UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

Scalable and Multileve Iterative Methods
(Ph.D. Thesis)

Wing Lok Wan

June 1998

CAM Report 98-29

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

UNIVERSITY OF CALIFORNIA

Los Angeles

Scalable and Multilevel Iterative Methods

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in Mathematics

by

Wing Lok Wan

1998

The dissertation of Wing Lok Wan is approved.

Bjorn Engquist

Stanley Osher

Xiaolin Zhong

Tony Chan, Committee Chair

University of California, Los Angeles

1998

DEDICATION

To my love Winnie:

Words cannot express my heartfelt thanks and gratitude to you. Your presence and support complete the fullness of my study.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I would like to thank Tony Chan, my advisor, mentor and teacher, for his guidance, patience and support. I am indebted to his help to make my Ph.D. study abroad a reality. His inspiration and sharing of knowledge provide perspective and wit. I would like to thank other members of the committee: Bjorn Engquist, Stanley Osher and Xiaolin Zhong, for their comments on my research and teaching during my years at UCLA.

Certain other individuals have played important roles in my graduate school career. I would like to thank Barry Smith for granting me an opportunity to experience state-of-the-art parallel computing at Argonne National Laboratory, and enlightening my work on multigrid and wavelet preconditioning. I would like to thank my academic uncle, Wei-Pai Tang, for his paternal advice and encouragement, and his illuminating ideas on my sparse approximate inverse work. My sincere thanks also go to Jinchao Xu and Jun Zou for deepening my understanding of multigrid and finite element theory. I would also like to thank Raymond Chan for his help on my multigrid work for differential-convolution equations. Special thanks go to two dear mathematicians: late Dr. Wong for his enthusiastic teaching, and Dr. Ng for his kind concerns and caring during my study at CUHK.

Finally, I am grateful to my parents for encouraging me to study abroad and to make my childhood dream come true.

This work was partially supported by the Dissertation Year Fellowship granted by the Alfred P. Sloan Foundation, the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract W07405-Eng-48, the NSF under contract ASC-9201266, DMS-9626755, ASC-9720257, the ONR under contract N00014-92-J-1890, N00014-96-I-0277, and the ARO under contract DAAL-03-91-G-150, DAAL-03-91-C-0047.

VITA

August 7, 1970	Born, Aberdeen, Hong Kong.
1992	B.Sc.(First Class Honors), Mathematics The Chinese University of Hong Kong
1992	Teaching Assistant The Chinese University of Hong Kong
1992–93	Croucher Foundation Scholarship
1993–98	Teaching Assistant and Research Assistant University of California, Los Angeles
1995	M.A., Mathematics University of California, Los Angeles
1995	Givens Research Associates Mathematics and Computer Science Division Argonne National Laboratory
1996–97	Alfred P. Sloan Foundation Doctoral Dissertation Year Fellowship
1997–98	ISCR Grant Award Lawrence Livermore National Laboratory

PUBLICATIONS AND PRESENTATIONS

Chan, T. F., and Wan, W. L. (1997). *Analysis of Projection Methods for Solving Linear Systems with Multiple Right-hand Sides.* SIAM Journal on Scientific Computing, 18:1698–1721.

Chan, T. F., Tang, W-P., and Wan, W. L. (1997). *Wavelet Sparse Approximate Inverse Preconditioning.* BIT, 37:644–660.

Chan, R., Chan, T. F., and Wan, W. L. (1997). *Multigrid for Differential-Convolution Problems Arising from Image Processing.* Proceedings of the Workshop on Scientific Computing, Hong Kong, Springer-Verlag.

Tang, W-P., and Wan, W. L. (1998). *Sparse Approximate Inverse Smoothing for Multigrid.* UCLA Mathematics Department CAM Report 98-18.

Wan, W. L. (April, 1996). *Fast Wavelet Based Sparse Approximate Inverse Preconditioner.* Talk presented at the 1996 Copper Mountain Conference on Iterative Methods, Colorado.

Wan, W. L. (April, July, August, 1997). *An Energy-Minimizing Interpolation for Multigrid Methods.* Talks presented at the 1997 Copper Mountain Conference on Multigrid Methods in Colorado, the SIAM 45th Anniversary Meeting in Stanford, California, and the International Conference on Domain Decomposition Methods in Boulder, Colorado.

Wan, W. L. (October, 1997). *Wavelet Sparse Approximate Inverse Preconditioning.* Talk presented at the Sixth SIAM Conference on Applied Linear Algebra, Snowbird, Utah.

Wan, W. L., Chan, T. F., and Smith, B. F. (1998). *An Energy-Minimizing Interpolation for Robust Multigrid Methods.* UCLA Mathematics Department CAM Report 98-6.

Wan, W. L. (April, 1998). *Sparse Approximate Inverse Smoothing for Multigrid.* Talk presented at the 1998 Copper Mountain Conference on Iterative Methods, Colorado.

Wan, W. L. (1998). *An Interface Preserving Coarsening Multigrid for Elliptic Problems with Highly Discontinuous Coefficients.* UCLA Mathematics Department CAM Report 98-27.

ABSTRACT OF THE DISSERTATION

Scalable and Multilevel Iterative Methods

by

Wing Lok Wan
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 1998
Professor Tony Chan, Chair

In this dissertation, we analyze three classes of iterative methods which are often used as preconditioners for Krylov subspace methods, for the solution of large and sparse linear systems arising from the discretization of partial differential equations. In addition, we propose algorithms for image processing applications and multiple right-hand side problems.

The first class is the *incomplete LU factorization* preconditioners, an intrinsic sequential algorithm. We develop a parallel implementation of ILU(0) and devise a strategy for a priori memory allocation crucial for ILU(k) parallelization.

The second class is the *sparse approximate inverse* (SPAI) preconditioners. We improve and extend its applicability to elliptic PDEs by using wavelets which converts smoothness, often found in the Green's functions of PDE operators, into small wavelet coefficients, thus producing a more efficient approximation. We also give analytic estimates for the wavelet SPAI preconditioners.

The third class is the *multigrid* preconditioners. We propose and analyze new techniques for the basic components of multigrid: smoothing, coarsening and interpolation. First, we present a new class of parallel SPAI smoothers which offer more robustness than the relaxation smoothers by improving their quality using more nonzeros in the approximate inverse. Second, we propose an interface preserving coarsening, a new technique for solving discontinuous coefficient PDEs. It selects coarse grid points in such a way that the interfaces will align with or are resolved by all the coarse grids, allowing linear interpolation to give fast multigrid convergence. Finally, we propose and analyze a new robust interpolation method based on the concepts of energy minimization and approximation. In one dimension, we prove a super-optimal convergence rate independent of both the mesh size and the PDE coefficients. In two dimensions, we prove a two level convergence and show that the resulting multigrid is effective for rough coefficient problems. The energy minimization principle can be also applied to indefinite problems.

As an application, we propose the use of fast transform preconditioned conjugate gradient methods as smoothers to solve differential-convolution equations arising from PDE-based image problems by multigrid. These smoothers remove the high frequency errors which the usual relaxation smoothers cannot damp away. The resulting multigrid is effective for the Tikhonov regularization approach.

We analyze a class of projection methods for solving multiple right-hand side problems. We observe and prove two fundamental properties, namely: the superconvergence of the conjugate gradient refinement process and the automatic exploitation of the near rank deficiency of the right-hand side matrix. Furthermore, we propose a block generalization of it.

CHAPTER 1

Introduction

1.1 Overview

With continued improvements in computer hardware and architecture design, scientists and engineers are attempting numerical simulations of complex phenomena involving increasingly large number of unknowns. The underlying mathematical models are often described by partial differential equations (PDEs), a discretization of which often results in extremely large and sparse linear systems. An efficient solution approach is the combination of two iterative method techniques, namely, Krylov subspace methods and preconditioning. While Krylov subspace methods are a subject of interest by themselves, this dissertation is mainly devoted to the study of scalable, multilevel and robust preconditioning and issues related to it.

A success of iterative methods for large scale computing is their simplicity. They often just need perform simple operations such as matrix-vector product and vector arithmetics. Hence the computer storage is minimal; we need only to store the nonzero entries of the sparse matrix, if necessary, and a few number of vectors. Consequently, the structures of a matrix, for instance, sparsity and storage format, can be fully exploited. On the other hand, however, their convergence rate may be slow. Also, they may not be as robust as direct methods. A standard way to improve both the efficiency and robustness is preconditioning; a powerful technique to improve the condition number of linear systems which often determines the convergence rate of Krylov subspace methods. In fact, the development of effective preconditioning techniques helps bring the iterative methods into widespread use.

There is a wide spectrum of preconditioners. On one extreme of the spectrum, there are *black box* solvers; they are purely algebraic and do not require any information of the problem from which the linear system arises. Moreover, they are often quite robust with respect to variations in the coefficients of the PDEs. However, their convergence may deteriorate as the problem size increases. On the other extreme, preconditioners can be designed for a specific class of problems and may require the analytic and geometric information of the problem being solved. They are often characterized by the optimal convergence rate which is independent of the mesh size. However, their convergence may depend on the PDE coefficients and special techniques are needed for specific problems. Preconditioners between the two extremes, which may be known as *gray box* solvers, apply techniques of one extreme to the other and hence possess a mixture of their advantages and disadvantages. In fact, one of the goal of this dissertation is to bridge the gap between the two extremes. In particular, we enhance the purely algebraic preconditioners for specific and yet important problem classes, and on the other hand, generalize the problem-specific preconditioners to a broader setting.

Apart from preconditioning, there are other issues in employing iterative methods. For example, unlike LU factorization, iterative methods are not readily suitable for solving linear systems with multiple right-hand sides which arise frequently in various applications.

In this dissertation, we address the issues of iterative methods and the tradeoff between generality, parallelism and robustness (fast convergence) for large scale scientific computing. We intentionally provide a good mix of theory, algorithm and application. As the main theme of the dissertation, we introduce and analyze three of the most useful and interesting scalable preconditioners ranging from algebraic and problem-independent to analytic and problem-specific preconditioners. Applications to other interesting problems such as image processing are also studied. In addition, we analyze a class of projection methods for solving multiple right-hand side problems and propose a generalization to it.

1.2 Scalable and Multilevel Preconditioners

Although there is a broad number of topics, this dissertation has a unifying theme–scalable and multilevel methods.

Classical iterative methods such as Jacobi, Gauss-Seidel and SOR were evolved from the algebraic extreme of the spectrum, where a matrix is approximated by the diagonal or the triangular parts of itself. A more sophisticated and powerful example is the *incomplete LU factorization* preconditioners [34, 98], where the L, U factors of a matrix are approximated by sparse matrices. Since their algorithms resemble Gaussian elimination, they possess the robustness of direct methods. However, their convergence may deteriorate with the problem size. Although there are some analyses regarding the existence and convergence of ILU preconditioners for M-matrices [52, 98, 125], the theoretical aspect is still an open issue. Furthermore, their sequential character makes them hard to parallelize on modern computer architectures. Nevertheless, there is a practical need for general purpose preconditioning techniques, since linear systems arising from applications often do not have many structures other than sparsity to exploit and for those problems, specialized preconditioners may fail. Efforts have been made on the parallelization aspect for the symmetric case, the incomplete Cholesky factorization, for example [82]. In Chapter 2, we generalize the parallel implementation to the ILU(0) factorization for general nonsymmetric matrices with a symmetric structure. We also discuss how to tactically add nonzeros to the approximate L and U factors to improve the quality of the ILU(0) preconditioners.

Another class of purely algebraic methods is the *sparse approximate inverse* preconditioners [7, 8, 89]. They construct a sparse approximation to the inverse of a matrix, instead of a sparse approximation to the matrix itself as the ILU factorizations do. They have not received much attention until recently, where their intrinsic parallelism is recognized. Robust algorithms and theory are still under development. Nevertheless, their potential as an alternative to ILU preconditioners has drawn a lot of attention recently. In contrast to ILU preconditioners, they are designed for parallelism. Yet they have to face other issues such as how to determine an effective sparsity pattern *a priori*, and how to maintain parallelism if the sparsity pattern is chosen adaptively. More importantly, there is a potential weakness of sparse approximate inverse preconditioners for solving matrices arising from PDE problems since these inverses are dense and may not be accurately approximated by sparse matrices. In Chapter 3, we improve their efficiency for PDE problems by incorporating the wavelet techniques [41]. These wavelet sparse approximate inverse preconditioners can be considered as *gray box* solvers belonging to the middle of the spectrum, since they are specialized for linear systems arising from differential equations.

On the other extreme, preconditioners can be designed for a specific class of problems and may require the analytic and geometric information of the problem being solved. For example, an important class of problems in scientific computing is second order self-adjoint elliptic PDEs. Given the fact that a good discretization method retains the essential properties of the PDEs by the discretization matrix, preconditioners in this category must be able to uncover and exploit the hidden properties, for instance, symmetry, positive definiteness, etc, to obtain an efficient solution procedure. In particular, a well-known fact about elliptic PDEs is that the solution value at a point is influenced by all the values, far and near, of the given right-hand side function. Thus, there must be an efficient mechanism in the numerical methods to pass around information, and a key technique is the multilevel approach.

Multilevel methods can be obtained by a change of basis which possesses a multilevel structure, for instance, the hierarchical basis for the hierarchical basis preconditioners [103, 150] and the wavelet basis for the wavelet sparse approximate inverse preconditioners [33] mentioned above. Another approach to obtain multilevel methods is by a hierarchy of grids, for instance, *multigrid*. Multigrid [73] and domain decomposition [121] methods are among the most efficient solution approaches for self-adjoint elliptic PDEs. They are often characterized by their mesh and subdomain size independent convergence rate. Optimal convergence analysis for smooth coefficient problems is quite well established in the literature [13, 14, 15, 73, 148, 149]. Moreover, algorithms and theory have also been extended to solve nonsymmetric and indefinite problems [17, 94, 148]. Despite that the current trend of multigrid research has been migrating to nonsymmetric problems, there are yet many issues of multigrid for solving self-adjoint problems which are either less known, not understood, or just left behind.

Multigrid methods can be extremely slow for PDEs whose coefficients are discontinuous [1, 46, 86], oscillatory [54, 93, 128], or anisotropic [74], although they have been proved successful for smooth coefficient problems. Besides, multigrid methods for unstructured grid computations are not easy to design. The main issue is that there is no natural choice of coarse grids [31, 70, 96, 90]. Another is how to define interpolation between grids [29, 32, 35, 134]. Also, multigrid methods for nonelliptic or integral equations [17, 22, 75], whose analytical properties are entirely different from differential operators, are not well understood and usually applicable to some special cases only. In practice, however, this kind of problems are quite common, and hence there is a need to design robust multigrid methods on general computational domains in any dimensions. The challenge of multigrid is to construct appropriate interpolations and smoothers for these problems. Besides, the parallelization is also a practical open issue for multigrid.

Chapters 4-8 mark the highlight of this dissertation. We address all the above issues and present solutions to some of them. In particular, we propose and analyze new multigrid techniques in smoothing, coarsening and above all interpolation, to obtain robust and efficient multigrid methods. In Chapter 5, we present a new class of sparse approximate inverse smoothers which are intrinsically parallel and can improve the robustness of multigrid for anisotropic problems without determining the direction of anisotropy. In Chapter 6, we propose an interface preserving coarsening technique to solve discontinuous coefficient problems. Special coarsening technique specific for these problems has not been studied in the literature, which turns out to be useful and effective. Chapter 7 is the climax of our multigrid work. We propose and analyze a new robust interpolation based on the concepts of energy minimization and approximation. The analysis for one dimension proves a convergence rate independent of the PDE coefficients and the mesh size, which has not been shown in the literature before. These techniques shed new light on the design of robust multigrid. In Chapter 8, we start a promising attempt to apply multigrid with preconditioned conjugate gradient smoothing to solve differential-convolution equations, and in particular, PDE-based image problems.

1.3 Content of this Dissertation

This dissertation contains a number of works on different aspects of iterative methods; some are more in-depth (Chapter 7, 9) and some just touch the surface of the problem (Chapter 8). The central parts are Chapters 2–7, in which we study three scalable preconditioners across the algebraic to problem-specific spectrum: parallel ILU, wavelet sparse approximate inverse and multigrid. The application and multiple right-hand side issues are discussed in Chapter 8 and 9. Figure 1.1

gives a pictorial overview of the various work on iterative methods. A more detailed account of the dissertation is given as follows.

Parallel Incomplete LU Factorizations

ILU preconditioners have been extensively used in a broad range of applications. They resemble Gaussian elimination but control the amount of fill-in. Let the allowable fill-in positions be given by an index set $\mathcal{I}$. Then

$$l_{i,j} = 0 \quad \text{and} \quad u_{i,j} = 0 \quad \text{if} \quad (i,j) \notin \mathcal{I},$$

where $L = (l_{i,j})$ and $U = (u_{i,j})$ are the incomplete factors. If $\mathcal{I}$ is chosen to be the nonzeros of A, the resulting incomplete factorization is called ILU(0), whose algorithm of computing the incomplete L and U factors can be obtained by a simple modification of Gaussian elimination. Thus, the computation is done column by column, which is very sequential. Also, in each preconditioning step, a sequential forward and backward solve are performed. These generic sequential steps are unavoidable, unless the matrix possesses further structures that allow parallelism.

A common technique to parallelize sequential methods is multicoloring. That is, we choose an ordering based on a coloring of the matrix graph. In such ordering, the unknowns with the same color form a block variable where we can perform a block incomplete factorization in parallel. Jones and Plassmann [83] proposed a numerically scalable coloring scheme and developed a scalable incomplete Cholesky factorization preconditioner for symmetric matrices. Incomplete Cholesky factorization, however, is not enough since many applications are nonsymmetric. On the other hand, general ILU factorization is difficult because the memory allocations and message passing patterns are hard to determine *a priori* without symmetry. In Chapter 2, we make a trade-off between efficiency and generality and develop a scalable ILU preconditioner for general matrices with a symmetric structure, which include most matrices arising from finite element calculations.

The quality of ILU(0) may not be sufficient to yield a fast convergence. More accurate factorizations are often needed. One approach is to allow more fill-in in the incomplete factors based on the concept of *level-of-fill* [142]. The resulting ILU(k) preconditioner, where k corresponds to the level-of-fill, is even harder to implement efficiently since the nonzero structures of the incomplete factors are not predictable for $k > 0$ and hence the level-of-fill is not known until we have performed the factorization. Thus, the memory allocations have to be done dynamically which will result in a very inefficient process. An important observation is that the legitimate fill-in positions of ILU(1) introduced by eliminating the ith node have to be neighbors of an entry in row i. Similar observation can be generalized to ILU(k). Based on this observation, we develop a scalable ILU(1) algorithm. Numerical results on the IBM SP2 show good scalability and performance.

Wavelet Sparse Approximate Inverses

Although scalable ILU is possible, there is still a trade-off between parallelism (number of colors) and convergence rate. This motivates the recent interest in sparse approximate inverse preconditioning. Benson [7] and Frederickson [8] obtained a preconditioner M by considering a minimization problem:

$$\min_{M \in S} \|AM - I\|_F^2,$$

where S is a class of sparse matrices, for instance, matrices with fixed sparsity pattern. The above minimization problem is equivalent to the following system of minimization problems:

$$\min_{m_j \in S_j} \|Am_j - e_j\|_2, \qquad j = 1, \dots, n,$$

where m_j is the jth column of M and S_j is the collection of the jth column of matrices in S. Each least squares problem is independent and can be solved in parallel. Fast convergence results for Harwell-Boeing matrices have been reported by Grote and Huckle [67] and Chow and Saad [37].

A weakness of sparse approximate inverses for elliptic PDEs is the assumption that the inverse matrix A^{-1} can be well approximated by a sparse matrix M. In Chapter 3, we show that for this class of matrices, their inverse entries need not have rapid decay, and hence sparse approximate inverses do not work. As discussed in the previous section, the lack of multilevel structure is the core failure of sparse approximate inverse preconditioners for elliptic PDEs. The crucial observation we made is that these inverses are often piecewise smooth. Hence we consider using a multi-scale method based on wavelets (e.g. Daubechies wavelets [40]) to capture the local smoothness while retaining scalable efficiency. Moreover, the resulting preconditioners benefit from the multilevel structure built in the wavelets. Our new minimization problem becomes:

$$\begin{aligned} & \min_{M \in S} \|AM - I\|_F^2 \\ = \; & \min_{M \in S} \|WAW^TWMW^T - I\|_F^2 \\ = \; & \min_{\tilde{M} \in \tilde{S}} \|\tilde{A}\tilde{M} - I\|_F^2 \\ \Leftrightarrow \; & \min_{\tilde{m}_j \in \tilde{S}_j} \|\tilde{A}\tilde{m}_j - e_j\|_2^2, \qquad j = 1, \dots, n, \end{aligned}$$

where W denotes an orthogonal wavelet transform. The symbols $\tilde{A}$, $\tilde{M}$, $\tilde{S}_j$ and $\tilde{m}_j$ denote the representation of the corresponding matrices or vectors in the wavelet space. Due to the localization and approximation properties of wavelets, the sparse approximation $\tilde{M}$ in the wavelet space gives a better approximation to $\tilde{A}$ than M to A, thus producing a more effective preconditioner for a larger class of problems where the matrices have a piecewise smooth inverse whose singularities clustered around the diagonal. Besides, the storage for the preconditioner is also dramatically reduced by the wavelet compression.

Multigrid

Elliptic PDEs constitute a significant part of scientific computing. Multigrid methods are among the most efficient solution procedures for linear systems arising from the discretization of self-adjoint elliptic PDEs. They consist of two main steps: smoothing and coarse grid correction. The smoothing process, usually carried out by a relaxation method, damps away the high energy error components. The coarse grid correction process, carried out by an interpolation and a coarse grid solve, solves the low energy components on the coarser grids. The idea is that the combination of the two will result in a significant error reduction independent of the problem size and hence lead to a fast solution procedure. Moreover, we gain efficiency since the coarse grid solve is less expensive than the fine grid one.

There is a vast amount of literature on the optimal convergence of multigrid methods; see e.g. [13, 14, 15, 73, 97, 121, 148, 149]. The classical analyses, however, usually ignore the effects of the PDE coefficients on the convergence of multigrid methods. In fact, standard multigrid may converge slowly for coefficients which are discontinuous, oscillatory or near singular.

To minimize the influence of the coefficients, we must be able to capture the low energy error components during interpolation so that the coarse grid correction is effective. In other words, the range of the interpolation operator must span the low energy subspace. Mathematically, the coarse grid basis functions given by the interpolation must satisfy two inequalities [148]:

$$\begin{aligned} \|Q_1 v\|_A^2 + \sum_{k=2}^{J} \|(Q_k - Q_{k-1})v\|_A^2 &\leq C_0 \|v\|_A^2, && (1.1) \\ \|(Q_k - Q_{k-1})v\| &\leq C_1 h_k \|Q_k v\|_A, \qquad \forall k > 1, && (1.2) \end{aligned}$$

where $V_1 \subset V_2 \cdots \subset V$ is a sequence of subspaces of the fine grid space V, $Q_k : V \to V_k$ is a linear operator onto V_k, and C_0 and C_1 are constants independent of the mesh size h_k. The inequalities (1.1) and (1.2) are sometimes known as the stability and the approximation inequalities, respectively. Applying the energy minimization and constant preserving principles, we study an energy-minimizing interpolation for multigrid methods in Chapter 7. In one dimension, we can prove that the convergence rate is indeed independent of the PDE coefficients and the mesh size. In two dimensions, numerical experiments also demonstrate that the convergence is robust with respect to the PDE coefficients.

Another issue of multigrid is smoothing. The Gauss-Seidel smoothers are effective but they are not parallel in their original form; the Jacobi smoothers, on the other hand, are parallel but they are less effective than Gauss-Seidel. Moreover, both methods are poor for anisotropic and discontinuous coefficient problems. In Chapter 5, we combine the sparse approximate inverse (SAI) techniques to obtain a robust and parallel smoother. Sparse approximate inverses may not be efficient preconditioners for elliptic PDEs due to a lack of multilevel structure. However, because of their local nature, they are in fact very efficient smoothers. Moreover, by adjusting the density of the approximate inverses, we are able to improve the quality of the SAI smoother for anisotropic and discontinuous coefficient problems. This flexibility feature makes the SAI smoothers more robust than the Gauss-Seidel and Jacobi smoothers.

Other than smoothing and interpolation, coarsening can also play a role in designing robust multigrid. For example, semicoarsening is a useful techniques for anisotropic problems [48, 124]. In Chapter 6, we demonstrate that coarsening can also be used to improve multigrid convergence for discontinuous coefficient problems. The key observation is based on both experience and theory. Intuitively, the parts of the solution on the regions of different constant coefficients behave independently and are glued together through a Neumann boundary condition on the interfaces. Theoretically, convergence analyses for interface problems often require that the discontinuities align with all coarse grids. Based on these observations, we derive an interface preserving coarsening algorithm so that the interfaces are aligned with the coarse grids, or as closely as possible, allowing the simple linear interpolation to be good enough to approximate errors on the coarse grids. Another explanation for the success of the interface preserving coarsening is that it improves the constant C_1 in the approximation inequality (1.2) since the finer grid functions can be correctly approximated by the coarser grid.

Application: Multigrid for Differential-Convolution Equations

One way to understand multigrid, especially the smoothing part, is based on the spectral property of the discrete elliptic differential operators, of which the eigenfunctions corresponding to small (large) eigenvalues coincide with the low (high) frequency sine/cosine functions. Unfortunately, this characteristic may not hold for other operators. For example, in PDE-based image processing, we often need to solve differential-convolution equations of the form:

$$\alpha R(u)(x) + \int_\Omega k(x-y)u(y)dy = f(x) \text{ in } \Omega, \tag{1.3}$$

where $u(x)$ is the recovered image, $k(x)$ is the kernel convolution function, $R(u)$ is a regularization functional often represented by an elliptic differential operator, and α is a positive parameter.

The challenge of multigrid for this kind of problems is that the spectral property of the convolution operator $k(x)$ is opposite to that of the elliptic differential operators. In fact, for convolution operators, the eigenfunctions corresponding to small (large) eigenvalues are the high (low) frequency sine/cosine functions. Thus, standard smoothers do not have the desired smoothing effect. This problem did not seem to have been studied before in the multigrid literature. In Chapter 8, we propose to use fast transform based preconditioned conjugate gradient methods as smoothers. For example, the fast cosine transform preconditioner [21] is obtained from the solution of the following minimization problem:

$$\min_{C \in \mathcal{C}} \|C - A\|_F,$$

where $\mathcal{C}$ is the class of matrices diagonalizable by the discrete cosine transform. It can be proved that the fast transform preconditioners are effective for convolution operators of certain types [23, 24, 26] as well as the Laplacian operator [20]. As a result, the resulting multigrid is efficient for $R(u) = I$, the identity operator, and $R(u) = \Delta$, the Laplacian operator. The result for $R(u) = -\nabla \cdot (\nabla u / |\nabla u|)$, which arises from the total variation denoising [27, 111], are not as good as the other two cases, but still satisfactory.

Multiple Right-Hand Sides

Multiple right-hand side problems arise frequently in various applications, for instance, wave scattering problems [123], time marching methods for PDEs [57] and structural mechanics problems [56]. Efficient solution procedure is important when the number of right-hand sides is large. Direct methods such as LU factorization do not have any difficulty for this kind of problems since the additional right-hand sides can be solved efficiently using the existing L and U factors. Iterative methods, however, have no such advantage, and a separate iterative solution procedure is needed for each right-hand side. The problem is that the desire for simplicity in iterative methods dictates keeping only a minimal amount of data sufficient for the convergence of the iterations. Suppose all the right-hand sides are available at the same time. The challenge is to extract and collect the information obtained from each iteration using only a small amount of additional storage. Various solution approaches including projection methods [123], block methods [101] and hybrid methods [119]) have been proposed to find a better alternative for solving

$$Ax^{(j)} = b^{(j)}, \qquad j = 1, \ldots, N,$$

where A is symmetric and positive definite. For example, Smith, Peterson and Mittra [123] devised a single seed method for an electromagnetic scattering application. It belongs to a class of Galerkin projection methods [104, 105, 114, 133] which project the unsolved right-hand sides, $b^{(j)}$, $j \neq s$, onto the Krylov subspace generated by the seed right-hand side, $b^{(s)}$. In Chapter 9, we analyze this method and observe two fundamental properties. One is the superconvergence of the conjugate gradient refinement process. It comes from the fact that Krylov subspace captures the extreme eigenvalues so that the effective spectrum of A is reduced after projection. Another is the efficient solve for continuously varying right-hand sides, which comes from the automatic exploitation of near rank deficiency of the right-hand side matrix $B = [b^{(j)}]$. We make these statements more precise and prove them in Chapter 9. Furthermore, we generalize their idea and propose a block extension which enjoys the fast convergence of the block conjugate gradient method while preserving the two desirable features.

List of Publications from this Dissertation

Tony F. Chan, Wei-Pai Tang and W. L. Wan. Wavelet Sparse Inverse Preconditioning. *BIT, 37:644–660, 1997.*

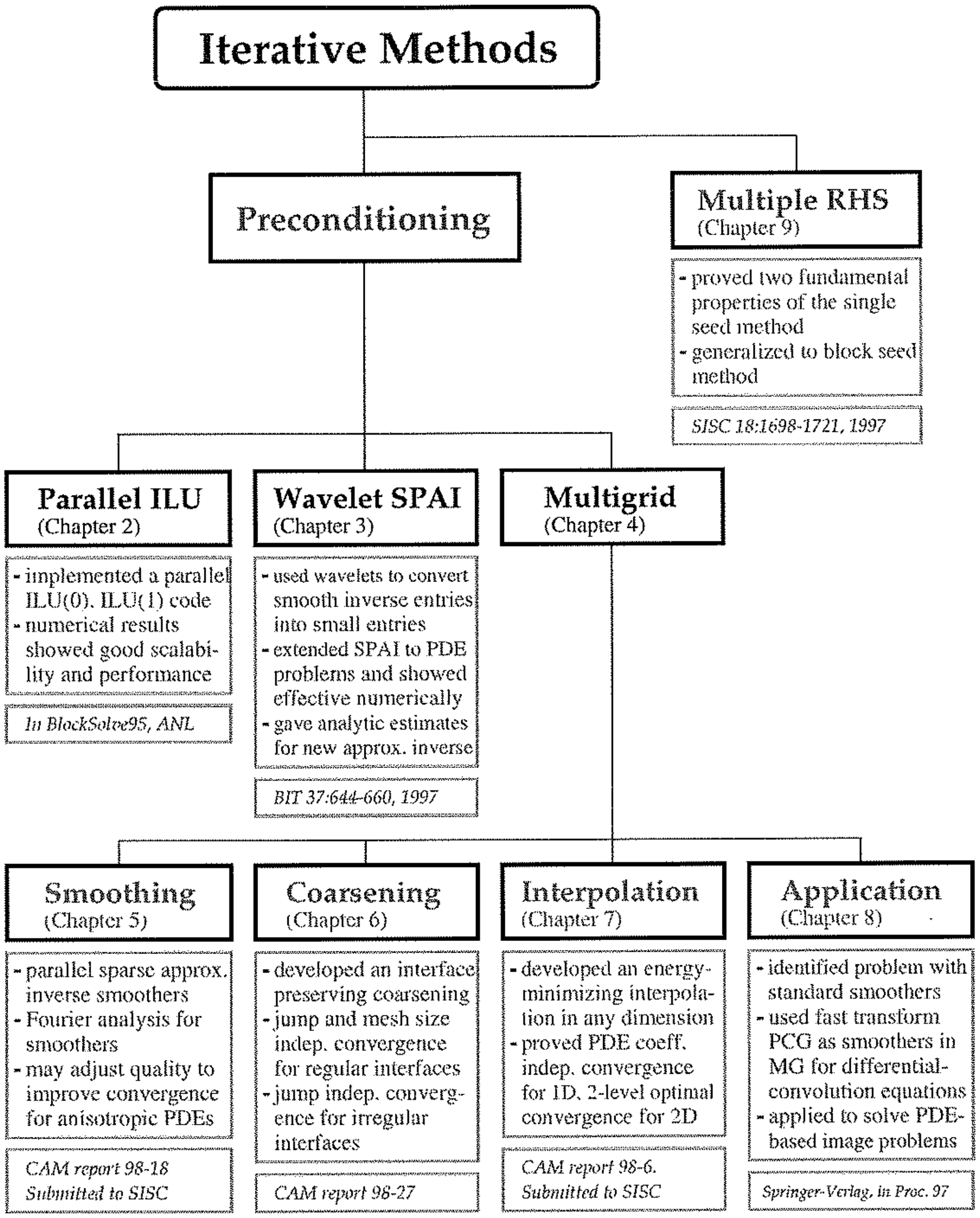
Wei-Pai Tang and W. L. Wan. Sparse Approximate Inverse Smoother for Multigrid. *CAM report 98-18, Dept. of Mathematics, UCLA, 1998.* Submitted to SIAM J. Sci. Comp.

W. L. Wan. Interface Preserving Coarsening Multigrid for Elliptic Problems with Highly Discontinuous Coefficients. *CAM report 98-27, Dept. of Mathematics, UCLA, 1998.*

W. L. Wan, Tony F. Chan and Barry Smith. An Energy-Minimizing Interpolation for Robust Multigrid Methods. *CAM report 98-6, Dept. of Mathematics, UCLA, 1998.* Submitted to SIAM J. Sci. Comp.

Raymond Chan, Tony F. Chan and W. L. Wan. Multigrid for Differential-Convolution Problems Arising from Image Processing, *Proceedings of the Workshop on Scientific Computing, March 10-12, 1997, Hong Kong, Springer-Verlag.*

Tony F. Chan and W. L. Wan. Analysis of Projection Methods for Solving Linear Systems with Multiple Right-hand Sides. *SIAM J. Sci. Comp., 18:1698–1721, 1997.*



Iterative Methods
Preconditioning
Multiple RHS
(Chapter 9)
- proved two fundamental properties of the single seed method
- generalized to block seed method
SISC 18:1698-1721, 1997
Parallel ILU
(Chapter 2)
- implemented a parallel ILU(0), ILU(1) code
- numerical results showed good scalability and performance
In BlockSolve95, ANL
Wavelet SPAI
(Chapter 3)
- used wavelets to convert smooth inverse entries into small entries
- extended SPAI to PDE problems and showed effective numerically
- gave analytic estimates for new approx. inverse
BIT 37:644-660, 1997
Multigrid
(Chapter 4)
Smoothing
(Chapter 5)
- parallel sparse approx. inverse smoothers
- Fourier analysis for smoothers
- may adjust quality to improve convergence for anisotropic PDEs
CAM report 98-18
Submitted to SISC
Coarsening
(Chapter 6)
- developed an interface preserving coarsening
- jump and mesh size indep. convergence for regular interfaces
- jump indep. convergence for irregular interfaces
CAM report 98-27
Interpolation
(Chapter 7)
- developed an energy-minimizing interpolation in any dimension
- proved PDE coeff. indep. convergence for 1D, 2-level optimal convergence for 2D
CAM report 98-6.
Submitted to SISC
Application
(Chapter 8)
- identified problem with standard smoothers
- used fast transform PCG as smoothers in MG for differential-convolution equations
- applied to solve PDE-based image problems
Springer-Verlag, in Proc. 97

Figure 1.1: Dissertation Overview

CHAPTER 2

A Parallel ILU Preconditioner

2.1 Introduction

Incomplete LU factorization (ILU) preconditioners have been widely used in a broad class of applications for its simplicity and robustness. ILU is essentially the Gaussian elimination but it controls the amount of fill-in (cf. Section 1.3). The idea of ILU is based on eliminating the entries only in the specified locations. The purpose is to avoid introducing extra nonzeros (fill-in) during elimination. More detail discussion of incomplete factorizations can be found, for example, in the books by Axelsson [3] and Saad [116], or in the survey paper by Chan and van der Vorst [34]. The parallelization aspects can be found in the survey paper by Demmel, Heath and van der Vorst [45].

A serious drawback of using ILU on parallel computer architectures is that the preconditioner is sequential in two ways. First, the incomplete factorization steps are basically Gaussian eliminations which are to be done column by column. Second, in each preconditioning step, a forward and backward solve are performed which are inherently sequential.

A common technique to parallelize sequential methods is multicoloring, or simply coloring. In other words, we choose an ordering based on a coloring of the graph representing the nonzero adjacency structure of the matrix. The idea is to assign a color to each unknown so that unknowns of the same color do not connect to each other with respect to the adjacency graph. Thus we may consider unknowns of the same color as a block of variable where we can perform a block incomplete factorization in parallel. Similarly, in each preconditioning step, the forward and backward solve can also be done in parallel for each color. Hence fewer colors results in more parallelism. In general, for matrices arising from physical models, the maximum degree of any node, which bounds the number of colors used (cf. Lemma 2.1), is bounded independently of the total number of nodes. Under this assumption, Jones and Plassmann [83] derived a scalable coloring heuristic which will be described in Section 2.2.

A scalable coloring scheme is not sufficient for scalable performance. One must also achieve good computation rates on each node. A standard implementation of a sparse matrix times vector multiplication does not necessarily exhibit good data locality and probably uses a large amount of indirect addressing. To improve these, we can take advantage of the special local structure typical in structural mechanics, for example. These matrices usually have large, dense cliques and can be easily identified. The sparse operations involving these cliques can utilize dense level 2 and 3 BLAS. Combining all these, Jones and Plassmann [84, 82] developed a scalable incomplete Cholesky factorization preconditioner for symmetric matrices. We extended the idea to the nonsymmetric case to obtain a scalable ILU preconditioner; see Section 2.3. The challenge is to keep track of the symmetry assumption used in the algorithm and to modify them accordingly to allow for the usual LU factorization.

The accuracy of the ILU(0) preconditioner discussed above may not be sufficient to yield an adequate rate of convergence for some applications. More accurate incomplete LU factorizations are sometimes needed. One way is to allow more fill-in. In [142], Watts introduced the concept of *level of fill-in*. In ILU(1), for instance, we also allow fill-in at the entries which are some of the neighbors of the nonzeros in ILU(0). In the following sections, we describe a scalable coloring heuristic used for the construction of a scalable ILU(0) preconditioner. Then we discuss the difficulties of implementing ILU(1) and how we resolve them. Numerical results on the SP2 are presented and finally we conclude this section by several remarks.

2.2 A Scalable Coloring Heuristic

In this section, we summarize the parallel coloring heuristic derived by Jones and Plassmann [83] which is to be used for the incomplete LU factorization. Given a structurally symmetric matrix A, the symmetric graph of A is defined as: $G(A) = (V, E)$, where $V = \{1, \ldots, n\}$ is the set of vertices and $E = \{(i,j) : A_{i,j} \neq 0, i \neq j\}$ is the set of edges. The function $\sigma : V \rightarrow \{1, \ldots, s\}$ is an *s-coloring* of $G(A)$, if $\sigma(i) \neq \sigma(j)$ for all edges $(i,j) \in E$. The chromatic number of $G(A)$, denoted by $\chi(G)$, is the minimum possible value for s.

The goal is to color the graph using $\chi(G)$ number of colors in an efficient way. Unfortunately, to determine whether a graph is s-colorable is NP-complete [59], not to mention finding the way to color it. Nevertheless, we can easily find an upper and lower bound for it.

Lemma 2.1 *Let* $\Delta(G) = \max_{i \in V} deg(i)$*, where* $deg(i) =$ *degree of vertex* $i \in V$. *Then* $\chi(G) \leq \Delta(G) + 1$.

The sequential (greedy) heuristic, Algorithm 2.1, always satisfies the upper bound above.

Let V' be a subset of V. The induced graph $G'(V') = (V', E')$, of G is given by the vertex set V' and the edge set $E' = \{(i,j) \in E : i, j \in V'\}$. An r-clique is a complete subgraph of G of size r. Since the r vertices in an r-clique must be assigned different colors, we have the following lower bound.

Lemma 2.2 *If* $G(A)$ *contains an* r*-clique, then* $\chi(G) \geq r$.

```
Algorithm 2.1: A Sequential Coloring Heuristic
$V' \leftarrow V$
for $i$=1, . . ., $n$
   choose $v_i \in V'$
   $\sigma(v_i)$ = smallest available consistent color
   $V' \leftarrow V' \backslash \{v_i\}$
end for
```

For many applications, it has been demonstrated that a number of sequential coloring heuristics [38, 84] are able to find colorings which are only one or two colors off the optimal one.

```
Algorithm 2.2: A Parallel Coloring Heuristic
$V' \leftarrow V$
while $V' \neq \emptyset$
   choose an independent set $I$ from $V'$
   color $I$ in parallel
   $V' \leftarrow V' \backslash I$
end while
```

A parallel version of Algorithm 2.1 can be obtained by choosing an independent set I from V' in place of a single node v_i and color I independently; see Algorithm 2.2. Thus the key issue now is to select an independent set in parallel. Luby proposed an Monte Carlo algorithm for determining I based on the following rule:

1. For each $v \in V'$, determine a distinct, random number $\rho(v)$.

2. $v \in I \Leftrightarrow \rho(v) > \rho(w) \quad \forall w \in \text{adj}(v)$.

The resulting algorithm is shown in Algorithm 2.3.

Jones and Plassmann improved the global synchronous step occurring at each new choice of random numbers and derived an asynchronous version of it; see Algorithm 2.4. In this algorithm, it is assumed that each v is assigned to a different processor.

It turns out that the asynchronous algorithm can also be applied to the general situation where each processor possesses a subset of nodes. Let V^S be the set of subdomain boundary nodes, i.e. $v \in V^S$, if v is adjacent to vertices belonging to different processors. Then the scalable coloring heuristic is performed in two main steps:

1. Color $G(V^S)$ using the asynchronous Monte Carlo heuristic.

Algorithm 2.3: A Parallel Heuristic for Determining MIS

```
$I \leftarrow \emptyset$
$V' \leftarrow V$
$G' \leftarrow G$
while $G' \neq \emptyset$
  choose an independent set $I'$ in $G'$
  $I \leftarrow I \cup I'$
  $N(I') \leftarrow$ set of neighboring nodes of $I'$
  $X \leftarrow I' \cup N(I')$
  $V' \leftarrow V'/X$
  $G' \leftarrow G(V')$
end while
```

Algorithm 2.4: An Asynchronous Parallel Coloring Heuristic

```
choose $\rho(v)$
n-wait = 0
send-queue = $\emptyset$
for each $w \in adj(v)$
  send $\rho(v)$ to processor responsible for $w$
  receive $\rho(w)$
  if $\rho(w) > \rho(v)$ then n-wait=n-wait+1
  else send-queue $\leftarrow$ send-queue $\cup\{w\}$
end for
n-recv = 0
while n-recv < n-wait
  receive $\sigma(w)$
  n-recv = n-recv+1
end while
$\sigma(v)$= smallest available color consistent with the previously
          colored neighbors of $v$
for each $w \in$ send-queue
  send $\sigma(v)$ to processor responsible for $w$
end for
```

2. On each processor, color the interior nodes using the colored nodes on the boundary obtained from Step 1.

2.3 A Scalable ILU(0) Algorithm

The idea of incomplete factorization is that the Gaussian elimination steps are only performed at the locations allowed so that the nonzeros of the incomplete L, U factors are restricted to those positions only. The advantage is that we can make the L, U factors sparse and hence the forward and backward solve can be performed more efficiently in the preconditioning steps. More precisely, let the allowable fill-in positions be given by the index set $\mathcal{I}$, i.e.

$$l_{i,j} = 0 \quad \text{and} \quad u_{i,j} = 0 \quad \text{if} \quad (i,j) \notin \mathcal{I}, \tag{2.1}$$

where $L = (l_{i,j})$ and $U = (u_{i,j})$. For ILU(0), the choice for $\mathcal{I}$ is the nonzeros of A:

$$\mathcal{I} = \{(i,j) : a_{i,j} \neq 0\}.$$

Let $M = LU$ be the resulting preconditioner from ILU(0). It must be a good approximation to A in some way. A common criterion is that

$$m_{i,j} = a_{i,j} \quad \text{if} \quad (i,j) \in \mathcal{I}. \tag{2.2}$$

The conditions (2.1) and (2.2) determine the incomplete L, U factors uniquely. The calculation can be implemented by a modification of the Gaussian Elimination; see Algorithm 2.5. Note that it is exactly Gaussian Elimination if the update for $a_{i,j}$ in the inner-most loop is not restricted by the index set $\mathcal{I}$.

Algorithm 2.5: ILU

```
for $k$=1, ..., $n-1$
   for $i$=$k+1$, ..., $n$
      if $(i,k) \in \mathcal{I}$ then
         $a_{i,k} = a_{i,k}/a_{k,k}$
         for $j$=$k+1$, ..., $n$
            if $(i,j) \in \mathcal{I}$ then
               $a_{i,j} = a_{i,j} - a_{i,k}a_{k,j}$
            end if
         end for
      end if
   end for
end for
```

The construction of ILU(0) is sequential since the updates of $a_{i,j}$ in the innermost loop depends on the value of $a_{i,k}$, $k < i$, which is not fixed, in general, until the elimination process proceeds to column k. A parallelization of ILU(0) is based on coloring, for example, the scalable coloring heuristic described in Section 2.2. After the matrix is reordered according to the assigned colors, the diagonal block of the block of rows with the same color will be a diagonal matrix. Thus we may update all these rows simultaneously. Similarly, the forward and backward solve in the preconditioning steps can also be done in parallel for each color. The parallel ILU(0) algorithm is summarized in Algorithm 2.6.

Algorithm 2.6: A Scalable ILU(0)

1. Construct a data structure for factorization
2. Color the variables
3. Perform the incomplete factorization

Step 1 is a technical preprocessing procedure which is described as follows. In our implementation, the matrix A is stored by rows, which is a very common format (CSR [116]) for sparse matrices. An advantage is that the common matrix operation, matrix-vector multiply , which is essentially a sequence of rowwise inner products, can be done efficiently. However, a disadvantage is that the positions of the nonzeros down a column may not be available directly. It causes a serious problem when we calculate the multipliers and the Schur complement during the incomplete factorization since these operations need to work on both the rows and columns. We solve the problem by constructing an additional data structure which stores the matrix A by columns. After that, the coloring and factorization can be proceeded as described above.

As mentioned in the introduction, it is not enough to achieve scalable performance by coloring. One must also be able to achieve good computation rate on each processor. Data locality and avoiding indirect addressing are two of the issues. For example, a standard implementation of a sparse matrix-vector multiply does not exhibit good data locality and involves great amount of indirect addressing. Note that for matrices arising from physical models, it is often that dense cliques exist in the matrix graph and they can be easily identified. We take advantage of this fact by utilizing dense level 2 and 3 BLAS in the computation. For problems involving multicomponents on each computational node, for instance, structural mechanics applications, rows corresponding to the same node often have identical structure. By exploiting this structure, we can reduce the amount of indirect addressing.

2.4 Implementation Issues of ILU(1)

For some applications, the quality of ILU(0) may not be good enough to give fast convergence. We may improve it by adding more nonzeros based on the *level of fill-in* [116]. Define the initial level of fill of an element $a_{i,j}$ by

$$\text{lev}_{i,j} = \begin{cases} 0 & \text{if } a_{i,j} \neq 0 \text{ or } i = j \\ \infty & \text{otherwise.} \end{cases}$$

When $a_{i,j}$ is modified in the incomplete factorization process, its level of fill must be updated by

$$\text{lev}_{i,j} = \min\{\text{lev}_{i,j}, \text{lev}_{i,k} + \text{lev}_{k,j} + 1\}. \tag{2.3}$$

The above procedure provides a systematic way of determining the fill-in positions of L and U. For ILU(k), we keep all the fill-in elements whose level of fill does not exceed k. In particular, for ILU(0), we keep precisely the nonzeros of A. For ILU(1), we keep those whose level of fill does not exceed 1; see Algorithm 2.7.

Algorithm 2.7: ILU(k)

```
define lev($a_{i,j}$)=0 for all nonzeros of $A$
for $l$=1, ..., $n-1$
    for $i$=$l+1$, ..., $n$
        if lev($a_{i,l}$) $\leq k$ then
            $a_{i,l} = a_{i,l}/a_{l,l}$
            for $j$=$l+1$, ..., $n$
                $a_{i,j} = a_{i,j} - a_{i,l}a_{l,j}$
                update lev($a_{i,j}$) using (2.3)
            end for
        end if
    end for
    replace any element in row $l+1$ and column $l$
    with lev($a_{i,j}$) $> k$ by 0
end for
```

For efficiency, especially in parallel computing, we often separate the incomplete LU factorization into two phases. In the symbolic phase, we determine the nonzero structures of L and U and allocate the necessary amount of memory for them. In the numeric phase, we perform the incomplete factorization as usual. For parallel ILU(0), the symbolic phase can be done easily since the nonzeros of L and U are just those of A. However, in general, the amount of fill-in and computational work for obtaining ILU(k) is not predictable for $k > 0$ since the level of fill is not known until one has performed the factorization. In other words, extra memory can only be allocated during the factorization, which will result in a very inefficient process.

