2019

On the Role of Ill-conditioning: Biharmonic Eigenvalue Problem and Multigrid Algorithms

Kasey Bray
University of Kentucky, kasey.bray@uky.edu
Digital Object Identifier: https://doi.org/10.13023/etd.2019.173

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Kasey Bray, Student

Dr. Qiang Ye, Major Professor

Dr. Peter Hislop, Director of Graduate Studies

On the Role of Ill-conditioning: Biharmonic Eigenvalue Problem and Multigrid Algorithms

DISSERTATION

A dissertation submitted in partial
fulfillment of the requirements for
the degree of Doctor of Philosophy
in the College of Arts and Sciences
at the University of Kentucky

By
Kasey Bray
Lexington, Kentucky

Director: Dr. Qiang Ye, Professor of Mathematics
Lexington, Kentucky

2019

ABSTRACT OF DISSERTATION

On the Role of Ill-conditioning: Biharmonic Eigenvalue Problem and Multigrid Algorithms

Very fine discretizations of differential operators often lead to large, sparse matrices A, where the condition number of A is large. Such ill-conditioning has well known effects on both solving linear systems and eigenvalue computations, and, in general, computing solutions with relative accuracy independent of the condition number is highly desirable. This dissertation is divided into two parts.

In the first part, we discuss a method of preconditioning, developed by Ye, which allows solutions of Ax=b to be computed accurately. This, in turn, allows for accurate eigenvalue computations. We then use this method to develop discretizations that yield accurate computations of the smallest eigenvalue of the biharmonic operator across several domains. Numerical results from the various schemes are provided to demonstrate the performance of the methods.

In the second part we address the role of the condition number of A in the context of multigrid algorithms. Under various assumptions, we use rigorous Fourier analysis on 2- and 3-grid iteration operators to analyze round off errors in floating point arithmetic. For better understanding of general results, we provide detailed bounds for a particular algorithm applied to the 1-dimensional Poisson equation. Numerical results are provided and compared with those obtained by the schemes discussed in part 1.

KEYWORDS: Accuracy, biharmonic operator eigenvalue problem, preconditioning, multigrid algorithms, rigorous Fourier analysis, roundoff errors

Author's signature: Kasey Bray

Date: May 2, 2019

On the Role of Ill-conditioning: Biharmonic Eigenvalue Problem and Multigrid Algorithms

By
Kasey Bray

Director of Dissertation: Qiang Ye

Director of Graduate Studies: Peter Hislop

Date: May 2, 2019

ACKNOWLEDGMENTS

I would first like to acknowledge my advisor, Dr. Qiang Ye, for providing continued support, guidance, insight, and suggestion. This project, it's approaches, ideas, and execution, would not have been possible with out him. Thank you to my committee members for your time and input over the years. And thank you to countless department faculty and staff for your availability and willingness to help.

A special shout out to R.Rogers for being the most supportive friend, even from 900 miles away; to SheOpenstheBills for keeping me grounded and not getting too big for my britches; to LaLa and Nicoleslaw for always making me feel like a rockstar; and to my okayest friend in KY, our unexpected friendship has been such a delight and truly made my time here better. Y'all the real MVP.

And finally to friends and family acquired through the math department: thanks for all the adventures in golf, concerts, karaoke, grilling, and backyard games. Thanks for therapeutic complaining sessions, keeping imposter syndrome in check, and some semblance of school-life balance. Here's to educated degenerates and lasting friendships.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1 Introduction

In this thesis we are concerned with accurate computations of solutions to large scale linear algebra problems where the coefficient matrix A is ill-conditioned. That is, where the condition number $\kappa(A) := \|A\|\|A^{-1}\|$ is large. We focus on the large, sparse systems which arise from the discretization of differential operators and, specifically, we use only the finite difference method for obtaining discretized systems. These matrices have condition numbers that increase as the discretization meshsize h decreases. In particular, the condition number of a second order operator (such as the Laplacian Δ) is of order $\mathcal{O}(h^{-2})$, while the condition number of a fourth order operator (such as the biharmonic operator Δ^2) is typically of order $\mathcal{O}(h^{-4})$. Moreover, ill-conditioning has well known (negative)effects on solving linear algebra problems, which puts us in an unfortunate situation: finer discretizations yield discrete systems whose exact solution better approximates the true continuous solution; however, the resulting large condition number produces roundoff errors in floating point arithmetic which prevent accurate computations of the discrete solutions.

In practice, ill-conditioning has two general effects on linear equation solving: it reduces the rate of convergence of iterative algorithms, and it limits the accuracy to which $A\mathbf{x} = \mathbf{b}$ can be solved in finite precision. Preconditioning is a well known technique for addressing the convergence issue, but, historically, there seems to be a lack of good remedies for addressing the accuracy issue. Standard backward stable algorithms (e.g. Cholesky) compute a solution $\hat{\mathbf{x}}$ with relative error bounded by $\mathcal{O}(\epsilon)\kappa_2(A)$ – which may be unsatisfactory for ill-conditioned problems. Ill-conditioning has similar effects on solving eigenvalue problems. In fact, as we will see in Chapter 2, the inability to accurately compute smaller eigenvalues boils down to an inability to accurately solve a linear system. In short, an ill-conditioned matrix is often associated with a clustered spectrum, which results in slow convergence of iterative eigenvalue solvers; slow convergence is circumvented by applying the algorithm at hand to A^{-1} (which has a much better spectral gap), but this requires the availability of A^{-1} explicitly, or that we solve a linear system to obtain it. Therefore, existing eigenvalue algorithms (e.g. Lanczos) compute smaller eigenvalues with relative error of order $\mathcal{O}(\epsilon)\kappa_2(A)$, and we may expect little accuracy out of these algorithms when A is ill conditioned.

For these reasons, computing solutions with relative accuracy independent of the condition number of A is highly desirable, and is the focus on this thesis. We primarily study a fourth order biharmonic (Δ^2) eigenvalue problem and a second order linear system based on the Laplacian operator Δ, and the methods for solving these problems separates this thesis into two parts.

Part I deals with accurate computations of the smallest eigenvalue of the biharmonic operator: discretizations which allow for the application of the accurate preconditioning scheme are paired with classical iterative eigenvalue algorithms in order to compute the smallest eigenvalue of the biharmonic eigenvalue problem with Dirichlet boundary conditions to high accuracy. We solve this problem on three different

domains: the unit interval, the unit square, and the unit circle.

In Part II we turn our attention to the accuracy of a different set of techniques for solving partial differential equations – multigrid algorithms. Using rigorous Fourier analysis, we analyze roundoff errors in a multigrid V-cycle, then apply the results to a Model Problem (Poisson's equation) using a Model Algorithm. In this case, explicit dependence on the condition number is not found, but we do find that the relative error of a solution computed using an l-grid V-cycle is of order $\mathcal{O}(N^2)$, and, for Poisson's equation, $\kappa_2(A)$ is $\mathcal{O}(N^2)$. This suggests that the condition number might play a small role; furthermore, a numerical example reveals that a full multigrid algorithm outperforms a backward stable algorithm (which depends on the condition number) for large N, but under-performs compared with condition number independent methods from Part I. The work of Chapter 7 and Chapter 8 does not yet explain the results of this example, but these preliminary results lay a foundation a deeper exploration of the explicit role of the condition number.

Future work on this problem involves the application of a standard multigrid algorithm to a fourth order operator in order to explore the N^2 dependence; extension to the full multigrid algorithm; and, revisiting the methods and approach of Chapter 7 in search of an explicit dependence or independence on the condition number.

1.1 Preliminaries and Notation

Throughout this paper, $\|\cdot\|$ denotes a general vector norm and its induced operator norm for matrices. $\|\cdot\|_p$ denotes the $p-$norm, while inequalities and absolute values involving matrices and vectors are entrywise. For future use and completion, we note a few essential properties of norms in the following lemma.

Lemma 1.1. *1.* $\|A\mathbf{x}\| \leq \|A\| \cdot \|\mathbf{x}\|$ *for a vector norm and its corresponding matrix norm. And,* $\|AB\| \leq \|A\| \cdot \|B\|$ *for any operator norm.*

2. $\|QAZ\| = \|A\|$ *if* Q *and* Z *are orthogonal.*

3. $\|A\|_2 = \max_i |\lambda_i(A)|$ *if* A *is normal. And for any* A, $\|A\|_2 = \sqrt{\lambda_{\max}(A^*A)}$.

Most commonly, we will employ the 2-norm $\|\cdot\|_2$, but note that in finite dimensional space, any two norms are equivalent. Unless explicitly stated otherwise, $\kappa(A)$ will denote the condition number of A using the 2-norm.

For error analysis in floating point arithmetic, ϵ denotes the machine round off unit, and $\mathcal{O}(\epsilon)$ denotes a term bounded by $p(N)\epsilon$ for some polynomial $p(N)$ in N. We use $fl(z)$ or $\hat{z}$ to denote the computed result of an algebraic expression z. We assume throughout that matrices and vectors given have floating point number entries. We assume the following standard model for roundoff errors in basic matrix computations:

$$fl(x+y) = x + y + e \text{ with } |e| \leq \epsilon(|x+y|), \tag{1.1}$$

$$fl(A\mathbf{x}) = A\mathbf{x} + e \text{ with } |e| \leq \epsilon M|A||\mathbf{x}| + \mathcal{O}(\epsilon^2), \tag{1.2}$$

where M is the maximal number of nonzero entries per row of A, and absolute value inequalities are entrywise. In general, we may rewrite eq. (1.2) as

$$\|fl(A\mathbf{x}) - A\mathbf{x}\| \leq \mathcal{O}(\epsilon)\|A\|\|\mathbf{x}\|. \tag{1.3}$$

It is worth noting that the condition number $\kappa(A) \coloneqq \|A^{-1}\|\|A\|$ is explicitly the condition number with respect to the problem of matrix inversion. It measures the relative change in the answer ($\hat{\mathbf{x}}$) as a multiple of the relative change in the data (A). Geometrically, the condition number can be characterized in the following way: if A is nonsingular, then the relative distance to the nearest singular matrix (ill-posed problem) is the reciprocal of the condition number.

Part I

The Biharmonic Eigenvalue Problem

Chapter 2 Ill-conditioning and the Biharmonic Eigenvalue Problem

Consider a vibrating plate on a domain Ω, clamped on the boundary $\partial\Omega$ as described by the biharmonic eigenvalue problem with Dirichlet boundary conditions:

$$\begin{cases} \Delta^2 v = \lambda v \text{ in } \Omega \\ v = \dfrac{\partial v}{\partial n} = 0 \text{ on } \partial\Omega. \end{cases} \tag{2.1}$$

A discretization with mesh-size $h \sim \frac{1}{N}$ of such an operator typically results in a large, sparse $N \times N$ matrix, A_N. Using a very fine discretization means the computed matrix eigenvalue is a better approximation to the true eigenvalue of eq. (2.1); however, a very fine discretization results in an A_N which may be extremely ill-conditioned, causing large round-off errors in the computed eigenvalue. In general, as h decreases, the condition number increases and the relative accuracy of smaller eigenvalues as computed by existing algorithms deteriorates ([11], [28]). In fact, for a fourth order operator, $\kappa(A_N)$ is of order $\mathcal{O}(h^{-4})$ which implies that existing eigenvalue algorithms may compute smaller eigenvalues with little or no accuracy in standard double precision for h smaller than $\approx 10^{-4}$.

Bjørstad and Tjøstheim [2] present solutions to eq. (2.1) and the buckling plate problem using a highly accurate spectral Legendre-Galerkin method in which a clever basis developed by Shen [20] is combined with a fast solution algorithm [3] to accurately compute eigenpairs. Their computations agree with various known theoretical properties of eigenpairs; however, in order to obtain both precision and accuracy, they had to use quadruple precision to compute the eigenvalues of a matrix obtained from up to 5000 basis functions. Existing matrix eigenvalue solvers implemented in standard double precision do not provide satisfactory accuracy at this resolution.

Here, in Part I, we use the work of Ye in [26], [27], and [28] on combining existing matrix eigenvalue solvers together with a preconditioning scheme to solve a matrix eigenvalue equation. Specifically, we use standard finite difference methods applied to eq. (2.1) to accuratly compute the smallest eigenvalue λ_1. Ill-conditioning encountered in this approach presents two general issues: convergence and accuracy.

The slow convergence of iterative algorithms on ill-conditioned matrices can be overcome by preconditioning, but this is only practical if the preconditioned system can be formed exactly or sufficiently accurately. In particular, if the preconditioner can be solved with inverse-equivalent accuracy (which is equivalent to multiplying exact inverses), then we can solve both the convergence and the accuracy issue caused by ill-conditioning in eigenvalue problems [26]. We will show that, in spite of extreme ill-conditioning, high accuracy is achieved in the computed eigenvalues. Since the eigenvalues at the left end of the spectrum of the differential operator are typically of interest in practical problems, we focus on computing the smallest eigenvalue, λ_1.

We begin by addressing in some detail the convergence and accuracy issues encountered when computing a few smaller eigenvalues of an extremely ill-conditioned

matrix A. Subsequent chapters are dedicated to the schemes for addressing these issues, and an application of the schemes to eq. (2.1).

2.1 Issues in Computing Smaller Eigenvalues of Ill-conditioned Matrices

In this section, we summarize the discussion and some conclusions of [28, Section 3] on the computations of smaller eigenvalues of extremely ill-conditioned matrices. The discussion is restricted to a matrix A that is symmetric positive definite.

Consider an ill-conditioned matrix A with eigenvalues $0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N$. In this case, λ_1 has relative spectral gap

$$\frac{\lambda_2 - \lambda_1}{\lambda_N - \lambda_2} = \frac{\lambda_2 - \lambda_1}{\lambda_1} \frac{1}{\lambda_N/\lambda_1 - \lambda_2/\lambda_1} \approx \frac{\lambda_2 - \lambda_1}{\lambda_1} \frac{1}{\kappa(A)},$$

where, unless $\lambda_2 \approx \lambda_N$, this spectral gap is small (i.e. λ_1 is clustered). And since the speed of convergence of classical eigenvalue algorithms (such as Lanczos) is determined by the relative spectral gap, a direct application to A will result in slow or no convergence. A^{-1}, on the other hand, has a better relative spectral gap:

$$\frac{\lambda_1^{-1} - \lambda_2^{-1}}{\lambda_2^{-1} - \lambda_N^{-1}} > \frac{\lambda_2 - \lambda_1}{\lambda_1}.$$

This suggests that we can deal with clustering by applying Lanczos to A^{-1} to compute the largest eigenvalue $\mu_1 = \lambda_1^{-1}$, from which we can recover λ_1. Unfortunately, μ_1 can only be computed accurately if A^{-1} is explicitly available, or if matrix-vector multiplication with A^{-1} can be accurately computed (i.e. if we can solve a linear system $A\mathbf{x} = \mathbf{b}$ accurately). But when A is ill-conditioned, this is not possible because backward stable algorithms produce computed solutions $\hat{\mathbf{x}}$ to linear systems $A\mathbf{x} = \mathbf{b}$ of order $\epsilon \|A^{-1}\| \|A\| \|\hat{\mathbf{x}}\| = \epsilon \kappa(A) \|\hat{\mathbf{x}}\|$.

