A MULTISCALE PRECONDITIONING METHOD FOR THE NUMERICAL SOLUTION OF PARTIAL DIFFERENTIAL AND RELATED INTEGRAL EQUATIONS

by

Jennifer Bailey

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Mathematical and Computer Sciences).

Golden, Colorado

Date 3-31-04

Signed: _____

Jennifer Bailey

Approved: _____

Dr. Junping Wang
Mathematical and Computer Sciences
Thesis Advisor

Approved: _____

Dr. Bernard Bialecki
Mathematical and Computer Sciences
Thesis Co-Advisor

Golden, Colorado

Date 3-31-04

_____

Dr. Graeme Fairweather
Professor and Head
Mathematical and Computer Sciences

ABSTRACT

This thesis contributes to current research by exploring a multiscale preconditioning method for the numerical solution of partial differential equations. This thesis will describe the construction of a general multiscale preconditioner to be used with the preconditioned conjugate gradient method to solve linear systems generated from partial differential equations using the finite element method and the collocation boundary element method on integral equations. Integral equations are an important tool for solving certain partial differential equations, and the boundary element method is a numerical technique for approximating the solution to integral equations. The technique discretizes the boundary of the domain and forms a linear system of equations. Preconditioning is then used to decrease the time needed to solve the linear system with an iterative method.

Two problems will be discussed in this thesis. The first is the numerical solution of Poisson's equation with Dirichlet boundary conditions in one and two dimensions using the finite element method. Poisson's equation occurs in steady-state heat equations and many other mechanical and physical applications. The second problem is the numerical solution of the integral equation form of Laplace's equation, a special case of Poisson's equation, with Dirichlet boundary conditions using the collocation boundary element method.

The goal of multilevel, or multiscale, preconditioners, including the one discussed in this paper, is to decrease the time needed to solve a system of equations. The multilevel preconditioner used with the finite element method decreased the num-

ber of iterations needed for convergence of PCG. However, the preconditioner did not improve convergence with the collocation boundary element formulation of the integral equation. Thus, a replacement preconditioner is generated based upon a BPX-preconditioner.

The main contribution of this thesis is the exploration of spectral theory for the differential form of Poisson's equation and computational justification presented for the differential and integral forms. However, this preconditioner has a much broader impact than the two problems discussed here. It can be used in weather and climate modeling on error covariance matrices. It is also expected to have similar positive results with the solution to Helmholtz's equation, which is known to be difficult to approximate numerically.

TABLE OF CONTENTS

ABSTRACT iii

LIST OF FIGURES vii

LIST OF TABLES viii

ACKNOWLEDGMENTS ix

Chapter 1 INTRODUCTION 1

1.1 Problem Introduction 1
1.2 Significance and Previous Work 2

Chapter 2 PROBLEM DEFINITION 5

2.1 Problem Statement 5
2.2 Poisson's Equation 5

Chapter 3 THE PRECONDITIONED CONJUGATE GRADIENT METHOD 7

3.1 The Conjugate Gradient Method 7
3.2 The Preconditioned Conjugate Gradient Method 9
3.3 The Mathematics and Convergence of CG 10
3.4 Domain Decomposition Preconditioning 17
3.5 Multiscale Preconditioning 18

Chapter 4 MULTISCALE PRECONDITIONING FOR POISSON'S EQUATION 19

4.1 Introduction 19
4.2 Finite Element Formulation for One-Dimensional Problems 19
4.3 A Multiscale Preconditioner 23
4.4 PCG Algorithm for One-Dimensional Problems 29
4.5 Multiscale Preconditioning for Two-Dimensional Problems 33
4.6 A Two-dimensional Multiscale Preconditioner 37
4.7 Spectral Bounds for the Multiscale Preconditioner 43

Chapter 5 MULTISCALE PRECONDITIONING FOR INTEGRAL EQUATIONS 53

5.1 An Integral Representation for Laplace's Equation 53
5.2 The Collocation Boundary Element Method 54
5.3 Multiscale Preconditioning 57
5.4 Multiscale Preconditioning Variant 61

Chapter 6 COMPUTATIONAL INVESTIGATION 64

6.1 Finite Element Formulation of Poisson's Equation 64
6.1.1 One-Dimensional Results 64
6.1.2 Two-Dimensional Results 66
6.2 Collocation BEM Formulation of Laplace's Equation 68

Chapter 7 CONCLUSION AND DISCUSSION 71

7.1 Summary of Work Completed 71
7.2 Future Work 72

REFERENCES 73

LIST OF FIGURES

6.1 Plot of the number of PCG iterations for the 1-D Poisson's equation with and without a multiscale preconditioner. 65

6.2 Plot of the time results for the 1-D Poisson's equation with and without a multiscale preconditioner. 66

6.3 Plot of the number of PCG iterations for the 2-D Poisson's equation with and without a multiscale preconditioner. 67

6.4 Plot of the time results for the 2-D Poisson's equation with and without a multiscale preconditioner. 68

6.5 Plot of the number of PCG iterations for the 2-D Laplace's equation with and without multiscale preconditioners. 70

6.6 Plot of the time results for the 2-D Laplace's equation with and without multiscale preconditioners. 70

LIST OF TABLES

6.1 Computational performance for the 1-D Poisson's equation with and without a multiscale preconditioner. 65

6.2 Computational performance for the 2-D Poisson's equation with and without a multiscale preconditioner. 67

6.3 Computational performance for the 2-D Laplace's equation with and without the multiscale preconditioners. 69

ACKNOWLEDGMENTS

I would like to thank my advisors, Dr. Junping Wang and Dr. Bernard Bialecki, for their support through this process. I would also like to thank my committee members, Dr. Barbara Moskal and Dr. Dinesh Mehta. I am grateful for the encouragement and help of Dr. Graeme Fairweather, department head.

I would like to express my gratitude to the VSEP program at NASA, Goddard Space Flight Center for a wonderful and eye-opening summer. Finally, I would like to acknowledge my family and friends. Without their aid and unyielding support, this thesis would not have been completed.

Chapter 1

INTRODUCTION

1.1 Problem Introduction

The contribution of this thesis will be in exploring a multiscale preconditioning method for the numerical solution of partial differential equations. Integral equations provide an important tool for solving partial differential equations. The boundary integral formulation of a partial differential equation reduces the dimension of the problem by requiring solutions, in terms of integrals involving the solution and its first normal derivative, to be generated only on the boundary of the domain instead of the entire domain. The formulation also provides an efficient way to solve partial differential equations on infinite domains. Again, this is due to the fact that the solutions are determined on the boundary only, and that the boundary conditions at infinity are naturally satisfied.

Preconditioning is an important numerical technique which decreases the time needed to solve linear systems of equations using iterative methods. A good preconditioner reduces the number of iterations needed for convergence and thus the computation time. The finite element method is a numerical technique for solving partial differential equations. The technique discretizes the domain and forms a linear system of equations. The collocation boundary element method is a numerical technique for approximating boundary integral equations. This technique discretizes the boundary of the domain and forms a linear system of equations from the original

integral equation. Such discretized linear systems are often poorly conditioned with a condition number depending on a parameter called the mesh size; the smaller the mesh size, the larger the condition number. Thus, preconditioning is important in the numerical solution of partial differential equations.

This thesis will outline a multiscale preconditioning method which is to be used with the preconditioned conjugate gradient (PCG) method in the numerical solution of partial differential equations with the finite element method (FEM) and the boundary element method (BEM). The thesis will begin (Chapter 1, Chapter 2) by stating the thesis problem and its significance, including previous work done with multilevel or multiscale preconditioners, such as with the finite element method (FEM) solution to partial differential equations. In the next chapter (Chapter 3), it will continue on to describe the PCG method and types of preconditioners and their importance. The following two chapters (Chapter 4, Chapter 5) will describe problems on which the multiscale preconditioner was tested including a finite element application and an integral equation application of the multiscale preconditioner, respectively. The results from applying the preconditioner will be outlined in the next chapter (Chapter 6). Finally, the thesis will conclude with a discussion of the completed work and related future work (Chapter 7).

1.2 Significance and Previous Work

A significant amount of prior work has been done in solving partial differential equations with iterative schemes and domain decomposition. Schwarz has been recognized for his early foundational efforts in domain decomposition [18]. The first qualitative convergence proof was completed by Bramble, Pasciak, Wang, and Xu [5] who provided an abstract theory of additive and multiplicative Schwarz methods.

Other research in domain decomposition, especially on additive methods, includes work done by Widlund [24] who examined a two-level method and provided convergence results. Research concerning multigrid methods has also been extensive. Some examples include the efforts of Bank and Dupont [3] who worked with the finite element method applied to second order elliptic boundary value problems, and Bramble, Pasciak, Wang, and Xu [4, 5, 6] who examined spectral bounds and convergence rates for the multigrid method. Bank and Dupont [3] also worked on two-level hierarchical basis methods, as did Yserentant [26] who was a pioneer in the analysis of hierarchical basis methods. Bramble and Pasciak [6], and Xu [25] also studied multilevel preconditioners. They provided spectral bounds for symmetric elliptic positive definite problems such as the finite element formulation of Poisson's equation which will be used as an example in this thesis. Oswald [14, 15] is another mathematician who completed research on multigrid and multilevel preconditioning as did Xu [25] and Zhang [27].

Much attention has also been given to solving integral equations with iterative methods. Multigrid, conjugate gradient, and GMRES (Generalized Minimal Residual Method) are commonly used solvers. They are combined with many preconditioning techniques ranging from the basic diagonal, block-Jacobi, and ILU preconditioners, to more complicated preconditioners involving multiscale and wavelets. Some mathematicians who have investigated, or are currently investigating, this issue include Vavasis [22], and Kleeman, Rathsfield, and Scheider [13] who have examined a multiscale algorithm for the solution of boundary integral equations based on wavelets. Taush and White [20] are currently exploring wavelet compression, particle-mesh techniques, and panel clustering, including their use for preconditioners. Finally, there is an active field of research being done with multiscale preconditioners for bound-

ary integral equations, including work done by Schmidlin [17] who is implementing a preconditioner for use with GMRES involving wavelets with Haar basis. Warnick and Chew [23] are examining a new type of spectral multigrid, including multigrid preconditioners for long-range interactions in surface integral operations for electromagnetic scattering problems, and Vandewalle [21] is exploring multigrid methods to be used with high frequent antenna problems and low-frequency electromagnetic applications.

Chapter 2

PROBLEM DEFINITION

2.1 Problem Statement

The problem which will be addressed in this thesis is the construction of a general multiscale preconditioner for use with the preconditioned conjugate gradient (PCG) method on linear systems generated from partial differential equations using the finite element method (FEM) and the boundary element method (BEM).

2.2 Poisson's Equation

Poisson's equation occurs in time-independent (steady-state) heat equations, fluid flow, electrostatics, elasticity, gravitation, and other mechanical and physical applications [10, 16]. An example of Poisson's equation in two dimensions with zero Dirichlet boundary conditions is

$$\begin{aligned} -\triangle u &= f \text{ in } \Omega, \\ u &= 0 \text{ on } \partial\Omega, \end{aligned} \tag{2.1}$$

where Ω is an open domain with a Lipschitz boundary $\partial\Omega$, and $\triangle$ is the Laplacian operator defined by $\triangle u = \dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2}$.

The finite element solution to Poisson's equation involves the solution of a linear system of the form $Ax = b$, where A is a matrix and x and b are vectors [7]. This is

the first example to be explored in this thesis, and will be discussed in more detail in Chapter 4. Multilevel preconditioners will be used with the PCG method to speed up the solution of this system.

Laplace's equation is a special case of Poisson's equation with $f = 0$. The equation in two dimensions with a Dirichlet boundary condition is as follows:

$$\begin{aligned} -\triangle u &= 0 \text{ in } \Omega, \\ u &= g \text{ on } \partial\Omega, \end{aligned} \tag{2.2}$$

where Ω is an open domain with a Lipschitz boundary.

The Laplacian equation can be solved with integral equations. Numerically, a method can be obtained through the discretization of these equations on the boundary. The method consists of two equations, one defined only on the boundary of the domain, and one relating points in the domain to the solution on the boundary. Let S be the boundary $\partial\Omega$. The integral equations can be derived from the partial differential equation using Green's second theorem. The first integral equation is given by

$$\int_S \frac{\partial G}{\partial n_q}(p,q)u(q)dS_q + \frac{1}{2}u(p) = \int_S G(p,q)\frac{\partial u}{\partial n_q}(q)dS_q, \quad p \in S, \tag{2.3}$$

where $G(p,q) = -\frac{1}{2\pi}\log|p-q|$, $|p-q|$ is the distance between the two points, and $\frac{\partial}{\partial n_q}$ is the normal derivative [8]. This can be numerically approximated by the linear system, see section 5.1,

$$Lv = z. \tag{2.4}$$

This is the second example to be discussed in this thesis (Chapter 5).

Chapter 3

THE PRECONDITIONED CONJUGATE GRADIENT METHOD

3.1 The Conjugate Gradient Method

The conjugate gradient (CG) method is an algorithm which uses a gradient iterative method to solve a linear system $Ax = b$. It is typically, and most effectively, used when A is square, symmetric, and positive definite. This type of coefficient matrix occurs in finite difference and finite element methods for solving partial differential equations, and also in boundary element methods. It can be used when one or more of these conditions do not hold, but speed of convergence will be limited. Iterative methods are used with large sparse matrices because they are memory efficient.

The idea behind the CG method is that, if the coefficient matrix A is symmetric and positive definite, then the quadratic form $f(x) = \frac{1}{2}x^T Ax - b^T x$ is minimized by the solution to $Ax = b$. Thus, the linear system can be solved by finding an x that minimizes $f(x)$.

An iterative method works by choosing an initial approximation for x and then making a series of iterative corrections until the solution is reached. The idea of the CG method is to start at some initial point x_0 and move down the parabola formed by $f(x)$ until the bottom is reached. We take a step in each direction to line up with the correct solution x. Convergence then occurs in only n steps, where A is an $n \times n$ matrix. To do this, a set of A-orthogonal or conjugate search directions d_i ($d_i^T A d_j = 0, i \neq j$) is utilized. This is equivalent to finding a minimum point along

each search direction. Thus, $x_{i+1} = x_i + \alpha_i d_i$, where α_i is the step size and is chosen to minimize $f(x)$. The search directions are generated using the Gram-Schmidt process on a set of orthogonal vectors $r_i = -\nabla f(x_i) = b - Ax_i$, where r_i is the residual.

Both the space and time complexity are $O(m)$, where m is the number of nonzero entries in A. In practice, the CG method is not necessarily completed in n iterations. Accumulated floating point roundoff error causes the residual to lose accuracy and cancellation error causes the search directions to lose conjugacy. However, even without these factors, convergence analysis is important since CG is often used on matrices so large that it is not possible to run n iterations. Let the spectral radius of A be denoted by

$$\rho(A) = \max_{\lambda \in \Lambda(A)} |\lambda| , \tag{3.1}$$

where $\Lambda(A)$ is the set of eigenvalues of A. The condition number of A is

$$\kappa = \frac{\lambda_{max}}{\lambda_{min}} , \tag{3.2}$$

where λ_{max} is the maximum eigenvalue of A and λ_{min} is the minimum eigenvalue. For the error $e_i = x - x_i$, we define the energy norm by

$$\|e_i\|_A = (e_i^T A e_i)^{\frac{1}{2}} . \tag{3.3}$$

Then we establish convergence of the CG method by noting that

$$\|e_i\|_A^2 \leq \min_{P_i} \max_{\lambda \in \Lambda(A)} [P_i(\lambda)]^2 \|e_0\|_A^2 , \tag{3.4}$$

where P_i is a polynomial of degree $\leq i$ such that $P_i(0) = 1$. This gives

$$\|e_i\|_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^i \|e_0\|_A \tag{3.5}$$

[11, 12].

3.2 The Preconditioned Conjugate Gradient Method

The PCG method is a form of the CG method in which the technique of preconditioning is employed. Preconditioning is used to replace A with a matrix whose condition number is smaller than that of A. CG yields faster convergence when the eigenvalues of the matrix are clustered together. Preconditioning involves finding a symmetric, positive definite matrix M that approximates A and is easier to invert. The goal is then to solve the system of linear equations $M^{-1}Ax = M^{-1}b$. If $\kappa(M^{-1}A) \ll \kappa(A)$ or the eigenvalues of $M^{-1}A$ are better clustered than those of A, then CG will converge more rapidly. The goal is to find a preconditioning matrix that will approximate A well enough that the reduction in the number of iterations makes up for the computation cost of the inverse. It is generally accepted that for large and poorly conditioned coefficient matrices, CG should be implemented with preconditioning.

CG cannot directly be applied to the system $M^{-1}Ax = M^{-1}b$, since $M^{-1}A$ is not generally symmetric, even if the matrices M and A are. Therefore, let $M = Q\Lambda Q^T$ be the eigendecomposition of M and $M^{\frac{1}{2}} = Q\Lambda^{\frac{1}{2}}Q^T$. If M is symmetric positive definite, then $M^{\frac{1}{2}}$ is also, and $(M^{\frac{1}{2}})^2 = M$. Then, $M^{-1}Ax = M^{-1}b$ is equivalent to

$$M^{-\frac{1}{2}}AM^{-\frac{1}{2}}(M^{\frac{1}{2}}x) \;=\; M^{-\frac{1}{2}}b, \tag{3.6}$$

or

$$\hat{A}\hat{x} = \hat{b}, \tag{3.7}$$

where $\hat{A} = M^{-\frac{1}{2}}AM^{-\frac{1}{2}}$ ($\hat{A}$ is symmetric and positive definite), $\hat{x} = M^{\frac{1}{2}}x$, and $\hat{b} = M^{-\frac{1}{2}}b$. Since $M^{-1}A$ and $\hat{A}$ are similar matrices, they have the same eigenvalues. Thus to solve the problem, CG is applied implicitly to $\hat{A}\hat{x} = \hat{b}$, in such a way as there is no actual multiplication by $M^{\frac{1}{2}}$, and the same improvement in convergence is gained.

