



Two-level preconditioners for regularized ill-posed problems
by Kyle Lane Riley

A thesis submitted in partial fulfillment of the requirements for the degree ' Doctor of Philosophy in
Mathematical Sciences
Montana State University
© Copyright by Kyle Lane Riley (1999)

Abstract:

The goal of this thesis is the solution of large symmetric positive definite linear systems which arise when Tikhonov regularization is applied to solve an ill-posed problem. The coefficient matrix for these systems is the sum of two terms. The first term comes from the discretization of a compact operator and is a dense matrix, i.e., not sparse. The second term, which is called the regularization matrix, is a sparse matrix that is either the identity or the discretization of a diffusion operator. In addition, the regularization matrix is scaled by a small positive parameter, which is called the regularization parameter. In practice, these systems are moderately ill-conditioned.

To solve such systems, we apply the preconditioned conjugate gradient algorithm with two-level preconditioners. These preconditioners were previously developed by Hanke and Vogel for positive definite regularization matrices. The contribution of this thesis is extension to the case where the regularization matrix is positive semidefinite. Also there is a compilation of the two-level preconditioning algorithms, and an examination of computational cost issues. To evaluate performance the preconditioners are applied to a problem from image processing known as image deblurring. Finally, there is a detailed numerical comparison of the performance between these two-level preconditioners and circulant preconditioning, which is a standard preconditioner from the problem of image deblurring.

TWO-LEVEL PRECONDITIONERS FOR
REGULARIZED ILL-POSED PROBLEMS

by

Kyle Lane Riley

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Doctor of Philosophy

in

Mathematical Sciences

MONTANA STATE UNIVERSITY-BOZEMAN
Bozeman, Montana

July 1999

D378
R4533

APPROVAL

of a thesis submitted by

Kyle Lane Riley

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

7/19/99
Date

Curt R. Vogel
Curt R. Vogel
Chairperson, Graduate Committee

Approved for the Major Department

7/19/99
Date

John Lund
John Lund
Head, Mathematical Sciences

Approved for the College of Graduate Studies

7-21-99
Date

Bruce A. McLeod
Bruce McLeod
Graduate Dean

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment for a doctoral degree at Montana State University-Bozeman, I agree that the Library shall make it available to borrowers under rules of the Library. I further agree that copying of this thesis is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U. S. Copyright Law. Requests for extensive copying or reproduction of this thesis should be referred to University Microfilms International, 300 North Zeeb Road, Ann Arbor, Michigan 48106, to whom I have granted "the exclusive right to reproduce and distribute copies of the dissertation for sale in and from microform or electronic format, along with the non-exclusive right to reproduce and distribute my abstract in any format in whole or in part."

Signature

Kyle Riley

Date

7/20/99

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Curtis Vogel for his guidance and patience. I would also like to thank Dr. Jack Dockery and Dr. Mark Pernarowski for their help in preparing this manuscript. Lastly, I would like to thank Melvin and Isa Riley for their everlasting support.

TABLE OF CONTENTS

	Page
LIST OF TABLES	vii
LIST OF FIGURES	viii
1. Introduction	1
2. Mathematical Preliminaries	6
Ill-posedness and Regularization	6
Discretization and Special Structures	19
Conjugate Gradient Algorithm and Preconditioning	24
3. Convolution and Circulant Preconditioning	30
Discrete Fourier Transforms	30
Algorithms for Two Dimensional Convolution	33
Circulant Preconditioning	36
4. Two-Level Preconditioners for Positive Definite L	39
Additive Schwarz Preconditioning	42
Symmetric Multiplicative Schwarz Preconditioning	43
Schur Complement Conjugate Gradient	45
5. Two-Level Preconditioners for a Semidefinite L	50
Additive Schwarz Preconditioning	53
Symmetric Multiplicative Schwarz Preconditioning	54
Schur Complement Conjugate Gradient	54
6. Numerical Results	56
The Test Problem	56
Implementation Details	57
Cost	63

Results	66
Summary	72

REFERENCES CITED	75
-----------------------------------	----

LIST OF TABLES

Table		Page
1	The computational costs, in flops, for the multilevel grid transfers involving Φ . Note the expression for m_p is given in (6.9) with p being the number restrictions it takes to get to the coarse grid.	64
2	Matrix computations per CG iteration for each of the two-level preconditioners when using a positive definite regularization matrix. . . .	65
3	Matrix computations per CG iteration for each of the two-level preconditioners when using a positive semidefinite regularization matrix.	65
4	Cost comparisons between the different algorithms using L^2 regularization with $\alpha = 2 \times 10^{-13}$. N denotes the number of piecewise constant coarse basis elements that were used to implement the two-level preconditioners. The solution error stopping tolerance was 10^{-9}	70
5	Cost comparisons between using piecewise linear and piecewise constant basis elements using L^2 regularization with $\alpha = 2 \times 10^{-13}$. N will denote the number of coarse grid basis elements. In the case of piecewise constant basis elements $N = 16^2$ and for the case of piecewise linear basis elements $N = 31^2$. The solution error stopping tolerance was 10^{-9}	71
6	Cost comparisons between the different algorithms using the discretized negative Laplacian with homogeneous Dirichlet boundary conditions where $\alpha = 10^{-12}$. The two-level preconditioners were implemented using 31^2 piecewise linear basis elements. The solution error stopping tolerance was 10^{-3}	71
7	This is the same cost comparison as in Table 6, except the solution error stopping tolerance is 10^{-9}	71
8	Cost comparisons between the different algorithms using the discrete negative Laplacian with homogeneous Neumann boundary conditions where $\alpha = 10^{-12}$. The two-level preconditioners were implemented using 15^2 piecewise linear coarse grid basis elements. The solution error stopping tolerance was 10^{-9}	72
9	Cost comparisons between the different algorithms using TV regularization with $\alpha = 10^{-13}$ and $\beta = .1$. The two-level preconditioners were implemented using 15^2 piecewise linear coarse grid basis elements. The solution error stopping tolerance was 10^{-9}	73

LIST OF FIGURES

Figure		Page
1	Simulated image Data from the Starfire Optical Range. At the top left is the true image. Noisy blurred observed data appears at the top right. At the lower left is the PSF. The lower right is a reconstruction using total variation regularization where $\alpha = 10^{-13}$ and $\beta = .1$	58
2	Performance of the different algorithms, showing relative solution error norm vs. iteration count. Pluses (+) represent additive Schwarz PCG; stars (*) represent plain CG; circles (o) represent circulant PCG; crosses (x) represent SMS PCG; and squares (\square) represent Schur CG. The results are from L^2 regularization with $\alpha = 2 \times 10^{-13}$. The two-level preconditioners were implemented with and 16^2 piecewise constant coarse grid basis functions.	68
3	Performance of two-level preconditioners with different number of piecewise constant coarse grid basis elements. The results are the solution error norm vs. iteration count for L^2 regularization with $\alpha = 2 \times 10^{-13}$. N will denote the number of coarse grid basis elements. Crosses (x) represent SMS PCG with $N = 16^2$; squares (\square) represent Schur CG with $N = 16^2$; stars (*) represent SMS PCG with $N = 32^2$; and diamonds (\diamond) represent Schur CG with $N = 32^2$	69
4	The effect of approximating L^\dagger on the convergence of Schur CG with a semidefinite L . The results given are normalized residuals vs. CG iteration counts. The regularization is with the discrete negative Laplacian with homogeneous Neumann boundary conditions with $\alpha = 5 \times 10^{-12}$ and with 15^2 piecewise linear coarse grid basis elements. Circles (o) represent 2 MG-CG iterations per PCG iteration; Stars (*) represent 5 MG-CG iterations per PCG iteration; and pluses (+) denote 10 MG-CG iterations per PCG iteration.	74

ABSTRACT

The goal of this thesis is the solution of large symmetric positive definite linear systems which arise when Tikhonov regularization is applied to solve an ill-posed problem. The coefficient matrix for these systems is the sum of two terms. The first term comes from the discretization of a compact operator and is a dense matrix, i.e., not sparse. The second term, which is called the regularization matrix, is a sparse matrix that is either the identity or the discretization of a diffusion operator. In addition, the regularization matrix is scaled by a small positive parameter, which is called the regularization parameter. In practice, these systems are moderately ill-conditioned.

To solve such systems, we apply the preconditioned conjugate gradient algorithm with two-level preconditioners. These preconditioners were previously developed by Hanke and Vogel for positive definite regularization matrices. The contribution of this thesis is extension to the case where the regularization matrix is positive semidefinite. Also there is a compilation of the two-level preconditioning algorithms, and an examination of computational cost issues. To evaluate performance the preconditioners are applied to a problem from image processing known as image deblurring. Finally, there is a detailed numerical comparison of the performance between these two-level preconditioners and circulant preconditioning, which is a standard preconditioner from the problem of image deblurring.

CHAPTER 1

Introduction

The goal of this thesis is the solution of large symmetric positive definite linear systems

$$A\mathbf{u} = \mathbf{b} \quad (1.1a)$$

where $\mathbf{b} = K^*\mathbf{z}$ and

$$A = K^*K + \alpha L. \quad (1.1b)$$

System (1.1) arises from the minimization of the functional

$$T(\mathbf{u}) = \|K\mathbf{u} - \mathbf{z}\|^2 + \alpha\mathbf{u}^*L\mathbf{u}. \quad (1.2)$$

Here K is an ill-conditioned matrix which results from the discretization of a compact operator. Typically K is a full matrix, i.e., not sparse. L is called the regularization matrix and is symmetric and sparse. Moreover, L may be positive definite or positive semidefinite. In this thesis, L will be either the identity or the discretization of a diffusion operator. The α is known as the regularization parameter. This parameter is positive and will typically be small. The functional in (1.2) is engendered from Tikhonov regularization applied to a compact operator equation

$$\mathcal{K}u = z.$$

Such equations can arise in many fields of study. A few examples are: bioelectric source imaging [20], tomography, signal processing, and image processing [2] among

others [22]. In order to provide a focus, we will concentrate our discussion on one particular problem from image processing, which is known as image deblurring [39].

In image deblurring \mathcal{K} is a two dimensional convolution operator given by

$$\mathcal{K}u(x, y) = \iint k(x - x', y - y')u(x', y')dx' dy'. \quad (1.3)$$

In this problem the kernel function k in (1.3) is referred to as the point spread function (PSF) and it mathematically describes the blurring of the light. The model for the observed image data is

$$z(x, y) = \mathcal{K}u_{true}(x, y) + \eta(x, y), \quad (1.4)$$

where u_{true} is the true image and η is an additive noise term. See Figure 1 in Chapter 6 for an illustration. The goal is to estimate the true image given the kernel function k and the noisy blurred data z . One complication with this type of problem is ill-posedness, i.e., the instability of the estimated u with respect to perturbations in the data z . To address the ill-posedness Tikhonov regularization will be used. The idea is to solve a penalized least squares problem,

$$\min \left(\frac{1}{2} \|\mathcal{K}u - z\|^2 + \frac{\alpha}{2} J(u) \right), \quad (1.5)$$

where J is the penalty term. The choice of penalty term reduces the presence of artifacts that arise from the ill-posedness. For example, the penalty term given by

$$J(u) = \iint \left(\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 \right) dx dy, \quad (1.6)$$

penalizes artifacts that are not smooth. For more information on regularization methods we refer to [16] and [31].

Discretization of (1.5) yields (1.2). With certain discretizations of the two dimensional convolution operator \mathcal{K} in (1.3), one obtains a matrix K which is block

Toeplitz with Toeplitz blocks. The term $\mathbf{u}^*L\mathbf{u}$ in (1.2) arises from the discretization of penalty terms like the one in (1.6). For details see Chapter 2. An important aspect to consider is that the matrix A in (1.1a) can be quite large. For example, the 256×256 pixel array in the image deblurring problem to be presented in Chapter 6 has a corresponding matrix that is $256^2 \times 256^2$. This means the matrix A will have over 4 billion entries. Moreover, this matrix tends to be poorly conditioned for very small values of α . The obstacles due to the large system size and large condition number of the matrix A have made solving (1.1) a very active area of research.

It is not computationally feasible to solve the large system in (1.1) using direct methods. Thus iterative methods are utilized. Recall that A is symmetric positive definite (SPD). Thus, the conjugate gradient (CG) algorithm [42] is an iterative method that will produce the solution to (1.1). However, due to the large condition number of A , the convergence of the CG algorithm is slow. Preconditioners can be used to accelerate the convergence of CG, yielding a method known as preconditioned conjugate gradient (PCG) [42]. Multigrid [6] and multilevel [36] methods are two other iterative techniques that have been applied to solve (1.1).

This thesis deals with two-level preconditioners that were developed by Hanke and Vogel in [26] and [45]. These preconditioners are related to the multilevel methods presented by Rieder. In [36] Rieder solves the system (1.1) for the case when L is the identity. Rieder introduces additive Schwarz and multiplicative Schwarz methods, which are closely related to the classical Jacobi and Gauss Seidel iterations respectively. He also analyzes the convergence of these methods when the regularization parameter α is relatively large. Hanke and Vogel adopt the two-level versions of these multilevel methods and utilize them as preconditioners for conjugate gradient. In [26] Hanke and Vogel provide a detailed convergence analysis under the more relevant con-

sideration of the asymptotic behavior when $\alpha \rightarrow 0$, and they consider more general SPD regularization matrices L . Implementation issues for these two-level preconditioners are dealt with in [45]. The main contribution of this thesis is the extension of the work of Hanke and Vogel to regularization matrices that are semidefinite. This thesis also contains a compilation of the two-level preconditioning algorithms, as well as the addition of an examination of computational cost issues. Also there is a detailed comparison of the performance between these two-level preconditioners and a standard preconditioner for a test problem in two dimensional image deblurring. Some of this work has appeared in [37], or has been submitted for publication [38].

A standard preconditioning technique for systems with Toeplitz structure is circulant preconditioning. These preconditioners all offer a relatively low computational cost for preconditioning since they can be implemented with fast Fourier transforms. The fast convergence rate of circulant preconditioning relies on the fact that circulant systems can be chosen "close" to Toeplitz systems. The idea was first introduced by Strang in [43]. The method of Strang's was later extended to block Toeplitz systems in the work of T. Chan and Olkin [13], and also in the work of R. Chan and X. Q. Jin [7]. The application of this preconditioning technique to Toeplitz least squares problems was carried out by R. Chan, Nagy, and Plemmons in [8]. T. Chan modified Strang's original algorithm so that given any matrix one could construct a circulant matrix that was closest in the Frobenius norm [11]. A good survey of preconditioning techniques for Toeplitz systems can be found in [9].

An outline of this thesis is as follows. Chapter 2 will briefly present the background mathematical material we will use for the subsequent chapters. Well-posedness will be defined and conditions that provide for the problem in (1.4) to be ill-posed will be established. Tikhonov regularization will be the tool that we will

use to compensate for the ill-posedness and solve this problem. Implementation of Tikhonov regularization and development of the regularization matrix L will be covered. An outline of the discretization of the convolution integral operator in (1.3) will also be produced. The chapter will conclude with a presentation of the conjugate gradient algorithm as well as the topic of preconditioning CG.

Chapter 3 will examine convolutions and provide a connection to the Fourier transform. The important role of the the fast Fourier transform will be established. An algorithm for circulant preconditioning of block Toeplitz systems will be the final topic of this chapter.

Chapters 4 and 5 will present the two-level preconditioners that are the main topic of this thesis. The implementation of the two-level preconditioners depends upon the matrix L . Chapter 4 presents the two-level preconditioning algorithms when a positive definite L is being used. Whereas, Chapter 5 discusses the alterations to the algorithms when using a positive semidefinite L .

Chapter 6 presents a numerical comparison of the various preconditioning algorithms for a test problem in image deblurring. Details of implementation will be covered and issues of computational cost will be discussed. Numerical results from applying the different PCG algorithms to a well referenced set of data will demonstrate the use of these algorithms in practice. The chapter will conclude with a summary of the results.

CHAPTER 2

Mathematical Preliminaries

The purpose of this chapter is to provide background and motivation for the system in (1.1). The first section will introduce notation that will be used in subsequent chapters. In addition the concepts of ill-posedness and regularization will be presented. The second section will examine the special structure that arises when discretizing convolution operators and regularization operators that will be used in Chapter 6. The final section will present the conjugate gradient and the preconditioned conjugate gradient algorithms.

Ill-posedness and Regularization

In this section \mathcal{H}_1 and \mathcal{H}_2 will denote separable Hilbert spaces with inner products $\langle \cdot, \cdot \rangle_i$, for $i = 1$ and 2 respectively. For these spaces the induced norm is defined by

$$\|u\|_i = \sqrt{\langle u, u \rangle_i}, \quad i = 1, 2,$$

where $u \in \mathcal{H}_i$. For this section, Ω will denote a simply connected, nonempty, measurable set in \mathbb{R}^n that has a piecewise Lipschitz continuous boundary. For smooth $u : \mathbb{R}^n \rightarrow \mathbb{R}^1$, define the *gradient* of u by

$$\nabla u = \left(\frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \dots, \frac{\partial u}{\partial x_n} \right).$$

For a vector valued function $\vec{v} = (v_1, v_2, \dots, v_n)$ where each $v_i : \mathbb{R}^n \rightarrow \mathbb{R}$ is smooth, we define the *divergence* of \vec{v} by

$$\nabla \cdot \vec{v} = \sum_{i=1}^n \frac{\partial v_i}{\partial x_i}.$$

If $\mathbf{u} \in \mathbb{R}^n$ then the *Euclidean norm* of \mathbf{u} is defined by

$$|\mathbf{u}| = \sqrt{\mathbf{u}^* \mathbf{u}}.$$

The following are three examples of Hilbert spaces that we will use in subsequent work.

Definition 2.1 The Hilbert space $L^2(\Omega)$ is made up of all measurable real-valued functions u such that $\int_{\Omega} u(x)^2 dx < \infty$. The L^2 inner product is denoted by

$$\langle u, v \rangle_{L^2} = \int_{\Omega} u(x)v(x)dx, \quad u, v \in L^2(\Omega). \quad (2.1)$$

The second Hilbert space makes use of the gradient operator.

Definition 2.2 The H^1 inner product of a pair of smooth functions is given by

$$\begin{aligned} \langle u, v \rangle_{H^1} &= \int_{\Omega} u(x)v(x)dx + \int_{\Omega} \nabla u \cdot \nabla v \, dx \\ &= \int_{\Omega} u(x)v(x)dx + \sum_{i=1}^n \int_{\Omega} \frac{\partial u}{\partial x_i} \frac{\partial v}{\partial x_i} dx. \end{aligned}$$

The Hilbert space $H^1(\Omega)$ is the completion of $C^\infty(\Omega)$, the space of infinitely continuously differentiable functions defined over Ω , under the norm induced by the H^1 inner product.

Definition 2.3 The Hilbert space $H_0^1(\Omega)$ is the completion of $C_0^\infty(\Omega)$, the space of infinitely continuously differentiable functions defined over Ω with compact support, in the space $H^1(\Omega)$ under the norm induced by the H^1 inner product.

In order to examine ill-posed problems we will first need to define well-posedness, which was a concept developed by Hadamard in [23]. Let \mathcal{K} represent a mapping from \mathcal{H}_1 to \mathcal{H}_2 .