This point is further illustrated in [28] by looking at the Rayleigh quotient. All iterative methods are based on constructing an approximate eigenvector x_1 then computing an approximate eigenvalue via the Rayleigh quotient

$$\rho_1 = \frac{x_1^T A^{-1} x_1}{x_1^T x_1}.$$

It is clear that the accuracy of the computed Rayleigh quotient depends on the accuracy of computing $A^{-1}x_1$, and hence it is shown ([28]) that the computed $\hat{\rho}_1$ satisfies

$$|\hat{\rho}_1 - \rho_1| \leq \mathcal{O}(\epsilon) \kappa_2(A) \mu_1.$$

With $\rho_1 \leq \mu_1$ the relative error of the computed Rayleigh quotient is expected to be of order $\mathcal{O}(\epsilon)\kappa(A)$. What's further, this bound is independent of the algorithm used to obtain the eigenvector x_1 – in fact, this is the case even if x_1 is available exactly. We refer the reader to [28] and the references therein for further details and theory on the computed accuracy of smaller eigenvalues of a matrix.

This brief survey, however, is enough to highlight that the root cause of an inability to compute eigenvalues accurately is in the roundoff errors encountered in computing matrix-vector multiplication with A^{-1}. This also suggests a remedy: computed A^{-1} (or solve $A\mathbf{x} = \mathbf{b}$) more accurately. Chapter 3 addresses a scheme for achieving this remedy.

Chapter 3 Accurate Preconditioning

This chapter addresses the issue of roundoff errors accrued in solving ill-conditioned linear systems. First, in Section 3.1, the inverse-equivalent accuracy measure of [26] is defined, and relevant discussion on when an how such an accuracy can be achieved is provided. Section 3.2 states the Accurate LDU algorithm, originally developed in [27], as the predominant inverse-equivalent algorithm used in our work of Chapter 4. Finally, Section 3.3 outlines the idea of accurate preconditioning from [26] and sufficient requirements for the process.

3.1 Inverse-equivalent Accuracy and Rank-revealing Decompositions

In solving $A\mathbf{x} = \mathbf{b}$, one may ideally strive for full accuracy from the computed solution $\hat{\mathbf{x}}$ such that

$$\frac{\|\hat{\mathbf{x}} - \mathbf{x}\|}{\|\mathbf{x}\|} \leq \mathcal{O}(\epsilon).$$

But in the absence of very stringent conditions on the original system, this is unrealistic. Traditionally, the best accuracy one can hope for is that of a backward stable algorithm in which the solution error is bounded as

$$\frac{\|\hat{\mathbf{x}} - \mathbf{x}\|}{\|\mathbf{x}\|} \leq \mathcal{O}(\epsilon)\kappa(A), \tag{3.1}$$

but when $\kappa(A)$ is large (as is the case in many differential eigenvalue problems), this bound is simply not good enough. Somewhere in between the unrealistic full-accuracy and the unsatisfactory backward stability is the inverse-equivalent accuracy introduced in [26], which proves to be sufficient in many applications.

Definition 3.1. *([26]) We say that an algorithm for solving linear systems with coefficient matrix A is an* ***inverse-equivalent algorithm*** *if, for any vector $\mathbf{b}$, it produces in floating point arithmetic a computed solution $\hat{\mathbf{x}}$ to $A\mathbf{x} = \mathbf{b}$ such that*

$$\|\hat{\mathbf{x}} - \mathbf{x}\| \leq \mathcal{O}(\epsilon)\|A^{-1}\| \cdot \|\mathbf{b}\|. \tag{3.2}$$

Similarly, we say that a solution $\hat{\mathbf{x}}$ satisfying eq. (3.2) is an ***inverse-equivalent solution****.*

Per its suggestive name, an inverse-equivalent algorithm produces solutions that are comparable to the one obtained by multiplying the exact inverse with the right-hand side vector, and such an accuracy is highly satisfactory in eigenvalue computations with inverse iteration [26]. The existence of inverse-equivalent algorithms arise most notably from accurate rank-revealing decompositions (RRD). First introduced by Demmel et. al. in [11], the accurate RRD was further explored by Dopico and Molera in [14] where solutions to linear systems computed via the accurate RRD are shown to be inverse-equivalent. Explicitly, we have the following.

Definition 3.2. *([11]) A factorization $A = XDY$ is said to be* ***rank-revealing*** *if X and Y are well conditioned and D is diagonal and nonsingular; furthermore, the factorization is said to be* ***accurate rank-revealing*** *if the computed factors $\hat{X}, \hat{D}$, and $\hat{Y}$ satisfy*

$$\frac{\|\hat{X} - X\|}{\|X\|} \leq \epsilon \cdot p(N); \quad \frac{\|\hat{Y} - Y\|}{\|Y\|} \leq \epsilon \cdot p(N); \quad \textit{and} \quad \frac{|\hat{D} - D|}{|D|} \leq \epsilon \cdot p(N), \tag{3.3}$$

where ϵ is the machine roundoff unit and $p(N)$ is a polynomial in N.

Using the accurate RRD of a coefficient matrix to solve linear systems is an inverse-equivalent algorithm.

Theorem 3.1. *([14]) Let $\hat{X}, \hat{D}$, and $\hat{Y}$ be the computed factors of an accurate rank-revealing factorization (i.e. they satisfy (3.3)), and let X, D, and Y be the corresponding exact factors of the rank revealing factorization $A = XDY$. Assume that the systems $X\mathbf{s} = \mathbf{b}$, $D\mathbf{w} = \mathbf{s}$, and $Y\mathbf{x} = \mathbf{w}$ are solved with a backward stable algorithm that, when applied to any linear system $B\mathbf{z} = \mathbf{c}$, computes a solution $\hat{\mathbf{z}}$ that satisfies $(B + \Delta B)\hat{\mathbf{z}} = \mathbf{c}$; with $\|\Delta B\| \leq \epsilon q(N)\|B\|$ where $q(N)$ is a modestly growing function of N such that $q(N) \geq 4\sqrt{2}/(1 - 12\epsilon)$. Let $g(N) := p(N) + q(N) + \epsilon p(N)q(N)$. Then, if $\hat{x}$ is the computed solution of $A\mathbf{x} = \mathbf{b}$ through solving*

$$\hat{X}\mathbf{s} = \mathbf{b}; \quad \hat{D}\mathbf{w} = \mathbf{s}; \quad \textit{and} \quad \hat{Y}\mathbf{x} = \mathbf{w}, \tag{3.4}$$

and if $\epsilon g(N)\kappa(Y) < 1$ and $\epsilon g(N)(2 + \epsilon g(N))\kappa(X) < 1$, then

$$\frac{\|\hat{\mathbf{x}} - \mathbf{x}\|}{\|\mathbf{x}\|} \leq (\epsilon g(N) + \mathcal{O}(\epsilon^2))\max\{\kappa(X), \kappa(Y)\}\frac{\|A^{-1}\|\|\mathbf{b}\|}{\|\mathbf{x}\|} \tag{3.5}$$

where $\|\cdot\|$ is any norm satisfying $\|\mathrm{diag}\{d_i\}\| = \max_i |d_i|$.

We note that if two matrices A_1 an A_2 both have an accurate RRD, then solving the system $A_1A_2\mathbf{x} = \mathbf{b}$ through $A_1\mathbf{w} = \mathbf{b}$ and $A_2\mathbf{x} = \mathbf{w}$ will produce an inverse-equivalent solution provided $\|A_1^{-1}\|\|A_2^{-1}\|/\|(A_1A_2)^{-1}\|$ is a modest number [28].

There are several classes of matrices that have been shown to have an accurate RRD (see [11]), but for our purposes we focus on the class of diagonally dominant matrices, whose accurate RRD can be obtained from the accurate LDU algorithm of [27].

3.2 Accurate LDU Factorization

Diagonally dominant matrices arise in many applications – specifically, the discretization matrix of the Laplacian operator, Δ, is diagonally dominant, and has a dominating presence throughout this work. In this section we present the algorithm for accurately factoring diagonally dominated matrices from [27], as well as a related result which is key in the success of the methods employed in Chapter 4.

Definition 3.3. *A matrix $A = (a_{ij})$ is said to be* ***column diagonally dominant*** *if*

$$|a_{ii}| \geq \sum_{j=1, j\neq i}^{N} |a_{ji}|. \tag{3.6}$$

Similarly, A is said to be ***row diagonally dominant*** *if*

$$|a_{ii}| \geq \sum_{j=1, j\neq i}^{N} |a_{ij}|. \tag{3.7}$$

Diagonally dominant matrices have several nice properties, many of which are exploited in [27] to develop a variation of Gaussian elimination called the accurate LDU ($ALDU$) factorization. This algorithm computes an accurate LDU factorization of a diagonally dominant matrix with computational complexity similar to that of the classic LDU factorization, and is shown to be an accurate rank-revealing decomposition. Throughout this section we will employ the $ALDU$ algorithm as our primary method for obtaining an accurate RRD. For completion and convenince, we include here both the accurate LDU algorithm, and relevant theorem characterizing the accuracy it achieves. We first need some notation associated with reparameterizing a diagonally dominant matrix.

Definition 3.4. *([27]) Given a matrix $M = (m_{ij}) \in \mathbb{R}^{N\times N}$ with zero diagonals and $\mathbf{v} = (v_i) \in \mathbb{R}^N$, we use $\mathcal{D}(\mathbf{M}, \mathbf{v})$ to denote the matrix $A = (a_{ij})$ whose off-diagonal entries are the same as M, and whose i^{th} diagonal entry is $v_i + \sum_{j\neq i} |m_{ij}|$. That is*

$$a_{ij} = \begin{cases} m_{ij} & i \neq j \\ v_i + \sum_{j\neq i} |m_{ij}| & else. \end{cases}$$

In this notation, a diagonally dominant matrix A satisfies

$$v_i = a_{ii} - \sum_{j\neq i} |a_{ij}| \geq 0. \tag{3.8}$$

Moreover, for any matrix $A = (a_{ij})$, let A_D be the matrix with zeros on the diagonal and off diagonal entries are the same as those of A, and let $\mathbf{v} = (v_i)$ be as in eq. (3.8). Then we can write $A = \mathcal{D}(A_D, \mathbf{v})$ – the representation of A by its diagonally dominant part $\mathbf{v}$. This representation is desirable because under entrywise perturbations, $(A_D, \mathbf{v})$ determines all the entries of A to the same relative accuracy, but not vice versa. Namely, $(A_D, \mathbf{v})$ contains more information than do the entries of A (see [27], [28]). Now, the accurate LDU factorization carries out Gaussian elimination on $(A_D, \mathbf{v})$ – in which case the entries of $\mathbf{v}$ can be carried out with no subtraction.

Algorithm 3.1. *([27])* ***Accurate*** *LDU* ***factorization of*** $\mathcal{D}(A_D, \mathbf{v})$

Input: $A_D = (a_{ij})$ *and* $\mathbf{v} = (v_i) \geq 0$*;*
Initialize: $P = I,\ L = I,\ D = 0,\ U = I.$
For $k = 1 : (N-1)$
    For $i = k : N$
        $a_{ii} = v_i + \sum_{j=k, j\neq i}^{N} |a_{ij}|;$
    End For.
    If $\max_{i\geq k} a_{ii} = 0$*, stop;*
    Else choose a permutation P_1 *for pivoting s.t.* $A = P_1 A P_1$ *satisfies one of:*
        (a) if diagonal pivoting: $a_{kk} = \max_{i\geq k} a_{ii}$*;*
        (b) if column diagonal pivoting: $0 \neq a_{kk} \geq \sum_{i=k+1}^{N} |a_{ik}|;$
    $P = P_1 P;\ L = P_1 L P_1;\ U = P_1 U P_1;\ d_k = a_{kk};$
    For $i = (k+1) : N$
        $l_{ik} = a_{ik}/a_{kk};\ u_{ki} = a_{ki}/a_{kk};\ a_{ik} = 0;$
        $v_i = v_i + |l_{ik}| v_k;$
        For $j = (k+1) : N$
            $p = sign(a_{ij} - l_{ik} a_{kj});$
            $s = sign(a_{ij}) p;$
            $t = -sign(l_{ik}) sign(a_{kj}) p;$
            If $j = i$
                $s = 1;\ t = sign(l_{ik}) sign(a_{ki});$
            End If
            $v_i = v_i + (1-s)|a_{ij}| + (1-t)|l_{ik} a_{kj}|;$
            $a_{ij} = a_{ij} - l_{ik} a_{kj};$
        End For
    End For
End For
$a_{nn} = v_n;\ d_n = a_{nn}.$

In the output of Algorithm 3.1 we have $PAP^T = LDU$, where the column diagonal dominance pivoting ensures that L is column diagonally dominant while U is still row diagonally dominant. This theoretically guarantees that L and U are well conditioned (see [27],[28]):

$$\kappa_\infty(L) \leq N^2 \quad \text{and} \quad \kappa_\infty \leq 2N.$$

Moreover, the computed factors $\hat{L}, \hat{D}$ and $\hat{U}$ are characterized by the following theorem.

Theorem 3.2. *([13],[27]) Let* $\hat{L} = (\hat{l}_{ik})$*,* $\hat{D} = diag\{\hat{d}_i\}$*, and* $\hat{U} = (\hat{u}_{ik})$ *be the computed factors of the LDU-factorization of* $\mathcal{D}(A_D, \mathbf{v})$ *by Algorithm 3.1 and let* $L = (l_{ik})$*,*

$D = diag\{d_i\}$ and $U = (u_{ik})$ be the corresponding factors computed exactly. Then we have

$$\begin{aligned}
&\|\hat{L} - L\|_\infty \leq \left(N\nu_{N-1}\epsilon + \mathcal{O}(\epsilon^2)\right)\|L\|_\infty, \\
&\left|\hat{d}_i - d_i\right| \leq \left(\xi_{N-1}\epsilon + \mathcal{O}(\epsilon^2)\right) d_i, \quad \textit{for} \quad 1 \leq i \leq N, \\
&\|\hat{U} - U\|_\infty \leq \left(\nu_{N-1}\epsilon + \mathcal{O}(\epsilon^2)\right)\|U\|_\infty,
\end{aligned}$$

where $\nu_{N-1} \leq 14N^3$ and $\xi_{N-1} \leq 6N^3$.

