/value pair is known to be $(\vec{1}, 0)$. Note that in the Ritz steps, we can improve our approximation to the second eigenvalue/vector pair if we have a larger subspace to Ritz on. Because of this, I usually find the first three vector/value pairs.

To recap, the components of the multilevel solver include:

- initial guess was found exactly on the coarse grid using the LAPACK symmetric eigenvalue routine `dsyev` for the coarse level problem $A_1 x_1 = \lambda x_1$ and then interpolated up through the levels to provide an initial fine level guess,
- 3-level FAS V-cycle multigrid,
- two pre- and post- symmetric Gauss-Seidel smoothings at each level,
- coarse level problems were constructed using Galerkin coarsening ($A_k = R_{k+1}^k A_{k+1} R_k^{k+1}$),
- the coarse level problem, $A_0 x_0 = \lambda x_0 + g_0$, was solved iteratively using 20 symmetric Gauss-Seidel followed by orthogonalization/normalization (with respect to the fine grid solution) and λ update using equation (4.8).
- 3 eigenvector/value pairs were found for the Ritz problem.

Figure 5.1 shows the FAS MG convergence histories for solving the discrete

Laplacian eigenvalue problem $Lu = \lambda u$ on unstructured grids of varying meshwidths. It can be observed that the method is independent of meshwidth.

When the discrete Laplacian eigenvalue problem with scaled identity is solved on the same three grids, we still get the usual multigrid convergence rates independent of meshwidth (see Figure 5.1).

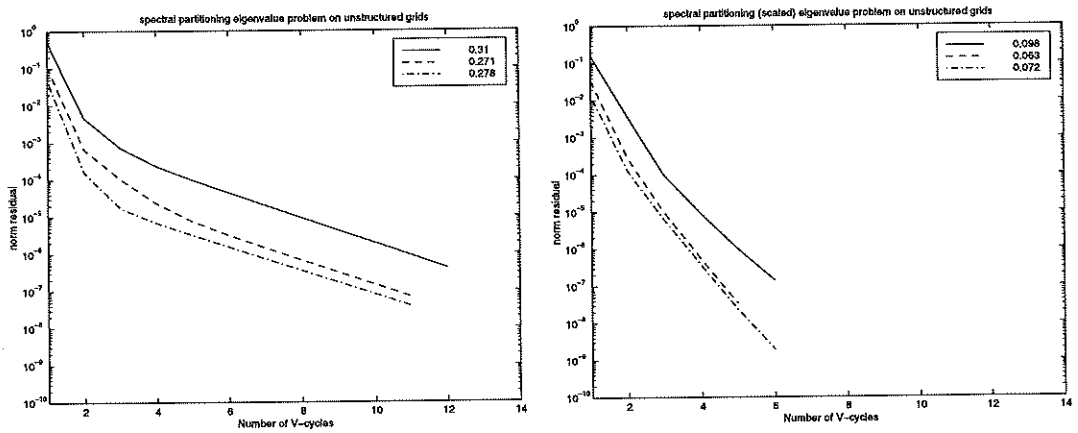


Figure 5.1: FAS MG for the spectral bisection problem on the unit square with unstructured grids of varying meshwidths. The coarse problems are defined to be $Lu = \lambda u$ (left) and $Lu = \lambda \tilde{I} u$ (right) on all levels. Solid line (310 unknowns), dashed line has (1064 unknowns), and dash-dotted line has (4164 unknowns).

The next series of figures (Fig. 5.2–5.11), show the convergence of the solver on different grids and compares efficiency of the Galerkin coarsened operator (left) with rediscretized operators (right) for the discrete spectral bisection problem. It should be noted that the Galerkin coarsened operators always performed better than the rediscretized operators and that the problems with the scaled identity $\tilde{I}$

(dashed lines) also performed a little better than the standard eigenvalue problems (solid lines).

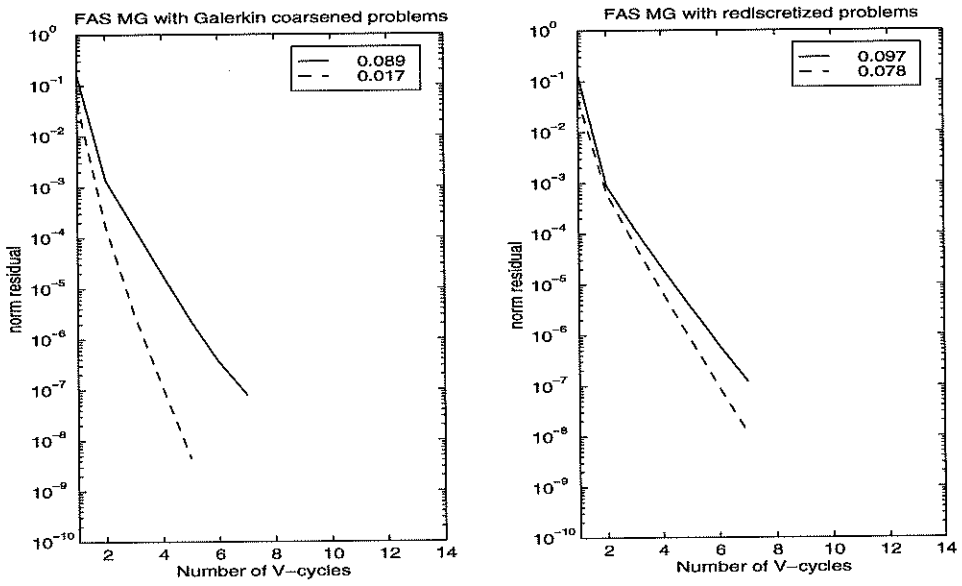


Figure 5.2: Uniform square grid (unknowns 289).

Next, we show a comparison for solving the spectral bisection partitioning problem using the FAS MG scheme with the various interpolants described in Section 3.3. Since the spectral bisection problem has Neumann boundary conditions, we expect that using a zero extension interpolation for intergrid transfers would not be accurate enough at non-matching boundaries, and this is in fact, what we observe. Interpolant $\mathcal{I}_h^0$ is the zero extension interpolation, interpolant $\mathcal{I}_h^1$ is the nearest edge interpolation, and interpolant $\mathcal{I}_h^2$ is the nearest element interpolation.

Figure 5.13 shows the bisection for the airfoil grid using the various partitioners available. The options for Chaco and Metis were set so a close comparison could be

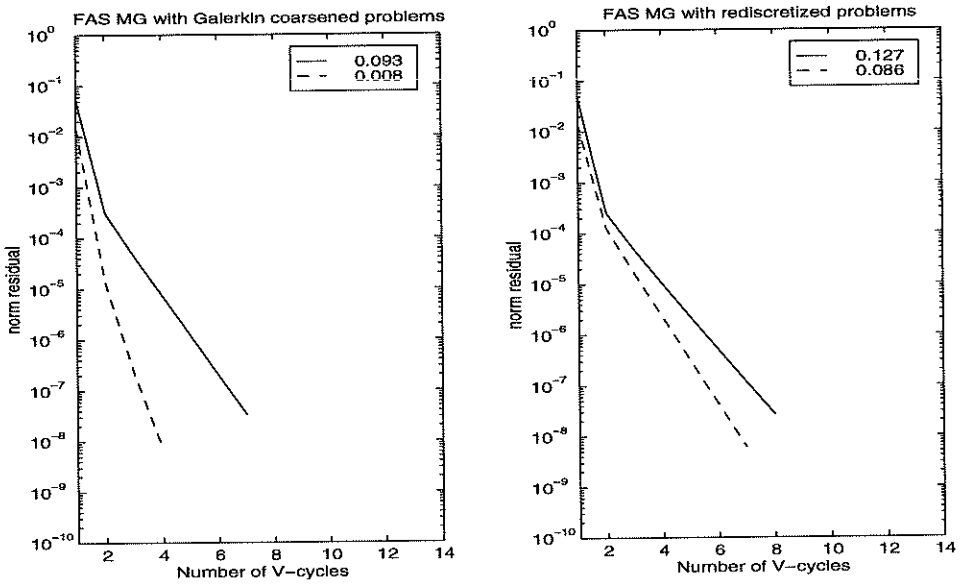


Figure 5.3: Uniform square grid (unknowns 1089).