Furthermore, in the factorization steps, we have already colored the graph of A. Since the coloring depends on the graph of A which is the same as the original graph of L and U, if we add extra nonzeros to L and U, the graphs of L and U will be changed and it is possible that two vertices of the same color are connected by the newly added nonzeros. One might consider reordering the graph of the expanded A. However, this will then change the extra fill-in positions which in turn require another recoloring. It might be an indefinite cycle.

We discuss a compromised solution based on the following observation [107]. The legitimate fill-in positions of ILU(1) introduced by eliminating the ith node have to be neighbors of an entry in row i. It does not mean, however, that every neighbor of an entry in row i will be a legitimate fill-in positions of ILU(1). We know which neighbors are legitimate fill-ins only when we are eliminating the ith node. This is the fundamental problem that causes the above difficulties. Our compromised solution is to allocate extra memory for all neighbors and put zeros in these potential fill-in positions. Hence the graph of A is changed. After that, we proceed Step 1-3 as in Algorithm 2.6. In this way, we always have already allocated the memory needed for the legitimate fill-ins no matter what coloring we have used in Step 2. However, we may have spent some amount of memory which have not been used in the factorization process.

2.5 A Scalable ILU(1) Algorithm

Based on the compromise discussed previously, the implementation of ILU(1) is just to add Step 0 in the ILU(0) Algorithm 2.6:

Step 0: Allocate memory for the neighbors of every entries in each row.

Now we describe Step 0 in more detail. Recall that each processor contains a sequence of disjoint rows of A. For each processor, we loop over each row and record all the column numbers in the index set $\mathcal{J}$. Then we gather all those rows with indices in $\mathcal{J}$. We collect all the column numbers in these rows and add them to the current row. The entire (complicated) process can be summarized concisely by a sentence if we think of A as a graph. That is, the new neighbors of node i are node j which are distance 1 or 2 away.

A technical problem is that the processor may not contain the rows that it needs. Moreover, those rows are not known until we come to the row that needs them. Also, the processor which contains that particular row may not be aware that some processor needs it at that moment. A solution to this situation is that each processor first looks over the rows it has and determines which rows it needs. After that, we make a global exchange of rows among processors. This procedure can be significantly simplified by the fact that the matrix A is structurally symmetric. In other words, if processor i needs k rows from processor j, processor j in turn needs k rows from processor i. Moreover, those rows are related in some way such that each processor can determine which processors have the rows it needs and the rows which other processors need in advance. This fact makes the global exchange possible and efficient.

A final remark. It seems that we eventually obtain a parallel ILU(1) algorithm. However we should caution that there is a minor difference between our algorithm and the classical one. For the classical ILU(1) algorithm, the structure of the incomplete factors L and U are the expanded version of those of ILU(0). However, in our algorithm, it may not be the case. Recall that the memory is allocated before factorization. In this way, the graph of A is changed, which implies the coloring used will be different from that of ILU(0). Consequently, the newly permuted A is probably different from the old one. Hence, the structure of the incomplete factors from ILU(0) and ILU(1) are different. Nevertheless, theoretically, we can think of the incomplete factors from ILU(1) as the expanded version of those from ILU(0) which uses the same ordering as ILU(1).

2.6 Numerical Results

In this section, we demonstrate and compare the efficiency of ILU(0) and ILU(1) in terms of factorization time, solution time and number of iterations to convergence. The testings are done in *BlockSolve95* running on IBM SP2. The model problem is a simple convection-diffusion equation:

$$-\Delta u + \beta u_x = f.$$

We solve the discretization linear system by GMRES(20) preconditioned by ILU(0) or ILU(1). The iteration stopped when the relative residual norm was less than 10^{-7}. The timings are the wall clock time recorded by MPI using the *MPI_Wtime()* command. Since the wall clock timings depend on the workload of the machines being run and some other system factors, they may not reflect accurately the time used for the operation being timed. Thus, we occasionally obtain speedups greater than the theoretical bounds.

Table 2.1 shows the factorization time of ILU(0) with different number of processors and mesh size. The speedups are relative to the time run by one processor. We see that the speedups are very close to the optimal. Moreover, for fixed problem size per processor, it requires roughly about the same amount of time to do the factorization. Hence the parallel factorization of ILU(0) described in Section 2.3 is scalable.

Table 2.2 shows the solution time of GMRES(20) preconditioned by ILU(0). The poor speedups in the solution phase is not known yet. It may be that the forward and backward solve during the preconditioning steps are not fully parallelized in *BlockSolve95*.

Table 2.3 shows the factorization time of ILU(1). We see that the speedups are similar to those of ILU(0) and are also close to the optimal. Also, like ILU(0), it requires roughly about the same amount of time to do the factorization for fixed problem size per processor. Hence ILU(1) described in Section 2.5 is also scalable. We note also that ILU(0) and ILU(1) took about the same amount of time to do the factorization.

Table 2.4 shows the solution time of ILU(1). Similar to ILU(0), the speedups are poor. However, we can see that ILU(1) generally takes less time than ILU(0) to solve the linear systems, and the saving is about 30%-50%.

Finally, table 2.5 shows that number of iterations to convergence of ILU(0) and ILU(1). The smaller number of ILU(1) iterations shows that ILU(1) indeed gives a better preconditioner than ILU(0).

	Factorization time					Speedup				
h	1	2	4	8	16	1	2	4	8	16
1/64	0.5454	0.2807	0.5223	0.3673	0.1946	1	1.94	1.04	1.48	2.80
1/128	2.321	1.178	0.6764	0.3286	0.4299	1	1.97	3.43	7.06	5.40
1/256	30.72	15.46	6.866	3.334	1.639	1	1.99	4.47	9.21	18.74
1/512	145.1	73.73	34.01	12.85	5.634	1	1.97	4.27	11.29	25.75

Table 2.1: Factorization time (sec) of ILU(0) with different number of processors and mesh size, and their corresponding speedup.

	Solution time					Speedup				
h	1	2	4	8	16	1	2	4	8	16
1/64	2.992	2.470	66.67	109.6	120.4	1	1.21	0.045	0.027	0.025
1/128	20.73	12.20	10.60	49.97	80.58	1	1.70	1.96	0.41	0.26
1/256	543.7	313.6	182.7	371.6	423.1	1	1.73	2.98	1.46	1.29
1/512	2865	1684	1226	1964	579.1	1	1.70	2.34	1.46	4.95

Table 2.2: Solution time (sec) of ILU(0) with different number of processors and mesh size, and their corresponding speedup.

	Factorization time					Speedup				
h	1	2	4	8	16	1	2	4	8	16
1/64	1.439	0.7532	0.4215	0.2521	0.5058	1	1.91	3.41	5.71	2.85
1/128	6.445	3.308	1.706	0.8870	0.9725	1	1.95	3.78	7.27	6.63
1/256	29.25	14.39	6.939	3.817	4.001	1	2.03	4.22	7.66	7.31
1/512	-	62.93	37.63	14.66	7.592	-	2	3.34	8.59	16.58

Table 2.3: Factorization time (sec) of ILU(1) with different number of processors and mesh size, and their corresponding speedup.

	Solution time					Speedup				
h	1	2	4	8	16	1	2	4	8	16
1/64	2.913	2.594	2.713	3.031	38.53	1	1.12	1.07	0.96	0.076
1/128	30.25	16.38	10.52	8.376	60.53	1	1.85	2.88	3.61	0.50
1/256	207.57	100.84	64.53	36.87	45.44	1	2.06	3.22	5.63	4.57
1/512	-	915.5	539.1	232.2	317.1	-	2	3.40	3.94	2.89

Table 2.4: Solution time (sec) of ILU(1) with different number of processors and mesh size, and their corresponding speedup.

	ILU(0)					ILU(1)				
h	1	2	4	8	16	1	2	4	8	16
1/64	163	157	180	187	155	83	85	92	97	98
1/128	257	259	277	250	253	199	177	169	173	183
1/256	561	549	538	544	518	348	317	307	271	318
1/512	*	*	*	*	*	-	740	819	646	657

Table 2.5: Number of GMRES(20) iterations. The numbers 1,2,4,8,16 indicate the number of processor. Convergence over 1000 iterations is denoted by $*$ and test fail due to insufficient memory is denoted by -.

2.7 Concluding Remarks

A simple parallel implementation of ILU(1) is derived from the existing ILU(0) algorithm. It preserves almost the same scalability of ILU(0) while possessing better quality.

In principal, we should be able to construct ILU(2) from ILU(1) and hence ILU(k) for any k. However, the compromise we have adopted by allocating excess amount of memory may be too expensive for ILU(k) when $k > 1$. Further investigation and testings are needed.

CHAPTER 3

Wavelet Approximate Inverse Preconditioners

3.1 Introduction

There have been many efforts to look for alternative robust preconditioners for ILU since the parallelization of the latter, though possible as described in Chapter 2, is quite difficult. Recently, there is an increasing interest in using sparse approximate inverse preconditioners for Krylov subspace iterative methods to solve:

$$Ax = b, \tag{3.1}$$

where A is large and sparse. On the one hand, they possess a conceptually straightforward parallel implementation. In contrast with ILU, the application of the preconditioner is simply a matrix-vector multiply instead of a forward and a backward solve and it can be done easily in parallel. On the other hand, similar to ILU preconditioners, they can be applied to both general and PDE problems due to their purely algebraic algorithm. The studies of Grote and Huckle [67] and Chow and Saad [37] showed that they are robust for a wide range of matrices in the Harwell-Boeing collection.

An approach of computing sparse approximate inverse is described by Benson [7] and Benson and Frederickson [8]. Consider the right preconditioned linear system:

$$AMy = b, \quad x = My, \tag{3.2}$$

where M is a right preconditioner. A sparse approximate inverse preconditioner M is defined as a solution to the following minimization problem:

$$\min_{M} \|AM - I\|_F^2, \tag{3.3}$$

subject to some constraint on the number and position of the nonzero entries of M. The Frobenius norm is particularly useful for parallel implementation. Notice that

$$\|AM - I\|_F^2 = \sum_{j=1}^{n} \|Am_j - e_j\|_2^2,$$

where m_j and e_j are the jth column of M and I respectively. Thus solving (3.3) leads to solving n independent least squares problems:

$$\min_{m_j} \|Am_j - e_j\|_2, \qquad j = 1, \ldots, n, \tag{3.4}$$

which can be done in parallel.

Another possibility is to use a weighted Frobenius norm which had been investigated intensively by Kolotilina, Yeremin and others [88, 89, 92]; see also the survey by Axelsson [3]. In the following, however, we focus primarily on the Frobenius norm approach. Other constructions of approximate inverse are discussed in [9, 10, 19, 36, 37, 61, 130]. A comparison of approximate inverse preconditioners and ILU(0) on Harwell-Boeing matrices can be found in [64].

In practice, it is desirable to look for sparse solutions of (3.4). However, this poses two difficulties: how to determine the sparsity pattern of M and how to solve (3.4) efficiently. Recently, two main approaches have been suggested. One is discussed by Cosgrove, Diaz and Griewank [39] and Grote and Huckle [67] and the other is by Chow and Saad [37]. For the former approach, the least squares problems (3.4) are solved by the QR factorization, which may seem costly. But since m_j is sparse, the cost of the QRF can be greatly reduced. Moreover, algorithms can be derived to determine the positions of the nonzero entries adaptively. Similar methods were also discussed in [68, 69, 88, 89, 92] where the sparsity pattern of M is typically fixed as banded or that of the nonzeros of A.

For the approach described by Chow and Saad, standard iterative methods (e.g. GMRES) are used to find an approximate solution to $Am_j = e_j$, and a dropping strategy is applied to m_j to control the amount of fill-in. The idea is to let the Krylov subspace build up the sparsity pattern gradually and then the nonzeros entries are selected automatically by size. Other adaptive searching heuristics are also proposed which will not be discussed here.

The idea of sparse approximate inverse is based on the assumption that A^{-1} can be approximated by a sparse matrix M. Thus sparse approximate inverses are particularly useful for matrices whose inverse contains only a small number of relatively large entries. For example, if the matrix is banded or diagonal dominant, the inverse entries decay away from the diagonal [12, 43, 44, 53]. However, as mentioned in [37, 39], if we require $\|AM - I\|_1 < 1$, we can always find a sparse matrix A for which the corresponding M has to be structurally dense. Another source of difficulty comes from matrices arising from elliptic PDEs with smooth coefficients, whose inverse may not necessarily have enough number of small entries so that sparse approximate inverses are effective.

The key idea is to transform A to a new basis in which A^{-1} has a sparse approximation. For example, the inverse of a discrete elliptic operators (corresponding to the discrete Green's function [99]) typically possesses some smoothness which can be converted into small wavelet coefficients. Specifically, we apply a wavelet transform to compress the piecewise smooth entries of A^{-1} and then apply the standard techniques (e.g. Grote and Huckle's implementation) to construct a sparse approximate inverse. The use of wavelets to solve integral and differential equations can also be found in [11, 19, 55, 61].

We should also mention that the factorized approximate inverse technique is another approach to deal with the case where the inverse contains only a few number of small entries. See [9, 10, 88, 89, 92] for details. An advantage of this approach is that the symmetry of matrix A can be preserved.

We also remark that even if A^{-1} has decay away from the diagonal (e.g. if A is the Laplace operator), the rate of decay may not be enough for the approximate inverse to have optimal convergence, in the sense that the number of iteration for convergence is independent of the mesh size. This is verified numerically in Table 3.1, where SPAI is the sparse approximate inverse given by Grote and Huckle's implementation [67]. The number in the bracket is the maximum allowable size of the residual norm of each column. In general, the smaller the number, the better (but also the denser) the approximate inverse is. We show numerically in Section 3.6 that our approach provides a means to improve the optimality of sparse approximate inverse preconditioners for solving elliptic PDEs. In Section 3.4, we show that the wavelet approximate inverse preconditioner is closely related to the well-known optimal hierarchical basis preconditioner.

h	no. of GMRES(20) iter			no. of nonzeros in precond.		
	SPAI(0.4)	SPAI(0.2)	ILU(0)	SPAI(0.4)	SPAI(0.2)	ILU(0)
1/8	16	10	9	208	696	288
1/16	29	17	14	1040	3640	1216
1/32	67	37	25	4624	16440	4992
1/64	160	63	57	19472	69688	20224

Table 3.1: Convergence of GMRES(20) where A=2D Laplacian.

In Section 3.2, we show how to adapt the wavelet transform in the least squares approach to solve (3.4). In Section 3.3, we justify theoretically that a smooth inverse has better decay in the wavelet basis. We also make an interesting connection between our wavelet based preconditioner and the classical hierarchical basis preconditioner. In Section 3.5, we estimate the extra cost for the wavelet transform and discuss the implementation issues of how to simplify the algorithm. In Section 3.6, we present several numerical examples to compare different methods.

3.2 Fast Wavelet Based Approximate Inverse

Since we are only interested in the application of wavelets to constructing approximate inverses, we only mention a few features of wavelets. See, for example, Daubechies' book [41] for more detail description of wavelets.

Given a set of orthogonal wavelet basis functions, there corresponds an orthogonal matrix W that transforms vectors in the standard basis to those in the wavelet basis. Furthermore, if v is a vector of smoothly varying numbers (with possibly local singularities), its wavelet representation $\tilde{v} = Wv$, will have mostly small entries. We can also represent two dimensional transforms by W. Let A be a matrix in the standard basis. Then $\tilde{A} = WAW^T$ is the representation of A in the wavelet basis. This wavelet representation $\tilde{A}$ is also called the standard form [11] of A. Nonstandard form [11] of A also exists but will not be discussed here.

Assuming A^{-1} is piecewise smooth, our idea is to apply a wavelet transform to compress A^{-1} and then use it as a preconditioner. Observe that

$$\widetilde{A^{-1}} \equiv WA^{-1}W^T = (WAW^T)^{-1} = \tilde{A}^{-1},$$

where W is an orthogonal wavelet transform matrix. Thus we can first transform A to its wavelet basis representation $\tilde{A}$ and then apply, for example, Grote and Huckle's method to find an approximate inverse for $\tilde{A}$, which is the preconditioner that we want to compute. In other words, we do not need to form A^{-1} but are still able to compute its transform.

We next show how we adapt the wavelet transform in the least squares approach. Consider equation (3.3) again. Let W be an orthogonal wavelet transform matrix, i.e. $\tilde{x} = Wx$ is the vector x in the wavelet basis. (Note that W can be 1-level or full $\log_2 n$-level wavelet transform matrix.) Then

$$\begin{aligned} \min_M \|AM - I\|_F &= \min_M \|WAW^TWMW^T - I\|_F \\ &= \min_{\tilde{M}} \|\tilde{A}\tilde{M} - I\|_F, \end{aligned} \tag{3.5}$$

where $\tilde{A} = WAW^T$ and $\tilde{M} = WMW^T$ are the representations of A and M in the wavelet basis respectively. Thus, our n least squares problems become

$$\min_{\tilde{m}_j} \|\tilde{A}\tilde{m}_j - e_j\|_2, \qquad j = 1, 2, \ldots, n. \tag{3.6}$$

Note that $\tilde{A}$ is sparse (but probably denser than A) since A is. Because of the wavelet basis representation, if M is piecewise smooth, we would expect $\tilde{M}$, neglecting small entries, to be sparse too. Therefore, the sparse solution of (3.6) would hopefully give rise to a more effective approximate inverse than the original approach without the wavelet transform. We justify our claim numerically in Section 3.6.

We summarize our algorithm as follows:

Algorithm 3.1: Wavelet Based Approx. Inverse

(a) Wavelet transform A to get $\tilde{A} = WAW^T$.
(b) Apply standard approx. inverse algorithm (e.g. SPAI) to $\tilde{A}$ and obtain $\tilde{M}$.
(c) Use $\tilde{M}$ as preconditioner to solve: $\tilde{A}\tilde{x} = \tilde{b}$, where $\tilde{b} = Wb$.
(d) Apply backward wavelet transform to $\tilde{x}$ to obtain $x = W^T\tilde{x}$.

It should be noted that if we know the sparsity pattern of the large entries of $\tilde{M}$ *a priori*, then it is more efficient and also much simpler to use that pattern to solve the least squares problems (3.6) instead of using Grote and Huckle's adaptive approach.

3.3 Theoretical Aspects

The effectiveness of the wavelet based approximate inverse preconditioners relies on the ability of wavelets to change (local) smoothness to small wavelet coefficients. In this section, we combine the classical result of Beylkin, Coifman and Rokhlin [11] and our construction to derive a residual estimate for our preconditioner.

In the discussion below, we follow the notation in [11]. Define the set of dyadic intervals on [0,1] by:

$$\mathcal{I} = \{[2^{-j}k, 2^{-j}(k+1)] : 0 \le k \le 2^j - 1, 0 \le j \le \log_2 n\}.$$

Let $I_{jk} = [2^{-j}k, 2^{-j}(k+1)] \in \mathcal{I}$. Then $|I_{jk}|$=length of I_{jk} is defined as: $2^{-j}(k+1) - 2^{-j}k = 2^{-j}$. In order to bound the size of the elements of $\tilde{A}^{-1}$ in the wavelet basis, a sufficient condition is the following smoothness assumptions on the Green's function $G(x,y)$:

$$|G(x,y)| \le \frac{1}{|x-y|}, \tag{3.7}$$

$$|\partial_x^m G(x,y)| + |\partial_y^m G(x,y)| \le \frac{C_m}{|x-y|^{m+1}}, \tag{3.8}$$

for some $m \ge 1$ and $C_m \ge 0$.

The following result of Beylkin, Coifman and Rokhlin [11] gives an estimate of the size of the entries in $\tilde{A}^{-1}$.

Theorem 3.1 *Suppose the Green's function $G(x,y)$ satisfies the smoothness assumptions (3.7) and (3.8). Let $\tilde{A}^{-1}$ be the discrete operator of $G(x,y)$ in the wavelet basis. Then the (k,l)th entry of $\tilde{A}^{-1}$ is bounded by:*

$$(\tilde{A}^{-1})_{k,l} \le C_m \left(\frac{|I_k|}{|I_l|}\right)^{\frac{1}{2}} \left(\frac{|I_k|}{d(I_k, I_l)}\right)^{m+1},$$

where $I_k, I_l \in \mathcal{I}$, $|I_k| \le |I_l|$ and $d(I_k, I_l)$=distance between I_k and I_l.

From the above bound, we can see that the lengths and positions of I_k and I_l determine the size of $(\tilde{A}^{-1})_{k,l}$. By the definition of dyadic intervals and as mentioned in [11], $d(I_k, I_l)$ is equal or close to 0 at $O(n\log_2 n)$ locations only. In other words, effectively $\tilde{A}^{-1}$ has only $O(n\log_2 n)$ number of elements for large enough n. More precisely, let $\epsilon > 0$ be given. Define a sparsity pattern $\mathcal{S}$ to be:

$$\mathcal{S} = \{(k,l) : (\tilde{A}^{-1})_{k,l} \geq \epsilon\}.$$

Then we have the following result by Theorem 3.1.

Theorem 3.2 *The number of elements in* $\mathcal{S} = O(n\log_2 n)$.

With this result, we are able to estimate the quality of our approximate inverse.

Theorem 3.3 *If we choose $\mathcal{S}$ as our sparsity pattern, then*

$$||AM - I||_F \leq n||A||_F\epsilon. \tag{3.9}$$

Proof. We first define an intermediate matrix $\tilde{N}$ which is essentially the truncation of $\tilde{A}^{-1}$ by:

$$(\tilde{N})_{i,j} = \begin{cases} (\tilde{A}^{-1})_{i,j} & (i,j) \in \mathcal{S}, \\ 0 & \text{otherwise,} \end{cases}$$

and denote the jth column of $\tilde{N}$ by $\tilde{n}_j$. The inequality (3.9) is a direct consequence of (3.5), (3.6), the definition of least squares solution and the definition of $\tilde{N}$ and is derived as follows:

$$\begin{aligned} ||AM - I||_F^2 &= ||\tilde{A}\tilde{M} - I||_F^2 \\ &= \sum_{j=1}^{n} ||\tilde{A}\tilde{m}_j - e_j||_2^2 \\ &\leq \sum_{j=1}^{n} ||\tilde{A}\tilde{n}_j - e_j||_2^2 \\ &= ||\tilde{A}\tilde{N} - I||_F^2 \\ &\leq ||A||_F^2 ||\tilde{N} - \tilde{A}^{-1}||_F^2 \\ &\leq n^2||A||_F^2\epsilon^2. \end{aligned}$$

□

Remark: Similar bound for the residual matrix can also be found in [67]. Our result is different in that we have an $O(n\log_2 n)$ estimate for the number of nonzeros of the computational approximate inverse M while that in [67] does not have such a bound.

The result of Theorem 3.3, however, is mainly of theoretical interest. First of all, the sparsity pattern $\mathcal{S}$ is not known in general. Besides, $O(n \log_2 n)$ elements for $\tilde{M}$ may be still too dense for practical purposes. Furthermore, because of the special finger-like distribution of the nonzero elements of $\tilde{M}$ given by $\mathcal{S}$, the amount of computation for solving the least squares problems may differ substantially from column to column. Thus in our implementation, we only choose a subset of $\mathcal{S}$ which corresponds to those entries near the main diagonal. We find that the preconditioning quality is still promising, as will be shown in Section 3.6. The implementation details will be discussed in Section 3.5.

3.4 Connection to HB Preconditioner

Because of the hierarchical structure of wavelets, there is a natural connection between the wavelet approximate inverse and the approximate inverse given by the hierarchical basis preconditioner [103, 150]. The wavelet approximate inverse, denoted by M^{WAI}, can be considered as an approximation to the transformed A^{-1}. That is,

$$\begin{aligned} M^{WAI} &= W^T \tilde{M} W, \\ \tilde{M} &= \mathit{approx}\,(\tilde{A}^{-1}), \end{aligned} \tag{3.10}$$

where $\mathit{approx}\,(\tilde{A}^{-1})$ is an approximation of $\tilde{A}^{-1}$. In our case, it is given by the solution of the least squares problems (3.6). We can also express the approximate inverse, M^{HB}, corresponding to the hierarchical basis preconditioner, in a similar form [103]:

$$\begin{aligned} M^{HB} &= S^T \hat{M} S, \\ \hat{M} &= (\mathit{approx}\,(\hat{A}))^{-1}, \end{aligned} \tag{3.11}$$

where S^T is the non-orthogonal transformation matrix from the hierarchical basis to the standard basis, $\hat{A} = SAS^T$, and $\mathit{approx}\,(\hat{A})$ is another approximation of $\hat{A}$, for instance, a coarse grid approximation of A.

These two approximate inverses are similar in that both possess a hierarchical structure. In fact, the hierarchical basis can also be considered as a special kind of wavelet since it consists of a hierarchy of piecewise linear functions and they are precisely the "hat" functions in the wavelet terminology. Our wavelet approximate inverse is more general in the sense that one is allowed to use other kind of wavelets, in particular, the orthogonal wavelets with compact support by Daubechies [40]. On the other hand, one could apply the hierarchical basis transform in a more general domain.

The main difference between the two approximate inverses is the way they approximate the original matrix A. For the approximate inverse given by the hierarchical basis in (3.11), we first approximate $\hat{A}$, for instance, by a block diagonal matrix, and then compute its exact inverse. For the wavelet approximate inverse, we compute $\tilde{A}$ exactly and then approximate its inverse by solving the least squares problems as discussed in Section 3.2. Typically, the approximate inverse given by the hierarchical basis is block diagonal with zero bandwidth except for the coarsest block while the one by wavelets can have nonzeros anywhere. If we choose the same block diagonal for the nonzeros of $\tilde{M}$, it will reduce to the same form (but probably with different values on the entries) of $\hat{M}$. Connections between wavelets and hierarchical basis are also made by Vassilevski and Wang [135, 136].

3.5 Complexity and Implementation

The naive algorithm in Section 3.2 needs quite an amount of overhead for doing the wavelet transformation. In this section, we will analyze each step of the algorithm and discuss some implementation issues of how to simplify and speed up the procedures. Meanwhile, we also analyze the sequential complexity of each step. We show that it is essentially $O(n)$, except *step(a)* which requires $O(kn)$ operations, where k is the number of levels of the wavelet transform.

In the following discussion, we assume that the wavelet used is orthogonal and of compact support, [40, 41]. Orthogonal wavelets are used so that the formulation developed in Section 3.2 makes sense. However, one could also use non-orthogonal wavelets. Compact support, on the other hand, is indispensable so that the wavelet transform is only $O(n)$ and $\tilde{A}$ does not become dense.

Step (a). In general, to compute the wavelet transform of a vector requires $O(n)$ operations. Computing $\tilde{A} = WAW^T$ is equivalent to transforming the columns and then the rows of A (or vice versa). Thus it will cost $O(n^2)$ operations. However, since A is sparse, if we assume that there are only $O(1)$ nonzeros in each column and each row, the cost will be reduced to $O(kn)$. In fact, in the parallel implementation, we do not need to form $\tilde{A}$ explicitly in each processor. Notice that solving each least squares problem only need a few columns of $\tilde{A}$. We just form those columns and the cost will be reduced to $O(n)$.

Step (b). In each level of the transform, we will introduce a fixed amount of nonzero entries. Even though there are only $O(1)$ nonzeros in each column and row of A, there will be $O(k)$ nonzeros in each column and row of $\tilde{A}$. We could choose k so small that the number of nonzeros introduced is acceptable. We may also reduce the cost significantly by taking advantage of the fact that for smooth coefficient problems, the Green's function typically has singularity only at a point. In other words, A^{-1} only has singularities along the diagonal, together with a few number of additional artificial singularities due to the wrapping of the 2D discrete Green's function to an one dimensional array in each row. Hence it is reasonable to fix the sparsity pattern of $\tilde{M}$ to be block diagonal. In fact, our current implementation already assumes block diagonal structure for $\tilde{M}$. This assumption saves an enormous amount of time used for searching for the next nonzero entries adaptively. By the way, adaptive searching procedure is usually less amenable to parallel implementation. We may further reduce the cost to $O(n)$ by adopting the concept of *local inverse* in [130].

Step (c). When we solve $\tilde{A}\tilde{x} = \tilde{b}$ by an iterative method, we need to perform $\tilde{A}$ times a vector. If we do it directly, the cost will be $O(kn)$ as $\tilde{A}$ has $O(k)$ nonzeros in each row. Note that

$$\tilde{A}v = W(A(W^T v)).$$

If we first backward transform v, apply A and then transform it back, the overall process will still be $O(n)$.

3.6 Numerical Results

In this section, we compare our preconditioner with SPAI (by Grote and Huckle [67]) to show that the wavelet sparse approximate inverse preconditioners gains both the efficiency and storage for PDE problems. The convergence results of ILU(0) are also compared to show that the wavelet sparse approximate inverse preconditioners are as efficient as ILU(0) and sometimes even better, in addition to the advantage that their parallel implementation is much easier than ILU(0). We choose several matrices arising from different elliptic PDEs. The inverses of all these matrices are piecewise smooth and the singularities are clustered around the diagonal. For efficiency, instead of applying SPAI to solve for $\tilde{M}$ adaptively in step (a) of our algorithm, we specify a block diagonal structure *a priori* for $\tilde{M}$ and then solve (3.6) by the QRF as discussed in Section 3.5. In all the tests, we use the compact support wavelet D_4 by Daubechies [40, 41]. We apply 6 levels of wavelet transform to matrices of order 1024 and 8 levels of transform to matrices of order 4096. Note that the number of levels is arbitrary. One could use different number in different situations.

We apply these preconditioners to GMRES(20). The initial guess was $x_0 = 0, r_0 = b$ and the stopping criterion was $\|r_n\|/\|r_0\| < 10^{-6}$. All the experiments were done in MATLAB in double precision.

We remark that GMRES was used because the matrices we are interested in are, in general, nonsymmetric and hence we do not adapt to using the conjugate gradient method even if A happens to be symmetric positive definite (Example 1 and 2). Nevertheless, it is true that our algorithm does not, in general, preserve symmetry. However, the conjugate gradient method can still be used if we replace $\tilde{M}$ by $(\tilde{M} + \tilde{M}^T)/2$ when $\tilde{M}$ is known to be positive definite. We also remark that another approach is to use factorized approximate inverses as described in [9, 88, 89, 92] for symmetric problems.

Example 1: We use two simple 1D matrices to show the benefit from using wavelet transforms. The first one is a near tridiagonal symmetric positive definite matrix of size 1024 $\times$ 1024, modified from an artificial periodic boundary value problem:

$$A = \begin{bmatrix} 2.00001 & -1 & & & & -1 \\ -1 & 2.00001 & -1 & & & \\ & -1 & 2.00001 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2.00001 & -1 \\ -1 & & & & -1 & 2.00001 \end{bmatrix}. \tag{3.12}$$

The second matrix is the 1D Laplacian operator derived from the following:

$$\begin{aligned} u''(x) &= f(x), \qquad \text{in } (0,1), \\ u(0) &= 0, \quad u'(1) = 0. \end{aligned}$$

Neumann boundary condition at $x = 1$ is used so that there is no decay in the Green's function near the boundary.

The bandwidth is 0,0,5,5,5,5 for the 1st to 6th level of the block diagonal structure of $\tilde{M}$ respectively.

Example 2: In this example, A is the 2D Laplacian operator with size 1024$\times$1024 and 4096$\times$4096. For n=1024, we choose the bandwidth of $\tilde{M}$ as before. For n=4096, we choose the bandwidth = 0,0,0,0,5,5,5,5 for the 1st to 8th level of the block diagonals respectively.

Example 3: Consider the following PDE with variable coefficients,

$$((1+x^2)u_x)_x + u_{yy} + (\tan y)^2 u_y = -100x^2.$$

We solve the 32$\times$32 and 64$\times$64 grid cases. The bandwidth of the block diagonal of $\tilde{M}$ is the same as before.

Example 4: In this case, A comes from a PDE of helical spring:

$$u_{xx} + u_{yy} + \frac{3}{5-y}u_x - 2G\lambda = 0,$$

where G and λ are some constants. Same setting as before.

Example 5: Finally, we show an example where our wavelet preconditioner does not work well. The matrix A comes from a discontinuous coefficients PDE:

$$(a(x,y)u_x)_x + (b(x,y)u_y)_y + u_x + u_y = \sin(\pi xy),$$

where the coefficients $a(x,y)$ and $b(x,y)$ are defined as:

$$a(x,y) = b(x,y) = \begin{cases} 10^{-3} & (x,y) \in [0,0.5] \times [0.5,1] \\ 10^{3} & (x,y) \in [0.5,1] \times [0,0.5] \\ 1 & \text{otherwise.} \end{cases}$$

The bandwidth is chosen to be 5,5,10,10,15,15 to make the number of nonzeros comparable to that of SPAI(0.2). Such modification is made so that sparsity is not a factor for the failure.

The convergence of GMRES(20) with different preconditioners in each example is shown in Figures 3.1–3.4 and is summarized in Table 3.2. In Example 1, we can see that SPAI(0.4) and SPAI(0.2) converge very slowly in this somewhat artificial but illustrating case. On the other hand, the wavelet based preconditioner converges much faster. This shows the advantage of wavelet transform in the case where A^{-1} is smooth with singularity only along the diagonal. We do not show the convergence of ILU(0) since it only takes 3 iterations to converge. This is exceptional because of the special near tridiagonal structure of A. Table 3.3 shows the number of nonzeros for each preconditioner. The wavelet based preconditioner requires much less amount of memory than SPAI does.

In Examples 2–4, the wavelet based preconditioner is most efficient in terms of convergence and storage. Although the convergence of the wavelet based preconditioner still depends on the mesh size (Figure 3.5), the dependence is less than that of ILU(0) and much less than that of SPAI. However, we would like to point out that this comparison is very rough since the preconditioners SPAI and ILU(0) take up many more nonzeros.

Besides rapid convergence, we can also see a tremendous gain in storage for the wavelet based preconditioner as n increases. This gain essentially comes from the wavelet compression. The larger n is, the more compression we can get. It is because the effect of singularity becomes less and less prominent as the singularity is only located along the diagonal.

Table 3.4 gives a comparison of the total operation counts for each method. The count estimate consists of the number of GMRES(20) iteration, the cost of matrix-vector multiply, application of the preconditioner and the number of inner products/saxpy operations. Since the number of inner products/saxpy operations depends on the iteration number, the operation count estimate for one GMRES(20) iteration, on the average, is:

$$\text{count} = nnz(A) + nnz(M) + 21n,$$

where $nnz(A)$ and $nnz(M)$ are the number of nonzeros of the matrix A and the preconditioner M respectively and n is the size of the matrix. The count for ILU(0) is normalized to one. The wavelet preconditioner shows a superior operation counts over all the other methods in Examples 2–4. In fact, the results are even better when n is larger. ILU(0) is exceptional good for the one dimensional problems in Example 1 as explained before. Despite that, the wavelet preconditioner still takes much smaller counts than the other two approximate inverses.

Finally, Figure 3.6(a) shows that the wavelet based preconditioner does not always work. As mentioned before, we assume that the singularity of the Green's function is only at a point so that the wavelet transformed inverse has large entries near the main diagonal and our implementation can capture those successfully as shown in previous examples. However, for discontinuous coefficient problems, the Green's function has addition singularity along the discontinuities of the coefficients as shown in Figure 3.6(b). Hence the inverse is not as smooth as before. Thus our block diagonal structure may not completely capture the significant elements of the exact inverse. We should remark that the failure is mainly due to our current implementation. In principle, if we can locate the significant elements by some adaptive procedure (e.g. the one given in [67] and [37]), we should be able to obtain an effective approximate inverse preconditioner. However, such a sophisticated adaptive searching technique has not been fully developed yet for this class of problems and further investigation is needed.

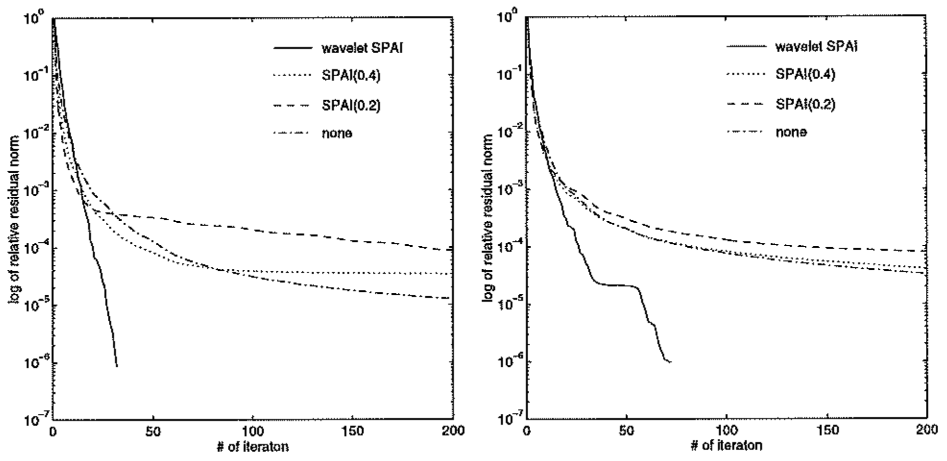


Figure 3.1: Convergence behavior of GMRES(20) where (a) A=artificial matrix (b) A=1D Laplacian with Dirichlet and Neumann boundary conditions.

Example	n	Wavelet SPAI	SPAI(0.4)	SPAI(0.2)	ILU(0)	No precond.
1a	1024	32	>200	>200	3	>200
1b	1024	71	>200	>200	1	>200
2	1024	26	62	34	28	116
	4096	47	160	63	57	>200
3	1024	26	100	40	34	>200
	4096	66	>200	129	93	>200
4	1024	26	84	36	31	183
	4096	68	>200	126	89	>200
5	1024	>200	64	34	21	>200

Table 3.2: Number of GMRES(20) iterations

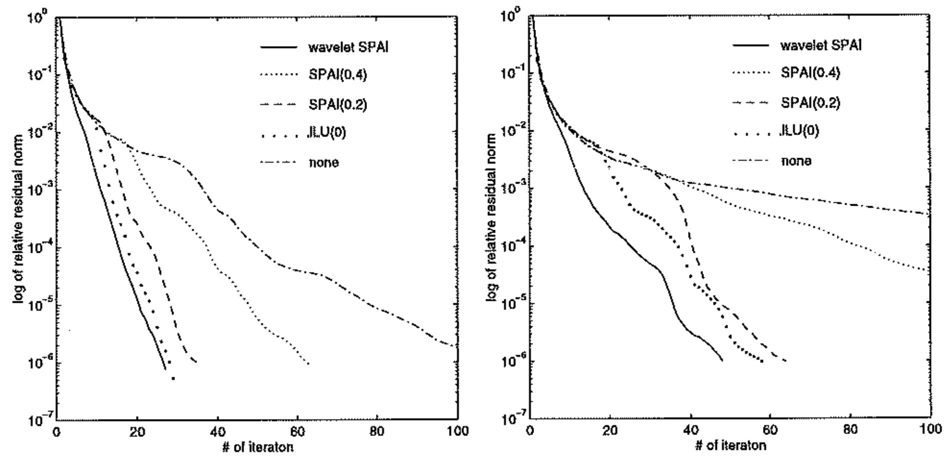


Figure 3.2: Convergence behavior of GMRES(20) where A=2D Laplacian with size (a) n=1024 (b) n=4096.

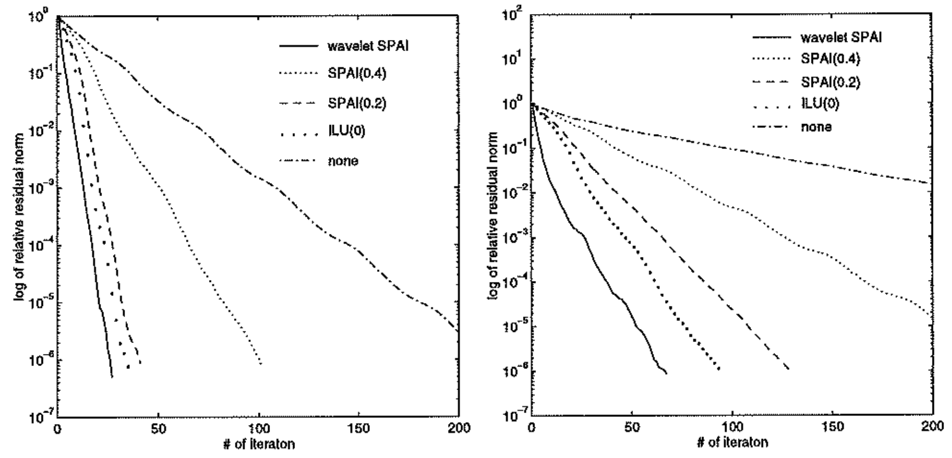


Figure 3.3: Convergence behavior of GMRES(20) where A=variable coefficient operator with size (a) n=1024 (b) n=4096.

Example	n	Wavelet SPAI	SPAI(0.4)	SPAI(0.2)	ILU(0)
1a	1024	3544	5120	21504	3072
1b	1024	3544	5121	25425	3072
2	1024	3544	4624	16440	4992
	4096	6616	19472	69688	20224
3	1024	3544	4514	17260	4992
	4096	6616	18618	73936	20224
4	1024	3544	4624	16387	4992
	4096	6616	19472	69628	20224
5	1024	13464	5677	18952	4992

Table 3.3: Number of nonzero in approximate inverse

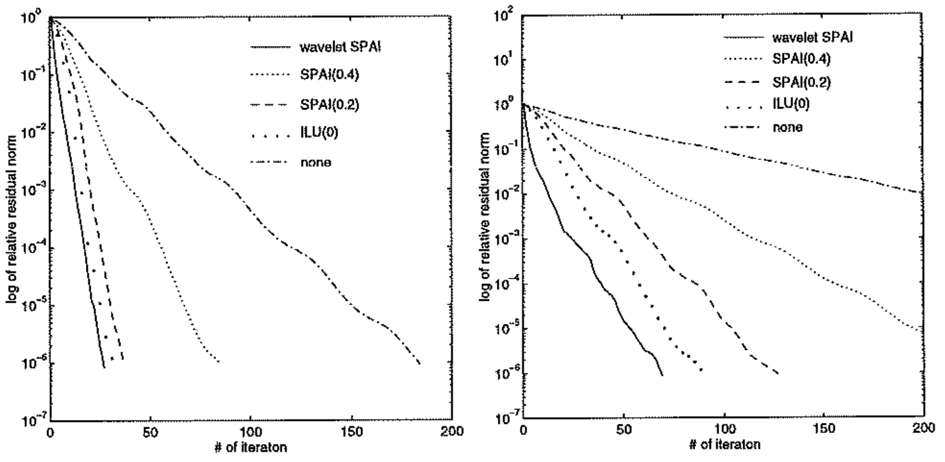


Figure 3.4: Convergence behavior of GMRES(20) where A=Helical spring operator with size (a) n=1024 (b) n=4096.

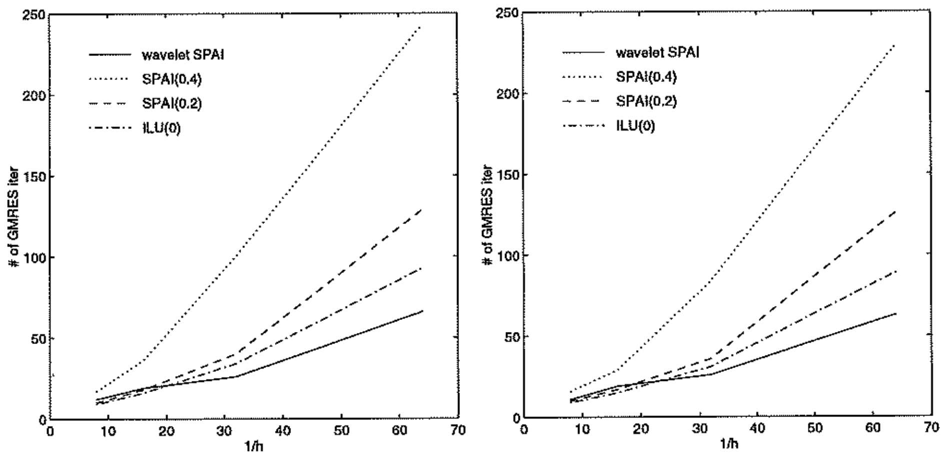


Figure 3.5: Iteration number vs $1/h$, h=mesh size, for (a) variable coefficient (b) Helical spring.

Example	n	Wavelet SPAI	SPAI(0.4)	SPAI(0.2)	ILU(0)
1a	1024	10	>72	>111	1
1b	1024	72	>215	>362	1
2	1024	0.95	2.65	2.02	1
	4096	0.78	2.79	1.54	1
3	1024	0.73	2.90	1.63	1
	4096	0.63	>2.12	1.98	1
4	1024	0.80	2.68	1.58	1
	4096	0.68	2.23	1.97	1
5	1024	>12	3.11	2.33	1

Table 3.4: Operation count estimate. The count for ILU(0) is normalized to 1.

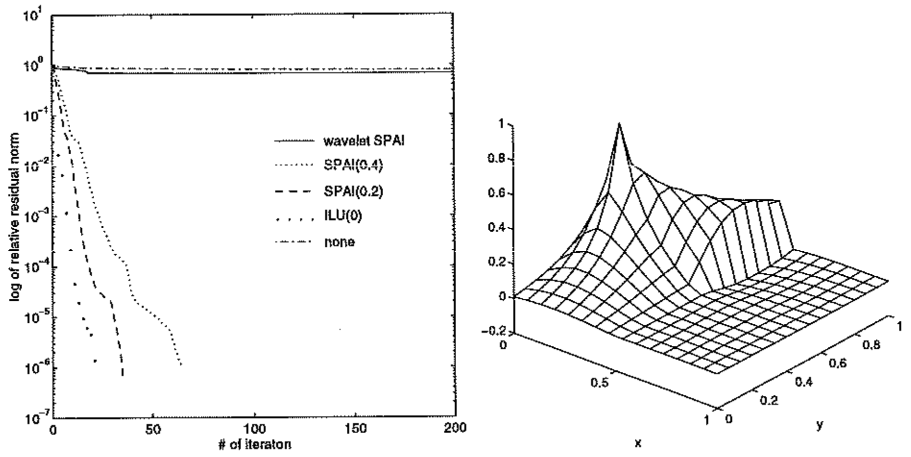


Figure 3.6: (a) Convergence behavior of GMRES(20) where A=discont. coeffs. (b) The Green's function of A at the point (0.1,0.5)

3.7 Concluding Remarks

We have extended the potential applicability of approximate inverse to a larger class of problems, namely, matrices with piecewise smooth inverses. There are two main factors concerning our preconditioner: choice of basis and sparsity pattern. We have shown that for our block diagonal implementation, the wavelet basis is suitable for matrices with piecewise smooth inverse and singularity mainly along the diagonal. Moreover, significant amount of storage can be saved. We should remark that other choices of basis are also feasible to solve specific problems, for example, higher order wavelets [41] and basis derived from multiresolution methods [76].

If the essential singularity of A^{-1} is along the diagonal, we have shown that block diagonal structure is sufficient. However, for more general situations, e.g. discontinuous coefficients, where the singularity is not necessarily near the diagonal, more sophisticated adaptive searching procedure is needed to locate the sparsity pattern correctly.

CHAPTER 4

Robust Multigrid Methods: Introduction

4.1 Model Problem

In this and the next three chapters, we focus on a specific and yet very important class of problems in scientific computing, namely, second order self-adjoint elliptic PDEs:

$$\begin{aligned} -\nabla \cdot a\nabla u &= f \qquad \text{in } \Omega \\ u &= 0 \qquad \text{on } \partial\Omega. \end{aligned} \tag{4.1}$$

The domain $\Omega \subset \Re^d, d = 1, 2$, or 3, is a polygon. Traditionally, a is assumed to be smooth and uniformly positive on $\bar{\Omega}$. Here we only assume that a is positive on Ω, and it is allowed to be oscillatory or discontinuous with large jump across the interfaces.

Let $H^1(\Omega)$ be the standard Sobolev space on Ω and $H_0^1(\Omega)$ its subspace whose functions vanish on $\partial\Omega$. The variational formulation of (4.1) is to find $u \in H_0^1(\Omega)$ such that

$$a(u, v) = (f, v) \qquad \forall v \in H_0^1(\Omega),$$

where the bilinear form $a(\cdot, \cdot)$ and the linear form $(\cdot, \cdot)$ are defined as

$$a(u, v) = \int_\Omega a(x)\nabla u \cdot \nabla v dx, \quad (f, v) = \int_\Omega f(x)v(x)dx.$$

Let $T^h = \{\tau_i^h\}$ be a triangulation of $\Omega = \cup_i \tau_i^h$ with simplexes τ_i^h. Define the linear finite element space to be

$$V^h = \{v^h \in H_0^1(\Omega) : v^h|_{\tau_i^h} \text{ is linear } \forall i\},$$

and denote the set of nodal basis by $\{\phi_j^h\}_{j=1}^n$. The finite element approximation to the solution of (4.1) is the function $u^h \in V^h$, so that

$$a(u^h, v^h) = (f, v^h) \qquad \forall v^h \in V^h. \tag{4.2}$$

Let $u^h = \sum_{j=1}^n \mu_j \phi_j^h$ and $f = \sum_{j=1}^n \beta_j \phi_j^h$. Then (4.2) is equivalent to a linear system:

$$\mathcal{A}^h \mu = \mathcal{M}^h b, \tag{4.3}$$

where $\mu = (\mu_1, \ldots, \mu_n)^T$, $b = (\beta_1, \ldots, \beta_n)^T$, $\mathcal{A}^h$ is the stiffness matrix, $\mathcal{A}^h_{i,j} = a(\phi^h_i, \phi^h_j)$, and $\mathcal{M}^h$ is the mass matrix, $\mathcal{M}^h_{i,j} = (\phi^h_i, \phi^h_j)$.