The above theorem states that the computed L and U are norwise accurate, while the computed D is entrywise accurate. But, perhaps more importantly, the theorem implies via Theorem 3.1 that the $ALDU$ factorization is a rank revealing factorization. Thus, in the event that a differential operator can be disctretized as a diagonally dominant matrix (or a product of diagonally dominant matrices) the $ALDU$ algorithm can be applied directly to A to obtain an inverse-equivalent solution $\hat{\mathbf{x}}$ to the linear system $A\mathbf{x} = \mathbf{b}$.

Unfortunately, the discretization matrix A often does not posses this necessary diagonal dominance structure, making the direct application of $ALDU$ impossible. In such a case, we look to solve $A\mathbf{x} = \mathbf{b}$ by preconditioning with a matrix for which an accurate RRD is known.

3.3 Accurate Preconditioning Method

In this section, we summarize the work of Ye in [26] which develops a preconditioning technique to address accuracy issues arising in the presence of ill-conditioning.

Preconditioning is a standard technique used to speed up convergence of iterative algorithms when solving ill-conditioned linear systems. The basic idea of preconditioning is as follows: Find a preconditioner M such that $M \approx A$ and $M^{-1}A$ is well conditioned. Then applying an iterative method to the new system

$$M^{-1}A\mathbf{x} = M^{-1}\mathbf{b} \tag{3.9}$$

results in accelerated convergence. Because this new system eq. (3.9) is well conditioned, one might argue that solving the system should produce more accurate solutions (thereby alleviating our need for the methods discussed in the previous sections). This, in general, is not the case.

Given the matrix M, suppose we solve for the desired preconditioner M^{-1} using a backward stable algorithm. Then by eq. (3.1) we have that $fl(M^{-1}\mathbf{b}) = M^{-1}\mathbf{b} + \mathbf{f}$ where $\|\mathbf{f}\| \leq \mathcal{O}(\epsilon)\kappa(M)\|M^{-1}\mathbf{b}\|$. Similarly, the computed matrix multiplication $M^{-1}A$ satisfies

$$fl(M^{-1}A) = M^{-1}A + F$$

where $\|F\| \leq \mathcal{O}(\epsilon)\kappa(M)\|M^{-1}A\|$.

Thus, in a best-case scenario, the *exact* solution $\mathbf{y}$ of the computed preconditioned system $fl(M^{-1}A)\mathbf{y} = fl(M^{-1}\mathbf{b})$ can only be bounded as

$$\frac{\|\mathbf{y} - \mathbf{x}\|}{\|\mathbf{x}\|} \leq \mathcal{O}(\epsilon)\kappa(M)\kappa(M^{-1}A). \tag{3.10}$$

At first glace, the bound in eq. (3.10) seems promising – it looks independent of $\kappa(A)$ after all. Let us take a closer look at the right hand side. First, note that

$$\frac{\kappa(A)}{\kappa(M^{-1}A)} \leq \kappa(M) \leq \kappa(M^{-1}A)\kappa(A). \tag{3.11}$$

This means that for $M^{-1}A$ to be well conditioned, M is necessarily ill-conditioned (or has a condition number comparable to that of A), *and* that the bound from eq. (3.10) $\mathcal{O}(\epsilon)\kappa(M)\kappa(M^{-1}A)$ is approximately $\mathcal{O}(\epsilon)\kappa(A)$ [26].

Roundoff errors accumulated in inverting M via a standard backward stable algorithm change both the preconditioned system and the final solution. This prevents the improvement of solution accuracy through general preconditioning, but also prompts the following question: Can more accurately inverting M lead to a more accurate solution of the original system? In fact, it can [26].

In general, the success of preconditioning (i.e. faster convergence *and* more accurate solutions) relies upon the ability to accurately invert the preconditioner, and preconditioning with a matrix whose inverse is solved for using an inverse-equivalent algorithm will produce inverse-equivalent solutions to the original system. The following two theorems from [26] provide an explicit error analysis to substantiate this claim. In particular, let $A = M + K$ where K is small in norm, and M is such that there is an inverse-equivalent algorithm for computing M^{-1}. Then we address the accuracy of solutions of the system

$$B\mathbf{x} = \mathbf{c} \quad \text{where} \quad B \coloneqq I + M^{-1}K \quad \text{and} \quad \mathbf{c} \coloneqq M^{-1}b \tag{3.12}$$

computed via a direct method and an iterative method, respectively.

Theorem 3.3. *([26]) Let $A = M + K$ with A and M being invertible and let $A\mathbf{x} = \mathbf{b}$. Assume that there is an inverse-equivalent algorithm for inverting M so that the computed results of $B \coloneqq I + M^{-1}K$ and $\mathbf{c} \coloneqq M^{-1}\mathbf{b}$ satisfy:*

$$\hat{B} = B + E \quad \textit{with} \quad \|E\| \leq \mathcal{O}(\epsilon)\left(1 + \|M^{-1}\|\|K\|\right), \tag{3.13}$$

and

$$\|\hat{\mathbf{c}} - \mathbf{c}\| \leq \mathcal{O}(\epsilon)\|M^{-1}\|\|\mathbf{b}\|. \tag{3.14}$$

Let $\hat{\mathbf{x}}$ be the computed solution to $\hat{B}\mathbf{x} = \hat{\mathbf{c}}$ by a backward stable algorithm so that $\hat{\mathbf{x}}$ satisfies

$$\left(\hat{B} + F\right)\hat{\mathbf{x}} = \hat{\mathbf{c}} \quad \textit{with} \quad \|F\|/\|\hat{B}\| \leq \mathcal{O}(\epsilon). \tag{3.15}$$

Let $\delta \coloneqq (\|E\| + \|F\|)\,\|B^{-1}\|$ and assume that $\delta < 1$. Then

$$\frac{\|\hat{\mathbf{x}} - \mathbf{x}\|}{\|A^{-1}\|\|\mathbf{b}\|} \leq \mathcal{O}(\epsilon)\frac{\kappa(B)}{1-\delta}\left(4 + \frac{\|K\|\|\mathbf{x}\|}{\|\mathbf{b}\|}\right). \tag{3.16}$$

In particular, if $\|M^{-1}\|\|K\| < 1$, then

$$\frac{\|\hat{\mathbf{x}} - \mathbf{x}\|}{\|A^{-1}\|\|\mathbf{b}\|} \leq \frac{\mathcal{O}(\epsilon)}{(1-\delta)(1 - \|M^{-1}\|\|K\|)^2}. \tag{3.17}$$

For large scale systems it is generally more desirable to solve eq. (3.12) using an iterative method. In this case, we have the following result.

Theorem 3.4. *([26]) Consider solving eq.* (3.12) *by an iterative method where the matrix-vector product* $B\mathbf{v} = \mathbf{v} + M^{-1}K\mathbf{v}$ *is computed by an inverse-equivalent algorithm for inverting* M*. Assume that the iterative method produces an approximate solution* $\mathbf{x}_L$ *with* $\|\mathbf{c} - B\mathbf{x}_L\| \leq \mathcal{O}(\epsilon)\left(1 + \|M^{-1}\|\|K\|\right)\|\mathbf{v}\|$ *and* $\|\mathbf{b} - A\mathbf{x}_L\| \leq \|\mathbf{b}\|$*. Then*

$$\frac{\|\mathbf{x} - \mathbf{x}_L\|}{\|A^{-1}\|\|\mathbf{b}\|} \leq \mathcal{O}(\epsilon)\kappa(B)\left(1 + \frac{\|K\|\|\mathbf{x}_L\|}{\|\mathbf{b}\|}\right) \tag{3.18}$$

$$\leq \mathcal{O}(\epsilon)\kappa(B)\left(1 + 2\|A^{-1}\|\|K\|\right). \tag{3.19}$$

In light of these results, we seek to write the discretization matrix A as the sum $A = M + K$, where K is small in norm and M has an accurate RRD. Then, solving the preconditioned system

$$\left(I + M^{-1}K\right)\mathbf{x} = M^{-1}\mathbf{b} \tag{3.20}$$

using any backward stable algorithm will result in an inverse-equivalent solution $\hat{\mathbf{x}}$ of the system $A\mathbf{x} = \mathbf{b}$. We reiterate that the necessary component here is that for any preconditioner M, M^{-1} is either available exactly, or has an inverse-equivalent algorithm. See [26] for proofs of Theorem 3.3 and Theorem 3.4 and further error analysis of this so-called accurate preconditioning process.

In Chapter 4 we use accurate preconditioning schemes together with standard iterative eigenvalue solvers in order to compute highly accurate smaller eigenvalues of the biharmonic operator eq. (2.1).

Chapter 4 Accurate Computations of a Biharmonic Eigenvalue

In this chapter we present discretizations for accurate computations of the smallest biharmonic eigenvalues across several domains. In particular, we solve a matrix eigenvalue problem $A_N\mathbf{v} = \lambda\mathbf{v}$ (where A_N depends on the domain) by applying inverse iteration together with accurate-preconditioning to compute accurate solutions of the differential eigenvalue problem eq. (2.1). In a demonstration of the success of this approach, we employ two particular inverse-equivalent algorithms for computing the preconditioner M^{-1}: the accurate LDU and a known accurate RRD.

Direct discretization matrices of the biharmonic operator Δ^2 are not typically diagonally dominant. However, under certain conditions, when Δ^2 is decoupled into the composition of two second order Laplacian operators Δ the resulting discretization matrix of Δ^2 is the product of two diagonally dominant matrices.

Consider, for example, the 1-d biharmonic eigenvalue problem $\Delta^2 v = \lambda v$ on $[0, 1]$, subject to the natural boundary conditions $v(0) = v''(0) = v(1) = v''(1) = 0$. The two decoupled Laplacian operators with which this problem is composed have the same natural boundary conditions and hence have the same discretization matrix

$$T_N = \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{bmatrix}. \tag{4.1}$$

This results in a final discretization matrix of Δ^2 which has the product form $A_N = T_N^2$. Clearly T_N is diagonally dominant, and thus inverse iteration together with the $ALDU$ factorization can be directly applied to accurately compute smaller eigenvalues. In the analogous 2-d case with domain $[0, 1]^2$ and natural boundary conditions, we have a final discretization matrix $A_{N\times N} = T_{N\times N}^2 = (I \otimes T_N + T_N \otimes I)^2$, where $\otimes$ is the Kronecker product. As before, this is the product of two diagonally dominant matrices. The eigenvalues of both T_N and $T_{N\times N}$ are known exactly and [28] demonstrates the success in employing the $ALDU$ algorithm to achieve high accuracy computations.

Unfortunately, under Dirichlet boundary conditions, this highly desired product form is not as readily available. It is possible to derive a suitable discretization at the boundary so that the resulting matrix can be written as the product of two diagonally dominant matrices; [28] does this for the 1d biharmonic operator with Dirichlet boundary conditions, but it is not clear that this method can be applied across many domains. Explicitly, [28] does not address the 2d biharmonic operator with Dirichlet boundary conditions, citing that the standard 13 point discretization does not possess the desirable product form. Instead, we focus on a more general approach of using standard discretizations of the biharmonic operator under Dirichlet boundary conditions together with the accurate preconditioning method described in Section 3.3 to achieve a desired accuracy.

We first solve eq. (2.1) on the unit square in both one and two dimensions, and to argue the applicability of this approach to other domains, we consider the unit circle $\Omega = B(0,1)$. We note that the results of Section 4.1 compare with those of [28], but here we have the advantage of not having to derive a suitable discretization that is the product of two diagonally dominant matrices. The results of Section 4.2 and Section 4.3, on the other hand, show an accuracy which cannot, to the authors knowledge, be found in the literature.

In each example inverse iteration is applied to A_N to compute the smallest eigenvalue, where at each iteration the linear system $A_N\mathbf{u} = \mathbf{v}$ is first solved using a standard backward stable algorithm (e.g. Cholesky factorization on A_N), and then again using GMRES (implemented with 50 restarts) on an accurately preconditioned system eq. (3.20). Similar results are obtained using Lanczos/Arnoldi, but we present only those from inverse iteration. All computations [1] were performed on a PC using MATLAB, version R2017a, and for each domain computed eigenvalues from the literature are used as a reference point to compute relative errors.

4.1 Unit Interval: $\Omega = [0,1]$

Discretizing $\Omega = [0,1]$ on the uniform mesh $0 = x_0 < x_1 < \cdots < x_N < x_{N+1} = 1$, where $x_i = ih$ and $h = \frac{1}{N+1}$, then using the center difference ([15], [28]) yields the matrix equation form of (2.1):

$$A_N\mathbf{v} = h^4\lambda\mathbf{v} + \mathbf{e} \tag{4.2}$$

where $\mathbf{v}^T = (v(x_1), v(x_2), \ldots v(x_N))$, $\mathbf{e}^T = (\mathcal{O}(h^{-1})\ \ \mathcal{O}(h^2)\ldots\mathcal{O}(h^2)\ \ \mathcal{O}(h^{-1}))$ and A_N is the 5-diagonal $N \times N$ matrix

$$A_N = \begin{bmatrix} 7 & -4 & 1 & & & & & \\ -4 & 6 & -4 & 1 & & & & \\ 1 & -4 & 6 & -4 & 1 & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & 1 & -4 & 6 & -4 \\ & & & & & 1 & -4 & 7 \end{bmatrix}.$$

A_N has a condition number of order h^{-4}, is not diagonally dominant, and does not appear to have a factorization as a product of diagonally dominant matrices, hence a direct computation of λ_1 will have little accuracy for large N. See Table 4.1. Instead, we write A_N as the sum

$$A_N = T_N^2 + E_N$$

where T_N is given by eq. (4.1) and E_N is the diagonal matrix $E_N = \mathrm{diag}(2, 0, \ldots, 0, 2)$. Since T_N^2 is the product of two diagonally dominant matrices and E_N is small, we can

[1] Except for $\lambda_1^{\mathrm{chol}}$ of Table 4.2 for $N = 2^{12}$; this was computed using a Hi-Mem node (512 GB) on the University of Kentucky's Lipscomb High Performance Computing Cluster, built by Dell Inc.

Table 4.1: The smallest eigenvalue of the 1-dimensional biharmonic eigenvalue problem on $\Omega = [0,1]$ is computed using inverse iteration together with the Cholesky factorization on A_N (λ_1^{chol}) and the accurate LDU factorization of T_N in an accurate preconditioning scheme (λ_1^{aldu}). For comparison, we include the results of using the built-in MATLAB function eigs on A_N (λ_1^{eigs}).