There are several ways of preconditioning. One way is to let $M = A$. However, the system $Mx = b$ must be solved in the iterative procedure. So, it provides no real advantages. Another way is diagonal or Jacobi preconditioning, where $M = diag[A]$ or M is defined by non-overlapping blocks on the diagonal of A. This method is easy to invert, but its effect on convergence is limited. Additive Schwarz preconditioning is a third way that involves overlapping blocks on the diagonal of A. Convergence is better with this method as it allows communication between blocks, or the domains they represent. Other methods involve multiple domains or grids [9, 11, 12, 19].

3.3 The Mathematics and Convergence of CG

As mentioned previously, the goal of the CG method is to minimize the quadratic form $f(x) = \frac{1}{2}x^TAx - b^Tx$, which is equivalent to solving $\nabla f(x) \equiv Ax - b = 0$ or $Ax = b$ [12]. The iteration equation is $x_{i+1} = x_i + \alpha_i d_i$, where α_i is chosen to minimize $f(x)$ along the direction d_i starting from the point x_i. Now, let $g(\alpha) = f(x_i + \alpha d_i)$. Then $\frac{d}{d\alpha}g(\alpha) = 0$ when $\alpha = \alpha_i$, which can be rewritten as $\nabla f(x_{i+1})d_i = 0$. Thus we choose α such that d_i and $\nabla f(x_{i+1})$ are orthogonal. We note that $\nabla f(x_{i+1}) = -r_{i+1}$.

Then we choose α such that d_i and r_{i+1} are orthogonal, that is

$$\begin{aligned} 0 &= r_{i+1}^T d_i \\ &= (b - Ax_{i+1})^T d_i \\ &= (b - A(x_i + \alpha_i d_i))^T d_i \\ &= (b - Ax_i)^T d_i - \alpha_i (Ad_i)^T d_i \,. \end{aligned} \tag{3.8}$$

Therefore,

$$(b - Ax_i)^T d_i = \alpha_i (Ad_i)^T d_i \,, \tag{3.9}$$

or

$$r_i^T d_i = \alpha_i d_i^T A d_i \,, \tag{3.10}$$

which gives

$$\alpha_i = \frac{r_i^T d_i}{d_i^T A d_i} \,. \tag{3.11}$$

Note that

$$r_{i+1} = r_i - \alpha_i A d_i \,. \tag{3.12}$$

The search directions d_i, where $d_i^T A d_j = 0 \,, i \neq j$, are conjugate and constructed using the Gram-Schmidt process. Let us assume that we have a set of n linearly independent vectors u_i. Then the Gram-Schmidt process gives

$$\begin{aligned} d_0 &= u_0 \\ d_i &= u_i + \sum_{k=0}^{i-1} \beta_{ik} d_k \\ \beta_{ij} &= -\frac{u_i^T A d_j}{d_j^T A d_j} \,. \end{aligned} \tag{3.13}$$

The search directions are constructed by applying the Gram-Schmidt process to the residuals, that is, $u_i = r_i$. Then we have $r_i^T r_j = 0\,, i \neq j$, and

$$\beta_{ij} = -\frac{r_i^T A d_j}{d_j^T A d_j}\,. \tag{3.14}$$

Thus, using (3.12), we have

$$\begin{aligned}
r_i^T r_{j+1} &= r_i^T r_j - \alpha_j r_i^T A d_j \\
\alpha_j r_i^T A d_j &= -r_i^T r_{j+1} + r_i^T r_j \\
r_i^T A d_j &= \begin{cases} \dfrac{1}{\alpha_i} r_i^T r_i & i = j \\ -\dfrac{1}{\alpha_{i-1}} r_i^T r_i & i = j+1 \\ 0 & \text{otherwise} \end{cases} \\
\beta_{ij} &= \begin{cases} \dfrac{1}{\alpha_{i-1}} \dfrac{r_i^T r_i}{d_{i-1}^T A d_{i-1}} & i = j+1 \\ 0 & \text{otherwise}\,. \end{cases}
\end{aligned} \tag{3.15}$$

Note that it is not necessary to store old search vectors. Now let $\beta_i = \beta_{i,i-1}$. Then

$$\begin{aligned}
\beta_i &= \frac{1}{\alpha_{i-1}} \frac{r_i^T r_i}{d_{i-1}^T A d_{i-1}} \\
&= \frac{r_i^T r_i}{r_{i-1}^T d_{i-1}} \\
&= \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}}\,.
\end{aligned} \tag{3.16}$$

The following is the CG algorithm:

$$\begin{aligned}
d_0 &= r_0 = b - A x_0 \\
\text{for } i &= 0, 1, \ldots
\end{aligned}$$

$$\begin{aligned} z &= Ad_i \\ \alpha_i &= \frac{r_i^T r_i}{d_i^T z} \\ x_{i+1} &= x_i + \alpha_i d_i \\ r_{i+1} &= r_i - \alpha_i z \\ \beta_{i+1} &= \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i} \\ d_{i+1} &= r_{i+1} + \beta_{i+1} d_i \,, \end{aligned} \tag{3.17}$$

where x_0 is an initial guess, typically set to zero, and the stopping criterion is

$$\|r_i\| < \epsilon \|r_0\| \,, \tag{3.18}$$

where ϵ is a prescribed error tolerance.

To understand the convergence of CG, notice that in each step of CG, e_i is chosen from $e_0 + D_i$ where

$$D_i = span[Ae_0, A^2 e_0, \ldots, A^i e_0] \,, \tag{3.19}$$

or the error is in a Krylov subspace. Krylov subspaces have the property that

$$e_i = \left(I + \sum_{j=1}^{i} \psi_j A^j \right) e_0 \,, \tag{3.20}$$

where ψ_j are chosen to minimize $\|e_i\|_A$ and are related to α_i and β_i. Let

$$P_i(\lambda) = 1 + \sum_{j=1}^{i} \psi_j \lambda^j \tag{3.21}$$

be a polynomial of degree i. Then $e_i = P_i(A)e_0$. Let $\Lambda(A)$ be the set of eigenvalues of A, and let Φ_i be the set of polynomials P_i such that $P_i(0) = 1$. Now, express e_0 as

a linear combination of orthonormal eigenvectors v_j of A, that is,

$$e_0 = \sum_{j=1}^{n} \varepsilon_j v_j \,. \tag{3.22}$$

Then,

$$e_i = \sum_{j=1}^{n} \varepsilon_j P_i(\lambda_j) v_j \,, \tag{3.23}$$

and

$$\|e_i\|_A^2 = \sum_{j=1}^{n} \varepsilon_j^2 \left[P_i(\lambda_j)\right]^2 \lambda_j \,. \tag{3.24}$$

Convergence is only as good as the convergence of the worst eigenvalue. Hence,

$$\begin{aligned} \|e_i\|_A^2 &\leq \min_{P_i \in \Phi_i} \max_{\lambda \in \Lambda(A)} \left[P_i(\lambda)\right]^2 \sum_{j=1}^{n} \varepsilon_j^2 \lambda_j \\ &= \min_{P_i \in \Phi_i} \max_{\lambda \in \Lambda(A)} \left[P_i(\lambda)\right]^2 \|e_0\|_A^2 \,. \end{aligned} \tag{3.25}$$

It can be proven that $\|e_i\|_A^2$ is minimized by choosing

$$P_i(\lambda) = T_i\left(\frac{a}{b}\right) \,, \tag{3.26}$$

where

$$a = \frac{\lambda_{max} + \lambda_{min} - 2\lambda}{\lambda_{max} - \lambda_{min}} \,, \quad b = \frac{\lambda_{max} + \lambda_{min}}{\lambda_{max} - \lambda_{min}} \,, \tag{3.27}$$

and

$$T_i(w) = \frac{1}{2}\left[(w + \sqrt{w^2 - 1})^i + (w - \sqrt{w^2 - 1})^i\right] \tag{3.28}$$

are Chebyshev Polynomials of the first kind of degree i. Finally,

$$\|e_i\|_A \leq T_i\left(\frac{\lambda_{max} + \lambda_{min}}{\lambda_{max} - \lambda_{min}}\right)^{-1} \|e_0\|_A$$

$$
\begin{aligned}
&= T_i \left(\frac{\kappa+1}{\kappa-1}\right)^{-1} \|e_0\|_A \\
&= 2\left[\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^i + \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^i\right]^{-1} \|e_0\|_A \qquad (3.29) \\
&\leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^i \|e_0\|_A .
\end{aligned}
$$

Thus, if ϵ is a prescribed tolerance, convergence requires [9, 19]

$$
2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^i \leq \epsilon , \qquad (3.30)
$$

or

$$
i \geq \frac{\ln\left(\frac{2}{\epsilon}\right)}{\ln\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)} . \qquad (3.31)
$$

The algorithm for the PCG method is as follows:

$$
\begin{aligned}
r_0 &= b - Ax_0 \\
d_0 &= y_0 = M^{-1} r_0 \\
\text{for } i &= 0, 1, \ldots \\
z &= Ad_i \\
\alpha_i &= \frac{r_i^T y_i}{d_i^T z} \\
x_{i+1} &= x_i + \alpha_i d_i \qquad (3.32) \\
r_{i+1} &= r_i - \alpha_i z \\
y_{i+1} &= M^{-1} r_{i+1} \\
\beta_{i+1} &= \frac{r_{i+1}^T y_{i+1}}{r_i^T y_i} \\
d_{i+1} &= y_{i+1} + \beta_{i+1} d_i .
\end{aligned}
$$

This is derived from applying the CG algorithm to $\hat{A}\hat{x} = \hat{b}$ in (3.7). Then the algorithm becomes:

$$\begin{aligned}
\hat{d}_0 &= \hat{r}_0 = \hat{b} - \hat{A}\hat{x}_0 \\
\text{for } i &= 0, 1, \ldots \\
\hat{z} &= \hat{A}\hat{d}_i \\
\hat{\alpha}_i &= \frac{\hat{r}_i^T \hat{r}_i}{\hat{d}_i^T \hat{z}} \\
\hat{x}_{i+1} &= \hat{x}_i + \hat{\alpha}_i \hat{d}_i \\
\hat{r}_{i+1} &= \hat{r}_i - \hat{\alpha}_i \hat{z} \\
\hat{\beta}_{i+1} &= \frac{\hat{r}_{i+1}^T \hat{r}_{i+1}}{\hat{r}_i^T \hat{r}_i} \\
\hat{d}_{i+1} &= \hat{r}_{i+1} + \hat{\beta}_{i+1} \hat{d}_i \,,
\end{aligned} \tag{3.33}$$

where $\hat{\beta}_i = \beta_i$, $\hat{\alpha}_i = \alpha_i$, $\hat{z} = M^{-\frac{1}{2}} z$, $\hat{x}_i = M^{\frac{1}{2}} x_i$, $\hat{r}_i = M^{-\frac{1}{2}} r_i$, and $\hat{d}_i = M^{\frac{1}{2}} d_i$. This means that x_i converges to $M^{-\frac{1}{2}} \hat{x}$, or

$$\begin{aligned}
x &= M^{-\frac{1}{2}} \hat{x} \\
&= M^{-\frac{1}{2}} \hat{A}^{-1} \hat{b} \\
&= A^{-1} b \,,
\end{aligned} \tag{3.34}$$

and the algorithm (3.33) simplifies to the PCG algorithm (3.32) [9]. As mentioned previously, the effectiveness of the preconditioner is determined by the condition number of $M^{-1}A$ and the clustering of its eigenvalues.

The CG method can be used on systems where the matrix A is not symmetric, positive definite, and/or not square, by solving $A^T A x = A^T b$. However, the condition

number of A^TA is the square of the condition number of A, and thus convergence is slower. There is also a nonlinear CG method. CG can minimize any continuous function f for which ∇f can be computed. This is used in optimization problems. There are three added difficulties to this algorithm. The recursive formula for the residual cannot be used; the step size α_i is more difficult to compute; and multiple choices exist for β_i [19].

3.4 Domain Decomposition Preconditioning

The additive Schwarz method is a domain decomposition algorithm which is used when the matrix A is represented on a physical region Ω, which can be broken into subregions, $\Omega = \bigcup_i \Omega_i$. The algorithm solves on each of the subregions independently which makes it highly parallelizable. The algorithm may not converge quickly if the blocks do not overlap much because information about the solution moves slowly from one subdomain to another. However, as long as the overlap between the blocks is a large enough fraction of the domain, the information will travel quickly enough for convergence.

The additive Schwarz preconditioner is defined as follows:

$$B_{as}^{-1} = \sum_{i=1}^{p} T_i \,, \tag{3.35}$$

where $T_i = R^T A_i^{-1} R$ and $A_i = R_i A R_i^T$. $R_i A R_i^T$ acts as a restriction of A to the domain i, and $R^T A_i^{-1} R$ acts as a prolongation of A_i^{-1} to the whole domain. T_i, R_i, and R are not computed explicitly. R_i and R are called restriction matrices, because they restrict a vector to the ith domain, or they make it the proper size for the ith domain. They are simple matrices composed of 0's and 1's and thus only their action

is explicitly used. Conversely, R_i^T and R^T are prolongation matrices, because they prolong, or extend, a vector to the ith subspace. When used in the PCG method the action of the preconditioner is given by

$$d = B_{as}^{-1} r = \sum_{i=1}^{p} R^T A_i^{-1} R r \,. \tag{3.36}$$

The linear systems $A_i d_i = Rr$ are solved using a direct solver. Then, d_i is prolongated

$$d = \sum_{i=1}^{p} R^T d_i \tag{3.37}$$

[9].

The additive Schwarz preconditioner can also be extended to multiple levels to better improve convergence. The two-level additive Schwarz method is given by

$$B_{as}^{-1} = \sum_{i=0}^{p} T_i \,, \tag{3.38}$$

where $T_0 = R_H^T A_H^{-1} R_H$. A_H is a weighted restriction to a coarse grid [5, 9, 24].

3.5 Multiscale Preconditioning

Domain decomposition preconditioning, such as the additive Schwarz preconditioner mentioned above, involves one or two levels. The two-level scheme involves a fine level and a coarse level to aid in communication and improve convergence. However, multilevel schemes involve multiple nested levels. The goal of these preconditioners, including the multiscale preconditioner discussed here, is to match the eigenvalue frequency of the matrix A on each level. Thus the inverse of the preconditioner will damp the frequency, cluster the eigenvalues, and improve convergence.

Chapter 4

MULTISCALE PRECONDITIONING FOR POISSON'S EQUATION

4.1 Introduction

The objective of this chapter is to describe a multiscale preconditioner to be used for the operator generated from solving Poisson's equation on the unit interval in one dimension, or the unit square in two dimensions, with Dirichlet boundary conditions by the finite element method. The preconditioner implicitly allows for communication between the grid points from level to level.

4.2 Finite Element Formulation for One-Dimensional Problems

Poisson's equation in one dimension on the unit interval is as follows:

$$\begin{aligned} -u'' &= f(x) \text{ in } (0\,,1)\,, \\ u(0) &= \alpha_1\,,\ u(1) = \alpha_2\,, \end{aligned} \tag{4.1}$$

where $f(x)$ is a given function in $L^2(0,1)$. Without loss of generality, it can be assumed that $\alpha_1 = \alpha_2 = 0$.

Let

$$\begin{aligned} V &= \{\phi | \phi \text{ is continuous on } [0,1], \phi' \text{ is piecewise continuous on } [0,1]\}\,, \\ V_0 &= \{v \in V | v(0) = 0, v(1) = 0\}\,. \end{aligned} \tag{4.2}$$

Also, let

$$a(u,v) = \int_0^1 u'(x)v'(x)dx\,, \tag{4.3}$$

and

$$f(v) = \int_0^1 f(x)v(x)dx\,. \tag{4.4}$$

Then the variational form of (4.1) is

$$a(u,v) = f(v)\,, \quad v \in V_0\,, \tag{4.5}$$

where $u \in V_0$.

The interval $[0,1]$ is partitioned into equal subintervals of length h, where h is called the mesh size. The number of subintervals is $n = \frac{1}{h}$, and the number of interior points, $x_i = ih$, is $m = n-1$. This is a finite element partition, and the finite element space is

$$V_h = \{\text{space of continuous piecewise linear polynomials on } [0\,,1]\} \tag{4.6}$$

(V_h is a subspace of V). Piecewise linear meaning, the restriction on each subinterval $[x_{i-1}\,, x_i]$ is linear. Then the solution u of (4.5) is approximated using the Galerkin method. This is done by finding $u_h \in V_h^0$ such that

$$a(u_h, v_h) = f(v_h)\,, \quad v_h \in V_h^0\,, \tag{4.7}$$

where $V_h^0 = V_0 \cap V_h$.

