Improving Coarsening and Interpolation for Algebraic Multigrid

by

Jeffrey S. Butler

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Applied Mathematics

Waterloo, Ontario, Canada, 2006

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Algebraic multigrid (AMG) is one of the most efficient algorithms for solving large sparse linear systems on unstructured grids. Classical coarsening schemes such as the standard Ruge-Stüben method [14] can lead to adverse effects on computation time and memory usage that affect scalability. Memory and execution time complexity growth is remedied for various large three-dimensional problems using the parallel modified independent set (PMIS) coarsening strategy developed by De Sterck, Yang, and Heys [18]. However, this coarsening strategy often leads to erratic grids without a regular structure that diminish convergence.

This thesis looks at two modifications of the PMIS algorithm that aim to improve scalability. These include a greedy implementation of PMIS and restricting PMIS coarsening to finer grid levels while Cleary-Jones-Luby-Plassman coarsening (based on the standard Ruge-Stüben method) is performed on all other grids. It is shown that, for a variety of problems, the greedy PMIS algorithm does little to improve convergence, while the second modification can improve convergence. However, it is also shown that the second modification can result in increased memory usage that is unfavorable to scalability.

The PMIS based algorithm can be improved by redefining interpolation. As shown by De Sterck and Yang [17], PMIS coarsening combined with *F-F* interpolation dramatically improves convergence, but often has negative effects on computation time per iteration and memory usage that affect scalability. A third modification is proposed that aims to remedy this problem by altering *F-F* interpolation. The new algorithm is called *F-F1* interpolation, and is shown to reduce computation time per iteration and memory usage compared to *F-F* interpolation, while maintaining fast convergence and good scalability, for a variety of problems.

Acknowledgements

First and foremost, I would like to thank my advisor Hans De Sterck for all of his help and support. It has truly been a pleasure to work with you, and I am grateful for all that you have taught me. Thank you for introducing me to AMG, for being a great resource on this fascinating subject, and for all of the financial support that made its study possible.

Thank you to the Center for Applied Scientific Computing and Lawrence Livermore National Laboratories (LLNL) in California for their financial support of my research, and for their provision of Hypre. In particular, I would like like to thank Ulrike Yang at LLNL for all of her ideas and insights on AMG and with regard to the work in this thesis.

Thank you to the Applied Mathematics Department for all of their financial support, and to the Waterloo Graduate Studies Office for their travel support in my attending the 2006 Copper Mountain Conference on iterative methods.

To the entire Applied Mathematics Department – all faculty, staff, and students – thank you for making Waterloo a great place to study, and for fostering an atmosphere of excellence.

Finally, thank you Mom, Dad, and Jordan for all of your love and support. Without you I would not be where I am today.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Although the field of numerical analysis dates back more than 2000 years, it has vastly grown in recent years due to the advent of the computer. The demands of modern scientific computation center on speed of execution and data-storage cost. Considerable research is being conducted around the world into algorithms that effectively solve problems while keeping both speed and storage at optimal levels.

The need for robust and efficient solution methods manifests itself in a range of applications: from engineering and atmospheric science, to finance and biological systems. Often, problems in these areas are inherently large by nature, or may require a high level of resolution. In either case, the number of unknowns can be in the billions, thus making the need for efficient solution strategies all the more evident. For many linear algebra problems, direct solution methods are impractical, and standard iterative methods are slow to converge. In 1964, Fedorenko introduced the first instance of a class of algorithms that could remedy these shortcomings and would come to be known as multigrid methods [6]. Brandt then introduced the first practical multigrid method in 1977 [2]. Multigrid methods have been very successful at solving a variety of problems, and a substantial part of this success has been due to a particular versatile form known as algebraic multigrid (AMG). This thesis will examine AMG and its range of applicability, and also present several modifications to current implementations that improve both execution speed and storage characteristics.

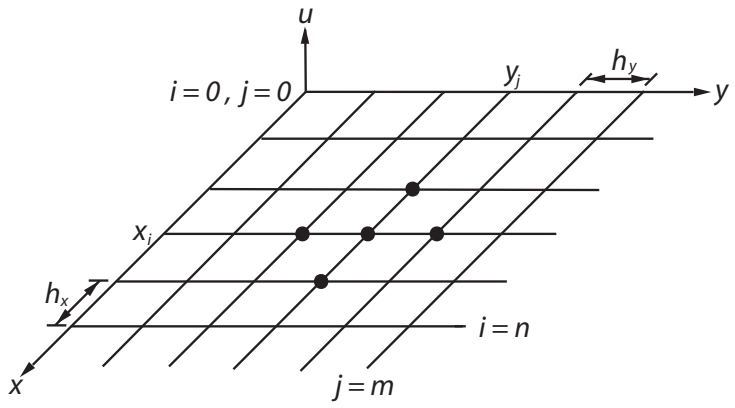

Figure 1.1: Discretized two-dimensional grid. The dots indicate points that appear in the discrete equation for point (i, j).

1.1 Preliminary Background: Finite Difference Discretizations

As described by Atkinson [1], numerical analysis provides computational methods for the study and solution of mathematical problems. Of interest to this thesis, is the numerical solution of partial differential equations (PDEs). For example, consider the standard two-dimensional (2D) Laplace equation with Dirichlet boundary conditions on the unit square,

$$-u_{xx} - u_{yy} = f(x,y), \qquad 0 < x < 1, \qquad 0 < y < 1, \tag{1.1.1}$$
$$u(0,y) = u(x,0) = 0.$$

This equation can be discretized by first discretizing the domain and then applying a Taylor series expansion at each of the grid points (as is done, for instance, in [4]). An example of a discretized domain is illustrated in Figure 1.1. The labeling indicates that the grid spacing in the x and y directions, respectively, can be defined as

$$h_x = \frac{1}{n}, \ where\ i = 0, 1, \ldots, n,$$
$$h_y = \frac{1}{m}, \ where\ j = 0, 1, \ldots, m,$$

such that $x_i = ih_x$ and $y_j = jh_y$. Then, applying a Taylor series in two variables at all grid points (i, j),[1]

$$u(x,y) \;=\; \sum_{k=0}^{\infty}\sum_{l=0}^{\infty} \frac{u^{(k,l)}(x_i,y_j)}{k!l!}(x-x_i)^k(y-y_j)^l, \tag{1.1.2}$$

and taking appropriate combinations, yields expressions for the second order x and y derivatives, respectively, as follows:

$$u_{xx}(x_i,y_j) \;\approx\; \frac{u(x_{i+1},y_j) - 2u(x_i,y_j) + u(x_{i-1},y_j)}{h_x^2}, \tag{1.1.3}$$

$$u_{yy}(x_i,y_j) \;\approx\; \frac{u(x_i,y_{j+1}) - 2u(x_i,y_j) + u(x_i,y_{j-1})}{h_y^2}. \tag{1.1.4}$$

When u_{xx} and u_{yy} in (1.1.1) are replaced by (1.1.3) and (1.1.4), respectively, the resulting discretization is referred to as a *5-point Laplace discretization*, since approximation of the x and y second derivatives involves five different grid points. Assuming that spacing in the x and y directions is equal (i.e. $h_x = h_y = h$, so that $n = m$), and letting $u_{i,j}$ be an approximation for $u(x_i, y_j)$, this results in a system of $(n-1)^2$ equations in $(n-1)^2$ unknowns (recall that solution values at the boundary are 0):

$$\begin{aligned}
\frac{4}{h^2}u_{1,1} - \frac{1}{h^2}u_{2,1} - \frac{1}{h^2}u_{1,2} &= f_{1,1}\\
&\vdots\\
-\frac{1}{h^2}u_{i+1,j} - \frac{1}{h^2}u_{i-1,j} + \frac{4}{h^2}u_{i,j} - \frac{1}{h^2}u_{i,j+1} - \frac{1}{h^2}u_{i,j-1} &= f_{i,j}\\
&\vdots\\
-\frac{1}{h^2}u_{n-1,n-2} - \frac{1}{h^2}u_{n-2,n-1} + \frac{4}{h^2}u_{n-1,n-1} &= f_{n-1,n-1}.
\end{aligned}$$

[1] $u^{(k,l)}(x_i, y_j)$ represents taking the k^{th} partial derivative in x and the l^{th} partial derivative in y of $u(x_i, y_j)$.

This system may be written in matrix form as

$$A\mathbf{u} = \mathbf{f}, \tag{1.1.5}$$

where entry a_{ij} in row i and column j of matrix A is the same as the coefficient of $u_{i,j}$ in equation i of the set of discretized equations. For example, in the $n = 4$ case,

$$A = \frac{1}{h^2}\begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix}, \ \mathbf{u} = \begin{bmatrix} u_{1,1} \\ u_{2,1} \\ u_{3,1} \\ u_{1,2} \\ u_{2,2} \\ u_{3,2} \\ u_{1,3} \\ u_{2,3} \\ u_{3,3} \end{bmatrix}, \ \mathbf{f} = \begin{bmatrix} f_{1,1} \\ f_{2,1} \\ f_{3,1} \\ f_{1,2} \\ f_{2,2} \\ f_{3,2} \\ f_{1,3} \\ f_{2,3} \\ f_{3,3} \end{bmatrix}.$$

Note that matrix A is sparse, as it contains a relatively small number of nonzero elements.

The linear algebraic system (1.1.5) was obtained using what is known as a *finite difference discretization*, and this is a very effective method for this problem. However, for other problems, finite element discretizations (see for example [16]), finite volume discretizations (see for example [12]), or other methods can be more suitable, and may still lead to a sparse system in the form (1.1.5). While considerable research focuses on obtaining effective discretizations, that is not the focus of this thesis. This thesis is concerned with the efficient solution of sparse linear algebraic systems of the form (1.1.5).

1.2 Outline

This thesis presents the standard algebraic multigrid algorithm, and then looks at three modifications of current components of the algorithm. In order to provide some

necessary background material, several standard iterative methods and an exact solution technique are first examined in Chapter 2. These methods include Gaussian Elimination, the standard Jacobi and Gauss-Seidel iterative processes, and the conjugate gradient and generalized minimal residual methods. For the Jacobi and Gauss-Seidel methods, a convergence and spectral analysis of error reduction capabilities is performed. Computational work requirements of the methods are also described.

Chapter 3 introduces the basic multigrid algorithm. All theory presented in this chapter is applicable in the algebraic multigrid context, and is meant to be a precursor to Chapter 4 where algebraic multigrid is described.

Chapter 4 presents the algebraic multigrid algorithm including all components relevant to the new developments introduced in this thesis. This includes the standard Ruge-Stüben (RS) coarsening algorithm [14], the parallel modified independent set (PMIS) coarsening algorithm introduced by De Sterck, Yang, and Heys [18], the Cleary-Jones-Luby-Plassman (CLJP) parallel coarsening algorithm [10], and *F*-*F* interpolation as defined by De Sterck and Yang [17].

Chapter 5 presents three proposed changes to current AMG components. These include a greedy version of the PMIS coarsening algorithm, restricting PMIS coarsening to finer grid levels followed by CLJP coarsening on coarser grid levels, and a modification of *F*-*F* interpolation that only considers one *distance-two* *C*-point in the interpolation formula for strong *F*-*F* connections without a common *C*-point. A variety of test problems are described in Chapter 6, and results along with a discussion of their significance are presented in Chapter 7.

The thesis finishes with concluding remarks and a discussion of future work in Chapter 8. An appendix containing algebra definitions that are used in the body of the text is also included.

Chapter 2

Standard Numerical Methods for Linear Algebraic Systems

This chapter examines several methods that have historically been used to solve linear systems of the form (1.1.5). These include Gaussian elimination, the Jacobi and Gauss-Seidel (GS) iterative procedures, and the conjugate gradient (CG) and generalized minimal residual (GMRES) methods. It should be noted that although these methods may perform well for small problem sizes, they may in fact be prohibitively computationally expensive for large ones. However, it is still useful to proceed with this examination – not only because it will be helpful by providing a framework for error analysis, but also because these methods do in fact play important roles within the multigrid algorithm, either as error smoothers, or as convergence accelerators.[1] This chapter will first look at the traditional exact solution technique, Gaussian elimination, and then proceed to discuss the aforementioned iterative methods.

2.1 Gaussian Elimination

Gaussian elimination is one of the cornerstones of linear algebra and is a historically significant method for exactly solving systems of equations of the form (1.1.5). The basic algorithm will not be presented here as it may be found in any standard textbook on linear algebra, for example [22].

[1] The conjugate gradient and generalized minimal residual methods can be used as accelerators for the multigrid algorithm.

In its most basic form, for a d-dimensional problem with n points in each dimension, Gaussian elimination requires $O\left(n^{3d}\right)$ operations. The reader may recognize that this is true since $O(n^d)$ columns must be treated by multiplication and subtraction for $O(n^d)$ rows, and all this must be done for $O(n^d)$ pivot elements.

The fact that Gaussian elimination requires $O\left(n^{3d}\right)$ operations makes it computationally inefficient for large n. However, it may be a reasonable choice when an exact solution technique is desired for a small problem, or when the matrix problem is not sparse. Gaussian elimination may be used in the multigrid algorithm for direct solution of coarse-grid problems. How this is done will be clarified in Chapter 3.

2.2 Relaxation Methods

This section introduces two iterative methods that are commonly referred to as relaxation methods, or smoothing methods. These include the Jacobi (or simultaneous displacement) method and the Gauss-Seidel (GS) method.[2] This section follows the treatment of [4], and will introduce the methods, perform a convergence analysis common to both methods, and then conclude with a spectral error analysis of the GS method. First, however, it is necessary to introduce the components needed for a standard error analysis of iterative methods. For simplicity, this is done in the context of the one-dimensional (1D) version of the model problem from Section 1.1. That is,

$$\begin{aligned} -u_{xx} &= f(x), \qquad 0 < x < 1, \\ u(0) &= u(1) = 0, \end{aligned} \tag{2.2.1}$$

on the domain illustrated in Figure 2.1, with grid spacing

$$h \;=\; \frac{1}{n}\;, \text{ where } i = 0, 1, \ldots, n.$$

[2]The Jacobi and Gauss-Seidel methods are referred to as *relaxation methods* when they are used for the purpose of reducing oscillatory error components. What is meant by "reduction of oscillatory error components" will be clarified in Section 2.2.4.

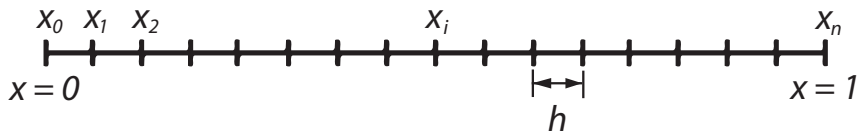

Figure 2.1: Discretized one-dimensional grid.

The resulting finite difference discretization is

$$\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} = f_i \ , \quad i = 1, 2, \ldots, n-1 \ , \quad u_0 = u_n = 0. \tag{2.2.2}$$

System (2.2.2) can then be written in the form (1.1.5). The error analysis can now be formulated.

Definition 2.2.1. Let $\mathbf{v}$ be an approximation to the exact solution of the discrete problem (1.1.5), $\mathbf{u}$, such that $v_i \approx u_i$.

Definition 2.2.2. Let the *error* in the approximation be defined as $\mathbf{e} \equiv \mathbf{u} - \mathbf{v}$.

Definition 2.2.3. Let the *residual* be defined as $\mathbf{r} \equiv \mathbf{f} - A\mathbf{v}$.

Since $\mathbf{u}$ is assumed to be unknown, it can be noted that $\mathbf{e}$ is just as inaccessible as $\mathbf{u}$, but that $\mathbf{r}$ does provide an easily computable measure of how well $\mathbf{v}$ approximates $\mathbf{u}$. Using Definitions 2.2.2 and 2.2.3, the following equations are obtained:

$$A\mathbf{e} = \mathbf{r}, \tag{2.2.3}$$

$$\mathbf{u} = \mathbf{v} + \mathbf{e}. \tag{2.2.4}$$

Equations (2.2.3) and (2.2.4) are known as the *residual equation* and the *residual correction*, respectively. Given an approximation $\mathbf{v}$, one can compute $\mathbf{r}$, solve (2.2.3) approximately for $\mathbf{e}$, and then use (2.2.4) to obtain a new (and hopefully improved) *approximation* for $\mathbf{u}$. If the error is only *approximately* computed from (2.2.3), the value calculated from (2.2.4) will in fact only be an *approximation* for $\mathbf{u}$. The entire process can be summarized as follows. Let the superscript 'k' refer to the k^{th} approximation, '$k+1$' to the $(k+1)^{st}$ approximation, and so on. Then from Definitions

2.2.2 and 2.2.3,

$$\mathbf{e}^{(k)} = \mathbf{u} - \mathbf{v}^{(k)}, \tag{2.2.5}$$

$$\mathbf{r}^{(k)} = \mathbf{f} - A\mathbf{v}^{(k)}. \tag{2.2.6}$$

Equations (2.2.3) and (2.2.4) then become,

$$A\mathbf{e}^{(k)} = \mathbf{r}^{(k)}, \tag{2.2.7}$$

$$\mathbf{u} = \mathbf{v}^{(k)} + \mathbf{e}^{(k)}, \tag{2.2.8}$$

such that

$$\mathbf{e}^{(k)} = A^{-1}\mathbf{r}^{(k)}. \tag{2.2.9}$$

If A^{-1} is replaced with an easily computable matrix approximation B, then (2.2.9) becomes,

$$\tilde{\mathbf{e}}^{(k)} = B\mathbf{r}^{(k)}, \tag{2.2.10}$$

where $\tilde{\mathbf{e}}$ represents an approximation to $\mathbf{e}$. Thus, the new *approximation* for $\mathbf{u}$ can be obtained from (2.2.8), resulting in the general form

$$\mathbf{v}^{(k+1)} = \mathbf{v}^{(k)} + \tilde{\mathbf{e}}^{(k)} = \mathbf{v}^{(k)} + B\mathbf{r}^{(k)}. \tag{2.2.11}$$

It can be noted that if $\mathbf{f} = \mathbf{0}$, the exact solution is "known" ($\mathbf{u} = \mathbf{0}$), and the error in $\mathbf{v}$ is simply $-\mathbf{v}$. This observation will be useful later. With this material in place, the stage is now set to introduce the relaxation methods.

2.2.1 Jacobi Method

The Jacobi, or simultaneous displacement, method amounts to solving the i^{th} equation for the approximation v_i while holding all other variables fixed. Therefore, rewriting the *1D* model problem (2.2.2) as

$$-v_{i-1} + 2v_i - v_{i+1} = h^2 f_i \ , \quad i = 1, 2, \ldots, n-1 \ , \quad v_0 = v_n = 0, \tag{2.2.12}$$

a new approximation for point v_i can be obtained according to the update equation

$$v_i^{(k+1)} = \frac{1}{2}(v_{i-1}^{(k)} + v_{i+1}^{(k)} + h^2 f_i). \tag{2.2.13}$$

Equation (2.2.13) can also be written in matrix form. This is accomplished by first expressing A as a combination of submatrices, that is,

$$A = D - L - U, \tag{2.2.14}$$

where D is the diagonal part of A, and -L and -U are the strictly lower and upper triangular parts of A, respectively. By absorbing the h^2 term in $\mathbf{f}$, the Jacobi method can be expressed in matrix form as follows:

$$\begin{aligned} (D - L - U)\mathbf{u} &= \mathbf{f} \\ D\mathbf{v}^{(k+1)} &= (L + U)\mathbf{v}^{(k)} + \mathbf{f} \\ \mathbf{v}^{(k+1)} &= D^{-1}(L + U)\mathbf{v}^{(k)} + D^{-1}\mathbf{f}. \end{aligned} \tag{2.2.15}$$

Furthermore, letting

$$R_J = D^{-1}(L + U), \tag{2.2.16}$$

equation (2.2.15) can be rewritten as

$$\mathbf{v}^{(k+1)} = R_J\mathbf{v}^{(k)} + D^{-1}\mathbf{f}. \tag{2.2.17}$$

R_J is known as the *Jacobi iteration matrix* or the *Jacobi error propagation matrix*. This is because R_J propagates the error as the iterations progress. This point will be clarified further in Section 2.2.3.

2.2.2 Gauss-Seidel Method

The Gauss-Seidel (GS) method is the same as the Jacobi method, except that when equation i is solved, the updated value replaces v_i immediately in the iteration process. The GS method may be written as:

$$v_i^{(k+1)} \leftarrow \frac{1}{2}\left(v_{i-1}^{(k+1)} + v_{i+1}^{(k)} + h^2 f_i\right), \tag{2.2.18}$$

or, in matrix form, taking the same approach as in Section 2.2.1,

$$\mathbf{v}^{(k+1)} \leftarrow (D - L)^{-1}U\mathbf{v}^{(k)} + (D - L)^{-1}\mathbf{f}. \tag{2.2.19}$$

Here, '$\leftarrow$' stands for *replacement* or *overwriting*. The Gauss-Seidel error propagation matrix is defined as:

$$R_{GS} = (D - L)^{-1}U, \tag{2.2.20}$$

such that (2.2.19) becomes

$$\mathbf{v}^{(k+1)} \leftarrow R_{GS}\mathbf{v}^{(k)} + (D - L)^{-1}\mathbf{f}. \tag{2.2.21}$$

Again, R_{GS} propagates the error in successive iterations, or *sweeps*. This point will be expanded on in the next section.

2.2.3 Convergence Analysis

For both the Jacobi and GS methods ((2.2.17) and (2.2.21), respectively), the update formula is linear in $\mathbf{v}$ and does not change from one iteration to the next. This type of formula is known as a *stationary linear iteration*. Sections 2.2.1 and 2.2.2 concluded by saying that the error propagation matrices propagate the error in each new approximation. This point can be clarified in a general sense by first considering that each method can be expressed in the form:

$$\mathbf{v}^{(k+1)} = R\mathbf{v}^{(k)} + C(\mathbf{f}). \tag{2.2.22}$$

Also, it can be noted that the exact solution is unchanged by the iteration (i.e. is a *fixed point*). This is verified for the GS method by using (2.2.20), (1.1.5), and (2.2.14) in (2.2.21) such that

$$\begin{aligned}
&\mathbf{v}^{(k+1)} \leftarrow R_{GS}\mathbf{v}^{(k)} + (D - L)^{-1}\mathbf{f} \\
&\mathbf{v}^{(k+1)} \leftarrow R_{GS}\mathbf{v}^{(k)} + (D - L)^{-1}A\mathbf{u} \\
&\mathbf{v}^{(k+1)} \leftarrow (D - L)^{-1}U\mathbf{v}^{(k)} + (D - L)^{-1}(D - L - U)\mathbf{u} \\
&\mathbf{v}^{(k+1)} \leftarrow (D - L)^{-1}U\mathbf{v}^{(k)} - (D - L)^{-1}U\mathbf{u} + \mathbf{u}
\end{aligned}$$

and thus

$$\mathbf{v}^{(k+1)} = \mathbf{v}^{(k)} = \mathbf{u} \text{ if } \mathbf{v}^{(k)} = \mathbf{u}.$$

The same result can be shown to hold for the Jacobi method. In general, it is true that

$$\mathbf{u} = R\mathbf{u} + C(\mathbf{f}). \tag{2.2.23}$$

Equation (2.2.22) can then be subtracted from (2.2.23) to show that

$$\begin{aligned} \mathbf{u} - \mathbf{v}^{(k+1)} &= R\mathbf{u} + C(\mathbf{f}) - \left(R\mathbf{v}^{(k)} + C(\mathbf{f})\right) \\ &= R\left(\mathbf{u} - \mathbf{v}^{(k)}\right) \\ \Rightarrow\ \mathbf{e}^{(k+1)} &= R\mathbf{e}^{(k)}. \end{aligned} \tag{2.2.24}$$

Thus, (2.2.24) explains why R is called the error propagation matrix. If the iteration is performed m times,

$$\mathbf{e}^{(m)} = R^m\mathbf{e}^{(0)}. \tag{2.2.25}$$

Here the superscript '0' corresponds to the initial approximation. With this result in hand, a convergence analysis of the methods can be performed. First, however, some preliminary work is required. Definition of the spectral radius, ρ, and a natural norm are given in Appendix A by (A.0.9) and (A.0.2), respectively.

Theorem 2.2.1. [11] Let A be an arbitrary square matrix. Then for any natural norm,

$$\rho(A) \leq \|A\|. \tag{2.2.26}$$

Theorem 2.2.2. [11] For any $\epsilon > 0$, there exists a natural norm such that

$$\|A\| \leq \rho(A) + \epsilon \tag{2.2.27}$$

Definition 2.2.4. A matrix A is convergent if $\lim_{n\to\infty} A^n = O$, where O is the matrix with all zero entries.

Theorem 2.2.3. [5] The following statements are equivalent:

i. A is convergent;

ii. $\lim_{n\to\infty} \|A^n\| = 0$ for some natural norm;

iii. $\lim_{n\to\infty} \|A^n\| = 0$ for all natural norms;

iv. $\rho(A) < 1$.

Corollary 2.2.4. [11] A is convergent if a natural norm exists for which $\|A\| < 1$.

Noting that (2.2.25) leads to

$$\|\mathbf{e}^m\| \leq \|R^m\|\|\mathbf{e}^0\|, \tag{2.2.28}$$

it follows from Theorem 2.2.3 that the iteration will converge if and only if $\rho(R) < 1$. In other words, the norm of the error will approach zero as the number of iterations is increased if and only if $\rho(R) < 1$. The spectral radius, $\rho(R)$, can be interpreted as the *asymptotic convergence factor* of the iterative error reduction equation (2.2.25), because it predicts the worst case error reduction over many iterations [4]. It is the asymptotic factor by which the norm of the error is reduced in each iteration, and it can be used to estimate the number of iterations required to reduce the error by a factor of 10^{-d}. This estimate is obtained by letting m be the smallest integer that satisfies

$$\frac{\|\mathbf{e}^m\|}{\|\mathbf{e}^0\|} \leq 10^{-d}. \tag{2.2.29}$$

It then follows approximately from (2.2.28) and Theorem 2.2.1 that

$$\rho^m(R) \leq 10^{-d}, \tag{2.2.30}$$

and that

$$m \geq -\frac{d}{\log_{10}(\rho(R))}. \tag{2.2.31}$$

The results of this section are valuable tools for the analysis of the convergence properties of any stationary linear iteration. However, this only provides a general picture of error reduction. To observe precisely how the error is reduced with each relaxation sweep, a spectral analysis is required. This is presented in the next section.

2.2.4 Spectral Error Analysis

This section presents the specific error reduction capabilities of the GS method. A similar analysis can be performed for the Jacobi method. Analysis of the GS method was selected for presentation since the GS method is a component of the AMG algorithm used in obtaining the results of Chapter 7.

As shown in Section 2.2.3, the convergence properties of stationary linear iterations depend directly on the spectral radius as highlighted in Theorem 2.2.3. While the spectral radius is the asymptotic factor by which the norm of the error is reduced in each iteration, it says nothing about how specific error components are reduced in each iteration. To examine this point for the GS method, consider the eigenvalue problem for the error propagation matrix,

$$R_{GS}\mathbf{w} = \lambda\mathbf{w}, \tag{2.2.32}$$

where λ is an eigenvalue, and $\mathbf{w}$ is an eigenvector, of R_{GS}. Using (2.2.20), (2.2.32) can be rewritten as

$$U\mathbf{w} = \lambda(D - L)\mathbf{w}. \tag{2.2.33}$$

For the 1D model problem (2.2.1), the resulting problem is

$$\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ 0 & 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_{n-1} \end{bmatrix} = \lambda \begin{bmatrix} 2 & 0 & 0 & \cdots & 0 \\ -1 & 2 & 0 & \ddots & \vdots \\ 0 & -1 & 2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_{n-1} \end{bmatrix},$$

where the indices of the eigenvector components, $0 \leq j \leq n$, refer to the grid location (w_0 and w_n are omitted since $\mathbf{w}$ is equal to 0 at the boundaries of the domain). It can be shown that the eigenvalues of R_{GS} are given by [3]

$$\lambda_k(R_{GS}) = \cos^2\left(\frac{k\pi}{n}\right), \quad 1 \leq k \leq n-1, \tag{2.2.34}$$

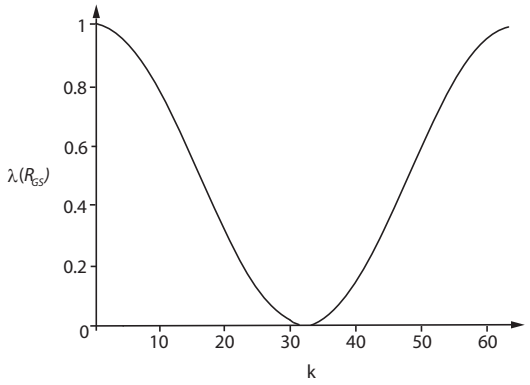

Figure 2.2: GS eigenvalue plot for the case $n = 64$.

and that the eigenvectors of R_{GS} are given by

$$\mathbf{w}_{k,j}(R_{GS}) = \cos^j\left(\frac{k\pi}{n}\right)\sin\left(\frac{k\pi j}{n}\right) \ , \quad 0 \leq j \leq n \ , \quad 1 \leq k \leq n-1. \tag{2.2.35}$$

To proceed, it is worth establishing the eigenvalues and eigenvectors of the matrix operator A. The eigenvalues and eigenvectors of A are given by [4]

$$\lambda_k(A) = 4\sin^2\left(\frac{k\pi}{2n}\right) \ , \quad 1 \leq k \leq n-1, \tag{2.2.36}$$

$$\mathbf{w}_{k,j}(A) = \sin\left(\frac{k\pi j}{n}\right) \ , \quad 1 \leq k \leq n-1 \ , \quad 0 \leq j \leq n. \tag{2.2.37}$$

There are two important things to note about these results. Firstly, the eigenvectors of A are actually *Fourier modes* forming a basis for an $(n-1)$-dimensional vector space, and k is the corresponding *wavenumber*. As such, the error in an iterative process may be expressed in terms of these eigenvectors such that

$$\mathbf{e}^{(m)} = \sum_{k=1}^{n-1} c_k \mathbf{w}_k(A), \tag{2.2.38}$$

where the coefficients $c_k \in \mathbb{R}$ simply indicate the weight of each mode present in the error. Secondly, the eigenvectors of R_{GS} do not coincide with those of A. Therefore, $\lambda_k(R_{GS})$ gives the convergence rate for the k^{th} mode of R_{GS}, and not for the k^{th}

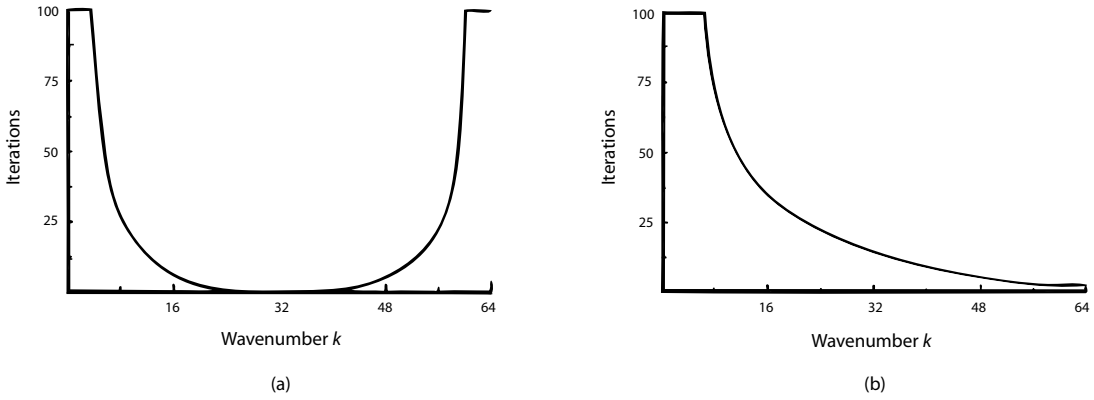

Figure 2.3: GS iteration matrix applied to the model problem with an initial guess consisting of (a) single eigenvectors of R_{GS} and (b) single eigenvectors of A. The figure shows the number of iterations required to reduce the norm of the initial error by a factor of 100 for each initial guess on a grid with $n = 64$ points (from [4]).

mode of A. This can be clarified by considering (2.2.32). There, it is noted that R_{GS} acting on the k^{th} mode of R_{GS}, $\mathbf{w}_k(R_{GS})$, is the same as the k^{th} eigenvalue of R_{GS}, $\lambda_K(R_{GS})$, acting on $\mathbf{w}_k(R_{GS})$. Thus, the magnitude of $\lambda_K(R_{GS})$ gives the rate at which the norm of $\mathbf{w}_k(R_{GS})$ approaches zero with each application of R_{GS}. This is important when examining the error reduction capabilities of the GS method. For example, consider the case where $n = 64$. The eigenvalues for R_{GS} are shown in Figure 2.2. This indicates that, when the initial guess (and error) consists of single eigenvectors of R_{GS}, one should expect modes near the middle of the spectrum to exhibit the fastest convergence. Indeed this is the case as indicated in Figure 2.3 (a). However, the result is quite different if the initial guess consists of single modes of A, as indicated in Figure 2.3 (b). Here it is observed that modes of higher frequency are damped much more effectively with each iteration than modes of lower frequency. Two definitions are now in order.