While ILU and wavelet sparse approximate inverse preconditioners are applicable to general linear systems and in particular (4.3), their convergence may not be optimal; the convergence rate decreases with the size of the linear systems. Among the optimal methods, multigrid is one of the most efficient and popular methods for the solution of (4.3).

The success of multigrid hinges on the choice of the coarse grids, the smoothing procedure, the interpolation operators, and the coarser grid discretization. In standard multigrid, full coarsening, Jacobi or Gauss-Seidel smoothing, and linear interpolation are usually used. Classical convergence theory [13, 14, 15, 73, 97, 121, 148, 149]. shows that these simple ingredients are enough to achieve optimal convergence for smooth coefficient problems. In general, however, these choices may lead to slow convergence. In the next sections, we give a general overview of the role that each multigrid component plays within the whole multigrid process, and how their choices may depend on the underlying PDE. In Chapter 5, we propose a robust parallel smoother for multigrid, which are effective for anisotropic coefficient PDEs on structured and unstructured grids. In Chapter 6, we study a special coarsening technique for discontinuous coefficient problems. The importance of coarsening for discontinuous coefficient problems has not been emphasized in the literature, and the geometric interpretation presented in Chapter 6 sheds a new insight into robust multigrid for discontinuous coefficient PDEs. In Chapter 7, we propose and analyze both theoretically and numerically a new interpolation for rough coefficient problems based on energy minimization. This energy-minimizing interpolation highlights the main theme of robust multigrid methods.

4.2 Smoothing

The idea of multigrid consists of two main components: *smoothing* and *coarse grid correction*. The smoothing process, usually carried out by a relaxation method, damps away the high energy error components. The coarse grid correction process, carried out by an interpolation, approximates the low energy error components on the coarser grids. The idea is that the combination of the two will result in a significant error reduction independent of the problem size and hence lead to a fast solution procedure. Moreover, we gain efficiency since the coarse grid solve is less expensive than the fine grid one.

Relaxation methods such as Richardson, Jacobi and Gauss-Seidel are often used as smoothers for multigrid, although other iterative methods, for instance, ILU [77, 146] and (preconditioned) conjugate gradient [5, 25], or even ODE solvers [80] such as Runge Kutta methods, have also been used for specific problems. The relaxation methods are particularly useful for multigrid since they all have a common property of removing the error eigencomponents corresponding to the large eigenvalues. A number of different smoothing analysis for relaxation methods can be found in the literature. For instance, Brandt [17] used a local mode approach which essentially came from the Fourier analysis for hyperbolic PDEs, whereas Hackbusch [73] and some others adopted a more functional analysis approach. Ruge and Stuben [112] etc. used a purely algebraic approach. In the following, we give a different algebraic smoothing analysis for the relaxation methods. Although the results used in the analysis are known, they are not usually discussed in the context of multigrid smoothing. The error reduction results in Theorem 4.2 and Theorem 4.3 are new, however.

4.2.1 Richardson

Richardson iteration is theoretically simplest and its analysis is also clearest. Let A be symmetric and positive definite and let $0 < \lambda_1 \leq \cdots \leq \lambda_n$ be its eigenvalues and $\{v_i\}$ the corresponding eigenvectors. The error e^{k+1} of the $k+1$st Richardson iteration is given by

$$e^{k+1} = (I - \frac{1}{\lambda_n}A)e^k. \tag{4.4}$$

Let the eigendecomposition of e^k be: $e^k = \sum_{i=1}^n \xi_i^k v_i$. By (4.4) and a direct computation, we have

$$e^{k+1} = \sum_{i=1}^n (1 - \frac{\lambda_i}{\lambda_n})\xi_i^k v_i = \sum_{i=1}^n \xi_i^{k+1} v_i,$$

where $\xi_i^{k+1} = (1 - \lambda_i/\lambda_n)\xi_i^k$. Thus, for $i \to n, \xi_i^{k+1} \to 0$, and for $i \to 1, \xi_i^{k+1} \to \xi_i^k$. Hence the large eigenvalue components are eliminated while the small eigenvalue components are retained by an iteration of Richardson.

4.2.2 Jacobi and Gauss-Seidel

The smoothing effect of the Jacobi iteration is much more subtle. In contrast with Richardson, it is not clear how each eigencomponent is damped by a Jacobi iteration. Nevertheless, we can show that the A-norm of the error in the next Jacobi iteration is minimized in some sense explained as follows. The following analysis is based on a well-known result [121, Lemma 1, Section 5.2] in the domain decomposition context and was also presented in the subspace correction framework by Xu [149].

Lemma 4.1 *Consider the minimization problem*

$$\min_{\mu_i} \|x^* - (x^k + \mu_i f_i)\|_A^2, \tag{4.5}$$

where x^* is the exact solution, x^k is an approximation of x^* and f_i is a vector of zeros with 1 at the ith entry. Then the minimizing solution μ_i is given by

$$\mu_i = \frac{(Ae^k)_i}{A_{i,i}}, \tag{4.6}$$

where $(Ae^k)_i$ is the ith component of the vector Ae^k and $A_{i,i}$ the (i,i)th entry of A.

Lemma 4.1 implies that if we want to modify the ith component of x^k so that the error is minimized in the A-norm, the update is given by (4.6).

Theorem 4.1 *Let $\mu = (\mu_1, \ldots, \mu_n)^T$, μ_i given by (4.6). Then the new approximate $x^k + \mu$ is the same as that given by one iteration of Jacobi:*

$$x^{k+1} \equiv x^k + \mu = x^k + D^{-1}Ae^k,$$

where D is the diagonal of A.

Proof. By Lemma 4.1, we have

$$\mu = \sum_{i=1}^{n} \mu_i f_i = \sum_{i=1}^{n} \frac{(Ae^k)_i}{A_{i,i}} f_i = D^{-1}Ae^k.$$

□

Hence Jacobi iteration minimizes the A-norm of the next iterate along each component. Moreover, we can compute precisely the amount of the error in the A-norm reduced by an iteration of Jacobi.

Theorem 4.2 *The error reduction in the A-norm of a Jacobi iteration is*

$$\|e^k\|_A^2 - \|e^{k+1}\|_A^2 = (e^k)^T AD^{-1}(2D - A)D^{-1}Ae^k. \tag{4.7}$$

Proof. Noting that the iteration matrix for the error of the Jacobi iteration is $I - D^{-1}A$, we can show by a direct computation that

$$(I - D^{-1}A)A(I - D^{-1}A) = A - AD^{-1}(2D - A)D^{-1}A. \tag{4.8}$$

The equality (4.7) follows from multiplying e^k to the left and the right of (4.8). □

By Theorem 4.2, we readily derive a sufficient condition for the convergence of the Jacobi iteration.

Corollary 4.1 *The Jacobi method is convergent if $2D - A$ is positive definite.*

Remark: It is not clear from the above analysis whether Richardson or Jacobi iteration is a better smoother. In practice, however, Jacobi is much more effective.

The smoothing effect of Gauss-Seidel can be understood similarly. However, instead of minimizing the entries at a time as in Jacobi, they are minimized one by one using the latest updated value for each entry. Hence, intuitively speaking, Gauss-Seidel smoothing should be more effective than Jacobi. Moreover, we can also compute precisely the error reduction in the A-norm in a similar way as in Theorem 4.2.

Theorem 4.3 *The error reduction in the A-norm of a Gauss-Seidel iteration is*

$$\|e^k\|_A^2 - \|e^{k+1}\|_A^2 = (e^k)^T A(D-L)^{-T} D(D-L)^{-1} A e^k.$$

Again, by Theorem 4.3, we can derive a sufficient condition for the convergence of the Gauss-Seidel iteration.

Corollary 4.2 *The Gauss-Seidel method is convergent for symmetric matrices whose lower triangular part, $D - L$, is nonsingular and whose diagonal entries are all positive. In particular, Gauss-Seidel is convergent for symmetric positive definite matrices.*

4.2.3 Other Smoothings

Relaxation smoothers are efficient for smooth coefficient problems. They may be very slow, in general. For instance, for anisotropic problems:

$$\epsilon u_{xx} + u_{yy} = f, \qquad \epsilon \ll 1, \tag{4.9}$$

we often use the block version of the relaxation methods instead. The analysis by Zhang and Bramble [152] shows that the convergence of the resulting multigrid is independent of the anisotropy ϵ. For discontinuous coefficient problems, Wesseling [144, 143], for instance, proposed to use ILU smoothers.

Another issue of smoothing is the parallel efficiency. In practice, Gauss-Seidel is usually the most effective smoothers among other relaxation methods. A drawback is that it is a very sequential algorithm. A parallel version of it may be obtained by a special ordering of the unknowns, for example, red-black ordering for the five-point stencil operator on a square grid. Jacobi method, on the other hand, is a very parallel method, but its smoothing efficiency is not as good as Gauss-Seidel.

In Chapter 5, we propose a new class of sparse approximate inverse smoothers. It has a similar smoothing efficiency as Gauss-Seidel and it is independent of ordering. Moreover, for hard problems, we can improve the smoothing efficiency by adaptively adjusting the quality of the approximate inverse, for instance, by adding more nonzeros.

4.3 Coarsening

The basic setting of multigrid is a hierarchy of grids. For structured square or triangular grids, it is very natural to define a subsequence of coarser grid; see Figure 4.1. Sometimes, it is more convenient to refer to the grid points than the grids themselves, in which case, we call the process of selecting a subset of coarse grid points *coarsening*. For structured grid problems, it is not hard to see that a set of coarse grid points defines a coarse grid, and vice versa (Figure 4.1). For unstructured grid problems, however, it is not as clear. Nevertheless, we may still use the coarse grid points to define a coarse grid, or more precisely, a coarse subspace; see Chapter 7.

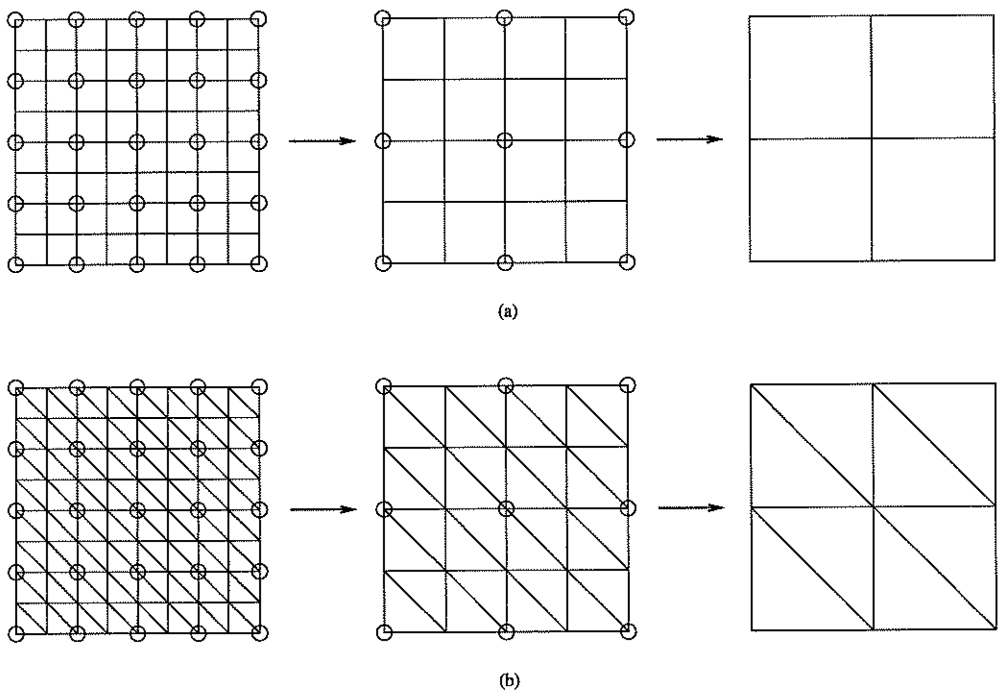


Figure 4.1: Standard sequence of grids, (a) square (b) triangular. The coarse grid points are denoted by $\circ$.

In the literature, coarsening is less emphasized among other issues of multigrid since the simple standard coarsening (Figure 4.1) already produces a fast multigrid method. For anisotropic problems, a special coarsening technique called semi-coarsening [48, 124] is often used to recover the fast convergence of multigrid.

In Chapter 6, we show numerically that coarsening may also play an important role to improve multigrid convergence for discontinuous coefficient problems. In some special case, it may be mandatory to use a carcfully selected set of coarse grid points.

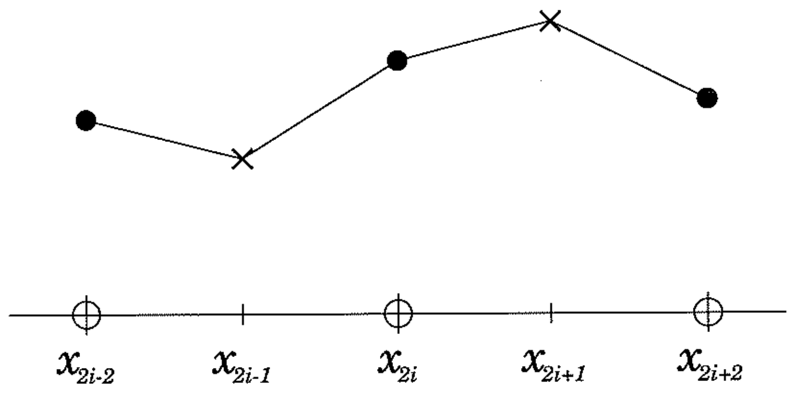


Figure 4.2: Interpolation for noncoarse grid points. The coarse grid points, the known values and the unknown values are denoted by $\circ$, $\bullet$ and $\times$, respectively.

4.4 Interpolation and Coarse Grid Basis

After a few smoothing steps on the fine grid, we transfer the fine grid residual function onto the coarse grid. This is called the *restriction.* On the coarse grid, after the error is solved either exactly or approximately, we transfer the coarse grid error back to the fine grid, and it is called the *prolongation.* If the Galerkin process is used, the restriction operator is just the adjoint of the prolongation operator.

The prolongation is essentially an interpolation: to determine the values at the noncoarse grid points from the values at the coarse grid points; see Figure 4.2. Typically, linear interpolation is used:

$$u_{2i-1} = \frac{1}{2}u_{2i-2} + \frac{1}{2}u_{2i}.$$

That is, the value of u_{2i-1} is given by a linear combination of values of u_{2i-2} and u_{2i} with weighting 1/2. Moreover, the interpolation used uniquely determines the coarse grid nodal basis functions and vice versa. Consider a linear element on the coarse grid, ϕ_i^H (Figure 4.3). It has a value of 1/2 at the noncoarse grid points x_{2i-1} and x_{2i}. Write ϕ_i^H as a linear combination of three fine grid basis functions:

$$\phi_i^H = \frac{1}{2}\,\phi_{2i-1}^h + 1\,\phi_{2i}^h + \frac{1}{2}\,\phi_{2i}^h.$$

The weightings for the terms ϕ^h_{2i-1} and ϕ^h_{2i} are $1/2$, which are precisely the weightings given by the linear interpolation. As a result, once we know the coarse grid basis functions, the prolongation and the restriction can be defined accordingly.

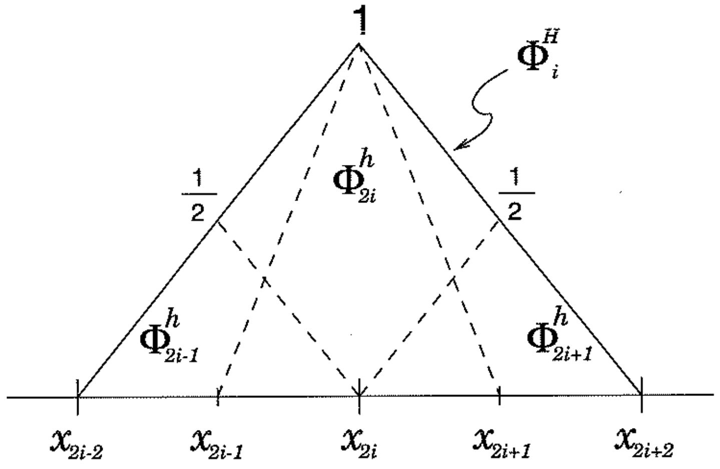


Figure 4.3: Coarse grid basis function as a (linear) combination of fine grid basis functions.

Good choices of interpolation have been the key in designing robust multigrid methods. For structured grid problems, Alcouffe et. al. [1] used special harmonic averaging techniques to define the interpolation weightings for discontinuous coefficient problems. Dendy [46] derived a Black Box Multigrid method in which the interpolation is defined solely from the nine-point stencils. Brandt, McCormick and Ruge [18] introduced purely algebraic techniques for defining interpolation solely from the given matrix. Since then, this approach has been studied extensively, for instance, by Ruge and Stuben [112]. Another related approach, which realized interpolations as matrices, led to the so called matrix-dependent interpolation [151].

The Black Box Multigrid approach exploits the underlying PDE, but is restricted to topological square grid problems. The algebraic or matrix-dependent approaches do not have any geometric restriction and hence may be applied to more general problems. However, since they pay less attention to the underlying PDE, it may lead to slower convergence rate for structured grid problems where one can easily take advantage of the geometric information and the special structure of the resulting matrix. In Chapter 7, we propose and analyze a robust interpolation through the construction of special coarse grid functions based on energy minimization, which has the advantage of exploiting the underlying PDE without any geometric restriction while maintaining the same efficiency on structured grids.

CHAPTER 5

Sparse Approximate Inverse Smoother for Multigrid

5.1 Introduction

In Chapter 4, we show that relaxation methods are effective for reducing the energy norm of the errors, and hence they are popular choice of smoothers for multigrid. In practice, Gauss-Seidel is usually more effective than the other relaxation methods. A drawback, however, is that Gauss-Seidel is a very sequential algorithm. A parallel version of it may be obtained by a special ordering of the unknowns, for example, red-black ordering for the five-point stencil operator on a square grid. The Jacobi method, on the other hand, is a very parallel method, but its smoothing efficiency is not as good as Gauss-Seidel. It is also well-known that Gauss-Seidel smoothers do not work well for anisotropic problems and discontinuous coefficient problems. In this chapter, we propose and analyze a new class of smoothers derived from sparse approximate inverse (SAI) preconditioners. It has a similar smoothing efficiency as Gauss-Seidel and it is independent of ordering. Moreover, for hard problems, we can improve the smoothing efficiency by adaptively adjusting the quality of the approximate inverse, for instance, by adding more nonzeros. Hence, this new technique is more robust than the Gauss-Seidel smoothers. Our numerical testings verify this statement.

We remark that Huckle [79] independently experimented with a sparse approximate inverse smoother for multigrid by modifying the standard Gauss-Seidel iteration. Specifically, instead of using the exact inverse of the lower triangular matrix, he used a sparse approximate inverse of it computed by the techniques described in [67] (cf. Chapter 3). In our approach, we do not restrict ourselves to Gauss-Seidel only. Indeed, we replace the Gauss-Seidel smoother completely by a spare approximate inverse smoother. The resulting multigrid is efficient, and we have more flexibility of improving the smoothing quality for hard problems.

In Section 5.2, we describe the construction of the sparse approximate inverse smoother. In Section 5.3, we analyze the smoothing property of the proposed smoother analytically and numerically for constant coefficient PDEs. Some techniques to improve the smoothing character for the anisotropic problems are presented. Finally, in Section 5.5, we show the effectiveness of the sparse approximate inverse as a smoother for multigrid by a variety of problems including anisotropic problems, discontinuous coefficient problems and unstructured grid problems. Finally, concluding remarks are made in Section 5.6.

5.2 SAI smoother

Various techniques have been proposed for an effective sparse approximate inverse preconditioner [7, 9, 36, 39, 68, 67, 88, 89, 131]. However, the goal of constructing an effective smoother is very different from finding a good preconditioner. For a powerful preconditioner, the capability of removing both the high and low energy errors is essential. In contrast, a good smoother may just remove the high energy errors effectively. In this respect, much of the weakness of SAI preconditioners [131] becomes the strength of SAI smoothers. Our new proposal is to explore them to construct a powerful smoother.

Most sparse approximate inverse approaches seek a sparse matrix M so that the error of the residual

$$E = AM - I$$

is minimized in some measure. The sparsity constraint often limits the effectiveness of M as a preconditioner due to the locality of the sparse approximation. The lack of a global approximation has created many difficulties for an effective preconditioner. Various additional techniques are required to improve the quality of a SAI preconditioner [33, 131]. The requirement for a good smoother, on the other hand, can take advantage of the locality of the sparse approximation. In this chapter, a simpler form of approximate inverse – *local least squares approximation* is studied, whose cost of computing a sparse approximate inverse is among the least of all others. Its construction is based on the Frobenius norm approach described in Chapter 3:

$$\min_M \|AM - I\|_F^2,$$

subject to some constraint on the number and position of the nonzero entries of M. The minimization problem is equivalent to:

$$\min_{m_j} \|Am_j - e_j\|_2, \qquad j = 1, \ldots, n. \tag{5.1}$$

We observe that the matrix A in (5.1) can be replaced by a submatrix of itself without changing the least squares problem. First, only those columns of A which correspond to the nonzero entries of m_j are needed. Second, the submatrix of A consisting of those columns often has many zero rows which can be deleted. As a result, we end up with a small submatrix A' for the least squares problems. Conversely, if we are given a submatrix of A, it will define a least squares problem for m_j. In this case, the nonzeros of m_j is given by the corresponding columns of the submatrix. In the following, we describe a systematic way of constructing submatrices of A which are used to define least squares problems for the m_j's. The advantage of this modified Frobenius norm approach is that we have direct control of the the number and location of the nonzeros of m_j's as well as the complexity of the least squares problems.

A sparse matrix A can be represented by a digraph $\mathcal{G} = (\mathcal{O}, E)$ [60]. For finite element methods, the graph often is the mesh on the PDE solution domain. Define $\mathcal{L}_k(o_i)$, the k-level neighbor set of node o_i, the nodes which are a distance $k+1$ or less from o_i. The use of level concept to define the sparsity pattern for incomplete factorizations is widely used for ILU preconditioning [71, 42]. The 0-level neighbor set $\mathcal{L}_0(o_i)$ contains all the nodes which directly connect to node o_i. For PDE problems with a second order finite difference/finite discretization, $\mathcal{L}_0(o_i)$ is just the set of stencil points. Similarly, define $\mathcal{W}_k(o_i)$, the k^{th}-wavefront of node o_i, as the set of nodes which are a distance $k+1$ from the node o_i.

It can be shown that the elements in a discrete Green's function decay in a wavefront fashion for many problems [129]. In particular, $\mathcal{L}_k(o_i)$ includes the k most influential wavefronts. That is the motivation to choose $\mathcal{L}_k(o_i)$ for the sparsity pattern to approximate the discrete Green's function at node o_i. The computation of these locations is also inexpensive. A good choice of the sparsity pattern has a significant impact on the quality of the SAI.

The rectangular submatrix

$$A_{kl} \equiv A(\mathcal{L}_k(o_i), \mathcal{L}_l(o_i)) \tag{5.2}$$

is defined as the (k,l)-level local matrix of node o_i. This matrix takes rows corresponding to nodes $\mathcal{L}_k(o_i)$ and columns corresponding to nodes $\mathcal{L}_l(o_i)$ from A. We introduce the *(k,l)-level local least squares approximation* of an inverse as follows. For each node o_i, the least squares solution[1] x of

$$A_{kl}x = e_{o_i} \tag{5.3}$$

is taken for the nonzero values at the corresponding location $\mathcal{L}_l(o_i)$ of the sparse approximation M at node o_i. More precisely, we inject each element of the least squares solution x into a zero vector of size of the matrix A at the locations in $\mathcal{L}_l(o_i)$ and use this sparse column to approximate the column corresponding to the node o_i of the inverse matrix A^{-1}.

We show the construction by an example. Consider the following constant coefficient second order elliptic PDE:

$$\begin{aligned} w_1 u_{xx} + w_2 u_{yy} + s u_x + t u_y &= f(x,y) \qquad (x,y) \in \Omega \\ u|_\Gamma &= g(x,y), \end{aligned}$$

on an $m \times n$ regular grid, the resulting matrix is a penta-diagonal matrix A of size $(m \times n)^2$, using the conventional finite difference or finite element method. The

[1]In (5.3), e_{o_i} is a unit basis vector of size $|\mathcal{L}_l(o_i)|$ with one in the o_i position and zeros in the rest of the location in $\mathcal{L}_k(o_i)$.

difference equations can be written as:

$$bu_{i+1,j} + du_{i,j+1} + au_{i,j} + cu_{i-1,j} + eu_{i,j-1} = h^2 f_{i,j}, \quad 1 \le i \le m, 1 \le j \le n.$$

The $(1,0)$-level local least squares problem for an interior grid point becomes:

$$\begin{pmatrix} a & & & & c \\ & a & & & d \\ & & a & & b \\ & & & a & e \\ c & d & b & e & a \\ c & & & & \\ d & c & & & \\ & d & & & \\ & b & d & & \\ & & b & & \\ & & e & b & \\ & & & e & \\ e & & & c & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{5.4}$$

It is clear that the sparsity pattern of the approximate inverse is determined by the locations $\mathcal{L}_l(o_i)$. A higher level l implies a denser approximation. The locations $\mathcal{L}_k(o_i)$ control the quality of the approximation given by x. A higher level k often produces a smaller error of the residual at the locations $\mathcal{L}_k(o_i)$. The $(0,0)$-level approach was used by Benson [7] originally, and was not a good choice for either preconditioner or smoother. Our experiences indicate that the $(1,0)$-level local least squares approximation provides a good trade-off between cost and efficiency in general. However, for difficult problems such as PDEs with anisotropic coefficient, higher level SAI smoother is required. This adaptable quality makes the SAI smoother a robust technique.

To compute these local least squares solutions is an easy task and can be implemented in parallel efficiently. For any constant coefficient PDE on a regular mesh, we may further simplify the computations as follows. We observe that the local least squares problems and hence the solutions corresponding to the interior points are identical (cf. (5.4)). However, they may be different near the boundary. We simply use the least squares solutions obtained from the interior points for the solutions at the boundary. Thus, we only need to solve one least squares problem. Our test and analysis indicate that this simple approach does not bring any noticeable penalty in performance. Various techniques can also be adopted to save the cost to compute the SAI. For example:

- The shape of the Green's function for different positions inside the solution region does not vary much for a PDE with constant coefficients. When an unstructured mesh is used, we may compute one approximation of the discrete

Green's function accurately and use interpolation to obtain the approximation for all other mesh point.

- For smooth variable coefficient PDEs, the use of one local least squares solution for several of its neighbors is feasible.
- For anisotropic problems, a higher level SAI smoother is required. However, the cost of the local least squares problems grows rapidly. A priori drop tolerance techniques [131] can significantly reduce the computations.

5.3 Smoothing Factor Analysis

We present an smoothing analysis for constant coefficient PDEs on a two dimensional rectangular region; see the example in Section 5.2. The analysis is only for the simplified (1,0)-level local least squares SAI smoother described above. However, we show by two numerical examples afterwards that the performance of the original and the simplified local least squares SAI smoothers are practically the same. We note that a related analysis techniques for discretization matrices with periodic boundary conditions are given by Chan and Elman [28].

Since (1,0)-level is used, the resulting sparse approximate inverse is also a pentadiagonal matrix. The smoothing analysis is based on the following result.

Lemma 5.1 *Given two tridiagonal matrices*

$$B = tridiag(b, a', c)_{n\times n}, \quad C = tridiag(d, a'', e)_{m\times m},$$

where b, c, d and e are non-negative, the eigenvalues of the matrix

$$B \otimes I_{m\times m} + I_{n\times n} \otimes C \tag{5.5}$$

are

$$a + 2\sqrt{bc}\cos\frac{k\pi}{n+1} + 2\sqrt{de}\cos\frac{j\pi}{m+1}, \quad 1 \le k \le n; \quad 1 \le j \le m,$$

where $a = a' + a''$. The corresponding eigenvectors are

$$\left\{\sin\frac{ks\pi}{n+1}\sin\frac{jt\pi}{m+1}\right\}, \quad 1 \le s \le n;\ 1 \le t \le m.$$

This result was used in many books and articles, for example, "Iterative Solutions of Large Linear Systems" by David Young.

In Lemma 5.1, we assume that b, c, d and e are non-negative. If a is negative, then $-A$ is an M-matrix and hence the solution values x_1, x_2, x_3, x_4 and x_5 in (5.4) are all non-positive. Denote by S the simplified smoother of matrix A. The matrix S can be written in the form of (5.5) where $-x_1, -x_2, -x_3$ and $-x_4$ correspond to b, c, d and e and $-x_5 = a' + a''$. Thus the eigenvalues of S can also be given by Lemma 5.1.

Corollary 5.1 *The eigenvalues of the matrix S are:*

$$-x_5 + 2\sqrt{x_2 x_4} \cos\frac{k\pi}{n+1} + 2\sqrt{x_1 x_3}\cos\frac{j\pi}{m+1}, \quad 1 \le k \le n; \quad 1 \le j \le m,$$

and the corresponding eigenvectors are

$$\left\{\sin\frac{ks\pi}{n+1}\sin\frac{jt\pi}{m+1}\right\}, \quad 1 \le s \le n; \ 1 \le t \le m.$$

Consequently, we have the following result.

Theorem 5.1 *The eigenvalues of the iterative matrix $I - SA$ are*

$$1 \ - \ \left(\left((-x_5 + 2\sqrt{x_1 x_3}\cos\frac{k\pi}{n+1} + 2\sqrt{x_2 x_4}\cos\frac{j\pi}{m+1}\right) \left(a + 2\sqrt{bc}\cos\frac{k\pi}{n+1} + 2\sqrt{de}\cos\frac{j\pi}{m+1}\right)\right),$$

where $1 \le k \le n, 1 \le j \le m$.

Proof. Since S and A have the same set of eigenvectors, they can be simultaneously diagonalizable. The result follows directly from Lemma 5.1 and Corollary 5.1. □

We illustrate the implication of Theorem 5.1 by the Laplacian example. In this case, $b = c = d = e = 1$ and $a = -4$. The solution of (5.4) gives:

$$x_1 = x_2 = x_3 = x_4 = -\frac{6}{61} \quad \text{and} \quad x_5 = -\frac{17}{61}.$$

For $k \to n, j \to m$, the corresponding eigenvalues of S and A approach 5/61 and 8, respectively. Thus, the corresponding eigenvalue of $I - SA$ approaches 1/3 approximately. Hence the high energy errors are efficiently damped away by a factor of 1/3. This analysis can also be generalized to three dimensional problems.

The above analysis is for the simplified (1,0)-level local least squares smoothers. We show numerically that the eigenvalues of the iteration matrix corresponding to the (1,0)-level local least squares smoothers are very much the same as the simplified version. We consider the Laplacian operator on a 20×20 grid. The

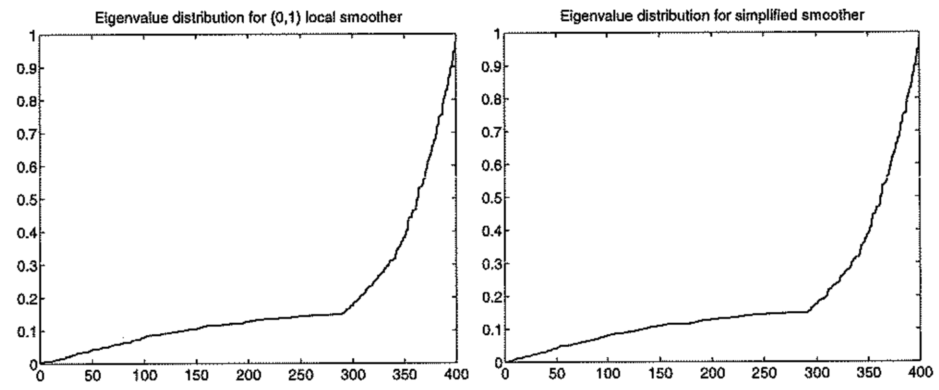


Figure 5.1: Comparison of the eigenvalue distributions. (a) (1,0)-level local smoother, (b) simplified smoother.

eigenvalues of $I - SA$ for S=(1,0)-level local least squares smoother and S=the simplified smoother are shown in Figure 5.1. There is no apparent difference between the two sets of eigenvalues.

For the single-direction anisotropic problem

$$\begin{aligned} 100u_{xx} + u_{yy} &= f(x,y) \qquad (x,y) \in \Omega \\ u|_{\Gamma} &= g(x,y), \end{aligned} \tag{5.6}$$

the difference in the eigenvalue distributions between the simplified SAI and the local least squares SAI can be seen in Figure 5.2. However, this difference does not have any significant impact on the smoothing factor, if the local least squares smoother is replaced by the simplified smoother.

In addition to the above Fourier analysis, we compare the smoothing efficiency of our SAI smoothers and the Gauss-Seidel smoother using an eigen-analysis. Figure 5.3 shows the errors after two smoothing steps of different smoothers. The x-axis represents the eigencomponents with respect to the Laplacian matrix, and the y-axis represents their magnitudes. For Laplacian matrices, the small and large eigenvalues correspond to the low and high frequencies, respectively. From the plots, there is no significant difference between the (1,0)-level local smoother and the simplified smoother as explained by the Fourier analysis above. Moreover, they both damp away high frequencies more effectively than the Gauss-Seidel smoother.

5.4 Complexity

Let n_k be the average number of indices in the k-level neighbor sets $\mathcal{L}_k(o_i)$. Then the number of nonzeros of the sparse approximate inverse given by the (k,l)-level SAI smoother is $n_l n$, which is also the complexity of applying one step of the (k,l)-level SAI smoother. For instance, the complexity of the (1,0)-level SAI

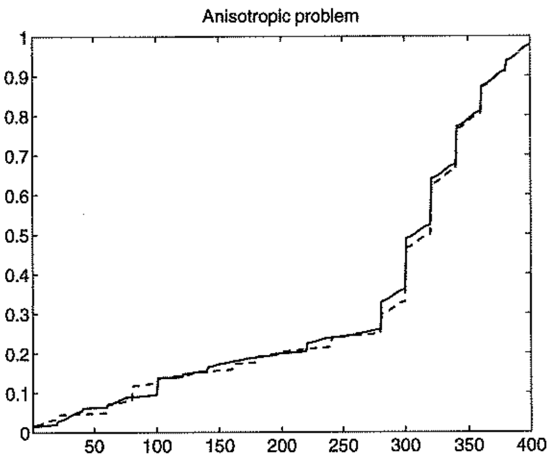


Figure 5.2: Comparison of the eigenvalue distributions. The solid line denotes the eigenvalues of the simplified smoother and the dashed line denotes the eigenvalues of the (1,0)-level local smoother.

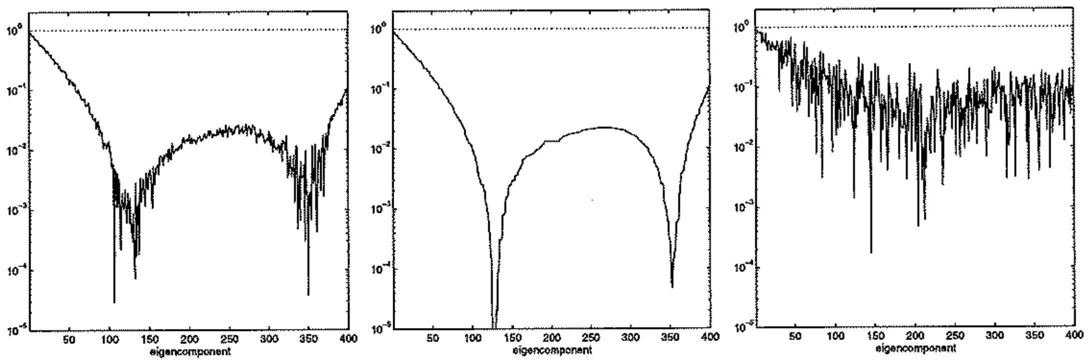


Figure 5.3: Errors after two smoothing steps of (a) (1,0)-level local smoother, (b) simplified smoother, (c) Gauss-Seidel smoother. The dotted line denotes the initial error.

and Gauss-Seidel smoothers is the same as one matrix-vector multiply. Another additional cost of the SAI smoothers compared to Gauss-Seidel is the construction cost. For the SAI smoothers, we have to solve a least squares problem of size $n_k \times n_l$, $n_k \geq n_l$, for each column. Thus, the construction complexity is: $O(n_k n_l^2 n)$. For the simplified SAI smoothers, since we solve one least squares problem only, the complexity is reduced to: $O(n_k n_l^2)$. Hence, for low level methods where k and l are small, this extra cost can be easily compensated by the faster multigrid convergence.

5.5 Numerical Results

In this section, we demonstrate the effectiveness of the proposed SAI smoothers. Examples 1-4 show that the SAI smoothers work when the Gauss-Seidel smoothers do or do not work. Example 5 shows further that we may improve the quality of the SAI smoothers by using higher level of fills in the approximate inverses for anisotropic problems. Thus the SAI smoothers are more robust than the Gauss-Seidel smoothers.

In all the examples, Dirichlet boundary conditions are used. In the multigrid procedure, a V-cycle is used with one pre- and one post- smoothing, unless otherwise stated. Linear interpolation is used for structured grid problems and a specialized energy-minimizing interpolation [140, 141] is used for unstructured grid problems. The number of multigrid levels is such that the coarsest grid is 3×3 for structured square grid problems, and a total of four levels for general unstructured grid problems. The iteration was terminated when the relative residual norm was less than 10^{-8}. The results are summarized in the form of a table where the number of V-cycles and the average convergence rate are shown.

Example 1: We compare the performance of different Gauss-Seidel and SAI smoothers by solving the Laplace equation on a 33×33 square grid. The results are shown in Table 5.1. We see that SAI is slightly better than GS in addition to the fact that it is much easily parallelized. In this simple case, SAI is slightly worse than GS(rb); maybe because of the special geometry. We will see in Example 4 that SAI performs better for unstructured grid problems. As described in Section 5.2, "SAI(1 pt)" is the low cost simplified version of SAI for constant coefficient PDEs, and it essentially performs the same as SAI.

Example 2: Table 5.2 shows that result for a discontinuous coefficient problem with a square interface on a 33×33 square grid. The jump across the interface is 10^4. We get a similar result as before. However, this time SAI(1 pt) does not work since the coefficient is not constant anymore, and in fact it changes enormously across the interface.

Example 3: In this example, we present the results for a number of variable coef-

Smoothers	Iteration	Conv. Rate
GS	14	0.25
GS(rb)	10	0.16
SAI	12	0.19
SAI(1 pt)	13	0.21

Table 5.1: Laplace equation. GS: Gauss-Seidel with natural ordering, GS(rb) Gauss-Seidel with red-black ordering, SAI: sparse approximate inverse smoother, SAI(1 pt): SAI generated by a single interior point (see Section 5.2).

Smoothers	Iteration	Conv. Rate
GS	16	0.30
GS(rb)	13	0.22
SAI	13	0.22
SAI(1 pt)	*	*

Table 5.2: A square interface problem. GS: Gauss-Seidel with natural ordering, GS(rb) Gauss-Seidel with red-black ordering, SAI: sparse approximate inverse smoother, SAI(1 pt): SAI generated by a single interior point (see Section). * indicates convergence more than 50 iterations.

ficient problems including a helical spring problem, a viscosity PDE problem, and a discontinuous coefficient problem where the interfaces now consist of a horizontal and a vertical line; see Chapter 3 for details of each problem. Table 5.3 shows that SAI may converge while the other two do not.

Problems	Iteration			Conv. Rate		
	GS	GS(rb)	SAI	GS	GS(rb)	SAI
Variable coeff.	10	8	8	0.14	0.08	0.08
Helical spring	9	7	7	0.11	0.06	0.07
Viscosity	9	7	7	0.11	0.06	0.07
Discont. coeff.	*	*	22	*	*	0.40

Table 5.3: Variable coefficient problems. GS: Gauss-Seidel with natural ordering, GS(rb) Gauss-Seidel with red-black ordering, SAI: sparse approximate inverse smoother. * indicates convergence more than 50 iterations.

Example 4: We show the effects of unstructured grids (Figure 5.4) on the smoothers by solving the Laplace equation. In this case, red-black ordering is not defined. Instead, we use a *generalized* red-black ordering for general sparse matrices, which is essentially a greedy coloring algorithm. We still denote the resulting Gauss-Seidel by GS(rb). Table 5.4 shows that SAI is slightly better than GS(rb), which is better than GS.

Example 5: We show how SAI smoothers can improve the convergence of multi-

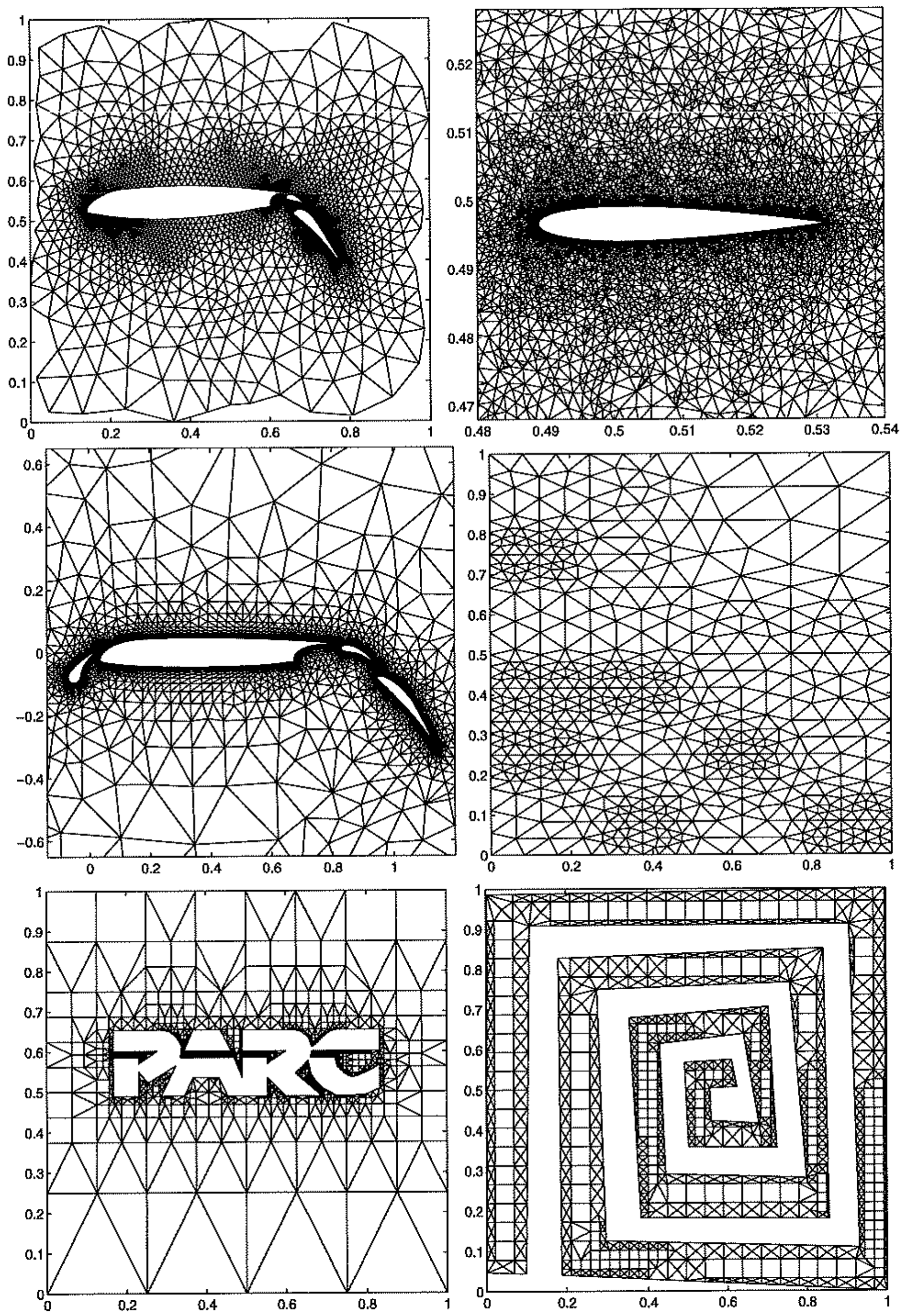


Figure 5.4: The unstructured grids for Example 4 and 5. Row 1 left: airfoil1, row 1 right: airfoil2, row 2 left: airfoil3, row 2 right: square, row 3 left: parc, row 3 right: spiral.

Grids	Iteration			Conv. Rate		
	GS	GS(rb)	SAI	GS	GS(rb)	SAI
airfoil1	16	15	14	0.35	0.32	0.29
airfoil2	15	14	13	0.40	0.39	0.35
airfoil3	27	26	27	0.61	0.59	0.61
square	16	13	11	0.36	0.28	0.20
parc	18	11	10	0.36	0.18	0.16
spiral	12	9	8	0.22	0.11	0.08

Table 5.4: Laplace equation on different unstructured grids. GS: Gauss-Seidel with natural ordering, GS(rb) Gauss-Seidel with *generalized* red-black ordering, SAI: sparse approximate inverse smoother.

grid for solving anisotropic coefficient PDEs. We consider two problems:

Problem 1: The single-direction anisotropic problem in (5.6).

Problem 2: There are anisotropic structures in both the x and y directions:

$$a(x,y)u_{xx} + b(x,y)u_{yy} = 1,$$

where the coefficients $a(x,y)$ and $b(x,y)$ are defined as:

$$a(x,y) = \begin{cases} 100 & (x,y) \in [0,0.5]\times[0,0.5] \text{ or } [0.5,1]\times[0.5,1] \\ 1 & \text{otherwise.} \end{cases}$$

$$b(x,y) = \begin{cases} 100 & (x,y) \in [0,0.5]\times[0.5,1] \text{ or } [0.5,1]\times[0,0.5] \\ 1 & \text{otherwise.} \end{cases}$$

The convergence results are shown in Table 5.5 and 5.6. The first three rows show the results of problem 1 on a 32×32, 64×64 and 128×128 grid, respectively. Similarly, row 4 and 5 show the results of problem 2. The last 4 rows show the results of problem 1 on unstructured grids.

As it is well-known, multigrid with Gauss-Seidel smoother is very slow for these problems, except for the results on the unstructured grids airfoil1 and airfoil2. Similar results are also obtained for Gauss-Seidel with red-black ordering, and hence its results are omitted. For problem 1 on a square grid, a usual technique to improve the multigrid convergence is to use block relaxation methods. As indicated in the table, block Jacobi is quite effective for small grids, but eventually slows down for bigger grids. Moreover, it is costly to invert each block. The previous (1,0)-level SAI smoother is not very effective in this case. We improve the performance by using higher level SAI smoothers (Section 5.2). For higher levels, however, the approximate inverse is much denser. We control the amount of fill-in by dropping small elements. $SAI(k,\epsilon)$ denotes the smoother with $(k+1,k)$-level and element whose absolute values below ϵ are dropped at the end. We may also drop small elements in the matrix A before we compute the approximate inverse. $SAI(\epsilon_1,k,\epsilon_2)$

indicates that elements of size $< \epsilon_1$ are dropped in the matrix A, and elements of size $< \epsilon_2$ are dropped in the the approximate inverse.

Grids	Iteration				
	GS	BJ	SAI(3,ϵ_2)	SAI(4,ϵ_2)	SAI(ϵ_1,4,ϵ_2)
problem 1 (32×32)	$*$	8	21	15	15
problem 1 (64×64)	$*$	27	27	20	19
problem 1 (128×128)	$*$	$*$	37	27	27
problem 2 (32×32)	$*$	$*$	49	13	13
problem 2 (64×64)	$*$	$*$	25	20	19
airfoil1	18	–	9	8	8
airfoil2	23	–	12	12	12
parc	46	–	11	9	10
spiral	$*$	–	18	21	21

Table 5.5: Anisotropic problems on different grids. GS: Gauss-Seidel with natural ordering, BJ: Block Jacobi, ϵ_1=2, ϵ_2= 0.0004 for the airfoil2 and spiral grid and 0.0008 for others. See text for the definitions of SAI(3,ϵ_2) and SAI(ϵ_1,4,ϵ_2).

Table 5.5 and 5.6 shows that the (4,3)-level SAI performs similarly as block Jacobi for problem 1 on structured grids and better for the other problems. The (5,4)-level SAI performs the best in general. Also, the SAI smoother does not deteriorate if the small elements of A are dropped first. Hence, we may afford to use higher level SAI if we drop small elements beforehand.