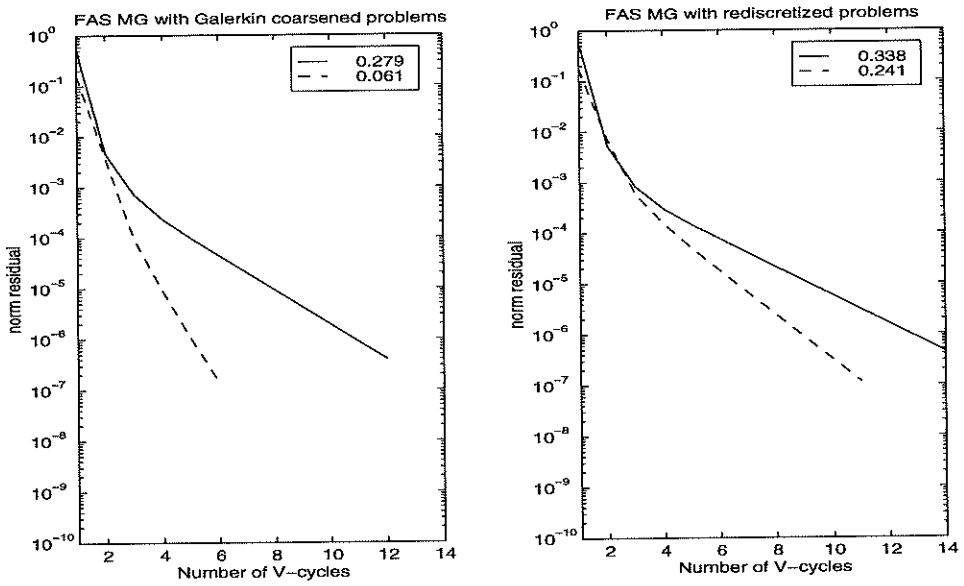


Figure 5.4: Unstructured square grid (unknowns 310).

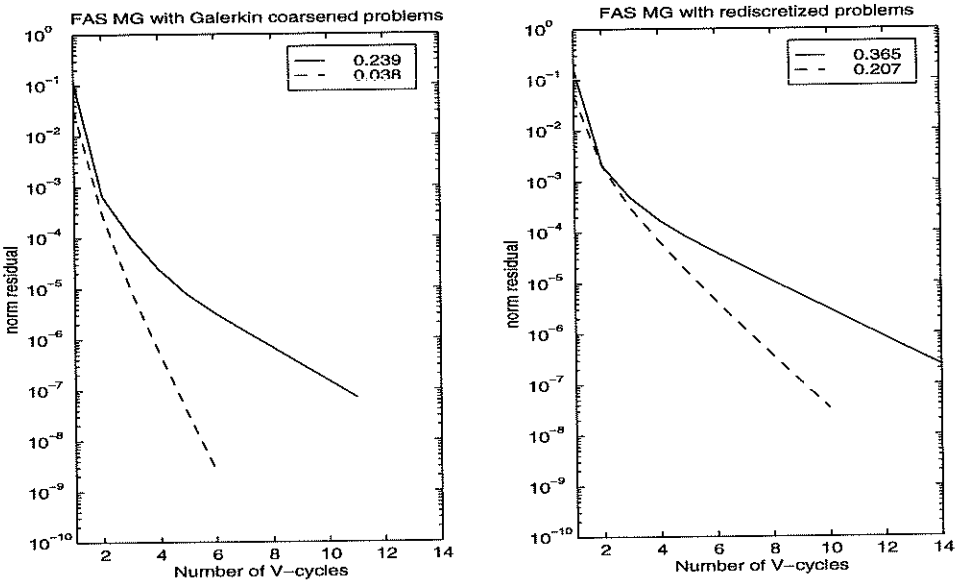


Figure 5.5: Unstructured square grid (unknowns 1064).

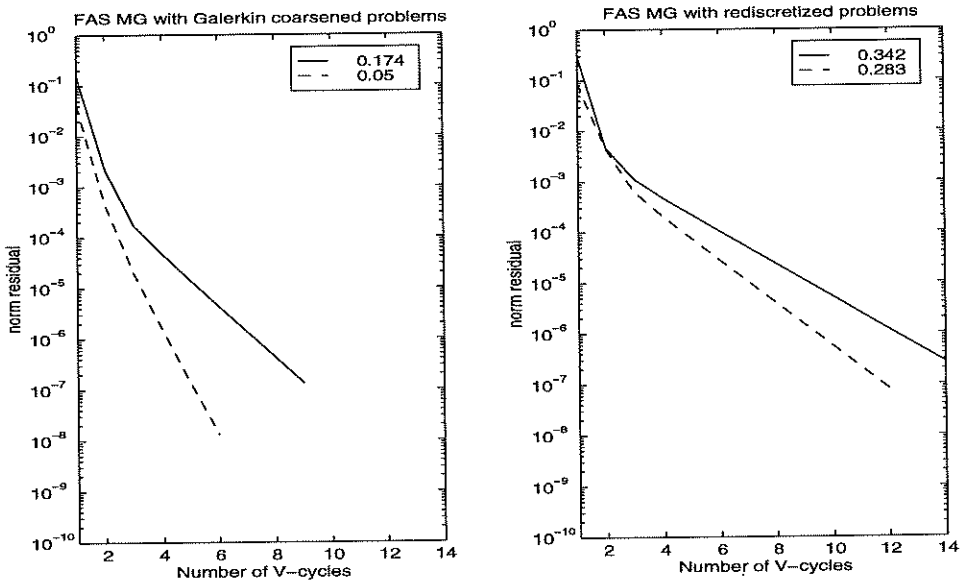


Figure 5.6: Structured circle grid (unknowns 654).