Definition 2.4 The problem

$$\mathcal{K}u = z, u \in \mathcal{H}_1, z \in \mathcal{H}_2, \quad (2.2)$$

is *well-posed* if and only if the following three properties hold:

- i) A solution exists, i.e. for any $z \in \mathcal{H}_2$ there is a $u \in \mathcal{H}_1$ such that $\mathcal{K}u = z$.
- ii) This solution is unique.
- iii) The solution depends continuously on the data, that is, given $u^* \in \mathcal{H}_1$ and $z^* \in \mathcal{H}_2$ for which $\mathcal{K}u^* = z^*$, then for every $\epsilon > 0$ there exists $\delta(\epsilon) > 0$ such that when $\|\mathcal{K}u - z^*\|_2 < \delta(\epsilon)$ then $\|u - u^*\|_1 < \epsilon$.

A problem that is not well-posed is referred to as an *ill-posed* problem.

For the case when \mathcal{K} is linear, well-posedness is equivalent to the requirement that the inverse operator, $\mathcal{K}^{-1} : \mathcal{H}_2 \rightarrow \mathcal{H}_1$, exists and is bounded. For a linear \mathcal{K} , we will denote the range space by $R(\mathcal{K})$, the null space by $N(\mathcal{K})$, and the domain for the operator will be represented by $D(\mathcal{K})$. In addition to these spaces, we will also use their orthogonal complement.

Definition 2.5 If $E \subset \mathcal{H}_i$ then $u \in \mathcal{H}_i$ is an element of the *orthogonal complement* of E if and only if $\langle u, v \rangle_i = 0$ for all $v \in E$. The orthogonal complement will be denoted by E^\perp .

When dealing with a linear operator it will become necessary to employ the adjoint operator.

Definition 2.6 Let \mathcal{K} be a linear operator with a dense domain in \mathcal{H}_1 mapping into \mathcal{H}_2 . The *adjoint operator* $\mathcal{K}^* : \mathcal{H}_2 \rightarrow \mathcal{H}_1$ is a linear operator where for every $y \in D(\mathcal{K}^*)$ there exists a unique $y^* \in \mathcal{H}_1$ such that

$$\langle \mathcal{K}u, y \rangle_2 = \langle u, y^* \rangle_1,$$

for every $u \in D(\mathcal{K})$. The adjoint is defined by the mapping $\mathcal{K}^*y = y^*$ for all $y \in D(\mathcal{K}^*)$.

When \mathcal{K} is not 1-1 the pseudo-inverse can be used to resolve the non-uniqueness [16].

Definition 2.7 Let $\mathcal{K} : \mathcal{H}_1 \rightarrow \mathcal{H}_2$ be a continuous linear operator. Define $\tilde{\mathcal{K}}$ by $\tilde{\mathcal{K}} = \mathcal{K}|_{N(\mathcal{K})^\perp}$, thus $\tilde{\mathcal{K}}$ is 1-1 and maps onto the range of \mathcal{K} . The *pseudo-inverse*, \mathcal{K}^\dagger , is the linear extension of $\tilde{\mathcal{K}}^{-1}$ to the domain

$$D(\mathcal{K}^\dagger) = R(\mathcal{K}) + R(\mathcal{K})^\perp,$$

with minimal operator norm.

It is worth mention that for any $z_0 \in D(\mathcal{K}^\dagger)$ the pseudo-inverse gives the best least squares approximation to z_0 [16], i.e. $u_0 = \mathcal{K}^\dagger z_0$ satisfies

$$\|u_0\|_1 = \inf_{u \in D(\mathcal{K})} \left(\|u\|_1 \mid \|\mathcal{K}u - z_0\|_2 = \inf_{v \in D(\mathcal{K})} \|\mathcal{K}v - z_0\|_2 \right).$$

The problem in (2.2) is an ill-posed problem when \mathcal{K}^\dagger is unbounded. One type of operator that can have an unbounded pseudo-inverse is a compact operator.

Definition 2.8 An operator \mathcal{K} is *compact* if and only if the image of any bounded set is a relatively compact set. Note that a relatively compact set is a set whose closure is compact.

Example 2.1 An example of a compact operator is the Fredholm integral of the first kind on $L^2(\Omega)$. Suppose the function $k(x, y)$ is measurable on $\Omega \times \Omega$ and has the property

$$\int_{\Omega} \int_{\Omega} k(x, y)^2 dx dy < \infty. \quad (2.3)$$

Then the Fredholm integral operator of the first kind,

$$\mathcal{K}u(x) = \int_{\Omega} k(x, y)u(y)dy, \quad x \in \Omega, \quad (2.4)$$

is a compact operator with $\mathcal{K} : L^2(\Omega) \rightarrow L^2(\Omega)$. The function k in (2.4) is known as the *kernel* function for \mathcal{K} . Notice that (2.4) combined with (2.3) defines a Hilbert-Schmidt operator. The proof of compactness of a Hilbert-Schmidt operator can be found in [49, p. 277].

The following theorem establishes the conditions for which a compact operator has an unbounded pseudo-inverse. The proof can be found in [16, p. 38].

Theorem 2.9 For a compact operator whose range is infinite dimensional the pseudo-inverse operator is densely defined and unbounded.

With an unbounded pseudo-inverse the problem in (2.2) is ill-posed since it violates properties i) and iii) of Definition 2.4. A special tool that allows for further analysis of a compact linear operator is the singular value expansion [16].

Theorem 2.10 Let $\mathcal{K} : \mathcal{H}_1 \rightarrow \mathcal{H}_2$ be a linear operator. If \mathcal{K} then there exists positive values σ_n , with associated functions $\psi_n \in \mathcal{H}_1$, and $\phi_n \in \mathcal{H}_2$ such that $\mathcal{K}\psi_n = \sigma_n\phi_n$ and $\mathcal{K}^*\phi_n = \sigma_n\psi_n$. If \mathcal{K} has infinite dimensional range then $n = 1, 2, \dots$, otherwise there

will be finitely many terms. The functions $\{\psi_n\}$ are an orthonormal basis of $N(\mathcal{K})^\perp$, and the functions $\{\phi_n\}$ are an orthonormal basis of the closure of $R(\mathcal{K})$. Furthermore,

$$\mathcal{K}u = \sum_n \sigma_n \langle u, \psi_n \rangle_1 \phi_n \quad (2.5)$$

and

$$\mathcal{K}^*z = \sum_n \sigma_n \langle z, \phi_n \rangle_2 \psi_n, \quad (2.6)$$

for all $u \in \mathcal{H}_1$, and $z \in \mathcal{H}_2$. The expression in (2.5) is known as the *singular value expansion* of \mathcal{K} .

The σ_n 's are referred to as the *singular values* of \mathcal{K} . For a compact operator the singular values can be arranged in descending order, $\sigma_1 \geq \sigma_2 \geq \dots > 0$. The collection $\{\sigma_n; \phi_n, \psi_n\}$ is defined to be the *singular system* for \mathcal{K} . The singular system can also be used to give an expansion for \mathcal{K}^\dagger [16].

Theorem 2.11 Let \mathcal{K} have a singular system given by $\{\sigma_n; \phi_n, \psi_n\}$. Then the pseudo-inverse has the following expansion

$$\mathcal{K}^\dagger z = \sum_n \frac{\langle z, \phi_n \rangle_2}{\sigma_n} \psi_n, \quad (2.7)$$

for all $z \in D(\mathcal{K}^\dagger)$.

Theorem 2.11 give the singular value expansion for the pseudo-inverse. To illustrate ill-posedness we will need the following theorem [16].

Theorem 2.12 Let \mathcal{K} satisfy the hypothesis of Theorem 2.10 with the additional property that the range is infinite dimensional. Then

$$\lim_{n \rightarrow \infty} \sigma_n = 0.$$

The expansion in (2.7) reveals the lack of continuous dependence of the pseudo-inverse solution when the range of \mathcal{K} is infinite dimensional. If there is any error present in ϕ_n , ψ_n , or z , then that error may be greatly amplified by the division of the small singular values of \mathcal{K} .

The way to resolve ill-posedness is to use some type of regularization method. The idea of regularization is to approximate an ill-posed problem with one that is well-posed. We will use the following definition of a regularization operator [31].

Definition 2.13 A *regularization operator* for \mathcal{K} is a one parameter family of continuous operators $R_\alpha : \mathcal{H}_2 \rightarrow \mathcal{H}_1$ such that for $\mathcal{K}u_0 = z_0$ the following two conditions hold:

- i) There exist numbers α_0 and δ_0 such that $R_\alpha(z)$ is defined for all $0 < \alpha < \alpha_0$ and $\|z - z_0\|_2 < \delta_0$
- ii) There exists a function $\alpha(\delta)$ such that given any $\epsilon > 0$ there is a number $\delta(\epsilon) < \delta_0$ such that if $\|z - z_0\|_2 < \delta(\epsilon)$ then $\|u_\gamma - u_0\|_1 < \epsilon$ where $u_\gamma = R_\gamma(z)$ and $\gamma = \alpha(\delta(\epsilon))$.

Notice that when \mathcal{K} is linear then part i) of Definition 2.13 is equivalent to the requirement that R_α is bounded, and part ii) is equivalent to the requirement that $R_\alpha(z) \rightarrow \mathcal{K}^\dagger z$ as $\alpha \rightarrow 0$ for all $z \in D(\mathcal{K}^\dagger)$ [16]. There are many different regularization methods that can be applied to an ill-posed problem. For more details about regularization methods see [16], [21], and [44].

Example 2.2 An example of a regularization operator is the truncated singular value decomposition. Given an $\alpha > 0$ the truncated singular value decomposition is found

by

$$u_\alpha = R_\alpha(z) = \sum_{\{n \mid |\sigma_n| > \alpha\}} \frac{\langle z, \phi_n \rangle_2}{\sigma_n} \psi_n. \quad (2.8)$$

Notice that for any $\alpha > 0$ the expansion in (2.8) has only a finite number of terms.

Thus,

$$\begin{aligned} \|R_\alpha(z)\|_1 &= \sqrt{\sum_{\{n \mid |\sigma_n| > \alpha\}} \left| \frac{\langle z, \phi_n \rangle_2}{\sigma_n} \right|^2} \\ &\leq \frac{1}{\alpha} \|z\|_2, \end{aligned}$$

which implies R_α is bounded. In addition, as $\alpha \rightarrow 0$ the coefficients in (2.8) approach the same values as the expansion in (2.7), which establishes part ii) of Definition 2.13

The type of regularization that we will be using is Tikhonov regularization, which is also known as Tikhonov-Phillips regularization [34]. For the following examples we assume \mathcal{K} satisfies the assumptions in Theorem 2.10 and Theorem 2.12.

Example 2.3 In an abstract Hilbert space setting (i.e. $\mathcal{K} : \mathcal{H}_1 \rightarrow \mathcal{H}_2$), Tikhonov regularization is given by the following

$$u_\alpha = R_\alpha(z) = \arg \min_{u \in \mathcal{D}(\mathcal{K})} \left(\frac{1}{2} \|\mathcal{K}u - z\|_2^2 + \frac{\alpha}{2} \|u\|_1^2 \right), \quad (2.9)$$

where $\arg \min_{u \in \mathcal{D}(\mathcal{K})}$ denotes an element out of $\mathcal{D}(\mathcal{K})$ that obtains the minimum value for the given functional. This method can be thought of as penalized least squares with the second term in (2.9) being the penalty term. The regularization parameter, $\alpha > 0$, is used to assign a weight to the penalty term. Finding the u_α that minimizes the functional in (2.9) involves the use of calculus of variations [50, p. 45]. Consequently, we will need the first variation of the functional $F(u) = \frac{1}{2} \|\mathcal{K}u - z\|_2^2 + \frac{\alpha}{2} \|u\|_1^2$ in the

direction of $v \in \mathcal{H}_1$. The first variation is given by

$$\begin{aligned} \delta F(u; v) &\equiv \lim_{\tau \rightarrow 0} \left(\frac{F(u + \tau v) - F(u)}{\tau} \right) \\ &= \lim_{\tau \rightarrow 0} \frac{1}{2\tau} (\langle \mathcal{K}(u + \tau v) - z, \mathcal{K}(u + \tau v) - z \rangle_2 + \alpha \langle u + \tau v, u + \tau v \rangle_1) \\ &= \langle \mathcal{K}u - z, \mathcal{K}v \rangle_2 + \alpha \langle u, v \rangle_1. \end{aligned} \quad (2.10)$$

Using the adjoint in (2.10) and the fact that at the minimum the first variation must be zero for all $v \in \mathcal{H}_1$ engenders a representation for the solution to (2.9),

$$u_\alpha = (\mathcal{K}^* \mathcal{K} + \alpha I)^{-1} \mathcal{K}^* z. \quad (2.11)$$

If \mathcal{K} has the singular system $\{\sigma_n; \phi_n, \psi_n\}_{n=1}^\infty$ then the solution in (2.11) has a singular value expansion

$$u_\alpha = \sum_{n=1}^{\infty} \frac{\sigma_n}{\sigma_n^2 + \alpha} \langle z, \phi_n \rangle_2 \psi_n. \quad (2.12)$$

Using the condition that $\alpha > 0$, the expansion in (2.12), and the fact that both the ϕ_n 's and ψ_n 's are an orthonormal bases results in

$$\begin{aligned} \|R_\alpha(z)\|_1 &= \sqrt{\sum_{n=1}^{\infty} \left| \frac{\sigma_n}{\sigma_n^2 + \alpha} \langle z, \phi_n \rangle_2 \right|^2} \\ &\leq \frac{\sigma_1}{\alpha} \sqrt{\sum_{n=1}^{\infty} |\langle z, \phi_n \rangle_2|^2} \\ &\leq \frac{\sigma_1}{\alpha} \|z\|_2. \end{aligned}$$

Hence R_α is a bounded operator for each $\alpha > 0$. For any fixed n it is also true that $\lim_{\alpha \rightarrow 0} \frac{\sigma_n}{\sigma_n^2 + \alpha} = \frac{1}{\sigma_n}$, which matches the expansion in (2.7). Therefore, (2.9) defines a regularization operator.

From this point on, $\langle \cdot, \cdot \rangle$ will denote the $L^2(\Omega)$ inner product and $\|\cdot\|$ will represent the induced $L^2(\Omega)$ norm,

$$\|u\| = \sqrt{\int_{\Omega} u(x)^2 dx}.$$

We will assume \mathcal{K} is a compact linear operator such that $\mathcal{K} : L^2(\Omega) \rightarrow L^2(\Omega)$. For this thesis we will consider a slightly more general form of Tikhonov regularization,

$$u_\alpha = \arg \min \left(\frac{1}{2} \|\mathcal{K}u - z\|^2 + \frac{\alpha}{2} \langle \mathcal{L}u, u \rangle \right), \quad (2.13)$$

where \mathcal{L} , the regularization operator, is either the identity operator on $L^2(\Omega)$ or it is a differential operator whose domain is dense in $L^2(\Omega)$. The space for which we will minimize over depends on the regularization scheme that is being used. In the former case of Tikhonov regularization $\langle \mathcal{L}u, u \rangle = \|u\|^2$ and we will refer to this as L^2 regularization. The advantages of using the identity as a regularization operator are the ease of implementation, and the invertibility of the regularization operator. However, L^2 regularization does not penalize against solutions that have spurious oscillations. We next present a class of differential operators which are used to impose smoothness on regularized solutions.

For $u, v \in C_0^\infty(\Omega)$, consider the the quadratic regularization functional

$$\begin{aligned} J(u) &= \frac{1}{2} \int_{\Omega} \kappa(x) \sum_{i=1}^n \left(\frac{\partial u}{\partial x_i} \right)^2 dx \\ &= \frac{1}{2} \int_{\Omega} \kappa |\nabla u|^2 dx \end{aligned}$$

and the associated bilinear form

$$b(u, v) = \int_{\Omega} \kappa(x) \sum_{i=1}^n \frac{\partial u}{\partial x_i} \frac{\partial v}{\partial x_i} dx \quad (2.14)$$

$$= - \int_{\Omega} \nabla \cdot (\kappa(x) \nabla u) v dx \quad (2.15)$$

$$= \langle \mathcal{L}u, v \rangle, \quad (2.16)$$

where “ $\nabla \cdot$ ” in (2.15) denotes the divergence operator, and $\kappa \in C^1(\Omega)$ with $\kappa(x) \geq \kappa_0 > 0$ for all $x \in \Omega$. Integration by parts applied to (2.14) produces (2.15). From

(2.15) it follows that \mathcal{L} in (2.16) is a diffusion operator given by

$$\mathcal{L}u = -\nabla \cdot (\kappa(x)\nabla u) = -\sum_{i=1}^n \frac{\partial}{\partial x_i} \left(\kappa(x) \frac{\partial u}{\partial x_i} \right). \quad (2.17)$$

The operator \mathcal{L} has a well defined extension to $H_0^1(\Omega)$ and the condition that $\kappa \in C^1(\Omega)$ can be relaxed to $\kappa \in L^\infty(\Omega)$ [17]. With a domain of $H_0^1(\Omega)$ the associated boundary conditions are homogenous Dirichlet boundary conditions, that is, $u(x) = 0$ for $x \in \partial\Omega$. Note that $\delta J(u; v) = b(u, v)$. If $\kappa(x) = 1$ then (2.17) reduces to

$$\mathcal{L}u = -\nabla \cdot \nabla u = -\sum_{i=1}^n \frac{\partial^2 u}{\partial x_i^2}. \quad (2.18)$$

We will refer to this operator as the negative Laplacian with homogeneous Dirichlet boundary conditions. Note that \mathcal{L} is symmetric,

$$\langle \mathcal{L}u, v \rangle = \langle u, \mathcal{L}v \rangle \text{ for every } u, v \in H_0^1(\Omega),$$

and coercive,

$$\langle \mathcal{L}u, u \rangle \geq \gamma \|u\|^2 \text{ for every } u \in H_0^1(\Omega), \quad (2.19)$$

where $\gamma > 0$. With the domain of $H_0^1(\Omega)$ and the property (2.19) it follows that (2.13) is well-posed [50].

With the operator defined in (2.17) it is possible to extend the domain to $H^1(\Omega)$ [35]. In this case \mathcal{L} is symmetric and positive semidefinite,

$$\langle \mathcal{L}u, u \rangle \geq 0 \text{ for every } u \in H^1(\Omega).$$

With \mathcal{L} being semidefinite we will need extra criteria to make the problem in (2.13) well-posed. The *complementation condition* is defined in [16] as: for every $\alpha > 0$ there is a $\gamma > 0$ such that

$$\|\mathcal{K}u\|^2 + \alpha \langle \mathcal{L}u, u \rangle \geq \gamma \|u\|^2, \quad (2.20)$$

for all $u \in H^1(\Omega)$. With a domain of $H^1(\Omega)$ the associated boundary conditions are the homogeneous Neumann boundary conditions, i.e., $\nabla u \cdot \vec{\eta} = 0$ for all $x \in \partial\Omega$, where $\vec{\eta}$ denotes the outward unit normal vector. Note that the operator \mathcal{L} operator in (2.17) has a one dimensional null space consisting of all functions that are constant on Ω . When $\kappa = 1$, we will refer to this operator as the negative Laplacian with homogeneous Neumann boundary conditions.

To find the solution to (2.13) we once again use calculus of variations, see [33]. The minimizer to (2.13) solves the system

$$(\mathcal{K}^* \mathcal{K} + \alpha \mathcal{L}) u = \mathcal{K}^* z. \quad (2.21)$$

Condition (2.20) guarantees that $\mathcal{K}^* \mathcal{K} + \alpha \mathcal{L}$ has a bounded inverse. Regularization that uses either the definite, or semidefinite, negative Laplacian as a regularization operator penalizes against solutions which are not smooth. Unfortunately, this type of regularization does not preserve edges that are sometimes present in the true solution.