For problem 2, because of the anisotropy in both directions, block Jacobi does not improve the multigrid convergence. The SAI smoothers, which do not require the concept of *direction*, perform as well as in problem 1. This feature allows us to capture the anisotropy easily by adjusting only one number–the level of fill-in, with no need to track geometrically the anisotropic directions which are often hard to determine.

For unstructured grid problems, blocks defined along the direction of the anisotropy no longer exist. Thus we do not test block Jacobi in these problems. Gauss-Seidel does surprisingly well on some of the grids, but still very slow on the others. SAI smoothers are effective on all the grids.

Example 6: Finally, we show that SAI, like other standard smoothers, give rise to a multigrid method whose convergence rate is independent of the mesh size; see Table 5.7.

5.6 Concluding Remarks

We have presented a class of robust sparse approximate inverse smoothers for multigrid. They are simple to construct and yet are robust for PDE problems. For

Grids	Conv. Rate				
	GS	BJ	SAI(3,ϵ_2)	SAI(4,ϵ_2)	SAI(ϵ_1,4,ϵ_2)
problem 1 (32×32)	$*$	0.09	0.52	0.40	0.40
problem 1 (64×64)	$*$	0.62	0.51	0.51	0.50
problem 1 (128×128)	$*$	$*$	0.64	0.53	0.53
problem 2 (32×32)	$*$	$*$	0.77	0.31	0.31
problem 2 (64×64)	$*$	$*$	0.59	0.51	0.48
airfoil1	0.38	–	0.10	0.09	0.09
airfoil2	0.61	–	0.30	0.37	0.37
parc	0.71	–	0.19	0.11	0.13
spiral	$*$	–	0.37	0.47	0.47

Table 5.6: Anisotropic problems on different grids. GS: Gauss-Seidel with natural ordering, BJ: Block Jacobi, ϵ_1=2, ϵ_2= 0.0004 for the airfoil2 and spiral grid and 0.0008 for others. See text for the definitions of SAI(3,ϵ_2) and SAI(ϵ_1,4,ϵ_2).

Grids	Iteration			Conv. Rate		
	GS	GS(rb)	SAI	GS	GS(rb)	SAI
32×32	14	10	13	0.25	0.16	0.21
64×64	14	11	13	0.25	0.17	0.22
128×128	14	11	13	0.25	0.17	0.22

Table 5.7: Laplace equation on grids of different mesh size. GS: Gauss-Seidel with natural ordering, GS(rb) Gauss-Seidel with red-black ordering, SAI: sparse approximate inverse smoother.

constant coefficient problems, we gave a simplified version of SAI. We analyzed theoretically and numerically that the simplified SAI is essentially the same as SAI. In the numerical experiments, we have demonstrated that SAI is robust for discontinuous coefficient problems, anisotropic problems and unstructured grid problems. For anisotropic problems, we have shown how the quality of SAI can be improved by increasing the number of levels whereas other smoothers such as Gauss-Seidel do not have such property.

CHAPTER 6

Interface Preserving Coarsening for Highly Discontinuous Coefficient PDEs

6.1 Introduction

The standard multigrid method converges slowly for PDE problems whose coefficients have jumps of several orders of magnitude [1, 6, 46, 73, 86]. The finite element/difference discretization has to be handled carefully [4, 91, 127]. Despite this, suppose the discretization is properly done. We want to derive a multigrid method which is as insensitive to the jumps as possible.

A typical approach of improving the convergence is to define a sophisticated interpolation such as in black box multigrid [46, 47]. The idea is to capture the discontinuous behavior of the derivative of the solution along the interfaces, or equivalently to preserve the continuity of the flux across the interfaces [1, 46, 47, 73]. However, these methods usually only apply to structured grid problems.

As discussed in Chapter 4, a successful multigrid method depends not just on interpolation but on all components as a whole. Our key observation is that coarsening can play a crucial role for interface problems. Specifically, the coarse grid points should resolve the shape of the interface in a certain sense to be described later. Intuitively speaking, experiences from the literature [1, 65, 73] indicated that the parts of the solution on the regions of different constant coefficients behave independently and are glued together through a Neumann boundary condition on the interfaces. Theoretically speaking, convergence analysis for interface problems [16, 49, 148] often require that the discontinuities are preserved on all coarser grids. In view of these, we propose an interface preserving coarsening algorithm so that all the coarse grids will align with the interfaces for regular interface problems on structured grids, and that the interfaces are resolved as much as possible for irregular interface problems. Consequently, linear interpolation is sufficient to obtain fast multigrid convergence.

We remark that special coarsening techniques are quite common for anisotropic problems. However, to the best of our knowledge, no coarsening strategy has been studied specifically for discontinuous coefficient problems. In algebraic multigrid [112], coarsening is also done specially according to the notion of *strong coupling.* However, their motivation is purely algebraic, and no geometric interpretation is given. We also remark that much work has been done on domain decomposition for discontinuous coefficient problems; see, for example [30, 121]. For many of these methods, the subdomains are naturally divided by the interfaces. Thus the true interfaces coincide with the computational subdomain interfaces. In multigrid, however, we do not usually have large subdomains and hence the interfaces do not generally align with the coarse grids.

Another approach for discontinuous coefficient problems is to use conjugate gradient preconditioned by diagonal scaling [66, 108]. It is very simple and yet quite effective. Graham and Hagger [65] analyzed such approach theoretically, and extended the analysis to additive Schwarz preconditioners. They showed that the largest eigenvalues of the preconditioned system are bounded and that only a finite number of small eigenvalues approach to zero as the size of the jump increases. Thus the preconditioned conjugate gradient method converges with a number of iterations which grows only logarithmically in the size of the jump. In their analysis, the interfaces are not assumed to align with the coarse grids. Another related approach by Ashby et. al. [2] is to use multigrid preconditioned conjugate gradient.

In Section 6.2, we first discuss the issues of coarsening for discontinuous coefficient problems and explain why sometimes it is mandatory to select coarse grid points in a special way. We illustrate the idea in one dimension in Section 6.3. In two dimensions, we discuss the regular and irregular interface case separately in Section 6.4.1 and Section 6.4.2, respectively. The effectiveness of interface preserving coarsening is demonstrated by one-dimensional and two-dimensional examples in Section 6.5. Finally, concluding remarks are made in Section 6.6.

6.2 Failure of Multigrid with Standard Coarsening

As mentioned in the previous section, robust interpolations have been the key for improving multigrid convergence for discontinuous coefficient problems. However, in some circumstances, especially in unstructured grid computations, robust interpolation is not enough. Sometimes, it is mandatory to employ special coarsening strategy.

Consider a square interface on a 5×5 triangular mesh as shown in Figure 6.1(a). The circles denote the coarse grid points chosen by standard coarsening. The jump in coefficient is indicated by the shaded area, and zero Dirichlet boundary condition is used. Since it is a triangular mesh, it is natural to interpolate the noncoarse grid points by the two (and only two) connected coarse grid points. The robust energy-minimizing interpolation described in Chapter 7 is used along with the Gauss-Seidel smoothing. The slow multigrid convergence is shown in Figure 6.1(b). The key observation is that the interpolated values at the two noncoarse grid points marked by • are far from the true values of the errors. Figure 6.2(a) and (b) show the error after smoothing and the error given by interpolation. Due to the large jump in coefficient, the error has larger values at the region with large

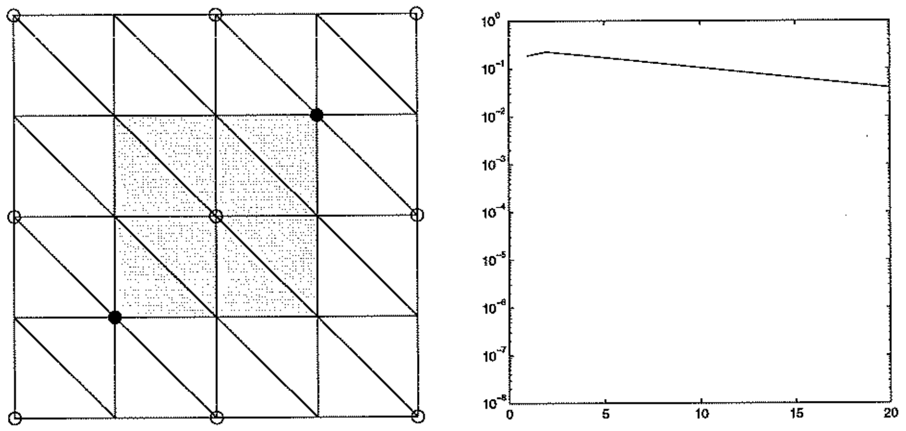


Figure 6.1: (a) A 5×5 grid with a jump in coefficient shown by the shaded area. Coarse grid points are denoted by $\circ$. (b) Standard multigrid convergence.

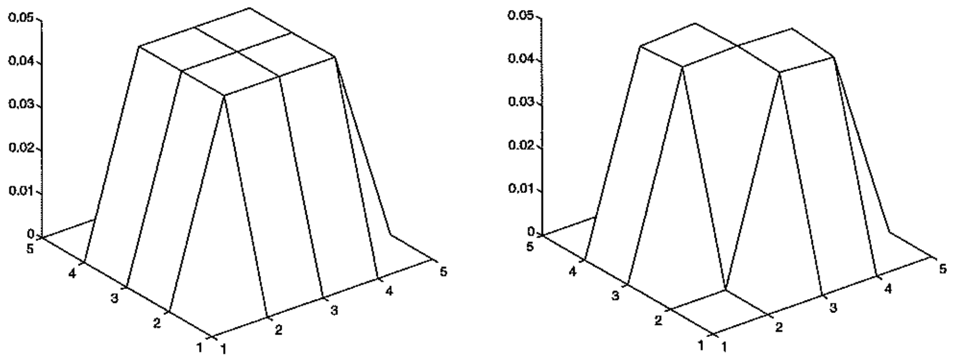


Figure 6.2: (a) Error after smoothing. (b) Error given by interpolation with standard coarsening.

coefficient and smaller values at the region with small coefficient. On the other hand, the interpolated values at the two corners are zero since their connected coarse grid points are on the boundary where the values are zero. In fact, even if other more sophisticated interpolations were used, the interpolated values had to be zero as long as they were interpolated only by the two connected coarse grid points on the boundary. Thus the approximation of the error on the coarse grid is very poor which leads to a slow multigrid convergence.

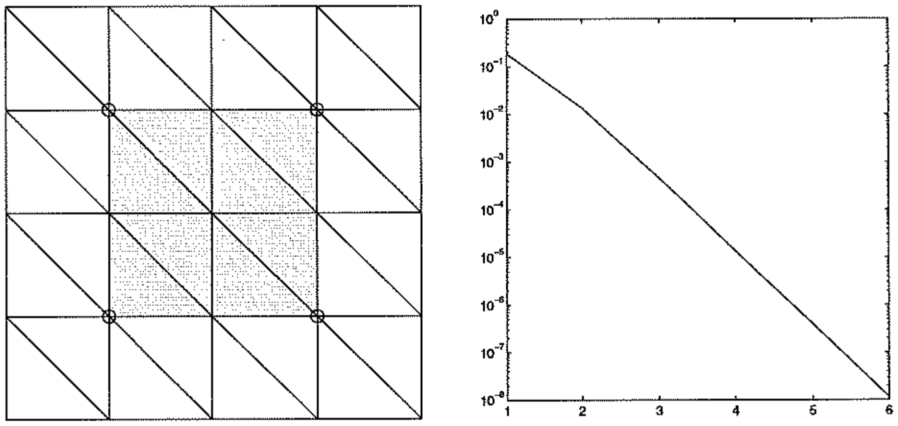


Figure 6.3: (a) Nonstandard coarsening, denoted by o, at the four corners of the square interface. (b) Standard multigrid convergence.

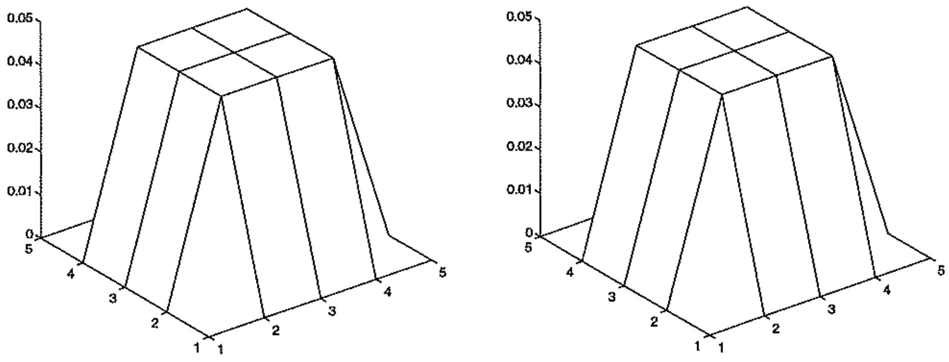


Figure 6.4: (a) Error after smoothing. (b) Error given by interpolation with non-standard coarsening.

Based on this observation, a natural remedy is to select coarse grid points so that the previous situation will not happen. For instance, if we choose the coarse grid points shown in Figure 6.3(a), along with linear interpolation and Gauss-Seidel smoothing, we obtain the usual rapid multigrid convergence (Figure 6.3(b)). As shown in Figure 6.4(a) and (b), the true error and the interpolated error are essentially the same, and hence the fine grid errors can be corrected accurately by the coarse grid. In conclusion, a special coarsening is required to maintain the efficient multigrid convergence in this case.

6.3 Interface Preserving Coarsening: One Dimension

The idea of the interface preserving coarsening is very intuitive. Since we want to align the interfaces with the coarse grids, we simply assign the points at the interfaces to be the coarse grid points. After that, for the remaining points, we perform standard coarsening; we select every other points as coarse grid points (Figure 6.5). The idea can be easily extended recursively to coarser grids (Figure 6.6). The algorithm is described mathematically as follows.

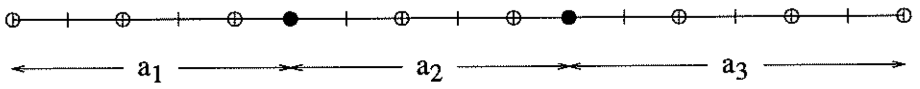


Figure 6.5: One-dimensional interface preserving coarsening. The regions of different coefficient are denoted by a_1, a_2 and a_3. Coarse grid points are denoted by o and the coarse grid points at the interfaces are denoted by •.

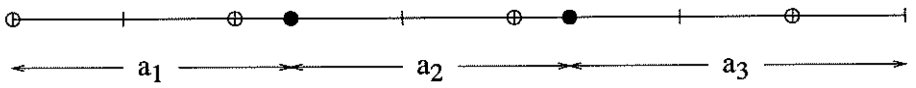


Figure 6.6: One-dimensional interface preserving coarsening on the coarse grid points given by Figure 6.5.

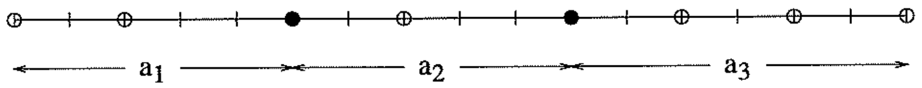


Figure 6.7: One-dimensional interface preserving coarsening with optional Step 2 (cf. Algorithm 6.1) in effect. No two consecutive points are coarse grid points.

Let $N^k = \{n_1^k, \ldots, n_{m_k}^k\}$ be the set of fine grid points at level k. Let N_C^k be the set of coarse grid points and $N_F^k = N^k \backslash N_C^k$ be the set of noncoarse grid points. The algorithm consists of the following three steps; see Algorithm 6.1.

Step 1 selects the interface points as coarse grid points. Step 2 ensures that no two coarse grid points are adjacent to each other unless they are both interface points. Thus the number of coarse grid points is not more than half of the fine grid points. The difference of applying and not applying Step 2 is shown in Figure 6.7 and Figure 6.5, respectively. The former has fewer coarse grid points but more

```
Algorithm 6.1: 1D Interface Preserving Coarsening

1. Set $N_C^k = \{$ interface points $\} \equiv \{n_{i_1}^k, n_{i_2}^k, \ldots, n_{i_p}^k\}$.
   Assume $n_{i_0}^k = n_1^k$ and $n_{i_{p+1}}^k = n_{m_k}^k$.
2. (optional) Set $N_F^k = \{$ neighbors of $j : j \in N_C^k\}$.
3. for $l = 0$ to $p$
     for $j = n_{i_l}^k$ to $n_{i_{l+1}}^k$
       if $j \notin N_F^k \cup N_C^k$ then
         $N_C^k = N_C^k \cup \{j\}$,   $N_F^k = N_F^k \cup \{$ neighbors of j $\}$
       end if
     end for
   end for
```

overlaps between basis functions and the opposite for the latter. Our default is not to apply Step 2. Step 3 performs the standard coarsening between the selected coarse grid points from Step 1.

The other multigrid components are standard. We use Gauss-Seidel as smoother and linear interpolation. We remark that the coordinate information of the computational points is needed to perform the linear interpolation due to the nonuniformity in spacing of the coarse grid points.

6.4 Interface Preserving Coarsening: Two Dimensions

We describe separately the algorithm for the regular and irregular interface. For the former, we may apply the one-dimensional technique. For the latter, we need another algorithm to maintain low complexity while still being able to resolve the interfaces.

6.4.1 Regular Interfaces

Suppose the grids are regular tensor product grids, and the regions of different coefficients are also formed by tensor products. More precisely, let

$$\begin{aligned} X &= \{x : x \text{ is the x-coordinate of an interface point}\}, \\ Y &= \{y : y \text{ is the y-coordinate of an interface point}\}. \end{aligned}$$

Let Ω_i be a region with a large jump in coefficient. If it is formed by a tensor product, then $\Omega_i = [x_1, x_2] \times [y_1, y_2]$ for some $x_1, x_2 \in X$ and $y_1, y_2 \in Y$. Figure 6.8 shows an example of a regular interface problem.

The two-dimensional regular interface preserving coarsening is obtained by a "tensor product" of the one-dimensional algorithm: A point $x \in X^k$ is an

```
Algorithm 6.2: 2D Regular Interface Preserving Coarsening

Let $\Omega^k$ be the set of grid points on level $k$.
1. Set $X^k = \{x : (x,y) \in \Omega^k\}$, $Y^k = \{y : (x,y) \in \Omega^k\}$.
2. Set $N^k_{Cx}$ = coarse grid points obtained from 1D coarsening on $X^k$.
3. Set $N^k_{Cy}$ = coarse grid points obtained from 1D coarsening on $Y^k$.
4. Set $N^k_C = \{(x,y) : x \in N^k_{Cx},\ y \in N^k_{Cy}\}$.
```

interface point if there exist a region with a large jump in coefficient Ω_i such that $\Omega_i = [x, \tilde{x}] \times [y, \tilde{y}]$ for some $\tilde{x} \in X^k$ and $y, \tilde{y} \in Y^k$. Thus $\tilde{x}$ is also an interface point in X^k. The interface points in Y^k are defined similarly. A result of the regular interface preserving coarsening is illustrated in Figure 6.8. We note that an extension of the algorithm to three dimensions is straightforward.

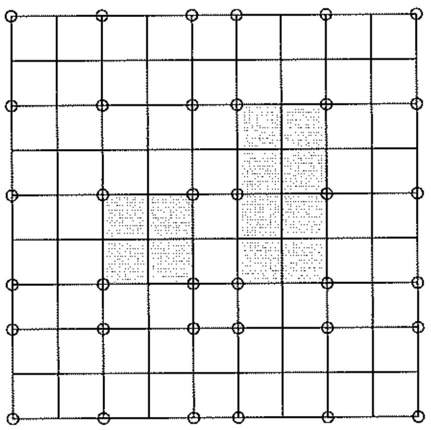


Figure 6.8: Tensor product interface preserving coarsening for a regular two-dimensional interface. Coarse grid points are denoted by o.

6.4.2 Irregular Interfaces

We can still apply the regular techniques if the interfaces are not too irregular. For example, for the interface shown in Figure 6.9(a), we may apply Algorithm 6.2 to obtain a coarsening of the grid points; we collect the x and y coordinates and determine their interface points as before, and then coarsen along the x and y direction independently. Figure 6.9(a) shows the resulting coarse grid points. For less regular interfaces, however, this tensor product procedure may create excessive number of coarse grid points to preserve the interfaces; see Figure 6.9(b). Consequently, the overall computational cost may increase.

In general, we want to resolve the shape of the interface while maintaining the coarse to fine grid point ratio to be 1:4 as closely as possible. The idea is as

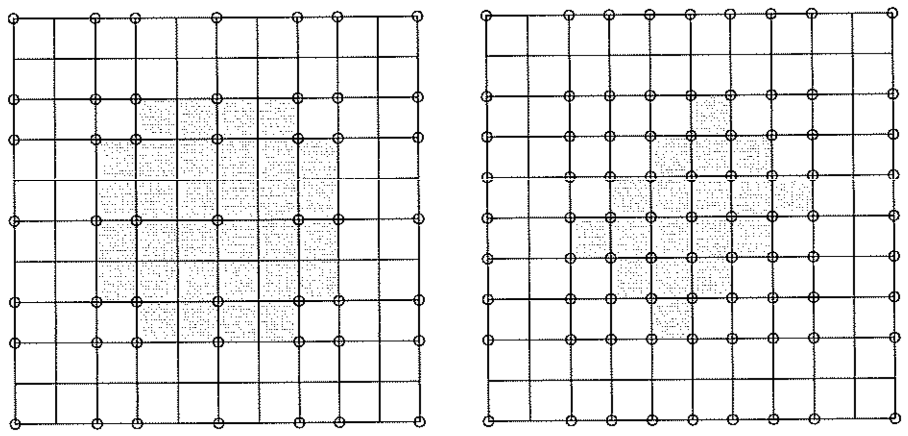


Figure 6.9: Tensor product interface preserving coarsening for (a) a less irregular interface, (b) very irregular interface. Coarse grid points are denoted by $\circ$.

follows. We first identify the points where the interfaces are located. Then we apply standard coarsening to these points and then to the remaining points. We convert some of the noncoarse grid points near the interfaces to coarse grid points so that the shapes of the interfaces are better resolved. We have not yet derived a rule of conversion which can be justified rigorously. Nevertheless, we suggest two heuristics to be discussed later.

The above procedures can be described mathematically as follows. Let Ω^+, Ω^- be disjoint open subsets of Ω such that $\Omega = \overline{\Omega^+} \cup \Omega^-$. Let $\Gamma = \partial\Omega^+$ be the interface and $\Gamma \cap \partial\Omega = \phi$. Let $a(x,y) \equiv a^+$ in Ω^+ and $a(x,y) \equiv a^-$ in Ω^- and $a^- \leq a^+$. The case of multiple interfaces is treated similarly and hence is omitted. The two-dimensional irregular interface preserving coarsening is given in Algorithm 6.3.

Remarks:
Step 1: Notice that the diagonal entries of the stiffness matrix $\mathcal{A}$ is

$$\mathcal{A}^h_{ii} = a^+ \int_\Omega |\nabla \phi^h_i|^2 dx,$$

for all $x^h_i \in N^+$. If $a^+ \gg a^-$, the set of points in N^+ can be easily identified by the large diagonal entries of $\mathcal{A}^h_{ii}$.

Step 2, 3: They are just standard coarsenings on the points in N^+ first followed by those in N^-.

Step 4: Two sets of heuristic criteria are suggested to ensure that the noncoarse grid points on Γ are properly interpolated so that the discontinuous derivative behavior of the solution is captured. Criterion a(i) & (ii) require noncoarse grid points to be interpolated by at least one coarse grid point in the same region. These conditions eliminate the situation given by the example in Section 6.2. The

Algorithm 6.3: 2D Irregular Interface Preserving Coarsening

```
1. Determine the set of fine grid points $N^+$ in $\overline{\Omega^+}$.
2. Full coarsening on $N^+$.
3. Full coarsening on $N^- = N \backslash N^+$, $N$=set of fine grid points.
4. Either (a) or (b) is used. Change a noncoarse grid point $x_i^h$ to a coarse grid point,
   if any of the condition is satisfied.
   Criterion set (a):
   i.   $x_i^h \in N^+$ and no coarse grid point $x_j^H \in N^+$ is connected to $x_i^h$.
   ii.  $x_i^h \in N^-$ and no coarse grid point $x_j^H \in N^-$ is connected to $x_i^h$.
   iii. $x_i^h$ is connected to one coarse grid point only.
   iv.  Coarse grid points in $N^+$ do not interpolate those in $N^-$, and vice versa.
   Criterion set (b):
   i.   $x_i^h \in N^+$ and less than two coarse grid points $x_j^H \in N^+$ connected to $x_i^h$.
   ii.  $x_i^h \in N^-$ and less than two coarse grid points $x_j^H \in N^-$ connected to $x_i^h$.
   iii. $x_i^h \in \partial\Omega$ and no $x_j^H \in \partial\Omega$ is connected to $x_i^h$.
   iv.  Coarse grid points in $N^+$ do not interpolate those in $N^-$, and vice versa.
```

purpose of criterion a(iii) is to take care of a special case in unstructured grid computations rather than for resolving the interfaces. If a noncoarse grid point is connected to one coarse grid only, the interpolated weight at the noncoarse grid point must be 1 due to the constant preserving constraint. In other words, we have a local piecewise constant instead of piecewise linear interpolation there which may affect convergence. Criterion a(iii) eliminates this case by making such noncoarse grid points to coarse grid points. In Criterion set (b), the first two criteria serve a similar purpose as a(i) & (ii), except that they require more coarse grid point connections in the same region. Criterion b(iii) requires that noncoarse grid points on the boundary must be connected to at least one coarse grid point on the boundary. For both criterion sets, we eliminate the influence of coarse grid points from the other region by criterion a(iv) & b(iv); we allow interpolation from coarse grid points to noncoarse grid points in the same region only.

Figure 6.10 shows an example of the irregular interface preserving heuristics. Since Algorithm 6.3 is based on the graph connection of a matrix, we use here a triangular mesh for a better illustration. Compared to Figure 6.9(b), we need much fewer coarse grid points. Moreover, within each constant coefficient region, the coarse grid points are selected as if in standard coarsening. More coarse grid points, especially in the case when the criterion set (b) is used, are selected near the interface to resolve its shape.

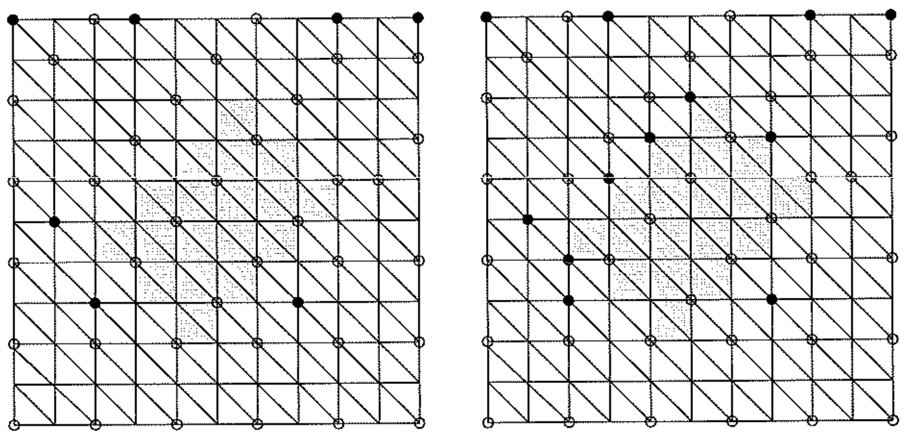


Figure 6.10: Irregular interface preserving coarsening for an irregular interface using (a) Criterion set (a), (b) Criterion set (b), in Algorithm 6.3. Coarse grid points are denoted by o. The coarse grid points selected by either criterion set are denoted by •.

6.4.3 Complexity Issue

Resolving the interface, in general, increases the total number of coarse grid points which in turn increases the overall computational cost. However, if the interface forms a simple piecewise smooth curve, the increase in the number of coarse grid points is at most the total number of grid points on the interface, which is only $O(\frac{1}{h})$ compared to $O(\frac{1}{h^2})$ total number of coarse grid points. Suppose the complexity, η_0, of one V-cycle multigrid with standard coarsening is estimated to be:

$$\eta_0 = O(\frac{1}{h^2} + \frac{1}{4h^2} + \frac{1}{16h^2} + \ldots) = O(\frac{4}{3h^2}).$$

Then, the complexity, η_1, of one V-cycle multigrid with our special coarsening is:

$$\begin{aligned}\eta_1 &= O(\frac{1}{h^2} + (\frac{1}{4h^2} + \frac{1}{h}) + (\frac{1}{4}(\frac{1}{4h^2} + \frac{1}{h}) + \frac{1}{2h}) + (\frac{1}{4}(\frac{1}{4}(\frac{1}{4h^2} + \frac{1}{h}) + \frac{1}{2h}) + \frac{1}{4h}) + \ldots) \\ &= O(\frac{4}{3h^2} + \frac{8}{3h}).\end{aligned}$$

Thus the extra amount of work due to the increase in the number of coarse grid points is asymptotically small compared to the standard one.

6.5 Numerical Results

We demonstrate the effectiveness of the interface preserving coarsening by three examples. The multigrid settings are standard. We apply two pre- and post-

Gauss-Seidel smoothings. Linear interpolation is used for the one-dimensional problem and the two-dimensional regular interface problem. For the irregular interface problem, since the tensor product grid structure is destroyed after one level of coarsening, we have to use unstructured grid multigrid techniques for the subsequent coarser grids. In particular, we use the energy-minimizing interpolation described in Chapter 7. The multigrid iteration was terminated when the relative residual norm was less than 10^{-6}.

Example 1: We compare the convergence results of the multigrid method with the one-dimensional interface preserving coarsening and linear interpolation and that with the standard coarsening and flux preserving interpolation [73]. The model equation is

$$\begin{aligned} -\frac{d}{dx}a(x)\frac{d}{dx}u(x) &= 1 \qquad \text{in } (0,1) \\ u &= 0 \qquad \text{at } x = 0 \text{ and } x = 1, \end{aligned}$$

where

$$a(x) = \begin{cases} 10^4 & \text{if } x \le 1/4 + h \\ 1 & \text{if } 1/4 + h < x \le 1/2 + h \\ 10^2 & \text{if } x > 1/2 + h. \end{cases}$$

Here, h is the size of the fine grid. It is so designed that the interfaces will not align with any standard coarse grids. The convergence results of the two multigrid methods are shown in Table 6.1. There is no difference in performance for both methods. Moreover, their convergence results are independent of the mesh size h and the jump in the coefficient.

h	MG Method 1	MG Method 2	Standard MG
1/32	6	6	16
1/64	6	6	17
1/128	6	6	20
1/256	6	6	24

Table 6.1: Convergence of multigrid methods for a 1D discontinuous coefficient problem. Method 1 uses interface preserving coarsening and linear interpolation. Method 2 uses standard coarsening and flux preserving interpolation.

Example 2: We show the effectiveness of the two-dimensional tensor product interface preserving coarsening for regular interfaces. The equation is extracted from Example 3 in Chapter 7:

$$-\nabla \cdot a(x,y)\nabla u = 1,$$

where

$$a(x,y) = \begin{cases} a^+ & 0.25 \le x \le 0.75 \quad \& \quad 0.25 \le y \le 0.75 \\ a^- & \text{otherwise.} \end{cases}$$

We fix $a^- = 1$ and vary a^+ from 10 to 10^4. The convergence results of the multigrid method with interface preserving interface and linear interpolation and that with standard coarsening and energy-minimizing interpolation are shown in Table 6.2. Again, their performances are essentially the same, and their convergences do not depend on the mesh size nor the size of the jump.

	MG Method 1			MG Method 2			Standard MG		
h	10	10^2	10^4	10	10^2	10^4	10	10^2	10^4
1/16	5	5	6	6	5	5	14	$*$	$*$
1/32	5	6	6	6	6	6	14	$*$	$*$
1/64	6	6	6	6	6	6	14	$*$	$*$
1/128	6	6	6	7	6	6	14	$*$	$*$

Table 6.2: Convergence of multigrid methods for a 2D discontinuous coefficient problem. Method 1 uses interface preserving coarsening and linear interpolation. Method 2 uses standard coarsening and energy-minimizing interpolation. The numbers $10, 10^2, 10^4$ indicate the values of a^+. $*$ denotes convergence more than 100 iterations.

Example 3: Finally, we present the results of an irregular interface problem. The star-shaped interface is shown in Figure 6.11(a), and the irregular interface coarsening with criterion set (b) on the 16×16 triangular mesh is shown in Figure 6.11(b). Since the coefficient is assumed to be constant on each triangle, a wrinkleness is resulted on the interface. We can see that more coarse grid points are selected near the interface and the remaining parts are coarsened in a standard way.

The convergence results of the multigrid method using the irregular interface preserving coarsening and energy-minimizing interpolation are given in Table 6.3. (Note that we cannot use the standard linear interpolation in this case since the regular nested grid structure is destroyed after one level of coarsening.) First, the convergence is independent of the jump. Second, for a fixed number of levels, the convergence is independent of the mesh size. However, the convergence rate does deteriorate with the number of levels. To alleviate this problem, we do not go through many levels, if we can afford to solve exactly, for instance, a coarse grid problem of size 100. In that case, it only needs about 20 iterations to convergence.

The "number of nodes" column shows the number of nodes at each level. As discussed in Section 6.4.2, it is important to maintain a reasonable ratio between the fine and coarse grid points so that the computational cost is not much higher than that of standard multigrid. From the column, we see that the ratio is between 1:3 and 1:4, which is close to the standard ratio 1:4.

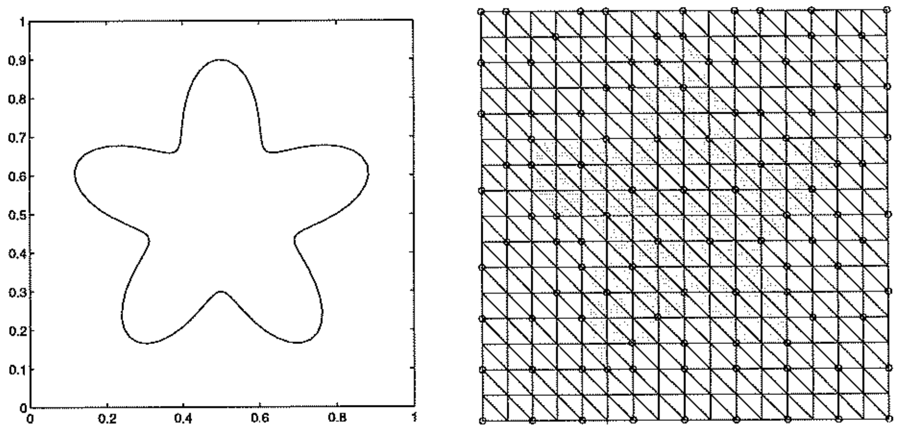


Figure 6.11: (a) A star-shaped interface is formed by: (x, y), $x = 0.5 + r\cos\theta$, $y = 0.5 + r\sin\theta$ where $r = 0.3 + 0.1\sin(5\theta)$, $0 \leq \theta \leq 2\pi$. (b) The star-shaped interface in (a) on a 16×16 triangular mesh.

Grid size	Level	# of nodes	$a^+ = 10$	$a^+ = 10^2$	$a^+ = 10^3$	$a^+ = 10^4$
1/32	2	344	12	13	14	14
	3	116	14	14	15	15
	4	40	20	22	22	22
1/64	2	1261	13	13	13	13
	3	367	14	14	14	14
	4	102	18	18	18	18
	5	28	30	37	39	39
1/128	3	1258	16	16	16	16
	4	319	19	19	19	19
	5	86	27	28	28	28
	6	25	43	52	54	54

Table 6.3: Convergence of the interface preserving coarsening multigrid for the star-shaped interface problem.

6.6 Concluding Remarks

We have demonstrated numerically that coarsening can be an efficient alternative for robust interpolation to solve discontinuous coefficient problems using multigrid. In fact, we have shown by an example that special coarsening sometimes is mandatory no matter what the interpolation is. We have shown that multigrid with the proposed interface preserving coarsening and the simple linear interpolation is an effective solution method for discontinuous coefficient problems on structured grids. In general, we need to combine the interface preserving coarsening and an unstructured grid interpolation for irregular interface problems on general domains, and we have shown by an example how this can be done.

CHAPTER 7

Energy-Minimizing Multigrid

7.1 Introduction

Sophisticated interpolation methods have been the key in designing robust multigrid methods (cf. Chapter 4). In one dimension, a robust interpolation [73, 109, 145] can be obtained by solving local homogeneous PDEs, which are equivalent to minimizing the energy of the coarse grid basis functions. The extension to higher dimensions of this approach is not obvious. Nonetheless, many attempts [1, 46, 73, 72, 86, 110, 145] have been made to set up similar local PDEs for defining a robust interpolation. In place of setting up PDEs, we consider an equivalent minimization formulation and derive a so-called energy-minimizing interpolation with special emphasis on its stability and approximation properties which are essential for optimal convergence. This approach to determining appropriate interpolation operators has also been used for iterative substructuring [50], and algebraic multigrid [134]. It will be made more precise in Section 7.3.

Although it is well-known that the one-dimensional interpolation mentioned above will produce a robust multigrid method, a convergence analysis has not been given in the literature. In Section 7.4, we analyze the one-dimensional method derived from the energy-minimizing interpolation. We prove the novel result that the convergence rate is independent of the coefficient of the underlying PDE, in addition to the mesh size. In Section 7.5, we give numerical examples mainly in two dimensions, including a discontinuous coefficient problem, an oscillatory coefficient problem, and a Helmholtz problem. Finally, we summarize our experience by several remarks in Section 7.6.

We now set up some notations to be used in the following sections. Let $V = V^h$ and $V_1 \subset V_2 \subset \cdots \subset V_J = V$ denote a sequence of nested subspaces of V defined by the span of nodal basis functions, $\{\phi_i^k\}_{i=1}^{n_k}, k = 1, \ldots, J$, at level k. The operator $A : V \rightarrow V$ is self-adjoint and induces the A-inner product: $(\cdot,\cdot)_A \equiv (A\cdot,\cdot)$. Also, we define $A_i : V_i \rightarrow V_i$ by $(A_i u_i, v_i) = (A u_i, v_i), u_i, v_i \in V_i$. Correspondingly, we have $R_i : V_i \rightarrow V_i$, which is an approximate inverse of A_i. Let $Q_i : V \rightarrow V_i$ and $P_i : V \rightarrow V_i$ be the projection operators with respect to the L^2 and the A inner product respectively. In the following analysis, the generic constant C is independent of the mesh size h.

7.2 Stability and Approximation Property

Before we explain the formulation of the energy-minimizing interpolation, we first discuss our motivation from the classical results of multigrid and domain decomposition methods. Two key properties: stability and approximation, must be satisfied by the coarse subspaces and the smoothers [73] in order to have optimal convergence results. These two terms occurred frequently in the literature but often appear in slightly different forms. For example, in the subspace correction framework [149], these two properties are built into the estimate of a constant K_0, which in turn is used to prove optimal convergence together with another constant K_1. The definition of K_0 and K_1 are as follows:

K_0*:* For any $v \in V$, there exists a decomposition $v = \sum_{i=1}^{J} v_i$ for $v_i \in V_i$ such that

$$\sum_{i=1}^{J}(R_i^{-1}v_i, v_i) \leq K_0(Av, v), \tag{7.1}$$

where R_i is usually known as the smoother in the multigrid context.

K_1*:* For any $S \subset \{1, \ldots, J\} \times \{1, \ldots, J\}$ and $u_i, v_i \in V$ for $i = 1, \ldots, J$,

$$\sum_{(i,j)\in S}(T_i u_i, T_j u_j)_A \leq K_1\Big(\sum_{i=1}^{J}(T_i u_i, u_i)_A\Big)^{\frac{1}{2}}\Big(\sum_{j=1}^{J}(T_j v_j, v_j)_A\Big)^{\frac{1}{2}}, \tag{7.2}$$

where $T_i = R_i A_i P_i$.

Theorem 7.1 *Let E_J be the iteration matrix given by the V-cycle multigrid, i.e.,*

$$u - u^{k+1} = E_J(u - u^k),$$

where u is the exact solution and u^k and u^{k+1} are two consecutive multigrid iterates. Then

$$E_J = (I - T_J)(I - T_{J-1}) \cdots (I - T_1),$$

and

$$\|E_J\|_A^2 \leq 1 - \frac{2 - \omega_1}{K_0(1 + K_1)^2},$$

where $\omega_1 = \max_{1 \leq i \leq J} \rho(R_i A_i)$.

Proof. See [149]. □

By Theorem 7.1, the convergence rate can be improved by producing a smaller K_0 or K_1. In this chapter, we propose an interpolation that will potentially decrease the size of the constant K_0 by reducing its dependence on the coefficients of the underlying elliptic PDE.

As shown in [149], the estimate of K_0 relies on two inequalities:

$$\begin{aligned} \|\tilde{Q}_1 v\|_A^2 + \sum_{k=2}^{J} \|(\tilde{Q}_k - \tilde{Q}_{k-1})v\|_A^2 &\le C_0 \|v\|_A^2, && (7.3)\\ \|(\tilde{Q}_k - \tilde{Q}_{k-1})v\| &\le C_1 h_k \|\tilde{Q}_k v\|_A, \qquad \forall k > 1, && (7.4)\end{aligned}$$

where $\tilde{Q}_k : V \to V_k$ is any linear operator onto V_k.

Inequality (7.3) appears in the Partition Lemma which is well-known in the domain decomposition literature [51, 121]. In the multigrid context, however, this inequality is usually only used implicitly. Intuitively speaking, (7.3) says that given any $v \in V$, we must be able to decompose v into the subspaces such that the total energy of all the pieces v_i is bounded by a small constant factor of the original energy of v. Besides (7.3), we also require that functions on the coarser grids approximate those on the finer ones to at least first order accuracy in h_k. This requirement is quantified by the inequality (7.4). If we have both (7.3) and (7.4), we can bound K_0 by a constant independent of the mesh size h.

Lemma 7.1 *Let* $\omega_0 = \min_{2 \le i \le J} (\rho(A_i)\lambda_{\min}(R_i))$. *Suppose (7.3) and (7.4) are satisfied. Then*

$$K_0 \le \frac{C}{\omega_0},$$

where C is a constant independent of the mesh size.

Proof. For any $v \in V$, let $v = \sum_{k=1}^{J} v_k$ be a decomposition of v given by (7.3), i.e. $v_k = (\tilde{Q}_k - \tilde{Q}_{k-1})v$. In view of the definition of K_0, and for each $k > 1$, we consider

$$(R_k^{-1} v_k, v_k) \le \frac{1}{\lambda_{\min}(R_k)}(v_k, v_k) \le \frac{\rho(A_k)}{\omega_0}\|(\tilde{Q}_k - \tilde{Q}_{k-1})v\|^2 \le \frac{C}{\omega_0}\|\tilde{Q}_k v\|_A^2,$$

by (7.4) and $\rho(A_k) = O(h_k^{-2})$. Substituting v by $\tilde{v} = (\tilde{Q}_k - \tilde{Q}_{k-1})v$ in (7.4) and rewriting $\tilde{v} = v$, we have

$$\|(\tilde{Q}_k - \tilde{Q}_{k-1})v\| \le C h_k \|(\tilde{Q}_k - \tilde{Q}_{k-1})v\|_A.$$

Hence,

$$(R_k^{-1} v_k, v_k) \le \frac{C}{\omega_0}\|(\tilde{Q}_k - \tilde{Q}_{k-1})v\|_A^2.$$

For $k = 1$, since $R_1 = A_1^{-1}$, we obtain instead

$$(R_1^{-1} v_1, v_1) = ((R_1 A_1)^{-1} v_1, A_1 v_1) = \|v_1\|_A^2 = \|\tilde{Q}_1 v\|_A^2.$$

Combining with (7.3), we have

$$\begin{aligned}
\sum_{k=1}^{J}(R_k^{-1}v_k, v_k) &= \|\tilde{Q}_1 v\|_A^2 + \sum_{k=2}^{J}(R_k^{-1}v_k, v_k) \\
&\le \|\tilde{Q}_1 v\|_A^2 + \frac{C}{\omega_0}\sum_{k=2}^{J}\|(\tilde{Q}_k - \tilde{Q}_{k-1})v\|_A^2 \\
&\le \max\left\{1, \frac{C}{\omega_0}\right\}\left(\sum_{k=1}^{J}\|(\tilde{Q}_k - \tilde{Q}_{k-1})v\|_A^2\right) \\
&= \max\left\{1, \frac{C}{\omega_0}\right\}\|v\|_A^2.
\end{aligned}$$

By the definition of K_0, the estimate follows. □

To summarize, if the stability and the approximation property (7.3) and (7.4) are satisfied, optimal convergence follows. Thus these two properties characterize a good coarse subspace. It is interesting to note that linear finite element subspaces are not compulsory for the V_k, though they are typically used or assumed in the classical analysis of multigrid methods. Moreover, the $\tilde{Q}_k$ in the approximation inequality (7.4) need not necessarily be the L^2 projections Q_k. Linear finite element and L^2 projections are simply two convenient and powerful tools for showing the stability and the approximation property, but are not necessary the only choice.

Optimal convergence, however, need not mean rapid convergence. The reason is that, in general, K_0 will depend on the PDE coefficients. The implicit dependence of the coefficient of the underlying PDE in the convergence rate may cause the multigrid method to converge very slowly, for example, when the coefficients are not smooth. In the following section, we construct coarse subspaces whose basis functions are, in general, different from piecewise linear finite elements but possess the stability and the approximation properties. In addition, the resulting multigrid algorithm is less sensitive to the coefficients than the standard multigrid method. Furthermore, we show that these two concepts lead to an optimal convergence for a one-dimensional multigrid method, and we illustrate how they motivate a two-dimensional multigrid algorithm.

7.3 Energy-minimizing Interpolation

In this section, we introduce the energy minimization approach to constructing the interpolation. The resulting formulation in the one-dimensional case is well-known in the literature [73, 109, 145]. We explain the energy-minimizing interpolation in one dimension first and then the higher dimensions in the next section.

7.3.1 One Dimension

We consider the following model problem:

$$\begin{aligned} -\frac{d}{dx}a(x)\frac{d}{dx}u(x) &= f \qquad \text{in } (0,1) \\ u &= 0 \qquad \text{at } x=0 \text{ and } x=1, \end{aligned} \tag{7.5}$$

where $a(x)$ and $f(x)$ are integrable and $a(x)$ is uniformly positive.

Given a uniform grid with grid size $h = 1/n$, let $x_j^h = jh, j = 0, \ldots, n$. Define the fine grid linear finite element space to be:

$$V^h = \{v^h \in H_0^1(0,1) : v^h \text{ is linear on } [x_j^h, x_{j+1}^h], j = 0, \ldots, n-1\},$$

and denote the set of nodal basis by $\{\phi_j^h\}_{j=1}^n$. The finite element approximation to the solution of (7.5) is the function $u^h \in V^h$ so that,

$$a(u^h, v^h) = (f, v^h) \qquad \forall v^h \in V^h. \tag{7.6}$$

Let $u^h = \sum_{j=1}^n \mu_j \phi_j^h$ and $f = \sum_{j=1}^n \beta_j \phi_j^h$. Then (7.6) is equivalent to a linear system:

$$\mathcal{A}^h \mu = \mathcal{M}^h b,$$

where $\mu = (\mu_1, \ldots, \mu_n)^T$, $b = (\beta_1, \ldots, \beta_n)^T$, $\mathcal{A}^h$ is the stiffness matrix and $\mathcal{M}^h$ is the mass matrix. Define $\tilde{\mathcal{A}}^h$ to be the augmented stiffness matrix that includes also the boundary points. Thus, $\tilde{\mathcal{A}}^h$ is singular with the null space consisting of constant functions, and $\mathcal{A}^h$ is a submatrix of it.

Let $x_i^H = x_{2i}^h, i = 0, \ldots, n/2$ be the set of coarse grid points. Now we define a coarse subspace V^H for multigrid by defining the coarse grid nodal basis functions $\{\phi_i^H\}$. That is,

$$V^H = \text{span}\{\phi_i^H : i = 1, \ldots, m\},$$

and $m = n/2 - 1$. Since $\{\phi_i^H\}$ are nodal basis functions on the coarse grid, $\phi_i^H(x_{2i}^h) = 1$ and $\phi_i^H(x_{2i-2}^h) = \phi_i^H(x_{2i+2}^h) = 0$. We only need to define $\phi_i^H(x_{2i-1}^h)$ and $\phi_i^H(x_{2i+1}^h)$ (see Figure 7.1). For example, if we let them equal $1/2$, the basis functions $\{\phi_i^H\}$ are just linear finite elements, implying that the interpolation from the coarse grid to the fine grid is piecewise linear.

