Algebraic Multigrid Methods for Parallel Computing, Systems, and Graphs

by

Tobias M. Jones

B.S., University of Colorado, 2010

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Applied Mathematics
2013
This thesis entitled:
Algebraic Multigrid Methods for Parallel Computing, Systems, and Graphs
written by Tobias M. Jones
has been approved for the Department of Applied Mathematics

Prof. Stephen F. McCormick

Prof. Thomas A. Manteuffel

Date ________________

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.
In modern large-scale supercomputing applications, Algebraic Multigrid (AMG) is a leading choice for solving linear systems. However, on the newest architectures, the relatively high cost of communication versus computation is a concern for the scalability of traditional implementations. Introduced here are Algebraic Multigrid Domain Decomposition (AMG-DD) and Algebraic Multigrid Range Decomposition (AMG-RD) which trade communication for computation by forming composite levels that replace many stages of multilevel communication with local computation using redundant information.

Another open topic in the application of AMG is in the context of solving systems of partial differential equations. Adaptive Smoothed Aggregation was developed as a method to address the potential difficulties with not only generating the aggregates in this setting, but also to generate the kernel components required to efficiently solve these problems. New variants on this approach are introduced that aim to more effectively identify the local and global near null spaces as well as form more robust multilevel solvers.

Historically, AMG was used to solve linear systems that arise from the discretization of differential equations. However, due to the $O(N)$ scalability of the method, it seems natural to investigate it in other contexts that generate large sparse linear systems. Data mining in graph theory applications generate very large, but extremely sparse, linear systems called Graph Laplacians. As a step in the process of targeting AMG for these problems, eigenvectors of matrices formed from graphs are investigated.
Dedication

For my Mother . . . who would have enjoyed reading this
Acknowledgements

It seems like a daunting task to actually acknowledge all the people who helped make this happen. As I sit here typing, I feel like I can hear the music playing because I have already gone on too long . . . but here goes anyway.

Probably the people most responsible (to blame?) for my thesis are my advisor Steve and my co-advisor Tom. I never would have made it to this point without their constant advice and guidance. I am pretty sure I leapt into the deep end without looking, and they were always there to teach and encourage me. Oh, and the skiing and biking, that was good too.

There has also been a series of people throughout the department to thank for their support. Anne Dougherty for believing in me from the beginning, and Ian and Matt for their support wading through the quagmire that is the “system.” Enough thanks cannot be given to all the professors who patiently helped me understand what I didn’t know I needed to know.

I would be remiss to not acknowledge all the help I received from the computational group at CU and the folks in CASC at LLNL: John and Marian, for their willing patience with my endless questions about AMG and SA and benevolent guidance in my attempts to reproduce their work: Rob, who always manged in three days to put more on a white board than I could possibly hope to remember(and that I couldn’t photograph): Geoff for his never-ending series of notes: and Van for his support, and more importantly, his one-liners. Of course, thanks must be given to Annette for her wrangling, and to the denizens of Grandview: Uno, Deuce, Lei, Kuo, David, Jacob, and Jeff.

My family undoubtedly deserves recognition for their patience and support through this entire process. Thanks to my Dad for his guidance, Jimdad for his help with homework, Eileen for
her backup, to my wife Colleen for putting up with me, and my daughter Aubriana for providing perspective. Special thanks must be given to Mary Ann, without whom this manuscript might have been considered an assault on the progress of grammar in today’s society.
Contents

Chapter

1 Introduction 1

2 Multigrid Background 4
   2.1 The Multigrid Cycle .................................. 4
      2.1.1 Relaxation Methods ................................ 4
      2.1.2 Coarse-Grid Correction .......................... 5
      2.1.3 V-Cycle ....................................... 6
   2.2 Geometric Multigrid .................................. 7
   2.3 Algebraic Multigrid .................................... 9
   2.4 Smoothed Aggregation .................................. 17
      2.4.1 Adaptive Smoothed Aggregation .................... 22

3 AMG-DD/RD 24
   3.1 Overview .............................................. 24
      3.1.1 High-Level Description ............................ 24
      3.1.2 Composite-Grid Creation .......................... 26
      3.1.3 Composite-Grid Operators ......................... 28
      3.1.4 AMG-DD .......................................... 29
      3.1.5 Communicating the Residual in AMG-DD ............. 30
      3.1.6 AMG-RD .......................................... 32
7.3 Multigrid for Scale-Free Graphs .............................................. 108

Bibliography ................................................................. 109
Tables

Table

3.1 Convergence of AMG-DD for 9,16,25,36 processors with 5,000 nodes per processor for varying padding ($\eta$) and number of cycles ($\rho$) ........................................... 55

3.2 Convergence of AMG-RD for 9,16,25,36 processors with 5,000 points per processor for varying padding ($\eta$) and number of cycles ($\rho$) ........................................... 56

6.1 Adjacency and Modularity eigenvalues of $pattach1$ with no cliques attached and with cliques attached. Parentheses highlight eigenvalues that have changed significantly with the addition of the cliques. ........................................... 99

6.2 Adjacency and Modularity eigenvalues of $pattach2$ with no cliques attached and with cliques attached. Parentheses highlight eigenvalues that have changed significantly with the addition of the cliques. ........................................... 100

6.3 Most positive eigenpair for which embedded cliques of various size are visible for $pattach1$ and $pattach2$ ................................................................. 104

6.4 Most positive eigenvalues of $A$ and $B$ with locations of associated eigenvector maxima and local graph statistics for $amazon0312$ ........................................... 106
Figures

Figure

2.1 Demonstration of the complementary processes of relaxation and coarse-grid correction 5

2.2 Example of geometric coarsening with coarse nodes colored in blue ............... 8

3.1 Sample composite grid constructed from the Laplace equation on a uniform mesh . . 28

3.2 $W_0 = 100,000$ and $W_0 = 25,000$, respectively ...................................... 44

3.3 Growth of the composite grid as a function of $\eta$ ............................................. 47

3.4 $W_0 = 100,000$, $\eta = 2$, $\rho = 1, 2, 3$ ..................................................... 50

3.5 $W_0 = 25,000$, $\eta = 2$, $\rho = 1, 3$ ......................................................... 51

3.6 $W_0 = 25,000$ and $W_0 = 100,000$, $\eta = 20$, $\rho = 2$ ................................ 51

3.7 Convergence and complexity as functions of uniform padding ....................... 53

3.8 Convergence and complexity as functions of non-uniform padding .................. 54

4.1 Aggregates for standard Poisson ................................................................. 65

4.2 Small aggregates for non-grid-aligned anisotropy ......................................... 65

4.3 Larger aggregates for non-grid-aligned anisotropy ....................................... 66

4.4 Aggregates for a curved anisotropy .............................................................. 66

5.1 Directed graph with 6 vertices ................................................................. 73

5.2 The graph from Figure 5.1 as an undirected graph ....................................... 74

5.3 Graph and degree distribution for $|V| = 20$ ................................................ 77

5.4 Graph and degree distribution for $|V| = 100$ .............................................. 77
5.5 Graph and degree distribution for $|V| = 1,000$ .................................................. 77

6.1 Spectral mapping for the three largest eigenvectors of $A$, for $|V| = 20,100,1000$ .... 82

6.2 Magnitude for eigenvectors associated with the 1st and 5th most positive eigenvalues of $A$ (top row) and $B$ (bottom row) for pattach1, $c = 11$ and $N = 250$. The vertices from the graph generator are ordered by level (distance from the initial hub), as indicated by the vertical gray lines, and the vertices associated with the community structure are listed last. Red dots represent values associated with the $c$-cliques. For $A$ (top left) a decay bound of order $(2/|\lambda_1|)^{\text{level}}$ is plotted to demonstrate that the decay is exponential and related to the size of the eigenvalue, as Theorem 6.3.2(ii) suggests. On the top and bottom right we see that the 11-cliques are large enough to be maximums for the 5th eigenvalue of both matrices, as determined in Theorems 6.3.2(i), 6.3.3(i). Note the logarithmic vertical axes have different scales. .... 101

6.3 Magnitude for eigenvectors associated with the 1st and 5th most positive eigenvalues of $A$ (top row) and $B$ (bottom row) for pattach2, $c = 16$ and $N = 167$. The vertices from the graph generator are ordered by level (distance from the initial hub), as indicated by the vertical gray lines, and the vertices associated with the community structure are listed last. Red dots represent values associated with the $c$-cliques. On the left, we see exponential decay as we move away from the vertex of maximal degree. On the right we see that the 16-cliques are large enough to be maximums for the 5th eigenvalue, as determined in Theorems 6.3.2(i), 6.3.3(i). Note the logarithmic vertical axes have different scales. ................. 102

6.4 Decay and sparsity plots for real-world example amazon0312 .............................................. 103
Chapter 1

Introduction

What are typically referred to in modern literature as multigrid methods were originally developed in the context of solving partial differential equations on structured grids. The earliest forms of the methodology that can be viewed as the basis for modern variations were introduced to the wider community by Brandt in 1973, and it was referred to as the multi-level adaptive technique (MLAT) [20]. One of the most attractive features of these multi-level approaches is that they offer the possibility of a solver for discretizations of partial differential equations (PDEs) that require only costs proportional to the number of degrees of freedom in the discrete equations.

Forming the solvers required for these optimal methods, however, can become greatly complicated once they are applied to anything but the most basic of problems. Some examples of these complications arise for PDEs with non-constant coefficients or problems discretized on unstructured meshes. In response to these complications, algebraic multigrid (AMG) [23, 24, 90, 91] was developed. AMG generates solvers for these linear systems based on the matrix, and possibly some underlying assumptions about the discretization of the PDE. Another similar multi-level methodology developed for solving the linear systems resulting from discretization of PDEs is smoothed aggregation (SA) [96, 97].

As research into utilizing multigrid in new areas continues, new challenges continue to arise. This thesis presents new research addressing these problems in three different contexts: the scalability of AMG on highly parallel architectures; the solution of systems of PDEs using adaptive SA; and finally AMG for treating the linear systems generated from graph problems.
The layout of the thesis is as follows. In Chapter 2, background is introduced for multigrid methods. Basic theory is provided to demonstrate the efficacy and the computational scalability of a variety of multigrid methods. Particular attention is paid to developing a comprehensive understanding of the algorithms for AMG and SA, since in later chapters this knowledge is fundamental to understanding the development of new models and algorithms.

A new variation on the AMG algorithm is outlined in Chapter 3. While AMG is known to scale optimally with respect to computational cost, algorithms developed for solving problems on modern architectures must now be concerned with how they scale with respect to communication. The new method is a domain decomposition multigrid algorithm that trades computation for communication.

Chapter 4 introduces a new variant of adaptive SA.

Chapter 5 is devoted to providing background and notation surrounding graphs. Thorough descriptions and examples are used to motivate terminology and understanding of the difficulties of a particular subset of graphs, referred to as scale-free. Examples are then provided of applications that utilize solutions of linear systems formed from this class of graphs. Finally, some useful linear algebra is introduced that is helpful in understanding the investigation of eigenspaces of graph-associated matrices derived from scale-free graphs presented in Chapter 6.

Classical variants of AMG struggle when directly applied to graph-associated matrices. AMG methods were developed to solve problems on meshes that typically come from some low-dimensional space, where connections are local. This strong locality allows the construction of coarse problems with similar sparsity structure, which is essential to assembling AMG solvers. The linear equations resulting from graph-mining applications, however, do not have this same locality. The structure being modeled often contains nodes with high connectivity and very little underlying structure; therefore there is usually no straightforward embedding to a low-dimensional space that preserves locality. Optimal AMG methods (as well as most optimal preconditioners) are developed based on an understanding of the eigenspaces of the linear systems. Chapter 6 investigates eigenspaces of matrices currently employed in data mining. Fully describing these eigenspaces is a fundamental
step in developing AMG preconditioners for linear systems used for data mining.

Finally, conclusions are made in Chapter 7 about the three main foci of the research. Future directions of all three approaches are also discussed.
Chapter 2

Multigrid Background

2.1 The Multigrid Cycle

2.1.1 Relaxation Methods

Multigrid methods have grown more complicated over the years, but, no matter the context, an underlying juxtaposition still forms the basis for all approaches. When using iterative methods to solve linear systems, especially those rising from the discretization of elliptic PDEs, simple relaxation techniques like Weighted Jacobi (WJ), Gauss-Seidel (GS), and Symmetric Successive over Relaxation (SSOR) all suffer from stalled convergence after several iterations [92].

Much investigation has been done into the cause of this stalled convergence. A simple illustration is to consider using WJ to solve a bilinear finite element discretization of Poisson \((\Delta u = f)\) on a uniform grid. It can be shown that the error that is quickly reduced by relaxation is rich in eigenvectors associated with eigenvalues at the upper end of the spectrum. Conversely, error that is slow to converge under iterations is rich in eigenvectors associated with eigenvalues at the lower end of the spectrum. For simple cases, this can be shown straightforwardly using Fourier analysis.

For more complicated PDEs, or on unstructured grids, this may not be as readily apparent using straightforward analysis techniques, but it is typically still true that the upper end of the spectrum is reduced quickly through relaxation for the linear systems arising from the discretization of elliptic PDEs. For clarity, error that is easily addressed through the process of relaxation is defined as oscillatory error, and error that is slow to converge as smooth error. In the setting where
little is known about the discretization, such error is referred to as *algebraically oscillatory and smooth error*, that is, error $e$ such that $||Ae||$ is “large” or “small” respectively. Note that this is consistent with the simple case where smooth error is associated with the low end of the spectrum.

(For a more thorough treatment of this, please refer to [8].)

2.1.2 Coarse-Grid Correction

Since relaxation can be used to reduce oscillatory error, it now remains to address the smooth components of the error. To do this, a complementary process is created that can efficiently address the smooth error. Referring back to the simple example of 1D Poisson on a grid of size $n$ nodes, then the eigenvectors associated with the lower half of the spectrum look like $\omega_{k,i} = \sin\left(\frac{ik\pi}{n}\right)$, where $k < \frac{n}{2}$. However, taking this same function on a grid with half as many nodes, now the upper half of the smooth error is oscillatory on this coarser grid. This oscillatory error is treated by relaxation fairly quickly, and any smooth error left is taken again to another coarser grid. This process is illustrated in Figure 2.1.

Once the goal of the two complementary processes of relaxation and coarse-grid correction is clear, then what must be asked is how to form a problem that exposes the desired coarse-grid error. Suppose the true solution, $x$, to the problem $Ax = b$ is known. Then an approximate solution, $x^{(n)}$, gives the error $e^{(n)} = x - x^{(n)}$. However, computing $e^{(n)}$ would require already having the
solution in hand. But, since the true solution solves the original problem, then the error must satisfy

\[ A e^{(n)} = A \left( x - x^{(n)} \right) = b - A x^{(n)}. \]

Coarse-grid correction thus involves solving for the error left after smoothing, and is obtained via the residual correction equation in a coarser space, \( A_c e_c = r_c \), where the residual is defined as \( r^{(n)} = b - A x^{(n)} \). Assuming that there are only two levels, then one iteration of a multigrid process takes the form:

1. Smooth on the original problem: \( A x = b \).
2. Form the residual: \( r = b - A x \).
3. Solve the coarse problem: \( A_c e_c = r_c \).
4. Update the solution: \( x = x + e_c \).
5. Smooth again on the original problem: \( A x = b \).

2.1.3 V-Cycle

This complementary process is then applied in a recursive fashion, where the work of smoothing on the coarser levels is much reduced in comparison to that on the fine level. With this in mind, to construct a full multigrid process, first a sequence of \( L + 1 \) grids must be selected: \( \Omega_0, \Omega_1, \ldots, \Omega_L \), where \( \Omega_0 \) is a grid with \( n_0 \) nodes and a solution, \( x_0 \in \mathbb{R}^{n_0} \), is sought that satisfies \( A x_0 = b \), \( A \in \mathbb{R}^{n_0 \times n_0} \). For most applications where multigrid is used as an iterative solver, the assumption is made that \( A \) is a symmetric positive definite (SPD) matrix. While this is not strictly required, it makes a good deal of the theory and implementation more straightforward.

For each grid, \( \Omega_l \), noting that \( \Omega_l \) has \( n_l \) nodes, it is now necessary to define an operator \( A_l \in \mathbb{R}^{n_l \times n_l} \). Along with each pair of grids, it is also necessary to define \( P_l : \mathbb{R}^{n_{l+1}} \rightarrow \mathbb{R}^{n_l} \), referred to as the Prolongation Operator. The last step is to define the sequence of Restriction Operators,
So the components required for a multigrid process composed of $L + 1$ levels are
\[
\begin{cases}
\Omega_l \in \mathbb{R}^{n_l \times n_l} & \text{for } l = 0, \ldots, L \\
A_l \in \mathbb{R}^{n_l \times n_l} & \text{for } l = 0, \ldots, L \\
P_l \in \mathbb{R}^{n_l \times n_{l+1}} & \text{for } l = 0, \ldots, L - 1 \\
R_l \in \mathbb{R}^{n_{l+1} \times n_l} & \text{for } l = 0, \ldots, L - 1.
\end{cases}
\]

With these components in hand, it is now possible to define the Multigrid V-Cycle, Algorithm 2.1.1. At this point the natural question is to ask how to form these components, $\Omega_l, A_l, P_l, R_l$.

**Algorithm 2.1.1** $x_l = \text{VCycle}(\nu_1, \nu_2, x_l, b_l, l)$ to update $x_l$.

**Require:** $\nu_1$: Number of pre-smoothes.
**Require:** $\nu_2$: Number of post-smoothes.
**Require:** $x_l$: Initial guess.
**Require:** $b_l$: Right-hand side.

- if $l == L$ then
  - Direct solve $A_l x_l = b_l$
- else
  - Relax $\nu_1$ times on $A_l x_l = b_l$.
  - Form residual $r_l = b_l - A_l x_l$.
  - Obtain $e_{l+1} = \text{VCycle}(\nu_1, \nu_2, 0, R_l r_l, l + 1)$.
  - Update $x_l = x_l + P_l e_{l+1}$.
  - Relax $\nu_2$ times on $A_l x_l = b_l$.
- end if

**return** $x_l$.

The choices here is exactly what differentiates the various multigrid methods, so exposition of this is delayed to the later sections where the details of each method are provided.

### 2.2 Geometric Multigrid

As outlined in earlier sections, efficient multigrid algorithms are founded on the principles of eliminating smooth error on fine grids and using coarse grids to reduce the smooth components of error. The first use of these two principles to form solvers for PDEs was in early Geometric Multigrid (GMG) codes. The earliest practical work in geometric multigrid methods was introduced by Brandt in the 1973 approach called the multi-level adaptive technique (MLAT) [20].
In the beginning GMG was intended as a fast solver for linear systems resulting from the discretization of PDEs. As such, the assumption was made that it was possible to formulate the discretization of the problem on any grid that the user chooses. This approach is also based upon the knowledge that the nodes are associated with actual nodes in some physical space. This is most easily understood in the setting of uniform spacing of nodes on some simple grid, say a square in $\mathbb{R}^2$.

Making the assumption that uniform coarsening allows coarse errors to be accurately represented, then, if the assumption is that the grid was formulated on a grid of $(2^n - 1) \times (2^n - 1)$ nodes, the coarse grid formed by skipping every other node is a grid of $(2^{n-1} - 1) \times (2^{n-1} - 1)$ nodes. Thinking about this elementwise, these grids generate $2^n \times 2^n$ elements, which is often convenient when combining GMG with finite element discretizations. Uniform coarsening of this nature in two dimensions is demonstrated in Figure 2.2.

It is possible to show that for a finite element discretization of a purely isotropic Laplacian problem, this coarse-grid selection leads to an optimal multigrid solver, a fact that is useful in later sections when trying to understand what an ideal graded coarsening might look like. While this outlines a purely geometric approach to formulating the coarse grids, the question remains as to how to formulate the inter-grid transfer operators.

One simple approach would be to select injection, $R$, from the fine grid to the coarse grid, and, for a 2-D problem, select bilinear interpolation $P_1$ from the coarse grid to the fine grid. In
this way all the components have now been formulated that allow the V-Cycle algorithm (2.1.1) to be used as a solver. However, there are many points in this process that lead to complications. To formulate any geometric approach, one needs to have an understanding of the geometry for the problem and the ability to reformulate problems on the coarser grids.

There are also many instances of problems for which standard coarsening leads to poor convergence. Take, for instance, a finite element discretization of $\nabla \cdot \mathbf{a}\nabla u = f$, with matrix $\mathbf{a}$ inducing some anisotropy. If this is a grid-aligned anisotropy then it is fairly simple to construct a coarsening that leads to a cycle with good properties, but in many cases this is a hard problem of its own. Because formulation of problems on non-regular grids can also quickly lead to very difficult-to-construct cycles, these problems led to the development of algebraic versions of multigrid that have a less restrictive construction process.

2.3 Algebraic Multigrid

Algebraic multigrid (AMG) was formulated in the early 1980s in a series of work involving Brandt et al [24,90,91]. AMG was developed primarily for the solution of the linear systems derived from discretizing PDEs on non-uniform grids. In this setting, it is no longer easy or always even possible to rely on the underlying geometry to coarsen the nodes.

AMG has since proven to be a very robust solver. It has been shown to be effective in many more complicated settings, such as solving problems with discontinuous coefficients and irregular geometries. Part of the attractiveness of this method is that it can be applied with no direct knowledge of the PDE or the geometry on which the problem was formulated. While AMG is often discussed in the setting of symmetric linear systems, it extends with some care to non-symmetric problems.

However, AMG is not purely an out-of-the box solver for any sparse linear system that a user could imagine. It was originally developed for the solution of the linear systems that arise from discretization of scalar elliptic PDEs. The assumption is made that the near nullspace can be well approximated by the constant, and this is then used to ensure that the correction from the coarse
grid is appropriate. When using AMG methods to solve problems that violate this assumption, care
must be taken in the application of the process to ensure the desired convergence and scalability
properties are retained.

The term AMG is now used in reference to a wider class of algorithms that rely on this mul-
tiscale approach. Here a template is provided for implementing a standard Ruge-Stüber approach
to AMG that was developed by McCormick, Brandt and Ruge in their original work. The purpose
is to provide a framework for an AMG implementation that later work in Chapter 3 utilizes. It
also provides an introduction to the terminology used in AMG approaches.

The initial information provided to an AMG solver is a \( n \times n \) linear system \( A_0 \), where the
desired result is the solution, \( u_0 \), to the equation \( A_0 u_0 = f_0 \). (Notice the use of subscript 0 to denote
the finest “level”, in contrast to its usage to denote the coarsest level in geometric approaches. This
reflects common usage in the algebraic methods community.) As in all multigrid methods the goal
is to leverage relaxation and coarse-grid correction. However, without geometry to guide the coarse-
grid selection, it is necessary to use the matrix to guide selection of coarse-grid nodes.

AMG methods generally rely on some common underlying principles. The first is that al-
gebraically smooth errors have small residuals, \( r \approx A e \approx 0 \). The second assumption is that
neighboring nodes that have “strong” connections can be considered good neighbors in that they
can be used to locally approximate such smooth error. Finally, the assumption is made that locally
the error is almost constant, and interpolation must be selected to match this smooth error.

Since the goal is to solve a system of size \( n \times n \), then the assumption is made (for a scalar
problem) that there is some underlying grid, \( \Omega_0 \), with \( n \) nodes in it, and node \( i \) has connections
specified by the off diagonals in the matrix \( A_0 \). Now the goal is to form a coarse grid, denoted \( \Omega_1 \),
and a prolongation operator, \( P_1^0 : \Omega_1 \to \Omega_0 \). Once these two elements are created the standard
Galerkin coarse-grid operator \( A_1 = (P_1^0)^T A_0 P_1^0 \) can be formed. This operation is repeated to form
$L$ levels, and the multigrid hierarchy

\[ \Omega_0, A_0, P_1^0, \]

\[ \Omega_1, A_1, P_2^1 \]

\[ \vdots \]

\[ \Omega_{L-1}, A_{L-1}, P_{L}^{L-1} \]

\[ \Omega_L, A_L \]

is defined. Grids with a larger number are denoted a “coarser” grid, and the process is stopped at some level, $L$, where the problem is small enough that it can be treated efficiently with a direct solver or simple relaxation. Notice that no prolongation operator needs to be formed to this coarsest level.