N	h	λ_1^{eigs}	$\frac{\lvert\lambda_1-\lambda_1^{\text{eigs}}\rvert}{\lambda_1}$	λ_1^{chol}	$\frac{\lvert\lambda_1-\lambda_1^{\text{chol}}\rvert}{\lambda_1}$	λ_1^{aldu}	$\frac{\lvert\lambda_1-\lambda_1^{\text{aldu}}\rvert}{\lambda_1}$
2^4	6.3e-02	4.866235035910463e+02	2.7e-02	4.866235035910569e+02	2.7e-02	4.866235035910537e+02	2.8e-02
2^5	3.1e-02	4.967978712848667e+02	7.5e-03	4.967978712851951e+02	7.5e -03	4.967978712850298e+02	7.5e-03
2^6	1.6e-02	4.995884940389652e+02	1.9e-03	4.995884940399146e+02	1.9e-03	4.995884940386783e+02	1.9e-03
2^7	7.8e-03	5.003159428431272e+02	4.9e-04	5.003159428352059e+02	4.9e-04	5.003159428323790e+02	4.9e-04
2^8	3.9e-03	5.005014087109788e+02	1.2e-04	5.005014085433866e+02	1.2e-04	5.005014086419910e+02	1.3e-04
2^9	2.0e-03	5.005482155644552e+02	3.1e-05	5.005482167485592e+02	3.1e-05	5.005482162287630e+02	3.1e-05
2^{10}	9.8e-04	5.005599901607670e+02	7.8e-06	5.005599685751752e+02	7.8e-06	5.005599726273552e+02	7.9e-06
2^{11}	4.9e-04	5.005629837852146e+02	1.8e-06	5.005630775610463e+02	1.6e-06	5.005629184981394e+02	1.9e-06
2^{12}	2.4e-04	5.005628603461304e+02	2.1e-06	5.005661883657695e+02	4.5e-06	5.005636558107462e+02	4.9e-07
2^{13}	1.2e-04	5.005752767371457e+02	2.3e-05	5.005582938119225e+02	1.1e-05	5.005638402501275e02	1.2e-07
2^{14}	6.1e-05	5.004440381056732e+02	2.4e-04	5.007178259919213e+02	3.0e-04	5.005638863671624e+02	3.1e-08
2^{15}	3.1 e-05	5.005701431041793e+02	1.2e-05	5.025944960403944e+02	4.0e-03	5.005638978912588 e+02	7.7e-09
2^{16}	1.5e-05	3.745357909797928e+02	2.5e-01	5.158748033027155e+02	3.0e-02	5.005639008277676 e+02	1.8e-09
2^{17}	7.6e -06	-1.076609301288449e+03	3.1e+00	2.655558644301438e+02	4.7e-01	5.005639014178888e+02	6.4e-10

accurately solve the matrix eigenvalue problem $(T_N^2 + E_N)\mathbf{v} = h^4\lambda\mathbf{v}$ by forming the preconditioned system

$$\left(I + T_N^{-2}E_N\right)\mathbf{u} = T_N^{-2}\mathbf{v}, \tag{4.3}$$

to be solved at each step of inverse iteration. To compare various results, $A_N\mathbf{u} = \mathbf{v}$ is solved for in two different ways:

1. Using the Cholesky factorization of A_N, and
2. Using GMRES on the preconditioned system in eq. (4.3) where T_N^{-2} is computed using the accurate LDU factorization of T_N.

The eigenvalues of the clamped plate problem are not known exactly, but in [28], λ_1 is computed in high precision as the root of an algebraic equation, transcendental in the eigenvalue parameter. To fifty digits we have

$$\lambda_1 = 500.56390174043259597023906145469523385520808092739. \tag{4.4}$$

Table 4.1 shows the resulting relative errors for increasingly large values of N. As h decreases the the discretization matrix becomes increasingly ill-conditioned. The errors associated with MATLAB's built-in solver `eigs` (λ_1^{eigs}) and with inverse iteration paired with the standard Cholesky (λ_1^{chol}) steadily worsen. In fact, when the mesh-size $h \approx$ 1.2e-04, the error actually begins to increase until we lose all accuracy, and at $N = 2^{17}$, `eigs` cannot even produce the correct sign. The $ALDU$ accurate-preconditioning scheme (λ_1^{aldu}) on the other hand continues to produce accurate computations as the error decreases quadratically to reach full machine accuracy.

4.2 Unit Square: $\Omega = [0,1]^2$

We now consider eq. (2.1) in two dimensions with $\Omega = [0,1]^2$ and Dirichlet boundary conditions $v(x,y) = \partial_n v(x,y) = 0$ when $x, y \in \{0,1\}$. As discussed in [1], when

the standard thirteen point difference approximation is employed (with x and y discretized on the same mesh as in Section 4.1), the matrix equation

$$A_{N\times N}\mathbf{v} = h^4\lambda\mathbf{v} + \mathbf{e} \tag{4.5}$$

results, where $\mathbf{e}$ is an N^2-length error vector analogous to that in Section 4.1, $\mathbf{v} = [\mathbf{v}_j]_{j=1}^N \in \mathbb{R}^{N^2}$ with $\mathbf{v}_j = [v(x_i, y_j)]_{i=1}^N \in \mathbb{R}^N$ and $A_{N\times N} = [V_i,\ W_i,\ X_i,\ Y_i,\ Z_i]$ is an $N^2 \times N^2$ block five-diagonal matrix:

$$A_{N\times N} = \begin{bmatrix} X_1 & Y_1 & Z_1 & & & & & \\ W_2 & X_2 & Y_2 & Z_2 & & & & \\ V_3 & W_3 & X_3 & Y_3 & Z_3 & & & \\ & & & \ddots & \ddots & & & \\ & & & & \ddots & \ddots & & \\ & & & & V_{N-1} & W_{N-1} & X_{N-1} & Y_{N-1} \\ & & & & & V_N & W_N & X_N \end{bmatrix} \tag{4.6}$$

with

$$\begin{aligned} X_1 = X_N &= X + I, \\ X_i &= X \text{ for } i = 2, \cdots, N-1, \end{aligned}$$

$$W_i = Y_{i-1} = W \text{ for } i = 2, \cdots, N, \tag{4.7}$$

and

$$V_i = Z_{i-2} = I \text{ for } i = 3, \cdots, N.$$

Here, I is the $N \times N$ identity matrix, and W and X are $N \times N$ three and five-diagonal matrices, respectively, defined by

$$\begin{aligned} W &:= [2, \quad -8, \quad 2], \\ X &:= [1, \quad -8, \quad \tilde{c}_k, \quad -8, \quad 1], \end{aligned} \qquad \tilde{c}_k = \begin{cases} 21 \text{ if } k = 1, N \\ 20 \text{ if } k = 2, \cdots, N-1. \end{cases} \tag{4.8}$$

To employ the desired preconditioning methods we rewrite eq. (4.6) as the sum

$$A_{N\times N} = T_{N\times N}^2 + E_{N\times N}$$

where $T_{N\times N} = I \otimes T_N + T_N \otimes I$ is the diagonally dominant block tri-diagonal matrix

$$T_{N\times N} = \begin{bmatrix} T_N + 2I & -I & & & \\ -I & T_N + 2I & -I & & \\ & & \ddots & & -I \\ & & & -I & T_N + 2I \end{bmatrix},$$

and $E_{N\times N}$ the block diagonal matrix

$$E_{N\times N} = \begin{bmatrix} E_N + 2I & & & \\ & E_N & & \\ & & \ddots & \\ & & & E_N + 2I \end{bmatrix}.$$

Similar to the one dimensional case, we drop the error term $\mathbf{e}$, and solve eq. (4.5) using two approaches to inverse iteration.

1. Using the Cholesky factorization to solve for $A^{-1}_{N\times N}$, and
2. Using GMRES on the preconditioned system

$$\left(I + T^{-2}_{N\times N} E_{N\times N}\right)\mathbf{u} = T^{-2}_{N\times N}\mathbf{v}, \tag{4.9}$$

at each iteration of the inverse power method. The preconditioner $T^{-2}_{N\times N}$ is solved for using the eigenvalue decomposition of $T_{N\times N}$ together with the Fast-Fourier Transform.

We note that $T_{N\times N}$ is diagonally dominant and thus has an accurate LDU decomposition which can be used for accurate preconditioning in eq. (4.9); however, the $\mathcal{O}((N^2)^3)$ cost of $ALDU$ on this $N^2 \times N^2$ matrix increases unreasonably quick with N. To test for large N, we consider instead an accurate RRD of $T_{N\times N}$ based on the known eigenvalue decomposition $T_N = Z\Lambda Z^T$ [12]. Since $\|Z\| = 1$ and $\Lambda(j,j) = 2(1 - \cos\frac{\pi j}{N+1}) > 0$ is diagonal and nonsingular, this is an accurate RRD. Moreover, the 2-dimensional eigenvalue decomposition

$$T_{N\times N} = (I \otimes T_N) + (T_N \otimes I) = (Z \otimes Z)\cdot(I \otimes \Lambda + \Lambda \otimes I)\cdot(Z \otimes Z)^T \tag{4.10}$$

is an exact rank-revealing decomposition with $Z \otimes Z$ being orthogonal, and hence it is an accurate RRD.

Using the eigenvalue decomposition to solve for $T^{-1}_{N\times N}$ is particularly desirable for the large systems arising in the 2-dimensional case because it allows us to use a Fast-Fourier Transform (FFT) operation to cheaply solve for $T^{-1}_{N\times N}$. First, note that the (j,k) entry of the $N \times N$ matrix Z is

$$z_{jk} = \sqrt{\frac{2}{N+1}}\sin\left(\frac{\pi jk}{N+1}\right), \tag{4.11}$$

and the (j,k) entry of the $(2N+2)\times(2N+2)$ Discrete-Fourier Transform (DFT) matrix is

$$\exp\left(\frac{-2\pi ijk}{2N+2}\right) = \exp\left(\frac{-\pi ijk}{N+1}\right) = \cos\frac{\pi jk}{N+1} - i\cdot\sin\frac{\pi jk}{N+1}. \tag{4.12}$$

Thus, the $N \times N$ eigenvector matrix Z consists of $-\sqrt{\frac{2}{N+1}}$ times the imaginary part of the second through $(N+1)$st rows and columns of the DFT matrix [12].

Now, we can rewrite $T_{N\times N}\mathbf{u} = \mathbf{v}$ such that the computed $\hat{\mathbf{u}}$ requires only left and right multiplications by Z. The system $T_{N\times N}\mathbf{u} = \mathbf{v}$ is equivalent to the matrix equation

$$T_N U + U T_N = V, \tag{4.13}$$

where $\mathbf{u}, U, \mathbf{v}$, and V are such that

$$\text{vec}(U) = \mathbf{u} \text{ and } \text{vec}(V) = \mathbf{v}.$$

Substituting the eigenvalue decomposition $T_N = Z\Lambda Z$ into (4.13) and multiplying on the left and right by $Z = Z^T$, we obtain the following algorithm [12] for computing the solution U:

Algorithm 4.1. *Input Z and V.*

1. $\tilde{V} = ZVZ$
2. *For all j and k,* $\tilde{\mathbf{u}}_{jk} = \dfrac{\tilde{\mathbf{v}}_{jk}}{\lambda_j + \lambda_k}$
3. $U = Z\tilde{U}Z$.

That is, obtaining solution U requires only the ability to multiply on the left (and right) by Z. And as can be seen from eq. (4.11), such a multiplication is an FFT-like operation on the columns(rows) of V and $\tilde{U}$. We note that Algorithm 4.1 in an inverse-equivalent algorithm since it is mathematically equivalent to inverse-equivalent algorithm

$$\text{vec}(U) = T_{N\times N}^{-1} \cdot \text{vec}(V)$$

obtained using the accurate RRD of eq. (4.10) as discussed above, thus allowing us to cheaply obtain an inverse-equivalent solution $\hat{\mathbf{u}}$.

The eigenvalues of the biharmonic problem on the unit square with Dirichlet boundary conditions are not known exactly, but the smallest eigenvalue is computed in quadruple precision in [2] as

$$\lambda_1 = 1294.93397959171280817030264797 44. \tag{4.14}$$

A word about this chosen reference eigenvalue λ_1: We have no guarantee of accuracy of this computation; nor, in general, do we know the degree to which quadruple precision calculations can be trusted in the presence of ill-conditioning. Ideally, in the absence of knowing the exact eigenvalue, we would like for the desired reference eigenvalue to be computed symbolically from an algebraic equation as eq. (4.4) was calculated for [28] by M. Embree. We leave this to future work, and settle for the time being with eq. (4.14) as a reference point. Now, this is not to say that, in the event that eq. (4.14) *is* accurate to all its presented digits, the current methods applied here are irrelevant or unnecessary. The computation of eq. (4.14) in [2] required the combination of a matrix obtained from up to 5000 basis functions and the use of quadruple precision. We've presented here an alternative, perhaps slightly more accessible, method for achieving highly accurate results in double precision (for

Table 4.2: Computations of the smallest eigenvalue of the 2-dimensional biharmonic eigenvalue problem on $\Omega = [0,1]^2$ using MATLAB's built in eigs on $A_{N\times N}$, and then inverse iteration together with the Cholesky factorization of $A_{N\times N}$ (λ_1^{chol}) and the eigenvalue decomposition of $T_{N\times N}$ in an accurate preconditioning scheme (λ_1^{FFT}).

N	h	λ_1^{eigs}	$\frac{\lvert\lambda_1 - \lambda_1^{\text{eigs}}\rvert}{\lambda_1}$	λ_1^{chol}	$\frac{\lvert\lambda_1 - \lambda_1^{\text{chol}}\rvert}{\lambda_1}$	λ_1^{FFT}	$\frac{\lvert\lambda_1 - \lambda_1^{\text{FFT}}\rvert}{\lambda_1}$
2^4	6.3e-02	1.253504496973189e+03	3.2e-02	1.253504496973134e+03	3.2e-02	1.253504496973189e+03	3.2e-02
2^5	3.1e-2	1.283580514513306e+03	8.7e-03	1.283580514514243e+03	8.7e-03	1.283580514515036e+03	8.7e-03
2^6	1.6e-2	1.291980776720620e+03	2.3e-03	1.291980776738552e+03	2.3e-03	1.291980776741693e+03	2.3e-03
2^7	7.8e-3	1.294182378905437e+03	5.8e-04	1.294182379179328e+03	5.8e-04	1.294182379123644e+03	5.8e-04
2^8	3.9e-3	1.294744492745101e+03	1.4e-04	1.294744497477243e+03	1.4e-04	1.294744497650665e+03	1.4e-04
2^9	2.0e-3	1.294886343614766e+03	3.7e-05	1.294886416807854e+03	3.7e-05	1.294886407148133e+03	3.7e-05
2^{10}	9.8e-4	1.294920852462902e+03	1.0e-05	1.294921637849319e+03	9.6e-06	1.294922050338627e+03	9.2e-06
2^{11}	4.9e-4	1.294911817799253e+03	1.7e-05	1.294923489483935e+03	8.1e-06	1.294931003151992e+03	2.3e-06
2^{12}	2.4e-4	1.294625457735924e+03	2.4e-04	1.294933138689635e+03	9.5e-05	1.294933430638977e+03	4.2e-07

which direct hardware support is far more common). Regardless, what we do have is the clear convergence of the accurate preconditioning scheme to a value near that of eq. (4.14), and a lack of convergence for the backward stable Cholesky approach. For comparison, we've included the results of MATLAB'S `eigs`, where we also see the computed values beginning to diverge for large N.

