Improving Robustness in Multiscale Methods

by

Scott Patrick MacLachlan

B.Sc., University of British Columbia, 2000

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

2004

This thesis entitled:
Improving Robustness in Multiscale Methods
written by Scott Patrick MacLachlan
has been approved for the Department of Applied Mathematics

Stephen F. McCormick

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.

MacLachlan, Scott Patrick (Ph.D., Applied Mathematics)

Improving Robustness in Multiscale Methods

Thesis directed by Prof. Stephen F. McCormick

An important constraint on our ability to simulate physical processes numerically is our ability to solve the resulting linear systems. Multiscale methods, such as multigrid, provide optimal or near-optimal order solution techniques for a wide range of problems. Current multigrid methods (both geometric and algebraic) rely on the use of effective coarse-scale operators to achieve this efficiency. Classical geometric multigrid methods construct these operators based on the geometry of a given problem and are, thus, most effective when this geometry is known and simple. Algebraic multigrid methods are free from most geometric constraints, instead relying on assumptions about the character of the matrix that limit the applicability of such methods to a relatively small class of problems.

In this thesis, we first study the applicability of multigrid methods to the problem of flow through porous media, particularly motivated by the size of the linear systems resulting from discretization. For such problems, it is well known how to construct effective coarse-scale models for a correction-based multigrid method. For these robust multigrid solvers, we consider whether sufficient information is contained in the multigrid hierarchy to allow for efficient approximation of solutions of these linear systems. In particular, we examine important properties of the coarse-scale operators and the solutions to these coarse-scale equations and demonstrate their relevance to both the fine-scale and continuum-scale models.

Additionally, we consider the question of improvements to multigrid algorithms in the form of adaptivity or self-correction. By utilizing the multigrid method itself to expose slow-to-converge components, we may adapt the multigrid hierarchy in order

to improve performance, resulting in enhanced efficiency for a new class of problems. We present results for both the classical algebraic multigrid and smoothed aggregation frameworks illustrating typical multigrid efficiency with fewer assumptions on the given matrix systems. We consider theoretical issues of this adaptive procedure in the reduction-based algebraic multigrid setting.

For the friends and family who've made this possible.

Acknowledgements

This thesis is the result of a collaboration with a very talented group of people. First and foremost, I must thank my advisor, Steve McCormick, for his invaluable tutoring, advice, and perspective. Tom Manteuffel has also provided much needed guidance and insight into the problems considered here. I have benefitted immensely from my collaboration with Dave Moulton and my time spent at Los Alamos National Laboratory. Rob Falgout, John Ruge, and Marian Brezina have all contributed in their own ways to this work since its beginning, while more recent collaboration with James Brannick and Oren Livne has provided new insight and new direction. This work was sponsored by the Department of Energy under grant numbers DE-FC02-01ER25479 and DE-FG02-03ER25574, Lawrence Livermore National Laboratory under contract number B533502, and Los Alamos National Laboratory under the auspices of the U.S. Department of Energy, under contract W-7405-ENG-36.

My attraction to the University of Colorado was both academic and environmental. The comradeship amongst the graduate students here has been incredible and made my graduate school experience more enjoyable than I could ever have hoped. I have been fortunate to share an office with Chad Westphal and Luke Olson, which proved to be a wonderful working (and playing) environment. Just next door have been more colleagues and friends in Hans De Sterck, Jeff Heys, Eunjung Lee, Oliver Roerhle, Schorsch Schmidt and the rest of the computational math group, both past and present. Further afield, the friendship of Dan, Dave, Keith, Mark, Matt, Matt, Ryan, Uli, and others has

made my time in Boulder utterly enjoyable. Any remaining sanity I have is a testament as much to these people as to anything I have done.

After 21 years of formal education, there is no question that getting here was a result of the many wonderful teachers I've had over the years. Steve and Tom have been amazing technical instructors, but have also encouraged me to follow my other interests, often following them on foot or on bike. My instructors at the University of British Columbia were the best I could hope for, and I am especially grateful for the mentoring and advice of Eldad Haber, Anthony Peirce, and Jim Varah. My studies in mathematics were initially inspired by Cary Chien, one of the most dedicated teachers I have ever encountered.

I could not have achieved this degree without the support and encouragement of my family. Many thanks are due to my parents, who have always encouraged me to pursue my education and provided many helping hands. Finally, I wish to thank Alison for her support, friendship, and understanding while this thesis was being written.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

The focus of this thesis is on the numerical solution of the linear partial differential equations (PDEs) resulting from the mathematical modeling of physical systems. We assume that these PDEs have already been discretized in a sensible manner through the use of finite differences or finite elements [8], and our primary focus is on efficient solution of the linear systems, of the form $Ax = b$, that arise from such discretizations. These systems are typically large (current problems of interest involve millions or even billions of degrees of freedom (DOFs)), sparse (a fixed number of non-zero entries per row or column, regardless of problem size), and ill-conditioned (with condition number approaching infinity as problem size increases).

While these systems are sparse, in many cases they possess full inverses (though with banded factors), resulting in significant computational expense for direct methods of solution. Conventional iterative methods for these linear systems involve sparse, or easily computed, approximations to this inverse that damp, but do not completely eliminate, errors in an approximation to the solution, $x = A^{-1}b$.

Classical, stationary iterative methods choose an approximate inverse to the matrix A, $B \approx A^{-1}$, and iterate based on the residual equation. The residual equation relates the error in a current approximation, $\tilde{x}$, to the residual, $r = b - A\tilde{x}$. Writing $b = AA^{-1}b = Ax$, we see that $r = A(x - \tilde{x}) = Ae$. Thus, an exact correction to a given approximation is of the form $A^{-1}r$, leading to the iterative scheme,

$x^{k+1} = x^k + Br = (I - BA)x^k + Bb$. Considering this in error propagation form, we have

$$\begin{aligned} e^{k+1} = x - x^{k+1} &= x - x^k - Br \\ &= e^k - BAe^k = (I - BA)e^k, \end{aligned}$$

giving us the error propagation operator, $I - BA$. Thus, we can bound the ℓ_2 norm of e^{k+1}, as $\|e^{k+1}\| \leq \|I - BA\| \cdot \|e^k\|$, so $\|e^{k+1}\| \leq \|I - BA\|^{k+1} \cdot \|e^0\|$ and if $\|I - BA\| < 1$, the iterative method must converge.

Traditional choices of B, such as in the Jacobi, Gauss-Seidel, or SOR methods, result in error propagation matrices, $I - BA$, whose norm is not bounded uniformly below 1 with refinement in h, the discretization mesh size. As modern applications demand highly refined meshes, these methods alone are not appropriate for solving problems of interest.

Krylov subspace methods, such as Hestenes' and Steifel's conjugate gradient method [55], rely on a different approach to the solution of $Ax = b$. In these methods, the closest approximation to the solution of the linear system is found in a subspace whose size is iteratively increased. The convergence of these methods is dependent on the condition number of the matrix A, which again typically increases as h decreases.

To avoid the increase in iterations, it is common practice to modify the Krylov subspace method in the hopes of reducing the difficulties in solving the given system. Such preconditioning often takes the form of left preconditioning, where the system $Ax = b$ is multiplied on the left by a matrix, M^{-1}, chosen to reduce the condition number of the matrix to be iterated with, from $\kappa(A)$ to $\kappa(M^{-1}A)$. The generalized conjugate gradient algorithm [54] (a Krylov subspace method applied to $M^{-1}A$) is equivalent to applying conjugate gradient to the system $M^{-\frac{1}{2}}AM^{-\frac{1}{2}}y = M^{-\frac{1}{2}}b$, for $y = M^{\frac{1}{2}}x$, where M is assumed to be symmetric and positive definite.

Preconditioned conjugate gradient methods have been shown to be more effec-

tive than non-preconditioned Krylov methods for many problems of interest. For easily inverted M (that is, when $M^{-1}r$ is easily computed), the additional cost of the preconditioning in the Krylov iteration can easily pay dividends if it results in a more amenable spectrum of $M^{-1}A$. That is, if the condition number of $M^{-1}A$ (or $M^{-\frac{1}{2}}AM^{-\frac{1}{2}}$) is significantly less than that of A, or if the spectrum is more clustered than that of the original matrix, we can expect significantly fewer iterations to be needed. However, even with very sophisticated preconditioners (such as the incomplete Cholesky factorization), the condition number of the preconditioned system still increases as h decreases.

As an example of this situation, consider the case of a 2D Poisson problem,

$$-\Delta p = f$$

with Dirichlet boundary conditions on the unit square, discretized via finite differences on an $n \times n$ regular grid (eliminating boundary conditions). The resulting matrix, A, is an $n^2 \times n^2$ sparse array, with 5 non-zero entries in each row/column. The half-bandwidth of this matrix is n when the unknowns are ordered lexicographically.

Direct methods are clearly not scalable for this problem. A naive Gaussian elimination approach would require $O(n^6)$ operations, although would not need pivoting (at least in exact arithmetic) if boundary conditions are chosen to ensure positive definiteness (i.e., at least one Dirichlet or Robin boundary region exists). Elimination can, of course, be done in a way to preserve the banded structure of this matrix, reducing the operation count to $O(n^4)$.

Classical, stationary iterative approaches also scale poorly. The spectral radius of the (stationary) iteration matrix can be computed and then used to estimate the number of iterations needed to reduce the error to a particular level. As the matrix equation approximates a continuous equation, one reasonable choice of permissible error would be to allow errors in the solution of the linear system to be of the size of the error already present in the discretization. In this regular setting, that error is $O(h^2)$, where

$h = \frac{1}{n+1}$.

For the three most common iterative methods, this is easily done. Jacobi, Gauss-Seidel, and SOR for this problem can all be analyzed in terms of a Fourier sine basis. For Jacobi, the spectral radius of the matrix, A, is $\rho_J = 1 - 2\sin^2\left(\frac{\pi}{2(n+1)}\right)$. The approximate number of steps, s, necessary to reduce a given error by a factor of ϵ is the determined by $\epsilon = \rho_J^s$. Based on asymptotic approximations for large n, we see that $s \approx \log\left(\frac{1}{\epsilon}\right)\frac{2n^2}{\pi^2}$. Substituting $\epsilon \approx \frac{1}{n^2}$ yields the approximate number of Jacobi iterations, $s \approx \log(n^2)\frac{2n^2}{\pi^2}$. Each iteration costs one matrix-vector multiply, or approximately $5n^2$ operations. Thus, the ultimate operation count for Jacobi is $\log(n^2)\frac{10n^4}{\pi^2} = O(n^4\log(n))$.

For Gauss-Seidel, Varga [83, p. 121] demonstrates that the spectral radius of the iteration matrix is exactly the square of that of the Jacobi iteration matrix. The approximate operation count for Gauss-Seidel is then $\log(n^2)\frac{5n^4}{\pi^2} = O(n^4\log(n))$, half of that for Jacobi. Varga [83, pp. 123–125] also discusses a result of Young that gives the optimal overrelaxation parameter, $\omega_b = \frac{2}{1+\sqrt{1-\rho_{GS}^2}}$, and the convergence factor for the resulting iteration as $\rho_b = \omega_b - 1$. Using this, the needed number of SOR iterations can be calculated, giving the total operation count of approximately $\log(n^2)\frac{5n^3}{2\pi} = O(n^3\log(n))$.

Since the original linear system is SPD, the conjugate gradient method (CG) is a good representative of Krylov methods for this problem. Greenbaum [50, p. 51] shows that one step of CG on a matrix, A, with condition number κ reduces the error (measured in the A-norm) by a factor of at least $\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$. Thus, for large κ, the number of CG iterations to reduce the error by a factor of ϵ can be approximated as $\log\left(\frac{1}{\epsilon}\right)\frac{\sqrt{\kappa}-1}{2}$. For our model problem, $\epsilon \approx h^2$ and $\kappa = O(n^2)$. So, the required number of CG iterations is $O(n\log(n))$, giving $O(n^3\log(n))$ operations, and CG is (asymptotically) no faster than SOR. However, the modified incomplete Cholesky factorization, as analyzed by Gustafsson [51], can be shown to produce a preconditioned system with condition number $O(n)$, resulting in a preconditioned conjugate gradient (PCG) method that requires only $O(\sqrt{n}\log(n))$ iterations and $O(n^{2.5}\log(n))$ operations.

Considering these results graphically, as in Figure 1.1, we see that while non-optimal methods can be very competitive for large mesh sizes, they must lose out as $h \to 0$, regardless of the factor hidden in the $O(\cdot)$ notation.

Thus, none of these methods allows for solution of the model problem with an optimal number of computations, proportional (or nearly so) to the total number of degrees of freedom, N ($O(N)$ or $O(N \log N)$ behavior).

However, we notice that for this simple model problem, the techniques of fast Fourier transforms do allow solution of the same differential equation in $O(N \log N)$ operations [33, §6.7]. This suggests that there is some advantage to considering multiscale solution techniques for the linear system we are considering. Indeed, multiscale ideas can lead to efficient solution schemes for this and many other linear systems. Originally proposed by Fedorenko [46] and Bakhvalov [3], multigrid methods have seen a tremendous growth in their applicability since the initial work of Brandt [14, 17]. In particular, Bramble et al. [10, 11, 12] show that there is an optimal multigrid method for elliptic partial differential equations discretized by finite elements, while local mode analysis, such as in [16, 19], establishes multigrid optimality for discretizations of a particular class of PDEs.

The name multigrid comes from the fact that the original algorithms were based on a nested grid hierarchy, with intergrid transfer operators based on the geometry of these grids. For this reason, these classical multigrid algorithms are often referred to as geometric multigrid methods. While these methods perform well for PDEs with smooth coefficients, it has been observed that performance of classical, geometric multigrid methods breaks down when there is significant variation in the coefficients of the PDEs. As discontinuous-coefficient diffusion is an important problem, Alcouffe et al. [2] and Dendy [34] developed the black box multigrid method (BoxMG), by introducing the idea of operator-induced interpolation. In this method, the intergrid transfer operators are chosen based on the entries in A, assuming a regular tensor-product mesh. It can

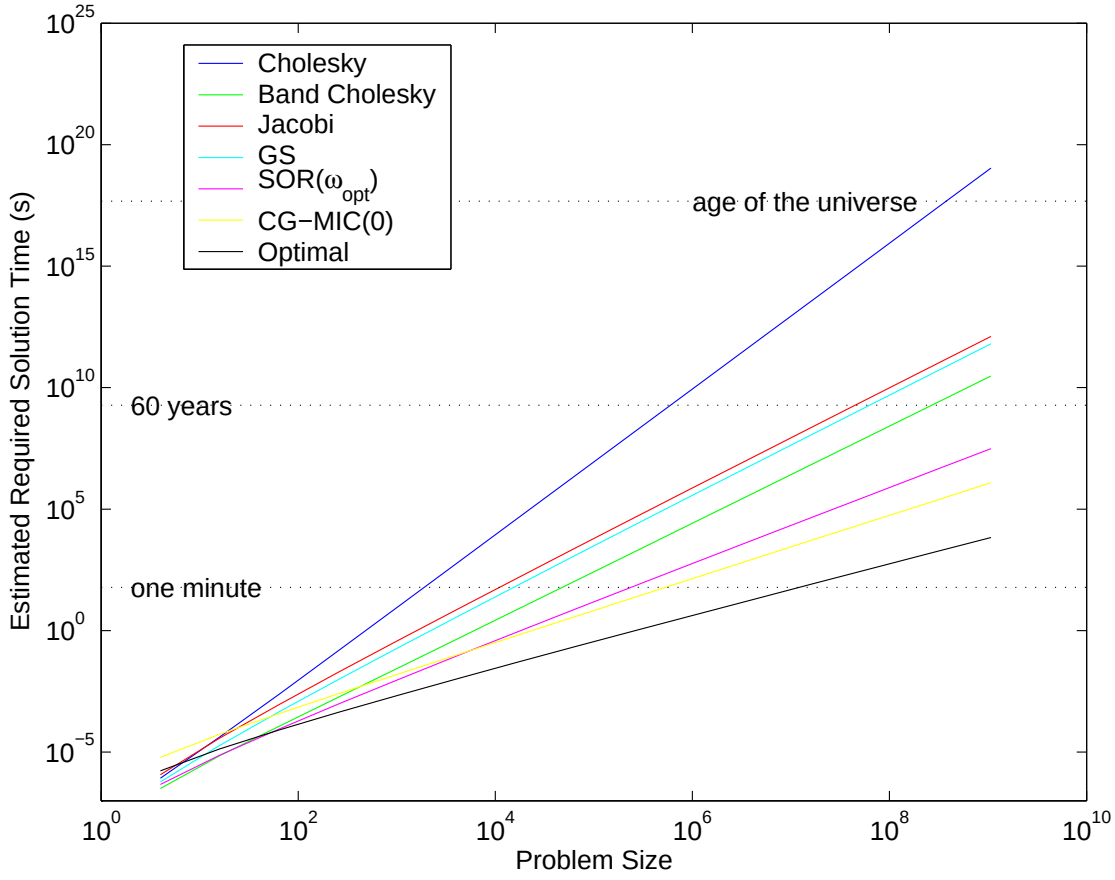

Figure 1.1: Scalability of Direct and Iterative Solution Techniques for the 2D Finite-Difference Laplacian as Problem Size Increases

be shown that the scheme developed in [2] approximately preserves the continuity of normal flux in the interpolation procedure [67].

As computational power has developed, so has the sophistication of discretizations. In particular, the finite element method allows discretization of a PDE on an arbitrary domain with a nearly-arbitrary mesh. The multigrid methods discussed so far are limited in their use of geometric properties of the mesh, either in the intergrid transfer operators as in geometric multigrid or in the assumption of a logically rectangular mesh in the black box multigrid algorithm. Algebraic multigrid methods were developed by Brandt, McCormick, and Ruge [23] to handle problems where regularity of the mesh could not be assumed. Further developed by Ruge and Stüben [72, 73], algebraic multigrid (AMG) has been shown to be quite efficient on a wide range of matrices, including those resulting from irregular meshes and discontinuous coefficients.

An interesting variant on AMG is the smoothed aggregation (SA) method, as developed by Vanek et al. [79, 80]. While classical AMG is developed based on the null space properties of typical discretizations of elliptic operators, the intergrid transfer operators used in SA are based on explicit knowledge of the near null space of the discretized operator. These methods are particularly effective for problems such as elasticity where the near null space is the span of several distinct vectors, all of which can be easily incorporated into the interpolation and restriction operators.

The problem of flow in porous media is one for which multigrid methods, particularly the BoxMG and AMG algorithms, are ideally suited. Modeled via Darcy's law and mass conservation, the resulting equation for pressure in a single-phase flow is simply a variable-coefficient diffusion equation. The main difficulty encountered in this problem is the disparity between the scale of variation of the permeability and the size of the domain. Typical simulations involve media with material properties (such as permeability) that vary on a scale of millimeters, while the domain may be several kilometers in length in each dimension. A fully-resolved, three-dimensional simulation

in this case would require more than 10^{18} degrees of freedom, resulting in a linear system that cannot be practically solved, even on the most powerful of modern computers.

There are many possible approaches to resolving this problem. At the simplest, one can choose a rule for averaging the variations below an acceptable computational scale independently of the permeability. These simple averages are typically insufficient for arbitrary permeability tensors. Numerical homogenization, or upscaling, techniques are used to seek a coarser discretization that preserves the effects of the fine-scale variation in the fully resolved problem. We consider the relationship between existing computational techniques and classical theoretical results and show that the two are quite closely linked. We also propose and examine an extension to the multigrid homogenization techniques of Moulton et al. [67] and Knapek [60] making use of operator-induced coarsening techniques (such as in BoxMG and AMG) to derive useful, multiscale basis functions, and the corresponding coarse-scale models.

In many other practical settings, full details of the discretization, or even of the continuum model, may not be available to the solver. The dependence of multigrid on complementarity, however, requires knowledge of the modes that are slow to converge under relaxation (and typically span the near null space of the discrete operator) to ensure multigrid optimality. Adaptive multigrid methods attempt to recover these modes iteratively and improve the complementarity of the coarse-grid correction and relaxation processes through their use. We introduce these ideas of adaptivity into the AMG and SA methods. Although much of this work is numerical, we also present new theoretical results that, while incomplete, indicate the success of the adaptive framework. Our research primarily focuses on the use of these components in the definition of interpolation. Related work, addressing the question of their use in choosing the coarse-grid points, is also underway.

Chapter 2 provides a brief introduction to the continuum models considered here. We focus primarily on the problem of single-phase, saturated flow through porous me-

dia (and the equivalent problem of discontinuous-coefficient diffusion) and discuss the importance of the permeability coefficient, $\mathcal{K}$, in the behavior of the solution. Discretizations of this equation are considered, and the finite element formulation we use is discussed. We will also make use of a few examples from linear elasticity, and this problem is also introduced here.

Chapter 3 discusses the aforementioned multigrid methods in more detail. Section 3.1 discusses the basic multigrid concepts and the classical, geometric multigrid methods. In Section 3.2, we discuss the details of the black box multigrid algorithm. Of particular interest is the relationship between the physical concept of flux and the multigrid operators used. Sections 3.3 and 3.4 review the fundamental components of the AMG and smoothed aggregation methods, particularly focusing on the automatic determination of coarse grids and of intergrid transfer operators.

Chapter 4 introduces the single-phase, saturated flow through porous-media problem in more detail, discussing both existing and new upscaling techniques in more detail. Here, we show the equivalence between common computational approaches to the problem and the classical theory. We also demonstrate that the operator-induced coarsening used in robust multigrid methods is appropriate for creating coarse-scale models that provide useful global-scale approximations.

Chapter 5 focuses on adaptivity in algebraic multigrid techniques. Important principles of adaptive multigrid methods are discussed in general, and then in specifics as applied to the AMG and SA methods. A theory of convergence is presented in the simplified AMGr framework, which indicates optimal performance of the adaptive setting. Important implementation issues for these methods are also discussed.

Chapter 2

Continuum Models

This thesis focuses primarily on issues of solving linear systems accurately and efficiently. These linear systems come from a variety of sources, including both model problems and realistic applications. Here we describe the continuum problems we seek to model, as well as our usual choice for discretization of finite elements.

2.1 The Diffusion Equation and Flow in Porous Media

Most of the results in this thesis are for the matrices that arise when discretizing the diffusion equation,

$$-\nabla \cdot \mathcal{K}(\mathbf{x}) \nabla p(\mathbf{x}) = Q(\mathbf{x}),$$

including the special case of Poisson's or Laplace's equation, when $\mathcal{K} \equiv 1$, $\Delta p = Q$. We primarily consider the two-dimensional case, although we briefly consider three dimensions in Sections 4.3 and 5.4.

Just as Poisson's equation can be used to model many different physical problems, the discontinuous-coefficient diffusion equation also has far-reaching applications. We focus here on the simulation of fluid flow in porous media, important in the study of oil reservoir performance and ground-water transport. This situation exhibits characteristics of other, similar applications, such as single-group monoenergetic neutron transport and electromagnetics, particularly in the dependence of the PDE coefficients on the properties of the simulated materials.

Saturated, single-phase flow in porous media may be approximated using Darcy's law. Assuming pointwise knowledge of the permeability of the media, $\mathcal{K}(\mathbf{x})$, fluid velocity is given by

$$\mathbf{u}(\mathbf{x}) = -\mathcal{K}(\mathbf{x})\nabla p(\mathbf{x}), \tag{2.1}$$

where $p(\mathbf{x})$ represents the pressure of the fluid, for $\mathbf{x} \in \Omega \subset \mathbb{R}^d$, for d = 1,2, or 3. In this case, $\mathcal{K}(\mathbf{x})$ is physically constrained to be a tensor-valued (allowing anisotropic flow characteristics), pointwise-symmetric, and positive-definite operator [4]. The flow is then fully characterized by imposing conservation of mass,

$$\nabla \cdot \mathbf{u}(\mathbf{x}) = Q(\mathbf{x}), \tag{2.2}$$

where $Q(\mathbf{x})$ represents any external fluid sources or sinks in the system. Physically, $Q(\mathbf{x})$ is quite important, representing injection and production wells in a reservoir. Mathematically, we simply require $Q(\mathbf{x}) \in H^{-1}(\Omega)$ (admitting useful models of these wells as point- or step-function sources). We do not concern ourselves with particular choices of Q, since we consider primarily characteristics of the matrix in the discretized linear system and not the right-hand side.

Many of the difficulties encountered in solving this PDE exist only because we consider the case of multi-dimensional flow. Indeed, the one-dimensional flow problem, $-\frac{\partial}{\partial x}\mathcal{K}(x)\frac{\partial}{\partial x}p(x) = Q(x)$, may be solved exactly, as

$$p(x) = -\int_0^x \frac{\int_0^y Q(t)dt + c_1}{\mathcal{K}(y)}dy + c_2,$$

where the constants, c_1 and c_2, are chosen to match the boundary conditions of the PDE. When $Q \equiv 0$, notice that there is no difference between this solution and that with $\mathcal{K}$ replaced by its harmonic mean, $\hat{\mathcal{K}} = \left(\frac{1}{x}\int_0^x \frac{dy}{\mathcal{K}(y)}\right)^{-1}$. In higher dimensions, no analogue of this approach exists, and so we consider numerical approaches instead.

Generalizations of this system are possible and commonly arise from physical situations of flow in porous media. For instance, we have chosen the permeability, $\mathcal{K}$, to

be dependent only on $\mathbf{x}$. In some flow regimes, such as when the fluid pressure is greater than the confining pressure of the media, the effect of the fluid pressure on permeability cannot be ignored. In the case of unsaturated flow, the flux becomes dependent on the saturation, S, and Equation 2.1 can be effectively replaced by an equation for the flux, $\mathbf{q}(\mathbf{x}) = D(S, \mathbf{x})\nabla S(\mathbf{x})$, where $D(S, \mathbf{x})$ is the diffusivity coefficient and is itself dependent on $\mathcal{K}(S, \mathbf{x})$. Conservation of mass is then imposed on this flux, $\nabla \cdot \mathbf{q}(\mathbf{x}) = Q(\mathbf{x})$, as in Equation 2.2. In this thesis, we consider only the linear case of saturated, single-phase flow.

2.2 Model and Realistic Permeability Fields

The solution of the system in Equations 2.1 and 2.2 is complicated by the nature of $\mathcal{K}(\mathbf{x})$. Permeability is a property of the medium and can thus vary significantly between media in the same simulation. A common example of this is in systems that involve a mixture of rocks and sands or gravels, where the permeability of a mixture of sand and gravel may be 6 orders of magnitude higher than that of sandstone or limestone. Even within a single medium, such as sand, permeabilities can vary by 4 orders of magnitude or more, depending on the composition and fineness of the sand. Thus, $\mathcal{K}(\mathbf{x})$ can be expected to vary significantly, both across the domain and within small volume elements.

We consider several model problems, where the permeability field is chosen to mimic characteristics of a physical problem, as well as synthetic data examples where the permeability field is generated geostatistically, modeling a physical situation. The results in Chapter 4 deal with both of these cases, while the focus in Chapter 5 is on issues of solving linear systems and only model problems are considered.

The simplest model problem considered, aside from Laplace's equation, is a two-material media with jumps in the material properties between the media. This may be viewed as an abstraction of the sand/shale problem common in flow through porous

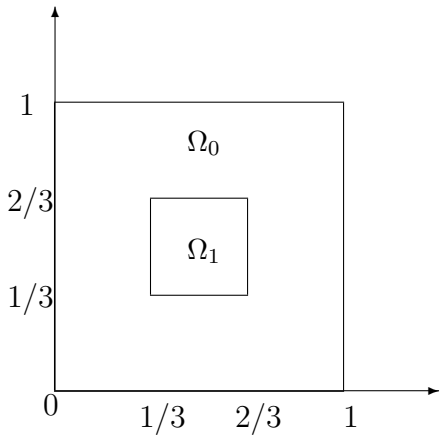

Figure 2.1: Square Inclusion Problem

media, where the domain is composed of a porous background medium (sand) with relatively impermeable inclusions.

We consider variations in both the shape and permeability of such inclusions. A simple example of a composite material, considered in both Chapters 4 and 5, is when the inclusion takes a regular shape and/or pattern, as in Figure 2.1. In this two-medium example, we consider choosing the permeability as piecewise constant,

$$\mathcal{K}(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in \Omega_0, \\ \lambda & \mathbf{x} \in \Omega_1. \end{cases}$$

Similar examples are prevalent in the homogenization and upscaling literature, and we introduce particular variations, including Bourgat's examples [7], in Section 4.1.2.

Another sand/shale problem common in the literature may be modeled by choosing a length scale for the impermeable inclusions and distributing them randomly throughout the domain. That is, given the inclusion length scale, H, divide the domain, Ω, into a grid of size H, and choose a fixed percentage of the elements on this scale (at random) to represent the impermeable media, as depicted in Figure 2.2, where Ω is the unit square, $H = \frac{1}{4}$, and approximately 20% of the volume is chosen to represent shale. A more difficult version of this problem occurs when $H = h$, i.e., the inclusion

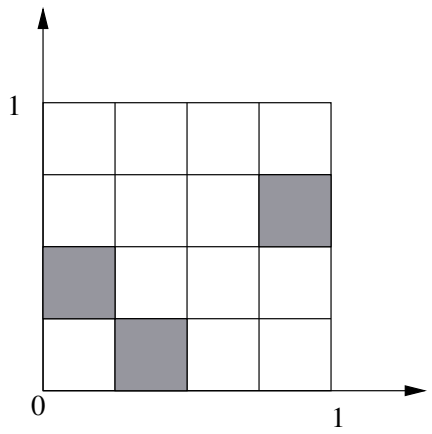

Figure 2.2: Sand/Shale Model Problem

length scale is also the discretization mesh size. This is considered as an example in Section 5.2.

Complete permeability fields for actual porous media are almost never available. In practice, permeability is known only at a few data points within the physical domain and geostatistical techniques must be used to generate a realization of the desired properties conditioned on the known values. We use one such program, gslib [38], to generate geostatistical permeability fields with no prior data for conditioning. In particular, we consider two fields chosen to mimic those used by He et al. [52]. The first field, pictured in Figure 2.3, exhibits mild layering effects aligned with the grid, due to the relatively close correlation lengths of $\mathcal{K}(\mathbf{x})$ of 0.5 in the x-directions and 0.1 in the y-direction. The second field, as in Figure 2.4, has principle axes rotated 30° from the horizontal and vertical, with a strong correlation in $\mathcal{K}$ of length 0.8 along the rotated horizontal axis, compared with a correlation length of 0.04 along the rotated vertical axis. In both of these cases, the permeability was chosen as piecewise constant on a 64×64 mesh from a log-normal distribution with variance (of $\log(\mathcal{K})$) of 4, resulting in permeabilities in the range $[10^{-2}, 10^{2}]$, where dark pixels in these figures correspond to low values of $\mathcal{K}$ and light pixels to high values.

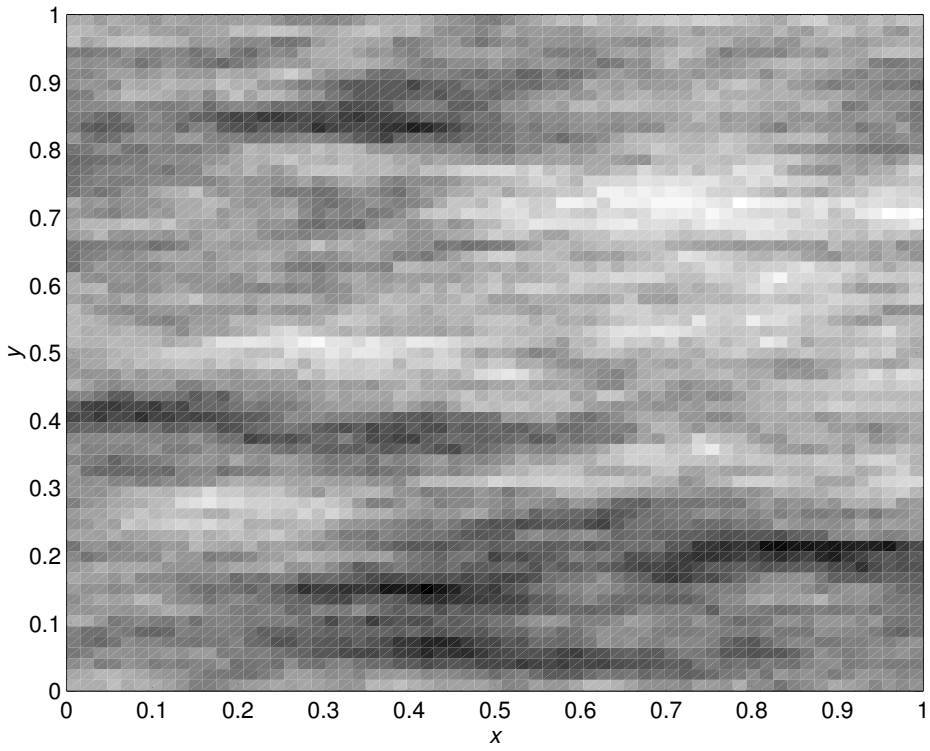

Figure 2.3: Mildly Layered Geostatistical Permeability Field

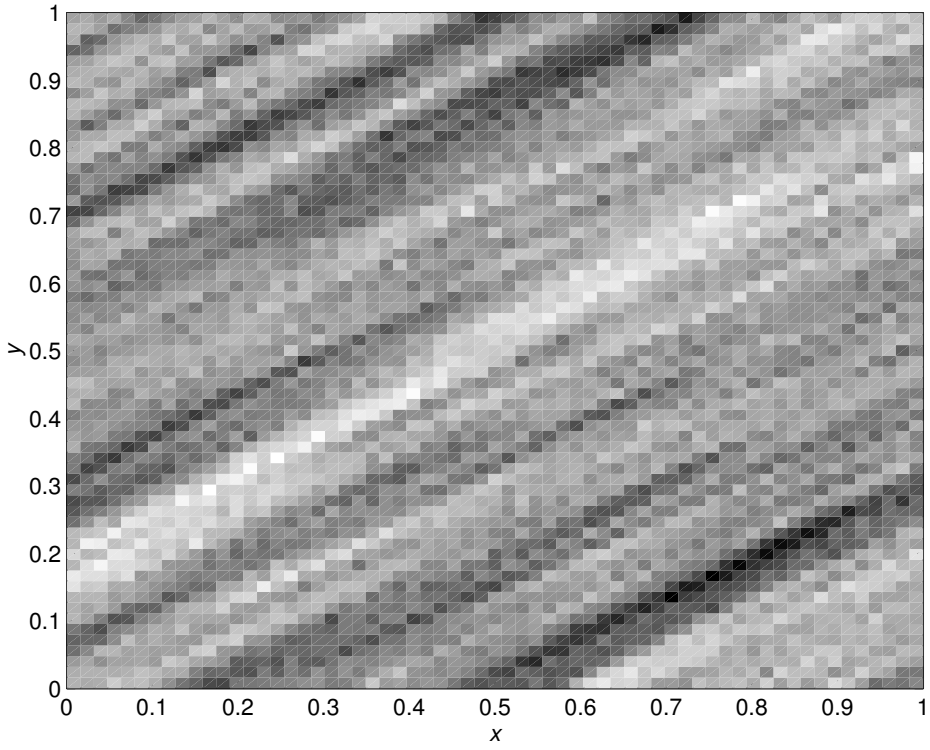

Figure 2.4: Strongly Layered Geostatistical Permeability Field

2.3 Discretizations

The primary factor in any discretization of Equations 2.1 and 2.2 is the severe variability of $\mathcal{K}(\mathbf{x})$ and its effect on the accuracy of the discretization. Many schemes have been employed to account for this variation, in the contexts of finite differences, finite volumes, and finite elements.

Finite difference and finite volume techniques typically rely on an averaging of cell permeabilities to ensure the appropriate continuity of flux. In finite differences, values of $\mathcal{K}(\mathbf{x})\frac{\partial}{\partial x_i}p(\mathbf{x})$ are approximated on auxiliary nodes located at the midpoints of the grid lines, using usual centered differences approaches. If $\mathcal{K}(\mathbf{x})$ is not given at these nodes, an averaging of the nearest known values is needed, and is typically taken to be the arithmetic or harmonic mean of cell-centered permeability data. Once these values have been approximated, the divergence may be imposed, again using a centered differences scheme.

Finite volume techniques typically place the unknowns at cell centers and focus on the approximation of fluxes into and out of each cell. If these fluxes are explicitly represented, Equation 2.2 is easily imposed by requiring balance between the integrated sources and sinks on a cell, $\int Q(\mathbf{x})d\mathbf{x}$, and the net flux along the boundary. Approximation of these fluxes may be done in many ways, but a simple, low-order technique is to reduce the problem to one dimension, taking the cell-centered values as boundary values and the known cell permeabilities as the diffusion coefficients for the one-dimensional problem. This problem may be solved explicitly, as discussed in Section 2.1, and results in the flux being proportional to the pressure gradient between the two nodes and to the harmonic average of the cell permeabilities.

Both of these techniques may be extended to higher order, using longer range differences or better approximations of the fluxes across cell boundaries, and such extensions are commonly used in simulations of flow in porous media. However, we consider

here discretizations via primal finite elements, due to their advantageous theoretical properties, particularly when posed variationally, as in Chapter 4.

We consider a conforming, Galerkin finite element discretization of the second-order PDE, $-\nabla\cdot\mathcal{K}(\mathbf{x})\nabla p(\mathbf{x}) = Q(\mathbf{x})$, with a nodal basis for our test/trial space. Writing $p(\mathbf{x}) = \sum_i p_i\phi_i(\mathbf{x})$, we ask that, for every j,

$$\sum_i p_i \int_\Omega \left(\mathcal{K}(\mathbf{x})\nabla\phi_i(\mathbf{x})\right)\cdot\left(\nabla\phi_j(\mathbf{x})\right) d\Omega = \int_\Omega Q(\mathbf{x})\phi_j(\mathbf{x}) d\Omega.$$

We consider the space spanned by $\{\phi_i\}$ to be piecewise-bilinear functions on a rectangular mesh in two dimensions and piecewise-trilinear functions on a cubic mesh in three dimensions. $\mathcal{K}(\mathbf{x})$ is taken to be piecewise constant on each element of the mesh, and each integral is evaluated exactly under this assumption. The resulting linear system of equations may be written as $A\mathbf{x} = \mathbf{b}$, where the matrix, A, has entries $a_{ij} = \int_\Omega \left(\mathcal{K}(\mathbf{x})\nabla\phi_i(\mathbf{x})\right)\cdot\left(\nabla\phi_j(\mathbf{x})\right) d\Omega$, and the vector, $\mathbf{x}$, has values corresponding to p_i.

2.4 Elasticity

In Section 5.4, we consider matrices arising from discretizations of a PDE model describing linear elasticity. The basic equations of linear elasticity may be expressed as the system of equations in $\mathbf{u}$, the displacement of an elastic body:

$$(\lambda+\mu)\nabla\nabla\cdot\mathbf{u} + \mu\boldsymbol{\Delta}\mathbf{u} = \mathbf{f},$$

where λ and μ are parameters of the material, known as Lamé coefficients.

We consider such problems only briefly, with a 2D discretization with bilinear finite elements on the unit square and a 3D discretization with trilinear finite elements on the unit cube. In both cases, we consider Dirichlet boundary conditions on one face of the domain, fixing the displacements along that face, and some form of Neumann boundary condition on the remaining faces.

Difficulty of the problem is controlled primarily by the material properties, λ and μ, which can be thought of as controlling the relative importance of the Laplacian and

grad-div terms in the equation. Another choice of parametrization is with the Poisson ratio, defined as $\nu = \frac{\lambda}{2(\lambda+\mu)}$, and the Young modulus, $E = \frac{\mu(3\lambda+2\mu)}{\lambda+\mu}$.

The linear elasticity problem becomes harder as $\nu \to \frac{1}{2}$, as in this limit the grad-div term dominates the system and the effect of the Laplacian term is lost. In the examples considered in Section 5.4, we fix $\nu = 0.3$, resulting in a one-parameter system in E. Most of the results presented are for fixed E; however, in Table 5.17, we also consider the case of E chosen randomly between 1 and 10^σ, for $\sigma = 2, 3, 4$, and 5.

Chapter 3

Background

While modern interest in multigrid methods dates back only about 30 years, the basic principles of multiscale ideas have a longer history [16, §10]. Indeed, as early as 1935, Southwell [74, 75] proposed what he called block or group relaxation strategies for resolving displacements (and, thus, stresses) in structural problems. In these works, Southwell proposes a method where blocks of nodes corresponding to particular sub-structures are relaxed as one, keeping their internal configuration, but moving as a unit relative to the remainder of the grid. Stiefel [76] considers general relaxation methods, including Southwell's, and discusses ways of accelerating the relaxation process. Also discussed is the property that (Gauss-Seidel) relaxation is, in general, least efficient at resolving modes associated with small eigenvalues of the matrix, A.

In the 1960's, two-level methods attracted notable attention. Fedorenko [45] considered a two-level correction method where smooth error is resolved by considering a restriction of the fine-grid equations to a sub-grid. He demonstrates numerically that this method is more efficient than SOR, by a factor of approximately 3 for a 50×50 mesh. Ahamed [1] applied Southwell's techniques to vector potential problems, commenting on the lack of adequate numerical methods for electromagnetic problems and demonstrating solution of finite-difference Poisson problems with a few hundreds of unknowns. Wachspress [84] considers non-symmetric matrices, with equations in A and A^*, using a variational technique to arrive at a set of contracted equations for the

fine-grid solution. This two-level method bears significant similarity to the geometric multigrid discussed in Section 3.1. De la Vallee Poussin [32] shows that the method outline by Stiefel is convergent, and demonstrates convergence independent of the size of the jump in diffusion coefficient.

True multilevel techniques also originally appeared in the 1960's. Fedorenko introduced the idea for theoretical purposes in [46]. In fact, he proposed a multigrid technique and then demonstrated that the number of iterations needed to reduce the residual of the 2D Laplacian on a regular, rectangular grid of N points by a factor ϵ was $O\left((N \log\left(\frac{1}{\epsilon}\right)\right)$. Bakhvalov [3] provided a more general result, applicable to second-order elliptic operators with continuous coefficients. Both papers provide significant theoretical results, but these results obscure the true usefulness of multilevel techniques because the cost estimates are quite high - on the order of $10^5 N$ operations for a reduction in the residual by a factor of 0.01 [16].

Multigrid methods as we consider them can be said to have taken form in the mid 1970's, particularly through the work of Brandt [14]. In the intervening years, many different variations and implementations have appeared in the literature. Here, we present details of four significant flavors of multigrid. First, we introduce the classical, geometric multigrid method in Section 3.1. This lays the foundation for the particular variants that become important in the remaining chapters of this thesis. In Section 3.2, we introduce the black box multigrid method, or BoxMG, introduced by Alcouffe et. al. [2] and later improved by Dendy [34, 35, 36, 37]. While BoxMG has proven very successful for structured domains with logically-structured grids, algebraic multigrid (AMG) [23, 72, 73] is often the method of choice for irregular grids. In Section 3.3, we discuss the details of AMG, particularly to motivate the work that appears in Section 5.2. Finally, we review the smoothed aggregation (SA) method [78, 79, 80], an algebraic technique quite distinct from the classical AMG method, which is extended in Section 5.4.

3.1 Multigrid

Geometric multigrid methods were first introduced by Brandt in 1973, with the multi–level adaptive technique or MLAT [14]. While the presentation here differs from the early descriptions of multigrid, the essential form of this basic method has not changed significantly since that time. This is, perhaps, largely due to the simplicity of the method and of the ideas that are central to it.

It has long been observed that, for many linear systems, convergence of stationary iterative methods is slow (see, for example, Southwell [75]). Multigrid methods take advantage of the fact that the slowness of these relaxation algorithms is due to a certain structure of the slow-to-converge error. Consider, for example, performing the weighted Jacobi iteration (with weight 0.8) on the 2D finite-difference Laplacian on a 33×33 grid, as shown in Figure 3.1. This behavior is typical of many stationary iterations applied to elliptic operators, that oscillatory error is reduced quite quickly, while the slow overall convergence is seen in smooth modes. While continuum elliptic operators have immediate global propagation of information, these local processes result in finite propagation speeds on the discrete grid.

Such disparity between the convergence of smooth and oscillatory errors cannot lead to an efficient algorithm. In [15], Brandt claims

> When much computational effort is invested for little real (physical) effect - something must be wrong. It is wasteful to continue slowly convergent relaxations; or to solve evolution problems with tiny time-steps imposed only by requirements of numerical stability; or to have mesh intervals far smaller than any local scale of the solution; etc. In each such case a much more efficient way does exist.

Multigrid methods overcome this stalling behavior through the use of complementary processes. We know that relaxation quickly eliminates oscillatory error, and thus we look for a process that efficiently eliminates smooth error. A key to optimal performance is that the cost of any global computation is (at least) proportional to the

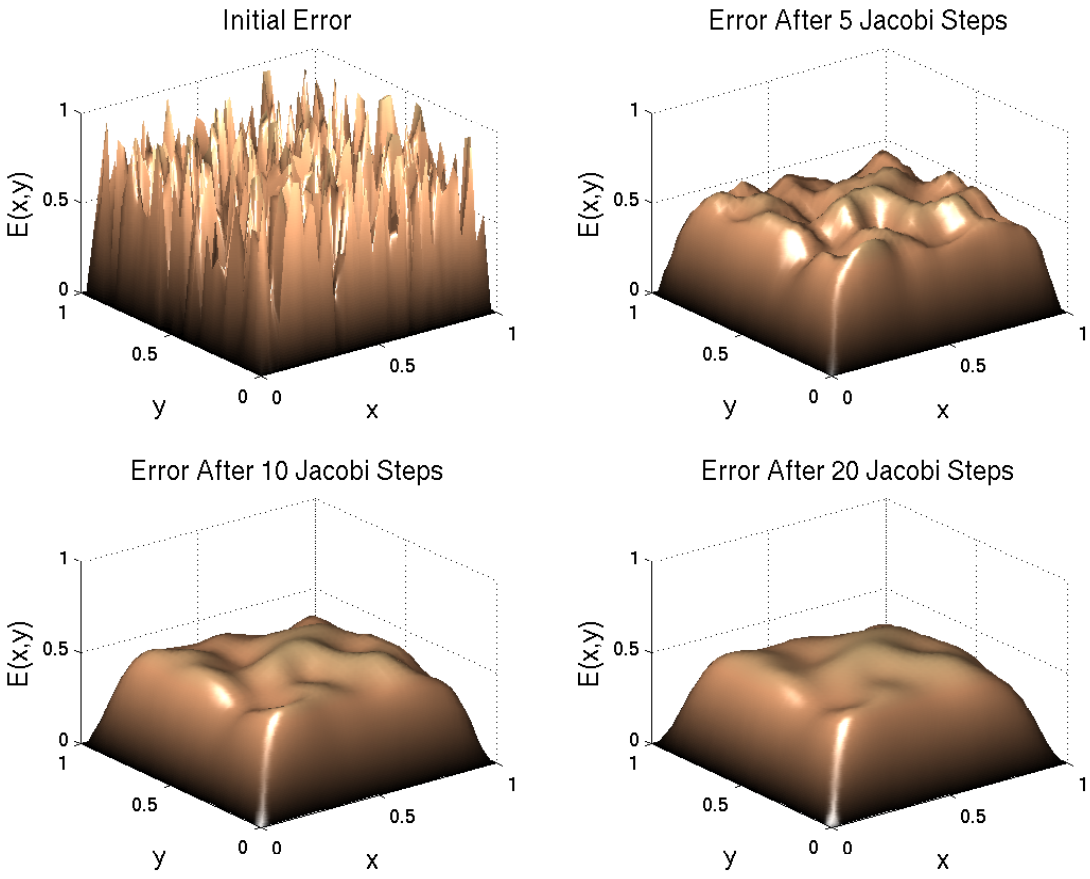

Figure 3.1: Smoothing of Random Error by the Weighted Jacobi Iteration

grid size. That is, if we could represent the error in an approximate solution on a coarser grid, we can attempt to resolve it there where global computation is cheaper.

A difficulty then lies in the fact that we would like to have a process through which we could represent the error in our solution on a coarser grid, but it is exactly this quantity we do not know. This is resolved by considering the only indication of the error in the algebraic equation, $A\mathbf{x} = \mathbf{b}$, that is available, the residual. Given an approximation, $\tilde{\mathbf{x}}$, to the exact solution, $\mathbf{x} = A^{-1}\mathbf{b}$, the residual is defined as

$$\mathbf{r} \equiv \mathbf{b} - A\tilde{\mathbf{x}} = A(\mathbf{x} - \tilde{\mathbf{x}}).$$

So, although the error is unknown, it can be represented through the residual. The error, $\mathbf{e}$, in the solution of $A\mathbf{x} = \mathbf{b}$ solves the residual equation, $A\mathbf{e} = \mathbf{r}$.

These two realizations are enough to recognize the basics of a two-grid solution method. First, relax on the fine grid to eliminate oscillatory error components and restrict the residual to a coarser grid. On that grid, solve the coarse-grid residual equation, $A_c\mathbf{e}_c = \mathbf{r}_c$, for the coarse-grid error, $\mathbf{e}_c$, and then interpolate this correction to the fine grid. Finally, it may be necessary to relax again on the corrected fine grid approximation, to ensure that some oscillatory error was not introduced through the coarse-grid correction step.

Such an algorithm is more efficient than a direct method applied to the fine grid if there is a sufficient reduction in grid size for the coarse-grid problem. The cost of a single sweep of fine-grid relaxation is typically $O(N)$, and only a fixed, small number of these sweeps are usually needed. If the reduction in the number of degrees of freedom is by a multiplicative factor, say $\frac{1}{c}$, then the reduction in the cost of a Cholesky factorization would be by a factor of $\frac{1}{c^3}$. Even for small c, this results in a significant decrease in cost, and for many problems it is reasonable to have $c = 2^d$, where d is the dimension in which the original PDE was posed.

The true efficiency in a multigrid procedure, however, lies in further elimination of

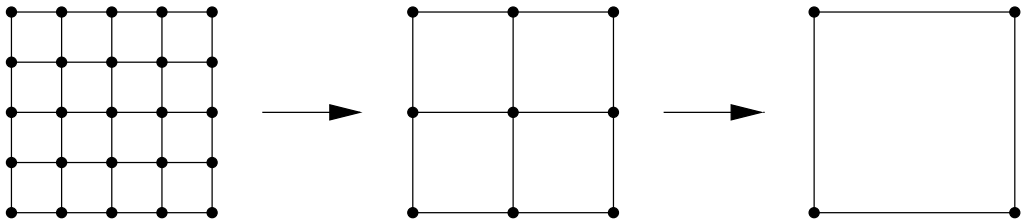

Figure 3.2: Standard Geometric Coarsening of a Tensor-Product Grid

work, pushing the factorization and direct solution down to an arbitrarily small system, usually of size independent of N. This is possible only if we restrict the residual in such a way that the coarse-level error equation has similar character as the original, fine-level equations. In the setting of a geometrically regular grid and an isotropic operator, such as the Laplacian, this is readily accomplished.

Consider the 2D Laplacian discretized on an $n \times n$ grid, for $n = 2^\ell + 1$. Relaxing several times with a simple pointwise smoother such as damped Jacobi or Gauss-Seidel leaves error that is geometrically smooth (as in Figure 3.1). With this smoothness, the remaining error can be accurately represented with fewer degrees of freedom (DOFs). Omitting every other grid point in each direction, such as is depicted in Figure 3.2, significantly reduces the total number of DOFs while retaining a dense enough grid so as not to lose accuracy on the smooth vectors. Error thus coarsened still satisfies the residual equation, although now the relevant operator is the $\left(2^{\ell-1}+1\right) \times \left(2^{\ell-1}+1\right)$ grid Laplacian.

On this grid, we can repeat the relaxation process to eliminate components of the error which are oscillatory relative to the coarse mesh size, then repeat the coarsening process. Proceeding in this manner until a coarsest grid, of small size (independent of n or ℓ), is reached, we can solve exactly for the remaining error using a simple direct solver such as Cholesky. These corrections must then be interpolated upward to the

relevant grids. Since the error on the finer grid is always smooth (relative to that grid), a simple interpolation of the correction from the coarser grid is sufficient to get the accurate representation that is sought. For this reason, a bilinear interpolant is used to represent the correction.

The method is most easily specified in a recursive manner. As such, we number grids and operators from 1, indicating the finest level (that is, the level on which the problem was posed), to $L \leq \ell$, the coarsest level. Assuming a $\left(2^\ell + 1\right) \times \left(2^\ell + 1\right)$ fine grid, reduced by a factor of approximately 2 each time (a $\left(2^k + 1\right) \times \left(2^k + 1\right)$ grid is reduced to a $\left(2^{k-1} + 1\right) \times \left(2^{k-1} + 1\right)$ grid as in Figure 3.2), this means the coarsest grid can be as small as 2×2 or 3×3, where inversion of the discrete Laplacian is easily accomplished. We denote a vector on grid k as $\mathbf{x}^{(k)}$, a matrix operator on grid k as $A^{(k)}$, and the intergrid transfer operators, I_{k+1}^k (interpolation) and I_k^{k+1} (restriction). Thus, we give the following, recursive definition of the multigrid algorithm.

Algorithm 1 ($\mathbf{x}_{\mathbf{MG}}^{(k)} = \mathbf{MG}_k(\mathbf{x}_0^{(k)},\mathbf{b}^{(k)},\nu_1,\nu_2,\mu)$).

(1) Relax ν_1 times on $A^{(k)}\mathbf{x}^{(k)} = \mathbf{b}^{(k)}$, with $\mathbf{x}_0^{(k)}$ as the initial guess, to produce the approximation $\mathbf{x}_1^{(k)}$.

(2) Form the residual, $\mathbf{r}^{(k)} = \mathbf{b}^{(k)} - A^{(k)}\mathbf{x}_1^{(k)}$.

(3) Restrict the residual, $\mathbf{b}^{(k+1)} = I_k^{k+1}\mathbf{r}^{(k)}$.

(4) If $k + 1 \neq L$, call MG_{k+1}, μ times, replacing the initial guess with the previous result each time:

$$\mathbf{x}_0^{(k+1)} = \mathbf{0}$$

for $j = 1$ to μ,

$$\mathbf{x}_j^{(k+1)} = \mathrm{MG}_{k+1}(\mathbf{x}_{j-1}^{(k+1)}, \mathbf{b}^{(k+1)}, \nu_1, \nu_2, \mu).$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}_\mu^{(k+1)}.$$

If $k + 1 = L$, solve the level L system directly to obtain $\mathbf{x}^{(L)}$.

(5) Interpolate and add the correction, $\mathbf{x}_2^{(k)} = \mathbf{x}_1^{(k)} + I_{k+1}^k \mathbf{x}^{(k+1)}$.

(6) Relax ν_2 times on $A^{(k)}\mathbf{x}^{(k)} = \mathbf{b}^{(k)}$, with $\mathbf{x}_2^{(k)}$ as the initial guess, to produce the approximation $\mathbf{x}_{\mathrm{MG}}^{(k)}$.

In this definition, we implicitly use the definitions of the coarse level matrices, $A^{(k)}$, and intergrid transfer operators, I_{k+1}^k and I_k^{k+1}. Due to the geometric context, we may consider defining these operators based on the coarse-grid geometry. The standard geometric MG definitions, as discussed above, give $A^{(k)}$ as the Laplacian discretized on grid k, I_{k+1}^k as bilinear interpolation (in 2D), and I_k^{k+1} as injection of values. These choices are also possible in 1D and 3D, with linear and trilinear interpolation, respectively.

An important question to answer is, of course, how the multigrid cycle converges. While for large ν_1, ν_2, and μ the cycle might be viewed as a direct solver, it is much more practical to consider small ν_1, ν_2, and μ and analyze it as an iterative method or preconditioner. In fact, it is quite common to consider only the special cases of $\mu = 1$ and $\mu = 2$. When $\mu = 1$, the cycle takes the form of a traverse downward to the coarsest grid, followed by a return directly upward to the finest grid. For this reason, a multigrid cycle with $\mu = 1$ is often referred to as a V-cycle, from the usual depiction as in Figure 3.3. When $\mu = 2$, the cycle takes the form of a downward traverse, followed by a single step up, then down, then up two levels, and so on. When depicted graphically, as in Figure 3.4, this appears in the shape of a W, thus giving the name of a W-cycle.

Brandt [16, 19] demonstrates through local mode analysis the convergence of these cycles for the Laplacian and many other problems. This analysis considers the action of the components of the multigrid cycle on the standard Fourier basis. Through it, quite sharp bounds on the reduction of the Fourier modes can be obtained, translating into sharp bounds on the asymptotic performance of the multigrid method.

Another form of analysis, due to Bramble et al. [9, 11, 12], considers the behavior

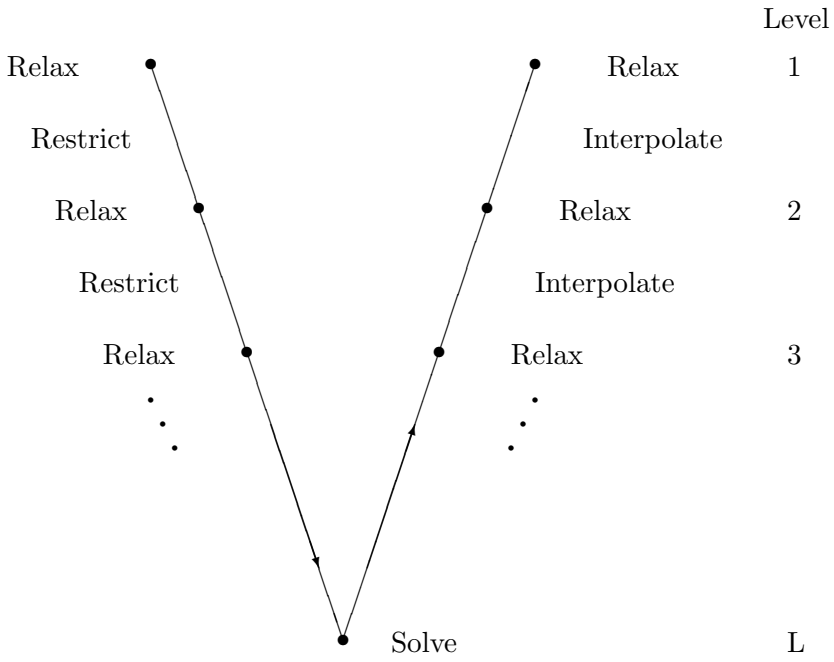

Figure 3.3: The Multigrid V-cycle

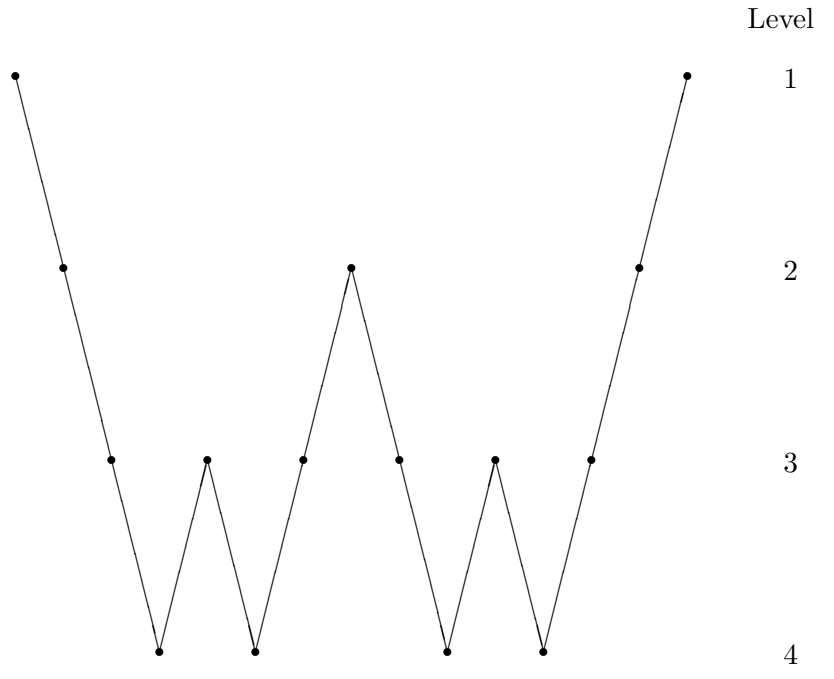

Figure 3.4: The Multigrid W-cycle

of the multigrid V-cycle as a stationary linear iteration. Writing the multigrid iteration in this way, they show that the A-norm of the error propagation matrix, $I - BA$, is bounded by a constant less than one. While this guarantees convergence of the multigrid method, the bound is not sharp. In particular, the bound depends on a number of properties of relaxation and of the matrix, A, which themselves are not easily made sharp, and the combination is then certainly not so.

Regardless of the analysis, one arrives at a bound of the form

$$\|\mathbf{e}^{\text{new}}\| \leq \rho \|\mathbf{e}^{\text{old}}\|.$$

This says that a reduction of the norm of the error by any fixed factor requires only a constant number of multigrid cycles. From local mode analysis, it can often be shown that, for a given PDE, ρ is independent of N. Thus, no matter what resolution the finest grid has, only a fixed number of multigrid cycles are required to reduce the error by a fixed factor.

Perhaps a more appropriate bound, however, is the effort required to reduce the error by a factor proportional to the error in the discretization of the PDE. If we discretize the symmetric PDE, $Lu = f$, appropriately, we can often realize the bound $\|u - \tilde{u}\| \leq \frac{c}{n^2}\|u\|$, for some constant c, where u denotes the solution of the continuous problem and $\tilde{u}$ the solution of the discretized problem on an $n \times n$ grid. In this case, asking that the error in the discrete solution be of the same order as the error between the exact discrete and continuous solutions means reducing the original, $O(1)$ error by a factor of $\frac{1}{n^2}$. This reduction requires $O(\log n)$ steps.

To consider the total cost of multigrid, we must also analyze the cost of each iteration. Considering the case where A is sparse, with a fixed number of non-zeros per row regardless of N, a matrix-vector multiply costs $O(N)$ operations. If this sparsity is preserved on all coarse grids, each cycle of the multigrid algorithm on a grid k problem of size $N^{(k)}$, MG_k, costs $O((\nu_1 + \nu_2)N^{(k)} + \mu \times \text{cost}(\text{MG}_{k+1}))$ operations, with MG_L

costing a constant number of operations. Undoing the recursion, we see that for $\mu = 1$, the total cost of a cycle from the finest grid is $O((\nu_1 + \nu_2)N)$, providing that coarsening is accomplished rapidly enough. Thus, depending on the criterion for reducing error, we can consider multigrid to be either an $O(N)$ or $O(N \log N)$ solver.

In fact, one can also consider multigrid in a nested iteration context. By continuation in N, we can have an initial guess on level k whose error is only a constant factor times the discretization error for the level k problem. An analysis of this situation, where only a fixed number of multigrid (V-)cycles is performed on each level shows that the total cost of nested-iteration multigrid is, in fact, $O(N)$ to reduce the error in the discrete solution to the size of discretization error on a particular level. We do not explicitly consider such iterations here, but it is important to realize that this is usually possible.

While geometric multigrid can be quite effective, its performance is also quite problem dependent, and, in many situations, this performance breaks down. The bilinear interpolation used is appropriate in the case when we expect small changes in the fine-grid solution, $\mathbf{x}$, between coarse and fine grids. For some problems, particularly those with discontinuous coefficients, this is not necessarily the case. While the continuum solution, p, is continuous for the discontinuous-coefficient diffusion operator, $-\nabla \cdot \mathcal{K}\nabla$, the gradient, ∇p, is not. The normal flux, $(\mathcal{K}\nabla p) \cdot \mathbf{n}$, however, is continuous across any interface with unit normal vector $\mathbf{n}$. Bilinear interpolation preserves continuity of ∇p, which is, in general, not continuous across an interface between values of $\mathcal{K}$. Thus, designing a multigrid method for the discontinuous-coefficient diffusion problem requires a different definition for interpolation near the discontinuities in the diffusion coefficients.

A similar breakdown occurs in the case of strong anisotropy or convection. In these cases, interpolation that ignores the directionality of the operator cannot properly approximate error components whose smoothness shares this directionality. In the

extreme anisotropic case, the operator is essentially one dimensional and interpolation should follow suit. Similarly, in the case of a convection-dominated flow, information should flow downstream and thus interpolation should be biased toward the flow. A naive interpolation based solely on geometric location of the nodes cannot take these situations into account, and, thus, the definition of interpolation for an efficient multigrid method for these problems must be reconsidered in the presence of such complexities.

Accurate interpolation is also difficult when the geometry is not regular. Simple examples of situations that cause trouble for multigrid with predefined interpolation schemes include non-regular mesh spacing in a tensor-product grid and non-regularly shaped elements. In both of these cases, a simple bilinear interpolation based on the assumption of equal mesh spacing can significantly over- or under-estimate the appropriate correction factor. In case of non-regular spacing, a solution can easily be seen in allowing the interpolation to depend on the grid spacing, if it is known. For a non-regular grid, however, achieving bilinear interpolation may require computation of a unique interpolation stencil for each fine-grid node.

Instead of developing a new multigrid code for each unique complication, we consider the approach that many of these situations can be handled by a more intelligent algorithm. Such an algorithm was first introduced in 1981, by Alcouffe et al. [2]. It is this algorithm that we discuss next, followed by further generalizations of the geometric multigrid ideas that allow more extreme complications to be addressed automatically.

3.2 The Black Box Multigrid Method

Early multigrid studies focused primarily on fast algorithms for the isotropic Poisson problem, $-\Delta p = f$. However, algorithms developed for this problem [16] were quickly recognized to lack robustness. One particular variation of significant interest is the variable-coefficient Poisson problem, $-\nabla \cdot \mathcal{K}\nabla p = f$. Here, we consider the coefficient, $\mathcal{K}(\mathbf{x})$, to vary significantly and discontinuously across the domain. $\mathcal{K}$ is

allowed to be tensor valued, although for most applications it is constrained to be symmetric and positive definite. Alcouffe et al. [2] first considered a multigrid algorithm for this problem, which Dendy later extended in a series of papers also addressing non-symmetry, systems of PDEs, periodicity, and singularity [34, 35, 36, 37].

The result of this work is the black box multigrid algorithm, also known as BoxMG. The name reflects the idea that the method can be employed as a black-box technique: the user need only provide the matrix, A, and right-hand side, $\mathbf{b}$, for solution of $A\mathbf{x} = \mathbf{b}$. The overall form of the algorithm is the same as that in the geometric context, as outlined in Section 3.1. The differences between BoxMG and geometric multigrid (and, indeed, between almost any two multigrid methods) are in the choice of the relaxation method, coarse-grid selection procedure, intergrid transfer operators, and coarse-grid operator definition.

The choice of a relaxation scheme is complicated by the assumption of jumps in $\mathcal{K}$. For a region where $\mathcal{K}$ is constant or varies slowly, pointwise relaxation is quick to smooth the error. However, when $\mathcal{K}$ changes significantly across an interface, point-wise relaxation on a given level may be slow to propagate changes in $\mathbf{x}^{(k)}$ away from that interface. This is acceptable in the view that the coarsest-level system, $A^{(L)}\mathbf{x}^{(L)} = \mathbf{b}^{(L)}$, is solved exactly, resolving the interfaces on a coarse mesh, and under the assumption that this solution suitably represents the interface when interpolated to the finer grids. A similar slowing of smoothing is seen in the case of anisotropy, when $\mathcal{K}$ is tensor valued and $\mathcal{K}_{1,1} \ll \mathcal{K}_{2,2}$ in 2D (or vice-versa). In this case, the reduction factor of the oscillatory modes (those in the upper half of the matrix spectrum) using pointwise Gauss-Seidel relaxation approaches 1 as the anisotropy becomes stronger [16]. A better strategy in this case is the use of line relaxation in the direction of the larger coefficient value. In the anticipation of added robustness, it is then often useful to choose line relaxation (line Gauss-Seidel) as the smoother in BoxMG. To avoid preselecting the direction of any anisotropy in the PDE, a common practice is to perform an iteration of Gauss-

Seidel on lines in x, followed by one on lines in y, as each relaxation sweep. In three dimensions, this extends to successive solves along xy-, xz-, and yz-planes. Each of these 2D solves (for a plane Gauss-Seidel sweep) can be accomplished by an application of a two-dimensional BoxMG cycle, thus preserving the multigrid optimality as relaxation remains an $O(N)$ process.