Definition 2.2.5. Loosely define *oscillatory modes*, or *oscillatory error components*, as those Fourier modes whose wavenumber is *greater than* $\frac{n}{2}$.

Definition 2.2.6. Loosely define *smooth modes*, or *smooth error components*, as those Fourier modes whose wavenumber is *less than* $\frac{n}{2}$.

The efficiency of an iterative scheme is largely dependent on characteristics as exhibited in Figure 2.3 (b). While a great deal of error elimination may be possible early in the iteration process – this being due to reduction of oscillatory components – it will not be beneficial to continue if the smooth components can not be effectively reduced. This observation is at the heart of multigrid, which aims to represent smooth modes as oscillatory ones by working on a grid with fewer points. This will be explained further in Chapter 3.

In closing this section, it is worth stating two facts about the Jacobi method in order to better justify the choice to use GS as a smoother in the multigrid algorithm for the test cases that will be discussed later in this thesis. Firstly, Jacobi does not effectively reduce oscillatory error components, and is therefore not a good candidate for multigrid. The reason for this becomes clear by looking at the eigenvalue spectrum of the Jacobi error propagation matrix. The interested reader may find this discussed in [4]. Also, while there exists a modified Jacobi relaxation scheme, known as weighted Jacobi, that effectively reduces oscillatory error components and has similar performance characteristics to the GS method (this is also examined in [4]), the GS relaxation method is more straightforward to analyze from an implementation perspective, and is therefore chosen for use in this thesis.

2.3 Acceleration Methods

This section presents the conjugate gradient (CG) and generalized minimal residual (GMRES) methods for solving systems of the form (1.1.5). While these are historically and practically significant methods, they are not a component of the algebraic multigrid algorithm. They are however often used as accelerators for algebraic multigrid. It is in this context that GMRES is used in the work of this thesis. As such, an outline of the algorithm is in order. While CG is not used in the work of this thesis, its introduction provides a good starting point for describing GMRES, and so will be

presented. The discussion of CG and GMRES will be brief, and the reader is referred to the literature for more information.

2.3.1 Conjugate Gradient Method

The classical conjugate gradient algorithm is presented here as in [8]. Although CG is an iterative method, it does converge to the exact solution in a finite number of iterations; however, it is only guaranteed to converge if A is symmetric and positive definite (SPD) [9].

The CG method can be derived by modifying the method of steepest descent. The latter is obtained using the procedure described below. First, consider the functional

$$F(\mathbf{v}) = \frac{1}{2}\langle \mathbf{v}, A\mathbf{v}\rangle - \langle \mathbf{f}, \mathbf{v}\rangle. \tag{2.3.1}$$

Since A is SPD, it follows that solving $A\mathbf{u} = \mathbf{f}$ is equivalent to minimizing $F(\mathbf{v})$ over $\mathbf{v}$ (the proof is left to the reader). It can also be shown that the gradient of $F(\mathbf{v})$ is given by

$$\nabla F(\mathbf{v}) = \mathbf{f} - A\mathbf{v} = \mathbf{r}. \tag{2.3.2}$$

This indicates the direction of greatest instantaneous rate of change of $F(\mathbf{v})$. If $\mathbf{v}^{(0)}$ is the initial guess, then successively better approximations $\mathbf{v}^{(1)}$, $\mathbf{v}^{(2)}, \ldots,$ $\mathbf{v}^{(i)}, \ldots,$ $\mathbf{v}^{(m)}$, where m is the approximation at which convergence criteria are satisfied, can be obtained by continually moving in the direction $\nabla F\left(\mathbf{v}^{(i)}\right)$. The procedure is given by

$$\mathbf{v}^{(i+1)} = \mathbf{v}^{(i)} + \alpha_i \nabla F\left(\mathbf{v}^{(i)}\right) = \mathbf{v}^{(i)} + \alpha_i \mathbf{r}^{(i)}, \tag{2.3.3}$$

where α_i should be chosen to minimize $F\left(\mathbf{v}^{(i+1)}\right)$. Using (2.3.1), it follows that

$$\alpha_i = \frac{\left(\mathbf{r}^{(i)}, \mathbf{r}^{(i)}\right)}{\left(\mathbf{r}^{(i)}, A\mathbf{r}^{(i)}\right)}. \tag{2.3.4}$$

Therefore, the steepest descent method can be described as

$$
\begin{aligned}
&\mathbf{v}^{(0)} = arbitrary \\
&for\ i = 0\ to\ m\ \{ \\
&\quad \mathbf{r}^{(i)} = \mathbf{f} - A\mathbf{v}^{(i)} \\
&\quad \alpha_i = \frac{\left(\mathbf{r}^{(i)}, \mathbf{r}^{(i)}\right)}{\left(\mathbf{r}^{(i)}, A\mathbf{r}^{(i)}\right)} \\
&\quad \mathbf{v}^{(i+1)} = \mathbf{v}^{(i)} + \alpha_i \mathbf{r}^{(i)} \\
&\}.
\end{aligned}
$$

Turning now to the CG method, take the same approach as for the steepest descent method, but do not specify the direction for the next approximation by $\nabla F\left(\mathbf{v}^{(i)}\right)$. Instead, let $\mathbf{v}^{(i+1)} = \mathbf{v}^{(i)} + \alpha_i \mathbf{p}^{(i)}$, where the direction vectors $\mathbf{p}^{(i)}$ are to be chosen such that they are A-orthogonal to each other $\left(i.e.\ \langle \mathbf{p}^{(i+1)}, A\mathbf{p}^{(i)} \rangle = 0\right)$. The CG method can then be written as follows:

$$
\begin{aligned}
&\mathbf{v}^{(0)} = arbitrary \\
&for\ i = 0\ to\ m\ \{ \\
&\quad \mathbf{r}^{(i)} = \mathbf{f} - A\mathbf{v}^{(i)} \\
&\quad \mathbf{p}^{(i)} = \begin{cases} \mathbf{r}^{(i)}\ , & i = 0 \\ \mathbf{r}^{(i)} + \beta_i \mathbf{p}^{(i-1)}\ , & i = 1,\ 2, \ldots,\ m \end{cases} \\
&\qquad with\ \ \beta_i = -\frac{\left(\mathbf{r}^{(i)}, A\mathbf{p}^{(i-1)}\right)}{\left(\mathbf{p}^{(i-1)}, A\mathbf{p}^{(i-1)}\right)} \\
&\quad \alpha_i = \frac{\left(\mathbf{p}^{(i)}, \mathbf{r}^{(i)}\right)}{\left(\mathbf{p}^{(i)}, A\mathbf{p}^{(i)}\right)} \\
&\quad \mathbf{v}^{(i+1)} = \mathbf{v}^{(i)} + \alpha_i \mathbf{p}^{(i)} \\
&\}.
\end{aligned}
$$

While the method of steepest descent can be very slow to converge, CG is guaranteed to converge to the exact solution in at most N iterations, where N is the number of unknowns in the linear algebraic system (of the form (1.1.5)) that is to be solved.

In practice, and for reasons too complicated to be explained here, the CG method is often able to closely approximate the exact solution in much less than N steps.

2.3.2 Generalized Minimal Residual Method

Basic aspects of the GMRES method are summarized here as in [21]. For additional information, the reader is also referred to [15].

Although it was not mentioned in Section 2.3.1, the CG method is a Krylov subspace method,[3] where the Krylov subspace K^m is defined as

$$K^m = span\left[\mathbf{r}^{(0)},\ A\mathbf{r}^{(0)}, \ldots,\ A^{m-1}\mathbf{r}^{(0)}\right], \tag{2.3.5}$$

and the Krylov subspace approximation is given by

$$\mathbf{v}^{(m)} \in \mathbf{v}^{(0)} + K^m = \mathbf{v}^{(0)} + span\left[\mathbf{r}^{(0)},\ A\mathbf{r}^{(0)}, \ldots,\ A^{m-1}\mathbf{r}^{(0)}\right]. \tag{2.3.6}$$

The vectors in (2.3.5) form an orthogonal basis for an m dimensional space, and (2.3.6) corresponds to calculating the approximation with minimal residual in a suitable norm for all $m = 1,\ 2, \ldots, until\ convergence\ criteria\ are\ satisfied.$ In the case of classical CG, which applies to SPD matrices, the residual is minimized in the norm

$$\|\mathbf{r}\| = \langle \mathbf{r}, A^{-1}\mathbf{r} \rangle. \tag{2.3.7}$$

GMRES is not, however, restricted to SPD matrices, and the minimization is performed using the $\|\cdot\|_2$ norm, where $\|\cdot\|_2$ is the matrix p-norm with $p = 2$ as defined by equation (A.0.4). One difficulty with this approach is that all vectors in the Krylov subspace must be stored in order to ensure that the norm of the residual is minimized (note that for the CG method this is not needed). For large problems, this can present storage complications. A fix for this problem is given by *restarted* GMRES(m), in which the subspace is completely removed and restarted with a new one after m iterations. GMRES(m) with a right preconditioner C is summarized below. Note that $h_{i,j}$ refers to the entry in row i and column j of matrix H.

[3] A presentation of other Krylov subspace methods, for example the *biconjugate gradient method*, may be found in [23].

$\mathbf{v}^{(0)} = arbitrary$

$Set\ the\ matrix\ H = O\ with\ dimension\ (m+1) \times m$

$\mathbf{r}^{(0)} = \mathbf{f} - A\mathbf{v}^{(0)}$

$\beta = \|\mathbf{r}^{(0)}\|_2$

$\mathbf{b}_1 = \dfrac{\mathbf{r}^{(0)}}{\beta}$

$for\ j = 1\ to\ m\ \{$

$\quad \mathbf{r}_j = C^{-1}\mathbf{b}_j$

$\quad \mathbf{w} = A\mathbf{r}_j$

$\quad for\ i = 1\ to\ j\ \{$

$\quad\quad h_{i,j} = \langle \mathbf{w}, \mathbf{b}_i \rangle$

$\quad\quad \mathbf{w} = \mathbf{w} - h_{i,j}\mathbf{b}_j$

$\quad\quad h_{j+1,j} = \|\mathbf{w}\|_2$

$\quad\quad \mathbf{b}_{j+1} = \dfrac{\mathbf{w}}{h_{j+1,j}}$

$\quad \}$

$\}$

$Define\ B_m = [\mathbf{b_1}, \ldots,\ \mathbf{b_m}]$

$y_m = \min\limits_{y} \|\beta\mathbf{e}_1 - Hy\|_2\ , \quad \left(\mathbf{e}_1 = [1,\ 0, \ldots,\ 0]^T\right)$

$\mathbf{v}^{(m)} = \mathbf{v}^{(0)} + C^{-1}B_m y_m$

$\mathbf{r}^{(m)} = \mathbf{f} - A\mathbf{v}^{(m)}$

$If\ \mathbf{r}^{(m)}\ satisfies\ convergence\ criteria:\ stop,$

$\quad else:\ restart\ \mathbf{v}^{(0)} \leftarrow \mathbf{v}^{(m)}$

Often, Krylov methods can be slow. This is because their convergence largely depends on the condition number of the matrix A. However, their performance can be substantially improved if they are effectively preconditioned. It is in this context that multigrid was used to obtain some of the test results in Chapter 7.

Chapter 3

Multigrid

This chapter introduces the basic components and structure of the multigrid algorithm following the treatment of [4]. Multigrid is a multilevel iterative method used for solving systems of the form $A\mathbf{u} = \mathbf{f}$. Multigrid can be separated into two categories known as geometric multigrid and algebraic multigrid. Geometric multigrid uses successively coarser grids that are constructed based on the geometric grid information from the discretized PDE problem. Since grid information is required by geometric multigrid, it is unable to handle unstructured grids. AMG, in contrast, does not rely on geometric grid information to construct coarse grids, but instead uses information found in the matrix operator A. In this way, PDE problems on unstructured grids, and even problems for which no physical grid exists, can be solved by AMG. This chapter is meant to provide an overview of the multigrid algorithm and theory that is common to both geometric and algebraic multigrid. The complete AMG algorithm will be presented in Chapter 4. The reader interested in more details on geometric multigrid is encouraged to consult the literature, for example [4] and [21]. This chapter will first provide the motivation for multigrid based on the results developed in Chapter 2. It will then expand on this motivation and present a detailed overview of the entire algorithm that is common to both geometric and algebraic multigrid. Several real applications of AMG will also be presented. The chapter will conclude with an investigation of storage and computational costs of the multigrid algorithm.

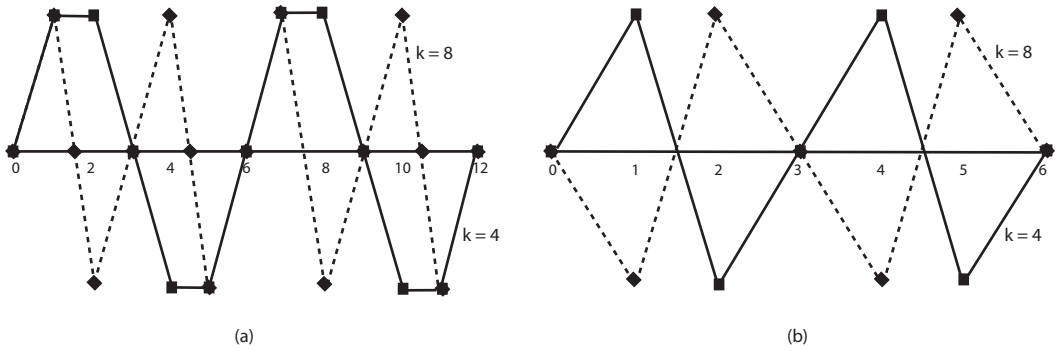

Figure 3.1: The k=4 and k=8 modes on a grid with (a) n=12 points (Ω^h), and (b) n=6 points (Ω^{2h}).

3.1 Motivation

A significant result from Chapter 2 is the error reduction capability of many relaxation methods. In particular, for the elliptic model problem (2.2.1), it was shown for the GS method that only oscillatory Fourier modes could be effectively reduced. It was then alluded to that multigrid attempted to remedy this weakness by expressing smooth error modes as more oscillatory ones on coarser grids. This section clarifies this point. The discussion that follows is meant to be interpreted in the context of GS relaxation applied to the elliptic model problem (2.2.1). That is, it is assumed that relaxation effectively reduces oscillatory error modes. This assumption allows the motivation of multigrid to be explained in an intuitive fashion, but is not a requirement for AMG algorithms. More will be said about this issue for AMG in Section 4.2.

Consider two waves, one with wavenumber $k = 4$ and the other with wavenumber $k = 8$, on 1D grids with $n = 12$ and $n = 6$ points as illustrated in Figure 3.1. First, note that Figure 3.1 refers to the n=12 grid as Ω^h and the n=6 grid as Ω^{2h}.This is because the multigrid algorithm uses a series of grids with successively fewer points. In geometric multigrid, the grid with the greatest number of points is often called Ω^h, the grid with the second greatest number of points Ω^{2h}, the grid with the third greatest Ω^{4h}, and so on. The reason that the superscript labels contain powers of two

is because in geometric multigrid, successive grids are often selected by choosing half of the points from each dimension of one grid to form the grid with the next greatest number of points. Therefore, Ω^h would have 2^d as many points as Ω^{2h}, where d is the dimension of the problem, and so a power of two is used in the grid labeling. Regardless of the type of grid structure, this thesis will adopt the notation that Ω^h represents the grid (in a series of grids with successively fewer points) that has the greatest number of points, Ω^{2h} represents the grid with the second greatest number of points, Ω^{4h} the grid with the third greatest number of points, and so on. To simplify grid description, a definition is now introduced.

Definition 3.1.1. Call grid a *finer* than grid b if grid a contains more points than grid b, and *coarser* than grid b if grid a contains fewer points than grid b.

Figure 3.1 illustrates that smooth modes ($k < \frac{n}{2}$) become more oscillatory on coarser grids. For example, the $k = 4$ mode in Figure 3.1 is the 4^{th} mode out of a possible 12 on Ω^h, but is the 4^{th} mode out of a possible 6 on Ω^{2h}. This implies that smooth modes which could not be effectively reduced on Ω^h, could be on Ω^{2h} if some method for transferring between grids could be developed. With such a transfer method, one could reduce oscillatory error on Ω^h by relaxation, transfer to Ω^{2h} such that smooth error from Ω^h appears more oscillatory, reduce the oscillatory error on Ω^{2h} (which is the smooth error from Ω^h) by relaxation, and transfer back to Ω^h such that both smooth and oscillatory error components are now reduced. It should be noted, however, that even when a mode is considered oscillatory by Definition 2.2.5, this does not mean that it will be completely reduced by just a few relaxations on a particular grid. As indicated in Figure 2.3, oscillatory modes of higher frequency exhibit the best error reduction. Therefore, in order to effectively reduce smooth modes on Ω^h by the grid transfer method just described, it might be necessary to transfer several times to successively coarser grids. In this way, the smooth error from Ω^h becomes sufficiently oscillatory on coarser grids to allow for effective error reduction. This idea leads to the concept known in multigrid as a *V-cycle*, the grid configuration of which is illustrated pictorially in Figure 3.2. In a V-cycle, an approximation of the exact solution is relaxed on, and then transferred to a coarser grid. The process is repeated

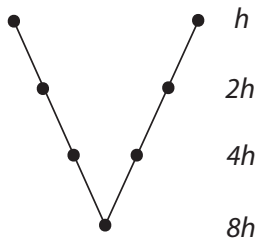

Figure 3.2: Four-level grid configuration for a V-cycle.

for some predetermined number of grids. The approximation is then transferred back to successively finer grids until the finest grid is reached. In this way, all components of the error on Ω^h can hopefully be effectively reduced. This is the fundamental idea behind the multigrid algorithm. As it turns out, the V-cycle is an effective method for solving a variety of problems. This thesis will always assume that a V-cycle is employed even though other types of cycles exist. Much more needs to be said about the V-cycle, and this will be done in the remainder of this chapter and in Chapter 4.

The question remains as to what becomes of the oscillatory modes ($k > \frac{n}{2}$) on Ω^{2h}. As in Figure 3.1 for the $k = 8$ case, it can be shown that the k^{th} mode on Ω^h becomes the $(n-k)^{th}$ mode on Ω^{2h}. This is caused by an effect known as *aliasing*, and occurs when there is not enough information available to accurately construct a signal.[1] Therefore, oscillatory modes on Ω^h are represented as smooth ones on Ω^{2h}. This is obviously an undesirable effect since relaxation will not be able to effectively reduce these modes on Ω^{2h}; however, this does not present a problem since these modes can be effectively reduced on Ω^h.

In summary, oscillatory modes on Ω^h can be effectively reduced on Ω^h, and smooth modes on Ω^h can be represented as oscillatory ones and effectively reduced on Ω^{2h}, or subsequently coarser grids. Therefore, in order to effectively reduce all components of the error, it seems appropriate to use the recursive error reduction method proposed in this section. This central idea behind the multigrid algorithm is further explained in the next section.

[1] This is caused by violation of Nyquist's Sampling Theorem. For more information on the subject of aliasing, the reader is referred to [19].

3.2 Multigrid Overview

This section first gives a summary of the basic multigrid algorithm that applies to both geometric and algebraic multigrid. The motivation for multigrid from an application perspective is then discussed, and several real applications of AMG are presented. The section concludes with a more detailed introduction of the multigrid algorithm.

3.2.1 Summary of the Multigrid Algorithm

This summary is meant to serve as a reference for the reader in the remainder of this thesis. All of the components listed in this summary are not meant to be understood at this point, nor is their motivation. They will be further explained in the rest of this chapter and in Chapter 4.

Multigrid (MG) is an algorithm that exploits the grid transfer method introduced in Section 3.1. Assuming that a sequence of successively coarser grids has been defined, where on a given level coarse grid points are a subset of fine grid points, and assuming that a matrix operator A is defined on all grids, the multigrid algorithm (using V-cycles) can be summarized as follows [4]:

$$\mathbf{v}^h \leftarrow MG\left(\mathbf{v}^h, \mathbf{f}^h\right)$$

While convergence criteria not satisfied, do (perform a V-cycle):

- Relax on $A^h\mathbf{u}^h = \mathbf{f}^h$ ν_1 times with initial guess $\mathbf{v}^h$.
- Compute $\mathbf{f}^{2h} = I_h^{2h}\mathbf{r}^h$.
  - Relax on $A^{2h}\mathbf{u}^{2h} = \mathbf{f}^{2h}$ ν_1 times with initial guess $\mathbf{v}^{2h} = \mathbf{0}$.
  - Compute $\mathbf{f}^{4h} = I_{2h}^{4h}\mathbf{r}^{2h}$.
    - Relax on $A^{4h}\mathbf{u}^{4h} = \mathbf{f}^{4h}$ ν_1 times with initial guess $\mathbf{v}^{4h} = \mathbf{0}$.
    - Compute $\mathbf{f}^{8h} = I_{4h}^{8h}\mathbf{r}^{4h}$.

      $\vdots$

      - Solve $A^{Lh}\mathbf{u}^{Lh} = \mathbf{f}^{Lh}$.

      $\vdots$

- Correct $\mathbf{v}^{4h} \leftarrow \mathbf{v}^{4h} + I_{8h}^{4h}\mathbf{v}^{8h}$.
- Relax on $A^{4h}\mathbf{u}^{4h} = \mathbf{f}^{4h}$ ν_2 times with initial guess $\mathbf{v}^{4h}$.

- Correct $\mathbf{v}^{2h} \leftarrow \mathbf{v}^{2h} + I_{4h}^{2h}\mathbf{v}^{4h}$.
- Relax on $A^{2h}\mathbf{u}^{2h} = \mathbf{f}^{2h}$ ν_2 times with initial guess $\mathbf{v}^{2h}$.

- Correct $\mathbf{v}^{h} \leftarrow \mathbf{v}^{h} + I_{2h}^{h}\mathbf{v}^{2h}$.
- Relax on $A^{h}\mathbf{u}^{h} = \mathbf{f}^{h}$ ν_2 times with initial guess $\mathbf{v}^{h}$.

end do

Here L is used to label the coarsest grid, ν_1 and ν_2 are positive integers, and I_a^b is an operator that takes a vector from grid a and represents it on grid b. Although it was assumed that a set of successively coarser grids had already been defined, it should be noted that a significant and important part of the AMG algorithm centers on the coarse-grid selection process. This is commonly referred to as *coarsening*, and will be discussed in Chapter 4. A method used to define the matrix operator A on all grids will also be presented in Chapter 4.

As illustrated in the algorithm summary, multigrid is an iterative method. One V-cycle corresponds to one iteration, and several V-cycles are usually required to satisfy a convergence criterion.

3.2.2 Multigrid Objective

This section describes the motivation for the multigrid algorithm from an application perspective. To aid in this discussion, several definitions are first in order.

Definition 3.2.1. Define the *convergence factor* of the k^{th} iteration to be $\frac{\|\mathbf{r}^k\|_2}{\|\mathbf{r}^{k-1}\|_2}$.[2]

Definition 3.2.2. Define a *scalable algorithm* to be one for which the convergence factor is independent of problem size, and for which storage and computational cost per V-cycle are linearly proportional to problem size.

[2] $\frac{\|\mathbf{r}^k\|_2}{\|\mathbf{r}^{k-1}\|_2}$ is also known as the *residual ratio*.

Definitions 3.2.1 and 3.2.2 are somewhat idealized. Typically the convergence factor will not be exactly the same for consecutive iterations. However, if an algorithm converges in a constant number of iterations for all problem sizes, the convergence factor can be considered independent of problem size, and therefore the algorithm is scalable (assuming that storage and computational cost per V-cycle are linearly proportional to problem size). In a multigrid context, a scalable algorithm is one for which the number of V-cycles required for convergence is independent of problem size, while also satisfying the V-cycle cost requirements of Definition 3.2.2. It should be noted that scalability by itself does not guarantee an efficient algorithm. It could be true that an algorithm is perfectly scalable by Definition 3.2.2, but requires a large number of iterations to converge, and a large V-cycle cost. As such, the additional requirement that V-cycle cost and number of iterations needed for convergence be as low as possible is implied when discussing scalability. In this way, fast execution time and low memory size can be attained even for large problems if an algorithm is scalable.

The primary objective of multigrid is to be able to solve large problems efficiently. As such, scalability is the most important factor in designing or improving a multigrid algorithm, and will be the central focus in evaluating the effectiveness of the changes to the AMG algorithm proposed in Chapter 5. It is also desirable that an AMG algorithm be effective in a parallel implementation, as this allows problem size to be substantially increased. Although the results of this thesis were obtained using serial AMG, the potential for each of the algorithms tested to excel in parallel will also be considered.

The theory of multigrid was originally developed for linear elliptic PDEs, but has since been extended to a larger class of problems for both geometric and algebraic multigrid (see for example [21]). In the AMG context, much of this work has focused on problems for which the fine-grid operator is a symmetric M-matrix (see for example [20]).[3] It has also been found, however, that the concepts of multigrid can be applied much more generally to a larger class of problems. While this is a positive result,

[3]Definition of an M-matrix is given in Definition A.0.1.

the drawback is that no single multigrid algorithm exists that optimally achieves scalability across a wide-range of problems. With this in mind, this thesis aims to improve the robustness of AMG by modifying several existing components of the algorithm such that scalability is improved for a variety of problems.

3.2.3 Real Applications of AMG

This section is meant to provide the reader with an idea of real applications for which AMG can be applied. Two problems from the literature will be presented that demonstrate the potential of AMG.

The first problem is taken from [26], and is a computational fluid dynamics (CFD) problem that simulates physical flow phenomena for a Boeing 747 over the entire body of the aircraft.[4] The equations that need to be solved are the Navier-Stokes equations. The surface mesh used is illustrated in Figure 3.3, and indicates that this is a large problem.

As stated in [26], GMRES and Gauss-Seidel exhibit convergence difficulties due to the complex configuration of the domain, while AMG works well in both serial and parallel implementations.

The second problem is taken from [25], and deals with the field of Electro- and MagnetoEncephaloGraphy (EEG/MEG) source localization. Using high resolution finite element modeling, a current distribution inside the human brain can be noninvasively reconstructed using electric field measurements taken outside the head. The equations that need to be solved are omitted here since they require a detailed presentation; however, it can be stated that the resulting system is large, sparse, linear, and has many different right-hand sides. The domain of the problem is illustrated in Figure 3.4, where the mesh for the head model is shown. This is a tetrahedrae mesh

[4]

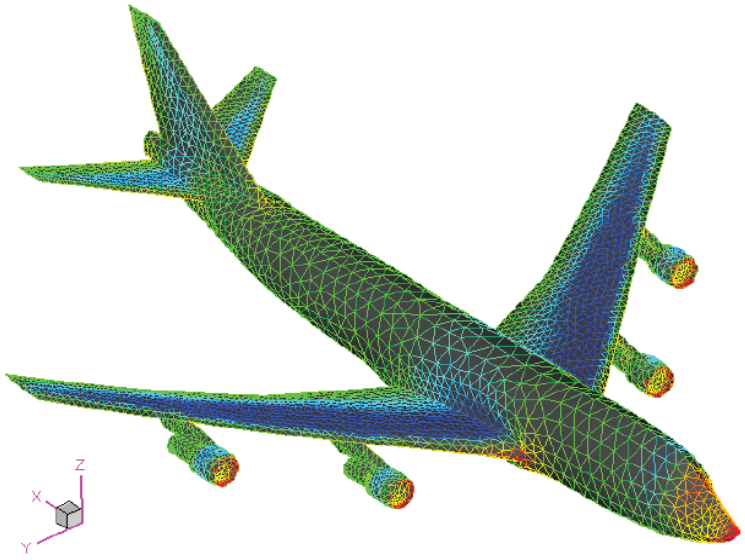

Figure 3.3: Boeing 747 configuration and surface mesh.

with 118299 nodes. Since medical and neuropsychological diagnosis and research are time sensitive, efficient solution of such a large problem is necessary.