To coarsen a level in the hierarchy, two items need to be formed: a coarse grid and prolongation from that coarse grid to the current one. For a given level, $l$, this is accomplished in three steps:

1. Form the strength of connection matrix $S_l$.

2. Perform the Fine-Coarse (F-C) splitting of the nodes.

3. Construct $P_{l+1}^l$ from $S_l$ and the (F-C) splitting.

The strength of connection matrix (SOC), denoted $S_l$, defines which nodes in the grid $\Omega_l$ are strongly connected to each other. Recall that the connections from a node, $i$, are defined by the non zero off diagonals in the $i^{th}$ rows and columns of $A_l$. This means that $S_l$ has the same dimensions as $A_l$, and has a subset of the non zero sparsity structure of $A_l$, which is convenient for implementation purposes.

The strength of connection matrix, $S_l$, and its transpose, $S_l^T$, tell us how strongly a node depends on the other nodes in the grid, and how strongly other nodes in the grid depend on this node. $S_l$ is typically constructed row by row.
The goal is to decide which nodes in the grid are strongly connected to each other. The hope is that, if two elements are strongly connected, the solution can be obtained at one node, and then interpolated solution will be fairly accurate at its neighbor. Let $S^i$ be the $i^{th}$ row of $S_l$. To fill out the row, the value

$$\alpha^i = \max_{1 \leq j \leq n, j \neq i} |a_{ij}|$$

is first computed. This measures the largest connection for this node, and all other connections are deemed to be strong relative to this connection. A relative strength approach allows connections to be judged locally, which is beneficial for many problems. Now the elements of the row $S^i$ are defined by

$$s_{ij} = \begin{cases} 
1, & i = j \\
1, & -a_{ij} > \theta \alpha_i \\
0, & \text{otherwise.}
\end{cases}$$

Selection of the value $\theta$ is very important in the performance of the algorithm. Selecting too small a value can lead to connections being deemed strongly connected when they should not be, and vice versa. Typically, one value is appropriate for wide varieties of problems.

Once the SOC matrix has been formed, it is then necessary to divide the grid into two disjoint sets, fine-grid nodes (F nodes) and coarse-grid nodes (C nodes). C nodes are repeated on the coarser grid, while F nodes are removed from the coarser nodes, and need to have their values updated in the MG cycle by interpolation. There are two criteria informing the splitting of nodes to which attention must be paid. The first is that for every fine node, each of its strong connections must be a coarse node, or strongly connected to a coarse node. The second is that the set of coarse nodes is a maximal subset with the property that no two C nodes are strongly connected to each other. This is often impossible in practice, so the goal is to enforce the first condition while trying to satisfy the second.

The process for satisfying these constraints is best understood by seeing it in practice. Through this explanation note the level $l$ is dropped from $S$ to make the notation more concise.
(1) Assign an initial weight to all the nodes, and then add them to a list $U$. Define

$$w(i) = K_1^i + K_2^i,$$

where

$$K_1^i = \text{Number of nodes strongly connected to } i,$$

and

$$K_2^i = \text{Number of nodes } i \text{ connects strongly to.}$$

At this point it is convenient to sort the list from largest to smallest; however, many elements may have the same weight, so the order in which elements are added to the list affects the coarsening. This is one example of when having knowledge of the problem can matter, i.e., if the nodes are ordered lexicographically, the coarsening obtained is often preferable to selecting the ordering of the nodes at random.

(2) Label the node $i \in U$ with the largest weight as a coarse node and add it to the list of coarse nodes $C$.

(3) Label all nodes strongly connected to $i$ from the list $U$ as fine-grid nodes and then add them to the list of fine nodes $F$. The nodes strongly connected to $i$ form the set

$$S^T_i = \{j : 1 \leq j \leq n, [S^T]_{ij} \neq 0\},$$

where $S^T$ is just the transpose of the SOC matrix. Now move the set $J$ to $F$ where

$$J = S^T_i \cap U.$$ 

(4) All nodes left in $U$ that are strongly connected to each element just added to $F$ have their weights increased by the number of items just added to $F$ to which they are strongly connected.

$$w(k) = w(k) + 1, \ \forall j \in J, k \in S^T_j \cap U.$$
(5) All nodes left in $U$ that $i$ strongly connects to have their weights decreased by 1. Define the set

$$S_i = \{ j : 1 \leq j \leq n, [S]_{ij} \neq 0 \},$$

then

$$w(j) = w(j) - 1, \forall j \in S_i \cap U.$$

(6) If $U$ is not empty, go back to Step 2; otherwise, stop.

Once this process is concluded, it defines the two sets of nodes $F$ and $C$. In practice, this can be slightly more complicated; care must be taken to handle some nodes that have “special” properties. One example of this is nodes that have no connections. Values at these nodes are directly solved for through relaxation and are handled separately.

Once the coarse grid is selected, the second part of the coarsening process can now be performed. At this point it is necessary to construct the interpolation operator. Recall the assumption that relaxation has worked, and that the error left is algebraically smooth. In other words, the error left at this level satisfies the approximation $A_l e_l \approx 0$. Making the assumption that the error, $e_l$, can be solved for on the coarse grid, and interpolated up, leads to the requirement that for one node $i$

$$(P_{l+1}^l e^{l+1})_i = \begin{cases} e_i & i \in C \\ \sum_{j \in C_i} w_{ij} e_j & i \in F. \end{cases}$$

For a fine-grid node this leads to the relation that the error at that node should be able to be written as

$$e_i = \sum_{j \in C_i} w_{ij} e_j,$$  \hspace{1cm} (2.1)$$

where $C_i$ is the set of coarse-grid nodes that strongly influence $i$, $C_i = S_i \cap C$. However, it is not always necessary to ensure that every element that is required to be in $C_i$ is actually on the coarse grid. To address this, it is necessary to find coarse-grid nodes that are appropriate to interpolate from. To aid this, it is necessary to define three more sets of nodes.
(i) $N_i = \text{all the nodes } j \text{ such that in } A_i \text{ that } a_{ij} \neq 0 \text{ and } i \neq j$.

(ii) $F_i^s$ is the set of fine-grid nodes that strongly influence the nodes $i$, which in set notation is

$$F_i^s = S_i \cap F.$$  

(iii) $F_i^w$ is the set of fine-grid nodes that weakly influence the node $i$, which in set notation is

$$F_i^w = N_i - (S_i),$$

where $N = \text{all the nodes on the global fine grid}$.

To find appropriate nodes on the coarse grid to interpolate from, recall the assumption that $Ae \approx 0$ for smooth error, so for any node it is safe to assume that

$$a_{ii}e_i \approx - \sum_{j \in N_i} a_{ij}e_j.$$  

All the points in $N_i$ then can be split into three types

$$a_{ii}e_i \approx - \sum_{j \in C_i} a_{in}e_j - \sum_{n \in F_i^s} a_{in}e_n - \sum_{m \in F_i^w} a_{im}e_m.$$  

(2.2)

Now the work remains to replace the $e_j$ in the second and third terms with relations of either $e_i$ or from $e_j$ for $j \in C_i$. To handle the right-most term it is possible to rely on the smoothness of the error. The fact that they are weak influences means they are small and may be “distributed” to the diagonal, giving the new relation

$$ \left( a_{ii} + \sum_{m \in F_i^w} a_{im} \right) e_i \approx - \sum_{j \in C_i} a_{ij}e_j - \sum_{n \in F_i^s} a_{in}e_n.$$  

(2.3)

For the $m$ elements associated with $F_i^s$, it is possible to instead choose to replace these $e_m$ with their representation from nodes in $C_i$ which can be written as

$$e_n \approx \frac{\sum_{j \in C_i} a_{nj}e_j}{\sum_{k \in C_i} a_{nk}}.$$  

(2.4)

To see where this comes from, note that the numerator $\sum_{j \in C_i} a_{mj}e_j$ can be used as an approximation because this $e_m$ is strongly influenced by the nodes $j$, and the denominator ensures that a
constant will be interpolated appropriately because if \( e_j = c, \forall j \) then
\[
e_n = \frac{\sum_{j \in C_i} a_{nj} c}{\sum_{k \in C_i} a_{nk}} = \frac{c}{\sum_{k \in C_i} a_{nk}} = c.
\]
Now combining (2.3) and (2.4)
\[
\left(a_{ii} + \sum_{m \in F^w_i} a_{im}\right) e_i \approx - \sum_{j \in C_i} a_{ij} e_j - \sum_{n \in F^s_i} \left[\frac{\sum_{j \in C_i} a_{nj} e_j}{\sum_{k \in C_i} a_{nk}}\right],
\]
and factoring the right-hand side to write the outer sums in terms of \( j \in C_i \) gives
\[
\left(a_{ii} + \sum_{m \in F^w_i} a_{im}\right) e_i \approx - \sum_{j \in C_i} \left(a_{ij} + \sum_{n \in F^s_i} \frac{a_{jn} a_{nj}}{\sum_{k \in C_i} a_{nk}}\right) e_j
\]
and moving the left to the right gives
\[
e_i \approx - \sum_{j \in C_i} \left(a_{ij} + \sum_{n \in F^s_i} \frac{a_{jn} a_{nj}}{\sum_{k \in C_i} a_{nk}}\right) e_j.
\] (2.5)
Utilizing the equation for the residual at this node yields
\[
w_{ij} = - \frac{\left(a_{ij} + \sum_{n \in F^s_i} \frac{a_{jn} a_{nj}}{\sum_{k \in C_i} a_{nk}}\right)}{\left(a_{ii} + \sum_{m \in F^w_i} a_{im}\right)} e_j.
\] (2.6)
For convenience, if one were now to order the vector in terms of the grid nodes,
\[c_1, c_2, \ldots, c_{n_c}, f_1, f_2, \ldots, f_{n_f},\]
where \( n_c \) is the number of coarse-grid nodes and \( n_f \) is the number of fine-grid nodes, then the interpolation operator can be written as
\[
P_{l+1}^l = \begin{bmatrix} I_{n_c \times n_c} \\ W_{n_f \times n_c} \end{bmatrix},
\]
where
\[
W = \begin{bmatrix} \vdots & \vdots & \vdots \\ \cdot & \cdot & \cdot \\ \cdot & w_{ij} & \cdot \\ \vdots & \vdots & \vdots \end{bmatrix}.
\]
The coarse operator is now formed by \( A_{l+1} = (P_{l+1}^l)^T A_L P_{l+1}^l \). This process is repeated recursively until some coarse level is formed that is small enough, and this hierarchy is used in the same cycling strategies as any other multigrid solver.
2.4 Smoothed Aggregation

Most algebraic multigrid algorithms can be posited as adaptations of the traditional Ruge-Stüben algorithm, but a noticeable exception is Smoothed Aggregation (SA) algorithms. This idea was first explored in the setting of a two-level algorithm for accelerating a relaxation approach and smoothing transfer operators. [74,95]. For simplicity in the majority of the discussion herein, application to symmetric positive definite matrices is assumed. While this is not always strictly necessary, many of the algorithms and much of the theory is developed with this assumption in mind.

Aggregation-based methods differ from typical AMG methods in that instead of repeating a node on the coarse grid and then interpolating from that node to surrounding neighbors, a group of nodes is collected together and forms an aggregate. Interpolation at nodes is defined based upon the aggregate they were assigned to (and possibly neighboring aggregates).

Let the set of degrees of freedom (DOF) on the \(i\)th grid be denoted by \(\mathcal{D}_i = \{1, 2, \ldots, n_i\}\) where \(n_i\) is the number of degrees of freedom. To perform aggregation the goal is to form \(n_{i+1}\) aggregates \(\{A_1, \ldots, A_{n_{i+1}}\}\) such that

(i) \(\mathcal{D}_i = \bigcup_{j=1}^{n_{i+1}} A_i,\)

(ii) \(A_i \cap A_j = \emptyset, \forall i \neq j,\)

(iii) \(\forall k \in \mathcal{D}_i, \exists j \text{ such that } k \in A_j.\)

Restriction can then be defined by

\[
R = \begin{cases} 
1 & j \in A_i \\
0 & j \notin A_i,
\end{cases}
\]

in other words by piece-wise constants across the aggregates. Coarse-grid operators are then defined by letting \(P = R^T\), and forming \(A_c = RAP\). Mika and Vanek showed that the multi-level algorithm composed of this process, often called unsmoothed aggregation, was a convergent process [74]. However, convergence of these unsmoothed methods is often unsatisfactory, especially in a
multi-level setting, and Vanek showed [95] that by “smoothing” these transfer operators, improved convergence is obtained. Another advantage of smoothed aggregation is that it is easily applied in the setting where the null space of the linear system (generally ignoring boundary conditions) is known to have a complicated kernel that needs more than one component to be addressed on the coarse grids.

Construction of the multigrid hierarchy for smoothed aggregation is similar to AMG, in that it has two components: selection of a coarse grid and construction of interpolation; however, these can look dramatically different. In order to understand the setup for smoothed aggregation it is necessary to understand the two-grid construction process. For this assume the goal is to solve a linear system of the form

\[ A\mathbf{x} = \mathbf{b}, \]

where \( A \) is a matrix of size \( n \times n \) degrees of freedom. It is also assumed that there is a set of kernel components \( K \in \mathbb{R}^{n \times k} \) that must be addressed on the coarse grid. For the purpose of this description the assumption is made that the algorithm is applied to a scalar problem. It is fairly straightforward to generalize the problem to systems where there are multiple DOF that must be treated as one “node.”

To form aggregates, the first step is to once again construct the strength of connection matrix, \( S \), where a typical measure is

\[
s_{ij} = \begin{cases} 
1, & i = j \\
1, & |a_{ij}| > \theta \sqrt{a_{ii}a_{jj}} \\
0, & \text{otherwise.}
\end{cases}
\]

Aggregation of nodes can be viewed as grouping together nodes that could essentially be represented by one item on the coarser grid. Unlike AMG, this does not necessarily require that this representative node is repeated on the fine grid, so there is no strict requirement to have injection at any node. This is typically split into two phases; the first phase, (Algorithm 2.4.1), loops over all the elements on the fine grid. For each unaggregated node, a set is formed from its strongly
connected neighbors. If none of those nodes have been aggregated, a new aggregate is formed from this set. To see this more clearly define the sets

- $\mathcal{R}$, the set of nodes that at any point in time have not been assigned to an aggregate, initially
  $$\mathcal{R} = \mathcal{D}_i.$$  

- $N_i$, the set of nodes that are strongly connected to a node $i$. So if $S$ is the strength-of-connection matrix then
  $$N_i = \{ j : s_{ij} = 1, i \neq j \}.$$  

- Let $k$ be the number of aggregates that have been formed so far, initially set to 0.

**Algorithm 2.4.1 Phase 1**

\[
\begin{align*}
\mathcal{R} &= \mathcal{D}_i \\
k &= 0 \\
\text{for } i = 1, \ldots, n_1 \text{ do} \\
&\quad \text{Form } N_i \\
&\quad \text{if } N_i \cap \mathcal{R} = N_i \text{ then} \\
&\quad \quad A_k = N_i \cup \{i\} \\
&\quad \quad k = k + 1 \\
&\quad \quad \mathcal{R} = \mathcal{R}/(N_i \cup \{i\}) \\
&\quad \text{end if} \\
\text{end for} \\
n_2 &= k
\end{align*}
\]

After Phase 1 is completed, an initial set of aggregates exists, but there is still a subset of nodes, denoted $\tilde{\mathcal{R}}$, which is the set of nodes $\tilde{\mathcal{R}} = \{k_1, \ldots, k_{n_3}\}$ that have yet to be assigned to an aggregate.

The goal of Phase 2, (Algorithms 2.4.2, 2.4.3), is to attach the set of nodes in $\tilde{\mathcal{R}}$ to aggregates in the list $A_1, \ldots, A_{n_2}$. There are two approaches to doing this. The first loops over aggregates and finds all nodes that are left in $\tilde{\mathcal{R}}$ that are strongly connected to a node in that aggregate, and then pulls them into the aggregates. This is denoted Approach 1. The second loops over all the nodes in $\tilde{\mathcal{R}}$ and finds which aggregate they are most strongly connected to, and adds them to that aggregate. This is denoted Approach 2.
Algorithm 2.4.2 Phase 2 - Approach 1

for $i = 1, \ldots, n_2$ do
  Let $m_i$ be the number of elements in $A_i$
  for $j = 1, \ldots, m_i$ do
    Form $N_j$
    Let $P_j = N_j \cap \hat{R}$
    for $k \in P_j$ do
      $A_i = A_i \cup \{k\}$
      $\hat{R} = \hat{R} \setminus \{k\}$
    end for
  end for
end for

Algorithm 2.4.3 Phase 2 - Approach 2

for $i \in \hat{R}$ do
  Find $j$ such that it is the maximizer of $a_{ij}$ and $j \notin \hat{R}$
  Find $k$ such that $j \in A_k$
  $A_k = A_k \cup \{i\}$
  $\hat{R} = \hat{R} \setminus \{i\}$
end for
Once Phases 1 and 2 are complete, all the nodes in the fine grid are either in an aggregate or are part of a special node that does not need to be represented on the coarse grid. As in traditional AMG implementations the goal is now to form interpolation. The first step in the process is to form the tentative prolongation operator $\hat{P}$ from the coarse grid to the fine grid that is subject to two constraints

$$\hat{P}K_c = K \quad \text{and} \quad \hat{P}^T\hat{P} = I,$$

where $K_c$ is the coarse kernel. Recall that $K$ is the $n \times k$ kernel component set that is selected to be an approximation of the null space of the operator on the fine level. The process for satisfying these requirements also forms the coarse-grid kernel $K_c$, and has the attractive property that it can be constructed entirely locally.

Recall that on the fine grid the DOF are organized into $n_1$ aggregates, $\{A_1, \ldots, A_{n_1}\}$. Suppose that $A_i$ has $m_i$ DOF in it. Then selecting the $m_i$ rows of $K$ associated with these DOF gives the matrix $K_i \in \mathbb{R}^{n_1 \times k}$. This local portion of the kernel is then decomposed using a QR algorithm to form

$$Q_iR_i = K_i.$$

These local parts are then used to form

$$\hat{P} = \begin{bmatrix} Q_1 & 0 & 0 & \ldots & 0 \\ 0 & Q_2 & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \ldots & Q_{n_1-1} & 0 \\ 0 & \ldots & 0 & Q_{n_1} \end{bmatrix}, \quad K_c = \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_{n_1-1} \\ R_{n_1} \end{bmatrix},$$

where $\hat{P} \in \mathbb{R}^{n \times n_1}, K_c \in \mathbb{R}^{n_1 \times k}$. Notice that by construction $\hat{P}$ satisfies the constraint $\hat{P}^T\hat{P} = I$. If the coarse-grid problem is selected to be $A_c = \hat{P}^T A \hat{P}$ then this is the unsmoothed aggregation algorithm, which has been shown not to scale in a multi-level setting. Overcorrection can be employed to help with scaling issues, but it does not ameliorate all the issues. One way to understand this is to picture the kernel component selected as purely the constant, then the only null space that
is exactly approximated is piece-wise constants. This is often not a rich enough space, especially
given the larger size of the aggregates.

To address this problem the tentative prolongater is smoothed to form

\[ P = S \hat{P}, \]

where \( S \) is typically selected to be \( S = I - \frac{4}{\det(A)} D^{-1/2}A \), where \( D \) is the (block) diagonal of \( A \). The coarse-grid operator is now defined via \( A_c = P^TAP \), and the coarse-grid kernel is the same as in unsmoothed aggregation. This process is repeated until the series of operators are once again defined for utilization in the multigrid cycle.

Smoothed aggregation has some attractive properties. Like all multigrid methods it is computationally scalable, with the advantage over standard AMG approaches that it allows for more aggressive coarsening. The approach allows the treatment of multiple kernel components which means that it can be applied to many systems of PDEs, and operator complexities stay reasonable. (This typically improves with the spatial dimension of the problem.)

There are two drawbacks to the generic SA algorithms, the first being, that unlike AMG they rely on the symmetry of the linear system explicitly. This can be generalized, but at both cost of implementation and computation. While SA allows the treatment of multiple kernel components, without a priori knowledge of these kernel components this advantage cannot be utilized. To address this issue the kernel can be discovered adaptively; this process is referred to as Adaptive Smoothed Aggregation (aSA).

2.4.1 Adaptive Smoothed Aggregation

In response to the disadvantage that SA requires prior knowledge of the kernel components, and therefore the near null space of the linear system, Brezina et al developed Adaptive Smoothed Aggregation [26]. The approach is based on the desire to have a setup process that can be formed solely using the matrix \( A \).

To accomplish this, the approach appeals to the standard assumption that algebraically
smooth error is left after relaxation, and it is exactly this error that must be addressed by the coarse
grids. Recall, also that the kernel is typically selected to be such that it presents an approximation
to the near null space of the operator $A$. The key to the method then is to have it compute the
slow-to-converge components of the error, and to enrich the kernel with these components.

This enrichment process is done by understanding that a random vector must initially be
rich in all components of the error. If multigrid cycles are applied to the homogeneous problem
$Ax = 0$, then the error left after several cycles is composed of error components that are in the null
space of the method. These components should therefore be added to the kernel, with the hope
that requiring these modes to be in the range of interpolation will then accelerate the solver.
Chapter 3

AMG-DD/RD

3.1 Overview

3.1.1 High-Level Description

The goal of AMG-DD/RD is to replace the V-Cycle iteration from AMG with another iteration that requires less communication. The assumption is that the setup phase of the AMG process has been completed, forming $A_i, P_i, \Omega_i$. Thus, a traditional AMG setup process has already formed a partitioning of the original fine grid, $\Omega_0$, across all the processors, $p = 1, 2, \ldots, N_P$.

Let $D_0^p$ denote the indices of $\Omega_0$ assigned to processor $p$. (When it is clear which processor is referred to, the superscript is dropped and just written $D_0$, and similarly for other notation.) After the AMG setup is complete, each processor owns the region of the fine grid it is assigned, as well as any points in that region that are repeated on any of the coarser grids ($D_0 \cap \Omega_k$, $k = 1, \ldots, L$).

It is possible to associate a graph with the matrices for a grid, and define distance between two nodes, $dist(x, y)$, to be the minimal number of links over all paths that connect them in the graph. The path is allowed to transition between grids by allowing connections in the graph between a node on grid $k + 1$ and its corresponding node on grid $k$. Thus, paths between two nodes, even on the same grid, are often shorter if they pass through coarser grids, a property exploited in the communication scheme detailed below. The definition of a distance from a node, $x$, to a set, $W$, is further extended by defining $dist(x, W)$ to be the minimal distance between $x$ and nodes in the set $W$. 
To understand the AMG-DD algorithm, first consider the ideal setting in which each processor separately solves the original problem, \( A_0 u_0 = f_0 \), on the original fine grid, \( \Omega_0 \). Each processor is assigned its region, \( D_0 \). Now letting each processor solve \( A_0 u_p = f_0 \), the final solution \( u \) can then be given exactly by
\[
    u = \sum_p Q_p u_p. \tag{3.1}
\]
Here, \( \{Q_p\}_{p=1}^{N_p} \) is the partition of unity defined by letting \( Q_p \) be the identity on \( D_0 \) and 0 otherwise. It is generally absurd, however, for each processor to solve the original global equations, so each instead solves a much smaller problem, \( A_p^c u_p^c = f_p^c \), that is fine only near the processor’s domain and increasingly coarse as it extends away to the boundary. More precisely, processor \( p \) iterates on residual equations by applying V-Cycles to
\[
    A_p^c u_p^c = r_p^c = (P_p^c)^T (f_0 - A_0 u_0), \tag{3.2}
\]
where \( P_p^c \) is an interpolation operator from composite-grid to fine-grid functions. The global approximation is then corrected according to
\[
    u_i^{(n+1)} = u_i^{(n)} + \sum_p Q_p P_p^c u_p^c. \tag{3.3}
\]
Notice that, in the ideal case, the composite grid is just the fine grid, so one iteration of AMG-DD is equivalent to traditional V-Cycles if the composite problems are solved by V-Cycles. The questions that remain are how the inaccuracies introduced by the composite grids are accumulated in forming the approximate solution of the original problem, and how much useful work can be done on each processor before a new residual must be calculated.