Define the inner product in V_h^0 as the L^2 inner product, that is,

$$(\varphi, \psi) = \int_0^1 \varphi(x)\psi(x)\, dx\,. \tag{4.8}$$

Also, define the operator $A_h : V_h^0 \to V_h^0$ such that for $w_h \in V_h^0$, $A_h w_h \in V_h^0$ is defined by

$$(A_h w_h, v_h) = a(w_h, v_h)\,, \quad v_h \in V_h^0\,. \tag{4.9}$$

Let $f_h \in V_h^0$ be defined by

$$(f_h, v_h) = (f, v_h) = f(v_h)\,, \quad v_h \in V_h^0\,. \tag{4.10}$$

Then it follows from (4.9) and (4.10) that the operator form of (4.7) is

$$A_h u_h = f_h\,. \tag{4.11}$$

Lemma 4.2.1 *A_h is self-adjoint and positive definite, that is,*

$$\begin{aligned} (A_h w_h, v_h) &= (w_h, A_h v_h)\,, \quad w_h\,, v_h \in V_h^0\,, \\ (A_h v_h, v_h) &> 0, \quad 0 \neq v_h \in V_h^0\,. \end{aligned} \tag{4.12}$$

Proof: The self-adjoint property follows from

$$\begin{aligned} (A_h w_h, v_h) &= a(w_h, v_h) \\ &= a(v_h, w_h) \\ &= (A v_h, w_h) \\ &= (w_h, A_h v_h)\,. \end{aligned} \tag{4.13}$$

To prove the operator is positive definite, note that

$$\begin{aligned}(A_h v_h, v_h) &= a(v_h, v_h) \\ &= \int_0^1 (v_h')^2(x)\,dx \\ &\geq 0\,.\end{aligned} \tag{4.14}$$

However, if $\int_0^1 (v_h')^2(x)\,dx = 0$, then $v_h' = 0$ and hence $v_h = C$, where C is a constant. Since $v_h \in V_h^0$, $v_h = 0$, which cannot be true by our assumption. Thus, we have a contradiction and the positive definite property is proven.

□

Let ψ_i be the piecewise linear function defined as follows:

$$\psi_i(x) = \frac{1}{h}\begin{cases} x - x_{i-1}\,, & x_{i-1} \leq x \leq x_i\,, \\ x_{i+1} - x\,, & x_i \leq x \leq x_{i+1}\,, \\ 0\,, & \text{otherwise.} \end{cases} \tag{4.15}$$

Then

$$\psi_i(x_j) = \delta_{ij} = \begin{cases} 1\,, & i = j\,, \\ 0\,, & i \neq j\,. \end{cases} \tag{4.16}$$

Moreover, $\{\psi_i\}_{i=1}^m$ is a basis for V_h^0. This implies that

$$u_h = \sum_{j=1}^m \alpha_j \psi_j\,, \tag{4.17}$$

and hence from (4.7) we have

$$a(u_h, v_h) = f(v_h)\,,$$

$$\begin{aligned} a\left(\sum_{j=1}^{m}\alpha_j\psi_j, v_h\right) &= f(v_h)\,, \\ \sum_{j=1}^{m}\alpha_j a(\psi_j, v_h) &= f(v_h)\,. \end{aligned} \tag{4.18}$$

If $v_h = \psi_i$, we obtain

$$\sum_{j=1}^{m}\alpha_j a(\psi_i, \psi_j) = f(\psi_i)\,, \tag{4.19}$$

which is the linear system $A\alpha = b$, where $\alpha = [\alpha_1, \ldots, \alpha_m]$,

$$A = (a_{ij})_{i,j=1}^{m}\,, \quad a_{ij} = a(\psi_i, \psi_j)\,, \tag{4.20}$$

and

$$b = [b_1, \ldots, b_m]\,, \quad b_i = f(\psi_i)\,. \tag{4.21}$$

4.3 A Multiscale Preconditioner

The goal is to find a self-adjoint and positive definite operator $M_h : V_h^0 \to V_h^0$, that is spectrally equivalent to A_h, that is,

$$c_1(M_h v_h, v_h) \leq (A_h v_h, v_h) \leq c_2(M_h v_h, v_h)\,, \quad v_h \in V_h^0\,, \tag{4.22}$$

where c_1 and c_2 are positive constants independent of the mesh size h. To begin, a multilevel, or multigrid, structure is applied to the domain. The domain is partitioned into different subintervals for each level. The subintervals are of length $h_l = 2^{-l}$ for level $l = 1, \ldots, L$, with $h_L = h$. For level l, the number of subintervals is $n_l = \dfrac{1}{h_l} = 2^l$, and the number of interior points, $x_{il} = ih_l$, is $m_l = n_l - 1$. This is a finite element

partition on the level l, and the finite element space for this level is

$$V_{h_l} = \{\text{space of continuous piecewise linear polynomials on } [0\,,1]\} \tag{4.23}$$

(V_{h_l} is a subspace of V). Let $V_{h_l}^0$ be the space formed by the intersection of the spaces V_0 and V_{h_l}, or $V_{h_l}^0 = V_0 \cap V_{h_l}$. Now let ψ_{il} be the piecewise linear function defined as follows:

$$\psi_{il}(x) = \frac{1}{h_l} \begin{cases} x - x_{i-1,l}\,, & x_{i-1,l} \le x \le x_{i,l}\,, \\ x_{i+1,l} - x\,, & x_{i,l} \le x \le x_{i+1,l}\,, \\ 0\,, & \text{otherwise}\,. \end{cases} \tag{4.24}$$

Then

$$\psi_{il}(x_{jl}) = \delta_{ij} = \begin{cases} 1\,, & i = j\,, \\ 0\,, & i \ne j\,. \end{cases} \tag{4.25}$$

Moreover, $\{\psi_{il}\}_{i=1}^{m_l}$ is a basis for $V_{h_l}^0$.

To begin the preconditioner discussion, consider finding $w_h \in V_h^0$ such that $A_h w_h = z_h$, where $z_h \in V_h^0$ is given. Using (4.9), we have

$$a(w_h, v_h) = (A_h w_h, v_h) = (z_h, v_h)\,, \quad v_h \in V_h^0\,. \tag{4.26}$$

Since

$$w_h = \sum_{l=1}^{L} \sum_{i=1}^{m_l} \alpha_{il} \psi_{il}\,, \tag{4.27}$$

we obtain

$$\begin{aligned} a\left(\sum_{l=1}^{L} \sum_{i=1}^{m_l} \alpha_{il} \psi_{il}, v_h \right) &= (z_h, v_h)\,, \\ \sum_{l=1}^{L} \sum_{i=1}^{m_l} \alpha_{il}\, a(\psi_{il}, v_h) &= (z_h, v_h)\,. \end{aligned} \tag{4.28}$$

If $v_h = \psi_{jk}$, then

$$\sum_{l=1}^{L}\sum_{i=1}^{m_l} \alpha_{il} a(\psi_{il}, \psi_{jk}) \;=\; (z_h, \psi_{jk})\,. \tag{4.29}$$

Since the goal is a preconditioner, nonessential terms can be dropped. In this situation, the nonessential terms are those corresponding to $i \neq j$ or $k \neq l$. Hence,

$$\alpha_{jk} a(\psi_{jk}, \psi_{jk}) \;\approx\; (z_h, \psi_{jk})\,. \tag{4.30}$$

This implies that

$$\alpha_{il} \approx \frac{(z_h, \psi_{il})}{a(\psi_{il}, \psi_{il})}\,. \tag{4.31}$$

Therefore,

$$w_h \approx \sum_{l=1}^{L}\sum_{i=1}^{m_l} \frac{(z_h, \psi_{il})}{a(\psi_{il}, \psi_{il})}\psi_{il}\,. \tag{4.32}$$

We define $M_h^{-1} : V_h^0 \to V_h^0$, and hence M_h, as follows: for $z_h \in V_h^0$, $M_h^{-1} z_h \in V_h^0$ is such that

$$M_h^{-1} z_h = \sum_{l=1}^{L}\sum_{i=1}^{m_l} \frac{(z_h, \psi_{il})}{a(\psi_{il}, \psi_{il})}\psi_{il}\,. \tag{4.33}$$

The next step is to find $a(\psi_{il}, \psi_{il})$. Since $h_l = 2^{-l}$ is the size of the subintervals on level l, using (4.24), we have

$$\begin{aligned} a(\psi_{il}, \psi_{il}) &= \int_0^1 \psi_{il}'(x)\psi_{il}'(x)dx \\ &= \int_{x_{i-1,l}}^{x_{il}} \left(\frac{1}{h_l}\right)^2 dx + \int_{x_{il}}^{x_{i+1,l}} \left(\frac{1}{-h_l}\right)^2 dx \\ &= \frac{h_l}{h_l^2} + \frac{h_l}{h_l^2} \\ &= \frac{2}{h_l}\,. \end{aligned} \tag{4.34}$$

Hence, from (4.33), for $w_h = M_h^{-1} z_h$, we have

$$w_h = \sum_{l=1}^{L} w_{h_l}\,, \tag{4.35}$$

where

$$w_{h_l} = \frac{1}{2} \sum_{i=1}^{m_l} h_l(z_h, \psi_{il})\psi_{il}\,. \tag{4.36}$$

Lemma 4.3.1 *For the basis functions ψ_{il} defined in (4.24), we have*

$$\psi_{il}(x) = \frac{1}{2}\psi_{2i-1,l+1}(x) + \psi_{2i,l+1}(x) + \frac{1}{2}\psi_{2i+1,l+1}(x)\,. \tag{4.37}$$

Proof: The proof of this lemma uses the definition (4.24) of ψ_{il}. Since $\psi_{i,l}$, $\psi_{2i-1,l+1}$, $\psi_{2i,l+1}$, and $\psi_{2i+1,l+1}$ are piecewise linear functions, and therefore determined by their endpoint values, it is sufficient to verify (4.37) for $x = x_{i-1,l}$, $x_{2i-1,l+1}$, $x_{i,l}$, $x_{2i+1,l+1}$, and $x_{i+1,l}$.

We begin by verifying (4.37) for $x = x_{i-1,l} = x_{2i-2,l+1}$. Using (4.25), we have

$$\frac{1}{2}\psi_{2i-1,l+1}(x_{i-1,l}) + \psi_{2i,l+1}(x_{i-1,l}) + \frac{1}{2}\psi_{2i+1,l+1}(x_{i-1,l}) = 0 = \psi_{i,l}(x_{i-1,l})\,. \tag{4.38}$$

Next we verify for $x = x_{2i-1,l+1}$. Using (4.24) and (4.25), we have

$$\begin{aligned} \psi_{i,l}(x_{2i-1,l+1}) &= \frac{1}{2} \\ &= \frac{1}{2}\psi_{2i-1,l+1}(x_{2i-1,l+1}) + \psi_{2i,l+1}(x_{2i-1,l+1}) + \frac{1}{2}\psi_{2i+1,l+1}(x_{2i-1,l+1})\,. \end{aligned} \tag{4.39}$$

Finally, for $x = x_{i,l} = x_{2i,l+1}$, we have

$$\frac{1}{2}\psi_{2i-1,l+1}(x_{i,l}) + \psi_{2i,l+1}(x_{i,l}) + \frac{1}{2}\psi_{2i+1,l+1}(x_{i,l}) = 1 = \psi_{i,l}(x_{i,l})\,. \tag{4.40}$$

Verifications for the other two values of x follow by the symmetry argument.

□

Using (4.37), we obtain

$$(z_h, \psi_{il}) = \frac{1}{2}(z_h, \psi_{2i-1,l+1}) + (z_h, \psi_{2i,l+1}) + \frac{1}{2}(z_h, \psi_{2i+1,l+1})\,, \tag{4.41}$$

and for $w_{h_{l-1}} = \sum_{i=1}^{m_{l-1}} \beta_{i,l-1}\psi_{i,l-1}$, we have

$$\begin{aligned} w_{h_{l-1}} &= \sum_{i=1}^{m_{l-1}} \beta_{i,l-1}\left(\frac{1}{2}\psi_{2i-1,l} + \psi_{2i,l} + \frac{1}{2}\psi_{2i+1,l}\right) \\ &= \beta_{1,l-1}\left(\frac{1}{2}\psi_{1,l} + \psi_{2,l} + \frac{1}{2}\psi_{3,l}\right) + \beta_{2,l-1}\left(\frac{1}{2}\psi_{3,l} + \psi_{4,l} + \frac{1}{2}\psi_{5,l}\right) \\ &\quad + \ldots + \beta_{m_{l-1},l-1}\left(\frac{1}{2}\psi_{m_l-2,l} + \psi_{m_l-1,l} + \frac{1}{2}\psi_{m_l,l}\right) \\ &= \sum_{i=1}^{m_l} \beta_{i,l}\psi_{i,l}\,, \end{aligned} \tag{4.42}$$

where

$$\beta_{i,l} = \begin{cases} \frac{1}{2}\beta_{\hat{i},l-1}\,, & i = 1 \text{ or } i = m_l\,, \\ \beta_{\frac{i}{2},l-1}\,, & i = 2, 4, \ldots, m_l - 1\,, \\ \frac{1}{2}\beta_{\frac{i-1}{2},l-1} + \frac{1}{2}\beta_{\frac{i+1}{2},l-1}\,, & i = 3, 5, \ldots, m_l - 2\,, \end{cases} \tag{4.43}$$

and

$$\hat{i} = \begin{cases} 1\,, & i = 1\,, \\ m_{l-1}\,, & i = m_l\,. \end{cases} \tag{4.44}$$

From (4.35), (4.36), (4.41), and (4.43), we obtain the following algorithm for computing $w_h = M_h^{-1} z_h$. First compute

$$(z_h, \psi_{iL})\,, \quad i = 1, \ldots, m_L\,, \tag{4.45}$$

and

$$w_{h_L} = \frac{1}{2}\sum_{i=1}^{m_L} h_L(z_h, \psi_{iL})\psi_{iL}\,. \tag{4.46}$$

Then

$$\begin{aligned}
&\text{for} \quad l = L-1, \ldots, 1 \\
&\quad \text{for } i = 1, \ldots, m_l \\
&\qquad (z_h, \psi_{il}) = \frac{1}{2}(z_h, \psi_{2i-1,l+1}) + (z_h, \psi_{2i,l+1}) + \frac{1}{2}(z_h, \psi_{2i+1,l+1}) \\
&\quad \text{end} \\
&\quad w_{h_l} = \frac{1}{2}\sum_{i=1}^{m_l} h_l(z_h, \psi_{il})\psi_{il} \\
&\quad \text{for } k = l+1, \ldots, L \\
&\qquad \text{for } i = 1, \ldots, m_k \\
&\qquad\quad \beta_{ik} = \begin{cases} \frac{1}{2}\beta_{\hat{i},k-1}\,, & i = 1 \text{ or } i = m_k \\ \beta_{\frac{i}{2},k-1}\,, & i = 2, 4, \ldots, m_k - 1 \\ \frac{1}{2}\beta_{\frac{i-1}{2},k-1} + \frac{1}{2}\beta_{\frac{i+1}{2},k-1}\,, & i = 3, 5, \ldots, m_k - 2 \end{cases} \\
&\qquad \text{end} \\
&\qquad \text{if } k = L \\
&\qquad\quad w_{h_L} = w_{h_L} + \sum_{i=1}^{m_L} \beta_{iL}\psi_{iL} \\
&\quad \text{end} \\
&\text{end}\,.
\end{aligned} \tag{4.47}$$

We implement the preconditioning algorithm (4.47) using a tree structure. To set up the tree, let each node of the tree represent an interior point x_{il} on the corresponding level. Each node contains a level number l, a point number i, $F_{il} = (z_h, \psi_{il})$, $d_{il} = \frac{1}{2}h_l F_{il}$, a pointer to its parents (there is a possibility of 2 parents per node), and

a pointer to its children (each node has 3 children). The tree contains a pointer to the root of the tree and a pointer to the current node on the tree. The nodes are labeled left to right on each level l (1 to $2^l - 1$). The parents can share a child and children can have up to two parents. Each node i has children $2i - 1$, $2i$, $2i + 1$ on the next level.