Since $\{\phi_i^H\}$ is a basis of V^H which is a subspace of V^h, there exists a unique matrix $\mathcal{I}_h^H$ of size $n \times m$ such that

$$[\phi_1^H \cdots \phi_m^H] = [\phi_1^h \cdots \phi_n^h]\mathcal{I}_h^H.$$

The matrix $\mathcal{I}_h^H$ is usually known as the prolongation (or interpolation) matrix and its transpose $(\mathcal{I}_h^H)^T = \mathcal{I}_H^h$ as the restriction matrix in the multigrid context. Hence the set of coarse grid basis functions defines an interpolation and vice versa. In

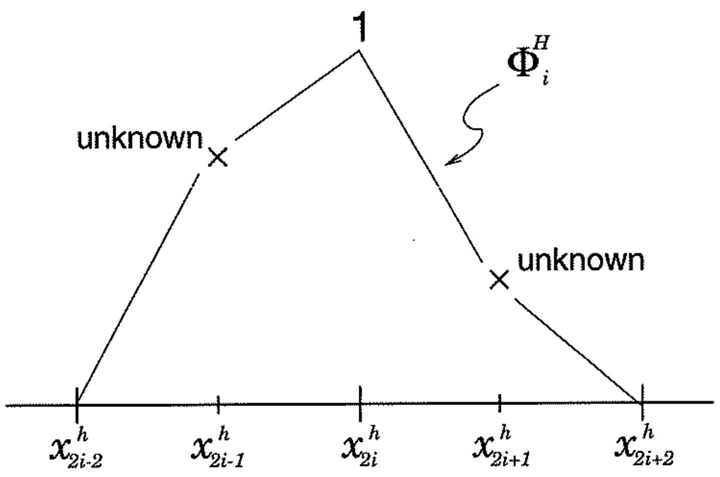


Figure 7.1: 1D coarse grid basis function ϕ_i^H on its support $[x^h_{2i-2}, x^h_{2i+2}]$.

the following, instead of deriving an interpolation method directly, we construct an energy-minimizing basis.

Formulation. As noted above, the interpolation is uniquely determined if the coarse grid basis functions $\{\phi_i^H\}$ are known. We can define $\phi_i^H(x)$ by solving the following local PDE problem in $[x^H_{i-1}, x^H_i] = [x^h_{2i-2}, x^h_{2i}]$:

$$\begin{aligned} &-\frac{d}{dx}a(x)\frac{d}{dx}\phi_i^H = 0 \qquad\qquad \text{in } [x^h_{2i-2}, x^h_{2i}], \\ &\phi_i^H(x^h_{2i-2}) = 0, \;\; \phi_i^H(x^h_{2i}) = 1. \end{aligned} \tag{7.7}$$

We observe that the PDE formulation of the basis functions has a "physical" meaning attached to it. Specifically, it looks for basis functions which have small energy. It is best illustrated by the following result.

Lemma 7.2 *An equivalent formulation of (7.7) is*

$$\begin{aligned} &\min \;\; a(\phi_i^H, \phi_i^H) \qquad\qquad \textit{in } [x^h_{2i-2}, x^h_{2i}], \\ \textit{subject to} \quad &\phi_i^H(x^h_{2i-2}) = 0, \;\; \phi_i^H(x^h_{2i}) = 1. \end{aligned} \tag{7.8}$$

Thus, the solution of the local PDE minimizes the energy of the coarse grid basis functions. This observation turns out to be very convenient to extend the idea to higher dimensions.

The solution of $\phi_i^H(x)$ on $[x^h_{2i-2}, x^h_{2i}]$ defines $\phi_i^H(x^h_{2i-1})$ implicitly. We can do the same for $\phi_i^H(x^h_{2i+1})$ in $[x^h_{2i}, x^h_{2i+2}]$. The local PDE formulation calculates the "harmonic" function ϕ_i^H which minimizes the energy on its support. If $a(x) \equiv 1$, ϕ_i^H is a linear function and we get back linear interpolation, i.e. $\phi_i^H(x^h_{2i-1}) =$

$\phi_i^H(x_{2i+1}^h) = 1/2$. In fact, in this case, ϕ_i^H is harmonic in the usual sense and it has minimum energy. In general, instead of 1/2, we have

$$\phi_i^H(x_{2i-1}^h) = -\frac{a(\phi_{2i-1}^h, \phi_{2i}^h)}{a(\phi_{2i-1}^h, \phi_{2i-1}^h)} = -\frac{\mathcal{A}_{2i-1,2i}^h}{\mathcal{A}_{2i-1,2i-1}^h}, \tag{7.9}$$

where $(\mathcal{A}_{ij}^h)$ is the stiffness matrix. Since our interpolation depends on the matrix $\mathcal{A}^h$, sometimes it is called a matrix-dependent interpolation in the algebraic multigrid context. The resulting interpolation was also described in [73, 109, 145] but from a different point of view. Ours is novel in the sense that we interpret it from the energy-minimization principle, which provides a clue to developing similar interpolation operators in higher dimensions.

The approximation property (7.4) is closely related to preserving constant functions. In fact, the coarse space V^H constructed in this way automatically contains constant functions on the fine grid.

Lemma 7.3

$$\sum_{i=1}^{m} \phi_i^H(x) = 1.$$

Proof. Let $\psi^H(x) = \sum_{i=1}^m \phi_i^H(x)$. By (7.7), for $i = 1, \ldots, m$, ψ^H satisfies the following:

$$\begin{aligned} -\frac{d}{dx}a(x)\frac{d}{dx}\psi^H &= 0 \qquad\qquad \text{in } [x_{2i-2}^h, x_{2i}^h], \\ \psi^H(x_{2i-2}^h) = 1, &\quad \psi^H(x_{2i}^h) = 1. \end{aligned}$$

By uniqueness, $\psi^H \equiv 1$ on $[x_{2i-1}^h, x_{2i}^h]$, and hence the result follows. $\square$

Thus, the interpolation derived from the energy-minimizing coarse grid basis functions preserves constants.

Remarks: (1) If $a(x)$ is piecewise constant, this interpolation preserves the continuity of the flux, $a(x)\nabla u$, at the discontinuities [73]. (2) If red-black Gauss-Seidel is used as smoother, the resulting multigrid method coincides with the cyclic reduction method in the numerical linear algebra context.

7.3.2 Higher Dimensions

The construction of the energy-minimizing interpolation described in this section is valid for two and three dimensions. However, to facilitate understanding, we focus on the standard structured grid on the square domain Ω: $[0,1]\times[0,1]$ in two dimensions. The model problem is

$$\begin{aligned} -\nabla\cdot a(x,y)\nabla u(x,y) &= f(x,y), && \text{in } \Omega \\ u &= 0 && \text{on } \partial\Omega, \end{aligned} \tag{7.10}$$

with the same assumptions on $a(x,y)$ and $f(x,y)$ as before. Again, we use finite element method to discretize (7.10).

Formulation. The extension to higher dimensions of the local PDE approach is difficult because there is no natural analog between one dimension and higher dimensions. For instance, in one dimension, the coarse grid points form the boundaries of the local subdomains so that well-posed PDEs can be easily defined. In higher dimensions, however, the boundaries consist of both coarse and noncoarse grid points and hence local boundary value problems apparently do not exist. Nevertheless, several possibilities for setting up local PDEs are discussed in the literature, for instance, the stencil or the so-called black-box multigrid approach [1, 46, 47, 72, 73, 85, 86, 145, 151], the Schur complement approach [58, 87, 110] and the algebraic multigrid approach [29, 35, 134], each of which mimics the one-dimensional case in some way.

Our approach is based on the observation (7.8). The coarse grid basis functions $\{\phi_i^H\}$ should possess the least amount of energy while preserving constant functions. The precise mathematical formulation is explained in the following.

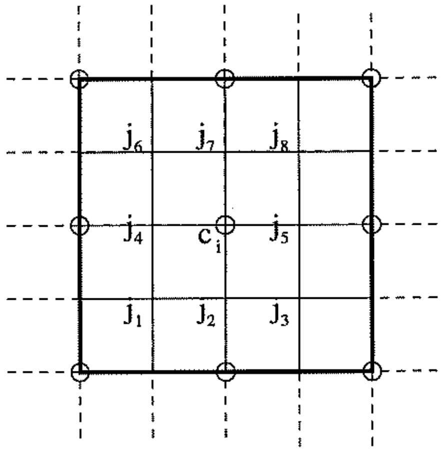


Figure 7.2: 2D coarse grid basis function ϕ_i^H on its support. ϕ_i^H is a linear combination of fine grid basis functions ϕ_j^h, $j = j_1,\ldots,j_8$ and c_i.

Suppose a maximal independent set of the vertices of the finer grid is selected as coarse grid points and denote the index set by $M=\{c_1,\ldots,c_m\}$, $m=(n/2+1)^2$. Write the coarse grid nodal basis function ϕ_i^H at node x_{c_i} as a linear combination of the fine grid ones:

$$\phi_i^H = \sum_{j \in \{l : \tilde{A}^h_{c_i,l} \neq 0\} \setminus M} \varphi_j^i \phi_j^h + \phi_{c_i}^h. \tag{7.11}$$

Thus, ϕ_i^H is a local combination of the fine grid basis functions whose corresponding node is adjacent to node x_{c_i} but not itself a coarse grid point. Figure 7.2 shows the support of ϕ_i^H in two dimensions. The indices j in the sum on the right-hand side of (7.11) correspond to $j_1, \ldots, j_8$. Since ϕ_i^H is a nodal basis function, the coefficient of $\phi_{c_i}^h$ is equal to 1. We define the interpolation by solving a constrained minimization problem for $\{\varphi_j^i\}$:

$$\min \ \frac{1}{2}\sum_{i=1}^{m} \|\phi_i^H\|_A^2 \qquad \text{subject to} \qquad \sum_{i=1}^{m} \phi_i^H(x) = 1 \text{ in } \bar{\Omega}. \tag{7.12}$$

Notice that the minimization problem is solved up to and including the boundary of Ω. Usually, the grid points on the boundary with Dirichlet boundary condition are treated separately, and no coarse grid point is placed there. However, in our formulation, we compute all ϕ_i^H including the ones at the boundary, but only those not on the boundary with Dirichlet condition are used in the interpolation.

In one dimension, this minimization reduces to (7.8).

Lemma 7.4 *An equivalent formulation of (7.7) and (7.8) is the global minimization:*

$$\min \ \frac{1}{2}\sum_{i=1}^{m} \|\phi_i^H\|_A^2 \qquad \textit{subject to} \qquad \sum_{i=1}^{m} \phi_i^H(x) = 1 \ \textit{on} \ [0,1].$$

Thus, we see a way to naturally generalize the approach for generating a robust interpolation from one dimension to multiple dimensions.

Remarks: (1) The values of the basis functions are defined implicitly by the solution of (7.12) and are not known explicitly in general. However, for the Laplacian, we recover exactly the bilinear interpolation on tensor-product grids, which is known to lead to optimal multigrid convergence for Poisson equations.

Lemma 7.5 *The solution of (7.12) gives the bilinear interpolation if $a(x) \equiv 1$.*

Proof. Let $\Phi_0 = [\varphi_0^1; \cdots; \varphi_0^m]$ be the vector corresponding to the bilinear interpolation. Thus, the $n \times 1$ sparse vector φ_0^i, corresponding to the coefficients of ϕ_j^h in the expansion of ϕ_i^H, has nonzeros 1/4, 1/2, and 1 only. We verify by direct substitution that Φ_0 satisfies the Euler-Lagrange equation (7.15) with an appropriately defined Λ_0.

Since φ_0^i is sparse, we may consider the nonzeros of φ_0^i only when computing the product $\tilde{A}_i^h \varphi_0^i$. Define

$$\begin{aligned}
\Omega_1 &= \{x_k^h : x_k^h \text{ is an interior noncoarse grid point which does not connect to} \\
&\qquad \text{any coarse grid points on the mesh.}\} \\
\Omega_2 &= \{x_k^h : x_k^h \text{ is an interior noncoarse grid point which connects to exactly 2} \\
&\qquad \text{coarse grid points on the mesh.}\} \\
\Omega_3 &= \{x_k^h : x_k^h \text{ is a noncoarse grid boundary point.}\}
\end{aligned}$$

By the definition of $\tilde{A}_i^h$, after some calculation, we can verify that

$$(\tilde{A}_i^h \varphi_0^i)_k = \begin{cases} 0 & \text{if } x_k^h \in \Omega_1 \\ 3/2 & \text{if } x_k^h \in \Omega_2 \\ 3/4 & \text{if } x_k^h \in \Omega_3. \end{cases} \tag{7.13}$$

Here $\tilde{A}_i^h$ is the nine point stencil

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}.$$

The values at the coarse grid points are not considered because the solution Φ must have a value of 1 there.

Let Λ_0 be the vector of Lagrange multipliers defined by

$$(\Lambda_0)_k = \begin{cases} 0 & \text{if } x_k^h \in \Omega_1 \\ -3/2 & \text{if } x_k^h \in \Omega_2 \\ -3/4 & \text{if } x_k^h \in \Omega_3. \end{cases}$$

First, from (7.13), we see that the values of $\tilde{A}_i^h \varphi_0^i$ depend not on the location of φ_0^i but on the location corresponding to the component k only. Second, $\mathcal{B}$ is a column of restriction matrices that have either 1 or 0 on the diagonal. It is not hard to see that $\mathcal{Q}\Phi_0 + \mathcal{B}\Lambda_0 = 1$. Hence, the result follows. □

We also remark that if triangular grids are used, the linear interpolation is almost recovered; numerical experiments show that the interpolation values are close to 1/2.

(2) Like algebraic multigrid, the construction of the interpolation operator is purely algebraic. In other words, geometry and in particular the grid information are not needed. Besides, the formulation of the interpolation is still valid if the coarse grid points do not form an independent set. Independent sets are certainly beneficial to efficiency but are not necessary. In some situations, we may want to remove this requirement, for example, when semi-coarsening is used.

(3) Finally, we remark that we may generalize the formulation further by putting in positive weights θ_i in front of $\|\phi_i^H\|_A^2$. Similarly, we have the following equivalence.

Lemma 7.6 *An equivalent formulation of (7.7) and (7.8) is the global weighted minimization*

$$\min \ \frac{1}{2}\sum_{i=1}^{m} \theta_i \|\phi_i^H\|_A^2 \qquad \textit{subject to} \qquad \sum_{i=1}^{m} \phi_i^H(x) = 1 \ \textit{on} \ [0,1],$$

for any sets of positive θ_i.

In our experience, special scalings, for instance, $\theta_i = 1/\tilde{A}^h_{c_i,c_i}$, may improve the performance for problems such as discontinuous coefficient PDEs where the discontinuities do not align with any coarser grids. However, an optimal choice of θ_i has not yet been fully analyzed, and hence we shall not discuss this generalization further.

7.3.3 Solution of the Minimization Problem

We describe a solution procedure for the minimization problem (7.12) below. For each i, write $\phi_i^H = \sum_{j=1}^n \varphi_j^i \phi_j^h$ and $\varphi^i = (\varphi_1^i, \ldots, \varphi_n^i)^T$. By (7.11), ϕ^i is a sparse vector. For example, in two dimensions, φ^i has at most 9 nonzeros. For structured triangular grids, φ^i has at most 7 nonzeros. Let $\Phi = [\varphi^1; \cdots; \varphi^m]$ be an $mn \times 1$ vector obtained by appending all the φ's. Note that $\|\phi_i^H\|_A^2 = \|\sum_{j=1}^n \varphi_j^i \phi_j^h\|_A^2 = (\varphi^i)^T \tilde{A}^h \varphi^i$. (Recall that $\tilde{A}^h$ is the augmented stiffness matrix on the fine grid without incorporating any Dirichlet boundary condition.) Thus, (7.12) can be written as the following equivalent discrete linear constrained quadratic minimization problem:

$$\min \ \frac{1}{2}\Phi^T \mathcal{Q} \Phi \quad \text{s.t.} \quad \mathcal{B}^T \Phi = \mathbf{1}. \tag{7.14}$$

The symbol $\mathbf{1}$ denotes a vector of all 1's. The $mn \times mn$ SPD matrix $\mathcal{Q}$ is block diagonal with each block equals $\tilde{\mathcal{A}}_i^h$ which is defined as

$$(\tilde{\mathcal{A}}_i^h)_{kl} = \begin{cases} \tilde{A}^h_{kl} & \text{if } \varphi_k^i \neq 0 \text{ and } \varphi_l^i \neq 0 \\ \delta_{kl} & \text{otherwise.} \end{cases}$$

The $n \times mn$ rectangular matrix $\mathcal{B}^T = [\mathcal{J}_1^T \cdots \mathcal{J}_m^T]$, where $\mathcal{J}_i = \mathcal{J}_i^T$ is a matrix corresponding to the restriction operator that maps v to v_i such that $(v)_k = (v_i)_k$ on $\text{supp}(\phi_i^H)$ and $(v_i)_k = 0$ otherwise. More precisely,

$$(\mathcal{J}_i)_{kl} = \begin{cases} 1 & \text{if } k = l \text{ and } \varphi_k^i \neq 0 \\ 0 & \text{otherwise.} \end{cases}$$

It is clear that $\mathcal{J}_i^T \varphi^i = \varphi^i$ and hence $\mathcal{B}^T \Phi = \sum_{i=1}^m \mathcal{J}_i^T \varphi^i = \sum_{i=1}^m \varphi^i = \mathbf{1}$. We solve the discrete linearly constrained minimization problem (7.14) by the Lagrange multiplier formulation, which is equivalent to:

$$\begin{bmatrix} \mathcal{Q} & \mathcal{B} \\ \mathcal{B}^T & 0 \end{bmatrix} \begin{bmatrix} \Phi \\ \Lambda \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix}, \tag{7.15}$$

where Λ is an $n \times 1$ vector of Lagrange multipliers. If Λ is known, Φ can be computed by solving:

$$\mathcal{Q}\Phi = -\mathcal{B}\Lambda. \tag{7.16}$$

Since $\mathcal{Q}$ is block diagonal and inverting each block corresponds to solving a matrix of at most 9×9 in size, it is trivial to compute Φ. Thus the entire minimization procedure is reduced to solving for the Lagrange multipliers Λ via

$$(\mathcal{B}^T \mathcal{Q}^{-1} \mathcal{B})\Lambda = -\mathbf{1}. \tag{7.17}$$

Note that $\mathcal{B}$ and $\mathcal{Q}^{-1}$ are sparse matrices. We can solve the linear system by conjugate gradient (CG).

The solution process of (7.17) could be costly. Depending on the conditional number of $\mathcal{Q}^{-1}$, the CG iteration may converge slowly. We shall discuss how to speed up the process. First, we need not compute $(\mathcal{B}^T \mathcal{Q}^{-1} \mathcal{B})^{-1}\mathbf{1}$ exactly since we are merely computing the interpolation to be used in the multigrid method. In fact, the numerical results in Section 7.5 indicate that Λ is usually accurate enough when the relative residual of (7.17) is less then 10^{-2}.

Besides, we have a readily obtainable initial guess for Λ. Consider equation (7.16). Multiplying both sides by $\mathcal{B}^T$, we have

$$\Lambda = -(\mathcal{B}^T \mathcal{B})^{-1} \mathcal{Q}\Phi.$$

Since $\mathcal{B}$ is the restriction operator, it is not hard to see that $\mathcal{B}^T\mathcal{B}$ is a diagonal matrix. So this gives an easy way to compute an initial guess for Λ from Φ. Since the interpolation weights are between 0 and 1, the solution Φ usually is not very far from the linear interpolation. It may be advantageous to use the linear interpolation as an initial guess for Φ, which in turn provides an initial guess for Λ.

It is interesting to note that $\tilde{\mathcal{A}}^h$ is a free and natural preconditioner for $\mathcal{B}\mathcal{Q}^{-1}\mathcal{B}$. By the definition by $\mathcal{B}$ and $\mathcal{Q}$, rewrite the product $\mathcal{B}^T \mathcal{Q}^{-1} \mathcal{B}$ as a sum of matrices:

$$\mathcal{B}^T \mathcal{Q}^{-1} \mathcal{B} = \sum_{i=1}^m \mathcal{J}_i^T (\tilde{\mathcal{A}}_i^h)^{-1} \mathcal{J}_i = \sum_{i=1}^m \mathcal{R}_i^T (\mathcal{R}_i \tilde{\mathcal{A}}^h \mathcal{R}_i^T)^{-1} \mathcal{R}_i,$$

where $\mathcal{R}_i$ is the submatrix of the nonzero rows of $\mathcal{J}_i$ and it is sometimes known as the restriction matrix in the domain decomposition context. Now it is clear that $\mathcal{B}\mathcal{Q}^{-1}\mathcal{B}$ is an overlapping additive Schwarz preconditioner of $\tilde{\mathcal{A}}^h$. Unfortunately,

$\tilde{A}^h$ is singular in our case. A simple remedy is to use $\tilde{A}^h + \eta\mathcal{I}$ instead as the preconditioner.

Because of the potential high cost of computing Φ, the energy-minimizing interpolation is aimed at problems that linear interpolation does not work well. Quite often, we may need to solve the same system many times, for instance, time dependent problems, the expensive setup cost can be compensated by the rapid convergence of each multigrid solve.

Remarks: (1) The solution of Φ from (7.16) and (7.17) is equivalent to Newton's method for solving (7.14). (2) We may also solve (7.14) by projected steepest descent method [95].

7.3.4 Connections to Other Approaches

As noted above, the entire procedure of constructing the interpolation is algebraic, and so it can be considered as a type of algebraic multigrid. In fact, it is related to the one derived by Vanek, Mandel and Brezina [134]. In their approach, groups of fine grid elements are agglomerated to form larger elements, or macroelements. In each agglomerated region (which can be thought of a subdomain in the domain decomposition context), a value of 1 is assigned to each node as an initial guess of the coarse grid basis. Because of the high energy of the piecewise constant basis functions, they are *smoothed* by a few steps of Jacobi iteration. Our energy-minimizing coarse grid basis can also be thought of being formed by agglomerating nearby fine grid elements, but the agglomeration only occurs at elements whose node is a coarse grid node. Also there are overlaps among agglomerated regions while there is none in the approach of Vanek et al. Moreover, the support of their basis functions will increase when the Jacobi "smoothing" steps are applied to the basis functions. In our approach, the supports are fixed and the energy is minimized by solving the minimization problem (7.12).

Because of the agglomeration view of the construction, our approach is also related to the one derived by Chan et al. [29, 35]. They explicitly form the macroelements by agglomeration using standard graph theoretical techniques. Then they have several way of defining the coarse grid basis functions. One way is the following. The noncoarse grid points on the edge of a macroelement are assigned a value using the graph distance, and those noncoarse grid points in the interior are obtained by solving a local homogeneous PDE. Our approach does not prescribe a value on the edges of the macroelements first and then solve for the interior points. Rather, we take all the unknowns together and solve for all the values simultaneously by solving the minimization problem.

7.4 Convergence Analysis

Much of the classical multigrid convergence analysis cannot be applied directly to the proposed multigrid algorithm because the coarse spaces defined by the basis functions are not standard finite element spaces which is usually assumed in the literature. The one-dimensional analysis is complete and will be presented first. It is then followed by a two level analysis of the two dimensions.

7.4.1 One Dimension

First, we show the stability property (7.3). The proof is based on the observation that the coarse grid basis functions contain a hierarchy of A-orthogonal basis functions; in other words, they are orthogonal in the A-inner product. Recall that the coarse grid points are chosen to be the even fine grid points (cf. Section 7.3.1), i.e. $x_i^{k-1} = x_{2i}^k$.

Lemma 7.7 *For any $l < k, i = 1, \ldots, n_l, j = 1, \ldots, n_k/2$, we have*

$$a(\phi_i^l, \phi_{2j-1}^k) = 0. \qquad \textit{(7.18)}$$

Proof. Let k be fixed. We first prove the case $l = k - 1$ using a technique suggested by Xu [147]. In this case, (7.18) is just the direct consequence of the fact that the equivalent variational formulation of (7.7) implies that $a(\phi_i^{k-1}, \phi_{2i-1}^k) = 0$, and the support of ϕ_i^{k-1} is only on $[x_{2i-2}^k, x_{2i+2}^k]$.

Now suppose it is true for $l = \bar{k}$. By definition,

$$\phi_i^{\bar{k}-1} = \sum_{j=2i-1}^{2i+1} \alpha_j \phi_j^{\bar{k}},$$

where $\alpha_{2i} = 1$, $\alpha_{2i-1} = \phi_i^{\bar{k}-1}(x_{2i-1}^{\bar{k}})$ and $\alpha_{2i+1} = \phi_i^{\bar{k}-1}(x_{2i+1}^{\bar{k}})$ are given by (7.9). Thus

$$a(\phi_i^{\bar{k}-1}, \phi_{2j-1}^k) = \alpha_{2i-1} a(\phi_{2i-1}^{\bar{k}}, \phi_{2j-1}^k) + a(\phi_{2i}^{\bar{k}}, \phi_{2j-1}^k) + \alpha_{2i+1} a(\phi_{2i+1}^{\bar{k}}, \phi_{2j-1}^k) = 0,$$

since all the terms vanish by assumption and the result follows from induction. □

Lemma 7.7 implies that the interpolation algorithm generates implicitly a set of A-orthogonal hierarchical basis functions. The orthogonality property immediately implies the stability of the nested subspaces.

Theorem 7.2 *For any $v \in V$, there is a nontrivial decomposition $v = \sum_{k=1}^{J} v_k$ with $v_k \in V_k$ such that*

$$\sum_{k=1}^{J} (v_k, v_k)_A = (v, v)_A. \tag{7.19}$$

Proof. For any $v \in V$, Lemma 7.7 implies that there exists an orthogonal hierarchical decomposition of v constructed as follows. We first define v_1 to be the nodal value interpolant of v at the coarsest level V_1. Then we subtract v_1 from v to obtain w_2. Because of the nodal interpolation, the values of w_2 at $x_i^1, i = 1, \ldots, n_1$, are zero. We proceed similarly by defining v_2 to be the nodal value interpolant of w_2 and so on. Formally, we have the following:

$$v_1 = \sum_{i=1}^{n_1} v(x_i^1)\phi_i^1 \qquad \text{and} \qquad v_k = \sum_{i=1}^{n_k} w_k(x_i^k)\phi_i^k, \qquad k = 2, \ldots, J, \tag{7.20}$$

where $w_k = v - \sum_{i=1}^{k-1} v_i$. Our decomposition implies that $v_k(x_j^k) = w_k(x_j^k) = 0, j$ even. Therefore, by Lemma 7.7, the v_k's are A-orthogonal since if $l < k$,

$$\begin{aligned} a(v_l, v_k) &= a(\sum_{i=1}^{n_l} w_l(x_i^l)\phi_i^l, \sum_{j=1}^{n_k} w_k(x_j^k)\phi_j^k) \\ &= \sum_{i=1}^{n_l} \sum_{j=1}^{n_k/2} w_l(x_i^l) w_k(x_{2j-1}^k) a(\phi_i^l, \phi_{2j-1}^k) \\ &= 0. \end{aligned}$$

The equality (7.19) follows immediately from the orthogonality of v_k's. □

Corollary 7.1 *Let $W_1 = V_1$ and $W_k = V_k \ominus V_{k-1}, k = 2, \ldots, J$, in the A-inner product. Then V can be expressed as a direct sum of W_k's:*

$$V = W_1 \oplus W_2 \oplus \cdots \oplus W_J.$$

Corollary 7.1 induces a projection operator $\tilde{Q}_k : V \to V_k$ defined by

$$\tilde{Q}_k v = v_1 + v_2 + \cdots + v_k, \tag{7.21}$$

where $v = v_1 + \cdots + v_J, v_k \in W_k$, is the unique representation of v defined in (7.20). This operator $\tilde{Q}_i$ will be used to prove the approximation property (7.4). Here we do not use the L^2 projection Q_k because $\tilde{Q}_k$ is a more natural and convenient choice in the one-dimensional case. In view of Theorem 7.2 and Corollary 7.1, the stability property (7.3) is satisfied.

In the literature, the approximation property (7.4) is typically proved by making use of the fact that the interpolation preserves constant functions. In the two level case, we have shown in Lemma 7.3 that constant functions are indeed preserved by the coarse grid basis functions. Using the same proof technique, we can easily show that it is also true for the multilevel case.

Lemma 7.8 *For any $k = 1, \ldots, J$,*

$$\sum_{i=1}^{n_k} \phi_i^k(x) = 1.$$

With this result, we can now prove the approximation property.

Theorem 7.3 *For any $v \in V$ and any $k = 2, \ldots, J$,*

$$\|(\tilde{Q}_k - \tilde{Q}_{k-1})v\| \le C h_k \|\tilde{Q}_k v\|_A. \tag{7.22}$$

Proof. We compute the quantities on both sides explicitly to see how preserving constant functions comes into play. Since $\tilde{Q}_k$ is a projection, we can always change v to $\tilde{Q}_k v$ in the left-hand side of (7.22). Without loss of generality, we assume $v \in V_k$, i.e. $v = \sum_{i=1}^{n_k} \nu_i \phi_i^k$. Thus, we need only prove

$$\|v - \tilde{Q}_{k-1}v\| \le C h_k \|v\|_A.$$

By the definition of $\tilde{Q}_{k-1}$ in (7.21),

$$\tilde{Q}_{k-1}v = \sum_{i=1}^{n_{k-1}} \nu_{2i}\phi_i^{k-1}.$$

Let $w = v - \tilde{Q}_{k-1}v = \sum_{i=1}^{n_k} \omega_i \phi_i^k$. Then we can verify that

$$\begin{aligned} \omega_{2i} &= 0, \qquad i = 1, \ldots, n_k/2, \\ \omega_{2i-1} &= \nu_{2i-1} - (\alpha\nu_{2i-2} + \beta\nu_{2i}), \end{aligned}$$

where $\alpha = \phi_{i-1}^{k-1}(x_{2i-1}^k)$ and $\beta = \phi_i^{k-1}(x_{2i-1}^k)$. Because the coarse grid basis functions preserve constant, we have $\alpha + \beta = 1$ and hence

$$\begin{aligned} \omega_{2i-1} &= (\alpha+\beta)\nu_{2i-1} - (\alpha\nu_{2i-2} + \beta\nu_{2i}), \\ &= \alpha(\nu_{2i-1} - \nu_{2i-2}) - \beta(\nu_{2i} - \nu_{2i-1}). \end{aligned}$$

Now we estimate the L^2 norm of w on $[x_{2i-2}^k, x_{2i}^k]$:

$$\begin{aligned} \int_{x_{2i-2}^k}^{x_{2i}^k} w^2 dx &= \int_{x_{2i-2}^k}^{x_{2i}^k} (\omega_{2i-1}\phi_{2i-1}^k)^2 dx \\ &= \omega_{2i-1}^2 \int_{x_{2i-2}^k}^{x_{2i}^k} (\phi_{2i-1}^k)^2 dx \\ &= \omega_{2i-1}^2 \mathcal{M}_{2i-1,2i-1}^k, \end{aligned}$$

where $\mathcal{M}^k_{2i-1,2i-1}$ is the $(2i-1, 2i-1)$ entry of $\mathcal{M}_k$ which is the mass matrix with respect to $\{\phi_i^k\}_{i=1}^{n_k}$. Using the formula for ω_{2i-1} and the elementary inequality: $(\alpha A - \beta B)^2 \leq \alpha A^2 + \beta B^2$ for $\alpha + \beta = 1$, we have that

$$\begin{aligned} \int_{x^k_{2i-2}}^{x^k_{2i}} w^2 dx &= \mathcal{M}^k_{2i-1,2i-1}[\alpha(\nu_{2i-1} - \nu_{2i-2}) - \beta(\nu_{2i} - \nu_{2i-1})]^2 \\ &\leq \mathcal{M}^k_{2i-1,2i-1}[\alpha(\nu_{2i-1} - \nu_{2i-2})^2 + \beta(\nu_{2i} - \nu_{2i-1})^2]. \end{aligned} \tag{7.23}$$

On the other hand, the A-norm of v is given by,

$$\begin{aligned} &\int_{x^k_{2i-2}}^{x^k_{2i}} a(x)(v')^2 dx \\ = &\int_{x^k_{2i-2}}^{x^k_{2i-1}} a(x)(\nu_{2i-2}{\phi^k_{2i-2}}' + \nu_{2i-1}{\phi^k_{2i-1}}')^2 dx + \int_{x^k_{2i-1}}^{x^k_{2i}} a(x)(\nu_{2i-1}{\phi^k_{2i-1}}' + \nu_{2i}{\phi^k_{2i}}')^2 dx \\ = &-(\nu_{2i-1} - \nu_{2i-2})^2 \int_{x^k_{2i-2}}^{x^k_{2i-1}} a(x){\phi^k_{2i-2}}'{\phi^k_{2i-1}}' dx \\ &-(\nu_{2i} - \nu_{2i-1})^2 \int_{x^k_{2i-1}}^{x^k_{2i}} a(x){\phi^k_{2i-1}}'{\phi^k_{2i}}' dx, \end{aligned}$$

since $\phi^k_{2i-2}(x) + \phi^k_{2i-1}(x) = 1$ on $[x^k_{2i-2}, x^k_{2i-1}]$, which implies ${\phi^k_{2i-2}}'(x) + {\phi^k_{2i-1}}'(x) = 0$. Similar argument also holds for the second integral. Together with the formula of α and β in (7.9), we have

$$\int_{x^k_{2i-2}}^{x^k_{2i}} a(x)(v')^2 dx = [\alpha(\nu_{2i-1} - \nu_{2i-2})^2 + \beta(\nu_{2i} - \nu_{2i-1})^2]\mathcal{A}^k_{2i-1,2i-1},$$

where $\mathcal{A}^k_{2i-1,2i-1}$ is the $(2i-1, 2i-1)$ entry of $\mathcal{A}_k$ which is the stiffness matrix at level k with respect to the basis $\{\phi_i^k\}_{i=1}^{n_k}$. Combining with (7.23), we have

$$\int_{x^k_{2i-2}}^{x^k_{2i}} w^2 dx \leq \frac{\mathcal{M}^k_{2i-1,2i-1}}{\mathcal{A}^k_{2i-1,2i-1}} \int_{x^k_{2i-2}}^{x^k_{2i}} a(x)(v')^2 dx \tag{7.24}$$

$$\tag{7.25}$$

It is easy to show that

$$\mathcal{M}^k_{2i-1,2i-1} = \int_{x^k_{2i-2}}^{x^k_{2i}} (\phi^k_{2i-1})^2 dx \leq O(h_k),$$

and

$$\mathcal{A}^k_{2i-1,2i-1} = \int_{x^k_{2i-2}}^{x^k_{2i}} a(x)({\phi^k_{2i-1}}')^2 dx \geq O(h_k^{-1}).$$

Thus, (7.24) becomes

$$\int_{x^k_{2i-2}}^{x^k_{2i}} w^2 dx \leq C h_k^2 \int_{x^k_{2i-2}}^{x^k_{2i}} a(x)(v')^2 dx,$$

where C is independent of h_k. Summing over i, we obtain the approximation property. □

Corollary 7.2 *For any $v \in V$ and any $k = 2, \ldots, J$,*

$$\|(\tilde{Q}_k - \tilde{Q}_{k-1})v\| \le Ch_k \|(\tilde{Q}_k - \tilde{Q}_{k-1})v\|_A.$$

Proof. Let $\tilde{v} = (\tilde{Q}_k - \tilde{Q}_{k-1})v$ and apply Theorem 7.3. □

Hence, by Lemma 7.1, K_0 is bounded by a constant independent of the mesh size h, although the constant may depend on the coefficient $a(x)$. The coefficient dependence comes from the bound given by the approximation property (cf. Theorem 7.3). It turns out that we can eliminate the coefficient dependence by estimating K_0 directly from its definition (7.1) if the damped Jacobi or Gauss-Seidel smoothings are used.

Theorem 7.4 *Let R_k^{DJ} and R_k^{GS} be the approximate inverses of A_k given by the damped Jacobi method and the Gauss-Seidel method, respectively:*

$$\begin{aligned} (R_k^{DJ} v_k, v_k) &= \frac{1}{\rho((\mathcal{D}^k)^{-1}\mathcal{A}^k)} (\nu^k)^T (\mathcal{D}_k)^{-1} \nu^k, \\ (R_k^{GS} v_k, v_k) &= (\nu^k)^T (\mathcal{D}^k - \mathcal{L}^k)^{-1} \nu^k, \end{aligned}$$

where $\mathcal{A}^k = \mathcal{D}^k - \mathcal{L}^k - (\mathcal{L}^k)^T, \mathcal{D}^k =$ diagonal of $\mathcal{A}^k$ and $v_k = \sum_{j=1}^{n_k} \nu_j^k \phi_j^k$. Then

$$K_0 \le 3,$$

for the damped Jacobi smoothing, and

$$K_0 = 1,$$

for the Gauss-Seidel smoothing.

Proof. This proof is a modification of the proof of Lemma 7.1. For any $v \in V$, let $v = \sum_{k=1}^J v_k$ be a decomposition of v given by (7.3), i.e. $v_k = (\tilde{Q}_k - \tilde{Q}_{k-1})v$. We first estimate K_0 for the damped Jacobi smoothing and then for Gauss-Seidel smoothing. In view of the definition of K_0, for each $k > 1$, we consider

$$((R_k^{DJ})^{-1} v_k, v_k) = \rho((\mathcal{D}^k)^{-1}\mathcal{A}^k)\ (\nu^k)^T \mathcal{D}^k \nu^k.$$

From the calculations in the proof of Theorem 7.3, we showed that $\nu_j^k = 0$, j even. Thus

$$(\nu^k)^T \mathcal{D}^k \nu^k = (\nu^k)^T \mathcal{A}^k \nu^k = \|v_k\|_A^2.$$

The estimate of $\rho((\mathcal{D}^k)^{-1}\mathcal{A}^k) = \rho((\mathcal{D}^k)^{-1/2}\mathcal{A}^k(\mathcal{D}^k)^{-1/2})$ is purely algebraic. Note that the product $(\mathcal{D}^k)^{1/2}\mathcal{A}^k(\mathcal{D}^k)^{1/2}$ is simply the matrix obtained by the diagonal scaling of $\mathcal{A}^k$. Hence, it is still tridiagonal and its diagonal elements are all 1's. The element of the $(i, i+1)$th entry is given by $\mathcal{A}^k_{i,i+1}/\sqrt{\mathcal{A}^k_{i,i}\mathcal{A}^k_{i+1,i+1}}$. Since $\mathcal{A}^k$ is SPD, it is easy to show that the $(i, i+1)$th entry is bounded by 1 in size. By the Gershgorin Circle Theorem, $\rho((\mathcal{D}^k)^{-1}\mathcal{A}^k) \le 3$. Hence

$$\begin{aligned}
\sum_{k=1}^{J}((R_k^{DJ})^{-1}v_k, v_k) &= \|v_1\|_A^2 + \sum_{k=2}^{J}((R_k^{DJ})^{-1}v_k, v_k) \\
&\le \|v_1\|_A^2 + 3\sum_{k=2}^{J}\|v_k\|_A^2 \\
&\le 3\|v\|_A^2.
\end{aligned}$$

By the definition of K_0, the estimate follows.

Similarly for the Gauss-Seidel smoothing, we consider

$$\begin{aligned}
((R_k^{GS})^{-1}v_k, v_k) &= (\nu^k)^T(\mathcal{D}^k - \mathcal{L}^k)\nu^k \\
&= \frac{1}{2}(\nu^k)^T\mathcal{D}^k\nu^k + \frac{1}{2}(\nu^k)^T\mathcal{A}^k\nu^k \\
&= (\nu^k)^T\mathcal{A}^k\nu^k \\
&= \|v_k\|_A^2.
\end{aligned}$$

Thus $K_0 = 1$ since

$$\sum_{k=1}^{J}((R_k^{GS})^{-1}v_k, v_k) = \sum_{k=1}^{J}\|v_k\|_A^2 = \|v\|_A^2.$$

□

For the estimate for K_1, instead of V_k, we consider W_k defined in Corollary 7.1. It is not hard to see that all the previous results still hold. In addition, we have $P_iP_j = 0$, for any $i \ne j$.

Theorem 7.5 *Let ω_1 be the smallest constant such that*

$$(A_kv_k, v_k) \le \omega_1(R_k^{-1}v_k, v_k) \qquad \forall v_k \in V_k.$$

Then

$$K_1 \le \omega_1.$$

If $R_k = R_k^{DJ}$, then $\omega_1 = 1$. If $R_k = R_k^{GS}$, then $\omega_1 < 2$.

Proof. The bound for K_1 is a direct consequence of Lemma 4.6 in [149] and the fact that $P_iP_j = 0$ for $i \neq j$. If $R_k = R_k^{DJ}$, then

$$\omega_1 = \rho(R_k^{DJ}A_k) = \frac{1}{\rho((\mathcal{D}^k)^{-1}\mathcal{A}^k)}\rho((\mathcal{D}^k)^{-1}\mathcal{A}^k) = 1.$$

If $R_k = R_k^{GS}$, then

$$\frac{(A_kv_k, v_k)}{((R_k^{GS})^{-1}v_k, v_k)} = \frac{(\nu^k)^T\mathcal{A}^k\nu^k}{\frac{1}{2}(\nu^k)^T\mathcal{D}^k\nu^k + \frac{1}{2}(\nu^k)^T\mathcal{A}^k\nu^k} < 2.$$

Hence $\omega_1 < 2$. □

7.4.2 Two Dimensions

Since our interpolation is implicitly defined by the minimization problem, the analysis is difficult and has not yet been fully investigated. Nevertheless, a two level convergence analysis is presented. In this special case, only the constant preserving property is used. The role of the minimization is explained afterwards.

In contrast to the one-dimensional analysis, a more classical approach is adopted for the following analysis, where techniques in Sobolev space and finite element methods are heavily used. The proof is based on the analyses given by Xu [147] and Chan, Xu and Zikatanov [35]. In [35], fine grid elements are agglomerated to form *macroelements* which are defined explicitly in their algorithm. It turns out the set of coarse grid basis functions $\{\phi_i^H\}$ constructed in Section 7.3.2 also implicitly defines a set of macroelements.

Lemma 7.9 *The set of coarse grid basis functions $\{\phi_i^H\}$ defines a set of macroelements covering Ω. If Ω is a regular triangular mesh obtained by successive refinements, the macroelements are precisely the standard coarse triangular elements.*

Proof. For any ϕ_i^H, let S_i be its support. Let $\mathcal{G}_1 = \{S_i\}_{i=1}^m$. Take finite intersections of elements in $\mathcal{G}_1$ to form another set of subsets:

$$\mathcal{G}_2 \equiv \{\cap_i G_i : G_i \in \mathcal{G}_1\}.$$

Clearly, $\mathcal{G}_2 \supset \mathcal{G}_1$. Now remove the large redundant subsets from $\mathcal{G}_2$ and keep only the elementary ones:

$$\mathcal{G}_3 \equiv \mathcal{G}_2 \backslash \{G \in \mathcal{G}_2 : G = \cup_i G_i, G_i \neq G, G_i \in \mathcal{G}_2\}.$$

In general, elements in $\mathcal{G}_3$ may overlap. We agglomerate the overlapped macroelements to form a bigger macroelements. Then, the resulting set, $\mathcal{G}$, consists of nonoverlapping macroelements covering Ω.

Suppose Ω is a triangular mesh obtained by successive refinements. For any standard big triangular element, we have three coarse grid basis functions at the vertices associated with it. The intersection of their supports is exactly the big triangular element. Thus it is in $\mathcal{G}_2$. It is in fact in $\mathcal{G}_3$ since no other coarse grid basis function whose support intersects with this element. Hence $\mathcal{G} = \mathcal{G}_3$ consists of all such triangular elements and only them. □

Remark: In Lemma 7.9, we construct $\mathcal{G}$ from $\mathcal{G}_3$ by agglomerating overlapping macroelements. In the worst case, we may obtain a singleton–the domain Ω. However, in practice, the number of overlap is very small. Hence, we assume that the number of overlap is bounded by a constant independent of the mesh refinement.

Once the macroelements are defined, the convergence can be analyzed in a classical way. Let V^h and V^H be the coarse and fine space as before. We assume that for any $G^H \in \mathcal{G}$ obtained from Lemma 7.9, there exists an auxiliary big triangle K^H of diameter H containing G^H and its neighboring elements in the fine grid. Also, we assume that H/h is bounded by a constant. We verify the stability and the approximation properties (7.3), (7.4) which are equivalent to (7.26), (7.27) in the two level setting.

Theorem 7.6 *There exists an interpolant I^H such that for any $v \in V^h \subset H_0^1(\Omega)$,*

$$|I^H v|_{1,\Omega} \le C_0 |v|_{1,\Omega} \tag{7.26}$$

$$\|v - I^H v\|_{0,\Omega} \le C_1 h |v|_{1,\Omega}, \tag{7.27}$$

where $\|\cdot\|_{0,\Omega}$, $\|\cdot\|_{1,\Omega}$ and $|\cdot|_{1,\Omega}$ are the usual L^2, H^1 and H^1-semi norms, respectively.

Proof. The following proof is based on [35]. Define an averaged nodal value interpolation similar to that in [117] as follows. For any coarse grid point x_j^H, let e_j be an edge on the fine grid which contains x_j^H. Let ψ_j be the linear function on e_j such that

$$(v, \psi_j)_{0,e_j} = v(x_j^H) \qquad \forall v \in P_1(e_j),$$

where $P_1(e_j)$ is the set of piecewise linear functions on e_j and $(\cdot,\cdot)_{0.e_j}$ is the L^2 inner product on e_j. Define $I^H : V^h \to V^H$ by:

$$(I^H v)(x) = \sum_{j=1}^{n} (v, \psi_j)_{0,e_j} \, \phi_j^H(x).$$

We now prove (7.27). Let $G^H \in \mathcal{G}$ be a macroelement obtained from Lemma 7.9. By assumption, there exists an auxiliary triangle K^H containing G^H and its neighboring elements. Let $F_{K^H} : \hat{K} \to K^H$ be an affine mapping that maps the standard reference element $\hat{K}$ to K^H. Define $\widehat{G} \equiv F_{K^H}^{-1}(G^H) \subset \hat{K}$. Similarly, we have

$$\begin{aligned} \hat{v}(\hat{x}) &= v(F_{K^H}(\hat{x})) && \forall \hat{x} \in \hat{K} \\ \text{and} \qquad \widehat{I^H}\hat{v} &= \widehat{I^H v}. \end{aligned}$$

(Note: v is extended by zero if K^H is outside Ω). By the trace theorem, it can be proved that

$$\|\hat{v} - \widehat{I^H}\hat{v}\|_{0,\widehat{G}} \le C\|\hat{v}\|_{1,\hat{K}}.$$

Since the macroelements are nonoverlapping and the coarse grid basis functions $\{\phi_i^H\}$ preserve constants, constant functions are invariant under I^H and so are under $\widehat{I^H}$. Hence

$$\begin{aligned} \|\hat{v} - \widehat{I^H}\hat{v}\|_{0,\widehat{G}} &= \inf_{\hat{c}\in\Re} \|\hat{v} + \hat{c} - \widehat{I^H}(\hat{v} + \hat{c})\|_{0,\widehat{G}} \\ &\le C \inf_{\hat{c}\in\Re} \|\hat{v} + \hat{c}\|_{1,\hat{K}} \\ &\le C|\hat{v}|_{1,\hat{K}}. \end{aligned}$$

Transforming back to K^H, we have

$$\|v - I^H v\|_{0,G^H} \le CH|v|_{1,K^H}.$$

Summing over the macroelements, we obtain

$$\begin{aligned} \|v - I^H v\|_{0,\Omega}^2 &= \sum_{G^H\in\mathcal{G}} \|v - I^H v\|_{0,G^H}^2 \\ &\le CH^2 \sum_{K^H\supset G^H} |v|_{1,K^H}^2 \\ &\le CH^2 |v|_{1,\Re^2}^2. \end{aligned}$$

By the extension theorem in finite element theory, we may assume that $v \in H_0^1(\Re^2)$ satisfies:

$$|v|_{1,\Re^2} \le C|v|_{1,\Omega}.$$

Hence, inequality (7.27) follows.

Inequality (7.26) can be proved similarly using the H^1 norm instead and hence is omitted. □

By the convergence theory of Bramble, Pasciak, Wang and Xu [14] and Theorem 7.6, we conclude that the two level convergence is independent of the mesh size h.

Remark: The convergence analysis for two dimensions is only proved for the two level case. However, the numerical results in Section 7.5 show that the optimal convergence is in fact true for multilevel. Furthermore, optimal convergence is also demonstrated for unstructured grids in Chapter 5.

In the two level setting, it turns out that the stability and the approximation properties are implied by the constant preserving property only. In our experience, however, constant preserving is not sufficient to guarantee optimal convergence in the multilevel case. We need also control the energy of the basis functions.

The role of the minimization can be shown by estimating $\|I^H v\|_A$. Here I^H is the nodal value interpolant. For easy illustration, we assume that (7.10) has a positive lower order term $b(x,y)u$. By the triangle and the Cauchy-Schwarz inequalities, we have

$$\begin{aligned}
\|I^H v\|_A &= \|\sum_i v(x_i^H)\phi_i^H\|_A \\
&\leq \sum_i \|\phi_i^H\|_A |v(x_i^H)| \\
&\leq (\sum_i \|\phi_i^H\|_A^2)^{1/2} (\sum_i |v(x_i^H)|^2)^{1/2} \\
&\leq \frac{C}{h} (\sum_i \|\phi_i^H\|_A^2)^{1/2} \|v\|_2 \\
&\leq \frac{C}{h \min b(x,y)} (\sum_i \|\phi_i^H\|_A^2)^{1/2} \|v\|_A.
\end{aligned}$$

Although the bound may not be sharp, the minimization problem essentially minimizes the constant C_0 in the stability inequality (7.26).