BoxMG assumes a fine grid with logically rectangular connections. In two dimensions, this implies that the discrete operator is structured based on a mesh around quadrilateral elements. The non-zero structure of the matrix is limited to the 8 nearest-neighbor connections as in a bilinear finite-element stencil. In three dimensions, this concept is generalized to 27-point stencils, structured as in a regular division of $\Omega \subset \mathbb{R}^3$ into cubes. As such, we can consider the operator to be posed on a regular, tensor-product mesh in two or three dimensions. Thus, we refer to both coordinate directions and cardinal directions when discussing the discretized operator, acknowledging that the actual geometry of the discretization may not exactly correspond with these directions.

Using this structure, coarsening can easily be done geometrically, as in the case of a regular grid structure. While the physical grid points may not lie in a regular array, the assumption of a logically rectangular structure allows us to treat the matrix as if they did. In the bilinear interpolation case, performance would suffer if the grid had a significantly irregular structure. So, the disadvantage of this approach is that any stretching, compression, and irregularity in the grid must be accounted for in the interpolation operator. This is not a new inconvenience, however, as BoxMG is already based on the premise that interpolation is not predetermined by geometry.

The major innovation in BoxMG is that the coefficients of interpolation, I_{k+1}^k, are chosen to depend on the fine-grid operator, $A^{(k)}$. Complications, such as jumps in the diffusion coefficient, $\mathcal{K}$, or unequal grid spacings, are reflected in the matrix coefficients, and thus it should be possible to choose an operator-dependent (or operator-induced) interpolation in a manner to accommodate these situations. In fact, Dendy shows that

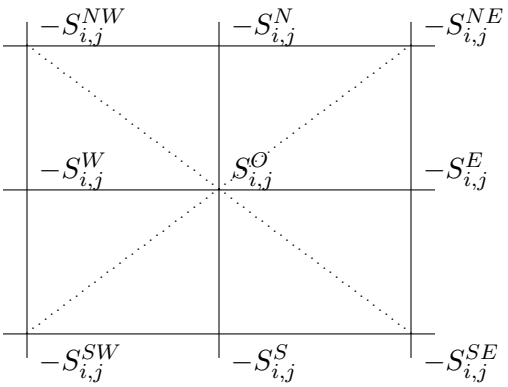

Figure 3.5: Compass-Based Notation for the Stencil at Node (i, j)

it is possible to do so in the case of a symmetric, positive-definite operator, $A^{(1)}$ [34], and also in more general situations [35, 36, 37].

The choice of interpolation in [34] is based on the desire to approximately preserve the continuity of the normal flux, $(\mathcal{K}\nabla p) \cdot \mathbf{n}$, across an internal boundary (grid line). Moulton et al. [67] shows that this is, in fact, exactly the case for a 9-point bilinear finite element discretization. For this reason, the BoxMG algorithm has proven to be quite useful for problems with large jumps in $\mathcal{K}$. We use BoxMG in this context in Chapter 4.

In two dimensions, we can consider a row of the matrix in stencil notation, writing the stencil using compass-based notation, as shown in Figure 3.5. It is important, however, to exploit the symmetry of the operator in order to achieve an efficient implementation of the method, as this can reduce the needed storage by a factor of approximately 0.5. A second representation of the matrix in terms of cells is also possible, where each entry in the stencil is associated with an edge of the matrix graph and, thus, an element that it crosses or is adjacent to. Reducing storage in this representation confuses the notation, which is shown in Figure 3.6, where stencil entries are numbered according to the node in the Northeast corner of the element they are associated with. Note that, in both cases, the off-diagonal entries are negated, so that (in general) the values ref-

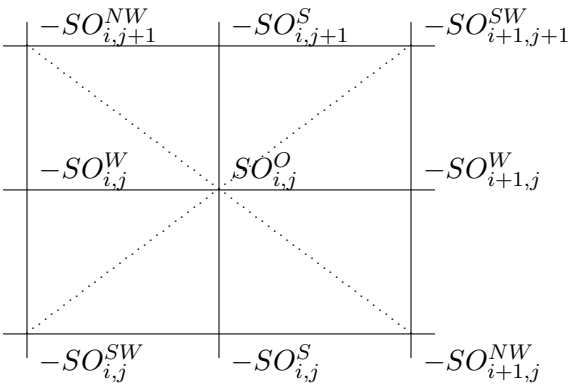

Figure 3.6: Element-Based Notation for the Stencil at Node (i, j)

erenced are positive. We derive the interpolation operator based on the compass-based notation, although the element-based notation is typically used in the referenced papers.

Considering interpolation of a grid function $\mathbf{x}^{(k+1)}$ from grid $k+1$ to grid k, we examine the interpolation to a node (i, j) on the fine grid, in four distinct situations. Denote a coarse-grid node, (i_C, j_C), to be the closest node to (i, j), referring to the particular node on grid $k+1$ that either corresponds to the fine-grid node, lies directly to its west, lies directly south, or lies south-west of the fine-grid node. As always, interpolation is needed only for functions that are smooth on the fine-grid. This smoothness is represented by a small residual, and thus an error interpolated to grid k satisfies an equation with small right side. Thus, when considering an equation to represent this error, we present it with a zero right side. It is possible to consider the non-zero residual in this context, which may be significant in certain situations [34].

There are exactly four possibilities for interpolation. The simplest is when (i, j) corresponds to a coarse-grid node. In this case, the value of $\mathbf{x}$ is preserved:

$$(I^k_{k+1}\mathbf{x}^{(k+1)})_{(i,j)} = \mathbf{x}^{(k+1)}_{(i_C,j_C)}.$$

The case of the fine-grid node lying between two coarse-grid nodes occurs in two variations, along horizontal lines and along vertical lines. In these situations, the interpolation formula is arrived at by collapsing the grid k operator stencil. When the

fine-grid node lies on a horizontal line between coarse-grid nodes (i_C, j_C) and $(i_C{+}1, j_C)$, averaging the stencil for (i, j) vertically to arrive at an equation involving only these three nodes' values gives

$$\left(-S_{i,j}^{NW} - S_{i,j}^{W} - S_{i,j}^{SW}\right)\mathbf{x}_{i_C,j_C}^{(k+1)} + \left(-S_{i,j}^{N} + S_{i,j}^{O} - S_{i,j}^{S}\right)\mathbf{x}_{i,j}^{(k)} + \left(-S_{i,j}^{NE} - S_{i,j}^{E} - S_{i,j}^{SE}\right)\mathbf{x}_{i_C+1,j_C}^{(k+1)} = 0.$$

This equation can be rearranged to express $\mathbf{x}_{i,j}^{(k)}$ in terms of the coarse-grid values, arriving at the definition of interpolation as

$$\left(I_{k+1}^{k}\mathbf{x}^{(k+1)}\right)_{i,j} = \frac{\left(S_{i,j}^{NW} + S_{i,j}^{W} + S_{i,j}^{SW}\right)\mathbf{x}_{i_C,j_C}^{(k+1)} + \left(S_{i,j}^{NE} + S_{i,j}^{E} + S_{i,j}^{SE}\right)\mathbf{x}_{i_C+1,j_C}^{(k+1)}}{-S_{i,j}^{N} + S_{i,j}^{O} - S_{i,j}^{S}}.$$

Similarly, for a fine-grid node lying on the vertical line between coarse-grid nodes (i_C, j_C) and $(i_C, j_C{+}1)$, averaging the stencil in the horizontal direction derives the interpolation formula to be

$$\left(I_{k+1}^{k}\mathbf{x}^{(k+1)}\right)_{i,j} = \frac{\left(S_{i,j}^{SE} + S_{i,j}^{S} + S_{i,j}^{SW}\right)\mathbf{x}_{i_C,j_C}^{(k+1)} + \left(S_{i,j}^{NE} + S_{i,j}^{N} + S_{i,j}^{NW}\right)\mathbf{x}_{i_C,j_C+1}^{(k+1)}}{-S_{i,j}^{E} + S_{i,j}^{O} - S_{i,j}^{W}}.$$

The fourth and final case is when a fine-grid node is embedded in the center of a coarse-grid element. In this orientation, the analogous situation would be to average the stencil based on the four nearest coarse-grid neighbors. It is, however, much simpler to consider determining the value based on the homogeneous fine-grid matrix equation. That is, consider interpolation to the four fine-grid neighbors of this node first, then use these values to interpolate to node (i, j). This results in the interpolation formula

$$\begin{aligned}\left(I_{k+1}^{k}\mathbf{x}^{(k+1)}\right)_{i,j} = \Big(&S_{i,j}^{SW}\mathbf{x}_{i_C,j_C}^{(k+1)} + S_{i,j}^{S}(I_{k+1}^{k}\mathbf{x}^{(k+1)})_{i,j-1} + S_{i,j}^{SE}\mathbf{x}_{i_C+1,j_C}^{(k+1)} \\ &+ S_{i,j}^{W}(I_{k+1}^{k}\mathbf{x}^{(k+1)})_{i-1,j} + S_{i,j}^{E}(I_{k+1}^{k}\mathbf{x}^{(k+1)})_{i+1,j} \\ &+ S_{i,j}^{NW}\mathbf{x}_{i_C,j_C+1}^{(k+1)} + S_{i,j}^{N}(I_{k+1}^{k}\mathbf{x}^{(k+1)})_{i,j+1} + S_{i,j}^{NE}\mathbf{x}_{i_C+1,j_C+1}^{(k+1)}\Big) \Big/ \left(S_{i,j}^{O}\right),\end{aligned}$$

which, when the formulae for $I_{k+1}^{k}\mathbf{x}^{(k)}$ above are substituted, explicitly defines interpolation to these nodes based on their four nearest coarse-grid neighbors.

When considering the Laplacian, we chose the coarse-grid operator simply as the coarse-scale discretization of the continuum operator. The fine-scale fluctuations in $\mathcal{K}$ may cause difficulty in directly discretizing the variable-coefficient PDE on a coarse grid, however. Instead, considering the original PDE to be discretized in a variational context (such as a finite-element discretization), we can make a very different choice.

Posing the symmetric matrix equation, $A\mathbf{x} = \mathbf{b}$, as a minimization problem, we define the functional, $F(\mathbf{y}) = \frac{1}{2}\langle A\mathbf{y}, \mathbf{y}\rangle - \langle \mathbf{b}, \mathbf{y}\rangle$, so that $\mathbf{x} = \underset{\mathbf{y}}{\operatorname{argmin}} F(\mathbf{y})$. A coarse-grid correction is then chosen to reduce $F(\mathbf{y})$ optimally, for a given approximation, $\mathbf{y}$. That is, we look to minimize $F(\mathbf{y} + I_c^f \mathbf{w})$ over all choices of $\mathbf{w}$ on the coarse grid (here we use the generic interpolation operator, I_c^f, to denote interpolation from some coarse grid to some fine grid). Doing so, the optimal choice of $\mathbf{w}$ then satisfies the equation

$$\left(\left(I_c^f\right)^T A \left(I_c^f\right)\right) \mathbf{w} = \left(I_c^f\right)^T (\mathbf{b} - A\mathbf{y}) .$$

Notice that the right side of this equation has the form of a restriction of the current fine-grid residual, $\mathbf{b} - A\mathbf{y}$, with the restriction operator chosen as the transpose of interpolation. Also, notice that the coarse-grid operator is a symmetric reduction of the fine-grid operator. This minimization viewpoint is applicable for any symmetric fine-grid operator, given the functional formulation. Thus, the choices of restriction as the transpose of interpolation and the RAP (restriction-A-prolongation) form of the coarse-grid operator are often called a variational formulation. The RAP form of A_c with $R = P^T$ is also referred to as a Galerkin formulation, borrowing from the finite-element theory to which it is quite similar.

The simplicity of the BoxMG algorithm has allowed it to be extended to a number of additional situations. For non-symmetric problems, interpolation is often based on the symmetric part of the operator (which, in the case of reaction-convection-diffusion models, is dominated by the diffusion terms), whereas restriction is chosen by considering the transpose of the fine-grid operator [35]. Systems can easily be handled by

generalizing the definitions above so that the matrix entries are themselves block matrices connecting co-located degrees of freedom [36]. In Chapter 4, we make use of this algorithm for singular problems, with periodic or full-Neumann boundary conditions. In [37], it is shown that this complication can be addressed through a simple adjustment to the solution of the coarsest-scale problem, for example, making it non-singular by adding a condition on the mean of the coarse-grid solution.

While the BoxMG algorithm has proven very useful for logically structured grids, many discretizations lack this regularity. The main ideas of the method could be extended based on a known geometry of a discretization, but it is not practical to require a new implementation for each new geometry. In fact, the geometry may not be available to the solver or may not even be known. For this reason, Brandt et al. [23] introduced the algebraic multigrid algorithm, which uses the same idea of operator-induced interpolation, but allows for more general grid geometries to be addressed automatically.

3.3 Algebraic Multigrid Methods

Algebraic multigrid (AMG) methods were first introduced by Brandt et al. [22] for the solution of geodetic problems. This early motivation called for a departure from geometric multigrid methods because of the inherent discreteness of the problem and the lack of useful information about its geometry (as grid locations are the unknowns in geodetic computations). The usefulness of AMG extends far beyond this particular application, however, as the algorithm has proved effective for problems with discontinuous coefficients, such as those for which BoxMG was designed, and also for problems with irregular or unknown geometry.

While the term algebraic multigrid now covers many different algorithms, a significant number of these are variations on the classical algorithm, which was first introduced in the early 1980's [22, 23] and most often linked to the version implemented by Ruge and Stüben [72, 73]. Here, we present the Ruge-Stüben algorithm, which we refer

to simply as AMG. The terms AMG and algebraic multigrid have come to mean both the Ruge-Stüben algorithm and the entire class of multigrid methods that are algebraic in approach. We follow this convention as the intent is usually clear by context. In the next section, we detail the smoothed aggregation multigrid method, an algebraic multigrid method that differs significantly from that presented by Ruge and Stüben. In Chapter 5, we discuss some generalizations of both of these algorithms.

As the AMG algorithm uses only information from the matrix, A, a description of it requires certain notation to account for the lack of geometric intuition. Each row, i, in the matrix is identified with an unknown (as always) and a node in the matrix graph, numbered i. The connections in the matrix then represent weights on the edges of the graph (if $a_{ij} = 0$, there is no edge between nodes i and j). Node (row) i is said to strongly depend on node (row) j if $-a_{ij} \geq \theta \max_{k \neq i}\{-a_{ik}\}$. Likewise, node i strongly influences node j if $-a_{ji} \geq \theta \max_{k \neq j}\{-a_{jk}\}$. Here, θ is a predetermined threshold value, $0 \leq \theta \leq 1$, often chosen to be 0.25. The negative signs in the equation reflect the origin of AMG in finite-difference and finite-element applications, where A is often an M-matrix (and so has negative off-diagonals). An alternate definition of strong influence and dependence uses $|a_{ij}|$ instead. Notice also that these definitions are adjoints: if i strongly depends on j, then j strongly influences i and vice-versa.

The major differences between AMG and geometric multigrid methods are in the choice of the coarse grids and intergrid transfer operators. While geometric multigrid makes use of additional information, AMG makes these choices based solely on the matrix, A, in the equation, $A\mathbf{x} = \mathbf{b}$. This is possible as relaxation processes are well understood for a large class of matrices. In particular, it is known [18] that point relaxation schemes slow down when the residual in the equation becomes small when compared with the error. Since relaxation and coarse-grid correction must be complementary, we seek to define interpolation so that all such errors are in its range. We also look to define coarse-grid operators so that this smooth, fine-level error can be easily

and accurately resolved on the coarse level.

The key to the efficiency of any multigrid process lies in the complementarity of the relaxation and coarse-grid correction processes. For this reason, any discussion of the classical AMG algorithm begins with the defining property of algebraic smoothness that the residual is, on average, small after a few sweeps of relaxation: $(A\mathbf{e})_i \approx 0$ for each point i. Considering pointwise relaxation, such as Gauss-Seidel, and a symmetric positive-definite operator, such errors are typically associated with the small eigenvalues of the operator, and this is often what is meant when discussing algebraically smooth errors. Also central is the assumed property that such errors vary slowly along strong connections in the matrix. This additional assumption is generally applicable for discretizations of scalar elliptic PDEs, for which AMG was originally designed. Other problems, such as systems of PDEs, may require other assumptions on the nature of algebraically smooth errors. In any case, awareness of this assumption is important in analyzing the performance of AMG, particularly when it is poor.

To construct interpolation to approximate a general algebraically smooth error, $\mathbf{e}$, we use the premise of a small residual (that, for example, $\|A\mathbf{e}\| \ll \|A\| \cdot \|\mathbf{e}\|_A$) to conclude that, for a given row, i, $(A\mathbf{e})_i \approx 0$ or

$$a_{ii}e_i \approx -\sum_{j \neq i} a_{ij}e_j. \tag{3.1}$$

Now, suppose that a coarse set of points that forms a subset of the fine DOFs has been chosen (one strategy for which is discussed below). Then, the fine-level DOFs can be represented as $\{1, 2, \ldots, N\} = C \cup F$, where C is the set of coarse-level points and F is the set of remaining fine-level points (so $C \cap F = \emptyset$). Since A is sparse, we introduce "neighborhood" notation: $N_i = \{j : a_{ij} \neq 0\}$, $C_i = C \cap N_i$, and $F_i = F \cap N_i$. Equation (3.1) can then be rewritten as

$$a_{ii}e_i \approx -\sum_{j \in C_i} a_{ij}e_j - \sum_{k \in F_i} a_{ik}e_k. \tag{3.2}$$

Were the last sum not present in Equation (3.2), this expression could be used to define interpolation because it would give the F-point value, e_i, approximately as a sum of the C_i-point values. The aim is therefore to "collapse" the connections from point i to points $k \in F_i$ onto the points $\{j \in C_i\} \cup \{i\}$. That is, we want to set a_{ik}, $k \in F_i$, to 0 while adjusting a_{ij}, for $j \in C_i$, and a_{ii} in some way to compensate for the inaccuracies this elimination introduces. The main assumption needed to collapse the stencil is that the values of algebraically smooth $\mathbf{e}$ at F_i points can be written in terms of its values at points in $C_i \cup \{i\}$:

$$e_k \approx \sum_{j \in C_i} \omega_{kj}^i e_j + \omega_{ki}^i e_i, \text{ for } k \in F_i. \tag{3.3}$$

Then, we could substitute this expression into the latter sum in Equation (3.2) to obtain an expression for e_i in terms of $e_j, j \in C_i$, which is exactly the aim. Note that Equation (3.3) is a special form of interpolation from $C_i \cup \{i\}$ to F_i. This special interpolation formula is used in reducing the stencil connections to determine the final interpolation formula, so this overall process is sometimes called "twice-removed" or "iterated" interpolation.

Now, in classical AMG algorithms, the $\{\omega_{kj}^i\}$ in Equation (3.3) are determined from $\{a_{kj}\}$ based on the additional assumption that $\mathbf{e}$ is constant along strong connections. So, first we must ask the question as to whether the connection between i and k is important. If k does not strongly influence i, then its value is not helpful in interpolation as it is not a strong interpolation point. The set of such k is denoted F_i^w, and referred to as the set of weak connections. These connections are not simply discarded; rather, their values are collapsed onto the diagonal. That is, $\omega_{kj}^i = 0$ for $j \neq i$, and $\omega_{ki}^i = 1$. Notice that this approach is, in fact, quite stable. If the classification of $k \in F_i$ misidentifies a strong influence as a weak one, it replaces the value e_k with e_i, which (because of the assumption about smooth error and strong connections) are, in fact, approximately equal.

The remaining $k \in F_i \setminus F_i^w \equiv F_i^s$ all strongly influence i. Adding the requirement that a node $k \in F_i^s$ is also strongly dependent on some $j \in C_i$, then we can hope to determine the value at point k based on the coarse-grid neighbors it has in common with i. Since the strength of k's connection to a point $j \in C_i$ is proportional to the matrix coefficient a_{kj}, the value e_k can be approximated by the average of these e_j weighted by their coefficients. That is,

$$e_k = \frac{\sum_{j \in C_i} a_{kj} e_j}{\sum_{l \in C_i} a_{kl}}.$$

An important property of this intermediate interpolation formula is that it is an average; if $\mathbf{e}$ is constant for all $j \in C_i$, then it takes the same value at k. For problems such as the Laplacian, where the smoothest mode is the constant vector (up to boundary conditions), it is quite important to ensure the accuracy of interpolation for this mode.

Combining these approximations yields an overall interpolation formula for $i \in F$: $e_i = \sum_{j \in C_i} w_{ij} e_j$, with

$$w_{ij} = -\frac{a_{ij} + \sum_{k \in F_i^s} \left(\frac{a_{ik} a_{kj}}{\sum_{l \in C_i} a_{kl}} \right)}{a_{ii} + \sum_{m \in F_i^w} a_{im}}.$$

The effectiveness of this interpolation operator is now very dependent on the quality of the coarse grid. This is apparent from the assumption that each point $k \in F_i^s$ has a strong dependence on some $j \in C_i$. In addition, we need the coarse grid to satisfy an often contradictory condition that it must be small enough that there is a real benefit to coarsening. This is typically expressed by saying that the size of C must be less than a fixed factor of N (as we saw in the geometric case, this is necessary for each cycle to have $O(N)$ cost), where we typically seek a factor of $\frac{1}{2^d}$ or smaller (when the fine-grid matrix comes from a problem in $\mathbb{R}^d$). At the same time, we do not want the coarse grid to be too small, as that would limit the correction to only a very small-dimensional

space.

These opposing views are best expressed through two heuristics that are often used to describe the AMG coarsening process:

- For each point $i \in F$, every point j that strongly influences i should either be a coarse-grid point in the set C_i or be, itself, strongly influenced by at least one point in C_i.

- The set C should be a maximal subset of $\{1, \ldots, N\}$, where no point in C strongly depends on another point in C.

The first heuristic clearly addresses the concern that a node $k \in F_i^s$ must strongly depend on something in C_i. The second heuristic leads to a sort of independence in C, attempting to guarantee that the size of C is significantly less than N. It also ensures that the coarse grid is not too small, as of all subsets, C, that have no pairs of strongly-connected nodes, we look to take the largest of them as the coarse grid.

Once we have chosen the coarse points, C, and interpolation operator, P, we must still choose a restriction operator, R (for transferring the residuals to the coarse level), and a coarse-level operator, A_c (for defining the coarse-level correction equation). Assuming that A is a symmetric positive-definite matrix, it is natural to define these operators by the Galerkin conditions (cf. [73] and the previous discussion): $R = P^T$ and $A_c = RAP$.

AMG is a generalization of classical geometric multigrid. It is an efficient solver for many problems, including those involving discretizations on stretched or irregular grids, or discretizations of many problems with anisotropic or variable coefficients. There are, however, still many problems for which AMG is not entirely effective as a black-box solver, including problems with highly anisotropic or highly variable coefficients, and those coming from the discretization of certain systems of PDEs such as Stokes or nearly incompressible linear elasticity. Simply put, the further the algebraically smooth

components of a problem are from being locally constant along strong connections, the more the performance of AMG suffers without additional changes to the algorithm.

Many variations on the Ruge-Stüben have been proposed since its first introduction. These involve either an alternate choice of interpolation, perhaps to fit different smooth components or take a different stencil, or an alternate choice of the coarse grid. Much recent work has been devoted to choosing the coarse grid, particularly for implementations in parallel, or problems that do not have the classical ellipticity upon which to rely. In Chapter 5, we consider a generalization of the interpolation definition. A new idea for choosing coarse-grid points is discussed in the remarks on future work in Chapter 6.

3.4 Smoothed Aggregation Multigrid

While many algebraic multigrid methods can be viewed as adaptations of the Ruge-Stüben algorithm, this is not the case for smoothed aggregation (SA). First introduced as a two-level algorithm for accelerating a given relaxation scheme [77], this method demonstrates significant improvement to the aggregation multigrid method. For this reason, we first consider some details of the aggregation multigrid method, followed by the extension to smoothed aggregation. In this context, we assume that A is symmetric and positive definite, coming from the discretization of an elliptic second- or fourth-order PDE.

Aggregation methods originated in economics, where similar products are considered together instead of individually. This procedure allows significant reduction in the problem size, while maintaining accurate representation of the global dynamics. If detail within a product aggregate is required, solution of a much smaller set of equations (corresponding only to products within a class) is necessary. In multigrid terminology, the coarse grid is selected as a collection of subsets of the fine grid. A node on the coarse grid is associated with several fine-grid nodes; each coarse-grid node is an aggregate of

fine-grid ones. Interpolation is typically then chosen as piecewise constant over each aggregate. Following the notation of Míka and Vaněk [64], the fine grid, $U = \{u_1, \ldots, u_N\}$, is partitioned into aggregation classes $\{\mathcal{A}_j\}_{j=1}^{N^{(2)}}$, where $N^{(2)}$ is the size of the first coarse grid, such that

(1) $\forall u_i \in U, \exists j$ such that $u_i \in \mathcal{A}_j$.

(2) $\mathcal{A}_j \cap \mathcal{A}_k = \emptyset$ if $j \neq k$.

(3) If $u_p, u_q \in \mathcal{A}_j$, then $\exists u_i \in U$ such that $u_p, u_q \in N_i$.

The restriction operator, R, is then defined as

$$R_{ij} = \begin{cases} 1 & \text{for } u_j \in \mathcal{A}_i, \\ 0 & \text{for } u_j \notin \mathcal{A}_i. \end{cases}$$

For symmetric matrices, A, the remaining operators are given variationally, with interpolation given by $P = R^T$ and the coarse-grid matrix by the Galerkin condition, $A_c = RAP$. Míka and Vaněk show that, using these operators, the two-level multigrid process is convergent [64]. They also show that a more efficient version of this method can be arrived at by amplification of the coarse-grid correction by an easily computed factor [65].

Vaněk further extended this method by noticing that improved convergence could be obtained with an intuitive modification of the interpolation operator, P [77]. In unsmoothed aggregation, columns of P correspond to step functions over the aggregates. Thus, the range of P includes only piecewise-constant vectors. While the exact null space of a differential operator is usually in this range, many algebraically smooth vectors are not well represented by their projections onto a piecewise-constant basis. In SA, the interpolation operator from unsmoothed aggregation is left multiplied by the weighted Jacobi operator, $I - \omega D^{-1} A$. One can view this process as smoothing the columns of the interpolation operator; instead of a column of P corresponding to a step function

over an aggregate, the columns of P now have smooth shape. The effect on the range of interpolation is significant.

With this modification, the performance bound of the two-level, non-smoothed aggregation method discussed in [64, 65] can be improved [77]. Choice of the weighting parameter, ω, is, of course, very important in the actual performance of the method. This importance is reflected by its appearance in the performance bound, but a good choice of ω is obscured by the other, matrix-dependent parameters that appear in this bound. The algorithm can be naturally extended to a multilevel one, as was done in [78], and similar performance bounds can be determined. The use of overcorrection, as in [65], is again important in the overall performance of the algorithm, giving, at low additional cost, an optimal reduction of error in the direction of the coarse-grid correction. How to best choose ω is still obscured by the proofs that the smoothed aggregation method converges both without and with overcorrection. While this theory provides no insight, in a series of numerical examples Vaněk chooses $\omega = 0.63$ and shows good numerical convergence rates [78].

An important extension of this algorithm, due to Vaněk et al. [79, 80], allows the application of the smoothed aggregation method to systems of PDEs and to higher-order problems. In this generalization, the fine-grid problem is assumed to have a near null space spanned by r vectors. An aggregation multigrid approach for this situation would ask for each of these r vectors to be in the range of interpolation. Accommodating this requires multiple columns of interpolation over each aggregate, which is clearly allowable.

Given r vectors that span the near null space of A, these vectors are partitioned into subvectors over each aggregate. These local representations of the near null space are then orthogonalized via the QR factorization. The Q factor for a given aggregate spans the same local space as the original r vectors restricted to that aggregate. The tentative prolongation operator, P, is then formed by assembling these local factors;

the first r columns contain the Q from the first aggregate in the rows corresponding to that aggregate and zeros elsewhere. Similarly, columns $(j-1)r+1$ through jr contain the Q from the j^{th} aggregate on rows corresponding to nodes in that aggregate, and zeros elsewhere. Reordering the rows of A, such that those corresponding to degrees of freedom in each aggregate are grouped together, P has the block form

$$P = \begin{bmatrix} Q_1 & 0 & \cdots & 0 \\ 0 & Q_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q_{N_2} \end{bmatrix}.$$

This operator is then left-multiplied by the weighted-Jacobi smoothing matrix to form interpolation. Restriction and the coarse-grid matrix are then formed using the variational conditions.

This method can be extended to a multilevel method because the QR factorizations used in creating the tentative prolongation also produce sufficient information about the coarse-grid near null space. Notice the block structure in P, that the only nonzeros on the rows corresponding to a given aggregate come from the QR factorization. Thus, if P is applied to a block vector in which rows $(j-1)r+1$ through jr contain the R factor, the resulting vector matches the original r vectors' values on the fine grid. This block vector (with r columns and r times the number of aggregates rows) gives the near null space of the Galerkin coarse-grid operator of non-smoothed aggregation, P^TAP. The fine-grid vectors are taken to be algebraically smooth, however, and so the weighted-Jacobi smoothing applied to these vectors does not significantly change them. Similarly, while weighted Jacobi smoothing does significantly change the interpolation, it does not change the coarse-grid representations of the near null space. So, the coarse-grid problem can again be solved by aggregating the coarse-level nodes and using the coarse-level near null space representation to apply the smoothed aggregation methodology.

The generalized SA method can then be summarized as follows. Given the fine-grid matrix, $A^{(1)} \equiv A$, and right-hand side, $\mathbf{b}^{(1)} \equiv \mathbf{b}$, a hierarchy of coarse problems is generated by the Galerkin recurrence:

$$A^{(k+1)} = (I_{k+1}^k)^T A^{(k)} I_{k+1}^k, \tag{3.4}$$

where the interpolation operator, I_{k+1}^k, is defined as the product of a given smoother, S_k, and a tentative interpolation operator or prolongator, P_{k+1}^k:

$$I_{k+1}^k = S_k P_{k+1}^k, \tag{3.5}$$

for $k = 1, \ldots, L-1$. Given also a smoothing procedure for each level $k \in \{1, \ldots, L-1\}$ system, $A^{(k)}\mathbf{x}^{(k)} = \mathbf{b}^{(k)}$, of the form

$$\mathbf{x}^{(k)} \leftarrow (I - R^{(k)} A^{(k)})\mathbf{x}^{(k)} + R^{(k)}\mathbf{b}^{(k)},$$

where $R^{(k)}$ is some simple approximation to the inverse of $A^{(k)}$ (e.g., $R^{(k)} = s_k I$, where $s_k \approx \frac{1}{\rho(A^{(k)})}$) for $l = 1, \ldots, L-1$, we have all of the ingredients necessary for a multigrid scheme, as in Algorithm 1.

To make use of the existing convergence estimates, we assume that

$$\lambda_{\min}(I - R^{(k)} A^{(k)}) \geq 0 \quad \text{and} \quad \lambda_{\min}(R^{(k)}) \geq \frac{1}{C_R^2 \rho(A^{(k)})},$$

with a constant $C_R > 0$ independent of the level.

A good choice for S_k is described in [79] and scrutinized in more generality in [27]. For our purposes, it suffices to assume that the prolongation smoother, S_k, corresponds to the Richardson iteration for the level k problem, with the particular choice of damping suggested in [79]:

$$S_k = I - \frac{4}{3\ \lambda_k} A^{(k)},$$

where $\lambda_k = 9^{1-k}\bar{\lambda}$ and $\bar{\lambda}$ is a bound on the spectral radius of the fine-level matrix: $\rho(A^{(1)}) \leq \bar{\lambda}$. Starting with the given matrix, B^1, whose columns represent the near

kernel of the fine-level operator, we construct the tentative prolongators and the coarse-level representation of the near kernel components simultaneously to satisfy

$$P_{k+1}^k B^{k+1} = B^k, \quad (P_{k+1}^k)^T P_{k+1}^k = I. \tag{3.6}$$

Obtaining (3.6) amounts to solving a set of local independent orthonormalization problems in which the basis given by the fine-level near kernel matrix, B^k, restricted to the degrees of freedom of an aggregate, is orthonormalized using the QR algorithm. The resulting orthonormal basis forms the values of a block column of P_{k+1}^k, while the coefficients representing the old basis with respect to the new basis define B^{k+1}, as described above.

With these choices of smoothing components and a coarsening procedure utilizing (3.6), the standard smoothed aggregation scheme can be proven to converge under certain assumptions on the near kernel components alone. The following such result motivates the need for standard smoothed aggregation to have access to the near kernel components.

Let $\langle \mathbf{u}, \mathbf{v} \rangle_{\mathcal{A}}$ denote the Euclidean inner product over the degrees of freedom corresponding to an agglomerate $\mathcal{A}$, and $\| \cdot \|_{\mathcal{A}}$ be the associated norm. Denote the $A^{(1)}$-norm by $|||\mathbf{u}||| = \langle A^{(1)}\mathbf{u}, \mathbf{u} \rangle^{1/2}$. Let B^1 denote an $N \times r$ matrix whose columns are thought to form a basis for the near kernel components corresponding to $A^{(1)}$.

Theorem 1 (Theorem 4.2 of [79]). *Let $\tilde{\mathcal{A}}_i^k$ denote the set of fine-level degrees of freedom corresponding to aggregate $\mathcal{A}_i^k$ on level k, and assume that there exists a constant, $C_a > 0$, such that for every $\mathbf{u} \in \mathbb{R}^{\mathrm{N}}$ and every $k = 1, \ldots, L-1$, the following approximation property holds:*

$$\sum_i \min_{\mathbf{w} \in \mathbb{R}^{\mathrm{r}}} \| \mathbf{u} - B^1 \mathbf{w} \|_{\tilde{\mathcal{A}}_i^k}^2 \leq C_a \frac{9^{k-1}}{\rho(A^{(1)})} |||\mathbf{u}|||^2. \tag{3.7}$$

Then

$$|||\tilde{\mathbf{x}} - MG^{(1)}(\mathbf{x}, \mathbf{b}^{(1)})||| \leq \left(1 - \frac{1}{c(L)}\right) |||\tilde{\mathbf{x}} - \mathbf{x}||| \qquad \forall \tilde{\mathbf{x}} \in \mathbb{R}^{\mathrm{N}},$$

where $A^{(1)}\mathbf{x} = \mathbf{b}^{(1)}$*, and* $c(L)$ *is a polynomial of degree 3 in* L*.*

Since the use of (3.6) is assumed, condition (3.7) reflects an assumption on all tentative prolongators, P_{k+1}^k, and can be equivalently restated as

$$\sum_i \min_{\mathbf{w}\in\mathbb{R}^r} \|\mathbf{u} - P_2^1 P_3^2 \dots P_{k+1}^k B^{k+1}\mathbf{w}\|^2_{\tilde{\mathcal{A}}_i^k} \leq C_a \frac{9^{k-1}}{\rho(A^{(1)})} |||\mathbf{u}|||^2 \tag{3.8}$$

for every $\mathbf{u} \in \mathbb{R}^N$ and every $k = 1, \dots, L-1$. Thus, in the context of smoothed aggregation, condition (3.7) can be viewed as an alternative formulation of the weak approximation property [13]. Note that the required approximation of a fine-level vector is less stringent for coarser levels. Also, convergence is guaranteed even though no regularity assumptions have been made. Although the convergence bound naturally depends on the number of levels, computational experiments suggest that the presence of elliptic regularity for standard test problems yields optimal performance (i.e., convergence with bounds that are independent of the number of levels).

This generalized smoothed aggregation method has a clear advantage over the Ruge-Stüben algorithm when applied to problems with higher-dimensional near null spaces, such as the biharmonic or second-order systems like elasticity. Vaněk et al. [80] demonstrate numerical performance of the method on a number of elasticity problems with quite reasonable convergence factors per iteration, under the assumption of a known discretization geometry. This comes, however, with a high setup cost. In practice, both the Ruge-Stüben algorithm and SA can be used to solve problems such as discretizations of linear elasticity models with reasonable performance.

The main drawback of the SA method is the need for knowledge of the near null space. For problems such as elasticity, detailed knowledge of the discretization is needed to represent the translations and rotations which make up the so-called rigid body modes. Without control of the discretization, this information may not be available. In fact, if local rotations of the discretization matrix are employed (such as if the matrix is rescaled for a block-Jacobi-preconditioned conjugate gradient method), these

components may not even be easily recovered. We consider this situation in Chapter 5, where effective solvers are designed without assuming the availability of this information.

Chapter 4

Multigrid and Upscaling for Flow in Porous Media

Modeling the flow of a single-phase, saturated fluid may be done with Darcy's law and conservation of mass, as in Equations 2.1 and 2.2:

$$\mathbf{u}(\mathbf{x}) = -\mathcal{K}(\mathbf{x})\nabla p(\mathbf{x}),$$
$$\nabla \cdot \mathbf{u}(\mathbf{x}) = Q(\mathbf{x}).$$

Here, we consider methods of upscaling to cope with the large linear systems that arise from discretizing these equations on the fine scales needed to capture the appropriate physical behavior of the fluid.

4.1 Background

Regardless of the method chosen to discretize Equations 2.1 and 2.2, the result is a linear system of the form $A\mathbf{x} = \mathbf{b}$ that we seek to solve. The primary difficulty in solving such systems is the large size of the matrix, A, due to the fine mesh necessary to capture fluctuations in $\mathcal{K}(\mathbf{x})$, which, in many physical examples, is highly variable, often over a wide range of spatial scales. For this reason, it is difficult to perform computation on a mesh that fully resolves all of the spatial scales in the permeability and, hence, the solution. For example, a typical reservoir simulation involves a three-dimensional physical domain several kilometers long in each direction. The primary material properties of porosity and permeability can, however, vary on the scale of

millimeters. Thus, simply representing the computational domain may require $O(10^{18})$ degrees-of-freedom (DOFs).

An alternate approach is clearly necessary in order to make any progress in such a simulation. The primary concern in evaluating possible methods must be that of accuracy versus computational cost. Many of the apparent possibilities offer very low cost, but similarly low accuracy. Likewise, there is no shortage of high-accuracy, high-complexity methods, although these are typically not feasible. Indeed, the high cost of a fully resolved simulation is the reason we investigate these alternative techniques.

In practice, there is very little need or interest in having access to a complete, accurate, fine-scale approximation of the primary variable, $p(\mathbf{x})$. Rather, macroscopic properties of the flow, such as total flux, breakthrough time, or extreme pressures, are the important characteristics for simulation. While this suggests a coarse-scale simulation, the fine-scale material properties may significantly influence these macroscopic properties. We are, thus, constrained to consider methods that somehow utilize the full fine-scale specification of the permeability.

We consider approaches of upscaling the fine-scale permeability and the fine-scale PDE. That is, we look to define a coarse-scale permeability (or PDE) that captures the essential features of the fine-scale variation, but does so on a scale that is computationally tractable. A permeability determined by such techniques is known in the literature as the equivalent, effective, upscaled, or homogenized permeability. We also consider techniques of representing the PDE itself on a coarser scale, regardless of the the properties of upscaled permeabilities, but which achieve the goal of accurately approximating the properties of interest using fewer DOFs.

4.1.1 Two-scale Asymptotic Theory of Homogenization

This question of upscaling can be posed mathematically: find an appropriately averaged form of the PDE,

$$-\nabla \cdot \mathcal{K}(\mathbf{x})\nabla p(\mathbf{x}) = Q(\mathbf{x}), \tag{4.1}$$

of the same form, but with constant (homogenized) coefficient, $\hat{\mathcal{K}}$. Supposing that the symmetric and positive definite tensor, $\mathcal{K}(\mathbf{x})$, varies with two spatial scales, $\mathbf{x}$ and $\mathbf{y} = \dfrac{\mathbf{x}}{\epsilon}$ for small ϵ, we can introduce a classical, two-scale asymptotic analysis of Equation 4.1 subject to periodic boundary conditions [5]. The explicit introduction of the second length-scale is indicative of a separation of the scales in $\mathcal{K}$ into a slowly varying scale ($\mathbf{x}$) and a fast scale ($\frac{\mathbf{x}}{\epsilon}$). This two-scale behavior of the coefficient necessarily introduces two-scale behavior into the solution, p, and so we write $p(\mathbf{x}) \equiv p(\mathbf{x}, \frac{\mathbf{x}}{\epsilon}) \equiv p(\mathbf{x}, \mathbf{y})$. Now, expanding p in powers of ϵ, we can perform the standard asymptotic analysis. In what follows, we assume that variation of $\mathcal{K}$ is negligible on the slow scale and that $\mathcal{K}$ is periodic in the fast scale. That is, $\mathcal{K}(\mathbf{x}, \mathbf{y}) \equiv \mathcal{K}(\mathbf{y})$.

Writing

$$p = p_0(\mathbf{x}, \mathbf{y}) + \epsilon p_1(\mathbf{x}, \mathbf{y}) + \epsilon^2 p_2(\mathbf{x}, \mathbf{y}) + \dots$$

and formalizing $\nabla = \nabla_{\mathbf{x}} + \frac{1}{\epsilon}\nabla_{\mathbf{y}}$, we separate powers of ϵ in the resulting two-scale equations, $-\nabla \cdot \mathcal{K}(\mathbf{y})\nabla p(\mathbf{x}, \mathbf{y}) = Q(\mathbf{x})$, as

$$O\left(\frac{1}{\epsilon^2}\right): \qquad A_1 p_0(\mathbf{x}, \mathbf{y}) = 0, \tag{4.2}$$

$$O\left(\frac{1}{\epsilon}\right): \qquad A_1 p_1(\mathbf{x}, \mathbf{y}) + A_2 p_0(\mathbf{x}, \mathbf{y}) = 0, \tag{4.3}$$

$$O(1): \qquad A_1 p_2(\mathbf{x}, \mathbf{y}) + A_2 p_1(\mathbf{x}, \mathbf{y}) + A_3 p_0(\mathbf{x}, \mathbf{y}) = Q(\mathbf{x}), \tag{4.4}$$

splitting the operator, $-\nabla \cdot \mathcal{K}(\mathbf{y})\nabla$, into

$$\begin{aligned} O\left(\tfrac{1}{\epsilon^2}\right): &\quad A_1 = -\nabla_{\mathbf{y}} \cdot \mathcal{K}(\mathbf{y})\nabla_{\mathbf{y}}, \\ O\left(\tfrac{1}{\epsilon}\right): &\quad A_2 = -\nabla_{\mathbf{y}} \cdot \mathcal{K}(\mathbf{y})\nabla_{\mathbf{x}} - \nabla_{\mathbf{x}} \cdot \mathcal{K}(\mathbf{y})\nabla_{\mathbf{y}}, \\ O(1): &\quad A_3 = -\nabla_{\mathbf{x}} \cdot \mathcal{K}(\mathbf{y})\nabla_{\mathbf{x}}. \end{aligned}$$

While we could proceed further in this expansion, the first three terms are sufficient for recovery of the homogenized (upscaled) operator. Equation 4.2 asks that

$$-\nabla_{\mathbf{y}} \cdot \mathcal{K}(\mathbf{y}) \nabla_{\mathbf{y}} p_0(\mathbf{x}, \mathbf{y}) = 0,$$

with periodic boundary conditions (in $\mathbf{y}$) on $p_0(\mathbf{x}, \mathbf{y})$. The unique solution to this problem is $p_0(\mathbf{x}, \mathbf{y}) = p_0(\mathbf{x})$, i.e., the leading term of the asymptotic expansion of $p(\mathbf{x}, \mathbf{y})$ depends only on the slow scale, $\mathbf{x}$.

Equation 4.3, for $p_1(\mathbf{x}, \mathbf{y})$, then becomes

$$\begin{aligned} -\nabla_{\mathbf{y}} \cdot \mathcal{K}(\mathbf{y}) \nabla_{\mathbf{y}} p_1(\mathbf{x}, \mathbf{y}) &= \left(\nabla_{\mathbf{y}} \cdot \mathcal{K}(\mathbf{y}) \nabla_{\mathbf{x}} + \nabla_{\mathbf{x}} \cdot \mathcal{K}(\mathbf{y}) \nabla_{\mathbf{y}}\right) p_0(\mathbf{x}) \\ &= \nabla_{\mathbf{y}} \cdot \mathcal{K}(\mathbf{y}) \nabla_{\mathbf{x}} p_0(\mathbf{x}). \end{aligned}$$

Introducing functions χ_j, for $j = 1, 2$, to be the periodic (in $\mathbf{y}$) solutions of

$$A_1 \chi_j = A_1 y_j = \sum_i \frac{\partial}{\partial y_i} k_{ij}(\mathbf{y}), \tag{4.5}$$

gives us the general solution of Equation 4.3,

$$p_1(\mathbf{x}, \mathbf{y}) = -\sum_j \chi_j(\mathbf{y}) \frac{\partial p_0}{\partial x_j}(\mathbf{x}) + \tilde{p}_1(\mathbf{x}),$$

where $\tilde{p}_1(\mathbf{x})$ is any function independent of $\mathbf{y}$. Trying to solve Equation 4.4 for $p_2(\mathbf{x}, \mathbf{y})$ leads to the homogenized equation for $p_0(\mathbf{x})$, due to the periodic boundary conditions (in $\mathbf{y}$) on $p_2(\mathbf{x}, \mathbf{y})$. Since the solution to the equation, $A_1 \phi = f$, ϕ periodic in y, exists (and is unique up to an additive constant) if and only if $\iint f(\mathbf{y}) d\mathbf{y} = 0$, the solution, $p_2(\mathbf{x}, \mathbf{y})$, to Equation 4.4 exists only if

$$\iint A_2 p_1(\mathbf{x}, \mathbf{y}) + A_3 p_0(\mathbf{x}) d\mathbf{y} = \iint Q(\mathbf{x}) d\mathbf{y} = |Y| Q(\mathbf{x}), \tag{4.6}$$

where $|Y|$ is the area of the subdomain parametrized by $\mathbf{y}$. Applying the divergence theorem,

$$\iint \nabla_{\mathbf{y}} \cdot (\mathcal{K}(\mathbf{y}) \nabla_x p_1(\mathbf{x}, \mathbf{y})) d\mathbf{y} = \int (\mathcal{K}(\mathbf{y}) \nabla_x p_1(\mathbf{x}, \mathbf{y})) \cdot ds,$$

and this boundary integral is zero by periodicity. Notice then that

$$\begin{aligned}\iint A_2 p_1(\mathbf{x},\mathbf{y})d\mathbf{y} &= -\sum_i \frac{\partial}{\partial x_i}\iint \left(\sum_l k_{il}(\mathbf{y})\frac{\partial p_1}{\partial y_l}(\mathbf{x},\mathbf{y})\right) d\mathbf{y} \\ &= \sum_{i,j}\frac{\partial}{\partial x_i}\iint\left(\sum_l k_{il}(\mathbf{y})\frac{\partial \chi_j}{\partial y_l}(\mathbf{y})\right)d\mathbf{y}\frac{\partial p_0}{\partial x_j}(\mathbf{x}),\end{aligned}$$

and, thus, we can rewrite Equation 4.6 as

$$-\sum_{i,j}\frac{\partial}{\partial x_i}\left(\frac{1}{|Y|}\iint\left(k_{ij}(\mathbf{y})-\sum_l k_{il}(\mathbf{y})\frac{\partial\chi_j}{\partial y_l}(\mathbf{y})\right)d\mathbf{y}\right)\frac{\partial p_0}{\partial x_j}(\mathbf{x}) = Q(\mathbf{x}),$$

or as $-\nabla_{\mathbf{x}}\cdot\hat{\mathcal{K}}\nabla_{\mathbf{x}}p_0(\mathbf{x}) = Q(\mathbf{x})$, with

$$\hat{k}_{ij} = \frac{1}{|Y|}\iint\left(k_{ij}(\mathbf{y})-\sum_l k_{il}(\mathbf{y})\frac{\partial\chi_j}{\partial y_l}(\mathbf{y})\right)d\mathbf{y}.$$

If we also define a weak form of the operator A_1,

$$a_1(\phi,\psi) = \sum_{i,j}\iint k_{ij}(\mathbf{y})\frac{\partial\phi}{\partial y_j}\frac{\partial\psi}{\partial y_i}d\mathbf{y}, \tag{4.7}$$

we can express the homogenized permeability, $\hat{\mathcal{K}}$, as

$$\hat{k}_{lm} = -\frac{1}{|Y|}a_1(\chi_m - y_m, y_l). \tag{4.8}$$

By Equation 4.5, $a_1(\chi_m - y_m,\psi) = 0$ for any ψ periodic in $\mathbf{y}$. Thus, $a_1(\chi_m - y_m,\chi_l) = 0$, and we can also express $\hat{\mathcal{K}}$ as

$$\hat{k}_{lm} = \frac{1}{|Y|}a_1(\chi_m - y_m, \chi_l - y_l). \tag{4.9}$$

Since $\mathcal{K}(\mathbf{y})$ is symmetric (by our original assumption), bilinear form $a_1(\cdot,\cdot)$ is also symmetric and Equation 4.9 implies the symmetry of $\hat{\mathcal{K}}$. Similarly, one can establish the positive-definiteness of $\hat{\mathcal{K}}$ [5].

4.1.2 Bourgat's Examples

This method was implemented by Bourgat [7] and tested for a number of model problems discretized by triangular finite elements in two dimensions. Three particular

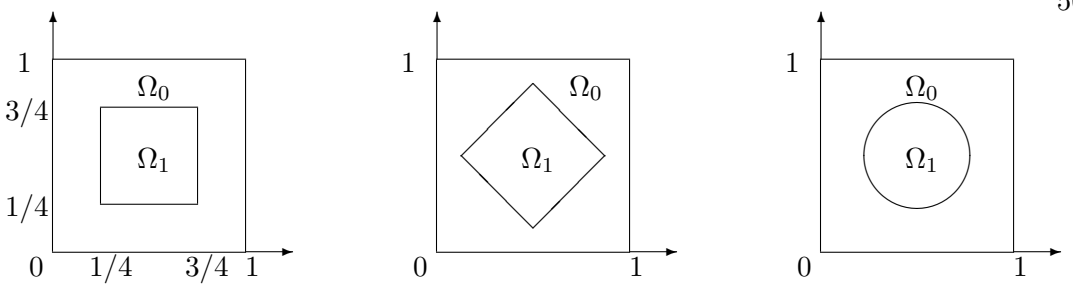

Figure 4.1: Bourgat's Shape-Variation Inclusion Problems

questions and related examples are considered in detail, which we also look at in Section 4.3.

In a periodic structure problem, such as the sand/shale examples of Section 2.2, it may happen that the media is known to be composed of two materials, but the exact geometry of the structure is unknown, even if some other properties (such as the volume of each material present) are known. A simple model of this phenomenon may be created by varying the shape of an inclusion within a homogeneous background, such as in Figure 4.1, where the area of Ω_1 is fixed to be $\frac{1}{4}$ for all three shapes. A scalar permeability, $\mathcal{K}(\mathbf{x})$, may then be given as, for example,

$$\mathcal{K}(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in \Omega_0, \\ 10 & \mathbf{x} \in \Omega_1. \end{cases} \tag{4.10}$$

Bourgat's calculations of the homogenized permeabilities for these three cases show that there is shape dependence in the homogenized permeability.

To test the dependence of the homogenized coefficients on the magnitudes of permeability, consider fixing the geometry of the structure, as in Figure 2.1, but allowing the permeability to vary. If scalar $\mathcal{K}(\mathbf{x})$ is given as

$$\mathcal{K}(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in \Omega_0, \\ \lambda & \mathbf{x} \in \Omega_1, \end{cases}$$

with λ varying between 0 and ∞, the homogenized permeability can be compared against simple averages of the fine-scale $\mathcal{K}$, such as the arithmetic and harmonic means.

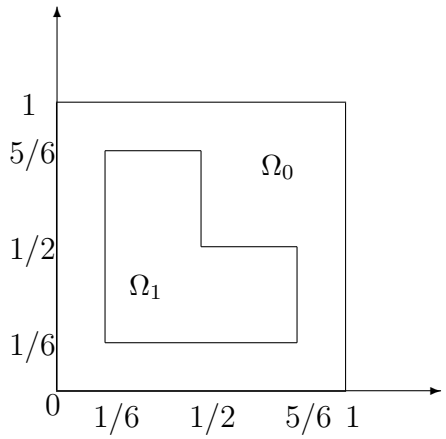

Figure 4.2: Bourgat's L-Shaped Inclusion Problem

In both these examples, the given fine-scale permeability tensor is diagonal (in fact, scalar) and symmetric with respect to lines $x = \frac{1}{2}$ and $y = \frac{1}{2}$. A theorem of Bourgat and Dervieux [7] implies that the homogenized tensors must then also be diagonal. To test a case where such symmetry is broken, Bourgat considers the L-shaped inclusion, shown in Figure 4.2. Even with a simple scalar permeability, such as in Equation 4.10, the homogenized tensor is shown to be full.

These three simple problems demonstrate quite a range in the behavior of the homogenized permeabilities. Thus, we also consider them as tests of our upscaling method.

4.2 Upscaling Techniques

Bourgat shows that it is possible to compute the homogenized coefficient, $\hat{\mathcal{K}}$, numerically using a sufficiently fine mesh on a periodic cell. In practice, however, we are interested in determining equivalent operators (and recovering macroscopic solution features of the flow) in situations where this theory does not strictly apply. While the assumption of periodic cells may be reasonable in some material science applications, it is not realistic in the case of flow in an aquifer or reservoir. For this reason, we

distinguish between the theory of homogenization, as discussed above (and in [5, 58]), and numerical methods of upscaling. While homogenization refers to the theoretical process of determining the appropriately averaged operator through analytic techniques, upscaling is the process of numerically determining a coarse-scale discrete operator or function with relevant macroscopic properties.

4.2.1 Local Techniques

The earliest investigations into upscaling were focused on the applicability of explicit averages in the determination of effective permeabilities. Cardwell and Parsons [28] show that the effective permeability must lie between the arithmetic and harmonic averages of a given permeability distribution. These bounds are, in fact, sharp: the effective permeability of a layered medium is simply the arithmetic average of the layered permeabilities for flow aligned with the layers, because the flux through each layer is proportional to the permeability and height of that layer; the effective permeability of flow perpendicular to the layering is the harmonic average of the layered permeabilities, as can be determined by reducing the problem to a one-dimensional flow.

Warren and Price [85] examine these averages as well as the geometric mean, and conclude that the geometric mean gives the best match to the effective permeability when $\mathcal{K}(\mathbf{x})$ is generated from a known probability density function. This agrees with much other work, as reviewed in [86, §3.1.1]. As we have already seen, this result does not apply in many situations, including when the media is regularly structured. In general, we do not expect any simple average to predict the effective permeability in all situations.

Another local technique is the renormalization technique, first introduced by King [59]. Based on effective media theory, this technique utilizes successive upscaling of two by two blocks on which the permeability is known, where the permeability is assumed to be constant on each cell of a regular mesh. The block permeability is determined by

posing the problem as a network of resistors, while imposing pressure on two opposing sides of the domain and no-flow boundary conditions on the remaining two sides.

The resistance problem is posed by considering each cell of constant permeability, $\mathcal{K}$, to be composed of a network of four resistors of resistance $\frac{1}{2\mathcal{K}}$, each attached in series to one edge and the cell midpoint. Four such cells are then considered in unison, discarding inputs from two opposing edges of a 2×2 block (the no-flow boundary conditions), and constant pressures given on the remaining two edges. The network then consists of a circuit of resistors arranged in both parallel and serial, from which the equivalent resistance of the circuit may be computed. The equivalent permeability of the cell can then be recovered by similar means.

This method has the distinct advantages that it is cheap and also accurate in some geometrically structured cases. Indeed, in the case of layered media, it predicts exactly the arithmetic mean for flow parallel to the layers and the harmonic average for flow perpendicular to the layers. The calculation of the equivalent permeability of a 2×2 block requires only a small number of operations, and larger blocks may be upscaled recursively. A significant disadvantage, however, is that the method only produces diagonal tensors, due to the imposed local boundary conditions. For this reason, the renormalization method is not appropriate for simulations of many composite media, including the simple example shown in Figure 4.2.

In general, local upscaling techniques for determining equivalent permeabilities have many drawbacks outweighing their low costs. The increased accuracy of the non-local techniques discussed below typically more than compensates for the significant increase in computation required. Recall that our measure of interest is that of accuracy per computational cost, and thus we must compare these methods based on both criteria and seek a balance between the two.

4.2.2 Laplacian Methods

A large group of nonlocal techniques are commonly called Laplacian methods, due to their reliance on the solution of local diffusion problems. In these techniques, a block to be upscaled is considered in isolation from the remainder of the domain, and local boundary conditions are imposed on the block in such a way as to induce a flux across the subdomain. Comparison of this flux to the local pressure gradient then yields an upscaled permeability.

In the "simple Laplacian" approach, the imposed boundary conditions are Dirichlet along two edges and homogeneous Neumann boundary conditions on the remaining two edges (e.g., to compute the permeability in the x-direction, constant pressures are prescribed along the lines $x = x_0$ and $x = x_1$ and no-flow conditions are imposed along $y = y_0$ and $y = y_1$). Computing the flux through any fixed vertical line, $x = x_f$ for $x_0 \leq x_f \leq x_1$, and knowing the difference in the pressures prescribed at x_0 and x_1 allows for computation of the component of the upscaled permeability in the x-direction. These boundary conditions may then be rotated 90 degrees and the procedure repeated to compute the y-component of the upscaled permeability. This approach was used by Warren and Price [85] to compare with the geometric mean.

While this technique does offer significant improvement over the local, additive techniques above, it still suffers from some of the same failings. In particular, the upscaled permeability is constrained to be a diagonal tensor due to the boundary conditions imposed on the subdomain problems. Indeed, these boundary conditions may not even be relevant to the global flow problem whose simulation is of interest. Wen and Gómez-Hernández [86] outline a number of modifications of this technique used to improve upon these negative features.

The "simple Laplacian extended" technique takes advantage of the fact that an iterative method, such as conjugate gradient, is usually used to solve the subdomain

flow problems. To improve the efficiency of the simple Laplacian technique, after each step of the iterative solution method, the necessary flux of the approximate solution is computed, and, from this, the upscaled permeability component. The solution iteration is not carried out to solution of the linear system for the pressures, but rather only until the permeability is resolved, typically in many fewer iterations.

Considering the problem of multi-phase flow, White [87] proposes using the global single-phase flow solution to determine local upscaled permeabilities. That is, the global single-phase flow problem is solved for the global pressure, with various sets of boundary conditions. For each solution, the local flux-pressure relationship, $\mathbf{q} = -\mathcal{K}\nabla p$, is then used to give two equations for the three entries in the symmetric tensor $\mathcal{K}$ (in two dimensions), assumed now to be constant on each subdomain. An overdetermined system for $\mathcal{K}$ on each subdomain is then formed by coupling the underdetermined systems from each set of boundary conditions. This system may then be solved in a least-squares sense. Such an approach allows for full tensors and, thus, may be much more accurate than the simple Laplacian technique. It is, however, significantly more expensive (several times the cost of solving the fine-scale single-phase flow problem), and is therefore only useful when considering situations such as multi-phase flow.

While this technique is considerably more expensive than is practical, a simplification of it is much more attractive. A main disadvantage of the simple Laplacian method is the effect of imposing local boundary conditions and so, in the "Laplacian with skin" method, the flow equation is solved for each subdomain over a region that includes both the subdomain and a surrounding buffer region. In this way, the subdomain is insulated from the imposed boundary conditions, yet a global solve is avoided. The same overdetermination as used in the global solve case may also be used to further buffer the effects of the boundary conditions.

4.2.3 Periodic Laplacian

Durlofsky presents a similar method [39], which utilizes two sets of periodic boundary conditions instead of Dirichlet and Neumann boundaries. This approach has the advantage that the resulting upscaled permeability may be tensor-valued, and is constrained to be symmetric and positive definite. Indeed, it must have these features, as we show in the following theorem, as it is simply an implementation of the classical, periodic theory as in [5]. Durlofsky demonstrates that the method is also reasonably robust in situations where this classical theory does not apply, such as fractally-generated permeability fields.

In [39], it is argued that a form of periodic boundary conditions is more appropriate within the Laplacian scheme. In particular, if the local pressure near point $\mathbf{x}_0$ satisfies

$$p = p_0 + \mathbf{G} \cdot (\mathbf{x} - \mathbf{x}_0),$$

then the local fluid velocity, $\mathbf{u}$, satisfies

$$\mathbf{u} = -\mathcal{K} \cdot \mathbf{G}.$$

Extending this to the subdomain, flows there are induced with known, approximately constant pressure gradients, $\mathbf{G}$. The flux, $\mathbf{u}$, for these flows is then computed and $\mathcal{K}$ is recovered. Thus, on each subdomain, we look to solve $-\nabla \cdot \mathcal{K}\nabla p = 0$ subject to boundary conditions that give an appropriate pressure gradient. Considering a 2D square subdomain mapped to $[0,1]^2$, Durlofsky introduces the subdomain coordinates, (y_1, y_2), and solves for $p(\mathbf{y})$ using boundary conditions

$$\begin{aligned} p(0, y_2) &= p(1, y_2) - G_1, \\ p(y_1, 0) &= p(y_1, 1) - G_2, \end{aligned} \tag{4.11}$$

to induce the pressure gradient. The average fluid velocity of the solution to the subdomain problem is then calculated and used to compute the effective permeability, $\mathcal{K}^*$,

as

$$\begin{aligned} \langle u_1 \rangle &= -(k_{11}^* G_1 + k_{12}^* G_2), \\ \langle u_2 \rangle &= -(k_{21}^* G_1 + k_{22}^* G_2), \end{aligned} \tag{4.12}$$

where $\langle u_1 \rangle$ and $\langle u_2 \rangle$ are computed by integrating $-\mathbf{u} \cdot \mathbf{n}$, for outward normal vector, $\mathbf{n}$, across a boundary of the subdomain. The four components of the tensor can be recovered by solving the PDE for two independent vectors, $\mathbf{G}$, typically taken to be $\binom{1}{0}$ and $\binom{0}{1}$.

Theorem 2. *Let $\mathcal{K}^*$ be defined as in Equation 4.12, where $p(\mathbf{x})$ solves $-\nabla \cdot \mathcal{K} \nabla p = 0$ with boundary conditions as in Equation 4.11 and $\mathbf{u} = -\mathcal{K} \nabla p$, and let $\hat{\mathcal{K}}$ be defined by $\hat{k}_{lm} = -\frac{1}{|Y|} a_1(\chi_m - y_m, y_l)$, as in Equation 4.8. Then*

$$\mathcal{K}^* = \hat{\mathcal{K}}.$$

Proof. To relate the classical theory to Durlofsky's method, first recast the periodic problem,

$$-\nabla_{\mathbf{y}} \cdot \mathcal{K}(\mathbf{y}) \nabla_{\mathbf{y}} p(\mathbf{y}) = 0, \tag{4.13}$$

$$\begin{aligned} p(0, y_2) &= p(1, y_2) - G_1 \\ p(y_1, 0) &= p(y_1, 1) - G_2, \end{aligned} \tag{4.14}$$

as a problem with homogeneous periodic boundary conditions. To do this, we consider explicitly the case $G_1 = 1$, $G_2 = 0$; the case $G_1 = 0$, $G_2 = 1$ follows analogously. Writing $q(\mathbf{y}) = p(\mathbf{y}) - y_1$, $q(\mathbf{y})$ must then satisfy the desired homogeneous boundary conditions from Equation 4.14. Substituting $p(\mathbf{y}) = q(\mathbf{y}) + y_1$ into Equation 4.13, we have

$$-\nabla_{\mathbf{y}} \cdot \mathcal{K} \nabla_{\mathbf{y}} q(\mathbf{y}) = \nabla_{\mathbf{y}} \cdot \mathcal{K} \nabla_{\mathbf{y}} y_1,$$

so $q(\mathbf{y}) = -\chi_1(\mathbf{y})$ and $p(\mathbf{y}) = y_1 - \chi_1(\mathbf{y})$. Similarly, for the case $G_1 = 0$, $G_2 = 1$,

$p(\mathbf{y}) = y_2 - \chi_2(\mathbf{y})$. Durlofsky then computes the fluxes,

$$\langle u_1 \rangle = \int (\mathcal{K}(\mathbf{y}) \nabla_{\mathbf{y}} p(\mathbf{y})) \cdot \mathbf{n}_1 dy_2,$$
$$\langle u_2 \rangle = \int (\mathcal{K}(\mathbf{y}) \nabla_{\mathbf{y}} p(\mathbf{y})) \cdot \mathbf{n}_2 dy_1,$$

where the integrals are taken along the so-called outflow boundary, i.e., $\langle u_i \rangle$ is evaluated along $y_i = 0$. Here, $\mathbf{n}_i$ is taken to be the unit normal out of the domain from line $y_i = 0$, although it does not matter which line these integrals are evaluated along because mass is conserved. To see this conservation, consider differentiating the integral over y_2 with respect to y_1:

$$\begin{aligned}
&\frac{\partial}{\partial y_1} \int_0^1 (\mathcal{K}(\mathbf{y}) \nabla_{\mathbf{y}} p(\mathbf{y})) \cdot \left(\begin{smallmatrix} -1 \\ 0 \end{smallmatrix}\right) dy_2 \\
&\qquad = -\frac{\partial}{\partial y_1} \int_0^1 \left(k_{11} \frac{\partial p}{\partial y_1} + k_{12} \frac{\partial p}{\partial y_2} \right) dy_2 \\
&\qquad = -\int_0^1 \frac{\partial}{\partial y_1} \left(k_{11} \frac{\partial p}{\partial y_1} + k_{12} \frac{\partial p}{\partial y_2} \right) dy_2 \\
&\qquad = \int_0^1 \frac{\partial}{\partial y_2} \left(k_{21} \frac{\partial p}{\partial y_1} + k_{22} \frac{\partial p}{\partial y_2} \right) dy_2 \\
&\qquad = \left(k_{21} \frac{\partial p}{\partial y_1} + k_{22} \frac{\partial p}{\partial y_2} \right) \Bigg|_{y_2=0}^{y_2=1} \\
&\qquad = 0,
\end{aligned}$$

using the properties that $(\mathcal{K} \nabla_{\mathbf{y}} p) \cdot \mathbf{n}$ is continuous, $\nabla_{\mathbf{y}} \cdot \mathcal{K} \nabla_{\mathbf{y}} p = 0$, and p is periodic in y_2. The analogous result holds for the integral with respect to y_1, and so the fluxes parallel to the edges of the subdomain are constant and can also be expressed as

$$\langle u_1 \rangle = \iint (\mathcal{K}(\mathbf{y}) \nabla_{\mathbf{y}} p(\mathbf{y})) \cdot \mathbf{n}_1 d\mathbf{y},$$
$$\langle u_2 \rangle = \iint (\mathcal{K}(\mathbf{y}) \nabla_{\mathbf{y}} p(\mathbf{y})) \cdot \mathbf{n}_2 d\mathbf{y}.$$

Notice, now, that $\mathbf{n}_1 = \left(\begin{smallmatrix} -1 \\ 0 \end{smallmatrix}\right) = -\nabla y_1$ and that $\mathbf{n}_2 = -\nabla y_2$, so we can again rewrite

$\langle u_1 \rangle$ and $\langle u_2 \rangle$:

$$\langle u_1 \rangle = -\iint (\mathcal{K}(\mathbf{y}) \nabla_{\mathbf{y}} p(\mathbf{y})) \cdot \nabla y_1 d\mathbf{y},$$
$$\langle u_2 \rangle = -\iint (\mathcal{K}(\mathbf{y}) \nabla_{\mathbf{y}} p(\mathbf{y})) \cdot \nabla y_2 d\mathbf{y}.$$

Recalling the definition of $a_1(\cdot, \cdot)$ from Equation 4.7, we have

$$\langle u_1 \rangle = -a_1(p(\mathbf{y}), y_1),$$
$$\langle u_2 \rangle = -a_1(p(\mathbf{y}), y_2).$$

When $p(\mathbf{y}) = y_1 - \chi_1(\mathbf{y})$ (i.e. $G_1 = 1, G_2 = 0$),

$$\langle u_1 \rangle = a_1(\chi_1 - y_1, y_1),$$
$$\langle u_2 \rangle = a_1(\chi_1 - y_1, y_2),$$

and, when $p(\mathbf{y}) = y_2 - \chi_2(\mathbf{y})$ (i.e. $G_0 = 0, G_2 = 1$),

$$\langle u_1 \rangle = a_1(\chi_2 - y_2, y_1),$$
$$\langle u_2 \rangle = a_1(\chi_2 - y_2, y_2).$$

Now, considering Equation 4.12 in these two cases, we recover

$$k^*_{11} = -a_1(\chi_1 - y_1, y_1),$$
$$k^*_{12} = -a_1(\chi_2 - y_2, y_1),$$
$$k^*_{21} = -a_1(\chi_1 - y_1, y_2),$$
$$k^*_{22} = -a_1(\chi_2 - y_2, y_2).$$

Noticing that Durlofsky's calculation is on a unit cell, so that $|Y| = 1$, this is in exact agreement with the the periodic theory, as in Equation 4.8. □

Corollary 1. *The tensor, $\mathcal{K}^*$, recovered from Equation 4.12 is symmetric and positive definite.*

In [39], these subdomain problems are discretized and solved with a nonconforming finite element technique. Desiring a continuous approximation of the flux, this is an appropriate choice of finite elements. Mixed finite elements were again considered in that study, but discarded due to the lack of an efficient solver. In [40], mixed finite elements were also considered, along with a control volume finite element formulation. While a conservative scheme, the control volume discretization was found to be more efficient only for mild heterogeneity. In the case of highly variable permeability fields, the mixed finite element discretization considered gave more accurate approximations to the flow variables than the control volume scheme. In [52], the nonconforming method is again used, but now extended to more general geometries. In particular, the problem of upscaling is considered using a flow-based partitioning into subdomains. When subdomains are chosen based on the global flow pattern, an improvement in accuracy over a uniform partitioning with the same number of subdomains is found.