In particular, Table 4.2 displays the results of our 2-dimensional experiment using inverse iteration. λ_1^{FFT} denotes the eigenvalue computed using Algorithm 4.1 and the FFT to solve for $T_{N\times N}^{-1}$ in the accurate preconditioning scheme at each iteration, while λ_1^{chol} refers the use of the Cholesky factorization to solve for $A_{N\times N}^{-1}$ at each iteration, and λ_1^{eigs} denotes the eigenvalue computed using MATLAB's built in `eigs` function directly on $A_{N\times N}$.

Moreover, if we take eq. (4.14) to be true, then the relative errors of Table 4.2 indicate that the FFT inverse-equivalent scheme performs as expected, as it continually produces accurate solutions as N increases. However, The relative error of the Cholesky preconditioning method on the other hand begins to stagnate, then begins to increase. Indeed, $N = 2^4$ through $N = 2^9$ all three methods preform comparably, after which the Cholesky and `eigs` based methods start to fall behind. In particular, for $N = 2^{11}$ and $N = 2^{12}$ we see that the error for Cholesky preconditioning is slightly larger than that for λ_1^{FFT}, and then for $N = 2^{12}$ the error actually increases. We see similar results for λ_1^{eigs}, but in this case, the error begins to stagnate earlier, around $N = 2^{10}$, and it does not reach the same accuracy that λ_1^{chol} does.

We further note that our computations live within the upper-bound 1294.933988 calculated in [25], and look to be monotonically increasing to λ_1. We also note that similarly accurate solutions are obtained when the preconditioner $T_{N\times N}^{-1}$ is solved for using the $ALDU$ factorization of $T_{N\times N}$ for N up to 2^9, but we report only the computationally efficient FFT results.

4.3 Unit Circle: $\Omega = B(0,1)$

While the problems examined thus far have exploited the structure of the very specific matrix T_N, the success of our method hinges only on the ability to write the discretization as the sum of a matrix possessing an accurate RRD and a matrix small in norm; thus, the method can be extended to many domains. As an example, we present accurate solutions to the biharmonic eigenvalue problem with Dirichlet boundary conditions on the unit circle.

For $\Omega = B(0,1) = \{x, y : x^2 + y^2 \leq 1\}$ eq. (2.1) is converted to polar coordinates [10] by writing v as a function of the radius $r \in [0,1]$ and angle $\theta \in (0, 2\pi]$. In polar coordinated Δ is given as $\Delta = \frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2}$. This two dimensional problem can be further simplified by assuming the function v is radial (i.e. $v(r,\theta) = v(r)$), and thus reducing it to a one dimensional problem in r:

$$\begin{cases} \left(\partial_r^2 + \dfrac{1}{r}\partial_r\right)^2 v(r) = \lambda v(r) \text{ for } r \in (0,1) \\ v(1) = v'(1) = 0. \end{cases} \tag{4.15}$$

Everitt, et. al discuss in [16] the Dirichlet boundary value problem of the biharmonic partial differential equation on a disc of finite radius in the plane. In this paper, a quasi-separation of variables method is presented which leads to solutions of the partial differential equation which are products of solutions of two ordinary linear differential equation: a fourth order radial equation, and a second order angular differential equation. In particular, [16, Theorem 9.1] addresses the eigenvalues of the biharmonic eigenvalue problem with Dirichlet boundary conditions in relation to the radial and angular factors of the solution $v(r,\theta)$. This, together with [16, Lemma 4.1] suggests the validity of using eq. (4.15) to compute λ_1 of eq. (2.1) with $\Omega = B(0,1)$.

Note that this equation has no left-end boundary condition, so to derive a suitable discretization matrix equation, we first decouple the product in eq. (4.15). Let

$$w(r) = -\left(\partial_r^2 + \frac{1}{r}\partial_r\right) v(r), \tag{4.16}$$

and

$$u(r) = -\left(\partial_r^2 + \frac{1}{r}\partial_r\right) w(r) = \lambda v(r). \tag{4.17}$$

Then, for $0 = r_0 < r_1 < \cdots < r_N < r_{N+1} = 1$, where $r_i = \left(i - \frac{1}{2}\right) h$ and $h = \frac{2}{2N+1}$ for $i = 1, \ldots, N$, applying the center difference yields

$$u_i = \frac{1}{h^2}\left(\frac{2-2i}{2i-1} w_{i-1} + 2w_i + \frac{-2i}{2i-1} w_{i+1}\right) + \mathcal{O}(h^2),$$

where we employ the notation $u_i = u(r_i)$. This half mesh shift takes care of a left boundary condition since when $i = 1$ the w_0 coefficient is zero, and hence eq. (4.17)

in matrix form is

$$\frac{1}{h^2}\begin{bmatrix} 2 & q_2 & & & & \\ l_1 & 2 & q_3 & & & \\ & \ddots & & & & \\ & & & \ddots & q_N & \\ & & & l_{N-1} & 2 & q_{N+1} \end{bmatrix}\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \\ w_{N+1} \end{bmatrix} + \mathcal{O}(\mathbf{h^2}) = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix}$$

where $l_i = \frac{2-2(i-1)}{2(i+1)-1}$ and $q_i = \frac{-2(i-1)}{2(i-1)-1}$. For notational purposes we write this as

$$\frac{1}{h^2}[B \mid \mathbf{y}]\hat{\mathbf{w}} + \mathbf{f} = \mathbf{u} \tag{4.18}$$

where

$$B = \begin{bmatrix} 2 & q_2 & & & \\ l_1 & 2 & q_3 & & \\ & \ddots & & & \\ & & & \ddots & q_N \\ & & & l_{N-1} & 2 \end{bmatrix} \text{ and } \mathbf{y} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ q_{N+1} \end{bmatrix}.$$

Similarly, for eq. (4.16) we have

$$w_i = \frac{1}{h^2}\left(\frac{2-2i}{2i-1}v_{i-1} + 2v_i + \frac{-2i}{2i-1}v_{i+1}\right) + \mathcal{O}(h^2),$$

where the v_0 coefficient vanishes when $i = 1$, and when $i = N$ we have the originally imposed boundary condition $v_{N+1} = v(1) = 0$. Finally, we must derive an equation for w_{N+1}.

Since $v'(r_{N+1}) = 0$, by eq. (4.16) we have

$$w_{N+1} = -v''(r_{N+1}) = -v''(1) = \frac{-v_N + 2v_{N+1} - v_{N+2}}{h^2} + \mathcal{O}(h^2).$$

Then, writing $v_{N+2} = v(1+h) = \frac{1}{2}h^2v''(1) + \mathcal{O}(h^2)$ gives $w_{N+1} = -\frac{1}{h^2}2v_N + \mathcal{O}(h^2)$, and therefore the matrix form of eq. (4.16) is

$$\frac{1}{h^2}\begin{bmatrix} B \\ \mathbf{z}^T \end{bmatrix}\mathbf{v} + \mathcal{O}(\mathbf{h^2}) = \hat{\mathbf{w}} \text{ where } \mathbf{z}^T = (0 \cdots 0 \ -2). \tag{4.19}$$

Plugging this into eq. (4.18) and dropping the (order h^2) error term gives the desired discretization of the biharmonic eigenvalue problem on the unit circle:

$$\left(B^2 + E_N\right)\mathbf{v} = h^4\lambda\mathbf{v} \tag{4.20}$$

where

$$E_N = \mathbf{y}\mathbf{z}^T = \operatorname{diag}\left(0, \ldots, 0, \ \frac{4N}{2N-1}\right);$$

furthermore, the diagonally dominant part of B, given by the N-vector

$$(0,\ldots,0,2-|l_{N-1}|) = \left(0,\ldots,0,\frac{2N}{2N-1}\right),$$

is nonnegative. Thus, B is diagonally dominant and we can use the accurate LDU algorithm to compute B^{-1}, then use GMRES to solve the preconditioned system

$$\left(I + B^{-2}E_N\right)\mathbf{u} = B^{-2}\mathbf{v}$$

at each step of inverse iteration to accurately compute the desired eigenvalue.

In [18], Luo develops a numerical method and error bounds for biharmonic eigenvalue equations in the spherical domain based on the spectral theory of compact operators. By adopting orthogonal spherical basis functions to establish a discrete model with sparse matrices, they compute the smallest eigenvalue of eq. (2.1) to ten-digit accuracy as $\lambda = 104.3631056$. This reference provides additional confirmation that our method (specifically the results in Table 4.3) are converging to the correct value. However, we will compute our reference eigenvalue λ_1 using the methods described in [9, Ch.5 §6]. We summarize some of these details here, but refer the reader to [7], [9], [16] and the references therein for more details.

Using polar coordinates and writing $\lambda = k^4$, we rewrite $\Delta^2 v = \lambda v$ as

$$(\Delta\Delta - k^4)v = 0 \Longleftrightarrow (\Delta - k^2)(\Delta + k^2)v = 0.$$

If we assume that v can be expanded in a Fourier series as

$$v = \sum_{n=-\infty}^{\infty} y_n(r)e^{in\theta},$$

then each term $y_n(r)$ must satisfy the following differential equation:

$$\left(\frac{d^2}{dr^2} + \frac{1}{r}\frac{d}{dr} - \frac{n^2}{r} - k^2\right)\left(\frac{d^2}{dr^2} + \frac{1}{r}\frac{d}{dr} - \frac{n^2}{r} + k^2\right)y_n(r) = 0.$$

The Bessel functions $J_n(kr)$ and $J_n(ikr)$ are two linearly independent solutions of this equation which are regular for $r = 0$. Thus, solutions v take the form

$$v(r,\theta) = J_n(kr)(a_1\cos(n\theta) + b_1\sin(n\theta)) + J_n(ikr)(a_2\cos(n\theta) + b_2\sin(n\theta)).$$

Now, imposing the boundary conditions $v(1,\theta) = 0 = v_r(1,\theta)$ we obtain the four equations

$$\begin{aligned} J_n(k)a_1 + J_n(ik)a_2 &= 0 & \qquad J_n(k)b_1 + J_n(ik)b_2 &= 0 \\ J_n'(k)a_1 + iJ_n'(ik)a_2 &= 0 & \qquad J_n'(k)b_1 + iJ_n'b_2 &= 0, \end{aligned}$$

which immediately imply that k must satisfy

$$\frac{J_n'(k)}{J_n(k)} = \frac{iJ_n'(k)}{J_n(ik)}. \tag{4.21}$$

Table 4.3: Computations of the smallest eigenvalue of the biharmonic eigenvalue problem on $\Omega = B(0,1)$ using inverse iteration together with the LU factorization of A_N (λ_1^{lu}) and the $ALDU$ factorization of B in an accurate preconditioning scheme (λ_1^{aldu}). For comparison, we include the built-in MATLAB eigenvalue function eigs on A_N (λ_1^{eigs}).

N	h	λ_1^{eigs}	$\frac{\lvert\lambda_1-\lambda_1^{\text{eigs}}\rvert}{\lambda_1}$	λ_1^{lu}	$\frac{\lvert\lambda_1-\lambda_1^{\text{lu}}\rvert}{\lambda_1}$	λ_1^{aldu}	$\frac{\lvert\lambda_1-\lambda_1^{\text{aldu}}\rvert}{\lambda_1}$
2^4	6.1e-02	1.034452748267384e+02	8.8e-03	1.034452748267304e+02	8.7e-03	1.034452748267304e+02	8.8e-03
2^5	3.1e-02	1.041254297202660e+02	2.3e-03	1.041254297203683e+02	2.3e-03	1.041254297202937e+02	2.3e-03
2^6	1.6e-02	1.043026887084155e+02	5.8e-04	1.043026887094575e+02	5.8e-04	1.043026887078057e+02	5.8e-04
2^7	7.8e-03	1.043478788628527e+02	1.5e-04	1.043478789031381e+02	1.5e-04	1.043478788785113e+02	1.5e-04
2^8	3.9e-03	1.043592837598627e+02	3.7e-05	1.043592840329114e+02	3.7e-05	1.043592837359561e+02	3.7e-05
2^9	2.0e-03	1.043621478534335e+02	9.2e-06	1.043621514032815e+02	9.1e-06	1.043621482183564e+02	9.2e-06
2^{10}	9.7e-04	1.043628345412665e+02	2.6e-06	1.043628944316454e+02	2.0e-06	1.043628659883233e+02	2.3e-06
2^{11}	4.8e-4	1.043626247152904e+02	4.6e-06	1.043633926591254e+02	2.7e-06	1.043630456363610e+02	5.7e-07
2^{12}	2.5e-04	1.043477740649407e+02	1.5e-04	1.043624032249038e+02	6.7e-06	1.043630905734745e+02	1.4e-07
2^{13}	1.2e-04	1.041958255025062e+02	1.6e-03	1.043665105526316e+02	3.2e-05	1.043631018007365e+02	3.6e-08
2^{14}	6.1e-05	1.018930268791083e+02	2.4e-02	1.043378363402431e+02	2.4e-04	1.043631046095882e+02	9.5e-09
2^{15}	3.1e-05	6.417530335512389e+01	3.8e-01	1.090929215448091e+02	4.5e-02	1.043631053197114e+02	2.7e-09
2^{16}	1.5e-05	-5.195041504138685e+02	6.0e+00	1.659797814880537e+02	5.9e-01	1.043631055128566e+02	8.3e-10
2^{17}	7.6e-06	4.274851634003073e+03	4.0e+01	1.834063679583271e+03	1.6e+01	1.043631055054487e+02	9.0e-10

Using properties of Bessel functions

$$\frac{d}{dx}J_n(x) = \frac{1}{2}\left(J_{n-1}(x) - J_{n+1}(x)\right) \quad \text{and} \quad J_n(ix) = i^n I_n(x),$$

where $I_n(x)$ is the modified Bessel function, we see that eq. (4.21) becomes (the real equation)

$$\frac{J_{n-1}(k) - J_{n+1}(k)}{J_n(k)} = \frac{I_{n-1}(k) + I_{n+1}(k)}{I_n(k)}.$$

Since we are interested in the smallest eigenvalue, we take $n = 0$ and solve

$$\frac{-J_1(k)}{J_0(k)} = \frac{I_1(k)}{I_0(k)}$$

for the smallest positive root k using Maple's `fsolve` to 100 digits. Then $\lambda_1 = k^4$ is

$$\lambda_1 = 104.36310555884430692172261967334785049263237543989337598568 40..$$
$$..10469863692197011426120976624952638 4382 \tag{4.22}$$

Table 4.3 uses eq. (4.22) as the reference eigenvalue for computing the relative errors in computing the smallest eigenvalue of eq. (4.15) to demonstrate that by decreasing h, our method continues to produce accurate computations. We present results from using MATLABs built in `eigs` (λ_1^{eigs}) on the discretization matrix $A_N = B^2 + E_N$, inverse iteration using the LU factorization of A_N (λ_1^{lu}), and finally, inverse iteration together with preconditioning using the $ALDU$ factorization of B (λ_1^{aldu}). We can see that both `eigs(AN)`, and the standard LU factorization begin to lose accuracy around $N = 2^{11}$, while the implementation of the $ALDU$ scheme produces increasingly accurate solutions.