7.5 Numerical Results

In this section, we present results of numerical experiments mainly in two dimensions to verify that the multigrid algorithm resulting from the energy-minimizing interpolation has optimal convergence behavior and is robust with respect to the coefficients of the PDEs. In all the numerical examples, the computational domain is $\Omega = [0,1] \times [0,1]$ with homogeneous Dirichlet boundary condition. In the multigrid procedure, a V-cycle is used with two pre- and two post- pointwise Gauss-Seidel smoothings. The iteration was terminated when the relative residual norm was less than 10^{-6}. The number of multigrid levels is such that the coarsest grid is a single point, or as otherwise stated.

In Section 7.3.2, we mentioned that it is not necessary to compute the Lagrange multipliers to machine precision. In all cases discussed below, we used piecewise linear or bilinear interpolation as our initial guess for the minimization problem. In the numerical results, we show how the accuracy of the Lagrange multipliers affect the efficiency and convergence of the resulting multigrid method. Moreover, as discussed in 7.3.3, the augmented stiffness matrix $\tilde{\mathcal{A}}^h$, or more precisely, $\tilde{\mathcal{A}}^h + \eta\mathcal{I}$, is a free preconditioner for solving the Lagrange multiplier equation (7.17). In the numerical examples, this preconditioner is used with η chosen as 10^{-3}.

Example 1: In the appendix, we proved that the energy-minimizing interpolation recovers the bilinear interpolation if $a(x) \equiv 1$ in the case when the structured square grid is used. But linear interpolation is not exactly obtained in the triangular grid case. In this example, we solve the following Poisson equation:

$$-\Delta u = 1,$$

on the triangular grid. The result is shown in Table 7.1. We vary the grid size from $h = 1/16$ to $h = 1/64$ and the number of multigrid levels from 3 to 6. We see that both the linear and the energy-minimizing interpolations give convergence rate independent of the mesh size and the number of multigrid level.

	Linear				Energy-min			
h	3	4	5	6	3	4	5	6
1/16	7	7	-	-	7	7	-	-
1/32	6	7	7	-	6	7	7	-
1/64	6	7	7	7	6	7	7	7

Table 7.1: Number of V-cycles using linear and energy-minimizing interpolations when $a(x) \equiv 1$.

Example 2: In this example, we verify numerically that the convergence does not depend on the number of levels. Here we consider the following PDE with a smooth coefficient:

$$-\nabla \cdot (1 + x\exp(y))\nabla u = 1.$$

Table 7.2 shows the number of MG iteration to convergence. We denote the multigrid method with bilinear interpolation by MGBL and our energy-minimizing multigrid method by EMMG(ϵ) where ϵ specifies the stopping criterion for the conjugate gradient (CG) method applied to the Lagrange multiplier equation (7.17). More precisely, the CG iteration stopped when the relative residual norm is less than ϵ. We see that when the optimization problem is effectively solved ($\epsilon = 10^{-12}$), the convergence is independent of the mesh size h and the number of levels. In fact, we observe that same convergence can be achieved even if the optimization problem is solved approximately ($\epsilon = 10^{-1}$). Thus we may reduce the cost by applying significantly fewer number of CG iterations as shown in Table 7.3 which gives the number of CG iterations at each multigrid level to solve (7.17).

We remark that this example is used to illustrate the optimal convergence of EMMG(ϵ) and the effect of varying ϵ only. It is not cost effective to use energy-minimizing interpolation when bilinear interpolation works well.

Example 3: We compare the multigrid method using bilinear interpolation with that using energy-minimizing interpolation by solving the following discontinuous coefficient problem [1, modified Example I]:

$$-\nabla \cdot a(x, y)\nabla u = 1,$$

	MGBL				EMMG(10^{-1})				EMMG(10^{-12})			
h	4	5	6	7	4	5	6	7	4	5	6	7
1/16	5	-	-	-	5	-	-	-	5	-	-	-
1/32	5	5	-	-	5	5	-	-	5	5	-	-
1/64	5	5	5	-	5	5	5	-	5	5	5	-
1/128	5	5	5	5	5	5	5	5	5	5	5	5

Table 7.2: Number of V-cycles using bilinear and energy-minimizing interpolations when $a(x) = 1 + x\exp(y)$.

h	level	EMMG(10^{-1})	EMMG(10^{-12})
1/16	4	1	53
	2	1	22
1/32	5	1	98
	3	1	33
	2	1	22
1/64	6	1	180
	4	1	53
	2	1	22
1/128	7	1	309
	5	1	98
	3	1	33
	2	1	22

Table 7.3: Number of CG iterations at each multigrid level with varying ϵ when $a(x) = 1 + x\exp(y)$.

where

$$a(x,y) = \begin{cases} a^+ & 0.25 \le x \le 0.75 \ \ \& \ \ 0.25 \le y \le 0.75 \\ a^- & \text{otherwise.} \end{cases}$$

We fix $a^- = 1$ and vary a^+ from 10 to 10^4. The convergence results are given in Table 7.4. Same notations are used as in Example 1. Here $*$ denotes convergence beyond 100 multigrid iterations. Consisted with the classical theory, the convergence rate of the standard multigrid does not depend on the mesh size h. However, the convergence rate deteriorates substantially as the jump of the discontinuity increases. On the other hand, the convergence of the energy-minimizing multigrid method does not depend both on the mesh size and the size of the jump. Again, EMMG(10^{-1}) shows similar convergence as EMMG(10^{-12}).

Table 7.5 shows the average number of CG iterations on the fine grid, in place of the number of CG iterations on each grid level shown in Table 7.3 does. It is computed as follows. One CG iteration on the first coarse grid is counted as 1/2 CG iteration on the fine grid and so on. By applying only three extra CG iterations to construct the energy-minimizing interpolation, the convergence of the multigrid is improved significantly. This demonstrates that extra cost of solving the minimization problem is justified by the much faster convergence of the multigrid method.

	MGBL				EMMG(10^{-1})				EMMG(10^{-12})			
h	10	10^2	10^3	10^4	10	10^2	10^3	10^4	10	10^2	10^3	10^4
1/16	14	*	*	*	6	5	6	6	6	5	5	5
1/32	14	*	*	*	6	6	6	6	6	6	6	6
1/64	14	*	*	*	6	6	7	7	6	6	6	6
1/128	14	*	*	*	7	7	7	7	7	6	6	6

Table 7.4: Number of V-cycles using bilinear and energy-minimizing interpolations for the discontinuous coefficient problem. The jump $a^+ = 10, 10^2, 10^3, 10^4$.

	EMMG(10^{-1})		EMMG(10^{-12})	
h	10	10^4	10	10^4
1/16	3.00	3.00	3.00	5.50
1/32	2.50	2.50	2.50	3.63
1/64	2.25	2.25	2.25	2.88
1/128	2.13	2.13	2.13	2.44

Table 7.5: Average number of CG iterations on the fine grid for the discontinuous coefficient problem. The jump $a^+ = 10, 10^4$.

Example 4: We solve another PDE to demonstrate the robustness of the energy-minimizing multigrid method. This time, the coefficient is oscillatory and the equation is [78, Example 7.4]:

$$-\nabla \cdot \frac{1}{(2+P\sin(x/\epsilon))(2+P\sin(y/\epsilon))}\nabla u = 1.$$

We chose $P = 1.99$ and ϵ=0.1 and 0.01. The results are shown in Table 7.6 and 7.7. This time, the coefficient is very rough and the minimization problem is more difficult to solve. In this case, EMMG(10^{-1}) is not accurate enough to have good convergence. However, with an slightly increase in the accuracy, EMMG(10^{-2}) recovers the same rapid convergence of EMMG(10^{-12}).

We remark that the nonuniform number of V-cycles to convergence for the case $\epsilon = 0.01$ may be because the mesh size h is not small enough to resolve the coefficient $a(x, y)$ for the first couple of values of h.

	MGBL		EMMG(10^{-1})		EMMG(10^{-2})		EMMG(10^{-12})	
h	0.1	0.01	0.1	0.01	0.1	0.01	0.1	0.01
1/16	*	4	7	5	7	5	7	5
1/32	51	*	23	14	7	14	7	14
1/64	65	58	*	11	7	7	7	7
1/128	66	*	*	11	7	10	7	10

Table 7.6: Number of V-cycles using bilinear and energy-minimizing interpolations for the oscillatory coefficient problem. $\epsilon = 0.1, 0.01$.

	EMMG(10^{-1})		EMMG(10^{-2})		EMMG(10^{-12})	
h	0.1	0.01	0.1	0.01	0.1	0.01
1/16	38.75	1.75	42.25	1.75	90.75	56.75
1/32	13.63	63.88	62.38	71.00	124.13	184.13
1/64	3.81	82.94	115.94	127.94	228.06	308.69
1/128	2.22	211.22	177.94	316.22	388.78	664.22

Table 7.7: Average number of CG iterations on the fine grid for the oscillatory coefficient problem. $\epsilon = 0.1, 0.01$.

Example 5: We show by a one-dimensional Helmholtz equation that the energy minimization principle is not restricted to positive definite second order elliptic PDEs. The model equation is

$$\Delta u + \alpha u = 1, \tag{7.28}$$

where α is a positive constant. This operator is indefinite.

We use multigrid to solve the linear system A^h. For this problem, we obtained ϕ_i^H from solving the local PDEs (7.7), not from the minimization problem (7.12), since constant functions are not in the kernel of A^h. The convergence results of

h	Linear	Energy
1/32	$*$	5
1/64	$*$	5
1/128	$*$	5

Table 7.8: Number of V-cycles using linear and energy-minimizing interpolations for the Helmholtz problem.

multigrid methods using linear and energy-minimizing interpolations are shown in Table 7.8. The $*$ in the first column indicates that standard multigrid takes more than 100 V-cycles to convergence. The poor convergence comes from the effect of smoothing and the way the interpolation is done. The eigenfunctions of the operator $\mathcal{A}^h$ corresponding to small energy are oscillatory whereas those corresponding to large energy are relatively smooth. As a result of standard relaxation smoothings (cf. Chapter 4), the errors become more oscillatory. Figure 7.3 shows the effect of 4 and 8 iterations of Gauss-Seidel smoothing applied to a smooth initial error. Such phenomenon was also discussed in [25]. Hence if we use linear interpolation, it will not be able to approximate the oscillatory error on the coarser subspaces. This causes the failure of the standard multigrid method.

On the other hand, the multigrid method using energy-minimizing interpolation works fine and does not show any deterioration. It is because the energy minimization captures the property of this type of operators and produces oscillatory coarse grid basis functions (see Figure 7.4). This consistency enables a good approximation on the coarser subspaces, and hence the multigrid convergence is much better.

Remark: The coarse grid basis functions obtained by solving the local PDEs do not preserve constants, an approach that is natural because the operator $\mathcal{A}$ does not annihilate constant functions. If we were to extend our minimization formulation to this case in higher dimensions, we would have to modify the constraint in (7.12).

7.6 Concluding Remarks

Through the analytical and numerical results, we have demonstrated that energy-minimizing and constant preserving are two key properties of the coarse grid interpolation required to have a robust multigrid method. An obvious drawback to the construction of the robust interpolation is the expensive solve of the minimization problem. An inexact preconditioned conjugate gradient method with the linear interpolation as initial guess is proposed to overcome this problem and the numerical results showed that the setup cost is not too expensive, especially when the system is to be solved many times. Nevertheless, more efficient methods to solve the minimization problem need to be derived and studied.

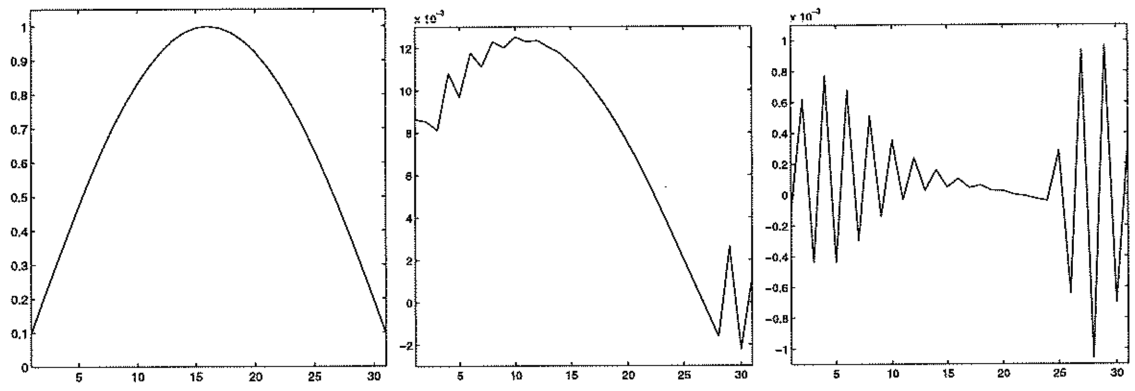


Figure 7.3: Left to right: errors after 0, 4 and 8 Gauss-Seidel iterations when $\mathcal{A}^h$ is the Helmholtz operator.

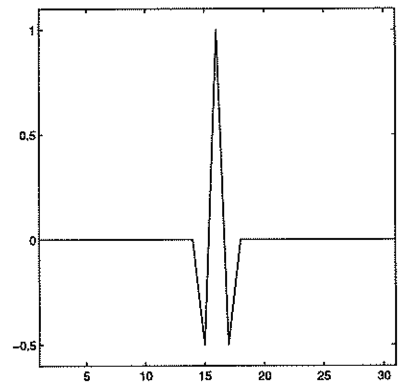


Figure 7.4: A coarse grid basis function obtained by the energy minimization when $\mathcal{A}^h$ is the Helmholtz operator.

Finally, because of the algebraic nature of the construction of the interpolation, we remark that our method is also applicable to complicated geometries, for instance, unstructured grids; see Chapter 5.

CHAPTER 8

Application: Multigrid Methods for Differential-Convolution Equations

8.1 Introduction

In PDE-based image processing, we often need to solve differential-convolution equations of the form:

$$\alpha R(u)(x) + \int_{\Omega} k(x-y)u(y)dy = f(x) \text{ in } \Omega, \tag{8.1}$$

where $u(x)$ is the recovered image, $k(x)$ is the kernel convolution function, $R(u)$ is a regularization functional and α is a positive parameter. Typical forms of $R(u)$ are:

$$R(u) = \begin{cases} u & \text{Tikhonov} \\ -\Delta u & \text{Isotropic Diffusion (ID)} \\ -\nabla \cdot (\nabla u / |\nabla u|) & \text{Total Variation (TV).} \end{cases}$$

The kernel function $k(x)$ represents the blurring effect to the image $u(x)$, which makes the problem ill-posed. The functional $R(u)$, on the other hand, regularizes the ill-posedness of the problem, and the function $f(x)$ represents the given blurred and noisy image.

The discretization of (8.1) gives rise to a linear system of the form:

$$(\alpha A + K)u = f, \tag{8.2}$$

with the following properties. The matrix A, corresponding to the regularization part, is typically sparse, symmetric and positive-definite (positive semi-definite for the ID and TV case because the boundary condition is Neumann). The matrix K, corresponding to the convolution part, is typically ill-conditioned, symmetric and dense but with a Toeplitz structure. In this chapter, we are interested in using iterative methods to solve a large system of the form (8.2).

For the matrix K, various preconditioners have been proposed, for example, circulant preconditioners [24, 26, 126], sine transform preconditioners [23], cosine transform preconditioners [21], etc. For these types of preconditioners, the eigenvalues of the preconditioned system typically cluster around one which is a very desirable condition for the conjugate gradient method. Recently, a MG preconditioner [22] has also been proposed and optimal convergence is proved for a class of Toeplitz systems.

The construction of preconditioners for the sum of operators $L = \alpha A + K$, however, is difficult. Suppose M_A and M_K are two efficient preconditioners for A and K respectively. Then $M_L = \alpha M_A + M_K$ would be a good approximation to L. Unfortunately, M_L is not easily invertible in general even if M_A and M_K are.

A simple strategy is to use either M_A or M_K alone to precondition L. Oman [102] and Vogel [138] constructed a MG preconditioner for $\tilde{L} = \alpha A + \gamma I$ which in turn is used to precondition L, hoping that the matrix K is well approximated by γI. A potential drawback is that γI may be a poor approximation to K.

In such situations, the operator splitting method of Vogel and Oman [139] may be more effective. This preconditioner approximates the inverse of L by a product of factors each involving only either A or K:

$$M = (K + \gamma I)^{1/2}(\alpha A + \gamma I)(K + \gamma I)^{1/2},$$

where γ is an appropriately chosen constant. This preconditioner is very effective for both very large and very small values of α but the performance can deteriorate for intermediate values of α.

To alleviate this problem, R. Chan, T. Chan and Wong [21] proposed a class of optimal fast transform based preconditioners to precondition L. The main idea is to select as preconditioner the best approximation to L from a fast transform invertible class of matrices by solving the following optimization problem:

$$\min_{M \in C} ||M - L||_F,$$

where C is the class of matrices diagonalizable by the cosine transform. Such optimal fast transform based preconditioners have proven to be very effective for convolution type problems [24] and they have also been extended to elliptic problems [20]. It turns out that the optimal M for L can be computed very efficiently by exploiting the Toeplitz structure of K and the banded structure of A. Since L is not "split" in arriving at a preconditioner, the performance is not sensitive to the value of α. However, even though the performance is very satisfactory for Tikhonov and ID regularization, the convergence behavior for the TV regularization case may still depend on the mesh size. This is caused by the highly varying coefficient in the TV operator.

In view of the effectiveness of MG for A and the fast transform preconditioners for K, our idea is to combine the benefits of both. Specifically, we use the fast transform based preconditioned conjugate gradient as a smoother for MG. Our analysis and numerical results show that this is an effective smoother, whereas the standard relaxation type preconditioners are totally ineffective for convolution type problems. In the following, we focus on two one dimensional cases: (1) $A = I$ (identity) (2) $A = -\Delta$ (Laplacian operator). In section 8.2 and 8.3, we discuss the difficulties of using MG for $L = \alpha I + K$ and $L = -\alpha\Delta + K$ and how we tackle it through the use of fast transform based smoothers. In section 8.4, we discuss the total variation case which is much more difficult. Although we have some encouraging results, we still have not arrived at an effective method. In 8.5, we estimate the complexity of some of the methods discussed. Finally, some conclusions are made in section 8.6.

8.2 The Case $A = I$

In this section, we consider operators of the form $L = \alpha I + K$, where K arises from the discretization of an integral operator of the first kind. It is well-known that K is very ill-conditioned and MG with traditional smoothers does not work well for K. The regularization term αI improves the conditioning by shifting the spectrum a distance α away from zero. However, this is not enough to make MG work well. The reason is related to the set of eigenvectors of K which is the same as L and is explained as follows.

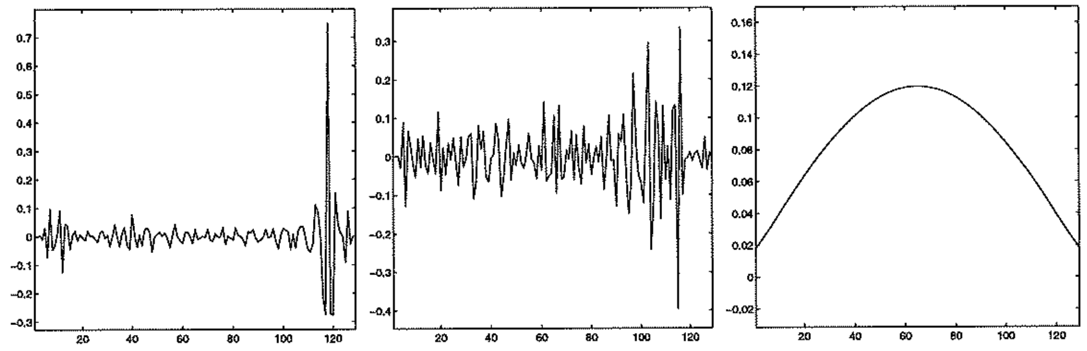


Figure 8.1: Eigenvectors corresponding to (a) the smallest (b) the middle (c) the largest eigenvalue of $L = 10^{-4}I + K$. The oscillatory eigenvectors corresponding to the small eigenvalues.

Our observation is that common relaxation methods, for instance, Richardson, Jacobi or Gauss-Seidel, fail to smooth the error in the geometric sense. The reason is that, unlike in the elliptic case, eigenvectors corresponding to small eigenvalues are highly oscillatory while those corresponding to large eigenvalues are smooth. It is known that relaxation methods reduce the error components corresponding to large eigenvalues only and therefore they in fact remove the smooth error components; see Chapter 4 Section 4.2.

We illustrate the smoothing phenomenon of the Richardson iteration applied to $L = \alpha I + K$ by a simple example. A typical choice for the parameters in image processing is: $\alpha = 10^{-4}$, $k(x) = \frac{1}{C}\exp(-x^2/0.01), C = \int_0^1 \exp(-x^2/0.01)dx$, (Gaussian blurring operator). Let $0 < \lambda_1 \leq \cdots \leq \lambda_n$ be the eigenvalues of L and $v_1, \ldots, v_n$ be the corresponding eigenvectors. Figure 8.1 shows the plots of v_1, $v_{n/2}$ and v_n. Relaxation methods, for example, Richardson, essentially reduces

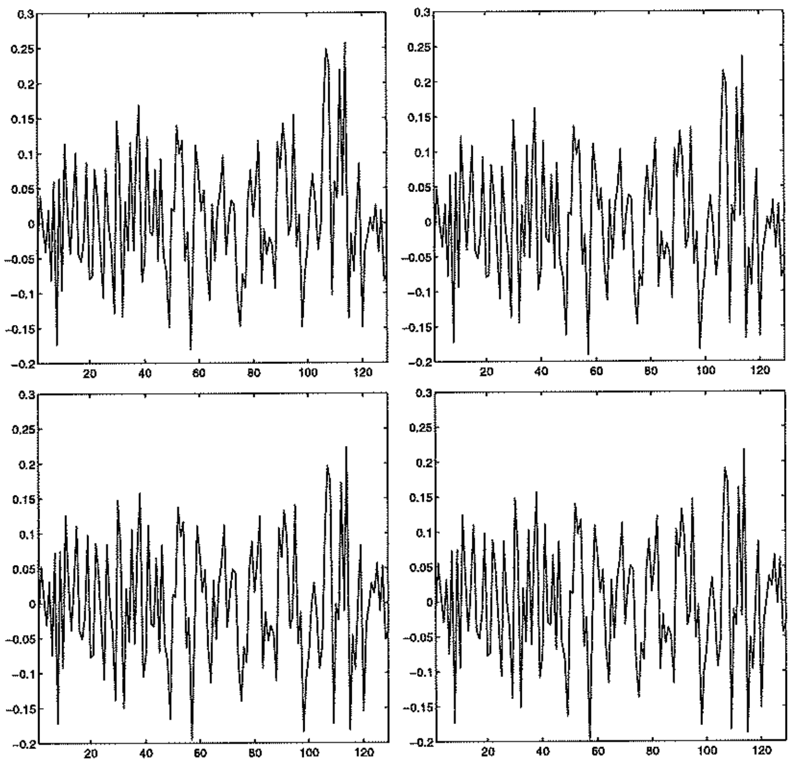


Figure 8.2: Error vectors after 0 iteration (top left), 1 iteration (top right), 5 iterations (bottom left), and 10 iterations (bottom right) of Richardson smoothing applied to $L = 10^{-4}I + K$. Note that there is no smoothing effect.

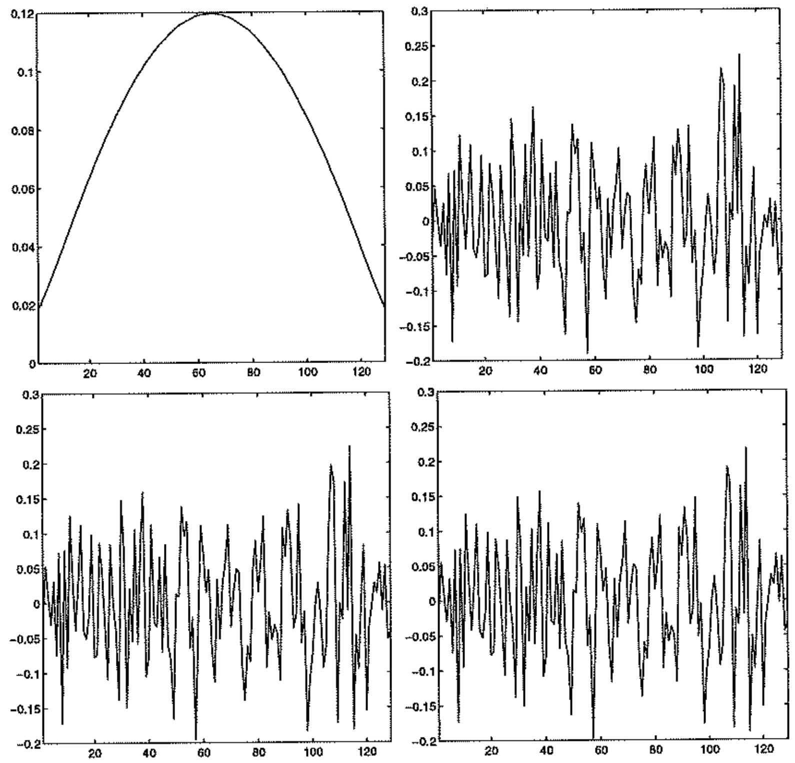


Figure 8.3: Error vectors after 0 iteration (top left), 1 iteration (top right), 5 iterations (bottom left), and 10 iterations (bottom right) of Richardson smoothing applied to $L = 10^{-4}I + K$. The smooth component is removed completely after only 1 iteration whereas the oscillatory components persist. All the plots are scaled so that the l_2-norm of the vector is equal to 1.

the error components corresponding to large eigenvalues, not necessary the high frequencies. Because of the special spectrum of L, these methods do not reduce the high frequency errors. Figure 8.2 shows the plots of the initial (oscillatory) error and errors after 1, 5, 10 number of Richardson iterations. No smoothing effect can be seen. In fact, as shown in Figure 8.3, if the initial error consists of low frequency and a small perturbation of high frequency vectors, after one Richardson iteration, the low frequency components will be removed and the error is left with high frequency only.

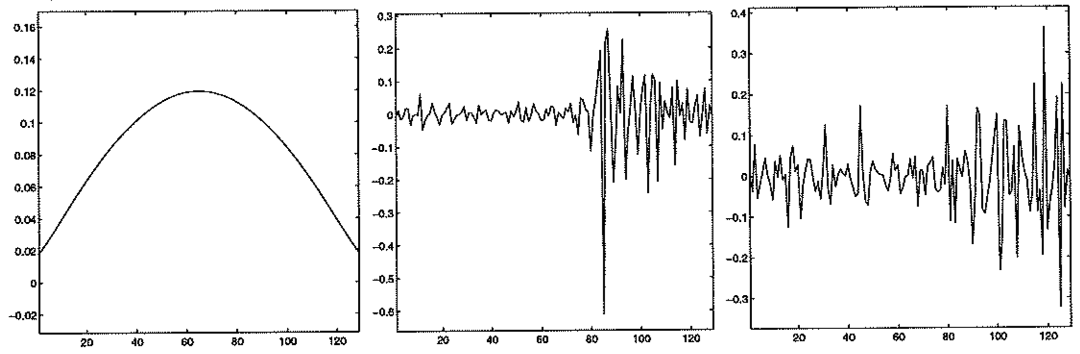


Figure 8.4: Eigenvectors corresponding to (a) the smallest (b) the middle (c) the largest eigenvalue of $L = I - K$. The oscillatory eigenvectors correspond to the largest eigenvalues.

In contrast, MG converges rapidly for integral operators of the second kind of the form $L = I - K$ and this can also be explained by the smoothing argument. Figure 8.4 shows the eigenvectors of $L = I - K$ with K as before. Because of the minus sign, eigenvectors corresponding to small eigenvalues are smooth while those of large eigenvalues are oscillatory as in the standard elliptic case. Thus the Richardson iteration has no difficulty removing high frequency errors as shown in Figure 8.5 and 8.6.

With the above understanding, it is clear that MG does not work well for $L = \alpha I + K$ because the standard smoothers are not effective and we need to devise smoothers which can remove high frequency error components more effectively. Our approach is based on two observations. First, fast transform preconditioners are effective for clustering the eigenvalues of L around one. Second, conjugate gradient annihilates efficiently error components corresponding to clusters of eigenvalues, in addition to those at both ends of the spectrum. Hence we propose to use PCG with fast transform preconditioners as smoother in the MG cycle.

Figure 8.7 shows the eigenvectors of the preconditioned system using the cosine transform preconditioner. It is interesting to note that low frequency vectors are located at both ends of the spectrum while high frequency vectors concentrate at the cluster. Figure 8.8 shows the smoothing effect of PCG using the cosine transform preconditioner (PCG(Cos)). We remark that MG with the optimal circulant

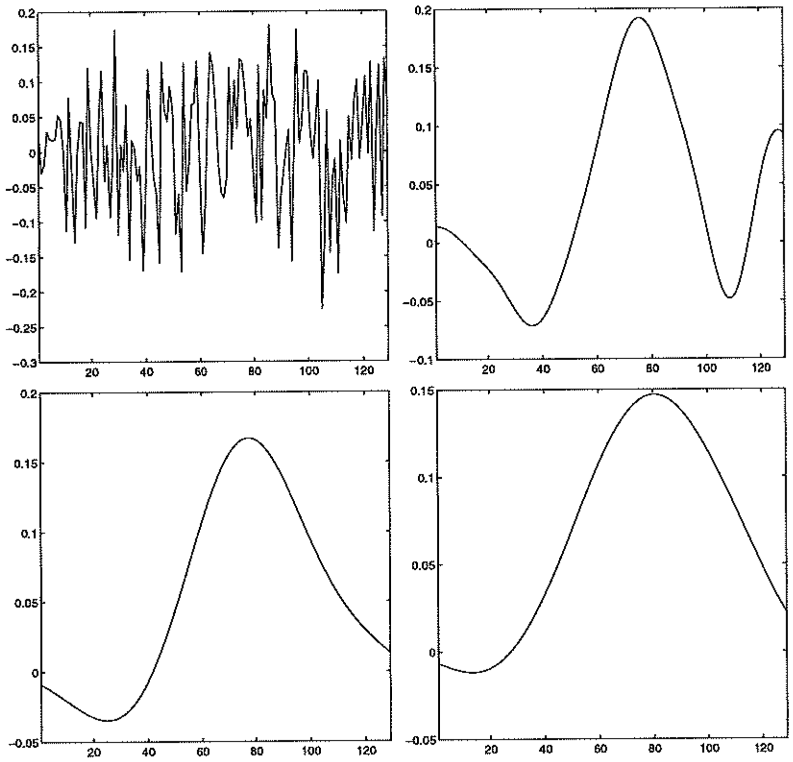


Figure 8.5: Error vectors after 0 iteration (top left), 1 iteration (top right), 5 iterations (bottom left) and 10 iterations (bottom right) of Richardson smoothing applied to $L = I - K$. The oscillatory components are quickly smoothed out.

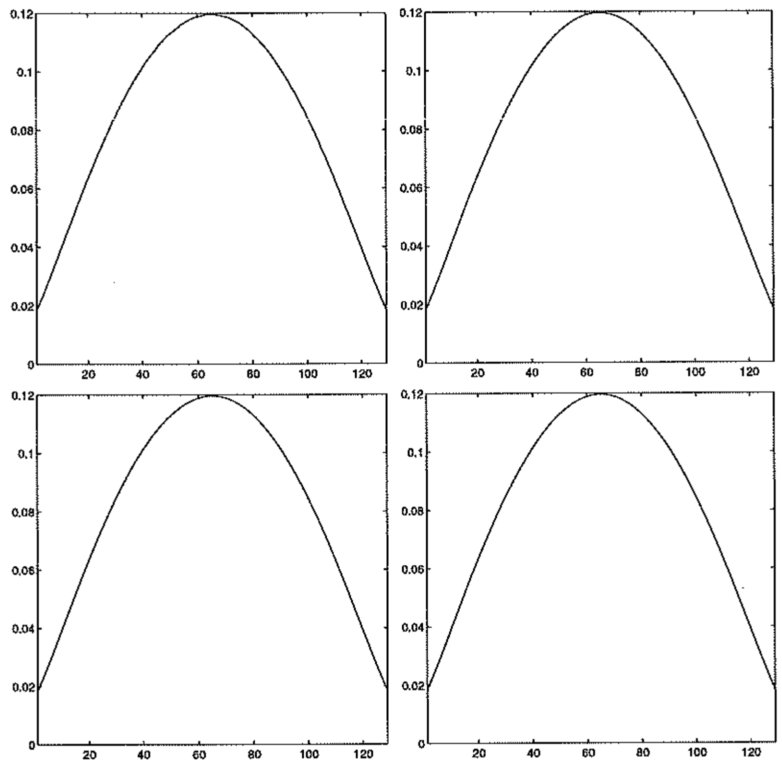


Figure 8.6: Error vectors after 0 iteration (top left), 1 iteration (top right), 5 iterations (bottom left) and 10 iterations (bottom right) of Richardson smoothing applied to $L = I - K$. The smooth components remain after many iterations.

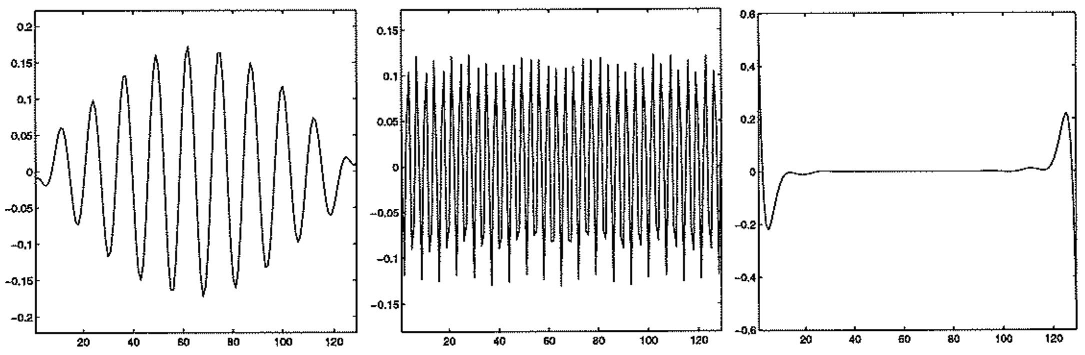


Figure 8.7: Eigenvectors corresponding to (a) the smallest (b) the middle (c) the largest eigenvalue of the cosine transform preconditioned system of $L = 10^{-4}I + K$. The oscillatory eigenvectors are clustered in the middle of the spectrum.

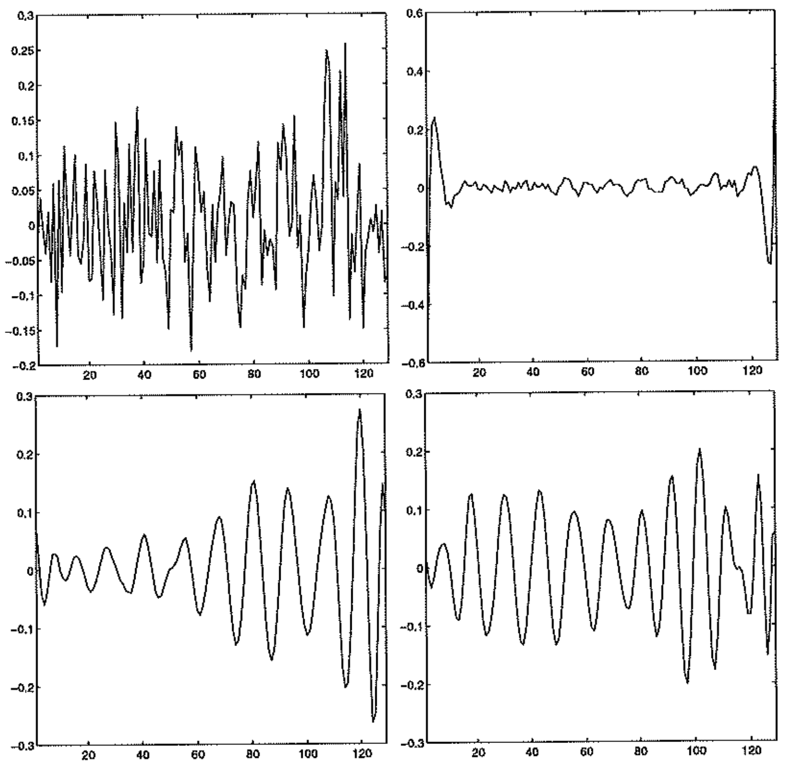


Figure 8.8: Error vectors after 0 iteration (top left), 1 iteration (top right), 5 iterations (bottom left) and 10 iterations (bottom right) of PCG(Cos) smoothing applied to $L = 10^{-4}I + K$. The smoothing effect is much improved over Richardson in Figure 5.2.

preconditioner also produces similar plots and hence we do not show it. Table 8.1 shows the MG convergence (MG(*)) of different smoothers specified in the brackets. The Richardson smoother is denoted by R and the PCG smoother with the cosine transform preconditioner is denoted by PCG(Cos). The convergence of PCG(Cos) alone is also given for comparison. Here we use two pre-smoothing and no post-smoothing step. The iteration is stopped when the relative residual is less than 10^{-10}. The matrix K is the Gaussian blurring operator as before. From the table, we see that PCG as smoother is much more efficient than standard relaxation methods in all cases. For large α, MG with PCG as smoother is about as efficient as PCG alone, taking into the account of two smoothing steps in each MG iteration. But for small α, MG is significantly better. In fact, its performance improves as the mesh size approaches zero whereas that of PCG alone remains constant.

α	h	1/64	1/128	1/256	1/512
	MG(R)	*	*	*	*
10^{-2}	MG(PCG(Cos))	5	4	4	4
	PCG(Cos)	8	8	8	8
	MG(R)	*	*	*	*
10^{-3}	MG(PCG(Cos))	6	5	4	4
	PCG(Cos)	11	11	11	11
	MG(R)	*	*	*	*
10^{-4}	MG(PCG(Cos))	11	7	6	6
	PCG(Cos)	18	18	18	18
	MG(R)	*	*	*	*
10^{-5}	MG(PCG(Cos))	40	18	14	11
	PCG(Cos)	33	37	36	38

Table 8.1: Convergence of different MG and PCG with varying α and mesh size h. $L = \alpha I + K$. $*$ indicates more than 100 iterations. The results show that PCG(Cos) is an effective smoother.

8.3 The Case $A = -\Delta$

In the following, we assume Neumann boundary condition for the Laplacian operator. The situation of $L = -\alpha\Delta + K$ is much more complicated. First of all, the regularization term $\alpha\Delta$ does not simply shift the spectrum; it actually alters the spectrum. For large α, the eigenvectors of L resemble those of Δ and for small α, they resemble those of K, where the high and low frequency vectors are flipped over each other. For α in between, it is a mixture but the precise nature of the mixing is not known. We pick three different size of α to illustrate the changing spectrum of L in Figure 8.9, 8.10 and 8.11.

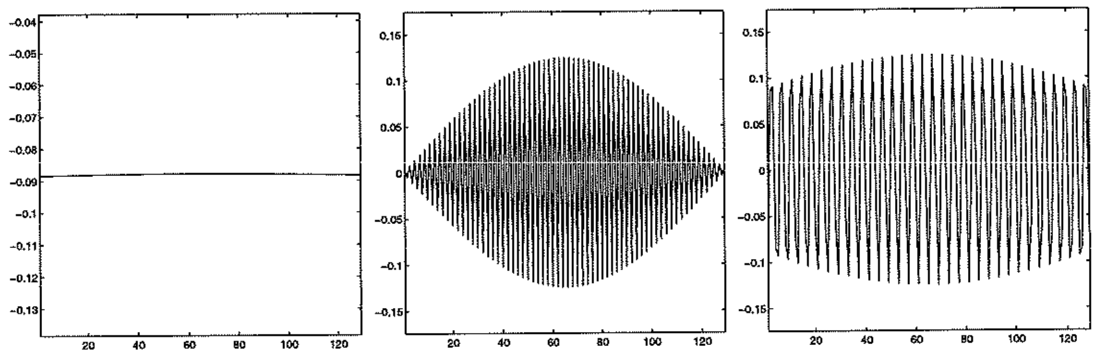


Figure 8.9: Eigenvectors corresponding to (a) the smallest (b) the middle (c) the largest eigenvalue of $L = -\Delta + K$. When α is large, the eigenvectors of L resemble those of $-\Delta$.

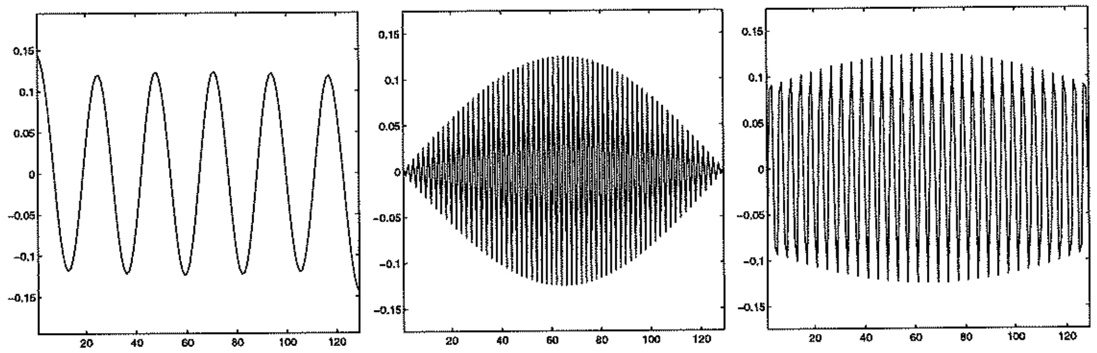


Figure 8.10: Eigenvectors corresponding to (a) the smallest (b) the middle (c) the largest eigenvalue of $L = -10^{-4}\Delta + K$. For intermediate value of α, the eigenvectors corresponding to large eigenvalues resemble those of $-\Delta$ and the eigenvectors corresponding to small eigenvalues are oscillatory and resemble those of K.

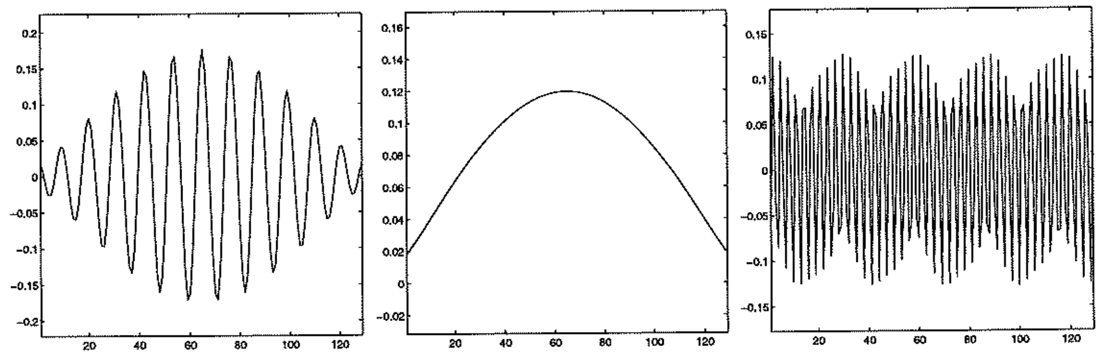


Figure 8.11: Eigenvectors corresponding to (a) the smallest (b) the middle (c) the largest eigenvalue of $L = -10^{-8}\Delta + K$. When α is small, the eigenvectors of L resemble those of K.

The numerical results for this case is given in Table 8.2. As expected, MG with standard relaxation methods as smoother deteriorates when α decreases because L approaches the convolution operator K which we have shown in section 8.2 that standard smoothers do not work well. Again, MG(PCG(Cos)) shows better performance over PCG(Cos) alone for small values of α and h.

α	h	1/64	1/128	1/256	1/512
10^{-2}	MG(R)	18	19	20	21
	MG(PCG(Cos))	3	3	3	3
	PCG(Cos)	6	6	6	6
10^{-3}	MG(R)	17	18	19	20
	MG(PCG(Cos))	3	3	3	3
	PCG(Cos)	7	7	7	7
10^{-4}	MG(R)	32	32	32	32
	MG(PCG(Cos))	4	4	3	3
	PCG(Cos)	8	8	8	8
10^{-5}	MG(R)	*	*	*	*
	MG(PCG(Cos))	4	4	3	3
	PCG(Cos)	9	9	9	9

Table 8.2: Convergence of different MG and PCG with varying α and mesh size h. $L = -\alpha\Delta + K$. The results show that PCG(Cos) is an effective smoother.

8.4 MG for TV deblurring

In this section, we discuss our preliminary experience in solving the TV based deblurring problem [27] by MG. The governing differential-convolution equation is slightly different from (8.1) and is given here:

$$\alpha R(u)(x) + \mathcal{K}^*\mathcal{K}(u) = \mathcal{K}^*z,$$

where $R(u) = -\nabla\cdot(1/|\nabla u|)\nabla u$, $\mathcal{K}u = \int_\Omega k(x-y)u(y)dx$, $\mathcal{K}^*$ is the adjoint operator of $\mathcal{K}$ and z is the observed blurred and noisy image. Basically, the convolution operator is replaced by a product of itself and its adjoint. The corresponding linear system becomes:

$$(\alpha A + K^T K)u = f, \tag{8.3}$$

which is similar to (8.2) with K replaced by K^TK. The additional challenges of solving (8.3) are two fold. First, the matrix A now comes from an elliptic operator with highly varying coefficient (which is $1/|\nabla u|$). It is not known if MG can handle this case efficiently. Second, the product K^TK is no longer Toeplitz which complicates the implementation. For instance, while it is trivial to construct the

Jacobi preconditioner for K, it is not so for K^TK at the first place, although it turns out that it can also be done in $O(n)$ operations. Moreover, the conditioning of K^TK is worse than K alone.

In this case, MG with PCG as smoother does not work well. A natural way to improve its performance is to use it as a preconditioner for conjugate gradient. However, this is not feasible as MG with PCG as smoother gives rise to a nonstationary preconditioner. One solution to this problem is based on the following observation. The success of PCG as smoother is that CG takes advantage of the clustered eigenvalues of the cosine transform preconditioned system. We notice that it is probably advantageous but not necessary to apply CG (which gives rise to a nonstationary preconditioner) to the preconditioned system. An alternative is to use standard relaxation methods on the cosine transform preconditioned system.

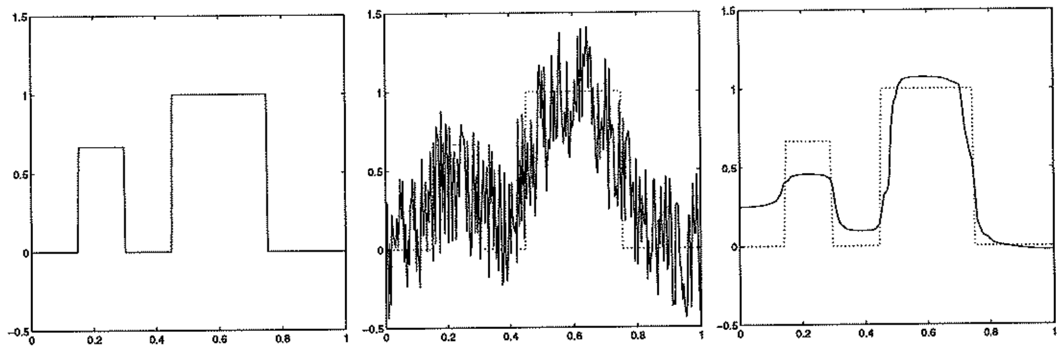


Figure 8.12: (a) Original image (b) Blurred and noisy image (c) Recovered image. Gaussian blur is used and SNR=13.

We have tested out several possibilities and the results are shown in Table 8.3, 8.4 and 8.5 for a TV deblurring example. The original and the blurred noisy 1D image together with the recovered image are shown in Figure 8.12. The signal-to-noise ratio SNR=13. Here we have used the Gaussian blur again. For each grid size h, we use the *optimal* α_{opt} for L which is chosen so that the recovered image has the specified SNR, if it were to be blurred and added the same noise. We test three cases with varying α: $\alpha = 10 * \alpha_{opt}$, $\alpha = \alpha_{opt}$ and $\alpha = 0.1 * \alpha_{opt}$, and the results are shown in Tables 8.3- 8.5 respectively. In each table, the second to fourth column show the convergence in the first fixed point iteration and the fifth to seventh ones show the convergence at the 11th fixed point iteration. (For our examples, the fixed point iteration has already converged at the 11th iteration.) We show these two sets of results because the coefficient $1/|\nabla u|$ is quite different for the two cases; see Figure 8.12. In the first fixed point iteration, the coefficient is very oscillatory whereas at the eleventh iteration, it is almost piecewise constant. With the same notation as before, the bracket followed PCG specifies the preconditioner used for CG and the bracket followed MG specifies the smoother. Here GS+Cos denotes the Gauss-Seidel (GS) method applied to the cosine transform preconditioned system.