The final regularization scheme is the most complicated. Total variation (TV) as a regularization functional was introduced in [40] by Rudin, Osher, and Fatemi. In this case, the domain is the space of all functions with bounded variation [33]. First we will define the total variation of a function. This will be followed by the definition of the space of functions with bounded variation. The definitions we are using come from Giusti [18].

Definition 2.14 Let u be a real-valued function defined on Ω . The *total variation* of u is defined by to be

$$|u|_{\text{TV}} = \sup_{\vec{w} \in \mathcal{W}} \int_{\Omega} -u \nabla \cdot \vec{w} \, dx, \quad (2.22)$$

where

$$\mathcal{W} = \{ \vec{w} \in C_0^1(\Omega) : |\vec{w}(x)| \leq 1, \text{ for all } x \in \Omega \},$$

Definition 2.15 The space of functions with *bounded variation* on Ω is comprised of all functions u such that $\int_{\Omega} |u| dx < \infty$ and $|u|_{\text{TV}} < \infty$. We will denote this space by $\text{BV}(\Omega)$.

To motivate the choice of the regularization functional for total variation consider the case when $u \in C^1(\Omega) \cap \text{BV}(\Omega)$. Integration by parts applied to (2.22) yields

$$|u|_{\text{TV}} = \sup_{\vec{w} \in \mathcal{W}} \int_{\Omega} \vec{w} \cdot \nabla u dx. \quad (2.23)$$

If, in addition, $|\nabla u| \neq 0$ the supremum for (2.23) occurs when $\vec{w} = \frac{\nabla u}{|\nabla u|}$, resulting in

$$|u|_{\text{TV}} = \int_{\Omega} |\nabla u| dx. \quad (2.24)$$

Note that the Euclidean norm $|\cdot|$ is not differentiable at the origin. To overcome the resulting numerical difficulties, we use the modified TV functional,

$$J_{\beta}(u) = \frac{1}{2} \int_{\Omega} \sqrt{|\nabla u|^2 + \beta^2} dx, \quad \beta > 0. \quad (2.25)$$

Note that $\beta = 0$ implies that $J_{\beta}(u)$ is the same as (2.24). For smooth u and v , the first variation of J_{β} is given by

$$\begin{aligned} \delta J_{\beta}(u; v) &= \int_{\Omega} \frac{\nabla u \cdot \nabla v}{\sqrt{|\nabla u|^2 + \beta^2}} dx \\ &= - \int_{\Omega} \nabla \cdot (\kappa(u) \nabla u) v dx \\ &= \langle \mathcal{L}(u)u, v \rangle, \end{aligned} \quad (2.26)$$

under the proviso of homogeneous Neumann boundary conditions where

$$\kappa(u) = \frac{1}{\sqrt{|\nabla u|^2 + \beta^2}}.$$

The corresponding regularization operator has the form

$$\mathcal{L}(u)v = -\nabla \cdot (\kappa(u) \nabla v), \quad (2.27)$$

see [33]. Notice that the operator in (2.27) is symmetric and has the same null space as the negative Laplacian on $H^1(\Omega)$. Total variation penalizes against spurious oscillations. Moreover, this regularization scheme allows for jump discontinuities in a solution, as illustrated in [15] and [46]. A disadvantage with TV regularization is that the amount of computation involved is much greater than for the previous regularization schemes. One must solve

$$(\mathcal{K}^*K + \alpha\mathcal{L}(u))u = \mathcal{K}^*z, \quad (2.28)$$

where $\mathcal{L}(u)$ is given by (2.27). This regularization operator is nonlinear and has variable coefficients, which complicates implementation. A method for solving (2.28) is the “lagged diffusivity” fixed point method presented by Vogel and Oman in [46] and [47]. This iterative method is expressed by

$$(\mathcal{K}^*K + \alpha\mathcal{L}(u^\nu))u^{\nu+1} = \mathcal{K}^*z, \quad \nu = 0, 1, \dots,$$

where u^ν , the previous estimate for u , is used to form the regularization operator.

Discretization and Special Structures

In the remainder of this chapter Ω will denote a square region in \mathbb{R}^2 . The type of operator \mathcal{K} that we will be particularly concerned with is a two dimensional Fredholm integral operator of the first kind with convolution form,

$$\mathcal{K}u(x, y) = \iint_{\Omega} k(x - x', y - y')u(x', y')dx'dy'. \quad (2.29)$$

From Example 2.1 we know that if k is measurable on $\Omega \times \Omega$ and satisfies (2.3) then the convolution operator is compact. If the operator also has infinite dimensional range then by Theorem 2.9 the problem $\mathcal{K}u = z$ is ill-posed.

The matrix K that is generated by discretizing the two-dimensional convolution operator, (2.29), often has a block Toeplitz structure with Toeplitz blocks (BTTB). A Toeplitz matrix has the property that it is constant on the diagonals, i.e.

$$T = \begin{bmatrix} t_0 & t_1 & t_2 & \dots & t_{m-1} \\ t_{-1} & t_0 & t_1 & \dots & t_{m-2} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ t_{2-m} & & \ddots & \ddots & t_1 \\ t_{1-m} & t_{2-m} & \dots & t_{-1} & t_0 \end{bmatrix}.$$

Note that T is uniquely determined by the first row and column of the matrix. For this reason we define the *generator* of the Toeplitz matrix T to be the vector $\mathbf{t} = (t_{1-m}, t_{2-m}, \dots, t_0, \dots, t_{m-1})$. This relationship will be denoted by $T = \text{toeplitz}(\mathbf{t})$.

Another structured matrix that we will make use of is the circulant matrix. A circulant matrix is a Toeplitz matrix with the added property that each column "wraps around" to start its successor, i.e.,

$$C = \begin{bmatrix} c_0 & c_{n-1} & c_{n-2} & \dots & c_1 \\ c_1 & c_0 & c_{n-1} & \dots & c_2 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ c_{n-2} & & \ddots & \ddots & c_{n-1} \\ c_{n-1} & c_{n-2} & \dots & c_1 & c_0 \end{bmatrix}. \quad (2.30)$$

Note that the first column of a circulant matrix uniquely determines that matrix, hence the notation $C = \text{circulant}(c_0, c_1, \dots, c_{n-1})$, and $\mathbf{c} = [c_0, c_1, \dots, c_{n-1}]$ becomes the generator for C .

A BTTB matrix has the following structure

$$K = \begin{bmatrix} T_0 & T_1 & T_2 & \dots & T_{n-1} \\ T_{-1} & T_0 & T_1 & \dots & T_{n-2} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ T_{2-n} & & \ddots & \ddots & T_1 \\ T_{1-n} & T_{2-n} & \dots & T_{-1} & T_0 \end{bmatrix},$$

where each T_j is an $m \times m$ Toeplitz matrix. In this case the generator of the BTTB matrix is defined to be the $(2m - 1 \times 2n - 1)$ array $k = [\mathbf{t}_{1-n} \ \mathbf{t}_{2-n} \ \dots \ \mathbf{t}_0 \ \dots \ \mathbf{t}_{n-1}]$,

where $T_j = \text{toeplitz}(t_j)$ for each $j = 1 - n, 2 - n, \dots, n - 1$. Hence, the size of the matrix K is $nm \times nm$. We use the notation $K = \text{BTTB}(k)$ to represent the relationship between k and K . Similarly, the generator of a block circulant matrix with circulant blocks (BCCB) is given by the array $b = [c_0 \ c_1 \ \dots \ c_n]$, where c_j is the first column of the circulant block C_j . The analogous expression $\text{BCCB}(b)$ will represent the BCCB matrix generated by the array b .

Example 2.4 This example illustrates a discretization of a one dimensional convolution operator.

$$\mathcal{K}u(x) = \int_{\Omega} k(x - x')u(x')dx', \quad (2.31)$$

that results in a Toeplitz matrix. In this example we will assume $\Omega = [-p, p] \subset \mathbb{R}^1$. With a uniform grid spacing, $\Delta x = \frac{2p}{n}$, the midpoints are given by: $x_j = -p + (2j - 1)\frac{\Delta x}{2}$, $j = 1, 2, \dots, n$. By midpoint quadrature, the expression in (2.31) takes the form

$$\begin{aligned} \mathcal{K}u(x_i) &= \int_{-p}^p k(x_i - x')u(x')dx' \\ &\approx \sum_{j=0}^{m-1} k(x_i - x_j)u(x_j)\Delta x \\ &\approx [K\mathbf{u}]_i, \end{aligned} \quad (2.32)$$

where $K = \Delta x \text{toeplitz}[k((n-1)\Delta x), k((n-2)\Delta x), \dots, k(0), \dots, k((1-n)\Delta x)]$ and $\mathbf{u} = [u(x_1), u(x_2), \dots, u(x_n)]^*$.

The two dimensional case, see (2.29), is analogous with the block structure arising from the double integration [2].

Example 2.5 Assume $\Omega = [-p, p] \times [-p, p]$ With grid spacing $\Delta x = \frac{2p}{m}$ and $\Delta y = \frac{2p}{n}$, the midpoints are given by $x_i = -p + (2i-1)\frac{\Delta x}{2}$, $i = 1, 2, \dots, m$, and $y_r = -p + (2r-1)\frac{\Delta y}{2}$,

$r = 1, 2, \dots, m$. By midpoint quadrature it follows that

$$\begin{aligned} \mathcal{K}u(x_i, y_r) &= \iint_{\Omega} k(x_i - x', y_r - y')u(x', y')dx'dy' \\ &\approx \sum_{s=0}^{m-1} \sum_{j=0}^{n-1} k(x_i - x_j, y_r - y_s)u(x_j, y_s)\Delta x\Delta y \\ &\approx [K\mathbf{u}]_{i,r} \end{aligned} \quad (2.33)$$

The vector \mathbf{u} is formed using lexicographical ordering by rows, i.e.

$$\mathbf{u} = [u(x_1, y_1), u(x_2, y_1), \dots, u(x_n, y_1), u(x_1, y_2), u(x_2, y_2), \dots, u(x_n, y_n)]^*$$

The row ordering implies that $K = \Delta x\Delta y \text{ BTTB}([\mathbf{k}_{m-1}, \mathbf{k}_{m-2}, \dots, \mathbf{k}_{1-m}])$, where

$$\mathbf{k}_r = [k((n-1)\Delta x, r\Delta y), k((n-2)\Delta x, r\Delta y), \dots, k((1-n)\Delta x, r\Delta y)]^*.$$

For the regularization operators we will use a uniform version of the grid in Example 2.5 (that is, $\Delta x = \Delta y = \frac{2p}{n_x}$). We will gain the approximation

$$\langle \mathcal{L}u, u \rangle \approx \mathbf{u}^* L \mathbf{u}.$$

The vector \mathbf{u} results from enforcing lexicographical ordering by rows on the array u with $u_{i,j} = u(x_i, y_j)$. When discretized, each of the regularization operators of the previous section correspond to a matrix L that is sparse and symmetric. Recall that L^2 regularization gives rise to the identity as the regularization operator. Another definite regularization operator is the negative Laplacian with homogeneous Dirichlet boundary conditions. Using a second order finite difference approximation [42], it follows that the corresponding matrix has BTTB form

$$L = \frac{1}{(\Delta x)^2} \begin{bmatrix} M & -I_{n_x} & 0 & & \dots & 0 \\ -I_{n_x} & M & -I_{n_x} & 0 & & 0 \\ 0 & -I_{n_x} & M & -I_{n_x} & 0 & \dots \\ \vdots & \dots & & \ddots & \ddots & 0 \\ 0 & \dots & & 0 & -I_{n_x} & M \\ 0 & \dots & & & 0 & -I_{n_x} & M \end{bmatrix}, \quad (2.34)$$

where I_{n_x} is the $n_x \times n_x$ identity and each 0 in (2.34) represents a $n_x \times n_x$ matrix of zeros. The $n_x \times n_x$ submatrix M in (2.34) is given by

$$M = \begin{bmatrix} 4 & -1 & 0 & & \dots & 0 \\ -1 & 4 & -1 & 0 & & \dots & 0 \\ 0 & -1 & 4 & -1 & 0 & \dots & 0 \\ \vdots & \dots & & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & & 0 & -1 & 4 & -1 \\ 0 & \dots & & & 0 & -1 & 4 \end{bmatrix}$$

To approximate the negative Laplacian with Neumann boundary conditions [24], the finite difference representation changes slightly to

$$L = \frac{1}{(\Delta x)^2} \begin{bmatrix} M_0 & -I_{n_x} & 0 & & \dots & 0 \\ -I_{n_x} & M_1 & -I_{n_x} & 0 & & \dots & 0 \\ 0 & -I_{n_x} & M_1 & -I_{n_x} & 0 & \dots & 0 \\ \vdots & \dots & & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & & 0 & -I_{n_x} & M_1 & -I_{n_x} \\ 0 & \dots & & & 0 & -I_{n_x} & M_0 \end{bmatrix}$$

The Neumann boundary conditions change the diagonal blocks so that

$$M_0 = \begin{bmatrix} 2 & -1 & 0 & & \dots & 0 \\ -1 & 3 & -1 & 0 & & \dots & 0 \\ 0 & -1 & 3 & -1 & 0 & \dots & 0 \\ \vdots & \dots & & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & & 0 & -1 & 3 & -1 \\ 0 & \dots & & & 0 & -1 & 2 \end{bmatrix},$$

and

$$M_1 = \begin{bmatrix} 3 & -1 & 0 & & \dots & 0 \\ -1 & 4 & -1 & 0 & & \dots & 0 \\ 0 & -1 & 4 & -1 & 0 & \dots & 0 \\ \vdots & \dots & & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & & 0 & -1 & 4 & -1 \\ 0 & \dots & & & 0 & -1 & 3 \end{bmatrix}$$

Note that the matrix that approximates the negative Laplacian with Neumann boundary conditions is not BTTB and it has a one dimensional null space consisting of constant valued vectors.

Finally, the TV regularization operator, (2.27), can also be approximated with finite differences subject to lexicographical row ordering and homogeneous Neumann boundary conditions. The operator can be defined as an operator on the array u where $u_{i,j} = u(x_i, y_j)$. For the array operator we will use \bar{u} to represent the array from the previous fixed point iteration. The array operator will take the form:

$$L(\bar{u})u = \Delta_-(\kappa_+(\bar{u})(\Delta_+u)) \quad (2.35)$$

where

$$\begin{aligned} \kappa_+(\bar{u}) &= \frac{1}{\sqrt{(\Delta_+^x \bar{u})^2 + (\Delta_+^y \bar{u})^2 + \beta^2}} \\ \Delta_+ u &= (\Delta_+^x u + \Delta_+^y u) \\ \Delta_- u &= (\Delta_-^x u + \Delta_-^y u) \\ \Delta_\pm^x u &= \pm \left(\frac{u_{i\pm 1, j} - u_{i, j}}{\Delta x} \right) \\ \Delta_\pm^y u &= \pm \left(\frac{u_{i, j\pm 1} - u_{i, j}}{\Delta y} \right) \end{aligned}$$

The corresponding matrix L is symmetric positive semidefinite. It is also sparse with the same band structure as the discrete Laplacian, (2.34). In general, this matrix is not BTTB. Moreover, it has the same one dimensional null space as the negative Laplacian with homogeneous Neumann boundary conditions. In the discretization of both semidefinite regularization matrices the vector we will use as a basis vector for the null space is the vector comprised of all ones.

Conjugate Gradient Algorithm and Preconditioning

In this section we will present the CG algorithm, discuss convergence, and present the related topic of preconditioning. We refer to [1], [3], [19], and [42] for more details on these topics.

The goal is to solve $A\mathbf{u} = \mathbf{b}$ where A is a $n \times n$ matrix that is symmetric positive definite (SPD). Notice that solving $A\mathbf{u} = \mathbf{b}$ is the same as minimizing the functional

$$f(\mathbf{u}) = \frac{1}{2}\mathbf{u}^*A\mathbf{u} - \mathbf{u}^*\mathbf{b}, \quad (2.36)$$

since the gradient of f is given by

$$\nabla f(\mathbf{u}) = A\mathbf{u} - \mathbf{b},$$

and f is strictly convex. Given any $\mathbf{u} \in \mathbb{R}^n$ we will define the *residual* to be $\mathbf{r} = \mathbf{b} - A\mathbf{u}$. Notice that $\mathbf{r} = -\nabla f(\mathbf{u})$. The form of the algorithm that we will use is stated in Algorithm 2.1 [19, p. 527].

Algorithm 2.1 *Conjugate Gradient Method*

```

k = 0;
r0 = b - Au0;
δ0 = r0*r0;
Begin CG iterations
  if k = 0
    p1 = r0;
  else
    βk+1 = δk / δk-1;
    pk+1 = rk + βk+1pk;
  end
  dk+1 = Apk+1;
  αk+1 = δk / pk+1*dk+1;
  uk+1 = uk + αk+1pk+1;
  rk+1 = rk - αk+1dk+1;
  k = k + 1;
  δk = rk*rk;
end CG iterations
u = uk+1

```

The computational cost of the CG algorithm is dominated by one matrix-vector product involving A each iteration of CG. The other relatively costly feature is the

computation of the 2 inner products (e.g., $\delta_k = \mathbf{r}_k^* \mathbf{r}_k$). In Algorithm 2.1 \mathbf{u}_{k+1} is the approximation to the solution of $A\mathbf{u} = \mathbf{b}$ after k CG iterations. We will refer to the vector \mathbf{p}_k as the CG descent vector at the k -th iteration. The residuals \mathbf{r}_k and descent vectors adhere to certain properties [3, p. 466].

Theorem 2.16 Let \mathbf{u}_k , \mathbf{p}_k , and \mathbf{r}_k be the terms from the k -th iteration of Algorithm 2.1. If A is a $n \times n$ SPD matrix then the following three properties will hold.

- i) The residuals are mutually orthogonal, i.e., $\mathbf{r}_k^* \mathbf{r}_j = 0$, for $0 \leq j < k < n$.
- ii) $\mathbf{r}_k^* \mathbf{p}_j = 0$, for $0 \leq k < j < n$.
- iii) Descent vectors are A -conjugate, i.e., $\mathbf{p}_j^* A \mathbf{p}_k = 0$, for $1 \leq j \neq k < n$.

From property i) of Theorem 2.16 it follows that, in the absence of round-off error, the CG algorithm generates the exact solution in at most n iterations. By construction, the descent vectors are elements of a subspace of \mathbb{R}^n that is known as the Krylov subspace [42].

Definition 2.17 Let A represent a $n \times n$ matrix and \mathbf{r} be an element of \mathbb{R}^n . The k -th Krylov subspace is given by

$$\mathcal{S}_k(A, \mathbf{r}) = \text{span}(\mathbf{r}, A\mathbf{r}, \dots, A^{k-1}\mathbf{r}).$$

The CG descent vectors $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_k\}$ are elements of $\mathcal{S}_k(A, \mathbf{r}_0)$.

Since A is SPD the *energy norm* $\|\cdot\|_A$,

$$\|\mathbf{u}\|_A = \sqrt{\mathbf{u}^* A \mathbf{u}},$$

can be used in assessing convergence. Axelsson establishes the following result [3, p. 466].

Theorem 2.18 Let \mathbf{u}_* be the solution to $A\mathbf{u} = \mathbf{b}$, \mathbf{u}_0 be the initial guess in Algorithm 2.1, and \mathbf{u}_k denote the k -th CG iterate. Then

$$\|\mathbf{u}_* - \mathbf{u}_k\|_A \leq \|\mathbf{u}_* - \mathbf{w}\|_A,$$

for all $\mathbf{w} \in (\mathbf{u}_0 + S_k(A, \mathbf{r}_0))$.

