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 | Curtis R. Vogel
Date | Curt R. Vogel
Chairperson, Graduate Committee

Approved for the Major Department

7/19/99 | John Lund
Date | John Lund
Head, Mathematical Sciences

Approved for the College of Graduate Studies

7-21-99 | Bruce R. McLeod
Date | Bruce McLeod
Graduate Dean

STATEMENT OF PERMISSION TO USE

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

LIST OF TABLES

LIST OF FIGURES

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

$$A\mathbf{u} = \mathbf{b} \tag{1.1a}$$

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

$$A = K^*K + \alpha L. \tag{1.1b}$$

System (1.1) arises from the minimization of the functional

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

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

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

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

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

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,

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

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

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

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 consideration of the asymptotic behavior when $\alpha \to 0$, 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

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

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 $u : \mathbb{R}^n \to \mathbb{R}^1$, define the *gradient* of u by

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

For a vector valued function $\vec{v} = (v_1, v_2, ..., v_n)$ where each $v_i : \mathbb{R}^n \to \mathbb{R}$ 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

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

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

$$\mathcal{K}u = z, \;\; u \in \mathcal{H}_1, \;\; z \in \mathcal{H}_2, \tag{2.2}$$

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, $\mathcal{K}^{-1} : \mathcal{H}_2 \to \mathcal{H}_1$, exists and is bounded. For a linear $\mathcal{K}$, we will denote the range space by $\mathsf{R}(\mathcal{K})$, the null space by $\mathsf{N}(\mathcal{K})$, and the domain for the operator will be represented by $\mathsf{D}(\mathcal{K})$. 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* $\mathcal{K}^* : \mathcal{H}_2 \to \mathcal{H}_1$ is a linear operator where for every $y \in \mathsf{D}(\mathcal{K}^*)$ 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 $u \in \mathsf{D}(\mathcal{K})$. The adjoint is defined by the mapping $\mathcal{K}^* y = y^*$ for all $y \in \mathsf{D}(\mathcal{K}^*)$.

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

Definition 2.7 Let $\mathcal{K} : \mathcal{H}_1 \to \mathcal{H}_2$ be a continuous linear operator. Define $\tilde{\mathcal{K}}$ by $\tilde{\mathcal{K}} = \mathcal{K}|_{\mathsf{N}(\mathcal{K})^\perp}$, 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

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

with minimal operator norm.

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

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

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

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

Then the Fredholm integral operator of the first kind,

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

is a compact operator with $\mathcal{K} : L^2(\Omega) \to L^2(\Omega)$. 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 $\mathcal{K} : \mathcal{H}_1 \to \mathcal{H}_2$ 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 $n = 1, 2, ...$, otherwise there will be finitely many terms. The functions $\{\psi_n\}$ are an orthonormal basis of $\mathsf{N}(\mathcal{K})^\perp$, and the functions $\{\phi_n\}$ are an orthonormal basis of the closure of $\mathsf{R}(\mathcal{K})$. Furthermore,

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

and

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

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 , $\sigma_1 \geq \sigma_2 \geq ... > 0$. 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

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

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

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

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

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 $R_\alpha : \mathcal{H}_2 \to \mathcal{H}_1$ 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 $R_\alpha(z) \to \mathcal{K}^\dagger z$ as $\alpha \to 0$ for all $z \in \mathsf{D}(\mathcal{K}^\dagger)$ [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

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

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 $\alpha \to 0$ 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. $\mathcal{K} : \mathcal{H}_1 \to \mathcal{H}_2$), Tikhonov regularization is given by the following

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

where $\arg \min\limits_{u \in \mathsf{D}(\mathcal{K})}$ denotes an element out of $\mathsf{D}(\mathcal{K})$ 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

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

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),

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

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

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

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

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

Hence R_α is a bounded operator for each $\alpha > 0$. For any fixed n it is also true that $\lim_{\alpha\to 0}\frac{\sigma_n}{\sigma_n^2+\alpha} = \frac{1}{\sigma_n}$, 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 $\mathcal{K} : L^2(\Omega) \to L^2(\Omega)$. For this thesis we will consider a slightly more general form of Tikhonov regularization,

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

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

$$\begin{aligned} b(u, v) &= \int_\Omega \kappa(x) \sum_{i=1}^n \frac{\partial u}{\partial x_i}\frac{\partial v}{\partial x_i} dx && (2.14) \\ &= -\int_\Omega \nabla \cdot (\kappa(x)\nabla u)\, v\, dx && (2.15) \\ &= \langle \mathcal{L}u, v\rangle, && (2.16) \end{aligned}$$

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

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

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 Dirchlet 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

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

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,

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

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

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

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

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

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

$$|u|_{\mathrm{TV}} = \sup_{\vec{w} \in \mathcal{W}} \int_\Omega -u \nabla \cdot \vec{w}\, dx, \tag{2.22}$$

where

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

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 $|u|_{\mathrm{TV}} < \infty$. We will denote this space by $\mathrm{BV}(\Omega)$.