The dual of this process is AMG-RD, which decomposes the right-hand side (as opposed to the solution), and then reconstructs the approximation as the sum of the global processor components as opposed to local processor pieces. So, in AMG-RD, each processor solves a problem of the form
\[
    A_p^c u_p^c = (P_p^c)^T Q_p \left( f_0 - A_0 u_0^{(n)} \right),
\]
and the approximate solution, $u$, is then formed according to

$$u_{0}^{(n+1)} = u_{0}^{(n)} + \sum_{p} P_{p} u_{c}^{p}.$$ 

Notice that if each processor is assigned the entire fine-grid problem, then under the same assumptions as in the ideal AMG-DD setting, this could also correspond exactly to standard V-Cycles. However, once again it must be verified how well the composite problems combine to represent the original problem.

### 3.1.2 Composite-Grid Creation

A traditional V-Cycle requires each processor to obtain a solution for its own region, $D_{0}^{p}$. This is accomplished by communicating on each grid of the V-Cycle with neighboring processors. The AMG-DD algorithm instead uses global composite grids to approximate the solution on $D_{0}^{p}$. A composite grid is best understood by examining how it is created.

Construction of the composite grid, $\Omega_{c}^{p}$, associated with processor $p$ starts with the nodes that belong to $D_{0}^{p}$ and all nodes that are within distance $\eta_{0}$ from $D_{0}^{p}$

$$D_{0,\eta}^{p} = \{ x \in \Omega_{0} \mid \text{dist}(x, y) \leq \eta_{0} \text{ for some } y \in D_{0}^{p} \}.$$ 

Now, let $D_{1}^{p}$ denote the nodes in $D_{0,\eta}^{p}$ that are repeated on the first coarse grid, $\Omega_{1}$. Adding all the nodes within distance $\eta_{1}$ on grid $\Omega_{1}$ forms the set

$$D_{1,\eta}^{p} = \{ x \in \Omega_{1} \mid \text{dist}(x, y) \leq \eta_{1} \text{ for some } y \in D_{1}^{p} \}.$$ 

Proceeding recursively in this fashion for all grids, $\Omega_{0}, \Omega_{1}, \ldots, \Omega_{L}$, it is possible to obtain the final composite grid

$$\Omega_{c}^{p} = \bigcup_{i=0}^{L} D_{i,\eta}^{p}.$$ 

Notice that, for any node in the composite grid, it is possible to trace a path from that node through the graph of the matrix to a point in $D_{0}^{p}$. 

Theorem 3.1.1. There exists a node \( x_0 \in D^p_0 \) such that the distance to a node \( x_{k,\eta} \in D^p_{k,\eta} \) is bounded by

\[
\text{dist}(x_0, x_{k,\eta}) \leq \sum_{i=0}^{k} \eta_i. \tag{3.4}
\]

Proof. Use induction to show that \( \text{dist}(x_0, x_{k,\eta}) \leq \sum_{i=0}^{k} \eta_i \)

- case: \( k = 0 \)
  By definition, since \( x_{0,\eta} \in D^p_{0,\eta} \), then there must exist \( y \in D^p_0 \) such that \( \text{dist}(x_{0,\eta}, y) < \eta_0 \). Thus, for \( k = 0 \), the bound holds.

- case: \( k = m + 1 \)
  Assume that if \( x_{m,\eta} \in D^p_{m,\eta} \), then there exists \( x_0 \in D^p_0 \) such that \( \text{dist}(x_0, x_{m,\eta}) \leq \sum_{i=0}^{m} \eta_i \).
  Let \( x_{m+1,\eta} \in D^p_{m+1,\eta} \). Then there exists \( x_{m+1} \in D^p_{m+1} \) with \( \text{dist}(x_{m+1}, x_{m+1,\eta}) \leq \eta_{m+1} \).
  But \( x_{m+1} \) has a corresponding node, \( x_{m,\eta} \in D^p_{m,\eta} \), that satisfies the assumption. Hence, we have

\[
\text{dist}(x_0, x_{m+1,\eta}) = \text{dist}(x_0, x_{m,\eta}) + \text{dist}(x_{m,\eta}, x_{m+1,\eta})
\]

\[
\leq \sum_{i=0}^{m} \eta_i + \eta_{m+1}
\]

\[
= \sum_{i=0}^{m+1} \eta_i,
\]

which proves the result.

Theorem 3.1.1 is illustrated by the path

\[
x_0 \rightarrow x_{0,\eta} \downarrow x_1 \rightarrow x_{1,\eta} \downarrow x_2 \rightarrow x_{2,\eta} \downarrow x_3 \rightarrow \ldots \downarrow x_k \rightarrow x_{k,\eta}. \tag{3.5}
\]

Here, a right arrow signifies a path through the graph on the current grid, and a down arrow signifies a transition to a repeated node on a coarser grid.

Notice that this creation process constructs a composite grid, \( \Omega^p_c \) (see Figure 3.1), such that the grid contains the original region of the fine grid assigned to it and a graded mesh that becomes
increasingly coarse further away from the region, until it reaches the boundaries on some coarsest grid, \( i \leq L \). Notice that it is possible to select \( \eta_L \) to ensure that the composite grid reaches all boundaries on the coarsest grid.

### 3.1.3 Composite-Grid Operators

Recall that the coarse grids are formed from the original grid hierarchy that AMG created, so a natural fit is to use parts of the already-created operators. In the previous section, the composite grid, \( \Omega_c \) was defined, and now it is possible to form an operator from the associated function space, \( G_c \), to the fine-grid function space, \( G_0 \), on the global fine grid.

For ease of discussion, first assume that there are only two grids in the original hierarchy. It is now possible to reorder the global fine grid as \( \Omega_0 = \{ D_{0,\eta}, \left( \Omega_0 - D_{0,\eta} \right) \} \), yielding the composite grid \( \Omega_c^p = \{ D_{0,\eta}, \left( \Omega_1 - D_{1}^p \right) \} \). Splitting the prolongation operator \( P_0 \) into blocks corresponding to \( D_1^p \) and \( \Omega_1 - D_1^p \) gives the \( 2 \times 2 \) block

\[
P_0 = \begin{bmatrix}
* & * \\
P_{10} & P_{11}
\end{bmatrix}.
\]

From this form, define the prolongation operator \( P_c : G_c \rightarrow G_0 \) expressed by

\[
P_{p_c} = \begin{bmatrix}
I & 0 \\
\hat{P}_{10} & P_{11}
\end{bmatrix},
\]
where $\hat{P}_{10}$ has had zero columns added to match the size of the identity block above it. Supposing that there are now three grids in the hierarchy, $\Omega_0, \Omega_1, \Omega_2$, then the composite grid would be of the form $\Omega_c = \left\{ D_{0\eta}^p, (D_{1\eta}^p - D_1^p), (\Omega_2 - D_2^p) \right\}$. Now, reordering the operator $P_1$ in terms of $\left\{ D_2^p, (\Omega_1 - D_{1\eta}^p) \right\}$

$$
P_1 = \begin{bmatrix}
\ast & \ast \\
\hat{P}_{21} & \hat{P}_{22}
\end{bmatrix},
$$

then the composite operator takes the form

$$
P^p_c = \begin{bmatrix}
I & 0 \\
\hat{P}_{10} & \hat{P}_{11} \\
\hat{P}_{21} & \hat{P}_{22}
\end{bmatrix}.
$$

For four grids, the operator takes the form

$$
P^p_c = \begin{bmatrix}
I & 0 \\
\hat{P}_{10} & \hat{P}_{11} \\
\hat{P}_{21} & \hat{P}_{22} \\
\hat{P}_{32} & \hat{P}_{33}
\end{bmatrix}.
$$

Continuing in this way to the coarsest grid that covers the whole domain, $\Omega_L$, the prolongation operator from the composite-function space to the global finest-function space can be defined. Thus it is possible to define the composite operator $A^p_c = (P^p_c)^T A_0 P^p_c$ for each processor [11]. In practice, the goal is to utilize a FAC type solver [68], which means that it is not necessary to explicitly form this operator, but the algorithm will behave as if it were. Notice that this composite operator is the natural extension of what happens with a finite element discretization on the composite grid, and the grids can actually be chosen such that it exactly corresponds to the finite element discretization.

### 3.1.4 AMG-DD

The necessary components now exist to define AMG-DD in terms of the composite operators and grids. An AMG-DD cycle (Algorithm 3.1.1) starts with a given global initial guess, $\mathbf{u}_0^{(0)}$, and
right-hand side, $f_0$, partitioned among the processors. All processors then work in parallel to com-
pute their pieces of the global residual and restrict them to their associated pieces of coarser grids.
These residuals are efficiently communicated at appropriate composite grids to other processors
as outlined in the next section. Then each processor solves its composite-grid problem using a
multigrid algorithm based on the original AMG setup, or some other geometric multigrid or AMG
scheme based directly on composite-grid operator $A_c$. The parameter $\rho$ in Algorithm 3.1.1 speci-
fies the number of local (communication-free) cycles that are done before the solution is patched
together and the next iteration is performed.

**Algorithm 3.1.1** $u = \text{AMG-DDCycle}(\rho, u_0, f_0)$ for solving $A_0 u_0 = 0$

**Require:** $\rho$ : Number of on-processor iterations.
**Require:** $u_0$ : Initial guess.
**Require:** $f_0$ : Right-hand side.

Form $r_0 = f_0 - A_0 u_0$.
Restrict the residual to $\Omega_1, \ldots, \Omega_L$.
Communicate the restricted residuals to form $r_p^0$ on each processor.

**Execute in parallel** on all processors $p = 1, \ldots, N_P$
- Set $u_p^0 = 0$.
- Do $\rho$ cycles on $A_p^0 u_p^0 = r_p^0$.
- Update $u_0 \leftarrow u_0 + Q_p u_p^0$.

**return** $u_0$.

### 3.1.5 Communicating the Residual in AMG-DD

Since each processor only needs the residual on its composite grid, communication can be
minimized by starting the residual updates on the coarsest grid, $\Omega_L$, and bubbling the information
up as detailed in Algorithm 3.1.2. The algorithm proceeds up through the hierarchy, exchanging
data with neighbors on each grid. Here it is shown that the algorithm produces the desired residual
update, and later in Section 3.3.2 it is shown that the total amount of data communicated by each
processor is on the same order as the number of points on its boundary. In past work, such as the
algorithms from Mitchell/Bank/Holst, efficient communication of residuals has been an impediment
to scaling.
Algorithm 3.1.2 Collect residual for composite grids

for $k = L \to 0$ do
  Execute in parallel on all processors ($p = 1, 2, \ldots, N_P$)
  if $(\mathcal{D}_0^p \cap \Omega_k) \neq \emptyset$ then
    Identify the set of neighboring processors $\{p_1, \ldots, p_m\}$ that contain points within distance $\eta_k$ of $(\mathcal{D}_0^p \cap \Omega_k)$.
    for $j = 1 \to m$ do
      Find all points $x \in (\mathcal{D}_0^p \cap \Omega_k)$ such that $\text{dist}(x, (\mathcal{D}_0^p \cap \Omega_k)) < \eta_k$, let $\Psi$ be the union of these points.
      Form $\Psi_c$ from $\Psi$ as outlined in Section 3.2.
      Send the residual at all points and grids in $\Psi_c$ to $p_j$.
    end for
  end if
end for

Claim: On grid $\Omega_k$, processor $p$ owns region $\mathcal{D}_0^p \cap \Omega_k$. After the algorithm has executed for this grid, processor $p$ will have the correct residual for all nodes, denoted $\Omega_{c,k}^p$, on a composite grid built from $\mathcal{D}_0^p \cap \Omega_k$.

Proof. The needed information is assumed to be stored off processor; otherwise, the claim is trivially true.

- Suppose $k = L$ (the coarsest grid). Then, since $p$ communicates directly to all processors within $\eta_L$, it trivially has the correct residual for all $x \in \Omega_{c,L}^p$.

- Suppose $k = m$, and assume that each processor has obtained residual information at all nodes on $\Omega_{c,m+1}^p$, the composite grid formed from $\mathcal{D}_0^p \cap \Omega_{m+1}$. By construction, if $x \in \Omega_{c,m}^p$, then there exists $y \in \mathcal{D}_0^p \cap \Omega_m$ and a path of the form

$$y \to y_{m,\eta} \downarrow y_{(m+1)} \to y_{(m+1),\eta} \downarrow y_{(m+2)} \to \cdots \to x.$$ 

Notice, for some processor $q$, that $y_{(m+1)} \in \mathcal{D}_0^q \cap \Omega_{m+1}$ and $x \in \Omega_{c,m+1}^q$. Hence, processor $q$ has residual information for node $x$ by assumption and, since $y_{(m+1)}$ is within distance $\eta_m$, then processor $q$ passes that information to processor $p$.

The claim thus holds. \[\square\]
This claim shows that, after the \( k^{th} \) grid has been updated, each processor has the composite grid defined by the region it owns for that grid. Therefore, after all grids have been updated, each processor obtains the residual information it needs for its composite grid. The cost of this algorithm is detailed in Section 3.3.

3.1.6 AMG-RD

AMG-RD is based on the same composite grids and operators that are formed for AMG-DD. However, instead of patching together the solutions after an iteration, AMG-RD adds together the individual global approximations in each iteration. For the global summation of solutions to make sense, AMG-RD solves residual problems of the form \( A^p u^p = (P^p)^T Q_p r \). This approach relies on the fact that the solutions are smooth outside the region, and can therefore be easily represented and approximated on the composite grid.

\[
\text{Algorithm 3.1.3} \quad u = \text{AMG-RDCycle}(\rho, u_0, f_0) \text{ for solving } A_0 u_0 = f_0
\]

Require: \( \rho \) : Number of on-processor cycles.
Require: \( u_0 \) : Initial guess.
Require: \( f_0 \) : Right-hand side.

Form \( r_0 = f_0 - A_0 u_0 \).
Execute in parallel on all processors \( p = 1, \ldots, N_P \)
Set \( u^p_0 = 0 \).
Do \( \rho \) cycles on \( A^p u^p = (P^p)^T Q_p r_0 \).
Update \( u \leftarrow u_0 + \sum_p u^p_0 \).
return \( u_0 \).

3.2 Theory

This section establishes a convergence theory for AMG-DD and AMG-RD. It is first shown that the two methods are duals of each other in the energy inner product. An abstract theory is then developed that establishes optimal convergence under full regularity and approximation property assumptions on the origin of the global fine-grid matrix equation, \( A u = f \). In particular, it is assumed that \( A \) is symmetric and positive definite, and that this matrix equation was created by standard discretization of a two- or three-dimensional uniformly elliptic operator on a uniform
grid in a region that is either a convex polygon or has smooth boundary.

In what follows, $C$ is used to denote a generic constant, independent of the number of processors and problem dimensions, that may change meaning with each occurrence. Recall that $Q_p$ is the matrix that zeros out the entries in $f$ outside processor $p$’s subdomain, $D_0^p$, but leaves $f$ inside $D_0^p$ alone. As before, let $\Omega_c^i$ and $G_c^i$ denote the respective composite grid and associated function space determined by the global grids, $\Omega_i, i = 0, 1, \ldots, L$. Also, as before, let $P_{pc} : G_c^i \to G_0$ denote the interpolation operator from processor $p$’s composite grid to the fine grid, and write $A_c^p = (P_c^p)^T A P_c^p$. Then, one iteration of AMG-DD starting with a zero initial guess is

$$u \leftarrow \sum_p Q_p P_c^p (A_c^p)^{-1} (P_c^p)^T f.$$  

The error propagation matrix for this method is

$$M_d = I - \sum_p Q_p P_c^p (A_c^p)^{-1} (P_c^p)^T A. \quad (3.6)$$

After incorporating $\nu \geq 0$ iterations of a weighted fine-grid pre-relaxation process into the scheme with weight $0 < \omega < 2$, then the error propagation matrix becomes

$$M_d = (I - \sum_p Q_p P_c^p (A_c^p)^{-1} (P_c^p)^T A) \left( I - \frac{\omega}{||A||} A \right)^{\nu}. \quad (3.7)$$

While in practice this relaxation is not strictly necessary for convergence, it makes the theory more straightforward. Similarly, AMG-RD without fine-grid relaxation is

$$u \leftarrow \sum_p P_c^p (A_c^p)^{-1} (P_c^p)^T Q_p f,$$

with the error propagation matrix

$$M_r = I - \sum_p P_c^p (A_c^p)^{-1} (P_c^p)^T Q_p A. \quad (3.8)$$

With $\nu$ iterations of fine-grid post-relaxation, the error propagation matrix becomes

$$M_r = \left( I - \frac{\omega}{||A||} A \right)^{\nu} (I - \sum_p P_c^p (A_c^p)^{-1} (P_c^p)^T Q_p A). \quad (3.9)$$

Now, since the adjoint of any matrix $B$ in the energy inner product $< A, \cdot >$ is $A^{-1} B^T A$, then it is easy to see by the Euclidean symmetry of $P_c^p (A_c^p)^{-1} (P_c^p)^T$ and $Q_p$ that $M_d$ and $M_r$
are energy adjoints of each other. One take-home from this result is that the range and domain
decomposition approaches exhibit the same energy convergence bounds. Another is that it could
be beneficial to use one of these methods following the other to create a symmetric iteration.

While it may be more natural to measure convergence of either error propagation matrix
in the energy norm, the Euclidean norm of $M_d$ may be simpler by virtue of the orthogonality of
its individual terms. To see this for $M_d$ without pre-relaxation, let $S_p = P^p_c(A^p)^{-1}(P^p_c)^T A$, the
energy-orthogonal projection of fine-grid vectors onto the range of $P^p_c$. Since the $Q_p$ sum to $I$, the relation

$$\| M_d x \|^2 = \sum_p \| Q_p (I - S_p) x \|^2$$

holds. With pre-relaxation, this becomes

$$\| M_d x \|^2 = \sum_p \| Q_p (I - S_p)(I - \frac{\omega}{\| A \|} A)^r x \|^2.$$  

The goal now becomes to show that $\|M_d\|$ is bounded uniformly above by a constant less than 1.

The bottom line here is that it is necessary to know how well composite-grid vectors ap-
proximate fine-grid vectors in the corresponding subdomain. They can, of course, provide no
approximation at all to oscillatory vectors outside $D^p_0$ vectors that are energy orthogonal to the
composite grid. The key here is the presence of $Q_p$, which means that it is sufficient to know the
local accuracy in $D^p_0$, where there is no de-refinement. To avoid estimates of the above sum from
depending on the number of processors, it necessitates the requirement of localizing the effect of $x$
on these terms. The convergence theorem below shows that $I - S_p$ has enough structure to enable
writing $Q_p(I - S_p)x$ in terms of a component of $x$ that is nonzero only on $D^p_0$ and some close
neighborhood about it.

To achieve this structure, the result is restricted to a two-grid setting, but with minimal
padding in the sense that $\eta_0 = 1$. This means that it is possible to assume that each composite grid,
$\Omega^p_c$, consists of $D^p_{0,1}$, the fine grid with uniform mesh spacing $h$ on $D^p_0$ and its nearest neighbors, and
a $2h$ grid beyond that to the boundary in what is denoted the de-refinement region, $\mathcal{D} = \Omega_1 - D^p_1$.
Note that the composite grid has only coarse points in $\mathcal{D}$. It is necessary to begin with two
assumptions that hold for this problem setting. These assumptions are made with regard to a
global coarsening of the fine grid by a factor of 2. In particular, let $P$ (without a subscript) denote
the operator that interpolates from the global coarse grid to the fine grid, and let $P_1^p$ denote the
operator that interpolates from the global coarse grid to $p$'s composite grid. Notice $P_1^p : G_1 \to G_c^p$, and $A_1 = (P_1^p)^T A_c^p P_1^p$.

**Weak approximation property (WAP)** [8]: There exists a constant $C < \infty$ such that,
for all $y \in G_0$, there exists a $z \in G_1$ such that
\[
\|y - Pz\|^2 \leq \frac{C}{\|A\|} < A, y >.
\]

**Strong approximation property (SAP)** [47,102]:

Letting $T_p = I - P_1^p ((P_1^p)^T A_c^p P_1^p)^{-1} (P_1^p)^T A_c^p$, then
\[
|T_p w| \leq \frac{C}{\|A_c^p\|} \|A_c^p w\|, \quad \forall w \in G_c^p.
\]

Next, it is necessary to establish approximate Euclidean orthogonality ($\perp$) between the range
of interpolation, $R(P)$, and its energy-orthogonal ($\perp_A$) complement, $R^\perp_A(P)$.

**Lemma 3.2.1.** (A strengthened Cauchy-Schwarz inequality (SCSI).) There exists a $\xi \in (0, 1]$ such that
\[
|<s, t>| \leq (1 - \xi)\|s\| \cdot \|t\|, \quad \forall s \in R(P), t \in R^\perp_A(P).
\]

**Proof.** Another way to write the SCSI is as follows. Let $P = P(P^T P)^{-\frac{1}{2}}$ so that $R(P) = R(P)$ and $P^T P = I$. Any vector that is energy orthogonal to $R(P)$ can thus be written as $A^{-1} (I - PP^T)x$, so the bound sought becomes
\[
< A^{-1} (I - PP^T)bx, Py > \leq (1 - \xi)\|A^{-1} (I - PP^T)x\| \cdot \|Py\|, \forall x \in G_0, y \in G_1.
\]

But the worst case for $y$ here is that which makes $Py$ equal to the orthogonal projection of
$A^{-1} (I - PP^T)x$ onto $R(P)$, and that gives $Py = PP^T A^{-1} (I - PP^T)x$. This leaves the inequality
\[
< A^{-1} (I - PP^T)x, PP^T A^{-1} (I - PP^T)x > \leq
\]
(1 - \xi)||A^{-1}(I - \mathcal{P}\mathcal{P}^T)x|| \cdot ||\mathcal{P}\mathcal{P}^T A^{-1}(I - \mathcal{P}\mathcal{P}^T)x||, \forall x \in G_0.

Since ||Py|| = ||y|| for any y \in G_1, this becomes

||\mathcal{P}^T A^{-1}(I - \mathcal{P}\mathcal{P}^T)x|| \leq (1 - \xi)||A^{-1}(I - \mathcal{P}\mathcal{P}^T)x||, \forall x \in G_0,

which can be written as

||A^{-1}(I - \mathcal{P}\mathcal{P}^T)x||^2 - ||\mathcal{P}^T A^{-1}(I - \mathcal{P}\mathcal{P}^T)x||^2 \geq (1 - (1 - \xi)^2)||A^{-1}(I - \mathcal{P}\mathcal{P}^T)x||^2, \forall x \in G_0.

A little more algebra then leads to the problem of showing that there exists a constant C such that

||A^{-1}(I - \mathcal{P}\mathcal{P}^T)x||^2 \leq C||(I - \mathcal{P}\mathcal{P}^T)A^{-1}(I - \mathcal{P}\mathcal{P}^T)x||^2, \forall x \in G_0.

If this bound were established, then the SCSI follows with \xi = 1 - \sqrt{1 - 1/C}. (It is possible to choose C \geq 1, ensuring that 0 < \xi \leq 1.) To this end, note that the right-hand side is bounded from below as follows

C||A^{-1}(I - \mathcal{P}\mathcal{P}^T)x||^2 = C < [(I - \mathcal{P}\mathcal{P}^T)A^{-1}(I - \mathcal{P}\mathcal{P}^T)]^2x, x >
\geq C \frac{1}{||A||^2} < (I - \mathcal{P}\mathcal{P}^T)(I - \mathcal{P}\mathcal{P}^T)x, x >
= \frac{C}{||A||^2}||A^{-1}(I - \mathcal{P}\mathcal{P}^T)x||^2, \forall x.

All that remains is to establish the bound

||A^{-1}(I - \mathcal{P}\mathcal{P}^T)x||^2 \leq \frac{C}{||A||^2}||(I - \mathcal{P}\mathcal{P}^T)x||^2, \forall x \in G_0.