4.4 PCG Algorithm for One-Dimensional Problems

The PCG algorithm (3.32) generalized to the operator equation $A_h u_h = f_h$ of (4.11) is:

$$\begin{aligned}
r_h^0 &= f_h - A_h u_h^0 \\
d_h^0 &= y_h^0 = M_h^{-1} r_h^0 \\
\text{for } k &= 0, 1, \ldots \\
z_h &= A_h d_h^k \\
\alpha^k &= \frac{(y_h^k, r_h^k)}{(z_h, d_h^k)} \\
u_h^{k+1} &= u_h^k + \alpha^k d_h^k \\
r_h^{k+1} &= r_h^k - \alpha^k z_h \\
y_h^{k+1} &= M_h^{-1} r_h^{k+1} \\
\beta^{k+1} &= \frac{(y_h^{k+1}, r_h^{k+1})}{(y_h^k, r_h^k)} \\
d_h^{k+1} &= y_h^{k+1} + \beta^{k+1} d_h^k ,
\end{aligned} \tag{4.48}$$

where the inner product is defined in (4.8). In general, for $g_h \in V_h^0$,

$$g_h = \sum_{j=1}^{m} g_j \psi_j , \tag{4.49}$$

and the vector representation of g_h is

$$g = [g_1, \ldots, g_m]^T . \tag{4.50}$$

Also, for any g, we use $\hat{g} = Bg$, where

$$B = (b_{ij})_{i,j=1}^m , \quad b_{ij} = (\psi_i, \psi_j) . \tag{4.51}$$

Let $A_h w_h = z_h$, where $w_h , z_h \in V_h^0$, are represented by the vectors $w = [w_1, \ldots, w_m]^T$ and $z = [z_1, \ldots, z_m]^T$. Definition (4.9) of A_h implies

$$\begin{aligned} (A_h w_h, \psi_i) &= a(w_h, \psi_i) \\ &= \sum_{j=1}^m a(\psi_i, \psi_j) w_j . \end{aligned} \tag{4.52}$$

Also,

$$(z_h, \psi_i) = \sum_{j=1}^m (\psi_i, \psi_j) z_j . \tag{4.53}$$

Equations (4.52) and (4.53) imply that $Aw = Bz$, where A is defined in (4.20) and

$$\begin{aligned} a_{ij} &= a(\psi_i, \psi_j) \\ &= \frac{1}{h} \begin{cases} -1 , & j = i \pm 1 , \\ 2 , & j = i . \end{cases} \end{aligned} \tag{4.54}$$

Also, B as defined in (4.51) is

$$b_{ij} = (\psi_i, \psi_j)$$

$$= \frac{h}{6}\begin{cases} 1\,, & j = i \pm 1\,, \\ 4\,, & j = i\,. \end{cases} \tag{4.55}$$

Then for the PCG algorithm (4.48), where f_h, u_h^k, r_h^k, d_h^k, z_h and $y_h^k \in V_h^0$, we have

$$\begin{aligned} r_h^0 &= f_h - A_h u_h^0\,, \\ (r_h^0, \psi_i) &= (f_h, \psi_i) - (A_h u_h^0, \psi_i)\,, \\ \sum_{j=1}^{m}(\psi_i, \psi_j) r_j^0 &= \sum_{j=1}^{m}(\psi_i, \psi_j) f_j - \sum_{j=1}^{m} a(\psi_i, \psi_j) u_j^0\,, \\ \hat{r}^0 &= \hat{f} - A u^0\,. \end{aligned} \tag{4.56}$$

Assuming that $u_h^0 = 0$ ($u^0 = 0$). $\hat{r}^0$ is known since $\hat{f}_i = (f_h, \psi_i) = f(\psi_i) = b_i$ from (4.21). In the next step, $y_h^0 = M_h^{-1} r_h^0$ is computed as outlined in (4.45)–(4.47). To start this computation we need $(r_h^0, \psi_{iL}) = \hat{r}_i^0$ which are known, and we finish with y^0 which is a representation of $y_h^0 \in V_h^0$. We note that the vector $d^0 = y^0$ since $d_h^0 = y_h^0$. Next, for the vectors for each k, we have

$$\begin{aligned} z_h &= A_h d_h^k\,, \\ (z_h, \psi_i) &= (A_h d_h^k, \psi_i)\,, \\ \sum_{j=1}^{m}(\psi_i, \psi_j) z_j &= \sum_{j=1}^{m} a(\psi_i, \psi_j) d_j^k\,, \\ \hat{z} &= A d^k\,, \end{aligned} \tag{4.57}$$

$$\begin{aligned} \alpha^k &= \frac{(r_h^k, y_h^k)}{(z_h, d_h^k)} \\ &= \frac{\sum_{i=1}^{m} y_i^k \sum_{j=1}^{m}(\psi_i, \psi_j) r_j^k}{\sum_{i=1}^{m} d_i^k \sum_{j=1}^{m}(\psi_i, \psi_j) z_j} \end{aligned}$$

$$
\begin{aligned}
&= \frac{(y^k)^T B r^k}{(d^k)^T B z} \\
&= \frac{(y^k)^T \hat{r}^k}{(d^k)^T \hat{z}} .
\end{aligned} \tag{4.58}
$$

$$
\begin{aligned}
u_h^{k+1} &= u_h^k + \alpha^k d_h^k , \\
(u_h^{k+1}, \psi_i) &= (u_h^k, \psi_i) + \alpha^k (d_h^k, \psi_i) , \\
\sum_{j=1}^{m} (\psi_i, \psi_j) u_j^{k+1} &= \sum_{j=1}^{m} (\psi_i, \psi_j) u_j^k + \alpha^k \sum_{j=1}^{m} (\psi_i, \psi_j) d_j^k , \\
B u^{k+1} &= B u^k + \alpha^k B d^k , \\
u^{k+1} &= u^k + \alpha^k d^k ,
\end{aligned} \tag{4.59}
$$

$$
\begin{aligned}
r_h^{k+1} &= r_h^k - \alpha^k z_h , \\
(r_h^{k+1}, \psi_i) &= (r_h^k, \psi_i) - \alpha^k (z_h, \psi_i) , \\
\sum_{j=1}^{m} (\psi_i, \psi_j) r_j^{k+1} &= \sum_{j=1}^{m} (\psi_i, \psi_j) r_j^k - \alpha^k \sum_{j=1}^{m} (\psi_i, \psi_j) z_j , \\
\hat{r}^{k+1} &= \hat{r}^k - \alpha^k \hat{z} .
\end{aligned} \tag{4.60}
$$

$y_h^{k+1} = M^{-1} r_h^{k+1}$ is computed as outlined in (4.45) – (4.47). To start this we need $(r_h^{k+1}, \psi_{iL}) = \hat{r}_i^{k+1}$ which are known from (4.60) and we finish with y^{k+1} which is a representation of y_h^{k+1}. Finally, we have

$$
\begin{aligned}
\beta^k &= \frac{(r_h^{k+1}, y_h^{k+1})}{(r_h^k, y_h^k)} \\
&= \frac{\sum_{i=1}^{m} y_i^{k+1} \sum_{j=1}^{m} (\psi_i, \psi_j) r_j^{k+1}}{\sum_{i=1}^{m} y_i^k \sum_{j=1}^{m} (\psi_i, \psi_j) r_j^k}
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{(y^{k+1})^T B r^{k+1}}{(y^k)^T B r^k} \\
&= \frac{(y^{k+1})^T \hat{r}^{k+1}}{(y^k)^T \hat{r}^k},
\end{aligned}
\tag{4.61}
$$

$$
\begin{aligned}
d_h^{k+1} &= y_h^k + \beta^k d_h^k, \\
(d_h^{k+1}, \psi_i) &= (y_h^k, \psi_i) + \beta^k (d_h^k, \psi_i), \\
\sum_{j=1}^{m} (\psi_i, \psi_j) d_j^{k+1} &= \sum_{j=1}^{m} (\psi_i, \psi_j) y_j^k + \beta^k \sum_{j=1}^{m} (\psi_i, \psi_j) d_j^k, \\
B d^{k+1} &= B y^k + \beta^k B d^k, \\
d^{k+1} &= y^k + \beta^k d^k.
\end{aligned}
\tag{4.62}
$$

Thus, we have obtained the vector form of PCG (3.32) from the operator form (4.48) for the operator equation (4.11), which is implemented in terms of the vectors $\hat{f}$, u^k, $\hat{r}^k$, d^k, $\hat{z}$, and y^k.

4.5 Multiscale Preconditioning for Two-Dimensional Problems

Poisson's equation in two dimensions on a unit square domain and Dirichlet boundary values is as follows:

$$
\begin{aligned}
-\triangle u &= f(x,y) \text{ in } \Omega = (0,1) \times (0,1), \\
u &= 0 \text{ on } \partial\Omega,
\end{aligned}
\tag{4.63}
$$

where $f(x,y)$ is a given function in $L^2(\Omega)$.

Let

$$
H^1(\Omega) = \{v | v \in L^2, \nabla v \in [L(\Omega)]^2\}, \tag{4.64}
$$

$$H_0^1(\Omega) = \{v | v \in H^1 , v(x) = 0 , x \in \partial\Omega\} . \tag{4.65}$$

Also, let

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \, d\Omega , \tag{4.66}$$

and

$$f(v) = \int_\Omega f \, v \, d\Omega . \tag{4.67}$$

Then the variational form of (4.63) is

$$a(u,v) = f(v) , \quad v \in H_0^1(\Omega) , \tag{4.68}$$

where $u \in H_0^1(\Omega)$.

The domain Ω is now partitioned into equal square subdomains of area h^2, where h is again called the mesh size. The number of subdomains is n^2 $\left(n = \frac{1}{h}\right)$, and the number of interior points, $(x_i, y_j) = (ih, jh)$, is m^2 $(m = n - 1)$. This is the finite element partition and the finite element space is

$$S_h = \{\text{space of continuous piecewise bilinear polynomials on } [0,1] \times [0,1]\} \tag{4.69}$$

(S_h is a subspace of H^1). Then the solution u of (4.68) is approximated using the Galerkin method. This is done by finding $u_h \in S_h^0$ such that

$$a(u_h, v_h) = f(v_h) , \quad v_h \in S_h^0 , \tag{4.70}$$

where $S_h^0 = H_0^1 \cap S_h$.

Define the inner product in S_h^0 as the L^2 inner product, that is,

$$(\varphi, \psi) = \int_\Omega \varphi(x,y)\psi(x,y)\,dx\,dy\,. \tag{4.71}$$

Also, define the operator $A_h : S_h^0 \to S_h^0$ such that for $w_h \in S_h^0$, $A_h w_h \in S_h^0$ is defined by

$$(A_h w_h, v_h) = a(w_h, v_h)\,, \quad v_h \in S_h^0\,. \tag{4.72}$$

Let $f_h \in S_h^0$ be defined by

$$(f_h, v_h) = (f, v_h) = f(v_h)\,, \quad v_h \in S_h^0\,. \tag{4.73}$$

Then the operator form of (4.70) is

$$A_h u_h = f_h\,. \tag{4.74}$$

Lemma 4.5.1 *A_h is self-adjoint and positive definite, that is,*

$$\begin{aligned} (A_h w_h, v_h) &= (w_h, A_h v_h)\,, \quad w_h\,, v_h \in S_h^0 \\ (A_h v_h, v_h) &> 0, \quad 0 \neq v_h \in S_h^0\,. \end{aligned} \tag{4.75}$$

Proof: The self-adjoint property is proved in a similar way to that in (4.13). To prove the operator is positive definite, note that

$$\begin{aligned} (A_h v_h, v_h) &= a(v_h, v_h) \\ &= \int_0^1 \int_0^1 \nabla v_h \cdot \nabla v_h\,dx\,dy \\ &\geq 0\,. \end{aligned} \tag{4.76}$$

However, if $\int_0^1 \int_0^1 \nabla v_h \cdot \nabla v_h \, dx \, dy = 0$, then $\nabla v_h = 0$ and hence $v_h = C$, where C is a constant. Since $v_h \in H_0^1$, $v_h = 0$, which proves the positive definite property, as before.

□

Let ψ_{ij} be the piecewise bilinear function defined by

$$\psi_{ij}(x, y) = \psi_i(x)\psi_j(y)\,, \tag{4.77}$$

where ψ_i, ψ_j are defined in (4.15) Then, $\{\psi_{ij}\}_{i,j=1}^m$ is a basis for S_h^0. This implies that

$$u_h = \sum_{i=1}^m \sum_{j=1}^m \alpha_{ij}\psi_{ij}\,, \tag{4.78}$$

and hence

$$\begin{aligned} a(u_h, v_h) &= f(v_h) \\ a\left(\sum_{i=1}^m \sum_{j=1}^m \alpha_{ij}\psi_{ij}, v_h\right) &= f(v_h) \\ \sum_{i=1}^m \sum_{j=1}^m \alpha_{ij} a(\psi_{ij}, v_h) &= f(v_h)\,. \end{aligned} \tag{4.79}$$

If $v_h = \psi_{pq}$, then we obtain

$$\sum_{i=1}^m \sum_{j=1}^m \alpha_{ij} a(\psi_{ij}, \psi_{pq}) = f(\psi_{pq})\,, \tag{4.80}$$

which is the linear system

$$(A \otimes B + B \otimes A)\alpha = b\,, \tag{4.81}$$

where A is defined in (4.20), B is defined in (4.51), and $\otimes$ is the matrix tensor product.

4.6 A Two-dimensional Multiscale Preconditioner

Th goal is to find a self-adjoint and positive definite operator $M_h : S_h^0 \to S_h^0$, that is spectrally equivalent to the two-dimensional operator A_h. First, a multilevel, or multigrid, structure is applied to the domain. The domain is partitioned into different subdomains for each level. The subdomains are of area h_l^2, where $h_l = 2^{-l}$, for each level $l = 1, \ldots, L$, with $h_L = h$. For each level l, the number of subdomains is n_l^2 $\left(n_l = \dfrac{1}{h_l} = 2^l\right)$, and the number of interior points, $(x_{il}, y_{jl}) = (ih_l, jh_l)$, is m_l^2 $(m_l = n_l - 1)$. This is a finite element partition on the level l, and the finite element space for level l is

$$S_{h_l} = \{\text{space of continuous piecewise bilinear polynomials on } [0,1] \times [0,1]\}\,. \quad (4.82)$$

Let $S_{h_l}^0$ be the intersection of the spaces H_0^1 and S_{h_l}, or $S_{h_l}^0 = H_0^1 \cap S_{h_l}$. Now let ψ_{ijl} be the piecewise bilinear function defined by

$$\psi_{ijl}(x,y) = \psi_{il}(x)\psi_{jl}(y)\,, \quad (4.83)$$

where ψ_{il}, ψ_{jl} are defined in (4.24). Then, $\{\psi_{ijl}\}_{i,j=1}^{m_l}$ is a basis for $S_{h_l}^0$.

To begin the preconditioner discussion, consider finding $w_h \in S_h^0$ such that $A_h w_h = z_h$, where $z_h \in S_h^0$ is given. Using (4.72), we have

$$a(w_h, v_h) = (A_h w_h, v_h) = (z_h, v_h)\,, \quad v_h \in S_h^0\,. \quad (4.84)$$

Since

$$w_h = \sum_{l=1}^{L} \sum_{i=1}^{m_l} \sum_{j=1}^{m_l} \alpha_{ijl} \psi_{ijl} \,, \tag{4.85}$$

we obtain

$$\begin{aligned} a\left(\sum_{l=1}^{L} \sum_{i=1}^{m_l} \sum_{j=1}^{m_l} \alpha_{ijl} \psi_{ijl}, v_h \right) &= (z_h, v_h) \,, \\ \sum_{l=1}^{L} \sum_{i=1}^{m_l} \sum_{j=1}^{m_l} \alpha_{ijl} a(\psi_{ijl}, v_h) &= (z_h, v_h) \,. \end{aligned} \tag{4.86}$$

If $v_h = \psi_{pqk}$, then

$$\sum_{l=1}^{L} \sum_{i=1}^{m_l} \sum_{j=1}^{m_l} \alpha_{ijl} a(\psi_{ijl}, \psi_{pqk}) = (z_h, \psi_{pqk}) \,. \tag{4.87}$$

Since the goal is a preconditioner, the nonessential terms ($i \neq p$ or $j \neq q$ or $l \neq k$) can now be dropped. Hence,

$$\alpha_{pqk} a(\psi_{pqk}, \psi_{pqk}) \approx (z_h, \psi_{pqk}) \,. \tag{4.88}$$

This implies

$$\alpha_{ijl} \approx \frac{(z_h, \psi_{ijl})}{a(\psi_{ijl}, \psi_{ijl})} \,. \tag{4.89}$$

Therefore,

$$w_h \approx \sum_{l=1}^{L} \sum_{i=1}^{m_l} \sum_{j=1}^{m_l} \frac{(z_h, \psi_{ijl})}{a(\psi_{ijl}, \psi_{ijl})} \psi_{ijl} \,. \tag{4.90}$$

We define $M_h^{-1} : S_h^0 \to S_h^0$, and hence M_h, as follows. For $z_h \in S_h^0$, $M_h^{-1} z_h \in S_h^0$ is

such that

$$M_h^{-1} z_h = \sum_{l=1}^{L} \sum_{i=1}^{m_l} \sum_{j=1}^{m_l} \frac{(z_h, \psi_{ijl})}{a(\psi_{ijl}, \psi_{ijl})} \psi_{ijl} \,. \tag{4.91}$$

The next step is to a find representation for $a(\psi_{ijl}, \psi_{ijl})$. Since $h_l = 2^{-1}$ is the size of the subintervals on level l, and using (4.54) and (4.55),

$$\begin{aligned} a(\psi_{ijl}, \psi_{ijl}) &= \int_0^1 \int_0^1 \nabla\psi_{ijl}(x,y) \cdot \nabla\psi_{ijl}(x,y)\, dx\, dy \\ &= \int_0^1 \psi'_{il}(x)\psi'_{il}(x)\, dx \int_0^1 \psi_{jl}(y)\psi_{jl}(y)\, dy \\ &\quad + \int_0^1 \psi_{il}(x)\psi_{il}(x)\, dx \int_0^1 \psi'_{jl}(y)\psi'_{jl}(y)\, dy \\ &= \frac{2}{h_l}\frac{2h_l}{3} + \frac{2h_l}{3}\frac{2}{h_l} \\ &= \frac{8}{3} \,. \end{aligned} \tag{4.92}$$

Hence, from (4.91), for $w_h = M_h^{-1} z_h$, we have

$$w_h = \sum_{l=1}^{L} w_{h_l} \,, \tag{4.93}$$

where

$$w_{h_l} = \frac{3}{8} \sum_{i=1}^{m_l} \sum_{j=1}^{m_l} (z_h, \psi_{ijl}) \psi_{il} \,. \tag{4.94}$$

Lemma 4.6.1 *For the basis functions ψ_{ijl} defined in (4.83), we have*

$$\begin{aligned} \psi_{ijl}(x,y) &= \frac{1}{4}\psi_{2i-1,2j-1,l+1}(x,y) + \frac{1}{2}\psi_{2i,2j-1,l+1}(x,y) + \frac{1}{4}\psi_{2i+1,2j-1,l+1}(x,y) \\ &\quad + \frac{1}{2}\psi_{2i-1,2j,l+1}(x,y) + \psi_{2i,2j,l+1}(x,y) + \frac{1}{2}\psi_{2i+1,2j,l+1}(x,y) \\ &\quad + \frac{1}{4}\psi_{2i-1,2j+1,l+1}(x,y) + \frac{1}{2}\psi_{2i,2j+1,l+1}(x,y) + \frac{1}{4}\psi_{2i+1,2j+1,l+1}(x,y) \,. \end{aligned} \tag{4.95}$$

Proof: The proof follows directly from Lemma 5.3.1.