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

$$|u|_{\mathrm{TV}} = \sup_{\vec{w} \in \mathcal{W}} \int_\Omega \vec{w} \cdot \nabla u \, dx. \tag{2.23}$$

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

$$|u|_{\mathrm{TV}} = \int_\Omega |\nabla u| \, dx. \tag{2.24}$$

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

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

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

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

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

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

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

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

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

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

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,

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

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.

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

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 $\mathbf{t} = (t_{1-m}, t_{2-m}, ..., t_0, ..., t_{m-1})$. This relationship will be denoted by $T = \mathtt{teoplitz}(\mathbf{t})$.

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.,

$$C = \begin{bmatrix} c_0 & c_{n-1} & c_{n-2} & \cdots & c_1 \\ c_1 & c_0 & c_{n-1} & \cdots & c_2 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ c_{n-2} & & \ddots & \ddots & c_{n-1} \\ c_{n-1} & c_{n-2} & \cdots & c_1 & c_0 \end{bmatrix}. \tag{2.30}$$

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

A BTTB matrix has the following structure

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

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 $T_j = \texttt{toeplitz}(\mathbf{t}_j)$ for each $j = 1-n, 2-n, ..., n-1$. Hence, the size of the matrix K is $nm \times nm$. We use the notation $K = \texttt{BTTB}(k)$ 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 $b = [\mathbf{c}_0\ \mathbf{c}_1\ ...\ \mathbf{c}_n]$, where $\mathbf{c}_j$ is the first column of the circulant block C_j. The analogous expression $\texttt{BCCB}(b)$ will represent the BCCB matrix generated by the array b.

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

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

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: $x_j = -p + (2j - 1)\frac{\Delta x}{2}$, $j = 1, 2, ...n$. By midpoint quadrature, the expression in (2.31) takes the form

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

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

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 $x_i = -p + (2i-1)\frac{\Delta x}{2}$, $i = 1, 2, ...n$, and $y_r = -p + (2r-1)\frac{\Delta y}{2}$,

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

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

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

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

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

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

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

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

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

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

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:

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

where

$$\begin{aligned}
\kappa_+(\bar{u}) &= \frac{1}{\sqrt{(\Delta_+^x \bar{u})^2 + (\Delta_+^y \bar{u})^2 + \beta^2}} \\
\Delta_+ u &= (\Delta_+^x u + \Delta_+^y u) \\
\Delta_- u &= (\Delta_-^x u + \Delta_-^y u) \\
\Delta_\pm^x u &= \pm\left(\frac{u_{i\pm1,j} - u_{i,j}}{\Delta x}\right) \\
\Delta_\pm^y u &= \pm\left(\frac{u_{i,j\pm1} - u_{i,j}}{\Delta y}\right).
\end{aligned}$$

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

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

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*

```
$k = 0$;
$\mathbf{r}_0 = \mathbf{b} - A\mathbf{u}_0$;
$\delta_0 = \mathbf{r}_0^*\mathbf{r}_0$;
Begin CG iterations
    if $k = 0$
        $\mathbf{p}_1 = \mathbf{r}_0$;
    else
        $\beta_{k+1} = \frac{\delta_k}{\delta_{k-1}}$;
        $\mathbf{p}_{k+1} = \mathbf{r}_k + \beta_{k+1}\mathbf{p}_k$;
    end
    $\mathbf{d}_{k+1} = A\mathbf{p}_{k+1}$;
    $\alpha_{k+1} = \frac{\delta k}{\mathbf{p}_{k+1}^*\mathbf{d}_{k+1}}$;
    $\mathbf{u}_{k+1} = \mathbf{u}_k + \alpha_{k+1}\mathbf{p}_{k+1}$;
    $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_{k+1}\mathbf{d}_{k+1}$;
    $k = k + 1$;
    $\delta_k = \mathbf{r}_k^*\mathbf{r}_k$;
end CG iterations
$\mathbf{u} = \mathbf{u}_{k+1}$
```

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 abscence 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

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

The CG descent vectors $\{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_k\}$ 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