At this point it is possible to appeal to the SAP (3.11) and symmetry of (I - \mathcal{P}\mathcal{P}^T):

||A^{-1}(I - \mathcal{P}\mathcal{P}^T)x||^2 \leq ||A^{-1}(I - \mathcal{P}\mathcal{P}^T)||^2||x||^2
= ||(I - \mathcal{P}\mathcal{P}^T)A^{-1}||^2||x||^2
= \max_{y \neq 0} \frac{||(I - \mathcal{P}\mathcal{P}^T)A^{-1}y||^2}{||y||^2} ||x||^2
= \max_{(A^{-1}y) \neq 0} \frac{||(I - \mathcal{P}\mathcal{P}^T)(A^{-1}y)||^2}{||A(A^{-1}y)||^2} ||x||^2
\leq \frac{C}{||A||^2}||x||^2,
and now simply replace $x$ by $(I - \mathcal{P}\mathcal{P}^T)x$ to obtain the desired bound.

\[ ||M_d x||^2 \leq \epsilon ||x||^2, \quad \forall x \in G_0. \]

**Theorem 3.2.1. (Two-Grid Convergence)** AMG-DD converges uniformly in the Euclidean norm in the sense that there exists an $\epsilon < 1$, independent of $h$ and $N_p$, such that

**Proof.** The proof starts by establishing that relaxation either converges well or produces an error with a small relative Rayleigh quotient. Assume for this purpose that relaxation has stagnated in the sense that the squared Euclidean norm of the error in iteration $\mu \leq \nu$ converges slowly in that there exists a $\delta$ bounded uniformly above by 1 such that

\[ ||(I - \frac{\omega}{||A||} A)^\mu x||^2 \geq (1 - \delta)||I - \frac{\omega}{||A||} A)^\mu - 1 x||^2. \]

Then a little algebra shows that $y \equiv (I - \frac{\omega}{||A||} A)^\mu - 1 x$ has a small Rayleigh quotient

\[ \frac{\langle Ay, y \rangle}{\langle y, y \rangle} \leq \frac{\delta ||A||}{\omega(2 - \omega)}. \quad (3.13) \]

Next, it must be shown that the component, $t = (I - \mathcal{P}(P^TAP)^{-1}P^TA)y$, of $y$ that is energy orthogonal to $\mathcal{R}(P)$ must be relatively small in the Euclidean sense

\[ ||t||^2 \leq \frac{C}{\xi ||A||} < Ay, y >. \quad (3.14) \]

This bound then combines with (3.13) to prove that

\[ ||t||^2 \leq \frac{C\delta}{\xi \omega(2 - \omega)} ||y||^2. \quad (3.15) \]

To establish (3.14), first note, by the SCSI, that any $Px \in \mathcal{R}(P)$ satisfies

\[ ||t - Px||^2 = ||t||^2 - 2 < t, Px > + ||Px||^2 \]

\[ \geq ||t||^2 + ||Px||^2 - 2(1 - \xi)||t||||Px|| \]

\[ = \xi(||t||^2 + ||Px||^2) + (1 - \xi) (||t||^2 + ||Px||^2 - 2||t||||Px||) \]

\[ \geq \xi ||t||^2, \]
where the last inequality follows because strictly positive terms \((\xi \in (0, 1))\) can be dropped.

Thus, by the WAP with \(t\) replacing \(y\), the hope is that
\[
\frac{\langle At, t \rangle}{\langle t, t \rangle} \geq \frac{\xi \|A\|}{C}.
\]
Bound (3.14) then follows by writing \(y = s + t\), where \(s = P(P^TAP)^{-1}P^T Ay\), and noting that
\[
\langle Ay, y \rangle = \langle As, s \rangle + \langle At, t \rangle \geq \langle At, t \rangle \geq \frac{\xi \|A\|}{C} \langle t, t \rangle.
\]

Now, since relaxation cannot increase the error, either relaxation by itself leads to an error reduction of at least a factor of \((1 - \delta)^{\nu}\), or else (3.15) holds. This strong sense of smoothness of the error is the connection between the Euclidean and energy vector spaces that is needed. The next step is to establish that the composite-grid step effectively reduces error that exhibits this strong smoothness property. Specifically, the aim now is to prove the general property that there exists a constant \(c < \infty\) such that
\[
\sum_p \|Q_p(I - S_p)t\|^2 \leq c\|t\|^2, \quad \forall t \in \mathcal{R}^{1/A}(P).
\] (3.16)

It would then be possible to conclude that the method either converges by relaxation alone, or that relaxation is halted when it stalls, and instead appeal to the convergence bound
\[
\|M_x\|^2 = \sum_p \|Q_p(I - S_p)(I - \frac{\omega}{\|A\|}A)^{\nu}x\|^2 = \sum_p \|Q_p(I - S_p)t\|^2 \leq c\|t\|^2 \leq c\|x\|^2,
\]
where \(\epsilon = \frac{c\xi\delta}{\xi\omega(2-\omega)}\) can be made as small as desired by choosing \(\delta\) small enough. The last equality here follows because \(y = (I - \frac{\omega}{\|A\|})^{\nu}Ax\) differs from \(t\) by \(s \in \mathcal{R}(P) \subset \mathcal{R}(P^p)\), which is in the null space of \(I - S_p\). To complete the proof of this Theorem, it thus suffices to establish (3.16).

To this end, note that a little algebra yields
\[
\sum_p \|Q_p(I - S_p)t\|^2 \leq 2\|t\|^2 + 2\sum_p \|Q_pS_p t\|^2.
\]

Focusing on the last term of this inequality, for a given \(p\), define \(r^p_c = (P^p_c)^T At\). It suffices now to establish that
\[
\|Q_pP_c^p(A_c^p)^{-1}r^p_c\| \leq \frac{C}{\|A_c^p\|}\|r^p_c\|.
\] (3.17)
Since \(|Q_pP_c|^2\) = 1, this would follow if it could be proven that

\[ ||\tau_p|| \leq \frac{C}{||A_c||} ||A_c \tau_p||, \]  

where \(\tau_p = (A_c)^{-1}r_c^p\). Since \(T_p\tau_p = \tau_p\), and \(P = P_cP_1\), then a little algebra shows that \(\tau_p \in R(P_1)^{-1}A_c\), with superscript \(\perp A_c\) denoting \(A_c\)-orthogonal complement. But (3.18) follows directly from the SAP, so (3.17) has been established.

\(P\) and \(P_c\) agree outside \(D_0\), so \(P^TAt = 0\) implies that \(r_c^p\) is nonzero only on \(D_0\) and its nearest coarse-grid neighbors. Thus, \(||(P_c)^TAt||\) only involves values of \(t\) on that extended set and its nearest coarse-grid neighbors. Let \(t_{loc}^p\) denote \(t\) restricted to that doubly extended \(D_0\) region. At this point (3.17) can be used to conclude that

\[ ||Q_pP_c(A_c)^{-1}P_cAt|| \leq \frac{C}{||A_c||} ||A|| ||t_{loc}^p||. \]

Finally, noticing that \(||A|| ||A_c||\) is bounded

\[ \sum_p ||Q_pP_c(A_c)^{-1}(P_c)^TAt||^2 \leq 2||t||^2 + \sum_p \frac{C^2}{\min_p ||A_c||^2} ||A||^2 ||t_{loc}^p||^2 \leq C||t||^2. \]

The last inequality follows because the support of a particular \(t_{loc}^p\) overlaps, at most, a small number of other such supports (e.g., nine in two dimensions for small-enough padding). Therefore (3.16) has been established and, thus, (3.14), which completes the proof.

\[ \square \]

**Remark 3.2.1.** Theorem 1 establishes uniform convergence of AMG-DD in the Euclidean norm. A direct consequence of this result is uniform convergence of its dual, AMG-RD, in the residual norm

\[ ||M_r||_{A^2} = ||AM_rA^{-1}|| = ||M_r^T|| \leq \epsilon < 1. \]

### 3.3 Scaling Models

One of the first tools in judging the performance of a parallel algorithm is forming a model that allows exploration of the performance on different architectures. Here models are developed that give indications of how AMG and AMG-DD/RD are expected to scale. Much of the analysis
of the scalability of a parallel implementation of AMG is based on the work by Gahvari et al [42]. Although the selection of three parameters to model the costs is an old approach, in their work Gahvari et al established the relative costs for architectures of machines that can be used to create accurate AMG models. Throughout this section, three main parameters are utilized:

1. **α**: The cost of latency on the device per message. For now, it is assumed that this is a constant that does not depend on the distance of connections.

2. **β**: Inverse bandwidth cost or, more generically, the cost per amount of data sent.

3. **γ**: The flop rate of the machine, that is, the amount of work per computation.

Based on data in Table 2 from [42], the assumption is made that **α, β, γ** have the following relations for the models

\[ α = 10^4 γ \quad β = 10γ . \]

For the purpose of the model, it is safe to assume, as before, that the problem being modeled is a nine-point discretization of a Laplacian in two dimensions, which allows prediction of stencil patterns in a full-coarsening scenario. Also note that the models just describe the cost of V-Cycles; setup modeling is omitted at this time.

### 3.3.1 AMG V-Cycle Cost

A standard process for modeling a parallel algorithm is to analyze the cost on the most expensive processor, and this structure is utilized in this analysis. In its most basic form, the model of one AMG V-Cycle can be written as

\[ T_{amg} = T_{\text{latency}} + T_{\text{comm}} + T_{\text{comp}} . \]

The communication terms can be modeled as

\[ T_{\text{latency}} + T_{\text{comm}} = \sum_{i=0}^{L} (αm_i + βq_i) , \]
where $L + 1$ is the number of grids in the system (note that $L \approx \log(N)$, with $N$ the number of unknowns in the system), $m_i$ is the number of messages sent on grid $i$, and $q_i$ is the amount of data sent on grid $i$. To calculate actual numbers, the $V(1,1)$-Cycle (Algorithm 2.1.1) is modeled.

To actually specify the parameters $m_i, q_i$, it is necessary to completely understand the structure of the operators in the multigrid hierarchy. For this analysis, standard geometric full coarsening is assumed, so complexity and operators are identical on all grids.

### 3.3.1.1 Communication Costs per AMG V-Cycle

For calculating the parameters, let $\Gamma_i$ denote the number of unknowns on the boundary of the processor of interest on grid $i$. Since the goal is to model the amount of information required to advance computation, then, to be precise, if the number of nodes on a side for a processor is $r$, the amount of information that needs to be gathered is $4r + 4$. However, the effect of the four corner nodes is negligible, so it is safe to assume that, given $n_i$ nodes on a side on grid $i$, the boundary has size

$$\Gamma_i = 4n_i.$$

This also leads to the convenient relation $\Gamma_i = \frac{\Gamma_0}{2^i}$. The $m_i, q_i$ parameters can now be split into their components for relaxation, residual calculation, restriction, interpolation, and coarse-grid solve as follows

- $m_{i,\text{relax}} = 8, \quad q_{i,\text{relax}} = \Gamma_i$;
- $m_{i,\text{residual}} = 8, \quad q_{i,\text{residual}} = \Gamma_i$;
- $m_{i,\text{restrict}} = 8, \quad q_{i,\text{restrict}} = \Gamma_i/2$;
- $m_{i,\text{interp}} = 8, \quad q_{i,\text{interp}} = \Gamma_i/2$;
- $m_{L,\text{solve}} = 8, \quad q_{L,\text{solve}} = \Gamma_L$.

First, assemble the latencies

$$m_i = \begin{cases} 
2m_{i,\text{relax}} + m_{i,\text{residual}} + m_{i,\text{restrict}} + m_{i,\text{interp}} & i < L \\
2m_{L,\text{solve}} & i = L.
\end{cases}$$
For the case being investigated

\[ m_i = \begin{cases} 
2 \cdot 8 + 8 + 8 + 8 & i < L \\
2 \cdot 8 & i = L.
\end{cases} \]

The latency cost can then be represented as

\[ T_{\text{latency}} = \alpha \left( \sum_{i=0}^{L-1} 40 + 2 \cdot 8 \right) \]
\[ = \alpha (40L + 16). \]

The amount of data sent is

\[ q_i = \begin{cases} 
2q_{i,\text{relax}} + q_{i,\text{residual}} + q_{i,\text{restrict}} + q_{i,\text{interp}} & i < L \\
2 \cdot q_{L,\text{solve}} & i = L,
\end{cases} \]

which, for this case, is

\[ q_i = \begin{cases} 
2\Gamma_i + \Gamma_i + \frac{\Gamma_i}{2} + \frac{\Gamma_i}{2} & i < L \\
2\Gamma_L & i = L.
\end{cases} \]

The inverse bandwidth cost can be modeled as

\[ T_{\text{comm}} = \beta \left( \sum_{i=0}^{L-1} 4\Gamma_i + 2\Gamma_L \right) \]
\[ = \beta \left( \sum_{i=0}^{L-1} \frac{\Gamma_0}{2^i} + \frac{\Gamma_0}{2^L} \right) \]
\[ = \beta \Gamma_0 \left( 4 \left[ \frac{1 - (1/2)L}{1 - (1/2)} \right] + \frac{2}{2^L} \right) \]
\[ \approx 8\beta \Gamma_0. \]

### 3.3.1.2 Computation Costs per AMG V-Cycle

Assuming that there are \( W_0 = n^2 \) nodes on the finest grid, then the number of nodes, \( W_i \), on the \( i^{th} \) grid is

\[ W_i = \frac{W_0}{2^{2i}}. \]
Computation cost is then based on the amount of work done on each grid. So, for each grid, the model for the work in terms of the number of unknowns on individual grids is

- Relaxation (Gauss-Seidel) : 18 operations
- Restriction (Ideal) : $\frac{18}{4}$ operations
- Interpolation (Ideal) : $\frac{18}{4}$ operations
- Residual calculation : 18 operations
- Coarse-grid solve : 2 relaxations = 36 operations.

This translates to per-grid costs of

$$N^i_{relax} = 18W_i, \quad N^i_{restrict} = \frac{18}{4}W_i, \quad N^i_{interp} = \frac{18}{4}W_i,$$
$$N^i_{residual} = 18W_i, \quad N^i_{solve} = 36WL.$$

So the total computational cost is

$$T_{comp} = \gamma \left( \sum_{i=0}^{L-1} 2N^i_{relax} + \sum_{i=0}^{L-1} N^i_{residual} + \sum_{i=0}^{L-1} N^i_{restrict} + \sum_{i=0}^{L-1} N^i_{interp} + N_{solve} \right)$$
$$= \gamma W_0 \left[ 63 \left( \frac{1 - (1/4)^L}{1 - (1/4)} \right) + \frac{36}{2^{2L}} \right] \approx 84\gamma W_0.$$

### 3.3.1.3 Graphs for AMG Costs

For the purpose of modeling, the scale is defined by setting $\gamma = 1$. For a problem with $N_p$ processors and a limit of $k$ unknowns per processor, letting $n = \lceil \sqrt{k} \rceil$ then yields $L = log_4(P \cdot n^2)$, $\Gamma_0 = 4n$, and $W_0 = n^2$. Using the derived functions for the AMG costs, the models for a $V(1,1)$-Cycle for $P = 2^i$, $i = 1, \ldots, 20$, while holding the number of unknowns per processor constant yields Figure 3.2. Notice that this figure shows that the only term that grows significantly as the number of processors increases is latency.
Figure 3.2: \( W_0 = 100,000 \) and \( W_0 = 25,000 \), respectively
3.3.2 Derivation of Cost for AMG-DD Cycle

As in the derivation of costs for AMG, once again the problem is broken down into three parts

\[ T_{mdd} = T_{\text{latency}} + T_{\text{comm}} + T_{\text{comp}}. \]

First, the specific form of the algorithm being modeled must be stated. The assumption here is that there has been an AMG setup already, and the parts of the matrix needed to solve the composite problems have already been distributed. Therefore, the restriction is made that this model just describes the solve cycles of AMG-DD from Algorithm 3.1.1.

3.3.2.1 Size of Problem on a Processor

To model the computational cost of the AMG-DD algorithm, the first step is to calculate the number of unknowns needed to form a composite problem. This is a function of the number of unknowns assigned to a processor, \(|D_p^0|\), and the padding on each grid, \(\eta_i\). It is important to note that AMG-DD requires more memory than AMG to solve the same problem, so AMG can solve somewhat larger problems on any given machine, depending on the size of the padding.

As described earlier, the pad amount is the amount of nearest neighbors that are added to a problem on each grid. To develop a closed form for this, start with the assumption that the processor owns a region of the original problem of size \(n \times n\). The model of the length of the side of each problem on each grid, for \(i = 0, \ldots, L\), where \(\eta\) is a constant padding amount, is then

\[ l_k = \frac{n}{2^k} + \sum_{i=0}^{k} \frac{2\eta}{2^i}. \]

Each of these new side lengths defines a region on each grid whose sizes are given by

\[ |D_{0,\eta}| = l_0^2, \quad |D_{1,\eta}| = l_1^2, \quad |D_{2,\eta}| = l_2^2, \quad \ldots \quad |D_{L-1,\eta}| = l_{L-1}^2, \quad |D_L| = |\Omega_L|. \]

To get the total size of the composite grid, it is easier to first imagine that there are only two grids, and then see that the size of the composite grid, \(|\Omega_c|\), is the padded fine region combined with the
coarse grid outside of that region

\[ |\Omega_c| = |\mathcal{D}_{0,\eta}| + \left| \left( \mathcal{D}_{1,\eta} - \frac{\mathcal{D}_{0,\eta}}{2^d} \right) \right|, \]

where \( d \) is the dimension of the original problem (in this case, \( d = 2 \)). Now, using this same logic on all grids yields

\[ |\Omega_c| = |\mathcal{D}_{0,\eta}| + \left| \left( \mathcal{D}_{1,\eta} - \frac{\mathcal{D}_{0,\eta}}{2^d} \right) \right| + \cdots + \left| \left( \mathcal{D}_{L-1,\eta} - \frac{\mathcal{D}_{L-2,\eta}}{2^d} \right) \right| + \left| \left( \Omega_L - \frac{\mathcal{D}_{L-1,\eta}}{2^d} \right) \right| \]

\[ = \left( 1 - \frac{1}{2^d} \right) l_0^2 + \left( 1 - \frac{1}{2^d} \right) l_1^2 + \cdots + \left( 1 - \frac{1}{2^d} \right) l_{L-1}^2 + |\Omega_L| \]

\[ = \left( 1 - \frac{1}{2^d} \right) \sum_{k=0}^{L-1} l_k^2 + |\Omega_L| \]

\[ = \left( 1 - \frac{1}{2^d} \right) \left[ \frac{4}{3} (n - 2\eta)^2 (1 - (1/4)^L) + 16(n\eta - 2\eta^2)(1 - (1/2)^L) + 16L\eta^2 \right] + |\Omega_L|. \]

Notice that the size of the composite grid depends on: \( n^2 \), the size of the fine-grid region assigned to a processor; \( N_p \), the number of processors on the machine; and \( \eta \), the size of the padding region. Plotting this for several different fixed numbers of processors, where the number of unknowns on a processor is \( W_0 = 10,000 \), Figure 3.3 shows how it scales with \( \eta \).

Notice that Figure 3.3 shows that keeping \( \eta \) small, especially on the order of one or two, means that the size of the composite grid is essentially on the same order as the size of the problem that would be assigned to a processor in a typical parallel implementation. Changing the scale of the processors has very little effect on the growth of the composite problem size, especially for small padding amounts.

### 3.3.2.2 Communication Costs per AMG-DD Cycle

There are three main points of communication with the AMG-DD cycle: residual calculation; residual restriction; and communication of the composite residuals. Residual calculation and
Figure 3.3: Growth of the composite grid as a function of $\eta$
restriction, respectively, cost

\[
m_{\text{residual}} = 8, \quad q_{\text{residual}} = \Gamma_0
\]
\[
m_{\text{restriction}} = 8L, \quad q_{\text{restriction}} = \frac{\Gamma_0}{2} \left( \frac{1-(1/2)^L}{1-1/2} \right).
\]

For the communication of the residual, Algorithm 3.1.2 is modeled. Since nearest-neighbor communication is all that is required on each grid, then the latency cost is just

\[
m_{\text{res.comm}} = \sum_{i=0}^{L} 8 = 8(L + 1).
\]

Bounding the amount of data that must be sent in one cycle is more complicated. Given a fixed side length on a processor, \( n \), and a fixed padding, \( \eta \), on each grid, then, counting up from the coarsest grid, the amount of data communicated in the “composite boundary grid” is

\[
K_{L-i} = 4\eta n \left( \sum_{j=0}^{i} \left[ \sum_{l=0}^{j} \frac{1}{4^l} \right] \frac{1}{2^{L-j}} \right).
\]

The total amount of data sent over one cycle is then the sum over all grids

\[
q_{\text{res.comm}} = \eta n \sum_{i=0}^{L} \left( \sum_{j=0}^{i} \left[ \sum_{l=0}^{j} \frac{1}{4^l} \right] \frac{1}{2^{L-j}} \right)
\]

\[
\leq 4\eta n \sum_{i=0}^{L} \left( \sum_{j=0}^{i} \frac{4}{3} \frac{1}{2^{L-j}} \right)
\]

\[
\leq \frac{16}{3} \eta n \sum_{i=0}^{L} 2 \frac{1}{2^{L-i}}
\]

\[
\leq \frac{64}{3} \eta n.
\]

Notice that for the residual communication routine, the total amount of data that must be communicated is on the order of the size of the boundary on a processor.

So the total communication costs are

\[
N_{\text{latency}} = (8 + 8L + 8(L + 1)) = 16(L + 1)
\]

and

\[
N_{\text{comm}} = \Gamma_0 + \frac{\Gamma_0}{2} \left( \frac{1-(1/2)^L}{1-1/2} \right) + \frac{64}{3} \eta n \approx 2\Gamma_0 + \frac{64}{3} \eta n.
\]
3.3.2.3 Computation Costs per AMG-DD Cycle

Using the work from the AMG derivations and the assumption that $W_0$ is the size of the fine grid on a processor, it is understood that the residual computation costs

$$N_{\text{residual.comp}} = 19W_0.$$ 

Restriction requires 18 operations, and must be done for each grid, so

$$N_{\text{restrict.comp}} = \sum_{i=1}^{L} \frac{18W_0}{2^{2i}} = 18W_0 \left( \frac{1 - (1/4)^L}{1 - (1/4)} \right) \approx 36W_0.$$ 

Finally, solving the composite problems can be estimated in the same fashion as for an AMG V-Cycle, except that $W_0$ is replaced with the calculated size of the composite grid, $|\Omega_c|$, and then the parameter $\rho$ that represents the number of cycles done between calculations is added

$$N_{\text{MLDD.comp}} = \rho |\Omega_c| \left[ 63 \left( \frac{1 - (1/4)^L}{1 - (1/4)} \right) + \frac{36}{2^{2L}} \right],$$

where $\rho$ is usually on the order of two to three. This yields a total computation cost of

$$N_{\text{comp}} = 19W_0 + 18W_0 \left( \frac{1 - (1/4)^L}{1 - (1/4)} \right) + \rho |\Omega_c| \left[ 63 \left( \frac{1 - (1/4)^L}{1 - (1/4)} \right) + \frac{36}{2^{2L}} \right] \approx 43W_0 + \alpha 84|\Omega_c|.$$ 

3.3.2.4 Explicitly Calculating the Terms

Once again it can be assumed that $\alpha = 10^4 \gamma$, $\beta = 10 \gamma$

$$T_{\text{mldd}} \approx (10^4 \gamma N_{\text{latency}} + 10^1 \gamma N_{\text{comm}} + \gamma N_{\text{comp}}).$$

With $\gamma = 1$, the total times are now

$$T_{\text{latency}} = 10^4 \cdot 16(L + 1), \quad (3.19)$$

$$T_{\text{comm}} = 10 \left( \Gamma_0 + \frac{\Gamma_0}{2} \left( \frac{1 - (1/2)^L}{1 - 1/2} \right) + \left( 1 - \frac{1}{2^{2d}} \right) \Gamma_0 \frac{(L + 1)(L + 2)}{2} \right), \quad (3.20)$$

$$T_{\text{comp}} = 19W_0 + 18W_0 \left( \frac{1 - (1/4)^L}{1 - (1/4)} \right) + \rho |\Omega_c| \left[ 63 \left( \frac{1 - (1/4)^L}{1 - (1/4)} \right) + \frac{36}{2^{2L}} \right]. \quad (3.21)$$
3.3.2.5 Graphs for AMG-DD

Using these derivations, several scaling models (Figure 3.4) are presented for AMG-DD cycles for a select sample of parameters. First, padding is fixed at $\eta = 2$, the number of nodes per processor are fixed at $W_0 = 100,000$, and then the model is shown for $\rho = 1, 3$.