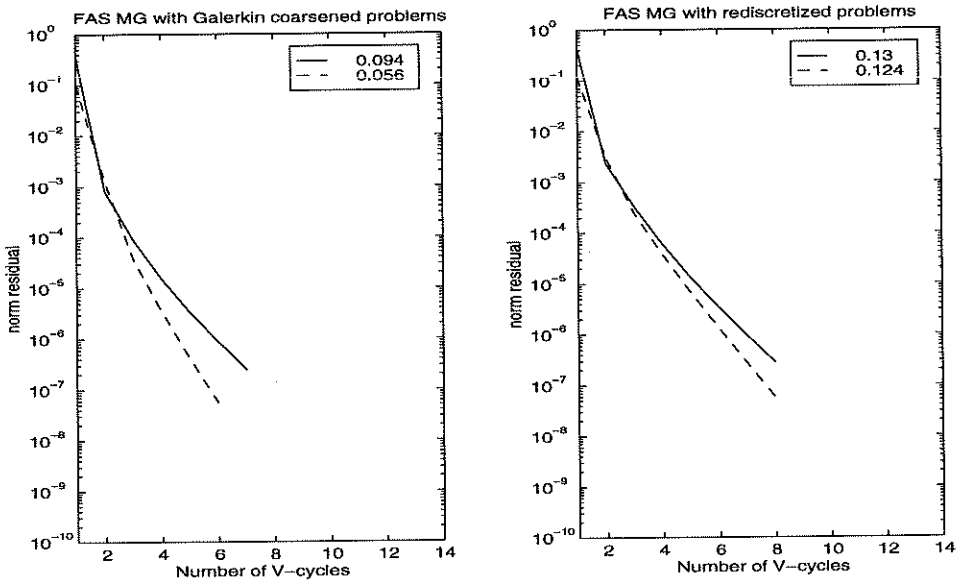


Figure 5.7: Unstructured annulus grid (unknowns 610).

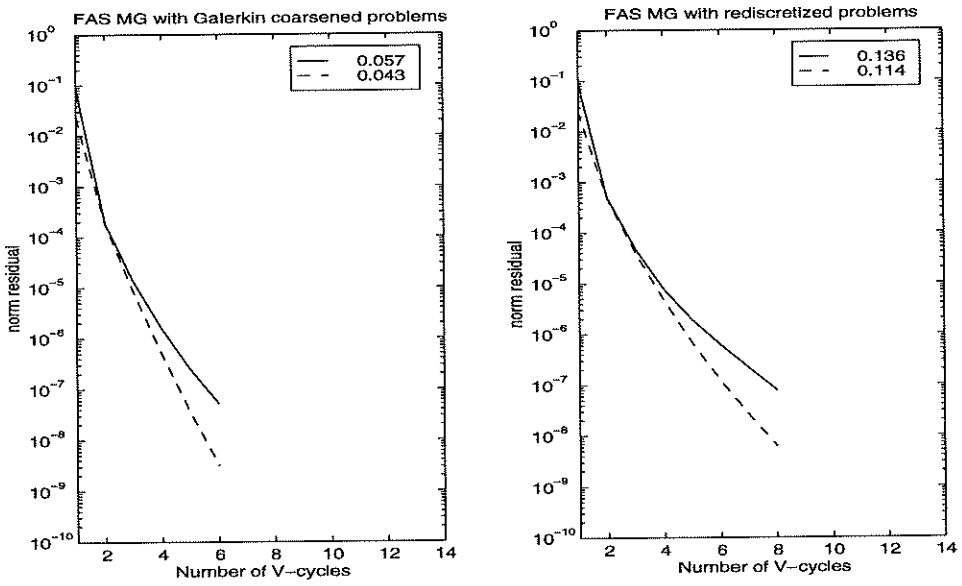


Figure 5.8: Unstructured annulus grid (unknowns 2175).

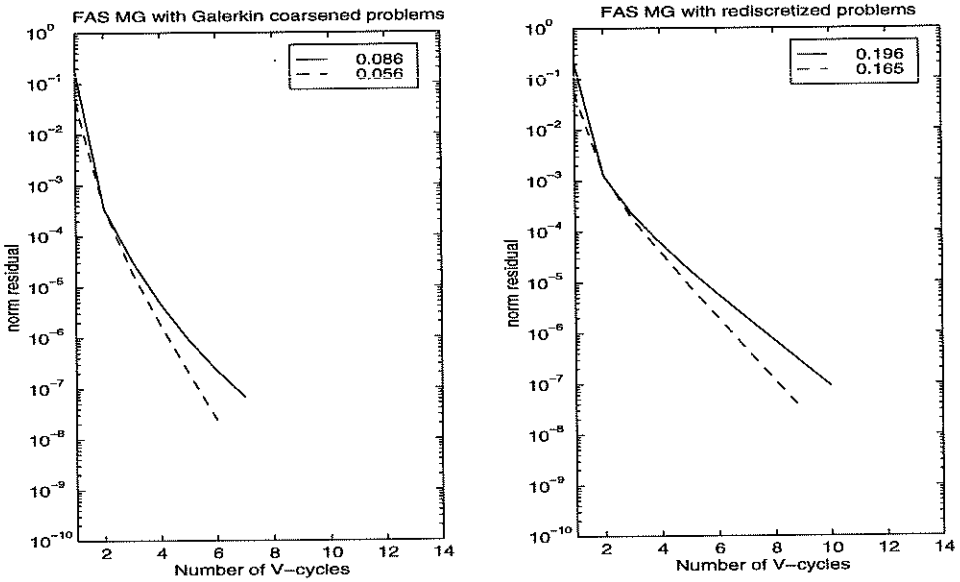


Figure 5.9: Unstructured annulus grid (unknowns 2430).

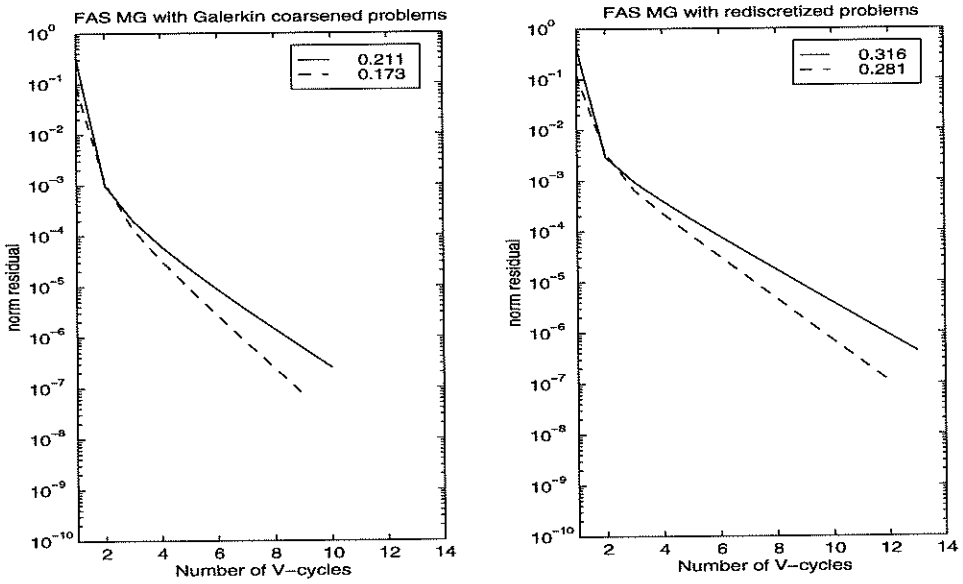


Figure 5.10: Unstructured 3-element airfoil grid (unknowns 1170).