□

Using (4.95), we obtain

$$\begin{aligned}(z_h,\psi_{ijl}) &= \frac{1}{4}(z_h,\psi_{2i-1,2j-1,l+1})+\frac{1}{2}(z_h,\psi_{2i,2j-1,l+1})+\frac{1}{4}(z_h,\psi_{2i+1,2j-1,l+1})\\ &\quad+\frac{1}{2}(z_h,\psi_{2i-1,2j,l+1})+(z_h,\psi_{2i,2j,l+1})+\frac{1}{2}(z_h,\psi_{2i+1,2j,l+1})\\ &\quad+\frac{1}{4}(z_h,\psi_{2i-1,2j+1,l+1})+\frac{1}{2}(z_h,\psi_{2i,2j+1,l+1})+\frac{1}{4}(z_h,\psi_{2i+1,2j+1,l+1})\,,\end{aligned} \tag{4.96}$$

and for

$$w_{h_{l-1}}=\sum_{i=1}^{m_{l-1}}\sum_{j=1}^{m_{l-1}}\beta_{i,j,l-1}\psi_{i,j,l-1}\,, \tag{4.97}$$

we have

$$\begin{aligned}w_{h_{l-1}} &= \sum_{i=1}^{m_{l-1}}\sum_{j=1}^{m_{l-1}}\beta_{i,j,l-1}\Big(\frac{1}{4}\psi_{2i-1,2j-1,l}+\frac{1}{2}\psi_{2i,2j-1,l}+\frac{1}{4}\psi_{2i+1,2j-1,l}\\ &\quad+\frac{1}{2}\psi_{2i-1,2j,l}+\psi_{2i,2j,l}+\frac{1}{2}\psi_{2i+1,2j,l}+\frac{1}{4}\psi_{2i-1,2j+1,l}\\ &\quad+\frac{1}{2}\psi_{2i,2j+1,l}+\frac{1}{4}\psi_{2i+1,2j+1,l}\Big)\\ &= \sum_{i=1}^{m_l}\sum_{j=1}^{m_l}\beta_{i,j,l}\psi_{i,j,l}\,,\end{aligned} \tag{4.98}$$

where

$$\beta_{i,j,l}=\begin{cases}\frac{1}{4}\beta_{\hat{i},\hat{j},l-1}\,, & i,j=1 \text{ or } i,j=m_l\,,\\ \frac{1}{2}\beta_{\frac{i}{2},\hat{j},l-1}\,, & i=2,4,\ldots,m_l-1\,,\ j=1 \text{ or } j=m_l\,,\\ \frac{1}{4}\beta_{\frac{i-1}{2},\hat{j},l-1}+\frac{1}{4}\beta_{\frac{i+1}{2},\hat{j},l-1}\,, & i=3,5,\ldots,m_l-2\,,\ j=1 \text{ or } j=m_l\,,\\ \frac{1}{2}\beta_{\hat{i},\frac{j}{2},l-1}\,, & i=1 \text{ or } i=m_l\,,\ j=2,4,\ldots,m_l-1\,,\\ \frac{1}{4}\beta_{\hat{i},\frac{j-1}{2},l-1}+\frac{1}{4}\beta_{\hat{i},\frac{j+1}{2},l-1}\,, & i=1 \text{ or } i=m_l\,,\ j=3,5,\ldots,m_l-2\,,\\ \beta_{\frac{i}{2},\frac{j}{2},l-1}\,, & i=2,4,\ldots,m_l-1\,,\ j=2,4,\ldots,m_l-1\,,\\ \frac{1}{2}\beta_{\frac{i}{2},\frac{j-1}{2},l-1}+\frac{1}{2}\beta_{\frac{i}{2},\frac{j+1}{2},l-1}\,, & i=2,4,\ldots,m_l-1\,,\ j=1,3,\ldots,m_l-2\,,\\ \frac{1}{2}\beta_{\frac{i-1}{2},\frac{j}{2},l-1}+\frac{1}{2}\beta_{\frac{i+1}{2},\frac{j}{2},l-1}\,, & i=3,5,\ldots,m_l-2\,,\ j=2,4,\ldots,m_l-1\,,\\ \frac{1}{4}\beta_{\frac{i-1}{2},\frac{j-1}{2},l-1}+\frac{1}{4}\beta_{\frac{i+1}{2},\frac{j-1}{2},l-1}&\\ +\frac{1}{4}\beta_{\frac{i-1}{2},\frac{j+1}{2},l-1}+\frac{1}{4}\beta_{\frac{i+1}{2},\frac{j+1}{2},l-1}\,, & i=3,5,\ldots,m_l-2\,,\ j=3,5,\ldots,m_l-2\,,\end{cases} \tag{4.99}$$

and

$$\hat{i} = \begin{cases} 1\,, & i = 1\,, \\ m_{l-1}\,, & i = m_l\,; \end{cases} \qquad \hat{j} = \begin{cases} 1\,, & j = 1\,, \\ m_{l-1}\,, & j = m_l\,. \end{cases} \tag{4.100}$$

From (4.93), (4.94), (4.96), and (4.99), we obtain the following algorithm for computing $w_h = M_h^{-1} z_h$. First compute

$$(z_h, \psi_{ijL})\,, \quad i = 1, \ldots, m_L\,, \quad j = 1, \ldots, m_l\,, \tag{4.101}$$

and

$$w_{h_L} = \frac{3}{8} \sum_{i=1}^{m_L} \sum_{j=1}^{m_l} (z_h, \psi_{ijL}) \psi_{ijL}\,. \tag{4.102}$$

Then

$$\begin{aligned}
&\text{for} \quad l = L-1, \ldots, 1 \\
&\quad \text{for } i = 1, \ldots, m_l \\
&\quad\quad \text{for } j = 1, \ldots, m_l \\
&\quad\quad\quad (z_h, \psi_{ijl}) = \frac{1}{4}(z_h, \psi_{2i-1,2j-1,l+1}) + \frac{1}{2}(z_h, \psi_{2i,2j-1,l+1}) + \frac{1}{4}(z_h, \psi_{2i+1,2j-1,l+1}) \\
&\quad\quad\quad\quad + \frac{1}{2}(z_h, \psi_{2i-1,2j,l+1}) + (z_h, \psi_{2i,2j,l+1}) + \frac{1}{2}(z_h, \psi_{2i+1,2j,l+1}) \\
&\quad\quad\quad\quad + \frac{1}{4}(z_h, \psi_{2i-1,2j+1,l+1}) + \frac{1}{2}(z_h, \psi_{2i,2j+1,l+1}) + \frac{1}{4}(z_h, \psi_{2i+1,2j+1,l+1}) \\
&\quad\quad \text{end} \\
&\quad \text{end} \\
&\quad w_{h_l} = \frac{3}{8} \sum_{i=1}^{m_l} \sum_{j=1}^{m_l} (z_h, \psi_{ijl}) \psi_{ijl} \\
&\quad \text{for } k = l+1, \ldots, L \\
&\quad\quad \text{for } i = 1, \ldots, m_k
\end{aligned} \tag{4.103}$$

for $j = 1, \ldots, m_k$

$$\beta_{i,j,k} = \begin{cases} \frac{1}{4}\beta_{\hat{i},\hat{j},k-1}\,, & i,j = 1 \text{ or } i,j = m_k\,, \\ \frac{1}{2}\beta_{\frac{i}{2},\hat{j},k-1}\,, & i = 2,4,\ldots,m_k-1\,, j = 1 \text{ or } j = m_k\,, \\ \frac{1}{4}\beta_{\frac{i-1}{2},\hat{j},k-1} + \frac{1}{4}\beta_{\frac{i+1}{2},\hat{j},k-1}\,, & i = 3,5,\ldots,m_k-2\,, j = 1 \text{ or } j = m_k\,, \\ \frac{1}{2}\beta_{\hat{i},\frac{j}{2},k-1}\,, & i = 1 \text{ or } i = m_k\,, j = 2,4,\ldots,m_k-1\,, \\ \frac{1}{4}\beta_{\hat{i},\frac{j-1}{2},k-1} + \frac{1}{4}\beta_{\hat{i},\frac{j+1}{2},k-1}\,, & i = 1 \text{ or } i = m_k\,, j = 3,5,\ldots,m_k-2\,, \\ \beta_{\frac{i}{2},\frac{j}{2},k-1}\,, & i = 2,4,\ldots,m_k-1\,, j = 2,4,\ldots,m_k-1\,, \\ \frac{1}{2}\beta_{\frac{i}{2},\frac{j-1}{2},k-1} + \frac{1}{2}\beta_{\frac{i}{2},\frac{j+1}{2},k-1}\,, & i = 2,4,\ldots,m_k-1\,, j = 1,3,\ldots,m_k-2\,, \\ \frac{1}{2}\beta_{\frac{i-1}{2},\frac{j}{2},k-1} + \frac{1}{2}\beta_{\frac{i+1}{2},\frac{j}{2},k-1}\,, & i = 3,5,\ldots,m_k-2\,, j = 2,4,\ldots,m_k-1\,, \\ \frac{1}{4}\beta_{\frac{i-1}{2},\frac{j-1}{2},k-1} + \frac{1}{4}\beta_{\frac{i+1}{2},\frac{j-1}{2},k-1} & \\ +\frac{1}{4}\beta_{\frac{i-1}{2},\frac{j+1}{2},k-1} + \frac{1}{4}\beta_{\frac{i+1}{2},\frac{j+1}{2},k-1}\,, & i = 3,5,\ldots,m_k-2\,, j = 3,5,\ldots,m_k-2\,, \end{cases}$$

end

end

if $k = L$

$$w_{h_L} = w_{h_L} + \sum_{i=1}^{m_L}\sum_{j=1}^{m_l}\beta_{ijL}\psi_{ijL}$$

end

end.

The algorithm (4.103) is again implemented with a tree. To set up the tree, let each node of the tree represent an interior point on the corresponding level (x_{il}, y_{jl}). Each node contains a level number l, a point number j, a row number i, $F_{ijl} = (z_h, \psi_{ijl})$, $d_{ijl} = \frac{3}{8}F_{ijl}$, a pointer to its parents (4 parents per node), and a pointer to its children (each node has 9 children). The tree contains a pointer to the root of the tree and a pointer to the current node on the tree. The nodes are labeled from left to right on each level l (1 to m_l^2). The parents can share multiple children, and

children can have up to four parents. Each node on row i and column j has children $(2i-1,2j)$, $(2i,2j)$, $(2i+1,2j)$, $(2i-1,2j-1)$, $(2i,2j-1)$, $(2i+1,2j-1)$, $(2i-1,2j+1)$, $(2i,2j+1)$, $(2i+1,2j+1)$.

The PCG algorithm follows from the one-dimensional one, with the following definition. In general, for $g_h \in S_h^0$,

$$g_h = \sum_{i=1}^{m}\sum_{j=1}^{m} g_{ij}\psi_{ij}\,, \tag{4.104}$$

the vector representation of g_h is

$$g = [g_{11}, g_{12}, \ldots, g_{1m}, g_{21}, \ldots, g_{2m}, \ldots, g_{m1}, \ldots, g_{mm}]^T\,. \tag{4.105}$$

4.7 Spectral Bounds for the Multiscale Preconditioner

The purpose of this section is to show that the operator M_h is spectrally equivalent to the operator A_h, where A_h and M_h are defined in (4.9) and (4.33), respectively.

Lemma 4.7.1 *The operator M_h^{-1} defined in (4.33) is self-adjoint and positive definite, that is,*

$$\begin{aligned} (M_h^{-1}w_h, v_h) &= (w_h, M_h^{-1}v_h)\,, \quad w_h, v_h \in V_h^0\,, \\ (M_h^{-1}v_h, v_h) &> 0\,, \quad 0 \neq v_h \in V_h^0\,. \end{aligned} \tag{4.106}$$

Proof: We begin by showing that M_h^{-1} is self-adjoint. For $w_h, v_h \in V_h^0$, we have

$$\begin{aligned} (M_h^{-1}w_h, v_h) &= \sum_{l=1}^{L}\sum_{i=1}^{m_l} \frac{(w_h, \psi_{il})}{a(\psi_{il},\psi_{il})}(\psi_{il}, v_h) \\ &= \sum_{l=1}^{L}\sum_{i=1}^{m_l} \frac{(v_h, \psi_{il})}{a(\psi_{il},\psi_{il})}(\psi_{il}, w_h) \\ &= (w_h, M_h^{-1}v_h)\,. \end{aligned} \tag{4.107}$$

Next we verify that M_h^{-1} is positive definite. For $0 \neq v_h \in V_h^0$, we have

$$\begin{aligned}(M_h^{-1}v_h, v_h) &= \sum_{l=1}^{L}\sum_{i=1}^{m_l} \frac{(v_h, \psi_{il})}{a(\psi_{il}, \psi_{il})}(\psi_{il}, v_h) \\ &= \sum_{l=1}^{L}\sum_{i=1}^{m_l} \frac{(v_h, \psi_{il})^2}{a(\psi_{il}, \psi_{il})} \\ &\geq 0\,,\end{aligned} \tag{4.108}$$

since $(v_h, \psi_{il})^2 \geq 0$, and $a(\psi_{il}, \psi_{il}) > 0$ by (4.34). However, there is an i such that $(v_h, \psi_{iL})^2 > 0$, since $\{\psi_{iL}\}_{i=1}^{m_L}$ forms a basis for V_h^0. Thus $(M^{-1}v_h, v_h) > 0$.

□

Lemma 4.7.1 implies that M_h is self-adjoint and positive definite.

Let V_h^0 be the finite element space defined for the one-dimensional Poisson's equation and let X be a subspace of V_h^0. Then the orthogonal projection of $v_h \in V_h^0$ onto X is $\rho_X v_h \in X$ defined by $(\rho_X v_h, w_h) = (v_h, w_h)\,, \forall w_h \in X$. Now , for $l = 1, \ldots, L$ and $i = 1, \ldots, m_l$, let X_{il} be the subspace spanned by the function ψ_{il}, that is, $X_{il} = \text{span}\{\psi_{il}\}$, and let ρ_{il} be the orthogonal projection onto X_{il} with respect to the a-inner product, that is, $\rho_{il} : V_h^0 \to X_{il}$,

$$a(\rho_{il}v_h, w_h) = a(v_h, w_h)\,, \forall w_h \in X_{il}\,. \tag{4.109}$$

If $w_h = \psi_{il}$,

$$a(\rho_{il}v_h, \psi_{il}) = a(v_h, \psi_{il})\,. \tag{4.110}$$

Since $\rho_{il}v_h \in X_{il}$, $\rho_{il}v_h = \alpha_{il}\psi_{il}$ for some constant α_{il}, and (4.110) gives

$$\alpha_{il}a(\psi_{il}, \psi_{il}) = a(v_h, \psi_{il})$$

$$\alpha_{il} = \frac{a(v_h, \psi_{il})}{a(\psi_{il}, \psi_{il})} . \tag{4.111}$$

Now, let

$$\rho = \sum_{l=1}^{L} \sum_{i=1}^{m_l} \rho_{il} \tag{4.112}$$

be the sum of those projections.