The work of [25] shows that for a parallel implementation, CG accelerated AMG significantly outperforms the CG accelerated Jacobi method (a well-known solution technique in finite element based source localization). This is accomplished for both the 118299 node problem illustrated in Figure 3.4, and a 325384 node mesh with cubic elements.

The two examples highlighted in this section illustrate the power of AMG for solving real applications. AMG is able to handle complex geometries and large problem sizes, which allows for high resolution. These examples also demonstrate the robustness of AMG, and its ability to effectively solve problems in parallel. Nevertheless, the scalability of existing AMG algorithms for complex applications is often not optimal. The modifications proposed in this thesis aim to improve scalability for difficult applications.

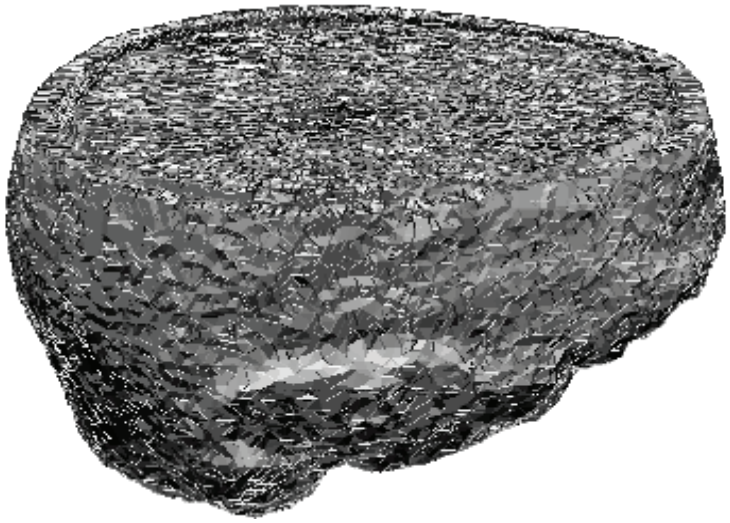

Figure 3.4: Head model mesh.

3.2.4 Detailed Description of the Multigrid Algorithm

This section presents a detailed description of the multigrid algorithm, and expands on the summary given in Section 3.2.1. It should be noted that this introduction applies to both geometric and algebraic multigrid. Any component not explicitly defined in this section will be covered for AMG in Chapter 4.

Recall from Chapter 2 that for a linear algebraic system (1.1.5), and an initial guess $\mathbf{v}^{(0)}$, the error and residual are given, respectively, by

$$\mathbf{e}^{(0)} = \mathbf{u} - \mathbf{v}^{(0)}, \tag{3.2.1}$$

$$\mathbf{r}^{(0)} = \mathbf{f} - A\mathbf{v}^{(0)}. \tag{3.2.2}$$

From (3.2.1) and (3.2.2), it follows that solving $A\mathbf{u} = \mathbf{f}$ is equivalent to solving $A\mathbf{e}^{(0)} = \mathbf{r}^{(0)}$. Furthermore, it can be shown that solving $A\mathbf{u} = \mathbf{f}$ with an arbitrary initial guess $\mathbf{v}$, is equivalent to solving the associated residual equation ($A\mathbf{e} = \mathbf{r}$) with the specific initial guess $\mathbf{e} = \mathbf{0}$. Also, recall that after several fine-grid relaxations, the remaining error is smooth, and may therefore be calculated accurately on a coarser

grid. These observations lead to the formulation of the multigrid algorithm, which rests on the fact that the residual equation can be used to relax on the error on the coarse grid. The first step in arriving at the multigrid algorithm is to present what is known as the *coarse-grid correction scheme*, which as in [4], is defined as follows:

- Relax on $A\mathbf{u} = \mathbf{f}$ on Ω^h to obtain an approximation $\mathbf{v}^h$.
- Compute the residual, $\mathbf{r} = \mathbf{f} - A\mathbf{v}^h$.
- Relax on $A\mathbf{e} = \mathbf{r}$ on Ω^{2h} with the initial guess $\mathbf{e} = \mathbf{0}$ to obtain an approximation to the error $\mathbf{e}^{2h}$.
- Correct the approximation obtained on Ω^h with the error estimate obtained on Ω^{2h}: $\mathbf{v}^h \leftarrow \mathbf{v}^h + \mathbf{e}^{2h}$.

Thus, the idea is to relax on Ω^h until convergence deteriorates (oscillatory modes have been reduced), transfer the residual from Ω^h to a coarser grid (Ω^{2h}), relax on the residual equation on Ω^{2h} to obtain an approximation to the error $\mathbf{e}^{2h}$, and then transfer the error back to Ω^h to obtain a new approximation $\mathbf{v}^h$. Of course, several issues still need to be addressed. In particular, how is the residual transferred from Ω^h to Ω^{2h}? How is the matrix operator A defined on Ω^{2h}? How is the error transferred from Ω^{2h} to Ω^h so that the correction on $\mathbf{v}^h$ can be made? The short answer to all of these questions is through the use of suitable matrix operators. Definition of these operators is now given.

Definition 3.2.3. Define the *restriction operator*, I_h^{2h}, to be the matrix operator that takes a vector from Ω^h and expresses it on Ω^{2h}: $\mathbf{x}^{2h} = I_h^{2h}\mathbf{x}^h$.

Definition 3.2.4. Define the *interpolation* (or *prolongation*) *operator*, I_{2h}^h, to be the matrix operator that takes a vector from Ω^{2h} and expresses it on Ω^h: $\mathbf{x}^h = I_{2h}^h\mathbf{x}^{2h}$.

The restriction and interpolation operators are also used to define the coarse-grid version of the matrix operator (A^{2h}), called the *coarse-grid operator*. Explicit definition of the restriction and interpolation operators and the coarse-grid operator are left

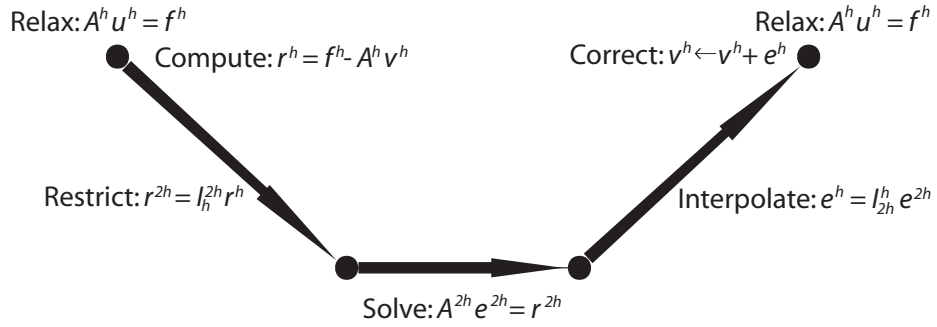

Figure 3.5: Two-grid correction scheme.

until Chapter 4 where they are introduced for AMG. With Definitions 3.2.3 and 3.2.4 in place, the coarse-grid correction scheme can now be reformulated to give what is known as the *two-grid correction scheme* (TG) [4]:

$$\mathbf{v}^h \leftarrow TG\left(\mathbf{v}^h, \mathbf{f}^h\right)$$

- Relax ν_1 times on $A^h\mathbf{u}^h = \mathbf{f}^h$ on Ω^h with a given initial guess $\mathbf{v}^h$.
- Compute the residual $\mathbf{r}^h = \mathbf{f}^h - A^h\mathbf{v}^h$ on Ω^h and restrict it to the coarse grid by $\mathbf{r}^{2h} = I_h^{2h}\mathbf{r}^h$.
- Solve $A^{2h}\mathbf{e}^{2h} = \mathbf{r}^{2h}$ on Ω^{2h}.
- Interpolate the coarse-grid error to the fine grid by $\mathbf{e}^h = I_{2h}^h\mathbf{e}^{2h}$ and correct the fine-grid approximation by $\mathbf{v}^h \leftarrow \mathbf{v}^h + \mathbf{e}^h$.
- Relax ν_2 times on $A^h\mathbf{u}^h = \mathbf{f}^h$ on Ω^h with initial guess $\mathbf{v}^h$.

In this discussion, ν_1 and ν_2 are positive integers, and typically take values of one or two. It has been found experimentally that these values generally provide a good balance between the cost of relaxation and the benefit gained in convergence. The two-grid correction scheme is also illustrated pictorially in Figure 3.5.

Instead of solving the coarse grid problem exactly, its solution can be approximated by recursively invoking the two-grid correction scheme. This leads to the *V-cycle scheme* described in Section 3.1, and further discussed below. To economize

on notation, the right-hand side of the residual equation is called $\mathbf{f}^{2h}$ rather than $\mathbf{r}^{2h}$, and the solution of the residual equation is called $\mathbf{u}^{2h}$ instead of $\mathbf{e}^{2h}$ (since they represent new right-hand side and solution vectors, respectively). It then follows that $\mathbf{v}^{2h}$ can be used to denote the approximation to $\mathbf{u}^{2h}$. As in [4], the V-cycle scheme (V) can be defined recursively as:

$$\mathbf{v}^h \leftarrow V\left(\mathbf{v}^h, \mathbf{f}^h\right)$$

1. Relax ν_1 times on $A^h\mathbf{u}^h = \mathbf{f}^h$ with a given initial guess $\mathbf{v}^h$.
2. If Ω^h = coarsest grid, go to step 4.

   Else

   $$\mathbf{f}^{2h} \leftarrow I_h^{2h}\left(\mathbf{f}^h - A^h\mathbf{v}^h\right),$$
   $$\mathbf{v}^{2h} \leftarrow \mathbf{0},$$
   $$\mathbf{v}^{2h} \leftarrow V^{2h}\left(\mathbf{v}^{2h}, \mathbf{f}^{2h}\right).$$
3. Correct $\mathbf{v}^h \leftarrow \mathbf{v}^h + I_{2h}^h\mathbf{v}^{2h}$.
4. If Ω^h = coarsest grid, solve $A^h\mathbf{u}^h = \mathbf{f}^h$.

   Else, relax ν_2 times on $A^h\mathbf{u}^h = \mathbf{f}^h$ with initial guess $\mathbf{v}^h$.

This is simply the recursive definition of what is inside the *while loop* in the multigrid summary in Section 3.2.1. Typically, when the coarsest grid is encountered in step 4, an exact solution method such as Gaussian elimination or a large number of relaxation sweeps is applied in order to obtain an accurate solution. Doing so is usually acceptable (i.e. computationally efficient) since the problem will have been coarsened to a reasonably small size.

Remark 3.2.1. The multigrid V-cycle is often referred to as a $V(\nu_1, \nu_2)$ cycle to signify that ν_1 relaxation sweeps are performed before each restriction step, and that ν_2 relaxation sweeps are performed after each interpolation step.

The framework for the multigrid algorithm is now complete. The next section will consider implementation costs. Specific definitions of all components of the algorithm not yet explained will be presented for AMG in Chapter 4.

3.3 Implementation Costs

The storage and computational costs of the multigrid algorithm will now be examined based on the assumption that a V-cycle scheme is employed. It is also assumed that a set of grids and coarse-grid operators have been defined. The analysis performed in this section is done in the context of structured grids; however, a similar extension to unstructured grids can be made that yields comparable results.

3.3.1 Storage Cost

The *storage cost* of the multigrid V-cycle algorithm is assessed here. Since AMG has a unique method for measuring the storage cost of matrices (which will be presented in Chapter 4), this section will only consider the storage cost of vectors.

A d-dimensional grid with n points in each dimension has n^d total points. Two arrays, $\mathbf{v}$ and $\mathbf{f}$, must be stored on the finest level for a total of $2n^d$ storage locations. Assuming that n is a power of 2, and that Ω^{2h} is recursively constructed by halving the number of points in each dimension of Ω^h (as is done in geometric multigrid), subsequently coarser grids require 2^{-d} times the amount of storage of the next finest grid. Therefore, the total storage requirement (number of locations) can be expressed as the following geometric series:

$$Storage\ Cost = 2n^d\{1 + 2^{-d} + 2^{-2d} + \ldots + 2^{-nd}\} < \frac{2n^d}{1 - 2^{-d}}. \tag{3.3.1}$$

Therefore, in one, two, and three dimensions, the total storage cost is less than 2, $\frac{4}{3}$, and $\frac{8}{7}$ times the cost of storage of the fine grid quantities, respectively. This shows that the multigrid algorithm requires storage costs on the order of the fine grid problem, and that the storage requirement increases linearly with respect to problem size. This is a desirable result with regard to scalability.

3.3.2 Computational Cost

When considering the computational cost of the multigrid V-cycle algorithm, the cost of intergrid transfer operations is usually neglected as they typically only amount to

between 10-20% of the cost of the entire cycle [4]. To proceed with the analysis, a definition is first in order.

Definition 3.3.1. Define a *work unit, WU*, to be the cost of performing one relaxation sweep on the finest grid ($O\left(n^d\right)$ operations).

As in Section 3.3.1, assume that n is a power of 2 and that Ω^{2h} is constructed by halving the number of points in each dimension of Ω^h. Then, the work required on a coarse grid is 2^{-d} times the amount of work required on the next finest grid. Since each level is visited twice, and assuming that a V(1,1) cycle is employed, it follows that the computational cost is given by the following geometric series:

$$\textit{Computational Cost} = 2\{1 + 2^{-d} + 2^{-2d} + \ldots + 2^{-nd}\}\ WU < \frac{2}{1 - 2^{-d}}\ WU. \quad (3.3.2)$$

In one, two, and three dimensions, this corresponds to approximately 4, $\frac{8}{3}$, and $\frac{16}{3}$ *WU*, respectively. Note that the computational cost, when measured in terms of work units, is not adversely affected if the problem size is increased. While this result is desirable for the purpose of scalability, it does not indicate how multigrid performs compared to other algorithms in terms of total computational cost. To facilitate such a comparison, the number of iterations (V-cycles) required to reduce the error by some standard amount needs to be known. This parameter is calculated below.

Although not stated explicitly until now, there are actually two forms of error present in the numerical solution of a PDE problem. The first is called the *discretization error*, and is a direct result of moving from the continuous problem $\mathbf{u}$, to the discrete problem $\mathbf{u^h}$. Considering the 1D model problem (2.2.1), the discretization error can be defined as:

$$E_i^h = u(x_i) - u_i^h\ , \quad 1 \leq i \leq n-1. \quad (3.3.3)$$

Using the definition of the discrete L^2 norm in (A.0.5), it can be shown that (3.3.3) can be bounded by

$$\|\mathbf{E}^h\|_h \leq Kh^p\ , \quad K > 0\ \in\ \mathbb{R}\ , \quad p > 0\ \in\ \mathbb{I}. \quad (3.3.4)$$

A second type of error is that generated in approximating the exact solution of the discrete problem $\mathbf{u}^h$ with $\mathbf{v}^h$. This error is called the *algebraic error*, and is simply that defined by Definition 2.2.2, namely,

$$\mathbf{e}^h = \mathbf{u}^h - \mathbf{v}^h. \tag{3.3.5}$$

Returning now to the issue of computational cost, the goal of an iterative method is to minimize the algebraic error efficiently, and hopefully attain a result that is of the order of the discretization error. As in Section 3.3.1, consider a d-dimensional problem with n^d unknowns such that the grid spacing is $h = \frac{1}{n}$ in each dimension. Furthermore, assume that the V-cycle has a convergence factor bound, γ, that is independent of h. If the scheme is to reduce the algebraic error from $O(1)$ to the order of the discretization error (that is $O\left(h^p\right) = O\left(n^{-p}\right)$), the number of V-cycles required, υ, must satisfy:

$$\gamma^\upsilon = O\left(n^{-p}\right), \tag{3.3.6}$$

such that

$$\upsilon = O\left(\log n\right). \tag{3.3.7}$$

As can be noted from (3.3.2), the cost of one V-cycle is $O\left(n^d\right)$. Therefore, the cost of obtaining a solution whose error is of the order of the discretization error using the multigrid V-cycle algorithm is $O\left(n^d \log n\right)$. This is clearly much better than the $O\left(n^{3d}\right)$ result obtained for Gaussian elimination in Section 2.1, and is often much less work than that required by standard iterative methods since multigrid has the ability to reduce all error components efficiently, whereas many other methods do not.

While the results presented in this section are desirable with regard to the scalability and work requirements of the multigrid V-cycle algorithm, it should be noted that another type of cycle, the *full multigrid* (FMG) cycle, actually has a total computational cost that is $O\left(n^d\right)$, and is therefore optimal. Only the V-cycle scheme is examined in this thesis since it is the basic building block of many multigrid schemes, including the optimal FMG cycle.

Chapter 4

Algebraic Multigrid

This chapter expands on the multigrid algorithm introduced in Chapter 3 in the context of AMG. This chapter will first present the underlying concepts and theoretical background that motivate the design of an AMG algorithm, along with several conventions that are unique to AMG for measuring storage and computational costs. The ideas of coarsening (selection of coarse grids) and interpolation introduced in Chapter 3 are then fully explained by introducing several different coarsening and interpolation methods. Advantages and disadvantages of each of these methods are also discussed. The chapter concludes by defining the restriction and coarse-grid operators, and provides a theoretical justification for their definition.

4.1 Introduction

The most basic differences between geometric and algebraic multigrid are the way in which coarsening is performed and the interpolation operators are defined. In geometric multigrid, where structured grids are considered, coarsening and interpolation are defined based on the known grid structure (i.e. fixed grid spacing). In this way, coarsening and interpolation can be performed in exactly the same way regardless of the location on the grid. For example, as stated in Chapter 3, coarsening in geometric multigrid is often performed by simply selecting half the points (every second point) in each dimension of a structured grid. Interpolation, the process by which coarse

grid quantities are transferred to fine grids, also relies on spatial information in geometric multigrid. This geometrical approach for coarsening and interpolation cannot be adopted for unstructured grids (where there is no regular spatial structure), and this is where the main differences lie between AMG and geometric multigrid. AMG is able to handle both structured and unstructured grids, and problems for which no physical grid exists, but coarsening and interpolation must be defined in a completely different way than in geometric multigrid. AMG instead uses the information present in the matrix operator A to select coarse grids and determine interpolation operators. This process will be explained in detail in the remainder of this chapter.

4.2 Background Theory

Fourier modes cannot simply be constructed on an unstructured grid due to a lack of spatial information. This observation is even more profound when one considers problems for which no physical grid even exists. Therefore, while it makes sense to discuss the reduction of smooth error components as defined by Definition 2.2.6 for structured grids, the same can not be said from an analytical perspective for unstructured grids, or problems for which a grid is not defined. This does not imply that the relaxation method reduces the error modes in a different way from the structured case, it simply means that the analysis can not proceed in the same way. To accommodate an analysis in AMG, it is necessary to redefine what is meant by smooth error.

Definition 4.2.1. Define *algebraically smooth error* to be any error that is not effectively reduced by relaxation.

For simplicity, algebraically smooth error will be referred to as *smooth error* in the AMG context. When geometrically smooth error is considered, it will be clearly stated. Note that it is possible for error to be algebraically smooth, but geometrically unsmooth (oscillatory). This is described in greater detail in [4, 20].

The significance of smooth error in AMG can now be illustrated. This will be done for the subset of symmetric M-matrices that are diagonally dominant (as defined by

(A.0.1)), and in the context of GS smoothing. It should be noted that a similar analysis can be performed for other relaxation methods. It follows that smooth error is characterized by

$$\|R\mathbf{e}\|_A \approx \|\mathbf{e}\|_A, \tag{4.2.1}$$

where the *A-norm*, $\|\cdot\|_A$, is defined by (A.0.8). It can be shown that, for smooth error,

$$\sum_{i=1}^{N} \frac{r_i^2}{a_{ii}} \ll \sum_{i=1}^{N} r_i e_i, \tag{4.2.2}$$

where r_i refers to the i^{th} entry in $\mathbf{r}$ (the same meaning applies for e_i and $\mathbf{e}$), and N is the number of unknowns in the linear algebraic system (of the form (1.1.5)). As discussed in [4], this implies that, *on average*, smooth error satisfies

$$|r_i| \ll a_{ii}|e_i|. \tag{4.2.3}$$

This indicates that smooth error has relatively small residuals, which can be written loosely as

$$A\mathbf{e} \approx \mathbf{0}. \tag{4.2.4}$$

Equation (4.2.4) is a very important result for AMG. It says that if the error is smooth, then e_i can be approximated (interpolated) well by a weighted average of the error in its neighbouring points:

$$a_{ii}e_i \approx -\sum_{j \neq i} a_{ij}e_j. \tag{4.2.5}$$

As will be shown, this result can then be used to define the interpolation and restriction operators used to transfer between grids. Before this can be accomplished, however, some preliminary work is needed.

It follows that since A is assumed to be diagonally dominant, there exists a dominant entry a_{ii} in each row of A, or equivalently in each equation i of the discretized system. As such, it makes sense to say that equation i primarily influences the value of u_i. Of course, the entire set of equations is required to accurately solve the system, but one can still think of equation i as being principally responsible for calculating the i^{th} unknown. Now consider equation i exclusively, and assume that there are nonzero

coefficients at positions j and k – that is, in row i of A, a_{ii}, a_{ij}, and a_{ik} are the only entries that are nonzero. A critical observation of AMG is that not all of these entries may be equally important in determining u_i. For instance, if a_{ik} is relatively small in comparison to a_{ij}, then the error in variable j may have more impact on the accuracy of the iterative approximation of variable i than the error in variable k. If, in the iterative process, u_j is changed by a small amount, it will have a significant effect on the calculation of u_i. However, even if u_k changes by a large amount, it will not have a significant impact on the calculation of variable u_i provided that the coefficient a_{ik} is sufficiently small. This means that variable k is not as important as variable j in determining the error of variable i in (4.2.5). Obviously what is meant by "sufficiently small" needs to be clarified, and this leads to the following definitions.

Definition 4.2.2. [4] Given a threshold value $0 < \theta \leq 1$, the variable (point) u_i *strongly depends* on the variable (point) u_j if

$$-a_{ij} \geq \theta \max_{k \neq i} \{-a_{ik}\}. \tag{4.2.6}$$

Definition 4.2.3. [4] If the variable u_i strongly depends on the variable u_j, then the variable u_j *strongly influences* the variable u_i.

If u_i strongly depends on u_j, u_j will be a good interpolatory candidate for u_i. This is true for two reasons. Firstly, a larger coefficient in equation i has a greater effect on the calculation of u_i than does a smaller one, and is therefore a better candidate for interpolation if only a limited number of points can be used. The second reason requires a more accurate description of how the error in point i relates to the error in all other points, j, in equation i. It can be shown that, on average for each i,[1]

$$\sum_{j \neq i} \frac{|a_{ij}|}{a_{ii}} \frac{(e_i - e_j)^2}{e_i^2} \ll 1 \ , \quad 1 \leq i \leq n. \tag{4.2.7}$$

For the inequality in (4.2.7) to be satisfied, it follows that either or both of the fractions must be small. However, if variable i is strongly connected to variable j,

[1]This can be derived for symmetric M-matrices by first noting that $\|\mathbf{e}\|_A \ll \|\mathbf{e}\|_D$ and then performing a careful expansion. A detailed derivation may be found in [20].

$|a_{ij}|/a_{ii}$ will be $O(1)$. Therefore, e_i must be approximately equal to e_j. As a result, *smooth error is said to vary slowly in the direction of strong connection.* It is in this direction that interpolation will be most accurate. Consequently, when interpolating fine grid quantities from coarse grid ones, it is best to do so in the direction of strong connection. Likewise, it is desirable that coarse grids be selected from fine ones such that strong connections may be exploited in interpolation.

The motivation for defining the restriction and interpolation operators introduced in Chapter 3 has now been established. The remainder of this chapter focuses on methods for defining these operators, as well as the coarse grid operators, in preparation for the algorithmic modifications and results presented in Chapters 5 and 7. First, however, several methods used to measure computational cost in AMG will be introduced.

4.3 Computational Cost Measurement in AMG

As will be illustrated in the next section, the ratio of fine to coarse grid points is not known until the coarsening process has been completed. Thus, unlike geometric multigrid, a predictive cost analysis cannot be performed for AMG. This section will present several tools that are used to measure computational cost in AMG.

One measure of computational cost used in AMG is *grid complexity*, which is defined as follows.

Definition 4.3.1. [4] Define *grid complexity* to be the total number of grid points on all grids, divided by the number of grid points on the finest grid.

The grid complexity for a specific coarsening method applied to a specific problem is a useful tool for measuring aspects of computational cost. For geometric multigrid, if coarse grids are selected by choosing half the points in each dimension from the fine grid, the grid complexities for one, two, and three dimensions are 2, $\frac{4}{3}$, *and* $\frac{8}{7}$, respectively. Grid complexity provides a direct measure of the storage required for

right-hand side and solution vectors, and is useful for comparing the performance of different coarsening strategies.

Another measure of computational cost used in AMG is *operator complexity*, which is defined as follows.

Definition 4.3.2. [4] Define *operator complexity*, C_{op}, to be the total number of nonzero entries, in all matrices A^{kh}, divided by the number of nonzero entries in the fine-grid operator A^h.

Like grid complexity, operator complexity is also useful for measuring storage cost, as it indicates precisely the amount of storage required by all operators A^{kh} on all grids. Furthermore, the amount of work required by relaxation and residual computations is directly proportional to the number of nonzeros in the A^{kh}, and these processes dominate a V-cycle. Therefore, operator complexity is also a good indicator of the amount of work required in each iteration of an AMG algorithm. With all this in mind, it follows that small operator complexity that increases linearly with problem size signifies a scalable algorithm (if also accompanied by good convergence).

To introduce two other useful measures of computational cost in AMG, two definitions are required.

Definition 4.3.3. Define the *setup phase* to consist of those processes in an AMG algorithm that are responsible for generating the components needed to perform a V-cycle. This includes the coarsening procedure, definition of the interpolation operators, and definition of the coarse-grid operators A^{kh} on all grids.

Definition 4.3.4. Define the *solve phase* to consist of the iterative application of the V-cycle scheme.

The time required for the setup and solve phases of an AMG algorithm will be referred to simply as *setup time* and *solve time*, respectively.

4.4 Coarsening

Coarsening is a somewhat heuristic coarse grid selection procedure that aims to exploit the result presented in the previous section – that smooth error varies slowly in the direction of strong connection. Coarsening can be performed in many different ways, and several methods that are relevant to the work of this thesis will be presented in this section.

For any coarsening strategy, a trade-off exists in the resulting AMG algorithm between storage and computational cost and convergence, which ultimately affect scalability. Obviously, the greater the number of points that are kept on coarser grids, the greater the accuracy that can be achieved in interpolation. However, retaining a large number of grid points when moving from a fine grid to a coarse grid increases storage cost, and may result in a large execution time per V-cycle. The primary objective of multigrid is to move to suitably coarser grids, in order to (hopefully) avoid high storage cost and execution time per V-cycle, while approximating smooth error accurately through restriction and interpolation. Indeed, this thesis examines modifications to current coarsening algorithms that aim to improve AMG convergence properties and execution speed while reducing storage cost. This section will introduce the standard Ruge-Stüben [14], parallel modified independent set [18], and Cleary-Jones-Luby-Plassman [10] algorithms – all effective coarsening strategies in their own respects. Before proceeding, however, several definitions are in order.

Definition 4.4.1. When referring to two subsequent grids, let the subset of fine-grid points selected to form the coarse grid be called *C-points*, and let the fine-grid points that are not C-points be called *F-points*.

Definition 4.4.2. [4] For each fine-grid point i, define the *neighbourhood* of i, N_i, to be the set of all points $j \neq i$ such that $a_{ij} \neq 0$.

Definition 4.4.3. Let $S_i \subset N_i$ be the set of points that strongly influence point i, i.e. the points on which i strongly depends.

Definition 4.4.4. Let S_i^T be the set of points that strongly depend on point i, i.e. the points strongly influenced by i.

Definition 4.4.5. For each F-point i, let $C_i \subset S_i \subset N_i$ be the set of C-points that strongly influence i.

Definition 4.4.6. Define the *auxiliary strength matrix* S as:

$$S_{ij} = \begin{cases} 1 & if\ i \neq j\ and\ u_i\ strongly\ depends\ on\ u_j \\ 0 & otherwise \end{cases} \tag{4.4.1}$$

The strength matrix provides a useful computational tool for accessing the strength information of an operator. The nonzero entries in row i of S indicate the points in S_i, and the nonzero entries in column i of S indicate the points in S_i^T. Calculation of S is easy to implement, and S lends itself to sparse matrix storage which aids in computational efficiency.
With this material in place, specific coarsening methods can now be introduced.

4.4.1 Classical Ruge-Stüben Coarsening

This section presents the classical Ruge-Stüben (RS) coarsening algorithm originally introduced in [14]. This is perhaps the most historically significant coarsening strategy, and is still commonly used today. Although RS was not used to produce any of the results presented in this thesis (the reason for which is stated at the end of this subsection), it is still useful for introducing coarsening and for comparing the design of other coarsening schemes.

The RS algorithm selects coarse grid points, and therefore aims to approximate smooth error accurately in restriction and interpolation, based on two heuristic criteria. As in [4], these may be stated as follows:

H-1: For each F-point i, every point $j \in S_i$ that strongly influences i should either be in the coarse interpolatory set C_i, or (if j is an F-point) should strongly depend on at least one point in C_i (in short, strong F-F connections require a common C-point).

H-2: The set of coarse-grid points should be a maximal subset of all fine-grid points with the property that no C-point strongly depends on another C-point (maximal independent set).