4.3 Multigrid Upscaling

All of these non-local techniques require a certain amount of fine-scale computation on each subdomain, typically two or more fine-scale solutions over each subdomain. Thus, the primary cost of these methods is in the multiple solves of subdomain-scale PDEs.

Multigrid methods for solving these problems typically exhibit the optimal behavior discussed in Section 3.1. That is, the cost of solution of a subdomain problem is proportional to the number of degrees of freedom in the subdomain. This principle also applies at the fine scale: multigrid solution of the fully-resolved problem requires time proportional to the total number of degrees of freedom. When the problem is subdivided, however, each fine-scale node belongs to a subdomain, and so solution of the complete set of subdomain problems has cost the same order as the solution of the original, fine-scale problem. In fact, the solution of the complete set of subdomain

problems can cost more than the solution of the fine-scale problem, because multiple problems must be solved on each subdomain. This is clearly efficient only in the case of a truly periodic media, where solution of a single subdomain problem may be used to upscale a much larger part of the fully-resolved system.

Due to the strong dependence of the macroscopic properties of interest on the fine-scale structure of the porous media, a certain amount of fine-scale computation is inevitable. Allowing this (to maintain hope of accurate results), we look instead to perform a minimal amount of computation on the fine scale.

The goal of coarsening in a multigrid process is to create coarse-scale problems that accurately capture features of the fine-scale solution not resolved by relaxation on the fine grid. As fine-scale relaxation is (usually) a local process, the unresolved features typically represent larger-scale features of the solution, including the macroscopic properties that we are interested in. For this reason, we examine the multigrid coarse-grid operators in an upscaling context. If these operators adequately represent the phenomena of interest, then the only fine-scale computation necessary is the Galerkin product to form the first coarse-grid operator.

A key aim of the upscaling procedure, however, must be to preserve significant fine-scale influence in the coarse-scale operator. Variational multigrid methods can preserve this information in their variational approach, because a coarse-scale operator is a restricted version of the fine-scale operator. Consider the equivalent minimization form satisfied by the solution of $A\mathbf{x} = \mathbf{b}$ when A is symmetric and positive definite:

$$\mathbf{x} = \underset{\mathbf{y}\in\mathbb{R}^N}{\operatorname{argmin}} \left(\frac{1}{2}\langle A\mathbf{y}, \mathbf{y}\rangle - \langle \mathbf{b}, \mathbf{y}\rangle \right).$$

In a coarse-level correction step, we ask for the correction, $\mathbf{e}_c$, to the fine-grid approxi-

mation, $\tilde{\mathbf{x}}$, to satisfy

$$\begin{aligned}\mathbf{e}_c &= \underset{\mathbf{w}_c \in \mathbb{R}^{N_c}}{\operatorname{argmin}} \frac{1}{2}\langle A(\tilde{\mathbf{x}} + P\mathbf{w}_c), \tilde{\mathbf{x}} + P\mathbf{w}_c\rangle - \langle \mathbf{b}, \tilde{\mathbf{x}} + P\mathbf{w}_c\rangle \\ &= \underset{\mathbf{w}_c \in \mathbb{R}^{N_c}}{\operatorname{argmin}} \left(\frac{1}{2}\left(\langle A\tilde{\mathbf{x}}, \tilde{\mathbf{x}}\rangle - \langle \mathbf{b}, \tilde{\mathbf{x}}\rangle\right) + \left(\frac{1}{2}\langle P^T AP\mathbf{w}_c, \mathbf{w}_c\rangle - \langle P^T(\mathbf{b} - A\tilde{\mathbf{x}}), \mathbf{w}_c\rangle\right)\right).\end{aligned}$$

Thus, the optimal coarse-grid correction of form $P\mathbf{e}_c$ comes when $\mathbf{e}_c$ satisfies

$$P^T AP\mathbf{e}_c = P^T\mathbf{r}.$$

Using this choice for the coarse-scale operator, some properties of the fine-scale operator are preserved on the coarse scale. When P is itself chosen in an operator-induced fashion (as is the case in BoxMG and AMG), this coarse-scale operator reflects sufficient fine-scale information to be useful in an upscaling context.

Notice that the variational coarse-grid operator is still symmetric and positive (semi-)definite. If the fine-grid operator, A, has row sums of zero (as is the case in finite-element discretization of $-\nabla \cdot \mathcal{K}\nabla p$ with periodic or Neumann boundary conditions), $P^T AP$ will also have zero row sums if P preserves constant vectors. That is, if $A\mathbf{1} = \mathbf{0}$ and $P\mathbf{1}_c = \mathbf{1}$, then $P^T AP\mathbf{1}_c = \mathbf{0}$, so the sum of the elements in any row of $P^T AP$ must be zero.

4.3.1 Periodic Multigrid Upscaling

Moulton et al. [67] and Knapek [60] both consider the 2D problem and analyze the coarse-scale stencil as a symmetric, zero-row-sum, sparse operator to determine the effective coarse-scale permeability. In [67], this is done through a local flux analysis, whereas [60] uses a basis function approach, which we follow here.

Coarsening variationally, the form of the coarse-grid operator depends on the choices of coarse grids and interpolation, and both [60, 67] utilize the coarsening routines of BoxMG [34]. These routines assume a 9-point fine-grid operator from discretization on a rectangular mesh is given. Coarsening preserves the rectangular mesh structure and

interpolation is chosen so that the variational coarsening preserves the 9-point structure in all the coarse-grid operators (see Section 3.2). Additionally, interpolation has the property that $P\mathbf{1}_c = \mathbf{1}$.

Discretizing Equation 4.1 subject to periodic boundary conditions with the usual nodal bilinear finite elements, the coarse-scale matrices produced by BoxMG inherit many of the properties of the fine-scale discretization. Symmetry and positive semi-definiteness are preserved, as are the 9-point sparsity pattern and the zero row sums. In fact, these matrices have the same properties as a diffusion operator discretized on the coarse-scale.

The question of upscaling is then one of determining which continuum operator corresponds to the coarse-grid matrix. Under the assumption that the fine-scale system is upscaled to the point of homogeneity (that is $\mathcal{K}(\mathbf{x}) \equiv \hat{\mathcal{K}}$, a constant), this can be determined from the matrix entries. Assuming enough points exist on the coarse scale such that the operator remains a 9-point operator on the periodic domain, it can be represented in the basis of the 9 non-trivial bilinear finite element stencils: I, ∂_x, ∂_y, ∂_{xx}, ∂_{xy}, ∂_{yy}, ∂_{xxy}, ∂_{xyy}, and ∂_{xxyy}. The zero-row-sum property of the stencil requires that the coefficient of I be zero, since it is the only basis function with a non-zero row sum. Symmetry of the homogeneous operator requires that the East and West coefficients be equal, as well as the North and South, the Northwest and Southeast, and the Northeast and Southwest. These four conditions imply that the coefficients of ∂_x, ∂_y, ∂_{xxy}, and ∂_{xyy} are all zero as well. The remaining coefficients may then be solved for based on the stencil entries, resulting in the upscaled equation, $-\nabla \cdot \hat{\mathcal{K}} \nabla \hat{p} + \partial_{xy} \hat{E} \partial_{xy} \hat{p}$, with

$$\hat{\mathcal{K}} = \begin{bmatrix} \frac{h_x}{h_y}(S^E + S^{NE} + S^{NW}) & S^{NE} - S^{NW} \\ S^{NE} - S^{NW} & \frac{h_y}{h_x}(S^N + S^{NE} + S^{NW}) \end{bmatrix},$$

where the stencil notation is as in Figure 3.5. Although derived with a different approach, this expression for $\hat{\mathcal{K}}$ is in perfect agreement with the results of Moulton et al.

[67]. That the higher-order coefficient,

$$\hat{E} = -\frac{h_x h_y}{6}\left(S^N + S^E - S^{NE} - S^{NW}\right),$$

is non-zero reflects that the multigrid upscaling process is not directly aimed at determining the effective permeability. Rather, coarse-scale matrix equations are chosen such that their solution has a particular relevance to the fine-scale linear system. This additional term in the coarse-scale model can be thought of as a regularization term, introduced to maintain desirable properties of the fine-scale operator as it is coarsened, while still appropriately representing the upscaled behavior in which we are interested.

Using the work of Dvořák [41], Moulton et al. [66] demonstrated that the periodic boundary condition multigrid upscaling technique can be used to generate upper and lower bounds of the true homogenized permeability. In fact, the variational finite element setting naturally shows that the upscaling technique described here provides an upper bound on the effective permeability. A lower bound is possible through the solution of an auxiliary problem of similar form. While no such result has been obtained for the case of Neumann boundary conditions, the discussion of which follows, there is no obvious impediment to extending Dvořák's results to this case. It is useful in analysis of the numerical results in the following section to consider that the computed upscaled permeabilities may be upper bounds of the true homogenized permeability.

4.3.2 Neumann Boundary Conditions

Extending the periodic boundary condition multigrid upscaling technique to other boundary conditions is possible with a similar analysis. Any symmetric, positive semi-definite, zero-row-sum 9-point operator may, in fact, be considered as the weak form,

$$a_{ij} = \int (\hat{\mathcal{K}}\nabla\phi_j)\cdot(\nabla\phi_i) + \hat{E}(\partial_{xy}\phi_j)(\partial_{xy}\phi_i)d\Omega, \tag{4.15}$$

where ϕ_i and ϕ_j are nodal bilinear basis functions for the underlying grid. Here, $\hat{\mathcal{K}}$ and $\hat{E}$ are assumed to be piecewise constant on each coarse-scale element, but we do not

assume to have coarsened to the point of homogeneity in the stencil.

Computing the nodal stencils associated with the weak form in Equation 4.15, we again look to recover piecewise-constant values of $\hat{K}$ and $\hat{E}$ that generate the same stencil. Labeling the element Northeast of node (i,j) as element $(i+\frac{1}{2}, j+\frac{1}{2})$ and the permeability tensor in that element as

$$\hat{\mathcal{K}}^{(i+\frac{1}{2},j+\frac{1}{2})} = \begin{bmatrix} \hat{\mathcal{K}}_{11}^{i+\frac{1}{2},j+\frac{1}{2}} & \hat{\mathcal{K}}_{12}^{i+\frac{1}{2},j+\frac{1}{2}} \\ \hat{\mathcal{K}}_{12}^{i+\frac{1}{2},j+\frac{1}{2}} & \hat{\mathcal{K}}_{22}^{i+\frac{1}{2},j+\frac{1}{2}} \end{bmatrix},$$

we recover the relations that

$$\begin{aligned} \hat{\mathcal{K}}_{11}^{i+\frac{1}{2},j-\frac{1}{2}} + \hat{\mathcal{K}}_{11}^{i+\frac{1}{2},j+\frac{1}{2}} &= 2\frac{h_x}{h_y}\left(S_{i,j}^{E} + \frac{1}{2}\left(S_{i,j}^{NE} + S_{i+1,j}^{NW}\right) + \frac{1}{2}\left(S_{i,j-1}^{NE} + S_{i+1,j-1}^{NW}\right)\right), \\ \hat{\mathcal{K}}_{12}^{i+\frac{1}{2},j+\frac{1}{2}} &= S_{i,j}^{NE} - S_{i+1,j}^{NW}, \\ \hat{\mathcal{K}}_{22}^{i-\frac{1}{2},j+\frac{1}{2}} + \hat{\mathcal{K}}_{22}^{i+\frac{1}{2},j+\frac{1}{2}} &= 2\frac{h_y}{h_x}\left(S_{i,j}^{N} + \frac{1}{2}\left(S_{i,j}^{NE} + S_{i+1,j}^{NW}\right) + \frac{1}{2}\left(S_{i-1,j}^{NE} + S_{i,j}^{NW}\right)\right). \end{aligned}$$

Thus, a closed form expression for the off-diagonal coefficient, $\hat{\mathcal{K}}_{12}^{i+\frac{1}{2},j+\frac{1}{2}}$, is available, but recovery of the diagonal coefficients, $\hat{\mathcal{K}}_{11}^{i+\frac{1}{2},j+\frac{1}{2}}$ and $\hat{\mathcal{K}}_{22}^{i+\frac{1}{2},j+\frac{1}{2}}$, requires solution of a recurrence relationship, necessitating examination of the stencils near boundary cells.

It is here that the effect of the higher-order regularization term is most pronounced. In order for the strong form of Equation 4.15, $-\nabla \cdot \hat{\mathcal{K}}\nabla p + \partial_{xy}\hat{E}\partial_{xy}p = f$, to be well-posed, it requires two sets of boundary conditions. The natural boundary conditions (in the finite element sense) are Neumann, $\left(\hat{\mathcal{K}}\nabla p\right) \cdot \mathbf{n} = g_1$ on $\partial\Omega$ (where $\mathbf{n}$ is the unit normal vector to $\partial\Omega$), and a second-order condition, $\hat{E}\partial_{xy}p = g_2$ on $\partial\Omega$. These conditions are consistent with a symmetric, zero-row-sum boundary stencil, and so we use them to determine the initial conditions for the recurrence relationships for

$\hat{\mathcal{K}}_{11}$ and $\hat{\mathcal{K}}_{22}$ as

$$\begin{aligned}
\hat{\mathcal{K}}_{11}^{i+\frac{1}{2},\frac{1}{2}} &= 2\frac{h_x}{h_y}\left(S_{i,0}^{E} + \frac{1}{2}\left(S_{i,0}^{NE} + S_{i+1,0}^{NW}\right)\right), \\
\hat{\mathcal{K}}_{11}^{i+\frac{1}{2},n_y-\frac{1}{2}} &= 2\frac{h_x}{h_y}\left(S_{i,n_y}^{E} + \frac{1}{2}\left(S_{i,n_y-1}^{NE} + S_{i+1,n_y-1}^{NW}\right)\right), \\
\hat{\mathcal{K}}_{22}^{\frac{1}{2},j+\frac{1}{2}} &= 2\frac{h_y}{h_x}\left(S_{0,j}^{N} + \frac{1}{2}\left(S_{0,j}^{NE} + S_{1,j}^{NW}\right)\right), \\
\hat{\mathcal{K}}_{22}^{n_x-\frac{1}{2},j+\frac{1}{2}} &= 2\frac{h_y}{h_x}\left(S_{n_x,j}^{N} + \frac{1}{2}\left(S_{n_x-1,j}^{NE} + S_{n_x,j}^{NW}\right)\right).
\end{aligned}$$

We now compare this algorithm to both the periodic case and the homogenized coefficients, as computed by Bourgat, for the examples discussed in Section 4.1.2. A limitation of the BoxMG code restricts us from coarsening down to a single element (a grid of 2×2 nodal values), and so we consider a periodic tiling of these permeability distributions. To avoid potential complications from the boundaries, we consider a 3×3 periodic tiling of the permeability fields and measure the upscaled permeability in the center element after coarsening a $3 \cdot 2^{\ell} \times 3 \cdot 2^{\ell}$ element grid down to a 3×3 element grid (although, when the upscaled permeabilities are diagonal, all elements give the same permeability).

For the shape-variation problems (as in Figure 4.1), with permeability

$$\mathcal{K}(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in \Omega_0, \\ 10 & \mathbf{x} \in \Omega_1, \end{cases}$$

the Neumann boundary condition multigrid upscaling method performs the same as the periodic version. For a triangulation of the unit square with 2048 triangles, Bourgat reports homogenized permeabilities of 1.548 for the square, 1.573 for the lozenge shape (rotated square), and 1.516 for the disk. Interestingly, while there is some shape-dependent variation in the permeability, that dependence is also rather weak. Using our multigrid upscaling method, we recover upscaled permeabilities of 1.598 for the square, 1.608 for the lozenge, and 1.563 for the disk, discretizing each problem on a 768×768 element rectangular fine grid and coarsening to a 3×3 element coarse grid. Refinement

of the grid produces little change in these results, with upscaled permeabilities of 1.598 for the square, 1.612 for the lozenge, and 1.563 for the disk on a 1536×1536 element fine grid.

An interesting advantage of this approach is that we may also calculate $\hat{E}^{i+\frac{1}{2},j+\frac{1}{2}}$, for each element, $(i+\frac{1}{2}, j+\frac{1}{2})$, from the matrix coefficients and the recovered tensor, $\hat{\mathcal{K}}$. While relations such as those for $\hat{\mathcal{K}}$ could be derived to express $\hat{E}$, a simpler approach is to use the recovered values of $\hat{\mathcal{K}}^{i+\frac{1}{2},j+\frac{1}{2}}$, as in

$$\hat{E}^{i+\frac{1}{2},j+\frac{1}{2}} = h_x h_y \left(S_{i,j}^{NE} + \frac{1}{6}\frac{h_y}{h_x}\hat{\mathcal{K}}_{11}^{i+\frac{1}{2},j+\frac{1}{2}} + \frac{1}{6}\frac{h_x}{h_y}\hat{\mathcal{K}}_{22}^{i+\frac{1}{2},j+\frac{1}{2}} - \frac{1}{2}\hat{\mathcal{K}}_{12}^{i+\frac{1}{2},j+\frac{1}{2}} \right).$$

Computing $\hat{E}$ in the center element of the 3×3 element coarse grid, we recover $-0.0708 h_x h_y$ for the square, $-0.097 h_x h_y$ for the lozenge, and$-0.0829 h_x h_y$ for the disk when coarsening from a 768×768 element fine grid. For a 1536×1536 element fine grid, the recovered values of $\hat{E}$ are $-0.0708 h_x h_y$ for the square, $-0.0976 h_x h_y$ for the lozenge, and $-0.0827 h_x h_y$ for the disk. While these numbers appear small when considering h_x and h_y of size $\frac{1}{768}$ or $\frac{1}{1536}$, the stencil of the fourth-order operator, $\partial_{xy}\hat{E}\partial_{xy}$, has a natural scaling of $\frac{1}{h_x h_y}$ relative to that of the second-order operator, $-\nabla \cdot \hat{\mathcal{K}} \nabla$, and, thus, these small coefficients still realize a significant change in the discrete operator. That these coefficients are not converging to zero indicates the multigrid coarse-grid operators from which $\hat{\mathcal{K}}$ and $\hat{E}$ are recovered are not equivalent to classical homogenization.

The simple geometry in Figure 2.1 allows analysis of the effects of variation in a scalar, but discontinuous permeability,

$$\mathcal{K}(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in \Omega_0, \\ \lambda & \mathbf{x} \in \Omega_1, \end{cases}$$

where λ is allowed to vary between 0 (corresponding to an impermeable material) and ∞ (corresponding to a perfect conductor). Figure 4.3 shows the variation in the upscaled permeability computed with the Neumann BC multigrid upscaling method, as compared with homogenization results like Bourgat's (computed using an implementation of the

periodic Laplacian method, as in Section 4.2.3, equivalent to Bourgat's computations (Theorem 2)), as well as the arithmetic and harmonic means that we know to bound the effective permeability [28].

Notice that both the upscaled and homogenized permeabilities do lie between the arithmetic and harmonic means. Also, the upscaled permeability (which is identical to that computed in the periodic boundary condition case) tracks very well with the homogenized permeability. Here, the upscaled permeability is computed on a 768×768 mesh, with a 3×3 tiling of the geometry of Figure 2.1. The homogenized permeability, for comparison, was computed on a 256×256 mesh consisting of just the domain in Figure 2.1, without any periodic extension.

While both of these cases have scalar upscaled permeabilities, the L-shaped domain in Figure 4.2 violates the necessary symmetry, and so has a homogenized permeability given by the full tensor,

$$\hat{\mathcal{K}} = \begin{bmatrix} 1.915 & -0.101 \\ -0.101 & 1.915 \end{bmatrix},$$

when the piecewise constant permeability is specified as in Equation 4.10. Noting that any symmetric 2×2 matrix, A, with $a_{11} = a_{22}$ can be diagonalized by $Q = \frac{1}{\sqrt{2}} \left[\begin{smallmatrix} -1 & 1 \\ 1 & 1 \end{smallmatrix} \right]$, we see that the homogenized permeability, $\hat{\mathcal{K}}$, represents anisotropic diffusion aligned with the principal axes obtained by a rotation of 45° from the usual coordinate system,

$$\hat{\mathcal{K}} = \begin{bmatrix} 1.915 & -0.101 \\ -0.101 & 1.915 \end{bmatrix} = Q \begin{bmatrix} 2.016 & 0 \\ 0 & 1.814 \end{bmatrix} Q^T.$$

The Neumann BC multigrid upscaling method also generates tensors that can be diagonalized by Q, and so, in Table 4.1, we report the entries of the upscaled permeability, $\hat{\mathcal{K}}$, as well as those of the diagonalized tensor, $Q^T \hat{\mathcal{K}} Q$. Notice here that the multigrid upscaling method converges to a solution that is near to, but slightly different from that reported by Bourgat [7] in scaling, but that exactly matches the principal

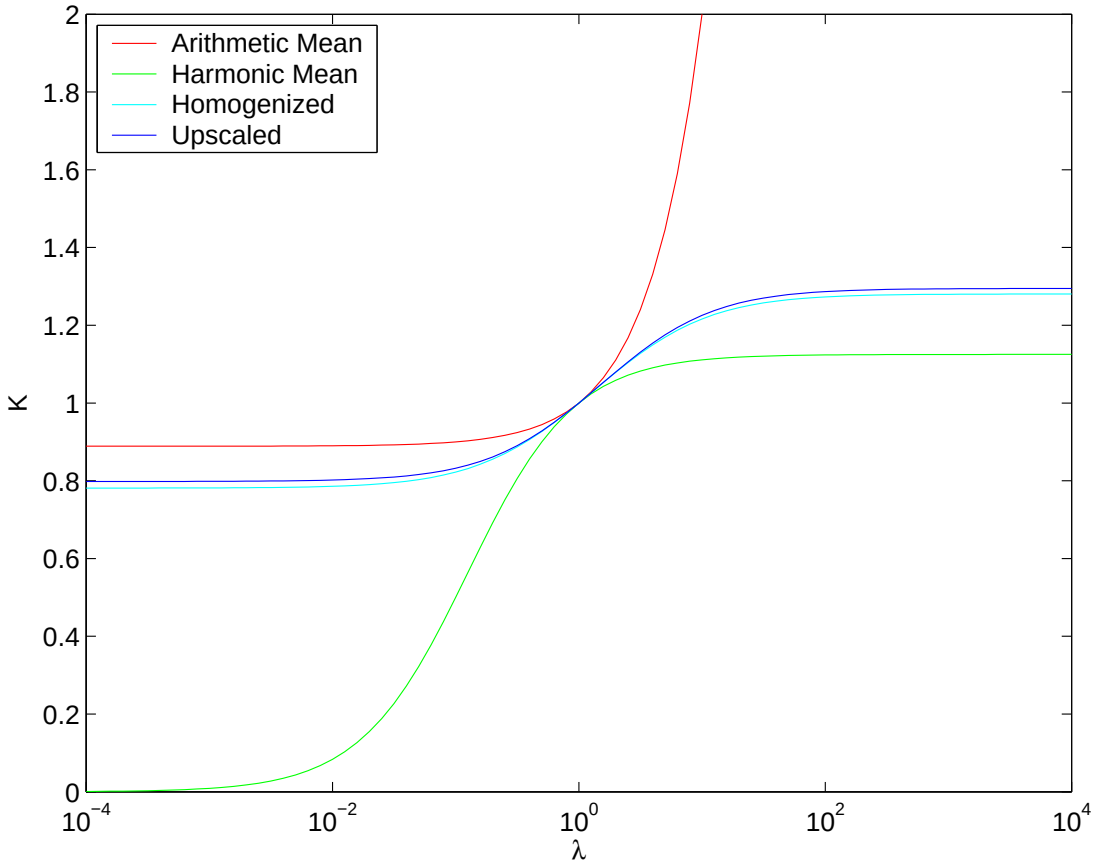

Figure 4.3: Predicted Effective Permeability vs. Jump in Permeability

Fine grid size	$\hat{K}_{11} = \hat{K}_{22}$	$\hat{K}_{12} = \hat{K}_{21}$	$(Q^T \hat{K} Q)_{11}$	$(Q^T \hat{K} Q)_{22}$
12×12	1.5649	-0.0853	1.6501	1.4796
24×24	2.0864	-0.1760	2.2624	1.9103
48×48	1.8619	-0.1401	2.0021	1.7218
96×96	1.9719	-0.1588	2.1307	1.8131
192×192	1.9207	-0.1509	2.0716	1.7697
384×384	1.9467	-0.1552	2.1019	1.7916
768×768	1.9339	-0.1532	2.0870	1.7807
1536×1536	1.9403	-0.1542	2.0945	1.7861
3072×3072	1.9371	-0.1537	2.0908	1.7834

Table 4.1: Upscaled Permeability and Diagonalized Tensor for L-shaped Domain

axes of diffusion. Similar behavior is seen in the periodic upscaling case [67], where the principal axes are again recovered exactly. Our Neumann BC upscaling method predicts the same off-diagonal coefficient, but more accurate diagonal coefficients than the periodic upscaling method.

One simple problem for which the Neumann BC multigrid upscaling method yields unsatisfying results is that of an essentially one-dimensional permeability distribution, such as the striped problem in Figure 4.4(a), with permeability given by

$$\mathcal{K}(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in \Omega_0, \\ 10^{-2} & \mathbf{x} \in \Omega_1. \end{cases}$$

The one-dimensional nature of this problem allows the homogenized permeability to be easily calculated as diagonal, with the harmonic mean in the y-direction (across the stripes) and the arithmetic mean in the x-direction (along the stripes). Even on a very coarse grid (as coarse as 6×6 elements, so that the variations in $\mathcal{K}$ are just represented), the Neumann BC multigrid upscaling technique accurately computes the harmonic mean for the y-component. Values for the x-component, however, vary across the coarse-scale elements, alternating between 10^{-2} and 1, regardless of the fine-grid mesh size. Rotating this example by 90°, to that in Figure 4.4(b), shows the same behavior: the x-component of the upscaled permeability is computed correctly as the

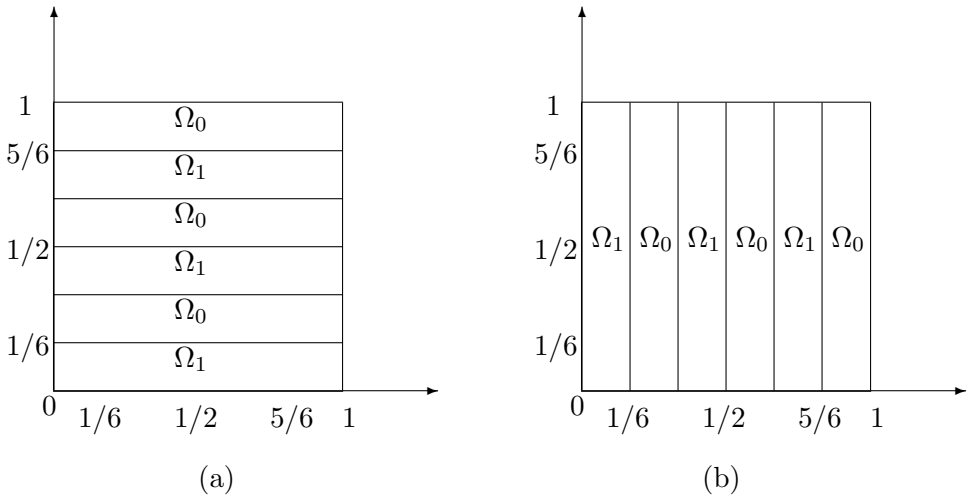

Figure 4.4: Striped Permeability Problem

harmonic mean, but the y-component alternates between the two fine-scale permeability values instead of taking the value of the arithmetic mean. The recovered values of $\hat{E}$ also demonstrate this alternating behavior, taking value $-0.001634 h_x h_y$ when the stripe-aligned $\hat{\mathcal{K}}$ is predicted as 10^{-2}, and $0.1634 h_x h_y$ when the stripe-aligned $\hat{\mathcal{K}}$ value is predicted as 1. It is an open question as to whether this is an effect of the recovery formula used, the boundary coarsening in the BoxMG code, or a combination of the two. These results compare unfavorably with the periodic BC multigrid upscaling method which is exact for this problem [67], and are also not clear upper bounds for the effective permeability, as suggested by the work of Dvořák [41], suggesting that further analysis of this problem may be of interest.

An interesting observation about this problem is that if the correct value was picked up from the starting condition,

$$\hat{\mathcal{K}}_{11}^{i+\frac{1}{2},\frac{1}{2}} = 2\frac{h_x}{h_y}\left(S_{i,0}^{E} + \frac{1}{2}\left(S_{i,0}^{NE} + S_{i+1,0}^{NW}\right)\right),$$

then the remaining cells would correctly calculate the arithmetic average. We thus believe the difficulty in this case to be caused by the coarse-scale boundary conditions. That is, it is possible that the coarsening used by BoxMG does not accurately produce

the natural boundary conditions assumed on the coarse-scale problem for our upscaled permeability recovery. In fact, the precise details of the definition of interpolation in BoxMG are somewhat more complicated than those presented in Section 3.2, and so it is possible that some of the additional checks in the code do cause us some difficulty in our recovery algorithm. These issues are the subject of current research.

4.3.3 Three-Dimensional Upscaling

A similar analysis can be performed in three dimensions, with analogous results. The 27-point operator of a trilinear finite element discretization on a tensor-product mesh may be considered as a linear combination of the 27 non-trivial trilinear FE stencils. Assuming a zero-row-sum operator, the coefficient of the term corresponding to the identity operator must again be zero. The symmetry condition can be used to eliminate all stencils of first, third, and fifth order derivatives, leaving only the second, fourth, and sixth order derivatives whose trilinear FE stencils are nontrivial in the expansion. In the case of a fully upscaled problem (such as with periodic boundary conditions), the upscaled 3×3 tensor, $\hat{\mathcal{K}}$, can be directly recovered as

$$
\begin{aligned}
\hat{\mathcal{K}}_{11} &= \frac{-h_x}{h_y h_z}\left(S^E + S^{UW} + S^{UE} + S^{NW} + S^{NE} + S^{UNW} + S^{UNE} + S^{USE} + S^{USW}\right), \\
\hat{\mathcal{K}}_{22} &= \frac{-h_y}{h_x h_z}\left(S^N + S^{UN} + S^{US} + S^{NW} + S^{NE} + S^{UNW} + S^{UNE} + S^{USE} + S^{USW}\right), \\
\hat{\mathcal{K}}_{33} &= \frac{-h_z}{h_x h_y}\left(S^U + S^{UN} + S^{US} + S^{UW} + S^{UE} + S^{UNW} + S^{UNE} + S^{USE} + S^{USW}\right), \\
\hat{\mathcal{K}}_{12} &= \frac{-1}{h_z}\left(S^{NE} - S^{NW} - S^{UNW} + S^{UNE} - S^{USE} + S^{USW}\right), \\
\hat{\mathcal{K}}_{13} &= \frac{-1}{h_y}\left(S^{UE} - S^{UW} - S^{UNW} + S^{UNE} + S^{USE} - S^{USW}\right), \\
\hat{\mathcal{K}}_{23} &= \frac{-1}{h_x}\left(S^{UN} - S^{US} + S^{UNW} + S^{UNE} - S^{USE} - S^{USW}\right),
\end{aligned}
$$

plus some higher-order terms. Here, the stencil entries, S, are not assumed to be negative, and so we have not reversed their natural signs as we did in the 2D case.

In the case of full Neumann boundary conditions, we again cannot expect to be able to upscale to the point of homogeneity in the matrix. Thus, given a symmetric, 27-point, zero-row-sum coarse-scale operator, we interpret it as a discretization of the symmetric, positive semi-definite PDE,

$$-\nabla \cdot \hat{\mathcal{K}} \nabla + [\partial_{xy}, \partial_{xz}, \partial_{yz}] \, \hat{E} \begin{bmatrix} \partial_{xy} \\ \partial_{xz} \\ \partial_{yz} \end{bmatrix} - \partial_{xyz} \hat{E}^{(2)} \partial_{xyz},$$

where $\hat{\mathcal{K}}$ and $\hat{E}$ are symmetric, 3×3 coefficient matrices, and $\hat{\mathcal{K}}$, $\hat{E}$ and $\hat{E}^{(2)}$ are taken to be piecewise constant over each element of the coarse mesh.

Considering the cases of the off-diagonal coefficients, for example, the sum for $\hat{\mathcal{K}}_{12}^{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}$ from the periodic case can be interpreted locally. Suppose the matrix entries in that sum correspond to edges passing through the elements Northwest of the node (i,j,k) (both above and below this node), we derive

$$\begin{aligned} S_{i,j,k}^{NE} - S_{i+1,j,k}^{NW} - S_{i+1,j,k}^{UNW} + S_{i,j,k}^{UNE} - S_{i,j+1,k-1}^{USE} + S_{i+1,j+1,k-1}^{USW} \\ = -\frac{h_z}{2} \hat{\mathcal{K}}_{12}^{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} - \frac{h_z}{2} \hat{\mathcal{K}}_{12}^{i+\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}}. \end{aligned}$$

Note that this is consistent with the periodic case, and that if we consider the natural boundary conditions for $k = 0$, we find the starting condition,

$$S_{i,j,0}^{NE} - S_{i+1,j,0}^{NW} - S_{i+1,j,0}^{UNW} + S_{i,j,0}^{UNE} = -\frac{h_z}{2} \hat{\mathcal{K}}_{12}^{i+\frac{1}{2},j+\frac{1}{2},\frac{1}{2}}.$$

A similar condition is found using the natural BCs, when $k = n_z - 1$. Similarly, we may derive expressions for the other off-diagonal elements of $\mathcal{K}$,

$$\begin{aligned} S_{i,j,k}^{UE} - S_{i+1,j,k}^{UW} - S_{i+1,j,k}^{UNW} + S_{i,j,k}^{UNE} + S_{i,j,k}^{USE} + S_{i+1,j,k}^{USW} \\ = -\frac{h_y}{2} \hat{\mathcal{K}}_{13}^{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} - \frac{h_y}{2} \hat{\mathcal{K}}_{13}^{i+\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}}, \\ S_{i,j,k}^{UN} - S_{i,j+1,k}^{US} + S_{i+1,j,k}^{UNW} + S_{i-1,j,k}^{UNE} - S_{i-1,j+1,k}^{USE} - S_{i+1,j+1,k}^{USW} \\ = -\frac{h_x}{2} \hat{\mathcal{K}}_{23}^{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} - \frac{h_x}{2} \hat{\mathcal{K}}_{23}^{i-\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}, \end{aligned}$$

with boundary conditions,

$$S^{UE}_{i,0,k} - S^{UW}_{i+1,0,k} - S^{UNW}_{i+1,0,k} + S^{UNE}_{i,0,k} = -\frac{h_y}{2}\hat{\mathcal{K}}^{i+\frac{1}{2},\frac{1}{2},k+\frac{1}{2}}_{13},$$
$$S^{UN}_{0,j,k} - S^{US}_{0,j+1,k} + S^{UNW}_{1,j,k} - S^{USW}_{1,j+1,k} = -\frac{h_x}{2}\hat{\mathcal{K}}^{\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}_{23}.$$

Whether such a matching technique can recover the diagonal entries of $\mathcal{K}$ is still unresolved. We expect that it is possible; however, the matching would need to take four adjacent elements into account, and so it is much more difficult to obtain the necessary balances to cleanly recover the coefficients of interest.

The extension of the periodic multigrid upscaling method to Neumann boundary conditions in two and three dimensions allow treatment of the underlying finite element discretizations with their natural boundary conditions. Interpreting the coarse-scale operator as a bilinear discretization again produces higher-order terms, not present in the fine-scale discretization of the $-\nabla \cdot \mathcal{K}(\mathbf{x})\nabla$ operator. From a multigrid point of view, these terms produce a regularization effect, allowing efficient multigrid treatment of many fine-scale operators whose upscaled characteristics alone introduce additional complexity that would require sophisticated handling.

Consider, for example, the geometry shown in Figure 4.4. The homogenized behavior of this operator is anisotropic, more so as the ratio between the permeability in Ω_0 and that in Ω_1 increases. A direct discretization on coarse scales of such an anisotropic problem would yield poor performance from a multigrid scheme using full coarsening and pointwise relaxation because of the poor smoothing properties of the pointwise relaxation, typically leading to the choice of either semicoarsening multigrid or some form of line (or block) relaxation. Yet, when the striped configuration exists on the fine-grid, pointwise relaxation has no difficulty resolving an appropriate correction on full-coarsening based coarse grids. This is because the Galerkin coarsening used produces these higher-order terms that cancel out the effect of the anisotropy while yielding a relevant coarse-grid problem. The size of $\hat{E}$ appears to directly reflect this

behavior, being largest when the upscaled coefficient is least accurate. The sign of $\hat{E}$ typically also provides diagnostic information, with negative sign when the computed permeability is greater than the homogenized permeability.

4.4 Coarse-Scale Models

An alternative to these approaches of generating upscaled effective permeability data and then reposing the problem on the coarse scale is to consider an appropriate basis for the coarse-grid problem. While the finite-element formulations discussed above are all based on low-order (piecewise-linear) elements, there is no need to consider such simple basis functions, especially given the desire to capture fine-scale fluctuations of the permeability on some coarse grid. A more efficient approach is to consider a discretization on a computationally feasible grid, with basis functions chosen to capture the fine-scale variations in permeability that we are interested in.

The multiscale finite element method (MSFEM) [56, 57] provides one way of generating such a basis. For a given partitioning of the domain, Ω, into subdomains, nodal basis functions are constructed by solving the homogeneous flow through porous media problem,

$$-\nabla \cdot \mathcal{K}(\mathbf{x}) \nabla \phi_i(\mathbf{x}) = 0,$$

over each subdomain neighboring the coarse-grid node, i. Boundary conditions for these problems are determined based on the need for local support of the coarse-scale basis functions, and on the fine-scale approximation sought.

In [56], these boundary conditions are posed based on the fine-scale structure in the subdomain. We seek a nodal basis, so the basis function associated with point i should have value 1 at that node and value 0 at all others. For each element adjacent to i, the boundary conditions along the edges not adjacent to i are simply taken as homogeneous Dirichlet conditions. For the edges adjacent to i, the flow in porous media

problem is restricted to the one-dimensional edge (by projecting $\mathcal{K}$ onto that edge), and the solution of this problem is used to determine the appropriate boundary data for the subdomain problem. These two-point boundary-value problems have Dirichlet boundary conditions of value 1 at point i and 0 at the opposite endpoint, and can be solved analytically. In this manner, the subdomain problem may be posed with full Dirichlet boundary data, and the set of nodal elements may be created. That such a basis may provide a good approximation to the fine-scale solution is demonstrated in [57].

Creating the basis in this manner, however, has a similar cost as the Laplacian techniques discussed above. For each coarse-scale element, a fine-scale PDE must be solved on this subdomain for each node on the boundary of the coarse-scale element. Thus, for a quadrilateral coarse mesh, four fine-scale problems must be solved to compute each coarse-scale basis function. In practice, a resonance effect was found between the small-scale variation in $\mathcal{K}$ and the fine-scale mesh size, requiring an oversampling technique that further increases this cost [56].

The techniques of wavelets and multiresolution analysis can also be applied to derive an appropriately averaged operator [6, 48]. Gilbert [48] develops the upscaling approach in the case of a one-dimensional flow problem where it can easily be related to the classical theory. The multiresolution scheme is based on a combination of upscaling (to reduce the fine-scale equation to a suitably coarse scale) followed by augmentation (to arrive at an equivalent fine-scale operator). This upscaling algorithm follows the usual Haar basis multiresolution technique of writing the fine-scale variables as averages and differences and then reducing the fine-scale equations to a system in only the averages. This reduction can be viewed as using a Schur complement to derive the coarse-scale equations, and may be applied recursively until an operator on a suitably coarse scale is derived. This coarsest-scale operator may then be augmented to the original fine scale by considering the reduction process in reverse, with the ansatz of a homogeneous

fine-scale operator.

While Gilbert [48] compares results for the one-dimensional problem with the classical theory, this approach does not easily generalize to multiple dimensions due to the denseness of the Schur complements in more than one dimension. That is, when the two-dimensional (or, in general, multi-dimensional) Haar basis is used to discretize the flow in porous media problem, the differences-differences block of the operator (analogous to the block of F to F connections, A_{ff}, in a multigrid context) is banded, but has a full inverse, leading to a dense system for the averages when this block is eliminated through a Schur complement. Beylkin and Coult [6] alleviate this difficulty by using higher order wavelets (with more vanishing moments) in the discretization, resulting in control of the sparsity of the reduced operator. The multiresolution LU approach [49] used to compute the reduced operator is, in fact, quite similar to the coarsening used in an algebraic multigrid method.

4.4.1 Multigrid Coarse-Scale Bases

In Section 4.3, we considered the coarse-scale operators constructed by a variational multigrid method, BoxMG, as bilinear (in 2D and trilinear in 3D) discretizations of the coarse-scale PDE,

$$-\nabla \cdot \hat{\mathcal{K}} \nabla p(\mathbf{x}) + \partial_{xy} \hat{E} \partial_{xy} p(\mathbf{x}) = Q(\mathbf{x}).$$

While appropriate for the upscaling framework considered there, we can also consider these coarse-scale operators as discretizations of the fine-scale PDE, $-\nabla \cdot \mathcal{K}(\mathbf{x}) \nabla p(\mathbf{x})$, using multiscale finite-element basis functions.

Consider the fine-grid operator, A, to come from discretizing $-\nabla \cdot \mathcal{K}(\mathbf{x}) \nabla p(\mathbf{x})$ via bilinear finite elements on a known meshing of $\Omega = [0,1]^2$. Thus,

$$a_{ij} = \mathbf{e}_j^T A \mathbf{e}_i = \int_\Omega \langle \mathcal{K}(\mathbf{x}) \nabla \phi_i, \nabla \phi_j \rangle d\Omega,$$

where $\mathbf{e}_i$ is the i^{th} canonical unit vector and ϕ_i is the standard, nodal finite-element basis function associated with node i.

Within the variational formulation, each multigrid coarse-grid operator can be written as P^TAP, where P represents the accumulation of interpolation operators from a given coarse scale to the finest grid. In other words, if we consider the coarse grid to be grid k (after $k-1$ coarsenings), then $P = P_k^{k-1} \cdots P_3^2 P_2^1$. The coarse-grid operator is then also a finite-element discretization of the fine-scale PDE,

$$\begin{aligned}
(P^TAP)_{ij} &= \sum_{k,l} p_{kj} p_{li} \int_\Omega \langle \mathcal{K}(\mathbf{x}) \nabla \phi_l, \nabla \phi_k \rangle d\Omega \\
&= \int_\Omega \left\langle \mathcal{K}(\mathbf{x}) \nabla \left(\sum_l p_{li} \phi_l \right), \nabla \left(\sum_k p_{kj} \phi_k \right) \right\rangle d\Omega \\
&= \int_\Omega \langle \mathcal{K}(\mathbf{x}) \nabla \hat{\phi}_i, \nabla \hat{\phi}_j \rangle d\Omega,
\end{aligned}$$

where $\{\hat{\phi}_i\}$ are new basis functions associated with the nodes of the fine grid that exist on the coarse grid [60].

That these are, in fact, nodal basis functions is not difficult to see. Considering multigrid methods where the coarse grid is a subset of the fine grid (such as BoxMG and AMG), interpolation is of the form $P = \left[\begin{smallmatrix} W \\ I \end{smallmatrix} \right]$, where W represents the interpolation from C to the fine-grid points, and the identity matrix is used to take coarse-grid values to their nodal fine-grid equivalents. So, the nodal values of $\hat{\phi}_i = \sum_l p_{li} \phi_l$ are simply the values of Pe_i. At node l, this is simply $(Pe_i)_l = p_{li}$ and so, if $l \in C$, then $p_{li} = \delta_{li}$, i.e., $\hat{\phi}_i(\mathbf{x}_l) = \delta_{li}$ for $i, l \in C$, exactly as is required for nodal basis functions.

This viewpoint also allows us to visualize the coarse-scale basis functions and see how fine-scale information is integrated into the coarse-scale operator. The coarse-scale basis function, $\hat{\phi}_i = \sum_l p_{li} \phi_l$, is a fine-grid function that is piecewise bilinear (as it is the sum of piecewise bilinear, fine-grid basis functions) and takes the nodal values $\hat{\phi}_i(\mathbf{x}_l) = p_{li}$. Thus, it is the interpolant of the coarse-scale function whose finite-element representation is the canonical, coarse-scale unit vector, $\mathbf{e}_i$, which is simply the coarse-

scale nodal basis function, ϕ_i. So, we may generate the coarse-scale basis function associated with node i by interpolating the coarse-grid vector, $\mathbf{e}_i$, up to the fine grid.

We first consider the case of a geometrically-regular permeability field, as pictured in Figure 4.5, representing a periodic sand/shale structure. In this example, the black regions represent zones of high permeability, $\mathcal{K} = 1000$, against the background media with $\mathcal{K} = 1$. The periodic pattern is a 4×4 tiling of the unit cell with an inclusion of high permeability on $[\frac{5}{16}, \frac{11}{16}]^2$, scaled back onto the unit cell. The fine scale for this problem is $h = \frac{1}{64}$, and so this permeability field is represented exactly on the fine scale, but not on the coarse scale, with $H = \frac{1}{4}$. Coarse-scale basis functions for this field are shown in Figure 4.6. These plots represent the evolution of the coarse-scale basis function as finer-scale information is added in. Thus, the surface in the upper left corner shows a bilinear function on the coarse mesh. In the upper right surface, information from interpolation to the 8×8 mesh is included, and we can see that the basis function already has significantly different character than the original bilinear hat function. As the basis function is interpolated to the 16×16 and 32×32 grids on the bottom of Figure 4.6, we see even more fine-scale structure appear. The finest-scale basis function, shown in Figure 4.7, distinctly shows the effect of the high-permeability inclusions and that this basis function accurately represents the fine-scale structure of the problem. As regions of high permeability typically result in low pressure gradients, we see that this basis function is ideally suited to resolving the expected features of a fine-scale flow over this field.

The geostatistical fields of Figures 2.3 and 2.4 provide more complex fine-scale structures to capture in the coarse-scale basis functions. For the mildly-layered field (Figure 2.3), the basis function associated with the node at $(\frac{1}{2}, \frac{1}{2})$ is located in a region of relatively high permeability, and thus does not exhibit the same dramatic behavior as that for the periodic problem above. Figure 4.8 shows the evolution of the basis function from the standard bilinear on the 4×4 grid up to the 32×32 grid, where we

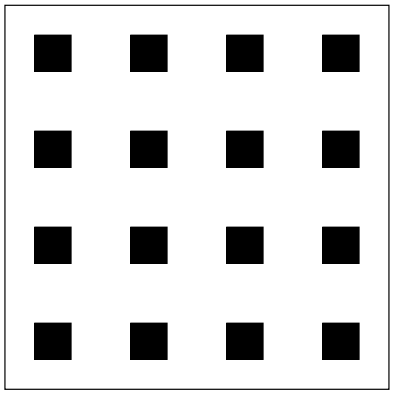

Figure 4.5: Periodic Sand/Shale Permeability Field

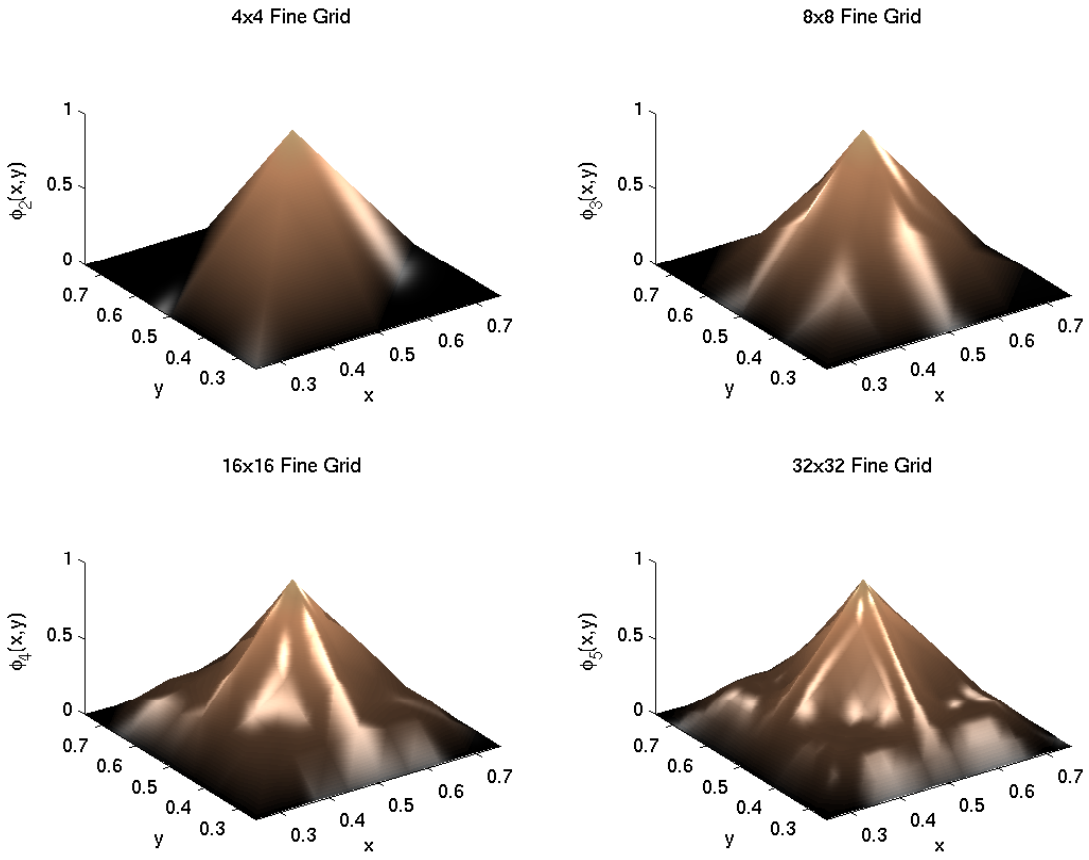

Figure 4.6: Coarse-Scale Basis Functions for Periodic Sand/Shale Problem

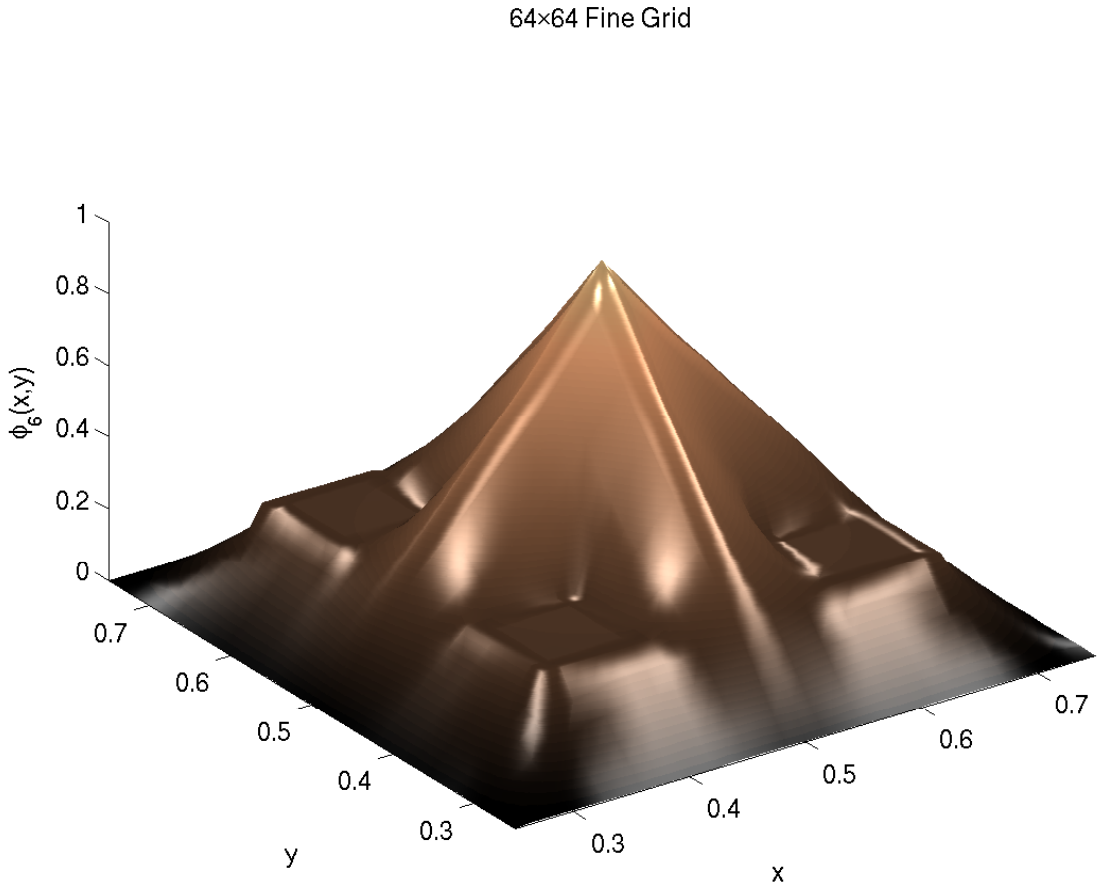

Figure 4.7: Fine-Scale Basis Function for Periodic Sand/Shale Problem

still see significant influence of the (random) fine-scale structure. Figure 4.9 shows the full multiscale basis function as represented on the original fine grid. Here, we can see the effects of the layering in fine-scale structure of the basis function that is primarily aligned with the x-axis. Notice also the change of character in the basis function near $y = 0.25$, corresponding to the lower permeability there.

The effects of the fine-scale heterogeneity in the strongly-layered media of Figure 2.4 are also well-represented within the multiscale basis function. On coarser levels, as in Figure 4.10, we see that the bilinear basis function of the 4×4 mesh begins to align with the layering of the flow already on the 8×8 grid and reflects noticeable fine-scale structure on the 32×32 grid. The full multiscale basis function, shown in Figure 4.11, shows the macroscopic quality of aligning with the layering as well as representing the fine-scale structure of $\mathcal{K}(\mathbf{x})$.

We begin to see a natural comparison between our method and the multiscale finite element method (MSFEM) [56]. In the MSFEM, coarse-scale basis functions can also be expressed as sums of the fine-scale basis functions, although they are determined as solutions of fine-scale problems on the coarse-scale elements. Our method is, thus, significantly less computationally expensive than the MSFEM, as it involves only the computation of interpolation and coarse-grid operators, as opposed to the complete solution of many fine-scale problems.

4.4.2 Multigrid Coarse-Scale Models

The multiscale information contained in the coarse-scale operator suggests that we should consider its use as an approximation to both the fine-scale operator and the continuum-scale PDE. In particular, we ask whether a solution of the coarse-scale problem provides accurate and relevant approximations to certain macroscopic properties of the flow, such as maxima and minima of the pressure field and the Darcy-law flow rate. Taking the fine-scale matrix as a starting point, we compare our technique to

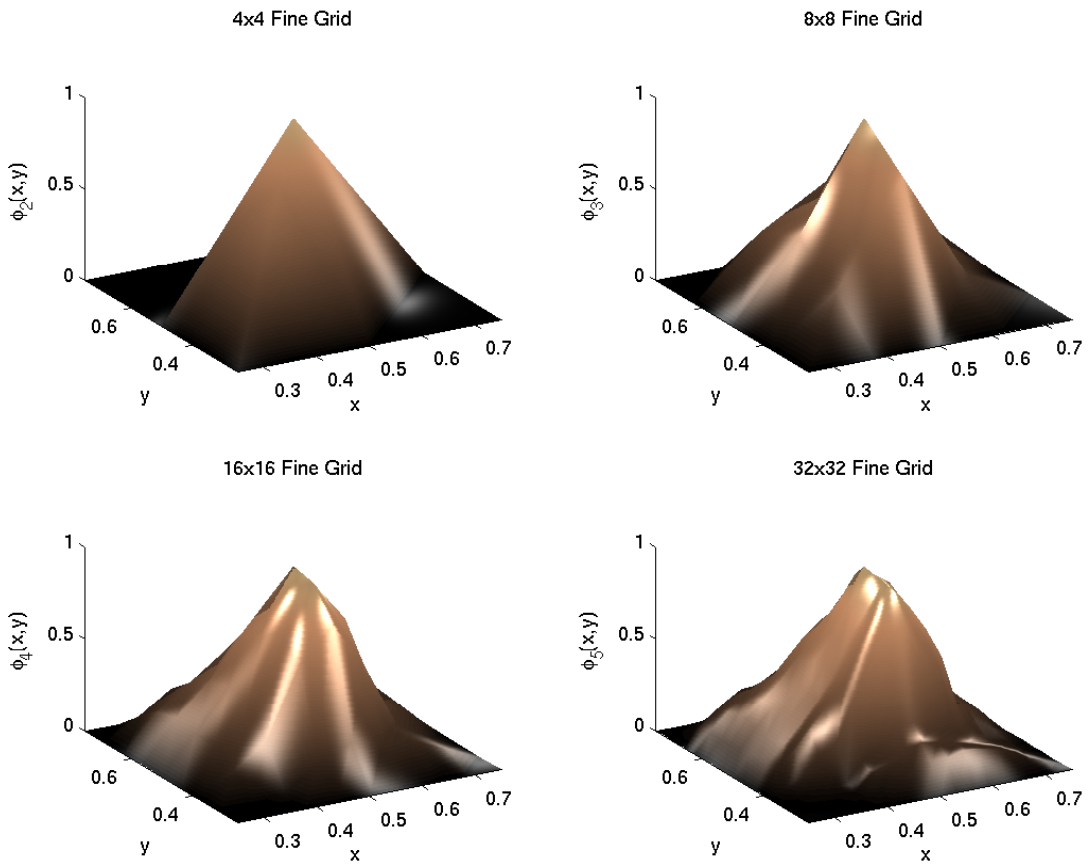

Figure 4.8: Coarse-Scale Basis Functions for Mildly-Layered Geostatistical Field

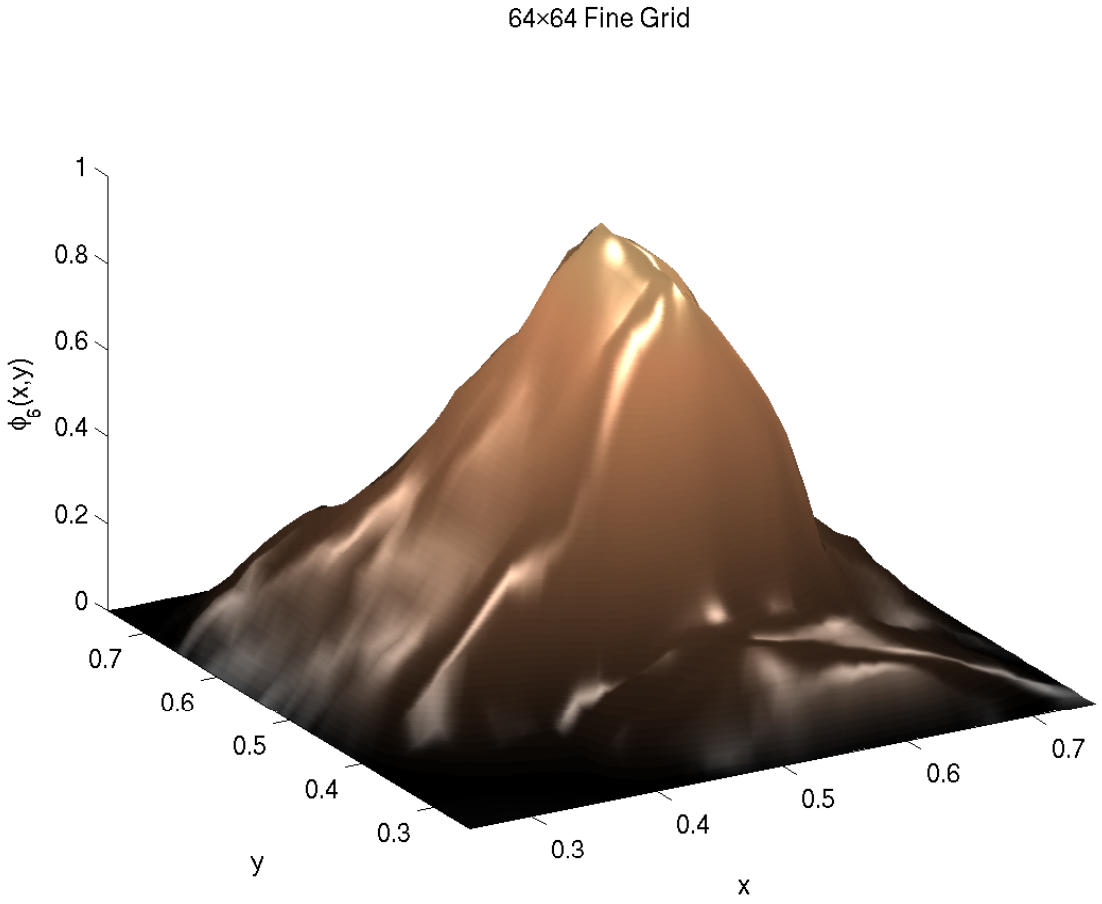

Figure 4.9: Fine-Scale Basis Function for Mildly-Layered Geostatistical Field

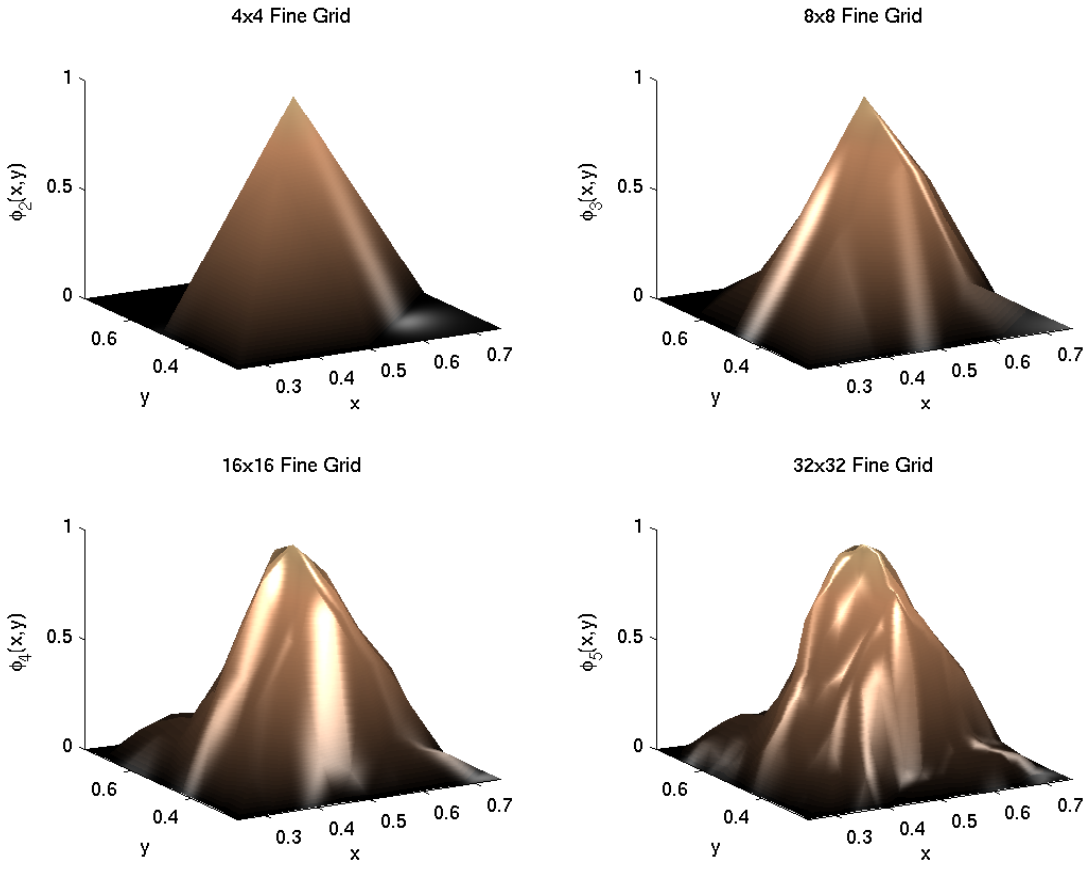

Figure 4.10: Coarse-Scale Basis Functions for Strongly-Layered Geostatistical Field

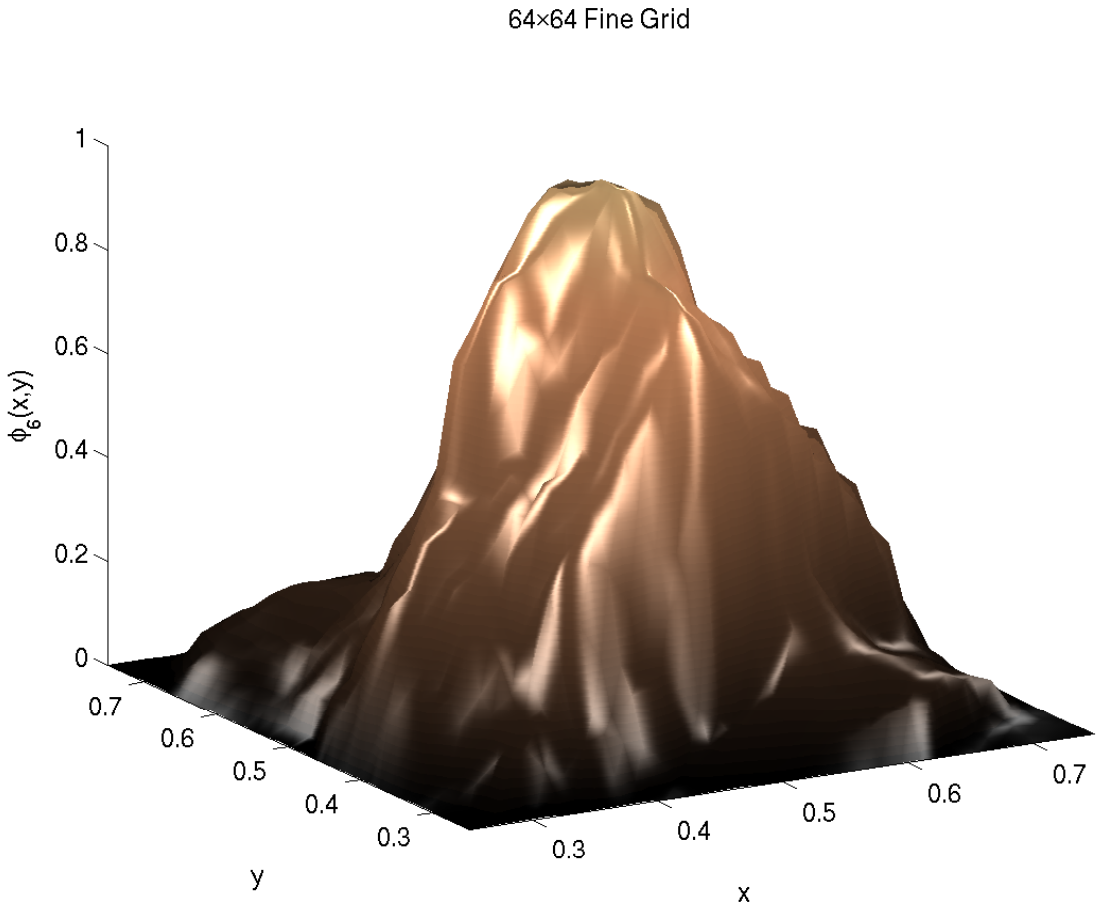

Figure 4.11: Fine-Scale Basis Function for Strongly-Layered Geostatistical Field

the results obtained by other upscaling techniques, noting that using a robust multigrid scheme to compute the coarse-scale operators requires much less work than solving a single problem over each subdivision of the domain.