Kelly uses Theorem 2.18 to prove the next result [28].

Theorem 2.19 Let \mathbf{u}_* be the solution to $A\mathbf{u} = \mathbf{b}$, \mathbf{u}_0 will represent the initial guess, and \mathbf{u}_k will denote the k -th CG iterate. Then the following bound holds

$$\|\mathbf{u}_* - \mathbf{u}_k\|_A = \min_{p \in P_k} \|p(A)(\mathbf{u}_* - \mathbf{u}_0)\|_A,$$

where P_k is the set of k degree polynomials with the property that $p(0) = 1$.

From Theorem 2.19 we can conclude that if A has m distinct eigenvalues then the CG algorithm will converge in at most m iterations. Kelly also uses Theorem 2.19 to explain rapid convergence when the eigenvalues of A cluster [28]. Saad uses the result in Theorem 2.19 to acquire the following bound on the convergence of the CG algorithm [42, p. 194].

Theorem 2.20 Let \mathbf{u}_* be the solution to $A\mathbf{u} = \mathbf{b}$, \mathbf{u}_0 be the initial guess, and \mathbf{u}_k denote the k -th CG iterate. Then

$$\|\mathbf{u}_* - \mathbf{u}_k\|_A \leq 2 \left[\frac{\sqrt{\text{cond}(A)} - 1}{\sqrt{\text{cond}(A)} + 1} \right]^k \|\mathbf{u}_* - \mathbf{u}_0\|_A, \quad (2.37)$$

where $\text{cond}(A) = \frac{\lambda_{\max}}{\lambda_{\min}}$ with λ_{\max} being the maximum eigenvalue of A and λ_{\min} is the minimum eigenvalue.

$\text{cond}(A)$ is called the spectral condition number for A . For brevity we will often refer to $\text{cond}(A)$ as the condition number of A . If the condition number is close to 1 then it is evident from (2.37) that the convergence for the CG algorithm will be fast. Recall from Chapter 1 that the system in (1.1) is poorly conditioned. In this case we will employ the technique of preconditioning to accelerate convergence of CG.

Preconditioning is based on applying the CG algorithm to the transformed system

$$C^{-1}AC^{-1}\mathbf{y} = C^{-1}\mathbf{b}, \quad (2.38)$$

where C is SPD and $\mathbf{y} = C\mathbf{u}$. By backtransforming with $\mathbf{u} = C^{-1}\mathbf{y}$ we can generate the solution to $A\mathbf{u} = \mathbf{b}$ from the solution to (2.38). If we use the backtransformation step in the CG algorithm for (2.38) and define $M = C^2$ then one can derive the preconditioned conjugate gradient (PCG) algorithm [19, p. 533].

Algorithm 2.2 *Preconditioned Conjugate Gradient Algorithm*

```

k = 0;
r0 = b - Au0;
Begin CG iterations
  Solve Mzk = rk
  δk = rk*zk;
  if k = 0
    p1 = z0;
  else
    βk+1 =  $\frac{\delta_k}{\delta_{k-1}}$ ;
    pk+1 = zk + βk+1pk;
  end
  dk+1 = Apk+1;
  αk+1 =  $\frac{\delta_k}{\mathbf{p}_{k+1}^* \mathbf{d}_{k+1}}$ ;
  uk+1 = uk + αk+1pk+1;
  rk+1 = rk - αk+1dk+1;
  k = k + 1;
end CG iterations
u = uk+1

```

Notice that C is never explicitly used in the actual implementation of PCG. We will refer to the step that involves solving $M\mathbf{z}_k = \mathbf{r}_k$ as the preconditioning step, and M will be called the *preconditioning matrix*. When comparing the computational cost of Algorithm 2.1 with Algorithm 2.2 the preconditioning step is the only added computation. There are two goals in preconditioning: i) Choose the preconditioner such that $C^{-1}AC^{-1}$ has a more desirable spectrum, e.g. small condition number or clustering of eigenvalues; and ii) the preconditioning step must be relatively inexpensive to implement. Note that $\text{cond}(C^{-1}AC^{-1}) = \text{cond}(M^{-1}A)$. Thus $\text{cond}(M^{-1}A)$ can be used in (2.37) to measure the effectiveness of a preconditioner.

CHAPTER 3

Convolution and Circulant Preconditioning

In this chapter we will introduce the discrete Fourier transform and discuss its connection to discrete convolutions and circulant matrices. In the second section, a few algorithms will be presented. One algorithm will be used to apply the discretized version of the operator in (2.29) using Fourier transforms. A subsequent algorithm will be developed that computes matrix-vector products involving a symmetric block Toeplitz matrix with Toeplitz blocks. The subject of the last section is circulant preconditioning.

Discrete Fourier Transforms

A valuable tool in signal and image processing is the discrete Fourier transform (DFT). In this section we will introduce the discrete Fourier transform. Following the introduction will be a discussion of the corresponding convolution theorem. To simplify the discussion, all of the theorems and proofs will involve vectors, $\mathbf{f} \in \mathbb{R}^n$. However, everything presented can be extended to two dimensional arrays, $f \in \mathbb{R}^{n \times n}$ [5].

Definition 3.1 The *discrete Fourier transform* is a mapping from \mathbb{C}^n to \mathbb{C}^n . Given a vector $\mathbf{f} = [f_0, f_1, \dots, f_{n-1}] \in \mathbb{C}^n$ the transform is denoted by

$$[\text{DFT}\{\mathbf{f}\}]_k \equiv [\mathbf{F}]_k = \sum_{j=0}^{n-1} f_j \exp\left(\frac{-2\pi i k j}{n}\right); \quad k = 0, 1, 2, \dots, n-1. \quad (3.1)$$

The DFT from (3.1) has a corresponding *inverse discrete Fourier transform* that will be defined by

$$[\text{IDFT}\{\mathbf{F}\}]_j = \frac{1}{n} \sum_{k=0}^{n-1} F_k e^{2\pi i \frac{jk}{n}}, \quad (3.2)$$

for any complex valued n -vector $\mathbf{F} = [F_0, F_1, \dots, F_{n-1}]$. Note that $\text{IDFT}\{\text{DFT}(\mathbf{f})\} = \mathbf{f}$ and $\text{DFT}(\text{IDFT}\{\mathbf{F}\}) = \mathbf{F}$ [5]. The convolution product of vectors $\mathbf{f} = \{f_k\}_{k=-n+1}^{n-1}$ ($\mathbf{f} \in \mathbb{C}^{2n}$) and $\mathbf{g} = \{g_j\}_{j=0}^{n-1}$ ($\mathbf{g} \in \mathbb{C}^n$) is a vector that is given by

$$[\mathbf{f} \star \mathbf{g}]_k \equiv \sum_{j=0}^{n-1} f_{k-j} g_j, \quad k = 0, 1, 2, \dots, n-1. \quad (3.3)$$

Given a vector $\mathbf{f} \in \mathbb{C}^n$ we define the n -periodic extension of \mathbf{f} as a sequence such that $f_{j+n} = f_j$ for any integer j . With the definitions above it is now possible to state the Convolution Theorem for discrete Fourier transforms.

Theorem 3.2 Let $\mathbf{f}, \mathbf{g} \in \mathbb{C}^n$ where \mathbf{f} has a n -periodic extension. The DFT of \mathbf{f} and \mathbf{g} will be denoted by \mathbf{F} and \mathbf{G} respectively. Then the DFT of the convolution between \mathbf{f} and \mathbf{g} has the property

$$[\text{DFT}\{\mathbf{f} \star \mathbf{g}\}]_k = F_k G_k, \quad (3.4)$$

for $k = 0, 1, 2, \dots, n-1$.

Proof:

To begin, we will use the fact that \mathbf{f} has a n -periodic extension, the definition of an inverse discrete Fourier transform in (3.2), and the definition of discrete convolution

in (3.3) to obtain the following:

$$\begin{aligned}
[\mathbf{f} \star \mathbf{g}]_k &= \sum_{j=0}^{n-1} \left[\left(\frac{1}{n} \sum_{r=0}^{n-1} G_r e^{2\pi i \frac{rj}{n}} \right) \left(\frac{1}{n} \sum_{m=0}^{n-1} F_m e^{2\pi i \frac{m(k-j)}{n}} \right) \right] \\
&= \sum_{j=0}^{n-1} \left[\left(\frac{1}{n} \sum_{r=0}^{n-1} G_r e^{2\pi i \frac{rj}{n}} \right) \left(\frac{1}{n} \sum_{m=0}^{n-1} F_m e^{2\pi i \frac{m(k-j)}{n}} \right) \right] \\
&= \frac{1}{n} \sum_{j=0}^{n-1} \left[\left(\sum_{r=0}^{n-1} G_r e^{2\pi i \frac{rj}{n}} \right) \left(\frac{1}{n} \sum_{m=0}^{n-1} F_m e^{-2\pi i \frac{mj}{n}} e^{2\pi i \frac{mk}{n}} \right) \right] \\
&= \frac{1}{n} \sum_{r=0}^{n-1} \sum_{m=0}^{n-1} G_r F_m e^{2\pi i \frac{mk}{n}} \left(\frac{1}{n} \sum_{j=0}^{n-1} e^{2\pi i \frac{rj}{n}} e^{-2\pi i \frac{mj}{n}} \right). \tag{3.5}
\end{aligned}$$

Now if $m = r$ then

$$\frac{1}{n} \sum_{j=0}^{n-1} e^{2\pi i \frac{rj}{n}} e^{-2\pi i \frac{mj}{n}} = 1,$$

or if $m \neq r$ then

$$\frac{1}{n} \sum_{j=0}^{n-1} e^{2\pi i \frac{rj}{n}} e^{-2\pi i \frac{mj}{n}} = 0.$$

This orthogonality property will reduce (3.5) to

$$[\mathbf{f} \star \mathbf{g}]_k = \frac{1}{n} \sum_{r=0}^{n-1} G_r F_r e^{2\pi i \frac{rk}{n}}. \tag{3.6}$$

Applying the DFT to both sides of (3.6) will result in the desired expression of (3.4). □

DFT's can be computed at a low computational cost using the fast Fourier transform (FFT). The FFT of an n -vector can be computed in $\mathcal{O}(n \log n)$ operations [32]. Next we will provide the connection between discrete convolution and circulant matrices. If $\mathbf{k} \in \mathbb{C}^{2n}$ is n -periodic and $\mathbf{u} \in \mathbb{C}^n$ then

$$\mathbf{k} \star \mathbf{u} = K\mathbf{u},$$

where $K = \text{circulant}(\tilde{\mathbf{k}})$ with $\tilde{\mathbf{k}}$ being the vector composed of the first n elements of \mathbf{k} . These tools will be put to use to develop the algorithms in the next section.

Algorithms for Two Dimensional Convolution

The algorithms developed in this section are for two dimensional arrays and will be utilized in Chapter 6. A major computational burden is the application of the discretized convolution operator, (2.33). The goal is to use two dimensional FFT's to exactly compute BTTB matrix-vector products. For an $n_x \times n_x$ image the corresponding matrix, given in Example 2.5, is a full BTTB matrix that is of size $n_x^2 \times n_x^2$. Fortunately, it is not necessary to construct the full matrix in order to compute the matrix-vector products.

First we will demonstrate how two dimensional FFT's can be used to compute applications of a block circulant matrix that has circulant blocks (BCCB). The impetus behind this is that BCCB matrix-vector multiplication can be performed with two dimensional FFT's [14]. Let C be an $n_x^2 \times n_x^2$ BCCB matrix with blocks that are of size $n_x \times n_x$. To compute a matrix-vector product involving C we will only need the first column, \mathbf{c}_1 . Recall from Chapter 2 that \mathbf{c}_1 is the generator for C . Define `array` to be the operation of converting an n_x^2 -vector into a $n_x \times n_x$ array via filling in by columns. For example, given $\mathbf{a} = [a_1, a_2, a_3, a_4]^*$ then

$$\text{array}(\mathbf{a}) = \begin{bmatrix} a_1 & a_3 \\ a_2 & a_4 \end{bmatrix}.$$

Let `vector` denote the reverse operation of taking an array and rearranging the elements back into a column vector. Finally, let $\hat{\mathbf{c}} = \text{FFT2}(\text{array}(\mathbf{c}_1))$, where FFT2 is the two dimensional FFT [48]. For a given vector $\mathbf{v} \in \mathbb{R}^{n_x^2}$ the matrix-vector product $C\mathbf{v}$ is computed by

$$C\mathbf{v} = \text{vector}(\text{IFFT2}(\text{FFT2}(\text{array}(\mathbf{v}).*\hat{\mathbf{c}}))), \quad (3.7)$$

where `.*` denotes component-wise multiplication and IFFT2 is the two dimensional inverse FFT.

The next algorithm computes an application of the convolution operator, (2.29), with FFT's. The method we will use is presented by Hanke and Nagy in [25]. The basic idea is to take the discrete kernel array, k from (2.29), and construct a larger array, k_{ext} , that will be a generator for a BCCB matrix. With k_{ext} we will use (3.7) to compute the matrix-vector products. Here n_x is an even integer so that the kernel array can be partitioned into blocks

$$k = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix},$$

with each block being of size $\frac{n_x}{2} \times \frac{n_x}{2}$. Recall that Theorem 3.2 demands vectors in the convolution to have n -periodic extensions. The analogous requirement for the same theorem to hold for arrays is that the arrays should have a doubly n -periodic extension [5]. An array f is *doubly n -periodic* when

$$f_{(i+n)j} = f_{ij} \quad \text{and} \quad f_{i(j+n)} = f_{ij},$$

where i and j are integers. In practice, the actual arrays that we will use are not necessarily doubly n -periodic. By extending the arrays with zeros it is possible to retain the convolution theorem for DFT's [5]. The extension for the kernel array is defined by

$$k_{ext} = \begin{bmatrix} k_{22} & 0 & k_{21} \\ 0 & 0 & 0 \\ k_{12} & 0 & k_{11} \end{bmatrix}, \quad (3.8)$$

where each 0 in (3.8) is a $\frac{n_x}{2} \times \frac{n_x}{2}$ array made up of zeros. Since k_{ext} is a generator for a BCCB matrix we will need the array $\hat{k}_{ext} = \text{FFT2}(k_{ext})$ to compute matrix-vector products. If r is an $n_x \times n_x$ array then it can be extended by the array operator `extend`,

$$\text{extend}(r) = \begin{bmatrix} r & 0 \\ 0 & 0 \end{bmatrix},$$

so that `extend`(r) is of size $2n_x \times 2n_x$. The array operator `restrict` simply restricts a $2n_x \times 2n_x$ array to the elements that are in the first n_x rows and n_x columns. With

Theorem 3.2 and the array operations defined as above we obtain

$$K\mathbf{r} = \text{vector}(\text{restrict}(\text{IFFT2}(\hat{k}_{ext} * \text{FFT2}(\text{extend}(\text{array}(\mathbf{r})))))). \quad (3.9)$$

Note that if K is BTTB, K^*K need not be BTTB. In order to ease computation and reduce storage for the numerical examples in Chapter 6 we will approximate K^*K by a BTTB matrix, T . In place of (1.1b) we will use

$$A = (T + \alpha L),$$

and solve (1.1a). With the same array operators it is possible to develop a similar technique for computing the matrix-vector products involving T . With k_{ext} given in (3.8) we will construct a $n_x \times n_x$ array t by

$$t = \text{restrict}(\text{IFFT2}(|\text{FFT2}(k_{ext})| \wedge 2)), \quad (3.10)$$

where $\wedge 2$ is component-wise squaring of each component of the array and $|\cdot|$ denotes the complex modulus of the components. With this array we are defining $t = \text{array}(\mathbf{t})$, where \mathbf{t} is the generator of T . The array in (3.10) can be embedded in an extended array, t_{ext} , that generates a BCCB matrix. For the $n_x \times n_x$ array let

$$t = [\mathbf{t}_1 \quad \mathbf{t}_2 \dots \mathbf{t}_{n_x}],$$

where \mathbf{t}_i is the i th column of \mathbf{t} . Define the array operator `embed` by extending t in the following way

$$\text{embed}(t) = [\mathbf{t}_1 \quad \mathbf{t}_2 \dots \mathbf{t}_{n_x} \quad \mathbf{t}_1 \quad \mathbf{t}_{n_x} \quad \mathbf{t}_{n_x-1} \dots \mathbf{t}_2].$$

The extension of the array t to a $2n_x \times 2n_x$ array is accomplished by the following

$$t_{ext} = (\text{embed}(\text{embed}(t)^*))^* \quad (3.11)$$

When arranged in vector form t_{ext} is the first column of a BCCB matrix that contains T in the first n_x^2 rows and the first n_x^2 columns. It follows that the computation of the matrix-vector product $T\mathbf{r}$ is simply

$$T\mathbf{r} = \text{vector}(\text{restrict}(\text{IFFT2}(\hat{t}_{ext} * \text{FFT2}(\text{extend}(\text{array}(\mathbf{r})))))), \quad (3.12)$$

where $\hat{t}_{ext} = \text{FFT2}(t_{ext})$. The expressions in both (3.9) and (3.12) have a double benefit. These matrix-vector products that involve $n_x^2 \times n_x^2$ matrices can be computed in $\mathcal{O}(n_x^2 \log(n_x))$ operations due to the FFT's and the Convolution Theorem. Moreover, all that is needed for computation is the first column of the matrix, which drastically decreases the amount of required storage.

Circulant Preconditioning

In 1986, Strang [43] proposed a circulant preconditioner for Toeplitz systems, and presented numerical results which showed very fast rates of convergence. It was later established that circulant matrices can approximate Toeplitz matrices closely, and that the preconditioned system is well conditioned [10]. In Chapter 6 circulant PCG will be used as a standard of comparison for the two-level preconditioners.

For a BTTB matrix T the circulant preconditioning step is carried out using two dimensional FFT's. In general, the first column of T is put into array form using `array`. The extended array, t_{ext} , is constructed by way of (3.11). Given a vector $\mathbf{r} \in \mathbb{R}^{n_x^2}$, Algorithm 3.1 shows how to compute $M_{circ}^{-1}\mathbf{r}$ using FFT's. Note that `./` denotes component-wise division.

Algorithm 3.1 *Circulant Preconditioning*

$$\mathbf{s} = M_{\text{circ}}^{-1} \mathbf{r}$$

```

 $r_{\text{ext}} = \text{extend}(\text{array}(\mathbf{r}));$ 
 $\hat{s}_{\text{ext}} = \text{FFT2}(r_{\text{ext}}) ./ \hat{t}_{\text{ext}};$ 
 $s_{\text{ext}} = \text{IFFT2}(\hat{s});$ 
 $\mathbf{s} = \text{vector}(\text{restrict}(s_{\text{ext}}));$ 

```

Recall that the matrix corresponding to the negative Laplacian with Neumann boundary conditions is not BTTB. Moreover, the matrix associated with TV regularization (that comes from (2.35)) is not BTTB. In this case, a BTTB approximation will be found [12]. To address this issue let us first consider the task of approximating a general $n \times n$ matrix A by a Toeplitz matrix. The Toeplitz approximation that we will use is from a technique developed by T. Chan and Mulet [12]. This technique gives the Toeplitz matrix T that is closest to A in terms of the Frobenius norm. Let a_{ij} denote the entries of A and the generator of T will be denoted by $\mathbf{t} = [t_1, t_2, \dots, t_n]$. The computation of \mathbf{t} is given by

$$\begin{aligned}
 t_1 &= \frac{\sum_{k=1}^n a_{k,k}}{n} \\
 t_j &= \frac{\sum_{k=1}^{n-j+1} (a_{(k+j-1),k} + a_{k,(k+j-1)})}{2(n-j+1)}; \quad 1 < j < n \\
 t_n &= \frac{a_{n,1} + a_{1,n}}{2}.
 \end{aligned}$$

This method generalizes to matrices with block structure. The sparsity structure of the regularization matrix, corresponding to the regularization operators in Chapter 2, allows the approximation to be determined by computing just three elements. Here we will denote a_{ij} as the entry in the regularization matrix that is in the i -th row and j -th column. With these specific matrices, the generator of the BTTB approximation

is given by

$$\begin{aligned}
 t_1 &= \frac{\sum_{k=1}^{n_x^2} a_{k,k}}{n_x^2} \\
 t_2 &= \frac{\sum_{j=0}^{n_x-1} \sum_{k=2}^{n_x} a_{(jn_x+k),(jn_x+k-1)}}{n_x(n_x-1)} \\
 t_{n_x} &= \frac{\sum_{j=1}^{n_x^2-n_x} (a_{(n_x+j),j})}{n_x(n_x-1)},
 \end{aligned}$$

with the rest of the elements for the generator being zero.