Heuristics **H-1** and **H-2** represent attempts to achieve good convergence with each V-cycle, and minimal computational cost per V-cycle, respectively. The motivation for **H-1** follows from the fact that an effective coarsening scheme should allow accurate interpolation of smooth functions. Since smooth error varies slowly in the direction of strong connection, smooth error will be interpolated well between points that are strongly connected. Thus, it is desirable that an F-point i have as many points as possible from S_i in C_i. However, as will be illustrated by example, this is not always possible. Therefore, if a point j is in S_i, but is not in C_i, it is desirable that j be strongly influenced by a point in C_i. The reason for this is basically that, if point k is in C_i and strongly influences point j, point j can be accurately interpolated from point k, and then point i can be accurately interpolated from point j. In this sense, point i interpolates from point j *indirectly*, in order to achieve accurate interpolation along directions of strong connection. This is illustrated in Figure 4.1, where an edge indicates a strong connection, an arrow indicates the direction of a strong connection, a black dot indicates a C-point, and a white dot indicates an F-point.

Heuristic **H-2** is much easier to justify, and is in place to ensure that coarse grids are in fact sufficiently coarsened in order to keep computational costs per V-cycle at scalable levels. As will be explained in Section 4.5 on interpolation, error values at C-points are known from computations on the coarse grid, and can therefore be interpolated directly to the fine grid. Thus, **H-2** ensures that a maximal number of C-points are chosen, which allows interpolation to be as accurate as possible, but also states that C-points should not be strongly connected (on the fine grid), since this would increase computation cost while providing no clear advantage to interpolation.

As will be shown by example, it is not always possible to enforce both **H-1** and **H-2** simultaneously. When this occurs, **H-1** is enforced rigorously while **H-2** is used only as a guide. Generally the reduction in computational cost per V-cycle that would be achieved if **H-2** were enforced rigorously is lost in the convergence behaviour of the iteration. Thus, the RS algorithm proceeds in two passes. The first pass creates

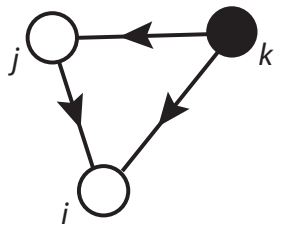

Figure 4.1: Example illustrating *indirect* interpolation for a strong F-F connection. Point i interpolates from point k, both directly, and also through point j.

an initial partition of the grid into C- and F-points, and the second pass enforces **H-1** rigorously.

The first pass is accomplished by assigning to each point a measure of its candidacy to serve as a C-point. Since interpolation is based on the fact that smooth error varies slowly in the direction of strong connection, a good way to define the suitability measure of a point i, is simply to count the number of points strongly influenced by i. Call this count λ_i, and it follows that it is the cardinality of S_i^T. From an implementation perspective, λ_i is simply the column sum of column i of the strength matrix S. The greater the value of λ_i, the more useful point i will be in interpolation if defined as a C-point. Thus, the first pass commences by arbitrarily choosing one point, i, that attains $\max_i \lambda_i$ to be a C-point. In accordance with **H-2**, all points that are strongly influenced by i are then made F-points. Since it would be best for a newly defined F-point, j, to interpolate from as many C-points as possible, the measure of all *unassigned* points that strongly influence j is increased. In this way, the points that strongly influence j are more likely to be chosen as C-points. The process of choosing C-points based on maximal measure, and making all strongly influenced points F-points is then repeated. This is done until all points are either C- or F-points. Before proceeding, it should be noted that all fine-grid points, i, that strongly influence no other point ($S_i^T = \emptyset$), are defined as F-points. This is done because there would be no benefit in making such a point a C-point, since it would not be used by any other point in interpolation (interpolation is performed along the

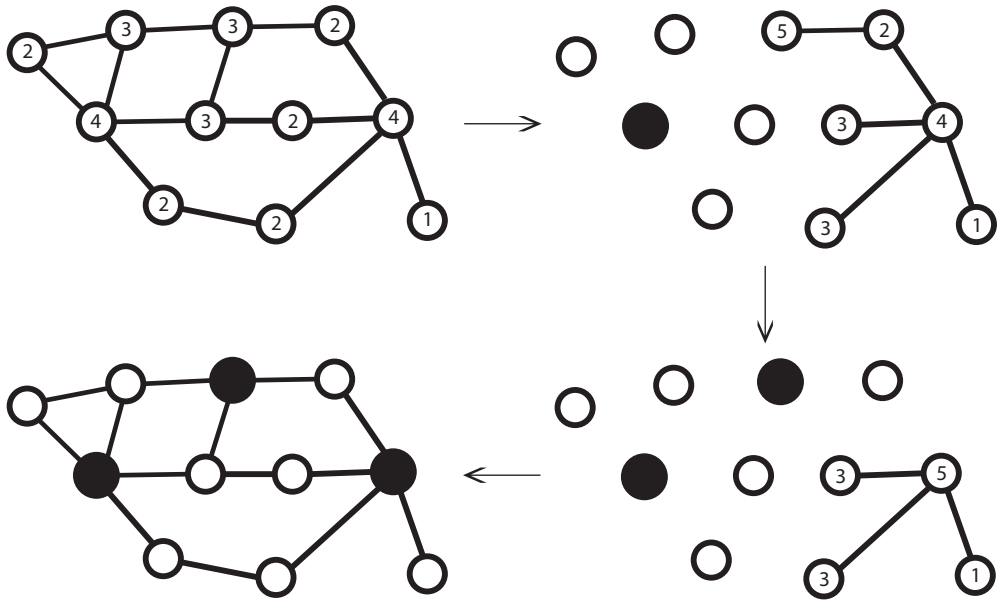

Figure 4.2: Illustration of the first pass of classical RS coarsening.

direction of strong connection). An example of the first pass is illustrated for an unstructured grid in Figure 4.2. Edges are meant to indicate a strong connection in both directions, a black dot indicates a C-point, a white dot indicates an F-point, and a white dot containing a number indicates an unassigned point along with that point's measure at the corresponding stage in the coarsening process. Note that after a point is assigned, all edges associated with that point are removed since they are no longer needed.

It can be noted in Figure 4.2, that at the conclusion of the first pass, **H-2** is satisfied, but **H-1** is not. The purpose of the second pass is to enforce **H-1** rigorously. This means that some F-points must be changed to C-points. This is usually done so that a minimum number of C-points are added. In the case of the example in Figure 4.2, the resulting grid becomes that shown in Figure 4.3.

As can be observed from the above example, the RS coarsening algorithm is highly sequential. As a result, RS coarsening does not perform well in parallel, and can lead to undesirable execution times in both serial and parallel implementations.

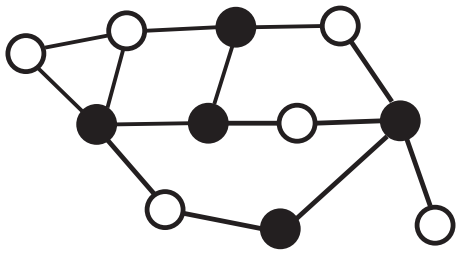

Figure 4.3: Final coarsened grid using RS coarsening.

In addition, the second pass often contributes to high grid and operator complexities such that storage and computational costs are adversely affected (this is especially true along processor boundaries in parallel). However, RS coarsening does exhibit almost optimal scalability in terms of convergence for a variety of problems. This illustrates the trade-off between convergence and computational cost discussed earlier. As a result, RS coarsening often causes an AMG algorithm to be unscalable when both convergence and computational cost are considered. This is especially true for three-dimensional (3D) problems. Since one goal of the work in this thesis is to be able to solve large-scale problems effectively in parallel, RS was not used in any of the algorithms tested.

4.4.2 PMIS Coarsening

This section presents the parallel modified independent set (PMIS) algorithm introduced by De Sterck, Yang, and Heys [17, 18]. This is a maximal independent set algorithm similar to that of Luby [13]. It also resembles RS coarsening, except that heuristic **H-1** is not enforced rigorously; that is, F-F connections without a common C-point are permitted.

As in the case of RS coarsening, PMIS initially assigns a measure λ_i to all points to quantify their suitability to become a C-point. In addition, PMIS then assigns a random number between zero and one, $Rand([0,1])$, to break ties between all points that have maximal measure. Instead of then choosing the point with maximal measure

as in the RS case, PMIS chooses as C-points those whose measure is greater than that of all of their strongly influencing *and* strongly dependent neighbours ($\lambda_i > \lambda_k \; \forall \; k \in S_i \bigcup S_i^T$). Points that are strongly influenced by these new C-points are then made F-points. The process is repeated – unassigned points whose measure is greater than that of all of their unassigned neighbours are made C-points and so on – until all points have been declared as C- or F-points. It should be noted that, as is done for RS coarsening, all fine-grid points that strongly influence no other point ($S_i^T = \emptyset$) are defined as F-points.

The PMIS algorithm is summarized as follows (as in [18]), and assumes that the strength matrix S has already been defined:

- Given S, define weights $\lambda_i \; \forall \; i \in \Omega$: $\lambda_i = \sum_j S_{ji} + Rand([0,1])$.
- Define the initial set of F-points: $F = \{i \in \Omega \mid \sum_j S_{ji} = 0\}$.
- Define the initial set of C-points: $C = \emptyset$.
- Remove the F-points from the remaining point set: $\Omega' = \Omega \backslash F$.
- While $\Omega' \neq \emptyset$ do:
  - Choose an independent set Υ of Ω': $i \in \Upsilon$ iff $\lambda_i > \lambda_j \; \forall \; j : S_{ij} \neq 0$ or $S_{ji} \neq 0$.
  - Make all elements of Υ C-points: $C = C \bigcup \Upsilon$.
  - Make all elements of $\Omega' \backslash \Upsilon$ that are strongly influenced by a new C-point, F-points: $F = F \bigcup F_{new}$, where $F_{new} = \{j \in \Omega' \backslash \Upsilon \mid \exists \; i \in \Upsilon : i \in S_j\}$.
  - Remove all new C- and F-points from Ω': $\Omega' = \Omega' \backslash \{\Upsilon \bigcup F_{new}\}$.

  End do.

An example of the PMIS coarsening algorithm is illustrated in Figure 4.4 for the 2D 5-point Laplace problem (the model problem from Section 1.1) on a structured grid.

Unlike RS coarsening, PMIS is not sequential in nature, and can be easily implemented in parallel with minimal processor boundary communication. This translates into positive benefits for execution time per V-cycle. De Sterck, Yang, and Heys [18]

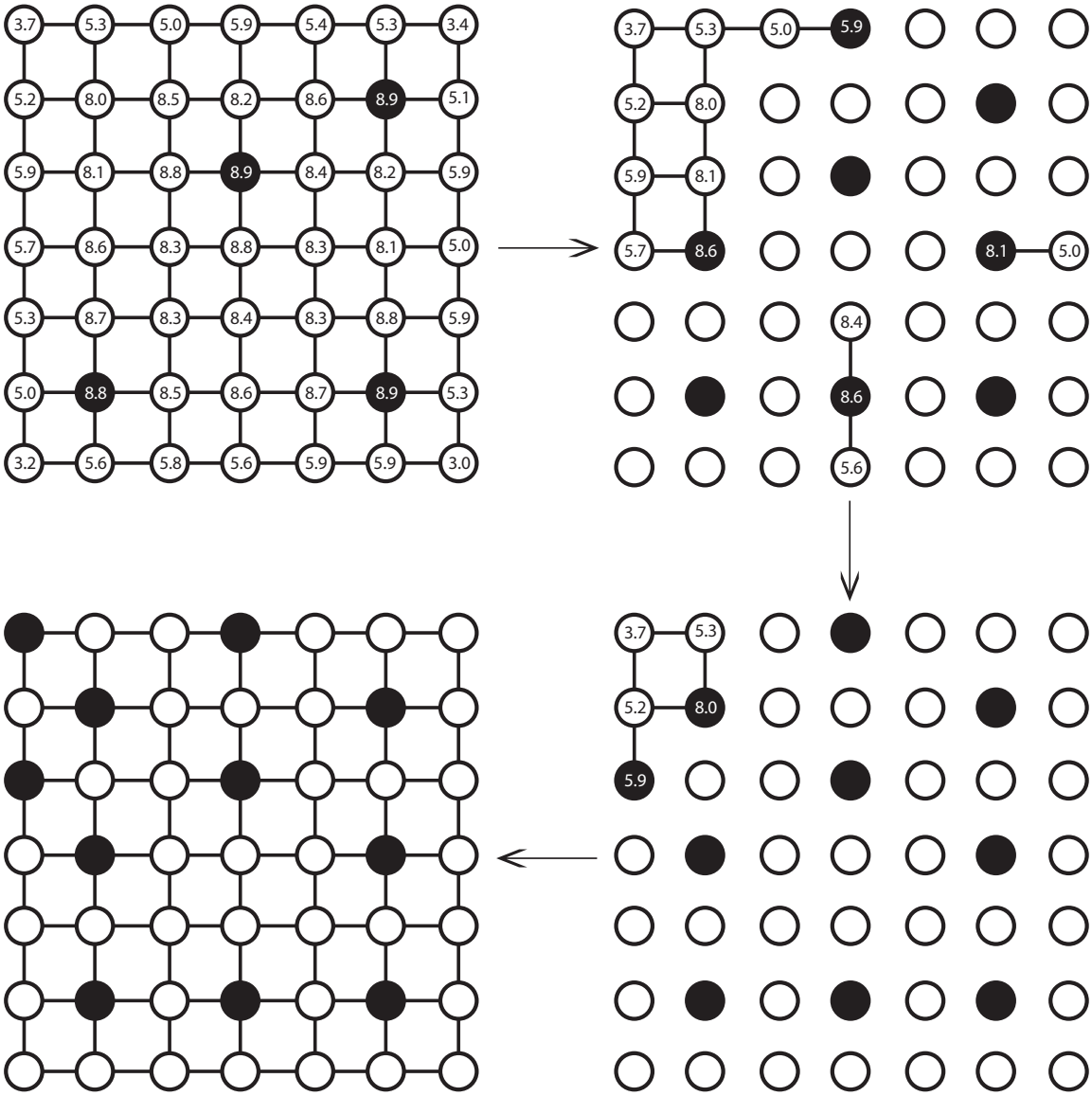

Figure 4.4: PMIS coarsening for the 2D 5-point Laplace operator, (adapted from [18]).

show that PMIS coarsening works well for many problems when the resulting AMG method is used as a preconditioner for GMRES. This thesis also shows that PMIS can cause AMG to perform well as a stand-alone algorithm for many problems if interpolation is modified to take into account certain *distance-two* C-points (see Chapters 5 and 7).

As illustrated in Figure 4.4, PMIS coarsening permits strong F-F connections without a common C-point. This has the benefit of lower grid and operator complexities (nearly optimal with regard to scalability); however, as shown in [18], convergence can be degraded for some problems compared to schemes based on RS coarsening. This is due to the fact that an inadequate amount of C-points and their location reduces the accuracy of interpolation. Thus, PMIS coarsening exhibits an opposite effect to that observed for RS coarsening; that is, low computational cost but poor convergence. Therefore, PMIS coarsening leads to AMG algorithms that are not scalable. This thesis will present several modifications to current uses of PMIS that aim to overcome this problem.

4.4.3 CLJP Coarsening

This section presents the Cleary-Luby-Jones-Plassman (CLJP) coarsening algorithm. This coarsening scheme was proposed by Cleary, and is based on parallel graph partitioning algorithms introduced by Luby, and developed by Jones and Plassman [10].

The behaviour of the CLJP algorithm resembles that of both RS and PMIS coarsening. Like PMIS, CLJP is highly parallel, but CLJP also exhibits similar convergence scalability as RS. As a result, CLJP can be used to replace RS in parallel implementations. Unfortunately, CLJP can lead to operator complexities and execution times that are not scalable, and that may be significantly higher than those of RS.

The first phase of CLJP is similar to that of PMIS. An initial measure λ_i is assigned to all points, along with a random number, $Rand([0,1])$, to break ties between all points that have maximal measure. Points whose measure is greater than that of all of their strongly influencing *and* strongly dependent neighbours ($\lambda_i > \lambda_k \; \forall \; k \in S_i \bigcup S_i^T$) are declared as C-points. CLJP then adopts a similar approach to RS by enforcing

two heuristic criteria that attempt to achieve a balance between good convergence and minimal computational cost per V-cycle. These heuristics may be stated as follows (where a lower case 'h' is used to distinguish them from **H-1** and **H-2** for RS coarsening):

h-1: Since C-points are interpolated directly to the fine-grid (this is explained in Section 4.5), their strongly influencing neighbours are less valuable as potential C-points.

h-2: If a C-point i strongly influences two points k and j, and j strongly influences k, then j is less valuable as a potential C-point since k can be interpolated from i.

Heuristics **h-1** and **h-2** represent attempts to achieve efficient interpolation. Heuristic **h-1** ensures that C-points are not declared where their benefit to interpolation would be small, and **h-2** is in place in an attempt to make interpolation at F-points sufficiently accurate for good convergence, while minimizing computational cost per V-cycle. From a practical perspective, these heuristics can be implemented as follows [10]:

```
for each new C−point, i
    for each j that strongly influences i
        decrement λ_j
        set S_ij ← 0
    for each j that strongly depends on i
        set S_ji ← 0
        for each k that strongly depends on j
            if k strongly depends on i
                decrement λ_j
                set S_kj ← 0
```

Decrementing the measure of point j decreases its potential to become a C-point subject to **h-1** and **h-2**. The appropriate entries in S are set to zero to indicate that a strong influence has been considered (since S is used to determine new C-points). Whenever the decrementing of λ_j results in $\lambda_j < 1$, point j is declared as an F-point. The entire process is then repeated until all points have been declared as C- or F-points. An example of CLJP coarsening is illustrated in Figure 4.5 for the 2D 9-point Laplace problem (which will be introduced in Chapter 6). Note that setting an entry $S_{ij} \leftarrow 0$ when applying the heuristics corresponds to removing edge ij from the graph in Figure 4.5. Also note that F-F connections without a common C-point are not possible with CLJP. This is due to the design of the method, about which more information can be found in [10].

It should be noted that, like the second pass of RS, the computational cost of implementing **h-1** and **h-2** can be high, and is increased in parallel due to the added communication required between processors when λ values are changed. Furthermore, since all $j \in S_i$ for a new C-point i are not immediately declared as F-points, the number of C-points on each grid is significantly larger than that obtained with RS coarsening [7]. While this is beneficial for convergence, it can lead to large operator complexities that negatively affect AMG performance and scalability. Nevertheless, since CLJP is highly parallel and generally exhibits good convergence, it is used to compare with the performance of other algorithms in Chapter 7.

4.5 Interpolation

This section presents two types of interpolation known as classical interpolation and F-F interpolation. This entails defining the interpolation operator that is used to transfer functions from coarse to fine grids. The goal is to define interpolation such that smooth functions are interpolated accurately. Since smooth error varies slowly in the direction of strong connection, interpolation will be most effective if strong connections can be taken into account appropriately.

Before proceeding with the classical and F-F interpolation formulations, some general terminology common to all AMG interpolation methods needs to be introduced.

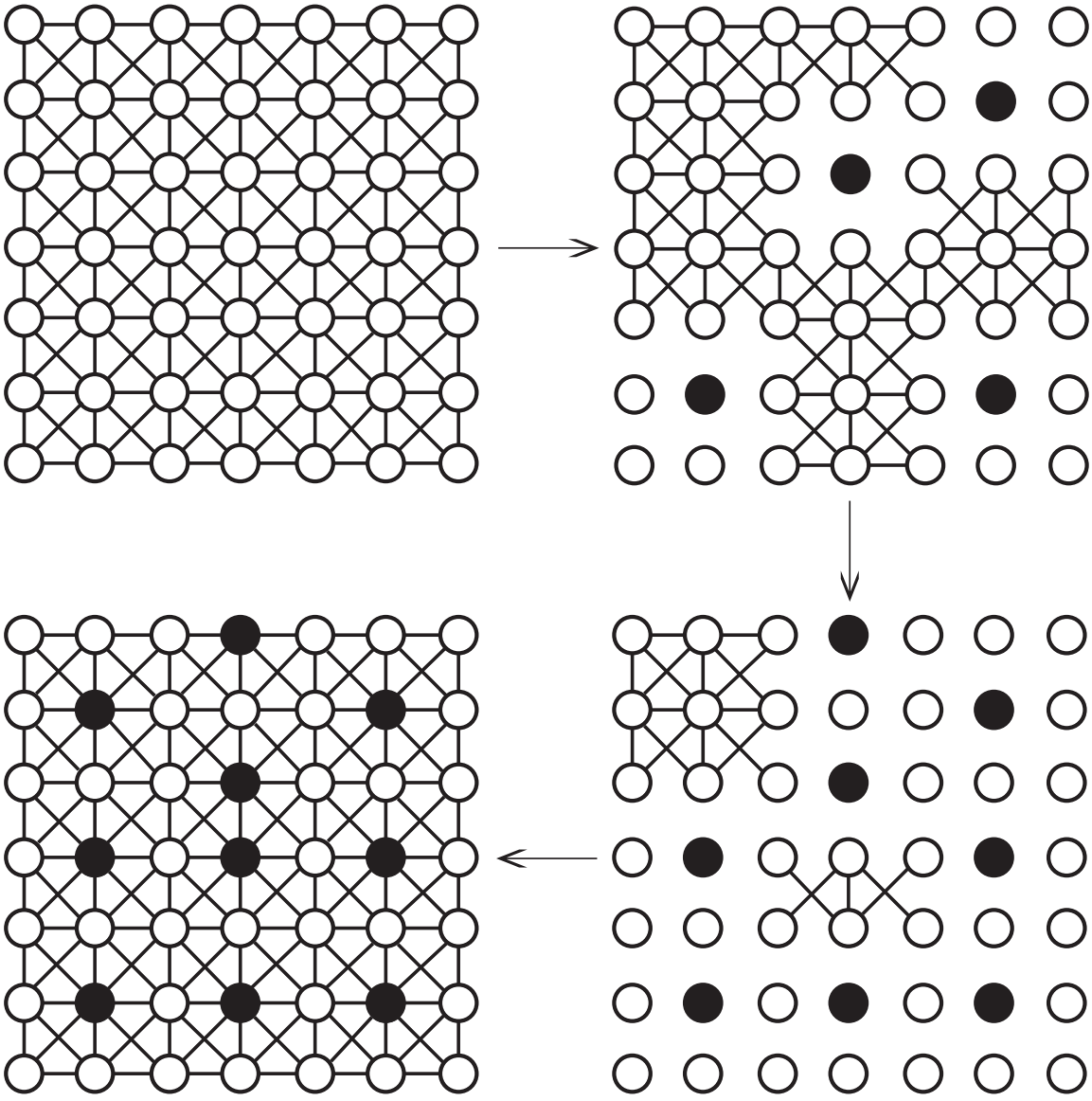

Figure 4.5: CLJP coarsening for the 2D 9-point Laplace operator, (from [10]).

As in [4], the neighbourhood N_i of each fine-grid point i can be divided into three subsets, namely,

- the neighbouring C-points that strongly influence i, called the *coarse interpolatory set* for i, and denoted by C_i;
- the neighbouring F-points that strongly influence i, denoted by D_i^s; and
- the points that do not strongly influence i, denoted by D_i^w. This set is called the set of *weakly connected neighbours*, and may contain both C- and F-points.

The material is now in place to precisely define the interpolation operator, I_{2h}^h.

4.5.1 Classical Interpolation

Classical interpolation has been a standard component in many AMG algorithms, and is an effective method for a variety of problems. This section will present classical interpolation following the treatment done in [4].

As mentioned in Section 4.4.1, error values at C-points are known from computations on the coarse grid, and can therefore be interpolated directly to the fine grid. It would not make sense to try and improve the approximation in a C-point, i, by interpolating from its strongly influencing neighbours, since these neighbours have an uncertainty that is similar in magnitude to the error in i. The question still remains however, as to how exactly one should interpolate error values at F-points using the coarse-grid error. The i^{th} component of the interpolated error, $I_{2h}^h\mathbf{e}$, is given by

$$\left(I_{2h}^h\mathbf{e}\right)_i = \begin{cases} e_i \ , & if \quad i \in C; \\ \sum_{j\in C_i} \omega_{ij} e_j \ , & if \quad i \in F. \end{cases} \tag{4.5.1}$$

Therefore, the only task that remains is to define the interpolation weights, ω_{ij}, for all $i \in F$. This can be accomplished by first recalling that, according to (4.2.5), e_i can be approximated well by a weighted average of the errors of its neighbours. Rewriting

this weighted average using Definition 4.4.2, and expanding N_i into its component sets, gives:

$$a_{ii}e_i \approx -\sum_{j\in N_i} a_{ij}e_j, \tag{4.5.2}$$

$$a_{ii}e_i \approx -\sum_{j\in C_i} a_{ij}e_j - \sum_{j\in D_i^s} a_{ij}e_j - \sum_{j\in D_i^w} a_{ij}e_j. \tag{4.5.3}$$

In order to determine the weights, ω_{ij}, it follows that the component sums must be expressed either in terms of e_i, or in terms of e_j where $j \in C_i$, in accordance with (4.5.1). For the sum over weakly connected points, redistribution to the diagonal is permissible, such that (4.5.3) becomes

$$\left(a_{ii} + \sum_{j\in D_i^w} a_{ij}\right) e_i \approx -\sum_{j\in C_i} a_{ij}e_j - \sum_{j\in D_i^s} a_{ij}e_j. \tag{4.5.4}$$

This step is justified by considering the nature of the sum over the weakly connected points. If it turns out that an error was made in defining the strength threshold θ, and that some $j \in D_i^w$ should in fact be considered to strongly influence i, then, since error varies slowly in the direction of strong connection, the error made in approximating e_j by e_i is small, and the transfer to the diagonal is acceptable. If however point j is only weakly connected to point i, as assumed, then the coefficient a_{ij} is in fact small, and the error generated in multiplying a_{ij} by e_i instead of e_j will be relatively insignificant.

According to [4], in the case of the sum over D_i^s, experience has shown that it is better to approximate the e_j's with weighted sums of e_k for $k \in C_i \bigcap C_j$, rather than to distribute the terms to the diagonal. This can be done by taking a linear combination of values of e_k that are in $C_i \bigcap C_j$. Therefore, replacing the e_j with the e_k corresponds to taking into account strong F-F connections using C-points that are common between the F-points (as described in Section 4.4.1 and illustrated in Figure (4.1)). Because smooth error varies slowly in the direction of strong connection, the error introduced in making this approximation is relatively insignificant. It also follows that, again since smooth error varies slowly in the direction of strong connection,

the e_j are strongly influenced by the e_k in proportion to the matrix entries a_{jk}. Thus, an appropriate approximation may be defined as

$$e_j \approx \frac{\sum_{k \in C_i} a_{jk} e_k}{\sum_{k \in C_i} a_{jk}}. \tag{4.5.5}$$

The denominator simply ensures that constants are interpolated exactly. It is important to note that in order for (4.5.5) to be well-defined, at least one C-point is required to be in both C_i and C_j; otherwise, the coefficients a_{jk} are small or vanishing, and e_j can not be approximated well by this method. Since RS coarsening always ensures that all strong F-F connections have a common C-point, this does not present a problem. However, for PMIS coarsening, F-F connections without a common C-point are permitted, and interpolation for these points must be handled differently. In this case, the appropriate terms of the sum over D_i^s may be redistributed to the diagonal [4] in a manner similar to that done for the weak connections. It will be assumed that this approach is taken whenever classical interpolation is used on PMIS coarsened grids further on in this thesis.

Equation (4.5.5) may be substituted into (4.5.4) to obtain

$$\left(a_{ii} + \sum_{j \in D_i^w} a_{ij} \right) e_i \approx - \sum_{j \in C_i} a_{ij} e_j - \sum_{j \in D_i^s} a_{ij} \frac{\sum_{k \in C_i} a_{jk} e_k}{\sum_{k \in C_i} a_{jk}}. \tag{4.5.6}$$

Then, since $j,\ k\ \in C_i$, and treating the denominator of the fraction as a constant, (4.5.6) can be rewritten as:

$$\left(a_{ii} + \sum_{j \in D_i^w} a_{ij} \right) e_i \approx - \sum_{j \in C_i} \left(a_{ij} + \sum_{m \in D_i^s} \frac{a_{im} a_{mk}}{\sum_{k \in C_i} a_{mk}} \right) e_j. \tag{4.5.7}$$

However, e_i represents the i^{th} component of the interpolated error. Therefore, the interpolation weights follow directly as:

$$\omega_{ij} = - \frac{a_{ij} + \sum_{m \in D_i^s} \left(\frac{a_{im} a_{mj}}{\sum_{k \in C_i} a_{mk}} \right)}{a_{ii} + \sum_{n \in D_i^w} a_{in}}. \tag{4.5.8}$$

The interpolation/prolongation operator is now completely defined by (4.5.1) and (4.5.8) for classical interpolation.

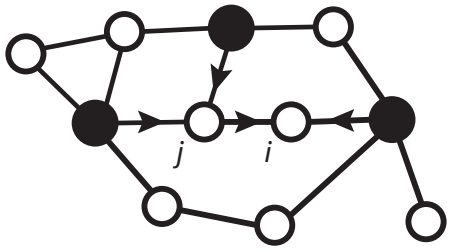

Figure 4.6: Example indicating points used in *F*-*F* interpolation to construct the interpolation formula for i – a point that has a strong *F*-*F* connection with point j, but no common *C*-point.

4.5.2 F-F Interpolation

The definition of classical interpolation in Section 4.5.1 required that all strong *F*-*F* connections have a common *C*-point. While (4.5.8) can be modified to handle these *F*-*F* connections by redistributing them to the diagonal, this method is not guaranteed to be sufficiently accurate. Another method for handling strong *F*-*F* connections without a common *C*-point is to use so-called *F*-*F* interpolation. This section defines and discusses *F*-*F* interpolation.

F-*F* interpolation was introduced by De Sterck and Yang in [17], and is similar to Stüben's standard interpolation [20]. It defines I_{2h}^h in exactly the same way as classical interpolation, except when strong *F*-*F* connections without a common *C*-point are encountered. When this occurs, *F*-*F* interpolation extends interpolation to *distance-two* *C*-points. For strongly influencing *F*-points j that do not have a common *C*-point with *F*-point i, the coarse interpolatory set C_i is extended to C_i^*, which additionally contains all *C*-points that strongly influence j.