Reducing the number of nodes per processor to $W_0 = 25,000$, while using the same $\eta, \rho$ parameters, yields Figure 3.5. Finally, the scaling characteristics for larger paddings are modeled with $\eta = 20$, fixing $\rho = 2$, for processor sizes $W_0 = 100,000$ and $W_0 = 25,000$ (Figure 3.6).

Figures 3.4, 3.5, and 3.6 show that, as desired, the AMG-DD algorithm achieves the goal of reducing communication for computation. While the communication does increase as the number of processors increases, the computation cost is now what drives the overall time per iteration of the algorithm. The amount of computation that has been added to each processor, as shown in Section 5.2.3, is on the same order of magnitude as the standard approach, so these figures also serve to highlight the balance between computation and communication that already exists in traditional AMG implementations.
Figure 3.5: $W_0 = 25,000, \eta = 2, \rho = 1, 3$

Figure 3.6: $W_0 = 25,000$ and $W_0 = 100,000, \eta = 20, \rho = 2$
3.4 Numerics

One question that needs to be answered is how well these composite grids can be expected to approximate the solution on the global patches. Since this method is utilized to replace a V-Cycle, the convergence rate of the AMG-DD cycle is compared to that of a V-Cycle to solve global problems of the same size. The results are presented in the context of weak scaling, that is, the number of unknowns per processor is fixed as the number of processors is increased.

Figure 3.7 compares optimal convergence of the AMG-DD cycle, with $\eta$ constant for each grid, to a standard AMG V-Cycle. To achieve this optimal convergence, each processor’s problem is solved as well as possible before doing any communication. This would, of course, only make sense if computation were actually “free.” In Figure 3.7, the complexity of the problem on each processor is highlighted as the padding increases for the case of uniform padding, which is presented in relation to the number of unknowns in the global fine grid.

Figure 3.7 indicates that the composite problems can give good approximation to the global solution and, as expected, increasing the overlap between the processors enhances the convergence rate for the algorithm. However, convergence must be understood in the context of the attendant complexity, which can be thought of as the number of nonzeros in the problem assigned to each processor. The complexity graph shows that, for minimal padding, the size of the problem on each processor is almost independent of the size of the global problem, which is directly proportional to the number of processors. One question this leads to is how important is it to maintain the high level of padding outside of the fine-grid region.

Figure 3.8 is the same test as Figure 3.7, but now only $\eta_0$ is allowed to grow, while $\eta_k$ is fixed at 2 for all $k > 0$. Figure 3.8 shows that increasing the padding only on the fine grid greatly reduces its effectiveness; however, growth of the problem in this setting is now directly tied just to the size of the fine-grid patch on each processor, and is nearly independent of the number of processors and the global problem size.
Figure 3.7: Convergence and complexity as functions of uniform padding
Figure 3.8: Convergence and complexity as functions of non-uniform padding
In Table 3.1, sample convergence rates are presented for parameters that are commensurate with the scaling models created earlier for AMG-DD/RD. For these results, each processor was assigned 5,000 points in the original fine grid, and then allotted minimal padding. Also, the number of local solves were kept to a minimal amount between communications. One important takeaway from these results is that they appear to be almost independent of the number of processors that were selected. There is mild growth as the number of processors increases, but it is similar to growth exhibited by AMG. Table 3.1 also indicates that, while preforming two cycles on each processor before the residual is calculated is beneficial, the benefit of any additional cycles is very mild. The table also suggests that any padding past $\eta = 2$ does not generate significant increase in
convergence of the overall iteration. The results for AMG-RD in Table 3.2 are, as expected, nearly identical to those for AMG-DD.

Table 3.2: Convergence of AMG-RD for 9, 16, 25, 36 processors with 5,000 points per processor for varying padding ($\eta$) and number of cycles ($\rho$)
4.1 Algebraic Multigrid for Systems

AMG and SA have proved to be optimal for solving many linear systems arising from discretization of scalar PDEs. However, in practice, solvers also need to be optimal for the solution of problems arising from systems of PDEs, which are composed of \( m \) unknowns connected by a system of equations. Each unknown refers to an individual physical quantity; for instance, in three dimensions, \( u, v, w \) refer respectively to the \( x-, y-, z-\)components of velocity. Historically, there are two ways of handling this problem: unknown and nodal approaches.

The unknown-based approach can be traced back to \([89,90]\). Assume that the problem is formulated on a grid, \( \Omega \), with \( n \times n \) nodes. If there are \( m \) unknowns, then the discretization results in matrix \( A_U \) that has \( m \times m \) blocks of size \( n \times n \). Labeling the unknowns \( u_1, \ldots, u_m \), then the unknown approach is predicated on the unknowns being ordered (or at least mapped) to form

\[
A_U = \begin{bmatrix}
A_{u_1u_1} & A_{u_1u_2} & \cdots & A_{u_1u_m} \\
\vdots & \ddots & \ddots & \vdots \\
A_{u_mu_1} & \cdots & A_{u_mu_{m-1}} & A_{u_mu_m}
\end{bmatrix}.
\]

Notice that non zeros in a block are the discretization of connections within the PDE coupling those blocks. Therefore, \( A_{u_iu_j} \) is only non zero if \( u_i, u_j \) are coupled in the PDE.

The unknown approach then constructs an interpolation operator for each diagonal block as
if it were a scalar problem, and then forms the associated matrix

\[
P = \begin{bmatrix}
P_{u1} & 0 & \cdots & 0 \\
0 & P_{u2} & \cdots & 0 \\
\vdots & \ddots & \ddots & \vdots \\
0 & \cdots & 0 & P_{um}
\end{bmatrix}.
\]

The coarse-grid operator is then formed using the typical Galerkin variational formulation. This unknown method typically works well when the off-diagonal couplings are small and AMG has little trouble inverting each of the diagonal blocks.

Instead, an alternative approach is to group the unknowns together at each nodal point as outlined in [89,90], and explored further in [29,46]. This creates a matrix \( A_N \) composed of \( n \times n \) blocks of size \( m \times m \)

\[
A_N = \begin{bmatrix}
A_{11} & A_{12} & \cdots & A_{1m} \\
\vdots & \ddots & \ddots & \vdots \\
A_{m1} & \cdots & A_{m(m-1)} & A_{mm}
\end{bmatrix}.
\]

Coarsening, or aggregation, is then based on a \( m \times m \) matrix that measures how strongly the nodes are connected. One measure that is commonly used in SA is

\[
S_{ij} = \rho \left( \frac{A_{ii}^{-1/2}A_{ij}A_{jj}^{-1/2}}{2} \right),
\]

where this measure is reasonable because of the assumption that \( A_N \) is SPD. In this fashion aggregates are formed by combining all degrees of freedom at nodes together. This approach is a natural combination with SA, since implementation of SA in scalar settings on coarse grids requires nodal coarsening if more than one component exists in the kernel.

To find a better SA solver for systems of PDEs there are two major elements to be addressed. The first open question surrounds what are the correct aggregates for the nodes and the unknowns. Both unknown and nodal methods have strict requirements on the aggregates that need to be formed, and one motivation for having another approach is presented in the next section. The second question surrounds what is the best approach for adaptively selecting the kernel components.
For a system like linear elasticity where the rigid-body modes can be used to generate a kernel these approaches are easy to apply, but, for the wide range of systems that can be imagined, the kernel is not as readily apparent.

4.2 Motivation

One of the motives behind the work here is to develop an effective approach for determining aggregates for PDE systems. Consider a 2-D elasticity-type operator, whose energy norm has the form

$$
\frac{1}{2} \left( \left\langle \nabla \cdot \begin{bmatrix} u \\ v \end{bmatrix}, \nabla \cdot \begin{bmatrix} u \\ v \end{bmatrix} \right\rangle + \epsilon \left\langle \nabla \times \begin{bmatrix} u \\ v \end{bmatrix}, \nabla \times \begin{bmatrix} u \\ v \end{bmatrix} \right\rangle \right)^{\frac{1}{2}} = \left( \|u_x + v_y\|^2 + \epsilon \|u_y + v_x\|^2 \right)^{\frac{1}{2}},
$$

where $\epsilon > 0$ is much smaller than 1. Clearly, two of the near null space components are

$$
\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}.
$$

However, to understand the near null space more thoroughly, suppose $u = f(x)g'(y)$ and $v = -f'(x)g(y)$ (where $'$ indicates derivative with respect to the variable of the function). Then this vector function is in the null space of the first term of the energy norm

$$
u_x + v_y = \frac{\partial}{\partial x} (f(x)g'(y)) + \frac{\partial}{\partial y} (-f'(x)g(y))
= f'(x)g'(y) - f'(x)g(y)
= 0.
$$

To understand the second term, consider the more specific vector function of this form given by

$$
\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \begin{bmatrix} \sin(M\pi x) \\ -M\pi \cos(M\pi x)y \end{bmatrix},
$$

where $M$ is an integer comparable to the number, $n$, of nodes in each coordinate direction in the discretization (i.e., $\frac{n}{2} \leq M \leq n$). Note then that $\|\epsilon(-u_y + v_x)\| = \mathcal{O}(M^2\pi^2\epsilon)$. This represents an increase in the energy norm over its $L_2$ norm of $\mathcal{O}(M\pi)$ by a factor of $\mathcal{O}(M\pi\epsilon)$. Comparing this factor to the expansion factor of $\mathcal{O}(n\pi)$ for the vector function $(\sin(n\pi x), 0)^T$, the function in (4.1) is considered to be in the near null space of the operator because it is slow to converge. This is
especially true if $M$ is somewhat small relative to $n$ (e.g., $M = \frac{n}{2}$). The point here is that this function is not smooth enough for $v$ in $x$ or $y$ to interpolate within $v$ itself, so interpolation between $u$ and $v$ is essential. This could be done using the fact that the near null space component is a rotation that is not necessarily smooth within $u$ or $v$ in any way, or by simply using the stencil based on assuming that $u_x + v_y = 0$.

### 4.3 Aggregation

The idea presented here is inspired by “affinity”-based coarsening [62, 87] that uses several relaxed vectors to determine pairs of nodes that are strongly connected. This process of choosing the coarse level can then be followed by Adaptive AMG [26] or Bootstrap AMG [21], both of which determine interpolation weights based on a series of vectors that are initially relaxed and later subjected to the evolving AMG method itself. Another relaxation-based coarsening approach to AMG is Compatible Relaxation [25, 61], which uses one relaxed vector to inform the decision about what coarse grid is suitable for reducing the error that remains after relaxation.

The aim here is to develop a unified process of determining effective aggregates and interpolation weights at once, based not just on pairwise strength of connection, but rather on strongly interconnected sets of nodes. This is accomplished by forming a rectangular matrix whose columns are the vectors of values of several relaxed random vectors at a set of potential aggregate nodes. The aggregate is first pruned down to a set of strongly interconnected nodes by applying a rotated Gram-Schmidt-type process to the rows. A suitable aggregate basis is then determined by a similar process applied to the columns.

For a given symmetric positive definite matrix, $A \in \mathbb{R}^{n \times n}$, and vector, $b \in \mathbb{R}^n$, consider the linear system

$$Ax = b.$$  \hspace{1cm} (4.2)

Assume a given standard relaxation scheme, like Gauss-Seidel or weighted Jacobi (or their block variants for systems), with error propagation matrix $M$. For each node let $x_i \in \mathbb{R}^n$ be the delta
function that is 1 at node $i$ and zero elsewhere. Perform $\nu$ relaxation steps (where $\nu$ is some small number) to form

$$y_i = M^{\nu} x_i \text{ for } i = 1, \ldots, m.$$ 

Picking $\nu$ to be a small number is relevant here, because $y_i$ are intended to represent the nodes that are strongly connected in the resulting AMG scheme. Therefore $\nu$ should be on the same order as the number of relaxations on each level. While it is not actually formed, this set defines the global matrix $Y \in \mathbb{R}^{n \times m}$ given by

$$Y = \begin{bmatrix} y_1 & y_2 & \cdots & y_m \end{bmatrix}.$$ 

Using this data, the next step is to form aggregates that are strongly connected, where “strength” is not just restricted to matrix connections, but with the hope that the inner products between delta functions can be used to tell information about the spread of error in the low eigenmodes. The first step in aggregation is to pick a seed. There are many possible strategies for this: selecting randomly from unaggregated nodes; selecting from the top of the list in order; using the strength-of-connection matrix; using some Compatible Relaxation inspired approach, i.e., taking the average across all the test vectors; and then picking the one with the highest value under the assumption this is one possible source of slow-to-converge error. In practice the aggregation is less sensitive to these choices than other approaches because once a seed is selected it may be discarded in deference to a better seed.

Once the seed has been selected, it is necessary to pick some nodes, $A$, in its neighborhood (including the seed itself) as the potential aggregate. How this might be done is an open question, but the current approach selects everything within two steps of the seed in the graph of matrix $A$. It is, of course, necessary here to consider complexity issues in making these choices.

Letting $k = |A|$, a window of $Y$ is determined by selecting the $k$ rows that correspond to the nodes in $A$, yielding the $k \times m$ matrix $Z$. To determine if $A$, or a pared subset of it, is a suitable
aggregate, consider the $k \times k$ normalized Gram matrix

$$
G = \begin{bmatrix}
|\langle \bar{z}_1, \bar{z}_1 \rangle| & |\langle \bar{z}_1, \bar{z}_2 \rangle| & \cdots & |\langle \bar{z}_1, \bar{z}_k \rangle| \\
|\langle \bar{z}_2, \bar{z}_1 \rangle| & |\langle \bar{z}_2, \bar{z}_2 \rangle| & \cdots & |\langle \bar{z}_2, \bar{z}_k \rangle| \\
\vdots & \ddots & \vdots \\
|\langle \bar{z}_k, \bar{z}_1 \rangle| & |\langle \bar{z}_k, \bar{z}_2 \rangle| & \cdots & |\langle \bar{z}_k, \bar{z}_k \rangle|
\end{bmatrix}
= \begin{bmatrix}
1 & |\langle \bar{z}_1, \bar{z}_2 \rangle| & \cdots & |\langle \bar{z}_1, \bar{z}_k \rangle| \\
|\langle \bar{z}_2, \bar{z}_1 \rangle| & 1 & \cdots & |\langle \bar{z}_2, \bar{z}_k \rangle| \\
\vdots & \vdots & \ddots & \vdots \\
|\langle \bar{z}_k, \bar{z}_1 \rangle| & |\langle \bar{z}_k, \bar{z}_2 \rangle| & \cdots & 1
\end{bmatrix},
$$

where $\bar{z}_i$ is row $i$ of $Z$ and $\bar{z}_i$ denotes $z_i/\|z_i\|$. Note that entry $G_{ij}$ measures the angle between vectors $z_i$ and $z_j$, so the extent to which $G_{ij} \in [0, 1]$ is near 1 is the extent to which the vectors are nearly parallel. The premise is that two nearly parallel vectors implies a strong connection between the corresponding nodes in the sense that algebraically smooth vectors have values at these nodes that are correlated. This is the basis for the affinity approach. However, the aim now is to go beyond this concept to identify a set of correlated nodes that become an aggregate. Set correlation is necessary for cases like linear elasticity, where two nodes are not always significantly correlated by themselves. The methodology described here goes even further in that it produces the aggregate basis vectors so that an adaptive process is not strictly necessary, although it could be used later to possibly improve interpolation.

To determine suitable aggregates, the idea is to process $G$ much the same way AMG determines the C nodes and the F nodes that they interpolate to. But this process is simplified because the only task is to find a correlated set of nodes, and identifying correlation in $G$ is fairly straightforward. More precisely, if $G_{ij}$ is above a certain threshold (e.g., $G_{ij} \geq \theta$), then nodes $i$ and $j$ are deemed to be strongly correlated. Identifying a good aggregate might then begin by selecting a row in $G$ with the most strong connections, then adding the rows to which it is strongly connected. The associated nodes would then form the aggregate, although more could be added, if deemed advantageous, by including twice-removed nodes, i.e., those that are strongly connected to the current set.
For a simple example, suppose six nodes yield the normalized Gram matrix

\[
G = \begin{bmatrix}
1 & 0 & \frac{1}{\sqrt{2}} & \epsilon & \epsilon & \epsilon \\
0 & 1 & \frac{1}{\sqrt{2}} & \epsilon & \epsilon & \epsilon \\
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 1 & \epsilon & \epsilon & \epsilon \\
\epsilon & \epsilon & \epsilon & 1 & 0 & \frac{1}{\sqrt{2}} \\
\epsilon & \epsilon & \epsilon & 0 & 1 & \frac{1}{\sqrt{2}} \\
\epsilon & \epsilon & \epsilon & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 1
\end{bmatrix}
\]

This leads to two aggregates with respective nodes 1, 2, 3 and 4, 5, 6.

The above process determines at least one aggregate, \(\mathcal{A}\), possibly pruned from the original aggregate so that its size \(k = |\mathcal{A}|\) may be smaller than the original \(k\). (A tentative aggregate of size 6 becomes two of size 3 each in the above example.) To determine a tentative interpolation basis for \(\mathcal{A}\), a Gram-Schmidt-type process is now applied to the normalized columns of the new (possibly pruned) associated window \(\mathcal{Y}\). The goal of this process is to find a basis that approximately spans the error left on this aggregate after relaxation.

Considering the column vectors that form \(\mathcal{Y}\), the first element is selected to be the pivot and all the columns are normalized, which amounts to post-multiplication by a positive diagonal matrix \(\mathcal{D}\). At this point, a Gram-Schmidt-type process is applied to \(\mathcal{YD}\) that orthogonalizes (but does not normalize!) subsequent columns from the first. This yields

\[
\mathcal{YD} = QR.
\]

Note that \(Q\) retains the first column of \(\mathcal{YD}\), but subsequent columns are orthogonal to it. To the extent that these other columns in \(\mathcal{YD}\) are parallel to, and therefore dependent on, the first, is the extent to which they are of reduced magnitude in \(R\). If the subsequent columns in \(R\) are “small”, then just this first column is needed to represent interpolation. However, if there are still many “large” columns left, the next largest is moved to the first column and the process is repeated in this fashion until the columns are all small. This process is, in practice, accomplished using a rank-revealing QR algorithm.
The process of determining one aggregate and its basis is described here as a sequential one, identifying one aggregate/basis pair at a time. However, it might make sense to form all of the aggregates first to ensure that all nodes are accumulated, and then determine the basis for each, which could be performed separately. Some sample results are presented in Figures 4.1, 4.2, 4.3, 4.4 for applying this approach to a series of scalar problems. For scalar problems the method generates aggregates that have desired properties. Using the method within a multi-level algorithm yields the same convergence rates as the standard methods for using the relaxation-generated aggregates and the standard kernel. Initial tests using the kernel based on fitting the gram matrices locally also exhibits convergence rates on the same order as standard SA techniques.

4.4 Adaptively Improving Kernels

Once aggregates have been formed, the question then remains as to how to form the correct kernel for the solver. Standard adaptive smoothed aggregation methods are restricted by two basic requirements; that the sizes of blocks are constant over the entire hierarchy, and that the size of the kernel is constant across the entire domain. One might imagine a situation where there are two regions of the mesh, with elasticity dominating in one region and a simple Poisson equation over the other. There is no reason why the whole domain needs to have enough kernels to resolve the more complicated region of the domain.

Instead, one can imagine an approach that is based on the idea that the goal is to satisfy a weak approximation property (WAP) locally. Unlike the strong approximation property (SAP), the WAP only confirms good two-grid rates. But as in standard SA methods, the hope is that the basis function smoothing done in SA leads to effective V-Cycle rates. One of the tenets of this approach is that for each aggregate the number of local kernels needed to satisfy the WAP might be different.

Understanding that the process can only be initialized with global kernels, it is important to start by isolating the nearest kernel components out of the targets by applying a Ritz projection to the columns of $K_1$. Ritz projection is also applied to coarser-level targets as they are developed.
Figure 4.1: Aggregates for standard Poisson

Figure 4.2: Small aggregates for non-grid-aligned anisotropy
Figure 4.3: Larger aggregates for non-grid-aligned anisotropy

Figure 4.4: Aggregates for a curved anisotropy
This allows the method to highlight and properly scale the components in the span of the target matrices that are most important in coarsening. On a level the method does not assume that the same restrictions are needed from the previous level, but instead it approaches the new linear system as a new one that needs to be solved as efficiently as possible.

Once the kernel is selected now the coarse grid and the operators required for that grid must be formed. Assuming that $A_1, K_1$ have been provided, the first step to forming the coarse operator and transfer operators is to perform a Ritz projection on $K_1$. To do this, it is necessary to solve the generalized eigenproblem

$$K_1^T A_1 K q_k = \lambda_k K_1^T K_1 q_k \quad \text{for } k = 1, \ldots, \kappa_1,$$

subject to the restrictions

$$(K_1 q_i)^T (K_1 q_j) = \frac{1}{\lambda_i} \delta_{ij},$$

and

$$||K_1 q_k||_A = 1,$$

where $\delta_{ij}$ is the Kronecker delta function. In essence, the $q_i$ are ordered such that $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{\kappa_1}$. This ensures that most important targets are listed first and trims away targets that are not strictly necessary. (Targets with relatively large Rayleigh quotients are algebraically oscillatory and need not be considered in coarsening.) The targets (without trimming) now become $\hat{K}_1 = K_1 \begin{bmatrix} q_1 & \cdots & q_{\kappa_1} \end{bmatrix}$, but, for brevity throughout the rest of this note, the hat is omitted and is simply written $K_1$.

Assume that a set of aggregates has been selected for this level

$$A_1 = \{A_1, A_2, \ldots, A_{n_2}\}.$$
approach outlined earlier. Focusing on aggregate $A_i$, assume that it has $m_i$ degrees of freedom (DOFs) from the matrix $A_1$. Next, form the local target matrix

$$K_{A_i} = \begin{bmatrix} k^i_1 & k^i_2 & \cdots & k^i_{\kappa_1} \end{bmatrix},$$

and perform an eigendecomposition on $(K_{A_i})^T K_{A_i}$

$$(K_{A_i})^T K_{A_i} = \begin{bmatrix} q_1 & \cdots & q_{\kappa_1} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & \lambda_{\kappa_1} \end{bmatrix}^T q^T_k = \lambda_j \delta_{jk}, \quad (4.6)$$

where $q_j \in \mathbb{R}^{\kappa_1}$. Assume that the eigenvalues are ordered $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{\kappa_1}$. With a given threshold $\theta$, for the $\alpha_i$ eigenvalues that are smaller than $\theta$, choose the corresponding $q_j$, $j = 1, 2, \ldots, \alpha_i$, to define $p_j = K_{A_i} q_j$ and then form the matrix

$$\hat{P}_{A_i} = \Lambda^{-\frac{1}{2}}_{A_i} \begin{bmatrix} p_1 & \cdots & p_{\alpha_i} \end{bmatrix}$$

to enforce that $(\hat{P}_{A_i})^T \hat{P}_{A_i} = I$.

Once this chore has been accomplished for each aggregate, form

$$\hat{P}_1 = \begin{bmatrix} \hat{P}_{A_1} & 0 & \cdots & 0 \\ 0 & \hat{P}_{A_2} & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \hat{P}_{A_{\kappa_2}} \end{bmatrix},$$

and finally smooth to obtain

$$P_1 = S \hat{P}_1.$$ 

Now the coarse operator is formed

$$A_2 = P_1^T A_1 P_1,$$

where the number of DOF for $A_2$ is $\sum_{i=1}^{\kappa_2} \alpha_i$. Note that this doesn’t preclude $\alpha_i$ from being 0, i.e., there is the possibility that there are aggregates that have no need to go to, or interpolate from, the coarser level.
One more task is left to be performed for the coarse operator: to take the targets on the finest level to the coarse grid, which is done by computing their best Euclidean approximation there. Since \( (\hat{P}_A)\hat{P}_A = I \), the new targets for the coarse problem can be defined in two possible ways

\[
K_2 = \hat{P}_1^T K_1, \quad (4.7)
\]
or

\[
K_2 = P_1^T K_1. \quad (4.8)
\]

On a side note, when thinking about coarsening this new level, it might make most sense in the standard approach to aggregate these “blocks” together. Even though they are of the same size, a SOC matrix can be formed using the measure

\[
\rho \left( A_{ii}^{-1/2} A_{ij} A_{jj}^{-1} \right)
\]
to determine if two blocks corresponding to nodes with different numbers of DOF are strongly connected, as long as the code supports different size blocks.