CHAPTER 4

Two-Level Preconditioners for Positive Definite L

Recall that the focus of this thesis is to solve

$$A\mathbf{u} = \mathbf{b}, \quad (4.1a)$$

where

$$A = K^*K + \alpha L, \quad (4.1b)$$

and $\mathbf{b} = K^*\mathbf{z}$ is a vector in \mathbb{R}^n . The implementation for the two-level preconditioners is different depending on whether L is definite or semidefinite. In this chapter, it is assumed that L is symmetric positive definite (SPD). The following chapter will address the case when L is symmetric positive semidefinite.

The approach taken will be to split the solution space, \mathbb{R}^n , into two subspaces. Let the “coarse grid” subspace, V_{Φ} , have a basis $\{\phi_i\}_{i=1}^N$ with $1 \leq N < n$. The second subspace will be given by $V_{\Psi} = \text{span}(\{\psi_j\}_{j=1}^{n-N})$. In principle, the bases for both subspaces can be chosen such that for every appropriate i and j ,

$$\phi_i^* L \psi_j = 0, \quad (4.2a)$$

$$\psi_i^* L \psi_j = \delta_{ij}, \quad (4.2b)$$

where δ_{ij} is the Kronecker delta. Two subspaces are L -orthogonal when property (4.2a) holds, where $\{\phi_i\}$ is the basis for the first subspace and $\{\psi_j\}$ is the basis for the second subspace. It follows that V_{Φ} and V_{Ψ} are L -orthogonal and we will sometimes

refer to V_Ψ as the L -orthogonal complement of V_Φ . As a consequence of (4.2) and L being SPD, $\{\phi_i\}_{i=1}^N \cup \{\psi_j\}_{j=1}^{n-N}$ forms a basis for \mathbb{R}^n .

Example 4.1 This is a one-dimensional example of the type of bases that can be used for the two-level methods in this chapter. Assume the solution space is \mathbb{R}^4 and the regularization matrix is the identity, $L = I$. An appropriate set of basis elements would be:

$$\phi_1 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}; \quad \phi_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}; \quad \psi_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \end{pmatrix}; \quad \psi_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1 \end{pmatrix}.$$

Notice that $\phi_i^* L \psi_j = 0$ and $\psi_i^* L \psi_j = \delta_{ij}$ for $i = 1, 2$ and $j = 1, 2$.

Let Φ and Ψ represent the matrices whose columns are composed of the ϕ_i 's and ψ_j 's respectively. The matrices Φ and Ψ can be adjoined to form the nonsingular matrix $[\Phi \Psi]$ that will transform the system (4.1) into

$$[\Phi \Psi]^* A [\Phi \Psi] \mathbf{x} = [\Phi \Psi]^* \mathbf{b},$$

where

$$\mathbf{x} = [\Phi \Psi]^{-1} \mathbf{u}. \quad (4.3)$$

The transformed system has a block representation given by

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \Phi^* A \Phi & \Phi^* A \Psi \\ \Psi^* A \Phi & \Psi^* A \Psi \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \Phi^* \mathbf{b} \\ \Psi^* \mathbf{b} \end{bmatrix}. \quad (4.4)$$

The system in (4.4) can be rewritten using the basis properties in (4.2), which results in

$$\begin{bmatrix} \Phi^* A \Phi & \Phi^* K^* K \Psi \\ \Psi^* K^* K \Phi & \Psi^* K^* K \Psi + \alpha I_{n-N} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \Phi^* \mathbf{b} \\ \Psi^* \mathbf{b} \end{bmatrix}, \quad (4.5)$$

where I_{n-N} is the $(n-N) \times (n-N)$ identity matrix. We assume that Ψ can be picked in such a way so that $\Psi^*K^*K\Psi$ is small. A sketch of how it is possible for $\Psi^*K^*K\Psi$ to be small is in the following. Consider the case when \mathcal{K} is given by (1.3). If \mathcal{K} has smooth kernel and u is highly oscillatory then $\mathcal{K}u$ is very small [27]. Thus, if the basis elements for V_Ψ are highly oscillatory then $K\Psi$ will be small. We refer to [26] for technical details. Using this property it follows that the block matrix in (4.5) can be approximated by

$$\tilde{A} = \begin{bmatrix} \Phi^*A\Phi & \Phi^*K^*K\Psi \\ \Psi^*K^*K\Phi & \alpha I_{n-N} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & \tilde{A}_{22} \end{bmatrix}. \quad (4.6)$$

For implementation of the two-level methods, define the operators G , P , and Q by

$$G = \Phi^*L\Phi \quad (4.7a)$$

$$P = \Phi G^{-1}\Phi^*L \quad (4.7b)$$

$$Q \equiv I - P = \Psi\Psi^*L. \quad (4.7c)$$

Lemma 4.1 The operator P is an L -orthogonal projector onto V_Φ and Q is the complementary L -orthogonal projector onto V_Ψ .

Proof: From direct computation it follows that $P^2 = P$, $Q^2 = Q$, and $P^*LQ = 0$. For any $\check{y} \in V_\Phi$ there exist $\mathbf{y} \in R^N$ such that $\check{y} = \Phi\mathbf{y}$. To compute $P\check{y}$ the equations in (4.7) are used and this results in $P\check{y} = \check{y}$. Moreover,

$$Q\check{y} = (I - P)\check{y} = 0,$$

which implies that V_Φ is in the null space of Q . For any $\tilde{\mathbf{x}} \in V_\Psi$ let $\mathbf{x} \in R^{n-N}$ be an element such that $\tilde{\mathbf{x}} = \Psi\mathbf{x}$. Using (4.2a) results in

$$P\tilde{\mathbf{x}} = \Phi G^{-1}\Phi^*L\Psi\mathbf{x} = 0.$$

On the other hand, using the basis properties in (4.2b) produces

$$Q\tilde{\mathbf{x}} = (\Psi\Psi^*L)\tilde{\mathbf{x}} = \Psi\mathbf{x} = \tilde{\mathbf{x}}.$$

By construction V_Φ and V_Ψ are L -orthogonal subspaces, thus P and Q are L -orthogonal projection operators [30].

□

With use of the projection operators it is possible to implement the two-level methods without ever constructing Ψ , or choosing any of the ψ_j 's. To illustrate how this is possible, let us consider the additive Schwarz preconditioner for the system (4.1).

Additive Schwarz Preconditioning

Additive Schwarz preconditioning can be derived as follows: (i) Transform the system in (4.1a) to get (4.4); (ii) Replace the block matrix in (4.4) by the block matrix \tilde{A} in (4.6); (iii) Apply block Jacobi preconditioning [42, p. 103] to the resulting approximate system; and finally (iv) Backtransform using (4.3). It is important to note that \tilde{A} differs from the coefficient matrix in (4.5) by only the block that is in the last row and column, i.e.,

$$\tilde{A}_{22} = \alpha I_{n-N}. \quad (4.8)$$

Let \tilde{D} , \tilde{L} , and \tilde{U} be defined by the following

$$\tilde{L} = \begin{bmatrix} 0 & 0 \\ A_{21} & 0 \end{bmatrix} \quad \tilde{D} = \begin{bmatrix} A_{11} & 0 \\ 0 & \tilde{A}_{22} \end{bmatrix} \quad \tilde{U} = \begin{bmatrix} 0 & A_{12} \\ 0 & 0 \end{bmatrix}. \quad (4.9)$$

Thus $\tilde{A} = \tilde{D} + \tilde{L} + \tilde{U}$. Let $\mathbf{u}_{AS} = M_{AS}^{-1}\mathbf{r}$ denote the result of applying the additive Schwarz preconditioner to a vector \mathbf{r} , i.e.,

$$\mathbf{u}_{AS} = M_{AS}^{-1}\mathbf{r} = [\Phi\Psi]\tilde{D}^{-1}[\Phi\Psi]^*\mathbf{r}.$$

From (4.9) this requires solving the block diagonal system

$$\begin{aligned} A_{11}\mathbf{x}_1 &= \Phi^*\mathbf{r} \\ \alpha\mathbf{x}_2 &= \Psi^*\mathbf{r}, \end{aligned}$$

and computing

$$\mathbf{u}_{AS} = \Phi\mathbf{x}_1 + \Psi\mathbf{x}_2. \quad (4.10)$$

Using the equations in (4.7) the following transpires:

$$\begin{aligned} \Psi\mathbf{x}_2 &= \frac{1}{\alpha}\Psi\Psi^*LL^{-1}\mathbf{r} \\ &= \frac{1}{\alpha}(I - P)L^{-1}\mathbf{r} \\ &= \frac{1}{\alpha}(L^{-1}\mathbf{r} - \Phi G^{-1}\Phi^*\mathbf{r}). \end{aligned} \quad (4.11)$$

It follows that the preconditioning step, without any explicit use of Ψ , is given by

$$M_{AS}^{-1}\mathbf{r} = \Phi(A_{11}^{-1}\Phi^*\mathbf{r} - \frac{1}{\alpha}G^{-1}\Phi^*\mathbf{r}) + \frac{1}{\alpha}L^{-1}\mathbf{r}. \quad (4.12)$$

Each iteration of the Additive Schwarz PCG method is dominated by a matrix-vector product involving A and an inversion of A_{11} , G , and L . This method is most effective for very diagonally dominant systems, because the Jacobi preconditioner uses only the diagonal of \tilde{A} . When \tilde{A} has significant off-diagonal terms there is a need for a different iterative method that takes these terms into consideration.

Symmetric Multiplicative Schwarz Preconditioning

To derive the symmetric multiplicative Schwarz preconditioner, it is necessary to replace the block Jacobi preconditioning step of additive Schwarz with that of block

symmetric Gauss Seidel preconditioning [42, p. 103]. The result of applying the preconditioner to a vector, $\mathbf{r} \in \mathbb{R}^n$, can be represented by

$$\mathbf{u}_{\text{SMS}} = M_{\text{SMS}}^{-1}\mathbf{r} = [\Phi\Psi](\tilde{D} + \tilde{U})^{-1}\tilde{D}(\tilde{D} + \tilde{L})^{-1}[\Phi\Psi]^*\mathbf{r}, \quad (4.13)$$

where \tilde{D} , \tilde{L} , and \tilde{U} are given in (4.9). Define

$$\tilde{\mathbf{r}} = [\Phi\Psi]^*\mathbf{r} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix}. \quad (4.14)$$

Then $\mathbf{r}_1 \in \mathbb{R}^N$ and $\mathbf{r}_2 \in \mathbb{R}^{n-N}$, and

$$\begin{aligned} (\tilde{D} + \tilde{L})^{-1}\tilde{\mathbf{r}} &= \begin{bmatrix} A_{11}^{-1}\mathbf{r}_1 \\ \tilde{A}_{22}^{-1}(\mathbf{r}_2 - A_{21}A_{11}^{-1}\mathbf{r}_1) \end{bmatrix} \\ \tilde{D}(\tilde{D} + \tilde{L})^{-1}\tilde{\mathbf{r}} &= \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 - A_{21}A_{11}^{-1}\mathbf{r}_1 \end{bmatrix} \\ (\tilde{D} + \tilde{U})^{-1}\tilde{D}(\tilde{D} + \tilde{L})^{-1}\tilde{\mathbf{r}} &= \begin{bmatrix} A_{11}^{-1}(\mathbf{r}_1 - A_{12}\tilde{A}_{22}^{-1}(\mathbf{r}_2 - A_{21}A_{11}^{-1}\mathbf{r}_1)) \\ \tilde{A}_{22}^{-1}(\mathbf{r}_2 - A_{21}A_{11}^{-1}\mathbf{r}_1) \end{bmatrix}. \end{aligned} \quad (4.15)$$

Let \mathbf{x}_1 and \mathbf{x}_2 denote the components of the right hand side of (4.15). From (4.14), (4.4), and (4.8) it follows that

$$\begin{aligned} \mathbf{x}_1 &= A_{11}^{-1}(\Phi^*\mathbf{r} - \Phi^*K^*K\Psi\frac{1}{\alpha}(\Psi^*\mathbf{r} - \Psi^*K^*K\Phi A_{11}^{-1}\Phi^*\mathbf{r})) \\ \mathbf{x}_2 &= \frac{1}{\alpha}(\Psi^*\mathbf{r} - \Psi^*K^*K\Phi A_{11}^{-1}\Phi^*\mathbf{r}). \end{aligned}$$

Finally, as in (4.10), we backtransform to obtain $\mathbf{u}_{\text{SMS}} = \Phi\mathbf{x}_1 + \Psi\mathbf{x}_2$. To simplify $\Psi\mathbf{x}_2$, the projection operators P and Q are used in the same manner as in (4.11).

Consequently,

$$\begin{aligned} \Psi\mathbf{x}_2 &= \frac{1}{\alpha}\Psi\Psi^*(\mathbf{r} - K^*K\Phi A_{11}^{-1}\Phi^*\mathbf{r}) \\ &= \frac{1}{\alpha}(L^{-1} - \Phi G^{-1}\Phi^*)(\mathbf{r} - K^*K\Phi A_{11}^{-1}\Phi^*\mathbf{r}). \end{aligned}$$

Simplification of the expression for $\Phi\mathbf{x}_1$,

$$\begin{aligned} \Phi\mathbf{x}_1 &= \Phi A_{11}^{-1}(\Phi^*\mathbf{r} - \Phi^*K^*K\Psi\frac{1}{\alpha}(\Psi^*\mathbf{r} - A_{21}A_{11}^{-1}\Phi^*\mathbf{r})) \\ &= \Phi A_{11}^{-1}\Phi^*(\mathbf{r} - K^*K\Psi\mathbf{x}_2), \end{aligned}$$

shows that $\Phi \mathbf{x}_1$ explicitly depends on $\Psi \mathbf{x}_2$. The algorithm to implement Symmetric Multiplicative Schwarz Preconditioning can now be given.

Algorithm 4.1 *Symmetric Multiplicative Schwarz Preconditioning* $\mathbf{u} = M_{\text{SMS}}^{-1} \mathbf{r}$

$$\begin{aligned}
 \mathbf{v} &= \Phi A_{11}^{-1} \Phi^* \mathbf{r} \\
 \mathbf{e} &= \mathbf{r} - K^* K \mathbf{v} \\
 \mathbf{w} &= L^{-1} \mathbf{e} \\
 \mathbf{x} &= \mathbf{w} - \Phi G^{-1} \Phi^* \mathbf{e} \\
 \mathbf{u}_Q &= \frac{1}{\alpha} \mathbf{x} \\
 \mathbf{y} &= \mathbf{r} - K^* K \mathbf{u}_Q \\
 \mathbf{u}_P &= \Phi A_{11}^{-1} \Phi^* \mathbf{y} \\
 \mathbf{u} &= \mathbf{u}_P + \mathbf{u}_Q
 \end{aligned}$$

For every multiplicative Schwarz PCG iteration the computation is dominated by two matrix-vector products involving $K^* K$, one matrix-vector product involving $A = K^* K + \alpha L$, two inversions of A_{11} , one inversion of G and one inversion of L .

Schur Complement Conjugate Gradient

This algorithm starts with the transformed system, (4.4). To this system we apply block Gaussian elimination to get

$$\begin{bmatrix} A_{11} & A_{12} \\ 0 & S \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \Phi^* \mathbf{b} \\ \Psi^* \mathbf{b} - A_{21} A_{11}^{-1} \Phi^* \mathbf{b} \end{bmatrix} \quad (4.16)$$

where

$$S \equiv A_{22} - A_{21} A_{11}^{-1} A_{12} = \Psi^* A (I - \Phi A_{11}^{-1} \Phi^* A) \Psi, \quad (4.17)$$

is the Schur complement relative to the coefficient matrix (4.4). Note that the submatrix $A_{22} = \Psi^* A \Psi$ is not the same as $\tilde{A}_{22} = \alpha I_{n-N}$, the submatrix used in both of the Schwarz methods. Back substitution in (4.16) produces

$$\mathbf{x}_1 = A_{11}^{-1} \Phi^* \mathbf{b} - A_{11}^{-1} A_{12} \mathbf{x}_2 \quad (4.18)$$

where

$$S \mathbf{x}_2 = \Psi^* \mathbf{b} - A_{21} A_{11}^{-1} \Phi^* \mathbf{b}. \quad (4.19)$$

The key idea for this algorithm is to solve the Schur complement system, (4.19), by using the conjugate gradient method to obtain an approximation to \mathbf{x}_2 . The result will be used in the back substitution step, (4.18). We then backtransform using (4.10) to approximate the solution $\mathbf{u} = A^{-1} \mathbf{b}$.

We now focus on the CG solution of (4.19). To simplify notation define

$$\begin{aligned} \tilde{\mathbf{b}} &= \Psi^* \mathbf{b} - A_{21} A_{11}^{-1} \Phi^* \mathbf{b} \\ &= \Psi^* (I - A \Phi A_{11}^{-1} \Phi^*) \mathbf{b}. \end{aligned}$$

As presented in Algorithm 2.1, the CG algorithm requires the computation of the following, where $\tilde{\mathbf{x}}$ is the CG approximation to the solution of (4.19), $\tilde{\mathbf{r}}$ denotes the residual, and $\tilde{\mathbf{z}}$ denotes the CG descent direction:

$$\tilde{\mathbf{r}} = \tilde{\mathbf{b}} - S \tilde{\mathbf{x}} \quad (4.20a)$$

$$\tilde{\delta} = \tilde{\mathbf{r}}^* \tilde{\mathbf{r}} \quad (4.20b)$$

$$\tilde{\tau} = \frac{\tilde{\delta}}{\tilde{\mathbf{z}}^* S \tilde{\mathbf{z}}} \quad (4.20c)$$

Note that all the vectors in (4.20) are elements of \mathbb{R}^{n-N} . This CG algorithm will be reformulated into an algorithm that directly computes the desired solution to (4.1a), $\mathbf{u} \in \mathbb{R}^n$. Moreover, there will not be any explicit dependence on Ψ .