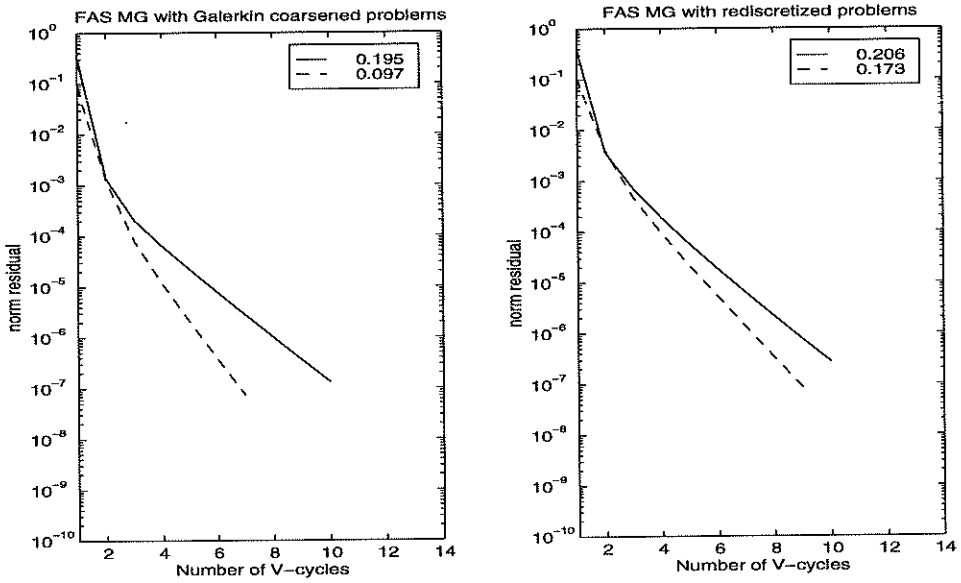


Figure 5.11: Unstructured 1-element airfoil grid (unknowns 1067).

made with the FAS MG partition. Options were chosen so they would coarsen to about 100 nodes and do spectral bisection at the coarse level. No local refinement was done after the partition was refined back to the fine level.

The various edges cut and times required for bisecting the airfoil are shown in Table 5.1. We should note that the current implementation was not optimized in any way and that the main objective was in achieving provable optimal multigrid performance, that is, obtaining a solution process which was grid-size independent.

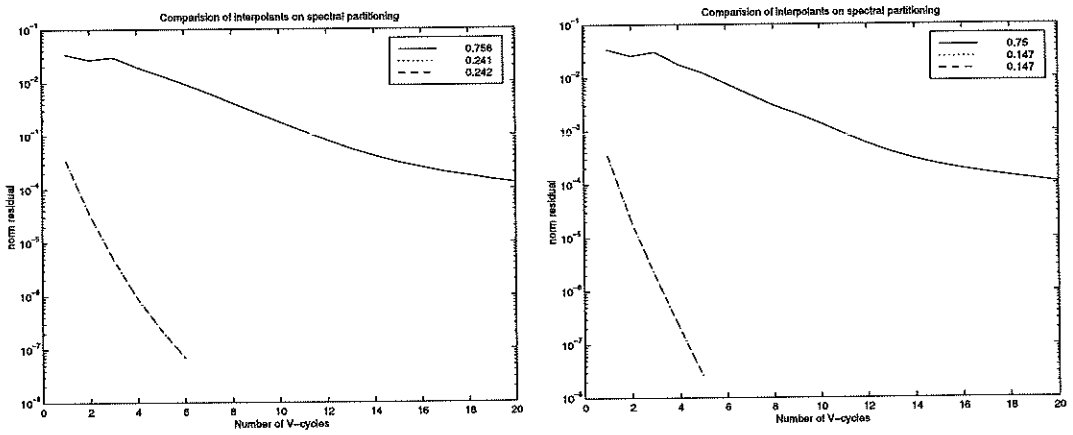


Figure 5.12: 3-level FAS MG for the spectral bisection problem on an unstructured annulus grid using different interpolants. The coarse problems are defined to be $Lu = \lambda u$ (left) $Lu = \lambda \tilde{I} u$ (right). Solid line is $\mathcal{I}_h^0$, dashed line is $\mathcal{I}_h^1$, and dash-dotted line is $\mathcal{I}_h^2$.

5.6 Concluding remarks

The main cost of the spectral method is in solving for the Fiedler vector (the lowest nontrivial eigenvector of the associated graph Laplacian). Attempts at accelerating the computation of the Fiedler vector by multilevel methods encounters the difficulty of adapting standard elliptic multilevel algorithms to solving the discrete graph Laplacian eigenvalue problem. Thus, for example, it is difficult to construct such algorithms that have convergence rate independent of the mesh size.

We have shown that when the graph is a standard finite element mesh, the graph Laplacian is spectrally equivalent (up to a diagonal scaling) to the mesh

Table 5.1: Performance of various bisection methods on the airfoil grid(4253 nodes).

Partitioner	# of Edges cut (%)	Time (sec)
Multilevel RSB	111 (0.90%)	2.99
Chaco	132 (1.07%)	12.34
Metis	122 (0.99%)	0.59
FAS MG	132 (1.07%)	45.63

Laplacian. This equivalence can be exploited to adapt recently developed multilevel elliptic algorithms for unstructured grids to solving the graph partitioning problem with a true multigrid convergence rate. Our current implementation, though slow, shows that optimal convergence rates can be achieved for the discrete problem.