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

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

$\mathtt{cond}(A)$ is called the spectral condition number for A. For brevity we will often refer to $\mathtt{cond}(A)$ 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

$$C^{-1}AC^{-1}\mathbf{y} = C^{-1}\mathbf{b}, \tag{2.38}$$

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*

```
$k = 0$;
$\mathbf{r}_0 = \mathbf{b} - A\mathbf{u}_0$;
Begin CG iterations
        Solve $M\mathbf{z}_k = \mathbf{r}_k$
        $\delta_k = \mathbf{r}_k^* \mathbf{z}_k$;
        if $k = 0$
            $\mathbf{p}_1 = \mathbf{z}_0$;
        else
            $\beta_{k+1} = \dfrac{\delta_k}{\delta_{k-1}}$;
            $\mathbf{p}_{k+1} = \mathbf{z}_k + \beta_{k+1}\mathbf{p}_k$;
        end
        $\mathbf{d}_{k+1} = A\mathbf{p}_{k+1}$;
        $\alpha_{k+1} = \dfrac{\delta_k}{\mathbf{p}_{k+1}^* \mathbf{d}_{k+1}}$;
        $\mathbf{u}_{k+1} = \mathbf{u}_k + \alpha_{k+1}\mathbf{p}_{k+1}$;
        $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_{k+1}\mathbf{d}_{k+1}$;
        $k = k + 1$;
end CG iterations
$\mathbf{u} = \mathbf{u}_{k+1}$
```

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 $\texttt{cond}(C^{-1}AC^{-1}) = \texttt{cond}(M^{-1}A)$. Thus $\texttt{cond}(M^{-1}A)$ 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 $\mathbf{f} = [f_0, f_1, ..., f_{n-1}] \in \mathbb{C}^n$ the transform is denoted by

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

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

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

for any complex valued n-vector $\mathbf{F} = [F_0, F_1, ..., F_{n-1}]$. 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

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

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

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

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

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:

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

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

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

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 $K = \texttt{circulant}(\tilde{\mathbf{k}})$ 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

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

Let `vector` denote the reverse operation of taking an array and rearranging the elements back into a column vector. Finally, let $\hat{c} = \mathtt{FFT2}(\mathtt{array}(\mathbf{c}_1))$, 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

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

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

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

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 $\hat{k}_{ext} = \texttt{FFT2}(k_{ext})$ 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`,

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

so that $\texttt{extend}(r)$ 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

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

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

$$t = \texttt{restrict}\left(\texttt{IFFT2}(\left|\texttt{FFT2}(k_{ext})\right|. \wedge 2)\right), \tag{3.10}$$

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 $t = \texttt{array}(\mathbf{t})$, 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

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

where $\mathbf{t}_i$ is the ith column of t. Define the array operator **embed** by extending t in the following way

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

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

$$t_{ext} = (\texttt{embed}(\texttt{embed}(t)^*))^* \ . \tag{3.11}$$

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

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

where $\hat{t}_{ext} = \texttt{FFT2}(t_{ext})$. 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*
$\mathbf{s} = M_{circ}^{-1}\mathbf{r}$

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

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 $\mathbf{t} = [t_1, t_2, ..., t_n]$. The computation of $\mathbf{t}$ is given by

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

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

$$\begin{aligned}
\mathbf{t}_1 &= \frac{\sum_{k=1}^{n_x^2} a_{k,k}}{n_x^2} \\
\mathbf{t}_2 &= \frac{\sum_{j=0}^{n_x-1} \sum_{k=2}^{n_x} a_{(jn_x+k),(jn_x+k-1)}}{n_x(n_x-1)} \\
\mathbf{t}_{n_x} &= \frac{\sum_{j=1}^{n_x^2-n_x} \left(a_{(n_x+j),j}\right)}{n_x(n_x-1)},
\end{aligned}$$

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

$$A\mathbf{u} = \mathbf{b}, \tag{4.1a}$$

where

$$A = K^*K + \alpha L, \tag{4.1b}$$

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 $V_\Psi = \text{span}\left(\{\psi_j\}_{j=1}^{n-N}\right)$. In principle, the bases for both subspaces can be chosen such that for every appropriate i and j,

$$\phi_i^* L \psi_j = 0, \tag{4.2a}$$

$$\psi_i^* L \psi_j = \delta_{ij}, \tag{4.2b}$$

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, $\{\phi_i\}_{i=1}^{N} \cup \{\psi_j\}_{i=1}^{n-N}$ 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

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

The transformed system has a block representation given by

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

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

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

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

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

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

$$G = \Phi^* L \Phi \tag{4.7a}$$

$$P = \Phi G^{-1} \Phi^* L \tag{4.7b}$$

$$Q \equiv I - P = \Psi \Psi^* L. \tag{4.7c}$$

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 $P^* L Q = 0$. For any $\check{\mathbf{y}} \in V_\Phi$ there exist $\mathbf{y} \in R^N$ such that $\check{\mathbf{y}} = \Phi \mathbf{y}$. To compute $P\check{\mathbf{y}}$ the equations in (4.7) are used and this results in $P\check{\mathbf{y}} = \check{\mathbf{y}}$. Moreover,

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

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.,

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

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

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

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

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

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

$$\mathbf{u}_{\mathrm{AS}} = \Phi\mathbf{x}_1 + \Psi\mathbf{x}_2. \tag{4.10}$$

Using the equations in (4.7) the following transpires:

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

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

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

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