Lemma 4.7.2 *For A_h and M_h defined by (4.9) and (4.33), we have*

$$\rho = M_h^{-1} A_h . \tag{4.113}$$

Proof: For any $v_h \in V_h^0$, using (4.111), (4.9), and (4.33), we have

$$\begin{aligned} \rho v_h &= \sum_{l=1}^{L} \sum_{i=1}^{m_l} \rho_{il} v_h \\ &= \sum_{l=1}^{L} \sum_{i=1}^{m_l} \alpha_{il} \psi_{il} \\ &= \sum_{l=1}^{L} \sum_{i=1}^{m_l} \frac{a(v_h, \psi_{il})}{a(\psi_{il}, \psi_{il})} \psi_{il} \\ &= \sum_{l=1}^{L} \sum_{i=1}^{m_l} \frac{(A_h v_h, \psi_{il})}{a(\psi_{il}, \psi_{il})} \psi_{il} \\ &= M_h^{-1} A_h v_h . \end{aligned} \tag{4.114}$$

□

Theorem 4.7.3 *There exist positive constants, c_1 and c_2, independent of h, such that*

$$c_1 a(v_h, v_h) \leq a(\rho v_h, v_h) \leq c_2 L a(v_h, v_h) , \quad v_h \in V_h^0 . \tag{4.115}$$

Proof: We start by verifying the decomposition condition, namely that there exists a constant $c > 0$, independent of h, such that, for any $v_h \in V_h^0$, there exists a

decomposition

$$v_h = \sum_{l=1}^{L} \sum_{i=1}^{m_l} v_{il} \,, \tag{4.116}$$

where $v_{il} \in X_{il}$, and

$$\sum_{l=1}^{L} \sum_{i=1}^{m_l} a(v_{il}, v_{il}) \leq c a(v_h, v_h) \,. \tag{4.117}$$

For $l = 1, \ldots L$, let $P_l : V_h^0 \to V_{h_l}^0$ be the projection onto $V_{h_l}^0$, with respect to the a-inner product, that is

$$a(P_l v_h, w_h) = a(v_h, w_h) \,, \quad w_h \in V_{h_l}^0 \,. \tag{4.118}$$

With $P_0 = 0$, the decomposition of $v_h \in V_h^0$ is defined by

$$\begin{aligned} v_h &= \sum_{l=1}^{L} v_l \,, \\ v_L &= v_h - P_{L-1} v_h \,, \\ v_l &= (P_l - P_{l-1}) v_h \,, \quad l = 1, \ldots L-1 \,, \end{aligned} \tag{4.119}$$

where

$$v_l = \sum_{i=1}^{m_l} v_{il} \,, \quad v_{il} = v_l(x_{il}) \psi_{il} \,. \tag{4.120}$$

First, we will show that, for $l = 1, \ldots, L$,

$$\sum_{i=1}^{m_l} a(v_{il}, v_{il}) \leq 12 h_l^{-2} \|v_l\|^2_{L^2(0,1)} \,. \tag{4.121}$$

Using (4.120) and (4.34), we have

$$\begin{aligned}\sum_{i=1}^{m_l} a(v_{il}, v_{il}) &= \sum_{i=1}^{m_l} v_l^2(x_{il}) a(\psi_{il}, \psi_{il}) \\ &= 2h_l^{-1} \sum_{i=1}^{m_l} v_l^2(x_{il}) \,. \end{aligned} \tag{4.122}$$

Using (4.120) and (4.24), we also have

$$\begin{aligned}\|v_l\|^2_{L^2(0,1)} &= \int_0^1 v_l^2(x)\, dx \\ &= \sum_{i=0}^{m_l-1} \int_{x_{i,l}}^{x_{i+1,l}} v_l^2(x)\, dx \\ &= \sum_{i=0}^{m_l-1} \int_{x_{i,l}}^{x_{i+1,l}} \left[v_l(x_{i,l})\psi_{il} + v_l(x_{i+1,l})\psi_{i+1,l}\right]^2 dx \\ &= \sum_{i=0}^{m_l-1} \int_{x_{i,l}}^{x_{i+1,l}} \left[v_l(x_{i,l})\frac{(x_{i+1,l}-x)}{h_l} + v_l(x_{i+1,l})\frac{(x-x_{i,l})}{h_l}\right]^2 dx \\ &= \sum_{i=0}^{m_l-1} h_l \left[\frac{1}{3}v_l^2(x_{i,l}) + \frac{1}{3}v_l^2(x_{i+1,l}) + \frac{1}{3}v_l(x_{i,l})v_l(x_{i+1,l})\right] \\ &= \sum_{i=0}^{m_l-1} h_l \left[\frac{1}{6}v_l^2(x_{i,l}) + \frac{1}{6}v_l^2(x_{i+1,l}) + \left(\frac{v_l(x_{i,l})}{\sqrt{6}} + \frac{v_l(x_{i+1,l})}{\sqrt{6}}\right)^2\right] \\ &\geq \frac{1}{6}h_l \sum_{i=1}^{m_l} v_l^2(x_{il}) \,. \end{aligned} \tag{4.123}$$

Thus, (4.122) and (4.123) imply (4.121).

Next, we show that

$$v_l = (I - P_{l-1})v_l \,. \tag{4.124}$$

Clearly it is sufficient to show $P_{l-1}v_l = 0$. First we verify that $P_{l-1}P_l v_h = P_{l-1}v_h$ and $P_{l-1}P_{l-1}v_h = P_{l-1}v_h$. For $w_h \in V^0_{h_{l-1}} \subset V^0_{h_l}$ using (4.118) we obtain

$$\begin{aligned} a(P_{l-1}P_l v_h, w_h) &= a(P_l v_h, w_h) \\ &= a(v_h, w_h) \\ &= a(P_{l-1}v_h, w_h)\,. \end{aligned} \tag{4.125}$$

Hence $P_{l-1}P_l v_h = P_{l-1}v_h$. The other result follows similarly. Then, by (4.119),

$$\begin{aligned} P_{l-1}v_l &= P_{l-1}(P_l - P_{l-1})v_h \\ &= P_{l-1}P_l v_h - P_{l-1}P_{l-1}v_h \\ &= P_{l-1}v_h - P_{l-1}v_h \\ &= 0\,. \end{aligned} \tag{4.126}$$

A finite element result gives

$$\|(I - P_{l-1})v_l\|^2_{L^2(0,1)} \le c h_{l-1}^2 a(v_l, v_l)\,, \tag{4.127}$$

where $c > 0$ is independent of h_l. Hence, (4.121), (4.124), (4.127), and $h_l = \dfrac{h_{l-1}}{2}$ imply

$$\begin{aligned} \sum_{i=1}^{m_l} a(v_{il}, v_{il}) &\le 12 h_l^{-2} \|(I - P_{l-1})v_l\|^2_{L^2(0,1)} \\ &\le 12 \cdot 2^2 \cdot c a(v_l, v_l)\,. \end{aligned} \tag{4.128}$$

Next we prove that

$$a(v_l, v_k) = 0\,, \quad \text{for } l \neq k\,. \tag{4.129}$$

Without loss of generality we assume $l < k$. Using (4.119),

$$\begin{aligned} P_l v_h \in V^0_{h_l} &\subset V^0_{h_{k-1}} \subset V^0_{h_k}, \\ P_{l-1} v_h \in V^0_{h_{l-1}} &\subset V^0_{h_{k-1}} \subset V^0_{h_k}, \end{aligned} \tag{4.130}$$

and (4.118), we have

$$\begin{aligned} a(v_l, v_k) &= a\left((P_l - P_{l-1})v_h, (P_k - P_{k-1})v_h\right) \\ &= a\left(P_l v_h, P_k v_h\right) - a\left(P_l v_h, P_{k-1} v_h\right) - a\left(P_{l-1} v_h, P_k v_h\right) + a\left(P_{l-1} v_h, P_{k-1} v_h\right) \\ &= a\left(P_l v_h, v_h\right) - a\left(P_l v_h, v_h\right) - a\left(P_{l-1} v_h, v_h\right) + a\left(P_{l-1} v_h, v_h\right) \\ &= 0. \end{aligned} \tag{4.131}$$

Then from (4.119), (4.129), and (4.128) we have

$$\begin{aligned} a(v_h, v_h) &= a\left(\sum_{l=1}^{L} v_l, \sum_{k=1}^{L} v_k\right) \\ &= \sum_{l=1}^{L}\sum_{k=1}^{L} a(v_l, v_k) \\ &= \sum_{l=1}^{L} a(v_l, v_l) \\ &\geq \frac{1}{12 \cdot 2^2 \cdot c} \sum_{l=1}^{L}\sum_{i=1}^{m_l} a(v_{il}, v_{il}), \end{aligned}$$

which gives (4.117).

To prove the first inequality of (4.115), we note by (4.119) and (4.120), that for $v_h \in V^0_h$,

$$\begin{aligned} a(v_h, v_h) &= a\left(v_h, \sum_{l=1}^{L}\sum_{i=l}^{m_l} v_{il}\right) \\ &= \sum_{l=1}^{L}\sum_{i=1}^{m_l} a(v_h, v_{il}) \\ &= \sum_{l=1}^{L}\sum_{i=1}^{m_l} a(\rho_{il} v_h, v_{il})\,, \end{aligned} \tag{4.132}$$

since $a(v_h, v_{il}) = a(\rho_{il} v_h, v_{il})$ by (4.109). The Cauchy-Schwarz inequality, (4.109), and (4.117) give

$$\begin{aligned} a(v_h, v_h) &\le \sum_{l=1}^{L}\sum_{i=1}^{m_l} a(\rho_{il} v_h, \rho_{il} v_h)^{\frac{1}{2}} a(v_{il}, v_{il})^{\frac{1}{2}} \\ &\le \left[\sum_{l=1}^{L}\sum_{i=1}^{m_l} a(\rho_{il} v_h, \rho_{il} v_h)\right]^{\frac{1}{2}} \left[\sum_{l=1}^{L}\sum_{i=1}^{m_l} a(v_{il}, v_{il})\right]^{\frac{1}{2}} \\ &\le \left[\sum_{l=1}^{L}\sum_{i=1}^{m_l} a(\rho_{il} v_h, v_h)\right]^{\frac{1}{2}} \sqrt{c}\, a(v_h, v_h)^{\frac{1}{2}} \\ &= a(\rho v_h, v_h)^{\frac{1}{2}} \sqrt{c}\, a(v_h, v_h)^{\frac{1}{2}}\,. \end{aligned} \tag{4.133}$$

Dividing both sides of (4.133) by $\sqrt{c}\, a(v_h, v_h)^{\frac{1}{2}}$ and squaring both sides, we get

$$\frac{1}{c} a(v_h, v_h) \le a(\rho v_h, v_h)\,. \tag{4.134}$$

The proof of the second inequality in (4.115) is as follows. By the Cauchy-Schwarz inequality, with $\rho_l = \sum_{i=1}^{m_l} \rho_{il}$,

$$a(\rho_l v_h, v_h) \le a(\rho_l v_h, \rho_l v_h)^{\frac{1}{2}} a(v_h, v_h)^{\frac{1}{2}}\,. \tag{4.135}$$

Then, using the fact that $\psi_{jl}(x) = 0$ for $x \in [x_{i-1}, x_{i+1}]$ if $|i-j| > 1$,

$$\begin{aligned} a(\rho_l v_h, \rho_l v_h) &= a\left(\sum_{i=1}^{m_l} \rho_{il} v_h, \sum_{j=1}^{m_l} \rho_{jl} v_h\right) \\ &= \sum_{i=1}^{m_l}\sum_{j=1}^{m_l} a(\rho_{il} v_h, \rho_{jl} v_h) \\ &= \sum_{i=1}^{m_l} \left[a(\rho_{il} v_h, \rho_{i-1,l} v_h) + a(\rho_{il} v_h, \rho_{i,l} v_h) + a(\rho_{il} v_h, \rho_{i+1,l} v_h)\right] , \end{aligned} \tag{4.136}$$

where we set $\rho_{0,l} = \rho_{m_l+1,l} = 0$. By the Cauchy-Schwarz inequality, and the inequality $\alpha\beta \le \frac{1}{2}(\alpha^2 + \beta^2)$, $\alpha, \beta \in \mathbb{R}$, we have

$$\begin{aligned} a(\rho_{il} v_h, \rho_{i\pm1,l} v_h) &\le a(\rho_{il} v_h, \rho_{il} v_h)^{\frac{1}{2}} a(\rho_{i\pm1,l} v_h, \rho_{i\pm1,l} v_h)^{\frac{1}{2}} \\ &\le \frac{1}{2}\left[a(\rho_{il} v_h, \rho_{il} v_h) + a(\rho_{i\pm1,l} v_h, \rho_{i\pm1,l} v_h)\right] . \end{aligned} \tag{4.137}$$

Hence (4.136), (4.137) and (4.109) imply

$$\begin{aligned} a(\rho_l v_h, \rho_l v_h) &\le 3\sum_{i=1}^{m_l} a\left(\rho_{il} v_h, \rho_{il} v_h\right) \\ &= 3\sum_{i=1}^{m_l} a\left(\rho_{il} v_h, v_h\right) \\ &= 3a(\rho_l v_h, v_h) . \end{aligned} \tag{4.138}$$

Using (4.135) and (4.138), we have

$$a(\rho_l v_h, v_h) \le \sqrt{3} a(\rho_l v_h, v_h)^{\frac{1}{2}} a(v_h, v_h)^{\frac{1}{2}} . \tag{4.139}$$

Dividing both sides of (4.139) by $a(\rho_l v_h, v_h)^{\frac{1}{2}}$ and squaring both sides, we obtain

$$a(\rho_l v_h, v_h) \le 3a(v_h, v_h) . \tag{4.140}$$

Finally, since $\rho = \sum_{i=1}^{L} \rho_l$, (4.140) implies

$$a(\rho v_h, v_h) = \sum_{l=1}^{L} a(\rho_l v_h, v_h) \leq 3La(v_h, v_h)\,. \tag{4.141}$$

□

(See Zhang [27] for better bounds, not involving the number of levels L.)

Corollary 4.7.4 *We have*

$$c_1(M_h v_h, v_h) \leq (A_h v_h, v_h) \leq c_2 L(M_h v_h, v_h)\,, \quad v_h \in V_h^0\,. \tag{4.142}$$

Proof: ρ is self-adjoint with respect to the a-inner product, since M_h^{-1} is self-adjoint with respect to the L^2-inner product. Moreover, ρ is positive definite with respect to the a-inner product, since M_h^{-1} is positive definite with respect to the L^2-inner product. Hence with $v_h = \rho^{-\frac{1}{2}} w_h$, Theorem 4.7.3, Lemma 4.7.2, and (4.9) give

$$\begin{aligned}
c_1 a(\rho^{-\frac{1}{2}} w_h, \rho^{-\frac{1}{2}} w_h) &\leq a(\rho^{\frac{1}{2}} w_h, \rho^{-\frac{1}{2}} w_h) \leq c_2 La(\rho^{-\frac{1}{2}} w_h, \rho^{-\frac{1}{2}} w_h)\,, \\
c_1 a(\rho^{-1} w_h, w_h) &\leq a(w_h, w_h) \leq c_2 La(\rho^{-1} w_h, w_h)\,, \\
c_1 a(A_h^{-1} M_h w_h, w_h) &\leq a(w_h, w_h) \leq c_2 La(A_h^{-1} M_h w_h, w_h)\,, \\
c_1(M_h w_h, w_h) &\leq (A_h w_h, w_h) \leq c_2 L(M_h w_h, w_h)\,.
\end{aligned} \tag{4.143}$$

□

The two dimensional case is virtually the same except A_h is the operator given by a finite element discretization to Poisson's equation in two dimensions, and the projections and domains are naturally altered.

Chapter 5

MULTISCALE PRECONDITIONING FOR INTEGRAL EQUATIONS

5.1 An Integral Representation for Laplace's Equation

The Laplace's equation can be solved numerically with the boundary element method. With this method, the problem is reformulated as an integral equation on the boundary of the domain using a fundamental solution and Green's theorem. This integral equation involves the solution and its normal derivative on the boundary only, and thus reduces the dimension of the problem by one. If Dirichlet boundary conditions are given, the BEM will be used to approximate the flux on the boundary. Then, an approximation at an interior point of the domain can be calculated using the given boundary data and the approximated flux data. In this application of the BEM, piecewise constant functions will be used to approximate the flux on the boundary of the square domain [8].

Let D be the unit square domain $(0,1) \times (0,1)$ and S be the boundary of D. We consider the problem

$$\begin{aligned} -\triangle u &= 0 \text{ in } D, \\ u &= g \text{ on } S. \end{aligned} \tag{5.1}$$

The fundamental solution is

$$G(p,q) = -\frac{1}{2\pi} \log r = -\frac{1}{2\pi} \ln r, \tag{5.2}$$

where $r = |p - q|$. For $q \in S$, n_q is the outward unit normal vector to S at q, and $v(q) = \dfrac{\partial u}{\partial n_q}$ is the outward normal derivative of u on S at q. Then, the integral equation for (5.1) is

$$\int_S \frac{\partial G}{\partial n_q}(p,q)g(q)dS_q + \frac{1}{2}g(p) \;=\; \int_S G(p,q)v(q)dS_q\,, \quad p \in S\,, \tag{5.3}$$

[2, 8]. (5.1) is a Fredholm equation of the first-kind, and yields a unique solution for v on S. We also have

$$u(p) = \int_S G(p,q)v(q)dS_q - \int_S \frac{\partial G}{\partial n_q}(p,q)g(q)dS_q\,, \quad p \in D\,, \tag{5.4}$$

[8], which allows us to approximate in D using v and g on S.

5.2 The Collocation Boundary Element Method

Collocation can now be used to approximate (5.3), to obtain a linear system of equations. Each side of S is divided into subintervals of length h. Then $S = \bigcup_{j=1}^{n} S_j$, where S_j is the jth subinterval of the partition and $n = \dfrac{4}{h}$. The subintervals are labeled 1 to n going clockwise around the square. For each j, let p_j be the midpoint of S_j. To approximate the boundary functions by a constant on each subinterval, we introduce

$$V_h = \{\text{space of piecewise constants on the partition of S with step-size } h\}\,. \tag{5.5}$$

Now let ψ_i be the piecewise constant function defined as follows:

$$\psi_i(q) = \begin{cases} 1\,, & q \in S_i\,, \\ 0\,, & \text{otherwise}\,. \end{cases} \tag{5.6}$$

Then, $\{\psi_i\}_{i=1}^n$ is a basis for V_h.