We derive a coarse-scale model by first discretizing Equations 2.1 and 2.2 on a scale fine enough to resolve the variations in $\mathcal{K}$, using bilinear finite elements and Neumann (natural) boundary conditions. The fine-scale matrix is then coarsened to a given coarse (or computational) scale using a variational multigrid formulation, as in BoxMG and AMG. Care must be taken to ensure sufficient representation of the fine-scale boundary on the coarse mesh. To avoid overcoarsening near Dirichlet boundaries (as can occur in correction-based multigrid schemes where little or no correction is needed near the diagonally-dominant Dirichlet boundaries), we coarsen the Neumann problem so that all rows of the fine-scale operator have the same zero-row-sum property. It may, however, be necessary to add additional constraints on the coarse grid nodes to ensure that any Dirichlet conditions on the flow are adequately represented.

Knowing that the coarse grid contains points upon which the Dirichlet boundaries of the problem are represented, we coarsen these independently of the PDE. As Neumann boundaries are coarsened naturally with the variational coarsening on the natural boundary condition problem, care need only be taken in the coarsening of Dirichlet conditions. The coarsening of the matrix entries for a Dirichlet condition is natural, as they may be imposed nodally, by setting the diagonal to 1 and (symmetrically) zeroing off-diagonal connections, but coarsening of the Dirichlet data must be done more carefully. In particular, the Dirichlet data should be coarsened in order to ensure conservation of global pressure gradients. Since we consider smooth Dirichlet data, we simply coarsen the data nodally. However, in the presence of significant variation in the Dirichlet data, an explicit averaging would likely be more appropriate, and would need to be carefully chosen.

Once we have derived the coarse-scale model with appropriate boundary con-

ditions, it can be easily solved with a correction-based multigrid scheme, such as the method used to derive the coarse-scale model. The coarse-scale solution can be considered on the coarse grid, but little additional effort is required to interpolate it to the finest grid (the original grid), where full use may be made of the multiscale information contained in the coarse-scale basis functions and, thus, the coarse-scale solution. Once the fine-grid representation of the coarse-scale solution has been computed, we may compute its macroscopic properties for comparison with other techniques.

We consider the results of this upscaling on both geometrically regular and geostatistical permeability fields as compared to various coarse-scale discretizations of the problem. In these examples, we consider a fine grid of 64×64 elements on the unit square (so that $h = \frac{1}{64}$) with a coarse grid of 16×16 elements. We impose no-flow (Neumann) boundary conditions on the top and bottom of the domain (along $y = 0$ and $y = 1$) and pressure (Dirichlet) boundary conditions on the left and right edges. In order to induce a flow across the domain, we set the pressure at the left edge ($x = 0$) to 1 and at the right edge ($x = 1$) to 0. The results presented below use the BoxMG method [34] for constructing the coarse-scale models analyzed.

4.4.3 Numerical Results for Structured Permeability Fields

As an initial test, we consider a fine-scale structure that can also be resolved on the coarse scale, as in Figure 4.12. Here, the fine-scale permeability is given as a 2×2 tiling of the unit square with a high-permeability inclusion, where $\mathcal{K} = 1000$, on $[\frac{1}{4}, \frac{3}{4}]^2$. Cross-sections are considered both passing through and outside the inclusions, as indicated by the dotted lines in Figure 4.12.

As this permeability can be resolved directly on the coarse scale, the appropriate comparison is to a direct discretization with the permeability sampled from the center of each coarse-scale element. Considering a fine-scale discretization on a 64×64 element mesh coarsened to a 16×16 element mesh, all multigrid coarsening occurs within homo-

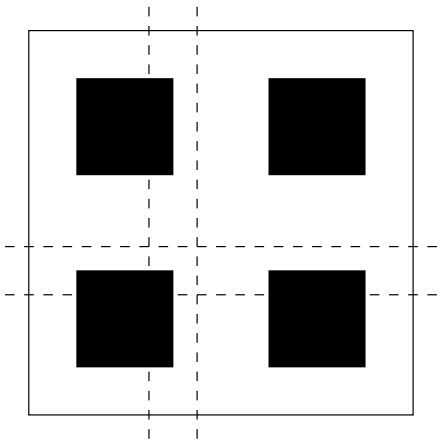

Figure 4.12: Periodic Sand/Shale Permeability Field and Cross-Sections Considered

geneous regions. Thus, the multigrid-upscaled equations are the same as those directly discretized on the coarse scale. Cross-sections of pressure in both x and y, passing both through and outside the high-permeability inclusions are shown in Figure 4.13. Here, we see that there is some loss of accuracy in the coarse-scale solution due to the curvature in the fine-scale solution that cannot be represented on the coarse computational scale.

The 4×4 periodic example of Figure 4.14 provides a more challenging test because the fine-scale structure cannot be exactly represented on the coarse scale. This field is the same as that in Figure 4.5 that produced the coarse-scale basis function shown in Figure 4.7. The fine-scale structure is created by tiling the unit square with an high-permeability inclusion, where $\mathcal{K} = 1000$, on $[\frac{5}{16}, \frac{11}{16}]^2$. As in the fully-resolved example above, we again consider cross-sections both passing through the inclusions and that pass only through the background media.

As the fine-scale structure of the permeability cannot be resolved directly on the coarse-scale, a discretization on that scale, where the permeabilities are simply sampled from the coarse-scale elements, is expected to perform poorly. Thus, we also consider discretizing the problem, again with finite elements, but now choosing the permeability used in discretizing over a coarse-scale element to be the harmonic average of the fine-

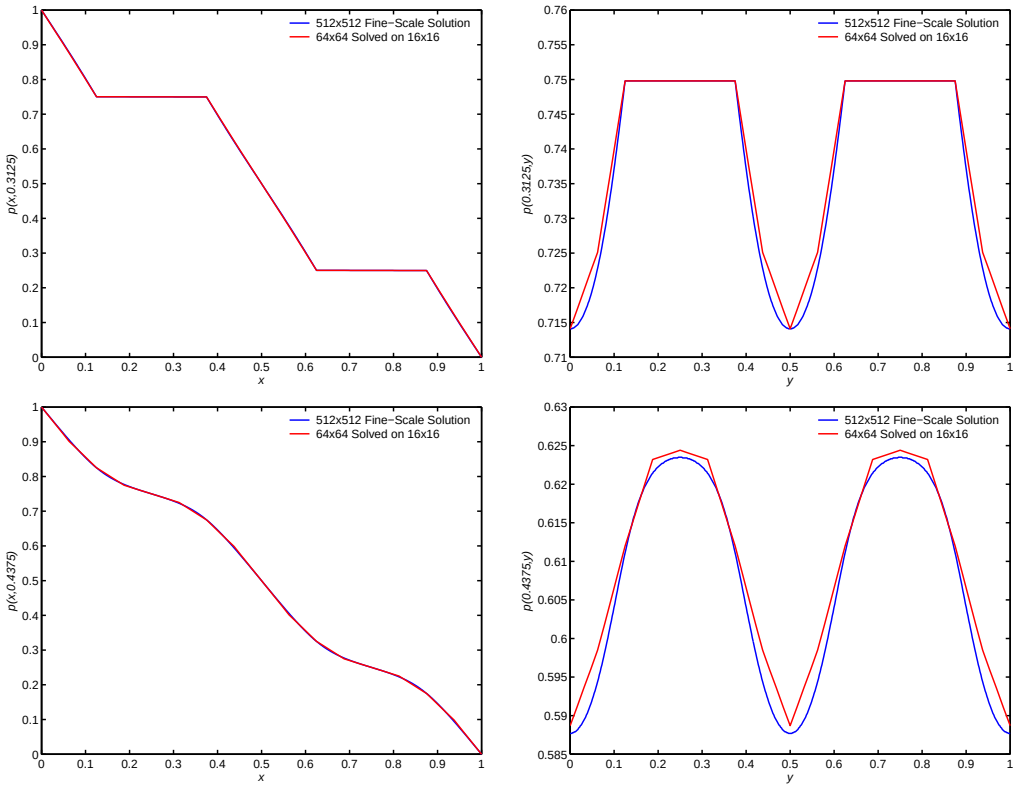

Figure 4.13: Cross-Sections of Pressure for 2×2 Periodic Permeability Example

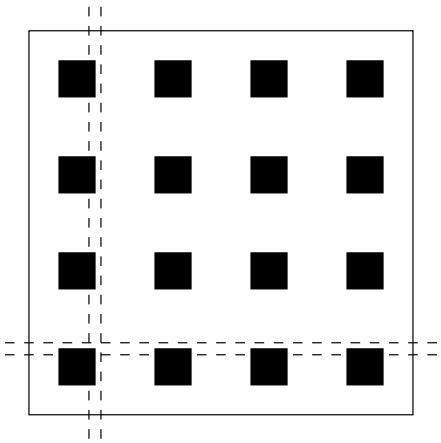

Figure 4.14: Periodic Sand/Shale Permeability Field and Cross-Sections Considered

scale permeability on the element. The choice of this average is motivated by the one-dimensional result discussed in Section 2.1 and because it is often used in practice.

Figure 4.15 shows the pressure fields of the solutions along the line $y = \frac{5}{32}$, which passes through four of the inclusions. While the coarse-scale solution resulting from sampling matches the macroscopic features of the fine-scale solution, it does not match the size of the inclusions, overestimating their width of $\frac{6}{64}$ as $\frac{8}{64}$, because two coarse-scale elements on this line have centers within the inclusion. The harmonic average case simply smears out the effect of the the inclusions, resulting in a much smoother gradient than is seen in the fine-scale solution. In contrast, the upscaled result matches the fine-scale solution exactly.

Cross-sections along $y = \frac{3}{16}$ show similar errors, as in Figure 4.16. The coarse-scale solution from the sampled case fails to resolve any character of the flow, as it chooses this line to lie within the inclusions. The harmonic mean now matches the fine-scale solution better, although again has a smoother gradient than the fine-scale solution exhibits. The fine-scale behavior outside the inclusions is smeared out when the harmonic mean is used to average the fine-scale permeability. The upscaled solution, however, is again quite close to the fine-scale solution, in both macroscopic and

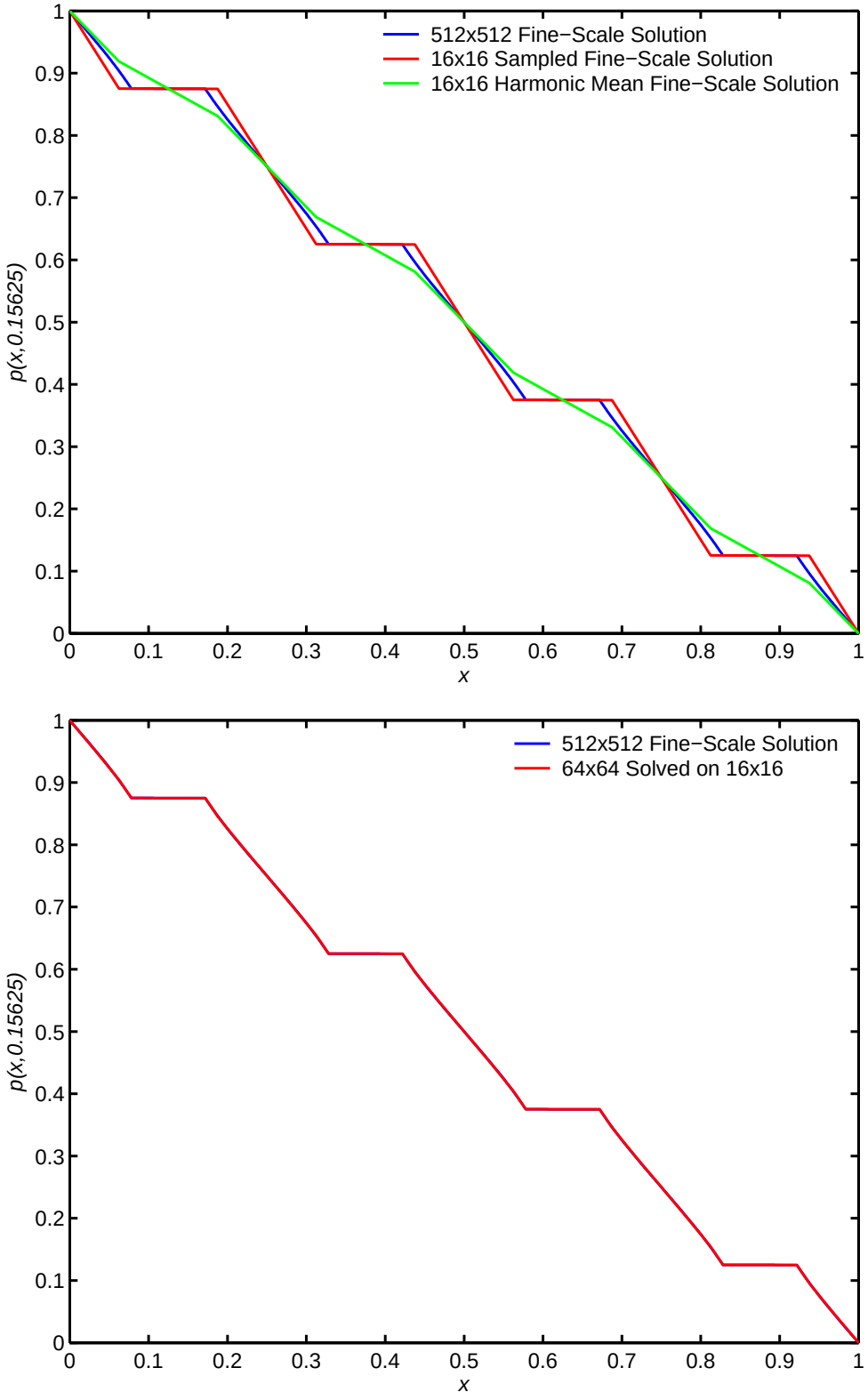

Figure 4.15: Cross-Sections of Pressure for 4×4 Periodic Permeability Example

microscopic detail.

Cross-sections in y are somewhat more illuminating. Along $x = \frac{5}{32}$, we see inaccuracies in all three solutions in Figure 4.17. The coarse-scale discretization with a sampled permeability tracks the coarse-scale solution well macroscopically, accurately predicting both its maximum and minimum values, but fails to resolve finer-scale details such as the extent of the plateaus from within the inclusions and the curvature of the troughs between these plateaus. The harmonic average does a very poor job of tracking the fine-scale solution, only matching the largest-scale trend and even doing that relatively poorly. It neither resolves the maxima and minima of the fine-scale pressure nor accurately represents the local character of this cross-section. In contrast, the upscaled solution matches the fine-scale solution quite well, except at the very bottoms of the troughs where it undershoots the true value. This may be due to the inability of the coarse-scale basis functions to accurately resolve the curvature in the fine-scale solution at this point, as may be seen in the basis function of Figure 4.7, where the basis closely matches a bilinear function, except with flat regions within the permeabilities.

Considering the line, $x = \frac{3}{16}$, which does not pass through the inclusions, we again see inaccuracies in the approximations in Figure 4.18. As in the case of the cross-section in x, the sampled coarse-scale problem predicts this line to be within the inclusions and thus does a poor job of matching the fine-scale profile. The profile of the harmonic-mean based solution is similar to that from the inclusion case. In this discretization, the effect of the inclusions is smeared over a wider area, and so we again get a poor match to the fine-scale solution. Here, the solution obtained by our upscaling approach on a 16×16 grid matches the fine-scale solution with a linear approximation that matches the large-scale oscillation of the fine-scale solution, but fails to resolve the fine-scale details of the curvature.

The misfit between the multigrid upscaling results and the fine-scale solutions in Figures 4.17 and 4.18 may be considered with respect to the underlying minimization

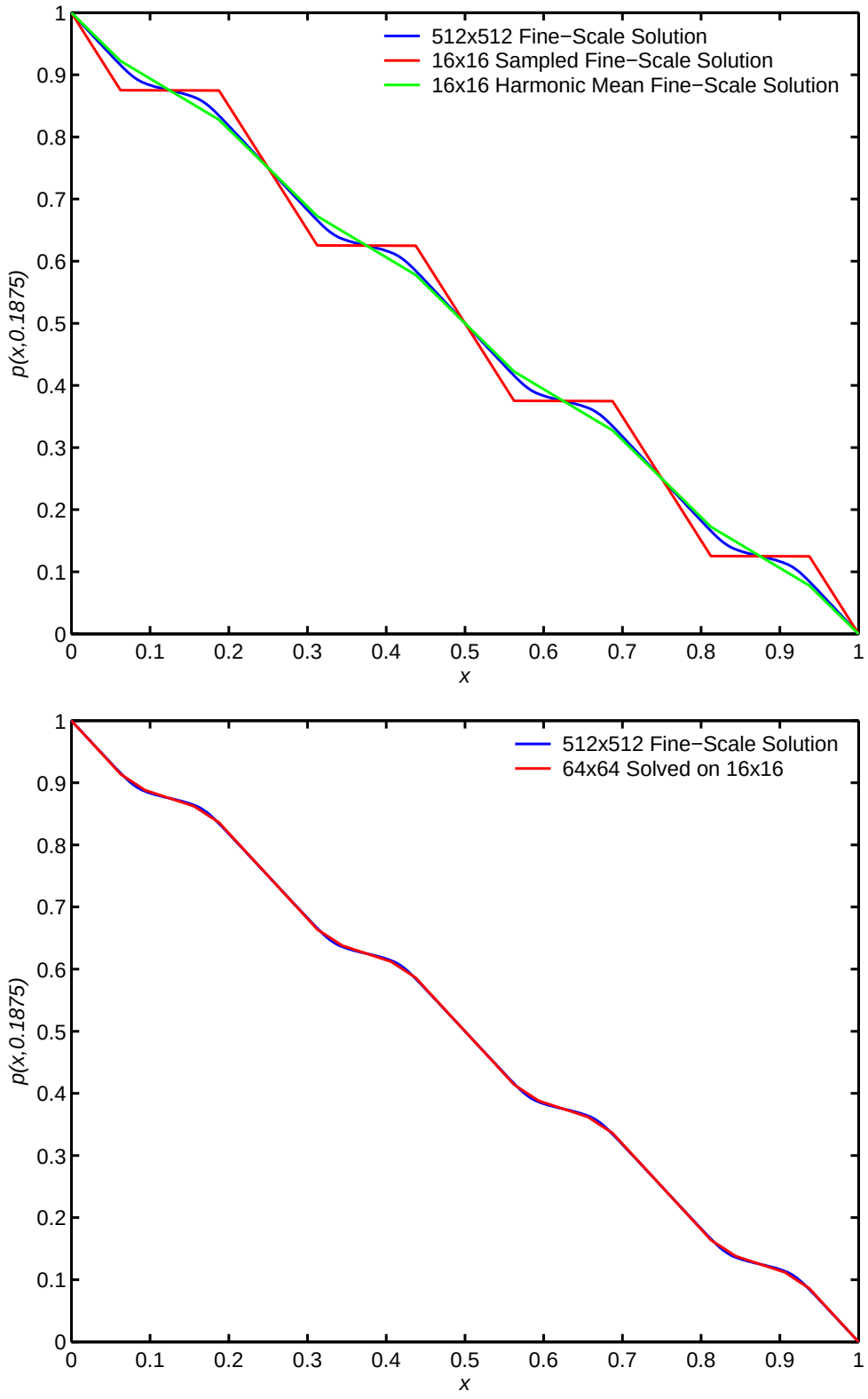

Figure 4.16: Cross-Sections of Pressure for 4×4 Periodic Permeability Example

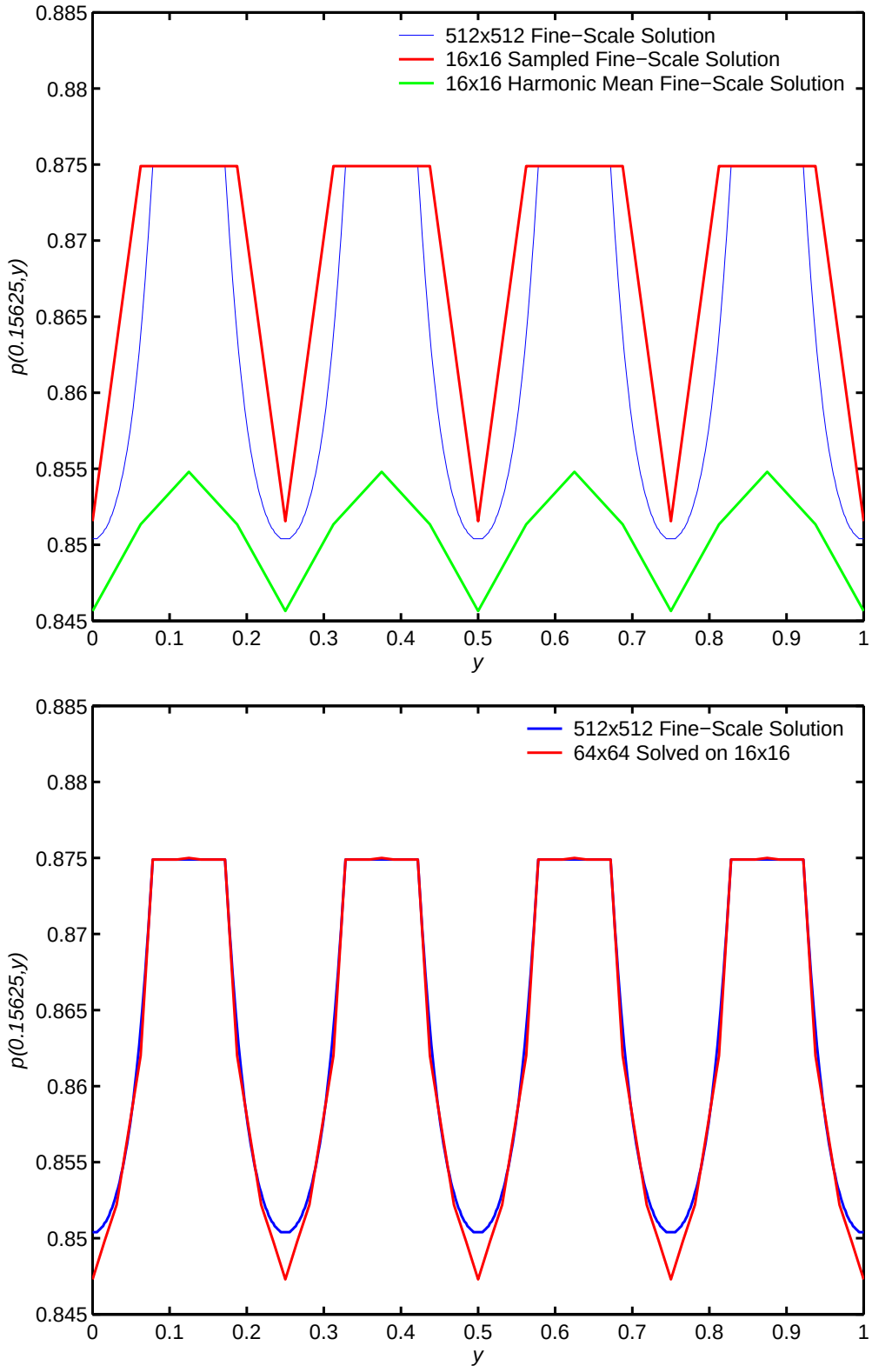

Figure 4.17: Cross-Sections of Pressure for 4×4 Periodic Permeability Example

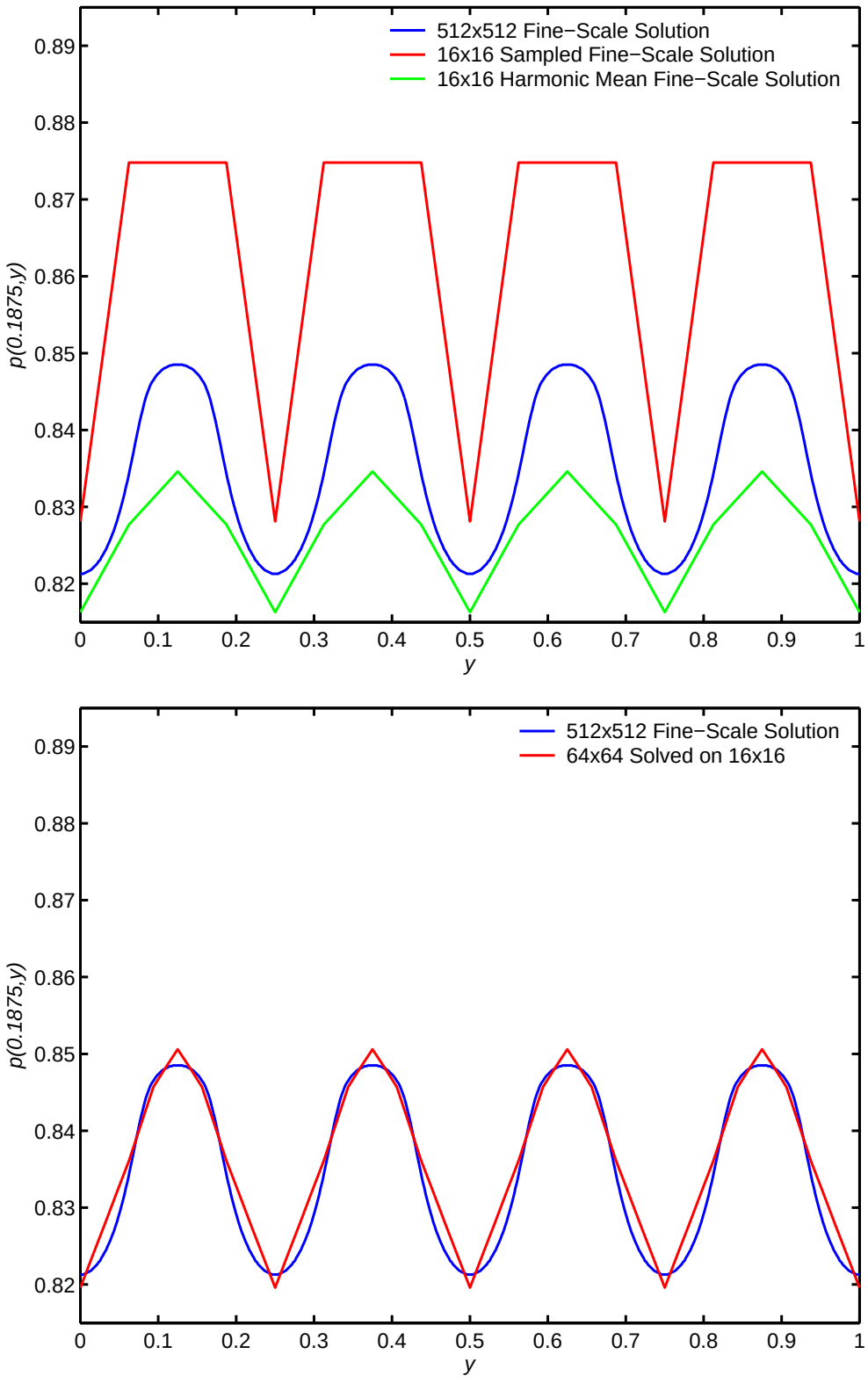

Figure 4.18: Cross-Sections of Pressure for 4×4 Periodic Permeability Example

problems of the finite element discretizations. Discretizing the diffusion equation,

$$-\nabla \cdot \mathcal{K} \nabla p = Q,$$

on the 64×64 element grid results in a minimization problem,

$$p = \underset{\tilde{p} \in \mathcal{V}}{\operatorname{argmin}} \iint \left(\frac{1}{2} (\mathcal{K} \nabla \tilde{p}) \cdot \nabla \tilde{p} - Q \tilde{p} \right) d\mathbf{x},$$

where $\mathcal{V}$ is the space of continuous, piecewise-bilinear functions on the 64×64 element grid. We, however, solve this minimization problem over an even smaller subspace of $\mathcal{V}$ and, thus, expect a more significant mismatch to the exact solution. Notice, however, that the errors are most significant in the background medium, where $\mathcal{K}$ is relatively small, and we see no lost accuracy in regions of high permeability, which is consistent with this minimization point of view.

4.4.4 Numerical Results for Geostatistical Permeability Fields

The geostatistical examples of Figures 2.3 and 2.4 have much richer fine-scale features, and thus we expect the problem of accurately representing the details of these fields on the coarse grid to be much more difficult than for the geometrically regular cases considered above. Since each element represents distinct material properties, the simple sampling approach is no longer appropriate. Instead, we consider direct coarse-level discretizations based on arithmetic and harmonic averaging of the permeability. We also consider the approach of Durlofsky, based on the periodic theory, of solving subdomain problems to compute an upscaled permeability for each coarse-scale element and then discretizing and solving the coarse-scale problem with these permeabilities.

For the weakly-layered field in Figure 2.3, these methods all capture, to some extent, the macroscopic trends in the pressure field. Cross-sections of the pressure along $y = \frac{5}{32}$, shown in Figure 4.19, show reasonable matches between the profiles of the fine-scale solution and those of the coarse-scale approximations. Both the arithmetic

and harmonic means capture the appropriate trend, although are noticeably off in exact details of the fine-scale pressure. The upscaling approach based on periodic theory actually does quite well at picking up these details, missing only a relatively small portion over the interval $[\frac{1}{2}, \frac{3}{4}]$. Our multilevel upscaling procedure, however, matches the fine-scale pressure nearly exactly for this problem.

Cross-sections along $x = \frac{5}{32}$, depicted in Figure 4.20, show more variation between the methods. The coarse-scale solution based on the arithmetic mean does a reasonable job of approximating the fine-scale solution as a linear function, matching the general trends, but certainly missing many of the finer-scale features. The harmonic-mean approach yields a pressure that matches the fine-scale behavior only in its broadest features. In contrast, the periodic theory approach yields a very good piecewise linear approximation to the fine-scale solution, outdone only by the multilevel upscaling approach that, by virtue of its use of a multiscale basis, matches many of the fine-scale features of this pressure cross-section.

This analysis may be quantified by considering qualities such as the prediction of location and values of maxima and minima in the pressure. In fact, accurately predicting such quantities is quite important in unsaturated or multiphase flow, where the PDE coefficients depend on these pressures and inaccurate values may result in non-physical simulation results. While tracking each local minima and maxima in the pressure is important, we present results here for finding only the absolute maximum and minimum pressures along the cross-section of $x = \frac{5}{32}$. The absolute extrema for the $y = \frac{5}{32}$ cross-section occur due to the Dirichlet boundary conditions and, thus, do not provide interesting data.

The maximum pressure of the 512×512 fine-grid solution occurs at $y = 0.043$ with value 0.8688, while the minimum pressure occurs at $y = 0.5176$ with value 0.5971. Errors in the computation of the value and location of these extrema for the four different upscaling methods are given in Table 4.2. The arithmetic and harmonic means result in

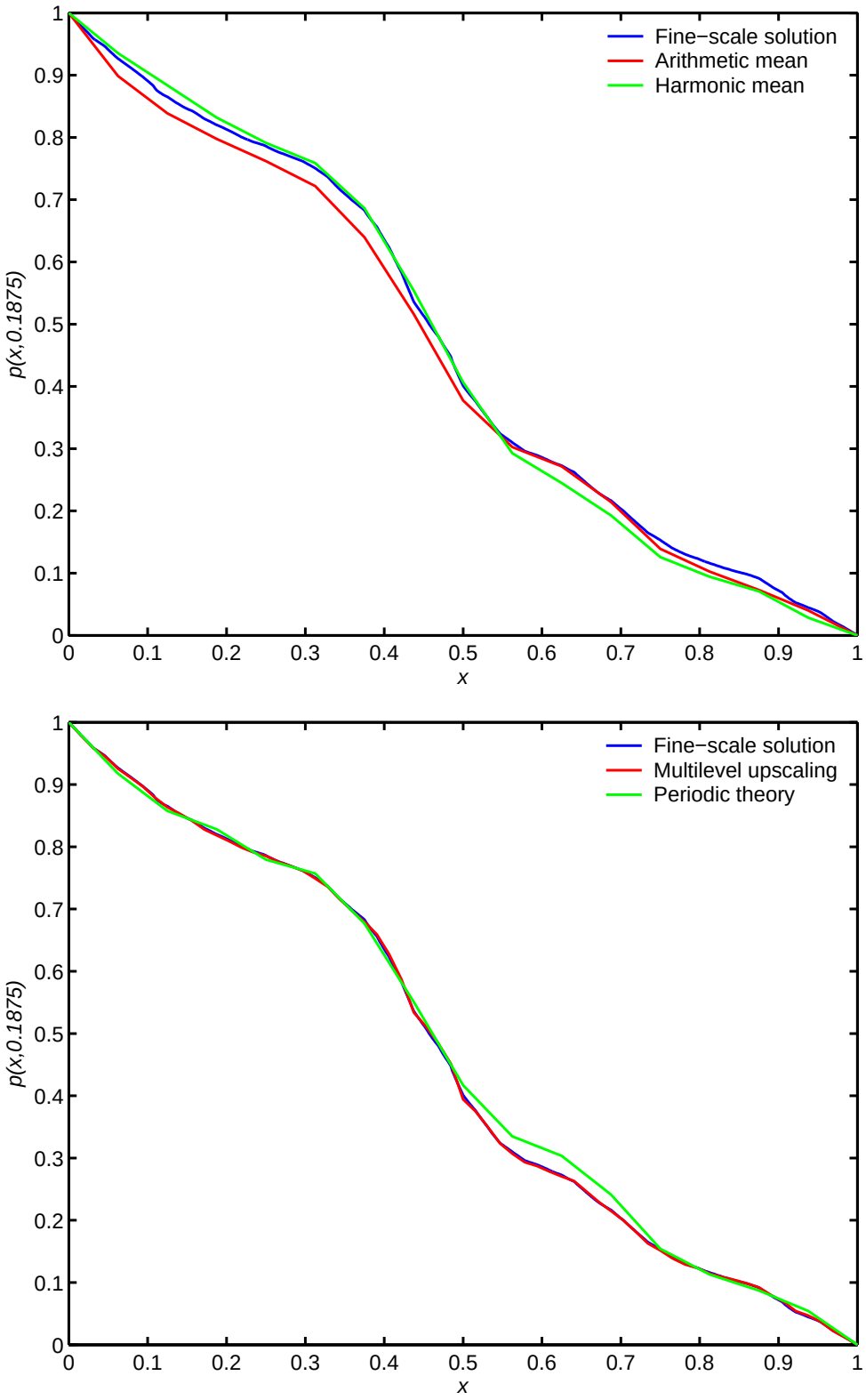

Figure 4.19: Cross-Sections of Pressure for Mildly-Layered Geostatistical Example

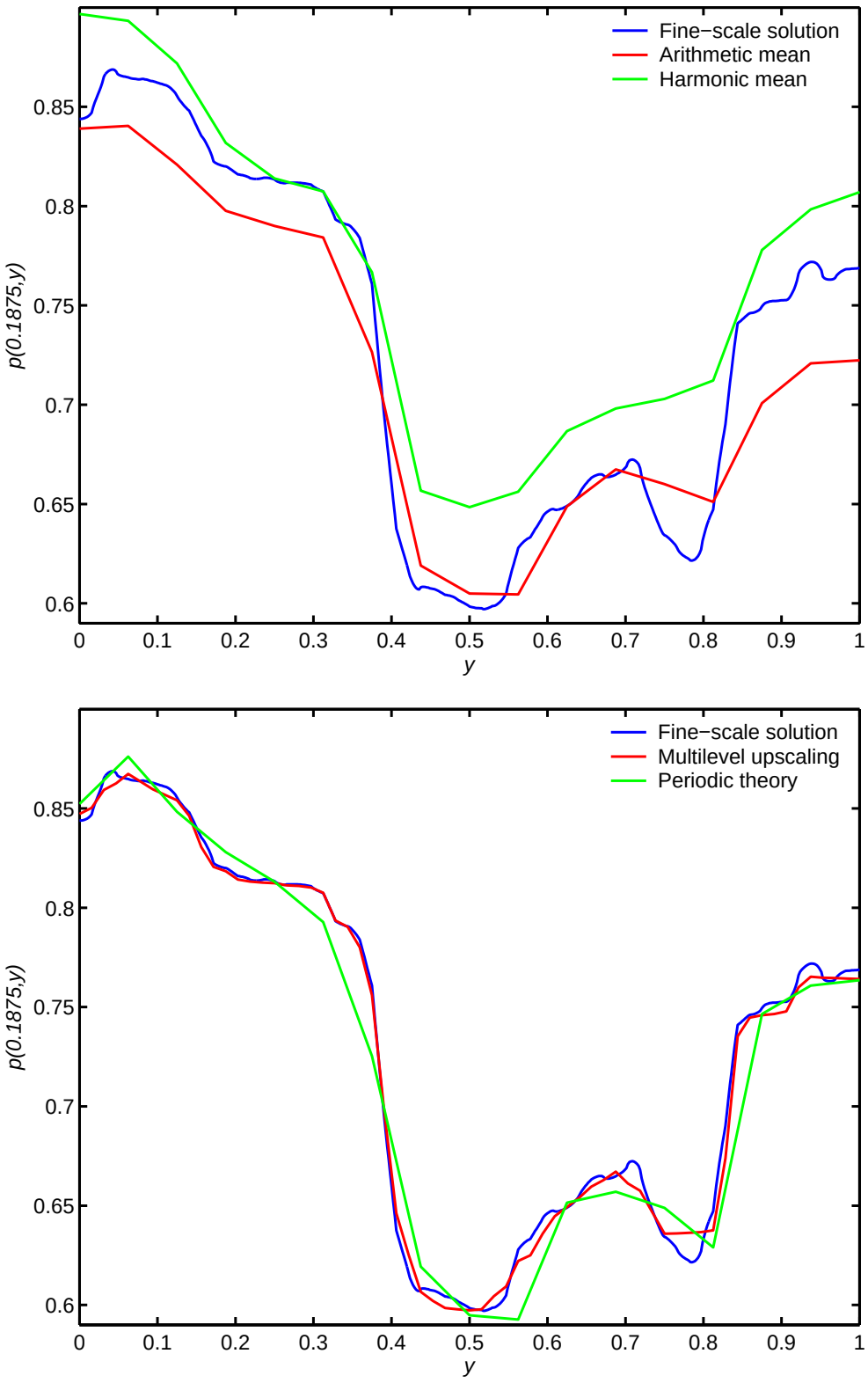

Figure 4.20: Cross-Sections of Pressure for Mildly-Layered Geostatistical Example

Upscaling Technique	Arithmetic Mean	Harmonic Mean	Periodic Theory	Multilevel Upscaling
Error in maximum pressure	2.8×10^{-2}	2.8×10^{-2}	7.3×10^{-3}	1.4×10^{-3}
Error in location of max. p	2.0×10^{-2}	4.3×10^{-2}	2.0×10^{-2}	2.0×10^{-2}
Error in minimum pressure	7.4×10^{-3}	5.1×10^{-2}	4.4×10^{-3}	2.0×10^{-4}
Error in location of min. p	4.5×10^{-2}	1.8×10^{-2}	4.5×10^{-2}	1.8×10^{-2}

Table 4.2: Errors in Computed Extreme Pressures for Mildly-Layered Geostatistical Permeability Field

similar errors in the maximum pressure, while the arithmetic mean is somewhat more accurate in predicting the minimum pressure. The periodic theory approach yields even more accurate maximum and minimum values than these averages; however, the multilevel upscaling approach yields the most accurate predictions of all four methods. All four methods yield comparable errors in the location of these extrema.

The strongly-layered permeability field provides the most difficulty of these test cases for all of the coarse-scale approximation methods. Figure 4.21 shows that both explicitly averaged coarse-scale models predict the general shape of the fine-scale pressure, but again miss details of a cross-section along $y = \frac{5}{32}$. Here, the arithmetic mean is much smoother than the fine-scale solution, whereas the harmonic mean does a better job of tracking the fine-scale variation. The periodic theory approach does a very poor job of predicting the pressure along this line, with a profile that bears little relation to that computed on the fine scale. Once again, the multilevel upscaling approach produces the best approximation to this cross-section and accurately captures most of the fine-scale features.

Cross-sections along $x = \frac{5}{32}$ show this even more dramatically, as in Figure 4.22. Both the harmonic and arithmetic averages provide a believable approximation to the fine-scale solution, particularly in the middle of this cross-section, but miss significant features near the endpoints. The periodic theory gives a horrific approximation to the fine-scale behavior that matches only some aspects of the changes in the fine-scale

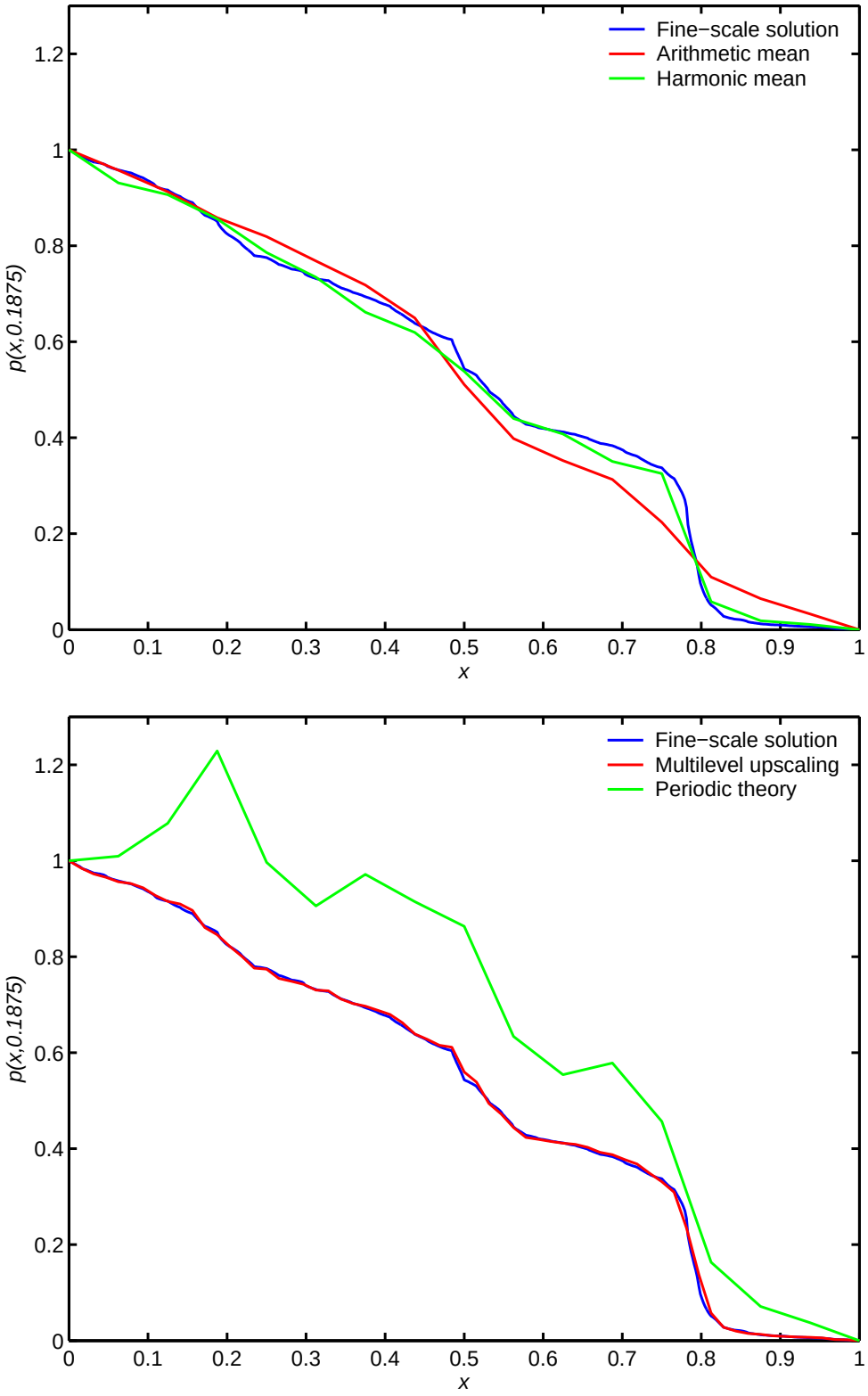

Figure 4.21: Cross-Sections of Pressure for Strongly-Layered Geostatistical Example

Upscaling Technique	Arithmetic Mean	Harmonic Mean	Periodic Theory	Multilevel Upscaling
Error in maximum pressure	5.5×10^{-2}	8.1×10^{-2}	2.7×10^{-1}	1.0×10^{-3}
Error in location of max. p	9.4×10^{-1}	6.6×10^{-2}	7.5×10^{-1}	3.9×10^{-3}
Error in minimum pressure	2.3×10^{-3}	7.0×10^{-4}	8.6×10^{-2}	1.4×10^{-3}
Error in location of min. p	7.8×10^{-3}	7.8×10^{-3}	3.0×10^{-1}	7.8×10^{-3}

Table 4.3: Errors in Computed Extreme Pressures for Strongly Layered Geostatistical Permeability Field

solution (that the approximation increases and decreases where the solution does), but does a very poor job of representing the structure of interest. The multilevel upscaling method gives a much better approximation to the fine-scale solution, although still misses some fine-scale details, such as the local maxima near $y = 0.2$.

To quantify this analysis, we again consider how well each approximation matches the location and values of the global maximum and minimum along the cross-section of $x = \frac{5}{32}$. The maximum pressure for the 512×512 grid fine-scale solution along this cross-section is $p = 0.9576$ at $y = 0.9336$, and the minimum is $p = 0.768$ at $y = 0.5703$. Errors in the computation of the value and location of these extrema for the four different upscaling methods are given in Table 4.3. The arithmetic and harmonic means both do a good job of predicting the minimum pressure, but fail to accurately approximate the maximum, with the arithmetic mean locating the maximum pressure at the left endpoint, and the harmonic mean locating the maximum at the right endpoint. The periodic-theory approach predicts these features dismally, worse than even the simple averages. Interestingly, the multilevel upscaling approach is comparable with the simple averages in errors in the location and value of the minimum pressure, but significantly more accurate in predicting the maximum.

Another important measure of the accuracy of the coarse-scale model is the ability to predict certain macroscopic properties, such as the net flux into or out of the domain. For both of the geostatistical examples, we measure the net flux out of the domain by

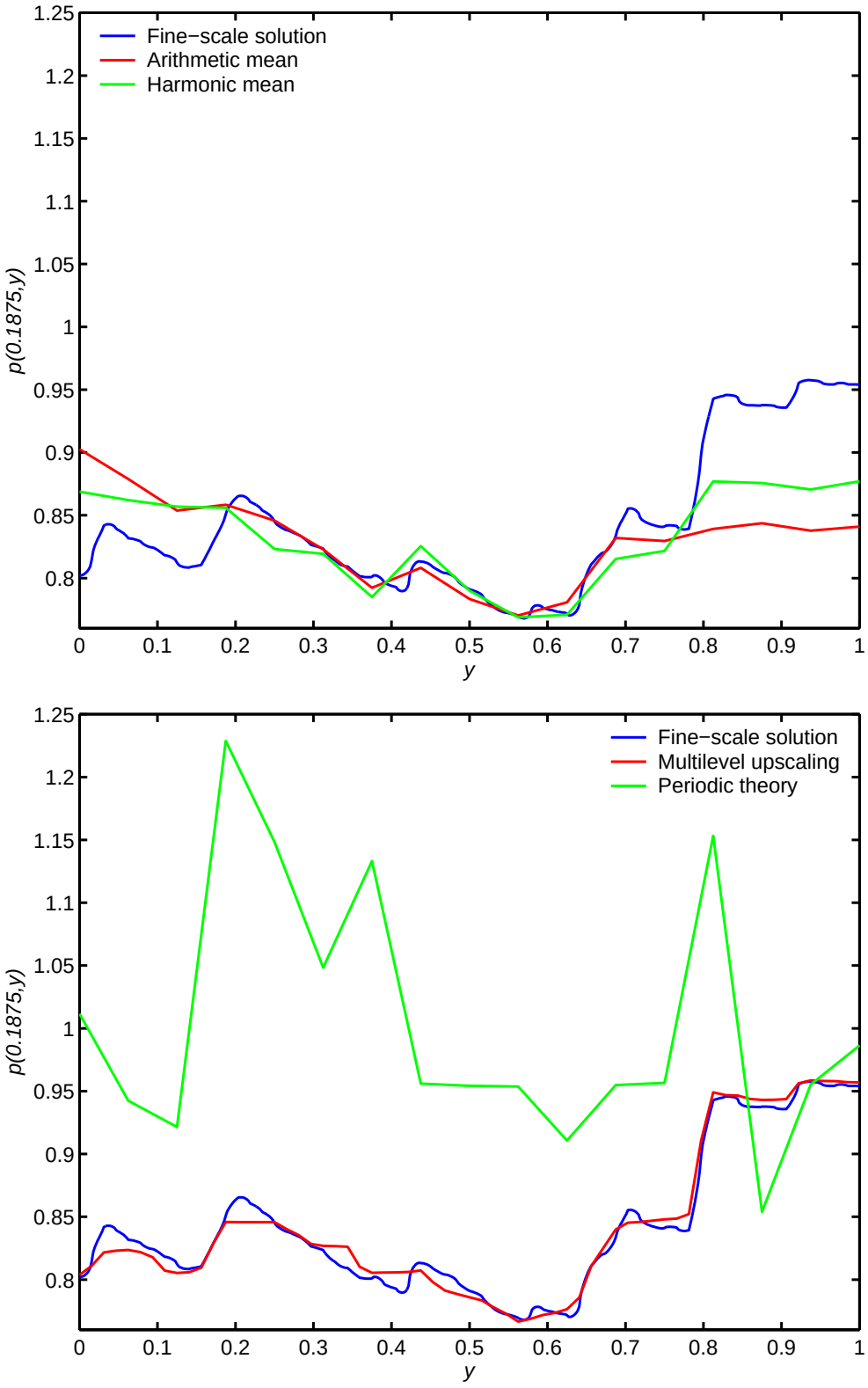

Figure 4.22: Cross-Sections of Pressure for Strongly-Layered Geostatistical Example

integrating $(\mathcal{K}\nabla p) \cdot \mathbf{n}$ along the outflow boundary, where $\mathbf{n}$ is the outward unit normal vector along this edge, for the various coarse-scale discretizations considered above. For the averaging and periodic-theory based discretizations, this integration is done using the coarse-scale solution and coarse-scale permeabilities (as determined by the method used). For the multilevel upscaling method, we consider the interpolated coarse-scale solution and integrate on the fine scale, taking advantage of the approximation provided by the multiscale basis functions.

We consider both the flows above, where Dirichlet boundary conditions are imposed so as to induce a flux from left to right across the domain, as well as the case when Dirichlet boundary conditions induce the flux from bottom to top across the domain, with Neumann (no-flow) boundary conditions along the left and right edges. In each case, we report both the computed approximate flux as well as the percentage error as compared to a fine-scale, 512×512 element approximation. We compute the fluxes for all four upscaling techniques discussed above, in all cases starting with the 64×64 permeability specification and coarsening down in factors of two as far as a 4×4 coarse scale.

Tables 4.4 and 4.5 show the computed flux values and associated errors for the mildly-layered field with flow in the x-direction, as compared to a fine-scale calculation of this flux as 2.696. As with the upscaled permeability calculations of Section 4.3, we see that the arithmetic mean is always an upper bound on the flux and the harmonic mean provides a lower bound. Both choices, however, give poor approximations to the fine-scale flux, with much more significant errors than either the periodic theory or multilevel upscaling techniques. Both of these techniques provide good approximations to the fine-scale flux, with errors under 2% when coarsening only a single level to a 32×32 coarse scale, and errors under 4% on all coarse scales down to the 4×4 grid. Similar behaviour is seen in the fluxes in the y-direction, as shown in Tables 4.6 and 4.7. Here, we see that the multilevel upscaling method is somewhat more accurate than the

Upscaling Technique	32×32	16×16	8×8	4×4
Arithmetic Mean	2.895	3.092	3.400	3.660
Harmonic Mean	2.153	1.558	1.087	0.944
Periodic Theory	2.748	2.755	2.751	2.797
Multilevel	2.743	2.753	2.802	2.795

Table 4.4: Computed Outflow Flux Values for Mildly-Layered Geostatistical Permeability Field with Flow in the x-direction

Upscaling Technique	32×32	16×16	8×8	4×4
Arithmetic Mean	7.4%	14.7%	26.1%	35.8%
Harmonic Mean	20.1%	42.2%	59.7%	65.0%
Periodic Theory	1.9%	2.2%	2.0%	3.7%
Multilevel	1.7%	2.1%	3.9%	3.7%

Table 4.5: Percentage Errors in Outflow Fluxes for Mildly-Layered Geostatistical Permeability Field with Flow in the x-direction

periodic theory, with errors of less than 2% relative to the fine-scale flux of 0.705 on all coarse scales. He et al. [52] consider an example with the same media characteristics, but with a 50×50 fine grid upscaled to a 10×10 coarse grid. They report errors of 5% and 8%, relative to the 50×50 solution, in computing the coarse-scale x- and y-fluxes, respectively.

Results for the strongly-layered media of Figure 2.4 show significantly more variation in the computed fluxes. The fine-scale calculation for outflow flux in the x-direction is 2.229, with coarse-scale fluxes shown in Table 4.8 and the percentage errors relative to the fine-scale value in Table 4.9. Again, the arithmetic and harmonic means give upper and lower bounds on the fine-scale permeability, but with significant error. The approximations from the periodic theory are much worse in this case, particularly on coarser grids, peaking with the 8×8 grid where the predicted value is more than twice the fine-scale computation. Performance of the multilevel upscaling method is worse than that in the mildly-layered case, but still much better than any of the other methods, with a maximum error of 16.6% occuring when upscaling 3 levels.

Flow in the y-direction produces more striking results, as shown in Tables 4.10

Upscaling Technique	32×32	16×16	8×8	4×4
Arithmetic Mean	0.864	1.344	1.874	2.073
Harmonic Mean	0.656	0.616	0.548	0.513
Periodic Theory	0.721	0.756	0.737	0.727
Multilevel	0.708	0.718	0.712	0.708

Table 4.6: Computed Outflow Flux Values for Mildly-Layered Geostatistical Permeability Field with Flow in the y-direction

Upscaling Technique	32×32	16×16	8×8	4×4
Arithmetic Mean	22.7%	90.7%	166.0%	194.3%
Harmonic Mean	6.9%	12.6%	22.2%	27.2%
Periodic Theory	2.4%	7.3%	4.6%	3.2%
Multilevel	0.5%	1.9%	1.1%	0.5%

Table 4.7: Percentage Errors in Outflow Fluxes for Mildly-Layered Geostatistical Permeability Field with Flow in the y-direction

Upscaling Technique	32×32	16×16	8×8	4×4
Arithmetic Mean	2.847	3.347	3.655	4.031
Harmonic Mean	1.565	0.982	0.579	0.423
Periodic Theory	2.319	3.482	4.923	3.124
Multilevel	2.430	2.558	2.599	2.493

Table 4.8: Computed Outflow Flux Values for Strongly-Layered Geostatistical Permeability Field with Flow in the x-direction

Upscaling Technique	32×32	16×16	8×8	4×4
Arithmetic Mean	27.7%	50.2%	64.0%	80.8%
Harmonic Mean	29.8%	55.9%	74.0%	81.1%
Periodic Theory	4.0%	56.2%	120.8%	40.1%
Multilevel	9.0%	14.8%	16.6%	11.8%

Table 4.9: Percentage Errors in Outflow Fluxes for Strongly-Layered Geostatistical Permeability Field with Flow in the x-direction

Upscaling Technique	32×32	16×16	8×8	4×4
Arithmetic Mean	1.571	2.370	3.093	3.700
Harmonic Mean	0.819	0.628	0.454	0.353
Periodic Theory	1.177	5.750	16.207	2.705
Multilevel	1.074	1.058	0.919	0.962

Table 4.10: Computed Outflow Flux Values for Strongly-Layered Geostatistical Permeability Field with Flow in the y-direction

Upscaling Technique	32×32	16×16	8×8	4×4
Arithmetic Mean	50.5%	127.0%	196.3%	254.4%
Harmonic Mean	21.5%	39.9%	56.5%	66.2%
Periodic Theory	12.7%	451.7%	1452.4%	159.1%
Multilevel	2.8%	1.3%	11.9%	7.8%

Table 4.11: Percentage Errors in Outflow Fluxes for Strongly-Layered Geostatistical Permeability Field with Flow in the y-direction

and 4.11. Here, the fine-scale flux of 1.044 is better approximated on every level of the multilevel upscaling method than on any level of any other method. While we expect the bounds of the arithmetic and harmonic means to be poor, as in Figure 4.3, the periodic theory approach gives its poorest performance on this problem, overestimating the flux on the upscaled 8×8 grid by a factor of over 15. In contrast, the multilevel upscaling technique remains quite accurate, with errors of less than 3% on the first two coarsening levels and under 12% on all levels.

These results convincingly indicate the advantages of the multilevel upscaling technique, particularly given its low cost. We are able to accurately reconstruct fine-scale characteristics of the flow through coarse-scale calculations and the interpolation of an operator-induced multigrid method. Not only does this method accurately predict macro-scale properties, such as the integrated flux or global trends in the pressure field, it also reconstructs accurate approximations to local variation, such as local maxima and minima of the pressure.

A simple approach to improving the accuracy of these results with a small additional cost is to augment the interpolation process from the coarse, computational scale

to the fine scale with a single post-smoothing step on each finer level. This creates, in effect, a V(0,1) multigrid cycle, with the computational-scale solve discussed above taking the place of a coarse-scale solve and the small amount of post-relaxation incorporating further fine-scale information into the solution. Figure 4.23 shows the cross-sections in the x-direction for the weakly-layered (top) and strongly-layered (bottom) geostatistical permeability fields when this post-relaxation strategy is employed.

Both figures show a noticeable increase in the resolution of fine-scale behavior, accurately resolving features missed in the computational-scale solution. In particular, we see a more accurate resolution of the local maxima and minima of the pressures along these cross-sections. The same improvement is not seen in the computed outflow fluxes, however, which appear to be marginally less accurate than those of the coarse-scale solution. Fluxes further upstream, such as those integrated along the line $x = \frac{1}{2}$, do show a measurable improvement in accuracy. As is typical, relaxation is slow to propagate the improved upstream pressure information to the outflow boundary.

These results offer insight into the cost-effectiveness of the multilevel upscaling approach. Once the multigrid hierarchy is constructed, solution of a suitably-coarse problem is quite cheap (costing several coarse-scale work units, but only a small amount of work relative to the fine scale). This iteration does not converge to the solution of the fine-scale problem, however, and additional relaxation on finer grids must be used to improve the fine-scale character of the approximate solution. Such additional accuracy comes at a cost of more fine-scale work units. Balancing the desired accuracy with a practical computational cost determines the optimal strategy for a given application.

There are many possible avenues of further research in this area, which are primarily discussed in Section 6.2.1. There are two avenues, however, that we feel are of particular interest in relation to these results. First, we are very interested in the effect of using other variational multigrid methods for the upscaling. BoxMG is well-suited to the task, particularly due to the approximate conservation of normal flux in interpola-

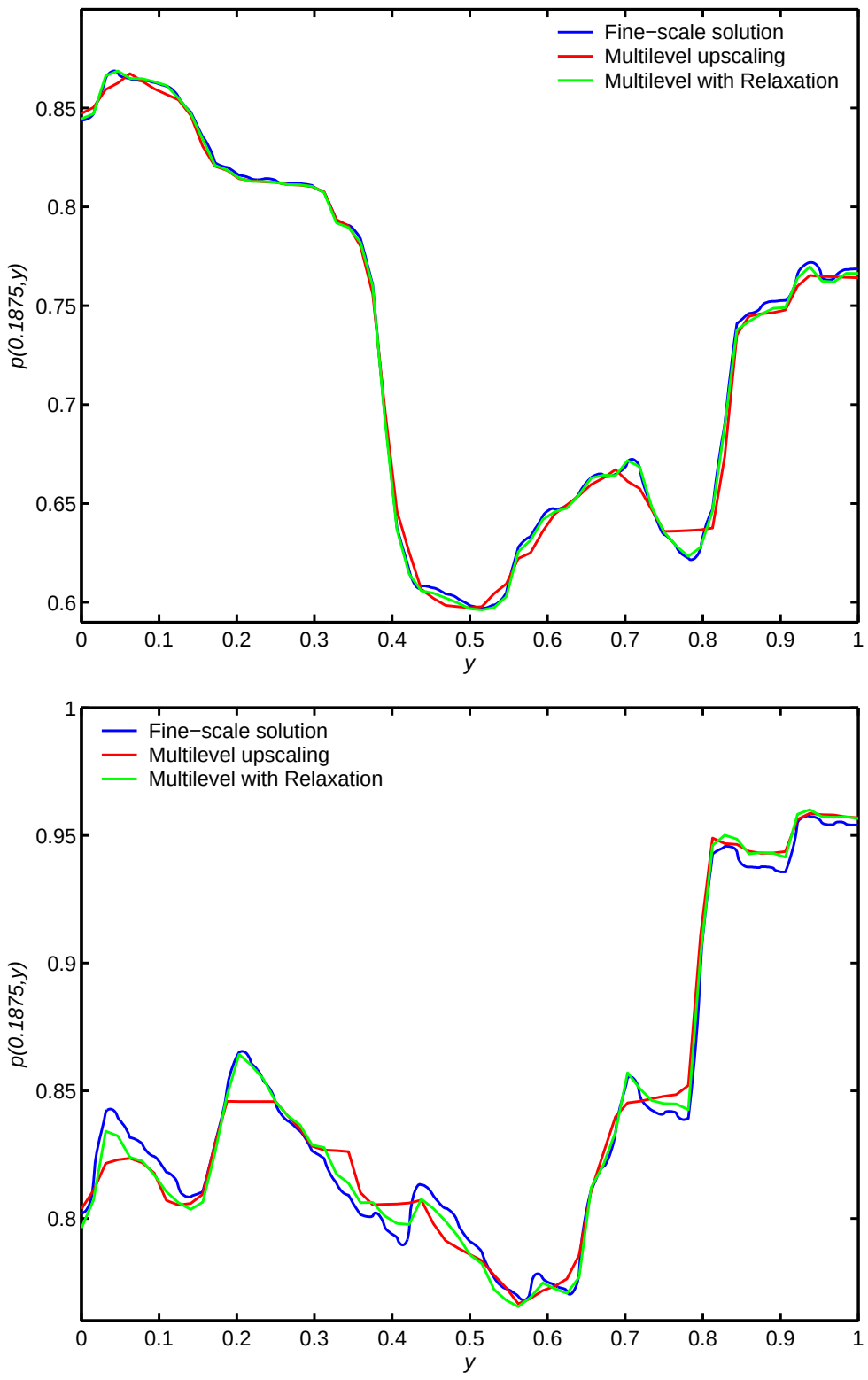

Figure 4.23: Cross-Sections of Pressure After Post-Relaxation for Geostatistical Examples

tion [67], but is restricted by the assumption of geometrically-regular coarse grids. In contrast, AMG is well-suited to handling the often-complex geometries of real reservoir simulations, but uses a much less physical interpolation scheme.

We also feel that a more detailed comparison to the multiscale finite element method (MSFEM) [56] is appropriate. The approach of using multiscale basis functions in the coarse-scale model is shown here to be very effective, and we expect this to be true also for the MSFEM. However, the cost of explicitly forming the finite element basis functions is much more significant than that of forming the interpolation operators necessary in our approach. Thus, we believe that the MSFEM method may provide comparable accuracy, although at a higher cost. Verification of this expectation is, of course, necessary in evaluating both methods.

Chapter 5

Adaptive Multigrid Methods

Consider the class of solvers based on multigrid principles that depend little or not at all on geometric information about the problem, but instead attempt to use basic concepts of "algebraic smoothness" (see Section 3.3) to determine effective coarsening and/or relaxation processes. Solvers of this type typically assume some defining characteristic of algebraic smoothness that specifies error components that are not quickly eliminated by the relaxation that is being used. For example, all such components are assumed, in standard AMG (cf. [72] and §3.3), to vary slowly along so-called strong connections in the matrix, or, in standard smoothed aggregation (SA; cf. [82] and §3.4), to be represented locally by a few prototype vectors supplied by the user. While appropriate use of the characteristic of algebraic smoothness seems essential for obtaining effective solvers, these additional assumptions limit the scope of applicability of these methods. In many important cases, errors missed by standard relaxation processes can vary substantially along strong matrix connections, and, in many cases, even the concept of strength of connection is not well understood. Moreover, supplying a fully representative set of prototypical smooth components is not always easy nor possible in practice.

The principal aim of the adaptive approach developed here is to eliminate or substantially reduce this reliance on the additional assumptions usually present in these methods. The basic idea is to test the initial version of the given solver on the homo-

geneous problem, $A\mathbf{x} = \mathbf{0}$, to determine its performance and expose whatever types of errors it cannot effectively handle. The resulting prototypical errors that these tests produce are then used to improve the algebraic multigrid process.