4.4 Conclusion

Implementing methods from [26] and [28] we have shown that we can compute the smallest eigenvalue of of the clamped plate problem in double precision to the order of machine precision across many domains using standard finite difference discretizations. The method exploits diagonal dominance in discretization matrices, but requires only that the discretization matrix be written as the sum of a matrix with an accurate rank-revealing decomposition and a matrix small in norm. Comparing results with the best known computations from the literature, the method is shown to be very effective. Future work includes verifying these computations, or determining exact eigenvalues on the unit square and unit circle symbolically. This would further confirm the suggestions of the numerical evidence presented here: that inverse equivalent algorithms can compute solutions to linear algebra problems to machine precision in the presence of ill-conditioning, while backward stable algorithms lose accuracy.

In Part II of this thesis we study a multigrid approach to solving differential equations, and compare the accuracy of multigrid-computed results with those computed by the methods here in Part I.

Part II

Multigrid Algorithms

Chapter 5 Motivation and Model Problem

5.1 Introduction

Historically developed for solving boundary value problems

$$\mathcal{L}v = f \text{ on } \Omega \text{ with } v = g \text{ on } \partial\Omega, \tag{5.1}$$

multilevel methods have evolved rapidly over the past thirty years to produce powerful new techniques for solving partial differential equations with a remarkable range of applicability. It has been verbally suggested that such methods are not subject to the accuracy issues caused by ill-conditioning; however, the author has not found references to substantiate such claims. Part II of this thesis is an investigation to determine if multigrid algorithms could indeed compute solutions comparably to the inverse-equivalent schemes described in Chapter 3 and implemented in Chapter 4.

Consider discretizing eq. (5.1) as

$$A_h \mathbf{v}_h = \mathbf{f}_h, \quad \text{where } A \in \mathbb{R}^{N\times N} \quad \text{with} \quad N = 2^L \text{ and } h = \frac{1}{N}. \tag{5.2}$$

We will conduct an error analysis of multigrid algorithms as applied to eq. (5.2). The broad goal of this work is to understand the role of the condition number of A_h on the relative error of a solution computed via a convergent l-grid V-cycle, for some $l = 1, \ldots, L$, or, to explicitly determine if a multigrid algorithm is an inverse-equivalent algorithm. A lofty goal, the work in this part (particularly Chapter 7 and Chapter 8) provides a starting point and opens the door for further discussion and analysis of multigrid algorithms in the presence of ill-conditioning and in the framework of inverse-equivalency.

In the remaining sections of this chapter we discuss existing error analysis the author could find in the literature, then introduce our Model Problem. This Model Problem will serve as our motivation for a floating point error analysis, as well as an example of how our generalized results can be used to derive concrete bounds. In Chapter 6 we introduce the basic multigrid principle, state specific multigrid algorithms, and develop Fourier analysis – a standard tool used to analyze multigrid algorithms; in this chapter we also state our Model Algorithm. Chapter 7 uses these tools to derive floating point error bounds on a 2-grid operator, a 3-grid operator, and then generalizes the results to an l-grid operator. Finally, in Chapter 8 we use the theory from Chapter 7 to explicitly derive error bounds on solutions to the Model Problem as computed by the Model Algorithm.

5.2 Motivation

Rigorous and local Fourier analysis are well known techniques for determining both convergence factors and bounds on norms of multigrid operators, and have been applied to many specific multigrid algorithms in the literature ([21], [22]). Moreover,

it is well know that, for many problems, full multigrid (FMG) can yield a solution whose error is comparable to the discretization error. In fact, it is often suggested that it is not worthwhile to solve the discrete problem more accurately by investing more computational work, but one should instead redirect that work to reducing the discretization error itself, for example, by appealing to yet a finer target grid. In [19], Rodrigo et al. develop a tool which yields insight into various components of the FMG algorithm and their effects on the final relative accuracy. They note a general lack of attention given to FMG in terms of error analysis and develop the so-called *FMG accuracy measure* as a suitable indicator for the performance of FMG.

However, this discretization-level-accuracy error analysis yields little insight into our question. While attention is paid in comparing a discrete solution to the continuous solution, to the author's knowledge, the literature does not address the comparison of a computed discrete solution to the exact discrete solution. That is, the general effects of roundoff errors in multilevel algorithms has not been address, nor has the specific ability of multigrid algorithms to compute solutions to full machine precision in the presence of ill-conditioning.

What's more, much of the analysis done in the field on multigrid algorithms is of the flavor "*if* the algorithm works, it works *this* well." That is, the analysis exists within the environment of a particular algorithm for a specific problem. In fact, the role of local Fourier analysis is often to aid in the design of a good algorithm for a concrete problem. Even more generally, much of the work of iterative methods involves exploiting the underlying mathematical or physical problem in order to design better iterative methods. We thus focus our analysis around the following Model Problem.

5.3 Model Problem

A classical model for a discrete elliptic boundary value problem, we introduce the finite difference approximation to Poisson's equation on the unit interval as our Model Problem. Arising in many applications, much attention has been paid to Poisson's equation, and many numerical solvers have been applied to this problem for comparison ([12], [19], [22], [26], [28], and the references therein.). Moreover, the coefficient matrix T_N of the discretization of Poisson's equation is large, sparse and has condition number of order $\mathcal{O}(h^{-2})$. This makes it the ideal Model Problem for understanding the roll of ill-conditioning in multigrid algorithms.

Model Problem. Let $\mathcal{L}$ be the one-dimensional Poisson operator, such that eq. (5.1) becomes the following one-dimensional Poisson's equation with Dirichlet boundary conditions:

$$\begin{cases} -\Delta v = f \text{ in } \Omega \\ v = \partial_n v = 0 \text{ on } \partial\Omega, \end{cases} \tag{5.3}$$

with $\Omega = [0, 1]$. We discretize Ω into grid $G_h = \{x_i = ih : i = 0, \cdots, N\}$ using an equidistant mesh $h = \frac{1}{N}$, where $N = 2^L$ for some $L \in \mathbb{N}$. Then using the standard

center difference approximation yields the linear system

$$\frac{1}{h^2}T_N\mathbf{v}_h = \mathbf{f}_h \text{ on grid } G_h, \tag{5.4}$$

or

$$A_h\mathbf{v}_h = \mathbf{f}_h \text{ on grid } G_h,$$

where $A_h = \frac{1}{h^2}T_N$, T_N is as in eq. (4.1), $\mathbf{v}_h(i) = v(x_i)$, and $\mathbf{f}_h(i) = f(x_i)$.

Our general goal is to accurately and efficiently solve eq. (5.4) using multigrid techniques. As further motivation for studying roundoff errors, we present some numerical results of various solvers on the Model Problem. We note that the Model Algorithm introduced in the next chapter details how these results are obtained. Moreover, while Chapter 7 outlines specific assumptions which allow for some general results and applicability, this Model Algorithm serves as a means for computing explicit error bounds in an effort to understand the results of Example 5.1.

Example 5.1. Consider eq. (5.3) with $f(x) = 3\sin(2\pi x)$, discretized as $\mathbf{f}_h(i) = f(x_i)$. This problem has the exact solution $v(x) = \frac{3}{4\pi^2}\sin(2\pi x)$, and hence $\mathbf{v}_h(i) = v(x_i)$. We solve this linear system using a well-behaved full multigrid algorithm that converges efficiently (Chapter 6 **Model Algorithm**, [6], [22]).

Table 5.1 shows numerical results for increasingly ill-conditioned discretization matrices $A_h = \frac{1}{h^2}T_N \in \mathbb{R}^{2^L\times 2^L}$, the table shows the relative error $\rho_x = \|\mathbf{v}_h - \hat{\mathbf{v}}_x\|_2/\|\mathbf{v}_h\|_2$, where the computed solution $\hat{\mathbf{v}}_x$ is obtained using three different algorithms:

1. $\hat{\mathbf{v}}_x = \hat{\mathbf{v}}_{\text{FMG}}$: the full multigrid algorithm, iterated until the ratio of the errors in consecutive iterations is 1, with tolerance .001;

2. $\hat{\mathbf{v}}_x = \hat{\mathbf{v}}_{\text{back}}$: MATLAB's built-in backslash for solving linear systems;

3. $\hat{\mathbf{v}}_x = \hat{\mathbf{v}}_{ALDU}$: the inverse-equivalent $ALDU$ factorization, from Section 3.2, applied directly to the diagonally dominant T_N.

These numerical results indicate that on fine grids, multigrid out preforms traditional backward stable algorithms, but under performs in comparison with inverse-equvalent algorithms. We can see that all three are comparable up until $L = 18$, or $\kappa(A_h) \approx 10^{10}$. At this point, ρ_{back} begins to increase, and continues to increase for the remainder of the table; meanwhile, ρ_{FMG} and ρ_{ALDU} continue to decrease quadratically. However, around $L = 20$ we begin to see significant differences in these remaining competing errors. For $L > 20$, ρ_{FMG} remains one to two orders of magnitude worse than those of an inverse-equivalent algorithm. More explicitly, at $L = 21$ the FMG error actually begins to increase, while ρ_{ALDU} remains consistent in its performance. From this table, it is not clear whether or not the ρ_{FMG} explicitly depends on $\kappa(A_h)$; although it seems to suggests that $\kappa(A_h)$ might play some small roll in the FMG error. The rest of this thesis is dedicated analysing roundoff errors in the building blocks of FMG, and setting a foundation for understanding the increasing errors of ρ_{FMG} for very large N.

Table 5.1: Example 5.1: The discrete boundary value problem eq. (5.4) with $\mathbf{f}_h(i) = 3\sin(2\pi\mathbf{x}(i))$ and exact solution $\mathbf{v}_h(i) = \frac{3}{4\pi^2}\sin(2\pi\mathbf{x}(i))$ is solved using full multigrid, MATLAB's built-in backslash, and the inverse-equivalent $ALDU$ algorithm. The table shows relative errors for increasingly ill-conditioned systems.

		Relative error $\rho_x = \frac{\|\mathbf{v}_h - \hat{\mathbf{v}}_x\|_2}{\|\mathbf{v}_h\|_2}$		
L	$\kappa(A_h)$	ρ_{FMG}	ρ_{back}	ρ_{ALDU}
2	8	2.3 e -01	2.3 e-01	2.3e-01
3	3.2e1	5.3e-02	5.3e-02	5.3e-02
4	1.3e2	1.3e-02	1.3e-02	1.3e-02
5	5.1e2	3.2e-03	3.2e-03	3.2e-03
6	2e3	8.0e-04	8.0e-04	8.0e-04
7	8e3	2.0e-04	2.0e-04	2.0e-04
8	3e4	5.0e-05	5.0e-05	5.0e-05
9	1.3e5	1.2e-05	1.2e-05	1.2e-05
10	5.2e5	3.0e-06	3.0e-06	3 .0e-06
11	2e6	7.8e-07	7.8e-07	7.8e-07
12	8e6	1.9e-07	1.9e-07	1.9e-07
13	3e7	4.9e-08	4.9e-08	4.9e-08
14	1e8	1.2e-08	1.2e-08	1.2e-08
15	5e8	3.1e-09	3.0e-09	3.0e-09
16	2e9	7.6e-10	7.6e-10	7.6e-10
17	8e9	1.2e-10	1.2e-10	2.0e-10
18	3e10	5.0e-11	1.5e-08	4.8e-11
19	1e11	1.3e-11	1.3e-07	1.2e-11
20	5e11	6.5e-12	4.6e-07	3.0e-12
21	1.8e12	2.6e-11	8.2e-07	7.9e-13
22	7e12	8.1e-11	8.0e-07	1.3 e-13
23	3e13	5.4e-11	1e-06	2.1e-13
24	1e14	3.0e-10	1e-05	4.5e-13

Chapter 6 Multigrid Algorithms

The general multigrid principle is fairly simple. In fact, each of its elements, considered separately, were already known and had been used for a long time before they were combined into what we know as modern multigrid methods. The 1960s saw the first studies introducing and investigating multigrid methods by R.P. Fedorenko and N.S. Bakhvalov. Later, Achi Brandt was the first to address the actual efficiency of multigrid solvers, and is often considered a general pioneer of the subject. We do not attempt to list or summarize the vast and important work of Brandt and the many other historically relevant contributors (namely, Hackbusch) to the subject, but instead refer the reader to [21] (and the references therein) for a nice historical overview, as well as a systematic introduction to multigrid methods for the solution of elliptic differential equation. A bit dated, the field has certainly expanded and generalized in many interesting ways since 1989 (e.g. multigrid preconditioning, algebraic multigrid, etc). However, for our purposes, we need only a somewhat basic framework of common multigrid methods. This chapter is an amalgamation of [6], [12, Chapter 6.7], and [22], to present the necessary background and understandings of a multigrid procedure. When necessary, we cite (possibly other) explicit references to appropriately steer the interested reader.

6.1 Basic Multigrid

We again start by considering solving eq. (5.1) discretized as the $N \times N$ linear equation

$$A_h \mathbf{v}_h = \mathbf{f}_h \quad \text{on grid} \quad G_h, \tag{6.1}$$

using a general iterative scheme based on the approximate solution of the residual equation.

Let $\mathbf{v}_h^{(m)}$ be an approximation to to eq. (6.1) and denote the error as

$$\mathbf{e}_h^{(m)} \coloneqq \mathbf{v}_h - \mathbf{v}_h^{(m)} \text{ and the residual by } \mathbf{r}_h^{(m)} \coloneqq \mathbf{f}_h - A_h \mathbf{v}_h^{(m)}.$$

We look to solve the residual equation $A_h \mathbf{e}_h^{(m)} = \mathbf{r}_h^{(m)}$ (which is equivalent to eq. (6.1)) by using some "simpler" operator $\hat{A}_h$ such that $\hat{A}_h \approx A_h$ and $\hat{A}_h^{-1}$ exists, then computing an updated approximation

$$\mathbf{v}_h^{(m+1)} \coloneqq \mathbf{v}_h^{(m)} + \hat{\mathbf{e}}_h^{(m)},$$

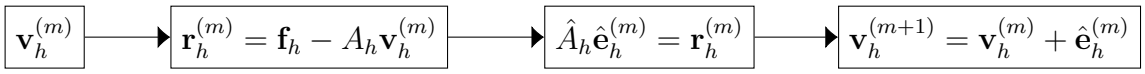
where $\hat{\mathbf{e}}_h^{(m)}$ satisfies $\hat{A}_h \hat{\mathbf{e}}_h^{(m)} = \mathbf{r}_h^{(m)}$. Figure 6.1 illustrates this type of iterative scheme.