Combining the equations (4.10), (4.18), and (4.19) generates the following expression for \mathbf{u} :

$$\begin{aligned}\mathbf{u} &= \Phi A_{11}^{-1} \Phi^* \mathbf{b} - \Phi A_{11}^{-1} A_{12} \mathbf{x}_2 + \Psi \mathbf{x}_2 \\ &= \Phi A_{11}^{-1} \Phi^* \mathbf{b} + (I - \Phi A_{11}^{-1} \Phi^* A) \Psi \mathbf{x}_2.\end{aligned}\quad (4.21)$$

To find the solution to (4.1) the equation (4.21) suggests that the initial guess for CG should be $\mathbf{u}_0 = \Phi A_{11}^{-1} \Phi^* \mathbf{b}$ and updated by $(I - \Phi A_{11}^{-1} \Phi^* A) \Psi \mathbf{x}_2$. Also from the definition of A_{11} it follows that

$$A(I - \Phi A_{11}^{-1} \Phi^* A) \Phi = 0 \quad (4.22a)$$

$$\Phi^* A(I - \Phi A_{11}^{-1} \Phi^* A) = 0. \quad (4.22b)$$

For any $\tilde{\mathbf{z}} \in \mathbb{R}^{n-N}$ define $\mathbf{w} \in \mathbb{R}^n$ by $\mathbf{w} = A(I - \Phi A_{11}^{-1} \Phi^* A) \Psi \tilde{\mathbf{z}}$, hence by (4.17)

$$S\tilde{\mathbf{z}} = \Psi^* \mathbf{w}.$$

Let \mathbf{g} be an element in V_Ψ such that $\mathbf{g} = \Psi S\tilde{\mathbf{z}}$. Thus

$$\begin{aligned}\mathbf{g} &= \Psi \Psi^* \mathbf{w} \\ &= (I - P)L^{-1} \mathbf{w} \\ &= L^{-1} \mathbf{w} - \Phi G^{-1} \Phi^* A(I - \Phi A_{11}^{-1} \Phi^* A) \Psi \tilde{\mathbf{z}} \\ &= L^{-1} \mathbf{w}.\end{aligned}\quad (4.23)$$

In the denominator of the term $\tilde{\tau}$, (4.20c), let $\mathbf{z} \in \mathbb{R}^n$ be such that $\tilde{\mathbf{z}} = \Psi^* \mathbf{z}$, hence

$$\begin{aligned}\tilde{\mathbf{z}}^* S\tilde{\mathbf{z}} &= (\Psi^* \mathbf{z})^* S\tilde{\mathbf{z}} \\ &= \mathbf{z}^* \mathbf{g}.\end{aligned}\quad (4.24)$$

To find the conversion expression for the numerator of $\tilde{\tau}$ will first require a conversion expression for $\tilde{\tau}$. To compute $S\tilde{\mathbf{x}}$ the expressions in (4.11) and (4.22) can be used to

get

$$\begin{aligned} S\tilde{\mathbf{x}} &= \Psi^*A(I - \Phi A_{11}^{-1}\Phi^*A)\Psi\Psi^*\mathbf{x} \\ &= \Psi^*A(I - \Phi A_{11}^{-1}\Phi^*A)L^{-1}\mathbf{x}. \end{aligned} \quad (4.25)$$

The residual in \mathbb{R}^{n-N} , $\tilde{\mathbf{r}} = \tilde{\mathbf{b}} - S\tilde{\mathbf{x}}$, is related to the residual in \mathbb{R}^n , $\mathbf{r} = \mathbf{b} - A\mathbf{u}$. This relationship results from using (4.19), (4.21), and (4.25) to get

$$\begin{aligned} \tilde{\mathbf{r}} &= \tilde{\mathbf{b}} - S\tilde{\mathbf{x}} \\ &= \Psi^*(\mathbf{b} - A\Phi A_{11}^{-1}\Phi^*\mathbf{b} - A(I - \Phi A_{11}^{-1}\Phi^*A)L^{-1}\mathbf{x}) \\ &= \Psi^*(\mathbf{b} - A\mathbf{u}) \\ &= \Psi^*\mathbf{r}. \end{aligned} \quad (4.26)$$

The next term from (4.20) to be considered is $\tilde{\delta}$. The combination of (4.11) and (4.26) yields

$$\begin{aligned} \tilde{\delta} &= \mathbf{r}^*\Psi\Psi^*\mathbf{r} \\ &= \mathbf{r}^*(L^{-1} - \Phi G^{-1}\Phi^*)\mathbf{r}. \end{aligned} \quad (4.27)$$

By using (4.21) the residual to original system can be rewritten as

$$\begin{aligned} \mathbf{r} &= \mathbf{b} - A\mathbf{u} \\ &= (I - A\Phi A_{11}^{-1}\Phi^*)\mathbf{b} - A(I - \Phi A_{11}^{-1}\Phi^*A)L^{-1}\mathbf{x} \\ &= (I - A\Phi A_{11}^{-1}\Phi^*)(\mathbf{b} - AL^{-1}\mathbf{x}). \end{aligned} \quad (4.28)$$

Combining (4.27) and (4.28) plus using the property in (4.22) results in

$$\tilde{\delta} = \mathbf{r}^*L^{-1}\mathbf{r}. \quad (4.29)$$

Efficient use of L^{-1} , A , A_{11}^{-1} , and the projection operators produces the following algorithm.

Algorithm 4.2 *Schur Complement CG*

```

u =  $\Phi A_{11}^{-1} \Phi^* \mathbf{b}$ ;
r =  $\mathbf{b} - A\mathbf{u}$ ; y =  $L^{-1}\mathbf{r}$ ;
 $\delta_0 = \mathbf{r}^* \mathbf{y}$ ;
z = r; d = y;
while  $\|\mathbf{r}\| \geq \epsilon$ 
    v =  $(I - \Phi A_{11}^{-1} \Phi^* A)\mathbf{d}$ ;
    w =  $A\mathbf{v}$ ; g =  $L^{-1}\mathbf{w}$ ;
     $\tau = \frac{\delta_0}{\mathbf{z}^* \mathbf{g}}$ ;
    u = u +  $\tau \mathbf{v}$ ;
    r = r -  $\tau \mathbf{w}$ ; y = y -  $\tau \mathbf{g}$ ;
     $\delta_1 = \mathbf{r}^* \mathbf{y}$ ;
     $\beta = \frac{\delta_1}{\delta_0}$ ;
     $\delta_0 = \delta_1$ ;
    z = r +  $\beta \mathbf{z}$ ; d = y +  $\beta \mathbf{d}$ ;
end while

```

For Schur complement CG the dominant terms computationally are: two matrix-vector products involving A , one inversion of A_{11} , and one inversion of L per CG iteration.

CHAPTER 5

Two-Level Preconditioners for a Semidefinite L

As indicated in Chapter 2 both H^1 and total variation regularization may have associated positive semidefinite regularization matrices. This chapter adapts the two-level preconditioners from Chapter 4 to handle regularization matrices that are semidefinite. The algorithms presented address the case when L has a one dimensional null space.

Just as in Chapter 4, the development starts with the decomposition of the solution space into a pair of L -orthogonal subspaces. Let the coarse grid subspace, V_Φ , have a basis $\{\phi_i\}_{i=1}^N$, with $1 \leq N < n$. The L -orthogonal complement, V_Ψ , will have basis functions $\{\psi_j\}_{j=1}^{n-N}$. The bases for both subspaces can still be chosen so that the properties in (4.2) hold. In contrast to Chapter 4, L has a null space that must be accounted for in the decomposition of the solution space. The first coarse grid basis element, ϕ_1 , will be chosen as the basis vector from the null space of L , as given in Chapter 2. In the construction of the projection operators, it will be necessary to construct an extra basis set that is a subset of V_Φ . Define $\Theta = [\theta_1 \theta_2 \dots \theta_{N-1}]$ to be an $n \times (N - 1)$ matrix such that

$$\theta_i = \phi_{i+1} - \left(\frac{\phi_{i+1}^* \phi_1}{\phi_1^* \phi_1} \right) \phi_1, \quad (5.1)$$

where $i = 1, 2, \dots, N - 1$. In addition, the basis vectors of V_Ψ can be chosen so that

$$\phi_1^* \psi_j = 0 \quad (5.2)$$

for $j = 1, 2, \dots, n - N$.

Example 5.1 This is a one-dimensional example of the type of bases that can be used for the two-level methods in this chapter. Assume the solution space is \mathbb{R}^7 and the regularization matrix is the discrete negative Laplacian with homogeneous Neumann boundary conditions, which is a 7×7 matrix that is given by

$$L = \frac{1}{(\Delta x)^2} \begin{bmatrix} 1 & -1 & 0 & & & & 0 \\ -1 & 2 & -1 & 0 & & & 0 \\ 0 & -1 & 2 & -1 & 0 & & 0 \\ \vdots & & & \ddots & \ddots & \ddots & 0 \\ 0 & & & & 0 & -1 & 2 & -1 \\ 0 & & & & & 0 & -1 & 1 \end{bmatrix},$$

where Δx is the uniform grid spacing. An example of an appropriate choice of basis elements are:

$$\begin{aligned} \phi_1 &= \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}; \quad \phi_2 = \begin{pmatrix} -\frac{1}{2} \\ -\frac{1}{3} \\ -\frac{1}{6} \\ 0 \\ \frac{1}{6} \\ \frac{1}{3} \\ \frac{1}{2} \end{pmatrix}; \quad \phi_3 = \begin{pmatrix} -\frac{3}{2} \\ -\frac{2}{21} \\ \frac{5}{21} \\ \frac{4}{21} \\ \frac{5}{21} \\ \frac{2}{21} \\ -\frac{2}{3} \end{pmatrix}; \\ \psi_1 &= \frac{1}{42} \begin{pmatrix} \frac{1}{2} \\ -\frac{2}{3} \\ -\frac{11}{6} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}; \quad \psi_2 = \frac{1}{42} \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ -\frac{11}{6} \\ -\frac{2}{3} \\ \frac{1}{2} \end{pmatrix}; \quad \psi_3 = \frac{1}{14} \begin{pmatrix} -\frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \\ 3 \\ -\frac{1}{2} \end{pmatrix}; \quad \psi_4 = \frac{1}{28} \begin{pmatrix} \frac{1}{2} \\ -3 \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} \end{aligned}$$

Notice that $\phi_i^* L \psi_j = 0$, $\psi_i^* L \psi_j = \delta_{ij}$, and $\phi_1^* \psi_j = 0$ for $i = 1, 2, 3$ and $j = 1, 2, 3, 4$.

For the formulation of the two-level preconditioners the following operators will be

needed:

$$G = \Theta^* L \Theta \quad (5.3a)$$

$$P_\Theta = \Theta G^{-1} \Theta^* L \quad (5.3b)$$

$$P_1 = \frac{\phi_1 \phi_1^*}{\phi_1^* \phi_1} \quad (5.3c)$$

$$P = P_\Theta + P_1 \quad (5.3d)$$

$$Q = \Psi \Psi^* L = I - P. \quad (5.3e)$$

Lemma 5.1 Assume the operators are defined as in (5.3), $\text{null}(L) = \text{span}(\phi_1)$, and the columns of Θ satisfy (5.1). If the bases for V_Φ and V_Ψ satisfy (4.2) and (5.2) then the operators P and Q are L -orthogonal projection operators onto V_Φ and V_Ψ respectively.

Proof: Since L is symmetric and the properties in (4.2) hold it follows readily that $Q^2 = Q$, $Q\Psi = \Psi$, and $Q\Phi = 0$. Combining (5.1) and (4.2) results in $\Theta^* L \Psi = 0$. Moreover, $\text{null}(L) = \text{span}(\phi_1)$. Hence by direct computation $P_\Theta^2 = P_\Theta$, $P^2 = P$, and $P\Psi = 0$. It has been shown that both P and Q are projection operators with Q projecting onto V_Ψ . However, showing that P projects onto V_Φ remains to be established. Any $\mathbf{v} \in V_\Phi$ can be rewritten into the form $\mathbf{v} = a_0\phi_1 + a_1\theta_1 + a_2\theta_2 + \dots + a_{N-1}\theta_{N-1}$. Since P_Θ and P_1 are both projection operators with the relationship given in (5.1) it follows that $P\mathbf{v} = \mathbf{v}$ for every $\mathbf{v} \in V_\Phi$. The fact that V_Φ and V_Ψ are L -orthogonal implies that P and Q are L -orthogonal projection operators.

□

We will start by deriving additive Schwarz preconditioning and then move on to the symmetric multiplicative Schwarz and the Schur complement preconditioners.

Additive Schwarz Preconditioning

Recall from Chapter 4 that the application of the Additive Schwarz preconditioner to a vector $\mathbf{r} \in \mathbb{R}^n$ is represented by

$$M_{AS}^{-1}\mathbf{r} = \Phi A_{11}^{-1}\Phi^*\mathbf{r} + \frac{1}{\alpha}\Psi\Psi^*\mathbf{r}. \quad (5.4)$$

Through the use of the projection operators the term $\frac{1}{\alpha}\Psi\Psi^*\mathbf{r}$ in (5.4) can be rewritten so that there is no explicit dependence on Ψ . Let L^\dagger denote the pseudo-inverse of L . It follows that LL^\dagger is an orthogonal projection operator onto the complement of the null space of L [30, pp. 432,434]. This fact combined with (5.2) produces the relation

$$\Psi^*LL^\dagger = \Psi^*. \quad (5.5)$$

Using (5.3), (5.5), and (4.2) yields

$$\begin{aligned} \Psi\Psi^* &= \Psi\Psi^*LL^\dagger \\ &= (I - P)L^\dagger \\ &= (L^\dagger - \Theta G^{-1}\Theta^* - P_1L^\dagger). \end{aligned} \quad (5.6)$$

It follows from (5.4) that

$$M_{AS}^{-1}\mathbf{r} = \Phi A_{11}^{-1}(\Phi^*\mathbf{r}) + \frac{1}{\alpha}((I - P_1)L^\dagger\mathbf{r} - \Theta G^{-1}\Theta^*\mathbf{r}). \quad (5.7)$$

The computational cost for Additive Schwarz PCG iteration is dominated by the one application of L^\dagger , one inversion of A_{11} , one inversion of G , and one application of A . Note that the cost of applying P_1 is insignificant, since for any $\mathbf{v} \in \mathbb{R}^n$, $P_1\mathbf{v} = \left(\frac{\phi_1^*\mathbf{v}}{\phi_1^*\phi_1}\right)\phi_1$.

Symmetric Multiplicative Schwarz Preconditioning

Recall that the application of the symmetric multiplicative Schwarz preconditioner to a vector $\mathbf{r} \in \mathbb{R}^n$ is given by

$$\begin{aligned} M_{\text{SMS}}^{-1}\mathbf{r} &= \Phi A_{11}^{-1}\Phi^*(\mathbf{r} - \alpha^{-1}K^*K\Psi\Psi^*(\mathbf{r} - K^*K\Phi A_{11}^{-1}\Phi^*\mathbf{r})) \\ &\quad + \alpha^{-1}\Psi\Psi^*(\mathbf{r} - K^*K\Phi A_{11}^{-1}\Phi^*\mathbf{r}). \end{aligned} \tag{5.8}$$

The identity in (5.6) shows the changes necessary in the implementation.

Algorithm 5.1 *Symmetric Multiplicative Schwarz Preconditioning* $M_{\text{SMS}}^{-1}\mathbf{r}$

$\begin{aligned} \mathbf{v} &= \Phi A_{11}^{-1}\Phi^*\mathbf{r}; \\ \mathbf{e} &= \mathbf{r} - K^*K\mathbf{v}; \\ \mathbf{w} &= L^\dagger\mathbf{e}; \\ \mathbf{x} &= \mathbf{w} - P_1\mathbf{w} - \Theta G^{-1}\Theta^*\mathbf{e}; \\ \mathbf{u}_Q &= \frac{1}{\alpha}\mathbf{x}; \\ \mathbf{y} &= \mathbf{r} - K^*K\mathbf{u}_Q; \\ \mathbf{u}_P &= \Phi A_{11}^{-1}\Phi^*\mathbf{y}; \\ \mathbf{u} &= \mathbf{u}_P + \mathbf{u}_Q; \end{aligned}$
--

The computational cost of each PCG iteration with symmetric multiplicative Schwarz preconditioning is dominated by the application of L^\dagger , two inversions of A_{11} , one inversion of G , two matrix-vector products involving the operator K^*K , and an application of $A = K^*K + \alpha L$.

Schur Complement Conjugate Gradient

As stated in Chapter 4, Schur complement CG relies on the computation of the

elements in (4.20). Using (5.6) in equation (4.23), we obtain

$$\begin{aligned}\mathbf{g} &= \Psi\Psi^*\mathbf{w} \\ &= (L^\dagger - \Theta G^{-1}\Theta^* - P_1L^\dagger)\mathbf{w}.\end{aligned}\tag{5.9}$$

Moreover, (5.6) will also change (4.27) to

$$\tilde{\delta} = \mathbf{r}^*(L^\dagger - \Theta G^{-1}\Theta^* - P_1L^\dagger)\mathbf{r}.\tag{5.10}$$

With these few changes, Algorithm 4.2 can be altered to accommodate a semidefinite L .

Algorithm 5.2 *Schur complement Conjugate Gradient Iteration*

```

 $\mathbf{u} = \Phi A_{11}^{-1} \Phi^* \mathbf{b};$ 
 $\mathbf{r} = \mathbf{b} - A\mathbf{u};$ 
 $\mathbf{q} = L^\dagger \mathbf{r};$ 
 $\mathbf{y} = \mathbf{q} - \Theta G^{-1} \Theta^* \mathbf{r} - P_1 \mathbf{q};$ 
 $\delta_0 = \mathbf{r}^* \mathbf{y};$ 
 $\mathbf{z} = \mathbf{r}; \quad \mathbf{d} = \mathbf{y};$ 
Begin CG iterations
   $\mathbf{v} = (I - \Phi A_{11}^{-1} \Phi^* A) \mathbf{d};$ 
   $\mathbf{w} = A\mathbf{v};$ 
   $\mathbf{f} = L^\dagger \mathbf{w};$ 
   $\mathbf{g} = \mathbf{f} - \Theta G^{-1} \Theta^* \mathbf{w} - P_1 \mathbf{f};$ 
   $\tau = \delta_0 / (\mathbf{z}^* \mathbf{g});$ 
   $\mathbf{u} = \mathbf{u} + \tau \mathbf{v};$ 
   $\mathbf{r} = \mathbf{r} - \tau \mathbf{w}; \quad \mathbf{y} = \mathbf{y} - \tau \mathbf{g};$ 
   $\delta_1 = \mathbf{r}^* \mathbf{y};$ 
   $\beta = \delta_1 / \delta_0;$ 
   $\delta_0 = \delta_1;$ 
   $\mathbf{z} = \mathbf{r} + \beta \mathbf{z}; \quad \mathbf{d} = \mathbf{y} + \beta \mathbf{d};$ 
end CG iterations

```

For every CG iteration, the computational cost is dominated by one application of L^\dagger , two matrix-vector products involving $A = K^*K + \alpha L$, one inversion of A_{11} , and one inversion of G .

CHAPTER 6

Numerical Results

The purpose of this chapter is to compare the two-level preconditioners with circulant preconditioning. The first section will describe the test problem to be used in the comparison. The second section will be concerned with implementation details. The third section examines the computational cost of the different algorithms. The fourth section will present the results, and the final section will contain a summary.