Thinking recursively, assume now that the process has formed a series of operators and target matrices, \( A_1, A_2, \ldots, A_L \) and \( K_1, K_2, \ldots, K_L \) (note that \( K_L \) is not strictly necessary). Starting with the operator \( A_{L-1} \), it is possible to then pick some random initial guess, \( x_{L-1} \), and perform several V-cycles on \( A_{L-1} x_{L-1} = 0 \).

If the convergence of the solver at this point is satisfactory, then proceed to level \( L - 2 \). Otherwise, append the remaining error to the current target matrix

\[
K_{L-1} = \begin{bmatrix}
k_1 & \cdots & k_{k_{L-1}} & \epsilon
\end{bmatrix}.
\]

Once again apply Ritz to emphasize the right targets, and then reform \( P_{L-1} \) as detailed above. Once this has formed the new \( A_L \), test again to see if another component is needed. This is repeated until the solver is good, or some maximum number of iterations are done, and then the process is repeated on level \( L - 2 \).
This process is repeated up to increasingly finer levels until there is a good V-Cycle for each level, at which point the solver can then be used. Notice that once a component is added to the grid at any level, then all coarser levels need to be revisited and modified.
AMG methods have become one of the preeminent options when choosing iterative solvers for discretizations of PDEs. Common graph-associated matrices also give rise to large sparse linear systems [16,27,107]. Data-mining decisions are made based on solutions to the linear systems generated, and at large scales iterative solvers must be utilized. Iterative linear solvers have importance as well in order to accelerate inner computations for eigensolvers and other non-linear optimization techniques which are also of considerable data-mining interest. Given that AMG methods provide solvers for PDE problems that can reach discretization error within cost proportional to the number of DOF. It would be natural to hope to use AMG as an iterative solver for graph-associated matrices.

In order to have confidence that AMG/SA methods are being applied optimally to linear systems arising from graphs, more needs to be understood about the characteristics of the linear systems. Due to the lack of locality present in the original graphs, the linear systems exhibit some properties that are problematic for AMG routines. One of the most dramatic problems for standard AMG algorithms is non-homogeneity of the matrix itself. Many of the graphs of interest generate non zero patterns that have widely different numbers of non zeros in some rows. These particularly dense rows lead to large fill-in for the coarse problem when treated with the standard Galerkin closures in AMG and SA, and can generate coarse problems with a larger number of non zeros than the original problem. This can lead to cycle and operator complexities that no longer exhibit optimal convergence.
AMG methods are constructed upon the assumption that a coarse problem can be easily formed that is used to approximate the lower part of the spectrum. Due to the large number of non-local connections an appropriate coarse grid is difficult to select solely based on connections in the matrix. Difficulty in picking a coarse grid with the correct characteristics often leads to a V-Cycle with very poor convergence characteristics.

Several approaches exist that use iterative linear and eigenproblem solvers to obtain information about importance of the vertices, estimates of distance, and to highlight sought-after community structures [27,48,64]. Work has already been done in creating AMG methods designed for solving graph Laplacians. The lean algebraic multigrid (lAMG) solver developed by Livne et al [63] makes strong improvements in using AMG for wide classes of diagonally dominant matrices. However there are cases where lAMG is non-optimal due to large cycle complexities. Analysis of eigenvectors will help modify AMG to accommodate wider classes of graph-associated matrices, as well as being optimal for complex topologies that the graphs were formulated upon.

Because of the strong connection to continuous PDEs on unstructured meshes they were designed to solve, AMG methods were developed with a close understanding of the eigenspaces and eigenvectors of the matrices. To improve AMG solvers for graph-associated problems, theory about the properties of these matrices needs to be developed. To this end, some preliminary definitions are introduced that narrow down the realm of graphs to a subset (that is still very wide) that are of specific interest for data-mining questions. These graphs exhibit strong degree dependent properties that are important in understanding the eigenspaces of some of the graph-associated matrices of interest.

Chapter 6 presents a step forward by establishing connections between the eigenspaces and properties of the graph. While most of the work in Chapter 6 is theoretical, some numerics are presented that establish how connections are made in practice based on understanding the theory. These connections would then provide part of a foundation for designing better AMG solvers for matrices associated with graphs of this type.

While the primary purpose of the investigation of the eigenspaces is to further develop AMG
solvers, the properties also serve use in some other related areas. In the setting of iterative solvers for the linear systems resulting from the discretization of PDEs, global convergence criteria typically are easily developed that are tied to some form of discretization error. However, when making decisions about community structure or relative decisions about the importance of connections, it is not as clear how well to “solve” the resultant linear systems. Knowledge of the eigenspaces can help guide decisions in the process of tuning solvers to get to data-mining decisions at optimal cost. It should also be noted that the eigenvectors themselves are important in making data-mining decisions, and theoretical understanding has value there as well.

5.1 Graphs

A graph $G(V, E)$ is used to represent a relational dataset, where $G$ is composed of $n := |V|$ vertices and $m := |E|$ edges. Typically the vertices represent an object of interest, such as a physical location or something more abstract like a word; and edges represent some manner of connections between these objects [78].

For a given graph $G$, then if $i, j \in V$ are two vertices in the graph, then the edge that connects is written $(i, j) \in E$. Notice this notation implies a direction in that the edge is rooted at $i$ and ends at $j$; graphs with edges that imply direction are denoted directed graphs. A graph is considered to be undirected if $(i, j) \in E \implies (j, i) \in E$. Undirected unweighted graphs are considerably easier to study using the tools of numerical linear algebra because of the symmetry properties of the matrices associated with them. Examples of directed and undirected graphs are shown in Figures 5.1, 5.2.

![Directed Graph with 6 Vertices](image)
A graph is considered to be **connected** if any pair of vertices in the graph are connected by a sequence of edges, where each successive edge is rooted at the tail of the previous edge. An edge of the graph that connects to itself is called a **self loop**; a graph with no self loops means that 

\[(i, i) \notin E.\]

The **degree** of a vertex \(i \in V\) is the number of edges that are incident to \(i\), denoted \(d_i\). The **graph neighborhood**, denoted \(N_i\), of a node \(i\) are all the vertices \(j\), such that \((i, j) \in E\). In set notation, this is 

\[N_i = \{ j \in V \mid (i, j) \in E \}.\]

Note if a graph has no self loops, then \(i \notin N_i\). An edge in the graph is also associated with a weight. An **unweighted graph** is a graph where all edges are considered to have an equal weight of 1. All graphs in this thesis will be considered to be unweighted, connected, and have no self loops.

## 5.2 Graph-Associated Matrices

The **unweighted adjacency matrix** (or more briefly **adjacency matrix**) \(A \in \mathbb{R}^{n \times n}\) is a binary matrix such that

\[
A_{ij} = \begin{cases} 
1 & \text{if } (i, j) \in E \\
0 & \text{otherwise.}
\end{cases}
\]
Notice since $G$ is undirected, $A$ is symmetric, and the matrix that is used to generate Figure 5.2 would then be

$$A = \begin{bmatrix}
0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0
\end{bmatrix}.$$

Another commonly used matrix for data-mining decisions is the Modularity matrix $B \in \mathbb{R}^{n\times n}$, which is a specific rank-one update of a matrix. Let $d$ be the vector such that $d_i = \sum_j A_{ij}$, often called the degree vector. Then

$$B = A - \frac{1}{2m}dd^t.$$

For the sake of completeness, graph Laplacians are also defined here. Ideally AMG solvers are going to be applied to the Laplacian matrices, not the Adjacency or Modularity matrices directly, but understanding of the eigenspaces of $A,B$ is a first step toward further understanding Laplacians. Recall that, for each vertex $i$, the degree is defined as the number of edges incident to the matrix. For this class of graphs, and assuming no self loops, then $d_i = \sum_{j=1}^n A_{ij}$, and the degree matrix $D$ is defined by

$$D_{ij} = \begin{cases} d_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}.$$

The combinatorial Laplacian can now be defined as $L = D - A$, and the normalized Laplacian is $\hat{L} = I - D^{-1/2}AD^{-1/2}$, of interest can also be the signless Laplacian $L_s = D + A$.

The matrix of interest is often dependent on the context in which a data-mining decision is to be made. One of the open areas of research is understanding which graph-associated information should be used in any setting. One property that all these matrices exhibit is that their sparsity is directly related to the sparsity of the graph. In most applications, while very high dimensional data sets are generated, they are usually very sparse in practice.
5.3 Scale-Free Graphs

One subset of graphs and networks that are of particular interest are scale-free graphs. Graphs of this type are assumed to have a highly skewed degree distribution: there are a relatively small number of vertices with a many connections and, conversely, many vertices with very few connections \[60\]. Traditionally, in the literature associated with scale-free graphs, one defining attribute is that in large networks the degree distribution follows a scale-free power law. Let \( \mathcal{V} \) be the set of all vertices in a graph, \( \mathcal{G} \). Picking any vertex \( v \in \mathcal{V} \), the probability that that degree is some integer \( k \) is written as

\[
P[\text{degree}(v) = k] \propto Ck^{-\alpha}, \quad \alpha > 1.
\]

Typically \( \alpha \in (2, 3) \), but there are graphs where it can fall outside this range. Graphs of this form are of interest because networks with power law distributions cause trouble for standard clustering approaches.

One clear way to get a sense of scale-free graphs is to look at some illustrative examples, and then to generate some simple spectral coordinate mappings to understand their relation. There are many ways of generating so called scale-free graphs. For later approaches the Barabasi-Albert model \[86\] was used, which starts with some initial set of vertices of small degree. New vertices are then added, and each added vertex is connected to some set of the original nodes with a probability proportional to the number of existing links. Formally the probability that a new vertex is connected to an existing vertex is

\[
p_i = \frac{k_i}{\sum_{j \in \mathcal{V}} k_j}.
\]

This leads to the behavior that highly connected nodes just end up gathering more and more connections. The model also has no self loops, and by construction only forms connected graphs. The distribution of the nodes is a power law that looks like \( k = 3 \). Three samples were used to generate these examples.

(1) \( |\mathcal{V}| = 20 \) Figure 5.3 shows a picture of a graph and the degree histogram for 20 vertices.
Figure 5.3: Graph and degree distribution for $|\mathcal{V}| = 20$

(2) $|\mathcal{V}| = 100$  Figure 5.4 shows a picture of a graph and the degree histogram for 100 vertices.

Figure 5.4: Graph and degree distribution for $|\mathcal{V}| = 100$

(3) $|\mathcal{V}| = 1000$  Figure 5.5 shows a picture of a graph and the degree histogram for 1,000 vertices.

Figure 5.5: Graph and degree distribution for $|\mathcal{V}| = 1,000$
In practice, scale-free graphs can show up in many settings. Some examples include social media networks, Internet connectivity graphs, or the power grid in the United States [1]. As technology has advanced, and storage has grown cheaper, larger and larger graphs of this type are generated. These graphs are often the preferred target of data-mining applications, so they are selected as the graphs for which the graph-associated matrices are studied in this thesis. Due to their large scale using AMG methods in this setting is the long-term goal.

5.4 Classical Eigenvalue and Perron-Frobenius Results

In Chapter 6, several new lemmas and theorems will be presented that rely on certain fundamental results from eigenvalue and Perron-Frobenius theory. Proofs and alternate statements of the following theorems are available in [54], but the theorems are presented here for reference, and have been modified to be consistent with definitions used in Chapter 6.

Theorem 5.4.1. (Gerschgorin) Let $A$ be a $n \times n$ matrix and $\sigma$ be the set of all eigenvalues of $A$. Then

$$\sigma \subset \bigcup_{i=1}^{n} \left\{ r \in \mathbb{C} : |a_{ii} - r| \leq \sum_{k=1, k \neq i}^{n} |a_{ik}| \right\}.$$ 

Theorem 5.4.2. (Courant-Fischer Min-max) Let $A$ be a $n \times n$ symmetric matrix, and let $k$ be an integer $1 \leq k \leq n$. Then

$$\lambda_k = \min_{w_1, w_2, \ldots, w_{n-k} \in \mathbb{R}^n} \max_{x \neq 0, x \in \mathbb{R}^n} \frac{x^T Ax}{x^T x},$$

and

$$\lambda_{k} = \max_{w_1, w_2, \ldots, w_{n-k} \in \mathbb{R}^n} \min_{x \neq 0, x \in \mathbb{R}^n} \frac{x^T Ax}{x^T x}.$$ 

Theorem 5.4.3. (Interlacing eigenvalues) Let $A$ be a $n \times n$ symmetric matrix, and let $z \in \mathbb{R}^n$ be a given vector. If the eigenvalues of $A$ and $A + zz^T$ are arranged in increasing order, then

(i) $\lambda_k(A + xx^T) \leq \lambda_{k+1}(A) \leq \lambda_{k+2}(A + zz^T)$ for $k = 1, 2, \ldots, n - 2$.

(ii) $\lambda_{k}(A) \leq \lambda_{k+1}(A + zz^T) \leq \lambda_{k+2}(A)$ for $k = 1, 2, \ldots, n - 2$. 

Definition 5.4.1. (Majorization) Let \( \mathbf{x}, \mathbf{y} \in \mathbb{R}^n \), be two vectors organized such that \( x_1 \leq x_2 \cdots \leq x_n \), and \( y_1 \leq y_2 \cdots \leq y_n \), then it is said \( \mathbf{x} \) is majorized by \( \mathbf{y} \), denoted \( \mathbf{x} \prec \mathbf{y} \), if

\[
\sum_{i=1}^{k} y_i \leq \sum_{i=1}^{k} x_i, \quad 1 \leq k < n,
\]

\[
\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} x_i.
\]

Theorem 5.4.4. (Schur) Let \( A \) be a \( n \times n \) hermetian matrix. Its vector of eigenvalues \( \lambda(A) \) majorizes its vector of main diagonal entries \( \mathbf{d} \), that is

\[
\sum_{i=1}^{k} d_i \leq \sum_{i=1}^{k} \lambda_i
\]

for each \( k = 1, 2, \ldots, n \), with equality for \( k = n \).

Theorem 5.4.5. Perron Let \( A \) be a \( n \times n \) positive matrix. Then

(i) \( \rho(A) \) is an eigenvalue of \( A \).

(ii) There is a positive eigenvector corresponding to \( \rho(A) \).

(iii) \( |\lambda| < \rho(A) \) for every eigenvalue such that \( \lambda \neq \rho(A) \).

(iv) \( \rho(A) \) is a simple eigenvalue of \( A \).
Chapter  6

Properties of Eigenpairs of Adjacency and Modularity Matrices

Data-mining applications use the eigenpairs of both the Adjacency and Modularity matrices to make decisions. Some examples of this involve spectral clustering [77, 83] and Katz scores [16]. However, when using these techniques on matrices with troublesome characteristics, such as small spectral gaps, preconditioners are needed for the eigensolves. Robust linear solvers based on both the adjacency matrix and modularity matrix are of research interest. While knowledge of the eigenspaces is necessary for designing AMG preconditioners, it also provides insight into the data-mining decisions.

When studying the adjacency and modularity matrices formed from scale-free graphs, several degree-based results are established. Initially it is established that there is a direct correlation between the largest eigenvalues and the largest degree vertices. With this information in hand, meaningful information can then be derived about the relative and absolute values of eigenvectors associated with these large eigenvalues at nodes in the graphs. Specifically, the largest absolute values of eigenvectors must be at large-degree nodes. Conversely, the values at very low-degree nodes are shown to very small.

With this understanding about the eigenspaces in hand, decisions can then be made for iterative solvers about which nodes are important to address. Due to the large spectral gaps that may exist for scale-free graphs, it is possible to show that a preconditioned conjugate gradient (pCG) solver can work out of the box on many scale-free graphs. However, on more complicated topologies, or if the problems are posed on any mesh-like structure, more robust iterative solvers
are needed.

In the work here, useful properties tying eigenspaces to degree-dependent structure are established for two graph-associated matrices of interest. The results here need to be extended and modified to fit the graph Laplacians in order to create AMG solvers for the full suite of matrices of interest.

6.1 Spectral Coordinates

While the goal is to use the theory to build iterative preconditioners, the degree-dependent results also have meaningful implications in spectral embedding. A short introduction to spectral embedding that provides a simple context for the numerics based on the eigenspaces is provided later in this chapter.

When embedding from $\mathcal{V}$ to a finite dimensional space using eigenvectors, the coordinates for a vertex associated with this embedding are called the spectral coordinates. This embedding is used with the hope that it preserves, in some way, the underlying structure of the graph. To this end recall that an eigenvalue, $\lambda_i$, of the adjacency matrix, $A$, satisfies the equation

$$Aw_i = \lambda_i w_i,$$

subject to the constraint $w_j \perp w_j$ for $i \neq j$, where $w_i \in \mathbb{R}^n$. Since $A$ is a symmetric $n \times n$ matrix, then all of its eigenvalues are real and are written in decreasing order: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. Typically the $K$ extremal eigenvalues are utilized, which is the set that consists of the $K_1$ largest and $K_2$ smallest eigenvalues, with $K = K_1 + K_2$. This set of extremal eigenvalues is formed as the set of $K$ elements $\{\lambda_1, \ldots, \lambda_{K_1}, \lambda_{n-K_2+1}, \ldots, \lambda_n\}$. The spectrum of $A$ is defined to be $\sigma(A) = \{\lambda_k \mid k = 1, \ldots, n\}$, and the spectral radius is defined to be $\rho(A) = \max_k |\lambda_k|$.

Defining the $n \times K$ matrix $W = \begin{bmatrix} w_1 & w_2 & \cdots & w_K \end{bmatrix}$, then $AW = \Lambda W$, where $\Lambda$ is a diagonal matrix with the $i^{th}$ diagonal element being the eigenvalue associated with $w_i$. For any vertex in $\mathcal{V}$, the row of $W$ associated with that vertex defines a unique $K$-tuple associated with that point. This is the spectral coordinate defined for mapping the graph into $\mathbb{R}^K$ based on the selected eigenspace.
So the spectral coordinate is just a row of $W$, e.g., for the $i^{th}$ node the spectral coordinate is obtained via $W^T e_i$, where $e_i$ is the basis vector that is 0 except in the $i^{th}$ position, where it is 1.

It has been observed that based on this embedding certain “eigenspokes” [83] are formed, and claims are made that these spokes reveal underlying community structure in the graphs. Figure 6.1 shows a three-dimensional spectral embedding of generated scale-free graphs of increasing size to provide a reference for this phenomenon. In these simple examples, as well as in real-world graphs, the tips of eigenspokes are almost always at the largest-degree nodes. The eigenspokes, and also that the majority of the nodes are concentrated around the origin, are explained by the theory presented in this thesis.

![Figure 6.1: Spectral mapping for the three largest eigenvectors of $A$, for $|\mathcal{V}| = 20, 100, 1000$](image)

### 6.2 Adjacency Matrix and Modularity Matrix

Consider the adjacency matrix $A \in \mathbb{R}^{n \times n}$ formed from an undirected connected scale-free graph $\mathcal{G}$ with no self loops. Much can be said about the spectral radius and the largest eigenvalue of $A$ due to Perron-Frobenius theory, and study of the spectral radius of $A$ in general has been a focus of research, largely summarized in [30]. The goal here is to focus on establishing results for
A when generated from scale-free graphs.

Often these matrices are used to make data-mining decisions, so the question becomes how much relevant information can be teased out of the eigenvectors of these matrices. Work in this thesis is similar to some of the results in [28, 69]. Here specific attention is tied to associating eigenvalues and the values of eigenvectors at particular vertices with factors dependent on the degrees of the vertices. Because of the power-law distribution of scale-free graphs this can have dramatic effects.

Several properties of adjacency matrices will become useful shortly. The first is that $diag(A^2) = diag(D)$. This is easy to see because

$$D_{ii} = \sum_{j=1}^{n} a_{ij}a_{ji} = \sum_{j=1}^{n} a_{ij}^{2} = \sum_{j \in N_i} 1 = d_i.$$ 

Another useful property is that $diag(A^3) = 2diag(T)$ where $T$ is the diagonal matrix such that $t_i$ is the number of triangles connected to vertex $i$. To see this, notice that

$$T_{ii} = \sum_{l=1}^{n} \sum_{j=1}^{n} a_{ij}a_{jl}a_{ji},$$ 

but $a_{ij}a_{jl}a_{ji} \neq 0$ if and only if there exists a path $i \rightarrow j \rightarrow l \rightarrow i$, i.e., a triangle rooted at vertex $i$. Since the graph is assumed to be undirected the path is represented both ways in $A$, so every triangle is represented by two paths. The summation is equivalent to counting all the paths of length three connected to a vertex, and since there are no self loops, this must count all triangles twice.

By Gershgorin’s Theorem [54] it is easy to establish that $\sigma(A) \in [-d_{max}, d_{max}]$. The singular values of a matrix $M$ are the positive square roots of the eigenvalues of $M^*M$, i.e., $\mu_i = \sqrt{\lambda_i} \mid \lambda_i \in \rho(M^*M)$. In the case where $A$ is symmetric then it can be written $A = QAQ^*$ and then $A^*A = QA^2Q^*$, so the eigenvalues of $A^*A$ are defined by $\rho(A^*A) = \{ \lambda^2 \mid \lambda \in \rho(S) \}$. $A$ then has singular values $\mu_i = |\lambda_i|$. Recall that the trace of a matrix satisfies the relation that

$$Tr(A^k) = \sum_{i=1}^{n} \lambda_i^k.$$
It is possible to use majorization (as defined in Definition 5.1.6) and Schur’s Inequality [54,67] to then show that

\[(a) \sum_{k=1}^{K} d_k \leq \sum_{k=1}^{K} \mu_k^2 \quad \text{and} \quad (b) \sum_{k=1}^{K} 2t_{p_k} \leq \sum_{k=1}^{K} \lambda_k^3, \tag{6.1}\]

where \(p_k\) has been selected so that \(t_{p_k} > t_{p_{k+1}}, \forall k\) in the same manner as the vertices are ordered to make the diagonals of the degree matrix increasing. By Gershgorin’s Theorem [54] \(\lambda_1 \in \left[-d_{\max}, d_{\max}\right]\), and combining this with 6.1(a) results in \(\lambda_1 \in \left[\sqrt{d_{\max}}\right]\). However, it is later shown that this bound is greatly improved for the cases where the neighborhood of a high-degree vertex is highly connected. For convenience define the induced subgraph of a vertex to be \(G_i(V_i, E_i)\), the set of vertices and edges in the neighborhood of \(i\). Note this includes the vertex \(i\), so specifically \(V_i = \{i\} \cup N_i\), and \(E_i = \{(j, k) \in E \mid j, k \in V_i\}\).

Lemma 6.2.1 establishes lower bounds of the largest eigenvalue of \(A\) in terms of the degree-based structure and the connectivity of the graph around the vertices. This improved lower bound is established as a function of the connectivity of the largest degree node.