Similarly for J+Cos where J denotes the Jacobi method.

$10 * \alpha_{opt}$	1st fixed pt. iter.			11th fixed pt. iter.		
	1/64	1/128	1/256	1/64	1/128	1/256
PCG(Cos)	42	85	108	13	49	75
PCG(MG(GS))	20	29	35	12	17	21
PCG(MG(GS+Cos))	14	28	37	4	11	12
PCG(MG(J+Cos))	17	51	79	13	17	22

Table 8.3: Convergence of PCG with varying h. $\alpha = 10 * \alpha_{opt}$.

α_{opt}	1st fixed pt. iter.			11th fixed pt. iter.		
	1/64	1/128	1/256	1/64	1/128	1/256
PCG(Cos)	38	81	98	42	92	106
PCG(MG(GS))	17	28	37	16	21	24
PCG(MG(GS+Cos))	14	26	34	13	15	14
PCG(MG(J+Cos))	17	45	73	20	28	28

Table 8.4: Convergence of PCG with varying h. $\alpha = \alpha_{opt}$.

$0.1 * \alpha_{opt}$	1st fixed pt. iter.			11th fixed pt. iter.		
	1/64	1/128	1/256	1/64	1/128	1/256
PCG(Cos)	35	75	93	55	90	128
PCG(MG(GS))	19	35	54	15	21	25
PCG(MG(GS+Cos))	13	24	33	15	19	17
PCG(MG(J+Cos))	16	43	67	22	35	36

Table 8.5: Convergence of PCG with varying h. $\alpha = 0.1 * \alpha_{opt}$.

We see that PCG(MG(GS)) and PCG(MG(GS+Cos)) are the best. They are not sensitive to α and their deterioration with smaller h is slow. Besides, PCG(MG(GS+Cos)) is better than PCG(MG(GS)) for smaller α which shows that cosine transform is effective in dealing with K. However, we have not come up with an efficient implementation for these two methods. For PCG(MG(GS)), Vogel [137] has also made a similar observation independently. Note also that PCG(MG(J+Cos)) shows a degradation over PCG(MG(GS+Cos)), similar to that of the ordinary GS over Jacobi. We should also remark that PCG(Cos) is quite effective among the methods we have tested.

8.5 Computation Complexity

Here we estimate the complexity of one iteration of some of the methods that we have described in section 8.2 and 8.3.

PCG(Cos): This method has been estimated in [21]. The construction of the preconditioner is $O(n)$ and the cost of the preconditioning is $O(n \log_2 n)$.

MG(R): On each level, the cost of a Richardson smoothing is essentially the cost of matrix-vector multiply. For the sparse matrix A, it can be done in $O(n_l)$ operations and for the Toeplitz matrix, it can be done in $O(n_l \log_2 n_l)$, where n_l is the size of the matrix at level l. Here we assume that K is Toeplitz at all levels. If the linear interpolation is used, the matrix K, indeed, is Toeplitz [22]. The construction of the coarse grid matrices can also be done in $O(n)$. Thus the overall complexity of an iteration of MG(R) is $O(n \log_2 n)$.

MG(PCG(Cos)): The method is almost the same as MG(R) but with different smoother. The cost of applying the PCG(Cos) is $O(n \log_2 n)$ and hence the overall complexity is $O(n \log_2 n)$.

We remark that we have not come up with an efficient implementation of the methods in the TV case and so we do not discuss the complexity issue of those methods here.

8.6 Concluding Remarks

We have shown in section 8.2 that standard smoothers do not work for matrices of the form $\alpha I + K$ arising from convolution operators. We have proposed to use PCG as smoother and demonstrated numerically that it is effective to reduce oscillatory errors. We have also tested the matrices of the form $-\alpha\Delta + K$ and the PCG smoother works as well.

For the TV image deblurring, the situation is complicated by the highly varying coefficient and the product of convolution operators. We have proposed several multigrid preconditioners and the numerical results are satisfactory. However the implementation issue is still left open. Further investigation is needed to devise a practical and efficient multigrid preconditioner in this case.

CHAPTER 9

Linear Systems with Multiple Right-Hand Sides

9.1 Introduction

After a series of study in preconditioners, we consider another fundamental issue of iteration method, namely, linear systems with multiple right-hand sides. For direct methods such as Gaussian Elimination, it can be done efficiently relative to the LU factorization. For iterative methods, however, we may have to start anew for each right-hand side, which may be very inefficient if there are many right-hand sides. In this chapter, we analyze the effectiveness of a class of Krylov projection methods, and propose a block generalization of it.

We consider solving the following systems:

$$AX = B, \tag{9.1}$$

where A is a real symmetric positive definite matrix of order n and the matrix $B = [b^{(1)} \cdots b^{(N)}]$ contains a number of right-hand sides to be solved. For iterative methods like conjugate gradient (CG), we may have to solve the system N times. This process can be speeded up if we can find good initial guesses for the unsolved systems and find some efficient refinement process to correct the initial guesses.

If the right-hand sides are arbitrary, we nearly have no hope to do that. For example, if the right-hand sides are eigenvectors, there is hardly any information sharable among the right-hand sides when the systems are to be solved by CG. Yet in practice, for example in wave scattering problems [123], in time marching methods for PDE's [57] or in structural mechanics problems [56], the right-hand sides are not arbitrary. They are usually discrete values of some function $b(t)$. In other words,

$$b^{(j)} = b(t_j) \qquad j = 1, \ldots, N.$$

Assume that $b(t) \in C^k$ and let $x^{(j)}$ be the solution of $Ax^{(j)} = b^{(j)}$. Then $x^{(j)}$ can be shown to be discrete values of some function $x(t) \in C^k$. Hence, the $x^{(j)}$'s are close if the $b^{(j)}$'s are.

There are several approaches to solve equation (9.1) more efficiently. An obvious approach is to solve the right-hand sides one by one using the CG method with initial guesses given by extrapolation of the previous solutions. This method is effective only when the $b^{(j)}$'s are close, which may impose a severe restriction on the step size t. Another approach is to select one seed system and solve it by the Lanczos or the CG method. Then one performs a Galerkin projection of the residuals onto the Krylov subspace generated by the seed to obtain approximate solutions for the unsolved ones. The approximate solutions are then refined by Lanczos or CG again with hopefully fewer steps. This Lanczos-Galerkin scheme was discussed by Parlett [105] and Saad [114]. In addition, they modified the refinement process so that each refinement continues previous Lanczos run instead of starting a new Lanczos process. However, this method requires a lot of memory storage for the Lanczos vectors.

Based on this projection idea, various approaches have been introduced to get rid of the storage requirement if the right-hand sides are available simultaneously. Papadrakakis and Smerou [104] used the Lanczos vectors to do the projection but derived a recursive update for the solution of each right-hand side so that no storage for the Lanczos vectors is needed. Van der Vorst [133] used the residual vectors generated from CG to do the projection. Smith, Peterson and Mittra [123] gave a simplified CG version by using the direction vectors to do the projection. They also introduced a systematic implementation of the seed method and discuss some seed selection strategies. Similar ideas were also discussed by Smith [122] and Joly [81]. Mathematically, all these methods are equivalent. They only differ in the way of implementation.

We may also use the block CG method proposed by O'Leary [101] and discussed by Nikishin and Yeremin [100]. Similarly, one may use the block Lanczos method [62] but we then have to face the storage problem again. Simoncini and Gallopoulos [119, 120] combined the idea of the seed method and hybrid techniques to solve nonsymmetric linear systems.

Among all these methods, we observed that the class of Galerkin projection methods are very efficient which leads us to a more detail analysis of the single seed method. There are several practical advantages. First, we observe a super-convergence behavior of the CG process of the seed system when compared with the usual CG process. Another advantage is that if the right-hand sides are close, it usually only takes very few number of restarts to solve all the systems. Moreover, it requires no storage for the Lanczos or the direction vectors doing the projection. These features, especially the first two, make this method very effective. In this chapter, we analyze these two properties and give analytical explanations for them. Furthermore, we combine the advantages of the seed method and the block CG method and propose a block generalization. Instead of selecting a single seed, we select several systems as seed. This block seed method enjoys the fast convergence of the block CG while preserving the basic properties of the single seed method.

In Section 9.2 and 9.3, we present and analyze the single seed algorithm. In Section 9.4 and 9.5, we present and analyze the block seed algorithm. Numerical experiments are given in Section 9.6. Finally, conclusions are made in Section 9.7.

In this chapter, we adopt the following notations. The eigenvalues and the normalized eigenvectors of A are denoted by λ_i and z_i respectively, and $0 < \lambda_1 \leq$

$\lambda_2 \leq \cdots \leq \lambda_n$. The acute angle between vectors a and b is denoted by $\angle(a, b)$, and similarly the acute angle between a vector a and a vector subspace V is denoted by $\angle(a, V)$. We use $O(h^s)$ to denote a scalar, a vector or a matrix whose norm is of order $O(h^s)$ for convenience.

9.2 Single Seed Method

The idea of the single seed method, proposed by Smith, Peterson and Mittra [123], is that for each restart, a seed system is selected from the unsolved ones and is then solved by the CG method. Meanwhile, an approximate solution for the non-seed systems is obtained from the space of direction vectors p_i such that the residual is minimized in the A-norm in the direction p_i. In other words, the approximate solution is obtained by Galerkin projection of the residuals onto the Krylov subspace generated by the seed system. After the seed system is solved to desired accuracy, a new seed system is selected and the whole procedure is repeated; see Algorithm 9.1.

Algorithm 9.1: Single Seed Method

```
for $k$=0, 1, 2 ... until all the systems are solved
  Select the $k+1$th system as seed
  for $i$=0, 1, 2, ... $m_{k+1}$                 % CG iteration
    for $j=k+1, k+2, k+3, \ldots, N$     % each remaining unsolved RHS
      if $j=k+1$ then perform usual CG steps
          $\delta_i^{k,k} = (r_i^{k,k})^T r_i^{k,k} / (r_{i-1}^{k,k})^T r_{i-1}^{k,k}$
          $p_i^{k,k} = r_i^{k,k} + \delta_i^{k,k} p_{i-1}^{k,k}$
          $\sigma_i^{k,k} = (r_i^{k,k})^T r_i^{k,k} / (p_i^{k,k})^T A p_i^{k,k}$
          $x_{i+1}^{k,k} = x_i^{k,k} + \sigma_i^{k,k} p_i^{k,k}$
          $r_{i+1}^{k,k} = r_i^{k,k} - \sigma_i^{k,k} A p_i^{k,k}$
      else perform Galerkin projection
          $\eta_i^{k,j} = (p_i^{k,k})^T r_i^{k,j} / (p_i^{k,k})^T A p_i^{k,k}$
          $x_{i+1}^{k,j} = x_i^{k,j} + \eta_i^{k,j} p_i^{k,k}$
          $r_{i+1}^{k,j} = r_i^{k,j} - \eta_i^{k,j} A p_i^{k,k}$
      end if
    end for
  end for
end for
```

Here, we assume that the $k+1$st system is the seed for the kth restart. We use the first superscript to denote the kth restart, the second superscript to denote the jth system and the subscript to denote the ith step of CG. We also assume that m_k steps are used to solve the kth system within a given tolerance.

What is so special about this method is the interesting phenomena shown in Figure 9.1. We apply the above algorithm to solve $Ax = b^{(1)}$ and $Ax = b^{(2)}$, where A=diag(1,...,100) and $b^{(1)}$, $b^{(2)}$ are random vectors. Suppose the first system is solved and an approximate solution for the second system is computed by the Galerkin projection. The result of using this projected solution as an initial guess and that of using a random initial guess having the same residual norm is shown in Figure 9.1(a). We see that the former converged much faster than the latter. This is the super-convergence phenomenon that we referred to earlier. The reason is that the projection process kills off the extreme eigenvector components of the initial error. More precisely, we know that the Ritz values approach the extreme eigenvalues rapidly in a few steps of the Lanczos process [106]. Usually, the Krylov subspace generated by the first few steps also contains the extreme eigenvectors well. As a result, after the Galerkin projection, the effective spread of the spectrum of A is reduced, which in turn increases the rate of convergence. We prove this claim in Section 9.3.2.

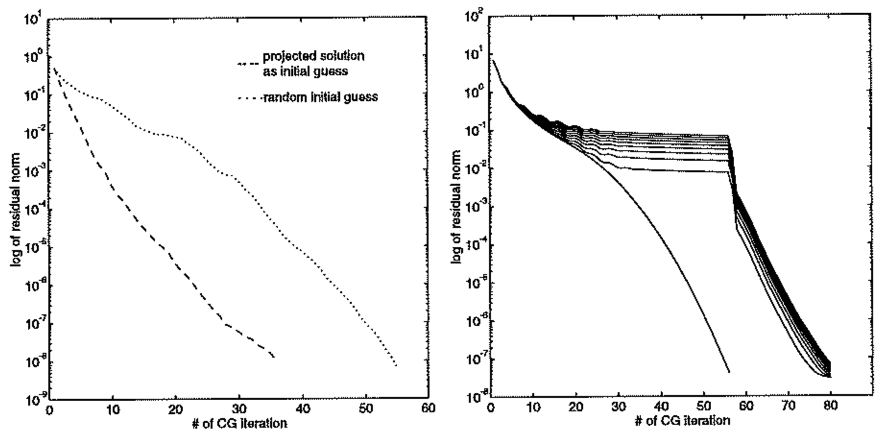


Figure 9.1: (a) The convergence behaviors of the CG process with projected solution as initial guess and with random vector as initial guess. (b) The convergence behaviors of all the systems when the right-hand sides are cyclic rotation of its elements.

Another special phenomenon is the small number of restarts needed to solve all the systems. Consider the linear systems with A as before and $b_i^{(j)}(t) = \sin(t + (i+j-2)\Delta t)$, $i = 1, \ldots, 100$; $j = 1, \ldots, 10$, where $\Delta t = 2\pi/100$. In other words, $b^{(j)}$ is obtained by shifting the components of $b^{(j-1)}$ by one position and the first component is replaced by the last one. The result is shown in Figure 9.1(b). The single seed method only needs one restart to solve all the systems. The explanation is that if B has rank k, then the single seed method only needs about $k-1$ restarts to solve all the systems. One can easily prove that the right-hand sides B generated by cyclic rotation of the components is only rank 2. We prove this claim in Section

9.3.3.

9.3 Analysis of the Single Seed Method

In this section, we analyze the two properties of the single seed method described in the previous section. Before we go on to our analysis, we state some facts about the Lanczos algorithm which will be used in the following analysis. Details of which can be found in [63].

9.3.1 Lanczos Connection

Suppose we solve $Ax = b$ by the Lanczos algorithm. Let the columns of $V_i = [v_1 \cdots v_i]$ be the orthonormal Lanczos vectors of the i-dim Krylov subspace, K_i, generated by i steps of the algorithm. Then we have the following well-known recurrence:

$$AV_m = V_m T_m + \beta_{m+1} v_{m+1} e_m^T, \qquad m = 1, \ldots, n-1 \tag{9.2}$$

where $T_m = V_m^T A V_m$ is a tridiagonal matrix, e_m is the mth column of the identity matrix and β_{m+1} is a scalar. Moreover, the solution at step m is given by

$$x_m = x_0 + V_m T_m^{-1} V_m^T r_0,$$

where x_0 is the initial guess and $r_0 = b - Ax_0$. Suppose we have another system: $A\tilde{x} = \tilde{b}$. Then the solution obtained from projection onto K_m is given by

$$\tilde{x}_m = \tilde{x}_0 + V_m T_m^{-1} V_m^T \tilde{r}_0, \tag{9.3}$$

where $\tilde{x}_0$ is the initial guess and $\tilde{r}_0 = \tilde{b} - A\tilde{x}_0$. See [114] for proof. The following is a useful lemma for our later analysis.

Lemma 9.1 *The projected solution of the non-seed systems given by the formula in the single seed algorithm is*

$$x_i^{k,j} = x_0^{k,j} + V_i^k (T_i^k)^{-1} (V_i^k)^T r_0^{k,j},$$

where V_i^k is the Lanczos vectors generated by i steps of the Lanczos algorithm if the kth system were solved by the Lanczos algorithm instead.

Proof. Smith, Peterson and Mittra [123] proved that the formula given in the algorithm computes the projected solution in the subspace generated by the direction vectors $\{p_i^{k,k}\}$. But this subspace is exactly the subspace spanned by the columns of V_i^k [63]. So, the result follows immediately from the formula (9.3). □

In the following sections, the superscript k for V_i^k and T_i^k is assumed to be the restart number and so it is dropped.

9.3.2 Rate of Convergence of the Seed System

Now, we prove the super-convergence of the single seed method. More precisely, we prove that if the previous Krylov subspace contains the extreme eigenvectors well, then the initial convergence rate of the seed is increased in a way as if the extreme ends of the spectrum of A is cut off. The technique of the proof is similar to that by van der Sluis and van der Vorst [132], in which they estimated the rate of convergence of the CG process. We compare the error reduction after i steps of the CG process with the error reduction of another CG process starting with an initial error vector obtained from the error vector of the original process by deleting the first l eigenvector components.

Remark: Saad also discussed the convergence of the seed system but from a different point of view; see [114].

Lemma 9.2 *Consider only the first two systems: $Ax^{(1)} = b^{(1)}$ and $Ax^{(2)} = b^{(2)}$. Suppose the first system is solved in m_1 CG steps. Let $x_0^{1,2}$ be the solution obtained by the projection of $b^{(2)}$ on K_{m_1} and let $\bar{x}_0^{1,2}$ be such that $x^{(2)} - \bar{x}_0^{1,2}$ is the projection of $x^{(2)} - x_0^{1,2}$ on span$\{z_k : k \in I\}^{\perp}$ where z_k is the unit eigenvector corresponding to the eigenvalue λ_k and I=$\{1,\ldots,l\}$. Let $\bar{x}_i^{1,2}$ be the ith iterate of the CG process for $Ax^{(2)} = b^{(2)}$ with $\bar{x}_0^{1,2}$ as initial guess. Then, for any i, we have*

$$\|x^{(2)} - x_i^{1,2}\|_A^2 \quad \leq \quad \|x^{(2)} - \bar{x}_i^{1,2}\|_A^2 + \delta_1, \tag{9.4}$$

where

$$\delta_1 = \|P_{m_1}^{\perp} x^{(2)}\|^2 \sum_{k \in I} \lambda_k \bar{p}_i^2(\lambda_k) \sin^2 \angle(z_k, K_{m_1}),$$

$P_{m_1}^{\perp} \equiv I - V_{m_1} T_{m_1}^{-1} V_{m_1}^T A$, is the A-orthogonal projection onto $K_{m_1}^{\perp}$, $\bar{p}_i$ is a polynomial of degree at most i and constant term 1.

Proof. Let $x^{(2)}$ solve the second system $Ax^{(2)} = b^{(2)}$ and $x_0^{1,2} = x_{m_1}^{0,1}$ be the $m_1 th$ CG iterate. Then the error of $x_0^{1,2}$ is given by

$$x^{(2)} - x_0^{1,2} = (I - V_{m_1} T_{m_1}^{-1} V_{m_1}^T A) x^{(2)} = P_{m_1}^{\perp} x^{(2)}. \tag{9.5}$$

Let the eigen-decomposition of $x^{(2)} - x_0^{1,2}$ be

$$x^{(2)} - x_0^{1,2} = \sum_{k=1}^{n} \phi_k z_k. \tag{9.6}$$

By the definition of $\bar{x}_0^{1,2}$, $x^{(2)} - \bar{x}_0^{1,2} = \sum_{k=l+1}^{n} \phi_k z_k$. It is well-known [63] that there exists a polynomial $\bar{p}_i(t)$ of degree at most i and constant term 1 such that

$$x^{(2)} - \bar{x}_i^{1,2} = \bar{p}_i(A)(x^{(2)} - \bar{x}_0^{1,2}) = \sum_{k=l+1}^{n} \bar{p}_i(\lambda_k) \phi_k z_k.$$

This implies that $\|x^{(2)} - \bar{x}_i^{1,2}\|_A^2 = \sum_{k=l+1}^{n} \bar{p}_i^2(\lambda_k)\phi_k^2\lambda_k$. Since $\bar{p}_i(t)$ is a polynomial of degree at most i, by the minimization property of CG, we have

$$\begin{aligned} \|x^{(2)} - x_i^{1,2}\|_A^2 &= \min_{p_i} \|p_i(A)(x^{(2)} - x_0^{1,2})\|_A^2 \\ &\leq \|\bar{p}_i(A)(x^{(2)} - x_0^{1,2})\|_A^2 \\ &= \sum_{k=1}^{n} \bar{p}_k^2(\lambda_k)\phi_k^2\lambda_k \\ &= \sum_{k=l+1}^{n} \bar{p}_i^2(\lambda_k)\phi_k^2\lambda_k + \sum_{k\in I} \bar{p}_i^2(\lambda_k)\phi_k^2\lambda_k \\ &= \|x^{(2)} - \bar{x}_i^{1,2}\|_A^2 + \sum_{k\in I} \bar{p}_i^2(\lambda_k)\phi_k^2\lambda_k. \end{aligned} \tag{9.7}$$

From (9.5) and (9.6), we can calculate ϕ_k to be

$$|\phi_k| = |(P_{m_1}^{\perp} x^{(2)}) \cdot z_k| \leq \|P_{m_1}^{\perp} x^{(2)}\| |\sin \angle(z_k, K_{m_1})|.$$

Substituting into (9.7), we obtain inequality (9.4). □

Lemma 9.2 implies that if δ_1 is small, the convergence rate will be increased in a way as if the first l eigenvalues of A have been removed. Since the Lanczos process captures the extreme eigenspaces, the Galerkin projection will kill off the extreme eigenvector components. Hence, $\angle(z_k, K_{m_1})$ is small for k close to 1. The estimates given by Parlett [106] and Saad [115] can be applied to bound $\sin \angle(z_k, K_{m_1})$.

Lemma 9.3 *Let $\theta_k = \angle(b^{(1)}, z_k)$ and $\tau_k = \frac{\lambda_k - \lambda_{k+1}}{\lambda_{k+1} - \lambda_n}$; k=1,...,l. Then*

$$\sin \angle(z_k, K_{m_1}) \leq \omega_k \tan \theta_k, \tag{9.8}$$

where

$$\omega_k = \frac{1}{T_{m_1-k}(1 + 2\tau_k)} \prod_{p=1}^{k-1} \frac{\lambda_p - \lambda_n}{\lambda_p - \lambda_k} \qquad k=1,\ldots,l,$$

and $T_q(x)$ is the Chebyshev polynomial of degree q.

Proof. Since

$$\sin \angle(z_k, K_{m_1}) \leq \tan \angle(z_k, K_{m_1}),$$

the result follows from Theorem 12-4-1 in [106] and Theorem 6.3 in [115]. □

First, we observe that θ_k is fixed. Thus, Lemma 9.3 implies that $\sin \angle(z_k, K_{m_1})$ is bounded by a constant times ω_k which decays exponentially with m_1. Using these two lemmas, we show that the bound for the convergence rate is the classical CG bound with a reduced condition number, plus a small perturbation term.

Theorem 9.1 *The bound for the A-norm of the error vector after i steps of the CG process is given by*

$$\|x^{(2)} - x_i^{1,2}\|_A^2 \;\leq\; 4\|x^{(2)} - \bar{x}_0^{1,2}\|_A^2 \left(\frac{\sqrt{\kappa_R}-1}{\sqrt{\kappa_R}+1}\right)^{2i} + \delta_2,$$

where

$$\delta_2 \;=\; \|P_{m_1}^{\perp} x^{(2)}\|^2 \sum_{k\in I} \lambda_k \omega_k^2 \tan^2\theta_k,$$

and the bound for δ_2 is independent of the number of iteration i. Here, $\kappa_R = \frac{\lambda_n}{\lambda_{l+1}}$ is the reduced condition number.

Proof. From Lemma 9.2, we have the following inequality:

$$\begin{aligned} \|x^{(2)} - x_i^{1,2}\|_A^2 &\leq \|x^{(2)} - \bar{x}_i^{1,2}\|_A^2 + \delta_1, \\ \delta_1 &= \|P_{m_1}^{\perp} x^{(2)}\|^2 \sum_{k\in I} \lambda_k \bar{p}_i^2(\lambda_k) \sin^2\angle(z_k, K_{m_1}). \end{aligned} \tag{9.9}$$

The term $\|x^{(2)} - \bar{x}_i^{1,2}\|_A^2$ can be bounded by the classical CG error estimate,

$$\|x^{(2)} - \bar{x}_i^{1,2}\|_A^2 \leq 4\|x^{(2)} - \bar{x}_0^{1,2}\|_A^2 \left(\frac{\sqrt{\kappa_R}-1}{\sqrt{\kappa_R}+1}\right)^{2i}. \tag{9.10}$$

Now we analyze the δ_1 term. By Lemma 9.3, we can bound δ_1 by,

$$\delta_1 \leq \|P_{m_1}^{\perp} x^{(2)}\|^2 \max_{k\in I} \bar{p}_i^2(\lambda_k) \sum_{k\in I} \lambda_k \omega_k^2 \tan^2\theta_k. \tag{9.11}$$

Recall that $\bar{p}_i$ is the optimal CG polynomial when solving $Ax^{(2)} = b^{(2)}$ using $\bar{x}_0^{1,2}$ as an initial guess (cf. proof of Lemma 9.2). It can be shown that $\bar{p}_i(t)$ has the following form [19, property 2.8]:

$$\bar{p}_i(t) = \frac{(\bar{\theta}_1^{(i)} - t)\cdots(\bar{\theta}_i^{(i)} - t)}{\bar{\theta}_1^{(i)}\cdots\bar{\theta}_i^{(i)}},$$

where $\bar{\theta}_1^{(i)} < \bar{\theta}_2^{(i)} < \cdots < \bar{\theta}_i^{(i)}$ are the Ritz values of A with respect to K_i, the Krylov subspace generated by $\bar{r}_0^{1,2} = b - A\bar{x}_0^{1,2}$. By a property of Ritz values, $\lambda_{l+1} \leq \bar{\theta}_j^{(i)} \leq \lambda_n$, $j = 1, \ldots, i$. From the formula of $\bar{p}_i(t)$ above, for $0 < t < \lambda_{l+1}$, we have $0 < \bar{p}_i(t) < 1$. Thus $\max_{k\in I} \bar{p}_i^2(\lambda_k) < 1$. The desired inequality follows from substituting (9.10) and (9.11) into (9.9). □

Since the Krylov subspace of the previous seed system contains the extreme eigenvectors well, ω_k and hence δ_2 are small. On the other hand, the quantity

$||P_{m_1}^{\perp} x^{(2)}||$ is bounded by $||x^{(2)}||$. (If we make additional assumptions on $b^{(1)}$ and $b^{(2)}$, we can estimate the bound for $||P_{m_1}^{\perp} x^{(2)}||$ more precisely; see Section 9.3.4). Thus the whole perturbation term is negligible which implies the initial rate of convergence can be as fast as if the spectrum of A is reduced. This explains the super-convergence behavior shown in Figure 9.1(a).

Remark: For the proof of the lemmas, we have made no assumptions about the right-hand sides. Thus, the single seed method may be useful to solve general multiple right-hand sides.

9.3.3 Convergence Analysis of the Non-seed System

We come to our second analysis. We prove that the systems are almost solved after $k-1$ restarts of the CG process if the rank of B is k. This result is the central idea of the following analysis that explains why the single seed method only needs a few restarts to solve all the systems if the right-hand sides are close to each other.

Theorem 9.2 *Suppose $B = [b^{(1)} \cdots b^{(N)}]$ and rank$(B) = k < N$. Then there exists $\alpha > 0$, independent of the iteration numbers m_p, such that the residual of the non-seed systems after $k-1$ restarts of the single seed method satisfies*

$$||\tilde{r}_0^{k,j}|| \le \alpha \sum_{p=1}^{k} |\beta_{m_p+1}|, \qquad j = k+1, \ldots, N, \tag{9.12}$$

where β_{m_p+1} comes from the Lanczos recurrence formula (9.2) if the pth seed is thought of solved by the Lanczos algorithm, $p = 1, \ldots, k$.

Proof. Let $B = \sum_{i=1}^{k} \sigma_i u_i w_i^T$ be the outer product form of a singular value decomposition of B, where σ_i, u_i and w_i are the singular values, left and right singular vectors of B respectively. So the jth system, $b^{(j)}$, can be written as:

$$b^{(j)} = \sum_{i=1}^{k} \sigma_i (w_i^T e_j) u_i, \tag{9.13}$$

where e_j is the jth column of the identity matrix I. Let

$$\begin{aligned} \tilde{r}_0^{0,j} &= b^{(j)}, \qquad \tilde{x}_0^{0,j} = 0, \qquad && j = 1, 2, \ldots, N, \\ \zeta_i^{0,j} &= \sigma_i w_i^T e_j, \qquad u_i^0 = u_i, \qquad && i = 1, 2, \ldots, k. \end{aligned} \tag{9.14}$$

Notice that

$$|\zeta_i^{0,j}| \le \sigma_i, \qquad j = 1, \ldots, N. \tag{9.15}$$

We take the first system as seed and suppose that it is solved in m_1 CG steps. By Lemma 9.1, the approximate solution of the non-seed systems obtained from the Galerkin projection is

$$\tilde{x}_0^{1,j} = \tilde{x}_0^{0,j} + V_{m_1} T_{m_1}^{-1} V_{m_1}^T \tilde{r}_0^{0,j} \qquad j = 1, \ldots, N,$$

and

$$\begin{aligned} \tilde{r}_0^{1,j} &= b^{(j)} - A\tilde{x}_0^{1,j} \\ &= (I - V_{m_1} V_{m_1}^T)\tilde{r}_0^{0,j} - \beta_{m_1+1} v_{m_1+1} e_{m_1}^T T_{m_1}^{-1} V_{m_1}^T \tilde{r}_0^{0,j}. \end{aligned} \tag{9.16}$$

We want to express $\tilde{r}_0^{0,j}$ in terms of $\tilde{r}_0^{0,1}$ since $(I - V_{m_1} V_{m_1}^T)\tilde{r}_0^{0,1} = 0$. Without loss of generality, we may assume that

$$|\zeta_1^{0,1}| = \max_{1 \le i \le k} |\zeta_i^{0,1}|. \tag{9.17}$$

From (9.13) and (9.15), we write $\tilde{r}_0^{0,j}$ as

$$\tilde{r}_0^{0,j} = \zeta_1^{0,j} u_1^0 + \sum_{i=2}^{k} \zeta_i^{0,j} u_i^0.$$

In particular, for $j = 1$, we have

$$\tilde{r}_0^{0,1} = \zeta_1^{0,1} u_1^0 + \sum_{i=2}^{k} \zeta_i^{0,1} u_i^0.$$

Now we rearrange the terms of $\tilde{r}_0^{0,j}$ and rewrite it in terms of $\tilde{r}_0^{0,1}$:

$$\tilde{r}_0^{0,j} = \zeta_1^{0,j} (\zeta_1^{0,1})^{-1} \tilde{r}_0^{0,1} + \sum_{i=2}^{k} [\zeta_i^{0,j} - \zeta_1^{0,j} (\zeta_1^{0,1})^{-1} \zeta_i^{0,1}] u_i^0. \tag{9.18}$$

Let

$$\zeta_i^{1,j} = \zeta_i^{0,j} - \zeta_1^{0,j} (\zeta_1^{0,1})^{-1} \zeta_i^{0,1} \qquad i = 2, \ldots, k.$$

Then formula (9.18) becomes

$$\tilde{r}_0^{0,j} = \zeta_1^{0,j} (\zeta_1^{0,1})^{-1} \tilde{r}_0^{0,1} + \sum_{i=2}^{k} \zeta_i^{1,j} u_i^0. \tag{9.19}$$

Moreover, by (9.15) and (9.17), we have

$$|\zeta_i^{1,j}| \le |\zeta_i^{0,j}| + |\zeta_i^{0,j}| |\zeta_i^{0,1} / \zeta_1^{0,1}| \le 2\sigma_i.$$

Combining (9.16) and (9.19), we have

$$\tilde{r}_0^{1,j} = (I - V_{m_1}V_{m_1}^T)\sum_{i=2}^{k}\zeta_i^{1,j}u_i^0 + \gamma_1^{1,j}v_{m_1+1},$$

where

$$\gamma_1^{1,j} = -\beta_{m_1+1}e_{m_1}^T T_{m_1}^{-1}V_{m_1}^T \tilde{r}_0^{0,j} = \alpha_1\beta_{m_1+1},$$

and

$$\alpha_1 \equiv -e_{m_1}^T T_{m_1}^{-1} V_{m_1}^T \tilde{r}_0^{0,j}.$$

We now show that α_1 is independent of m_1. Obviously, $e_{m_1}^T$ is bounded by 1. Besides, it is well-known [63] from the interlacing property of the eigenvalues of the consecutive $T_k's$ that the spectrums of $T_k's$ are bounded by the spectrum of A. Thus, $\|T_{m_1}^{-1}\| \leq \|A^{-1}\|$. Next, since the columns of V_{m_1} are orthogonal, we have $\|V_{m_1}\| \leq 1$. Finally, it is clear that

$$\|\tilde{r}_0^{0,j}\| \leq \|B\|.$$

Thus α_1 is bounded independent of m_1. In other words, it does not grow unboundedly as the iteration goes on. Let

$$u_i^1 = (I - V_{m_1}V_{m_1}^T)u_i^0.$$

Then we further simplify the notation and write $\tilde{r}_0^{1,j}$ as

$$\tilde{r}_0^{1,j} = \sum_{i=2}^{k}\zeta_i^{1,j}u_i^1 + \gamma_1^{1,j}v_{m_1+1}. \tag{9.20}$$

Hence, after one solve, the rank of the remaining columns of B is effectively reduced by one if $\gamma_1^{1,j}$ is small. We proceed in the same manner for the other seeds. Inductively, after $l-1$ restarts, the residual for the non-seed system is

$$\tilde{r}_0^{l,j} = \sum_{i=l+1}^{k}\zeta_i^{l,j}u_i^l + \sum_{p=1}^{l}\gamma_p^{l,j}\prod_{q=p+1}^{l}(I - V_{m_q}V_{m_q}^T)v_{m_p+1}, \tag{9.21}$$

where

$$\begin{aligned}
\zeta_i^{l,j} &= \zeta_i^{l-1,j} - \zeta_l^{l-1,j}(\zeta_l^{l-1,l})^{-1}\zeta_i^{l-1,l}, \\
u_i^l &= (I - V_{m_l}V_{m_l}^T)u_i^{l-1}, \\
\gamma_p^{l,j} &= \gamma_p^{l-1,j} - \zeta_l^{l-1,j}(\zeta_l^{l-1,l})^{-1}\gamma_l^{l-1,j}, \qquad p = 1, \ldots, l-1, \\
\gamma_l^{l,j} &= -\beta_{m_l+1}e_{m_l}^T T_{m_l}^{-1}V_{m_l}^T\tilde{r}_0^{l-1,j}.
\end{aligned}$$

Notice that $|\zeta_i^{l,j}| \leq 2^l \sigma_i$ since $|\zeta_i^{l-1,j}| \leq 2^{l-1}\sigma_i$, which can be derived inductively. In particular, when $l = k$, we have from (9.21) that the first term vanishes and

$$\tilde{r}_0^{k,j} = \sum_{p=1}^{k} \gamma_p^{k,j} \prod_{q=p+1}^{k} (I - V_{m_q} V_{m_q}^T) v_{m_p+1}, \tag{9.22}$$

where $\gamma_p^{k,j}$ is a constant multiple of β_{m_p+1}. As explained before, these constants are independent of the m_p's. Thus we can find $\alpha > 0$ such that

$$|\gamma_p^{k,j}| \leq \frac{\alpha}{k} |\beta_{m_p+1}|. \tag{9.23}$$

Moreover, for any q, $I - V_{m_q} V_{m_q}^T$ are projections and v_{m_p+1} is of unit length. Hence

$$\| \prod_{q=p+1}^{k} (I - V_{m_q} V_{m_q}^T) v_{m_p+1} \| \leq 1. \tag{9.24}$$

Finally, we obtain (9.12) from (9.23), (9.24) and (9.22). □

If B has near rank deficiency, then we have similar result but with an extra perturbation term.

Corollary 9.1 *Suppose the singular values of B are such that*

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > \epsilon \geq \sigma_{k+1} \geq \cdots \geq \sigma_N \geq 0.$$

Then there exists $\alpha' > 0$, independent of the m_p's, so that

$$\|\tilde{r}_0^{k,j}\| \ \leq \ \alpha' \, (\sum_{p=1}^{k} |\beta_{m_p+1}| \, + \, \epsilon).$$

Proof. By (9.21), with $k = N$, $l = k$, we have

$$\tilde{r}_0^{k,j} = \sum_{i=k+1}^{N} \zeta_i^{k,j} u_i^k + \sum_{p=1}^{k} \gamma_p^{k,j} \prod_{q=p+1}^{k} (I - V_{m_q} V_{m_q}^T) v_{m_p+1}. \tag{9.25}$$

Since $|\zeta_i^{k,j}| \leq 2^k \sigma_i$ and $\sigma_i \leq \epsilon$, $i = k+1, \ldots, N$, by Theorem (9.2), we get

$$\begin{aligned} \|\tilde{r}_0^{k,j}\| \ &\leq \ \sum_{i=k+1}^{N} |\zeta_i^{k,j}| \|u_i^k\| + \alpha \sum_{p=1}^{k} |\beta_{m_p+1}| \\ &\leq \ (N-k) 2^k \epsilon + \alpha \sum_{p=1}^{k} |\beta_{m_p+1}|. \end{aligned}$$

The result follows if we let $\alpha' = \max(\alpha, (N-k)2^k)$. $\square$

Remarks:

1. In practice, a stopping criterion of the following form is usually used:

$$\|r^{p,p}_{m_p}\| \le tol\ \|b^{(p)}\|.$$

It is well-known [63] that:

$$\|r^{p,p}_{m_p}\| = |\beta_{m_p+1}||e^T_{m_p} T^{-1}_{m_p} e_1|\|b^{(p)}\|.$$

Hence $|\beta_{m_p+1}|$ can be bounded as:

$$|\beta_{m_p+1}||e^T_{m_p} T^{-1}_{m_p} e_1| \le tol.$$

If $|e^T_{m_p} T^{-1}_{m_p} e_1|$ is not small, then $|\beta_{m_p+1}|$ is the same order of *tol*. Therefore, by Theorem 9.2, the systems are solved to the order of accuracy of *tol* in only $k-1$ restarts if rank(B)=k.

2. Consider the special case that $B = [b^{(1)} \cdots b^{(N)}]$, $b^{(j)} = b(t_j)$ and

$$b(t) = c_1(t)a_1 + \cdots + c_k(t)a_k,$$

where $c_i(t)$ are some bounded functions, not necessary continuous and a_i are some linearly independent vectors. Then Theorem (9.2) implies that B is almost solved after $k-1$ restarts. In particular, if $b(t)$ is a polynomial of degree $k-1$, then B can be solved in about $k-1$ restarts.

3. In general, if $b(t)$ is not a polynomial, Corollary (9.1) ensures that we can still have a similar result for $b(t)$ a continuous function. We rewrite this remark into the following theorem.

Theorem 9.3 *If $b(t)$ is a continuous function on $[t_1, t_N]$, then given $\epsilon > 0$, there exist integers $k \le N$ and α' such that*

$$\|\tilde{r}^{k,j}_0\| \le \alpha' (\sum_{p=1}^{k} |\beta_{m_p+1}| + \epsilon).$$

Proof. Since $b(t)$ is continuous, by the Weierstrass approximation theorem [118], there exists a polynomial $f(t)$ of degree $(k-1)$ such that

$$\|b(t) - f(t)\|_2 < \epsilon/\sqrt{N} \qquad \forall t \in [t_1, t_N]. \tag{9.26}$$

Notice that we can always find a polynomial of degree $(N-1)$ such that (9.26) is true since B has only N columns. So, we always have $k \le N$. We now estimate the size of σ_{k+1}.

Let

$$\tilde{B} = [f(t_1) \ \cdots \ f(t_N)].$$

Then rank($\tilde{B}$)=k. By the minimization property of σ_{k+1} (see [63]), we have

$$\sigma_{k+1} \leq \|B - \tilde{B}\|_2.$$

But by (9.26), we have $\|B - \tilde{B}\|_2 \leq \|B - \tilde{B}\|_F \leq \epsilon$. The result follows from Corollary (9.1). □

This theorem points out an important property. If $b(t)$ is continuous and the $b^{(j)}$'s are close to each other, then rank$(B) < N$. In other words, we can find a low order polynomial to approximate $b(t)$. Theorem 9.3 says that the single seed method will automatically exploit this fact. One may suggest using a QR factorization of B and then solving the linearly independent right-hand sides. But this is only effective when the rank is much smaller than N. Otherwise, it is costly to do the factorization and reconstruction. Furthermore, in practice, it is hard to tell the numerical rank of B. The usefulness of the single seed method is that we do not need to determine the rank by ourselves and it will automatically take care of it.

The number k somehow reveals the effective rank of B. If the $b^{(j)}$'s are close to each other, k is usually much smaller than N. Hence, the single seed method only needs a few restarts to solve all the systems.

9.3.4 Error Analysis of the Initial Guess

In this section, we estimate the error of the initial guess given by the Galerkin projection, assuming that $b(t)$ is smooth. Saad [114] also estimated the error of the projected solution but in a more general context. Since the right-hand sides of the non-seed systems differ from that of the seed system by $O(h)$, we can show that the initial guess given by the projection also has $O(h)$ error.

Theorem 9.4 *Assume that $b = b(t)$ and $b^{(j)} = b(t_1 + (j-1)h)$. Suppose we have applied m_1 CG steps to the first system: $Ax^{(1)} = b^{(1)}$ and obtain K_{m_1}. Let $\tilde{x}_0^{1,j}$ be the projected solution of $Ax^{(j)} = b^{(j)}$ onto K_{m_1} and $x_{m_1}^{0,j}$ be the solution given by applying m_1 CG steps to the jth system with initial guess 0. Then*

$$\|\tilde{x}_0^{1,j} - x_{m_1}^{0,j}\| = O(h).$$

Proof. Since $b(t)$ is smooth, the approximation of $b^{(j)}$ by $b^{(1)}$ satisfies

$$b^{(j)} = b^{(1)} + O(h).$$

By Lemma 9.1, the projected solution of $b^{(j)}$ with 0 initial guess is

$$\tilde{x}_0^{0,j} = V_{m_1} T_{m_1}^{-1} V_{m_1}^T b^{(j)} = x_{m_1}^{0,1} + O(h)$$

Since $x_{m_1}^{0,1} - x_{m_1}^{0,j} = O(h)$, the theorem is proved. □

This theorem shows that $\|P_{m_1}^{\perp} x^{(2)}\|$ in Theorem 9.1 is only $O(h)$ if the right-hand sides have smooth dependence.

9.4 Block Seed Method

In this section, we propose a block generalization of the single seed method. In the previous section, we notice that the initial guess given by the Galerkin projection requires h being small to have a good approximation since the approximation is only of first order. Practically, however, we may need to take large steps. A natural solution to this problem is to select more than one system as seed so that the subspace generated by the seed is larger and the initial guess obtained from the projection onto this subspace is hopefully better. Moreover, if we use the block CG method to solve the seed systems, the convergence rate is also improved. These statements will be made more precise in the following sections.

The block seed algorithm is exactly the same as the single seed method except the seed is a block of s systems. The seed is solved by the block CG method [101] while approximate solution for the non-seed system is obtained by the Galerkin projection onto the the subspace generated by the seed. The algorithm of the block seed method is as follows.

A drawback of the block method is that it may break down when the matrices $(R_{i-1}^{k,k})^T R_{i-1}^{k,k}$ and $(P_i^{k,k})^T A P_i^{k,k}$ become singular. On the other hand, in addition to the fast convergence inherited from the block CG, the block seed method possesses all the basic properties of the single seed method discussed in last sections. In fact, the bounds obtained are better than those for the single seed method. Analogous theorems are stated in the following sections.

From now on, we assume that the block size is s and the matrix A does not have eigenvalues with multiplicity greater than s. After m steps of the block CG, we let $\mathcal{V}_m = [U_1 \cdots U_m]$ whose columns denote The orthonormal blocks of Lanczos vectors and $\mathcal{K}_m$ denote the subspace spanned by columns of $\mathcal{V}_m$. Let $\mathcal{P}_m$ and $\mathcal{P}_m^{\perp}$ be the A-orthogonal projection onto $\mathcal{K}_m$ and $\mathcal{K}_m^{\perp}$ respectively. Lower case letters denote vectors while upper case letters denote matrices.

9.5 Analysis of the Block Seed Method

In the following sections, we give the analogous results of the single seed case. Since the techniques of the proofs are similar to the non-block case, we shall skip

```
Algorithm 9.2: Block Seed Method
for $k$=0, 1, 2 ... until all the systems are solved
   Select the $ks+1, \dots, ks+s$th systems as seed
   for i=0, 1, 2, ... $m_{k+1}$                          % block CG iteration
      for $j=k+1, k+2, k+3, \dots, \lceil N/s \rceil$     % each unsolved block of RHS
         if $j=k+1$ then perform usual block CG steps
            $\Delta_i^{k,k} = (R_i^{k,k})^T R_i^{k,k}((R_{i-1}^{k,k})^T R_{i-1}^{k,k})^{-1}$
            $P_i^{k,k} = R_i^{k,k} + \Delta_i^{k,k} P_{i-1}^{k,k}$
            $\Sigma_i^{k,k} = (R_i^{k,k})^T R_i^{k,k}((P_i^{k,k})^T A P_i^{k,k})^{-1}$
            $X_{i+1}^{k,k} = X_i^{k,k} + \Sigma_i^{k,k} P_i^{k,k}$
            $R_{i+1}^{k,k} = R_i^{k,k} - \Sigma_i^{k,k} P_i^{k,k}$
         else perform block Galerkin projection
            $H_i^{k,j} = (P_i^{k,k})^T R_i^{k,j}((P_i^{k,k})^T A P_i^{k,k})^{-1}$
            $X_{i+1}^{k,j} = X_i^{k,j} + H_i^{k,j} P_i^{k,k}$
            $R_{i+1}^{k,j} = R_i^{k,j} - H_i^{k,j} A P_i^{k,k}$
         end if
      end for
   end for
end for
```

Note: The second superscript for the matrices denotes the jth sub-block of B.

them.

9.5.1 Convergence Analysis of the Seed System

We show below that the bound for the convergence rate of the seed systems is superior to that by the single seed method since the block Lanczos method gives a better bound for the extreme eigenvalues than that by the classical Lanczos method. Yet the statement and the proof of the theorem is analogous to Theorem 9.1. Before we go on, we state the minimization property of the block CG iterate proved by O'Leary [101].

Theorem 9.5 *Let X_m be the approximate solution given by the block CG and X^* be the true solution. Then X_m minimizes $tr[(X - X^*)^T A(X - X^*)]$ over all X such that $X - X_0 \in \mathcal{K}_m$.*

Since the minimization property of the block CG algorithm is on all the systems together, we only estimate the convergence rate of the seed systems indirectly; we do not bound the error of the individual systems but by blocks instead. Same as before, we show the effect of the first l eigenvalues.

Lemma 9.4 *Consider two systems: $AX^{(1)} = B^{(1)}$ and $AX^{(2)} = B^{(2)}$. Suppose the first systems are solved to desired accuracy after m_1 steps of the block CG process. Let $X_0^{0,2}$, $\bar{X}_0^{0,2}$, $X_i^{0,2}$, $\bar{X}_i^{0,2}$ and I be defined similarly as in Lemma 9.2 and $x^{(j)}$ be the solution of the jth system of $X^{(2)}$. Then for any i, we have*

$$tr[(X^{(2)} - X_i^{1,2})^T A(X^{(2)} - X_i^{1,2})] \le tr[(X^{(2)} - \bar{X}_i^{1,2})^T A(X^{(2)} - \bar{X}_i^{1,2})] + \delta_3, \tag{9.27}$$

where

$$\delta_3 = \sum_{j=1}^{s} \|\mathcal{P}_{m_1}^{\perp} x^{(j)}\|^2 \sum_{k\in I} \lambda_k \bar{p}_j^2(\lambda_k) \sin^2 \angle(z_k, \mathcal{K}_{m_1}).$$

We can also estimate $\sin \angle(z_k, \mathcal{K}_{m_1})$ by the bounds given by Saad [113].