The Test Problem

All results for this chapter arise from applying the preconditioning algorithms from this thesis to a set of data provided by the Starfire Optical Range (SOR), USAF Phillips Laboratory at Kirkland AFB, New Mexico. The data is from a simulation of the effects of atmospheric blurring on an image of a satellite in earth orbit. The data consists of 256×256 arrays where the value in the i -th row and j -th column represents the light intensity at the pixel (x_i, y_j) . The observed image is modeled by the array

$$z_{i,j} = [\mathcal{K}\bar{u}](x_i, y_j) + \eta_{i,j}, \quad (6.1)$$

where \mathcal{K} is the convolution operator given in (2.29), \bar{u} is the true image, and η is an additive noise term [39] [2]. In this application the kernel function for \mathcal{K} is called

the point spread function (PSF). The PSF describes the blur that occurs in taking the image of a point source. The term $[\mathcal{K}\bar{u}]$ in (6.1) is approximated by the use of midpoint quadrature on (2.29)

$$[\mathcal{K}\bar{u}](x_i, y_j) = \sum_s \sum_r k_{i-r, j-s} \bar{u}_{r,s} \Delta x \Delta y + \zeta_{i,j}, \quad (6.2)$$

where $k_{i,j} = k(x_i, y_j)$ and ζ represents the remainder term from the quadrature method. Combining (6.1) - (6.2), setting $\mathbf{z} = \text{vector}(z)$, and defining $\mathbf{u} = \text{vector}(u)$, it follows that the resulting system has the form

$$\mathbf{z} = K\mathbf{u} + \nu \quad (6.3)$$

where ν is the resulting error term from η and ζ . The matrix K is the BTTB matrix generated by k . Recall that the size of the square array is $n_x \times n_x$, where $n_x = 256$, which implies that K is a $256^2 \times 256^2$ matrix. Figure 1 shows the true image, the noisy blurred image that is observed, the PSF, and a reconstruction of the image obtained with Tikhonov regularization. Tikhonov regularization applied to (6.3) yields,

$$\begin{aligned} \mathbf{u}_\alpha &= \arg \min_{\mathbf{u} \in \mathbb{R}^n} \|K\mathbf{u} - \mathbf{z}\|^2 + \alpha \mathbf{u}^* L \mathbf{u} \\ &= (K^*K + \alpha L)^{-1} K^* \mathbf{z}. \end{aligned} \quad (6.4)$$

We refer to Chapter 2 for details. Note that throughout this chapter the norm $\|\mathbf{u}\|$, where $\mathbf{u} \in \mathbb{R}^n$, will be the Euclidean norm.

Implementation Details

It was mentioned previously that K is a BTTB matrix and the matrix-vector products involving K and K^* can be computed via FFT's. However, K^*K may not be BTTB. To simplify computation and reduce storage we will use a symmetric BTTB

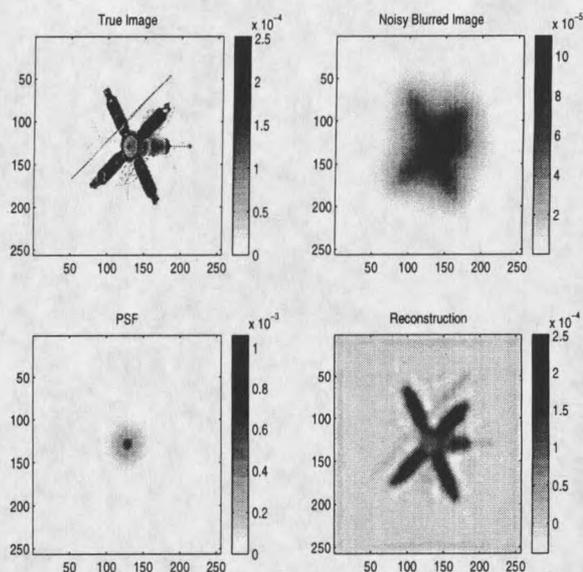


Figure 1: Simulated image Data from the Starfire Optical Range. At the top left is the true image. Noisy blurred observed data appears at the top right. At the lower left is the PSF. The lower right is a reconstruction using total variation regularization where $\alpha = 10^{-13}$ and $\beta = .1$.

approximation in place of K^*K . This approximation will be denoted by T and the system that will be solved is

$$(T + \alpha L)\mathbf{u} = \mathbf{b}, \quad (6.5)$$

where $\mathbf{b} = K^*\mathbf{z}$. Consult Chapter 3 for the construction of T using the PSF array k . For the test problem numerical solutions to (6.5) compare favorably with the numerical solutions to (1.1). The only major difference between the two types of reconstructions is along the boundary of the image [37].

To reduce computation cost of implementing the two-level preconditioners we will use an efficient method of projecting onto the space spanned by the coarse grid basis vectors. To compute $\Phi^*\mathbf{r}$ it is not necessary to construct the $n \times N$ matrix Φ . As discussed in [45], the computation of the matrix-vector products $\Phi^*\mathbf{r}$ and $\Phi\mathbf{v}$ can

be implemented using a multilevel restriction and a multilevel prolongation respectively. The element Φ^*r is generated by applying a series of multigrid restrictions on $\text{array}(r)$. To compute Φv the vector v can be rearranged into a $m_x \times m_x$ array and a series of multigrid prolongations are applied to generate Φv . For more details about multigrid see [4].

Example 6.1 This example is to illustrate how grid transfers from one level to the next can be accomplished by array operators. We will use piecewise constant "coarse grid" basis elements which are analogues of the basis elements mentioned in Example 4.1. The operators `vector` and `array` are the array operators that are defined in Chapter 3. Define the array r by

$$r = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{bmatrix}$$

With piecewise constant coarse grid basis elements, it follows that the grid restriction of r is given by

$$\text{array}(\Phi^*(\text{vector}(r))) = \begin{bmatrix} (r_{11} + r_{12} + r_{21} + r_{22}) & (r_{13} + r_{14} + r_{23} + r_{24}) \\ (r_{31} + r_{32} + r_{41} + r_{42}) & (r_{33} + r_{34} + r_{43} + r_{44}) \end{bmatrix}$$

Notice that the first entry is the sum of the four elements in the first quadrant, the second entry in the first row is the sum of the four elements in the second quadrant, and so on. Let the array v have the form

$$v = \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{bmatrix}$$

The prolongation of v using piecewise constant basis elements can be accomplished by the following

$$\text{array}(\Phi(\text{vector}(v))) = \begin{bmatrix} v_{11} & v_{11} & v_{12} & v_{12} \\ v_{11} & v_{11} & v_{12} & v_{12} \\ v_{21} & v_{21} & v_{22} & v_{22} \\ v_{21} & v_{21} & v_{22} & v_{22} \end{bmatrix}$$

Notice that prolongation takes an entry from the array and expands it to be the entry value for a whole quadrant. The computations in Example 6.1 show just one grid transfer, or one level in reduction. The coarse grid size that corresponds to this case is $N = \frac{n_x^2}{4}$. If implemented as a nested sequence of restrictions (or prolongations) then the coarse grid can be reduced (or increased) further.

The matrix $\Phi^*T\Phi$ can also be computed via array operations on the generator of T . The only positive definite regularization matrices we will consider are the identity and the discrete negative Laplacian with homogeneous Dirichlet boundary conditions. Both of these regularization operators are BTTB. Hence, $G = \Phi^*L\Phi$ can be computed in the same manner as $\Phi^*T\Phi$.

Example 6.2 The purpose of this example is to show how $\Phi^*T\Phi$ is computed using array operations. We will use piecewise constant coarse grid basis elements in Φ and T is a symmetric BTTB matrix with symmetric blocks. It follows that T has a generator that can be arranged into a $n_x \times n_x$ array. We will denote the generator by $t = [t_1, t_2, \dots, t_{n_x}]$, where t_j is the first column of the j th submatrix of T .

The reduction to the coarse grid is accomplished in two steps: i) Combine the columns of t to produce a smaller array v ; ii) Combine the rows of v which results in the array that will be the generator for $\Phi^*T\Phi$. The following is for the case where multiplying by Φ corresponds to a one level piecewise constant restriction operation. The columns of array v will be produced by

$$v_1 = 2t_1 + 2t_2$$

$$v_j = t_{2(j-1)} + 2t_{2j-1} + t_{2j},$$

where $j = 2, 3, \dots, \frac{n_x-2}{2}$. Define $(v)_j^*$ as the j th row of the array v . The next step

involves applying the same array operations on the rows of array v

$$(\mathbf{w})_1^* = 2(\mathbf{v})_1^* + 2(\mathbf{v})_2^*$$

$$(\mathbf{w})_j^* = (\mathbf{v})_{2(j-1)}^* + 2(\mathbf{v})_{2j-1}^* + (\mathbf{v})_{2j}^*,$$

where $j = 2, 3, \dots, \frac{n_x-2}{2}$. The final result is given by $\Phi^*T\Phi = \text{BTTB}(w)$, where $w = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{\frac{n_x-2}{2}}]$. More levels of reduction can be made by nesting the operations.

Examples 6.1 and 6.2 both illustrate the use of piecewise constant coarse grid basis elements. The implementation for piecewise linear basis elements is accomplished in a similar manner [4]. There is a caveat about the two types of basis elements. The piecewise constant basis elements are structured to apply on a grid where $n_x = 2^p$ with p being a positive integer. The piecewise linear basis elements are constructed to apply on a grid where $n_x = 2^p - 1$, for homogeneous Dirichlet boundary conditions and $n_x = 2^p + 1$ for homogeneous Neumann boundary conditions. The Dirichlet case will be considered in the following. We will accommodate this requirement by using an array operator to restrict our given array of size $n_x = 2^8$ to an array of size $(2^8 - 1) \times (2^8 - 1)$. Let a be a $n_x \times n_x$ array. To restrict a to an array that has size $(n_x - 1) \times (n_x - 1)$ we will use the array operator Υ^* that will be defined by

$$[\Upsilon^*a]_{i,j} = \frac{1}{4}(a_{i,j} + a_{i,j+1} + a_{i+1,j} + a_{i+1,j+1}), \quad i, j = 1, 2, \dots, n_x - 1. \quad (6.6)$$

The array operator that will extend the array a to an array that has size $(n_x + 1) \times$

$(n_x + 1)$ will be denoted by Υ and will be defined by

$$[\Upsilon a]_{i,j} = \begin{cases} \frac{1}{4}(a_{i-1,j-1} + a_{i,j-1} + a_{i-1,j} + a_{i,j}) & \text{if } i, j = 2, 3, \dots, n_x, \\ \frac{1}{4}(a_{i,j-1} + a_{i,j}) & \text{if } i = 1, n_x, \text{ and } j = 2, 3, \dots, n_x, \\ \frac{1}{4}(a_{i-1,j} + a_{i,j}) & \text{if } i = 2, 3, \dots, n_x, \text{ and } j = 1, n_x, \\ \frac{1}{4}a_{i,j} & \text{if } i = j = 1, \\ \frac{1}{4}a_{i-1,j-1} & \text{if } i = j = n_x + 1, \\ \frac{1}{4}a_{i,j-1} & \text{if } i = 1 \text{ and } j = n_x + 1, \\ \frac{1}{4}a_{i-1,j} & \text{if } i = n_x + 1 \text{ and } j = 1. \end{cases} \quad (6.7)$$

It follows that when we use piecewise linear basis elements on this test problem a restriction of the vector \mathbf{r} to the coarse grid will take the form $\Phi^* \text{vector}(\Upsilon^*(\text{array}(\mathbf{r})))$ and a prolongation of the vector \mathbf{v} will be accomplished by $\Phi \text{vector}(\Upsilon(\text{array}(\Phi \mathbf{v})))$.

The last implementation issue we will mention concerns the regularization matrix L . Recall that the two-level preconditioners requires solving a system of the form:

$$Lg = w. \quad (6.8)$$

When L is SPD we can use the Cholesky factorization [19] to solve (6.8). A disadvantage to Cholesky factorization is the increase in the amount of storage. However, for a banded matrix the Cholesky factor is also banded [19, p. 155].

When L is symmetric positive semidefinite a PCG method will be used to apply L^\dagger . One iteration of a multigrid V-cycle is used as a preconditioner for every CG iteration (MG-CG). To preserve symmetry, the V-cycle is constructed with a forward sweep of Gauss Seidel as a pre-smoother and a backward sweep of Gauss Seidel as a post-smoother. To solve the semidefinite problem the projection onto the null space is subtracted from the grid approximation before every pre-smoother and after every post-smoother. More discussion of MG-CG can be found in [29]. The

choice of using V-cycles to apply L^\dagger makes it necessary to use grids of size $n_x = 2^p + 1$. We will use the array operators Υ^* and Υ defined in (6.6) and (6.7) to restrict and extend the array for our test problem as needed.

Cost

We will focus our discussion of cost in terms of flops.

Definition 6.1 A *flop* is an elementary scalar floating point operation on a pair of real numbers: $\alpha + \beta$, $\alpha - \beta$, α/β , or $\alpha * \beta$, where $\alpha, \beta \in \mathbb{R}$.

Sometimes the flop counts will be given with just the higher order terms. For example, if an algorithm is classified as costing $\frac{3}{2}n^2$ flops then it is implied that the cost is $(\frac{3}{2}n^2 + \mathcal{O}(n))$ flops.

The first preconditioning algorithm we presented, Algorithm 3.1, corresponds to circulant preconditioning. The cost of implementing a two dimensional FFT on a $n_x \times n_x$ array is $\mathcal{O}(n_x^2 \log(n_x))$ flops [5]. Recall that for the matrix-vector products involving T , and the Algorithm 3.1, employ FFT's of arrays that are of size $2n_x \times 2n_x$. Thus, the cost to build the preconditioner is also $\mathcal{O}(n_x^2 \log(n_x))$ flops. The number of flops that are required for Algorithm 3.1 is on the order of $\mathcal{O}(n_x^2 \log(n_x))$. The matrix vector products involving T , (3.12), will also require $\mathcal{O}(n_x^2 \log(n_x))$ flops.

In the case of two-level preconditioning we will make two comparisons. The first comparison comes from varying the type of coarse grid basis elements. We will contrast the cost between choosing piecewise constant basis elements with the choice of piecewise linear basis elements. The second comparison involves the choice of regularization matrix.

Table 1: The computational costs, in flops, for the multilevel grid transfers involving Φ . Note the expression for m_p is given in (6.9) with p being the number restrictions it takes to get to the coarse grid.

	Piecewise constant	Piecewise linear
$\Phi^* \mathbf{r}$	$3m_p$	$11m_p$
$\Phi \mathbf{v}$	$4m_p$	$18m_p$
$\Phi^* T \Phi$	$9m_p$	$21m_p$

The main difference in computational cost between using piecewise constant and piecewise linear basis elements occurs in the computations that involve Φ . In the previous section we established that these computations can be done using nested multigrid operations. Here we will consider the case when piecewise constant coarse grid basis elements are used. If we denote p as the number of restrictions (or prolongations) in the nested sequence then the size of the coarse grid becomes $N = \left(\frac{n_x}{2^p}\right)^2$. Part of the calculation, when computing the total cost, involves a finite geometric sum,

$$m_p = \sum_{k=1}^p \left(\frac{n_x}{2^k}\right)^2 = \frac{n_x^2}{3} \left(1 - \left(\frac{1}{4}\right)^p\right), \quad (6.9)$$

that adds up the computational cost for each level in the nested sequence. Since p is a positive integer it follows that $m_p \leq \frac{n_x^2}{3}$. For piecewise constant coarse grid basis elements the expression in (6.9) is an exact way for computing the cost over all levels. Moreover, (6.9) is a good approximation for the cost involved with using piecewise linear coarse grid basis elements. Table 1 shows the approximate computational costs for the matrix-vector products involving Φ .

Table 2 displays a summary of the different matrix computations for the two-level preconditioners when using a positive definite regularization matrix. This table displays the count of the more costly operations that are involved per CG iteration. The matrix L is sparse for each of the different regularization schemes from Chapter

Table 2: Matrix computations per CG iteration for each of the two-level preconditioners when using a positive definite regularization matrix.

Matrix	Additive Schwarz	Multiplicative Schwarz	Schur Complement CG
T	1	3	2
L^{-1}	1	1	1
A_{11}^{-1}	1	2	1
G^{-1}	1	1	0
Φ	1	3	1
Φ^*	1	3	1

Table 3: Matrix computations per CG iteration for each of the two-level preconditioners when using a positive semidefinite regularization matrix.

Matrix	Additive Schwarz	Multiplicative Schwarz	Schur Complement CG
T	1	3	2
L^\dagger	1	1	1
A_{11}^{-1}	1	2	1
G^{-1}	1	1	1
$\Phi \mid \Theta$	1 1	2 1	1 1
$\Phi^* \mid \Theta^*$	1 1	2 1	1 1

2. As a result, the matrix-vector products involving L have a cost of $\mathcal{O}(n_x^2)$ flops. When L is SPD, we use the Cholesky factorization to invert L . The goal of Cholesky factorization is to find a factor R that is lower triangular and $L = RR^*$. Taking advantage of the banded structure of L permits the Cholesky factorization in $\mathcal{O}(n_x^4)$ flops. The cost to solve the system, $RR^*x = r$, using forward elimination followed by backward substitution is $\mathcal{O}(n_x^3)$ flops.

Recall, from Chapter 5, that the positive semidefinite regularization matrices required a change in implementation for the two-level preconditioners. Table 3 shows the corresponding change in matrix computations for each of the two-level algorithms. The multilevel prolongations and restrictions can still be accomplished by array oper-

ations. Grid operations similar to those used for Φ and Φ^* are used to apply Θ and Θ^* respectively. A key difference between the application of Φ and Θ is the subtraction of the projection onto the null space of L that is incorporated into the computation of Θ . The subtraction of the projection onto the null space requires $\mathcal{O}(n_x^2)$ flops.

The last computational cost issues involve the regularization matrices that are positive semidefinite. The discretization of the Laplacian with homogeneous Neumann boundary conditions has enough structure to compute $\Phi^*L\Phi$ via array operations, however it involves entries from three different columns of L since it is not a BTTB matrix. The construction of the matrix that corresponds to the TV regularization operator requires $\mathcal{O}(n_x^2)$ flops. Moreover, this matrix is not necessarily BTTB. Since L does not have Toeplitz structure it is necessary to explicitly construct Φ in order to compute $\Phi^*L\Phi$. The matrix Φ can be built with Kronecker products and it is a sparse $n \times N$ matrix with $N \ll n$. The computation for $G = \Theta^*L\Theta$ is done much the same way where the columns of Φ are used and the projection onto the null space of L is subtracted off. As was mentioned in the previous section, a semidefinite L requires the application of the pseudo-inverse of L to a vector. This computation is performed by using the MG-CG algorithm. One MG-CG iteration costs $\mathcal{O}(n_x^2)$ flops.

Results

One measure of performance that we will use is the relative solution error. The relative solution error is given by

$$error^k = \frac{\|\mathbf{u}_\alpha - \mathbf{u}_{approx}^k\|}{\|\mathbf{u}_\alpha\|}$$

where \mathbf{u}_α is the solution to (6.5) and \mathbf{u}_{approx}^k is the approximate solution to (6.5) at the

k^{th} CG iteration. We will often refer to the relative solution error as just the solution error. The solution error will be used to illustrate convergence results. In this section we will exhibit results for each of the regularization matrices presented in Chapter 2. The values that we will use for the regularization parameter α were values that yielded qualitatively good reconstructions. Consult [16] for a discussion on methods for choosing the value of α . For brevity we will use “SMS PCG” to denote symmetric multiplicative Schwarz PCG and “Schur CG” to denote Schur complement CG in the remainder of this section.