**Lemma 6.2.1.**

\[\rho(A) \geq \max_{i \in V} f(d_i, t_i),\]

where for each \(d_i \geq 1\), \(f(d_i, t)\) is a monotonically increasing function in \(t \in (0, \frac{d_i}{2} (d_i - 1))\) such that

\[f(d_i, 0) = \sqrt{d_i} \quad \text{and} \quad f\left(d_i, \frac{d_i(d_i - 1)}{2}\right) = d_i.\]

**Proof.** Define \(t_i = |E_i| - d_i\) to be the number of triangles that vertex \(i\) participates in. Note \(t_i \in [0, \frac{d_i}{2} (d_i - 1)]\). To see this is true notice that the maximum occurs when every edge in the neighborhood of \(i\) is connected. Every pair of nodes, \(j, k\), in a fully connected neighborhood, \(N_i\), forms a triangle. Thus the maximum number of triangles is defined by how many pairs can be selected from the neighborhood

\[t_{\max} = \binom{d_i}{2} = \frac{d_i}{2} (d_i - 1).\]
Pick \( i \in \mathcal{V} \). Then for this vertex define
\[
x = \begin{cases} 
x_j = 1 & j = i \\
x_j = a & j \in \mathcal{N}_i \\
x_j = 0 & \text{otherwise ,}
\end{cases}
\]
and note that since \( \rho(A) \geq \max_{z \in S} \langle Az, z \rangle / \langle z, z \rangle \). Then for any \( a \in \mathbb{R} \), \( \rho(A) \geq \langle Ax, x \rangle / \langle x, x \rangle \). Take any subgraph and look at the adjacency matrix for it. Since \( x \) is zero outside the neighborhood, the effect on any nodes outside this region can be ignored and instead restrict attention to the matrix \( A \) of size \( |\mathcal{N}_i| \times |\mathcal{N}_i| \). Suppose \( A \) has \( t_i \) triangles then it can be split such that
\[
A = \begin{bmatrix}
0 & 1 & 1 & \cdots & 1 \\
1 & 0 & 0 & \cdots & 0 \\
\vdots & 0 & 0 & \cdots & 0 \\
\vdots & 0 & 0 & \cdots & 0 \\
1 & 0 & 0 & \cdots & 0
\end{bmatrix} + \sum_{p=1}^{t_i} T_p
\]
\[
= B + \sum_{p=1}^{t_i} T_p,
\]
where \( T_p \) is the matrix representing the edge not involving \( i \) that makes up triangle \( p \), one of the \( t_i \) triangles in the sub-graph. Notice that \( T_p \) has exactly two non-zeros in it and they are in distinct rows and columns that are not in row \( i \). Now
\[
\langle Ax, x \rangle = \langle (B + \sum_{p=1}^{t_i} T_p)x, x \rangle = \langle Bx, x \rangle + \sum_{p=1}^{t_i} \langle T_p x, x \rangle,
\]
and note
\[
\langle Bx, x \rangle = 2d_ia,
\]
and
\[
\sum_{p=1}^{t_i} \langle T_p x, x \rangle = \sum_{p=1}^{t_i} 2a^2 = 2t_ia^2.
\]
So, for this specific \( x \)

\[
\langle A x, x \rangle = 2d_i a + 2t_i a^2,
\]

and

\[
\langle x, x \rangle = 1 + d_i a^2.
\]

This means a lower bound can be formed for \( \rho(A) \) of the form

\[
\rho(A) \geq \max_{a \in \mathbb{R}} \frac{2d_i a + 2t_i a^2}{1 + d_i a^2}.
\]

Defining

\[
f(d_i, t_i) = \max_{a \in \mathbb{R}} \frac{2d_i a + 2t_i a^2}{1 + d_i a^2},
\]

it is now possible to look at the continuous version of this function

\[
f(a, d, t) = \frac{2a^2 t + 2ad}{a^2 d + 1},
\]

and taking a derivative with respect to \( a \) yields

\[
\frac{\partial}{\partial a} f(a, d, t) = \frac{4at + 2d}{a^2 d + 1} - \frac{2ad(2a^2 t + 2ad)}{(a^2 d + 1)^2}
\]

\[
= \frac{4at - 2a^2 d^2 + 2d}{(a^2 d + 1)^2},
\]

which has the two roots

\[
\begin{cases}
  t - \sqrt{d^3 + t^2} \\
  t + \sqrt{d^3 + t^2}
\end{cases}
\]

Selecting the root that yields the largest maximum

\[
a_* = \frac{t + \sqrt{d^3 + t^2}}{d^2},
\]

so the maximized function takes the form

\[
f(d, t) = \frac{2t \left( \frac{\sqrt{d^3 + t^2} + t}{d^2} \right)^2 + \frac{2d(\sqrt{d^3 + t^2} + t)}{d^2}}{d \left( \frac{\sqrt{d^3 + t^2} + t}{d^2} \right)^2 + 1}.
\]
After some algebra this reduces to

\[ f(d,t) = \frac{t + \sqrt{d^3 + t^2}}{d}. \]

Notice it satisfies \( f(d_i, 0) = \sqrt{d_i} \) and \( f \left( d_i, \frac{d_i(d_i-1)}{2} \right) = d_i \), and is clearly monotonically increasing for \( t \in \left(0, \frac{d_i(d_i-1)}{2}\right) \).

\[ \square \]

With Lemma 6.2.1 in hand it is possible to now establish the next result regarding eigenvalues in terms of degree-based bounds.

**Theorem 6.2.1. (Most Positive Eigenvalues of \( A \))** Let \( I_k = \{i_1, i_2, \ldots, i_k\} \) be any set of \( k \) vertices that are three hops or greater from each other (an independent set in the two-step graph). Then

\[ \lambda_k \geq \max_{I_k} \left[ \min_{i \in I_k} f(d_i, t_i) \right], \]

where \( f \) is as established in Lemma 6.2.1.

**Proof.** Define \( x_i \) as in the proof of Lemma 6.2.1. Let \( S_k \) be the subspace spanned by \( x_i \) for each \( i \in I_k \). Note that for \( i, j \in I_k \)

\[ x_i^T x_j = \begin{cases} 0 & i \neq j \\ ||x_i||^2 & i = j. \end{cases} \]

This is due to the assumption that all nodes are at least three steps apart, which implies \( N_i \cap N_j = \emptyset \), so there are no non zero elements in common in \( x_i, x_j \). Define

\[ P_k = \begin{bmatrix} x_{i_1} & x_{i_2} & \cdots & x_{i_k} \\ ||x_{i_1}||^2 & ||x_{i_2}||^2 & \cdots & ||x_{i_k}||^2 \end{bmatrix} 
= [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k], \]

the matrix whose columns are the normalized versions of \( x_i, \hat{x}_i = \frac{x_i}{||x_i||^2} \). For convenience denote
\[ \mathcal{P} = \mathcal{P}_{I_k}. \] Notice that

\[
\mathcal{P}^T \mathcal{P} = [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k]^T [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k] \\
= \begin{bmatrix}
\langle \hat{x}_1, \hat{x}_1 \rangle & \ldots & \langle \hat{x}_1, \hat{x}_k \rangle \\
\vdots & \ddots & \vdots \\
\langle \hat{x}_k, \hat{x}_1 \rangle & \ldots & \langle \hat{x}_k, \hat{x}_k \rangle 
\end{bmatrix}
\]

which by the orthogonality of \( \langle x_i, x_j \rangle \) means \( \mathcal{P}^T \mathcal{P} = I \). Also notice that

\[
\mathcal{P}^T A \mathcal{P} = [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k]^T A [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k] \\
= [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k]^T [A\hat{x}_1, A\hat{x}_2, \ldots, A\hat{x}_k] \\
= \begin{bmatrix}
\langle A\hat{x}_1, \hat{x}_1 \rangle & \ldots & \langle A\hat{x}_1, \hat{x}_k \rangle \\
\vdots & \ddots & \vdots \\
\langle A\hat{x}_k, \hat{x}_1 \rangle & \ldots & \langle A\hat{x}_k, \hat{x}_k \rangle 
\end{bmatrix}
\]

However, suppose \( i \neq j \), if \( \langle A\hat{x}_i, \hat{x}_j \rangle \neq 0 \) there exists at least one index \( k \) such that \( (Ax_i)_k (x_j)_k \neq 0 \).

This means that there is some index \( k \in N_l \cap N_j \), where \( l \in N_i \). However, this is only true if there is a connection of length 2 between \( i, j \), which violates the initial assumption, so \( \langle A\hat{x}_i, \hat{x}_j \rangle = 0 \), if \( i \neq j \). Finally by construction

\[
\langle A\hat{x}_i, \hat{x}_i \rangle = \frac{1}{||x_i||^2} \langle Ax_i, x_i \rangle = \frac{\langle Ax_i, x_i \rangle}{\langle x_i, x_i \rangle} = f(d_i, t_i),
\]

so,

\[
\mathcal{P}^T A \mathcal{P} = \text{Diag}(f(t_i, d_i)).
\]
Now by applying the Courant-Fisher Min-Max Theorem

\[
\lambda_k = \max_{\dim(S) = k} \min_{x \in S} \frac{x^T Ax}{x^T x} \\
\geq \max \min_{S_k} \frac{x^T Ax}{x^T x} \\
= \max \min_{I_k} \frac{y^T P^T A P y}{y^T P^T P y} \\
= \max \left[ \min_{i \in I_k} f(d_i, t_i) \right],
\]

where the first step is just because this is a subset over all spaces of that dimension, the second step is by definition of the space, and the final one is that the minimum of

\[
\frac{y^T \text{Diag}(f_i, t_i) y}{y^T y} = \min_i f(d_i, t_i).
\]

The result that should be taken from Theorem 6.2.1 is that selecting \( K \) vertices that are separated by at least three hops allows a degree-based estimate on the \( k^{th} \) eigenvalue of \( A \). More specifically, if \( K \) large-degree nodes can be separated in this fashion then \( \lambda_k \geq \sqrt{d_k} \) for \( k = 1, \ldots, K \), and since the minimum defined by a node is increased if that node has higher connectivity, the lower bounds on these eigenvalues is usually higher in real world graphs.

Low-rank corrections to \( A \) are commonly used in cases of highly-skewed degree distribution. Here, focus is paid to a specific rank-one correction common in the literature: the modularity matrix, \( B = A - \frac{1}{m} d d^T \), where \( d \) is the vector with degree \( d_i \) as its \( i^{th} \) entry. The eigenpairs satisfying \( Bv_k = \theta_k v_k \) for \( K \) eigenpairs \((\theta_k, v_k)\) with eigenvalues in decreasing ordering are analyzed.

Recall the Cauchy Interlacing Theorem [54] shows the eigenvalues of a symmetric matrix interlace the eigenvalues of a low-rank update to that matrix. Here a useful corollary to the interlacing theorem for negative, rank-one corrections is presented.

**Corollary 6.2.1. (Negative, Rank-One Interlacing)** Let \( n \geq 2 \), \( A \in \mathbb{R}^{n \times n} \) be symmetric, and \( z \in \mathbb{R}^n \). Let \( \lambda_k(A) \) and \( \lambda_k(A - zz^T) \) be the \( k^{th} \) right-most eigenvalue of \( A \) and \( A - zz^T \),
respectively. For \( k = 1, \ldots, (n - 1) \)

\[
\lambda_{k+1}(A) \leq \lambda_k(A - zz^t) \leq \lambda_k(A), \quad \text{and}
\lambda_n(A - zz^t) \leq \lambda_n(A).
\]

This result implies the eigenvalues of \( B \) are tied closely to the eigenvalues of \( A \), and, for scale-free graphs, the extremal eigenvalues of \( B \) are also distributed according to the structure surrounding the high-degree vertices. In particular, applying Theorem 6.2.1 and Corollary 6.2.1 with \( z = (2m)^{-1/2}d \) shows that

\[
\lambda_1(B) \geq \sqrt{d_p},
\]

where \( p \) is the largest-degree vertex that is three or more hops from a vertex of maximal degree. (Note: if there are two vertices of maximal degree that are three or more hops from each other, then \( \lambda_1(B) \geq \sqrt{d_{\max}} \).) More connectivity within \( N_p \) implies a bound larger than \( \sqrt{d_p} \).

In the next section the theory that has been established is then used to demonstrate that the size of the extremal eigenvalues implies degree-related maximum principles and decay rates for the associated eigenvectors for \( A \) and \( B \).

### 6.3 Degree-Based Decay of Eigenvalues and Eigenvectors

The theorems presented in this section outline that for adjacency and modularity eigenpairs used for spectral embedding highlight the high-degree nodes and proximity of vertices to these nodes. The theory is developed using local analysis: that is, using the \( i^{th} \) row of the eigenequation to establish information about the magnitude of the \( i^{th} \) entry in the eigenvector based on the average values of the eigenvector in the neighborhood of a vertex and the associated eigenvalue.

**Theorem 6.3.1. (Local Analysis for Adjacency Eigenpairs)** Let \( W \) and \( \Lambda \) represent \( K \) eigenpairs of \( A \) with no zero eigenvalues. Then

\[
W^t e_i = d_i \Lambda^{-1} \left( \frac{1}{d_i} \sum_{j \in N_i} W^t e_j \right).
\]  \( (6.2) \)
Proof. Since by assumption $W$ and $\Lambda$ represent the first $K$ eigenpairs of $A$ then

$$AW = WA.$$  

Since $A, \Lambda$ are symmetric

$$\Lambda W^T = W^T A.$$  

Now applying both sides to any $e_i$

$$\Lambda W^T e_i = W^T A e_i$$

$$= W^T \left( \sum_{j \in N_i} e_j \right)$$

$$= d_i \frac{1}{d_i} \sum_{j \in N_i} W^t e_j.$$  

Since there are assumed to be no zero eigenvalues in $\Lambda$, then it is non-singular and can be inverted to obtain

$$W^t e_i = d_i \Lambda^{-1} \left( \frac{1}{d_i} \sum_{j \in N_i} W^t e_j \right).$$

What this theorem says then is that for any node its spectral coordinate can be written as an average of all its neighbors, with a factor that is dependent on the degree of the node. Notice that if the node has degree much smaller than the highest degree nodes that its value must be driven to the origin. However, for very high degree nodes the average value may even be magnified. In order to present the major result about decay rates in the eigenvectors of the adjacency matrix, a distinction must be made between a global and local maximum.

**Definition 6.3.1.** (Local and Global Extrema) For graph $G(V, E)$ and vector $g \in \mathbb{R}^n$ defined on $V$, $g$ is said to have a local maximum at vertex $i$ if $g_i \geq g_j$ for all $j \in N_i$. $g$ is said to have a global maximum at vertex $i$ if $g_i \geq g_p$ for all $p \in V$. Note that a global maximum is also a local maximum.

**Theorem 6.3.2.** Let $(\lambda_k, w_k)$ be eigenpairs of $A$ with $\lambda_k \neq 0$. Then the following results hold.
(i) **Local Maximum Principle** If \( w_k \) has a local maximum at vertex \( i \), then \( d_i \geq |\lambda_k| \).

Moreover, if \( k = 1 \), then \( d_i \geq \sqrt{d_{\text{max}}} \).

(ii) **Low-Degree Periphery Decay** Let \( S^k_\xi \) be the set of all vertices \( q \) such that \( d_q \geq \xi |\lambda_k| \) for \( \xi \in (0,1) \). Assume \( j \not\in S^k_\xi \) is at least \( s \) hops from \( S^k_\xi \). Then

\[
|w_j^{(k)}| \leq \xi^s \max_{p \in V} |w_p^{(k)}|.
\]

**Proof.** The proof is presented in two parts.

(i) For the first part: Recall from Theorem 6.3.1 that

\[
W^T e_i = d_i \Lambda^{-1} \left( \frac{1}{d_i} \sum_{j \in N_i} W^T e_j \right).
\]  

\[
W^T = \begin{bmatrix} w^{(1)} \\ w^{(2)} \\ \vdots \\ w^{(K)} \end{bmatrix} \implies W^T e_i = \begin{bmatrix} w^{(1)} \\ w^{(2)} \\ \vdots \\ w^{(K)} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} w_i^{(1)} \\ w_i^{(2)} \\ \vdots \\ w_i^{(K)} \end{bmatrix}.
\]

Combining this with (6.3)

\[
\begin{bmatrix} w_i^{(1)} \\ w_i^{(2)} \\ \vdots \\ w_i^{(K)} \end{bmatrix} = d_i \Lambda^{-1} \left( \frac{1}{d_i} \sum_{j \in N_i} w_j^{(1)} \right).
\]

Taking the \( k^{th} \) coordinate results in the expression

\[
w_i^{(k)} = \frac{d_i}{\lambda_k} \sum_{j \in N_i} w_j^{(k)},
\]
and after taking absolute values

\[ |w_i^{(k)}| = \left| \frac{d_i}{\lambda_k} \frac{1}{d_i} \sum_{j \in N_i} w_j^{(k)} \right| \]

\[ = \frac{d_i}{|\lambda_k|} \left| \frac{1}{d_i} \sum_{j \in N_i} w_j^{(k)} \right| \]

\[ \leq \frac{d_i}{|\lambda_k|} \frac{1}{d_i} \sum_{j \in N_i} \left| w_j^{(k)} \right| \]

\[ \leq \frac{d_i}{|\lambda_k|} \max_{j \in N_i} \left| w_j^{(k)} \right|. \]

Now assuming \( |w_i^{(k)}| \geq |w_j^{(k)}| \) for any \( j \in N_i \)

\[ |\lambda_k| \leq \frac{\max_{j \in N_i} |w_j^{(k)}|}{|w_i^{(k)}|} \implies |\lambda_k| \leq d_i. \]

For the case \( k = 1 \) recall the relation

\[ \sum_{k=1}^K d_k \leq \sum_{k=1}^K \mu_k^q \implies \sqrt{d_{\text{max}}} \leq |\lambda_1|. \]

(ii) For the second part recall by definition that

\[ S_{\xi}^k = \{ q \in V \mid d_q \geq \xi |\lambda_k| \text{ for } \xi \in (0, 1) \}. \]

Notice that

\[ p \notin S_{\xi}^k \implies 0 < \frac{d_p}{|\lambda_k|} < \xi < 1. \]

Also notice that if an item is a global extrema, then it is certainly a local extrema. This implies that if \( p \) is such that

\[ |w_p^{(k)}| = \max_{q \in V} |w_q^{(k)}|, \]

then from part (i)

\[ d_p \geq |\lambda_k|, \]
so for any $\xi \in (0,1)$, it is true that $\xi \leq \frac{d}{\lambda_k}$. Now define a new set $R^k_{\xi,s}$ to be the set of vertices at least $s$ hops away from $S^k_{\xi}$. In set notation this is

$$R^k_{\xi,s} = \{ p \in \mathcal{V} \mid \text{dist}(p, S^k_{\xi}) \geq s \text{ and } p \notin S^k_{\xi} \}.$$ 

By definition this set satisfies some criteria

- $S^k_{\xi} \cup R^k_{\xi,1} = \mathcal{V}$.
- If $s \geq 1$ then $S^k_{\xi} \cap R^k_{\xi,s} = \emptyset$.
- For all $s \geq 1$ $R^k_{\xi,s+1} \subset R^k_{\xi,s}$.

Relying on the fact that $\frac{1}{d_j} \sum_{i \in \mathcal{N}_j} |w_i^{(k)}| \leq \max_{i \in \mathcal{N}_j} |w_i^{(k)}|$, because the average of a set of numbers is bounded by their maximum. Using the fact that a maximum over a subset is less than the maximum over a whole set

$$\max_{j \in R^k_{\xi,1}} |w_j^{(k)}| \leq \max_{j \in R^k_{\xi,1}} \left[ \frac{d_j}{|\lambda_k|} \left( \frac{1}{d_j} \sum_{p \in \mathcal{N}_j} |w_p^{(k)}| \right) \right]$$

$$\leq \max_{j \in R^k_{\xi,1}} \left[ \frac{d_j}{|\lambda_k|} \left( \max_{p \in \mathcal{N}_j} |w_p^{(k)}| \right) \right]$$

$$\leq \max_{j \in R^k_{\xi,1}} \left[ \frac{d_j}{|\lambda_k|} \left( \max_{p \in \mathcal{V}} |w_p^{(k)}| \right) \right]$$

$$= \xi \max_{p \in \mathcal{V}} |w_p^{(k)}|.$$ 

The general result can be established by induction on $s$, starting with proving it for $s = 2$, and showing that it is true for any $s \in \mathbb{N}$. 
• Case: \( s = 2 \).

\[
\max_{j \in R_{\xi,2}^k} |w_j^{(k)}| \leq \max_{j \in R_{\xi,2}^k} \left[ \frac{d_j}{|\lambda_k|} \left( \frac{1}{d_j} \sum_{p \in N_j} |w_p^{(k)}| \right) \right]
\leq \max_{j \in R_{\xi,2}^k} \left[ \frac{d_j}{|\lambda_k|} \left( \max_{p \in N_j} |w_p^{(k)}| \right) \right]
\leq \max_{j \in R_{\xi,2}^k} \left[ \frac{d_j}{|\lambda_k|} \left( \max_{p \in R_{\xi,1}^k} |w_p^{(k)}| \right) \right]
\leq \max_{j \in R_{\xi,2}^k} \left[ \frac{d_j}{|\lambda_k|} \xi \max_{p \in V} |w_p^{(k)}| \right]
\leq \xi \max_{p \in V} |w_p^{(k)}|.
\]

• Case: \( s = 3, 4, 5, \ldots, m \). Assume now that

\[
\max_{j \in R_{\xi,m-1}^k} |w_j^{(k)}| \leq \xi^{m-1} \max_{p \in V} |w_p^{(k)}|.
\]

Then

\[
\max_{j \in R_{\xi,m}^k} |w_j^{(k)}| \leq \max_{j \in R_{\xi,m}^k} \left[ \frac{d_j}{|\lambda_k|} \left( \frac{1}{d_j} \sum_{p \in N_j} |w_p^{(k)}| \right) \right]
\leq \max_{j \in R_{\xi,m}^k} \left[ \frac{d_j}{|\lambda_k|} \left( \max_{p \in N_j} |w_p^{(k)}| \right) \right]
\leq \max_{j \in R_{\xi,m}^k} \left[ \frac{d_j}{|\lambda_k|} \left( \max_{p \in R_{\xi,(m-1)}^k} |w_p^{(k)}| \right) \right]
\leq \max_{j \in R_{\xi,m}^k} \left[ \frac{d_j}{|\lambda_k|} \xi^{m-1} \max_{p \in V} |w_p^{(k)}| \right]
\leq \xi^m \max_{p \in V} |w_p^{(k)}|.
\]

This theorem then says that the maximums of the eigenvectors can be found at high-degree nodes when the eigenvectors are associated with the largest eigenvalues. The quick rate of decay of
these eigenvectors away from these high-degree nodes is not the tightest it could possibly be, and in real-world examples decay is often more dramatic. These two facts would, of course, be of interest when designing iterative methods focused on the eigenspaces. Recall that for scale-free graphs, the extremal eigenvalues of low-rank corrections to $A$ are also related to the high-degree vertices and their surroundings. Specifically for $B$, a similar result to Theorem 6.3.1 can be established.

**Corollary 6.3.1. (Modularity Matrix)** Let $V = V(B)$ and $\Theta = \Theta(B)$ represent $K$ eigenpairs of $B$ with no zero eigenvalue

$$V^t e_i = d_i \Theta^{-1} \left( \frac{1}{d_i} \sum_{j \in N_i} V^t e_j - c \right),$$

where vector $c \in \mathbb{R}^K$ is independent of $i$, and is a degree-weighted average of all spectral coordinates

$$c = \left( \frac{1}{2m} \sum_{j=1}^{n} d_j V^t e_j \right).$$

**Proof.** Employing the symmetry of $\Theta$ and $B$

$$\Theta V^t e_i = V^t B e_i = V^t \left( A e_i - \frac{1}{2m} dd^t e_i \right) = \left( \sum_{j \in N_i} V^t e_j - \frac{d_i}{2m} \sum_{j=1}^{n} d_j V^t e_j \right).$$

By assumption, there are no zero eigenvalues in the partial eigendecomposition, so diagonal matrix $\Theta$ is invertible

$$V^t e_i = d_i \Theta^{-1} \left( \frac{1}{d_i} \sum_{j \in N_i} V^t e_j - \frac{1}{2m} \sum_{j=1}^{n} d_j V^t e_j \right).$$

The global connectivity in the rank-one term, $-\frac{1}{2m} dd^t$, makes it more difficult to make arguments about local maximums of $B$. However, a global maximum principle is established, as well as a degree-dependent decay factor that holds within one connection from the vertex.