Lemma 9.5 *Suppose that $\mathcal{P}_1 z_j$ are linearly independent for j=k, ... ,k+s-1; $k \le l$. Let $\hat{x}_k \in \mathcal{K}_1$ be defined such that*

$$z_j^T \hat{x}_k = \delta_{kj},$$

for j=k, ... , k+s-1; $k \le l$. Let $\hat{\theta}_k = \angle(z_k, \hat{x}_k)$ and $\hat{\tau}_k = \frac{\lambda_k - \lambda_{k+s}}{\lambda_{k+s} - \lambda_n}$; k=1, ... ,l. Then

$$\sin \angle(z_k, \mathcal{K}_{m_1}) \le \hat{\omega}_k \tan \hat{\theta}_k, \tag{9.28}$$

where

$$\hat{\omega}_k = \frac{1}{T_{m-k}(1 + 2\hat{\tau}_k)} \prod_{\lambda\in\Lambda_k} \frac{\lambda - \lambda_n}{\lambda - \lambda_k}; k = 1, \ldots, l.$$

Here Λ_k is the set of the first $k-1$ distinct eigenvalues.

Remark: Since $|\lambda_k - \lambda_{k+s}| \ge |\lambda_k - \lambda_{k+1}|$, we have $\hat{\tau}_k \ge \tau_k$, which implies $\hat{\omega}_k \le \omega_k$. Moreover, the subspace $\mathcal{V}_{m_1}$ is bigger than the subspace V_{m_1}, we have $\|\mathcal{P}_{m_1}^{\perp} x^{(j)}\| \le \|P_{m_1}^{\perp} x^{(j)}\|$ (cf. Lemma 9.2, 9.3).

By these two lemmas, we may give a bound similar to Theorem 9.1 for the convergence rate of the seed systems.

Theorem 9.6 *The bound for the sum of the error vector in the A-norm after i steps of the block CG process is given by*

$$tr[(X^{(2)} - X_i^{1,2})^T A(X^{(2)} - X_i^{1,2})] \le 4 \sum_{j=1}^{s} \rho_j \left(\frac{\sqrt{\kappa_R} - 1}{\sqrt{\kappa_R} + 1}\right)^{2i} + \delta_4,$$

where

$$\delta_4 = \sum_{j=1}^{s} \|\mathcal{P}_{m_1}^{\perp} x^{(j)}\|^2 \sum_{k\in I} \lambda_k \hat{\omega}_k^2 \tan^2 \hat{\theta}_k,$$

for some constant ρ_j.

This bound looks almost the same as the bound in Theorem 9.1 except ω_k and θ_k change to $\hat{\omega}_k$ and $\hat{\theta}_k$. From the previous remark, $\hat{\omega}_k \leq \omega_k$ and $\|\mathcal{P}_{m_1}^{\perp} x^{(j)}\| \leq \|P_{m_1}^{\perp} x^{(j)}\|$. This shows that the bound for the convergence rate of the seed systems in the block case is better than that in the single seed case.

9.5.2 Convergence Analysis of the Non-seed System

We now state two other analogous theorems of the block seed method. It says that all the systems are almost solved in $\lceil k/s \rceil - 1$ restart of the block seed method if rank(B)=k.

Theorem 9.7 *Suppose we want to solve the same problem as in Theorem 9.2 by the block seed method. Then there exists $\alpha > 0$, independent of the m_p's, such that the residual of the non-seed systems after $\lceil k/s \rceil - 1$ restarts satisfies*

$$\|\tilde{r}_0^{\lceil k/s \rceil, j}\| \leq \alpha \sum_{p=1}^{\lceil k/s \rceil} \|\hat{\beta}_{m_p+1}\|, \qquad j = k+1, \ldots, N,$$

where $\hat{\beta}_{m_p+1}$ comes from the recurrence of the block Lanczos algorithm [62],

$$A\mathcal{V}_{m_p} = \mathcal{V}_{m_p}\mathcal{T}_{m_p} + U_{m_p+1}\hat{\beta}_{m_p+1}E_{m_p}^T, \qquad p = 1, \ldots, \lceil k/s \rceil.$$

Finally, if $b(t)$ is continuous, we can bound the residual in the same way as in Theorem 9.3.

Theorem 9.8 *If $b(t)$ is a continuous function in t on $[t_1, t_N]$, then given $\epsilon > 0$, there exist integers k and α' such that the residual of the non-seed systems, after $\lceil k/s \rceil - 1$ restart of the block seed method, satisfies*

$$\|\tilde{r}_0^{\lceil k/s \rceil, j}\| \leq \alpha' \Big(\sum_{p=1}^{\lceil k/s \rceil} \|\hat{\beta}_{m_p+1}\| + \epsilon \Big).$$

The results of Theorem 9.7 and 9.8 are not surprising since each solve of the block CG generates an s times bigger subspace than CG does. Thus the number of restarts should be reduced by a factor of s for the block seed method.

9.5.3 Error Analysis of the Initial Guess

We point out in Section 9.4 that the initial guess given by the single seed method is good only if the right-hand sides are really close. In fact, the initial guess is only $O(h)$ accurate to the mth iterate given by the CG process. We now prove that the initial guess given by the block seed method is $O(h^s)$ accurate to the extrapolate approximation of $x^{(j)}$ given by the solutions of the seed systems.

Theorem 9.9 *Assume $b^{(j)} = b(t_1 + (j-1)h)$ as in Theorem 9.4. Suppose we have applied m_1 block CG steps to the seed systems: $AX^{(1)} = B^{(1)}$ and obtain $\mathcal{K}_{m_1}$. Let $\tilde{x}_0^{1,j}$ be the projected solution of $Ax^{(j)} = b^{(j)}$ onto $\mathcal{K}_{m_1}$. Then $\tilde{x}_0^{1,j}$ is an sth order approximation of the extrapolation of the columns of $X_{m_1}^{0,1}$ which are obtained from applying m_1 block CG steps to the first sub-block of B.*

Proof. Since $b(t)$ is smooth, the extrapolate approximation of $b^{(j)}$ by $b^{(i)}$'s, $i = 1, \ldots, s$, is of order s. Thus there exist χ_i such that

$$b^{(j)} = \sum_{i=1}^{s} \chi_i b^{(i)} + O(h^s).$$

Then, by the block version of Lemma 9.1, the projected solution of $b^{(j)}$ with initial guess 0 is

$$\begin{aligned}
\tilde{x}_0^{1,j} &= \mathcal{V}_{m_1} \mathcal{T}_{m_1}^{-1} \mathcal{V}_{m_1}^T b^{(j)} \\
&= \sum_{i=1}^{s} \chi_i \mathcal{V}_{m_1} \mathcal{T}_{m_1}^{-1} \mathcal{V}_{m_1}^T b^{(i)} + O(h^s) \\
&= \sum_{i=1}^{s} \chi_i x_{m_1}^{(i)} + O(h^s).
\end{aligned}$$

Result follows from the fact that $\sum_{i=1}^{s} \chi_i x_{m_1}^{(i)}$ is actually the extrapolated approximation given by $X_{m_1}^{0,1}$. □

Since the initial guesses given by the block seed method have higher order of accuracy, the block seed method is useful in the case when the right-hand sides are not close to each other.

9.6 Numerical Results

In this section, we present four examples to illustrate the analysis in Sections 9.3 and 9.5, and also compare the performance of these methods applied to an acoustic scattering problem. All the experiments were performed in MATLAB with machine precision 10^{-16}. The stopping criterion is: $\|r_i^{k,j}\| < 10^{-8}\|b^{(j)}\|$.

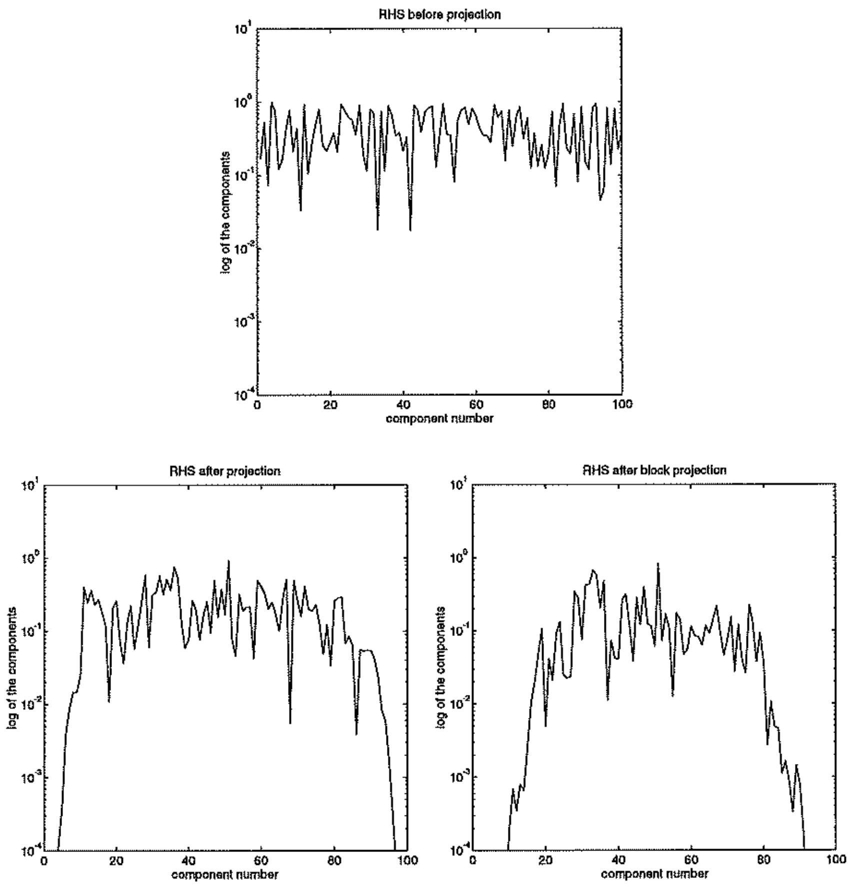


Figure 9.2: Size distribution of the components of (a) the original vector b_4, (b) b_4 after the projection, (c) b_4 after the block projection.

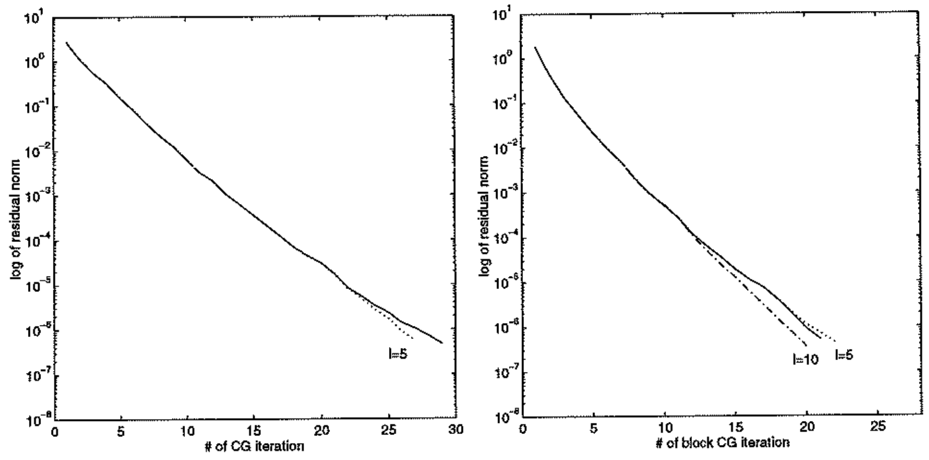


Figure 9.3: The convergence behavior of the CG/block CG process with different initial guess. (a) The single seed method, (b) The block seed method. Solid lines denote seed/block seed method. Dotted lines denote the case $l = 5$. Dashed dotted lines denote the case $l = 10$.

Example 1: In this example, we show the effects of projection to the error components and the convergence rate of the seed systems. Our test problem is:

$$AX = B,$$

where A=diag(1,2,...,100) and B is a random matrix with 4 columns.

Let b_j denotes the jth column of B. For the single seed method, we solve b_1 by CG (in 55 iterations) and project the other 3 systems to obtain an initial guess. Then each one is solved by CG again. We only consider the behavior of the last system, b_4. Before discussing the convergence behavior, we see from Figure 9.2(b) and (c) that the first number of eigenvector components of b_4 are removed by the projection. Thus the effective spectrum of A is reduced, which in turn increases the convergence rate of CG. In Figure 9.3(a), the solid line denotes the convergence of the last system using the projected solution as initial guess. The dotted line denotes the convergence of the same system using the same projected solution as initial guess but its error vector is such that the first l eigenvector components are set to zero. We choose $l = 5$ since about five eigenvector components are removed as shown in Figure 9.2(b). We see that the convergence behaviors of both initial guesses are almost identical as predicted by Theorem 9.1.

For the block seed method, we solve b_1 and b_2 by the block CG method (in 40 iterations) and project the other two systems to obtain an initial guess. Then the last two are solved by the block CG again. Again, we only consider the last system. Figure 9.2(c) shows that more eigenvector components are removed by the block projection. This implies more extreme eigenspaces are captured by one solve of the block CG than by the usual CG. Figure 9.3(b) shows that the convergence behavior of the last system is slightly better than the case l=5 and almost the same as the case l=10. This verifies that the block seed method has a better convergence bound than the single seed method.

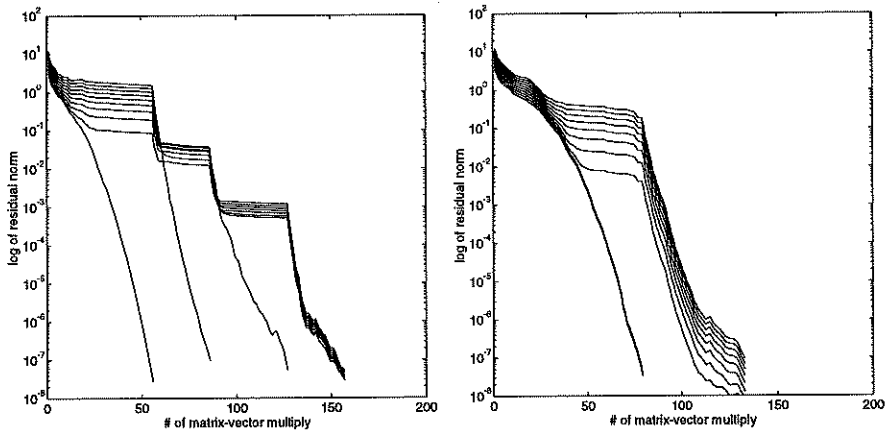


Figure 9.4: The convergence behavior of all the systems (10 right-hand sides) when solving $Ax(t) = b(t)$ where A=diag(1:100) and $b(t)$=$a_0 + ta_1 + t^2a_2 + t^3a_3$ by (a) the single seed method, (b) the block seed method.

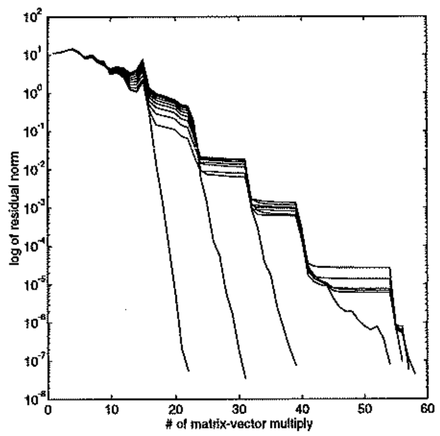
Example 2: In this example, we show the convergence rate of the non-seed systems. Our test problem is

$$Ax(t) \;=\; b(t),$$

where A=diag(1,2,...,100) and $b(t)$=$a_0 + ta_1 + t^2a_2 + t^3a_3$; a_0, a_1, a_2 and a_3 are some linearly independent vectors of unit length. Thus rank(B)=4 in this case. We choose $t_1 = 1$, $\Delta t = 0.1$ and $t_j = t_1 + (j-1)\Delta t$, j=1, ... ,10, and $b_j = b(t_j)$.

The convergence behaviors of all the systems are shown in Figure 9.4(a) and (b). From the plot, each steepest declining line denotes the convergence of a seed and also for the non-seed in the last restart. Figure 9.4(a) shows that four seeds (corresponding to three restart) are used to solve all the systems. Figure 9.4(b) shows that, in the block seed case, only two seeds (corresponding to one restart) are used to solve all the systems. These results match precisely the results of Theorem 9.2 and Theorem 9.7. Notice that we plot the residual norm against the cost (the number of matrix-vector multiply) in place of the iteration numbers as in Example 1 so that we may compare the effectiveness of these methods by cost. In this particular example, the block seed method is slightly more efficient.

Example 3: In this example, we apply the single seed method and the block seed method to solve an acoustic scattering problem. The object is circular and the wave number is arbitrarily chosen as 5. The Helmholtz equation or the reduced wave equation is solved by the double layer potential, using the trapezoidal rule to discretize the integral. We show the efficiency of both methods and illustrate the



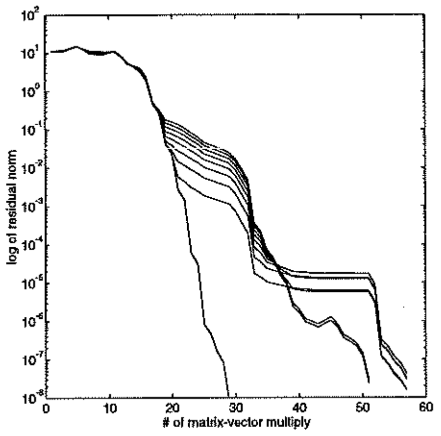


Figure 9.5: The convergence behavior of all the systems when solving $Ax(t) = b(t)$ where $b(t)$ corresponds to the boundary data of a radar scattering problem and $\Delta t = 1^\circ$ by (a) the single seed method, (b) the block seed method.

effect of Δt, the change of the incident angle t, on them. More precisely, our test problem is

$$Ax(t) = b(t),$$

where $A = \tilde{A}\tilde{A}^T$ and $\tilde{A}$ is a particular non-singular complex matrix of order 128×128 arising from the acoustic scattering problem. $b(t)$ is the right-hand side which depends on the incident angle t and corresponds to the boundary data that accompany the circle geometry. The formula for the ith component of the right-hand side is

$$b_i(t) = -\cos(5\cos(t - 2(i-1)\pi/128)) - \sqrt{-1}\sin(5\cos(t - 2(i-1)\pi/128)).$$

We solve the problem in two cases. The first case corresponds to $\Delta t = 1^\circ$ and the second case corresponds to $\Delta t = 10^\circ$. The number of right-hand sides is still 10.

The result of the first case is shown in Figure 9.5(a) and (b). In our previous analysis, we learn that the initial guess given by the block seed method is better than that by the single seed method when Δt is small. Figure 9.5(a) and (b) show that the residual of the initial guess given by the single seed method is $O(10^{-1})$ while that given by the block seed method is $O(10^{-2})$. On the other hand, since the right-hand sides are close for Δt small, the performance of the block seed method is not particularly good since some matrices are near singular. In fact, both methods solve the problem in about the same cost.

In Figure 9.5(a), we can easily see that the single seed method only takes 6 CG runs to solve all the systems. In fact, the first four CG runs almost solve all the systems. The last two are just minor corrections of the unsolved systems. Notice that the first 4 singular values of B are 35.24, 6.18, 0.37 and 0.02, and the remaining ones are only $O(10^{-4})$ or less. Thus, this experiment verifies that the single (and the block) seed method will automatically exploit the rank deficiency of B and hence save a lot of CG solves.

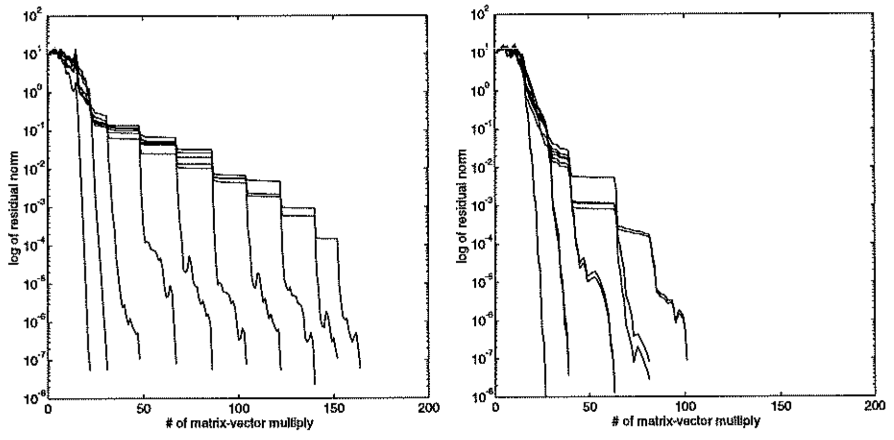


Figure 9.6: The convergence behavior of all the systems (10 right-hand sides) when solving $Ax(t) = b(t)$ where $b(t)$ corresponds to the boundary data of a radar scattering problem and $\Delta t = 10°$ by (a) the single seed method, (b) the block seed method.

For the second case, the result is shown in Figure 9.6(a) and (b). Since Δt is large, the initial guesses given by the single seed method are not good. Also, the $b^{(j)}$'s are not close to each other, and hence the matrix B has full rank. These unfavorable conditions weaken the performance of the single seed method. For the block seed method, however, it uses two instead of one right-hand side to generate a subspace in each restart, so the subspace is larger, which is good for the convergence of the seed systems as well as for reducing the residual of the non-seed systems by projection. Figure 9.6(a), (b) clearly shows that the block seed method performs much better in this case.

Example 4: In this example, we show the performance of the block seed method with block size, s=3, 4. We use the same matrices A and B as in Example 3, and choose $\Delta t = 10°$. Figure 9.7(a) and (b) show the result of s=3 and 4, respectively. When compared to Figure 9.6(a), (b), we see that the convergence rate of the seeds for s=3, 4 is better than for s=1, 2. In fact, we can see a trend of increasing convergence rate with increasing s. However, for large block size, the near singularity of some matrices may produce unstable results. Figure 9.7(a) and (b) show that some systems do not even converge. From our experience, s=2 is an optimal choice.

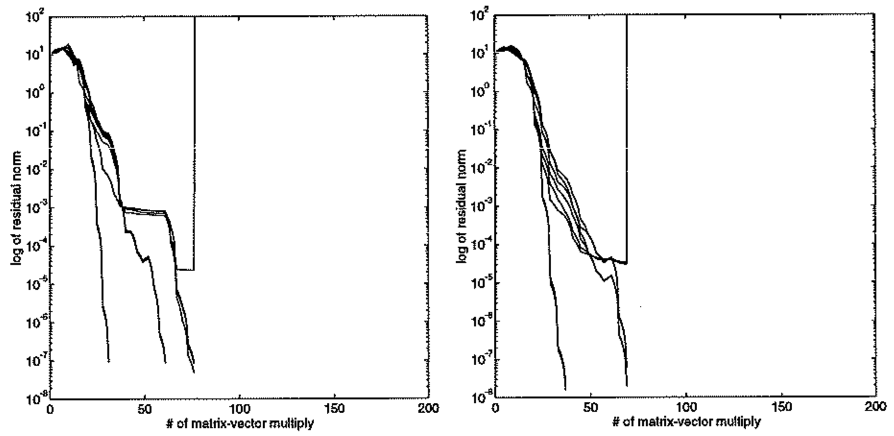


Figure 9.7: The convergence behavior of all the systems when solving $Ax(t) = b(t)$ where $b(t)$ corresponds to the boundary data of a radar scattering problem and $\Delta t = 10^\circ$ by the block seed method with (a) s=3, (b) s=4.

9.7 Concluding Remarks

We have claimed that the single seed method possesses two merits that make it an effective method to solve multiple right-hand sides with smooth dependence on t. One is the super-convergence of the seed system and the other is that it only takes a few number of restart to solve all the systems. These two properties are essentially derived from the intrinsic property of the Galerkin projection. The single seed method automatically makes use of this property and improves the efficiency of solving multiple right-hand sides. We have proved these two properties analytically and also estimated the initial error of the projected solution. We have also performed numerical experiments to justify the theoretical analysis and to show that the single seed method is effective to solve this problem when the right-hand sides are close to each other.

On the other hand, we have also noted the weakness of the single seed method when the right-hand sides are not close to each other. We therefore proposed a block generalization of the single seed method. The basic idea of this block seed method is to exploit a larger subspace by the block seed systems. We have proved that the block seed method possesses the same properties of the single seed method and found that the bounds are better than those for the single seed method. We have estimated the initial errors as well. However, we have noted that the block seed method inherits both the advantages and disadvantages of the block CG method. We have shown numerically that the performance of the block seed method is very good when the right-hand sides are not close to each other but not particularly good in the opposite case. It is not surprising since the subspace generated by the block seed is essentially the same as that generated by a single seed for the latter case. Nevertheless, the block seed method is shown to be an attractive alternative to the single seed method when the right-hand sides are not close to each other.

Bibliography

[1] R.E. Alcouffe, A. Brandt, J.E. Dendy, Jr., and J.W. Painter. The multi-grid method for the diffusion equation with strongly discontinuous coefficients. *SIAM J. Sci. Stat. Comput.*, 2(4):430–454, 1981.

[2] S. F. Ashby, R. D. Falgout, S. G. Smith, and T. W. Fogwell. Multigrid preconditioned conjugate gradients for the numerical simulation of groundwater flow on the Cray T3D. Technical report, Lawrence Livermore National Laboratory, Livermore, 1996.

[3] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, Cambridge, 1994.

[4] I. Babuška. The finite element method for elliptic equations with discontinuous coefficients. *Computing*, 5:207–213, 1970.

[5] R. E. Bank and C. C. Douglas. Sharp estimates for multigrid rates of convergence with general smoothing and acceleration. *SIAM J. Numer. Anal.*, 22:617–633, 1985.

[6] A. Behie and P.A. Forsyth, Jr. Multi-grid solution of three-dimensional problems with discontinuous coefficients. *Appl. Math. Comput.*, 13:229–240, 1983.

[7] M.W. Benson. Iterative solution of large scale linear systems. Master's thesis, Lakehead University, Thunder Bay, Ontario, 1973.

[8] M.W. Benson and P.O. Frederickson. Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems. *Utilitas Math.*, 22:127–140, 1982.

[9] M. Benzi, C. D. Meyer, and M. Tuma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM J. Sci. Comput.*, 17:1135–1149, 1996.

[10] M. Benzi and M. Tuma. A sparse approximate inverse preconditioner for nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 19:968–994, 1998.

[11] G Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms I. *Comm. Pure Appl. Math.*, 44:141–184, 1991.

[12] C. De Boor. Dichotomies for band matrices. *SIAM J. Numer. Anal.*, 17:894–907, 1980.

[13] J. Bramble. *Multigrid Methods.* Longman Scientific & Technical, Essex, UK, 1993.

[14] J. Bramble, J. Pasciak, J. Wang, and J. Xu. Convergence estimates for multigrid algorithms without regularity assumptions. *Math. Comp.*, 57:23–45, 1991.

[15] J. Bramble, J. Pasciak, J. Wang, and J. Xu. Convergence estimates for product iterative methods with applications to domain decomposition and multigrid. *Math. Comp.*, 57:1–21, 1991.

[16] J. Bramble and J. Xu. Some estimates for a weighted l^2 projection. *Math. Comp.*, 56:463–476, 1991.

[17] A. Brandt. Multigrid techniques: 1984 guide with applications to fluid dynamics. In *GMD-Studien Nr. 85. Gesellschaft für Mathematik und Datenverarbeitung*, St. Augustin, 1984.

[18] A. Brandt, S. McCormick, and J. Ruge. Algebraic multigrid for automatic multigrid solution with application to geodetic computations. Manuscript, 1983.

[19] F.X. Canning and J.F. Scholl. Diagonal preconditioners for the EFIE using a wavelet basis. *IEEE Trans. on Antennas. and Propagation.*, 44:1239–1246, 1996.

[20] R. Chan and T. Chan. Circulant preconditioners for elliptic problems. *Numer. Linear Algebra Appl.*, 1:77–101, 1992.

[21] R. Chan, T. Chan, and C. Wong. Cosine transform based preconditioners for total variation minimization problems in image processing. Technical Report 95-23, Dept. of Mathematics, UCLA, 1995.

[22] R. Chan, Q. Chang, and H. Sun. Multigrid method for ill-conditioned symmetric toeplitz systems. Technical Report 95-12, The Chinese University of Hong Kong, 1995.

[23] R. Chan, K. Ng, and C. Wong. Sine transform based preconditioners for symmetric toeplitz systems. *Linear Algebra Appls.*, 232:237–260, 1996.

[24] R. Chan and M. Ng. Conjugate gradient methods for toeplitz systems. *SIAM Review*, 38:427–482, 1996.

[25] R.H. Chan, T.F. Chan, and W.L. Wan. Multigrid for differential-convolution problems arising from image processing. In *Proceedings of the Workshop on Scientific Computing, March 10-12, 1997, Hong Kong.* Springer-Verlag, 1997.

[26] T. Chan. An optimal circulant preconditioner for toeplitz systems. *SIAM J. Sci. Stat. Comput.*, 9:766–771, 1988.

[27] T. Chan and P. Mulet. Iterative methods for total variation image restoration. Technical Report 96-38, Dept. of Mathematics, UCLA, 1996.

[28] T.F. Chan and H.C. Elman. Fourier analysis of iterative methods for elliptic problems. *SIAM Review*, 31:20–49, 1989.

[29] T.F. Chan, S. Go, and L. Zikatanov. Lecture notes on multilevel methods for elliptic problems on unstructured grids. Lecture Course 28th Computational Fluid Dynamics, 3-7 March, 1997, von Karman Institute for Fluid Dynamics, Belgium, 1997.

[30] T.F. Chan and T.P. Mathew. Domain decomposition algorithms. In *Acta Numerica*, pages 61–143. Cambridge University Press, 1994.

[31] T.F. Chan and B. Smith. Domain decomposition and multigrid methods for elliptic problems on unstructured meshes. In David Keyes and Jinchao Xu, editors, *Domain Decomposition Methods in Science and Engineering, Proceedings of the Seventh International Conference on Domain Decomposition, October 27–30, 1993, The Pennsylvania State University.* American Mathematical Society, Providence, 1994.

[32] T.F. Chan, B. Smith, and J. Zou. Multigrid and domain decomposition methods for unstructured meshes. In I.T. Dimov, Bl. Sendov, and P. Vassilevski, editors, *Proceedings of the Third International Conference on Advances in Numerical Methods and Applications, Sofia, Bulgaria*, pages 53–62. World Scientific, August 1994.

[33] T.F. Chan, W.P. Tang, and W.L. Wan. Wavelet sparse inverse preconditioning. *BIT*, 37:644–660, 1997.

[34] T.F. Chan and H.A. van der Vorst. Approximate and incomplete factorization. Technical Report CAM 94-27, Department of Mathematics, UCLA, 1994.

[35] T.F. Chan, J. Xu, and L. Zikatanov. Agglomeration strategies in multigrid method. In preparation, 1997.

[36] E. Chow and Y. Saad. Approximate inverse techniques for block-partitioned matrices. *SIAM J. Sci. Comput.*, 18:1657–1675, 1997.

[37] E. Chow and Y. Saad. Approximate inverse preconditioners via sparse-sparse iterations. *SIAM J. Sci. Comput.*, 19:995–1023, 1998.

[38] T. Coleman and J. Moré. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM J. Numer. Anal.*, 20:187–209, 1983.

[39] J.D.F. Cosgrove, J.C. Diaz, and A. Griewank. Approximate inverse preconditionings for sparse linear systems. *Intern. J. Computer Math*, 44:91–110, 1992.

[40] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Comm. Pure Appl. Math.*, 41:909–996, 1988.

[41] I. Daubechies. *Ten Lectures on Wavelets, CBMS-NSF Series Appl. Math.* SIAM, 1991.

[42] D. F. D'Azevedo, P. A. Forsyth, and W. P. Tang. Towards a cost effective ILU preconditioner with high level fill. *BIT*, 32:442–463, 1992.

[43] S. Demko. Inverses of band matrices and local convergence of spline projections. *SIAM J. Numer. Anal.*, 14:616–619, 1977.

[44] S. Demko, W.F. Moss, and P.W. Smith. Decay rates for inverses of band matrices. *Math. Comp.*, 43:491–499, 1984.

[45] J.W. Demmel, M.T. Heath, and H.A. van der Vorst. Parallel linear algebra. In *Acta Numerica*, pages 111–198. Cambridge University Press, 1993.

[46] J.E. Dendy, Jr. Black box multigrid. *J. Comp. Phys.*, 48:366–386, 1982.

[47] J.E. Dendy, Jr. Black box multigrid for nonsymmetric problems. *Appl. Math. Comput.*, 13:261–283, 1983.

[48] J.E. Dendy, Jr., M.P. Ida, and J.M. Rutledge. A semi-coarsening multigrid algorithm for SIMD machines. *SIAM J. Sci. Stat. Comput.*, 13:1460–1469, 1992.

[49] M. Dryja, M. Sarkis, and O. Widlund. Multilevel Schwarz methods for elliptic problems with discontinuous coefficients in three dimensions. *Numer. Math.*, 72:313–348, 1996.

[50] M. Dryja, B. Smith, and O. Widlund. Schwarz analysis of iterative substructuring algorithms for problems in three dimensions. *SIAM J. Numer. Anal.*, 31:1662–1694, 1994.

[51] M. Dryja and O.B. Widlund. Towards a unified theory of domain decomposition algorithms for elliptic problems. In Tony F. Chan, Roland Glowinski, Jacques Periaux, and Olof B. Widlund, editors, *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 3–21. SIAM, Philadelphia, 1990.

[52] T. Dupont, R. Kendall, and H. Rachford. An approximate factorization procedure for solving self-adjont elliptic difference equations. *SIAM J. Numer. Anal.*, 5:559–573, 1968.

[53] V. Eijkhout and B. Polman. Decay rates of inverses of banded m-matrices that are near to toeplitz matrices. *Linear Algebra and Its Appl.*, 109:247–277, 1988.

[54] B. Engquist and E. Luo. Multigrid methods for differential equations with highly oscillatory coefficients. In *Proceedings of the Sixth Copper Mountain Conference on Multigrid Methods*, pages 175–190, 1993.

[55] Bjorn Engquist, Stanley Osher, and Sifen Zhong. Fast wavelet basd algorithms for linear evolution equations. *SIAM J. Sci. Comput.*, 15:755–775, 1994.

[56] C. Farhat and F. X. Roux. Implicit parallel processing in structural mechanics. Technical Report CU-CSSC-93-26, Center for Aerospace Structures, University of Colorado, November 1993.

[57] Paul F. Fisher. Projection techniques for iterative solution of Ax=b with successive right-hand sides. Technical Report 93-90, ICASE, December 1993.

[58] J. Fuhrmann. A modular algebraic multilevel method. In *Virtual Proceedings of the Ninth International GAMM-Workshop on Parallel Multigrid Methods*, 1996.

[59] M. Garey and D. Johnson. *Computers and Intractability.* W. H. Freeman, New York, 1979.

[60] A. George and J. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[61] R. Glowinski, A. Rieder, R.O. Wells, Jr., and X. Zhou. A wavelet multigrid preconditioner for dirichlet boundary-value problems in general domains. Technical Report TR93-06, Rice Computational Mathematics Laboratory, 1993.

[62] G. H. Golub and R. Underwood. The block lanczos method for computing eigenvalues. In J. R. Rice, editor, *Mathematical Software III*, pages 361–377, New York, 1977. Academic Press.

[63] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins, 1989.

[64] N. I. M. Gould and J. A. Scott. On approximate-inverse preconditioners. Technical Report RAL-TR-95-026, The Central Laboratory of the Research Councils, 1995.

[65] I. G. Graham and M. J. Hagger. Unstructured additive schwarz - cg method for elliptic problems with highly discontinuous coefficients. Technical Report 9608, School of Mathematical Sciences, University of Bath, Bath, UK, 1996.

[66] A. Greenbaum, C. Li, and H. Chao. Parallizing PCG algorithms. *Comp. Phys. Comm.*, 53:295–309, 1989.

[67] M. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM J. Sci. Comput.*, 18:838–853, 1997.

[68] M. Grote and H. Simon. Parallel preconditioning and approximate inverses on the connection machine. In *Scalable High Performance Computing Conference (SHPCC), 1992 Williamsburg, VA*, pages 76–83. IEEE Computer Science Press, 1992.

[69] M. Grote and H. Simon. Parallel preconditioning and approximate inverses on the connection machine. In Richard Sincover et al, editor, *Sixth SIAM Conference on Parallel Processing for Scientific Computing II*, pages 519–523. SIAM, 1993.

[70] H. Guillard. Node-nested multigrid method with delaunay coarsening. Technical Report RR-1898, INRIA, Sophia Antipolis, France, March 1993.

[71] I. Gustafsson. A class of first order factorization methods. *BIT*, 18:142–156, 1978.

[72] W. Hackbusch. Ein iteratives Verfahren zur schnellen Auflosung elliptischer Randwert-probleme. Technical Report 76-12, Universitat zu Koln, 1976.

[73] W. Hackbusch. *Multi-grid Methods and Applications.* Springer-Verlag, Berlin, Heidelberg, 1985.

[74] W. Hackbusch. The frequency decomposition multigrid method, part I: Application to anisotropic equations. *Numer. Math.*, 56:229–245, 1989.

[75] W. Hackbusch. *Integral Equations: Theory and Numerical Treatment.* Birkhäuser Verlag, Basel, Switzerland, 1995.

[76] A. Harten. Discrete multi-resolution analysis and generalized wavelets. Technical Report 92-08, Dept of Mathematics, UCLA, 1992.

[77] P. W. Hemker. The incomplete ILU-decomposition as a relaxation method in multi-grid algorithms. In J. H. Miller, editor, *Boundary and Interior Layers - Computational and Asymptotic Methods*, pages 306–311. Boole Press, 1980.

[78] T.Y. Hou, X.H. Wu, and Z. Cai. Convergence of a multiscale finite element method for elliptic problems with rapidly oscillating coefficients. Submitted to Math. Comp., 1996.

[79] T. Huckle. Sparse approximate inverses and multigrid methods. Sixth SIAM Conference on Applied Linear Algebra, Snowbird, October29–November 1, 1997.

[80] A. Jameson. Solution of the Euler equation for two-dimensional flow by a multigrid method. *Appl. Math. and Comp.*, 13:327–355, 1983.

[81] P. Joly. Résolution de systèmes linéaires avec plusieurs seconds members par la méthode du gradient conjugué. Technical Report R-91012, Publications du Laboratoire d'Analyse Numérique, Université Pierre et Marie Curise, Paris, March 1991.

[82] M. Jones and P. Plassmann. Solution of large, sparse systems of linear equations in massively parallel applications. In *Proceedings of Supercomputing 92*, pages 551–560. IEEE Computer Society Press, 1992.

[83] M. Jones and P. Plassmann. A parallel graph coloring heuristic. *SIAM J. Sci. Comput.*, 14:654–669, 1993.

[84] M. Jones and P. Plassmann. Scalable iterative solution of sparse linear systems. *Parallel Computing*, 20:753–773, 1994.

[85] R. Kettler. Analysis and comparison of relazation schemes in robust multigrid and precondidtioned conjugate gradient methods. In W. Hackbusch and U. Trottenberg, editors, *Multigrid Methods*, Nov 1981.

[86] R. Kettler and J.A. Meijerink. A multigrid method and a combined multigrid-conjugate gradient method for elliptic problems with strongly discontinuous coefficients in general domains. Shell Publ 604, KSEPL, Rijswijk, 1981.

[87] S. Knapek. Matrix-dependent multigrid-homogenization for diffusion problems. In *1996 Copper Mountain Conference on Iterative Methods*, 1996.

[88] L.Yu. Kolotilina, A.A. Nikishin, and A.Yu. Yeremin. Factorized sparse approximate inverse (FSAI) preconditionings for solving 3d fe systems on massively parallel computers II. In R. Beauwens and P.de Groen, editors, *Iterative Methods in Linear Algebra, Proc of the IMACS International Symposium, Brussels, April 2-4, 1991*, pages 311–312, 1992.

[89] L.Yu. Kolotilina and A.Yu. Yeremin. Factorized sparse approximate inverse preconditionings I. theory. *SIAM J. Matrix Anal. Appl.*, 14:45–58, 1993.

[90] M.H. Lallemand, H. Steve, and A. Dervieux. Unstructured multigridding by volume agglomeration: Current status. *Computers and Fluids*, 21:397–433, 1992.

[91] R. LeVeque and Z. Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM J. Numer. Anal.*, 31:1019–1044, 1994.

[92] Ju.B. Lifshitz, A.A. Nikishin, and A.Yu. Yeremin. Sparse approximate inverse (FSAI) preconditionings for solving 3D CFD problems on massively parallel computers. In R. Beauwens and P.de Groen, editors, *Iterative Methods in Linear Algebra, Proc of the IMACS International Symposium, Brussels, April 2-4, 1991*, pages 83–84, 1992.

[93] E. Luo. *Multigrid Method for Elliptic Equation with Oscillatory Coefficients.* PhD thesis, Department of Mathematics, UCLA, California, US, 1993.

[94] J. Mandel. Multigrid convergence for nonsymmetric, infinite variational problems and one smoothing step. *Applied Math. Comput.*, pages 201–216, 1986.

[95] J. Mandel, M. Brezina, and P. Vanek. Energy optimization of algebraic multigrid bases. Technical Report 125, Dept. of Mathematics, UC Colorado, Denver, 1998.

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

[97] S. McCormick. *Multigrid Methods.* SIAM, Philadelphia, Penn, 1988.

[98] J. Meijerink and H.A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric m-matrix. *Math. Comp.*, 31:148–162, 1977.

[99] G. Meurant. A review on the inverse of symmetric tridiagonal and block tridiagonal matrices. *SIAM J. Matrix Anal. Appl.*, 13:707–728, 1992.

[100] A. A. Nikishin and A. Yu. Yeremin. Variable block cg algorithms for solving large sparse symmetric positive definite linear systems on parallel computers, I: General iterative scheme. Technical Report EM-RR 1/92, Elegant Mathematics, Inc. (USA), February 1992.

[101] Dianne P. O'Leary. The block conjugate gradient algorithm and related methods. *Linear Algebra and Its Applications*, 29:292–322, 1980.

[102] M. Omen. Fast multigrid techniques in total variation-based image reconstruction. In *Proceedings of the 1995 Copper Mountain Conference on Multigrid Methods*, 1995.

[103] E. Ong. *Hierarchical Basis Preconditioners for Second Order Elliptic problems in Three Dimensions.* PhD thesis, University of Washington, Seattle, 1989.

[104] M. Papadrakakis and S. Smerou. A new implementation of the lanczos method in linear problems. *International Journal for Numerical Methods in Engineering*, 29:141–159, 1990.

[105] B. N. Parlett. A new look at the lanczos algorithm for solving symmetric systems of linear equations. *Linear Algebra Appl.*, 29:323–346, 1980.

[106] B. N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice-Hall, 1980.

[107] P. Plassmann. Private communication.

[108] A. Ramage and A. Wathen. On preconditioning for finite-element equations on irregular grids. *SIAM J. Matrix Anal. and Appl.*, 15:909–921, 1994.

[109] A. Reusken. Multigrid with matrix-dependent transfer operators for a singular perturbation problem. *Computing*, 50:199–211, 1993.

[110] A. Reusken. Multigrid with matrix-dependent transfer operators for convection-diffusion problems. In *Multigrid Methods, Vol. IV, Springer Lectures in Math.* Springer-Verlag, 1993.

[111] L.I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[112] J.W. Ruge and K. Stuben. Algebraic multigrid. In S. McCormick, editor, *Multigrid Methods*, pages 73–130. SIAM, 1987.

[113] Y. Saad. On the rates of convergence of the lanczos and the block-lanczos methods. *SIAM J. of Numer. Anal.*, 17:687–706, 1980.

[114] Y. Saad. On the lanczos method for solving symmetric linear systems with several right-hand sides. *Math. Comp.*, 48:651–662, 1987.

[115] Y. Saad. *Numerical Methods for Large Eigenvalue Problems.* Manchester University Press, 1992.

[116] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing, 1996.

[117] L. Ridgway Scott and Shangyou Zhang. Finite element interpolation of non-smooth functions satisfying boundary conditions. *Math. Comp.*, 54:483–493, 1990.

[118] George F. Simmons. *Introduction to Topology and Modern Analysis.* McGraw-Hill, 1963.

[119] V. Simoncini and E. Gallopoulos. A memory-conserving hybrid method for solving linear systems with multiple right-hand sides. In *Copper Mountain Conf. Iterative Methods*, April 1992.

[120] V. Simoncini and E. Gallopoulos. An iterative method for nonsymmetric systems with multiple right-hand sides. *SIAM J. Sci. Comput.*, 16(4):917–933, 1995.

[121] B. Smith, P. Bjorstad, and W. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge University Press, Cambridge, 1996.

[122] Charles F. Smith. *The Performance of Preconditioned Iterative Methods in Computational electromagnetics.* PhD thesis, Dept. of Electrical Engineering, University of Illinois at Urbana-Champaign, 1987.

[123] Charles F. Smith, Andrew F. Peterson, and Raj Mittra. A conjugate gradient algorithm for the treatment of multiple incident electromagnetic fields. *IEEE Transactions On Antennas And Propagation*, 37(11):1490–1493, 1989.

[124] R.A. Smith and A. Weiser. Semicoarsening multigrid on a hypercube. *SIAM J. Sci. Stat. Comput.*, 13:1314–1329, 1992.

[125] H. Stone. Iterative solution of implicit approximations of multidimensional partial differential equations. *SIAM J. Numer. Anal.*, 5:530–558, 1968.

[126] G. Strang. A proposal for toeplitz matrix calculations. *Stud. Appl. Math.*, 74:171–176, 1986.

[127] G. Strang and G. J. Fix. *An Analysis of the Finite Element Method.* Prentice-Hall, Englewood Cliffs, N.J., 1973.

[128] S. Ta'asan. *Multigrid Methods for Highly Oscillatory Problems.* PhD thesis, Weizmann Institute of Science, Rehovat, Israel, 1984.

[129] W. P. Tang. *Schwarz Splitting and Template Operators.* PhD thesis, Computer Science Dept., Stanford University, Stanford, 1987.

[130] W. P. Tang. Effective sparse approximate inverse preconditioners. In preparation, 1995.

[131] W. P. Tang. Towards an effective approximate inverse preconditioner. *SIAM J. Sci. Comput.*, 19, 1998.

[132] A. van der Sluis and H. A. van der Vorst. The rate of convergence of conjugate gradients. *Numerische Mathematik*, 48:543–560, 1986.

[133] H. A. van der Vorst. An iteration solution method for solving f(A)x=b, using krylov subspace information obtained for the symmetric positive definite matrix a. *Journal of Computational and Applied Mathematics*, 18:249–263, 1987.

[134] P. Vanek, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56:179–196, 1996.

[135] P.S. Vassilevski and J. Wang. Stabilizing the hierarchical basis by approximate wavelets, I: Theory. Technical Report 95-47, Dept of Mathematics, UCLA, 1995.

[136] P.S. Vassilevski and J. Wang. Stabilizing the hierarchical basis by approximate wavelets, II: Implementation and Numerical Results. Technical Report 95-48, Dept of Mathematics, UCLA, 1995.

[137] C. R. Vogel. Private communication. March 97.

[138] C. R. Vogel. A multigrid method for total variation-based image denoising. In K. Bowers and J. Lund, editors, *Computation and Control IV*. Birkhauser, 1995.

[139] C.R. Vogel and M.E. Oman. Iterative methods for total variation denoising. *SIAM J. Sci. Comput.*, 17:227–238, 1996.

[140] W. L. Wan. An energy-minimizing interpolation for multigrid. Technical Report 97-18, Dept of Mathematics, UCLA, 1997.

[141] W. L. Wan, T. F. Chan, and B. Smith. An energy-minimizing interpolation for robust multigrid. Technical Report 98-6, Dept. of Mathematics, UCLA, 1998.

[142] J. W. Watts-III. A conjugate gradient truncated direct method for the iterative solution of the reservoir simulation pressure equation. *Society of Petroleum Engineer Journal*, 21:345–353, 1981.

[143] P. Wesseling. A robust and efficient multigrid method. In *Multi-grid Methods, Proceedings, Köln-Porz, November 1981*, Lecture Notes in Mathematics 960, pages 614–630. Springer-Verlag, 1982.

[144] P. Wesseling. Theoretical and practical aspects of a multigrid method. *SIAM J. Sci. Stat. Comput.*, 3:387–407, 1982.

[145] P. Wesseling. *An Introduction to Multigrid Methods.* Wiley, Chichester, 1992.

[146] G. Wittum. On the robustness of ILU smoothing. *SIAM J. Sci. Stat. Comp.*, 10:699–717, 1989.

[147] J. Xu. Private communication.

[148] J. Xu. *Theory of Multilevel Methods.* PhD thesis, Cornell University, 1989.

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

[150] H. Yserentant. On the multilevel splitting of finite element spaces. *Numer. Math.*, 49:379–412, 1986.

[151] P.M. De Zeeuw. Matrix-dependent prolongations and restrictions in a black-box multigrid solver. *J. Comp. Appl. Math.*, 33:1–27, 1990.

[152] X. Zhang and J. Bramble. Uniform convergence of the multigrid v-cycle for an anisotropic problem. In *A Numerical Analysis Conference in Honor of Olof Widlund, January 23–24*, Courant Institute, New York, 1998.