Starting with some initial vector $\mathbf{v}_h^{(0)}$, this iterative process can be written in terms of an iteration operator, M_h:

$$\mathbf{v}_h^{(m+1)} = M_h \mathbf{v}_h^{(m)} + \hat{A}_h^{-1} \mathbf{f}_h \tag{6.2}$$

$$M_h = I_h - \hat{A}_h^{-1} A_h. \tag{6.3}$$

Figure 6.1: General iterative scheme for solving $A_h\mathbf{v}_h = \mathbf{f}_h$



Multigrid algorithms are iterative algorithms of this general form but infused with two basic ingredients: coarse-grid correction and smoothing techniques. Coarse-grid correction defines the operator $\hat{A}_h$, while the smoothing eliminates oscillatory components of the error and ensures that the iterative process is well behaved. Together, these two components define the multigrid iteration operator. To understand multigrid algorithms, we first describe in detail how these two components come together to define the structure of the 2-grid operator.

6.2 Coarse Grid Correction and Structure of the 2-Grid Operator

The 2-grid operator, or the 2-grid cycle, is an essential building block of any multigrid algorithm. It combines a 2-level coarse grid correction with smoothing.

In coarse grid correction, the idea is to solve the residual equation using $\hat{A}_h = A_H$ – an approximation of A_h on some coarser grid G_H. There are of course several choices of H one could make; however, throughout this thesis we define successively coarser grids by doubling the mesh-size of the previous grid. And explicitly in the case of the 2-grid cycle, we consider only the next coarsest grid, taking $H = 2h$. That is, we approximate A_h by A_{2h}.

Taken on its own in the context of a general iterative scheme, however, coarse grid correction produces an algorithm which does not converge. In order to define a convergent 2-grid operator, we must combine this process with a few iterations of a smoothing algorithm. The basic idea of smoothing is as follows. In general, a smooth error term is well approximated on a coarse grid; thus, the role of the smoothing steps is to damp (or smooth) the high frequency components of the error $\mathbf{e}_h^{(m)}$ of approximations $\mathbf{v}_h^{(m)}$, thereby making coarse grid correction (i.e. transferring functions between grids) a more accurate process.

Thus, each iteration of a 2-grid cycle consists of pre-smoothing, coarse grid correction, and post-smoothing. Before we can state the 2-grid cycle (hence the 2-grid operator) then, we must define some operators.

Definition 6.1. *We denote the following operators.*

> $\mathbf{J_h^{2h}} : G_h \to G_{2h}$ *is a* ***restriction operator*** *which takes a vector* $\mathbf{v}_h$ *on grid* G_h *and maps it the vector* $\mathbf{v}_{2h}$*, which is an approximation on coarser grid* G_{2h}*.*
>
> $\mathbf{J_{2h}^h} : G_{2h} \to G_h$ *is a* ***prolongation operator*** *which takes a vector* $\mathbf{v}_{2h}$ *on grid* G_{2h} *and converts it to the vector* $\mathbf{v}_h$*, the approximation on the finer grid* G_h*.*
>
> $\mathbf{S_h} : G_h \to G_h$ *is an (iterative)* ***smoothing operator****, which takes and approximate solution* $\mathbf{v}_h$ *and computes an improved (smooth) solution* $\bar{\mathbf{v}}_h$ *by damping high frequency components.*

In Section 6.4 we present more details on these operators and comment on how to choose them.

Using this notation, we can now write down the 2-grid iteration cycle, which takes in approximate solution $\mathbf{v}_h^{(m)}$ and returns the updated approximation $\mathbf{v}_h^{(m+1)}$ using only the original grid G_h and the next coarsest grid G_{2h}. The structure of a single 2-grid cycle described by Algorithm 6.1 is depicted in Figure 6.2.

Algorithm 6.1. ***2-Grid Cycle*** $\mathbf{v}_h^{(m)} \to \mathbf{v}_h^{(m+1)}$

1. Pre-smoothing. *Apply ν_1 steps of some smoothing operator S_h:*

$$\bar{\mathbf{v}}_h^{(m)} = S_h^{\nu_1} \mathbf{v}_h^{(m)}$$

2. Coarse-grid Correction.

   a) *Compute residual:* $\bar{\mathbf{r}}_h^{(m)} = \mathbf{f}_h^{(m)} - A_h \bar{\mathbf{v}}_h^{(m)}$.

   b) *Restrict residual:* $\bar{\mathbf{r}}_{2h}^{(m)} = J_h^{2h} \bar{\mathbf{r}}_h^{(m)}$.

   c) *Solve residual equation on G_{2h}:* $A_{2h} \hat{\mathbf{e}}_{2h}^{(m)} = \bar{\mathbf{r}}_{2h}^{(m)}$.

   d) *Prolongate solution:* $\hat{\mathbf{e}}_h^{(m)} = J_{2h}^h \hat{\mathbf{e}}_{2h}^{(m)}$.

   e) *Update:* $\bar{\mathbf{v}}_h^{(m+1)} = \bar{\mathbf{v}}_h^{(m)} + \hat{\mathbf{e}}_h^{(m)}$

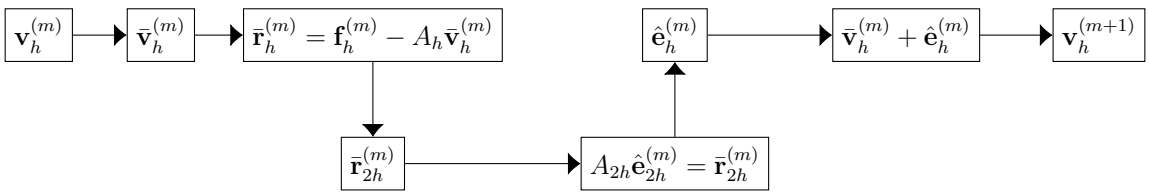
3. Post-smoothing. *Apply ν_2 post-smoothing steps:*

$$\mathbf{v}_h^{(m+1)} = S_h^{\nu_2} \bar{\mathbf{v}}_h^{(m+1)}$$

In matrix form, the 2-cycle iteration operator is thus given by

$$M_h^{2h} = S_h^{\nu_2} \left(I_h - J_{2h}^h A_{2h}^{-1} J_h^{2h} A_h \right) S_h^{\nu_1}. \tag{6.4}$$

Figure 6.2: Structure of a single iteration of the 2-grid cycle given by Algorithm 6.1. The two levels are representative of the two grids: G_h on top and G_{2h} on bottom. Vertical arrows denote transferring between grids while horizontal arrows represent computations on the current grid.



6.3 Multigrid V-cycle and Full Multigrid

The multigrid V-cycle embeds the 2-grid cycle within itself: instead of solving the residual equation $A_{2h}\hat{\mathbf{e}}_{2h}^{(m)} = \bar{\mathbf{r}}_{2h}^{(m)}$ on grid G_{2h}, we approximate it by performing coarse-grid correction on the next coarsest grid G_{4h}, etc. Repeating this process on successively coarser grids until an exact solution of the residual equation is possible defines an L-grid V-cylce, and is stated in its recursive form in Algorithm 6.2. First, we make the following remark on notation.

Remark. Beginning with a meshsize $h = \dfrac{1}{N} = \dfrac{1}{2^L}$ determines the number, L, of grids used. As depicted in Figure 6.3 (left), grid G_h is level 1, grid G_{2h} is level 2, etc. The l^{th} level represents a grid of mesh-size $h_l = 2^{l-1}h$, down to the coarsest level L which has grid of meshsize $h_L = 2^{L-1}h = \frac{1}{2}$. That is, on the coarsest grid, solving the residual equation is simple division. In the rest of this section, we will denote by G_l the grid of meshsize h_l, and, consequently, vectors and operators defined on grid G_l shall be denoted similarly. That is, on the finest grid, $G_1 \coloneqq G_h$, $\mathbf{v}_1 \coloneqq \mathbf{v}_h$, etc. In general we will employ this notation when dealing with general $l$grid scenarios but in the case of 2- and 3-grid cycles, we use explicit h notation.

Algorithm 6.2. ***L-Grid V-Cycle*** $\mathbf{v}_1^{(m)} \to V^L(\mathbf{v}_1^{(m)}, \mathbf{f}_1) = \mathbf{v}_1^{(m+1)}$

1. Pre-smoothing. *Apply ν_1 smoothing steps:* $\bar{\mathbf{v}}_1^{(m+1)} = S_1^{\nu_1}\mathbf{v}_1^{(m)}$
2. Coarse-grid Correction. *If $G_1 = G_L$ is coarsest grid, go to step 3. Else,*
   a) *Restrict computed residual:* $\bar{\mathbf{r}}_2^{(m)} = J_1^2\left(\mathbf{f}_1^{(m)} - A_1\bar{\mathbf{v}}_1^{(m)}\right)$.
   b) *Solve residual equation by $(L-1)$-grid V-cycle using zero vector as first approximation:*
   $$\hat{\mathbf{e}}_2^{(m)} = V^{L-1}(\mathbf{0}, \bar{\mathbf{r}}_2^{(m)})$$
   c) *Prolongate and update:* $\bar{\mathbf{v}}_1^{(m+1)} = \bar{\mathbf{v}}_1^{(m)} + J_2^1\hat{\mathbf{e}}_2^{(m)}$.
3. Post-smoothing. *Apply ν_2 smoothing steps:* $\mathbf{v}_1^{(m+1)} = S_1^{\nu_2}\bar{\mathbf{v}}_1^{(m+1)}$

The V-cycle is a building block of the full multigrid (FMG) algorithm – FMG combines the V-cycle with nested iteration; it uses a single iteration of a l-grid V-cycle (for $l = 1, \ldots, L$) to provide a good initial approximation for an $(l+1)$-grid V-cycle. The full multigrid scheme is represented pictorially in Figure 6.3 (right), and the recursive FMG algorithm is given in Algorithm 6.3.

Algorithm 6.3. ***Full Multigrid*** $\mathbf{v}_1^{(m)} \to FMG^L(\mathbf{f}_1) = \mathbf{v}_1^{(m+1)}$

1. *If $G_1 = G_L$ is the coarsest grid, go to step 3. Else,*
   a) *Restrict right-hand side:* $\mathbf{f}_2 = J_1^2\mathbf{f}_1$.
   b) *Solve:* $\mathbf{v}_2 = FMG^{L+1}(\mathbf{f}_2)$.

2. Update. $\mathbf{v}_1 = J_2^1 \mathbf{v}_2$.

3. $\mathbf{v}_1 = V^L(\mathbf{v}_1, \mathbf{f}_1)$.

Finally, we can use eq. (6.2) and eq. (6.4) to write down the iteration matrix for solving eq. (6.1) using the V-cycle described in Algorithm 6.2. Specifically, M_l^L denotes the iteration matrix starting on grid level l and corresponding to the multigrid cycle that employs L grid levels. It is given recursively by ([22], [24]):

$$M_l^L = S_l^{\nu_2}\left(I_l - J_{l+1}^l (I_{l+1} - M_{l+1}^L) A_{l+1}^{-1} J_l^{l+1} A_l\right) S_l^{\nu_1} \quad \text{for} \quad l = 1, \ldots, L-1, \quad (6.5)$$
$$M_0^0 = 0.$$

We note that there are other options for multigrid algorithms, as various structures can be created by combining different numbers of grids and cycling these grids different numbers of times (resulting in W-cycles, F-cycles, etc. See [6], [22] for details). We do not consider these other structures here. FMG is typically the most efficient multigrid method, thus this paper focuses on its V-cycle building blocks.

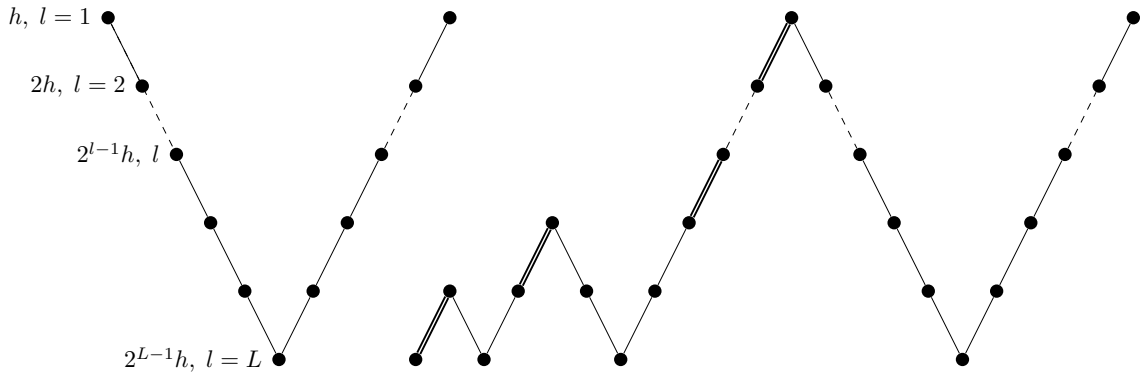
In the following section we provide discussion and more details on the various operators which compose a multigrid algorithm.

6.4 Multigrid Components and Convergence

As we have seen, there are many choices to be made when constructing a multigrid algorithm. And these choices, which are often highly problem dependent, can have a strong influence on the efficiency of the algorithm. In this section, we present various examples of how some of the multigrid components can be specified, discuss the implication of such choices, and the relationships between them. Finally, we state the Model Algorithm.

We have already fixed some choices. We consider only standard center difference discretizations, standard coarsening of doubling meshsize $H = 2h$ (other options include: semicoarsening, red-black coarsening, $4h$ coarsening), the natural coarse

Figure 6.3: Structure of a multigrid V-cycle (left) and the full multigrid scheme (right).



grid operator $\hat{A}_h = A_{2h}$ (other options include: the Galerkin coarse grid operator), and L-grid V-cycles (other options include F and W cycles). We are left to determine smoothing, restriction, and interpolation operators, and the number of pre- and post-smoothing steps.

Fortunately, for most multigrid purposes, the simplest transfer operators are quite effective. Moreover, many classical iterative methods, when applied to discrete elliptic problems, have a strong smoothing effect on the error of an approximation: they tend to damp oscillatory terms, making high frequency components become small while low frequency components are hardly changed. The following example outlines a few standard, and well studied, restriction, prolongation, and smoothing operators.

Example 6.1. We give several examples of choices of operators J_h^{2h}, J_{2h}^h and S_h composing a 2-grid cycle. We give details only on those operators which will be used later to compose the Model Algorithm.

(a) Restriction Operator J_h^{2h}.

(a.1) The simplest way to obtain $\mathbf{v}_{2h}$ is to sample $\mathbf{v}_h$ at the common grid points (injection), but averaging values of $\mathbf{v}_h$ at neighboring points is a better approach: full weight and half weight averaging are common choices.