The concept of using a multigrid algorithm to improve itself began with standard AMG [71], where interpolation was adjusted to fit vectors obtained by relaxation on the homogeneous problem. In [23], a variation of this idea was used for recovering typical AMG convergence for a badly scaled scalar elliptic problem. While the method there was very basic and used only one error prototype, it contained many of the ingredients of the adaptive process developed here. These concepts were developed further in [63, 69, 70, 73]. The idea of fitting of eigenvectors corresponding to the smallest eigenvalues was advocated in [63] and [73], where an AMG algorithm determining these eigenvectors through Rayleigh quotient minimization was outlined. These vectors were, in turn, used to update the AMG interpolation and coarse-level operators. Most of these ideas were later summarized in [63].

In many ways, using algebraically smooth vectors in the definition of interpolation represents the next logical step in improving the interpolation operators used in robust geometric and algebraic multigrid methods. Alcouffe et al. introduced the idea of operator-induced interpolation in [2]. This improvement on the previously used geometric interpolation approach opened up a much larger class of problems to black-box multigrid solution.

In the present chapter, we use operator-induced interpolation approaches as well, but also rely on an automatic process that supplies representative smooth components to ensure optimal performance. By integrating information regarding algebraically smooth vectors into the definition of interpolation, we develop multigrid schemes that are hopefully optimal for elliptic problems where the discrete system is not necessarily given in terms of an M-matrix. These operator- and relaxation-induced interpolation approaches can, if properly implemented, greatly enlarge the class of problems that admits optimal

performance by a black-box multigrid technique.

We also introduce this form of adaptivity into AMG and SA. While classical multigrid methods can be viewed as stationary iterative methods [9], the methods presented here are dynamic. In fact, we propose using the method itself to drive its own iterative improvement. A "bootstrap" AMG method that is similar to the approach developed here was recently proposed for the classical AMG setting by Brandt [21, 24].

Several other attempts have been made to allow for the solver itself to determine from the discrete problem the information required to solve it successfully, without a priori assumptions on the form of the smooth error, including the methods of [25, 27, 29, 47]. All of these methods, however, have in common their requirement that the local finite element matrices of the problem be available, and they construct the multigrid transfer operators based on the algebraically smooth eigenvectors corresponding to local stiffness matrices assembled over element agglomerates. Although they can achieve encouraging convergence rates, their need to construct, store, and manipulate the coarse-level element information typically leads to increased storage requirements compared to those of classical AMG or standard SA. The methods described below attempt to achieve the good convergence properties of the element-based methods without the overhead of element storage that they require.

We refer to the approaches developed here as adaptive because they involve self-testing to expose slowly converging error components and adaptation of the schemes' components to improve themselves. The acronym αMG is used to refer to the general class of multigrid methods of this type, to suggest their primal algebraic nature: we have in mind methods that use only the defining characteristic of algebraic smoothness and must use an automatic, algebraic process to determine additional characteristics that enable effective determination of the full MG algorithm. The additional abbreviations αAMG and αSA, respectively, are used to refer to the specific AMG and SA versions of αMG.

In the next section, we present the adaptive framework and discuss the fundamental principles of αMG methods. Section 5.2 presents the application of these ideas in the AMG context. In Section 5.3, theoretical results are motivated and shown in the particular setting of a reduction-based AMG algorithm. Finally, the adaptive SA method is discussed in Section 5.4.

5.1 The Adaptive MG Framework

The details of the αAMG and αSA algorithms are quite complex. We arrived upon them by careful consideration of the basic principles and methodologies of an adaptive algorithm. For this reason, our discussion focuses on these basic principles before the particular details. We restrict our attention for the remainder of the paper to the case that the $n \times n$ matrix, $A = (a_{ij})$, is symmetric and positive definite, although most of what is developed applies to more general cases. Our aim is to develop an algebraic multigrid process to solve the matrix equation,

$$A\mathbf{x} = \mathbf{b},$$

without a priori knowledge of the character of algebraically smooth error.

5.1.1 The Adaptive MG Algorithm

An efficient multigrid process for solving $A\mathbf{x} = \mathbf{b}$ relies on the appropriate complementarity of relaxation and coarse-grid correction. Thus, we view the goal of the adaptive process as the development of a representative collection of vectors for which the chosen relaxation process is inefficient. In its most basic form, the adaptive process would be quite simple: relax on a significant number of vectors to expose slow-to-converge components and then choose interpolation to fit these vectors. Such an approach is, however, quite inefficient and a multiscale viewpoint proves more useful.

Suppose we know the matrix, A, but nothing more. Relaxation is then the only

possible way to expose algebraically smooth error components. However, with no further knowledge, there is no way of predicting, a priori, how many distinct components are needed to achieve good results. Experience (and, in the case of SA, theory [79]) has shown that, for discretizations of second-order scalar elliptic PDEs, a single component is sufficient, whereas six components may be needed for a problem such as 3D linear elasticity. To arrive at an optimal solver with a minimal amount of work, it thus seems necessary to start with a single vector and introduce new prototypes as the evolving method proves inadequate.

The situation is then that we have a given matrix and seek to find a single algebraically smooth vector upon which to base interpolation. Relaxation alone can achieve this, simply by iteration on the homogeneous problem. However, it typically requires a significant number of relaxations to expose a global, algebraically smooth vector, and so we seek to expose such components through multiscale development. We use only enough relaxation to expose errors smooth enough to be handled on the first coarse grid. Smoother errors are exposed by relaxation on this coarse grid and then used to create a still coarser grid in the usual multigrid fashion. Just a few steps of relaxation on the homogeneous problem on the finest grid quickly reduces a significant portion of a random initial guess, leaving error that can then be said to be locally algebraically smooth. If this prototype is used locally to define interpolation from some preselected coarse grid, then a coarse-grid problem that adequately represents the algebraically smooth error on the fine grid can be created. We can now iterate to an appropriate coarsest grid and interpolate a prototype of the smooth error to all grids. Proceeding recursively, this resembles a full approximation scheme multigrid for the algebraically smooth component, rather than the usual correction scheme method.

In this manner, a good prototype of a single algebraically smooth component can be determined and the resulting solver tested. If it proves sufficient, then the adaptive stage is complete. Inefficiency in the resulting solver indicates that relaxation

and coarse-grid correction are not yet perfectly complementary, and that there are distinct algebraically smooth components that are not being accounted for. Since these components are being reduced neither by relaxation nor coarse-grid correction, they can be exposed by an iteration as above with the current solver taking the place of relaxation. This may be repeated until acceptable convergence rates are attained.

Thus, we sketch the adaptive procedure as

Algorithm 2 (Abstract Adaptive Process).

(1) Let $k = 1$ and $\mathbf{x}^{(1)}$ be a random vector. Define, for all grids l, the methods CYCLE$_l(\mathbf{x}^{(l)},\mathbf{b}^{(l)})$ to be ν relaxation sweeps on $A^{(l)}\mathbf{x}^{(l)} = \mathbf{b}^{(l)}$.

(2) CYCLE$_k(\mathbf{x}^{(k)},\mathbf{0})$.

(3) If not sufficiently coarsened, form interpolation and its coarse-grid operator. Let $\mathbf{x}^{(k+1)} = (\mathbf{x}^{(k)})_c$ (that is, $\mathbf{x}^{(k)}$ evaluated at the grid $k+1$ points), $k = k+1$, and goto Step 2.

Otherwise, continue.

(4) While $k > 1$, let $k = k - 1$, interpolate the coarse-grid approximation, $\mathbf{x}^{(k)} = I_{k+1}^{k}\mathbf{x}^{(k+1)}$, and perform CYCLE$_k(\mathbf{x}^{(k)},\mathbf{0})$.

(5) Let $k = 1$ and $\mathbf{x}^{(1)}$ be a random vector. For all grids l, redefine the cycle, CYCLE$_l(\mathbf{x}^{(l)},\mathbf{b}^{(l)})$, to be ν current V-cycles on $A^{(l)}\mathbf{x}^{(l)} = \mathbf{b}^{(l)}$. If the performance of CYCLE$_1(\mathbf{x}^{(1)},\mathbf{0})$ is not acceptable, go to Step 2.

5.1.2 Adaptive MG Principles

Perhaps the easiest way to understand the adaptive methodology is to begin with the principles upon which it is based. Here, we list the core ideas that motivate and provide a foundation for the αMG methods, with the primary focus on the αAMG

scheme. The pragmatic reader may prefer to defer reading this discussion until after Sections 5.2 and 5.4.

Smoothness: The concept of algebraic smoothness is of utmost importance in achieving an optimally efficient algebraic multigrid method. Since we only allow reduction of the error through the processes of relaxation and coarse-grid correction, the algebraically smooth error (which, by definition, is slow to be resolved by relaxation) must be accurately corrected from the coarse grid. That is, interpolation must be very accurate for algebraically smooth components. In fact, a stronger requirement is imposed by the eigenvector approximation criterion that, for a given eigenvector, $\mathbf{u}$, of A, interpolation must reconstruct $\mathbf{u}$ to an accuracy proportional to its eigenvalue [18, 62].

The algebraic multigrid methods considered here are based on some defining characteristic of what algebraic smoothness means. This definition generally amounts to articulating an algebraic property of the errors that the given relaxation process cannot reduce effectively. For example, classical AMG is developed based on properties of a polynomial iterative method such as the Richardson iteration:

$$\tilde{\mathbf{x}} \leftarrow \tilde{\mathbf{x}} - \frac{1}{\|A\|}(A\tilde{\mathbf{x}} - \mathbf{b}).$$

For this iteration, the error, $\mathbf{e} = A^{-1}\mathbf{b} - \tilde{\mathbf{x}}$, converges slowly in the A-norm,

$$\|\mathbf{e}\|_A = \sqrt{< A\mathbf{e}, \mathbf{e} >},$$

if and only if $\mathbf{e}$ yields a small generalized Rayleigh quotient:

$$RQ_A(\mathbf{e}) = \frac{< A\mathbf{e}, A\mathbf{e} >}{\|A\| < A\mathbf{e}, \mathbf{e} >}.$$

Proper use of this defining property of algebraic smoothness gives AMG its potential for optimal performance over a wide range of problems. It enables coarsening processes that rightfully depend on the matrix and hopefully capture the errors that relaxation cannot eliminate.

Almost all algebraic multigrid methods, however, make additional assumptions regarding algebraically smooth error that allow them to capitalize on the special nature of algebraic smoothness that is assumed. For example, classical AMG rests on two main assumptions: that the constant vector, $\mathbf{1}$, must be interpolated exactly; and that algebraically smooth errors vary slowly along strong connections. While this enables effective treatment of many problems, it also restricts the class to which these algorithms apply. Many discrete systems exhibit algebraically smooth errors that vary dramatically across strong connections and many others offer no clear understanding of what strength of connection even means. Also, as we discuss further in the next section, $\mathbf{1}$ is not necessarily a good representative of algebraically smooth error. A major goal of the adaptive process is to capitalize on the definition of algebraically smooth error without making additional specific assumptions about its character.

Prototypes: A central idea in the development of αAMG methods is the use of prototypes that serve as representatives of algebraically smooth error. In fact, prototypes are used in the development of nearly all multigrid methods. As mentioned above, for example, classical AMG uses $\mathbf{1}$ to build its matrix-based interpolation coefficients (see Equation 3.1 and the discussion in Section 3.3). αAMG and αSA differ in that they attempt to generate their prototypes automatically.

Given a set of these prototypes, it is important to recognize that they should only be used locally as representatives of algebraically smooth error. Otherwise, it would not be possible to achieve optimality. As an illustration of this point, consider the fact that a non-optimal preconditioned conjugate gradient method can be formulated as an αMG method that uses the generated prototypes globally as representatives of slow-to-converge error components (here, the smoother is the preconditioner and the coarse-grid corrections are the Krylov subspace projections; see [42] for details). In general, errors that are left by relaxation consist of a large fraction of the spectrum of the matrix, so that coarsening must effectively approximate $O(n)$ components with varying degrees of

accuracy. The goal is to achieve this approximation property by only using a small, $O(1)$, number of computed prototypes.

The use of a small number of prototypes to achieve approximation of a much larger space is a cornerstone of multigrid methods. It is essential that each prototype be used effectively as a representative of many components with similar local character. Remember that the constant vector, $\mathbf{1}$, is used in classical AMG to define an interpolation whose range not only includes $\mathbf{1}$, but also approximates all smooth errors. This is analogous to how local basis functions are used in finite elements: piecewise polynomial basis functions are used locally not only because they can reconstruct their global counterparts, but also because they can approximate all smooth components of the solution of the PDE. The global prototype is a representative of many algebraically smooth components, and thus is used locally to determine an interpolation operator that has many such smooth components in its range.

Self-Testing: Computation of a rich supply of prototypes can be done by carefully testing the algorithm as it evolves. These self-tests should be done on a problem with known solution. The homogeneous problem, $A\mathbf{x} = \mathbf{0}$, is especially appropriate because it avoids trouble with machine representation when the approximation is very close to that solution. For our αMG schemes, we can test the current version of the algorithm on $A\mathbf{x} = \mathbf{0}$ by measuring the A-norm of the error of successive iterates. This test serves a dual role: it signals when the algorithm is performing well enough and it produces a good prototype when it is not. Assuming that enough iterations are used, the prototype must be appropriate because it is algebraically smooth (relaxation is not eliminating it), yet poorly represented by whatever current coarsening process is being used (if any). This prototype can then be used in the underlying algorithm precisely where the additional smoothness assumptions were used. For classical AMG, this means that the prototype would provide information on the correct coefficients to use in eliminating the matrix connections to points that are only on the fine level.

While the homogeneous problem is important as a measure of performance because it has a known solution, other measures can be useful in monitoring the evolving behavior and improving the prototypes. This issue is most clearly exposed when a direct solver is used on the coarsest level, where solving the homogeneous problem seems paradoxical: why solve $A\mathbf{x} = \mathbf{0}$ when all you presumably get is $\mathbf{x} = \mathbf{0}$? At this point, a simple approach is to just accept the prototype computed on the next finer level so that the coarsest level is never really used in the adaptive process. This means that the prototype is never really improved there either. It may be better to enhance the coarsest-level prototype by using a more precise measure of smoothness. For our αMG schemes, we could choose to improve the prototype, $\mathbf{x}$, on the coarsest level by minimizing the generalized Rayleigh quotient, $RQ_A(\mathbf{x})$. This approach becomes less clear, however, when there are several prototypes because of the need to distinguish between them. It may be necessary to also use a Ritz projection and, perhaps, a Rayleigh quotient involving the correction operator from the current method (RQ_{BA}, where $I - BA$ represents the current error propagation matrix) or a more sophisticated measure [43]. In the scalar PDE case considered in §5.2, as well as in the αSA method in §5.4, we have not yet found any need for improving the coarsest-level prototype, so this is not addressed further in what follows. The Rayleigh quotient can, however, be useful as a diagnostic tool in assessing how the adaptive process is behaving, as in Section 5.3.

Range of Interpolation: The primary aim of coarsening in any multigrid process is to allow algebraically smooth error to be represented by the range of interpolation. For the adaptive process, this means approximating the prototypes as much as possible. When enough DOFs are used on the coarse level, it is possible to fit them exactly: $\mathbf{x} = P\mathbf{x}_c$ for some coarse-level $\mathbf{x}_c$. This is not, however, sufficient for achieving good multigrid performance because, for example, piecewise-constant interpolation can fit the constant vector exactly but leads to poor multigrid performance [16]. In fact, in our αAMG scheme (described in Section 5.2.1), $\mathbf{x}_c$ is chosen simply as $\mathbf{x}$ restricted to the coarse

points, and so $\mathbf{x} = P\mathbf{x}_c$ only in the case that $A\mathbf{x} = \mathbf{0}$.

The fine-level problem is coarsened so that the coarse-grid representation of a prototype is just as good as the fine-grid representation, but requires less effort to resolve. For this reason, we can improve our representation of the near null space on the coarse grid, but only if the range of interpolation admits an algebraically smoother component than the prototype upon which interpolation was based. By using the prototype locally, we ensure that this is possible. Thus, in the adaptive process, we overwrite each fine-level prototype by its coarse-level interpolant that is, in general, a better prototype.

Optimality: Multigrid methods are useful solution techniques because they exhibit optimal traits, such as $O(N)$ or $O(N \log N)$ scaling in both number of operations and storage. As such, any adaptive multigrid process should also retain this optimality. In particular, the adaptive process must not make requirements of the solver that compromise the optimality of the overall process and must itself scale optimally in operation count and storage.

Classical AMG controls complexity by its intricate way of determining the coarse points and its careful use of the matrix entries. The adaptive AMG approach assumes that a suitable coarsening process is available (such as the compatible relaxation in [43, 61] and Section 6.2.2), with the attendant assumption that there is sufficient reduction in grid sizes from fine to coarse levels. When fitting multiple prototypes, however, it is tempting to abandon the tight control on the stencil of interpolation (such as to the nonzero pattern of A_{fc}, the submatrix of A linking fine- to coarse-grid nodes, as is used in classical AMG) to allow for exact fitting of more prototypes. This must be done with utmost care, as each new nonzero entry in the interpolation operator can lead to new nonzero connections in the coarse-grid matrix. Care must also be taken to ensure that the stencil of interpolation is not too small: early experiments limited the size of the set of coarse-grid points from which any fine-grid point i is interpolated, C_i, to the number of prototypes being fit, which led to an extremely inefficient algorithm because

a single prototype could only be used to define one-sided interpolation, and so multiple prototypes were needed for good convergence even for second-order, scalar PDEs.

Constructing a prototype set of minimal size is important to practical success and control over the complexity of the algorithm. While the same near null space may be well-represented by a small number of very precise vectors or a larger number of less-resolved prototypes, the costs of the adaptive process and, possibly, the resulting method increase with the number of prototypes. This is seen in αSA, where the prototype set is locally orthogonalized when determining interpolation to ensure new columns of the prolongator (and thus coarse-grid points) are not introduced unnecessarily. For this reason, it is more efficient to consider improvement of the existing prototype(s) than to add a new prototype. As prototypes emerge, we can consider improvement of their representation by removing each in turn from the prototype set, constructing the multigrid method based on this reduced set, and then applying the reduced multigrid method to the removed prototype. This either improves the prototype as a representative of the smooth components that are not well-represented by the rest of the prototype set, or signals that the removed prototype is not needed in the set and the reduced multigrid algorithm can replace the previous one. In either case, the prototype set is improved, either by reducing its size or by enhancing its representation of algebraically smooth error.

To ensure that the adaptive process is also optimal, adaptations are made whenever sufficient new information becomes known, but also only when the change is expected to improve the overall algorithm. For example, we develop the algebraically smooth prototype in a full approximation scheme manner. This means that the prototype on a given level is discarded when it can be improved from a coarser grid. We do not, however, update interpolation or coarse-grid operators on the upward traverse of the setup V-cycle. Such an adaptation would be wasted because operators at higher levels will also change as the cycle moves toward the finest grid. For this reason, while

we allow for multiple V-cycles to be performed in the setup phase, the last cycle always terminates at the coarsest grid.

Termination of the adaptive process must also be properly implemented in order to maintain optimality. Experience has shown that improvement in the resulting multigrid process becomes less cost-effective with the number of setup phases and the total amount of relaxation in the adaptive step. A method with an acceptable convergence factor may be attained after even a single adaptive step, and a second adaptive step improves this factor by only a fraction of a percent. This may be addressed by reducing the amount of relaxation per adaptive step to a single sweep on each level, and monitoring the convergence of the prototype vector between sweeps (for example, measuring its Rayleigh quotient). Unfortunately, the majority of the cost of an adaptive step is in the computation of interpolation and coarse-grid operators and not relaxation, so performing many adaptive steps is undesirable. For this reason, we choose a strategy involving a minimal number of setup cycles, motivated in Section 5.2.4, with enough relaxations in these cycles to quickly achieve convergence factors within a few percent of the apparent optimal performance.

5.2 Adaptive Algebraic Multigrid Methods

Our goal in developing a new type of algebraic multigrid method is to extend the applicability of classical AMG schemes. Thus, a primary concern is the generalization of the definition of interpolation in AMG. The guiding principles for this generalization come from basic properties of all multigrid algorithms:

- simple relaxation is inefficient for solving $A\mathbf{x} = \mathbf{b}$ on error components, $\mathbf{e}$, whose residuals, $A\mathbf{e}$, are small relative to $\mathbf{e}$ in some sense; and
- efficient multigrid performance depends on effective complementarity of relaxation and coarsening so that they efficiently cooperate to eliminate all error

components.

In developing the new interpolation procedure, we consider the case of pure algebraic coarsening; however, for practical reasons, we chose to first implement the algorithm in the case of regular geometric coarsening. The numerical results presented in Section 5.2.5 are for this implementation in the case of a scalar PDE.

5.2.1 Definition of Interpolation

Since the success of our methods depends on the complementarity of relaxation and coarse-grid correction, a good starting point for defining interpolation is to consider a vector, $\mathbf{e}$, that is not quickly reduced by relaxation. Using a simple (pointwise) relaxation scheme, such as Gauss-Seidel, this also means that $A\mathbf{e} \approx \mathbf{0}$, or

$$a_{ii}e_i \approx -\sum_{j\in C_i} a_{ij}e_j - \sum_{k\in F_i} a_{ik}e_k,$$

where, as in Equation 3.2, we assume a splitting of N_i into C_i and F_i. Again writing error e_k, $k \in F_i$, as

$$e_k \approx \sum_{j\in C_i} \omega_{kj}^i e_j + \omega_{ki}^i e_i,$$

a general interpolation formula for point $i \in F$ is then:

$$e_i = -\sum_{j\in C_i} \left(\frac{a_{ij} + \sum_{k\in F_i} a_{ik}\omega_{kj}^i}{a_{ii} + \sum_{k\in F_i} a_{ik}\omega_{ki}^i} \right) e_j. \tag{5.1}$$

The αAMG interpolation is different from that used in classical AMG [73] in that $\{\omega_{kj}^i\}$ are chosen to depend on both the entries in A and a (computed) prototype, $\mathbf{x}^{(1)}$, that represents many algebraically smooth components. How this prototype is computed is the subject of Section 5.2.3.

To be specific about the choice of $\{\omega_{kj}^i\}$, consider the idea of twice-removed interpolation [23]. Suppose we have a point, i, whose neighbors have been partitioned into the two sets, C_i and F_i. The problem of collapsing the $F - F$ connections is equivalent

to that of determining a way to interpolate to point $k \in F_i$ from points $j \in C_i$ (or, more generally, $j \in C_i \cup \{i\}$). That is, we seek to write (as before)

$$e_k = \sum_{j \in C_i} \omega_{kj}^i e_j, \tag{5.2}$$

dropping the term $\omega_{ki}^i e_i$, under the assumption that k is as strongly connected to some point (or points) in C_i as it is to i. If there is a particular vector, $\mathbf{x}^{(1)}$, that we want to be in the range of interpolation, then we ask that Equation 5.2 hold when $\mathbf{e}$ is replaced by $\mathbf{x}^{(1)}$. This constraint with one vector, $\mathbf{x}^{(1)}$, fixes one DOF of the possibly many for set $\{\omega_{kj}^i\}$, but leads to a unique F_i-interpolation formula if it is restricted to be of the form $D^{-1}A_{fc}$, where D is a diagonal matrix. (This choice is motivated by the discussion in [25].) D is thus determined by

$$d_{kk}^i x_k^{(1)} = -\sum_{j \in C_i} a_{kj} x_j^{(1)}$$

or

$$d_{kk}^i = \frac{-\sum_{j \in C_i} a_{kj} x_j^{(1)}}{x_k^{(1)}}. \tag{5.3}$$

Thus, choosing $\omega_{kj}^i = (d_{kk}^i)^{-1} a_{kj}$ in Equation 5.2, the F_i-interpolation formula is

$$e_k = -\sum_{j \in C_i} \frac{a_{kj}}{d_{kk}^i} e_j.$$

Interpolation to $i \in F$, given by Equation 5.1, then has the particular form

$$e_i = -\sum_{j \in C_i} \frac{1}{a_{ii}} \left(a_{ij} + \sum_{k \in F_i} a_{ik} \frac{a_{kj} x_k^{(1)}}{\sum_{j' \in C_i} a_{kj'} x_{j'}^{(1)}} \right) e_j. \tag{5.4}$$

Note that the interpolation operator, P, as a mapping from C to $F \cup C$, then has the form

$$P = \begin{bmatrix} W \\ I \end{bmatrix},$$

where W is the matrix of coefficients determined by Equation 5.4.

This αAMG interpolation formula is a simple generalization of the classical AMG formula that allows for a sense of smoothness that may differ from what AMG conventionally uses. The primary assumption used in standard AMG to collapse $F-F$ connections is that the smoothest error component is constant [73]. Thus, classical AMG interpolation is recovered from the formula in Equation 5.4 by choosing $\mathbf{x}^{(1)} \equiv \mathbf{1}$.

The iterated interpolation of Equations 5.2 and 5.3 was chosen to exactly match the near null space approximation, $\mathbf{x}^{(1)}$. The final interpolation in Equation 5.4, however, does not necessarily match this vector exactly. For a fine-grid point, i, the misfit is easily calculated as

$$x_i^{(1)} - (P\mathbf{x}_c^{(1)})_i = \frac{1}{a_{ii}}(A\mathbf{x}^{(1)})_i. \tag{5.5}$$

This is in accord with the classical AMG point of view that interpolation must be more accurate for errors that yield smaller residuals. In fact, it may be directly compared with the eigenvector approximation criterion, as described by Brandt [18, Theorem 4.1], which relies on the existence of a constant, C_0, such that

$$C_0 \sum_i a_{ii}(e_i - (P\mathbf{e}_c)_i)^2 \leq \mathbf{e}^T A\mathbf{e},$$

for all $\mathbf{e} \in \mathbb{R}^N$. Squaring Equation 5.5, multiplying through by a_{ii}, and summing, we have

$$\begin{aligned}\sum_i a_{ii}(x_i^{(1)} - (P\mathbf{x}_c^{(1)})_i)^2 = \sum_i \frac{1}{a_{ii}}(A\mathbf{x}^{(1)})_i^2 &= (\mathbf{x}^{(1)})^T A \left(\operatorname{diag}(\frac{1}{a_{ii}})\right) A\mathbf{x}^{(1)} \\ &\leq \rho\left(A^{\frac{1}{2}}\left(\operatorname{diag}\left(\frac{1}{a_{ii}}\right)\right)A^{\frac{1}{2}}\right)(\mathbf{x}^{(1)})^T A\mathbf{x}^{(1)}.\end{aligned}$$

For a diagonally-dominant operator with constant diagonal, such as the finite-element Laplacian, $\rho\left(A^{\frac{1}{2}}\left(\operatorname{diag}\left(\frac{1}{a_{ii}}\right)\right)A^{\frac{1}{2}}\right)$ is easily bounded by 2 and, thus, the constant, C_0, in Brandt's bound is not made unduly small due to the misfit in the interpolation of $\mathbf{x}^{(1)}$. While such a bound is only for the prototype, $\mathbf{x}^{(1)}$, and not for any arbitrary fine-grid vector (as required by the eigenvector approximation criteria), we consider $\mathbf{x}^{(1)}$

to be an appropriate prototype of the algebraically smooth error for which this bound is most difficult to achieve and, thus, indicative of a good interpolation scheme.

While this section considers methods involving just one prototype vector, $\mathbf{x}^{(1)}$, appropriate for scalar PDEs, these concepts can also be generalized to systems. Consider discretizing a system so that its DOFs are located on the same grid, i.e., there are d DOFs co-located at each node. Since we seek to generalize the ideas from the scalar case, we start by generalizing the notation: A_{kj} becomes the $d \times d$ matrix of connections between the DOFs located at nodes k and those located at node j, the diagonal entries of D (D^i_{kk}) become $d \times d$ matrices, and $\mathbf{x}^{(1)}$ becomes the matrix, $X^{(1)}$, composed of d columns of distinct prototypes. Its restriction to the d DOFs at node k is denoted by $X^{(1)}_k$. The analogue of Equation 5.3 is then

$$D^i_{kk} = -\left(\sum_{j \in C_i} A_{kj} X^{(1)}_j\right)\left(X^{(1)}_k\right)^{-1}.$$

The F_i-interpolation formula for systems thus becomes

$$e_k = -\sum_{j \in C_i} (D^i_{kk})^{-1} A_{kj} e_j,$$

which yields the final nodal interpolation formula

$$e_i = -A^{-1}_{ii}\left(\sum_{j \in C_i}\left(A_{ij} + \sum_{k \in F_i} A_{ik}(D^i_{kk})^{-1} A_{kj}\right)\right) e_j.$$

5.2.2 Theoretical Properties

One situation that can cause difficulty for classical AMG is when the matrix is rescaled. For example, if A is the discretization of a Poisson-like problem, then it is generally true that A applied to $\mathbf{1}$ yields a relatively small residual: $A\mathbf{1} \approx \mathbf{0}$. This means that constant vectors are indeed algebraically smooth, as classical AMG presumes. Rescaling A by multiplying it on both left and right (to preserve symmetry) by a positive diagonal matrix can dramatically change this property. Thus, if A is replaced by $\hat{A} = SAS$ for some positive diagonal matrix S, then $\hat{A}(S^{-1}\mathbf{1}) \approx 0$, and

the new near null space component is actually $S^{-1}\mathbf{1}$. If the diagonal entries of S have significant variation in them, then $S^{-1}\mathbf{1}$ has a significantly different character than does $\mathbf{1}$. For classical AMG, this can cause a dramatic deterioration in convergence rates, although it can be prevented if the scaling is supplied to AMG so that the original matrix can essentially be recovered (as in [23]), but this is not always possible in practice. Fortunately, as the following result shows, such scaling causes no problem for αAMG, provided the scaled prototype can be accurately computed.

Theorem 3. *Given a positive diagonal matrix, S, vectors $\mathbf{x}^{(1)}$, $\hat{\mathbf{x}}^{(1)} = S^{-1}\mathbf{x}^{(1)}$, $\mathbf{b}$, and $\hat{\mathbf{b}} = S\mathbf{b}$, then the convergence of αAMG on $\hat{A}\hat{\mathbf{x}} = \hat{\mathbf{b}}$ with prototype $\hat{\mathbf{x}}^{(1)}$ (measured in the $\hat{A}$-norm) is equivalent to that of αAMG on $A\mathbf{x} = \mathbf{b}$ with prototype $\mathbf{x}^{(1)}$ (measured in the A-norm).*

Proof. Given a coarse-grid set, C, and the complementary fine-grid set, F, partition A and S to have the forms

$$A = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \text{ and } S = \begin{bmatrix} S_f & 0 \\ 0 & S_c \end{bmatrix},$$

so that

$$\hat{A} = \begin{bmatrix} S_f A_{ff} S_f & S_f A_{fc} S_c \\ S_c A_{cf} S_f & S_c A_{cc} S_c \end{bmatrix}.$$

The weights, $\hat{\omega}^i_{kj}$, for matrix $\hat{A}$ are given by

$$\begin{aligned} \hat{\omega}^i_{kj} &= \frac{\hat{a}_{kj}\hat{x}^{(1)}_k}{\sum_{j' \in C_i} \hat{a}_{kj'}\hat{x}^{(1)}_{j'}} \\ &= \frac{s_k a_{kj} s_j s_k^{-1} x^{(1)}_k}{\sum_{j' \in C_i} s_k a_{kj'} s_{j'} s_{j'}^{-1} x^{(1)}_{j'}} = s_k^{-1}\omega^i_{kj} s_j, \end{aligned}$$

where the weights, ω^i_{kj}, are chosen as for matrix A. Equation 5.4 then gives (with some

algebra)

$$e_i = -\sum_{j \in C_i} s_i^{-1} \left(\frac{a_{ij} + \sum_{k \in F_i} a_{ik}\omega_{kj}^i}{a_{ii}} \right) s_j e_j, \quad i \in F.$$

For $i \in C$, again simply take the value from the coarse-grid and assign it as the value on the fine-grid. Thus, the interpolation operator, $\hat{P}$, is of the form

$$\hat{P} = \begin{bmatrix} \hat{W} \\ I \end{bmatrix} = \begin{bmatrix} S_f^{-1} W S_c \\ I \end{bmatrix} = S^{-1} P S_c,$$

where P is the interpolation operator from the unscaled case. Further, considering the coarse-grid operator, $\hat{A}_c$, note that $\hat{A}_c = S_c P^T A P S_c = S_c A_c S_c$, where A_c is the coarse-grid operator from αAMG on A.

So, the coarse-grid operator for the scaled problem is simply the scaled version of the coarse-grid operator for the unscaled problem. Since standard relaxation techniques such as Gauss-Seidel or Jacobi (both pointwise and block forms) are scaling invariant (that is, if A is scaled to SAS as above, initial guess $\mathbf{x}^{(0)}$ to $S^{-1}\mathbf{x}^{(0)}$ and initial right side $\mathbf{b}$ to $S\mathbf{b}$, then the approximation generated changes from $\mathbf{x}^{(1)}$ to $S^{-1}\mathbf{x}^{(1)}$), we see that the entire process is independent of any diagonal scaling. □

Theorem 4. *Theorem 3 extends to the systems algorithm, which is invariant to diagonal scaling with pointwise relaxation and nodal scaling with nodal relaxation.*

Proof. The proof is identical in form to the scalar case, and is thus omitted. □

5.2.3 Determining $\mathbf{x}^{(1)}$

Successful implementation of these schemes for interpolation relies on having an appropriate prototype vector, $\mathbf{x}^{(1)}$ (or set of vectors, $X^{(1)}$). Since we rely on the complementarity of relaxation and coarsening, the best choice for $\mathbf{x}^{(1)}$ would be a representative of the vectors for which relaxation is inefficient. Thus, a straightforward method for generating this prototype would be to start with a vector that is hopefully rich in all

components (i.e., all eigenvectors of symmetric A), relax on $A\mathbf{x} = \mathbf{b}$ for some $\mathbf{b}$, and then determine the error in the approximate solution after a sufficient number of relaxations.

We typically make use of relaxation schemes whose error-propagation matrices have the form $I - BA$. While it is possible that the slow-to-converge modes of the relaxation iteration, $I-BA$, are not modes for which $A\mathbf{e} \approx \mathbf{0}$, in most practical situations they are. In particular, for the pointwise relaxation schemes considered here, the two descriptions of algebraically smooth error are equivalent. In fact, for many choices of B, the true near null space of A is accurately reflected in the vectors for which relaxation is inefficient. Knowledge of this space could be used as it is with standard AMG to determine an effective coarsening process. Our focus, however, is on the case where this knowledge is inadequate or even unavailable. We thus concentrate on the case that a good prototype, $\mathbf{x}^{(1)}$, is not known.

Start with a vector generated randomly from a uniform distribution on $(0, 1)$. (Consideration of positive vectors is motivated by the case of scalar, second-order operators, which tend to have positive near null space vectors. A more general choice is appropriate when considering problems such as linear elasticity, but care must be taken because the definition of interpolation for αAMG presented here breaks down if $x_j^{(1)} = 0$ for any coarse-grid node, j.) Such a vector is, in general, not equally rich in all error components. However, in the scalar PDE case, it tends to be rich enough that a few relaxation sweeps on the homogeneous problem, $A\mathbf{x} = \mathbf{0}$, produces a good representative of the slow-to-converge components. Note that the homogeneous problem is advantageous to use here because the prototype is simply the error in approximating the exact solution, $\mathbf{x} = \mathbf{0}$. Thus, starting with a random initial guess and performing relaxation on $A\mathbf{x} = \mathbf{0}$ generates a prototype vector, $\mathbf{x}^{(1)}$, that represents the slow-to-converge components and that can then be used in the interpolation formula developed in Section 5.2.1.

Unfortunately, generating the prototype by fine-grid relaxation alone is effective

only in terms of the first coarse level and is, in general, quite inefficient in a multilevel setting. To produce a prototype that is smooth enough to represent the components associated with smoothness on very coarse levels, a multilevel scheme is needed. Here, we again measure smoothness by the eigenvector approximation criterion. Basing every interpolation operator on a single component, whose Rayleigh quotient is near the minimal eigenvalue on the finest grid, requires significant smoothness in that component as the eigenvector corresponding to this eigenvalue must be interpolated with accuracy proportional to this small value. For coarser grids, such smoothness is much more efficiently represented by calculation on these grids. Thus, we start with a random guess on the fine level and perform a few (ν_0) relaxation sweeps there to generate a tentative $\mathbf{x}^{(1)}$. Using this current prototype, an interpolation operator is computed (as in Section 5.2.1) and the coarse-level and restriction operators are formed using the Galerkin condition. We use injection (direct restriction of the values on the C-points) to form a coarse-level initial guess, relax ν_1 times, and recurse to the coarsest level. From this coarsest level, interpolate and relax ν_2 times on the vector all the way to the finest level, but do not recompute the coarse-level and restriction operators.

The cycle can then be repeated, using the resulting vector as an overall initial guess in an attempt to improve the prototype. This cycle may proceed as before, simply by performing relaxation on each level, or it may be augmented by replacing the relaxation stage with the current multilevel solver. Use of the current solver may be preferred as this exposes errors that are both algebraically smooth and that are not being quickly reduced by the current coarse-level correction scheme. In the case of scalar-PDE based matrices, such error is a good prototype upon which to base relaxation. For matrices with a higher-dimensional near null space, interpolation should be adapted to account for this prototype as well as any other prototypes currently being approximated. This cycling strategy is illustrated in Figure 5.1, where boxes indicate stages where coarse-level operators are computed and circles indicate stages where only application

of the current solver is necessary. Note that since the multigrid operators are computed only on the downward part of the cycle, no relaxation is necessary on the upward part of the final setup cycle. As is discussed in Section 5.2.5, this procedure yields multigrid solvers with level-independent convergence factors tested up to 1024×1024 grids for many scalar problems.

One important benefit of generating the initial prototype vector in a multilevel fashion is the ability to implement a proper transition to simplicity in the algorithm. That is, since we begin by relaxing on a random vector (assumed to be rich in all components), it is easy to tell if relaxation is sufficient to solve either the fine grid problem or one of the generated coarse-level problems. If this is indeed the case, then no additional labor is needed in designing an algorithm because an efficient solver already exists.

For systems of PDEs and higher-order problems, an added wrinkle is the need to generate multiple prototype vectors. We expect that a technique similar to the one described above can generate the components well enough to produce an efficient multigrid scheme, but further investigation is necessary to ensure that the generated prototypes are rich enough and are not redundant. We expect that a strategy similar to the one developed for αSA can be used to produce an effective αAMG approach for systems, which is the subject of current research.

5.2.4 Setup Cost Considerations

An important factor to be considered in the cost of the αAMG algorithm is the cost of the setup phase. Normal (solution) cycling costs the same as any other multigrid method, but the setup costs incurred can be much higher than other multigrid algorithms, including those that use operator-induced coarsening. Here, we analyze the costs of the setup procedure in the structured case, and consider the question of what makes an efficient setup routine.

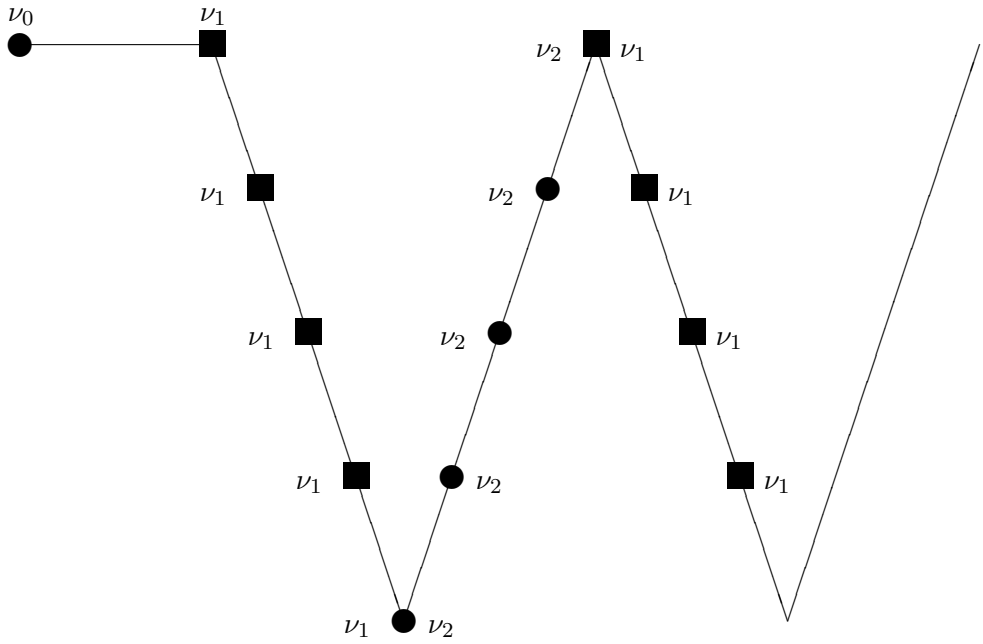

Figure 5.1: The Setup Scheme for Determining $x^{(1)}$

5.2.4.1 Theoretical Costs

We consider the general setup routine appropriate for a bilinear finite-element discretization of a scalar PDE. In the current code, this comprises some initial fine-grid relaxations, followed by adaptive cycles with significant costs from relaxation, the definition of interpolation, and the RAP multiplication. We define the following variables to describe the setup process:

$\nu_0 =$ the number of initial fine-grid relaxations.

$M =$ the number of setup cycles to be run.

$\nu_1 =$ the number of relaxations on the downward side of each setup cycle.

$\nu_2 =$ the number of relaxations on the upward side of each setup cycle.

We now describe our setup (with a fixed number of cycles, M) as

Algorithm 3 (Single-Prototype αAMG Setup).

(1) Let $m = 1$, $k = 1$ and $\mathbf{x}^{(1)}$ be a random vector. Define, for all grids l, the methods CYCLE$_l(\mathbf{x}^{(l)},\mathbf{b}^{(l)})$ to be 1 relaxation sweep on $A^{(l)}\mathbf{x}^{(l)} = \mathbf{b}^{(l)}$.

(2) Apply ν_0 cycles to the homogeneous fine-grid problem, CYCLE$_1^{\nu_0}(\mathbf{x}^{(1)},\mathbf{0})$.

(3) Apply ν_1 cycles to the homogeneous level k problem, CYCLE$_k^{\nu_1}(\mathbf{x}^{(k)},\mathbf{0})$. If convergence is fast, return to previous flow at Step 5a with $\mathbf{x}^{(k)}$ as before relaxation.

(4) If not sufficiently coarsened,

  (a) Compute interpolation, I_{k+1}^k.

  (b) Compute the coarse-grid operator, $A^{(k+1)} = \left(I_{k+1}^k\right)^T A^{(k)} I_{k+1}^k$.

  (c) Let $\mathbf{x}^{(k+1)} = (\mathbf{x}^{(k)})_c$ (that is, $\mathbf{x}^{(k)}$ evaluated at the grid $k+1$ points), $k = k+1$, and goto Step 3.

Otherwise, continue.

(5) While $k > 1$, let $k = k - 1$, and

   (a) Interpolate the coarse-grid approximation, $\mathbf{x}^{(k)} = I_{k+1}^{k}\mathbf{x}^{(k+1)}$

   (b) Apply ν_2 cycles to the homogeneous level k problem, $\mathrm{CYCLE}_k^{\nu_2}(\mathbf{x}^{(k)},\mathbf{0})$.

(6) Let $k = 1$ and $\mathbf{x}^{(1)}$ be a random vector. Define, for all grids l, $\mathrm{CYCLE}_l(\mathbf{x}^{(l)},\mathbf{b}^{(l)})$ to be ν current V-cycles on $A^{(l)}\mathbf{x}^{(l)} = \mathbf{b}^{(l)}$. If $m < M$, increment m and go to Step 3, otherwise exit.

Under the assumption of a structured (tensor-product) 2-D mesh, the bilinear FE discretization gives 9-point operators. Choosing full-coarsening based grids and constraining interpolation to have the same sparsity structure as A_{fc}, this 9-point structure is maintained on all coarse grids. Full coarsening results in a reduction in grid size by a factor of approximately $\frac{1}{4}$ in each coarsening. Thus, the cost of an operation performed on each grid can be approximately bounded by $\frac{4}{3}$ of its cost on the finest grid. We use fine-grid work units (the cost of a residual evaluation or matrix-vector product on the finest grid) as our primary cost measure.

We can now estimate the costs of the setup phase in fine-grid work units as

$$\nu_0 + \frac{4M}{3}\left(\mathrm{CGO} + \mathrm{INTERP} + \mathrm{P}\right) + \frac{4}{3}(\nu_1 + \nu_2) + \frac{4}{3}(M-1)\left(\frac{8}{3}\nu_1 + \frac{8}{3}\nu_2 + 2\mathrm{P}\right),$$

where INTERP represents the cost of forming the interpolation to the finest-grid, CGO represents the cost of performing the RAP triple product to form the Galerkin coarse-grid operator, and P represents the cost of interpolating the representative vector $\mathbf{x}$ to the finer grid. This estimation can be slightly improved by recalling that any relaxation or interpolation on the upward step of the final setup cycle can be omitted. The revised estimate is then

$$\nu_0 + \frac{4}{3}\left(\nu_1 + \mathrm{CGO} + \mathrm{INTERP}\right),$$

if $M = 1$, or

$$\nu_0 + \frac{4M}{3}\left(\text{CGO} + \text{INTERP} + \text{P}\right) + \frac{4}{3}\left(\nu_1 - \frac{5}{3}\nu_2 - \text{P}\right) + \frac{4}{3}(M-1)\left(\frac{8}{3}\nu_1 + \frac{8}{3}\nu_2 + 2\text{P}\right),$$

if $M > 1$.

The costs of Step 5a in Algorithm 3 are the easiest to compute, given the structure of interpolation in αAMG, as $P = \left[\begin{smallmatrix} W \\ I \end{smallmatrix}\right]$, with W having the same sparsity structure as A_{fc}. Noticing that, with a full-coarsening grid, A_{cc} is diagonal, a column of P has the same number of non-zero elements as a column of A. So, assuming an operator of fixed stencil size (such as our nine-point matrix A), the cost of the interpolation $P\mathbf{x}$ is simply $\frac{n_c}{N}$ work units, where n_c is the size of the coarse-grid and N is the size of the (full) fine grid. Assuming a coarsening ratio of $\frac{1}{4}$, this means that the cost of interpolation to the fine grid is also $\frac{1}{4}$ work units.

The cost of determining interpolation is much more significant, because, for each fine-grid point, i, a double sum must be evaluated for each nonzero in the i^{th} row of P, as in Equation 5.4.

For each fine-grid point i, for each of its coarse-grid neighbors, we evaluate the term in parenthesis in Equation 5.4. Each of these terms requires a negation, an addition, a division, and the evaluation of a sum over the fine-grid neighbors of i (requiring $|F_i| - 1$ additions). Each term in this sum, in turn, requires two multiplications and a division, as well as the evaluation of a sum over the coarse-grid neighbors of i (with $|C_i| - 1$ additions). Each term in this final sum requires a multiplication. Thus, the innermost sum requires $|C_i|$ multiplication and $|C_i| - 1$ additions. The middle sum then requires $|F_i|$ evaluations of the inner sum, $|F_i|$ times 3 flops, plus $|F_i| - 1$ additions in its evaluation. Determining the i^{th} row of P requires $|C_i|$ evaluations of the middle sum, as well as $|C_i|$ negations, additions, and divisions. Thus, the total cost in operations of defining P is given by

$$\sum_{i \in F} |C_i| \left(|F_i| \left((2|C_i| - 1) + 4\right) + 2\right).$$

There are now three types of fine-grid points in our fully-coarsened grid. One-fourth of the grid is composed of points that lie at the center of a coarse-grid cell, having 4 coarse-grid neighbors and 4 fine-grid neighbors. One-fourth lie at midpoints of horizontal coarse-grid lines, and one-fourth lie at midpoints of vertical coarse-grid lines, both having 2 coarse-grid neighbors and 6 fine-grid neighbors. The final fourth of the fine-grid nodes are also coarse-grid nodes. Thus, the sum above can be written as

$$\frac{N}{4}\left(4\left(4\left((2\cdot 4-1)+4\right)+2\right)\right)+\frac{2N}{4}\left(2\left(6\left((2\cdot 2-1)+4\right)+2\right)\right)=90N\text{flops}.$$

Now, one work unit equals the cost of a residual evaluation. For our 9-point operator, this takes $18N$ flops (one addition and one multiplication per non-zero in the matrix). Thus, we find that forming interpolation costs $\frac{90}{18}=5$ work units.

This calculation does, however, make use of the assumption that there is no cost to computing a needed entry in the matrix A. This is, unfortunately, not realistic with CSR storage. Instead, we must compute the location of a row of A and then search all of the non-zero entries in that row for the correct column. Thus, the true cost of computing P is expected to be higher than just that of the floating point operations needed to form each non-zero entry in P.

We could determine the cost of forming the Galerkin coarse-grid operator by analyzing the terms in the triple product, but find it more convenient to analyze the looping structure of the algorithm used to compute RAP in compressed-sparse-row form. This structure can be simplified to its essentials:

- for $i = 1$ to n_c,
  - for $ii = R_i(i)$ to $R_i(i+1)-1$,
    - for $j = A_i(R_j(ii))$ to $A_i(R_j(ii)+1)-1$,
      - Perform one multiplication.
      - for $jj = P_i(A_j(j))$ to $P_i(A_j(j)+1)-1$,
        perform one addition and one multiplication.

Considering the looping, the innermost for-loop is executed once for each nonzero in row $A_j(j)$ of P. The next loop is executed once for each nonzero in row $R_j(ii)$ in A, of which there are nine. The next loop is executed once for each nonzero in row i of $R = P^T$, of which there are also nine. Finally, the outermost loop is executed once for coarse-grid point, of which there are $\frac{N}{4}$. The number of non-zeros in row jj of P depends on which type of node jj corresponds to, but is either 1 (if jj is a coarse-grid node), 2 (if jj is a fine-grid node embedded in a coarse-grid line), or 4 (if jj is a fine-grid node at the center of a coarse-grid cell).

Assuming that the rows of P are accessed roughly equally, the innermost loop is executed an average of $\frac{9}{4}$ times. The next two loops are executed 9 times each, with the outermost loop being executed $n_c = \frac{1}{4}N$ times. Thus, the total number of operations is approximately $\frac{N}{4} \cdot 9 \cdot 9 \cdot (1 + \frac{9}{4} \cdot 2)$. Dividing this by the $18N$ operations in a work unit, we get that the cost of forming a coarse grid operator is approximately $\frac{9 \cdot 9 \cdot 11}{4 \cdot 2 \cdot 18} \approx 6$ work units.

Using these results, our setup cost estimate becomes

$$\nu_0 + \frac{4}{3}(\nu_1 + 11),$$

if $M = 1$, or

$$\nu_0 + 15M + \frac{4}{3}\left(\nu_1 - \frac{5}{3}\nu_2 - \frac{1}{4}\right) + \frac{4}{3}(M-1)\left(\frac{8}{3}\nu_1 + \frac{8}{3}\nu_2 + \frac{1}{2}\right),$$

if $M > 1$.

5.2.4.2 Measured Costs

Using a profiling tool, we can easily measure the costs of computing interpolation and coarse-grid operators relative to the cost of a known cycle. As a standard of measurement, consider the timing of a V(1,1) cycle, equivalent to approximately $\frac{8}{3}$ work units. Since such timings should not vary significantly based on the matrix coefficients,

Grid Size	V(1,1) Solution	Total Setup	CGO	INTERP	Setup Relaxation
2048×2048	1.45	26.75	9.36	13.71	0.43
1024×1024	0.34	6.59	2.40	3.32	0.10
512×512	0.087	1.46	0.43	0.81	0.03
256×256	0.027	0.36	0.10	0.20	0.02
128×128	0.006	0.08	0.02	0.05	0.01

Table 5.1: Timing Results (in Seconds) for V(1,0) αAMG Setup Cycles

we test using the 9-point finite element discretization of the Laplacian on $2^\ell \times 2^\ell$ element meshes. The test runs and samples were done on a dual-processor machine with 2.0GHz Intel XEON CPUs, and 4GB of RAM. Problems smaller than 128×128 elements ran too quickly on this machine to generate accurate timings. Results for grids from 128×128 to 2048×2048 elements, timing a single V(1,0) setup cycle are given in Table 5.1.

In general, the cost of computing the coarse-grid operators is about 6 times the cost of a V(1,1) cycle, while the cost of computing interpolation is closer to 9 times the cost of a V(1,1) cycle. This suggests that the true cost of computing a coarse-grid operator is about 13 work units, while the true-cost of computing interpolation is about 19 work units.

The relationship between the theoretically expected and computationally realized costs of computing both the interpolation and coarse-grid operators is quite poor. This may be due to a number of factors, including memory and cache access costs, the costs of computed addressing in CSR format, and inefficiencies in the programmed routines. Regardless of this mismatch, we see that the significant cost of the αAMG setup phase remains in the computation of the interpolation and coarse-grid operators.

5.2.4.3 Analysis

We consider this issue primarily to inform our decisions on the parameters of the setup algorithm, namely, M, ν_0, ν_1, and ν_2. Using the theoretically computed cost,

$$\nu_0 + \frac{4}{3}(\nu_1 + 11),$$

if $M = 1$, or

$$\nu_0 + 15M + \frac{4}{3}\left(\nu_1 - \frac{5}{3}\nu_2 - \frac{1}{4}\right) + \frac{4}{3}(M-1)\left(\frac{8}{3}\nu_1 + \frac{8}{3}\nu_2 + \frac{1}{2}\right),$$

if $M > 1$, keeping M small is very important. Note, for example, that for fixed nu's, decreasing M by 1 saves over 15 work units in theory and comparatively more in practice.

Following this reasoning, we suggest that M should, perhaps, be as small as possible. Forcing M to be 1, however, allows for no iterative improvement of the prototype vector, which has been shown numerically to be effective. So, we consider here also the case of $M = 2$, but seek strongly to avoid allowing any further growth.

5.2.4.4 Test Problems

The bottom line in this consideration is minimizing the real cost of setup, in computational time, over the (M, ν_0, ν_1, ν_2) parameter space. Guided by this, consider the performance of the αAMG solver (as measured by convergence factors of V(1,1) cycles) versus setup time for points in this parameter space. In particular, for fixed M, we find the point, (ν_0, ν_1, ν_2), that corresponds to the minimal setup time to achieve near-optimal asymptotic V(1,1) convergence factors for each problem. Considering these test problems, we can easily take M and (ν_0, ν_1, ν_2) large enough to achieve the optimal convergence factors that result from using the algebraically smoothest eigenvector of the matrix as the prototype. Knowing the truly optimal convergence factors, we then choose the values of M and (ν_0, ν_1, ν_2) such that the asymptotic V(1,1) convergence factor for a given problem is within 0.01 of the optimal factor. These points and the

times required for only the multigrid setup phase for the problems discussed below are shown in Table 5.2. Here, we disregard the solution time, under the assumption that the resulting solver will be used to solve the linear system for many right-hand sides and, thus, achieving a near-optimal convergence factor in a minimal amount of time is more important than the overall time to solution for any single right-hand side.

In Table 5.2, Problem 1 is the finite-element Laplacian, discretized on $[0,1]^2$ with Dirichlet boundary conditions. Problem 1r is a diagonally scaled version of Problem 1, with scaling of the form 10^{10r} for r chosen uniformly between 0 and 1.

Problem 2 is the finite-element Laplacian, discretized on $[0,1]^2$ with Neumann boundary conditions. Problem 2r is a diagonally scaled version of Problem 2, using the same scaling as in Problem 1r. Note that Problem 2 is, in general, a harder problem for any multigrid method, because the eigenvector approximation criterion suggests that we must approximate the null space exactly in interpolation.

Problem 3 is a finite-element discretization of $-\nabla \cdot \mathcal{K}(x,y)\nabla p(x,y)$ with Dirichlet boundary conditions on the left and right sides and Neumann boundary conditions top and bottom. Here, $\mathcal{K}$ is given to be 10^{-8} inside the square $[\frac{1}{3},\frac{2}{3}]^2$ and 1 everywhere else. Applying the AMG-like heuristic of the near-null space being constant achieves convergence factors of about 0.14, while the αAMG code can achieve factors of about 0.10 or better. Problem 3r is again a diagonally scaled version of Problem 3, using the same scaling as in Problems 1r and 2r.

These CPU timings indicate two things. First, for a fixed number of cycles, the amount of work per cycle increases with problem difficulty (size and jumps). This comes as no surprise, because relaxation takes longer to expose the smooth modes of these problems. The second indication is that the best setup scheme for the problems tested makes use of only one cycle. It is much harder to find the correct relaxation parameters for this, but, in all cases, the setup cycle with $M = 1$ was faster than choosing $M = 2$. Considering the trends, it is not clear if this would remain true for

Prob.	Size	$M = 1$				$M = 2$			
		ν_0	ν_1	ν_2	T	ν_0	ν_1	ν_2	T
1	128^2	2	2	0	0.10	1	1	0	0.17
1	256^2	3	3	0	0.35	1	1	0	0.67
1	512^2	4	5	0	1.55	1	1	0	2.71
1	1024^2	7	7	0	6.93	1	1	0	11.32
1r	128^2	7	6	0	0.08	1	1	1	0.18
1r	256^2	10	10	0	0.46	2	1	1	0.74
1r	512^2	16	15	0	2.04	1	2	1	3.00
1r	1024^2	23	22	0	10.01	1	2	1	12.39
2	128^2	4	4	0	0.09	1	1	0	0.20
2	256^2	6	6	0	0.44	2	1	1	0.74
2	512^2	9	9	0	1.83	2	2	1	3.22
2	1024^2	13	13	0	8.06	3	2	1	12.77
2r	128^2	12	11	0	0.14	3	2	1	0.22
2r	256^2	18	18	0	0.58	3	2	1	0.81
2r	512^2	27	27	0	2.69	5	3	3	3.53
2r	1024^2	38	38	0	13.34	5	4	3	15.01
3	128^2	4	4	0	0.10	1	1	0	0.20
3	256^2	4	4	0	0.38	1	1	0	0.68
3	512^2	6	6	0	1.58	2	1	1	3.00
3	1024^2	7	7	0	6.95	2	1	1	11.63
3r	128^2	6	6	0	0.10	2	2	1	0.19
3r	256^2	11	11	0	0.45	4	2	2	0.84
3r	512^2	13	13	0	2.02	4	3	2	3.43
3r	1024^2	21	20	0	9.73	5	4	3	15.08

Table 5.2: Parameters and CPU Times for Optimal Points in the Setup Parameter Space

larger grids, because in many cases, the relative costs of $M = 1$ and $M = 2$ became closer as the grid size increased.

5.2.5 Numerical Results

To examine the feasibility of this approach, we implemented a solver for the special case of a rectangular grid in two dimensions with full coarsening. This restriction in generality has a notable effect on the range of problems that are able to be reasonably considered (anisotropy, for example, becomes much more difficult to account for in this setting). However, examining problems without such difficulties, we feel that we can obtain a good initial indication of the performance of this approach. In particular, the aim here is to test the quality of the interpolation operator, not that of the coarsening procedure. Indeed, given current research into new coarsening techniques [61], it is difficult to say which coarsening method would be most appropriate for comparison.

We consider several measures of the effectiveness of the algorithm. Of primary importance is the total time to solution, given only the matrix equation, $A\mathbf{x} = \mathbf{b}$. Here, solving a problem is defined as reducing the residual by 10 orders of magnitude, and the wall-clock time to solution on a modern desktop workstation (2.66 GHz Intel Pentium 4) is measured. Another relevant measure is the asymptotic convergence factor of the resulting cycle. While setup costs may form a significant portion of the cost of solving a linear system with a single right side, many problems require repeated solution with multiple right sides (such as in implicit time stepping). In these cases, the (possibly large) setup cost can be amortized over the number of solutions and the cost of only the solution stage is important. The asymptotic convergence factor reflects this cost, as a lower factor requires fewer iterations in the solution phase. We discard the usual AMG measures of grid and operator complexity because, in the structured coarsening framework considered here, these measures are constant for all fine-grid operators of the same mesh and stencil sizes.

As discussed in Sections 5.2.3 and 5.2.4, the setup may be performed in an iterative fashion. This is an appealing feature because, in practice, convergence of the prototype vector can be measured as an indicator of convergence of the method. This iteration is, however, quite expensive as it involves computation of new interpolation and new Galerkin coarse-grid operators at each iteration. Testing, as in §5.2.4, indicates that it is usually significantly more efficient to perform more relaxation sweeps (i.e., increase ν_0 and ν_1) in a single setup cycle than it is to perform multiple setup cycles with fewer iterations per cycle to generate a single prototype. Thus, in the results that follow, we first consider performing only a single setup cycle and track the number of relaxations (values of ν_0 and ν_1) necessary to achieve highly efficient solver performance. This calibrated AMG method is typically more efficient than the adaptive AMG method, whose results appear later. In the adaptive procedure, we fix the number of relaxations used in each setup phase and perform setup iterations until an acceptable solver is determined (measured by the error reduction in the solver). Note that the adaptive approach is that of Algorithm 2 in Section 5.1.1 and not the iterative improvement approach. That is, the results in §5.2.5.3 reflect starting each subsequent cycle with a new random initial guess and redefining interpolation based only on this new prototype. Numerical results for iterative improvement of prototypes are not demonstrated here, rather they are shown in Sections 5.2.4.4 and 5.3.

We consider four PDEs as test problems, all discretized using bilinear finite elements on the unit square. Problem 1 is Laplace's Equation with pure Dirichlet boundary conditions. Problem 2 is Laplace's Equation with pure Neumann boundary conditions. Problems 3 and 4 are

$$-\nabla \cdot \mathcal{K}(\mathbf{x}) \nabla p(\mathbf{x}) = 0,$$

with Dirichlet boundary conditions on the East and West boundaries and Neumann

boundary conditions along the North and South boundaries. For Problem 3, $\mathcal{K}(\mathbf{x})$ is chosen as

$$\mathcal{K}(\mathbf{x}) = \begin{cases} 10^{-8} & \mathbf{x} \in [\frac{1}{3}, \frac{2}{3}]^2, \\ 1 & \text{otherwise.} \end{cases}$$

For Problem 4, $\mathcal{K}(\mathbf{x})$ is assumed to be constant on each element and chosen to have value 10^{-8} on 20% of the elements (chosen randomly) and value 1 everywhere else.

A significant advantage of the αAMG method is its invariance to diagonal scaling, as shown in Section 5.2.2. Thus, for each of these problems, we consider the results of such scaling. A common scaling is to make the diagonal entries of A all have value 1, using the diagonal matrix given by $s_{ii} = \frac{1}{\sqrt{a_{ii}}}$. For Problems 1-4 above, the problems with matrices thus scaled are referred to as Problems 1u-4u (where the u refers to the unit diagonal of the matrix). We also consider a more drastic scaling given by $s_{ii} = 10^{5r_i}$, where again r_i is chosen from a uniform distribution on $[0, 1]$ for each i. We call these Problems 1r-4r (where the r refers to the random scaling).

5.2.5.1 AMG Benchmarks

A baseline for these problems is established by considering the performance of standard AMG under the same assumptions (primarily that coarsening is performed geometrically). Since we expect the scalings employed to destroy any sense of strong connection in the matrix coefficients, we consider here a "strong-connection-only" version of AMG, equivalent to setting $\theta = 0$ in the definitions of strong influence and dependence in Section 3.3. Wall-clock times and iteration counts are shown in Table 5.3, while convergence factors are shown in Table 5.4.

These results show that classical AMG interpolation gives a scalable solver for Problems 1, 2, and 3, coming directly from discretization. If, however, the discretization matrices are scaled, then there is a significant increase in the needed work units for solution, especially noticeable as the problem grows (but also present with smaller fine

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.04 (9)	0.22 (9)	0.91 (9)	3.32 (9)	13.13 (9)
Problem 1u	0.05 (9)	0.24 (9)	0.86 (9)	3.33 (9)	13.15 (9)
Problem 1r	$3.3 \cdot 10^{-5}$	$3.6 \cdot 10^{-5}$	$2.5 \cdot 10^{-5}$	$1.8 \cdot 10^{-5}$	$1.3 \cdot 10^{-5}$
Problem 2	0.04 (8)	0.20 (8)	0.81 (8)	3.12 (8)	12.30 (8)
Problem 2u	0.09 (27)	0.93 (52)	5.64 (90)	36.94 (158)	$2.4 \cdot 10^{-9}$
Problem 2r	$4.5 \cdot 10^{-6}$	$3.8 \cdot 10^{-6}$	$4.3 \cdot 10^{-6}$	$3.4 \cdot 10^{-6}$	$3.5 \cdot 10^{-6}$
Problem 3	0.04 (9)	0.25 (9)	0.89 (9)	3.41 (9)	13.23 (9)
Problem 3u	0.11 (26)	0.78 (41)	4.73 (69)	29.81 (124)	$4.6 \cdot 10^{-10}$
Problem 3r	$4.0 \cdot 10^{-5}$	$3.0 \cdot 10^{-5}$	$1.8 \cdot 10^{-5}$	$1.4 \cdot 10^{-5}$	$9.6 \cdot 10^{-6}$
Problem 4	0.05 (12)	0.28 (12)	1.04 (11)	4.40 (13)	17.64 (14)
Problem 4u	0.18 (64)	3.14 (179)	$2.0 \cdot 10^{-6}$	$1.8 \cdot 10^{-5}$	$1.1 \cdot 10^{-5}$
Problem 4r	$4.5 \cdot 10^{-5}$	$2.5 \cdot 10^{-5}$	$1.7 \cdot 10^{-5}$	$1.2 \cdot 10^{-5}$	$9.3 \cdot 10^{-6}$

Table 5.3: Wall-Clock Time in Seconds (and Iteration Count), or Residual Reduction After 200 Iterations in Case of Failure for Standard AMG to Reduce Residuals by 10^{10}.

grids). Improving on the results from these unscaled problems is difficult, so our aim should be to determine a balance where significant improvement on the results from the scaled problems is found, while not excessively increasing the cost of solution for problems such as 1,2, and 3. The results for the unscaled Problem 4 are typical of the situation where, as h decreases, the problem becomes more difficult to solve due to the increase in the number of internal boundaries.

5.2.5.2 Calibrated AMG results

For the adaptive AMG method, we consider several questions around the same problems. Since experience has shown that a single setup cycle is often most efficient, parameters ν_0 and ν_1 must be chosen such that this cycle yields an effective solver. How to do so actually depends on our interests. For solving only the matrix equation, $A\mathbf{x} = \mathbf{b}$, for a single vector, $\mathbf{b}$, choose ν_0 and ν_1 such that the total time to solution is smallest. This may mean sacrificing performance of the solver to save cost of the setup stage. For solving the matrix equation for many right sides, the parameters should be chosen such that the solver performs optimally. We consider this latter situation, and

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.104	0.115	0.124	0.131	0.137
Problem 1u	0.104	0.115	0.124	0.131	0.137
Problem 1r	0.991	0.997	0.996	0.996	0.996
Problem 2	0.068	0.069	0.070	0.071	0.071
Problem 2u	0.594	0.754	0.864	0.929	0.964
Problem 2r	0.991	0.993	0.993	0.993	0.992
Problem 3	0.111	0.122	0.130	0.136	0.141
Problem 3u	0.488	0.656	0.793	0.886	0.940
Problem 3r	0.995	0.997	0.996	0.996	0.996
Problem 4	0.209	0.212	0.233	0.290	0.375
Problem 4u	0.760	0.914	0.976	0.994	0.998
Problem 4r	0.996	0.996	0.996	0.996	0.995

Table 5.4: Asymptotic Convergence Factors (Measured After at Most 200 Iterations) for Standard AMG

demonstrate that doing so does not severely impact the time-to-solution for a single right side. This approach may be called a calibrated AMG approach, reflecting that parameters ν_0 and ν_1 are chosen to calibrate the AMG performance, rather than a truly adaptive approach, where ν_0 and ν_1 remain fixed and setup is performed until an efficient solver is exposed.