For instance, consider the example in Figure 4.2. Rather than adding *C*-points to accommodate classical interpolation, *F*-*F* interpolation takes the approach illustrated in Figure 4.6 for point i. Strongly connected *C*-points are handled in the same way as for classical interpolation. However, when treating the strongly connected *F*-point j, which does not have a common *C*-point with i, all $k \in C_j$ are now included in the extended coarse interpolatory set, C_i^*. This is justified by the same argument as was

done for strong F-F connections with a common C-point in classical interpolation. Since error varies slowly along the direction of strong connection, e_j can be approximated by a weighted average of the e_k, and the weights will be proportional to a_{jk}. Therefore, when a strong F-F connection without a common C-point is encountered, (4.5.5) is replaced by

$$e_j \approx \frac{\sum_{k \in C_j} a_{jk} e_k}{\sum_{k \in C_j} a_{jk}}. \tag{4.5.9}$$

Then for an F-point i, letting D_i^{s*} represent the set of strongly connected F-points without a common C-point, and redefining D_i^s to represent the set of strongly connected F-points with a common C-point, (4.5.6) can be rewritten as

$$\left(a_{ii} + \sum_{j \in D_i^w} a_{ij} \right) e_i \approx - \sum_{j \in C_i^*} a_{ij} e_j - \sum_{j \in D_i^s} a_{ij} \frac{\sum_{k \in C_i^*} a_{jk} e_k}{\sum_{k \in C_i^*} a_{jk}} - \sum_{j \in D_i^{s*}} a_{ij} \frac{\sum_{k \in C_i^*} a_{jk} e_k}{\sum_{k \in C_i^*} a_{jk}}. \tag{4.5.10}$$

Taking the same approach that was used in the section on classical interpolation, it follows that the interpolation weights are given by

$$\omega_{ij} = - \frac{a_{ij} + \sum_{m \in D_i^s \bigcup D_i^{s*}} \left(\frac{a_{im} a_{mj}}{\sum_{k \in C_i^*} a_{mk}} \right)}{a_{ii} + \sum_{n \in D_i^w} a_{in}}. \tag{4.5.11}$$

Having completed the formulation, it can be concluded that F-F interpolation avoids the need for a second pass in RS and the added work in CLJP, and consequently promotes the use of PMIS which gives lower grid and operator complexities. However, an increase in the number of C-points considered in interpolation also produces more entries in the interpolation operator, which requires an increase in storage compared to classical interpolation. A denser interpolation operator in turn produces more entries in the coarse-grid operator which is directly derived from I_{2h}^h (this is covered in the next section), and thus computational costs are increased compared to classical interpolation. In addition, a considerable amount of work may be required by F-F interpolation to construct I_{2h}^h, since many *distance-two* C-points for all strong F-F connections may have to be considered. As will be shown in Chapter 7, F-F interpolation combined with PMIS coarsening may actually improve AMG scalability

compared to CLJP coarsening with classical interpolation; however, this is still not to a desirable level due to the larger coarse interpolatory set. This thesis will examine a version of *F-F* interpolation that seeks to avoid these complications, and would thus allow the advantages of PMIS and other coarsening algorithms that permit strong *F-F* connections without a common *C*-point to be exploited. This will be called *F-F1* interpolation and will be presented in Chapter 5.

4.6 Variational Properties

This section explicitly defines the restriction and coarse-grid operators on all grids. As is common in AMG, this will be done using what are known as the *variational properties*. Each of the two variational properties will be presented along with a theoretical justification of its use.

Before introducing the variational properties, two necessary assumptions must be stated. Firstly, all interpolation operators, $I_{2h}^{h} \in \mathbb{R}^{M \times N}$ with $M > N$, are assumed to have full rank (i.e. N linearly independent vectors form the columnspace of I_{2h}^{h}). This property will be shown to be necessary in order that the variational properties can be extended recursively to apply to all grid levels of a V-cycle. It can be noted that interpolation as defined in Section 4.5 is always guaranteed to have full rank. This is due to the fact that coarse-grid error is interpolated exactly to the fine grid. For example, consider the following interpolation operator that is used to transfer a vector-function from a coarse grid with three points (rows 1, 3, and 5) to a fine grid with five:

$$I_{2h}^{h} = \begin{bmatrix} 1 & 0 & 0 \\ \omega_{2,1} & \omega_{2,3} & \omega_{2,5} \\ 0 & 1 & 0 \\ \omega_{4,1} & \omega_{4,3} & \omega_{4,5} \\ 0 & 0 & 1 \end{bmatrix}.$$

Even if the weights, $\omega_{i,j}$, used to interpolate the *F*-points are considered arbitrary, the reduced-row-echelon form of I_{2h}^{h} is guaranteed to have three leading-ones – meaning

that I_{2h}^{h} has full rank. Since, in this thesis, interpolation is always defined such that the error of coarse-grid points is interpolated exactly on the fine grid, the interpolation operator is always guaranteed to have full rank. The second assumption is that the fine-grid operator, A^h, is symmetric and positive definite.

The restriction operator, I_h^{2h}, and coarse-grid operator, A^{2h} can now be defined, respectively, as follows:

$$A^{2h} = I_h^{2h} A^h I_{2h}^h, \tag{4.6.1}$$

$$I_h^{2h} = \left(I_{2h}^h\right)^T. \tag{4.6.2}$$

Equations (4.6.1) and (4.6.2) are called the *variational properties*, and will now be justified. Note that each of the variational properties will be used in motivating the definition of the other.

By defining restriction to be the transpose of interpolation as in (4.6.2), it follows that, as long as I_{2h}^h is full rank, A^{2h} will be symmetric and positive definite. This is observed using the *Euclidean* inner product as defined by (A.0.6), such that [20]

$$\begin{aligned}\left(A^{2h}\mathbf{v}^{2h}, \mathbf{v}^{2h}\right)_E &= \left(I_h^{2h} A^h I_{2h}^h \mathbf{v}^{2h}, \mathbf{v}^{2h}\right)_E \\ &= \left(A^h I_{2h}^h \mathbf{v}^{2h}, I_{2h}^h \mathbf{v}^{2h}\right)_E \\ &= \left(\mathbf{v}^{2h}, A^{2h}\mathbf{v}^{2h}\right)_E. \end{aligned} \tag{4.6.3}$$

This is a useful fact. Only the fine-grid operator was assumed to be symmetric and positive definite, but since the coarse-grid operator is also, any analysis applicable to a two-grid scheme can simply be recursively extended to accommodate a V-cycle. Defining restriction in the form (4.6.2) also has computational benefits. Only the interpolation operator needs to be stored, and the restriction operator can be easily obtained.

Turning now to the justification of (4.6.1), which is also referred to as the *Galerkin condition*, it is first useful to restate the two-grid correction scheme presented in Section 3.2.4 more concisely. An exact solve is assumed on the coarse grid, and to economize on notation, the right-hand side of the residual equation is called $\mathbf{f}^{2h}$ rather

than $\mathbf{r}^{2h}$, and the solution of the residual equation is called $\mathbf{u}^{2h}$ instead of $\mathbf{e}^{2h}$ (as is done for the V-cycle scheme in Section 3.2.4). The AMG two-grid correction scheme can then be written as follows [4]:

- Relax ν_1 times on Ω^h with scheme (error propagation matrix) R: $\mathbf{v}^h \leftarrow R^{\nu_1}\mathbf{v}^h + C(\mathbf{f})$.
- Restrict $\mathbf{r}^h$ to Ω^{2h}: $\mathbf{f}^{2h} \leftarrow I_h^{2h}\left(\mathbf{f}^h - A^h\mathbf{v}^h\right)$.
- Solve the residual equation exactly: $\mathbf{v}^{2h} = \left(A^{2h}\right)^{-1}\mathbf{f}^{2h}$.
- Correct the approximation on Ω^h: $\mathbf{v}^h \leftarrow \mathbf{v}^h + I_{2h}^h\mathbf{v}^{2h}$.
- Relax ν_2 times on Ω^h with scheme (error propagation matrix) R: $\mathbf{v}^h \leftarrow R^{\nu_2}\mathbf{v}^h + C(\mathbf{f})$.

Ignoring the relaxation steps for the moment, the two-grid correction scheme may be represented by the following operation:

$$\mathbf{v}^h \leftarrow \mathbf{v}^h + I_{2h}^h\left(A^{2h}\right)^{-1} I_h^{2h}\left(\mathbf{f}^h - A^h\mathbf{v}^h\right). \tag{4.6.4}$$

It can also be noted that the exact solution is unchanged by the two-grid correction scheme, such that

$$\mathbf{u}^h = \mathbf{u}^h + I_{2h}^h\left(A^{2h}\right)^{-1} I_h^{2h}\left(\mathbf{f}^h - A^h\mathbf{u}^h\right). \tag{4.6.5}$$

Thus, (4.6.4) can be subtracted from (4.6.5) to give

$$\mathbf{e}^h \leftarrow K_{op}\mathbf{e}^h \ , \quad with\ K_{op} \equiv I - I_{2h}^h\left(A^{2h}\right)^{-1} I_h^{2h}A^h. \tag{4.6.6}$$

This defines what is known as the *coarse-grid correction operator*, K_{op}. Including the pre- and post-relaxation steps, the two-grid correction scheme may be summarized as follows:

$$\mathbf{e}^h \leftarrow M_{op}\mathbf{e}^h \ , \quad with\ M_{op} \equiv R^{\nu_2}K_{op}R^{\nu_1}, \tag{4.6.7}$$

where M_{op} is called the *two-grid correction operator*.

With this material in place, the advantage of defining the coarse-grid operator A_{2h} by (4.6.1) can be expressed in the following theorem. Note that $\Re(X)$ is used to denote the *range* of a matrix operator X.

Theorem 4.6.1. [20] Let A^h be symmetric positive definite, let a coarse grid exist that is an arbitrary subset of fine-grid points, and let interpolation be defined in any way such that I_{2h}^h has full rank. Then, the coarse-grid correction operator K_{op} is an orthogonal projector with respect to the A-inner product. In particular, it is true that:

1. $\Re(K_{op}) \perp_A \Re(I_{2h}^h)$, i.e. $(A^h K_{op}\mathbf{x}^h, I_{2h}^h \mathbf{y}^h)_E = 0 \; \forall \; \mathbf{x}^h \in \Re(K_{op})$ *and* $\mathbf{y}^h \in \Re(I_{2h}^h)$.
2. For $\mathbf{x}^h \in \Re(K_{op})$ and $\mathbf{y}^h \in \Re(I_{2h}^h)$, it follows that $\|\mathbf{x}^h + \mathbf{y}^h\|_A^2 = \|\mathbf{x}^h\|_A^2 + \|\mathbf{y}^h\|_A^2$.
3. $\|K_{op}\|_A = 1$.
4. For all $\mathbf{e}^h$: $\|K_{op}\mathbf{e}^h\|_A = min_{\mathbf{e}^{2h}} \|\mathbf{e}^h - I_{2h}^h \mathbf{e}^{2h}\|_A$.

The last statement of this theorem illustrates precisely why it is best to define the coarse-grid operator according to the Galerkin condition. It states that, regardless of the coarsening and (full rank) interpolation methods used, the Galerkin coarse-grid correction operator is guaranteed to minimize the error over the space of all possible coarse-grid error functions, and this is optimal.

Theorem 4.6.1 also indicates why it is useful to define the coarse-grid operator by the Galerkin condition in terms of the convergence properties of the two-grid correction scheme. In particular, the third statement says that the two-grid correction scheme, (4.6.7), can never diverge as long as $\|R\| \leq 1$. This important result can be extended to consider complete V-cycles by recursive application of the following lemma. Again, it is assumed that a coarse grid exists that is an arbitrary subset of fine-grid points, and that interpolation is defined in any way such that I_{2h}^h has full rank.

Lemma 4.6.2. [20] Let the exact coarse-grid error (i.e. the error just prior to interpolation to Ω^h), $\mathbf{e}^{2h}$, implied in (4.6.6), be replaced by any approximation $\tilde{\mathbf{e}}^{2h}$ satisfying $\|\mathbf{e}^{2h} - \tilde{\mathbf{e}}^{2h}\|_{A^{2h}} \leq \|\mathbf{e}^{2h}\|_{A^{2h}}$. Then the approximate two-grid correction operator still satisfies $\|\tilde{K}_{op}\|_A \leq 1$.

Therefore, as long as a relaxation method is chosen such that $\|R\| \leq 1$, $\mathbf{e}^{2h}$ will satisfy the condition of Lemma 4.6.2, and the V-cycle iteration is guaranteed not to diverge.

Chapter 5

AMG Modifications

As indicated in Chapter 4, and shown by De Sterck, Yang, and Heys in [18], the PMIS coarsening algorithm works well for many problems when used as a preconditioner for GMRES, but causes convergence to degrade for other problems compared to CLJP coarsening. The reason for this is attributed to a lack of accuracy in interpolation due to a decreased number of C-points generated in the coarsening process compared to CLJP coarsening. One solution to this problem is simply to add C-points, as is done in the second pass of the RS algorithm. However, this can lead to high grid and operator complexities that drastically affect computational costs. The process of adding C-points can also be time consuming. This chapter presents three modifications to current AMG components that are new in this thesis, and that aim to improve the convergence properties of PMIS-coarsened AMG without adversely affecting computational costs. These include a greedy implementation of the PMIS coarsening algorithm, combining PMIS and CLJP in the coarsening process such that PMIS is used on finer grid levels and CLJP is used on coarser grid levels, and a modification to the F-F interpolation scheme. The motivation for each of these changes is also described.

5.1 Modification 1: PMIS Greedy

As mentioned before, one of the shortcomings of the PMIS coarsening method is its inability to generate an adequate number of C-points for accurate interpolation.

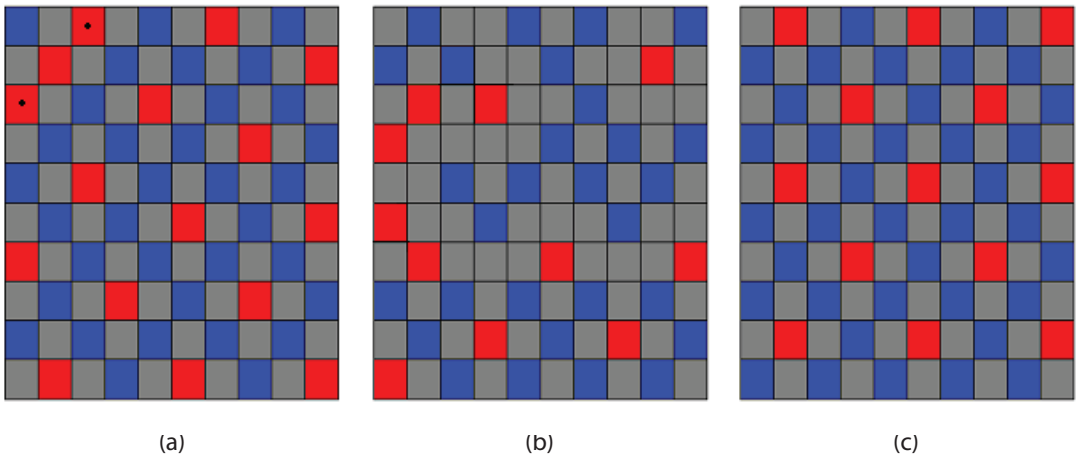

Figure 5.1: Sequence of coarsened grids for the 5-pt 2D Laplace problem on a structured grid with a) RS coarsening, b) PMIS coarsening, and c) PMIS greedy coarsening.

While inaccurate interpolation is partly caused by the low number of C-points that are produced, it is also due their arrangement. In particular, this problem is caused by strongly connected F-points without a common connected C-point which are not effectively accounted for in the classical interpolation formula. For example, consider the grids illustrated in Figure 5.1, where grey represents the finest grid, blue the second finest grid, and red the coarsest grid, and all points are assumed to both strongly depend on and strongly influence their adjacent neighbours. Figure 5.1 (a) illustrates that the grid produced by RS coarsening is very structured; however, the effect of the second pass is directly evident by the points marked with a dot. A CLJP coarsened grid would have a similar appearance. While RS and CLJP coarsening produce structured grids that are beneficial for accurate interpolation, they also require an increase in setup time, storage cost, and operator complexity that can be prohibitively high. Figure 5.1 (b) illustrates the result obtained using PMIS coarsening. While this is a rather extreme example, the lack of grid structure is evident, and there is a predominance of strong F-F connections without a common C-point. It would be advantageous if the PMIS coarsening algorithm could be modified in some

way to produce grids that are consistently more structured and allow for improved interpolation. This is the motivation for the PMIS greedy coarsening algorithm.

In the RS coarsening algorithm, after an F-point is defined, the measure of all of its strongly influencing neighbours is increased. This is done to increase the likelihood that these neighbours are selected as C-points. If selected, they will cause the F-point to have a greater number of strongly influencing C-points, and therefore greater accuracy in interpolation. Also, this method usually produces grids that are highly structured due to the sequential nature of the algorithm, and due to the updating of weights of neighbours of neighbours. Often, newly assigned points *spread out* from the first assigned C-point due to the fact that the measures of neighbours of new F-points are updated at each step. In contrast, the PMIS algorithm simply selects points whose measure is greater than that of all of their strongly influencing and dependent neighbours, while breaking ties randomly. It may happen that a PMIS coarsened grid possesses a large amount of structure, but this is much less likely than in the RS case. To attempt to remedy this problem, one can add the structure producing step of RS to PMIS. That is, one can update the measure of all strongly influencing neighbours of newly defined F-points. In this way, the *spreading-out* effect exhibited by RS will be present (at least to some extent) in PMIS, and F-points will hopefully have a better set of strongly influencing C-points to interpolate from. The resulting version of PMIS will be called PMIS greedy. The term "greedy" is used to indicate the fact that the method adopts the meta-heuristic of making the locally optimal choice at each stage with the hope of finding the global optimum [24].
The PMIS greedy algorithm may be summarized as follows:

- Perform PMIS coarsening as defined in Section 4.4.2, but after a point is defined as an F-point, increment the measure of all unassigned points that strongly influence that F-point.

By this method, coarsened grids will hopefully be more structured, and consequently may allow for better interpolation and AMG convergence. As indicated in Figure 5.1 (c), the PMIS greedy algorithm can improve grid structure considerably compared to PMIS. Also, compared to the grid generated using RS coarsening, the PMIS greedy

grid has fewer C-points, and therefore produces a lower grid complexity. While Figure 5.1 indicates that PMIS greedy coarsening is promising, the question still remains as to how it will perform on larger and more difficult problems. In particular, the question remains as to whether the extra work contributed to updating measures will in fact pay off in terms of improved convergence behaviour. These questions will be investigated numerically for a variety of problems in Chapter 7.

5.2 Modification 2: Restrict PMIS to Finer Grid Levels

This section proposes another method that aims to remedy the convergence problems sometimes caused by the PMIS coarsening algorithm when combined with classical interpolation. As indicated in Section 4.4, implementing the heuristic criteria for CLJP can be computationally intensive. Therefore, PMIS coarsening has an advantage over CLJP in terms of speed. It was also stated that CLJP coarsening can result in high operator complexities, but generally produces good convergence properties. On the other hand, PMIS generally produces lower operator complexities than CLJP, but can suffer from diminished convergence. In this sense, PMIS and CLJP coarsening exhibit an opposite trade-off between computational cost and convergence that prevent both from achieving good scalability. This section presents a modification that seeks to take advantage of the strengths of both methods; that is, the structure and convergence properties demonstrated by CLJP, and the lower storage and computational cost per V-cycle incurred by PMIS. In this way, the resulting AMG algorithm will hopefully be more scalable.

Consider the coarsening process of geometric multigrid for a symmetric problem on a 3D structured grid. If coarse grids are chosen by selecting half the points in each dimension from the fine grid, and coarse-grid operators are simply defined to be the coarse-grid version of the fine-grid operator, then the operator complexity may

be defined by the following geometric series:

$$C_{op} = 1 + \frac{1}{8} + \frac{1}{64} + \ldots < \frac{8}{7}. \tag{5.2.1}$$

While this calculation is easily performed for this example with geometric multigrid, the same reasoning can be applied to AMG for any problem. The critical observation is that C_{op} can be expressed as a series of fractions of the number of nonzero entries in the fine-grid operator, and that these fractions necessarily decrease as grids become coarser. Therefore, to most effectively reduce operator complexity, it makes sense to try and reduce the number of nonzero entries in the coarse-grid operator on finer grids, since these contribute the largest values to the operator complexity as illustrated by (5.2.1). The aim of Modification 2 is to use PMIS coarsening only on finer grid levels, where its effect on reducing operator complexity will be most valuable, and to use CLJP coarsening on coarser grid levels. Consequently, the increase in operator complexity compared to pure PMIS will be minimal since it will occur on coarser grid levels, and yet the better convergence properties of CLJP compared to PMIS may still be exploited on coarser grids.

In summary, for Modification 2, PMIS coarsening is performed on the first g grid levels, and CLJP coarsening is performed on all remaining levels. In this way, PMIS coarsening reduces operator complexity compared to CLJP on the finest grid levels where it makes the biggest difference, and CLJP produces more *structured* grids with only a small number of strong F-F connections without a common C-point – and therefore better interpolation and convergence – on coarser grid levels where the impact on operator complexity is reduced. While this method appears to take advantage of the strengths of both PMIS and CLJP, it is uncertain at this point how the method will cause the AMG algorithm to perform. Also, the number of PMIS coarsened grids that should be obtained before switching to CLJP coarsening still needs to be determined. These issues are investigated numerically for a variety of problems in Chapter 7.

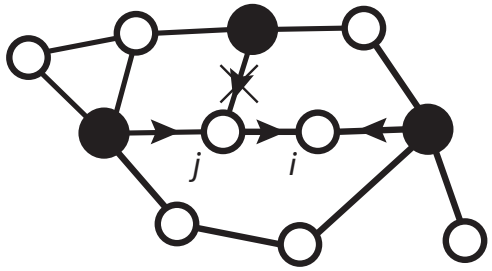

Figure 5.2: Example indicating points used in *F-F1* interpolation to construct the interpolation formula for i – a point that has a strong *F-F* connection with point j, but no common *C*-point.

5.3 Modification 3: *F-F1* Interpolation

A third modification of the AMG algorithm is proposed in this section. In this case the change does not involve the coarsening procedure, but instead targets interpolation. In Chapter 7 it will be shown that, although PMIS coarsening combined with *F-F* interpolation can reduce operator complexity while achieving similar convergence compared to CLJP coarsening with classical interpolation, the computational benefit is often not substantial. When implementing *F-F* interpolation, considerable work may be required to account for all *distance-two C*-points for strong *F-F* connections without a common *C*-point. Also, because a large number of additional *C*-points often need to be considered, the resulting operator complexities (although better than CLJP) may still be undesirable. If the amount of work could somehow be reduced while also reducing operator complexity and retaining good convergence, the true benefits of PMIS coarsening (fast execution, highly parallel, tendency for low operator complexities) could be exploited while also achieving good scalability from AMG. This section presents a modification that aims to do just that, and it will be referred to as *F-F1* interpolation.

The main difficulty with *F-F* interpolation is that all *distance-two C*-points for strong *F-F* connections without a common *C*-point are considered. *F-F1* interpolation is defined in the same way as *F-F* interpolation, except that it only considers the

first *distance-two* *C*-point encountered for each strong *F*-*F* connection. In this way, operator complexity and the amount of work required to generate the interpolation operator may be substantially reduced. At this point it is uncertain what the impact will be on the convergence properties of AMG; however, this will be investigated using numerical examples in Chapter 7. *F*-*F1* interpolation is depicted in Figure 5.2 for the example from Section 4.5.2. The arrow with an 'X' through it is meant to illustrate that a *C*-point is used in *F*-*F* interpolation, but not in *F*-*F1* interpolation.

Chapter 6

Model Problems

This chapter introduces all of the model problems used to generate the results in Chapter 7. Each PDE problem is presented separately along with its finite difference discretization. First, however, some elaboration is needed on the use of the finite difference method. This preliminary discussion will expand on the material covered in Section 1.1, and can be applied to all of the model problems presented in this chapter.

6.1 Background: More on Finite Differences

The explanation of the finite difference method in Section 1.1 was limited in its breadth. This section will clarify and expand on the details presented earlier by illustrating different ways of discretizing first and second order derivatives for PDEs in two dimensions on structured grids. The resulting expressions may be easily extended to accommodate problems of dimension greater than two, and are relevant to all of the model problems that follow.

Consider any two-dimensional problem that contains both x and y second derivatives, u_{xx} and u_{yy}. As in Section 1.1, these derivatives can be discretized by considering appropriate combinations of Taylor series in two variables. For an interior grid point on a structured grid, this can be done in a variety of ways. In Section 1.1, only the points north, south, east, and west of (x_i, y_j) were used to obtain an

approximation for a derivative at a point (x_i, y_j). These points are illustrated in Figure 6.1 (a), where a line indicates that a point is to be used in the approximation at point (x_i, y_j). This produced equations (1.1.3) and (1.1.4), which are rewritten here for completeness:

$$u_{xx}(x_i, y_j) \approx \frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)}{h_x^2}, \tag{6.1.1}$$

$$u_{yy}(x_i, y_j) \approx \frac{u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1})}{h_y^2}. \tag{6.1.2}$$

Both (6.1.1) and (6.1.2) are second order accurate, and together form what is known as a *5-point discretization*. This is because a total of five different points are involved in approximating the x and y second derivatives at each interior grid point. If instead all nine neighbouring grid points of (x_i, y_j) are used in the approximation, as indicated in Figure 6.1 (b), approximations of the second derivatives can be written as:

$$\begin{aligned} u_{xx}(x_i, y_j) \approx \frac{1}{3h_x^2}[&u(x_{i+1}, y_j) + \frac{1}{2}u(x_{i+1}, y_{j+1}) + \frac{1}{2}u(x_{i+1}, y_{j-1}) - 4u(x_i, y_j) \\ &+ u(x_{i-1}, y_j) + \frac{1}{2}u(x_{i-1}, y_{j+1}) + \frac{1}{2}u(x_{i-1}, y_{j-1})], \end{aligned} \tag{6.1.3}$$

$$\begin{aligned} u_{yy}(x_i, y_j) \approx \frac{1}{3h_y^2}[&u(x_i, y_{j+1}) + \frac{1}{2}u(x_{i+1}, y_{j+1}) + \frac{1}{2}u(x_{i-1}, y_{j+1}) - 4u(x_i, y_j) \\ &+ u(x_i, y_{j-1}) + \frac{1}{2}u(x_{i+1}, y_{j-1}) + \frac{1}{2}u(x_{i-1}, y_{j-1})]. \end{aligned} \tag{6.1.4}$$

Both (6.1.3) and (6.1.4) are second order accurate, and together form what is known as a *9-point discretization*, since a total of nine different points are used to approximate the x and y second derivatives at each interior grid point. A similar extension to three dimensions can be made to obtain the standard *7-point* and *27-point* discretizations in the same form as (6.1.1) and (6.1.2), and (6.1.3) and (6.1.4), respectively. This is left to the reader.

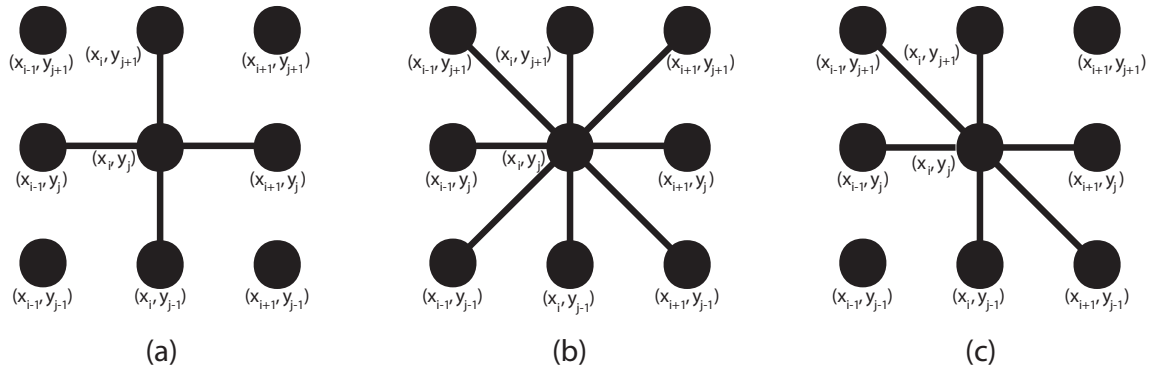

Figure 6.1: Points on a 2D structured grid used in (a) a 5-point discretization, (b) a 9-point discretization, and (c) a 4-point mixed discretization.

The mixed term u_{xy} (which will appear in the context of the 2D rotated anisotropic Laplace problem) is most naturally discretized using the *left-oriented 7-point discretization* [20]. Using Taylor series expansions of the points indicated in Figure 6.1 (c), the left-oriented 7-point discretization is given by

$$u_{xy}(x_i, y_j) \approx \frac{1}{2h_x h_y}[u(x_{i+1}, y_j) + u(x_{i-1}, y_j) + u(x_i, y_{j+1}) + u(x_i, y_{j-1}) - 2u(x_i, y_j) - u(x_{i+1}, y_{j-1}) - u(x_{i-1}, y_{j+1})], \tag{6.1.5}$$

which is second order accurate.

Finally, first order terms such as u_x (which will appear in the context of the 3D convection-diffusion problem) will be treated using standard *upwind discretizations*, like

$$u_x(x_i, y_j) \approx \frac{u(x_i, y_j) - u(x_{i-1}, y_j)}{h_x}, \tag{6.1.6}$$

which are first order accurate. The upwind discretization is used, rather than the *central difference discretization* (which is second order accurate), due to stability considerations. The reader is referred to [21] for more information on the use of upwind discretizations.

6.2 Model Problems

This section presents separately, all of the model problems used to generate the results in Chapter 7. This includes giving a description of the continuous PDE problem, followed by a discussion of how the problem was discretized for the work in this thesis. The discretizations are presented in the context of the components introduced in Section 6.1 for an interior grid point. Structured grids are considered, and it is assumed that grid spacing is the same in all dimensions; that is, $h_x = h_y = h_z = h$. Definition of domains, boundary conditions, and right-hand sides is left until Chapter 7.