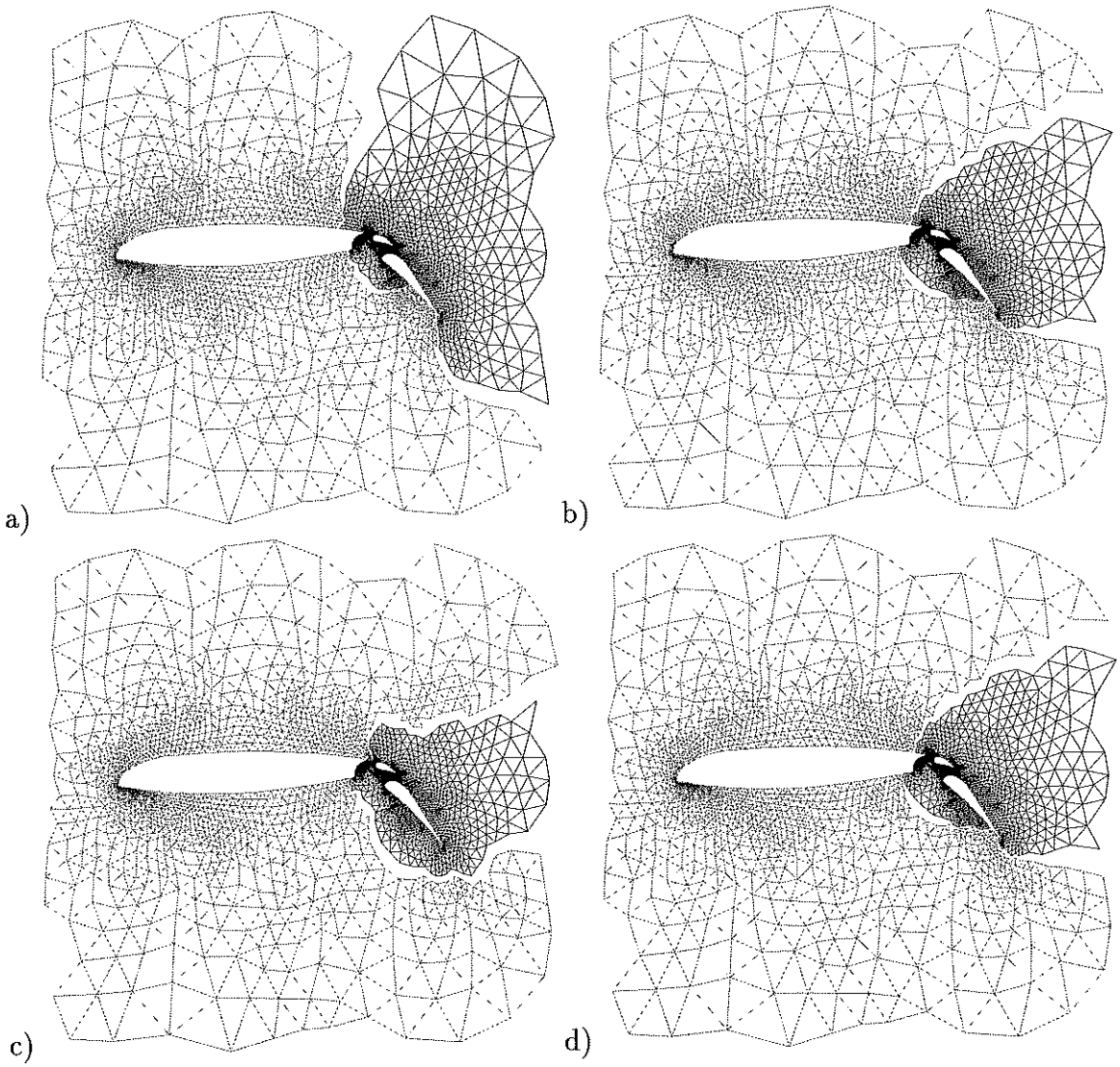


Figure 5.13: Bisection using (a) Multilevel RSB, (b) Chaco, (c) Metis, and (d) FAS MG partitioners.

APPENDIX A

Software

A.1 The Portable Extensible Toolkit for Scientific computation

The numerical results presented in this work were generated using the Portable Extensible Toolkit for Scientific computation (PETSc) software, developed by William Gropp and Barry Smith. This is an extensive, portable object-oriented library of software routines and data structures for both uni- and parallel processors. It is currently freely available on the Internet [27].

A.2 A Finite Element Flow Library for Unstructured Grids

An introduction to a library of finite element space discretizations for the fluid equations is presented in this section. These routines calculate the Jacobian matrix and the residual vector in Newton's method for solving compressible Euler flow and can be easily used in combination with the PETSc library of linear and non-linear solvers.

A.2.1 Compressible Euler equations

The two-dimensional compressible Euler equations, written in quasilinear form with conservative variables is

$$U_t + \frac{\partial f}{\partial U} U_x + \frac{\partial g}{\partial U} U_y = 0 \qquad in\ \Omega \tag{A.1}$$

where

$$u = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}, f = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{bmatrix}, \text{ and } g = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vH \end{bmatrix}.$$

We simplify the notation by writing the 4 x 4 Jacobian matrices of the fluxes as

$$A = \frac{\partial f}{\partial U} \text{ and } B = \frac{\partial g}{\partial U}. \tag{A.2}$$

Thus

$$U_t + AU_x + BU_y = 0 \tag{A.3}$$

For Euler flow, we impose boundary conditions only on the inflow boundaries:

$$U = G \quad \text{on } \Gamma^-$$

where

$$\Gamma^- = \{\vec{x} \in \Gamma : (A\ B) \cdot \hat{n} < 0, \hat{n} \text{ is the unit outward normal wrt } \Omega\}$$

In the next several sections, we discuss the spatial discretization of the compressible Euler equations. We note here that we are interested in steady-state calculations.

A.2.1.1 Galerkin method

The Galerkin finite element method with weakly imposed boundary conditions, for solving (A.3): Find $U^h \in V^h$ such that

$$B(W^h, U^h) = L(W^h) \quad \forall W^h \in V^h \tag{A.4}$$

where

$$B(W^h, U^h) = \int_\Omega W^h(AU_x + BU_y)d\Omega - \int_{\Gamma^-} W^h U d\Gamma \tag{A.5}$$

$$L(W^h) = -\int_{\Gamma^-} W^h G d\Gamma \tag{A.6}$$

where

$$V^h = \text{ space of piecewise polynomial functions}$$

A.2.1.2 Galerkin/Least squares method

To define a stabilized Galerkin method, we introduce the Galerkin/Least Squares method (GLS), which can be written: Find $U \in V^h$ such that

$$B_{GLS}(W^h, U^h) = L_{GLS}(W^h) \quad \forall W^h \in V^h \tag{A.7}$$

where

$$B_{GLS}(W^h, U^h) = B(W^h, U^h) \tag{A.8}$$

$$+ \int_\Omega (AW_x + BW_y)^T \cdot \tau \cdot (AU_x + BU_y) d\Omega \tag{A.9}$$

$$L_{GLS}(W^h) = L(W^h) \tag{A.10}$$

where τ is a 4 x 4 stabilization parameter. The exact definition of the least squares stabilization parameter, τ, will be discussed later.

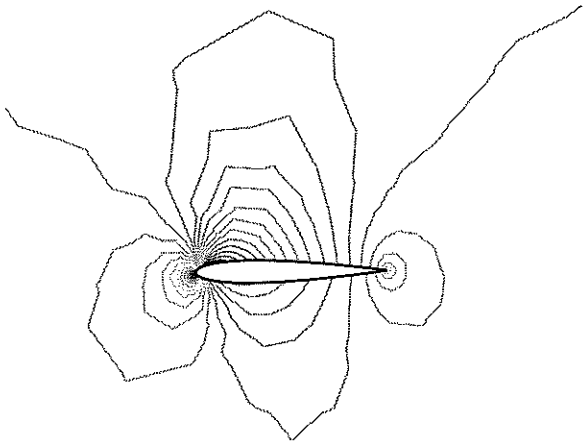


Figure 1.1: Pressure contours for Euler flow using GLS stabilized discretizations. Angle of attack 2, Mach number 0.63.

A.2.1.3 Discontinuous Galerkin method

In the discontinuous Galerkin method, the solution is allowed to be discontinuous across element boundaries. Because jumps may occur in the solution across element boundaries, element boundary terms are added to the standard Galerkin finite element formulation. The discontinuous Galerkin method is written

$$B_{DG}(W^h, U^h) = L(W^h) \quad \forall W^h \in V^h \tag{A.11}$$

where

$$B_{DG}(W^h, U^h) = B(W^h, U^h) + \sum_{T \in \mathcal{T}} \int_{\partial T} W^h((A\ B) \cdot \hat{n})^-[U] dl, \tag{A.12}$$

$$[U] = U_{exterior} - U_{interior} \tag{A.13}$$

Given a directed edge, we can make a distinction between the left and right element and in so doing, rewrite the discontinuous Galerkin method as the following edge-based method

$$\begin{aligned} B_{DG}(W^h, U^h) = & B(W^h, U^h) \\ & + \sum_{e \in \mathcal{E}} \int_e W^h((A\ B) \cdot \hat{n_L})^-(U_R - U_L) dl \\ & + \sum_{e \in \mathcal{E}} \int_e W^h((A\ B) \cdot \hat{n_R})^-(U_L - U_R) dl \end{aligned}$$

where $\hat{n_L}$ is the unit outward normal with respect to the left element, and $\hat{n_R}$ is the unit outward normal with respect to the right element (see Fig. 1.2).

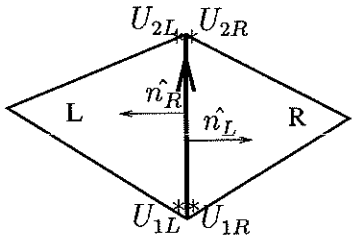


Figure 1.2: A directed edge with left and right elements L, R, outward normals $\hat{n_L}, \hat{n_R}$ and nodal values $U_{1L}, U_{2L}, U_{1R}, U_{2R}$.