Our experiments were thus performed with the goal of (approximately) minimizing the asymptotic convergence factors of the resulting methods. To do this, we computed the asymptotic convergence factors for very large ν_0 and ν_1, corresponding to the optimal case where $\mathbf{x}^{(1)}$ is the eigenvector associated with the smallest eigenvalue of the matrix, then chose the smallest values for ν_0 and ν_1 such that the convergence factor of the resulting method was within 0.005 of the optimal factor. We found that, for Problems 1, 2, and 3, good asymptotic convergence factors could be achieved with relatively small values of ν_0 and ν_1. For Problem 4, the same performance as standard AMG on the unscaled system can be recovered with small values of ν_0 and ν_1. Table 5.5 shows the time required for the setup phase for given values of these parameters and different grid sizes. As always happens when measuring computational performance,

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.02 (2,2)	0.09 (2,2)	0.38 (3,3)	1.67 (4,5)	7.21 (7,7)
Problem 1u	0.02 (2,2)	0.09 (2,2)	0.37 (3,3)	1.63 (4,5)	7.27 (7,7)
Problem 1r	0.02 (4,4)	0.11 (5,5)	0.47 (8,7)	2.13 (11,11)	9.52 (16,17)
Problem 2	0.02 (3,2)	0.10 (4,4)	0.42 (6,6)	1.88 (9,9)	8.57 (13,13)
Problem 2u	0.04 (3,2)	0.09 (4,4)	0.43 (7,6)	1.83 (10,9)	8.03 (14,13)
Problem 2r	0.03 (7,7)	0.11 (9,9)	0.51 (14,14)	2.50 (21,21)	11.84 (31,31)
Problem 3	0.02 (2,2)	0.11 (4,4)	0.45 (4,4)	1.69 (6,6)	6.86 (7,7)
Problem 3u	0.02 (2,2)	0.10 (4,4)	0.41 (4,4)	1.66 (6,6)	6.94 (7,7)
Problem 3r	0.03 (5,5)	0.10 (6,6)	0.45 (9,8)	1.90 (10,11)	8.50 (17,16)
Problem 4	0.02 (2,2)	0.06 (3,2)	0.35 (4,4)	1.60 (6,6)	6.80 (8,8)
Problem 4u	0.02 (2,2)	0.09 (3,2)	0.39 (5,5)	1.60 (6,6)	6.98 (9,9)
Problem 4r	0.02 (6,5)	0.12 (9,9)	0.50 (13,14)	2.37 (22,21)	17.09 (22,20*)

Table 5.5: Wall-Clock Time in Seconds (and Values of ν_0, ν_1) for Calibrated AMG Setup Phase. * Indicates 2 Setup Cycles Were More Efficient

the timings are accurate only to within a few hundredths of a second, and so there is some variation in the timings of program stages that require the same number and ordering of operations.

Table 5.6 presents the asymptotic convergence factors for the methods resulting from the setup stages as outlined in Table 5.5. Note that, for the first three problems and for all grid sizes, αAMG achieves convergence factors bounded well below 1, with very small growth as the mesh size decreases. For Problem 4, we see growth like that in the standard AMG results. Note also that scaling the matrices has no affect on our ability to determine an efficient solver for these problems.

Finally, in Table 5.7, we consider the total cost of solving $A\mathbf{x} = \mathbf{0}$ a single time, with random initial guess, using the near-optimal solver. Calibrated AMG did not (and could not be expected to) beat the overall performance of standard AMG on the four unscaled problems. However, the setup costs of calibrated AMG were not significantly higher than those of AMG. On a 1024×1024 mesh, AMG setup required approximately 5 seconds of CPU, and so the adaptive setup needed between 30% and 120% more time, but tended to produce a slightly better solver than classical AMG. For these reasons,

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.067	0.073	0.079	0.080	0.079
Problem 1u	0.067	0.073	0.079	0.080	0.079
Problem 1r	0.069	0.078	0.077	0.078	0.079
Problem 2	0.069	0.069	0.071	0.071	0.073
Problem 2u	0.069	0.071	0.071	0.071	0.072
Problem 2r	0.072	0.071	0.071	0.072	0.073
Problem 3	0.070	0.097	0.081	0.110	0.103
Problem 3u	0.072	0.097	0.080	0.109	0.106
Problem 3r	0.070	0.100	0.084	0.111	0.108
Problem 4	0.194	0.202	0.243	0.288	0.376
Problem 4u	0.189	0.212	0.231	0.294	0.374
Problem 4r	0.187	0.212	0.235	0.292	0.383

Table 5.6: Asymptotic Convergence Factors for Calibrated AMG.

the overall cost of calibrated AMG is close to that of standard AMG in the cases where standard AMG works well. When standard AMG fails, there is no contest. Calibrated AMG was able to solve those problems that caused difficulty for standard AMG in a small fraction of the time. Considering that 200 iterations of standard AMG for a 1024×1024 problem took approximately 180 seconds, calibrated AMG was able to reduce the residual by over 4 orders of magnitude more in under a tenth of the time on the first three randomly scaled 1024×1024 grid problems, and in a sixth of the time for the fourth.

5.2.5.3 Adaptive AMG (αAMG) results

While the calibrated AMG approach yields solvers with optimal performance characteristics, its results are achieved at a significant cost of user time in tuning the setup parameters of ν_0 and ν_1. An adaptive approach may be used instead, moving the burden of calibration from the user to the solver itself.

In the αAMG approach, a small, fixed number of relaxation sweeps are initially performed on the finest grid to locally expose algebraically smooth error. This error is used to create an interpolation operator and (through the Galerkin conditions) a

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.04 (8)	0.25 (8)	0.89 (8)	3.52 (8)	14.70 (8)
Problem 1u	0.06 (8)	0.21 (8)	0.90 (8)	3.55 (8)	14.73 (8)
Problem 1r	0.03 (7)	0.22 (7)	0.91 (7)	3.64 (7)	15.64 (7)
Problem 2	0.05 (8)	0.22 (8)	0.92 (8)	3.83 (8)	15.85 (8)
Problem 2u	0.04 (8)	0.24 (8)	0.95 (8)	3.83 (8)	15.94 (8)
Problem 2r	0.05 (7)	0.24 (7)	0.97 (7)	4.23 (7)	18.60 (7)
Problem 3	0.05 (8)	0.20 (8)	0.93 (8)	3.63 (8)	14.58 (8)
Problem 3u	0.03 (8)	0.21 (8)	0.92 (8)	3.66 (8)	14.43 (8)
Problem 3r	0.04 (7)	0.24 (8)	0.94 (8)	3.81 (8)	16.32 (8)
Problem 4	0.03 (11)	0.31 (11)	1.09 (11)	4.84 (13)	22.06 (16)
Problem 4u	0.04 (11)	0.32 (11)	1.10 (11)	4.79 (13)	20.40 (14)
Problem 4r	0.05 (10)	0.27 (10)	1.22 (11)	5.35 (12)	28.27 (12)

Table 5.7: Total Solution Wall-Clock Time in Seconds (and Iteration Count) for Calibrated AMG to Reduce Residuals by 10^{10}.

coarse-grid problem. Relaxation is then performed on this grid and the problem is further coarsened. After the initial multigrid hierarchy is created in this fashion, another setup cycle is performed with a new random initial guess. Now, relaxation is replaced by the current multigrid solver. This is done in order to expose errors that are both algebraically smooth (as relaxation does not quickly reduce them) and not reduced by the current coarse-grid correction scheme.

Ideally, such errors would then be used to improve the adaptive AMG interpolation, resulting in a more robust solver. We have, however, been considering problems similar to those for which classical AMG is effective. In particular, those where the near null space can be represented by a single vector. The question of how to improve an existing AMG interpolation operator with information from a second prototype vector is not easily addressed. Instead, here we redefine the interpolation operator based only on the new vector. This is quite dangerous, as the new prototype is representative of error distinct from the previous one, and so we may expect that in some cases the adapted method will be worse than its parent.

The results in Tables 5.8, 5.9, and 5.10 reflect performing 8 iterations of the

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.04 (1)	0.21 (1)	0.79 (1)	3.18 (1)	12.75 (1)
Problem 1u	0.03 (1)	0.19 (1)	0.83 (1)	3.13 (1)	12.75 (1)
Problem 1r	0.04 (1)	0.20 (1)	0.81 (1)	6.46 (2)	25.61 (2)
Problem 2	0.04 (1)	0.21 (1)	0.85 (1)	3.31 (1)	13.00 (1)
Problem 2u	0.04 (1)	0.21 (1)	0.81 (1)	3.25 (1)	12.65 (1)
Problem 2r	0.05 (1)	0.22 (1)	0.83 (1)	23.07 (7)	91.55 (7)
Problem 3	0.04 (1)	0.20 (1)	0.72 (1)	3.15 (1)	12.71 (1)
Problem 3u	0.03 (1)	0.18 (1)	0.83 (1)	3.28 (1)	12.40 (1)
Problem 3r	0.04 (1)	0.19 (1)	0.82 (1)	6.44 (2)	25.35 (2)
Problem 4	0.05 (1)	0.18 (1)	0.80 (1)	3.18 (1)	25.01 (2)
Problem 4u	0.05 (1)	0.17 (1)	0.81 (1)	6.44 (2)	25.39 (2)
Problem 4r	0.04 (1)	0.37 (2)	1.62 (2)	6.37 (2)	50.63 (4)

Table 5.8: Wall-Clock Time in Seconds (and Number of Setup Cycles) for Adaptive AMG Setup Phase

current solver on the homogeneous problem at each level of the setup phase. The fine-grid multigrid cycle is accepted when the error reduction (measured in the A-norm) by the last of these iterations is by a factor greater than 2.5. An additional cost in the adaptive setup phase is that of this "test drive", performing 8 cycles of the current V-cycle on the homogeneous problem (to ensure adequate performance) before considering the system $A\mathbf{x} = \mathbf{b}$. Performing fewer iterations of the solver results in less exposure of algebraically smooth error and typically increases the number of setup cycles needed. Considering the high cost in recomputation of the interpolation and coarse-grid operators, we prefer to avoid this. Performing more iterations of the solver on the homogeneous problem better exposes the error sought, but at the cost of potentially significant unnecessary computation. Choosing the threshold for accepting the solver is less ambiguous. Setting a high threshold reduces the setup time but admits poor solvers, whereas setting the threshold to be small improves the solver but potentially requires more setup cycles.

Table 5.8 demonstrates the typical higher computational cost of the adaptive approach as compared to the calibrated approach. Some of this cost is certainly un-

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.065	0.068	0.070	0.086	0.201
Problem 1u	0.065	0.068	0.070	0.086	0.201
Problem 1r	0.069	0.085	0.210	0.072	0.071
Problem 2	0.067	0.069	0.089	0.155	0.334
Problem 2u	0.067	0.070	0.091	0.158	0.337
Problem 2r	0.099	0.160	0.355	0.118	0.349
Problem 3	0.067	0.097	0.080	0.118	0.294
Problem 3u	0.067	0.097	0.080	0.121	0.298
Problem 3r	0.075	0.131	0.293	0.110	0.090
Problem 4	0.186	0.195	0.243	0.395	0.390
Problem 4u	0.185	0.195	0.231	0.300	0.389
Problem 4r	0.227	0.238	0.241	0.283	0.397

Table 5.9: Asymptotic Convergence Factors for Adaptive AMG.

avoidable. Fixing the number of iterations performed on the homogeneous problem undoubtedly results in some extraneous relaxations being performed, particular on coarser grids, where few relaxation sweeps are "necessary". Likewise, the cost of testing the solver can only be avoided if we know, a priori, whether the solver will be acceptable. While these costs are significantly higher than those for calibrated AMG (as in Table 5.5), they do remain relatively low, still under two minutes of CPU time in all cases with a 1024×1024 grid.

Convergence factors for the adaptive AMG approach (Table 5.9) demonstrate its robustness. While the requirement for a successful setup was a single=step convergence factor less than 0.4 when measured after 8 iterations, we see that the asymptotic convergence factors are all bounded by 0.4 as well. Some of these factors are worse than the corresponding convergence factors for the calibrated AMG approach, although they could also be improved at the cost of more expensive setup cycles. A constant convergence factor of 0.4 is still sufficient, however, to reduce the error by a factor of 10^{10} in 25 iterations.

The overall time to solution of the adaptive approach, as in Table 5.10, reflects the expected behavior. Iteration counts are low - at most 15 iterations are required

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.05 (8)	0.33 (8)	1.31 (8)	5.11 (8)	22.81 (11)
Problem 1u	0.06 (8)	0.31 (8)	1.36 (8)	5.05 (8)	22.72 (11)
Problem 1r	0.06 (7)	0.34 (8)	1.44 (10)	8.17 (7)	32.14 (7)
Problem 2	0.07 (8)	0.34 (8)	1.36 (8)	5.52 (9)	25.44 (13)
Problem 2u	0.06 (8)	0.35 (8)	1.33 (8)	5.46 (9)	25.35 (14)
Problem 2r	0.08 (8)	0.37 (8)	1.68 (13)	25.05 (8)	101.92 (11)
Problem 3	0.06 (8)	0.34 (8)	1.30 (8)	4.95 (8)	25.42 (14)
Problem 3u	0.05 (8)	0.32 (8)	1.35 (8)	5.24 (8)	25.01 (14)
Problem 3r	0.07 (7)	0.33 (9)	1.56 (12)	8.40 (8)	32.04 (7)
Problem 4	0.08 (11)	0.36 (11)	1.51 (11)	7.40 (18)	38.80 (15)
Problem 4u	0.08 (11)	0.37 (11)	1.50 (11)	9.56 (13)	39.10 (15)
Problem 4r	0.07 (11)	0.55 (11)	2.32 (11)	8.78 (10)	62.54 (13)

Table 5.10: Total Solution Wall-Clock Time in Seconds (and Iteration Count) for Adaptive AMG to Reduce Residuals by 10^{10}.

for residual reduction by a factor of 10^{10} - and so the time to solution, discarding the setup phase, is quite low. The increased setup costs due to the poor adaptations used are reflected in the total time to solution. Comparing these results to the calibrated AMG approach is somewhat disappointing; however, in all cases, the solution was still obtained quickly, with total time for the αAMG approach within a factor of at most 6 (and often 2) of the total time for the calibrated AMG results. Comparing with the standard AMG results shows that the overall time to solution for the unscaled problems is higher than that for standard AMG, whereas, for the scaled problems, adaptive AMG always yielded a quickly converging solver, and so yielded lower overall solution times in all cases where standard AMG performance suffered.

The calibrated and adaptive AMG results are very encouraging. For the scaled problems, there is a tremendous improvement in the amount of effort required for solution of the linear systems when compared to the results for standard AMG interpolation in Table 5.4. The solution phase of the algorithm scales well across all 5 grid sizes, and the actual costs are quite reasonable. It must be acknowledged, however, that once we account for the cost of the setup phase of our algorithm, classical AMG is still a slightly

more efficient solver for the unscaled matrices when we consider solving with only a single right-hand side. Put simply, if the algebraically smoothest component of an elliptic PDE is known exactly, αAMG can do no better than designing multigrid interpolation based on that component. Indeed, if this component is given as input to the adaptive AMG method for creating the multigrid hierarchy, we can solve the problem with the same cost as classical AMG on the unscaled problem, simply by using this prototype as $\mathbf{x}^{(1)}$ in Equation 5.4, as discussed in Section 5.2.2.

5.3 Theoretical Results

The results above give no quantitative indication of the growth in cost of the adaptive process with difficulty of the problem nor with grid size. To get an indication of this cost, we consider the case of using a simple V(1,0) setup cycle to iteratively improve a single prototype. As discussed in Section 5.1.2, a prototype may be improved by removing it from the prototype set, constructing the reduced method, and then using the prototype as an initial guess for iteration on the homogeneous problem with the reduced method. Since we limit ourselves to a single prototype, the reduced method will always be no method, and so the structure of the cycles used is always the same as the first setup cycle, contrary to the cycling scheme analyzed in Section 5.2.4. We monitor the performance of the solvers resulting from increasing numbers of such cycles.

The difficulty of the problem under consideration is controlled by utilizing the shifted Laplacian problem, $-\Delta u - 2\pi^2(1 - 2^{-\beta})u = 0$, with Dirichlet boundary conditions, discretized with finite elements on the unit square, with uniform mesh width, h, in both x and y. To maintain a relationship with the continuous problem, we consider the generalized eigenvalues of the discrete operator, A, values λ such that $A\mathbf{u} = \lambda M\mathbf{u}$ for some $\mathbf{u}$, where M is the finite element mass matrix. The minimal eigenvalue of this

problem is then

$$\frac{12}{h^2}\left(\frac{2-\cos(\pi h)-\cos^2(\pi h)}{4+4\cos(\pi h)+\cos^2(\pi h)}\right)-2\pi^2(1-2^{-\beta})=2^{1-\beta}\pi^2+\mathrm{O}(h^2).$$

By increasing β, the problem is made more difficult, as the eigenvector approximation principle requires a more accurate prototype to ensure good performance. Thus, we monitor the difference between the exact smallest eigenvalue and the Rayleigh quotient ($\mathrm{RQ}(\mathbf{x})=\frac{\langle A\mathbf{x},\mathbf{x}\rangle}{\langle M\mathbf{x},\mathbf{x}\rangle}$, where M is the finite element mass matrix) of the current prototype as the setup iterations proceed, with differing β and grid sizes.

Results for a 128×128 grid, with a random initial guess (with Rayleigh quotient of 1.35×10^4), are shown in Figure 5.2. The upper figure in 5.2 shows the size of the Rayleigh quotient of the prototype vector after each cycle, measured relative to the minimal eigenvalue of the matrix. The lower figure in 5.2 shows the convergence factors of the resulting multigrid methods after each setup cycle, measured relative to the convergence factor of the cycle based upon the eigenvector of minimum eigenvalue. Results for a 256×256 grid, with a random initial guess (with Rayleigh quotient of 5.25×10^4), are shown in Figure 5.3. Results for a 512×512 grid, with a random initial guess (with Rayleigh quotient of 2.06×10^5), are shown in Figure 5.4.

These results indicate that there is, in general, a constant reduction in the Rayleigh quotient per V(1,0) setup cycle. This is particularly noticeable for large β, where the gap between the Rayleigh quotient of the initial guess and the optimal Rayleigh quotient, $\lambda_{\min}$, is largest, and we see several cycles of setup with such a reduction before the Rayleigh quotient of the prototype nears $\lambda_{\min}$. They also indicate that solver performance approaches its best when the error between the Rayleigh quotient of the prototype and $\lambda_{\min}$ is $\mathrm{O}(\lambda_{\min})$. For all three grids, the optimal convergence factors range from approximately 0.07 for $\beta=0$ to 0.105 for $\beta\geq9$. Thus, a convergence factor within a factor of three of the optimal reflects a convergence factor less than approximately 0.3. On all three lower plots, we mark the line of a relative convergence factor of

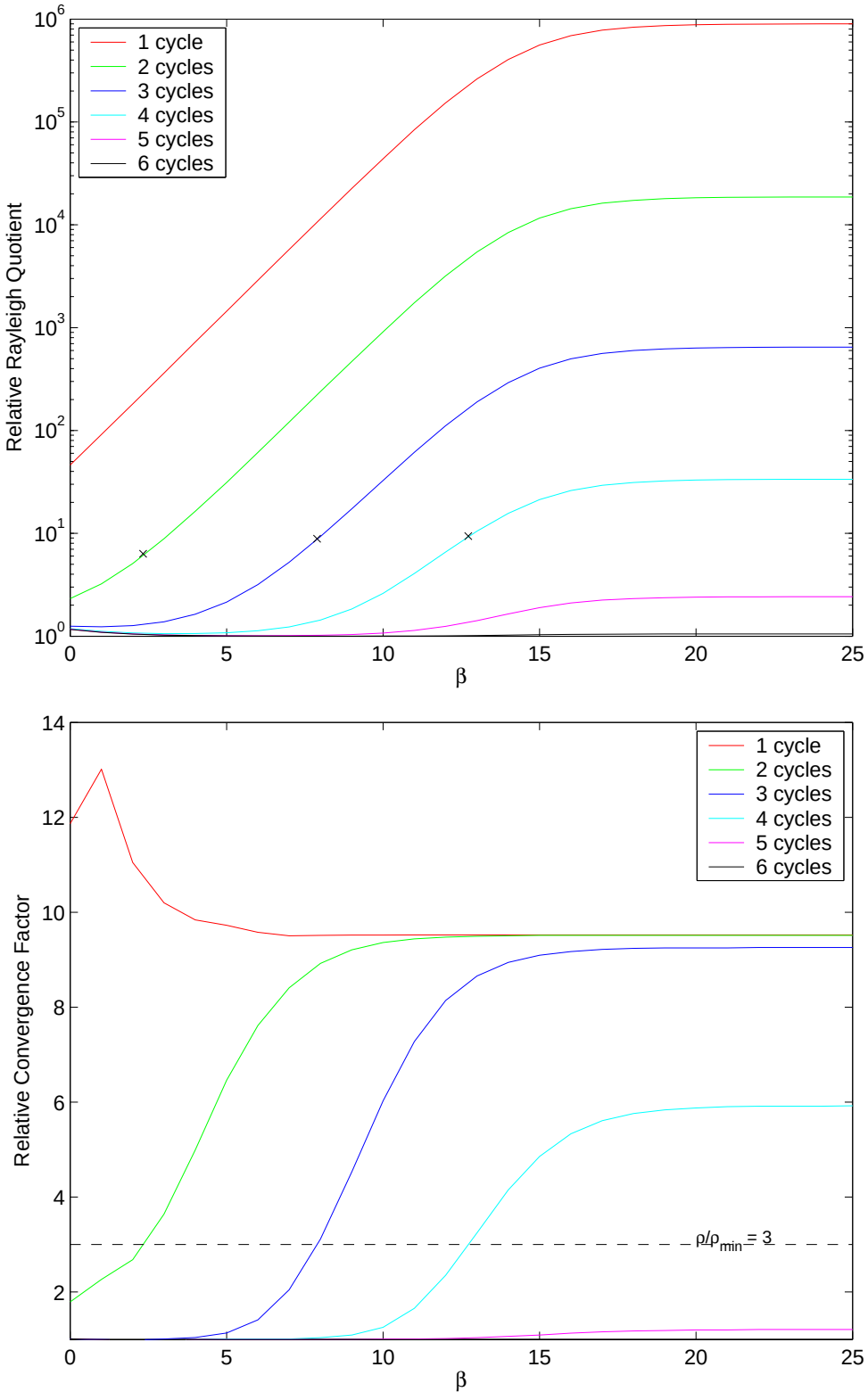

Figure 5.2: Relative Rayleigh Quotients and Relative Convergence Factors for Resulting Multigrid Method for Different β and Numbers of V(1,0) Setup Cycles, for $h = \frac{1}{128}$

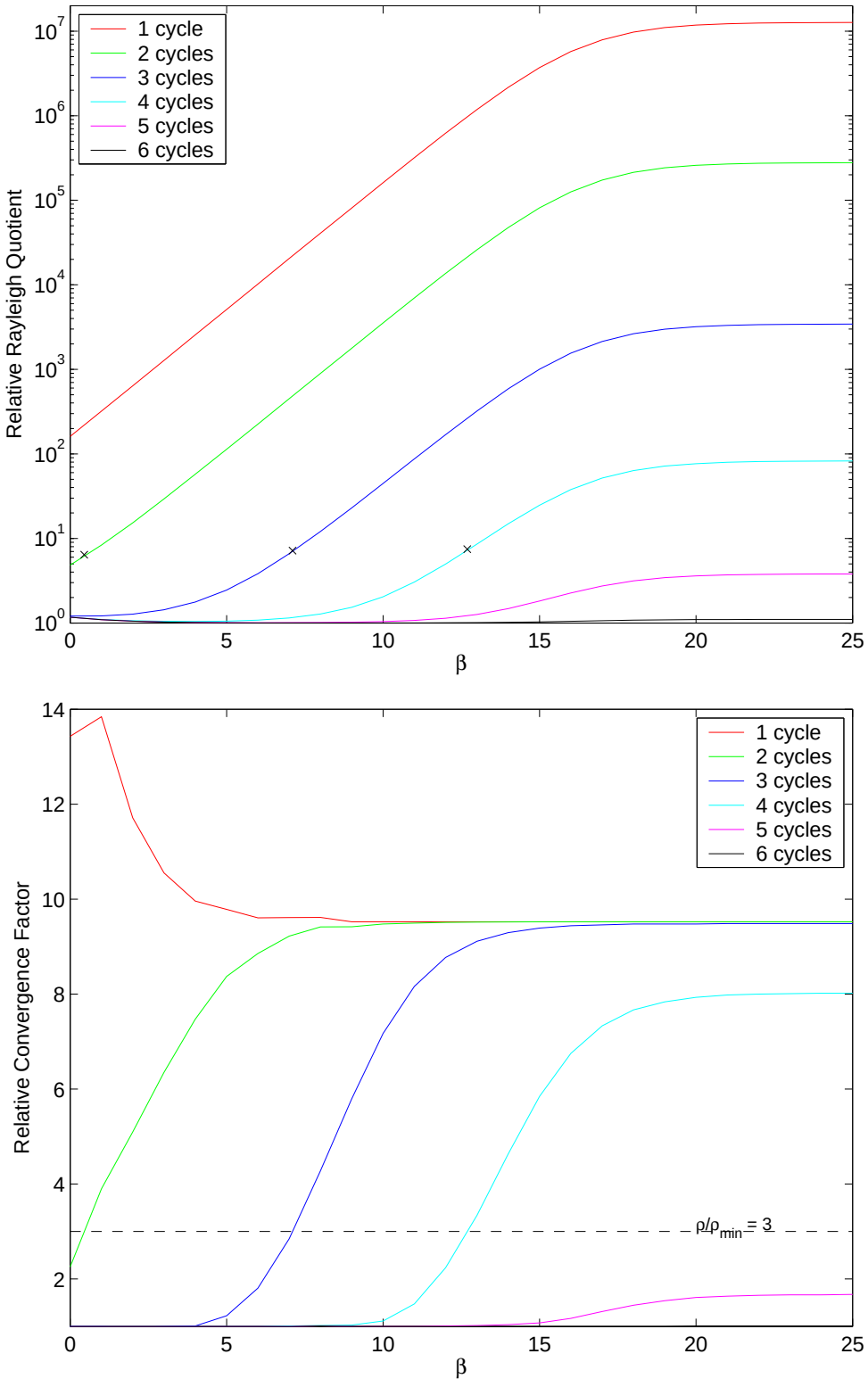

Figure 5.3: Relative Rayleigh Quotients and Relative Convergence Factors for Resulting Multigrid Method for Different β and Numbers of V(1,0) Setup Cycles, for $h = \frac{1}{256}$

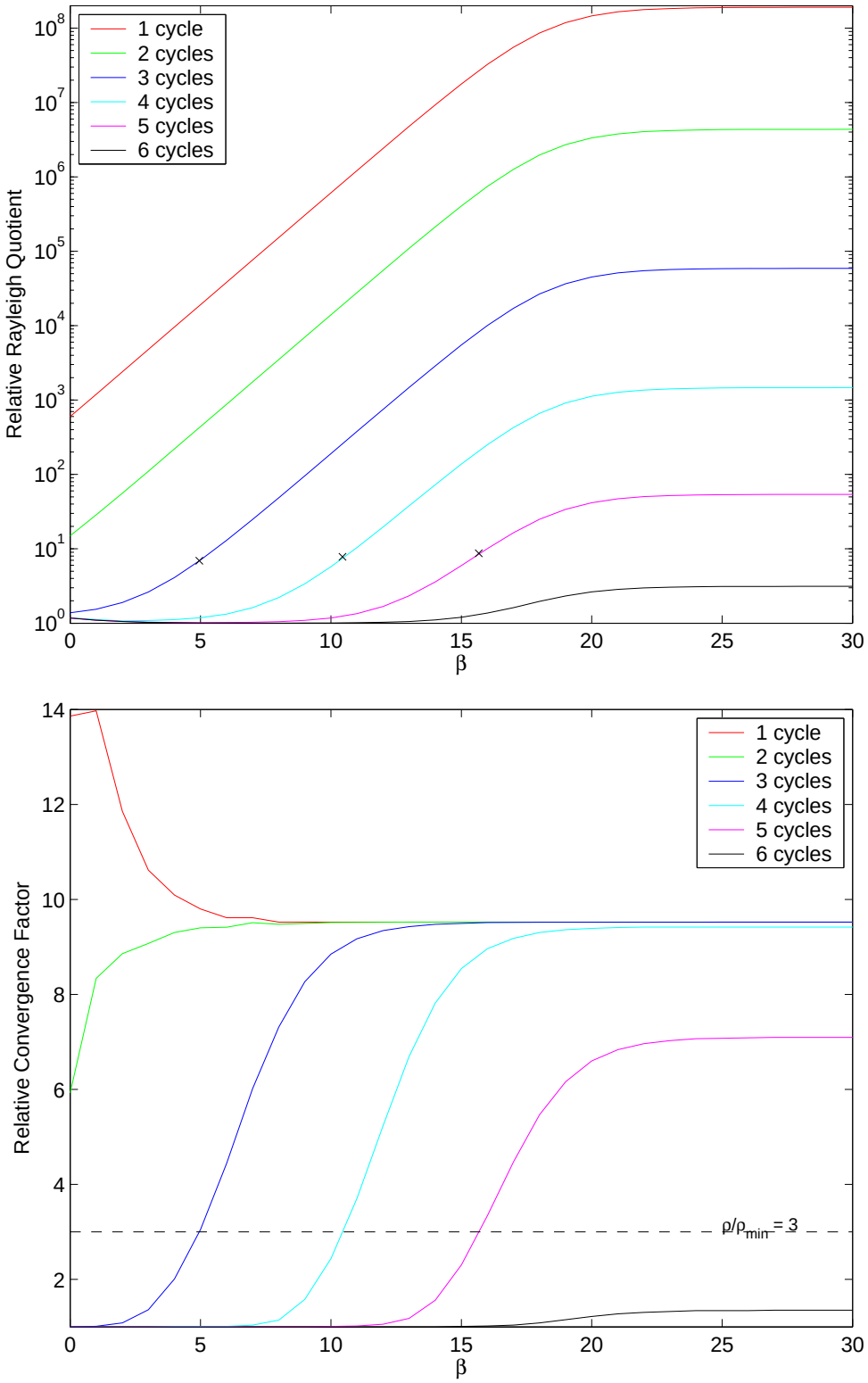

Figure 5.4: Relative Rayleigh Quotients and Relative Convergence Factors for Resulting Multigrid Method for Different β and Numbers of V(1,0) Setup Cycles, for $h = \frac{1}{512}$

3 with the dashed line. The points where relative convergence factors cross these lines for the different numbers of setup cycles are marked as crosses on the upper plots. Note that for all three grids, these points occur at roughly the same value of the Rayleigh quotient of the prototype, relative to the grid- and β-dependent $\lambda_{\min}$. The same is true for any for any fixed convergence factor not approaching 1, measured relative to the minimum. Since the Rayleigh quotient of a random initial guess for this problem has expected value $O(h^{-2})$, $O\left(\log\left(\frac{h^{-2}}{\lambda_{\min}}\right)\right)$ setup cycles are expected to be necessary to reduce the Rayleigh quotient of the prototype to be $O(\lambda_{\min})$ and, thus, obtain good solver performance.

To improve understanding of adaptive AMG, we seek to develop a theory in a simplified, two-level setting for a reduction-based method called AMGr. The basis for AMGr is a splitting of the finest-level variables into a set of fine-level-only variables, F, and coarse-level variables, C, in such a way that the matrix defining F-to-F connections is diagonally dominant. Such a splitting can be achieved using compatible relaxation [20] in the coarsening process. The theory in Section 5.3.1 shows that, given an appropriate splitting, we can obtain convergence factors bounded uniformly below 1 for any amount of diagonal dominance.

We also develop a simple, two-level version of adaptive AMGr (αAMGr). In this case, we aim to show that αAMGr recovers, to arbitrary tolerance, the assumed one-dimensional null space of the operator. Using this component to define an interpolation scheme then recovers convergence factors bounded below 1 for two-level AMGr. This indicates that, at the cost of a few αAMGr cycles, we can recover the performance of the AMGr method without explicit knowledge of the near null space components.

5.3.1 Reduction-Based AMG (AMGr)

AMGr is developed from a reduction point of view. We suppose that the original degrees of freedom (points, or variables in $\mathbb{R}^n$) are partitioned into those that are

associated with the coarse level (set C) and those that are not (set F): $\mathbb{R}^n = F \cup C$. We further assume that this is done in such a way that the F-to-F connections are subdominant (i.e., the submatrix associated with the fine-level-only points is diagonally dominant). If this is the case, then good performance can be achieved using relaxation only on the F points in combination with coarse-level correction (as shown below).

In what follows, the matrix, A, is assumed to be real, symmetric, and positive definite. Additional assumptions are made on A that specify the F-to-C dominance needed. We show that this algorithm obtains uniform convergence for any given scale of diagonal dominance.

Consider the representation for the $n \times n$ matrix A as

$$A = \begin{bmatrix} A_{ff} & -A_{fc} \\ -A_{fc}^T & A_{cc} \end{bmatrix},$$

where we choose to denote the F-C block as $-A_{fc}$, both to simplify notation in what follows and to suggest the overall balance in the block structure typical of the matrices we consider. To clarify the diagonal dominance we need, write $A_{ff} = D + \mathcal{E}$, where D is diagonal and positive definite and $0 \leq \mathcal{E} \leq \epsilon D$ for some $\epsilon > 0$. The matrix notation $A \leq B$ is taken to mean that $\mathbf{x}^T A\mathbf{x} \leq \mathbf{x}^T B\mathbf{x}$ for all $\mathbf{x}$. Assume further that A and D satisfy

$$A_D = \begin{bmatrix} D & -A_{fc} \\ -A_{fc}^T & A_{cc} \end{bmatrix} \geq 0. \tag{5.6}$$

Our primary interest is in applying such methods to finite difference and finite element discretizations of elliptic PDEs, using standard multigrid coarsening techniques. Five-point finite difference matrices on tensor-product meshes coarsened in a red-black manner satisfy the above assumptions trivially, as A_{ff} is diagonal. When coarsening of these operators is done in a full-coarsening manner, the assumptions still hold, however now parameter ϵ may be quite large as A_{ff} is only irreducibly diagonally dominant, not strictly so, and easy bounds on ϵ are not available with Gerschgorin's Theorem.

For nine-point finite element operators coarsened using full coarsening, the bound is typically easy to achieve with moderate ϵ.

The two-level AMGr scheme is defined by its basic components, relaxation and coarse-level correction. For relaxation, we choose a weighted Jacobi-like process applied to the F points only. Its error propagation matrix is given by

$$REL = \left(I - \omega \begin{bmatrix} D^{-1} & 0 \\ 0 & 0 \end{bmatrix} A \right), \tag{5.7}$$

with $\omega = \dfrac{2}{2+\epsilon}$. Complementing this is a variational coarse-level correction scheme using interpolation operator

$$P = \begin{bmatrix} D^{-1}A_{fc} \\ I \end{bmatrix}.$$

Note that the error propagation matrix for this coarse-level correction is given by

$$CLC = I - P(P^T AP)^{-1} P^T A. \tag{5.8}$$

The error propagation matrix for the two-grid scheme analyzed here, with $\nu \geq 1$ relaxation steps followed by a coarse-level correction, is then given by

$$MG_2 = CLC \cdot REL^{\nu}. \tag{5.9}$$

The analysis is simplified by noticing that any $\mathbf{e} \in \mathbb{R}^n$ can be written as the A-orthogonal sum

$$\mathbf{e} = \alpha \begin{bmatrix} A_{ff}^{-1}A_{fc} \\ I \end{bmatrix} \mathbf{v} + \beta \begin{bmatrix} I \\ 0 \end{bmatrix} \mathbf{w}, \tag{5.10}$$

where $\|\mathbf{v}\|_{\hat{A}_{ff}} = \|\mathbf{w}\|_{A_{ff}} = 1$ and $\hat{A}_{ff} = A_{cc} - A_{fc}^T A_{ff}^{-1} A_{fc}$ is the Schur complement of A_{ff} in A. Here, we again use the energy-norm notation, $\|\mathbf{x}\|_A = \langle A\mathbf{x}, \mathbf{x} \rangle^{\frac{1}{2}}$, for any symmetric and positive-definite matrix, A.

Theorem 5. *Given* $A = \begin{bmatrix} A_{ff} & -A_{fc} \\ -A_{fc}^T & A_{cc} \end{bmatrix} \geq 0$, *such that* $A_{ff} = D + \mathcal{E}$, $0 \leq \mathcal{E} \leq \epsilon D$, *and* $\begin{bmatrix} D & -A_{fc} \\ -A_{fc}^T & A_{cc} \end{bmatrix} \geq 0$, *the multigrid process given in Equations 5.7, 5.8, and 5.9,* MG_2,

satisfies, for any $\epsilon > 0$ *and* $\nu \geq 1$,

$$\|MG_2\|_A \leq \left(\frac{\epsilon}{1+\epsilon} + \left(\frac{\epsilon}{2+\epsilon} \right)^{2\nu} \right)^{\frac{1}{2}}.$$

The two-grid AMGr scheme thus converges uniformly for any $\epsilon > 0$ *with sufficiently many smoothing steps,* $\nu \geq 1$.

Proof. Considering the effect of relaxation, the A–orthogonal decomposition of (5.10) gives

$$\begin{aligned} REL\mathbf{e} &= \mathbf{e} - \omega \begin{bmatrix} D^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} \beta A_{ff}\mathbf{w} \\ \alpha \hat{A}_{ff}\mathbf{v} - \beta A_{fc}^T \mathbf{w} \end{pmatrix} \\ &= \mathbf{e} - \beta\omega \begin{pmatrix} D^{-1}A_{ff}\mathbf{w} \\ 0 \end{pmatrix} \\ &= \alpha \begin{bmatrix} A_{ff}^{-1}A_{fc} \\ I \end{bmatrix} \mathbf{v} + \beta \begin{bmatrix} I - \omega D^{-1}A_{ff} \\ 0 \end{bmatrix} \mathbf{w}. \end{aligned}$$

However,

$$\begin{aligned} \left\| \begin{bmatrix} I - \omega D^{-1}A_{ff} \\ 0 \end{bmatrix} \mathbf{w} \right\|_A &= \| \left(I - \omega D^{-1}A_{ff} \right) \mathbf{w} \|_{A_{ff}} \\ &\leq \rho \left(I - \omega A_{ff}^{1/2} D^{-1} A_{ff}^{1/2} \right) \\ &\leq \max \left(\left| 1 - \frac{2}{2+\epsilon} \right|, \left| \frac{2+2\epsilon}{2+\epsilon} - 1 \right| \right) = \frac{\epsilon}{2+\epsilon}. \end{aligned}$$

It is therefore easy to see that

$$REL^{\nu}\mathbf{e} = \alpha \begin{bmatrix} A_{ff}^{-1}A_{fc} \\ I \end{bmatrix} \mathbf{v} + \hat{\beta} \begin{bmatrix} I \\ 0 \end{bmatrix} \hat{\mathbf{w}},$$

where $\|\hat{\mathbf{w}}\|_{A_{ff}} = 1$, and

$$|\hat{\beta}| \leq |\beta| \left(\frac{\epsilon}{2+\epsilon} \right)^{\nu}. \tag{5.11}$$

To analyze the coarse-level correction step, we again make use of Equation 5.10 to see that

$$\begin{aligned}
\|CLC\mathbf{e}\|_A &= \min_{\mathbf{u}} \left\| \mathbf{e} - \begin{bmatrix} D^{-1}A_{fc} \\ I \end{bmatrix} \mathbf{u} \right\|_A \\
&\leq \min_{\theta} \left\| \alpha \begin{bmatrix} A_{ff}^{-1}A_{fc} \\ I \end{bmatrix} \mathbf{v} + \beta \begin{bmatrix} I \\ 0 \end{bmatrix} \mathbf{w} - \alpha\theta \begin{bmatrix} D^{-1}A_{fc} \\ I \end{bmatrix} \mathbf{v} \right\|_A \\
&\leq |\alpha| \min_{\theta} \left\| \left(\begin{bmatrix} A_{ff}^{-1}A_{fc} \\ I \end{bmatrix} - \theta \begin{bmatrix} D^{-1}A_{fc} \\ I \end{bmatrix} \right) \mathbf{v} \right\|_A + |\beta|. \qquad (5.12)
\end{aligned}$$

The key is then to write the correction as an A–orthogonal sum:

$$\begin{bmatrix} D^{-1}A_{fc} \\ I \end{bmatrix} \mathbf{v} = \begin{bmatrix} A_{ff}^{-1}A_{fc} \\ I \end{bmatrix} \mathbf{v} + \begin{bmatrix} \left(D^{-1} - A_{ff}^{-1}\right) A_{ff} \\ 0 \end{bmatrix} \mathbf{v},$$

giving

$$\begin{aligned}
&\left\| \left(\begin{bmatrix} A_{ff}^{-1}A_{fc} \\ I \end{bmatrix} - \theta \begin{bmatrix} D^{-1}A_{fc} \\ I \end{bmatrix} \right) \mathbf{v} \right\|_A^2 \\
&\qquad = (1-\theta)^2 \left\| \begin{bmatrix} A_{ff}^{-1}A_{fc} \\ I \end{bmatrix} \mathbf{v} \right\|_A^2 + \theta^2 \left\| \left(D^{-1} - A_{ff}^{-1}\right) A_{fc}\mathbf{v} \right\|_{A_{ff}}^2 \\
&\qquad \leq (1-\theta)^2 + \theta^2\epsilon \equiv f(\theta). \qquad (5.13)
\end{aligned}$$

This inequality follows because $A_D \geq 0$ implies that $A_{cc} \geq A_{fc}^T D^{-1} A_{fc}$, which in turn implies that

$$\begin{aligned}
\left\| \left(D^{-1} - A_{ff}^{-1}\right) A_{fc}\mathbf{v} \right\|_{A_{ff}}^2 &= \mathbf{v}^T A_{fc}^T A_{ff}^{-1/2} \left(A_{ff}^{1/2} D^{-1} A_{ff}^{1/2} - I \right)^2 A_{ff}^{-1/2} A_{fc}\mathbf{v} \\
&\leq \epsilon \mathbf{v}^T A_{fc}^T A_{ff}^{-1/2} \left(A_{ff}^{1/2} D^{-1} A_{ff}^{1/2} - I \right) A_{ff}^{-1/2} A_{fc}\mathbf{v} \\
&= \epsilon \mathbf{v}^T A_{fc}^T \left(D^{-1} - A_{ff}^{-1} \right) A_{fc}\mathbf{v} \\
&\leq \epsilon \mathbf{v}^T \left(A_{cc} - A_{fc}^T A_{ff}^{-1} A_{fc} \right) \mathbf{v} \\
&= \epsilon \|\mathbf{v}\|_{\hat{A}}^2 = \epsilon.
\end{aligned}$$

Now $f'(\theta) = 0$ implies that $\theta = \frac{1}{1+\epsilon}$ and that the minimum of $f(\theta)$ is $\frac{\epsilon}{1+\epsilon}$. We thus have

$$\|CLC\mathbf{e}\|_A \leq |\alpha|\sqrt{\frac{\epsilon}{1+\epsilon}} + |\beta|.$$

Combining this bound with Equations 5.10 and 5.11 yields

$$\begin{aligned}\|MG_2\mathbf{e}\|_A &\leq |\alpha|\sqrt{\frac{\epsilon}{1+\epsilon}} + |\hat{\beta}| \\ &\leq |\alpha|\sqrt{\frac{\epsilon}{1+\epsilon}} + |\beta|\left(\frac{\epsilon}{2+\epsilon}\right)^{\nu} \\ &\leq \left(\frac{\epsilon}{1+\epsilon} + \left(\frac{\epsilon}{2+\epsilon}\right)^{2\nu}\right)^{\frac{1}{2}} \|\mathbf{e}\|_A.\end{aligned}$$

□

5.3.2 Adaptive AMGr (αAMGr)

Construction of AMGr depends on knowledge of the splitting of A_{ff} into $D + \mathcal{E}$ with the assumed properties. Our aim is now to extend the applicability of AMGr to the case where such a splitting is not known beforehand. Here, the basic idea is to develop an adaptive AMGr scheme that automatically computes near null space components by "solving" $A\mathbf{x} = \mathbf{0}$ and using the result to determine D. To understand this process more clearly, we now assume that A is singular but positive semidefinite. It is easy to see that Theorem 5 still holds with the understanding that convergence is in $\mathcal{N}(A)^{\perp}$ (the orthogonal complement of the null space of A).

Although we no longer assume that D and $\mathcal{E}$ are known, we continue to suppose that the partition, $\mathbb{R}^N = F \cup C$, is given and that D and $\mathcal{E}$ exist under the assumptions of the previous section. We thus assume again that we can write the matrix as

$$A = \begin{bmatrix} A_{ff} & -A_{fc} \\ -A_{fc}^T & A_{cc} \end{bmatrix} \geq 0.$$

Below, n_f and n_c denote the dimensions of square matrices A_{ff} and A_{cc}, respectively, and subscripts f and c are used to refer to the corresponding components of $\mathbf{v} \in \mathbb{R}^N$, i.e., $\mathbf{v} = (\mathbf{v}_f, \mathbf{v}_c)^T$ and $N = n_f + n_c$. Assume also that A is singular with a one-dimensional null space spanned by a unit vector, $\mathfrak{r} \in \mathbb{R}^N$: $A\mathfrak{r} = 0$, $\|\mathfrak{r}\| = 1$. Since we continue to assume that $A_D \geq 0$ and $\mathcal{E} \geq 0$, we must conclude that $A_D\mathfrak{r} = 0$ and $\mathcal{E}\mathfrak{r}_f = 0$. To see this, note that the splitting $A_{ff} = D + \mathcal{E}$ yields

$$0 = \langle A\mathfrak{r}, \mathfrak{r}\rangle = \langle A_D\mathfrak{r}, \mathfrak{r}\rangle + \langle \mathcal{E}\mathfrak{r}_f, \mathfrak{r}_f\rangle.$$

By assumption, the two terms on the right are nonnegative, so they must both be zero. Since A_D and $\mathcal{E}$ are both positive semidefinite, we must have $A_D\mathfrak{r} = \mathbf{0}$ and $\mathcal{E}\mathfrak{r}_f = \mathbf{0}$.

Without loss of generality, and for convenience, we assume further that $D = I$. Our aim is to construct a two-level method that yields grid-independent convergence factors for solving $A\mathbf{x} = \mathbf{b}$ in general, and $A\mathbf{x} = \mathbf{0}$ in particular, without assuming knowledge of $\mathfrak{r}$ nor of the splitting of $A_{ff} = I + \mathcal{E}$.

Finally, suppose that the vector, $\mathbf{u}$, approximates the true null-space component, $\mathfrak{r}$. We may write $\mathbf{u} = \mathfrak{r} + \mathbf{e}$, rescaling $\mathbf{u}$ is necessary, where $\mathbf{e}$ is taken to be somehow orthogonal to $\mathfrak{r}$ (in a manner defined momentarily). Let $RQ(\mathbf{v}) = \dfrac{\langle A\mathbf{v}, \mathbf{v}\rangle}{\langle \mathbf{v}, \mathbf{v}\rangle}$, the standard Rayleigh quotient. The two-level adaptive AMGr algorithm then takes the following abstract form:

(1) Relax on $A\mathbf{u} = \mathbf{0}$.

(2) Define P such that $P\mathbf{u}_c = \mathbf{u}$.

(3) Set $\mathbf{u}^{\text{new}} = P\left(\underset{\mathbf{w}\in\mathbb{R}^{n_c}}{\operatorname{argmin}} RQ(P\mathbf{w})\right)$.

To simplify the theoretical development, we replace the iterative Jacobi-like F-point solver in AMGr with an exact F-point relaxation, $\mathbf{u}_f = A_{ff}^{-1}A_{fc}\mathbf{u}_c$. Interpolation, P, is then defined to match the values of $\mathbf{u}$ on the fine-level. This is done by choosing

the form of interpolation to be $\Lambda^{-1}A_{fc}$ for some diagonal matrix $\Lambda > 0$, and then determining Λ to match the fine-level values:

$$\Lambda^{-1}A_{fc}\mathbf{u}_c = \mathbf{u}_f = A_{ff}^{-1}A_{fc}\mathbf{u}_c. \tag{5.14}$$

Notice that the definition of Λ is independent of the norm of $\mathbf{u}$. Thus, we can define Λ based on a given $\mathbf{u}$, then scale $\mathbf{u}$ based on the definition of Λ, without affecting that of Λ.

The interpolation operator, P, is defined by

$$P = \begin{bmatrix} \Lambda^{-1}A_{fc} \\ I \end{bmatrix}.$$

For later use, it is convenient to also define an "ideal" interpolation operator:

$$Q = \begin{bmatrix} A_{ff}^{-1}A_{fc} \\ I \end{bmatrix}.$$

It is easy to see that using Q in place of P in AMGr yields an exact two-level solver, whence the term reduction-based AMG.

We first show that the two-grid αAMGr scheme produces a new iterate, $\mathbf{u}^{\text{new}}$, that gives a substantially smaller Rayleigh quotient. Since $\min_{\mathbf{u}\in\mathbb{R}^N} RQ(\mathbf{u}) = 0$, this gives us a measure of how close $\dfrac{\mathbf{u}^{\text{new}}}{\|\mathbf{u}^{\text{new}}\|}$ is to $\mathcal{N}(A)$ and, hence, $\mathfrak{r}$. Note then that successive αAMGr cycles would thus converge, in the sense of the Rayleigh quotient, to $\mathfrak{r}$.

Theorem 6. *Let* $A = \begin{bmatrix} A_{ff} & -A_{fc} \\ -A_{fc}^T & A_{cc} \end{bmatrix} \geq 0$ *be given such that* $A\mathfrak{r} = 0$ *and suppose that* $A_{ff} = I + \mathcal{E}$, $0 \leq \mathcal{E} \leq \epsilon I$, *and* $\begin{bmatrix} I & -A_{fc} \\ -A_{fc}^T & A_{cc} \end{bmatrix} \geq 0$. *Given a vector,* $\mathbf{u}$, *define* Λ *as in Equation 5.14,* $P = \begin{bmatrix} \Lambda^{-1}A_{fc} \\ I \end{bmatrix}$, *and assume that* $\eta = \min_i (A_{fc}\mathbf{u}_c)_i > 0$. *Then,*

$$RQ(\mathbf{u}^{new}) \equiv \min_{\mathbf{w}\in\mathbb{R}^{n_c}} RQ(P\mathbf{w}) \leq \left(1 - \frac{1}{2\left(\delta^2 + (1+\delta^2)\left(1 + \frac{\epsilon}{\eta^2}\right)\right)}\right) RQ(\mathbf{u}),$$

where $\mathbf{u} = \mathfrak{r} + \mathbf{e}$, $P\mathfrak{r}_c \perp P\mathbf{e}_c$, *and* $\delta = \frac{\|P\mathbf{e}_c\|^2}{\|P\mathfrak{r}_c\|^2}$.

Proof. It suffices to show that there is some $\mathbf{w} \in \mathbb{R}^{n_c}$ for which the stated bound holds. The choice we make is $\mathbf{w} = \mathbf{u}_c + s\mathbf{v}_c$ for a particular s determined below. Here, $\mathbf{v}_c = \delta\mathfrak{r}_c - \mathbf{e}_c$ and δ is chosen so that $P\mathbf{u}_c$ and $P\mathbf{v}_c$ are orthogonal:

$$\delta = \frac{\mathbf{e}_c^T P^T P(\mathfrak{r}_c + \mathbf{e}_c)}{\mathfrak{r}_c^T P^T P(\mathfrak{r}_c + \mathbf{e}_c)} = \frac{\mathbf{e}_c^T P^T P\mathbf{e}_c}{\mathfrak{r}_c^T P^T P\mathfrak{r}_c},$$

where the second equality comes from defining $\mathbf{e}$ so that $P\mathbf{e}_c$ is orthogonal to $P\mathfrak{r}_c$. This necessitates a rescaling of $\mathbf{u}$ such that $\mathbf{u} = \mathfrak{r} + \mathbf{e}$, which is permissible as such scaling does not affect the definition of P.

Note that

$$\begin{aligned}
RQ(P(\mathbf{u}_c + s\mathbf{v}_c)) &= \frac{\langle AP\mathbf{u}_c, P\mathbf{u}_c\rangle + 2s\langle AP\mathbf{u}_c, P\mathbf{v}_c\rangle + s^2\langle AP\mathbf{v}_c, P\mathbf{v}_c\rangle}{\|P\mathbf{u}_c\|^2 + s^2\|P\mathbf{v}_c\|^2} \\
&= \frac{\langle AQ\mathbf{u}_c, Q\mathbf{u}_c\rangle + 2s\langle AQ\mathbf{u}_c, P\mathbf{v}_c\rangle + s^2\langle AP\mathbf{v}_c, P\mathbf{v}_c\rangle}{\|Q\mathbf{u}_c\|^2 + s^2\|P\mathbf{v}_c\|^2} \\
&\leq \frac{\langle \hat{A}_{ff}\mathbf{u}_c, \mathbf{u}_c\rangle + 2s\langle \hat{A}_{ff}\mathbf{u}_c, \mathbf{v}_c\rangle + s^2\langle AP\mathbf{v}_c, P\mathbf{v}_c\rangle}{\|Q\mathbf{u}_c\|^2} \\
&= \frac{\langle \hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle - 2s\langle \hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle + s^2\langle AP\mathbf{v}_c, P\mathbf{v}_c\rangle}{\|Q\mathbf{u}_c\|^2} \\
&= \left(1 - 2s + s^2\frac{\langle AP\mathbf{v}_c, P\mathbf{v}_c\rangle}{\langle \hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle}\right)\frac{\langle \hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle}{\|Q\mathbf{u}_c\|^2} \\
&= \left(1 - 2s + s^2\frac{\langle AP\mathbf{v}_c, P\mathbf{v}_c\rangle}{\langle \hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle}\right)\frac{\langle AP\mathbf{u}_c, P\mathbf{u}_c\rangle}{\|P\mathbf{u}_c\|^2} \\
&= \left(1 - 2s + s^2\frac{\langle AP\mathbf{v}_c, P\mathbf{v}_c\rangle}{\langle \hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle}\right) RQ(\mathbf{u}).
\end{aligned}$$

Choosing $s = \dfrac{\langle \hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle}{\langle AP\mathbf{v}_c, P\mathbf{v}_c\rangle}$ then yields

$$RQ(P(\mathbf{u} + s\mathbf{v}_c)) \leq \left(1 - \frac{\langle \hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle}{\langle AP\mathbf{v}_c, P\mathbf{v}_c\rangle}\right) RQ(\mathbf{u}) = (1 - s)RQ(\mathbf{u}).$$

This implies that $0 \leq s \leq 1$, since both Rayleigh quotients are positive and s is a ratio of positive terms. Equivalently, this implies that $\langle \hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle \leq \langle AP\mathbf{v}_c, P\mathbf{v}_c\rangle$. It now suffices to show that $\langle AP\mathbf{v}_c, P\mathbf{v}_c\rangle \leq C_1\langle \hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle$ for some $C_1 \ll \infty$.

Writing $\mathbf{v} = \delta\mathbf{u} - (1+\delta)\mathbf{e}$, we have

$$\begin{aligned}\langle AP\mathbf{v}_c, P\mathbf{v}_c\rangle = \|P\mathbf{v}_c\|_A^2 &= \|\delta P\mathbf{u}_c - (1+\delta)P\mathbf{e}_c\|_A^2 \\ &\leq \left(\|\delta P\mathbf{u}_c\|_A + \|(1+\delta)P\mathbf{e}_c\|_A\right)^2 \\ &\leq 2\left(\|\delta P\mathbf{u}_c\|_A^2 + \|(1+\delta)P\mathbf{e}_c\|_A^2\right) \\ &= 2\left(\delta^2\langle\hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle + (1+\delta)^2\|P\mathbf{e}_c\|_A^2\right).\end{aligned}$$

Thus, if we can show that $\|P\mathbf{e}_c\|_A^2 \leq C_2\langle\hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle$, the theorem would follow. To this end, note that $A_{cc} \geq A_{fc}^T D^{-1} A_{fc} = A_{fc}^T A_{fc}$ and, hence,

$$\begin{aligned}\langle\hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle &\geq \langle(A_{fc}^T A_{fc} - A_{fc}^T A_{ff}^{-1} A_{fc})\mathbf{e}_c, \mathbf{e}_c\rangle \\ &= \langle(I - A_{ff}^{-1})A_{fc}\mathbf{e}_c, A_{fc}\mathbf{e}_c\rangle.\end{aligned}$$

Noting that

$$\begin{aligned}\langle AP\mathbf{e}_c, P\mathbf{e}_c\rangle &= \langle(\hat{A}_{ff} + A_{fc}^T\Lambda^{-1}A_{ff}\Lambda^{-1}A_{fc} - 2A_{fc}^T\Lambda^{-1}A_{fc} + A_{fc}^T A_{ff}^{-1}A_{fc})\mathbf{e}_c, \mathbf{e}_c\rangle \\ &= \langle\hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle + \langle(\Lambda^{-1}A_{ff}\Lambda^{-1} - 2\Lambda^{-1} + A_{ff}^{-1})A_{fc}\mathbf{e}_c, A_{fc}\mathbf{e}_c\rangle,\end{aligned}$$

it then suffices to show that

$$\langle(\Lambda^{-1}A_{ff}\Lambda^{-1} - 2\Lambda^{-1} + A_{ff}^{-1})A_{fc}\mathbf{e}_c, A_{fc}\mathbf{e}_c\rangle \leq C_3\langle(I - A_{ff}^{-1})A_{fc}\mathbf{e}_c, A_{fc}\mathbf{e}_c\rangle.$$

Defining $\tilde{\Lambda} = A_{ff}^{-\frac{1}{2}}\Lambda A_{ff}^{-\frac{1}{2}}$, we get

$$\begin{aligned}&\langle(\Lambda^{-1}A_{ff}\Lambda^{-1} - 2\Lambda^{-1} + A_{ff}^{-1})A_{fc}\mathbf{e}_c, A_{fc}\mathbf{e}_c\rangle \\ &\qquad = \langle A_{ff}^{-\frac{1}{2}}\tilde{\Lambda}^{-2}A_{ff}^{-\frac{1}{2}} - 2A_{ff}^{-\frac{1}{2}}\tilde{\Lambda}^{-1}A_{ff}^{-\frac{1}{2}} + A_{ff}^{-1})A_{fc}\mathbf{e}_c, A_{fc}\mathbf{e}_c\rangle \\ &\qquad = \langle(\tilde{\Lambda}^{-2} - 2\tilde{\Lambda}^{-1} + I)\mathbf{z}, \mathbf{z}\rangle \\ &\qquad = \langle(I - \tilde{\Lambda}^{-1})^2\mathbf{z}, \mathbf{z}\rangle = \|(I - \tilde{\Lambda}^{-1})\mathbf{z}\|^2,\end{aligned}$$

where $\mathbf{z} = A_{ff}^{-\frac{1}{2}}A_{fc}\mathbf{e}_c$.

To control this term, note that Λ was chosen so that $\Lambda^{-1}A_{fc}\mathbf{u}_c = A_{ff}^{-1}A_{fc}\mathbf{u}_c$. Hence,

$$\Lambda^{-1}A_{fc}\mathfrak{r}_c + \Lambda^{-1}A_{fc}\mathbf{e}_c = A_{ff}^{-1}A_{fc}\mathfrak{r}_c + A_{ff}^{-1}A_{fc}\mathbf{e}_c,$$

which yields

$$\Lambda^{-1}\mathfrak{r}_f - \mathfrak{r}_f = (A_{ff}^{-1} - \Lambda^{-1})A_{fc}\mathbf{e}_c,$$

or,

$$(\Lambda^{-1} - I)\mathfrak{r}_f = A_{ff}^{-\frac{1}{2}}(I - \tilde{\Lambda}^{-1})\mathbf{z}$$

because $A_{ff}\mathfrak{r}_f = (I + \mathcal{E})\mathfrak{r}_f = \mathfrak{r}_f$. Now, $(I - \tilde{\Lambda}^{-1})\mathbf{z} = A_{ff}^{\frac{1}{2}}(\Lambda^{-1} - I)\mathfrak{r}_f$, and we see that

$$\|(I - \tilde{\Lambda}^{-1})\mathbf{z}\|^2 = \|A_{ff}^{\frac{1}{2}}(\Lambda^{-1} - I)\mathfrak{r}_f\|^2 \leq (1 + \epsilon)\|(\Lambda^{-1} - I)\mathfrak{r}_f\|^2.$$

It thus suffices to bound $(\Lambda^{-1} - I)\mathfrak{r}_f$ in the L^2-norm. By prerelaxation (Step 1), we have $\mathbf{u}_f = A_{ff}^{-1}A_{fc}\mathbf{u}_c$, which yields

$$\begin{aligned}
(\Lambda^{-1} - I)\mathfrak{r}_f &= \operatorname{diag}\left(\frac{(\mathbf{u}_f)_i}{(A_{ff}\mathbf{u}_f)_i} - 1\right)\mathfrak{r}_f \\
&= \operatorname{diag}\left(\frac{(A_{ff}^{-1}A_{fc}\mathbf{u}_c)_i}{(A_{fc}\mathbf{u}_c)_i} - 1\right)\mathfrak{r}_f \\
&= \operatorname{diag}\left(\frac{((A_{ff}^{-1} - I)A_{fc}\mathbf{u}_c)_i}{(A_{fc}\mathbf{u}_c)_i}\right)\mathfrak{r}_f \\
&= \operatorname{diag}\left(\frac{((A_{ff}^{-1} - I)A_{fc}\mathbf{e}_c)_i}{(A_{fc}\mathbf{u}_c)_i}\right)\mathfrak{r}_f.
\end{aligned}$$

Thus,

$$\|(\Lambda^{-1} - I)\mathfrak{r}_f\|^2 = \sum_i \left(\frac{((A_{ff}^{-1} - I)A_{fc}\mathbf{e}_c)_i}{(A_{fc}\mathbf{u}_c)_i}\right)^2 (\mathfrak{r}_i)^2.$$

Taking $\eta = \min_i (A_{fc}\mathbf{u}_c)_i$, and noting that $I \leq A_{ff} \leq (1+\epsilon)I$ implies $I - A_{ff}^{-1} \leq \frac{\epsilon}{1+\epsilon}I$, we have

$$\begin{aligned}
\|(\Lambda^{-1} - I)\mathfrak{r}_f\|^2 &\leq \frac{1}{\eta^2}\sum_i \left(\left(\left(A_{ff}^{-1} - I\right)A_{fc}\mathbf{e}_c\right)_i\right)^2 (\mathfrak{r}_i)^2 \\
&\leq \frac{1}{\eta^2}\sum_i \left(\left(\left(A_{ff}^{-1} - I\right)A_{fc}\mathbf{e}_c\right)_i\right)^2 \sum_i (\mathfrak{r}_i)^2 \\
&= \frac{1}{\eta^2}\left\|\left(I - A_{ff}^{-1}\right)A_{fc}\mathbf{e}_c\right\|^2 \|\mathfrak{r}_f\|^2 \\
&\leq \frac{1}{\eta^2}\frac{\epsilon}{1+\epsilon}\left\langle\left(I - A_{ff}^{-1}\right)A_{fc}\mathbf{e}_c, A_{fc}\mathbf{e}_c\right\rangle.
\end{aligned}$$

Tracing back, this implies that

$$\begin{aligned}\|(I-\tilde{\Lambda}^{-1})\mathbf{z}\|^2 &\leq (1+\epsilon)\|(\Lambda^{-1}-I)\mathfrak{r}_f\|^2 \\ &\leq \frac{\epsilon}{\eta^2}\left\langle \left(I-A_{ff}^{-1}\right)A_{fc}\mathbf{e}_c, A_{fc}\mathbf{e}_c\right\rangle \\ &\leq \frac{\epsilon}{\eta^2}\langle \hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle,\end{aligned}$$

so that

$$\begin{aligned}\langle AP\mathbf{e}_c, P\mathbf{e}_c\rangle &= \langle \hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle + \|(I-\tilde{\Lambda}^{-1})\mathbf{z}\|^2 \\ &\leq \left(1+\frac{\epsilon}{\eta^2}\right)\langle \hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle.\end{aligned}$$

Thus,

$$\begin{aligned}\langle AP\mathbf{v}_c, \mathbf{v}_c\rangle &\leq 2\left(\delta^2\langle \hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle + (1+\delta)^2\|P\mathbf{e}_c\|_A^2\right) \\ &\leq 2\left(\delta^2 + (1+\delta)^2\left(1+\frac{\epsilon}{\eta^2}\right)\right)\langle \hat{A}_{ff}\mathbf{e}_c, \mathbf{e}_c\rangle,\end{aligned}$$

and so we arrive at the bound

$$RQ(P(\mathbf{u}_c + s\mathbf{v}_c)) \leq \left(1 - \frac{1}{2\left(\delta^2 + (1+\delta)^2\left(1+\frac{\epsilon}{\eta^2}\right)\right)}\right) RQ(\mathbf{u}).$$

□

To show that we are actually driving the Rayleigh quotient to zero uniformly, we must show that this constant is bounded independently of the iteration number. In particular, we want to show that we have the necessary control over constants δ and η to ensure that the reduction factor does not tend to 1 as we iterate.

Constant ϵ is given by our decomposition of the matrix $A_{ff} = I + \mathcal{E}$, where $0 \leq \mathcal{E} \leq \epsilon I$. It reflects how good of a job has been done in partitioning $\mathbb{R}^n$ into F and C in the sense of how well the F-F connections of A are dominated by the F-C connections. We can coarsely bound ϵ for a particular A_{ff} using Gerschgorin's theorem: $\epsilon \leq \max_{i\in F}\sum_{j\in F}|a_{ij}| - 1$. Considering standard finite-element or finite-difference matrices,

we can derive a slightly more optimistic bound (independent of h) by considering the fixed stencil-size and noticing that the diagonal scaling necessary to write $A_{ff} = I + \mathcal{E}$ cancels the powers of h in the matrix coefficients.

Parameter $\delta = \dfrac{\mathbf{e}_c^T P^T P \mathbf{e}_c}{\mathfrak{r}_c^T P^T P \mathfrak{r}_c} = \dfrac{\|P\mathbf{e}_c\|^2}{\|P\mathfrak{r}_c\|^2}$ is a measure of the relative size of $\mathbf{e}_c$. The normalization of $\mathbf{u} = \mathfrak{r} + \mathbf{e}$ suggests that, as the iteration converges to $\mathfrak{r}$, the size of $\|P\mathbf{e}_c\|$ relative to $\|P\mathfrak{r}_c\|$ should remain bounded, controlling the behavior of δ.

Parameter $\eta = \min_i (A_{fc}\mathbf{u})_i$ is more difficult to consider. There is nothing inherent in the iteration to suggest that η remains bounded away from zero. If, however, $\mathbf{u}$ is a good approximation to $\mathfrak{r}_c$, we expect that the variational coarsening preserves the relationship between small eigenvalues of discrete operators A and $A_P = P^T A P$, and those of the continuous operator. Smoothing $\mathbf{u}$ then results in a smooth $\mathbf{e}_c$, and the reduction in Rayleigh quotient from the coarse-grid correction must reflect a reduction in the size of $\mathbf{e}_c$ and not simply significant additional smoothing in $\mathbf{e}_c$ that could mask an increase in magnitude and an approach to the critical case of $(\mathbf{u}_c)_i = 0$ for some node i.

At present, we have not found acceptable analytic bounds on δ and η. Experience, however, suggests that such bounds are possible. Here, we demonstrate numerically that, for two simple examples, both of these parameters are well behaved.