The model problems are presented in an order that, historically, has represented an increase in difficulty for algebraic multigrid algorithms. Often an algorithm will perform well for simple problems, but eventually fails when applied to more difficult ones. Therefore, these problems will help to evaluate the robustness of the modifications proposed in Chapter 5. In addition, this set of problems is relatively standard in the multigrid community. Consequently, the results of this thesis can be compared with other work.

6.2.1 5-Point (2D) Laplace Problem

The 2D Laplace problem,

$$-u_{xx} - u_{yy} = f(x, y), \tag{6.2.1}$$

is discretized using the 5-point discretization, such that

$$\frac{1}{h^2}[-u(x_{i+1}, y_j) - u(x_{i-1}, y_j) - u(x_i, y_{j+1}) - u(x_i, y_{j-1}) + 4u(x_i, y_j)] = f(x_i, y_j). \tag{6.2.2}$$

6.2.2 9-Point (2D) Laplace Problem

The 2D Laplace problem,

$$-u_{xx} - u_{yy} = f(x, y), \tag{6.2.3}$$

is discretized using the 9-point discretization, such that

$$\frac{1}{3h^2}[-u(x_{i+1},y_j)-u(x_{i+1},y_{j+1})-u(x_{i-1},y_j)-u(x_{i-1},y_{j-1})$$
$$-u(x_i,y_{j+1})-u(x_{i-1},y_{j+1})-u(x_i,y_{j-1})-u(x_{i+1},y_{j-1})+8u(x_i,y_j)]=f(x_i,y_j). \tag{6.2.4}$$

6.2.3 7-Point (3D) Laplace Problem

The 3D Laplace problem,

$$-u_{xx}-u_{yy}-u_{zz}=f(x,y,z), \tag{6.2.5}$$

is discretized using the 7-point discretization, such that

$$\frac{1}{h^2}[-u(x_{i+1},y_j,z_k)-u(x_{i-1},y_j,z_k)-u(x_i,y_{j+1},z_k)-u(x_i,y_{j-1},z_k)$$
$$-u(x_i,y_j,z_{k+1})-u(x_i,y_j,z_{k-1})+6u(x_i,y_j,z_k)]=f(x_i,y_j,z_k). \tag{6.2.6}$$

6.2.4 27-Point (3D) Laplace Problem

The 3D Laplace problem,

$$-u_{xx}-u_{yy}-u_{zz}=f(x,y,z), \tag{6.2.7}$$

is discretized using the 27-point discretization, such that

$$\frac{1}{9h^2}[-u(x_{i+1},y_j,z_k)-u(x_{i-1},y_j,z_k)-u(x_i,y_{j+1},z_k)-u(x_i,y_{j-1},z_k)$$
$$-u(x_i,y_j,z_{k+1})-u(x_i,y_j,z_{k-1})-u(x_{i+1},y_{j+1},z_k)-u(x_{i-1},y_{j-1},z_k)$$
$$-u(x_{i+1},y_{j-1},z_k)-u(x_{i-1},y_{j+1},z_k)-u(x_{i+1},y_j,z_{k+1})-u(x_{i-1},y_j,z_{k-1})$$
$$-u(x_{i+1},y_j,z_{k-1})-u(x_{i-1},y_j,z_{k+1})-u(x_i,y_{j+1},z_{k+1})-u(x_i,y_{j-1},z_{k-1})$$
$$-u(x_i,y_{j+1},z_{k-1})-u(x_i,y_{j-1},z_{k+1})-u(x_{i+1},y_{j+1},z_{k+1})-u(x_{i-1},y_{j-1},z_{k-1})$$
$$-u(x_{i+1},y_{j-1},z_{k-1})-u(x_{i+1},y_{j+1},z_{k-1})-u(x_{i+1},y_{j-1},z_{k+1})-u(x_{i-1},y_{j+1},z_{k+1})$$
$$-u(x_{i-1},y_{j-1},z_{k+1})-u(x_{i-1},y_{j+1},z_{k-1})+26u(x_i,y_j,z_k)]=f(x_i,y_j,z_k). \tag{6.2.8}$$

6.2.5 3D Anisotropic Laplace Problem

The 3D anisotropic Laplace problem,

$$-cu_{xx} - u_{yy} - u_{zz} = f(x, y, z), \tag{6.2.9}$$

where $c \in \mathbb{R}$, is discretized using the 7-point discretization, such that

$$\begin{aligned}\frac{1}{h^2}[&- c\{u(x_{i+1}, y_j, z_k) + u(x_{i-1}, y_j, z_k)\} - u(x_i, y_{j+1}, z_k) - u(x_i, y_{j-1}, z_k) \\ &- u(x_i, y_j, z_{k+1}) - u(x_i, y_j, z_{k-1}) + (2c + 4)u(x_i, y_j, z_k)] = f(x_i, y_j, z_k).\end{aligned} \tag{6.2.10}$$

6.2.6 3D Convection-Diffusion Problem

The 3D convection-diffusion problem,

$$-c_x u_{xx} - c_y u_{yy} - c_z u_{zz} + a_x u_x + a_y u_y + a_z u_z = f(x, y, z), \tag{6.2.11}$$

where $\{c_x, c_y, c_z, a_x, a_y, a_z\} \in \mathbb{R}$, is discretized using the 7-point and upwind discretizations, such that

$$\begin{aligned}\frac{1}{h^2}[&- c_x\{u(x_{i+1}, y_j, z_k) + u(x_{i-1}, y_j, z_k)\} - c_y\{u(x_i, y_{j+1}, z_k) + u(x_i, y_{j-1}, z_k)\} \\ &- c_z\{u(x_i, y_j, z_{k+1}) + u(x_i, y_j, z_{k-1})\} + (2c_x + 2c_y + 2c_z)u(x_i, y_j, z_k)] \\ &+ \frac{1}{h}[a_x\{u(x_i, y_j, z_k) - u(x_{i-1}, y_j, z_k)\} + a_y\{u(x_i, y_j, z_k) - u(x_i, y_{j-1}, z_k)\} \\ &\quad + a_z\{u(x_i, y_j, z_k) - u(x_i, y_j, z_{k-1})\}] = f(x_i, y_j, z_k).\end{aligned} \tag{6.2.12}$$

6.2.7 2D Rotated Anisotropic Laplace Problem

The 2D rotated anisotropic Laplace problem,

$$-(c^2 + \epsilon s^2)u_{xx} + 2(1 - \epsilon)csu_{xy} - (s^2 + \epsilon c^2)u_{yy} = f(x, y), \tag{6.2.13}$$

where $\epsilon \in \mathbb{R}$, $c = \cos\gamma$, $s = \sin\gamma$, and γ is the angle of rotation, is discretized using the 5-point and left-oriented 7-point discretizations, such that

$$\begin{aligned}
&\frac{1}{h^2}[(c^2+\epsilon s^2)\{-u(x_{i+1},y_j)+2u(x_i,y_j)-u(x_{i-1},y_j)\} \\
&\quad+(1-\epsilon)cs\{u(x_{i+1},y_j)+u(x_{i-1},y_j)+u(x_i,y_{j+1})+u(x_i,y_{j-1}) \\
&\qquad\qquad -2u(x_i,y_j)-u(x_{i+1},y_{j-1})-u(x_{i-1},y_{j+1})\} \\
&\quad+(s^2+\epsilon c^2)\{-u(x_i,y_{j+1})+2u(x_i,y_j)-u(x_i,y_{j-1})\}] = f(x_i,y_j). \qquad (6.2.14)
\end{aligned}$$

6.2.8 3D Elliptic PDE with Jumps

The 3D elliptic PDE with jumps in the coefficients,

$$-(au_x)_x-(au_y)_y-(au_z)_z = f(x,y,z), \qquad (6.2.15)$$

is discretized using the 7-point discretization, such that

$$\begin{aligned}
&\frac{1}{h^2}[-a(x_{i+\frac{1}{2}},y_j,z_k)u(x_{i+1},y_j,z_k)-a(x_{i-\frac{1}{2}},y_j,z_k)u(x_{i-1},y_j,z_k) \\
&\quad-a(x_i,y_{j+\frac{1}{2}},z_k)u(x_i,y_{j+1},z_k)-a(x_i,y_{j-\frac{1}{2}},z_k)u(x_i,y_{j-1},z_k) \\
&\quad-a(x_i,y_j,z_{k+\frac{1}{2}})-u(x_i,y_j,z_{k+1})-a(x_i,y_j,z_{k-\frac{1}{2}})u(x_i,y_j,z_{k-1}) \\
&\quad+\{a(x_{i+\frac{1}{2}},y_j,z_k)+a(x_{i-\frac{1}{2}},y_j,z_k)+a(x_i,y_{j+\frac{1}{2}},z_k) \\
&\qquad+a(x_i,y_{j-\frac{1}{2}},z_k)+a(x_i,y_j,z_{k+\frac{1}{2}})+a(x_i,y_j,z_{k-\frac{1}{2}})\}u(x_i,y_j,z_k)] \\
&\quad= f(x_i,y_j,z_k). \qquad (6.2.16)
\end{aligned}$$

6.3 Multigrid Performance

This section discusses the performance characteristics of existing geometric and algebraic multigrid algorithms for the model problems in Section 6.2. This includes summarizing traditional modifications to the geometric multigrid algorithm for problems in which difficulties arise.

The model problems in Section 6.2 can be divided into four categories. These are *standard Laplace problems* (model problems 6.2.1 through 6.2.4), *anisotropic problems* (model problems 6.2.5 and 6.2.7), *convection-diffusion problems* (model problem 6.2.6), and *discontinuous coefficient problems* (model problem 6.2.8). Standard geometric multigrid algorithms perform well for standard Laplace problems (see for example [4, 21]); however, efficiency can be significantly degraded for the other three types of problems. For anisotropic problems, it follows that standard pointwise relaxation methods such as GS do not effectively reduce error in the direction of anisotropy since they only smooth error in the direction of strong coupling [21]. One solution to this problem is to use *block relaxation*, in which a block of unknowns is updated simultaneously (for example, a line in 2D or a plane in 3D). Another solution is to keep pointwise relaxation, but modify the coarsening method according to the problem. One such approach is to use *semicoarsening*, in which coarsening is performed only along the direction of strong coupling. In this way, pointwise relaxation can still be effective. More information on block relaxation and semicoarsening can be found in [4, 21]. Further difficulties arise when the anisotropy is not aligned with the grid, as is true for the rotated anisotropic problem. One method for overcoming this problem is to use a more robust smoother such as *modified ILU*, an "ILU-type" (incomplete LU matrix decomposition) smoother, which also reduces certain low frequency error components on the fine grid [21].

The ideas of semicoarsening and block relaxation can also be applied to convection-diffusion problems. For these problems, geometric multigrid performance can be improved by using *higher order upwind-biased discretizations* combined with *KAPPA smoothers* or *multistage Jacobi smoothers* [21].

Interpolation in standard geometric multigrid relies on the continuity of ∇u. Consequently, for problems with discontinuous coefficients, the algorithm must be modified accordingly. This can be accomplished using *operator-dependent interpolation* (which resembles classical interpolation in AMG) and the Galerkin condition to define the coarse-grid operator. Discussion of this method and several others that apply to problems with discontinuous coefficients may be found in [21].

Standard AMG (RS coarsening with classical interpolation) generally performs well for all four types of problems in terms of convergence scalability. Problem-dependent modifications, like those discussed above for geometric multigrid, are usually not required for AMG since it automatically performs coarsening and interpolation in the direction of strong connection. As a result, AMG automatically semi-coarsens and uses operator-dependent interpolation, and in this sense is a more robust algorithm than geometric multigrid. However, complexity problems often occur for AMG, especially for 3D problems. These complexity issues may lead to a significant loss of scalability for the AMG algorithm, which causes existing AMG methods to be inefficient for large problem sizes – in particular on parallel computers. The modifications proposed in Chapter 5 are aimed at improving AMG efficiency for these cases. Their performance for the model problems introduced in Section 6.2 will be evaluated in Chapter 7.

Chapter 7

Numerical Results

This chapter presents a numerical comparison of standard AMG coarsening and interpolation methods with the modified methods as described in Chapter 5. The standard methods include CLJP coarsening with classical interpolation, PMIS coarsening with classical interpolation, and PMIS coarsening with *F-F* interpolation. The modifications include PMIS greedy coarsening with classical interpolation, restricting PMIS to finer grid levels while CLJP is performed only on coarser levels with classical interpolation, and PMIS coarsening with *F-F1* interpolation. This study will be performed for all eight model problems introduced in Chapter 6, and will include results for both stand-alone AMG and GMRES(5)-accelerated AMG (AMG-GMRES(5)).[1] GMRES(5) was used for all problems, rather than using CG for the symmetric problems and GMRES(5) for the nonsymmetric problems, so that algorithms could be consistently compared.

In the case of the second modification (PMIS on the first g finest grids, and CLJP on all remaining grids), tests were performed for $g = 1, 2, 3, 4$, and 5. For simplicity, only the value of g that exhibited the best result in terms of a combination of operator complexity, convergence, and execution time is presented for each of the model problems. Since it was found that the algorithm with $g = 1$ always converges in the fewest number of iterations, it follows that if the algorithm listed has $g \neq 1$, then it executed faster and had a lower operator complexity than the algorithm with

[1] Choosing to restart GMRES after precisely five iterations was somewhat of an arbitrary choice, and is not necessarily the optimal choice.

$g = 1$. A more detailed analysis of modification 2 is presented in Section 7.2. This includes discussion of an interesting trade-off that was observed between operator complexity, convergence, and execution speed.

Tests were done on a serial computer with 2 GB of memory and a Pentium 4 processor with a speed of 3.2 GHz using Hypre. Hypre is an optimized AMG code developed by the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory in Livermore, California, and can be obtained from http://www.llnl.gov/CASC/linear_solvers/. Unless otherwise stated, the following parameters were used for all runs:

- Cycle type: V(1,1)
- Relaxation method: Gauss-Seidel, in CF order on the finest grid and on the downward part of the V-cycle (except for the coarsest grid, for which the coarsest-grid solve method is employed), and in FC order on the upward part of the V-cycle (except for the finest grid).[2]
- Strength threshold: $\alpha = 0.25$
- Maximum size of the coarsest level: 9
- Coarsest-grid solve method: Gaussian elimination
- Convergence tolerance (relative residual using the L^2 norm)[3]: 1×10^{-6}
- Initial guess on the finest level: $\mathbf{v}^{(0)} = \mathbf{0}$

The legend for all tables is as follows:

- CLJP: CLJP coarsening with classical interpolation

[2] *CF order* means to relax on all variables that are *C*-points, and then on all variables that are *F*-points (*FC order* is the reverse). GS relaxation in CF order is an efficient smoother in practice, and is related to *red-black Gauss-Seidel relaxation* in geometric multigrid [20]. More information on Gauss-Seidel in CF order and red-black Gauss-Seidel relaxation can be found in [20] and [4, 21], respectively.

[3] The relative residual is defined to be the norm of the residual divided by the norm of the right-hand side; that is, $\frac{\|\mathbf{r}\|}{\|\mathbf{f}\|}$.

- PMIS: PMIS coarsening with classical interpolation
- PMIS greedy: PMIS greedy coarsening with classical interpolation
- PMIS(g)-CLJP: PMIS coarsening performed on the first g finest grids and CLJP coarsening on all remaining grids, with classical interpolation
- PMIS-FF: PMIS coarsening with *F-F* interpolation
- PMIS-FF1: PMIS coarsening with *F-F1* interpolation
- n: problem size per dimension
- C_{op}: operator complexity
- $\#lev$: total number of grid levels in the V-cycle
- $iter$: number of iterations required to reach the convergence tolerance
- t_{setup}: time required for the setup phase (s)
- t_{solve}: time required for the solve phase (includes the GMRES(5) component for AMG-GMRES(5)) (s)
- t_{tot}: total run time (setup time + solve time) (s)

For all 2D and 3D problems, the domain was chosen to be the unit square and the unit cube, respectively. Dirichlet boundary conditions ($\mathbf{u} = \mathbf{0}$) were enforced for all problems, and discretization of the continuous problems was performed according to the methods described in Section 6.2.

It is generally observed that PMIS coarsened AMG methods are scalable with respect to storage cost since C_{op} normally does not increase with respect to problem size asymptotically. Any discussion of scalability for a PMIS-coarsened algorithm will assume this to be true, and therefore only applies to the convergence criteria. Unless otherwise stated, scalability of storage cost will be implied.

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP	2.21	8	2	*small*	*small*	*small*
PMIS	1.76	7	4	*small*	*small*	*small*
PMIS greedy	1.76	7	4	*small*	*small*	*small*
PMIS(5)-CLJP	1.76	7	4	*small*	*small*	*small*
PMIS-FF	1.77	7	4	*small*	*small*	*small*
PMIS-FF1	1.77	7	4	*small*	*small*	*small*

Table 7.1: AMG on a 1024×1024 structured grid for the 5-point Laplace problem.

7.1 Results and Discussion

7.1.1 5-Point (2D) Laplace Problem

The 2D Laplace problem,

$$-u_{xx} - u_{yy} = 1, \tag{7.1.1}$$

was considered using a 5-point discretization. As none of the three proposed modifications showed a marked improvement over current methods, only the data for the largest problem size tested is included here. This is presented in Table 7.1 for AMG on a 1024×1024 grid. A time value of 'small' is meant to indicate that a value close to zero was obtained, but that it could not be accurately determined. Results for AMG-GMRES(5) were exactly the same for this problem size, and showed no improvement of scalability over AMG for all problem sizes considered. As such, the AMG-GMRES(5) results will not be displayed. Consequently, it can be said that there is no advantage to accelerating AMG with GMRES for this problem for the problem sizes tested.

As expected, PMIS reduces operator complexity compared to CLJP, but requires more iterations to converge. As illustrated in Table 7.1 – in terms of operator complexity, convergence, and execution time – all three modifications basically exhibit the same behaviour as the PMIS algorithm. As such, there is no advantage to any of the proposed modifications for this simple problem compared to PMIS, at least for the problem sizes that were investigated. It should be noted that AMG with CLJP

coarsening is essentially optimal for the 5-point Laplace problem, so one should not expect to observe much, if any, improvement in comparison. It was also noted that the number of iterations required to reach the convergence criteria was almost constant as a function of problem size for all methods, and that C_{op} only increased linearly with respect to problem size for CLJP. Consequently, all methods can be considered scalable for this problem.

In summary, CLJP performs slightly better than all other methods in terms of convergence, and slightly worse in terms of operator complexity; however, scalability of all methods was found to be the same. None of the proposed modifications improve convergence or reduce operator complexity for this problem compared to PMIS.

7.1.2 9-Point (2D) Laplace Problem

The 2D Laplace problem,

$$-u_{xx} - u_{yy} = 1, \tag{7.1.2}$$

was considered using a 9-point discretization. Results for the largest problem size tested are included in Tables 7.2 and 7.3 for AMG and AMG-GMRES(5), respectively. A scalability study for several of the algorithms is also presented in Figure 7.1, where number of iterations is plotted as a function of problem size.

Table 7.2 illustrates that, while PMIS reduces operator complexity compared to CLJP, convergence is significantly degraded due to inaccurate interpolation. This is directly evidenced through the number of iterations required to reach the convergence criteria, which also negatively affects the corresponding solve times. AMG-GMRES(5) does accelerate convergence, but not to a satisfactory level.

The effect of the three proposed modifications is analyzed as follows.

1 The PMIS greedy modification improves both AMG and AMG-GMRES(5) in terms of convergence, and with only a slight increase in operator complexity compared to PMIS; however, this improvement is not sufficient, and still results in poor scalability properties as illustrated in Figure 7.1.

2 Performing PMIS coarsening on only the first grid and CLJP on all remaining grids does appear to have a positive effect. Execution time is comparable to that of

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP	1.92	12	18	10.65	11.31	21.96
PMIS	1.24	9	197	4.59	88.97	93.56
PMIS greedy	1.26	9	153	4.72	70.07	74.79
PMIS(1)-CLJP	1.40	12	33	5.62	16.19	21.81
PMIS-FF	1.45	8	16	7.78	8.05	15.83
PMIS-FF1	1.41	9	19	7.10	9.16	16.26

Table 7.2: AMG on a 1024×1024 structured grid for the 9-point Laplace problem.

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP	1.92	12	10	10.59	9.41	20.00
PMIS	1.24	9	46	4.66	32.99	37.65
PMIS greedy	1.26	9	43	4.73	31.03	35.76
PMIS(1)-CLJP	1.40	12	18	5.65	13.91	19.56
PMIS-FF	1.45	8	10	7.90	7.76	15.66
PMIS-FF1	1.41	9	11	7.26	8.67	15.93

Table 7.3: AMG-GMRES(5) on a 1024×1024 structured grid for the 9-point Laplace problem.

CLJP while operator complexity is considerably reduced. While this has a benefit in terms of storage, it is noted that many more iterations are needed for convergence compared to CLJP. This is less explicit for AMG-GMRES(5), but still undesirable. As illustrated in Figure 7.1, scalability for both AMG and AMG-GMRES(5) with modification 2, although significantly better than PMIS, is still not ideal for this problem.

3 The most notable improvement is observed for PMIS coarsening combined with *F-F* and *F-F1* interpolation. For both AMG and AMG-GMRES(5), convergence is comparable to that of CLJP with classical interpolation, but significant time is saved in both the setup and solve phases. Operator complexity is significantly reduced compared to that of CLJP, and the extra time spent in defining the interpolation operator far outweighs the penalty paid in convergence when using PMIS with classical interpolation. It can also be noted for this problem that, compared to *F-F*, the lack of

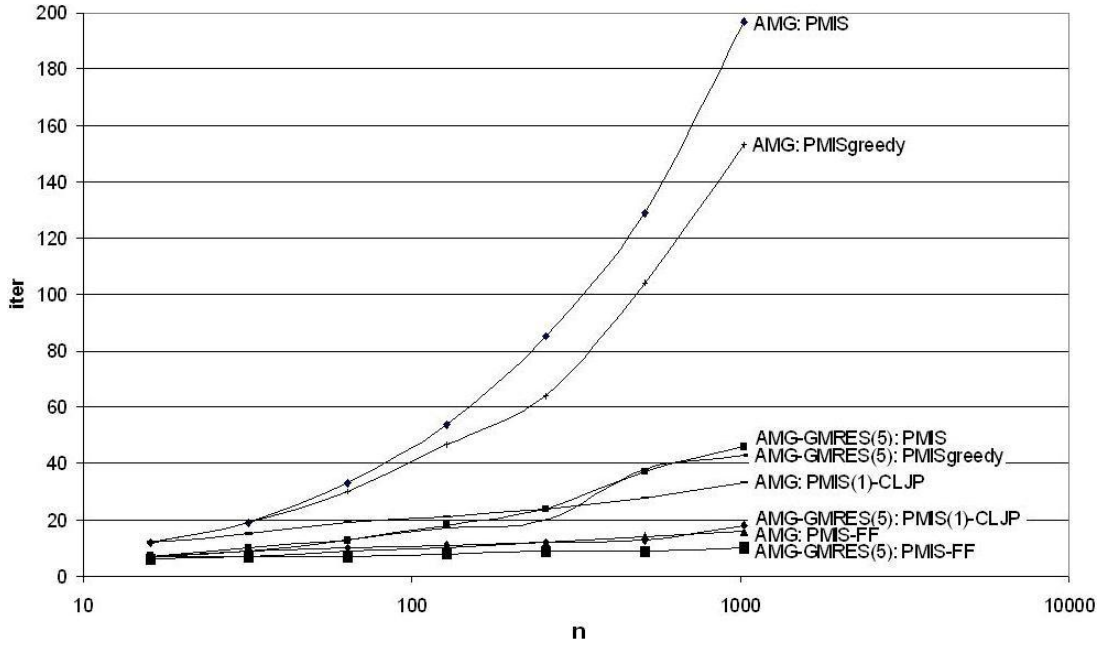

Figure 7.1: Scalability comparison for the 9-point Laplace problem.

accuracy in interpolation with *F-F1* resulted in poorer convergence, and only a slight decrease in operator complexity. As such, there does not appear to be an advantage to using *F-F1* interpolation, and PMIS coarsening with *F-F* interpolation seems to be the best choice of all algorithms tested for solving the 9-point Laplace problem. Note that the convergence results of CLJP and PMIS-FF1 were similar to those obtained with PMIS-FF for both AMG and AMG-GMRES(5), respectively, and are not shown in Figure 7.1.

7.1.3 7-Point (3D) Laplace Problem

The 3D Laplace problem,

$$-u_{xx} - u_{yy} - u_{zz} = 1, \tag{7.1.3}$$

was considered using a 7-point discretization. Results for AMG and GMRES(5)-accelerated AMG are shown for a $128 \times 128 \times 128$ grid in Tables 7.4 and 7.5, respectively. A scalability study for several of the algorithms is also presented in Figure 7.2, where number of iterations is plotted as a function of problem size.

It should be noted that, not only did the CLJP test run out of memory on the 128^3 grid, it was also about three times slower than AMG with PMIS, and about four times slower than AMG-GMRES(5) with PMIS, on a 64^3 grid. In addition, operator complexity was about ten times smaller for PMIS coarsening compared to that obtained using CLJP. Thus, even without any of the proposed modifications, PMIS coarsening appears to be a much better choice than CLJP for this problem with respect to execution time and storage cost. It was observed, however, that CLJP converged in significantly fewer iterations than PMIS, and is more scalable with respect to convergence.

1 Convergence with PMIS greedy coarsening is negligibly improved compared to PMIS for AMG, and is unchanged for AMG-GMRES(5). Also, as expected, PMIS greedy operator complexity is higher than that obtained using PMIS. As such, there is little or no advantage to using PMIS greedy instead of PMIS for this problem. This would also be true for parallel implementations, since an added communication cost would exist for PMIS greedy (the measure of unassigned neighbours of F-points must be incremented on adjacent processors), and would likely cause any advantage to be completely lost.

2 As illustrated in Tables 7.4 and 7.5, PMIS(3)-CLJP considerably improves execution time compared to PMIS for AMG, but the improvement is only small for AMG-GMRES(5). It should be noted that for both AMG and AMG-GMRES(5), PMIS(1)-CLJP and PMIS(2)-CLJP actually converge in fewer iterations than PMIS(3)-CLJP, but require larger setup times since more CLJP coarsening is performed. In addition, operator complexity is much higher for PMIS(1)-CLJP and PMIS(2)-CLJP compared to PMIS(3)-CLJP. For this reason, PMIS(3)-CLJP is considered the best performer out of all PMIS(g)-CLJP variants tested. As illustrated in Figure 7.2, PMIS(3)-CLJP shows an improvement over PMIS in terms of convergence scalability for both AMG and AMG-GMRES(5). Since the operator complexity generated using PMIS(3)-CLJP

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP (64^3)	22.51	14	9	20.36	5.83	26.19
PMIS	2.36	8	77	16.63	85.93	102.56
PMIS greedy	2.44	9	74	17.53	84.57	102.10
PMIS(3)-CLJP	2.49	12	47	17.70	53.82	71.52
PMIS-FF	4.80	8	13	83.81	22.86	106.67
PMIS-FF1	3.68	8	15	44.22	22.07	66.29

Table 7.4: AMG on a $128 \times 128 \times 128$ structured grid for the 7-point Laplace problem. CLJP results are for the 64^3 problem because the 128^3 test ran out memory.

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP (64^3)	22.51	14	5	20.32	4.23	24.55
PMIS	2.36	8	20	16.67	35.26	51.93
PMIS greedy	2.44	9	20	17.54	35.77	53.31
PMIS(3)-CLJP	2.49	12	16	17.70	29.89	47.59
PMIS-FF	4.80	8	9	83.87	22.97	106.84
PMIS-FF1	3.68	8	9	43.85	19.85	63.70

Table 7.5: AMG-GMRES(5) on a $128 \times 128 \times 128$ structured grid for the 7-point Laplace problem. CLJP results are for the 64^3 problem because the 128^3 test ran out memory.

is not much higher than that obtained using only PMIS coarsening, this modification does show a promising improvement in overall scalability.

3 For PMIS-FF and PMIS-FF1, a considerable reduction in the number of iterations required for convergence is observed for both AMG and AMG-GMRES(5). It is noted, however, that there is a significant increase in operator complexity for PMIS-FF compared to PMIS, with no saving in execution time for either AMG or AMG-GMRES(5). As evidenced by high setup times, this is due to the fact that a large number of *distance-two C*-points need to be considered in the interpolation procedure. *F-F1* interpolation does well to reduce operator complexity compared to *F-F*, and also demonstrates a significant time saving over the *F-F* runs. PMIS-FF1 shows the best overall improvement in execution time over PMIS with classical interpolation

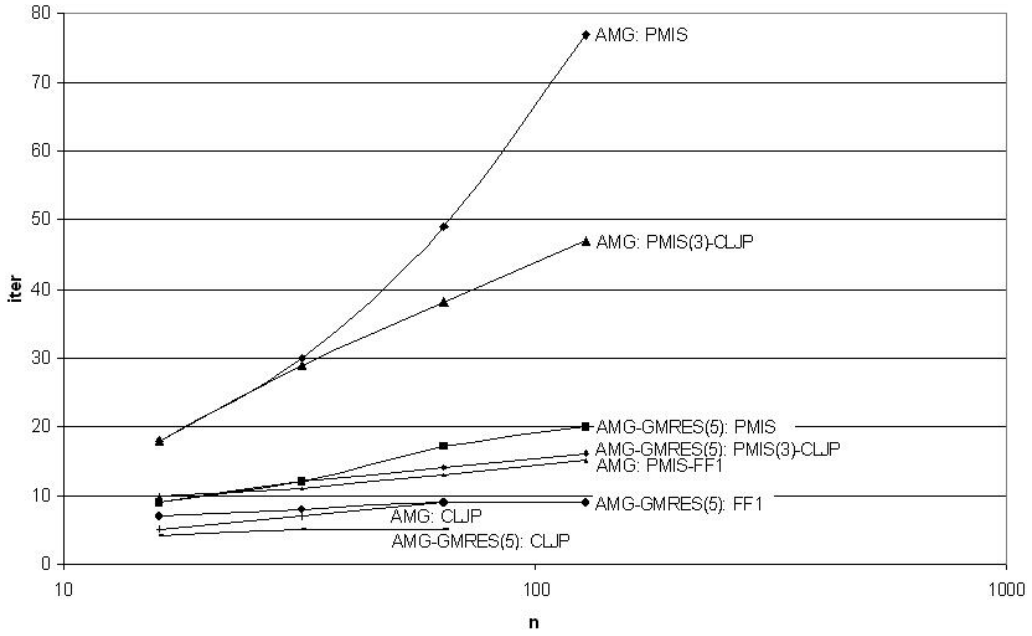

Figure 7.2: Scalability comparison for the 7-point Laplace problem.

of all AMG runs, and does relatively well for the AMG-GMRES(5) runs. The real benefit, however, is that *F-F1* interpolation is able to mirror the scalability of *F-F* interpolation while significantly reducing computational cost. As such, it appears that only considering one *distance-two* *C*-point for strong *F-F* connections without a common *C*-point is sufficient. Of all PMIS methods tested, PMIS-FF and PMIS-FF1 are the most scalable, and are comparable to CLJP in terms of convergence scalability. This is illustrated in Figure 7.2. Only the PMIS-FF1 results are included in Figure 7.2 since the PMIS-FF results are similar for both AMG and AMG-GMRES(5), respectively.