Equation (5.3) can be written as

$$\int_S G(p,q)v(q)dS_q = f(p)\,, \quad p \in S\,, \tag{5.7}$$

where

$$f(p) = \int_S \frac{\partial G}{\partial n_q}(p,q)g(q)dS_q + \frac{1}{2}g(p) \tag{5.8}$$

is known. Replacing v by $v_h \in V_h$,

$$v_h(q) = \sum_{j=1}^n v_j\psi_j(q)\,, \tag{5.9}$$

(5.7) becomes

$$\begin{aligned} \int_S G(p,q)\sum_{j=1}^n v_j\psi_j(q)dS_q &\approx f(p)\,, \quad p \in S\,, \\ \sum_{j=1}^n \left[\int_S G(p,q)\psi_j(q)dS_q\right] v_j &\approx f(p)\,, \quad p \in S\,. \end{aligned} \tag{5.10}$$

Taking $p = p_i$, for $i = 1, \ldots, n$, and using

$$\int_S G(p_i,q)\psi_j(q)dS_q = \int_{S_j} G(p_i,q)dS_q\,, \tag{5.11}$$

and replacing $\approx$ with $=$, we have

$$\sum_{j=1}^{n}\left[\int_{S_j} G(p_i,q)dS_q\right]v_j \;=\; f(p_i)\,,\quad i=1,\ldots,n\,. \tag{5.12}$$

This is the linear system

$$Lv = f\,, \tag{5.13}$$

where $L=\{L_{ij}\}_{i,j=1}^{n}$, $v=[v_1,\ldots,v_n]^T$, $f=[f_1,\ldots,f_n]^T$,

$$\begin{aligned} L_{ij} &= \int_{S_j} G(p_i,q)dS_q\,,\\ f_i &= f(p_i)=\int_S \frac{\partial G}{\partial n_q}(p_i,q)g(q)dS_q+\frac{1}{2}g(p_i)\,. \end{aligned} \tag{5.14}$$

To evaluate the necessary integrals, some properties of the kernel functions must be established [1]. Let $p=(p_x,p_y)$, $q=(q_x,q_y)$, and $n_q=[n_{qx},n_{qx}]^T$, then:

$$\begin{aligned} \frac{\partial G}{\partial r} &= -\frac{1}{2\pi r}\,,\\ \frac{\partial r}{\partial n_q} &= -\frac{(p_x-q_x)n_{qx}+(p_y-q_y)n_{qy}}{r}\\ \frac{\partial G}{\partial n_q} &= \frac{(p_x-q_x)n_{qx}+(p_y-q_y)n_{qy}}{2\pi r^2}\,. \end{aligned} \tag{5.15}$$

Thus, if L_{ii} is computed directly on an interval $S_i=[a_i,b_i]$ and p_{iz} equals p_{ix} or p_{iy} depending upon the side the interval is on,

$$\begin{aligned} L_{ij} &= \int_{S_j} -\frac{1}{2\pi}\log|p_i-q|dS_q\,,\quad i\neq j\\ L_{ii} &= -\frac{1}{2\pi}\left[(p_{iz}-a_i)\log|p_{iz}-a_i|-(p_{iz}-b_i)\log|p_{iz}-b_i|\right.\\ &\quad \left.-(p_{iz}-a_i)+(p_{iz}-b_i)\right]\,,\quad p_i\,,q_i\in S_i=[a_i,b_i]\,, \end{aligned} \tag{5.16}$$

$$f_i = \int_S \frac{(p_{ix} - q_{ix})n_{qx} + (p_{iy} - q_{iy})n_{qy}}{2\pi|p_i - q|^2} g(q)dS_q + \frac{1}{2}g(p_i)\,.$$

To compute the integrals necessary for the vector f, a three-point Gauss-Legendre quadrature rule is used, and for the matrix L the midpoint rule is used to produce a symmetric matrix. Tests showed that the integrals of f were sensitive to the integration scheme and thus the midpoint rule is inadequate. However, the elements of L were less sensitive, and the midpoint rule proved sufficient. Since the midpoint rule is used, the resulting matrix $\hat{L}$ is symmetric, since

$$\begin{aligned} \hat{L}_{ij} &= \int_{S_j} -\frac{1}{2\pi} \log|p_i - p_j| dS_q \\ &= \int_{S_j} -\frac{1}{2\pi} \log|p_j - p_i| dS_q \\ &= \hat{L}_{ji}\,. \end{aligned} \tag{5.17}$$

We expect $\hat{L}$ to be positive definite, although this has not been verified theoretically. Thus, PCG can be used to solve $\hat{L}v = \hat{f}$, where $\hat{L}$ and $\hat{f}$ are the approximations of L and f.

5.3 Multiscale Preconditioning

Our first attempt at the multiscale preconditioner for the matrix L was one similar to the preconditioner used in the finite element calculations (chapter 4). In other words, the action of a preconditioner of L, denoted by B, is computed by $w = B^{-1}z$, where z is a given vector.

To begin, a multilevel structure must be described. Each side of S is divided into subintervals of length $h_l = \frac{4}{2^l}$ for each level $l = 2, \ldots, L$, with $h_L = h$. Then

$S = \bigcup_{j=1}^{n_l} S_{jl}$, where S_{jl} is the jth subinterval of the partition on level l, and $n_l = \frac{4}{h_l}$. The subintervals are again labeled 1 to n_l going clockwise around the square. For each j on level l, let p_{jl} be the midpoint of S_{jl}. Let

$$V_{h_l} = \{\text{space of piecewise constants on the partition of S with step-size } h_l\}\,. \quad (5.18)$$

Now let ψ_{il} be the piecewise constant function defined by

$$\psi_{il}(q) = \begin{cases} 1\,, & q \in S_{il}\,, \\ 0\,, & \text{otherwise}\,. \end{cases} \quad (5.19)$$

Then, $\{\psi_{il}\}_{i=1}^{n_l}$ is a basis for V_{h_l}.

To begin the preconditioner discussion, consider finding the vector w such that $Lw = z$, where z is a given vector. Now, let $z_h \in V_h$ and $w_h \in V_h$. Then,

$$z_h = \sum_{j=1}^{n} z_j \psi_j\,, \quad w_h = \sum_{j=1}^{n} w_j \psi_j\,, \quad (5.20)$$

and the equivalent vector representations are

$$z = [z_1, \ldots, z_n]^T\,, \quad w = [w_1, \ldots, w_n]^T\,. \quad (5.21)$$

Lemma 5.3.1 *For the basis functions ψ_{il} defined in (5.19), we have*

$$\psi_{il}(q) = \psi_{2i-1,l+1}(q) + \psi_{2i,l+1}(q)\,, \quad q \neq p_{il}\,. \quad (5.22)$$

Proof: The proof of this lemma uses the definition (5.19) of ψ_{il}. Since $\psi_{i,l}$, $\psi_{2i-1,l+1}$, and $\psi_{2i,l+1}$ are piecewise constant functions, it is sufficient to verify (5.22) for $q =$

$p_{2i-1,l+1}$ and $p_{2i,l+1}$.

We begin by verifying (5.22) for $q = p_{2i-1,l+1}$. Using (5.19), we have

$$\psi_{i,l}(p_{2i-1,l+1}) = 1 = \psi_{2i-1,l+1}(p_{2i-1,l+1}) + \psi_{2i,l+1}(p_{2i-1,l+1})\,. \tag{5.23}$$

For $q = p_{2i,l+1}$, we have

$$\psi_{i,l}(p_{2i,l+1}) = 1 = \psi_{2i-1,l+1}(p_{2i,l+1}) + \psi_{2i,l+1}(p_{2i,l+1})\,. \tag{5.24}$$

□

Using (5.22), we obtain

$$z_h\psi_{il} = z_h\psi_{2i-1,l+1} + z_h\psi_{2i,l+1}\,. \tag{5.25}$$

Hence, for $w_{h_{l-1}} \in V_{h_{l-1}}$,

$$w_{h_{l-1}} = \sum_{i=1}^{n_{l-1}} \beta_{i,l-1}\psi_{i,l-1}\,, \tag{5.26}$$

we have

$$\begin{aligned} w_{h_{l-1}} &= \sum_{i=1}^{n_{l-1}} \beta_{i,l-1}\left(\psi_{2i-1,l} + \psi_{2i,l}\right) \\ &= \beta_{1,l-1}\left(\psi_{1,l} + \psi_{2,l}\right) + \beta_{2,l-1}\left(\psi_{3,l} + \psi_{4,l}\right) \\ &\quad + \ldots + \beta_{n_{l-1},l-1}\left(\psi_{n_l-1,l} + \psi_{n_l,l}\right) \\ &= \sum_{i=1}^{n_l} \beta_{i,l}\psi_{i,l}\,, \end{aligned} \tag{5.27}$$

where

$$\beta_{i,l} = \begin{cases} \beta_{\frac{i}{2},l-1}\,, & i = 2,4,\ldots,n_l\,, \\ \beta_{\frac{i+1}{2},l-1}\,, & i = 1,3,\ldots,n_l - 1\,. \end{cases} \tag{5.28}$$

From (5.25), and (5.28), we obtain the following algorithm for computing $w = B^{-1}z$, thinking of the vectors w and z in (5.21) in terms of w_h and z_h in (5.20). First compute

$$z_h\psi_{iL}\,, \quad i = 1,\ldots,n_L\,, \tag{5.29}$$

and set

$$w_{h_L} = \sum_{i=1}^{n_L} h_L^{-1} z_h \psi_{iL}\,. \tag{5.30}$$

Then

$$\begin{aligned} \text{for} \quad & l = L-1,\ldots,2 \\ & \text{for } i = 1,\ldots,n_l \\ & \quad z_h\psi_{il} = z_h\psi_{2i-1,l+1} + z_h\psi_{2i,l+1} \\ & \text{end} \\ & w_{h_l} = \sum_{i=1}^{n_l} h_l^{-1} z_h \psi_{il} \\ & \text{for } k = l+1,\ldots,L \\ & \quad \text{for } i = 1,\ldots,n_k \\ & \quad\quad \beta_{ik} = \begin{cases} \beta_{\frac{i}{2},k-1}\,, & i = 2,4,\ldots,n_k \\ \beta_{\frac{i+1}{2},k-1}\,, & i = 1,3,\ldots,n_k - 1 \end{cases} \\ & \quad \text{end} \\ & \quad \text{if } k = L \\ & \quad\quad w_{h_L} = w_{h_L} + \sum_{i=1}^{n_L} \beta_{iL}\psi_{iL} \end{aligned} \tag{5.31}$$

end

end .

We implement the preconditioning algorithm (5.31) using a tree structure. To set up the tree, let each node represent a midpoint on the corresponding level. The number of nodes in the tree on the finest level L must be $n = 2^L$. The coarsest level used was $l = 2$, or one point on each side. The elements are labeled 1 to n going clockwise around the square. Therefore, the last panel touches the first at the corner. Each node contains a level number l, a point number i, $F_{il} = z_h \psi_{il}$, $d_{il} = h_l^{-1} F_{il}$, a pointer to its parent (there is 1 parent per node), and a pointer to its children (each node has 2 children). The tree contains a pointer to the root of the tree and a pointer to the current node on the tree. Each parent has two children formed by dividing each element in half, and each child can only have one parent in this implementation of the tree. Therefore, no communication occurs between sides of the domain.

5.4 Multiscale Preconditioning Variant

The multiscale preconditioner (5.31) is of the form

$$w = \sum_{l=2}^{L} h_l^{-1} Q_l z \,, \tag{5.32}$$

where $z = [z_1, \ldots, z_n]$, $Q_l = R_{l+1}^l \cdots R_L^{L-1}$, and for $l = 2, \ldots, L-1$, R_{l+1}^l is the $n_l \times 2n_l$ matrix defined by

$$R_{l+1}^{l} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & 1 \end{bmatrix}. \tag{5.33}$$

On the finest level $Q_L = I$. However, it became apparent through numerical tests that the preconditioner given in (5.32) did not work. There is no communication across the levels, or the communication occurring is not accurate and was therefore limits the strength of the preconditioner. Thus, a different preconditioner was considered.

The BPX-form of a multiscale preconditioner [6] is as follows:

$$w = \sum_{l=2}^{L} h_l^{-1}(Q_l - Q_{l-1})z\,. \tag{5.34}$$

Thus, as a replacement for (5.31), the following preconditioner is used

$$w = \sum_{l=2}^{L} h_l^{-1}(\overline{Q}_l - \overline{Q}_{l-1})z\,, \tag{5.35}$$

where $\overline{Q}_l = \overline{R}_{l+1}^{l} \cdots \overline{R}_{L}^{L-1}$, and for $l = 2, \dots, L-1$, $\overline{R}_{l+1}^{l}$ is the $n_l \times 2n_l$ matrix defined by

$$\overline{R}_{l+1}^{l} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}. \tag{5.36}$$

The algorithm is as follows:

for $l = L-1, \ldots, 2$

for $i = 1, \ldots, n_l$

$$z_h \psi_{il} = \frac{1}{2} z_h \psi_{2i-1,l+1} + \frac{1}{2} z_h \psi_{2i,l+1}$$

end

$$w_{h_l} = \sum_{i=1}^{n_l} h_l^{-1} z_h \psi_{il}$$

end

for $l = L, \ldots, 3$

for $i = 1, \ldots, n_l$

$$z_h \psi_{il} = \begin{cases} z_h \psi_{il} - z_h \psi_{\frac{i}{2},l} & i = 2, 4, \ldots, n_l \\ z_h \psi_{il} - z_h \psi_{\frac{i+1}{2},l} & i = 1, 3, \ldots, n_l - 1 \end{cases}$$

end (5.37)

end

for $l = L-1, \ldots, 2$

for $k = l+1, \ldots, L$

for $i = 1, \ldots, n_k$

$$\beta_{ik} = \begin{cases} \beta_{\frac{i}{2},k-1}, & i = 2, 4, \ldots, n_k \\ \beta_{\frac{i+1}{2},k-1}, & i = 1, 3, \ldots, n_k - 1 \end{cases}$$

end

if $k = L$

$$w_{h_L} = w_{h_L} + \sum_{i=1}^{n_L} \beta_{iL} \psi_{iL}$$

end

end.

Chapter 6

COMPUTATIONAL INVESTIGATION

6.1 Finite Element Formulation of Poisson's Equation

6.1.1 One-Dimensional Results

Throughout this chapter, the test problems were programmed in C++ and ran on a 515 mHz, dual Pentium III machine, with 512 megabytes of RAM. The convergence results are all based on the PCG algorithm (3.32), with the convergence parameter $\epsilon = 10^{-5}$ in (3.18).

The preconditioner was tested on the finite element formulation of the one-dimensional Poisson's equation (4.1). The operator equation (4.11), or equivalently the linear system (4.19), was solved using the PCG algorithm described in (4.48), where the multiscale preconditioner is as outlined in (4.47). Since $b = \hat{f}$, selecting b is equivalent to selecting f_h. The results can be seen in the attached charts. Table 6.1 and Figure 6.1 show the number of iterations needed for the PCG method to converge with and without the multiscale preconditioner. The CG column corresponds to the computation with no preconditioner and the MSPCG column contains the result computed with the multiscale preconditioner. The preconditioner clearly offers a vast improvement. Table 6.1 and Figure 6.2 show the time for convergence for the PCG method (in seconds) with and without the preconditioner. Again, clearly as the size of the matrix grows, the time needed to solve with the preconditioner is compensated by the reduced number of iterations.

Num. of Points	CG Time	CG Num. of Iter.	MSPCG Time	MSPCG Num. of Iter.
1	0.0	1	0.01	1
3	0.0	2	0.0	2
7	0.0	4	0.0	4
15	0.0	8	0.0	8
31	0.0	16	0.0	11
63	0.0	32	0.01	14
127	0.0	64	0.04	15
255	0.02	128	0.09	17
511	0.07	256	0.23	19
1023	0.29	512	0.58	21
2047	1.27	1024	1.40	22
4095	5.09	2048	3.65	23
8191	21.57	4096	8.66	24
16383	95.82	8192	20.78	26
32767	467.12	16384	47.97	27

Table 6.1. Computational performance for the 1-D Poisson's equation with and without a multiscale preconditioner.

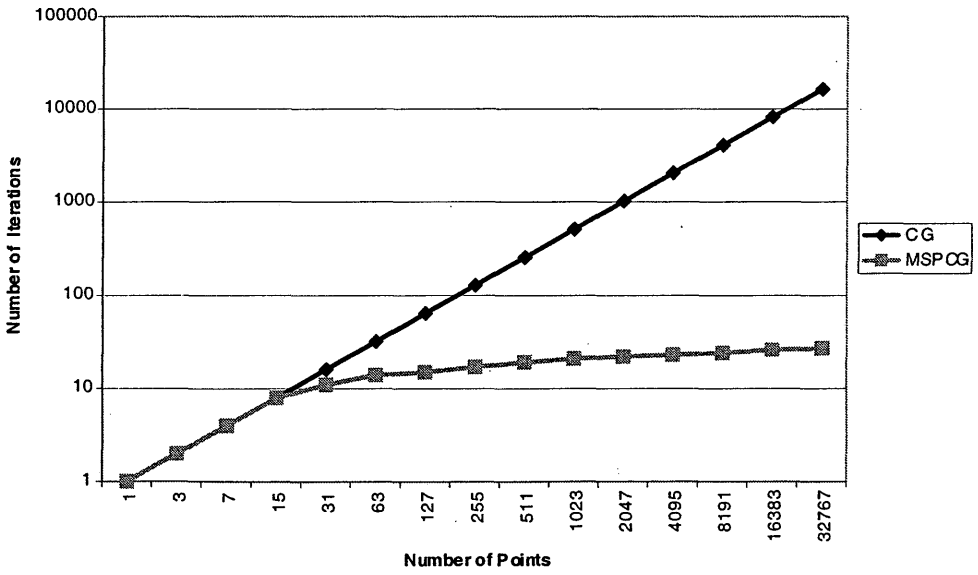

FIG. 6.1. Plot of the number of PCG iterations for the 1-D Poisson's equation with and without a multiscale preconditioner.