Consider the two-dimensional problem, $-\nabla \cdot \mathcal{K} \nabla p$, discretized via bilinear finite elements with natural boundary conditions (so that $\mathfrak{r} = \mathbf{1}$) on an $n \times n$ grid. Let the coarse grid be chosen by red-black coarsening, so that no row of A_{fc} is zero, and consider the iteration, starting with a random initial guess, $\mathbf{u}$, normalized so that $\mathbf{u} \cdot \mathbf{1} = \|\mathbf{1}\|^2$:

(1) $\mathbf{u}_f = A_{ff}^{-1} A_{fc} \mathbf{u}_c$,

(2) $\Lambda^{-1} = \mathrm{Diag}\left(\frac{(\mathbf{u}_f)_i}{(A_{fc}\mathbf{u}_c)_i}\right)$, $P = \begin{bmatrix} \Lambda^{-1} A_{fc} \\ I \end{bmatrix}$, $A_c = P^T A P$,

(3) Solve $A_c \mathbf{v}_c = \lambda_{\min} P^T P \mathbf{v}_c$,

(4) $\mathbf{u}^{\mathrm{new}} = \frac{\mathbf{1}^T \mathbf{1}}{(Q\mathbf{v}_c)^T \mathbf{1}} Q \mathbf{v}_c$.

Iteration #	RQ	η	δ
0	6.4×10^{-1}	4.5×10^{-3}	2.2×10^{-1}
1	1.6×10^{-2}	1.1×10^{-1}	1.5×10^{-1}
2	8.4×10^{-4}	6.3×10^{-1}	7.7×10^{-3}
3	3.1×10^{-5}	9.7×10^{-1}	1.1×10^{-5}

Table 5.11: αAMGr Iteration for 33×33 Laplacian

Iteration #	RQ	η	δ
0	4.3×10^{0}	1.7×10^{-3}	2.2×10^{-1}
1	3.6×10^{-3}	1.5×10^{-3}	3.2×10^{2}
2	1.9×10^{-3}	9.0×10^{-1}	6.3×10^{-3}
3	9.3×10^{-6}	1.0×10^{0}	1.9×10^{-6}

Table 5.12: αAMGr Iteration for 33×33 Variable Coefficient Problem

This iteration is exactly the αAMGr iteration from above, with the additional step of normalizing $\mathbf{u}$ so that we may always write $\mathbf{u} = \mathbf{1} + \mathbf{e}$. Performing this iteration, we monitor δ and η, and also note the reduction in the Rayleigh quotient in Table 5.11 for $\mathcal{K} = 1$ on a 32×32 element grid. Notice that η increases monotonically to 1 and δ decreases monotonically to zero. Results for a more difficult problem, with diffusion coefficient $\mathcal{K}(\mathbf{x}) = \frac{1}{1000} + 10\mathbf{x}^T\mathbf{x}$, are shown in Table 5.12. Here, we see an initial decrease in η and increase in δ after the first step, but monotonic improvement afterward.

In addition to a result such as that in Theorem 6, showing that the iterates, $\mathbf{u}$, converge nicely to $\mathfrak{r}$, we must analyze the performance of the resulting method. Assuming exact fine-grid relaxation as in the setup phase above, we consider only the coarse-level correction step from the AMGr algorithm presented in Section 5.3.1. Recall that Λ is the emerging approximation to I, the diagonal term from the decomposition of $A_{ff} = I + \mathcal{E}$ (referred to as D in Section 5.3.1). Analogous to Equation 5.12 with Λ replacing D, we have

$$\|CLC\mathbf{e}\|_A \leq |\alpha| \min_{\theta} \left\| \left(\begin{bmatrix} A_{ff}^{-1} A_{fc} \\ I \end{bmatrix} - \theta \begin{bmatrix} \Lambda^{-1} A_{fc} \\ I \end{bmatrix} \right) \mathbf{v} \right\|_A + |\beta|,$$

for arbitrary fine-grid error, $\mathbf{e}$, where $\mathbf{v}$ is now an $\hat{A}_{ff}$-unit vector from the A-orthogonal

decomposition of an arbitrary error, $\mathbf{e} \in \mathbb{R}^n$ (see Equation 5.10). Since $\mathbf{e}$ is the error in solving $A\mathbf{x} = \mathbf{0}$, choose $\mathbf{v}$ to be orthogonal to $\mathfrak{r}_c$, for use below. Now, following the analysis in Equation 5.13,

$$\left\| \left(\begin{bmatrix} A_{ff}^{-1} A_{fc} \\ I \end{bmatrix} - \theta \begin{bmatrix} \Lambda^{-1} A_{fc} \\ I \end{bmatrix} \right) \mathbf{v} \right\|_A = (1-\theta)^2 + \theta^2 \|(\Lambda^{-1} - A_{ff}^{-1}) A_{fc} \mathbf{v}\|_{A_{ff}}^2.$$

Again, minimizing over θ shows that

$$\|CLC\mathbf{e}\|_A \leq |\alpha| \left(\frac{\|(\Lambda^{-1} - A_{ff}^{-1}) A_{fc} \mathbf{v}\|_{A_{ff}}^2}{1 + \|(\Lambda^{-1} - A_{ff}^{-1}) A_{fc} \mathbf{v}\|_{A_{ff}}^2} \right)^{\frac{1}{2}} + |\beta|.$$

We now would like to uniformly bound $\|(\Lambda^{-1} - A_{ff}^{-1}) A_{fc} \mathbf{v}\|_{A_{ff}}^2$, which is equivalent to bounding $\|(\Lambda^{-1} - I) A_{fc} \mathbf{v}\|_{A_{ff}}$:

$$\begin{aligned} \|(\Lambda^{-1} - A_{ff}^{-1}) A_{fc} \mathbf{v}\|_{A_{ff}} &\leq \|(\Lambda^{-1} - I) A_{fc} \mathbf{v}\|_{A_{ff}} + \|(I - A_{ff}^{-1}) A_{fc} \mathbf{v}\|_{A_{ff}} \\ &\leq \|(\Lambda^{-1} - I) A_{fc} \mathbf{v}\|_{A_{ff}} + \sqrt{\epsilon}. \end{aligned}$$

Recall that $\mathbf{v}$ has unit $\hat{A}_{ff}$-length, and so achieving a uniform bound is difficult because this Schur complement is not naturally accounted for. A uniform bound on

$$\|A_{ff}^{\frac{1}{2}} (\Lambda^{-1} - I) A_{fc} \hat{A}_{ff}^{-\frac{1}{2}}\|,$$

is necessary, which is clearly possible if $\Lambda^{-1} - I$ is small enough, but requires that the size of this term depend on h (and the original operator, A) to achieve uniform bounds.

5.3.3 Other Convergence Results

The results above indicate the convergence of αAMGr, but fail to give uniform convergence bounds. Thus, we now consider alternative techniques and assumptions to establish further convergence results. In the special case of a two-dimensional coarse grid, much simpler results are available. In this case, we may again write $\mathbf{u}_c = \mathfrak{r}_c + \mathbf{e}_c$, now taking $\mathfrak{r}_c \perp \mathbf{e}_c$, and so $\{\mathfrak{r}_c, \mathbf{e}_c\}$ spans the entire coarse-grid space. This reduction in complexity allows for a complete analysis of the αAMGr algorithm, with the coarse-grid eigenproblem replaced by inverse iteration.

Theorem 7. *Let $n_c = 2$, and let approximation $\mathbf{u}$, $\mathbf{u}_c = \mathfrak{r}_c + \mathbf{e}_c$ with $\|\mathfrak{r}_c\| = 1$ and $\mathfrak{r}_c \perp \mathbf{e}_c$ be given. Define the usual αAMGr interpolation operator, $P = \begin{bmatrix} \Lambda^{-1}A_{fc} \\ I \end{bmatrix}$, with $\Lambda^{-1}A_{fc}\mathbf{u}_c = A_{ff}^{-1}A_{fc}\mathbf{u}_c$, and coarse-grid matrix $A_P = P^TAP$. Assume also that $A\mathfrak{r} = \mathbf{0}$, $A_{ff} = I + \mathcal{E}$, for $0 \leq \mathcal{E} \leq \epsilon I$, and that $\begin{bmatrix} I & -A_{fc} \\ -A_{fc}^T & A_{cc} \end{bmatrix} \geq 0$. Finally, let $\bar{\eta} = \min_i(\mathfrak{r}_f)_i$ and $\eta = \min(\min_i(\mathfrak{r}_f + A_{fc}\mathbf{e}_c)_i, \bar{\eta}) > 0$. Then, the coarse-grid solution, $\mathbf{v}_c$, to the inverse iteration, $\mathbf{v}_c = A_P^{-1}\mathbf{u}_c$, satisfies*

$$\mathbf{v}_c = k(\zeta\mathfrak{r}_c + \mathbf{e}_c),$$

for constant k, and $\zeta \geq 1 + \frac{1}{1+\|\mathbf{e}_c\|^2}\frac{\eta^2}{\epsilon(1+\epsilon)\|\mathfrak{r}_f\|^2}$.

Proof. Let $W = [\mathfrak{r}_c, \frac{\mathbf{e}_c}{\|\mathbf{e}_c\|}]$ be the orthogonal transformation from Cartesian coordinates to the $(\mathfrak{r}_c, \mathbf{e}_c)$ coordinate system. The inverse iteration, $\mathbf{v}_c = A_P^{-1}\mathbf{u}_c$, may then be rewritten as

$$\mathbf{v}_c = W(W^TA_PW)^{-1}W^T(\mathfrak{r}_c + \mathbf{e}_c).$$

Note that $P\mathbf{u}_c = Q\mathbf{u}_c$, where Q denotes the ideal interpolation operator, $Q = \begin{bmatrix} A_{ff}^{-1}A_{fc} \\ I \end{bmatrix}$, and that $Q\mathfrak{r}_c = \mathfrak{r}$, so that

$$P\mathfrak{r}_c = \mathfrak{r} + \begin{pmatrix} (\Lambda^{-1}-I)\mathfrak{r}_f \\ \mathbf{0} \end{pmatrix},$$
$$P\mathbf{e}_c = Q\mathbf{e}_c - \begin{pmatrix} (\Lambda^{-1}-I)\mathfrak{r}_f \\ \mathbf{0} \end{pmatrix},$$

and compute $\alpha \equiv \mathfrak{r}_c^TA_P\mathfrak{r}_c = \mathfrak{r}_f^T(\Lambda^{-1}-I)A_{ff}(\Lambda^{-1}-I)\mathfrak{r}_f$. Notice then that $\mathbf{e}_c^TA_P\mathfrak{r}_c = -\alpha$, because $Pe_c = Pu_c - P\mathfrak{r}_c = Qu_c - P\mathfrak{r}_c$, and Qu_c is A-orthogonal to any vector that is zero on all coarse points.

Thus, taking $\gamma = \|\mathbf{e}_c\|$ and $\beta = \mathbf{e}_c^T\hat{A}_{ff}\mathbf{e}_c$, the inverse iteration matrix may be rewritten as

$$W^TP^TAPW = \begin{bmatrix} \alpha & -\frac{\alpha}{\gamma} \\ -\frac{\alpha}{\gamma} & \frac{\alpha+\beta}{\gamma^2} \end{bmatrix}.$$

The inverse iteration then takes the form

$$\begin{aligned}
\mathbf{v}_c &= W \begin{bmatrix} \alpha & -\frac{\alpha}{\gamma} \\ -\frac{\alpha}{\gamma} & \frac{\alpha+\beta}{\gamma^2} \end{bmatrix}^{-1} W^T (\mathfrak{r}_c + \mathbf{e}_c) \\
&= W \frac{\gamma^2}{\alpha\beta} \begin{bmatrix} \frac{\alpha+\beta}{\gamma^2} & \frac{\alpha}{\gamma} \\ \frac{\alpha}{\gamma} & \alpha \end{bmatrix} \begin{pmatrix} 1 \\ \gamma \end{pmatrix} \\
&= \left(\frac{\alpha+\beta}{\alpha\beta} + \frac{\gamma^2}{\beta} \right) \mathfrak{r}_c + \left(\frac{\gamma+\gamma^3}{\beta} \right) \frac{\mathbf{e}_c}{\gamma} \\
&= \left(\frac{1+\gamma^2}{\beta} \right) \left(\left(\frac{\alpha+\beta}{(1+\gamma^2)\alpha} + \frac{\gamma^2}{1+\gamma^2} \right) \mathfrak{r}_c + \mathbf{e}_c \right).
\end{aligned}$$

Now consider $\beta = \mathbf{e}_c^T \hat{A}_{ff} \mathbf{e}_c$, and note that

$$\hat{A}_{ff} = A_{cc} - A_{fc}^T A_{fc} + A_{fc}^T (I - A_{ff}^{-1}) A_{fc} \geq A_{fc}^T (I - A_{ff}^{-1}) A_{fc},$$

as $A_{cc} - A_{fc}^T A_{fc} \geq 0$, by assumption as in Equation 5.6. Then, since

$$(I - A_{ff}^{-1}) = (I - (I + \mathcal{E})^{-1}) = \mathcal{E}(I + \mathcal{E})^{-1},$$

we see that

$$\hat{A}_{ff} \geq \frac{1}{\epsilon} A_{fc}^T \mathcal{E}^2 (I + \mathcal{E})^{-1} A_{fc},$$

as

$$\mathcal{E}^2 (I + \mathcal{E})^{-1} \leq \epsilon \mathcal{E} (I + \mathcal{E})^{-1}.$$

In particular, this implies that $\beta \geq \frac{1}{\epsilon} \mathbf{e}_c^T A_{fc}^T \mathcal{E}^2 (I + \mathcal{E})^{-1} A_{fc} \mathbf{e}_c$. We now turn our attention to α, which can be bounded above as follows:

$$\begin{aligned}
\alpha &= \|(\Lambda^{-1} - I)\mathfrak{r}_f\|_{A_{ff}}^2 \leq (1+\epsilon) \|(\Lambda^{-1} - I)\mathfrak{r}_f\|^2 \\
&= (1+\epsilon) \sum_i \left(\frac{((A_{ff}^{-1} - I) A_{fc} \mathbf{e}_c)_i}{(\mathfrak{r}_f + A_{fc}\mathbf{e}_c)_i} \right)^2 (\mathfrak{r}_f)_i^2 \\
&\leq \frac{1+\epsilon}{\eta^2} \sum_i ((A_{ff}^{-1} - I) A_{fc} \mathbf{e}_c)_i^2 \sum_i (\mathfrak{r}_f)_i^2 \\
&\leq \frac{(1+\epsilon)\|\mathfrak{r}_f\|^2}{\eta^2} \sum_i ((A_{ff}^{-1} - I) A_{fc} \mathbf{e}_c)_i^2
\end{aligned}$$

and so,

$$\begin{aligned}\alpha &= \frac{(1+\epsilon)\|\mathfrak{r}_f\|^2}{\eta^2}\|(A_{ff}^{-1}-I)A_{fc}\mathbf{e}_c\|^2 \\ &\leq \frac{(1+\epsilon)\|\mathfrak{r}_f\|^2}{\eta^2}\|(A_{ff}^{-1}-I)A_{fc}\mathbf{e}_c\|_{A_{ff}}^2 \\ &= \frac{(1+\epsilon)\|\mathfrak{r}_f\|^2}{\eta^2}\mathbf{e}_c^T A_{fc}^T A_{ff}^{-1}(A_{ff}-I)A_{ff}(A_{ff}-I)A_{ff}^{-1}A_{fc}\mathbf{e}_c \\ &= \frac{(1+\epsilon)\|\mathfrak{r}_f\|^2}{\eta^2}\mathbf{e}_c^T A_{fc}^T \mathcal{E}^2(I+\mathcal{E})^{-1}A_{fc}\mathbf{e}_c \leq \frac{\epsilon(1+\epsilon)\|\mathfrak{r}_f\|^2}{\eta^2}\beta.\end{aligned}$$

Thus, $\mathbf{v}_c = \left(\frac{1+\gamma^2}{\beta}\right)\left(\left(\frac{\alpha+\beta}{(1+\gamma^2)\alpha}+\frac{\gamma^2}{1+\gamma^2}\right)\mathfrak{r}_c+\mathbf{e}_c\right)$ can be seen to have the form, $\mathbf{v}_c = k(\zeta\mathfrak{r}_c+e_c)$, for $k=\left(\frac{1+\gamma^2}{\beta}\right)$, and

$$\begin{aligned}\zeta &= \left(\frac{\alpha+\beta}{(1+\gamma^2)\alpha}+\frac{\gamma^2}{1+\gamma^2}\right) \geq 1+\frac{\beta}{(1+\gamma^2)\alpha} \\ &\geq 1+\frac{1}{1+\gamma^2}\frac{\eta^2}{\epsilon(1+\epsilon)\|\mathfrak{r}_f\|^2}.\end{aligned}$$

□

Corollary 2. *Under the hypotheses of Theorem 7, the inverse iteration process,* $\mathbf{u}_c \leftarrow A_P^{-1}\mathbf{u}_c$ *converges uniformly to* $\mathbf{u}_c = \mathfrak{r}_c$.

Proof. Let $\bar{\eta} = \min_i(\mathfrak{r}_f)_i$. Renormalizing $\mathbf{v}_c = \mathfrak{r}_c + \frac{1}{\zeta}\mathbf{e}_c$, we see that $\frac{1}{\zeta} < 1$, thus

$$\eta^{\text{new}} = \min\left(\min_i\left(\mathfrak{r}_f+\frac{1}{\zeta}A_{fc}\mathbf{e}_c\right)_i, \bar{\eta}\right) \geq \eta^{\text{old}},$$

and that

$$\gamma^{\text{new}} = \left\|\frac{1}{\zeta}\mathbf{e}_c\right\| < \gamma^{\text{old}}.$$

Thus, for any initial $\eta^{(0)}$ and $\gamma^{(0)}$, we have ζ from Theorem 7 satisfying

$$\zeta \geq 1+\frac{1}{1+(\gamma^{(0)})^2}\frac{(\eta^{(0)})^2}{\epsilon(1+\epsilon)}.$$

□

The success of inverse iteration in this case suggests that it may be useful also in the case when $n_c > 2$. It has not yet been shown that coarse-grid inverse iteration

yields an N-independent bound in the more general case; however, we believe that the following avenue of approach may be useful in achieving such a bound.

Given the current approximation, $\mathbf{u}_c = \mathfrak{r}_c + \mathbf{e}_c$, define the orthonormal matrix, V, to span the orthogonal complement of $\{\mathfrak{r}_c, \mathbf{e}_c\}$, and then the orthonormal matrix, $W = (\mathfrak{r}_c, \frac{\mathbf{e}_c}{\|\mathbf{e}_c\|}, V)$, to span $\mathbb{R}^{n_c}$. Under the assumptions that $\mathbf{u}_c = \mathfrak{r}_c + \mathbf{e}_c$, $\|\mathfrak{r}_c\| = 1$ and $\mathfrak{r}_c \perp \mathbf{e}_c$, we have $\langle \mathbf{u}_c, \mathfrak{r}_c \rangle = 1$, and so we get

$$A_P^{-1}\mathbf{u}_c = W(W^T A_P W)^{-1} W\mathbf{u}_c = W(W^T A_P W)^{-1} \begin{pmatrix} 1 \\ \|\mathbf{e}_c\| \\ \mathbf{0} \end{pmatrix}.$$

Now, let $X = A_P - \hat{A}_{ff}$ and notice that $X(\mathfrak{r}_c + \mathbf{e}_c) = 0$. Thus, we have $X\mathbf{e}_c = -X\mathfrak{r}_c$, and so

$$\begin{aligned} W^T A_P W &= W^T \hat{A}_{ff} W + W^T X W \\ &= \begin{pmatrix} 0 & 0 & \mathbf{0}^T \\ 0 & \frac{\mathbf{e}_c^T \hat{A}_{ff}\mathbf{e}_c}{\|\mathbf{e}_c\|^2} & \frac{\mathbf{e}_c^T \hat{A}_{ff} V}{\|\mathbf{e}_c\|} \\ \mathbf{0} & \frac{V^T \hat{A}_{ff}\mathbf{e}_c}{\|\mathbf{e}_c\|} & V^T \hat{A}_{ff} V \end{pmatrix} + \begin{pmatrix} \mathfrak{r}_c^T X \mathfrak{r}_c & -\frac{\mathfrak{r}_c^T X \mathfrak{r}_c}{\|\mathbf{e}_c\|} & \mathfrak{r}_c^T X V \\ -\frac{\mathfrak{r}_c^T X \mathfrak{r}_c}{\|\mathbf{e}_c\|} & \frac{\mathfrak{r}_c^T X \mathfrak{r}_c}{\|\mathbf{e}_c\|^2} & -\frac{\mathfrak{r}_c^T X V}{\|\mathbf{e}_c\|} \\ V^T X \mathfrak{r}_c & -\frac{V^T X \mathfrak{r}_c}{\|\mathbf{e}_c\|} & V^T X V \end{pmatrix}. \end{aligned}$$

Now, take $\gamma = \|\mathbf{e}_c\|$, $\alpha = \mathfrak{r}_c^T X \mathfrak{r}_c$, $\mathbf{a} = V^T X \mathfrak{r}_c$, $\beta = \mathbf{e}_c^T \hat{A}_{ff}\mathbf{e}_c$, and $\mathbf{b} = V^T \hat{A}_{ff}\mathbf{e}_c$, reducing the inverse iteration step to

$$A_P^{-1}\mathbf{u}_c = (\mathfrak{r}_c, \frac{\mathbf{e}_c}{\|\mathbf{e}_c\|}, V) \begin{pmatrix} \alpha & -\frac{\alpha}{\gamma} & \mathbf{a}^T \\ -\frac{\alpha}{\gamma} & \frac{\alpha+\beta}{\gamma^2} & \frac{-\mathbf{a}^T+\mathbf{b}^T}{\gamma} \\ \mathbf{a} & \frac{-\mathbf{a}+\mathbf{b}}{\gamma} & V^T A_P V \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ \gamma \\ \mathbf{0} \end{pmatrix}.$$

Gaussian elimination then yields

$$\mathbf{v}_c = A_P^{-1}\mathbf{u}_c = \left(\frac{1}{\alpha} + \frac{\mathbf{a}^T\mathbf{x}}{\alpha} + \left(\frac{\gamma^2+1}{\beta} + \frac{\mathbf{b}^T\mathbf{x}}{\beta}\right)\right)\mathfrak{r}_c + \left(\frac{\gamma^2+1}{\beta} + \frac{\mathbf{b}^T\mathbf{x}}{\beta}\right)\mathbf{e}_c - V\mathbf{x},$$

where $\mathbf{x} = \left(V^T A_P V - \frac{\mathbf{a}\mathbf{a}^T}{\alpha} - \frac{\mathbf{b}\mathbf{b}^T}{\beta}\right)^{-1}\left(\frac{\mathbf{a}}{\alpha} + (\gamma^2+1)\frac{\mathbf{b}}{\beta}\right)$. Since the decrease orthogonal to $\mathfrak{r}_c$ is of primary importance, consider the relative size of the $\mathbf{e}_c$ and $V\mathbf{x}$ terms to the

$\mathfrak{r}_c$ term, rewriting $\mathbf{v}_c$ as

$$\mathbf{v}_c = \left(\frac{1}{\alpha} + \frac{\mathbf{a}^T\mathbf{x}}{\alpha} + \left(\frac{\gamma^2+1}{\beta} + \frac{\mathbf{b}^T\mathbf{x}}{\beta}\right)\right)\left(\mathfrak{r}_c + \frac{\frac{\gamma^2+1}{\beta} + \frac{\mathbf{b}^T\mathbf{x}}{\beta}}{\frac{1}{\alpha} + \frac{\mathbf{a}^T\mathbf{x}}{\alpha} + \left(\frac{\gamma^2+1}{\beta} + \frac{\mathbf{b}^T\mathbf{x}}{\beta}\right)}\mathbf{e}_c - \frac{1}{\frac{1}{\alpha} + \frac{\mathbf{a}^T\mathbf{x}}{\alpha} + \left(\frac{\gamma^2+1}{\beta} + \frac{\mathbf{b}^T\mathbf{x}}{\beta}\right)}V\mathbf{x}\right).$$

The dependence of the definition of P on the infinity-norm of $\mathbf{e}_c$ suggests that convergence is naturally measured in this norm. In particular, we ask that

$$\left\| \frac{\frac{\gamma^2+1}{\beta} + \frac{\mathbf{b}^T\mathbf{x}}{\beta}}{\frac{1}{\alpha} + \frac{\mathbf{a}^T\mathbf{x}}{\alpha} + \left(\frac{\gamma^2+1}{\beta} + \frac{\mathbf{b}^T\mathbf{x}}{\beta}\right)}\mathbf{e}_c - \frac{1}{\frac{1}{\alpha} + \frac{\mathbf{a}^T\mathbf{x}}{\alpha} + \left(\frac{\gamma^2+1}{\beta} + \frac{\mathbf{b}^T\mathbf{x}}{\beta}\right)}V\mathbf{x} \right\|_\infty < \|\mathbf{e}_c\|_\infty,$$

which follows easily if

$$\|V\mathbf{x}\|_\infty < \left(\frac{1}{\alpha} + \frac{\mathbf{a}^T\mathbf{x}}{\alpha}\right)\|\mathbf{e}_c\|_\infty,$$

under the assumption that $\frac{1}{\alpha} + \frac{\mathbf{a}^T\mathbf{x}}{\alpha} > 0$. Proof of such an inequality is an open question.

5.3.4 Contraction Argument

The above theory still does not provide a satisfactory bound on convergence of the adaptive process to the null-space component, $\mathfrak{r}$, of A. Convergence in a neighborhood of this solution may be guaranteed, however, by a contraction argument under further assumptions.

In what follows, we use the notation,

$$\mathcal{D}(\mathbf{w}) = \text{Diag}(\ldots, w_i, \ldots),$$

for a given vector, $\mathbf{w}$, to be the diagonal matrix with $\mathbf{w}$ as the diagonal terms. We also use the Jacobian notation,

$$J(G, \mathbf{x})[\mathbf{y}],$$

to denote the Jacobian of the map G with respect to $\mathbf{x}$, evaluated at $\mathbf{y}$, or $J(\mathbf{z}, \mathbf{x})[\mathbf{y}]$ to denote the Jacobian of the map from $\mathbf{x}$ to $\mathbf{z}$, evaluated at $\mathbf{y}$.

Theorem 8. *Suppose that* $n_c = n_f$ *and that* $A_{cf} = A_{fc} = I$. *Let* $\mathbf{u}_c = \mathfrak{r}_c + \mathbf{e}_c$, *for* $\mathfrak{r}_c = \mathbf{1}_c$, $\mathfrak{r}_c \perp \mathbf{e}_c$. *Define the map,* $G : \mathbf{e}_c \to \mathbf{e}_c^{new}$, *to be defining* Λ *based on* $\mathbf{u}_c$ *as in Equation 5.14, then finding the new* $\mathbf{e}_c$ *from the coarse-grid eigenvalue problem,* $P^T AP\mathbf{v}_c = \lambda_{min}\mathbf{v}_c$, $\langle \mathbf{v}_c, \mathfrak{r}_c \rangle = \|\mathfrak{r}_c\|^2$, *and* $\mathbf{e}_c^{new} = \mathbf{v}_c - \mathfrak{r}_c$.

The map, G, *is a contraction at the solution,* $\mathbf{e}_c = \mathbf{0}$. *Further, the norm of the Jacobian of* G *with respect to* $\mathbf{e}_c$ *satisfies*

$$\|J(G, \mathbf{e}_c)\| \leq \frac{\epsilon}{1+\epsilon},$$

and so G *remains a contraction in some neighborhood of* $\mathbf{e}_c = \mathbf{0}$.

Proof. Notice first that the coarse-grid operator,

$$\begin{aligned} A_P = P^T AP &= A_{cc} - 2A_{cf}\Lambda^{-1}A_{fc} + A_{cf}\Lambda^{-1}A_{ff}\Lambda^{-1}A_{fc} \\ &= (A_{cc} - A_{cf}A_{ff}^{-1}A_{fc}) + (A_{cf}A_{ff}^{-1}A_{fc} - 2A_{cf}\Lambda^{-1}A_{fc} + A_{cf}\Lambda^{-1}A_{ff}\Lambda^{-1}A_{fc}) \\ &= (A_{cc} - A_{cf}A_{ff}^{-1}A_{fc}) + A_{cf}(\Lambda^{-1} - A_{ff}^{-1})A_{ff}(\Lambda^{-1} - A_{ff}^{-1})A_{fc}, \end{aligned}$$

is the sum of two symmetric and positive semidefinite parts, with

$$(A_{cc} - A_{cf}A_{ff}^{-1}A_{fc})\mathfrak{r}_c = \mathbf{0}$$

$$\text{and } A_{cf}(\Lambda^{-1} - A_{ff}^{-1})A_{ff}(\Lambda^{-1} - A_{ff}^{-1})A_{fc}(\mathfrak{r}_c + \mathbf{e}_c) = \mathbf{0}.$$

The coarse-grid step is then to find

$$\mathbf{e}_c = \underset{\tilde{\mathbf{e}}_c}{\operatorname{argmin}} \operatorname{RQ}(A_P, \mathfrak{r}_c + \tilde{\mathbf{e}}_c),$$

where $\operatorname{RQ}(A, \mathbf{v}) = \frac{\langle A\mathbf{v}, \mathbf{v} \rangle}{\langle \mathbf{v}, \mathbf{v} \rangle}$, under the constraint $\langle \mathbf{e}_c, \mathfrak{r}_c \rangle = 0$. The fine-grid error, $\mathbf{e}_f$, is then defined by an exact fine-grid relaxation,

$$A_{ff}(\mathfrak{r}_f + \mathbf{e}_f) - A_{fc}(\mathfrak{r}_c + \mathbf{e}_c) = 0.$$

The iteration may be viewed as a mapping from $\mathbf{e}_c \to \mathbf{e}_c^{\text{new}}$. Defining $\Xi = \Lambda^{-1} - I$, and ξ to be the vector such that $\xi_i = \Xi_{ii}$ (i.e., $\Xi = \mathcal{D}(\xi)$), the iteration proceeds as

$$\mathbf{e}_c \to \xi \to \mathbf{e}_c.$$

At this point, we use the assumption that $A_{cf} = A_{fc} = I$ (and that $n_c = n_f$), define $E = \mathcal{D}(\mathbf{e}_c)$, and rewrite Equation 5.14 as

$$(I + \Xi)(\mathfrak{r}_c + \mathbf{e}_c) = (I + E)(\mathfrak{r}_c + \xi) = A_{ff}^{-1}(\mathfrak{r}_c + \mathbf{e}_c) = \mathfrak{r}_c + A_{ff}^{-1}\mathbf{e}_c,$$

where we make use of the fact that $\Xi\mathbf{e}_c = E\xi$, or

$$\xi = (I + E)^{-1}(\mathfrak{r}_c + A_{ff}^{-1}\mathbf{e}_c) - \mathfrak{r}_c.$$

Now, as $\xi = \xi(\mathbf{e}_c)$, we consider the Jacobian, $J(\xi, \mathbf{e}_c) = \frac{\partial \xi}{\partial \mathbf{e}_c}$, evaluated at $\mathbf{e}_c$,

$$J(\xi, \mathbf{e}_c)[\mathbf{e}_c] = (I + E)^{-1}A_{ff}^{-1} - (I + E)^{-2}(I + \mathcal{D}(A_{ff}^{-1}\mathbf{e}_c)),$$

recalling that $\mathfrak{r}_c = \mathbf{1}$. At the solution, $\mathbf{e}_c = \mathbf{0}$, and we have

$$J(\xi, \mathbf{e}_c)[\mathbf{0}] = A_{ff}^{-1} - I. \tag{5.15}$$

It is easy to see from the assumptions that

$$0 \leq I - A_{ff}^{-1} \leq \frac{\epsilon}{\epsilon + 1} I < I,$$

and so this part of the map is a contraction.

For the second part of the map, we first find

$$\mathbf{u}_c = \underset{\mathbf{v}_c}{\operatorname{argmin}}\, \mathrm{RQ}(A_P, \mathbf{v}_c) = \underset{\mathbf{v}_c}{\operatorname{argmin}}\, \frac{\langle A_P\mathbf{v}_c, \mathbf{v}_c\rangle}{\langle \mathbf{v}_c, \mathbf{v}_c\rangle} \tag{5.16}$$

subject to the constraint

$$\langle \mathbf{u}_c, \mathfrak{r}_c\rangle - \langle \mathfrak{r}_c, \mathfrak{r}_c\rangle = 0$$

and set $\mathbf{e}_c^{\text{new}} = \mathbf{u}_c - \mathfrak{r}_c$. This yields an implicit relationship between ξ and $\mathbf{e}_c^{\text{new}}$. Taking the gradient of (5.16), we have the set of equations

$$A_P\mathbf{u}_c - \frac{\langle A_P\mathbf{u}_c, \mathbf{u}_c\rangle}{\langle \mathbf{u}_c, \mathbf{u}_c\rangle}\mathbf{u}_c = 0$$

plus the original constraint equation

$$\langle \mathbf{u}_c, \mathfrak{r}_c\rangle - \langle \mathfrak{r}_c, \mathfrak{r}_c\rangle = 0.$$

Note that $A_P = A_P(\xi)$ and, to belabor the point, this has the form of a set of equations

$$F(\mathbf{u}_c, \xi) = A_P(\xi)\mathbf{u}_c - \frac{\langle A_P(\xi)\mathbf{u}_c, \mathbf{u}_c\rangle}{\langle \mathbf{u}_c, \mathbf{u}_c\rangle}\mathbf{u}_c = 0. \tag{5.17}$$

With the constraint there are n_c+1 equations, but they are consistent, as the first n_c do not determine the constant multiplier and are therefore singular. The constraint fixes the scaling.

Implicit differentiation of F yields

$$J(F, \mathbf{u}_c)J(\mathbf{u}_c, \mathbf{e}_c)J(\mathbf{e}_c, \xi) + J(F, \xi) = 0, \tag{5.18}$$

where, of course, $J(\mathbf{u}_c, \mathbf{e}_c) = I$. We deal with the constraint equation later.

We now need to construct $J(F, \mathbf{u}_c)$ and $J(F, \xi)$, which is messy but straightforward. Referring to the definition of A_P and recalling that $A_{fc} = A_{cf} = I$, $A_{ff} = I + \mathcal{E}$ and $\Lambda^{-1} = I + \Xi$, we have

$$A_P = (A_{cc} - 2I + A_{ff}) + \Xi^2 + \Xi\mathcal{E} + \mathcal{E}\Xi + \Xi\mathcal{E}\Xi.$$

This yields

$$J(F, \xi) = 2\Xi\mathcal{D}(\mathbf{u}_c) + \mathcal{D}(\mathcal{E}\mathbf{u}_c) + \mathcal{E}\mathcal{D}(\mathbf{u}_c) + \mathcal{D}(\mathcal{E}\Xi\mathbf{u}_c) + \Xi\mathcal{E}\mathcal{D}(\mathbf{u}_c) + \mathbf{u}_c(\nabla_\xi \mathrm{RQ}(A_P, \mathbf{u}_c))^T,$$

where

$$\nabla_\xi \mathrm{RQ}(A_P, \mathbf{u}_c) = \frac{1}{< \mathbf{u}_c, \mathbf{u}_c >}(2\Xi\mathcal{D}(\mathbf{u}_c) + \mathcal{D}(\mathcal{E}\mathbf{u}_c) + \mathcal{E}\mathcal{D}(\mathbf{u}_c) + \mathcal{D}(\mathcal{E}\Xi\mathbf{u}_c) + \Xi\mathcal{E}\mathcal{D}(\mathbf{u}_c))\mathbf{u}_c.$$

The important thing to notice is that at the solution, $\mathbf{u}_c = \mathfrak{r}_c$, $\Xi = 0$, and $\xi = \mathbf{0}$, yielding

$$\nabla_\xi \mathrm{RQ}(A_P, \mathbf{u}_c) = \mathbf{0},$$

and the only surviving term in $J(F, \xi)$ is $(\mathcal{E}_{ij}u_j) = \mathcal{E}$, because of the assumption that $\mathfrak{r}_c = \mathbf{1}_c$, yielding

$$J(F, \xi)[\mathbf{0}] = \mathcal{E}.$$

Now, to compute $J(F, \mathbf{u}_c)$, we have

$$J(F, \mathbf{u}_c) = A_P - \mathrm{RQ}(A_P, \mathbf{u}_c)I + \mathbf{u}_c(\nabla_{\mathbf{u}_c}\mathrm{RQ}(A_P, \mathbf{u}_c))^T,$$

where, as stated before,

$$\nabla_{\mathbf{u}_c}\mathrm{RQ}(A_P, \mathbf{u}_c) = \frac{1}{< \mathbf{u}_c, \mathbf{u}_c >}(A_P\mathbf{u}_c - \frac{< A_P\mathbf{u}_c, \mathbf{u}_c >}{< \mathbf{u}_c, \mathbf{u}_c >}\mathbf{u}_c),$$

and, at the solution, this yields

$$J(F, \mathbf{u}_c)[\mathfrak{r}_c] = A_{cc} - 2I + A_{ff} = A_{cc} - I + \mathcal{E}.$$

Now consider the constraint. If we think of the constraint equation as the last in Equation 5.17, this adds one extra equation to the set in 5.18. The last row of $J(F, \xi)$ is $\mathbf{0}^T$ because the constraint does not depend on ξ. The constraint equation also adds one row to $J(F, \mathbf{u}_c)$, and that last row is $\mathfrak{r}_c^T$. That is, the columns of $J(\mathbf{e}_c, \xi)$ are all orthogonal to $\mathfrak{r}_c$. We must check that this augmented system has a solution, because matrix $J(F, \mathbf{u}_c)[\mathfrak{r}_c] = A_{cc} - 2I + A_{ff}$ has $\mathfrak{r}_c$ as its null space. Note that the right side, $-J(F, \xi)[\mathbf{0}] = -\mathcal{E}$, also has $\mathfrak{r}_c$ as its null space and (since it is symmetric) each column of $\mathcal{E}$ is orthogonal to $\mathfrak{r}_c$. Then, since $J(F, \mathbf{u}_c)[\mathfrak{r}_c]$ is also symmetric, each column of $\mathcal{E}$ must be in its range (this is Fredholm's theorem: that $(\mathrm{Range}(A))^\perp = \mathrm{Null}(A^T)$). So, we know that a solution, $J(\mathbf{e}_c, \xi)$, exists. If we additionally ask for the solution to satisfy the constraint, $J(\mathbf{e}_c, \xi)\mathfrak{r}_c = \mathbf{0}$, with the minimal RQ, we then get a unique value for $J(\mathbf{e}_c, \xi)[\mathbf{0}]$.

Note that in order to get the result we want, we must assume that $\mathfrak{r}_c$ is the only null vector of $A_{cc} - I$ (as $\mathfrak{r}_c$ is the only null vector of $\mathcal{E}$). Using the results in Equations 5.15 and 5.18, we have

$$(A_{cc} - I + \mathcal{E})J(\mathbf{e}_c, \xi)J(\xi, \mathbf{e}_c) = \mathcal{E}(I - A_{ff}^{-1}) = \mathcal{E}^2(I + \mathcal{E})^{-1}.$$

Suppose now that V is the unitary matrix that diagonalizes $\mathcal{E}$, i.e.,

$$\Psi = \mathcal{D}(\psi) = V\mathcal{E}V^*.$$

Under this change of basis, we have

$$V^*(VA_{cc}V^* - I + \Psi)VJ(\mathbf{e}_c, \xi)J(\xi, \mathbf{e}_c) = V^*(\Psi^2(I + \Psi)^{-1})V.$$

So now we get the bound

$$\begin{aligned} C &= \underset{\mathbf{e}\perp V\mathbf{r}}{\operatorname{argmax}} \frac{< \Psi^2(I + \Psi)^{-1}\mathbf{e}, \mathbf{e} >}{< (V^*A_{cc}V - I)\mathbf{e}, \mathbf{e} > + < \Psi\mathbf{e}, \mathbf{e} >} \\ &\leq \underset{\mathbf{e}\perp V\mathbf{r}}{\operatorname{argmax}} \frac{< \Psi^2(I + \Psi)^{-1}\mathbf{e}, \mathbf{e} >}{< \Psi\mathbf{e}, \mathbf{e} >} \leq \frac{\epsilon}{1 + \epsilon}. \end{aligned}$$

Thus, the norm of the Jacobian of the mapping, $J(G, \mathbf{e}_c) = J(\mathbf{e}_c, \xi)J(\xi, \mathbf{e}_c)$, is bounded uniformly less than 1, yielding convergence in a neighborhood of the solution.

□

5.4 Adaptive Smoothed Aggregation

* Over the last decade, smoothed aggregation (SA; cf. [77, 82]) has emerged as an efficient multilevel algebraic solver for the solution of the algebraic systems obtained by discretizing certain classes of differential equations on unstructured meshes. In particular, SA is often very efficient at solving the systems that arise from problems of 3D thin-body elasticity, a task that can tax traditional algebraic multigrid techniques.

As with classical AMG [23, 72, 73], the standard smoothed aggregation method bases its transfer operators on certain assumptions about the nature of smooth error. For SA applied to discretizations of elliptic partial differential equations, this assumption usually takes the form of explicit knowledge of the near null space of the associated weak form. This knowledge is easy to obtain for large classes of problems. For example, it is simple to determine the near null space for finite element discretizations of second- or fourth-order PDEs, including many nonscalar problems. In more general situations, however, this knowledge may not be readily available. Here, we propose the αSA method to address this situation. The objective of the setup phase of αSA is to compute a set of prototype vectors, B, that represent error components that relaxation is slow to resolve.

* This section has also appeared as [26]

The setup phase for αSA is easiest to describe as an adaptive process. We start from a given, primitive parent method (possibly a simple relaxation scheme) with error propagation operator M_0, and a current, but possibly empty, set, B, of prototypes (error components that M_0 does not effectively reduce). We attempt to enhance B by first putting M_0 to the test: given a small number, ν, of iterations and a random initial guess, $\mathbf{e}_0$, compute

$$\mathbf{e}_\nu \leftarrow M_0^\nu \mathbf{e}_0. \tag{5.19}$$

If the method performs well in the sense that $\mathbf{e}_\nu$ is much smaller than $\mathbf{e}_0$ in an appropriate norm, then it is accepted as the solver and the adaptive scheme stops. Otherwise, the resulting approximation, $\mathbf{e}_\nu$, is expected to be rich in the error components that are not effectively reduced by M_0, so it is added to the prototype set, B. The new prototype set is then used to construct an improved child method, with error propagation operator M_1. The whole process can then be repeated with M_1 in place of M_0, continuing in this way to generate a sequence of hopefully improving methods.

Thus, we iterate on the method itself, improving the current version by having it compute its own troublesome components - those that it does not effectively reduce - and then adjusting the coarsening process accordingly to produce a new method. Old prototype components are also used in this adjustment process to ensure that the new method continues to reduce them efficiently. This improvement process repeats until the current method shows itself to be capable of efficient solution of the problem of interest. The iteration on the method is called the adaptive setup phase (or, simply, the setup phase) to distinguish it from the solver phase where the resulting method is applied to the target problem. The setup phase is terminated when either the latest incarnation of the method performs satisfactorily or when a prescribed number of steps is reached.

Each new child method is constructed based on components resulting from its

parent iteration (5.19). The method is modified to reflect the newly computed prototypes as soon as they become available. In other words, the method is kept up to date at all times and no more work is done than necessary.

The smoothed aggregation framework [82] lends itself to the efficient use of a prototype set. It offers fast automatic coarsening with well-understood control over operator complexity due to its typically fixed coarse-operator sparsity pattern. In addition, the process guarantees proper approximation of a given set of functions and their natural localizations during the coarsening process. The resulting coarse-level basis functions are smooth by design and thus suitable for use in a multilevel method. The prototypes obtained by iteration (5.19) play the roles of the near kernel components on which the SA method is based. Thus, in the αSA context, the notion of near kernel components depends not only on the problem, but also on the current method. In general, however, a troublesome component must have a small Rayleigh quotient, signifying ineffectiveness of relaxation.

Note that the convergence result in Theorem 1 (§3.4) hinges on the selection of B^1. In particular, B^1, and the coarse operators, B^{k+1} and P^k_{k+1}, $1 \leq k \leq L-1$, that it induces, must guarantee that the left side of Equation 3.8 is small for any vector $\mathbf{u}$ for which $\langle A^{(1)}\mathbf{u}, \mathbf{u}\rangle$ is small (where $A^{(1)} = A$, the fine-grid matrix). Since the standard SA method requires that matrix B^1 be given as input, with the columns of B^1 representing a basis for (a superset of) the near kernel of $A^{(1)}$, then the construction of P^k_{k+1} as in Equation 3.6 guarantees that all coarse-level representations, B^k, form a basis for (a superset of) the near kernel of $A^{(k)}$, $k > 1$.

The purpose of αSA is to enrich a given, incomplete (possibly even empty) set of near kernel components with approximations computed at runtime in such a way that good convergence can be recovered. The adaptive method can then be viewed as an iterative attempt to satisfy (3.8) heuristically. Our B^1 is computed only approximately, which means that the coarse-level B^k obtained by (3.6) may, alone, not be the optimal

representation of the near kernel. To remedy this, we carry out the setup computation also on the coarse levels to improve on the initial guess for the coarse-level prototypes given by (3.6).

5.4.1 Self-Correcting Adaptive Setup

Before describing the algorithm, we introduce the following notational conventions. The transfer operators and coarse-level problems, as well as other components of our multigrid scheme, change as our method evolves. Whenever possible, we use the same symbols for the updated components. Thus, symbol B^k may denote a single vector in one cycle of the setup procedure, or perhaps a two-column matrix in the next step of the setup. The intended meaning should be clear from context.

5.4.1.1 Initialization setup stage

The adaptive multigrid setup procedure considered in this section can be split into two stages. If no knowledge of the near kernel components of $A^{(1)}$ is available, then we start with the first stage to determine an approximation to one such component. This stage also determines the number of levels, L, to be used in the coarsening process. (Changing L in the next stage based on observed performance is certainly possible, but it is convenient to fix L and other constructs early in the setup phase.) Let $\varepsilon > 0$ be a given convergence tolerance.

Algorithm 4 (Initialization stage).

1. Set $k = 1$, select a random vector, $\mathbf{x}^{(1)} \in \mathbb{R}^{\mathrm{N}^{(1)}}$, and create copy, $\hat{\mathbf{x}}^{(1)} \leftarrow \mathbf{x}^{(1)}$.

2. With initial approximation $\mathbf{x}^{(1)}$, relax ν times on $A^{(1)}\mathbf{x} = \mathbf{0}$:

$$\mathbf{x}^{(1)} \leftarrow (I - R^{(1)}A^{(1)})^{\nu}\mathbf{x}^{(1)}.$$

3. If $\left(\frac{\langle A^{(1)}\mathbf{x}^{(1)},\mathbf{x}^{(1)}\rangle}{\langle A^{(1)}\hat{\mathbf{x}}^{(1)},\hat{\mathbf{x}}^{(1)}\rangle}\right)^{1/\nu} \leq \varepsilon$, then set $L = 1$ and **stop** (problem $A^{(1)}\mathbf{x} = \mathbf{b}^{(1)}$ can be solved fast enough by relaxation alone, so only one level is needed).

4. Otherwise, do the following:

(a) Set $B^k \leftarrow \mathbf{x}^{(k)}$.

(b) Create a set, $\{\mathcal{A}_i^k\}_{i=1}^{N^{(k+1)}}$, of nodal aggregates based on matrix $A^{(k)}$.

(c) Define tentative prolongator P_{k+1}^k and prototype set B^{k+1} using the prototype set B^k and relations (3.6) with structure based on $\{\mathcal{A}_i^k\}_{i=1}^{N^{(k+1)}}$.

(d) Define the prolongator: $I_{k+1}^k = S_k P_{k+1}^k$.

(e) Define the coarse matrix: $A^{(k+1)} = (I_{k+1}^k)^T A^{(k)} I_{k+1}^k$. If level $k+1$ is coarse enough that a direct solver can be used there, skip to Step 5; otherwise, continue.

(f) Set the next-level approximation vector: $\mathbf{x}^{(k+1)} \leftarrow B^{k+1}$.

(g) Make a copy of the current approximation: $\hat{\mathbf{x}}^{(k+1)} \leftarrow \mathbf{x}^{(k+1)}$.

(h) With initial approximation $\mathbf{x}^{(k+1)}$, relax ν times on $A^{(k+1)}\mathbf{x} = \mathbf{0}$:

$$\mathbf{x}^{(k+1)} \leftarrow (I - R^{(k+1)} A^{(k+1)})^{\nu} \mathbf{x}^{(k+1)}.$$

(i) If $\left(\frac{\langle A^{(k+1)}\mathbf{x}^{(k+1)}, \mathbf{x}^{(k+1)} \rangle}{\langle A^{(k+1)}\hat{\mathbf{x}}^{(k+1)}, \hat{\mathbf{x}}^{(k+1)} \rangle} \right)^{1/\nu} \leq \varepsilon$, skip Steps (f-i) in further passes through Step 4.

(j) Increment $k \leftarrow k+1$ and return to Step 4(a).

5. Set $L \leftarrow k+1$ and update the finest-level prototype matrix:

$$B^1 \leftarrow I_2^1 I_3^2 \ldots I_{L-1}^{L-2} \mathbf{x}^{(L-1)}.$$

6. Create the V-cycle based on B^1 using the standard smoothed aggregation procedure.

This initialization stage terminates whenever a level is reached in the coarsening process where a direct solver is appropriate. It does not involve level L processing because it is assumed that the coarsest level is handled by a direct solver, making the stopping criterion in Step 4(i) automatically true. Notice that the prototype set is

actually a single vector in this initial stage because we are computing only one prototype. Note also that this stage provides all of the components needed to construct the initial solver.

If the criterion tested in Step 4(i) is satisfied, we are assured that the current coarse level, $k+1$, can be easily solved by relaxation alone. At that point, we could choose not to coarsen further and use relaxation as a coarsest-level solver. However, it is possible that the general stage of the algorithm described below adds more prototypes. In case a new prototype approximates the low-energy modes of the problem better than the prototype obtained in the initial step, the coarse-level matrix may no longer be easily solved by relaxation alone. Thus, we choose to coarsen further, until we are certain that the coarsest problem can be handled well. This offers an added benefit of producing, at the end of the initial stage, a complete aggregation that can be reused in the general stage. Note that if the condition of 4(i) is satisfied, then the approximate solution of the homogeneous problem may be zero. In such a case, we restore the saved original vector $\hat{\mathbf{x}}^{(k+1)}$. We choose to skip the Steps 4(f-i) in further coarsening once 4(i) is satisfied, amounting to using standard SA coarsening from level $k+1$ down, which guarantees that the prototype computed on level k is exactly represented all the way to the coarsest level. Figure 5.5 illustrates Algorithm 4.

This initialization stage may not be necessary in practice. Consider a problem of 3D linear elasticity discretized by a standard, linear first-order finite element method over an unstructured grid. In this case, if the discretization package either generates the rigid body modes or supplies the nodal geometry to the solver, then the full set of null space vectors are presumably available [81] and the adaptive process may be unnecessary. When the full set of rigid body modes is unavailable, it is nevertheless often possible to obtain a subset of the rigid body modes consisting of the three independent constant displacements, regardless of the geometry of the grid. Such a subspace should be used whenever possible to create B^1 and to set up a V-cycle exactly as in the standard

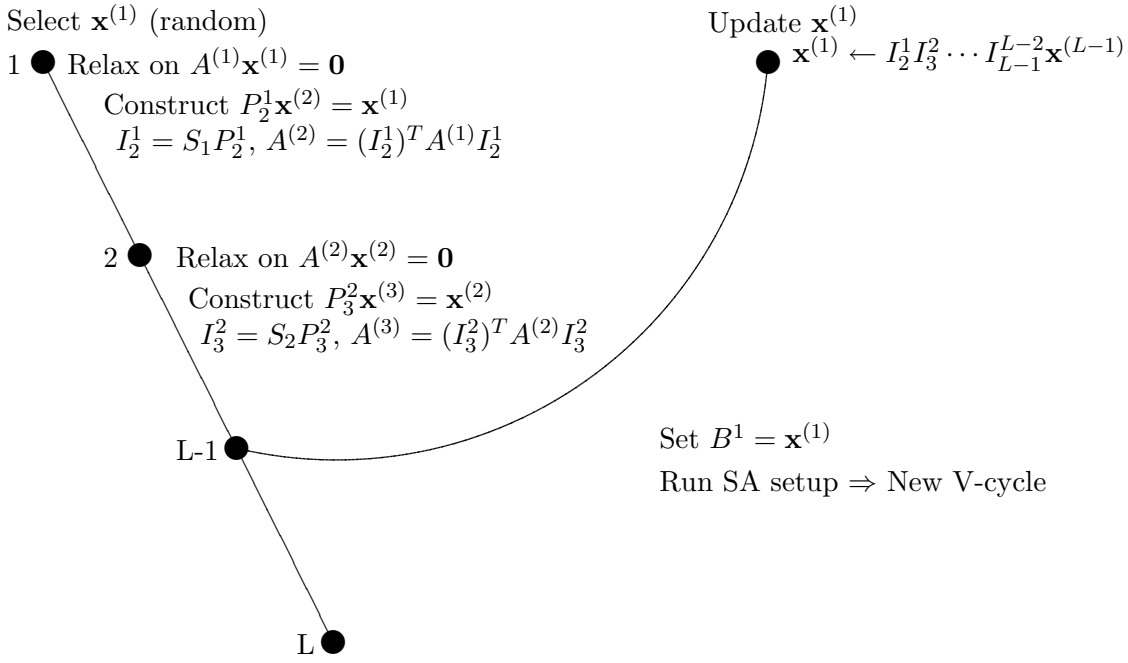

Figure 5.5: Initialization Stage, Algorithm 4

smoothed aggregation method. This initialization stage would then be omitted.

Thus, the initialization stage given by Algorithm 4 should be viewed as optional, to be done only if no information can be assumed about the system to be solved. If any such information is available, the initial V-cycle may be constructed as in standard SA. We can thus, in any case, assume that an initial B^1 having at least one column and a tentative V-cycle are available. This also means that aggregates $\mathcal{A}_i^k$, transfer operators P_{k+1}^k and I_{k+1}^k, and coarse operators $A^{(k+1)}$, $k = 1, \ldots, L-1$ have been constructed.

5.4.1.2 General setup stage

In each step of the second stage of the adaptive procedure, we apply the current V-cycle to the homogeneous problem to uncover error components that are not quickly attenuated. The procedure then updates its transfer operators to ensure that these components will be eliminated by the improved method, while preserving the previously established approximation properties. Thus, this stage essentially follows the initialization stage with relaxation replaced by the current V-cycle.

One of the subtleties of this approach lies in the method's attempt to update each level of the evolving V-cycle as soon as its ineffectiveness is exposed. Thus, on the finest level in the second stage, the current V-cycle simply plays the role of relaxation: if it is unable to quickly solve the homogeneous problem (i.e., Step 3 fails), then the resulting error becomes a new prototype and new degrees of freedom are generated accordingly on level 2 (i.e., columns are added to B^1). The level 2-to-L part of the old V-cycle (i.e., the part without the finest level) then plays the role of the level 2 relaxation in the initial setup phase and is thus applied to the homogeneous problem to assess the need to improve the coarser-level interpolation operators. The same is done on each coarser level, k, with the level k-to-L part of the old V-cycle playing the role of the level k relaxation step in the initial setup phase. The process continues until adequate performance is observed or the maximum permitted number of degrees of freedom per

node is reached on coarse levels.

Consider then a method in which, within each cycle of the adaptive setup, we attempt to update the current V-cycle level by level. One cycle of this adaptive setup traverses from the finest to the coarsest level; on each level, k, along the way, it updates B^k based on computing a new prototype from the current multigrid scheme applied to the homogeneous problem on level k. Thus, on level k in the setup process, a solver is applied that traverses from that level to level L and back. Now, once this new prototype is computed, it is incorporated into the current multigrid scheme and the previously existing V-cycle components are overwritten on level $k+1$, but temporarily retained from that level down. As a result, we redefine level-by-level the V-cycle components. Once the new B^k (and I_{k+1}^k in (3.5)) are constructed all the way to the coarsest level, we can then use them to update the current B^1 and, based on it, construct a new V-cycle on the finest level.

The general stage is therefore analogous to the initialization stage described in Algorithm 4, with relaxation replaced by the evolving V-cycle. Instead of using simple relaxation on each level as the initialization stage does, the general stage uses the current solver to identify new types of error that the earlier sweeps of the setup cycle may have missed, just as the initialization stage was designed to capture a type of error that relaxation cannot efficiently eliminate. This algebraically smooth error is the prototype that is generated by applying relaxation to the homogeneous problem on each level. Similarly, each cycle of the general stage is designed to capture a type of error that the current V-cycle cannot handle, and this, too, must be done on each level. It is, however, not enough for the coarsening process to eliminate only the particular prototype: typically, a fixed percentage of the spectrum of $A^{(1)}$ is algebraically smooth, so elimination of one prototype at a time would take $O(N^{(1)})$ setup cycles. To avoid this unacceptably large cost, each setup cycle must determine interpolation operators so that the solver eliminates a relatively large set of errors similar to each prototype.

Just as each rigid body mode is used locally in standard SA to treat errors of similar type (with constants representing errors that are smooth within variables and rotations representing inter-variable "smoothness"), so too must each prototype be used in αSA. Moreover, a full set of prototypes must be determined if the solver is to attain full efficiency (e.g., for 2D linear elasticity, three rigid body modes are generally needed). We, thus, think of each prototype as a sort of straw man that represents a class of smooth components. Efficient computation of a full set of straw men is the responsibility of the adaptive process. However, proper treatment of each straw man is the task of the basic solver, SA.

As we apply our current method to the homogeneous problem, the resulting prototype, $\mathbf{x}^{(k)}$, becomes rich in the components of the error that are slow to converge in the current method. Our goal in designing the adaptive algorithm is to ensure that $\mathbf{x}^{(k)}$ is approximated relatively well by the newly constructed transfer operator. That is, we want to control the constant, C_a, in the inequality

$$\min_{\mathbf{v}\in\mathbb{R}^{\mathrm{N}(\mathrm{k}+1)}} \|\mathbf{x}^{(k)} - P_{k+1}^{k}\mathbf{v}\|^2 \leq \frac{C_a}{\rho(A^{(k)})}\|\mathbf{x}^{(k)}\|_{A^{(k)}}^2. \tag{5.20}$$

The transfer operators must therefore be constructed to give accurate approximations to each prototype as it is computed. This can be guaranteed locally by requiring that, over every aggregate, $\mathcal{A}$, we have

$$\min_{\mathbf{v}\in\mathbb{R}^{\mathrm{N}(\mathrm{k}+1)}} \|\mathbf{x}^{(k)} - P_{k+1}^{k}\mathbf{v}\|_{\mathcal{A}}^2 \leq C_a\delta_{\mathcal{A}}(\mathbf{x}^{(k)}), \tag{5.21}$$

where $\delta_{\mathcal{A}}$ are chosen so that summing Equation 5.21 over all aggregates leads to Inequality 5.20, i.e., so that

$$\sum_{\mathcal{A}} \delta_{\mathcal{A}}(\mathbf{x}) = \frac{\langle A^{(k)}\mathbf{x}, \mathbf{x}\rangle}{\rho(A^{(k)})}. \tag{5.22}$$

For now, the only assumption we place on $\delta_{\mathcal{A}}(\mathbf{x})$ is that (5.22) holds. An appropriate choice for its definition is given in Section 5.4.2.

To relate the condition in (5.20) to the theoretical foundation of smoothed aggregation, we make the following observation. If P_{k+1}^{k} is constructed so that (5.20) is

satisfied for the prototype $\mathbf{x}^{(k)}$, the construction of our method automatically guarantees that

$$\min_{\mathbf{v}\in\mathbb{R}^{\mathrm{N}(\mathrm{k}+1)}} \|\mathbf{x}^{(1)} - P_2^1 P_3^2 \dots P_{k+1}^k \mathbf{v}\|^2 \leq \frac{C_a}{\rho(A^{(1)})} \|\hat{\mathbf{x}}^{(1)}\|^2_{A^{(1)}},$$

where $\mathbf{x}^{(1)} = P_2^1 P_3^2 \dots P_k^{k-1} \mathbf{x}^{(k)}$ and $\hat{\mathbf{x}}^{(1)} = I_2^1 I_3^2 \dots I_k^{k-1} \mathbf{x}^{(k)}$. Since it is easy to show that $\|\hat{\mathbf{x}}\|_{A^{(1)}} \leq \|\mathbf{x}\|_{A^{(1)}}$, we can thus guarantee that (3.8) holds for the particular fine-level prototype $\mathbf{x}^{(1)}$. Inequality 5.20 is easily satisfied for any component, $\mathbf{u}$, for which $\|\mathbf{u}\|_{A^{(1)}}$ is bounded away from zero, and so we focus on the troublesome subspace of components with small energy. Our experience with the standard SA method indicates that, for second- and fourth-order elliptic problems, it suffices to ensure that the components corresponding to the null space of the weak form of the problem are well approximated by the prolongation (the near null space components are then well approximated due to the localization and smoothing procedures involved in constructing the smoothed aggregation transfer operators). Further, as the set of prototypes constructed during the setup cycle is expected to eventually encompass the entire troublesome subspace, satisfaction of (5.20) for all prototypes would imply the satisfaction of (3.8) for any $\mathbf{u} \in \mathbb{R}^{\mathrm{N}^{(1)}}$. This, in turn, guarantees convergence.

Three practical matters must be addressed when coarsening in the adaptive stage:

Locally small components: Each new prototype is the result of applying the V-cycle based on the current prototype set, B^1, so it must be approximately $A^{(1)}$-orthogonal to all previously computed prototypes. This is, however, only a global property that the evolving prototypes tend to exhibit. It may be that a prototype is so small on some aggregate, relative to its energy, that its representation there can be ignored. More precisely, we could encounter situations in which

$$\|\mathbf{x}^{(k)}\|^2_{\mathcal{A}} \leq C_a \delta_{\mathcal{A}}(\mathbf{x}^{(k)}) \tag{5.23}$$

for a particular aggregate, $\mathcal{A}$, meaning that Equation 5.21 is automatically satisfied, no matter what choice we make for P_{k+1}^k. We can, therefore, test for this condition for

each prototype on every aggregate. When the test is positive, we can simply remove the prototype's segment from consideration in construction of that aggregate's transfer operator. This elimination can help control coarse-level complexity since small prototype segments are prevented from generating additional columns of P_{k+1}^k and I_{k+1}^k. (This test should be used in the initialization as well as the general setup stage. We did not include it there for simplicity and because there is generally less worry about complexity in the initial stage.)

Construction of P_{k+1}^k to minimize the number of columns: Although each prototype's segments may be too large to ignore, it may be that a nontrivial linear combination of them is small. Thus, we also need to use (5.23) to identify a minimal local basis for constructing the transfer operators so that the global approximation property is maintained. To this end, let a subscript $\mathcal{A}$ denote the restriction of the corresponding vector or matrix to the degrees of freedom in $\mathcal{A}$, and let $r_{\mathcal{A}}$ denote the number of columns of $B_{\mathcal{A}}^k$. One possibility for constructing the updated transfer operator, P_{k+1}^k, aggregate by aggregate, would then proceed as follows:

- Rescale each column, $\mathbf{y}$, of B^k globally: $\mathbf{y} \leftarrow \frac{\mathbf{y}}{\sqrt{\langle A^{(k)}\mathbf{y},\mathbf{y}\rangle}}$.
- Reorder the newly scaled columns of $B_{\mathcal{A}}^k$ so that their Euclidean norms over aggregate $\mathcal{A}$ are nonincreasing: $\|\mathbf{y}_1\|_{\mathcal{A}} \geq \|\mathbf{y}_2\|_{\mathcal{A}} \geq \ldots \geq \|\mathbf{y}_{r_{\mathcal{A}}}\|_{\mathcal{A}}$.
- Set $j = 1$.
- While $j \leq r_{\mathcal{A}}$:
  - Set $\gamma_{\mathcal{A}} = \frac{C_a \delta_{\mathcal{A}}(\mathbf{y}_j)}{\langle A^{(k)}\mathbf{y}_j,\mathbf{y}_j\rangle}$.
  - If $\|\mathbf{y}_j\|_{\mathcal{A}}^2 \leq \gamma_{\mathcal{A}}$, then stop. Otherwise, add $\frac{\mathbf{y}_j}{\|\mathbf{y}_j\|_{\mathcal{A}}}$ as a new column of P_{k+1}^k, make all remaining columns in $B_{\mathcal{A}}^k$ orthogonal to $\mathbf{y}_j$, and reorder them so that their Euclidean norms over $\mathcal{A}$ are nonincreasing.
  - $j \leftarrow j + 1$.

A disadvantage of this process is that P_{k+1}^k (hence, also I_{k+1}^k) must, in principle, be constructed from scratch in each cycle of the adaptive setup. We discuss other practical issues associated with this approach in the next section.

Reusing previously constructed components: To exploit the work done in the earlier steps of the setup as much as possible, we consider an alternate procedure that reuses parts of P_{k+1}^k that have already been computed. Thus, in each step of the setup, we only consider adding a single new column to P_{k+1}^k. This has the advantages that less work is required and that the storage used to hold the global prototypes can be reused as soon as they have been incorporated into P_{k+1}^k.

In this approach, to minimize the complexity of the transfer operators, we seek to ignore locally those components of prototype $\mathbf{x}^{(k)}$ that appear to be well approximated by the current transfer operators. This includes the case when $\mathbf{x}^{(k)}$ is locally small in the sense of (5.23). To decide whether to ignore $\mathbf{x}^{(k)}$ locally in the construction of new tentative prolongator, P_{k+1}^k, we test how well it is approximated by the current tentative prolongator, $\tilde{P}_{k+1}^k$. The following provides a test of how well the range of $\tilde{P}_{k+1}^k$ approximates $\mathbf{x}^{(k)}$ over aggregate $\mathcal{A}$:

$$\|\mathbf{x}^{(k)} - \tilde{P}_{k+1}^k (\tilde{P}_{k+1}^k)^T \mathbf{x}^{(k)}\|_{\mathcal{A}}^2 \leq C_a \delta_{\mathcal{A}}(\mathbf{x}^{(k)}). \tag{5.24}$$

Since $(\tilde{P}_{k+1}^k)^T \tilde{P}_{k+1}^k = I$, then $\tilde{P}_{k+1}^k (\tilde{P}_{k+1}^k)^T$ is the ℓ_2-projection onto the range of $\tilde{P}_{k+1}^k$; thus, (5.24) is just the approximation property in (5.21), using the tentative prolongator in place of the smoothed one.

If Equation 5.24 is satisfied, then $\mathbf{x}^{(k)}$ is assumed to be well approximated by the current transfer operator and is simply ignored in the construction of the new transfer operator on aggregate $\mathcal{A}$. (Practical implications of this local elimination from the coarsening process are considered in Section 5.4.2.) If the inequality is not satisfied, then we keep the computed vector $\mathbf{y} = \mathbf{x}^{(k)} - \tilde{P}_{k+1}^k (\tilde{P}_{k+1}^k)^T \mathbf{x}^{(k)}$, that, by construction, is orthogonal to all the vectors already represented in the current $\tilde{P}_{k+1}^k$. We then normalize

via $\mathbf{y} \leftarrow \mathbf{y}/\|\mathbf{y}\|_{\mathcal{A}}$ so that the new P_{k+1}^k has orthonormal columns: $(P_{k+1}^k)^T P_{k+1}^k = I$.