In summary, when considering a combination of operator complexity and execution time, PMIS(3)-CLJP is the best method for small problem sizes; however, it does not scale well in terms of convergence. Due to its superior scalability, PMIS-FF1 appears to be the best choice for large problem sizes.

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP	2.67	15	9	209.88	25.24	235.12
PMIS	1.10	8	47	33.67	65.08	98.75
PMIS greedy	1.12	8	38	36.37	54.01	90.38
PMIS(2)-CLJP	1.12	11	27	34.55	38.05	72.6
PMIS-FF	1.35	7	7	119.34	12.92	132.26
PMIS-FF1	1.27	7	8	84.29	13.52	97.81

Table 7.6: AMG on a $128\times128\times128$ structured grid for the 27-point Laplace problem.

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP	2.67	15	6	208.94	74.72	283.66
PMIS	1.10	8	17	33.72	42.09	75.81
PMIS greedy	1.12	8	15	36.39	36.99	73.38
PMIS(2)-CLJP	1.12	11	12	34.58	30.55	65.13
PMIS-FF	1.35	7	7	124.51	31.94	156.45
PMIS-FF1	1.27	7	7	85.31	36.60	121.91

Table 7.7: AMG-GMRES(5) on a $128 \times 128 \times 128$ structured grid for the 27-point Laplace problem.

7.1.4 27-Point (3D) Laplace Problem

The 3D Laplace problem,

$$-u_{xx} - u_{yy} - u_{zz} = 1, \tag{7.1.4}$$

was considered using a 27-point discretization. Results for AMG and GMRES(5)-accelerated AMG are shown for a $128 \times 128 \times 128$ grid in Tables 7.6 and 7.7, respectively. A scalability study for several of the algorithms is also presented in Figure 7.3, where number of iterations is plotted as a function of problem size. As indicated in Tables 7.6 and 7.7, PMIS coarsening provides a significant advantage over CLJP in terms of operator complexity and setup time, but not in the number of iterations required for convergence. For AMG, PMIS has a larger solve time and requires more iterations than CLJP, but ultimately outperforms CLJP in total execution time. The

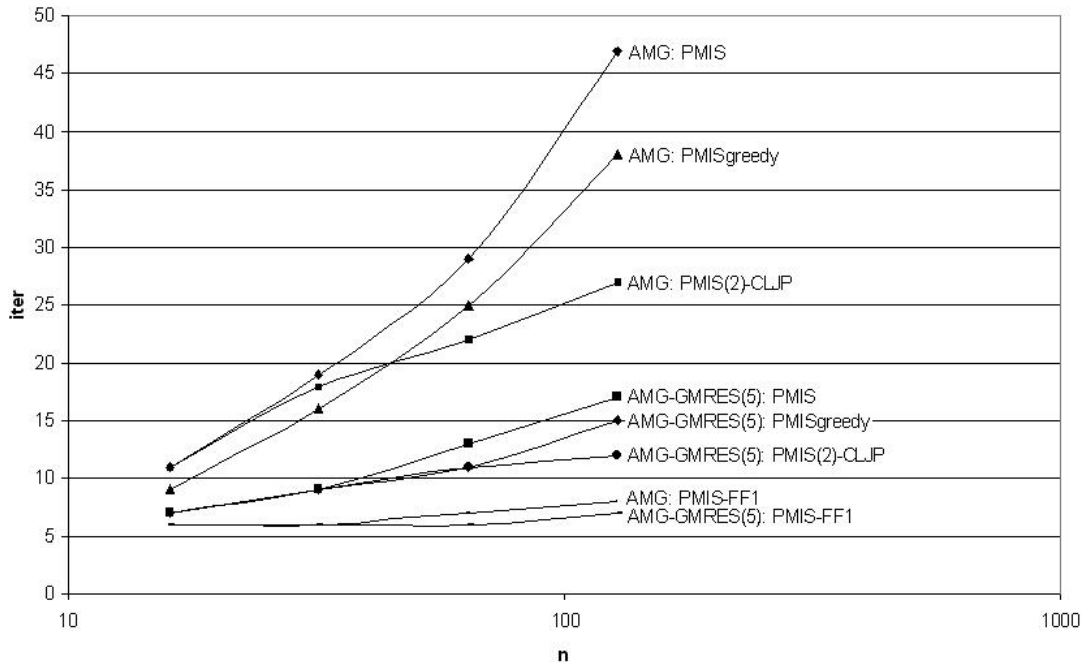

Figure 7.3: Scalability comparison for the 27-point Laplace problem.

larger solve time is due to a lack of accuracy in interpolation as a result of an insufficient number of C-points. While the PMIS solve time is smaller than that of CLJP for AMG-GMRES(5), it is estimated that this advantage would be lost for larger problem sizes since CLJP converges in fewer iterations and scales better in terms of convergence. Thus, for the problem sizes tested, PMIS appears to perform better overall than CLJP; however, convergence of PMIS needs to be improved since it does scale poorly (CLJP scales much better than PMIS). Scaling for this problem is illustrated in Figure 7.3. CLJP results are not shown in Figure 7.3 since they are similar to the PMIS-FF1 results. [1] The PMIS greedy algorithm improves convergence for both AMG and AMG-GMRES(5); however, the improvement is only small. As indicated in Figure 7.3, PMIS greedy scalability is still poor. Also, there is a slight increase in operator complexity and setup time as expected. The increase in setup time would be larger in a parallel implementation due to added communication cost, causing the advantage gained in execution time from improved convergence to likely be lost.

2 The PMIS(2)-CLJP algorithm provides a better improvement than PMIS greedy in terms of convergence and solve time, while requiring only a negligible increase in operator complexity and setup time compared to PMIS. This is a promising improvement; however, the number of iterations required to satisfy the convergence criteria is still excessively high which indicates poor scalability. Indeed, poor scalability is confirmed in Figure 7.3.

3 PMIS with *F-F* interpolation provides the desired improvement in convergence that PMIS greedy and PMIS(2)-CLJP could not. While setup times are considerably higher for PMIS-FF compared to PMIS greedy and PMIS(2)-CLJP, solve times and the number of iterations required for convergence are much lower. While operator complexity is slightly higher than that obtained for PMIS greedy and PMIS(2)-CLJP, it is not excessive, and still offers a considerable improvement over that obtained using CLJP only.

PMIS with *F-F1* interpolation proves to be the best choice for this problem when scalability is considered. It not only matches the convergence characteristics of PMIS-FF, but also provides a substantial reduction in operator complexity and setup time (compared to PMIS-FF), with only a minimal increase in solve time. Thus, including only one *distance-two* *C*-point for strong *F-F* connections without a common *C*-point in interpolation is sufficient. Although execution times are larger and operator complexity is slightly higher for PMIS-FF1 compared to PMIS, PMIS greedy, and PMIS(2)-CLJP, it still appears that PMIS-FF1 is the best algorithm for large problem sizes due to superior scalability. This is affirmed by considering the scalability profiles in Figure 7.3. Although not shown in Figure 7.3, the convergence results for CLJP were almost identical to those of PMIS-FF1. As such, it can be concluded that both CLJP and PMIS-FF1 are almost optimal methods with respect to convergence scalability.

It can be noted for many of the algorithms in Tables 7.6 and 7.7, that due to a faster solve time, AMG actually outperforms AMG-GMRES(5) in terms of total execution time. This is a confounding result (since GMRES usually accelerates AMG), and may indicate that an extra load was mistakenly added to the computer during execution of the AMG-GMRES(5) runs. However, setup times are not affected in the

same way, which is puzzling. Regardless of the cause, this issue is worth investigating further in future work.

In summary, when considering operator complexity and execution time, PMIS(2)-CLJP is the best algorithm for solving small problem sizes. However, PMIS-FF1 appears to be the best choice for large problem sizes due to its superior scalability.

7.1.5 3D Anisotropic Laplace Problem

The 3D anisotropic Laplace problem,

$$-cu_{xx} - u_{yy} - u_{zz} = 1, \tag{7.1.5}$$

was considered using a 7-point discretization, and $c = 0.001$. Results for AMG and GMRES(5)-accelerated AMG are shown for a $128 \times 128 \times 128$ grid in Tables 7.8 and 7.9, respectively. A scalability profile for several of the algorithms tested for this problem is presented in Figure 7.4, where number of iterations is plotted as a function of problem size.

It should be noted that, not only did the CLJP test run out of memory on the 128^3 grid, it was also about four times slower than PMIS for both AMG and AMG-GMRES(5) on a 64^3 grid, and setup and solve times for PMIS were faster than for CLJP. In addition, operator complexity was about ten times smaller for PMIS coarsening compared to that obtained using CLJP on the 64^3 grid. While CLJP did exhibit better convergence scalability than PMIS (similar to the PMIS-FF results shown in Figure 7.4), it can be concluded that, even without any of the proposed modifications, PMIS coarsening is preferable to CLJP for the problem sizes investigated.

1 For this problem, PMIS greedy does not significantly outperform PMIS for either AMG or AMG-GMRES(5). Although convergence is slightly improved, operator complexity is negatively affected. In addition, while not illustrated in Figure 7.4, scalability for PMIS greedy was similar to that obtained for PMIS. In also considering that a parallel implementation would only increase setup time due to processor boundary communication, it can be concluded that PMIS greedy does not provide a useful advantage over PMIS for this problem. 2 PMIS(3)-CLJP moderately

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP (64^3)	22.34	16	8	20.23	5.25	25.48
PMIS	2.36	9	52	16.60	58.20	74.00
PMIS greedy	2.43	9	48	17.48	54.61	72.09
PMIS(3)-CLJP	2.48	13	34	17.79	39.12	56.91
PMIS-FF	4.81	8	12	82.18	20.88	103.06
PMIS-FF1	3.68	8	13	43.51	18.93	62.44

Table 7.8: AMG on a $128 \times 128 \times 128$ structured grid for the 3D anisotropic Laplace problem. CLJP results are for the 64^3 problem because the 128^3 test ran out memory.

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP (64^3)	22.34	16	6	20.21	5.54	25.75
PMIS	2.36	9	25	16.62	43.53	60.15
PMIS greedy	2.43	9	24	17.56	42.68	60.24
PMIS(3)-CLJP	2.48	13	20	17.96	35.54	53.50
PMIS-FF	4.81	8	9	82.10	22.95	105.05
PMIS-FF1	3.68	8	10	43.43	21.70	65.13

Table 7.9: AMG-GMRES(5) on a $128 \times 128 \times 128$ structured grid for the 3D anisotropic Laplace problem. CLJP results are for the 64^3 problem because the 128^3 test ran out memory.

improves convergence for both AMG and AMG-GMRES(5), while only increasing operator complexity slightly compared to PMIS. Setup times are marginally worse than those obtained for PMIS, but the advantage of performing CLJP coarsening on coarser grid levels is directly evident in the reduction in solve times and the number of iterations required for convergence. Although this represents an improvement over PMIS, the scalability of the algorithm is still undesirable as illustrated in Figure 7.4.

3 Tables 7.8 and 7.9 show that PMIS-FF dramatically improves convergence compared to the other modifications, and it can be noted that convergence of PMIS-FF is comparable to that of CLJP coarsening with classical interpolation. The drawback of PMIS-FF is that operator complexity is rather large (although still much less than for CLJP), and setup time is increased due to the consideration of all *distance-two*

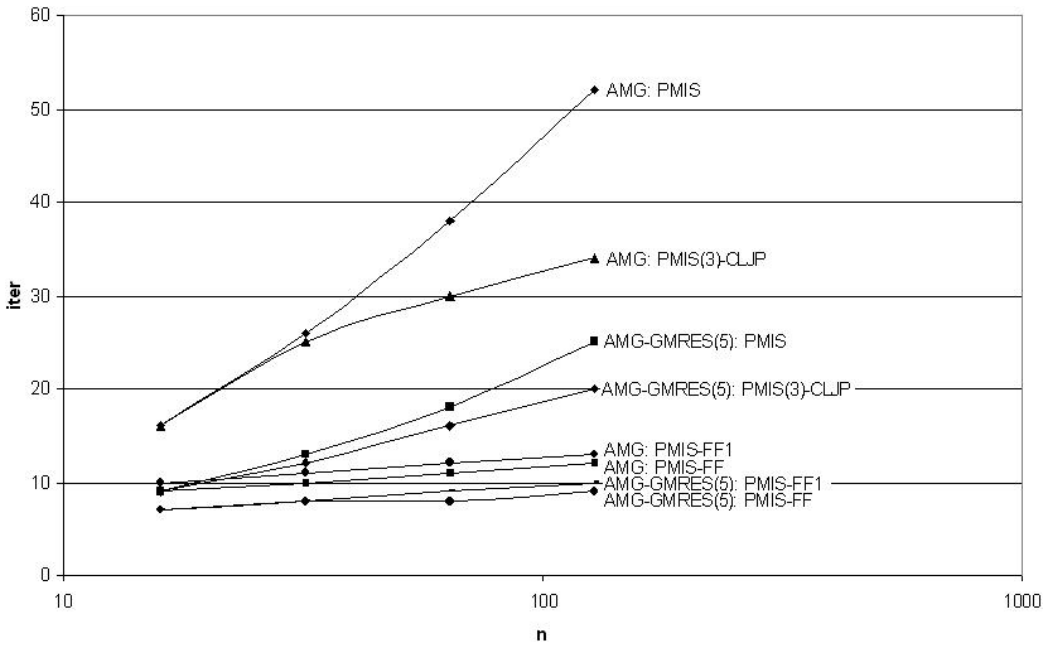

Figure 7.4: Scalability comparison for the 7-point anisotropic Laplace problem.

C-points for strong *F*-*F* connections without a common *C*-point in interpolation. However, the situation is noticeably improved for PMIS-FF1. Operator complexity and setup time are substantially reduced by only considering one *distance-two* *C*-point, while the convergence benefits of *F*-*F* interpolation are still maintained. As illustrated in Figure 7.4, PMIS-FF1 is comparable to PMIS-FF as being the best algorithm of all those tested in terms of convergence scalability, and as shown in Tables 7.8 and 7.9, performs well in terms of execution time for both AMG and AMG-GMRES(5), respectively. Although operator complexity for PMIS-FF1 is somewhat high, it is still acceptable, and is much better than that of CLJP.

In summary, when a combination of execution speed and operator complexity is considered, PMIS(3)-CLJP is the best method for solving small problem sizes like those tested in this thesis. PMIS-FF1 also performs well with respect to execution time, and, due to its superior scalability, appears to be the best method for large problem sizes.

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP (64^3)	22.49	15	9	20.38	5.93	26.31
PMIS	2.36	9	74	16.78	82.87	99.65
PMIS greedy	2.44	9	72	17.51	82.52	100.03
PMIS(3)-CLJP	2.49	12	45	18.05	51.58	69.63
PMIS-FF	4.80	7	13	84.02	22.59	106.61
PMIS-FF1	3.68	8	15	43.80	21.85	65.65

Table 7.10: AMG on a $128 \times 128 \times 128$ structured grid for the 3D convection-diffusion problem. CLJP results are for the 64^3 problem because the 128^3 test ran out memory.

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP (64^3)	22.49	15	6	20.45	5.64	26.09
PMIS	2.36	9	22	16.84	39.15	55.99
PMIS greedy	2.44	9	21	17.57	38.33	55.90
PMIS(3)-CLJP	2.49	12	17	17.92	31.15	49.07
PMIS-FF	4.80	7	9	83.93	23.01	106.94
PMIS-FF1	3.68	8	9	43.83	19.84	63.67

Table 7.11: AMG-GMRES(5) on a $128 \times 128 \times 128$ structured grid for the 3D convection-diffusion problem. CLJP results are for the 64^3 problem because the 128^3 test ran out memory.

7.1.6 3D Convection-Diffusion Problem

The 3D convection-diffusion problem,

$$-c_x u_{xx} - c_y u_{yy} - c_z u_{zz} + a_x u_x + a_y u_y + a_z u_z = 1, \tag{7.1.6}$$

was considered using 7-point and upwind discretizations. Convection and diffusion parameters were taken to be $c_x = c_y = c_z = 1$ and $a_x = a_y = a_z = 10$, respectively. Results for AMG and GMRES(5)-accelerated AMG are shown for a $128 \times 128 \times 128$ grid in Tables 7.10 and 7.11, respectively. A scalability study for several of the algorithms is also presented in Figure 7.5, where number of iterations is plotted as a function of problem size.

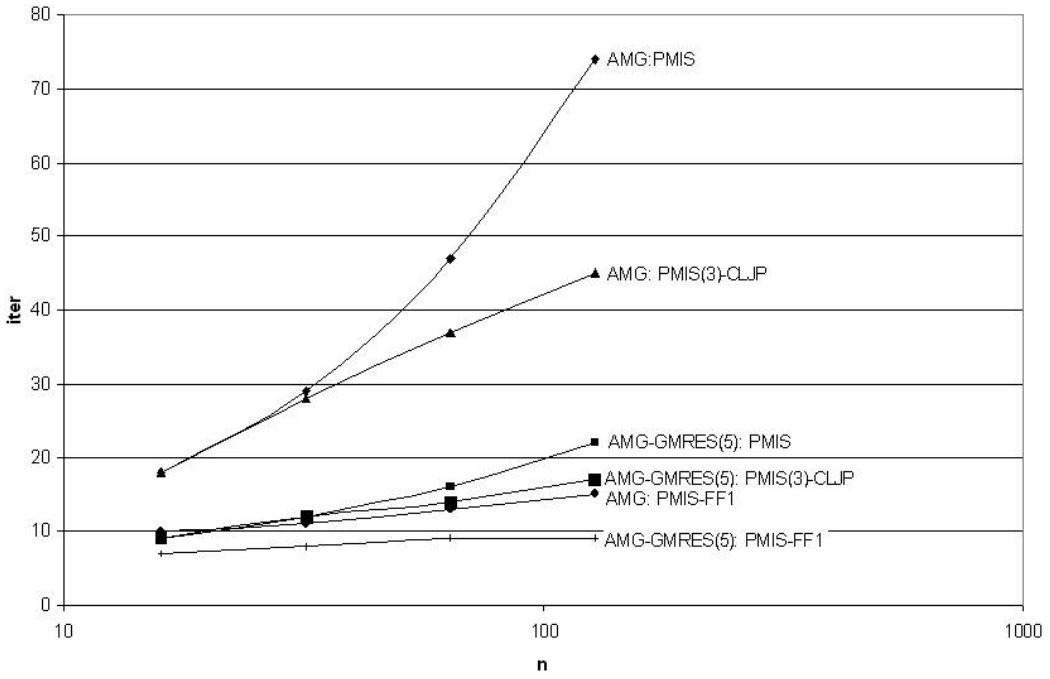

Figure 7.5: Scalability comparison for the 3D convection-diffusion problem.

While the CLJP test did run out of memory on the 128^3 grid, it should also be noted that it was about three times slower than PMIS for AMG, and about five times slower than PMIS for AMG-GMRES(5), on a 64^3 grid. In addition, operator complexity was about 10 times larger for CLJP compared to PMIS on a 64^3 grid. However, CLJP did converge in significantly fewer iterations than PMIS, indicating better convergence scalability. These results illustrate that, for the problem sizes tested, PMIS is a better choice than CLJP, even though CLJP is more scalable in terms of convergence.

1 For this problem, PMIS greedy essentially offers no improvement in terms of convergence or execution time for either AMG or AMG-GMRES(5) compared to PMIS. In addition, operator complexity is increased for PMIS greedy compared to PMIS, and scalability is approximately the same as for PMIS. Consequently, it can be concluded that the PMIS greedy algorithm does not improve PMIS for this problem.

2 The PMIS(3)-CLJP algorithm provides a considerable improvement over PMIS in terms of convergence, with only a small increase in setup time, for both AMG and AMG-GMRES(5). This is accompanied by only a small increase in operator complexity. As such, the improved structure on coarser grids resulting from CLJP coarsening aids convergence, but does not significantly increase storage cost. While this represents an improvement over PMIS, it can be noted that convergence scalability is still undesirable. This is illustrated in Figure 7.5.

3 PMIS with *F-F* interpolation significantly improves convergence compared to PMIS with classical interpolation, but produces an increase in setup time and operator complexity. PMIS-FF1 is able to match the convergence properties of PMIS-FF, while also significantly reducing setup time and operator complexity. Although the operator complexity generated by PMIS-FF1 is somewhat undesirable, it is still much better than that of CLJP. PMIS-FF1 also has excellent scalability as illustrated in Figure 7.5. Although not shown in Figure 7.5, CLJP convergence scalability is approximately equal to that of PMIS-FF1.

In summary, for small problem sizes like those tested, PMIS(3)-CLJP is the best method in terms of a combination of execution time and operator complexity. PMIS-FF1 performed well with respect to execution time, and appears to be the best method for large problem sizes due to its superior scalability.

7.1.7 2D Rotated Anisotropic Laplace Problem

The 2D rotated anisotropic Laplace problem,

$$-(c^2 + \epsilon s^2)u_{xx} - 2(1-\epsilon)csu_{xy} - (s^2 + \epsilon c^2)u_{yy} = 1, \tag{7.1.7}$$

where $\epsilon \in \mathbb{R}$, $c = \cos\gamma$, $s = \sin\gamma$, and γ is the angle of rotation, was considered with $\epsilon = 0.001$ and using 5-point and left-oriented 7-point discretizations. Results for AMG and GMRES(5)-accelerated AMG are shown for a 256×256 grid in Tables 7.12 through 7.15 for rotation angles of $\gamma = 45^o$ and $\gamma = 60^o$. A scalability study for several of the algorithms is also presented in Figures 7.6 and 7.7, where number of

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP	2.77	14	8	0.34	0.35	0.69
PMIS	1.88	10	89	0.21	2.73	2.94
PMIS greedy	1.88	10	77	0.22	2.40	2.62
PMIS(1)-CLJP	2.29	14	26	0.28	0.95	1.23
PMIS-FF	2.04	10	24	0.25	0.80	1.05
PMIS-FF1	2.03	10	24	0.24	0.79	1.03

Table 7.12: AMG on a 256×256 structured grid for the 2D rotated anisotropic Laplace problem with $\gamma = 45^o$.

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP	2.77	14	6	0.36	0.43	0.79
PMIS	1.88	10	25	0.22	1.19	1.41
PMIS greedy	1.88	10	26	0.21	1.27	1.48
PMIS(1)-CLJP	2.29	14	12	0.27	0.68	0.95
PMIS-FF	2.04	10	13	0.25	0.69	0.94
PMIS-FF1	2.03	10	12	0.24	0.65	0.89

Table 7.13: AMG-GMRES(5) on a 256×256 structured grid for the 2D rotated anisotropic Laplace problem with $\gamma = 45^o$.

iterations is plotted as a function of problem size, for rotation angles of $\gamma = 45^o$ and $\gamma = 60^o$, respectively.

Tables 7.12 and 7.13 indicate that for an angle of rotation of $\gamma = 45^o$, convergence of PMIS is much worse than that of CLJP for both AMG and AMG-GMRES(5). While there is some improvement in terms of operator complexity for PMIS compared to CLJP, it is only moderate, and the resulting execution times show that CLJP is a better choice than PMIS for this problem. Furthermore, while the PMIS results are not included in Figure 7.6, it was found that CLJP is much more scalable in terms of convergence than PMIS for this problem.

1: $\gamma = 45^o$ It can be noted in Tables 7.12 and 7.13 that PMIS greedy only slightly improves convergence compared to PMIS for AMG, while not at all for AMG-GMRES(5), and that the operator complexity for PMIS and PMIS greedy is the same.

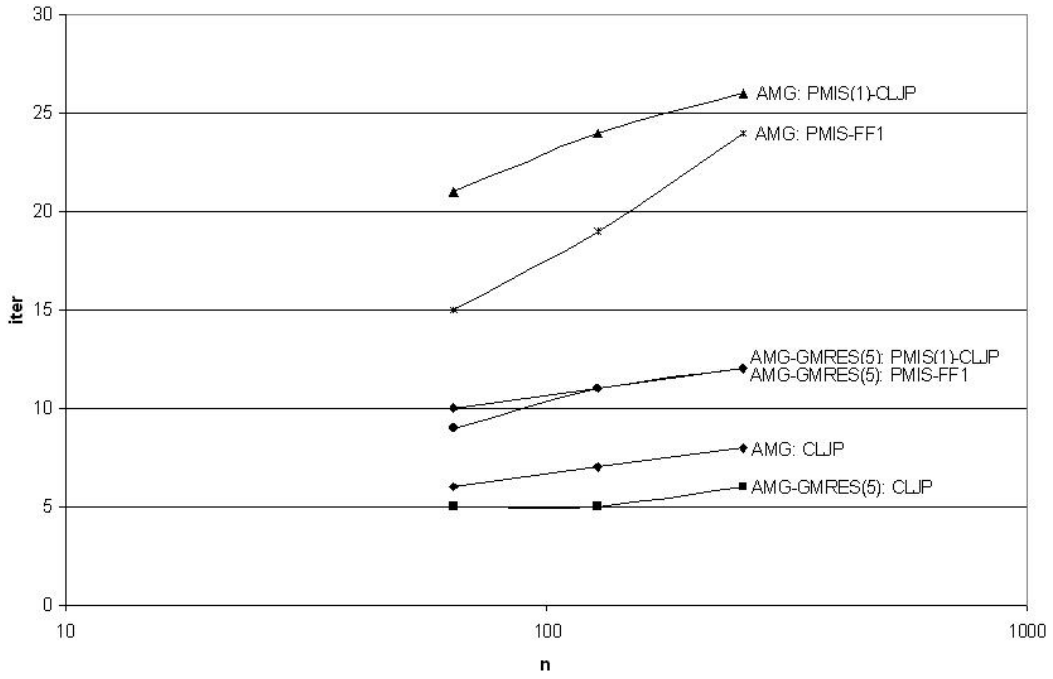

Figure 7.6: Scalability comparison for the 2D rotated anisotropic Laplace problem with $\gamma = 45^o$.

Thus, for both AMG and AMG-GMRES(5), CLJP far outperforms PMIS greedy.

2: $\gamma = 45^o$ PMIS(1)-CLJP does well to improve convergence compared to PMIS, while also reducing operator complexity compared to CLJP, for both AMG and AMG-GMRES(5), respectively. As illustrated in Figure 7.6, convergence scalability is only slightly worse than for CLJP for AMG-GMRES(5), but is much worse for AMG.

3: $\gamma = 45^o$ Tables 7.12 and 7.13 indicate that the extra work performed in considering all *distance-two* C-points in PMIS-FF is not necessary, as PMIS-FF1 produces similar results in terms of convergence (for this reason, the PMIS-FF results are not included in Figure 7.6). PMIS-FF and PMIS-FF1 show comparable performance to PMIS(1)-CLJP for AMG and AMG-GMRES(5), respectively, but provide a greater reduction in operator complexity. However, as was true for PMIS(1)-CLJP, convergence scalability of PMIS-FF and PMIS-FF1 is worse than for CLJP (slightly worse for AMG-GMRES(5), and much worse for AMG). It was also noted that operator

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP	4.71	13	40	0.84	2.34	3.18
PMIS	1.82	8	$\gg 200$	*Slow*	*to*	*converge*
PMIS greedy	1.85	8	$\gg 200$	*Slow*	*to*	*converge*
PMIS(5)-CLJP	1.81	8	$\gg 200$	*Slow*	*to*	*converge*
PMIS-FF	2.47	8	176	0.48	6.11	6.59
PMIS-FF1	2.25	8	197	0.41	6.50	6.91

Table 7.14: AMG on a 256×256 structured grid for the 2D rotated anisotropic Laplace problem with $\gamma = 60^o$.

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP	4.71	13	15	0.84	1.21	2.05
PMIS	1.82	8	87	0.28	3.95	4.23
PMIS greedy	1.85	8	83	0.28	3.77	4.05
PMIS(1)-CLJP	2.63	11	49	0.46	2.73	3.19
PMIS-FF	2.47	8	42	0.48	2.22	2.70
PMIS-FF1	2.25	8	45	0.40	2.23	2.63

Table 7.15: AMG-GMRES(5) on a 256×256 structured grid for the 2D rotated anisotropic Laplace problem with $\gamma = 60^o$.

complexity for CLJP only increased linearly with problem size. Thus, although CLJP produces slightly higher operator complexity, it is the best method for this problem with $\gamma = 45^o$.

For $\gamma = 60^o$, Table 7.14 shows that, except in terms of operator complexity, no improvement over CLJP is made by any of the methods for AMG. Convergence and scalability were found to be poor for all modifications. Of all algorithms tested, CLJP is the best choice for AMG.