A.2.2 Implementation

Consider a change of variables $U = U(Z)$ which symmetrizes A and B, so that the equations are now:

$$A(U(Z))\frac{\partial U}{\partial Z}Z_x + B(U(Z))\frac{\partial U}{\partial Z}Z_y = 0 \tag{A.14}$$

Let

$$\begin{aligned} \tilde{A}_0(Z) &= \frac{\partial U}{\partial Z} \\ \tilde{A}(Z) &= A(U(Z))\tilde{A}_0 \\ \tilde{B}(Z) &= B(U(Z))\tilde{A}_0 \end{aligned}$$

Then

$$\tilde{A}(Z)Z_x + \tilde{B}(Z)Z_y = 0 \tag{A.15}$$

A system of nonlinear equations is generated by multiplying with the weight functions, W^h. Define $R_i(U)$ such that

$$R_i(U) = B(W_i, U^h) - L(W_i) \quad i = 1, \ldots, M$$

To solve the system $U_t + R(U) = 0$, we use an unconditionally stable implicit discretization in time, that is, discretize the system with backward Euler:

$$\frac{U^{n+1} - U^n}{\Delta t} + R(U^{n+1}) = 0.$$

Linearizing the residual term $R(U^{n+1})$ about U^n yields

$$\frac{U^{n+1} - U^n}{\Delta t} + R(U^n) + \frac{\partial R}{\partial U}(U^n)(U^{n+1} - U^n) = 0.$$

Rearranging,

$$\left[\frac{1}{\Delta t} + \frac{\partial R}{\partial U}(U^n)\right] \Delta U = -R(U^n)$$

where $\Delta U = U^{n+1} - U^n$. The first term goes to zero as we approach steady state, $\Delta t \to \infty$, and we recover standard Newton's method. The construction of the linear system requires assembling the global Jacobian matrix, $\left[\frac{\partial R_i}{\partial U}(U^n)\right]$ and the global stiffness vector, $R_i(U^n)$. The algorithm would be implemented as follows:

Algorithm A.2.1

For each element, do

Get the indices of the vertices which make up the element

Get the coordinates and the values of the current iterate at these vertices

Calculate the element vector, $R_i(U^n)$

Calculate the element Jacobian matrix, $\left[\frac{\partial R_i}{\partial U}(U^n)\right]$

Assemble (accumulate) global RHS and Jacobian by scattering

the element vector and Jacobian matrix

end

This would be the general calling sequence for the Galerkin method. The Galerkin/Least Squares and Discontinuous Galerkin methods would add an extra call to Algorithm A.2.1, where the element vector and Jacobian are defined appropriately (in the discontinuous Galerkin case, we add an edge-based term to the standard Galerkin method, so we would loop over edges instead of elements). The definitions of of the Jacobians and stiffness vectors for each method will now be discussed in detail.

A.2.3 Construction of the Galerkin element Jacobian matrix and stiffness vector

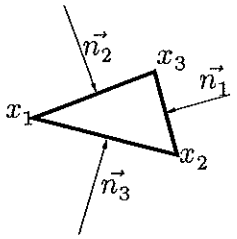


Figure 1.3: Triangular element with inward normal vectors scaled by the length of the corresponding side.

For a triangular element with indices $\{1,2,3\}$, and inward normal vectors $\{\vec{n_1},\vec{n_2},\vec{n_3}\}$, scaled by the length of the side (see Fig. 1.3), we will use the following identity which holds for linear functions, f:

$$\nabla f = \frac{1}{2\ meas(T)} \sum_{i=1}^{3} f_i \vec{n_i}. \tag{A.16}$$

where $f_i = f(x_i)$.

Proof: Since ∇f is constant for linear f, we have

$$\int_T \nabla f dA = \nabla f \int_T dA = \ meas(T) \nabla f. \tag{A.17}$$

By the Divergence theorem:

$$\int_T \nabla f dA = \int_{\partial T} f \hat{n} dl.$$

Using the facts that the scaled normals sum to zero ($\vec{n_1} + \vec{n_2} + \vec{n_3} = 0$), and that the line integral can be evaluated exactly for linear f gives

$$\int_{\partial T} f \hat{n} dl = -\frac{f_2 + f_3}{2}\vec{n_1} - \frac{f_3 + f_1}{2}\vec{n_2} - \frac{f_1 + f_2}{2}\vec{n_3} \tag{A.18}$$

$$= \frac{1}{2}\sum_{i=1}^{3} f_i \vec{n_i}. \tag{A.19}$$

Combining equations A.17 and A.19 yields the desired result.

A.2.3.1 Element stiffness vector

Assume the basis functions, W^h, and the unknowns, U, are piecewise linear on a triangular element and define K_i to be the 4 x 4 matrix defined by

$$K_i \equiv \frac{1}{2}[\bar{A}, \bar{B}] \cdot \vec{n_i} \quad i = 1, 2, 3$$

for some mean valued vector field $[\bar{A}, \bar{B}]$. Then using identity A.16, we see that the i^{th} component of the 3 x 1 block element stiffness vector is given by

$$\begin{aligned} R_i(U) &= B(W_i, U) \\ &= \int_T W_i(\bar{A}U_x + \bar{B}U_y)dA \\ &= [\bar{A}, \bar{B}] \cdot \nabla U \int_T W_i dA \\ &= \frac{1}{2\ meas(T)} \sum_{j=1}^{3} [\bar{A}, \bar{B}] \cdot \vec{n_j} U_j \int_T W_i dA \\ &= \frac{1}{meas(T)} \sum_{j=1}^{3} K_j U_j \int_T W_i dA \\ &= \frac{1}{3} \sum_{j=1}^{3} K_j U_j \end{aligned}$$

Here, we have used $\int_T W_i dA = \bar{W}_i \int_T dA = \frac{1}{3}\ meas(T)$, since the average of a basis function is $\bar{W}_i = 1/3$. In matrix notation the complete 3 x 1 element vector is written

$$R(U) = \frac{1}{3} \begin{bmatrix} K_1 & K_2 & K_3 \\ K_1 & K_2 & K_3 \\ K_1 & K_2 & K_3 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix}$$

A.2.3.2 Element Jacobian matrix

The element Jacobian matrix is a block 3 x 3 matrix where the ij^{th} block is the 4 x 4 matrix

$$\begin{aligned}\left[\frac{\partial R}{\partial U}\right]_{ij} &= \left(\frac{\partial R_i}{\partial U_j}\right) \\ &= \frac{\partial}{\partial U_j}\left(\frac{1}{3}K_l U_l\right) \\ &= \frac{1}{3}\left(\frac{\partial K_l}{\partial U_j}U_l + K_l\frac{\partial U_l}{\partial U_j}\right)\end{aligned}$$

Note that $\frac{\partial K_l}{\partial U_j}$ is a 4 x 4 x 4 matrix.

A.2.4 Construction of the least squares stabilization element Jacobian matrix and stiffness vector

A.2.4.1 Element vector

The least squares stabilization method adds to the standard Galerkin method, a block 3 x 1 term for each element, where the i^{th} block is of the form

$$\begin{aligned}RGLS_i &= \int_T (\bar{A}W_{ix} + \bar{B}W_{iy})^T \cdot \tau \cdot (\bar{A}U_x + \bar{B}U_y)dA \\ &= K_i\tau\sum_{j=1}^{3} K_j U_j \int_T dA \\ &= \frac{1}{2}K_i\tau K_j U_j\end{aligned}$$

There have been many alternatives for defining the stabilization parameter. One way to define the 4 x 4 least squares stabilization parameter, τ, is:

$$\tau = meas(T)\left[\sum_{i=1}^{3}|K_i|^p\right]^{1/p}, \quad p = 1,2 \tag{A.20}$$

where $|K_i|$ is the componentwise absolute value of K_i.