Before we introduce Algorithm 5 below, we stress that the description should be viewed as a general outline of the adaptive multigrid setup. We intentionally ignore several practical issues that must be addressed before this algorithm can be implemented. For instance, we do not include details on how the new B^k and I_{k+1}^k are efficiently constructed in the evolving method. Also, when using a coarse-level V-cycle constructed by previous applications of the setup stage, we must account for the possibility that the number of vectors approximated on coarse levels in previous cycles is smaller than the number of vectors approximated on the fine levels in the current cycle. These issues are discussed in Section 5.4.2, where we take advantage of the smoothed aggregation framework to turn the prototypical Algorithm 5 into a practical implementation.

Assume we are given a bound, $R \in \mathbb{N}$, on the number of degrees of freedom per node on coarse levels, convergence factor tolerance, $\varepsilon \in (0,1)$, and aggregate quantities, $\delta_{\mathcal{A}}(x)$, such that $\sum_{\mathcal{A}} \delta_{\mathcal{A}}(x) = \frac{\langle A^{(k)}\mathbf{x},\mathbf{x}\rangle}{\rho(A^{(k)})}$.

Algorithm 5 (One cycle of the general setup stage).

1. If the maximum number of degrees of freedom per node on level 2 equals R, **stop** (the allowed number of coarse-grid degrees of freedom has been reached).

2. Create a copy of the current B^1 for later use: $\hat{B}^1 \leftarrow B^1$.

3. Select a random $\mathbf{x}^{(1)} \in \mathbb{R}^{\mathrm{N}^{(1)}}$, create a copy $\hat{\mathbf{x}}^{(1)} \leftarrow \mathbf{x}^{(1)}$, and apply ν iterations of the current finest-level V-cycle:

$$\mathbf{x}^{(1)} \leftarrow \mathrm{MG}_1^{\nu}(\mathbf{x}^{(1)}, \mathbf{0}).$$

4. If $\left(\frac{\langle A^{(1)}\mathbf{x}^{(1)},\mathbf{x}^{(1)}\rangle}{\langle A^{(1)}\hat{\mathbf{x}}^{(1)},\hat{\mathbf{x}}^{(1)}\rangle}\right)^{1/\nu} \leq \varepsilon$, then **stop** ($A^{(1)}\mathbf{x} = \mathbf{b}^{(1)}$ can be solved fast enough by the current method).

5. Update B^1 by incorporating the computed $\mathbf{x}^{(1)}$ in its range:

$$B^1 \leftarrow [B^1, \mathbf{x}^{(1)}].$$

6. For $k = 1, \ldots, L-2$:

(a) Create a copy of the current B^{k+1} for later use: $\hat{B}^{k+1} \leftarrow B^{k+1}$,

(b) Define new coarse-level matrix B^{k+1} and transfer operators P_{k+1}^k, I_{k+1}^k using (3.6) and (3.5). In creating P_{k+1}^k, some local components in B^k may be eliminated as suggested above.

(c) Construct coarse operator $A^{(k+1)} = (I_{k+1}^k)^T A^{(k)} I_{k+1}^k$.

(d) Set the coarse-level approximation, $\mathbf{x}^{(k+1)}$, to be the last column of B^{k+1}.

(e) Make a copy: $\hat{\mathbf{x}}^{(k+1)} \leftarrow \mathbf{x}^{(k+1)}$.

(f) Apply ν iterations of the current level $k+1$ V-cycle:

$$\mathbf{x}^{(k+1)} \leftarrow MG_{k+1}^{\nu}(\mathbf{x}^{(k+1)}, \mathbf{0}).$$

(g) If $\left(\frac{\langle A^{(k+1)}\mathbf{x}^{(k+1)}, \mathbf{x}^{(k+1)}\rangle}{\langle A^{(k+1)}\hat{\mathbf{x}}^{(k+1)}, \hat{\mathbf{x}}^{(k+1)}\rangle}\right)^{1/\nu} \leq \varepsilon$, then skip Steps (d) through (h) in further passes through Step 6.

(h) Update B^{k+1} by ensuring that the newly computed $\mathbf{x}^{(k+1)}$ is in its range:

$$B^{k+1} \leftarrow [\hat{B}^{k+1}, \mathbf{x}^{(k+1)}].$$

7. Update the finest-level prototype:

$$\mathbf{x}^{(1)} \leftarrow I_2^1 I_3^2 \ldots I_{L-1}^{L-2} \mathbf{x}^{(L-1)}. \tag{5.25}$$

8. Update B^1 by adding the newly computed $\mathbf{x}^{(1)}$ to the range of $\hat{B}^1$:

$$B^1 \leftarrow [\hat{B}^1, \mathbf{x}^{(1)}].$$

9. Create a new V-cycle, MG, based on B^1 using the standard smoothed aggregation setup procedure.

This algorithm, illustrated in Figure 5.6, starts from a V-cycle on input and produces an improved V-cycle as output. It stops iterating when either the convergence factor for the fine-level iteration in Step 3 is acceptable (as measured in Step 4) or when the maximum number of iterations is reached. Note that, as with the initial stage, this general stage does not involve level L processing because the coarsest level is assumed to be treated by a direct solver. Also as in the initial stage, once a level is reached where the problem can be solved well by the current method, any further coarsening is constructed as in the standard SA.

5.4.2 Implementation Issues

Several issues must be addressed to make Algorithm 5 practical. We take advantage of certain features of the smoothed aggregation concept to carry out the method outlined above, as well as to control the amount of work required to keep the evolving multigrid hierarchy up to date.

As suggested above, a prototype may occasionally be eliminated locally over an aggregate. This results in varying numbers of degrees of freedom per node on the coarse levels. (Recall that a coarse-level node is defined as a set of degrees of freedom, with each DOF representing the restriction of a single prototype to a fine-level aggregate.) To simplify notation, we assume for the time being that the number of degrees of freedom per node is the same for all nodes on a given level (i.e., no prototypes are locally eliminated). It is important, however, to keep in mind that we are interested in the more general case. A generalization to varying numbers of degrees of freedom per node could be obtained easily at the cost of a much more cumbersome notation. We briefly remark on the more general cases below.

A practical implementation of the method requires strict attention to many details affecting accuracy and cost. These details are discussed here individually, starting with complexities introduced in trying to eliminate work in the setup phase.

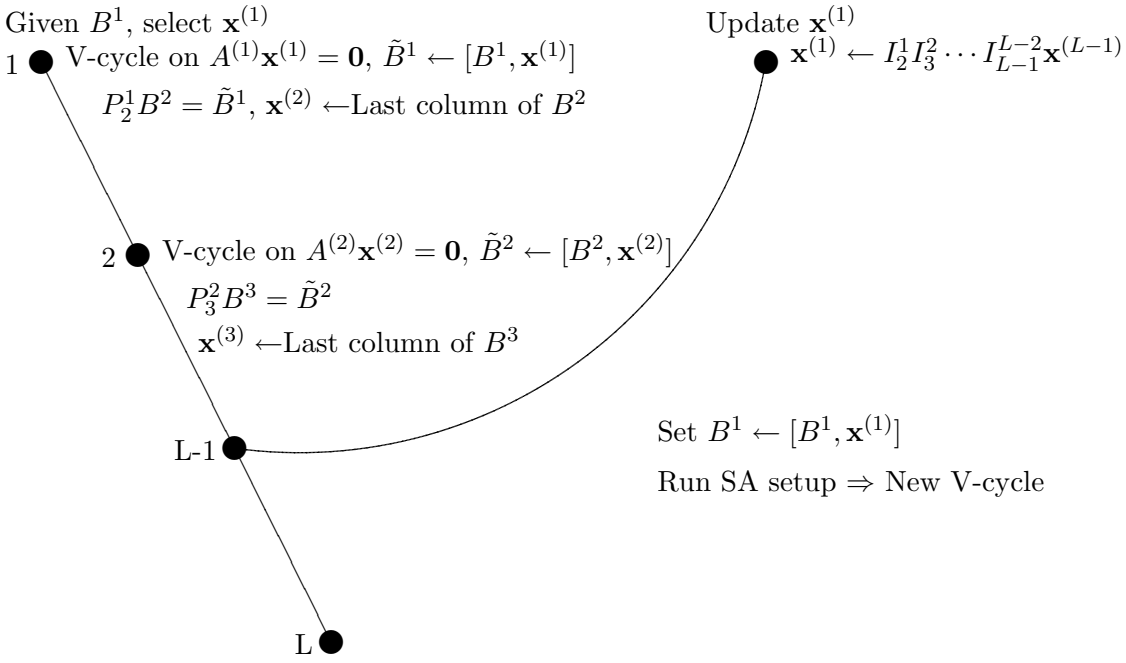

Figure 5.6: One Step of General Setup Stage, Algorithm 5.

Construction of temporary "bridging" transfer operators: One issue we must consider is the interfacing between the emerging V-cycle on finer levels and the previous V-cycle on coarser levels. Each setup cycle starts by selecting an initial approximation for a new prototype on the finest level (cf. Figure 5.6). This approximation is then improved by applying the previously constructed V-cycle to it. The resulting prototype is used to enrich B^1, necessitating an update of P_2^1, I_2^1, and $A^{(2)}$ from (3.6) and (3.4), and introducing an additional degree of freedom for the nodes on level 2. Since we now want to run the current solver on level 2 to obtain an improved prototype on that level, we need to temporarily modify P_3^2 and I_3^2 because these transfer operators have not yet been updated to reflect the added degrees of freedom on level 2. Once this modification has been made, a V-cycle on level 2 can be run to compute the new prototype there. This prototype is then incorporated into B^2 and new P_3^2 and I_3^2 are constructed, overwriting the temporary versions, and the new $A^{(3)}$ can be computed using (3.4). To perform the V-cycle on level 3, we then must temporarily modify operators P_4^3 and I_4^3 for the same reason we had to update P_3^2 and I_3^2 above. Analogous temporary modifications to the transfer operators are necessary on all coarser levels, as the setup cycle traverses sequentially through them.

Thus, on level k of a single cycle of the setup process, all transfer operators defining the V-cycle can be used without change, except for P_{k+1}^k and, consequently, I_{k+1}^k defined through (3.5). We can construct the temporary operator, P_{k+1}^k, by modifying (3.6) as

$$P_{k+1}^k B^{k+1} = \hat{B}^k,$$

where $\hat{B}^k$ is formed by removing the last column from B^k, which consists of the $r+1$ fine-level prototype vectors, including the newly added one (so that the first r prototypes are the same as in the previous cycle).

Since the tentative prolongator, P_{k+1}^k, produced in this way is based only on fitting

the first r vectors in B^k, the coarse-level matrix $A^{(k+1)}$ resulting from the previous cycle of the αSA setup can be used on the next level. Thus, all the coarse operators for levels coarser than k can be used without change. This has the advantage of reducing the amount of work to keep the V-cycle up to date on coarser, yet-to-be-traversed levels.

Eliminating prototypes locally: When we eliminate a prototype locally over an aggregate as suggested above, the construction of the bridging operator can be easily modified so that the multigrid hierarchy constructed in the previous setup cycle can be used to apply a level k V-cycle in the current one. Performing the elimination so that the previously selected prototypes are retained and only the newly computed prototype may be locally eliminated, the V-cycle constructed in the previous setup cycle remains valid on coarser grids. The only difference now is that aggregates may have a variable number of associated prototypes, and the construction of the temporary transfer operator, P_{k+1}^k, must account for this when removing the column of B^k to construct $\hat{B^k}$.

The situation is slightly more complicated when eliminating prototypes by complete orthogonalization over an aggregate. First, even if none of the old prototypes are eliminated, the elimination may result in a permutation of the prototypes over an aggregate, hence a permutation of the coarse degrees of freedom corresponding to the associated node. To match the fine-level V-cycle with the existing coarser levels, an appropriate permutation of the coarse degrees of freedom must then be done when performing the intergrid transfer in the application of the resulting V-cycle.

However, if some of the previously selected prototypes are eliminated in favor of the new prototype in the construction of the updated P_{k+1}^k, the coarse V-cycle should no longer be used without change. In such cases, we would have to generate all the coarse levels below level k before running the level $k+1$ V-cycle, significantly increasing the cost of the setup phase.

Selection of the local quantities $\delta_{\mathcal{A}}(\mathbf{x})$: Our algorithm relies on local aggregate quantities, $\delta_{\mathcal{A}}(\mathbf{x})$, to decide whether to eliminate prototype $\mathbf{x}$ in aggregate $\mathcal{A}$, and

to guarantee that the computed prototypes satisfy the global approximation property (5.20). This leads us to the choice

$$\delta_{\mathcal{A}}(\mathbf{x}) = \left(\frac{\text{card}(\mathcal{A})}{N^{(k)}}\right) \frac{\langle A^{(k)}\mathbf{x}, \mathbf{x}\rangle}{\rho(A^{(k)})}, \tag{5.26}$$

where $\text{card}(\mathcal{A})$ denotes the number of nodes in aggregate $\mathcal{A}$ on level k, and $N^{(k)}$ is the total number of nodes on that level. Note that $\sum_{\mathcal{A}} \delta_{\mathcal{A}}(\mathbf{x}) = \frac{\langle A^{(k)}\mathbf{x},\mathbf{x}\rangle}{\rho(A^{(k)})}$ for any $\mathbf{x}$, so this can be used in the local estimates in Equation 5.21 to guarantee Inequality 5.20.

Addressing these concerns, we may restate the αSA setup procedure (Algorithm 5), given the bound, R, and tolerance, ε, as

Algorithm 6 (One cycle of αSA).

1. If the maximum number of degrees of freedom per node on level 2 equals R, **stop** (the allowed number of coarse grid degrees of freedom has been reached).

2. Create a copy of the current B^1 for later use: $\hat{B}^1 \leftarrow B^1$.

3. Select a random $\mathbf{x}^{(1)} \in \mathbb{R}^{\mathrm{N}^{(1)}}$, create a copy $\hat{\mathbf{x}}^{(1)} \leftarrow \mathbf{x}^{(1)}$, and apply ν iterations of the current V-cycle:

$$\mathbf{x}^{(1)} \leftarrow \mathrm{MG}_1^{\nu}(\mathbf{x}^{(1)}, \mathbf{0}).$$

4. If $\left(\frac{\langle A^{(1)}\mathbf{x}^{(1)},\mathbf{x}^{(1)}\rangle}{\langle A^{(1)}\hat{\mathbf{x}}^{(1)},\hat{\mathbf{x}}^{(1)}\rangle}\right)^{1/\nu} \leq \varepsilon$, then **stop** ($A^{(1)}\mathbf{x} = \mathbf{b}^{(1)}$ can be solved fast enough by the current method).

5. Update B^1 by extending its range with the new column $\{\mathbf{x}^{(1)}\}$:

$$B^1 \leftarrow [B^1, \mathbf{x}^{(1)}].$$

6. For $k = 1, \ldots, L-2$:

(a) Define a new coarse-level matrix, B^{k+1}, and transfer operator, P_{k+1}^k, based on (3.6), using B^k and decomposition $\{\mathcal{A}_i^k\}_{i=1}^{N^{(k+1)}}$. In creating P_{k+1}^k, some local components in B^k may be locally eliminated.

(b) Construct the prolongator: $I_{k+1}^k = S_k P_{k+1}^k$.

(c) Construct the coarse operator: $A^{(k+1)} = (I_{k+1}^k)^T A^{(k)} I_{k+1}^k$.

(d) Reorder the columns of B^{k+1} so that its last is $\mathbf{x}^{(k+1)}$ and let $\hat{B}^{k+1}$ consist of all other columns of B^{k+1}.

(e) Create a "bridge" transfer operator P_{k+2}^{k+1} to the coarser level with the old B^{k+1} by fitting all the vectors in B^{k+1} except the last one.

(f) Set the new "bridging" prolongator: $I_{k+2}^{k+1} = S_{k+1} P_{k+2}^{k+1}$.

(g) Make a copy: $\hat{\mathbf{x}}^{(k+1)} \leftarrow \mathbf{x}^{(k+1)}$.

(h) Apply ν iterations: $\mathbf{x}^{(k+1)} \leftarrow \mathrm{MG}_{k+1}^{\nu}(\mathbf{x}^{(k+1)}, \mathbf{0})$.

(i) If $\left(\frac{\langle A^{(k+1)} \mathbf{x}^{(k+1)}, \mathbf{x}^{(k+1)} \rangle}{\langle A^{(k+1)} \hat{\mathbf{x}}^{(k+1)}, \hat{\mathbf{x}}^{(k+1)} \rangle} \right)^{1/\nu} \leq \varepsilon$, then skip (d) through (j) in further passes through Step 6.

(j) Update the coarse representation of the prototypes B^{k+1}:

$$B^{k+1} \leftarrow [\hat{B}^{k+1}, \mathbf{x}^{(k+1)}].$$

7. Update the latest fine-level prototype:

$$\mathbf{x}^{(1)} \leftarrow I_2^1 I_3^2 \ldots I_{L-1}^{L-2} \mathbf{x}^{(L-1)}. \tag{5.27}$$

8. Update B^1 by extending the old copy with the newly computed $\mathbf{x}^{(1)}$:

$$B^1 \leftarrow [\hat{B}^1, \mathbf{x}^{(1)}].$$

9. Create the V-cycle based on the current B^1 using the standard smoothed aggregation setup procedure.

Note that if we eliminate prototypes by a complete local orthogonalization step, we should modify the algorithm to construct a completely new multigrid hierarchy on levels $k+1$ through L before applying the level $k+1$ V-cycle in Step 6(h).

In general, we seek to eliminate all unneeded computation from the setup algorithm. Two important considerations are:

Improving the quality of existing prototypes: Many practical situations, including fourth-order equations and systems of fluid and solid mechanics, require a set of multiple prototypes to achieve optimal convergence. In the interest of keeping operator complexity as small as possible, it is imperative that the number of prototypes used to produce the final method be controlled. Therefore, ways of improving the quality of each prototype are of interest, to curb the demand for the growth in their number.

When the current V-cycle hierarchy is based on approximating at least two prototypes (in other words, the coarse problems feature at least two degrees of freedom per node), this can be easily accomplished as follows.

Assume that the currently available prototype vectors are $\mathbf{x}_1, \ldots, \mathbf{x}_r$. Consider one such prototype, say, $\mathbf{x}_j$, that we want to improve. We want to run a modified but current V-cycle on the homogeneous problem, $A^{(1)}\mathbf{x} = \mathbf{0}$, using $\mathbf{x}_j$ as the initial guess. The modification consists of disabling, in the coarse-grid correction process, the columns of the prolongator corresponding to the given prototype. That is, instead of $\mathbf{x}^{(k)} \leftarrow \mathbf{x}^{(k)} + I_{k+1}^k \mathbf{x}^{(k+1)}$ in Step 4 of Algorithm 1, we use

$$\mathbf{x}^{(k)} \leftarrow \mathbf{x}^{(k)} + I_{k+1}^k \hat{\mathbf{x}}^{(k+1)},$$

where $\hat{\mathbf{x}}^{(k+1)}$ is obtained from $\mathbf{x}^{(k+1)}$ by setting to zero every entry corresponding to fine-level prototype $\mathbf{x}_j$. Thus, the columns of I_{k+1}^k corresponding to $\mathbf{x}_j$ are not used in coarse-grid correction.

In this way, we come up with an improved prototype vector without restarting the entire setup iteration from scratch and without adding a new prototype. Since we focus on one component at a time and keep all other components intact, this modified V-cycle is expected to converge rapidly.

Saving work: The reuse of current coarse-level components reduces the amount of work

required to keep the V-cycle up to date. Additional work can be saved by performing the decomposition of nodes into disjoint aggregates only during the setup of the initial V-cycle and then reusing this decomposition in later cycles. Yet further savings are possible in coarsening, assuming the prototypes are allowed to be locally eliminated. For instance, we can exploit the second-level matrix structure:

$$A^{(2)} = \begin{bmatrix} \tilde{A}^{(2)} & X \\ Y & Z \end{bmatrix},$$

where $\tilde{A}^{(2)}$ is the second-level matrix from the previous cycle. Thus, $A^{(2)}$ need not be recomputed and can be obtained by a rank-one update of each block entry in $\tilde{A}^{(2)}$. In a similar fashion, the new operators, P_{k+1}^k and B^{k+1}, do not have to be recomputed in each new setup cycle by the local QR decomposition noted in Section 3.4. Instead, it is possible to update each nodal entry in $\tilde{P}_{k+1}^k$ and $\hat{B}^{k+1}$ by a rank-one update on all coarse levels, where $\tilde{P}_{k+1}^k$ and $\hat{B}^{k+1}$ are the operators created by the previous setup cycle.

5.4.3 Numerical Experiments

To demonstrate the effectiveness of the adaptive setup process, we present results obtained by applying the method to several model problems. In these tests, the solver was stopped when the residual was reduced by a relative factor of 10^{12} (unless otherwise specified). The value $C_a = 10^{-3}$ was used in (5.24) and the relaxation scheme for the multigrid solver was symmetric Gauss-Seidel. While a Krylov subspace process is used often in practice, we present these results for the multigrid V-cycle with no acceleration scheme for clarity, unless explicitly specified otherwise.

All the experiments have been run on a notebook computer with a 1.6 GHz mobile Pentium 4 processor and 512 MB of RAM. For each experiment, we report the following data. The column denoted by "Iter" contains the number of iterations required to reduce the residual by the prescribed factor. The "Factor" column reports

the convergence factor measured as the geometric average of the residual reduction in the last 10 iterations. In the "CPU" column, we report the total CPU time in seconds required to complete both the setup and iteration phases of the solver. In the column "RelCPU", we report the relative times to solution, with one unit defined as the time required to solve the problem given the correct near null space components. In the "OpComp" column, we report the operator complexity associated with the V-cycle for every run (defining operator complexity in the usual sense [73], as the ratio of the number of entries stored in all matrices on all levels divided by the number of entries stored in the finest-level matrix.) The "Prototypes" column indicates the number of near-kernel vectors computed in the setup iteration (a value of "provided" means that complete kernel information was supplied to the solver, assuming standard discretization and ignoring scaling). Parameter $\mu_{\max}$ denotes the maximal number of tentative V-cycles allowed in computing each prototype.

In all cases, we consider both the problem as discretized and when that discretization is modified by either scaling or rotating each nodal entry in the system by a random angle. These modifications pose serious difficulties for classical algebraic iterative solvers that are not aware of such modifications, as we assume here. For comparison, we include the results for the unmodified problem, with a supplied set of kernel components. Not surprisingly, the standard algorithm (without benefit of the adaptive process) performs poorly for the scaled or rotated systems when the details of this modification are kept from the solver. In all of the results that follow, whenever the solver is provided information about the problem, that information is the appropriate near kernel components for the unmodified discretization.

We start by considering a diagonally scaled problem:

$$A \leftarrow D^{-1/2} A D^{-1/2},$$

where the original matrix, A, is the matrix obtained by standard Q1 finite element

σ	Prototypes	$\mu_{\max}$	Iter	Factor	CPU	RelCPU	OpComp
Poisson problem with 68,921 degrees of freedom							
0	provided	N/A	9	0.100	3.65	1.00	1.038
0	1	5	9	0.100	4.09	1.12	1.038
6	provided	N/A	150	0.871	43.76	11.99	1.038
6	1	5	10	0.126	4.27	1.17	1.038
Poisson problem with 1,030,301 degrees of freedom							
0	provided	N/A	9	0.093	58.43	1.00	1.039
0	1	5	9	0.099	80.05	1.37	1.039
6	provided	N/A	690	0.970	3,252.80	55.67	1.039
6	1	5	9	0.096	88.23	1.51	1.039

Table 5.13: 3D Poisson Problems, 68,921 and 1,030,301 Degrees of Freedom; Reducing Residual by 10^8.

discretization of the 3D Poisson operator on a cube, and D is a diagonal matrix with entries 10^{β}, where $\beta \in [-\sigma, +\sigma]$ is chosen randomly. Table 5.13 shows the results for different values of parameter σ and different levels of refinement. Using the supplied kernel yields good convergence factors for the unmodified problem, but the performance is poor and deteriorates with increased problem size when used with $\sigma \neq 0$. In contrast, the adaptive process, starting from a random approximation, recovers the convergence properties associated with the standard Poisson problem ($\sigma = 0$), even for the scaled case, with convergence that appears independent of the problem size.

The second problem comes from a matrix arising in 2D elasticity. Diagonal scaling is again considered, with entries of D defined as 10^{β}, for $\beta \in [-\sigma, +\sigma]$ chosen randomly. The original matrix is the discrete operator for the plane-strain elasticity formulation over a square domain using bilinear finite elements on a uniform grid, with a Poisson ratio of $\nu = 0.3$ and Dirichlet boundary conditions specified only along the "West" side of the domain. The results in Table 5.14 follow a pattern similar to those for the Poisson problem. Note, however, that more than the usual three prototype vectors are now needed to achieve convergence properties similar to those of the unscaled problem when a correct set of three rigid-body modes is provided by the user. For the scaled

problem, however, supplying the rigid-body modes computed based on the problem geometry leads, as expected, to dismal performance of the standard solver.

The third set of experiments is based again on the 2D elasticity problem, but now each nodal block is rotated by a random angle $\beta \in [0, \pi]$:

$$A \leftarrow Q^T A Q,$$

where Q is a nodal block-diagonal matrix consisting of rotations with random angles. The results in Table 5.15 show that αSA can recover good convergence factors for both the unmodified and the modified systems. Without the adaptive procedure, our basic algebraic solver could not solve the modified matrix problem in a reasonable amount of time.

The fourth example demonstrates performance of the method when a higher number of prototypes is required. We consider a 3D elasticity problem with local rotations. This is done to maintain locally orthogonal coordinates, but is otherwise a random rotation of the three degrees of freedom at each node. The model problem we start from is linearized elasticity discretized using trilinear finite elements over a uniform grid. Dirichlet boundary conditions are specified on the "West" face of the cube and the Poisson ratio is set to $\nu = 0.3$. The results in Table 5.16 show that, even for the modified system, the adaptive method can again recover good convergence factors. Furthermore, our current method mimics the convergence of the smoothed aggregation for the unmodified problem with the supplied set of rigid-body modes. In this set of experiments, we can get close to the ideal iteration counts using just 6 prototypes. We see that using one extra prototype can improve convergence properties and, in some cases, actually lower the overall cost of the total time to solution. This is done at the price of a small increase in operator complexity. For problems with multiple right sides, the more expensive setup would be performed only once, and using the extra prototype may then be preferred. Once again, however, we see that the additional cost of the setup stage

σ	Prototypes	$\mu_{\max}$	Iter	Factor	CPU	RelCPU	OpComp
2D elasticity problem, 80,400 degrees of freedom							
0	3 provided	N/A	17	0.21	9.16	1.00	1.27
0	3	6	23	0.37	21.16	2.31	1.27
0	3	15	18	0.23	26.65	2.91	1.27
6	3 provided	N/A	299	0.92	133.55	14.58	1.27
6	3	6	25	0.38	22.26	2.43	1.27
6	3	15	18	0.25	27.30	2.98	1.27
2D elasticity problem, 181,202 degrees of freedom							
0	3 provided	N/A	23	0.35	22.85	1.00	1.28
0	3	15	267	0.937	272.14	11.91	1.27
0	4	15	26	0.422	75.18	3.29	1.50
0	4	20	26	0.439	86.60	3.79	1.50
0	5	15	20	0.314	88.20	3.86	1.78
6	3 provided	N/A	5,000*	0.996	4,559.95	199.56	1.28
6	4	15	23	0.367	74.95	3.28	1.50
6	4	20	19	0.302	76.78	3.36	1.50
6	5	10	14	0.173	69.46	3.04	1.78

Table 5.14: 2D Elasticity Problems with 80,400 and 181,202 Degrees of Freedom. Iteration Counts Marked with an Asterisk Indicate that Residual Reduction by 10^{12} was not Achieved Before the Maximum Number of Iterations was Reached.

Rotated	Prototypes	$\mu_{\max}$	Iter	Factor	CPU	RelCPU	OpComp
2D elasticity problem with 80,400 degrees of freedom							
NO	3 provided	N/A	17	0.21	9.16 s	1.00	1.27
NO	3	15	18	0.23	26.66	2.91	1.27
YES	3 provided	N/A	1,329	0.99	587.80	64.17	1.27
YES	3	15	19	0.27	27.84	3.04	1.27
2D elasticity problem with 181,202 degrees of freedom							
NO	3 provided	N/A	23	0.35	22.85 s	1.00	1.28
NO	3	15	18	0.23	66.49	2.91	1.28
YES	3 provided	N/A	5,000*	0.999	3,968.36	173.67	1.28
YES	3	20	135	0.885	170.23	7.45	1.28
YES	4	15	27	0.488	77.46	3.39	1.50
YES	4	20	21	0.395	79.29	3.47	1.50
YES	5	6	18	0.34	60.78	2.66	1.78
YES	5	10	15	0.233	72.66	3.18	1.78

Table 5.15: 2D Elasticity Problems with 80,400 and 181,202 Degrees of Freedom. Iteration Counts Marked with an Asterisk Indicate that Residual Reduction by 10^{12} was not Achieved Before the Limit on the Number of Iterations was Reached.

should not be undertaken unless it is necessary. For the problem with 201,720 degrees of freedom, the convergence factor for the unrotated problem with 6 given prototypes is the same as that for the rotated problem with 7 computed prototypes and $\mu_{\max} = 10$, yet the total algorithm time is a factor of 6 larger. As the convergence factors are the same, this increase can only be attributed to the additional setup costs.

The final example demonstrates performance of the adaptive method for an elasticity problem featuring discontinuities in the Young modulus. Here we consider a 3D elasticity problem in which the Poisson ratio is fixed at 0.32, while the Young modulus is allowed to vary randomly between the elements. We consider two cases: a case of coefficients varying randomly with uniform distribution in the interval $(1, 10^{\sigma})$, and the case where the distribution is exponential, i.e., the Young modulus is computed as $10^{(\sigma r)}$, where r is generated randomly with uniform distribution in (0, 1). Keeping with the usual practice of employing Krylov method acceleration for problems with coefficient discontinuities, in this experiment we use our adaptive method as a preconditioner in the conjugate gradient method. The iteration was stopped once the initial residual was reduced by 10^8. Table 5.17 compares the results obtained by using our adaptive scheme, started from random initial guess, to the results obtained when the method based on a priori knowledge of the rigid body modes is employed as a preconditioner. The table indicates that using the adaptive procedure, without a priori knowledge of the problem geometry, we can about recover the rates of the method based on the knowledge of the rigid-body modes.

Note that the operator complexities in all of the test problems remain below 2. Moreover, for the larger spatial dimension of 3D, these complexities improve somewhat, due largely to the increased speed of aggregation coarsening. It is also worth mentioning that the increasing size of the coarse-matrix block entries due to the increasing number of prototypes does not significantly impact the time needed to perform one iteration of the solver, apparently due to the more efficient memory access afforded by blocking.

Rotated	Prototypes	$\mu_{\max}$	Iter	Factor	CPU	RelCPU	OpComp
3D elasticity problem with $114,444$ degrees of freedom							
NO	6 provided	N/A	16	0.20	29.97	1.00	1.159
NO	6	15	20	0.27	189.11	6.31	1.159
NO	7	15	17	0.21	215.78	7.20	1.217
YES	6 provided	N/A	587	0.97	913.49	30.48	1.159
YES	6	15	16	0.22	184.32	6.15	1.159
YES	7	10	15	0.20	171.73	5.73	1.217
YES	7	15	15	0.20	210.99	7.04	1.217
3D elasticity problem with $201,720$ degrees of freedom							
NO	6 provided	N/A	16	0.20	50.33	1.00	1.153
NO	6	15	21	0.31	319.60	6.35	1.153
NO	7	10	17	0.216	297.95	5.92	1.209
NO	7	15	17	0.209	363.38	7.22	1.209
YES	6 provided	N/A	739	0.97	1,924.62	38.24	1.153
YES	6	15	16	0.23	308.02	6.12	1.153
YES	7	10	15	0.20	301.98	6.00	1.209
YES	7	15	14	0.16	357.85	7.11	1.209

Table 5.16: 3D Elasticity Problems with $114,444$ and $201,720$ Degrees of Freedom

σ	Prototypes	$\mu_{\max}$	Iter	Factor	CPU	RelCPU	OpComp
Elasticity problem with uniformly distributed coefficient jumps							
2	6 provided	N/A	8	0.073	24.22	1.00	1.15
2	6	10	13	0.221	219.34	9.05	1.15
2	7	10	11	0.187	266.29	10.99	1.21
3	6 provided	N/A	8	0.077	24.23	1.00	1.15
3	6	10	13	0.215	218.57	9.02	1.15
3	7	10	11	0.186	265.70	10.96	1.21
4	6 provided	N/A	8	0.077	24.56	1.00	1.15
4	6	10	12	0.208	219.26	8.93	1.15
4	7	10	11	0.165	264.38	10.76	1.21
Elasticity problem with exponentially distributed coefficient jumps							
2	6 provided	N/A	9	0.115	25.99	1.00	1.15
2	6	10	16	0.305	225.52	8.68	1.15
2	7	10	12	0.214	267.72	10.30	1.21
3	6 provided	N/A	14	0.247	35.68	1.00	1.15
3	6	10	22	0.418	237.56	6.76	1.15
3	7	10	16	0.310	275.62	7.84	1.21
4	6 provided	N/A	20	0.395	49.99	1.00	1.15
4	6	10	30	0.532	255.43	5.11	1.15
4	7	10	21	0.404	289.39	5.79	1.21
5	6 provided	N/A	32	0.555	73.63	1.00	1.15
5	6	10	46	0.670	292.35	3.97	1.15
5	6	20	36	0.598	402.75	5.47	1.21
5	7	10	37	0.602	324.19	4.40	1.21
5	7	15	27	0.497	381.16	5.17	1.21

Table 5.17: 3D Elasticity Problem, 201,720 Degrees of Freedom, with Young Modulus Featuring Random Jumps in $(1, 10^{\sigma})$.

Chapter 6

Conclusions and Future Work

The focus of this thesis is on the efficient solution of the linear systems that approximate certain PDEs, particularly those that come from diffusion or Darcy-law flow models. The techniques developed here may be placed into two main categories, upscaling and adaptive multigrid.

Upscaling techniques deal with computing an approximation to the solution of the linear system with significantly less computational cost than solving the linear system exactly, or even to the usual tolerances accepted of iterative methods. The techniques we develop rely on a robust multigrid solver, typically BoxMG (although we also consider the extension to AMG), to perform a coarsening of the fine-scale representation of the linear system to a scale suitable for computation. Thus, we do as little work as possible on the fine scale (and, in particular, solve no portion of the linear system on any subdomain) to derive a coarse-scale model whose solution may be used to represent fine-scale behavior of the PDE. We demonstrate notably better performance than competing methods from the literature.

Adaptive multigrid methods are proposed to allow more efficient solution of the fine-scale linear systems with fewer assumptions on their character and with less information supplied about their origin. With no knowledge of the problem except the given matrix, our adaptive methods show performance characteristic of non-adaptive multigrid methods applied to systems for which full knowledge of the near null space

is available. The cost of this robustness is a more complicated setup phase, but the resulting algorithms still demonstrate linear scaling and acceptable solution times.

6.1 Contributions of this Thesis

6.1.1 Upscaling

The permeability upscaling techniques presented in Chapter 4 build on the previous work of Moulton et al. [67] and Knapek [60]. We also introduce new techniques and analysis into the field of coarse-scale modeling that show dramatic improvements on the techniques of Durlofsky [39] and He et al. [52]. Furthermore, our multilevel upscaling method provides a new type of multiscale basis, competitive with the techniques of multiscale finite elements [56, 57].

As classical homogenization theory rests on the assumption of periodic media, the effective permeability calculations of Moulton et al. [67] and Knapek [60] also make use of periodic boundary conditions. We extended this work to the case of Neumann boundary conditions, because this is the important case of natural boundary conditions for finite elements. This also allows the computation of effective permeability coefficients when the coarsened matrix has not yet reached the point of homogeneity. Our Neumann boundary condition upscaling method yields similar accuracy to the periodic boundary condition techniques, although we do not yet fully understand the implications of coarsening of the boundary conditions. We also extended the two-dimensional theory to three dimensions, having done so fully in the periodic case, and nearly so in the case of Neumann boundary conditions.

The regularization terms that appear in both two and three dimensions have significant impact on both the recovery of effective material properties and the performance of the multigrid solvers on which the upscaling is based. Without accounting for these terms, coarse-scale coefficients may be recovered from the coarse-scale operators,

but they do not reproduce the operator from which they were derived. By including these higher-order terms in our analysis, we are able to recover physically meaningful coarse-scale coefficients. The role of these regularizing terms is also important in the performance of the multigrid solvers considered as, without them, isotropic fine-grid problems may have coarse-grid operators that exhibit significant anisotropy, which would lead to poor performance of the standard pointwise smoothers in resolving error on these grids.

Our search of the existing literature on such upscaling of permeabilities led us to a detailed analysis of the work of Durlofsky [39]. This analysis culminates in Theorem 2, which states that the upscaled material properties computed using the methods in [39] are simply finite element calculations of those predicted by the classical homogenization theory, as in [5, 58]. In particular, this implies that the results in [39] are equivalent to those of Bourgat [7], who implemented a finite element version of this theory.

We also introduced a multiscale technique for computing accurate approximations to the fine-scale solutions of these equations, requiring significantly less computation than even a multigrid V-cycle based approach would. The multiscale basis functions implicit in a variational coarsening multigrid scheme create coarse-grid operators that reflect significant fine-scale information. By solving these coarse-scale problems and representing the solutions in that basis, we determine quite accurate approximations of the fine-scale solutions at a small fraction of the computational cost. These approximations accurately reflect both macroscopic properties of the flow, such as global fluxes, while also capturing the finer-scale structures necessary for useful modeling of the related nonlinear problems. In both cases, our approach is much more accurate than that in [39, 52], both in fine-scale structure, as shown in the figures of Section 4.4, and in quantitative measures, such as integrated outflow fluxes.

6.1.2 Adaptive Multigrid Methods

Algebraic multigrid methods [23, 73] and smoothed aggregation multigrid [64, 82] are known to be efficient multigrid solvers based solely on the algebraic system to be solved. AMG has been shown to perform well for systems whose low energy modes are locally constant, regardless of geometry or problem size [31]. When provided with a full basis for the near null space of the matrix, smoothed aggregation can be shown to also exhibit optimal convergence behavior [79].

The principles of an adaptive multigrid method are centered on the idea of using a tentative solver to expose errors that must be accounted for in the multigrid coarsening process. Of primary importance in this task is the concept of algebraic smoothness, which is easily exposed through relaxation. Relaxation on the homogeneous problem, $A\mathbf{x} = \mathbf{0}$, produces prototypical errors which may be used to calibrate the interpolation choice in the multigrid scheme. That an interpolation based on only a few such components can be effective is the cornerstone of all multigrid methods. Interpolation is chosen to fit these prototypes locally, and thus all components with similar local character may be effectively represented by their coarse-grid counterparts. The principle of a self-testing method allows efficient adaptation: if performance on the homogeneous problem is sufficient, we know we have an acceptable solver; if it is not, we know the component exposed by application of the current solver is one to which more attention should be paid. Overriding all of these concerns is a desire for optimality of both the adaptive process and the resulting multigrid method. Thus, choices in the adaptation are always made so that they provide an optimal change with minimal wasted computation.

The adaptive algebraic multigrid scheme follows these adaptive principles in the framework of classical AMG. The definition of interpolation for αAMG is based on a similar derivation as that of classical AMG, but with the modified assumption that the global character of algebraically smooth error is given as a vector rather than assumed

to be locally constant. The interpolation can be shown to reduce to that of classical AMG if the prototype vector is constant, and can also be extended to a simple case for systems of PDEs. We show that the resulting multigrid algorithm is then invariant to diagonal scalings of the matrix, a problem that causes significant difficulties for the classical AMG method. Practical considerations lead us to consider a strategy for taking a small number of adaptive steps and the limiting case of allowing only a single adaptation with a hand-tuned number of relaxation sweeps in the setup shows good performance. Performance suffers somewhat for the adaptive approach compared to the calibrated approach, largely because we still base interpolation on a single prototype vector, which may not be the slowest-to-diminish component under an existing, but non-optimal, V-cycle.

Theoretical understanding of αAMG is considered in a two-level, reduction-based setting. We establish that the adaptive step reduces the Rayleigh Quotient of the approximation to the algebraically smoothest component, although without a uniform bound. We show better bounds in the special case of a 2-degree-of-freedom coarse grid, which motivates a discussion of the role of inverse iteration in this setting. Under further simplifying assumptions, the mapping of the error in the approximation from one step to the next may be shown to be a contraction.

The application of the adaptive framework to smoothed aggregation is, in some ways, more natural than to that of AMG. In particular, the introduction of additional prototypes into interpolation is natural and can be done without significant change to the basic method. However, important issues arise when considering the adaptive development of the prototype set, particularly to distinguish local redundancy in the case of global distinction among the prototypes, and to ensure the cost is kept to practical levels. Good numerical results are shown for the three-dimensional Laplacian, as well as a variety of linear elasticity problems, including those that have been diagonally scaled or had rotations of degrees of freedom introduced.

6.2 Future Work

6.2.1 Upscaling

The results presented in Chapter 4 show efficient and accurate performance of the multilevel upscaling techniques based on BoxMG for the problem of saturated flow in a porous media. Accurate comparison with the many other techniques currently in use or under development for this problem is, in itself, a significant task. Extending the results to more complicated domains and more complicated flows is also necessary for the technique to be useful in real-world applications, such as reservoir modeling. Additionally, the techniques developed for upscaling this class of PDEs may also be examined for extension to other, more complex, families of problems.

The Neumann boundary condition permeability upscaling technique offers similar performance to existing techniques based on periodic boundary conditions in many situations. There appears, however, to be an inconsistency in the coarsening of these natural boundary conditions that affects the results of the upscaling in certain cases. Tracking down this inconsistency will provide insight into both the upscaling results and the coarsening used in BoxMG. We would like to discover both why the results appear as they do, and what changes to the coarsening used might provide effective permeabilities that agree with existing homogenization theory in cases where it applies.

As discussed at the end of Section 4.4.4, a near-term goal is to compare our results with those obtained using the multiscale finite element method (MSFEM) of Hou et al. [56, 57]. We believe that there is a significant advantage in accuracy in using a multiscale basis for the coarse-scale model over techniques that use an upscaled permeability and a coarse-scale basis. Thus, we expect a similar level of accuracy as the MSFEM exhibits. A negative feature of the MSFEM method, however, is the need to create the multiscale basis functions from numerically solving fine-scale subdomain problems. Our analysis will thus center on the idea of accuracy versus the computational

cost of both approaches.

Allowing for upscaling of more complex geometries is an important goal for this technique, due to the irregular structure of real-world domains, such as reservoirs, and components of the model, such as wells. These features can be approximated by the geometrically-regular grids of BoxMG, but are much more naturally considered in the algebraic multigrid framework. We are currently implementing the multilevel upscaling technique in the AMG setting, and believe that this will allow us to tackle much more complex domains than we can currently consider. A concern of using the AMG code, however, is the purely algebraic (and, thus, likely non-physical) nature of the interpolation used there. The degree to which this impedes the accuracy of the coarse-scale model must be evaluated. In other words, while the coarse-scale matrices of an AMG hierarchy still represent a multiscale discretization of the original PDE, it is unknown as to whether the basis functions used in this approach reflect enough of the fine-scale structure to provide accurate coarse-scale approximations. Another interesting avenue of investigation is the upscaling properties of the adaptive multigrid methods considered in Chapter 5

We have tested out methods using realistic 2D permeability fields, generated using standard geostatistical techniques. Application of our method to real-world applications also requires an extension of these ideas to three dimensions. In principle, this is easily done, and is the subject of current research in the case of computing upscaled permeabilities. The principles of multilevel upscaling extend obviously to three dimensions, and, in the structured case, a 3D BoxMG code is available. We intend to develop these ideas in three dimensions, and, having done so, will seek to apply them to appropriate models of permeability.

The saturated single-phase flows considered in this thesis are the simplest in a series of mathematical models of flow in porous media. An important generalization is

to the case of unsaturated single-phase flow, where the saturation, $S(\mathbf{x})$, satisfies

$$-\nabla \cdot D(S, \mathbf{x}) \nabla S(\mathbf{x}) = Q(\mathbf{x}).$$

Here, the nonlinearity appears because the diffusivity, $D(S, \mathbf{x})$, depends upon the permeability, $\mathcal{K}(S, \mathbf{x})$, which depends on the level of saturation. Effective iteration on this nonlinearity, using, for example, Newton's method, or in the case of a time-dependent flow, requires an accurate representation of $S(\mathbf{x})$ in order to predict the coefficient of the PDE at the next time step. Our results indicate that we can achieve such an approximation using the multilevel upscaling technique, and thus derive better fine-scale and coarse-scale models for the next step in the iteration.

Theoretical approaches to upscaling, e.g. the method of moments [68], are, on the surface, very different from many numerical approaches, including the one considered here. The explicit closure required when considering a stochastic model of permeability also results in elliptic equations of similar form [68]. As we consider upscaling only a single realization of a permeability field, we have not introduced an explicit closure. Whether the implicit closure can be formulated and compared to the explicit closure used in analytic calculations of effective permeabilities is an open question that we are investigating.

Homogenization methods exist for problems other than the $-\nabla \cdot \mathcal{K} \nabla$ operator considered here, and upscaling techniques may prove useful for many operators where material properties vary on fine scales. The techniques developed in Chapter 4 are effective, in part, because of the depth of multigrid techniques for solving diffusion-type problems. Other PDE settings, such as elasticity or Maxwell's equations, do not have as rich of a class of effective multigrid solvers. As methods for these problems are developed, the techniques used here can be considered, within the confines of variational multigrid methods that are a simple combination of relaxation and coarse-grid correction (multiple coarse grids, for example, would not easily allow such an approach).

6.2.2 Adaptive Multigrid Methods

While we have demonstrated that adaptive multigrid methods, such as the αAMG and αSA methods described in Chapter 5, can provide significant improvements over the classical AMG and SA methods, there is still much to be done in improving the robustness of the multigrid framework. While it is unrealistic to expect such methods to solve all matrix equations, we believe it is possible to further extend the class of problems for which efficient, adaptive multigrid solution is possible.

The αAMG method of Section 5.2 removes the assumption on the local character of algebraically smooth error of classical AMG, but does not escape the implicit assumption of a one-dimensional near null space that is typical of discretizations of scalar, second-order PDEs. Expanding the definition of interpolation to allow fitting of multiple prototypical smooth-error vectors is the next step in improving the generality of this method.

As long as the size of the coarse-grid interpolatory set, $|C_i|$, is greater than the number of prototypes to be fit, we have the freedom to fit all prototypes as accurately as we wish (in fact, they can be fit exactly, although the eigenvector approximation criterion should be the guiding principle in this fit). We view the choice of this fit as a choice on collapsing the fine-fine grid connections. While we choose a one-dimensional modification of the coefficients in A_{ff} in the single prototype case (dividing them all by a constant factor for each i), multiple prototypes would require a multi-dimensional modification.

There are two approaches to the computational task of defining interpolation based on multiple prototypes. A straightforward approach would require recomputing the interpolation operator every time new information (in the form of a new prototype) becomes available. We can also consider an approach where interpolation is updated when a new prototype becomes available, fitting exactly the character of the prototype

that is missed by the current interpolation by adding an update to the existing operator. This approach seems simpler computationally, but it may be no easier to ensure that the update does not pollute the interpolation of the previous prototypes than it is to simply recompute the complete interpolation operator.

Once we are able to develop effective interpolation operators based on multiple prototypes, we will need to further explore the setup phase of αAMG to determine how to most effectively generate the needed prototypes. Our experience with the αSA algorithm should prove valuable here. The recursive form of the setup procedure, where the role of relaxation in the first cycle is always replaced by the current solver in subsequent cycles, ensures that error exposed in the setup phase is always algebraically smooth and also not effectively reduced by the current coarse-grid correction. Thus, it should be a very good prototype upon which to augment the current interpolation scheme.

One interesting idea is to explore the interaction of multigrid and the preconditioned conjugate gradient algorithm (PCG). Through the relationship with the Lanczos algorithm, we can find the smoothest mode of the preconditioned matrix, $M^{-1}A$, where M^{-1} represents the action of a multigrid V-cycle in very little additional work than the PCG iteration itself. Thus, given a tentative multigrid cycle, we could use it as a solver for $A\mathbf{x} = \mathbf{b}$ in PCG and, if the iteration is slow to converge, generate the necessary information to improve the solver at the same time as we iterate to find the solution. While this combination proved to be ineffective for scalar second-order PDEs, it may be effective for systems where multiple prototypes, all rich in the algebraically smooth components, are used together to represent the near null space of A. A similar idea, using block PCG to iterate on $A\mathbf{x} = \mathbf{0}$ simultaneously with $A\mathbf{x} = \mathbf{b}$, was recently explored in [30].

Chapter 5 deals primarily with the choices of interpolation in an adaptive multigrid setting. Of no less importance is the appropriate choice of a coarse grid. Compatible relaxation (CR) [61] may be a natural complement to the schemes considered here.

Just as we start with a sample of algebraically smooth error to determine interpolation, CR uses the same sort of prototypical error to choose the coarse grid. This arises naturally because of the need for complementarity in a multigrid process. Given a sample of the error that we are primarily interested in correcting, points may be tested as coarse-grid nodes by measuring the suitability of the correction to this component that they induce.

Exactly how to measure this suitability is a question of much current debate. Theory of Brandt [18] suggests that the suitability may be indicated simply by the relative size of the prototypical error when the matrix has been properly scaled [61]. Such scaling may, however, be counter-intuitive in the case of discretizations of discontinuous-coefficient diffusion problems, where all points are equally useful as coarse-grid nodes but the scaling induces a predisposition to choosing nodes where the fine-grid diffusion coefficient is small. The theory of Falgout and Vassilevski [43] can be used to suggest that such a scaling of the matrix is unnecessary if the smoother is taken into account in measuring the relative size of the prototype's nodal values.

Knowing that interpolation of the form $A_{ff}^{-1}A_{fc}$ leads to an exact coarse-grid correction, although dense Galerkin coarse-grid operators, choosing interpolation may be viewed as trying to approximate A_{ff}^{-1} with a local operator. The i^{th} column of A_{ff}^{-1} may be approximated by relaxation on the problem $A_{ff}\mathbf{x}_i = \hat{\mathbf{e}}_i$, where $\hat{\mathbf{e}}_i$ is the i^{th} canonical unit basis vector. Thus, the suitability of a particular node, i, to be a coarse-grid point may be measured by the performance of a local relaxation on the vector that is chosen to be 1 at point i and zero elsewhere. Such an approach also has the advantage that it exposes strong connections between i and its neighbors with the presence of large values in the relaxed vector. A similar local-relaxation-based approach considers the ability of a coarse-grid node to correct the particular prototypical error by assessing the reduction in the local size of the prototype when node i is fixed to 0 (as if it is exactly corrected from the coarse grid) and relaxation on the homogeneous

problem is performed.

Once the quality of the fine-grid points as coarse-grid nodes has been assessed, a subset must be selected to form the actual coarse grid. This is typically done through a maximal independent set algorithm, weighted by the coarse-grid candidate measures. This approach can, however, run into ambiguity in the case of ties in candidate measures, where a suboptimal choice in tie-breaking may lead to a significant increase in V-cycle complexity. An alternative is to use the candidate measures to threshold points into potential coarse-grid nodes and poor coarse-grid candidates, then weight the maximal independent set algorithm based on the number of strong connections, as in classical AMG.

Regardless of the details of the compatible relaxation scheme, there appears to be some advantage in closely integrating CR with the choice of interpolation within αAMG. Both rely on the exposure and use of a prototype of algebraically smooth error, and thus work may be saved by computing this component once for both uses. Early tests of this integration have shown slight improvements in convergence factors for many problems, such as those discussed in Section 5.2.5. Another important question in this integration is updating of the coarse grid as better prototypes of algebraically smooth error become available. Success was achieved in αSA while performing aggregation once, independent of the prototype set; however, slight improvements in the coarse grid may be more practical in the AMG framework where the coarse grid is simply a subset of the fine grid.

While the theory presented in Section 5.3 has contributed to our understanding of the adaptive process and the automatic exposure of algebraically smooth components, the lack of uniform control over the convergence bound suggests that this is an incomplete understanding. The αAMGr framework is an attractive two-level setting for considering convergence of the αAMG method, and we hope to utilize it to achieve a more satisfying result. Necessary conditions for αAMG V-cycle convergence would also

be of significant interest to us and, hopefully, enable us to improve our choices in the overall adaptive scheme.

Finally, the computational codes developed for this thesis have been primarily research-oriented codes. Practical applications require significantly more energy invested in performance, optimization, and parallelization than we have yet devoted. The lion's share of the computation in the adaptive MG methods presented here is in operations, such as matrix-vector and matrix-matrix products, that can be easily tuned and easily parallelized to achieve the performance necessary to be competitive with other methods that may be less attractive mathematically, but for which there exist more efficient implementations. Such optimizations can be implemented to improve performance of the current code, whereas parallel development would require more effort, although may be achievable within the HYPRE/BoomerAMG packages [44, 53].

Bibliography

[1] S. V. Ahamed, Accelerated convergence of numerical solution of linear and non-linear vector field problems, Comput. J., 8 (1965), pp. 73–76.

[2] R. E. Alcouffe, A. Brandt, J. E. Dendy, and J. W. Painter, The multi–grid method for the diffusion equation with strongly discontinuous coefficients, SIAM J. Sci. Stat. Comput., 2 (1981), pp. 430–454.

[3] N. S. Bakhvalov, On the convergence of a relaxation method under natural constraints on an elliptic operator, Z. Vycisl. Mat. i. Mat. Fiz., 6 (1966), pp. 861–883.

[4] J. Bear, Dynamics of Fluids in Porous Media, American Elsevier Publishing Company, New York, 1972.

[5] A. Bensoussan, J.-L. Lions, and G. Papanicolaou, Asymptotic Analysis For Periodic Structures, vol. 5 of Studies in Mathematics and its Applications, North–Holland, New York, 1978.

[6] G. Beylkin and N. Coult, A multiresolution strategy for reduction of elliptic PDEs and eigenvalue problems, Appl. Comput. Harmon. Anal., 5 (1998), pp. 129–155.

[7] J. F. Bourgat, Numerical experiments of the homogenization method for operators with periodic coefficients, in Computing Methods in Applied Science and Engineering, I, R. Glowinski and J.-L. Lions, eds., Versailles, December 5–9 1977, Springer, pp. 330–356.

[8] D. Braess, Finite Elements, Cambridge University Press, Cambridge, 2001. Second Edition.

[9] J. H. Bramble, Multigrid Methods, vol. 294 of Pitman Research Notes in Mathematical Sciences, Longman Scientific & Technical, Essex, England, 1993.

[10] J. H. Bramble and J. E. Pasciak, New convergence estimates for multigrid algorithms, Math. Comp., 49 (1987), pp. 311–329.

[11] ——, New estimates for multigrid algorithms including the V–cycle, Math. Comp., 60 (1993), pp. 447–471.

[12] J. H. Bramble, J. E. Pasciak, J. Wang, and J. Xu, Convergence estimates for multigrid algorithms without regularity assumptions, Math. Comp., 57 (1991), pp. 23–45.

[13] ——, Convergence estimates for product iterative methods with applications to domain decomposition, Math. Comp., 57 (1991), pp. 1–21.

[14] A. Brandt, Multi–level adaptive technique (MLAT) for fast numerical solution to boundary value problems, in Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics, H. Cabannes and R. Teman, eds., vol. 18 of Lecture Notes in Physics, Berlin, 1973, Springer–Verlag, pp. 82–89.

[15] ——, Multi–level adaptive techniques (MLAT) I. the multi-grid method, research report, IBM Thomas J. Watson Research Center, 1976.

[16] ——, Multi–level adaptive solutions to boundary–value problems, Math. Comp., 31 (1977), pp. 333–390.

[17] ——, Multi–level adaptive techniques (MLAT) for partial differential equations: ideas and software, in Mathematical Software III, J. R. Rice, ed., Academic Press, New York, 1977, pp. 277–318.

[18] ——, Algebraic multigrid theory: The symmetric case, Appl. Math. Comput., 19 (1986), pp. 23–56.

[19] ——, Rigorous local mode analysis of multigrid, in Preliminary Proc. of the 4th Copper Mountain Conference on Multigrid Methods, J. Mandel and S. F. McCormick, eds., vol. 1, Denver, 1989, Computational Mathematics Group, Univ. of Colorado, pp. 55–133.

[20] ——, General highly accurate algebraic coarsening, Elect. Trans. Numer. Anal., 10 (2000), pp. 1–20.

[21] ——. a lecture given at CASC, Lawrence Livermore National Lab, June 2001.

[22] A. Brandt, S. F. McCormick, and J. W. Ruge, Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations, tech. rep., Institute for Computational Studies, Colorado State University, 1982.

[23] ——, Algebraic multigrid (AMG) for sparse matrix equations, in Sparsity and Its Applications, D. J. Evans, ed., Cambridge University Press, Cambridge, 1984.

[24] A. Brandt and D. Ron, Multigrid solvers and multilevel optimization strategies. manuscript, 2002.

[25] M. Brezina, A. Cleary, R. Falgout, V. Henson, J. Jones, T. Manteuffel, S. McCormick, and J. Ruge, Algebraic multigrid based on element interpolation (AMGe), SIAM J. Sci. Comp., 22 (2000), pp. 1570–1592.

[26] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, Adaptive smoothed aggregation (αSA), SIAM J. Sci. Comp., 25 (2004), pp. 1896–1920.

[27] M. Brezina, C. I. Heberton, J. Mandel, and P. Vaněk, An iterative method with convergence rate chosen a priori, UCD/CCM Report 140, Center for Computational Mathematics, University of Colorado at Denver, February 1999. http://www-math.cudenver.edu/ccmreports/rep140.ps.gz.

[28] W. Cardwell and R. Parsons, Average permeabilities of heterogeneous oil sands, Trans. Am. Inst. Min. Metall. Pet. Eng., 160 (1945), pp. 34–42.

[29] T. Chartier, R. Falgout, V. Henson, J. Jones, T. Manteuffel, S. McCormick, J. Ruge, and P. Vassilevski, Spectral AMGe (ρAMGe), SIAM J. Sci. Comp., 25 (2003), pp. 1–26.

[30] E. Chow, An aggregation multilevel method using smooth error vectors, SIAM J. Sci. Comp., (2004). Submitted.

[31] A. Cleary, R. Falgout, V. Henson, J. Jones, T. Manteuffel, S. McCormick, G. Miranda, and J. Ruge, Robustness and scalability of algebraic multigrid, SIAM J. Sci. Comp., 21 (2000), pp. 1886–1908.

[32] F. de la Vallee Poussin, An accelerated relaxation algorithm for iterative solution of elliptic equations, SIAM J. Numer. Anal., 5 (1968), pp. 340–351.

[33] J. W. Demmel, Applied Numerical Linear Algebra, SIAM, Philadelphia, 1997.

[34] J. E. Dendy, Black box multigrid, J. Comput. Phys., 48 (1982), pp. 366–386.

[35] ——, Black box multigrid for nonsymmetric problems, Appl. Math. Comput., 13 (1982), pp. 261–284.

[36] ——, Black box multigrid for systems, Appl. Math. Comput., 19 (1986), pp. 57–74.

[37] ——, Black box multigrid for periodic and singular problems, Appl. Math. Comput., 25 (1988), pp. 1–10.

[38] C. Deutsch and A. Journel, GSLIB, geostatistical software library and user's guide, Oxford University Press, Oxford, second ed., 1998.

[39] L. Durlofsky, Numerical-calculation of equivalent grid block permeability tensors for heterogeneous porous-media, Water Resour. Res., 27 (1991), pp. 669–708.

[40] ——, Accuracy of mixed and control volume finite element approximations to Darcy velocity and related quantities, Water. Resour. Res., 30 (1994), pp. 965–973.

[41] J. Dvořák, A reliable numerical method for computing homogenized coefficients, mat-report 31, Mathematical Institute, Danish Technical University, Lyngby, Denmark, 1995.

[42] R. Falgout, A note on the relationship between adaptive AMG and PCG, technical report, Lawrence Livermore National Laboratory, 2004.

[43] R. Falgout and P. Vassilevski, On generalizing the AMG framework, SIAM J. Numer. Anal., (2003). submitted.

[44] R. Falgout and U. Yang, hypre: a library of high performance preconditioners, in Computational Science - ICCS 2002: International Conference, Amsterdam, The Netherlands, April 21-24, 2002. Proceedings, Part III, no. 2331 in Lecture Notes in Computer Science, Springer-Verlag, 2002, pp. 632–641.

[45] R. P. Fedorenko, A relaxation method for solving elliptic difference equations, Z. Vycisl. Mat. i. Mat. Fiz., 1 (1961), pp. 922–927. Also in U.S.S.R. Comput. Math. and Math. Phys., 1 (1962), pp. 1092–1096.

[46] ——, The speed of convergence of one iterative process, Z. Vycisl. Mat. i. Mat. Fiz., 4 (1964), pp. 559–563. Also in U.S.S.R. Comput. Math. and Math. Phys., 4 (1964), pp. 227–235.

[47] J. Fish and V. Belsky, Generalized aggregation multilevel solver, International Journal for Numerical Methods in Engineering, 40 (1997), pp. 4341–4361.

[48] A. Gilbert, A comparison of multiresolution and classical one-dimensional homogenizatio schemes, Appl. Comput. Harmon. Anal., 5 (1998), pp. 1–35.

[49] D. Gines, G. Beylkin, and J. Dunn, LU factorization of non-standard forms and direct multiresolution solvers, Appl. Comput. Harmon. Anal., 5 (1998), pp. 156–201.

[50] A. Greenbaum, Iterative Methods for Solving Linear Systems, Frontiers in Applied Mathematics, SIAM, Philadelphia, 1997.

[51] I. Gustafsson, A class of 1st order factorization methods, BIT, 18 (1978), pp. 142–156.

[52] C. He, M. Edwards, and L. Durlofsky, Numerical calculation of equivalent cell permeability tensors for general quadrilateral control volumes, Computational Geosciences, 6 (2002), pp. 29–47.

[53] V. Henson and U. Yang, Boomeramg: a parallel algebraic multigrid solver and preconditioner, Applied Numerical Mathematics, 41 (2002), pp. 155–177.

[54] M. Hestenes, The conjugate-gradient method for solving linear systems, in Proceedings of Symposia on Applied Mathematics, vol. VI, Numerical Analysis, New York, 1956, McGraw–Hill, pp. 83–102.

[55] M. Hestenes and E. Steifel, Methods of conjugate gradients for solving linear systems, J. Research NBS, 49 (1952), pp. 409–436.

[56] T. Hou and X. Wu, A multiscale finite element method for elliptic problems in composite materials and porous media, J. Comput. Phys., 134 (1997), pp. 169–189.

[57] T. Hou, X. Wu, and Z. Cai, Convergence of a multiscale finite element method for elliptic problems with rapidly oscillating coefficients, Math. Comp., 68 (1999), pp. 913–943.

[58] V. V. Jikov, S. M. Kozlov, and O. A. Oleinik, Homogenization of Differential Operators and Integral Functionals, Springer–Verlag, 1994. Translated from Russian.

[59] P. R. King, The use of renormalization for calculating effective permeability, Transport in Porous Media, 4 (1989), pp. 37–58.

[60] S. Knapek, Matrix–dependent multigrid homogenization for diffusion problems, SIAM J. Sci. Comput., 20 (1998), pp. 515–533.

[61] O. Livne, Coarsening by compatible relaxation, Num. Lin. Alg. Appl., 11 (2004), pp. 205–227.

[62] S. F. McCormick and J. W. Ruge, Multigrid methods for variational problems, SIAM J. Numer. Anal., 19 (1982), pp. 924–929.

[63] ——, Algebraic multigrid methods applied to problems in computational structural mechanics, in State-of-the-Art Surveys on Computational Mechanics, ASME, New York, 1989, pp. 237–270.

[64] S. Míka and P. Vaněk, Acceleration of convergence of a two level algebraic algorithm by aggregation in smoothing process, Appl. Math., 37 (1992), pp. 343–356.

[65] ——, A modification of the two-level algorithm with overcorrection, Appl. Math., 37 (1992), pp. 13–28.

[66] J. Moulton, S. Knapek, and J. Dendy, Multilevel upscaling in heterogeneous porous media, cnls report, Center for Nonlinear Studies, Los Alamos National Laboratory, Los Alamos, New Mexico, 1999.

[67] J. D. Moulton, J. E. Dendy, and J. M. Hyman, The black box multigrid numerical homogenization algorithm, J. Comput. Phys., 141 (1998), pp. 1–29.

[68] E. Paleologos, S. Neuman, and D. Tartakovsky, Effective hydraulic conductivity of bounded, strongly heterogeneous porous media, Water Resources Research, 32 (1996), pp. 1333–1341.

[69] J. Ruge, Algebraic multigrid applied to systems of partial differential equations, in Proc. Int'l Multigrid Conf., S. McCormick, ed., Amsterdam, Apr. 1985, North Holland.

[70] ——, Final report on amg02, tech. rep., Gesellschaft für Mathematik und Darenverarbeitung, St. Augustin, 1985.

[71] J. W. Ruge, Algebraic multigrid (AMG) for geodetic survey problems, in Prelimary Proc. Internat. Multigrid Conference, Fort Collins, CO, 1983, Institute for Computational Studies at Colorado State University.

[72] J. W. Ruge and K. Stüben, Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG), in Multigrid Methods for Integral and Differential Equations, D. J. Paddon and H. Holstein, eds., The Institute of Mathematics and its Applications Conference Series, Clarendon Press, Oxford, 1985, pp. 169–212.

[73] ——, Algebraic multigrid (AMG), in Multigrid Methods, S. F. McCormick, ed., vol. 3 of Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, 1987, pp. 73–130.

[74] R. V. Southwell, Stress calculation in frameworks by the method of systematic relaxation of constraints. I,II, Proc. Roy. Soc. London Ser. A, 151 (1935), pp. 56–95.

[75] ——, Relaxation Methods in Engineering Science, Oxford University Press, Oxford, 1940.

[76] E. Stiefel, Über einige methoden der relaxationsrechnung, Z. Angew. Math. Physik, 3 (1952), pp. 1–33.

[77] P. Vaněk, Acceleration of convergence of a two level algorithm by smoothing transfer operators, Appl. Math., 37 (1992), pp. 265–274.

[78] ——, Fast multigrid solver, Appl. Math., 40 (1995), pp. 1–20.

[79] P. Vaněk, M. Brezina, and J. Mandel, Convergence of algebraic multigrid based on smoothed aggregation, Numer. Math., 88 (2001), pp. 559–579.

[80] P. Vaněk, J. Mandel, and M. Brezina, Algebraic multigrid based on smoothed aggregation for second and fourth order problems, Computing, 56 (1996), pp. 179–196.

[81] P. Vaněk, M. Brezina, and R. Tezaur, Two-grid method for linear elasticity on unstructured meshes, SIAM J. Sci. Comp., 21 (1999), pp. 900–923.

[82] P. Vaněk, J. Mandel, and M. Brezina, Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems, Computing, 56 (1996), pp. 179–196.

[83] R. S. Varga, Matrix Iterative Analysis, Springer Series in Computational Mathematics, Springer, Berlin, 2000. Second Edition.

[84] E. L. Wachspress, Iterative Solution of Elliptic Systems and Applications to the Neturon Diffusion Equations of Reactor Physics, Prentice-Hall International Series in Applied Mathematics, Prentice-Hall, Englewood Cliffs, N.J., 1966.

[85] J. Warren and H. Price, Flow in heterogeneous porous media, Society of Petroleum Engineering Journal, 1 (1961), pp. 153–169.

[86] X.-H. Wen and J. J. Gómez-Hernández, Upscaling hydraulic conductivities in heterogeneous media: An overview, Journal of Hydrology, 183 (1996), pp. ix–xxii.

[87] C. White, Representation of heterogeneity for numerical reservoir simulation, PhD thesis, Stanford University, 1987.