Problem 6.1 Here we solve equation (6.5) with $L = I$ (L^2 regularization) and $\alpha = 2 \times 10^{-13}$. Figure 2 compares the convergence in solution error between the two-level preconditioners, circulant preconditioning, and CG without preconditioning. Notice that additive Schwarz PCG is not effective for this particular test problem and has convergence that is slower than plain CG. Hence, it will not be discussed in the subsequent problems. Figure 3 shows the effect of varying the number of coarse grid basis elements on two-level PCG convergence. This leads to the question of picking the optimal number of coarse grid basis elements. Using a large number of basis elements accelerates convergence, however, it also increases the computational cost of each CG iteration. Table 4 shows the results in flop count and number of CG iterations when varying the number of piecewise constant coarse grid basis elements. For the remainder of the chapter we will present the results for the number N that minimizes computational cost. The last thing we will consider out of this problem is the cost of implementing the two-level algorithms with piecewise constant coarse grid basis elements compared to the cost of using piecewise linear coarse grid basis elements. Table 5 shows that for L^2 regularization, the cost was lower when using piecewise constant basis elements than for the piecewise linear basis elements. Although the piecewise constant basis elements gave slower convergence, they were less costly to

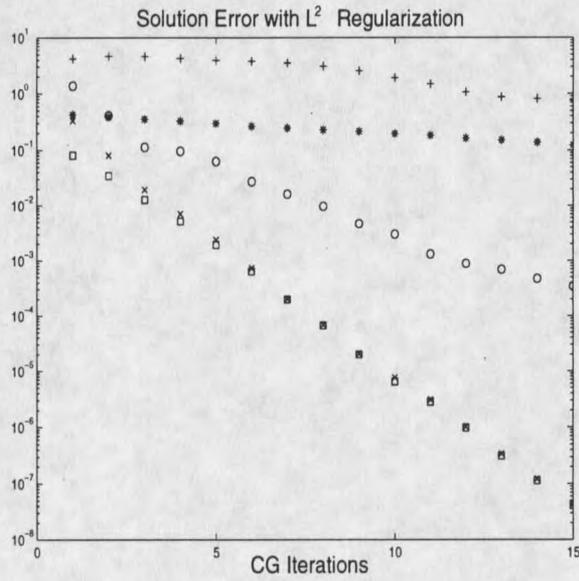


Figure 2: Performance of the different algorithms, showing relative solution error norm vs. iteration count. Pluses (+) represent additive Schwarz PCG; stars (*) represent plain CG; circles (o) represent circulant PCG; crosses (x) represent SMS PCG; and squares (\square) represent Schur CG. The results are from L^2 regularization with $\alpha = 2 \times 10^{-13}$. The two-level preconditioners were implemented with and 16^2 piecewise constant coarse grid basis functions.

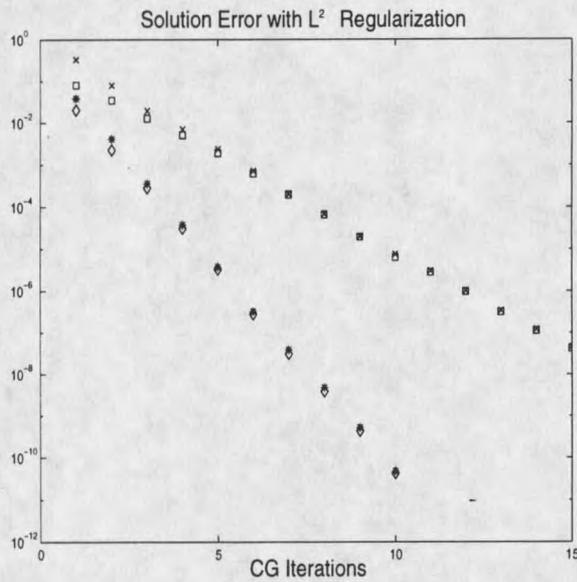


Figure 3: Performance of two-level preconditioners with different number of piecewise constant coarse grid basis elements. The results are the solution error norm vs. iteration count for L^2 regularization with $\alpha = 2 \times 10^{-13}$. N will denote the number of coarse grid basis elements. Crosses (x) represent SMS PCG with $N = 16^2$; squares (\square) represent Schur CG with $N = 16^2$; stars (*) represent SMS PCG with $N = 32^2$; and diamonds (\diamond) represent Schur CG with $N = 32^2$.

Table 4: Cost comparisons between the different algorithms using L^2 regularization with $\alpha = 2 \times 10^{-13}$. N denotes the number of piecewise constant coarse basis elements that were used to implement the two-level preconditioners. The solution error stopping tolerance was 10^{-9} .

Algorithm	# of Iterations	Cost
CG	183	2.7e+10
Circulant PCG	37	7.3e+09
SMS PCG		
N	# of Iterations	Cost
8^2	62	1.5e+10
16^2	19	4.9e+09
32^2	9	8.7e+09
Schur CG		
8^2	62	1.2e+10
16^2	20	3.9e+09
32^2	10	5.4e+09

use per CG iteration than the piecewise linear basis elements.

Problem 6.2 For this problem the regularization matrix L is the discretized negative Laplacian with homogeneous Dirichlet boundary conditions and $\alpha = 10^{-12}$. The convergence results are similar to Problem 6.1 [37]. For this test problem it is difficult to visually distinguish between reconstructions after a few iterations where the solution error is 10^{-3} compared to the reconstructions after many iterations when the solution error is smaller than 10^{-9} . Table 6 shows the cost and the number of CG iterations to yield a reconstruction with solution error smaller than 10^{-3} . These reconstructions are suitable for observation purposes. However, we will present results that correspond to a reconstruction with solution error smaller than 10^{-9} to better illustrate the asymptotic convergence and the computational differences between the algorithms. Table 7 displays the cost and number of CG iterations involved in a reconstruction that has a solution error smaller than 10^{-9} .

Table 5: Cost comparisons between using piecewise linear and piecewise constant basis elements using L^2 regularization with $\alpha = 2 \times 10^{-13}$. N will denote the number of coarse grid basis elements. In the case of piecewise constant basis elements $N = 16^2$ and for the case of piecewise linear basis elements $N = 31^2$. The solution error stopping tolerance was 10^{-9} .

SMS PCG			
Piecewise constant		Piecewise linear	
# of Iterations	Cost	# of Iterations	Cost
19	4.9e+09	13	1.8e+10
Schur CG			
# of Iterations	Cost	# of Iterations	Cost
20	3.9e+09	14	1.2e+10

Table 6: Cost comparisons between the different algorithms using the discretized negative Laplacian with homogeneous Dirichlet boundary conditions where $\alpha = 10^{-12}$. The two-level preconditioners were implemented using 31^2 piecewise linear basis elements. The solution error stopping tolerance was 10^{-3} .

Algorithm	# of Iterations	Cost
CG	68	9.9e+09
Circulant PCG	12	2.4e+09
SMS PCG	4	5.9e+09
Schur CG	4	4.3e+09

Table 7: This is the same cost comparison as in Table 6, except the solution error stopping tolerance is 10^{-9} .

Algorithm	# of Iterations	Cost
CG	189	2.7e+10
Circulant PCG	36	7.1e+09
SMS PCG	13	1.9e+10
Schur complement CG	15	1.4e+10

Table 8: Cost comparisons between the different algorithms using the discrete negative Laplacian with homogeneous Neumann boundary conditions where $\alpha = 10^{-12}$. The two-level preconditioners were implemented using 15^2 piecewise linear coarse grid basis elements. The solution error stopping tolerance was 10^{-9} .

Algorithm	# of Iterations	Cost
CG	193	2.8e+10
Circulant PCG	49	9.6e+09
SMS PCG	35	1.1e+10
Schur CG	34	1.1e+10

Problem 6.3 This problem uses the discrete negative Laplacian with homogeneous Neumann boundary conditions as the regularization matrix L with $\alpha = 10^{-12}$. Recall that this is a semidefinite regularization matrix. Thus, the two-level preconditioners will employ Algorithms 5.1 and 5.2. Table 8 displays the cost and the number of CG iterations for a reconstruction that has solution error smaller than 10^{-9} . For more convergence results we refer to [38].

Problem 6.4 Total variation regularization is applied in this last problem where $\alpha = 10^{-13}$ and $\beta = .1$. Recall that this yields a regularization matrix that is semidefinite. To illustrate the performance of the preconditioners only one fixed point iteration will be presented. In this case we will use the true image $\bar{u} = u_{true}$ in (2.35) to construct the matrix L . Table 9 shows the results for solving the resulting system (6.5).

Summary

In each of the problems the convergence rate for the two-level preconditioners varied with N , the number of coarse grid basis elements. With a small N the convergence

Table 9: Cost comparisons between the different algorithms using TV regularization with $\alpha = 10^{-13}$ and $\beta = .1$. The two-level preconditioners were implemented using 15^2 piecewise linear coarse grid basis elements. The solution error stopping tolerance was 10^{-9} .

Algorithm	# of Iterations	Cost
CG	191	3.7e+10
Circulant PCG	49	9.7e+09
SMS PCG	35	1.2e+10
Schur CG	33	1.2e+10

rate was low, but the rate increased with an increase in N ; see Table 4. Moreover, by using a large enough N the Schur complement conjugate gradient algorithm (Schur CG) and the symmetric multiplicative Schwarz preconditioned conjugate gradient algorithm (SMS PCG) will both converge more rapidly than circulant PCG.

Circulant preconditioning offers a low computational cost in implementation. However, it must be noted that in Problem 6.1 it was possible to use the two-level preconditioners at a lower computational cost than for the circulant preconditioner. In Chapter 1 it was mentioned that circulant preconditioners are efficient at preconditioning Toeplitz systems. The development of the two-level preconditioners in Chapters 4 and 5 does not rely on any Toeplitz structure. Thus, two-level preconditioners are not as restrictive in the type of problems that they can be applied to.

When comparing Schur CG with SMS PCG the former has a few distinct advantages. In early iterations the Schur CG method had smaller solution errors as evident in Figures 2 and 3. Schur CG also required lower computational cost to implement in the case of Problems 6.1 and 6.2. For the semidefinite operators the computational cost to implement Schur CG was quite similar to that of SMS PCG. One reason the two methods were closer in computational cost is the change in the

use of the matrix operators that is evident in Table 3. Figure 4 explains why Schur

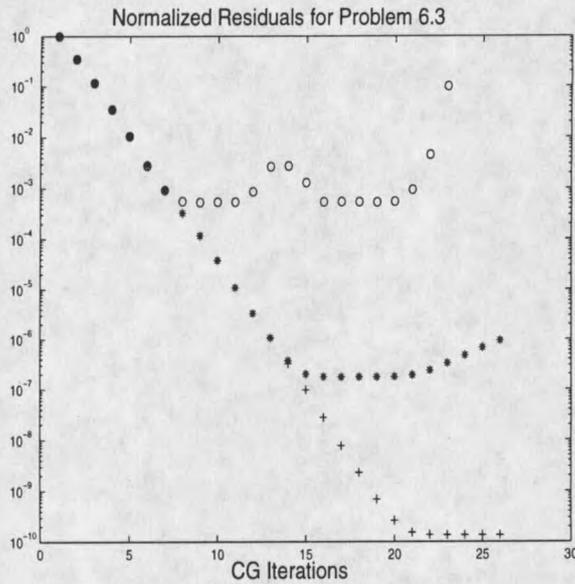


Figure 4: The effect of approximating L^\dagger on the convergence of Schur CG with a semidefinite L . The results given are normalized residuals vs. CG iteration counts. The regularization is with the discrete negative Laplacian with homogeneous Neumann boundary conditions with $\alpha = 5 \times 10^{-12}$ and with 15^2 piecewise linear coarse grid basis elements. Circles (o) represent 2 MG-CG iterations per PCG iteration; Stars (*) represent 5 MG-CG iterations per PCG iteration; and pluses (+) denote 10 MG-CG iterations per PCG iteration.

complement CG is comparatively more costly to implement with semidefinite regularization. Schur CG demands an accurate approximation of the matrix-vector products involving L^\dagger . This demand translates into several (8-10) iterations of multigrid conjugate gradient (MG-CG) to maintain convergence of the algorithm. On the other hand, symmetric multiplicative Schwarz PCG requires only one iteration of MG-CG to maintain convergence.

REFERENCES CITED

- [1] L. Adams and J.L. Nazareth, editors. *Linear and Nonlinear Conjugate Gradient-Related Methods*. SIAM, Philadelphia, PA, 1996.
- [2] H.C. Andrews and B.R. Hunt. *Digital Image Restoration*. Signal Processing Series. Prentice-Hall, Englewood Cliffs, NJ, 1977.
- [3] O. Axelsson. *Iterative Solution Methods*. Cambridge Press, New, NY, 1996.
- [4] W. L. Briggs. *A Multigrid Tutorial*. SIAM, Philadelphia, PA, 1987.
- [5] W. L. Briggs and V. E. Henson. *The DFT (An Owner's Manual for the Discrete Fourier Transform)*. SIAM, Philadelphia, PA, 1995.
- [6] R. Chan, T. F. Chan, and W. L. Wan. Multigrid for differential-convolution problems arising from image processing. *Proceedings of the Workshop on Sci. Comput.*, to appear, 1997.
- [7] R. Chan and X.Q. Jin. A family of block preconditioners for block systems. *SIAM J. Sci. Comput.*, 13(5):1218–1235, 1992.
- [8] R. Chan, J. Nagy, and R. Plemmons. Circulant preconditioned Toeplitz least squares iterations. *SIAM J. Matrix Anal. Appl.*, 15(1):80–97, 1994.
- [9] R. Chan and M. Ng. Conjugate gradient methods for Toeplitz systems. *SIAM Rev.*, 38:427–482, 1996.
- [10] R. Chan and G. Strang. Toeplitz equations by conjugate gradients with circulant preconditioner. *SIAM J. Sci. Stat. Comput.*, 10(1):104–119, 1989.
- [11] T. F. Chan. An optimal circulant preconditioner for Toeplitz systems. *SIAM J. Sci. Stat. Comput.*, 9:766–771, 1988.
- [12] T. F. Chan and P. Mulet. Iterative methods for total variation image restoration. *Computational and Applied Mathematics report at UCLA*, (96-38), 1996.
- [13] T. F. Chan and J. Olkin. Circulant preconditioners for Toeplitz-block matrices. *Numer. Algorithms*, 6:89–101, 1994.

- [14] P. J. Davis. *Circulant Matrices*. Pure and Applied Mathematics Interscience Monograph. John Wiley and Sons, New York, NY, 1979.
- [15] D. C. Dobson and F. Santosa. Recovery of blocky images from noisy and blurred data. *SIAM J. Appl. Math.*, 56(4):1181–1198, 1996.
- [16] H. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems*. Kluwer Academic Publishers, Dordrecht, 1996.
- [17] L. C. Evans. *Partial Differential Equations*. Graduate Studies in Mathematics. American Mathematical Society, Providence, RI, 1998.
- [18] E. Giusti. *Minimal Surfaces and Functions of Bounded Variation*. Monographs in Mathematics. Birkhauser, Boston, MA, 1984.
- [19] G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, MD, third edition, 1996.
- [20] F. Greensite. Second-order approximation of the pseudoinverse for operator deconvolutions and families of ill-posed problems. *J. Appl. Math.*, 59(1):1–16, 1998.
- [21] C. W. Groetsch. *The theory of Tikhonov regularization for Fredholm equations of the first kind*, volume 105 of *Research Notes in Mathematics*. Pitman Publishing Inc., London, 1984.
- [22] C. W. Groetsch. *Inverse Problems in the Mathematical Sciences*. Vieweg, Braunschweig, 1993.
- [23] J. Hadamard. *Lectures on the Cauchy's Problem in Linear Partial Differential Equations*, volume 105. Yale University Press, New Haven, CT, 1923.
- [24] C. A. Hall and T. A. Porsching. *Numerical Analysis of Partial Differential Equations*. Prentice Hall, Englewood Cliffs, NJ, 1990.
- [25] M. Hanke and J. G. Nagy. Restoration of atmospherically blurred images by symmetric indefinite conjugate gradient techniques. *Inverse Problems*, 12:157–173, 1996.
- [26] M. Hanke and C. R. Vogel. Two-level preconditioners for regularized inverse problems 1: Theory. *Numer. Math.*, to appear.
- [27] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems*. SIAM Monographs on Mathematical Modeling and Computation. SIAM, Philadelphia, PA, 1998.
- [28] C.T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. Frontiers in Applied Mathematics. SIAM, Philadelphia, PA, 1995.

- [29] R. Kettler. Analysis and comparison of relaxation schemes in robust multigrid and preconditioned conjugate gradient methods. In *Multigrid Methods*, volume 960 of *Lecture Notes in Mathematics*, pages 502–534, 1982.
- [30] P. Lancaster and M. Tismenetsky. *The Theory of Matrices with Applications*. Computer Science and Applied Mathematics. Academic Press, New York, NY, second edition, 1985.
- [31] M. M. Lavrent'ev, V. G. Romanov, and S. P. Shishat'skii. *Ill-posed Problems of Mathematical Physics and Analysis*. Translations of Mathematical Monographs. American Mathematical Society, Providence, RI, 1986.
- [32] C.F. Van Loan. *Computational Frameworks for the Fast Fourier Transform*. SIAM, Philadelphia, PA, 1992.
- [33] M. Oman. *Iterative Methods for Total Variation Based Image Reconstruction*. PhD thesis, Montana State Univ., 1995.
- [34] D. L. Phillips. A technique for the numerical solution of certain integral equations of the first kind. *J. Assoc. Comput. Mach.*, 9:84–97, 1962.
- [35] M. Renardy and R. C. Rogers. *An Introduction to Partial Differential Equations*. Texts in Applied Mathematics. Springer-Verlag, New York, NY, 1993.
- [36] A. Rieder. A wavelet multilevel method for ill-posed problems stabilized by Tikhonov regularization. *Numer. Math*, 75:501–522, 1997.
- [37] K. L. Riley and C. R. Vogel. Preconditioners for linear systems arising in image reconstruction. *SPIE Proceedings*, 3461-37:372–380, 1998.
- [38] K. L. Riley and C. R. Vogel. Two-level preconditioners for ill-conditioned linear systems with semidefinite regularization. *J. of Comput. and Appl. Math.*, submitted, 1999.
- [39] M. C. Roggemann and B. Welsh. *Imaging Through Turbulence*. CRC Press, New York, NY, 1996.
- [40] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60:259–268, 1992.
- [41] W. Rudin. *Functional Analysis*. McGraw-Hill Book Company, New York, NY, 1973.
- [42] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, Boston, MA, 1996.
- [43] G. Strang. A proposal for Toeplitz matrix calculations. *Stud. Appl. Math*, 74:171–176, 1986.

- [44] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-posed Problems*. V.H. Winston and Sons, Washington, D.C., 1977.
- [45] C. R. Vogel and M. Hanke. Two-level preconditioners for regularized inverse problems 2: Implementation and numerical results. *SIAM J. Sci. Comput.*, submitted, 1998.
- [46] C. R. Vogel and M. Oman. Iterative methods for total variation denoising. *SIAM J. Sci. Comput.*, 17(1):227–237, 1996.
- [47] C. R. Vogel and M. Oman. Fast, robust total variation-based reconstruction of noisy, blurred images. *IEEE Trans. on Image Processing*, 7:813–824, 1998.
- [48] R. G. Wilson. *Fourier Series and Optical Transform Techniques in Contemporary Optics (an introduction)*. John Wiley and Sons, New York, 1995.
- [49] Kôzaku Yosida. *Functional Analysis*. Springer-Verlag, New York, fourth edition, 1974.
- [50] E. Zeidler. *Applied Functional Analysis Main Principles and Their Applications*, volume 109 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 1995.

MONTANA STATE UNIVERSITY - BOZEMAN



3 1762 10331166 6