In matrix notation the complete 3 x 1 element vector is written

$$R(U) = \frac{1}{2}\begin{bmatrix} K_1\tau K_1 & K_1\tau K_2 & K_1\tau K_3 \\ K_2\tau K_1 & K_2\tau K_2 & K_2\tau K_3 \\ K_3\tau K_1 & K_3\tau K_2 & K_3\tau K_3 \end{bmatrix}\begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix}$$

A.2.4.2 Element Jacobian matrix

The element Jacobian matrix is a block 3 x 3 matrix where the ij^{th} block is the 4 x 4 matrix

$$\begin{aligned}\left[\frac{\partial RGLS}{\partial U}\right]_{ij} &= \left(\frac{\partial RGLS_i}{\partial U_j}\right) \\ &= \frac{\partial}{\partial U_j}\left(\frac{1}{2}K_i\tau K_k U_k\right) \\ &= \frac{1}{2}\left[\frac{\partial K_i}{\partial U_j}\tau K_k U_k + K_i\frac{\partial \tau}{\partial U_j}K_k U_k + K_i\tau\frac{\partial K_k}{\partial U_j}U_k + K_i\tau K_k\frac{\partial U_k}{\partial U_j}\right]\end{aligned}$$

A.2.5 Construction of the discontinuous Galerkin element Jacobian matrix and stiffness vector

A.2.5.1 Element vector

The discontinuous Galerkin method adds to the standard Galerkin method, a block 4 x 1 term for each interior edge, where the i^{th} block is of the form

$$\begin{aligned} RDG_{L_i} &= \int_e (W_i(\bar{A}\ \bar{B}) \cdot \hat{n_L})^-(U_R - U_L)dl \\ &= ((\bar{A}\ \bar{B}) \cdot \hat{n_L})^- \left(\frac{h}{4}\right) \sum_{j=1}^{2} (U_{Rj} - U_{Lj}) \end{aligned}$$

or

$$\begin{aligned} RDG_{R_i} &= \int_e (W_i(\bar{A}\ \bar{B}) \cdot \hat{n_R})^-(U_L - U_R)dl \\ &= ((\bar{A}\ \bar{B}) \cdot \hat{n_R})^- \left(\frac{h}{4}\right) \sum_{j=1}^{2} (U_{Lj} - U_{Rj}) \end{aligned}$$

where $h = ||e||_2$.

Define $\vec{n_L}$ to be the scaled outward normal with respect to the left element, and $\vec{n_R}$ be the scaled outward normal with respect to the right element, and define K_L and K_R as

$$\begin{aligned} K_L &= \frac{h}{4}((\bar{A}\ \bar{B}) \cdot \hat{n_L})^- = \frac{1}{4}((\bar{A}\ \bar{B}) \cdot \vec{n_L})^- \\ K_R &= \frac{h}{4}((\bar{A}\ \bar{B}) \cdot \hat{n_R})^- = \frac{1}{4}((\bar{A}\ \bar{B}) \cdot \vec{n_R})^- = -\frac{1}{4}((\bar{A}\ \bar{B}) \cdot \vec{n_L})^+ \end{aligned}$$

In matrix notation the complete 4 x 1 element vector is written

$$RDG(U) = \begin{bmatrix} -K_L & -K_L & K_L & K_L \\ -K_L & -K_L & K_L & K_L \\ K_R & K_R & -K_R & -K_R \\ K_R & K_R & -K_R & -K_R \end{bmatrix} \begin{bmatrix} U_{L1} \\ U_{L2} \\ U_{R1} \\ U_{R2} \end{bmatrix} \tag{A.21}$$

A.2.5.2 Element Jacobian matrix

The element Jacobian matrix for the discontinuous Galerkin method is a block 4 x 4 matrix where the ij^{th} block is the 4 x 4 matrix

$$\left[\frac{\partial RDG_L}{\partial U}\right]_{ij} = \left(\frac{\partial RDG_{Li}}{\partial U_j}\right) = \frac{\partial}{\partial U_j}\left(K_L \sum_{j=1}^{2}(U_{Rj} - U_{Lj})\right)$$
$$= \frac{\partial K_L}{\partial U_j}\sum_{j=1}^{2}(U_{Rj} - U_{Lj}) + K_L \sum_{j=1}^{2}\frac{\partial}{\partial U_j}(U_{Rj} - U_{Lj})$$

or

$$\left[\frac{\partial RDG_R}{\partial U}\right]_{ij} = \left(\frac{\partial RDG_{Ri}}{\partial U_j}\right) = \frac{\partial}{\partial U_j}\left(K_R \sum_{j=1}^{2}(U_{Lj} - U_{Rj})\right)$$
$$= \frac{\partial K_R}{\partial U_j}\sum_{j=1}^{2}(U_{Lj} - U_{Rj}) + K_R \sum_{j=1}^{2}\frac{\partial}{\partial U_j}(U_{Lj} - U_{Rj})$$

Bibliography

[1] R. Bank and J. Xu. An algorithm for coarsening unstructured meshes. *Numer. Math.*, 73:1–23, 1996.

[2] J. Bramble. *Multigrid methods.* Pitman, Notes on Mathematics, 1994.

[3] James H. Bramble, Joseph E. Pasciak, Junping Wang, and Jinchao Xu. Convergence estimates for multigrid algorithms without regularity assumptions. *Math. Comp.*, 57(195):23–45, 1991.

[4] A. Brandt. Multigrid techniques with applications to fluid dynamics: 1984 guide. *VKI Lecture Series*, pages 1–176, 1984.

[5] A. Brandt, S. McCormick, and J. Ruge. Multigrid methods for differential eigenproblems. *SIAM J. Sci. Stat. Comput.*, 4(2):244–260, 1983.

[6] S. Brenner and L.R. Scott. *The mathematical theory of finite element methods.* Springer-Verlag, 1994.

[7] William Briggs. *A multigrid tutorial.* SIAM Book, Philadelphia, 1987.

[8] Xiao-Chuan Cai. The use of pointwise interpolation in domain decomposition methods with nonnested meshes. *SIAM J. Sci. Comp.*, 16(1), 1995.

[9] Xiao-Chuan Cai and Youcef Saad. Overlapping domain decomposition algorithms for general sparse matrices. *Numer. Lin. Alg. Appls.*, 3(3):221–237, 1996.

[10] J. C. Cavendish. Automatic triangulation of arbitrary planar domains for the finite element method. *Int. J. Numer. Meth. Eng.*, 8:679–696, 1974.

[11] T. F. Chan, S. Go, and L. Zikatanov. Lecture notes on multilevel methods for elliptic problems on unstructured grids. In *28th Computational Fluid Dynamics*, volume LS 1997-02 of *Lecture Series at the von Karman Institute for Fluid Dynamics*, March 1997.

[12] T. F. Chan, S. Go, and L. Zikatanov. Multilevel methods for elliptic problems on unstructured grids. In M. Hafez, editor, *1997 Computational Fluid Dynamics*, 1997.