**Theorem 6.3.3.** Let $(\theta_k, v_k)$ be eigenpairs of $B$. Then the following results hold

(i) **Global Maximum Principle.** If $v_k$ has a global maximum at $i$, and $\theta_k \neq 0$, then $d_i \geq |\theta_k|/2$. 
(ii) Degree-Based Drop Factor. For any vertex $i$, with $\theta_k \neq -\frac{d_i^2}{2m}$

$$|v_i^{(k)}| \leq \frac{2d_i - \frac{d_i^2}{2m}}{|\theta_k + \frac{d_i^2}{2m}|} \max_{p \in \mathcal{V}} |v_p^{(k)}|.$$ 

Proof. First (ii) is proved. Looking at the $k^{th}$ coordinate in Equation (6.4)

$$\theta_k v_i^{(k)} = \sum_{j \in N_i} v_j^{(k)} - \frac{d_i}{2m} \sum_{p=1}^n d_p v_p^{(k)}$$

$$\theta_k v_i^{(k)} + \frac{d_i^2}{2m} v_i^{(k)} = \sum_{j \in N_i} v_j^{(k)} - \frac{d_i}{2m} \sum_{p \neq i} d_p v_p^{(k)}$$

$$\left(\theta_k + \frac{d_i^2}{2m}\right) v_i^{(k)} = \sum_{j \in N_i} v_j^{(k)} + \frac{d_i}{2m} \sum_{p \neq i} d_p (-v_p^{(k)}).$$

Taking absolute values of both sides and applying the triangle inequality gives

$$\left| \theta_k + \frac{d_i^2}{2m} \right| |v_i^{(k)}| \leq \sum_{p=1}^n \omega_p |v_p^{(k)}|,$$

where the right-hand side is a weighted sum of $v_k$ over all vertices with weights

$$\omega_p = \begin{cases} 0 & i = p \\ 1 + \frac{d_id_p}{2m} & p \in N_i \\ \frac{d_id_p}{2m} & p \notin N_i. \end{cases}$$

Notice that the sum of the weights is

$$\sum_{p=1}^n \omega_p = \left( \sum_{p \in N_i} 1 \right) + \left( \sum_{p \notin N_i} \frac{d_id_p}{2m} \right) - \frac{d_i^2}{2m} = 2d_i - \frac{d_i^2}{2m}. $$

Rewrite the weighted sum as a weighted average times a scale factor to see (ii)

$$|v_i^{(k)}| \leq \frac{2d_i - \frac{d_i^2}{2m}}{|\theta_k + \frac{d_i^2}{2m}|} \left( \frac{1}{2d_i - \frac{d_i^2}{2m}} \sum_{p=1}^n \omega_p |v_p^{(k)}| \right) \leq \frac{2d_i - \frac{d_i^2}{2m}}{|\theta_k + \frac{d_i^2}{2m}|} \max_{p \in \mathcal{V}} |v_p^{(k)}|.$$ 

To prove (i) for the case $\theta_k > 0$ implies $\theta_k \neq -\frac{d_i^2}{2m}$ for all $i \in \mathcal{V}$. Then it is possible to demonstrate the weaker, yet more succinct bound

$$|v_i^{(k)}| \leq \frac{2d_i - \frac{d_i^2}{2m}}{|\theta_k + \frac{d_i^2}{2m}|} \max_{p \in \mathcal{V}} |v_p^{(k)}| \leq \frac{2d_i}{|\theta_k|} \max_{p \in \mathcal{V}} |v_p^{(k)}|. $$
This implies the desired result, as \( \frac{2d_i}{|\theta_k|} \) needs to be greater than or equal to 1 for a global maximum to occur at vertex \( i \).

For the case \( \theta_k < 0 \), the algebra is similarly straightforward:

\[
|\theta_k| |v_i^{(k)}| \leq \sum_{j \in \mathcal{N}_i} |v_j^{(k)}| + \frac{d_i}{2m} \sum_{p=1}^{n} d_p |v_p^{(k)}|,
\]

and proceeding in a similar manner to the proof of (ii) (forming a weighted average and taking a maximum) yields the result.

\[ \square \]

Numerical results that illustrate the theory are shown in the next section.

6.4 Numerical Results

6.4.1 Synthetic Graphs

To demonstrate the decay rates and community resolution properties of extremal eigenvectors of \( A \) and \( B \), the first series of tests embeds cliques of various size into graphs from simple scale-free graph generators that yield graphs with very little community structure. The graph generator is initialized with a small star (a single hub vertex attached to nine vertices). Then preferential attachment [86] is used to introduce new vertices, one at a time, by connecting them to existing vertices with probability proportional to their degree, until there are 25,000 vertices. The initial hub is highly likely to be the vertex of highest degree when the generation process is complete. As a graph is generated each vertex’s level or distance from the initial hub vertex is tracked. Then level is used to order vertices to display the maximum principles and decay properties of extremal eigenvectors of \( A \) and \( B \).

Two different versions of this generator, \texttt{pattach1} and \texttt{pattach2} were used, where each new vertex connects to one or two existing edges, respectively. A graph that is generated by such processes has scale-free degree distribution and is reasonably small in diameter, yet exhibits virtually no community structure. Process \texttt{pattach1} generates a tree, and \texttt{pattach2} generates a
graph that is the union of two trees where presence of relatively many short cycles is extremely unlikely.

After a preferential attachment graph is generated, significant community structure is attached to vertices furthest away from the initial hub. Specifically, $N$ cliques of size $c$ are attached, each to a single vertex in the periphery of the preferential attachment graph, with $N$ chosen so that roughly 10 percent of vertices in the final graph are involved in a $c$-clique. The size of $c$ is varied, then the top several eigenpairs are recalculated, and properties of the eigenvalues and associated eigenvectors are investigated. With this model, it can generally be observed that when $c$ is smaller than $|\lambda_k|$, the $k^{th}$ eigenvalue of $A$ is insensitive to the presence of the community structure. Tables 6.1 and 6.2 provide a few examples of this phenomenon for $\text{pattach1}$ and $\text{pattach2}$ and show how related the eigenvalues are to the square roots of the largest degree, verifying the bounds in Theorem 6.2.1 are met. For $B$, however, the eigenvalues are no longer as insensitive to the addition of the low-degree community structure. This is due to the global effect of the rank-one update. However, as expected from Corollary 6.2.1, the tables show that the eigenvalues are still related to the highest degree nodes.

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\sqrt{d_k}$</th>
<th>$c = 0$, $\lambda_k$</th>
<th>$c = 0$, $\lambda_{n+1-k}$</th>
<th>$c = 11$, $\lambda_k$</th>
<th>$c = 11$, $\lambda_{n+1-k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+17.8606</td>
<td>+17.9994</td>
<td>-17.9994</td>
<td>+16.9559</td>
<td>-18.8972</td>
</tr>
<tr>
<td>3</td>
<td>+11.9583</td>
<td>+12.1465</td>
<td>-12.1465</td>
<td>+11.8502</td>
<td>-12.2748</td>
</tr>
<tr>
<td>4</td>
<td>+10.7238</td>
<td>+10.7443</td>
<td>-10.7443</td>
<td>+10.6742</td>
<td>-11.0603</td>
</tr>
<tr>
<td>5</td>
<td>+10.4881</td>
<td>+10.4183</td>
<td>-10.4183</td>
<td>+10.3902</td>
<td>-10.5619</td>
</tr>
<tr>
<td>$k$</td>
<td>$\sqrt{d_k}$</td>
<td>$c = 11$, $\lambda_k$</td>
<td>$c = 11$, $\lambda_{n+1-k}$</td>
<td>$c = 11$, $\lambda_k$</td>
<td>$c = 11$, $\lambda_{n+1-k}$</td>
</tr>
<tr>
<td>1</td>
<td>+17.8606</td>
<td>+17.9994</td>
<td>-17.9994</td>
<td>(+17.3807)</td>
<td>(-18.5572)</td>
</tr>
<tr>
<td>2</td>
<td>+12.6886</td>
<td>+12.9172</td>
<td>-12.9172</td>
<td>(+12.7838)</td>
<td>(-13.0253)</td>
</tr>
<tr>
<td>3</td>
<td>+11.9583</td>
<td>+12.1465</td>
<td>-12.1465</td>
<td>(+12.0546)</td>
<td>(-12.2277)</td>
</tr>
<tr>
<td>4</td>
<td>+10.7238</td>
<td>+10.7443</td>
<td>-10.7443</td>
<td>(+10.7314)</td>
<td>(-10.9266)</td>
</tr>
<tr>
<td>5</td>
<td>+10.4881</td>
<td>(+10.4780)</td>
<td>-10.4183</td>
<td>(+10.4614)</td>
<td>(-10.5325)</td>
</tr>
</tbody>
</table>

Table 6.1: Adjacency and Modularity eigenvalues of $\text{pattach1}$ with no cliques attached and with cliques attached. Parentheses highlight eigenvalues that have changed significantly with the addition of the cliques.

Community structure is considered visible to a certain eigenpair if the associated spectral
<table>
<thead>
<tr>
<th>k</th>
<th>( \sqrt{d_k} )</th>
<th>Adjacency Matrix</th>
<th>Modularity Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>( c = 0, \lambda_k )</td>
<td>( c = 0, \lambda_{n+1-k} )</td>
</tr>
<tr>
<td>1</td>
<td>+21.4709</td>
<td>+22.1004</td>
<td>-21.6228</td>
</tr>
<tr>
<td>2</td>
<td>+17.7482</td>
<td>+17.8817</td>
<td>-17.9628</td>
</tr>
<tr>
<td>3</td>
<td>+15.0333</td>
<td>+15.5960</td>
<td>-15.3273</td>
</tr>
<tr>
<td>4</td>
<td>+14.8997</td>
<td>+15.1071</td>
<td>-15.2066</td>
</tr>
<tr>
<td>5</td>
<td>+14.3527</td>
<td>+14.6567</td>
<td>-14.6572</td>
</tr>
<tr>
<td></td>
<td>( \sqrt{d_k} )</td>
<td>( c = 16, \lambda_k )</td>
<td>( c = 16, \lambda_{n+1-k} )</td>
</tr>
<tr>
<td>1</td>
<td>+21.4709</td>
<td>+22.1004</td>
<td>-21.6228 (20.9455)</td>
</tr>
<tr>
<td>2</td>
<td>+17.7482</td>
<td>+17.8817</td>
<td>-17.9628 (17.6697)</td>
</tr>
<tr>
<td>3</td>
<td>+15.0333</td>
<td>+15.5960</td>
<td>-15.3273 (15.4427)</td>
</tr>
<tr>
<td>4</td>
<td>+14.8997</td>
<td>+15.1071</td>
<td>-15.2066 (15.1072)</td>
</tr>
<tr>
<td>5</td>
<td>+14.3527</td>
<td>(+15.0761)</td>
<td>-14.6572 (15.0756)</td>
</tr>
</tbody>
</table>

Table 6.2: Adjacency and Modularity eigenvalues of \texttt{pattach2} with no cliques attached and with cliques attached. Parentheses highlight eigenvalues that have changed significantly with the addition of the cliques.

coordinates of any vertex in a community are maximal in magnitude. Vertices are considered invisible if their magnitude is several orders of magnitude lower than the maximum. For \( A \), a clique is only visible to \( v_k \) if the degree of its members is \((c - 1) \approx |\lambda_k| \) or larger, just as the maximum principle suggests. Table 6.3 shows how far one needs to dig into the spectrum for \texttt{pattach1} and \texttt{pattach2} to detect any of a large number of cliques embedded into the graph.

For \texttt{pattach1} with \( c = 11 \) and \( N = 250 \), the left-hand side of Figure 6.2 demonstrates the exponential decay of the eigenvectors associated with the largest eigenvalues of \( A \) and \( B \), as described in Theorems 6.3.2(ii) and 6.3.3(ii). In the right-hand side of Figure 6.2, the eigenvectors associated with the fifth-most-positive eigenvalue of each matrix is plotted, where the maximum is obtained on a vertex involved in a clique. This is possible because \((c - 1) \approx |\lambda_5| \) and the requirement imposed on a vertex that is a local maximum from Theorem 6.3.2(i) is satisfied. Figure 6.3 displays the analogous results for \texttt{pattach2} with \( c = 16 \) and \( N = 167 \). These simple synthetic examples give clear demonstrations of the implications of the theory.
Figure 6.2: Magnitude for eigenvectors associated with the 1\textsuperscript{st} and 5\textsuperscript{th} most positive eigenvalues of $A$ (top row) and $B$ (bottom row) for \texttt{pattach1}, $c = 11$ and $N = 250$. The vertices from the graph generator are ordered by level (distance from the initial hub), as indicated by the vertical gray lines, and the vertices associated with the community structure are listed last. Red dots represent values associated with the $c$-cliques. For $A$ (top left) a decay bound of order $(2/|\lambda_1|)^{\text{level}}$ is plotted to demonstrate that the decay is exponential and related to the size of the eigenvalue, as Theorem 6.3.2(ii) suggests. On the top and bottom right we see that the 11-cliques are large enough to be maximums for the 5\textsuperscript{th} eigenvalue of both matrices, as determined in Theorems 6.3.2(i), 6.3.3(i). Note the logarithmic vertical axes have different scales.
Figure 6.3: Magnitude for eigenvectors associated with the 1\textsuperscript{st} and 5\textsuperscript{th} most positive eigenvalues of $A$ (top row) and $B$ (bottom row) for \texttt{pattach2}, $c = 16$ and $N = 167$. The vertices from the graph generator are ordered by level (distance from the initial hub), as indicated by the vertical gray lines, and the vertices associated with the community structure are listed last. Red dots represent values associated with the $c$-cliques. On the left, we see exponential decay as we move away from the vertex of maximal degree. On the right we see that the 16-cliques are large enough to be maximums for the 5\textsuperscript{th} eigenvalue, as determined in Theorems 6.3.2(i), 6.3.3(i). Note the logarithmic vertical axes have different scales.
Figure 6.4: Decay and sparsity plots for real-world example amazon0312

Adjacency Decay of Eigenvector 1

Modularity Decay of Eigenvector 1
<table>
<thead>
<tr>
<th>Graph Generator</th>
<th>$c$</th>
<th>$N$</th>
<th>Adjacency Eigenpair</th>
<th>Modularity Eigenpair</th>
</tr>
</thead>
<tbody>
<tr>
<td>pattach1</td>
<td>6</td>
<td>500</td>
<td>77$^{th}$</td>
<td>68$^{th}$</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>358</td>
<td>31$^{st}$</td>
<td>31$^{st}$</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>250</td>
<td>5$^{th}$</td>
<td>5$^{th}$</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>139</td>
<td>1$^{st}$</td>
<td>1$^{st}$</td>
</tr>
<tr>
<td>pattach2</td>
<td>6</td>
<td>500</td>
<td>$&gt;100^{th}$</td>
<td>$&gt;100^{th}$</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>358</td>
<td>70$^{th}$</td>
<td>70$^{th}$</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>250</td>
<td>19$^{th}$</td>
<td>19$^{th}$</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>167</td>
<td>5$^{th}$</td>
<td>5$^{th}$</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>109</td>
<td>1$^{st}$</td>
<td>1$^{st}$</td>
</tr>
</tbody>
</table>

Table 6.3: Most positive eigenpair for which embedded cliques of various size are visible for pattach1 and pattach2

6.4.2 A Real-World Graph

Here amazon0312 is studied, the undirected version of an Amazon product co-purchasing network, downloaded from the SNAP collection [82], which was built by crawling amazon.com on March 12, 2003. Vertices represent products for sale on the Web site. An edge exists between two products $i$ and $j$ if product $i$ is deemed as frequently bought by the same set of users that bought product $j$, or vice versa. No self loops are allowed in the adjacency matrix used. The graph has $n = 400,727$ vertices, and $m = 3,200,440$ edges. The total number of triangles is 3,686,467 and average clustering coefficient [78] is 0.4113. Ninety percent of the vertices are contained in a subgraph with diameter 7.7.

The 15 most positive eigenvalues of $A$ and $B$ with their associated eigenvectors were solved for. In Table 6.4 these eigenvalues and some graph characteristics related to the theory are listed. For each eigenvalue of $A$ a vertex $i$ is found for which the associated eigenvector is maximized in magnitude to demonstrate that the eigenvectors indicate high-degree structure. Here degree, $d_i$, number of local triangles, $t_i$, and the associated local measure from Lemma 6.2.1, $f(d_i, t_i)$, for vertex $i$ is reported. Note that the vertices are not guaranteed to be three hops away from each other, and some of the assumptions in Theorem 6.2.1 are not met for $K > 1$ and the vertices of largest degree. Similarly, for each eigenvalue of $B$, a vertex $j$ is found for which the associate eigenvector is maximized. Note that the extremal eigenvectors of $A$ and $B$ often take on their
maximums at the same high-degree vertex.

Additionally, it can be observed that the eigenvectors associated with the most positive eigenvalues decay rapidly and that the structures detected by these eigenvectors are hubs and their surroundings. On the left side of Figure 6.4, the eigenvector associated with the right-most eigenvalue is plotted, ordered by breadth-first search away from the vertex of maximal degree. Exponential decay can be observed as vertices move away from this vertex. On the right side of Figure 6.4, sparsity structure of the subgraph associated with the vertices having the 100 largest magnitude values in the eigenvector are plotted. While there are 648 nonzero entries (324 internal edges) on this block, there are 19,060 edges into the rest of the graph (683 internal and 16,477 external, respectively, for modularity). This structure is represented clearly by the extremal eigenvector because of a few high-degree vertices and not because of a strong community.

Based upon the theory, the work here shows that degree-based influence on the value of eigenvectors associated with the smallest eigenvalues is significant in many areas of the graph. This is relevant not only for making data-mining decisions based upon these eigenvectors, but should also prove to have relevance in other areas. For instance, due to the exponential decay away from high-degree nodes, if decisions are being made on nodes with very small values, care must be taken with setting tolerances for solves in order to overcome the numerical instabilities in those regions. However, of more interest is the effect of having large and small values in the eigenvectors when determining coarsening and interpolation in AMG approaches.

These results, however, can be very different when studying graph Laplacians. This is a more involved result that can’t be shown just through straightforward degree-based analysis of eigenspaces, and is the subject of ongoing research. It is unlikely that AMG will be aimed for the adjacency or modularity matrices, but instead for the graph Laplacians. However, understanding the underlying architecture of these matrices is an important first step toward designing optimal AMG preconditioners.
<table>
<thead>
<tr>
<th>$k$</th>
<th>$\lambda_k$</th>
<th>$i$</th>
<th>$d_i$</th>
<th>$t_i$</th>
<th>$f(d_i, t_i)$</th>
<th>$\theta_k$</th>
<th>$j$</th>
<th>$d_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>56.1956</td>
<td>1</td>
<td>2747</td>
<td>1550</td>
<td>52.9784</td>
<td>54.2444</td>
<td>1</td>
<td>2747</td>
</tr>
<tr>
<td>2</td>
<td>48.2261</td>
<td>3</td>
<td>2247</td>
<td>1475</td>
<td>48.0624</td>
<td>47.9351</td>
<td>3</td>
<td>2247</td>
</tr>
<tr>
<td>3</td>
<td>46.5151</td>
<td>2</td>
<td>2249</td>
<td>660</td>
<td>47.7178</td>
<td>46.4265</td>
<td>2</td>
<td>2249</td>
</tr>
<tr>
<td>4</td>
<td>42.3643</td>
<td>4</td>
<td>1413</td>
<td>1050</td>
<td>38.3385</td>
<td>42.1369</td>
<td>4</td>
<td>1413</td>
</tr>
<tr>
<td>5</td>
<td>40.9791</td>
<td>5</td>
<td>1282</td>
<td>3118</td>
<td>38.2991</td>
<td>40.7422</td>
<td>5</td>
<td>1282</td>
</tr>
<tr>
<td>6</td>
<td>39.9788</td>
<td>12</td>
<td>698</td>
<td>2297</td>
<td>29.8640</td>
<td>39.7048</td>
<td>12</td>
<td>698</td>
</tr>
<tr>
<td>7</td>
<td>38.4942</td>
<td>23</td>
<td>482</td>
<td>2300</td>
<td>27.1133</td>
<td>38.4360</td>
<td>23</td>
<td>482</td>
</tr>
<tr>
<td>8</td>
<td>37.7312</td>
<td>67</td>
<td>333</td>
<td>1946</td>
<td>24.7866</td>
<td>37.6437</td>
<td>67</td>
<td>333</td>
</tr>
<tr>
<td>9</td>
<td>37.5905</td>
<td>52</td>
<td>367</td>
<td>1837</td>
<td>24.6497</td>
<td>37.5569</td>
<td>52</td>
<td>367</td>
</tr>
<tr>
<td>10</td>
<td>37.3198</td>
<td>9</td>
<td>885</td>
<td>2266</td>
<td>32.3920</td>
<td>37.0684</td>
<td>9</td>
<td>885</td>
</tr>
<tr>
<td>11</td>
<td>36.0873</td>
<td>26</td>
<td>469</td>
<td>1402</td>
<td>24.8002</td>
<td>35.7036</td>
<td>26</td>
<td>469</td>
</tr>
<tr>
<td>12</td>
<td>34.7883</td>
<td>25</td>
<td>474</td>
<td>1669</td>
<td>25.5056</td>
<td>34.7228</td>
<td>25</td>
<td>474</td>
</tr>
<tr>
<td>13</td>
<td>34.4431</td>
<td>42</td>
<td>412</td>
<td>1734</td>
<td>24.8323</td>
<td>34.4125</td>
<td>42</td>
<td>412</td>
</tr>
<tr>
<td>14</td>
<td>34.4044</td>
<td>48</td>
<td>379</td>
<td>2005</td>
<td>25.2932</td>
<td>34.3912</td>
<td>48</td>
<td>379</td>
</tr>
<tr>
<td>15</td>
<td>34.3602</td>
<td>4</td>
<td>1413</td>
<td>1050</td>
<td>38.3385</td>
<td>34.3385</td>
<td>4</td>
<td>1413</td>
</tr>
</tbody>
</table>

Table 6.4: Most positive eigenvalues of $A$ and $B$ with locations of associated eigenvector maxima and local graph statistics for Amazon0312
Chapter 7

Conclusions and Future Work

7.1 AMG-DD/RD

When evaluating the AMG-DD/RD algorithm it is important to understand the context in which it must be evaluated. When discussing parallel algorithms, attention must be paid to the architecture on which the methods are intended to be utilized. It seems unrealistic to believe that there is one algorithm that will scale optimally regardless of architecture. Multigrid methods will continue to be a first choice for many researchers. What must be understood is that no matter the popular narrative, there is a delicate balance between computation and communication for any algorithm in parallel scientific computing.

Part of the conclusion from this research is that in an ideal setting, current AMG algorithms can be expected to continue to scale well on modern architectures. The models all suggest, too, that the balance between computation and communication is not the bottleneck that some believe it to be. AMG-DD/RD trades communication for useful computation, but the generic models do not immediately show that it will be a significant improvement over standard AMG methods on current architectures.

This is not the end of the story. One could imagine an architecture where even for as simple a problem as Laplace, it could be very beneficial. Consider, for instance, if there were an architecture with many powerful “nodes,” combined with very high communication cost between nodes. Then one might see dramatic improvement from this algorithm. Future research on the method is divided into three fronts: a full parallel implementation of the algorithm; investigating
the efficacy of the method on harder problems; and finally seeing if there are other ways to better utilize the AMG-DD/RD cycles.

7.2 Adaptive Smoothed Aggregation

Minimal progress was made in terms of implementation of these approaches, due to the significant engineering challenges they impose. The often subtle details of multigrid methods can have major effects on the perceived value of a particular algorithm. Much effort was put into developing prototypes that could reproduce the results from standard methods on a suite of test problems, starting from simple scalar Laplacian-type problems, then working through well-understood systems (e.g., linear elasticity), and, finally, to less well-understood systems of PDEs.

The methods have so far been applied in the context of simple scalar problems, and have recovered comparable results to standard approaches. The next steps going forward are to develop a series of tests and problems that will allow research to quantify the theoretical approaches outlined in this thesis. The goal is not only to improve AMG methods on systems of PDEs, but also to provide insight and metrics to aid future research in this area.

7.3 Multigrid for Scale-Free Graphs

The research presented in this thesis is mainly concerned with understanding the associated eigenspaces of adjacency and modularity matrices in the setting of linear systems formed from scale-free graphs. This is one step in the process of building a fundamental understanding of using iterative methods and numerical linear algebra to answer data-mining questions about these graphs. Future research is focused not just on continuing to build theoretical understanding of graph-associated matrices, but also on improving the performance of AMG solvers on these linear systems. Current work is focused on improving aggregation methods for AMG-type solvers.
Bibliography


[87] Dorit Ron and et al. Relaxation-based coarsening and multiscale graph organization.