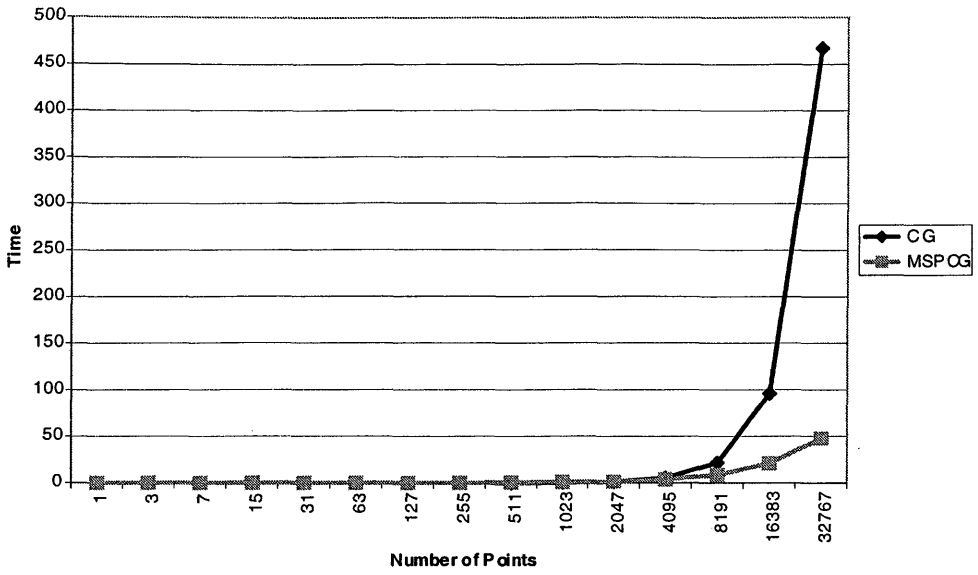

FIG. 6.2. Plot of the time results for the 1-D Poisson's equation with and without a multiscale preconditioner.

6.1.2 Two-Dimensional Results

The multiscale preconditioner was also implemented for the two-dimensional Poisson's equation (4.63). The operator equation (4.74), or equivalently the linear system (4.81), was solved using the PCG algorithm outlined in (4.48) with the multiscale preconditioner described in (4.103). The results can be seen in the attached charts (Table 6.2, Figure 6.3, and Figure 6.4). Table 6.2 and Figure 6.3 show the number of iterations needed for the PCG method to converge with and without the multiscale preconditioner. The preconditioner clearly offers a vast improvement. Again, the CG column is the computation with no preconditioner and the MSPCG column is the result computed with the multiscale preconditioner. Table 6.2 and Figure 6.4 show the time for convergence for the PCG method with and without the

Num. of Points	CG Time	CG Num. of Iter.	MSPCG Time	MSPCG Num. of Iter.
1	0.0	1	0.0	1
9	0.0	3	0.0	3
49	0.0	8	0.01	7
225	0.0	17	0.12	10
961	0.03	34	0.91	13
3969	0.25	69	5.80	15
16129	2.37	142	32.21	16
65025	24.11	288	180.44	18
261121	197.72	589	987.14	20

Table 6.2. Computational performance for the 2-D Poisson's equation with and without a multiscale preconditioner.

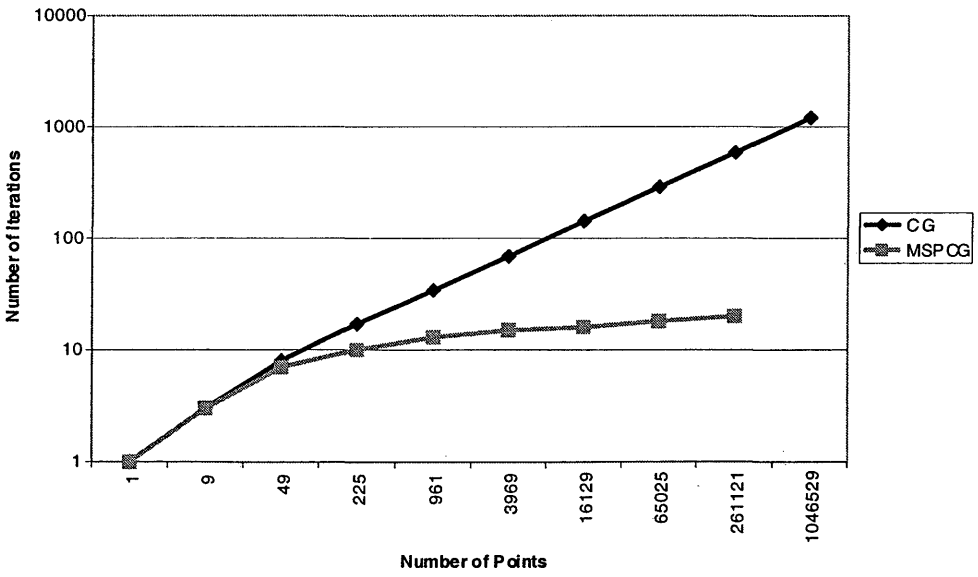

FIG. 6.3. Plot of the number of PCG iterations for the 2-D Poisson's equation with and without a multiscale preconditioner.

preconditioner. As the size of the matrix grows the time needed to solve with the preconditioner does not seem to be compensated by the reduced number of iterations. A better, more optimal, implementation of the multiscale preconditioner should yield a much improved computational performance.

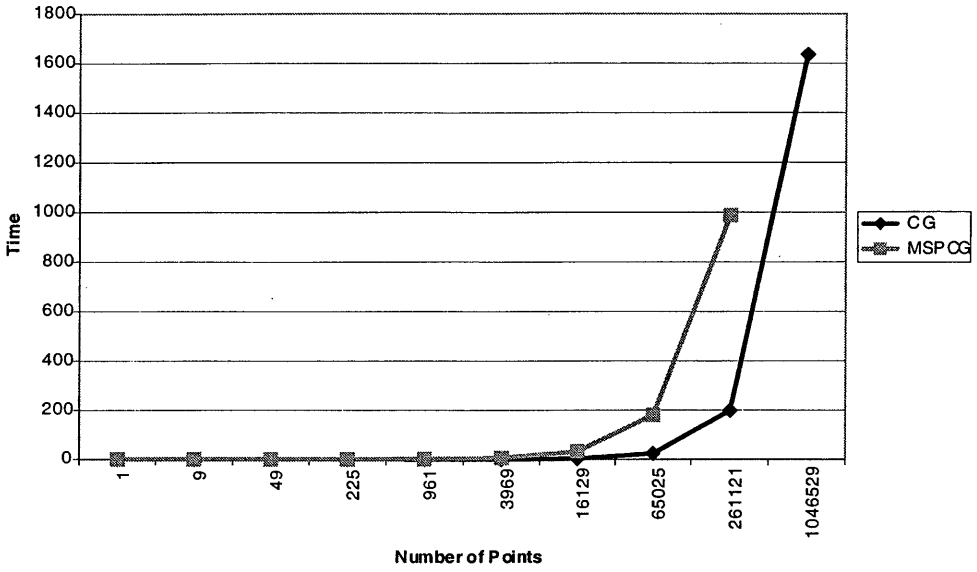

FIG. 6.4. Plot of the time results for the 2-D Poisson's equation with and without a multiscale preconditioner.

6.2 Collocation BEM Formulation of Laplace's Equation

The multiscale preconditioner was tested in two dimensions using the collocation BEM for the numerical solution to the integral formulation of Laplace's equation (5.1). The linear system (5.13) was solved using the PCG algorithm (3.32) where the multiscale preconditioner is as outlined in (5.37). The results can be seen in the attached charts. Table 6.3 and Figure 6.5 show the number of iterations needed for the

PCG method to converge with and without the multiscale preconditioner. As before, the CG column is the computation with no preconditioner and the MSPCG column is the computations with the multiscale preconditioner. The Old MSPCG line in Figure 6.5 shows the original, faulty, preconditioner (5.31), without the adjustment made in the argument from section 5.3. The preconditioner clearly offers an improvement over no preconditioner (Table 6.3, Figure 6.5). Table 6.3 and Figure 6.6 show the

Points	CG Time	CG Iter.	MSPCG Time	MSPCG Iter.	Old MSPCG Time	Old MSPCG Iter
8	0.0	3	0.0	3	0.0	3
16	0.0	5	0.0	4	0.0	5
32	0.0	8	0.0	8	0.0	8
64	0.01	11	0.02	10	0.01	11
128	0.02	15	0.04	13	0.05	15
256	0.11	17	0.17	15	0.20	21
512	0.61	23	0.61	16	0.87	25

Table 6.3. Computational performance for the 2-D Laplace's equation with and without the multiscale preconditioners.

time for convergence for the PCG method with and without the preconditioner. As the size of the matrix grows, the time needed to solve with the preconditioner is not compensated by the reduced number of iterations for small numbers of points. However, as the size of the matrix increases and with an optimal implementation, it should compensate (Table 6.3, Figure 6.6).

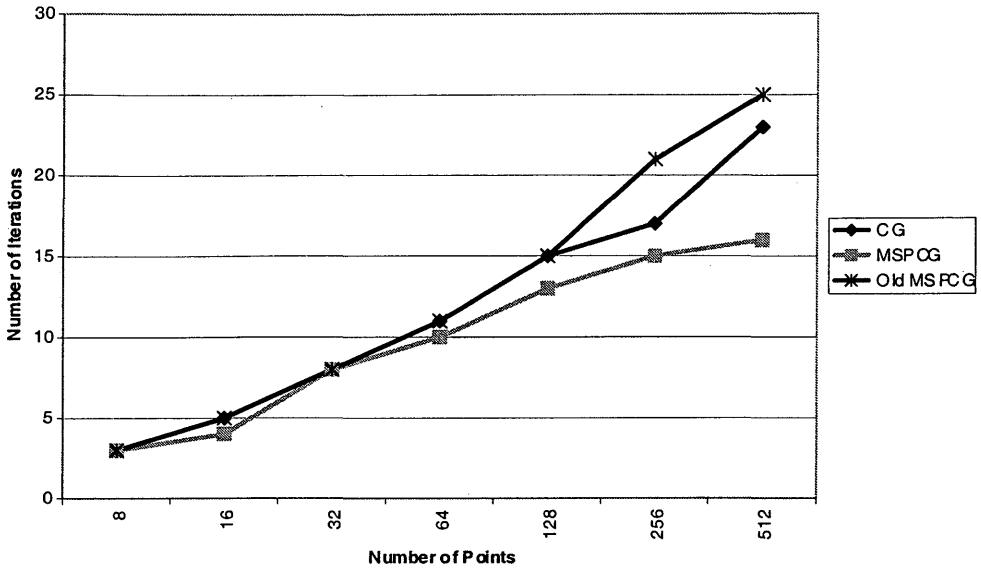

FIG. 6.5. Plot of the number of PCG iterations for the 2-D Laplace's equation with and without multiscale preconditioners.

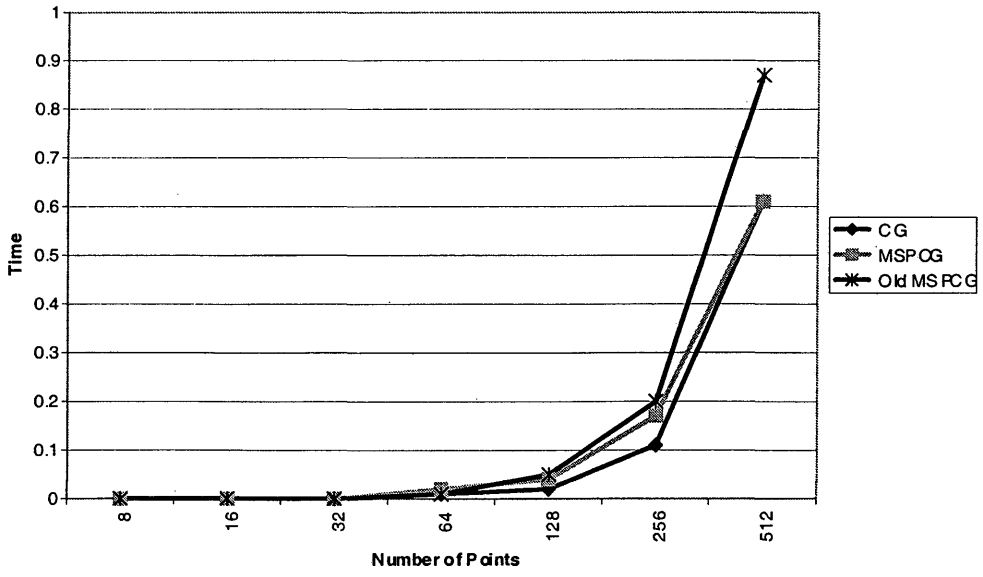

FIG. 6.6. Plot of the time results for the 2-D Laplace's equation with and without multiscale preconditioners.

Chapter 7

CONCLUSION AND DISCUSSION

7.1 Summary of Work Completed

The results from this thesis outline the importance of the multiscale preconditioner for both finite element and collocation boundary element methods for solving certain commonly occurring partial differential equations. As mentioned previously, preconditioning is an important technique in the solution of linear systems of equations using the preconditioned conjugate gradient method, or another iterative scheme. The multiscale preconditioner in this thesis is based on a decomposition of the finite element or boundary element space scaled differently for each nested level. Spectral results were outlined for the finite element formulation, and positive computational results were shown for this method in one and two dimensions on Poisson's equation. Although, the initial preconditioner attempted with the collocation boundary element method applied to Laplace's equation was flawed, once the adjustment based on the BPX-preconditioner was implemented, positive results, meaning a reduction in the number of iterations needed for convergence, were shown for this example also. Thus, this preconditioning method provides an important technique for a wide class of differential equations.

7.2 Future Work

Future work based on this thesis involves spectral justification for the collocation boundary element formulation, three dimensional extensions for the finite element and collocation boundary element formulations, and parallelizations for all the examples. All the examples and formulations can also be extended to nonuniform grids. The collocation boundary element formulation should be explored for Laplace's equation with Neumann and Robin boundary conditions. Finally, investigations, both theoretical and computational, should be done with the multiscale preconditioner on Helmholtz's equation.

REFERENCES

[1] *The Boundary Element Method (1994)*, www.boundary-element-method.com.

[2] K. Atkinson and W. Han, *Theoretical Numerical Analysis*, Springer–Verlag, New York, 2001.

[3] R. E. Bank and T. Dupont, *An optimal order process for solving finite element equations*, Math. Comp., 36 (1981), pp. 35–51.

[4] J. H. Bramble, *Multigrid Methods*, Cornell Mathematics Department Lecture Notes, Cornell University, New York, 1992.

[5] J. H. Bramble, J. E. Pasciak, J. Wang, and J. Xu, *Convergence estimates for product iterative methods with applications to domain decomposition*, Math. Comp., 57 (1991), pp. 1–22.

[6] J. H. Bramble, J. E. Pasciak, and J. Xu, *Parallel Multilevel Preconditioners*, Math. Comp., 55 (1990), pp. 1–22.

[7] S. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods*, Springer, New York, 1994.

[8] G. Chen and J. Zhou, *Boundary Element Methods*, Harcourt Brace Jovanovich, London, 1992.

[9] J. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[10] G. B. Folland, *Introduction to Partial Differential Equations*, Princeton University Press, Princetown, New Jersey, 1992.

[11] G. H. Golub and J. M. Ortega, *Scientific Computing and Differential Equations*, Harcourt Brace Jovanovich, Boston, 1992.

[12] G. H. Golub and C. F. Van Loan, *Matrix Computations, Third Edition*, John Hopkins University Press, Baltimore, 1996.

[13] B. H. Kleemann, A. Rathsfield, and R. Schneider, *Multiscale methods for boundary integral equations and their application to boundary value problems in scattering theory and geodesy*, Preprint series: Technische Universitat Chemnitz-Zwickau (Germany), SFB393–Preprint 96–17 (1996).

[14] P. Oswald, *On discrete norm estimates related to multilevel preconditioners in the finite element methods*, in Constructive Theory of Functions, K.G. Ivanov, ed., Varna 91 (1992), pp. 203–214.

[15] P. Oswald, *On a BPX preconditioner for P1 elements*, Computing, 51 (1993), pp. 125–133.

[16] A. D. Polyanin, *Handbook of Linear Partial Differential Equations for Engineers and Scientists*, Chapman and Hall/CRC, Boca Raton, 2002.

[17] G. Schmidlin, Project Leader: C. Schwab, *Fast algorithm for the discretization of integral equations*, www.math.ethz.ch/research/groups/sam/projects/schwab/schmidlin/.

[18] H. A. Schwarz, *Ueber einige Abbildungsaufgaben*, J. Reine Angew. Math., 70 (1869), pp. 105–120; Ges. Math. Abh., 2 (1890), pp. 65–83.

[19] J. R. Shewchuk, *An introduction to the CG method without the agonizing pain (1994)*, www-2.cs.cmu.edu/ jrs/jrspapers.htm/.

[20] J. Tausch and J. White, *Preconditioning and fast summation techniques for first-kind boundary integral equations*, Third IMAC Symposium on Iterative Methods in Scientific Computation, July 9–12, 1997.

[21] S. Vandewalle, *Multiscale methods for the solution of integral equations: matrix compression and preconditioning*, cwisdb.cc.kuleuven.ac.be/research/P/3E03/project3E030259.htm.

[22] S. Vavasis, *Preconditioning for Boundary Integral Equations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 905–925.

[23] K. F. Warnick and W. C. Chew, *Fast surface integral equation solvers for electromagnetic scattering*, www.ccem.uiuc.edu/reschew13.html.

[24] O. B. Widlund, *Some schwarz methods for symmetric and nonsymmetric elliptic problems*, in Domain Decomposition Methods for Partial Differential Equations, T.F. Chan, G. Meurant, J.S. Scroggs, R.G. Voight, eds., Society for Industrial and Applied Mathematics, Philadelphia, 1992, pp. 19–36.

[25] J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Review, 34 (1992), pp. 581–611.

[26] H. Yserentant, *Two preconditioners based on the multilevel splitting of finite element spaces*, Numer. Math., 58 (1990), pp. 163–184.

[27] X. Zhang, *Multilevel Schwarz Methods*, Numerische Mathematik, 63 (1992), pp. 521–539.