[13] T. F. Chan, S. Go, and J. Zou. Boundary treatments of multilevel methods on unstructured meshes. Technical Report CAM 96–30, Department of Mathematics, University of California at Los Angeles, September 1996.

[14] T. F. Chan, S. Go, and J. Zou. Multilevel domain decomposition and multigrid methods for unstructured meshes: Algorithms and theory. In R. Glowinski, J. Périaux, Z.-C. Shi, and O. Widlund, editors, *Domain Decomposition Methods in Science and Engineering*, pages 159–176. John Wiley and Sons, 1997.

[15] T. F. Chan and T. Mathew. Domain decomposition algorithms. *Acta Numerica*, pages 61–143, 1994.

[16] T. F. Chan and B. Smith. Domain decomposition and multigrid methods for elliptic problems on unstructured meshes. *Elec. Trans. Numer. Anal.*, 2:171–182, 1994.

[17] T. F. Chan, B. Smith, and J. Zou. Overlapping Schwarz methods on unstructured meshes using non-matching coarse grids. *Numer. Math.*, 73(2):149–167, April 1996.

[18] T. F. Chan and J. Zou. Additive Schwarz domain decomposition methods for elliptic problems on unstructured meshes. *Numer. Algorithms*, 8:329–346, 1994.

[19] T. F. Chan and J. Zou. A convergence theory of multilevel additive Schwarz methods on unstructured meshes. *Numer. Algorithms*, 13:365–398, 1996.

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

[21] P. Ciarlet. *The Finite Element Method for Elliptic Problems.* North-Holland, 1978.

[22] P. Clément. Approximation by finite element functions using local regularization. *R.A.I.R.O. Numer. Anal.*, R-2:77–84, 1975.

[23] H. Deconinck and Tim Barth, editors. *Special course on unstructured grid methods for advection dominated flows*, March 1992. AGARD REPORT 787, Special course at the VKI, Belgium.

[24] M. Dryja and O. Widlund. An additive variant of the Schwarz alternating method for the case of many subregions. Technical Report 339, also Ultracomputer Note 131, Department of Computer Science, Courant Institute, 1987.

[25] Maksymilian Dryja and Olof B. Widlund. Some domain decomposition algorithms for elliptic problems. In Linda Hayes and David Kincaid, editors, *Iterative Methods for Large Linear Systems*, pages 273–291, San Diego, California, 1989. Academic Press.

[26] Maksymilian Dryja and Olof B. Widlund. Additive Schwarz methods for elliptic finite element problems in three dimensions. In Tony F. Chan, David E. Keyes, Gérard A. Meurant, Jeffrey S. Scroggs, and Robert G. Voigt, editors, *Fifth Conference on Domain Decomposition Methods for Partial Differential Equations*, Philadelphia, PA, 1992. SIAM.

[27] William D. Gropp and Barry F. Smith. Portable Extensible Toolkit for Scientific Computation (PETSc). Available by anonymous ftp at `info.mcs.anl.gov` in the directory `pub/pdetools` or at `http://www.mcs.anl.gov/Projects/petsc/petsc.html`.

[28] H. Guillard. Node-nested multi-grid method with Delaunay coarsening. Technical Report RR-1898, INRIA, Sophia Antipolis, France, March 1993.

[29] W. Hackbusch. *Multi-grid methods and applications.* Springer Verlag, New York, 1985.

[30] B. Hendrickson and R. Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. Technical Report SAND94-1460, Sandia Nat'l Lab., Albuquerque, N.M., Sep. 1992.

[31] B. Hendrickson and R. Leland. The Chaco User's Guide Version 2.0. Technical Report SAND94-2692, Sandia Natl. Lab., Albuquerque, N.M., July. 1995.

[32] Claes Johnson. *Numerical solution of partial differential equations by the finite element method.* Cambridge University Press, Cambridge, 1990.

[33] P. Ciarlet Jr. and F. Lamour. On the validity of a front-oriented approach to partitioning large sparse graphs with a connectivity constraint. Technical Report 94-37, University of California at Los Angeles, December 1994.

[34] P. Ciarlet Jr. and F. Lamour. Spectral partitioning methods and greedy partitioning methods: A comparison on finite element graphs. Technical Report 94-9, University of California at Los Angeles, April 1994.

[35] P. Ciarlet Jr., F. Lamour, and B.F. Smith. On the influence of the partitioning schemes on the efficiency of overlapping domain decomposition methods.

Technical Report 94-23, University of California at Los Angeles, July 1994.

[36] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report TR 95-035, University of Minnesota, 1995.

[37] G. Karypis and V. Kumar. Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, 1997. available at http://www-users.cs.umn.edu/ karypis/metis/metis.html.

[38] B. Koobus, M. H. Lallemand, and A. Dervieux. Unstructured volume-agglomeration MG: solution of the Poisson equation. *Int. J. Numer. Meth. Fluids*, 18(1):27–42, 1994.

[39] R. Kornhuber and H. Yserentant. Multilevel methods for elliptic problems on domains not resolved by the coarse grid. In David Keyes and Jinchao Xu, editors, *Domain Decomposition Methods in Science and Engineering*, pages 49–60. AMS, Providence, 1994.

[40] P. Lions. On the Schwarz alternating method. I. In Roland Glowinski, Gene H. Golub, Gérard A. Meurant, and Jacques Périaux, editors, *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, 1988.

[41] A. M. Matsokin and S. V. Nepomnyaschikh. A Schwarz alternating method in a subspace. *Soviet Mathematics*, 29(10):78–84, 1985.

[42] D. J. Mavriplis and V. Venkatakrishnan. Agglomeration multigrid for two-dimensional viscous flows. *Comput. and Fluids*, 24(5):553–570, 1995.

[43] D.J. Mavriplis. Unstructured mesh algorithms for aerodynamic calculations. Technical Report 92-35, ICASE, NASA Langley, Virginia, July 1992.

[44] Stephen F. McCormick. *Multilevel Adaptive Methods for Partial Differential Equations.* SIAM, Philadelphia, PA, 1989.

[45] C. F. Ollivier-Gooch. Multigrid acceleration of an upwind Euler solver on unstructured meshes. *AIAA Journal*, 33(10):1822–1827, 1995.

[46] A. Pothen. Graph partitioning algorithms with applications to scientific computing. In D. E. Keyes, A. H. Sameh, and V. Venkatakrishnan, editors, *In Parallel Numerical Algorithms.* Kluwer Academic Press, 1996.

[47] A. Pothen, H. Simon, and K.P. Liou. Partitioning sparse matrices with eigenvector of graphs. *SIAM J. Mat. Anal. Appl.*, 11(3):430–452, 1990.

[48] B. Smith, P. Bjørstad, and W. Gropp. *Domain decomposition: parallel multilevel methods for elliptic partial differential equations.* Cambridge University Press, Cambridge, 1996.

[49] G. Strang and G. Fix. *An Analysis of the Finite Element Method.* Prentice Hall, 1973.

[50] P. Wesseling. Introduction to multigrid methods. Technical Report 95-11, ICASE, NASA Langley Research Center, February 1995.

[51] R. Williams. Performance of dynamic load balancing algorithms for unstructured mesh calculations. *Concurrency: Practice and Experience*, 3:457–481, 1991.

[52] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM Rev.*, 34:581–613, December 1992.

[53] J. Xu. *An introduction to multilevel methods.* To be published by Oxford University Press, 1996. Lecture notes: VIIth EPSRC Numerical analysis summer school, Leicester University, UK.