As shown in Table 7.15 for $\gamma = 60^o$, accelerating AMG with GMRES(5) provides a better result in terms of convergence than stand-alone AMG. However, as indicated in Figure 7.7, convergence scalability of all algorithms except CLJP is still undesirable. It was also noted that operator complexity for CLJP only increased linearly with problem size. Consequently, even though CLJP results in a large operator complexity,

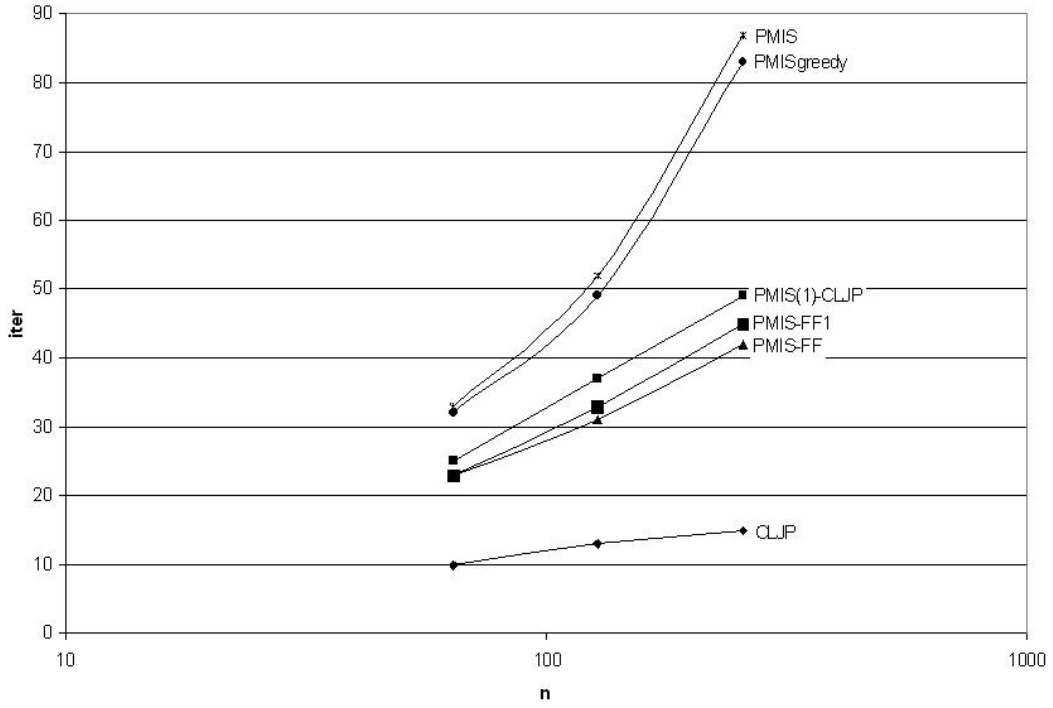

Figure 7.7: Scalability comparison for AMG-GMRES(5) for the 2D rotated anisotropic Laplace problem with $\gamma = 60^o$.

it is the best solution method. If operator complexity is an important consideration to the user, PMIS(1)-CLJP, PMIS-FF, or PMIS-FF1 may provide a reasonable alternative since the scalability of these three methods is not overly prohibitive. Of the three, Table 7.15 and Figure 7.7 indicate that PMIS-FF1 offers the best combination of operator complexity and convergence.

In summary, although CLJP produces higher operator complexities than the other algorithms tested, it still exhibits the best scalability, and is the best choice for this problem. Note that this is the first and only test problem where *F-F* and *F-F1* interpolation do not sufficiently improve the convergence of PMIS-coarsened AMG.

7.1.8 3D Elliptic PDE with Jumps

The 3D elliptic PDE with jumps in the coefficients on the unit cube,

$$-(au_x)_x - (au_y)_y - (au_z)_z = 1, \tag{7.1.8}$$

was considered using a 7-point discretization and

$$\begin{aligned} a(x,y,z) &= 1000, \quad \textit{on } 0.1 \le x,y,z \le 0.9 \\ &= 0.01, \quad \textit{on } 0 < x,y,z < 0.1, \textit{ and the other} \\ &\qquad \textit{cubes of size } 0.1 \times 0.1 \times 0.1 \textit{ located} \\ &\qquad \textit{on the corners of the domain} \\ &= 1, \qquad \textit{elsewhere.} \end{aligned} \tag{7.1.9}$$

Results for AMG and GMRES(5)-accelerated AMG are shown in Tables 7.16 and 7.17, respectively. A scalability study for several of the algorithms is also presented in Figure 7.8, where number of iterations is plotted as a function of problem size.

Note that, while the CLJP test ran out of memory for both AMG and AMG-GMRES(5) for the problem sizes named in Tables 7.16 and 7.17, it did take about the same amount of execution time as PMIS on the next smallest grid (80^3 points for AMG and 40^3 points for AMG-GMRES(5), respectively). Also, operator complexity was about nine times larger for CLJP compared to PMIS on the 80^3 grid, but CLJP converged in approximately ten iterations for both AMG and AMG-GMRES(5) (whereas convergence for PMIS was much worse). This illustrates the trade-off between CLJP and PMIS for this difficult problem – CLJP has excellent convergence scalability but poor operator complexity, and PMIS results in much better operator complexity but poor convergence scalability.

1 For this problem, convergence properties are poor for PMIS and PMIS greedy for both AMG and AMG-GMRES(5), respectively. Although convergence is improved slightly by PMIS greedy compared to PMIS for AMG-GMRES(5), the result is still impractical.

2 PMIS(1)-CLJP improves convergence significantly for both AMG and AMG-GMRES(5). As illustrated in Figure 7.8, convergence scalability of PMIS(1)-CLJP

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP (80^3)	21.54	14	11	39.58	14.6	54.18
PMIS	2.44	8	$\gg 200$	*Slow*	*to*	*converge*
PMIS greedy	2.52	8	$\gg 200$	*Slow*	*to*	*converge*
PMIS(1)-CLJP	7.21	13	23	47.93	44.32	92.25
PMIS-FF	4.94	7	14	62.95	20.54	83.49
PMIS-FF1	3.84	8	18	35.36	22.47	57.83

Table 7.16: AMG on a $120 \times 120 \times 120$ structured grid for the 3D elliptic PDE with varying coefficients. CLJP results are for the 80^3 problem because the 120^3 test ran out memory.

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP (40^3)	15.88	13	6	3.15	1.06	4.21
PMIS	2.46	7	188	4.06	77.34	81.40
PMIS greedy	2.54	8	144	4.19	59.90	64.09
PMIS(1)-CLJP	6.97	12	11	12.76	8.38	21.14
PMIS-FF	4.90	7	9	16.46	5.34	21.80
PMIS-FF1	3.85	7	10	9.67	4.99	14.66

Table 7.17: AMG-GMRES(5) on a $80 \times 80 \times 80$ structured grid for the 3D elliptic PDE with varying coefficients. CLJP results are for the 40^3 problem because the 80^3 test ran out memory.

for AMG is still poor (but may level-off for large problems), but is quite good for AMG-GMRES(5) (comparable to that achieved with CLJP). In considering AMG-GMRES(5) with PMIS(1)-CLJP, it is also noted that operator complexity is significantly increased compared to PMIS. While this increase is not as severe as the operator complexity obtained using CLJP, the result is still undesirable. Thus, although PMIS(1)-CLJP with AMG-GMRES(5) appears to be an effective solver for this problem, severe storage complications may arise for larger grids.

3 PMIS-FF improves convergence significantly compared to PMIS, and to an almost optimal level with regard to scalability as illustrated in Figure 7.8.[4] This is

[4]PMIS-FF and PMIS-FF1 results are not included for AMG-GMRES(5) in Figure 7.8 for the largest problem size because the runs could not be completed due to inadequate computational resources.

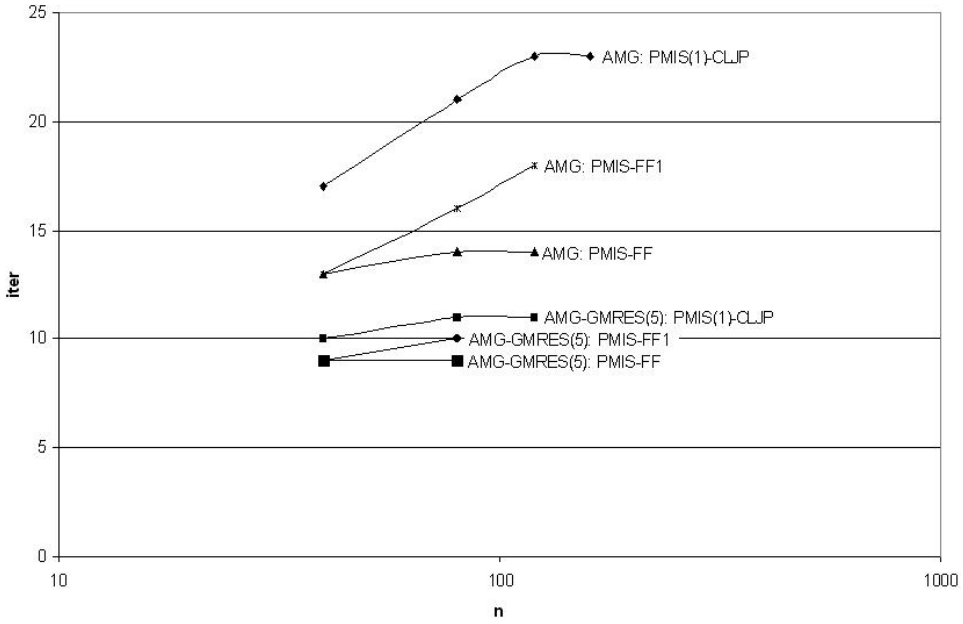

Figure 7.8: Scalability comparison for the 3D PDE with jumps in the coefficients.

best observed for AMG-GMRES(5) and to a lesser extent for AMG. Unfortunately, this is accompanied by a large increase in operator complexity. Compared to PMIS-FF, PMIS-FF1 significantly reduces operator complexity, and maintains an improved level of convergence; however, scalability may be degraded as illustrated in Figure 7.8. While accounting for only one *distance-two C*-point in interpolation may be sufficient, more tests should be performed to confirm whether or not this is true for larger problem sizes. Furthermore, although operator complexity is high for PMIS-FF1, it is still much better than that obtained with CLJP. It can also be noted in Tables 7.16 and 7.17 that PMIS-FF1 is considerably faster than all other algorithms for both AMG and AMG-GMRES(5), respectively. In summary, PMIS-FF1 is the best method for small problems. Since scalability results are limited, it is uncertain which of PMIS-FF and PMIS-FF1 would be the best method for large problems. As such, future work should be performed with appropriate computational resources to investigate this further.

7.2 More on Modification 2

As mentioned at the start of the chapter, modification 2 – performing PMIS on the first g finest grids and CLJP on all remaining grids – requires a separate discussion in order to best interpret the results. Recall that PMIS(g)-CLJP was performed for $g = 1,\ 2,\ 3,\ 4$ and 5; and that only the g that provided the best overall result, taking into account operator complexity, convergence, *and* execution time, was presented for each model problem in Section 7.1. This section will investigate trends observed in results across all values of g. While only a select set of data will be presented here, a short discussion of whether or not these observations apply to a each of the model problems will also be provided.

Modification 2 is designed with the aim of achieving low operator complexity resulting from PMIS, while also taking advantage of the strong convergence scalability of CLJP. However, it does not appear that these two coarsening methods can be combined in a way such that a scalable algorithm with low operator complexity is obtained. This is best illustrated for the 3D PDE with jumps in the coefficients. Tables 7.18 and 7.19 show the results across all g for the problem described in Section 7.1.8 for AMG and AMG-GMRES(5), respectively. In both of these tables, a trade-off between operator complexity and convergence is directly evident.

Tables 7.18 and 7.19 show that both operator complexity and the number of iterations required to converge for PMIS(g)-CLJP approach the values obtained for PMIS as g is increased. As a result, the best convergence is obtained when $g = 1$. Unfortunately, this also corresponds to the worst operator complexity, since the most CLJP coarsening is performed when $g = 1$. This trade-off between convergence and operator complexity diminishes scalability, and was observed for all problems. Operator complexity was reduced (often substantially) compared to CLJP with classical interpolation, and this was most evident for larger g (where more PMIS is performed). While convergence was improved for all problems with PMIS(1)-CLJP compared to PMIS, this was never to the same level as CLJP.

For every problem, PMIS(1)-CLJP always converged in the fewest number of iterations (of all g), and was therefore the best in terms of convergence scalability.

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP (80^3)	21.54	14	11	39.58	14.6	54.18
PMIS	2.44	8	$\gg 200$	*Slow*	*to*	*converge*
PMIS(1)-CLJP	7.21	13	23	47.93	44.32	92.25
PMIS(2)-CLJP	3.49	12	168	20.58	190.16	210.74
PMIS(3)-CLJP	2.56	11	$\gg 200$	*Slow*	*to*	*converge*
PMIS(4)-CLJP	2.45	11	$\gg 200$	*Slow*	*to*	*converge*
PMIS(5)-CLJP	2.44	10	$\gg 200$	*Slow*	*to*	*converge*

Table 7.18: Modification 2 comparison for AMG on a $120 \times 120 \times 120$ structured grid for the 3D elliptic PDE with varying coefficients. CLJP results are for the 80^3 problem because the 120^3 test ran out memory.

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP (40^3)	15.88	13	6	3.15	1.06	4.21
PMIS	2.46	7	188	4.06	77.34	81.40
PMIS(1)-CLJP	6.97	12	11	12.76	8.38	21.14
PMIS(2)-CLJP	3.42	11	30	5.69	14.20	19.89
PMIS(3)-CLJP	2.56	11	105	4.24	44.20	48.44
PMIS(4)-CLJP	2.48	9	146	4.10	60.26	64.36
PMIS(5)-CLJP	2.46	8	177	4.05	71.78	75.83

Table 7.19: Modification 2 comparison for AMG-GMRES(5) on a $80 \times 80 \times 80$ structured grid for the 3D elliptic PDE with varying coefficients. CLJP results are for the 40^3 problem because the 80^3 test ran out memory.

This occurred since PMIS(1)-CLJP incorporates the most CLJP coarsening of all g. However, the resulting scalability was often significantly worse than that of CLJP. Thus, while PMIS(1)-CLJP improves convergence scalability compared to PMIS, it is not to a level that one would hope for. Furthermore, for the 7-point Laplace problem, the 3D anisotropic Laplace problem, and the 3D convection-diffusion problem, an increase in operator complexity with problem size was observed at a rate that indicates that PMIS(1)-CLJP is not scalable with respect to storage cost. While this was not to the same extent as that observed for CLJP, it does not appear that C_{op} will cease

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP (64^3)	22.51	14	9	20.36	5.83	26.19
PMIS	2.36	8	77	16.63	85.93	102.56
PMIS(1)-CLJP	7.17	14	23	59.31	53.41	112.72
PMIS(2)-CLJP	3.43	13	36	25.15	48.46	73.61
PMIS(3)-CLJP	2.49	12	47	17.70	53.82	71.52
PMIS(4)-CLJP	2.37	11	61	16.73	68.74	85.47
PMIS(5)-CLJP	2.36	10	73	16.62	81.99	98.61

Table 7.20: Modification 2 comparison for AMG on a $128 \times 128 \times 128$ structured grid for the 7-point Laplace problem. CLJP results are for the 64^3 problem because the 128^3 test ran out memory.

Method	C_{op}	$\#lev$	$iter$	t_{setup}	t_{solve}	t_{total}
CLJP (64^3)	22.51	14	5	20.32	4.23	24.55
PMIS	2.36	8	20	16.67	35.26	51.93
PMIS(1)-CLJP	7.17	14	11	59.33	37.36	96.69
PMIS(2)-CLJP	3.43	13	14	25.00	28.59	53.59
PMIS(3)-CLJP	2.49	12	16	17.70	29.89	47.59
PMIS(4)-CLJP	2.37	11	19	16.79	33.59	50.38
PMIS(5)-CLJP	2.36	10	20	16.65	34.99	51.64

Table 7.21: Modification 2 comparison for AMG-GMRES(5) on a $128 \times 128 \times 128$ structured grid for the 7-point Laplace problem. CLJP results are for the 64^3 problem because the 128^3 test ran out memory.

to increase with respect to problem size asymptotically, and could therefore cause storage complications for large problem sizes.

It was also noted that, except for the 5- and 9-point Laplace and the rotated anisotropic Laplace problems, the version of PMIS(g)-CLJP that took the fewest number of iterations to converge, PMIS(1)-CLJP, was not always the fastest in terms of execution time. This means that a method with a larger value of g actually executed faster than PMIS(1)-CLJP. To illustrate this, consider the data for the 7-point Laplace problem presented in Tables 7.20 and 7.21 for AMG and AMG-GMRES(5),

respectively. These tables show that, although PMIS(1)-CLJP converges in the fewest iterations, PMIS(3)-CLJP is actually the fastest for both AMG and AMG-GMRES(5) for this problem size. This occurs for two reasons. First, larger setup times are required for PMIS(1)-CLJP than PMIS(3)-CLJP since more CLJP coarsening is performed in the former. Second, since operator complexity is larger for PMIS(1)-CLJP than PMIS(3)-CLJP, PMIS(1)-CLJP requires more time for matrix multiplication per iteration in the solve phase.

The results presented in this section illustrate the interplay between convergence, execution time, and operator complexity that exists when trying to combine PMIS and CLJP coarsening on different grid levels. From these results it does not appear that modification 2 can produce an algorithm that possesses the convergence properties of CLJP and operator complexities similar to PMIS. In addition, fine-tuning the balance between convergence and operator complexity is not straightforward, and was found to be problem dependent. While the user may find some of the results presented here interesting, modification 2 certainly lacks the robustness required for general application to large problems. It appears that PMIS-FF1 is the best choice in this respect. However, for moderately small problem sizes like those examined in this thesis, modification 2 produces a solver with an attractive balance between execution time and storage cost for a variety of problems.

7.3 Summary

This section presents a general summary of how each of the modifications performed for all of the problems.

1 PMIS greedy always produced a small increase in operator complexity compared to PMIS (except for the 5-point Laplace problem for which operator complexity was found to be the same for both methods), and never significantly improved convergence and scalability for any of the problems. Since PMIS greedy would require an increase in processor boundary communication compared to PMIS in parallel implementations, any advantage gained in execution time through improved convergence would likely be lost.

2 With respect to execution time and operator complexity, PMIS(g)-CLJP was the best solver for many of the problems with small to moderate problem size. However, due to poor scalability, PMIS(g)-CLJP can not be considered an effective solver for large problems.

3 PMIS-FF and PMIS-FF1 produced scalable results, similar to those of CLJP in terms of convergence, for most problems. PMIS-FF operator complexity was often high; however, it was still much better than that of CLJP. In addition, PMIS-FF1 generally reduced operator complexity significantly compared to PMIS-FF, while also achieving good scalability. For these reasons, PMIS-FF1 is considered the best algorithm for many of the problems tested. The exception was the rotated anisotropic Laplace problem, for which CLJP is considered the best solver. For this problem, scalability of PMIS-FF1 was poor for AMG, and more reasonable for AMG-GMRES(5); however, this was still not to the level achieved by CLJP. It was also found that, for the 3D PDE with jumps in the coefficients, PMIS-FF is slightly more scalable than PMIS-FF1 for small problems, although PMIS-FF1 still results in lower operator complexity. Since the problem sizes tested were limited for this problem, future work is necessary to accurately confirm this observation.

Chapter 8

Conclusion and Future Work

This thesis presented three modifications to current coarsening and interpolation procedures for algebraic multigrid. The goal was to improve AMG in terms of convergence, storage cost, scalability, and robustness. The modifications included a greedy implementation of the PMIS coarsening algorithm, performing PMIS coarsening only on finer grids while CLJP coarsening is performed on coarser grids, and a modified version of *F-F* interpolation, known as *F-F1* interpolation. The performance of these modifications in the context of the AMG algorithm was evaluated for a variety of test problems.

The PMIS greedy algorithm consists of performing standard PMIS, but incrementing the measures of unassigned strongly influencing neighbours when an F-point is assigned. This was done with the motivation that F-points would have a greater number of strongly influencing neighbours defined as C-points, and that this would in turn improve interpolation. It was found that the structure of coarsened grids could be improved compared to those obtained with PMIS. However, this did not provide a significant improvement in terms of convergence compared to PMIS. For all model problems tested, PMIS greedy at best provided only a small improvement in convergence compared to PMIS. In addition, due to the increased structure of coarsened grids, PMIS greedy always produced operator complexities that were equal to or larger than those of PMIS. Also, since PMIS greedy has to update the measures of strongly influencing neighbours after an F-point is declared, larger setup times were observed

compared to PMIS. In considering the operator complexities and setup times generated by PMIS greedy, and the fact that any improvement in convergence was only small for all problems tested, it can be concluded that PMIS greedy does not improve PMIS. Furthermore, PMIS greedy would increase setup time in a parallel implementation due to added processor boundary communication that would be required when updating the measures of unassigned points. Thus, it can be further concluded that PMIS greedy would provide no advantage over PMIS in a parallel implementation. As such, no future work incorporating PMIS greedy is recommended.

The second modification of AMG was to perform PMIS coarsening on finer grid levels, and CLJP coarsening on all remaining grids. This was done with the motivation that PMIS coarsening could reduce operator complexity on finer grids where it would make the biggest difference, while the strong convergence properties of CLJP coarsening could still be exploited on coarser grids where the impact on operator complexity would be reduced. The resulting algorithm was called PMIS(g)-CLJP, indicating that PMIS coarsening was performed on the first g finest grids, and CLJP coarsening was performed on all remaining grids. This was tested for a variety of problems with $g = 1,\ 2,\ 3,\ 4$ and 5. It was found that PMIS(1)-CLJP considerably improves convergence compared to PMIS; however, convergence scalability was never as good as that achieved by CLJP. Furthermore, it was found that operator complexity and the number of iterations required for convergence approached PMIS values as g was increased. Unfortunately, this means that operator complexity is always greatest when the number of iterations required for convergence is least. The magnitude of this trade-off was found to be problem dependent; consequently, a universal g that optimizes both operator complexity and the number of iterations required for convergence across all problems cannot be found. It was found, however, that PMIS(g)-CLJP did exhibit good convergence without prohibitively high operator complexity for some problems when problem size is not too large (with g being problem dependent). Future work could include confirmation of the results found here for a parallel implementation of PMIS(g)-RS.

The third attempt to improve AMG involved modifying F-F interpolation. As

shown by De Sterck and Yang in [17], PMIS coarsening combined with *F-F* interpolation can produce good convergence results, but may also lead to large increases in operator complexity and setup time for a variety of problems. PMIS-FF can lead to good convergence since it accurately accounts for strong *F-F* connections without a strongly connected *C*-point in interpolation. This is accomplished by considering all *distance-two C*-points for these *F-F* connections. In doing so, however, interpolation matrix densities, operator complexity, and setup time can be significantly increased. The third modification, PMIS-FF1, seeks to remedy this complication by only adding one *distance-two C*-point when defining interpolation for each strong *F-F* connection without a common strongly connected *C*-point. As illustrated in Chapter 7, this change reduces operator complexity and setup time significantly, while maintaining similar convergence properties as PMIS-FF, for almost all problems tested. This is a strong result, as it indicates that PMIS-FF1 is a robust and scalable algorithm with reasonable operator complexity and execution time for a wide range of problems. Future work should confirm the results found here for larger problem sizes. Parallel implementation of PMIS-FF1 also appears promising, and is a candidate for future research.

Appendix A

Algebra Definitions

Definition A.0.1. Define an *M-matrix* to be an $N \times N$ matrix A that is positive definite $\left(\mathbf{u}^T A\mathbf{u} > 0,\ \forall\ \mathbf{u} \neq \mathbf{0}\right)$, diagonally positive, and off-diagonally non-positive.

Definition A.0.2. Define a matrix a to be (*weakly*) *diagonally dominant* if

$$\sum_{j \neq i}^{n} |a_{ij}| \leq |a_{ii}|\ ,\quad 1 \leq i \leq N. \tag{A.0.1}$$

Definition A.0.3. [5] A *matrix norm* on the set of all $N \times N$ matrices is a real-valued function, $\|\cdot\|$, defined on this set, satisfying for all $N \times N$ matrices A and B and all real numbers α:

i. $\|A\| \geq 0$,

ii. $\|A\| = 0$ if and only if A is O, the matrix with all zero entries,

iii. $\|\alpha A\| = |\alpha|\|A\|$,

iv. $\|A + B\| \leq \|A\| + \|B\|$,

v. $\|AB\| \leq \|A\|\|B\|$.

Definition A.0.4. Define a *natural matrix norm* as:

$$\|A\| = \max_{\mathbf{x}\ s.t.\ \|\mathbf{x}\|=1} \|A\mathbf{x}\|, \tag{A.0.2}$$

where $\|\cdot\|$ is some vector norm on $\mathbb{R}^N$.

This is sometimes also referred to as the *matrix norm induced by the vector norm*, $\|\cdot\|$. Note that it can be shown that any natural norm is a matrix norm.

Definition A.0.5. Define a *vector p-norm* and its associated *matrix p-norm* for all $p \geq 1$ as:

$$\|\mathbf{x}\|_p = (\sum_{i=1}^{N} |x_i|^p)^{\frac{1}{p}}, \tag{A.0.3}$$

$$\|A\|_p = \max_{\mathbf{x} \ s.t. \ \|\mathbf{x}\|_p=1} \|A\mathbf{x}\|_p. \tag{A.0.4}$$

Note that the set of matrix p-norms is a subset of all natural norms.

Definition A.0.6. Define the *discrete L^2 norm* for a d-dimensional domain with uniform grid spacing h as:

$$\|\mathbf{x}^h\|_h = \left(h^d \sum_i \left(x_i^h\right)^2 \right)^{\frac{1}{2}}. \tag{A.0.5}$$

Definition A.0.7. Define the *Euclidean inner product* as:

$$(\mathbf{x}, \mathbf{y})_E = \langle \mathbf{x}, \mathbf{y} \rangle_2 = \sum_i x_i y_i. \tag{A.0.6}$$

Definition A.0.8. Define the *A-inner product* for a symmetric positive definite matrix, A, as:

$$(\mathbf{x}, \mathbf{y})_A = (A\mathbf{x}, \mathbf{y})_E = \langle A\mathbf{x}, \mathbf{y} \rangle_2. \tag{A.0.7}$$

Definition A.0.9. Define the *A-norm* for a symmetric positive definite matrix, A, induced by the A-inner product, as:

$$\|\mathbf{x}\|_A = (A\mathbf{x}, \mathbf{x})_E^{\frac{1}{2}}. \tag{A.0.8}$$

Definition A.0.10. Define the *spectral radius* of a matrix A as:

$$\rho(A) = \max_i |\lambda_i| \ , \ i = 1, \ 2, \ldots, \ N, \tag{A.0.9}$$

where λ_i are the eigenvalues of A.

Bibliography

[1] K. E. Atkinson, An introduction to numerical analysis, second ed., p. 3, John Wiley and Sons, Toronto, 1989.

[2] A. Brandt, Multi-level adaptive solutions to boundary-value problems, Mathematics of Computation **31** (1977), 333–390.

[3] W. L. Briggs, Problem solutions for a mutligrid tutorial – first edition, http://www.web.mit.edu/wax/www/Books/Briggs/briggs.html, retrieved 2006 06 10.

[4] W. L. Briggs, V. E. Henson, and S. F. McCormick, A Multigrid Tutorial, second ed., Society for Industrial and Applied Mathematics, Philadelphia, 2000.

[5] R. L. Burden and J. Douglas Faires, Numerical analysis, fifth ed., PWS Publishing Company, Boston, 1993.

[6] R. P. Fedorenko, The speed of convergence of one iterative process, U.S.S.R. Computational Mathematics and Mathematical Physics **4** (1964), 227–235.

[7] M. Griebel, B. Metsch, D. Oeltz, and M. A. Schweitzer, Coarse grid classification: A parallel coarsening scheme for algebraic multigrid methods, Numerical Linear Algebra with Applications **13** (2006), 193–214.

[8] L. A. Hageman and D.M. Young, Applied iterative methods, pp. 138–141, Academic Press, Inc., New York, 1981.

[9] W. W. Hager, Applied numerical linear algebra, pp. 340–341, Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[10] V. E. Henson and U. Meier Yang, BoomerAMG: A parallel algebraic multigrid solver and preconditioner, Applied Numer. Math **41** (2002), 155–177.

[11] E. Isaacson and H. B. Keller, Analysis of numerical methods, John Wiley and Sons, Inc., New York, 1966.

[12] R. J. Leveque, Finite volume methods for hyperbolic problems, Cambridge University Press, New York, 2002.

[13] M. Luby, A simple parallel algorithm for the maximal independent set problem, SIAM Journal on Computing **15** (1986), 1036–1053.

[14] J. W. Ruge and K. Stuben, Algebraic multigrid, in "Multigrid Methods" by S. F. McCormick (ed), pp. 73–130, Frontiers in Applied Mathematics (SIAM), Philadelphia, 1987.

[15] Y. Saad and M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. **7** (1986), 856–869.

[16] F. L. Stasa, Applied finite element analysis for engineers, CBS College Publishing, New York, 1985.

[17] H. De Sterck and U. Meier Yang, Coarsening and interpolation in algebraic multigrid: a balancing act, Presentation at the 9th Copper Mountain Conference on Multigrid Methods, 2004.

[18] H. De Sterck, U. Meier Yang, and J. J. Heys, Reducing complexity in parallel algebraic mulitgrid preconditioners, SIAM Journal on Matrix Analysis and Applications **27** (2006), 1019–1039.

[19] N. Storey, Electronics, a systems approach, second ed., pp. 602–603, Prentice Hall, Toronto, 1998.

[20] K. Stuben, An introduction to algebraic multigrid, in "Multigrid" by U. Trottenberg, C.W. Oosterlee and A. Schuller, pp. 413–532, Elsevier Academic Press, San Diego, 2001.

[21] U. Trottenberg, C. W. Oosterlee, and A. Schuller, Multigrid, Elsevier Academic Press, San Diego, 2001.

[22] P. R. Turner, Guide to scientific computation, second ed., pp. 212–216, CRC Press LLC, Boca Raton, 2001.

[23] H. A. van der Vorst, Iterative krylov methods for large linear systems, Cambridge University Press, New York, 2003.

[24] The Free Encyclopedia Wikipedia, Greedy algorithm, http://en.wikipedia.org/wiki/Greedy_algorithm, retrieved 2006 06 15.

[25] C. H. Wolters, M. Kuhn, A. Anwander, and S. Reitzinger, A parallel algebraic multigrid solver for finite element based source localization in the human brain, Computing and Visualization in Science **5** (2002), 165–177.

[26] X. Zhao, P. G. Richards, and S. J. Zhang, A parallel algorithm for algebraic multigrid, (2004), 285–290, Proceedings of the 42^{nd} Annual ACM Southeast Regional Conference.