(a.2) The 1D full weight restriction operator is given by

$$J_h^{2h}\mathbf{v}_h(x) = \frac{1}{4}\left(\mathbf{v}_h(x-h) + 2\mathbf{v}_h(x) + \mathbf{v}_h(x+h)\right) = \mathbf{v}_{2h}(x) \quad \text{for} \quad x \in G_{2h}$$

with the analogous matrix representation

$$J_h^{2h} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & & & & & \\ & & 1 & 2 & 1 & & & \\ & & & & & \ddots & & \\ & & & & & & 1 & 2 & 1 \end{bmatrix} \in \mathbb{R}^{N/2-1,N-1}.$$

(b) Prolongation Operator J_{2h}^h.

(b.1) Linear (or bilinear) interpolation are the most common prolongation operators. In 1d, linear interpolation is given by the following.
For $x \in G_h$,

$$\mathbf{v}_h(x) = J_{2h}^h\mathbf{v}_{2h}(x) = \begin{cases} \mathbf{v}_{2h}(x) & \text{if } x \in G_{2h} \\ \frac{1}{2}\left(\mathbf{v}_{2h}(x-h) + \mathbf{v}_{2h}(x+h)\right) & \text{otherwise.} \end{cases}$$

Which has the matrix representation

$$J_{2h}^{h} = \frac{1}{2}\begin{bmatrix} 1 & & & & \\ 2 & & & & \\ 1 & 1 & & & \\ & 2 & & & \\ & 1 & \ddots & & \\ & & & 1 & \\ & & & 2 & \\ & & & 1 & \end{bmatrix} \in \mathbb{R}^{N-1,N/2-1}.$$

We note that weighted restriction and linear interpolation have the desirable property that $J_{2h}^{h} = c\left(J_{h}^{2h}\right)^{T}$. In the case of full weight restriction we have $c = 2$.

(c) Smoothing Operator S_h.

(c.1) Gauss-Seidel-type iterations: lexicographic ordering, red-black ordering, zebra line, with a relaxation parameter (successive overrelaxation), etc.

(c.2) Jacobi-type iterations: A single iteration of weighted Jacobi, with weight ω, is written in component form as:

$$\mathbf{v}_h^{(m+1)}(x) = \frac{1}{2}\left(\mathbf{v}_h^{(m)}(x-h) + \mathbf{v}_h^{(m)}(x+h) + h^2\mathbf{f}_h(x)\right) \quad \text{for} \quad x \in G_h.$$

We can obtain an iteration matrix S_h as defined by weighted Jacobi by splitting A_h in the form $A_h = D_h - L_h - U_h$, where D_h is the diagonal of A_h, and $-L_h$ and $-U_h$ are the strictly upper and lower triangular parts of A_h, respectively. This yields [12]:

$$\begin{aligned} &\mathbf{v}_h^{(m+1)} = S_h\mathbf{v}_h^{(m)} + \omega D_h^{-1}\mathbf{f}_h \\ &\text{where} \quad S_h = [(1-\omega)I_h + \omega R_J], \\ &\text{and} \qquad R_J = D_h^{-1}\left(L+U\right). \end{aligned}$$

It is worth noting that interpolation is most effective when the error is smooth while relaxation is most effective when the error is oscillatory. Indeed, suppose the exact error on G_h is smooth and approximated on G_{2h}, then the interpolant of this coarse grid error *back* to the fine grid is also smooth. On the contrary, if the exact error is oscillatory, then even a very good coarse grid approximation my produce an interpolant that is not very accurate [6].

Relaxation schemes, on the other hand, tend to eliminate oscillatory modes and leave smooth modes. Think of the error as a sum of sine curves of different frequencies: then the work performed on a particular grid (i.e. averaging the solution at each point with its neighbors) makes the solution smoother, which is equivalent to getting rid of the high frequency error.

The complementarity at work here is truly wonderful: Relaxation on the fine grid eliminates oscillatory components and leaves us with a relatively smooth error, and because this error is smooth interpolation works well in accurately transferring this error back to the fine grid (after solving on G_{2h} of course) leaving us with an effective correction of the fine grid solution.

Now, the specific smoothing properties attained by a chosen smoothing operator S_h depends on the right choice of relaxation parameters, and also on the ordering of the grid points. As it turns out, appropriate Gauss-Seidel iterations are generally better smoothers than Jacobi iterations; however, we leave our attention on weighted Jacobi. In particular, for our Model Problem, weighted Jacobi with $\omega = \frac{2}{3}$ is good at decreasing the high-frequency error: the upper half of the error components are multiplied by $\frac{1}{3}$ or less at each iteration (independently of N) while the low-frequency error components are not decreased as much [12].

The number of pre- and post-smoothing (ν_1 and ν_2 respectively) are, again, problem dependent, but are usually small integers. A general observation is that it does not pay to use large values for ν_1 and ν_2, and one should instead put the work towards carrying out a few more multigrid cycles. The numbers are often chosen experimentally (or through a Fourier analysis), and it is common practice to choose $\nu = \nu_1 + \nu_2 \leq 3$ ([22], [21]).

These considerations all play in to what it means to design a "good" algorithm. It is necessary to look into both the speed of convergence of an algorithm and its cost: A typical multigrid analysis aims to show h-independent fast convergence, and a computational work cost proportional to the number of unknowns N. There are a variety of tools and approaches for analysing multigrid methods theoretically, each offering their own sets of pros and cons, but our main approach will be rigorous Fourier analysis. Our focus, however, is not on *designing* an appropriate algorithm, but rather, *given* a convergent algorithm, determining how accurate the computed solution is. Therefore, we establish a Model Algorithm (for solving the Model Problem) which will provide a foundation for our theoretical approach, and frame our primary analysis.

Model Algorithm. We solve eq. (5.4) using the full multigrid algorithm (but note that much of the analysis will focus on the building block of a single V-cycle) using the following components:

- J_h^{2h} is full weight restriction.
- J_{2h}^h is linear interpolation.
- S_h is weighted Jacobi with $\omega = \frac{2}{3}$. For T_N, this iteration matrix is $S_h = I_h - \frac{\omega}{2} T_N$.
- $\nu_1 = 2$ and $\nu_2 = 1$.

In [12] a convergence proof of this algorithm is sketched. In particular, the overall error in a V-cycle is decreased by a constant less than 1, independent of the grid size, and the total work is proportional to the cost of a single V-cycle.

In the remaining sections of this chapter, we introduce the tools of rigorous Fourier analysis and basic results which will be used in the following chapter to analyze roundoff errors. We focus primarily on a *rigorous* Fourier analysis, but, in light of limitations, make note of the widely used *local* Fourier analysis.

6.5 Rigorous Fourier Analysis

The tools and results presented in this section are standard in the multigrid literature. The original provision of this flavor of analysis is due to Achi Brandt, [5] (much of which originally appeared in [4]). Used explicitly to analyze the efficiency of a multigrid algorithm, or to design an efficient algorithm for a problem at hand, these tools have been widely applied to and studied on many specific scenarios and model problems. Here, we have primarily relied upon [19], [21], [22], and [24]. Furthermore, in [23], Wienands and Joppich comprehensively describe what is needed for multigrid algorithms in a realistic simulation environment, and they focus on practically and computationally answering questions such as *How good is the convergence?* within a very particular application. In this paper, because we care about solving a predetermined linear system, we present Fourier analysis with a more linear algebra flavor.

At its core, Fourier analysis of multigrid algorithms seeks to exploit properties of the various operators in order to "nearly" diagonalize the multigrid iteration matrix M_l^L by the eigenvectors of the coefficient matrix A_h. Since the operators composing M_l^L are inherently defined on different spaces, this process requires us to identify *high and low frequency components* of the eigenvectors of A_h.

This notion of high and low frequency components was present in the Section 6.4 discussion of smoothing operators. Recall that a property of good smoothing operators is that after a few iterations, the high frequency (oscillatory) components of the error are damped, while the lower frequency components hardly change. Further recall that a smooth error term is well approximated on a coarse grid. In this chapter we will see that the low frequency components of an error on G_h also represent meaningful grid functions on G_{2h}, whereas the high frequency components do not. Specifically, the high frequencies are not visible on the coarser grid G_{2h}.

Consider an arbitrary discrete operator L_h corresponding to a difference stencil

$$[s_\kappa]_h = [s_{\kappa_1,\kappa_2}]_h = \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & s_{-1,1} & s_{0,1} & s_{1,1} & \cdots \\ \cdots & s_{-1,0} & s_{0,0} & s_{1,0} & \cdots \\ \cdots & s_{-1,-1} & s_{0,-1} & s_{1,-1} & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix}_h$$

such that $L_h \mathbf{v}_h(x) = \sum_{\kappa \in V} s_\kappa \mathbf{v}_h(x + \kappa h)$, where V is a finite index set. If we assume that G_h is an infinite grid, (hence the influence of boundaries and of boundary conditions is not taken into account), then the grid functions of L_h

$$\phi_h^k(x) = e^{ik\pi x} \quad \text{where} \quad x \in G_h \quad \text{and} \quad k = 1, \ldots, N-1$$

are formal eigenfunctions of L_h. In particular, these grid functions satisfy

$$L_h\phi_h^k(x) = \left(\sum_{\kappa\in V} s_\kappa e^{(ik\pi)\cdot\kappa}\right)\phi_h^k(x) = \tilde{L}_h(k)\phi_h^k(x).$$

Moreover, we note that for any two grid functions $\phi_h^k(x)$ and $\phi_h^{k'}(x)$ we have

$$\phi_h^k(x) = \phi_h^{k'}(x) \text{ if and only if } k' \equiv k \bmod 2N.$$

This allows us to distinguish between high and low frequency grid functions–that is, grid functions which are representable on G_{2h} (so-called low frequency), and those which are not visible on G_{2h}. That is, a grid function ϕ_h^k either coincides with a low-frequency grid-function on G_{2h}, or it vanishes on G_{2h}, and we call it a high-frequency grid function. These L_h grid functions are the basis of *local Fourier analysis (LFA)*. In the literature, the goal of LFA is to determine smoothing factors for S_h, two-grid convergence factors, and error reduction factors for 2-grid iteration operator M_h^{2h} (eq. (6.4)). In the absence of boundary conditions, the objective of LFA is to determine quantitative results which a multigrid algorithm *can* attain if a proper boundary treatment is included.

We note that standard LFA denotes the grid functions as $\phi_h(\theta, x)$, introducing the continuous parameter $\theta \in [-\pi, \pi)$, then distinguishes high and low frequency components on G_h with respect to G_{2h} by establishing that

$$\phi(\theta, x) = \phi(\theta', x) \text{ for } x \in G_{2h} \text{ if and only if } \theta = \theta' \bmod \pi.$$

Then $\phi(\theta, x)$ is a low-frequency component if $\theta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)$, and a high-frequency component if $\theta \in \left[-\pi, -\frac{\pi}{2}\right) \cup \left[\frac{\pi}{2}, \pi\right)$.

Since our goals live in the fame work of solving a linear system of the form eq. (6.1), which includes boundary information, we will only consider grid functions $\phi_h^k(x)$ of a diagonalizable discrete operator A_h, where the $\phi_h^k(x)$ that are formal eigenfunctions of A_h. This is referred to as *rigorous Fourier analysis*. We note that the L_h-grid functions and A_h-grid functions agree on interior points and have the same eigenvalues, and, in fact, rigorous Fourier analysis results can be derived within the framework of LFA.

Although, we do advise one to proceed with caution, as the applicability of rigorous Fourier analysis is restricted. For example, it requires particular subspaces of span$\{\phi_h^k\}$ be invariant under the operators composing M_l^L and therefore cannot analyze Gauss-Seidel with lexicographic ordering as a smoother; it cannot handle non-symmetric operators nor nonrectangular domains, and requires constant coefficients in the original differential equation. On the other hand, rigorous Fourier analysis has been applied in detail to a multitude of situations. See [22] and the references therein for a comprehensive list, and [21] for many complete examples.

Now, in the final section of this chapter, we will describe rigorous Fourier analysis, as used in the 2- and 3-grid cycles, in detail.

6.6 Eigenmatrix Representations of Multigrid Operators using Rigorous Fourier Analysis

Let $(\phi_h^k(x), \lambda_h^k)$ be an eigenpair of A_h, where $x \in G_h$ and $k = 1, \ldots, N$. Note that under the assumption that A_h is invertible, the ϕ_h^k form a basis of R^N.

For the 2-grid case, recall that we have the iteration operator

$$M_h^{2h} = S_h^{\nu_2} \left(I_h - J_{2h}^h A_{2h}^{-1} J_h^{2h} A_h \right) S_h^{\nu_1}.$$

Let $k^{(1)}$ be an integer in some subset $\mathcal{N}$ of $[N]$ such that the $\phi_h^{k^{(1)}}$ represent the low frequency components of $\left\{\phi_h^k\right\}_{k=1}^N$, and for each $k^{(1)}$ there is an associated $k^{(2)}$ such that $\phi_h^{k^{(2)}}$ is a high frequency grid function. The high and low frequency grid functions are defined such that they satisfy:

$$\phi_h^{k^{(1)}}(x) \text{ and } \phi_h^{k^{(2)}}(x) \text{ agree up to sign for } x \in G_{2h}. \tag{6.6}$$

Often, $k^{(2)} \in [N] \backslash \mathcal{N}$ is a function of $k^{(1)}$ such that $\bigcup_{\mathcal{N}} \left\{k^{(1)}, k^{(2)}\right\} = [N]$.

High and low frequencies are used to define the $h-2h$ space of harmonics. We note that the definition of both high/low frequency grid functions and harmonic spaces occur in [22] and [21] as applied to a 2d model problem as well in the context of local Fourier analysis; here, we derived the 1d analogue and made some generalizations.

Definition 6.2. *Let $\phi_h^k(x)$ denote the discrete (orthogonal) eigenfunctions of an invertible matrix $A_h \in \mathbb{R}^{N \times N}$, and let $\mathcal{E}_h = span\left\{\phi_h^k\right\}_{k=1}^N$. Let $k^{(1)}$ and $k^{(2)}$ be as in eq.* (6.6).

Then we define $\mathcal{E}_h^{2h}$, the $2-$dimensional spaces of $h-2h$ ***harmonics*** *as*

$$\mathcal{E}_h^{2h} = span\{\phi_h^{k^{(1)}}(x), \pm\phi_h^{k^{(2)}}(x)\}$$

for $x \in G_h$ and $k^{(1)}, k^{(2)} \in [N]$ such that $\bigcup_{k^{(1)},k^{(2)}} \mathcal{E}_h^{2h} = \mathcal{E}_h$.

Note that the $\mathcal{E}_h^{2h}$ is defined by decomposing $\mathcal{E}_h$ into subspaces defined by the high and low frequency components. The precise functions $\phi_h^{k^{(1)}}$ and $\phi_h^{k^{(2)}}$ are determined by the operator A_h; the operator $\mathcal{L}$ and the boundary conditions will both effect how high and low frequency functions are defined. In the following, to simplify notation, we denote $\pm\phi_h^{k^{(2)}}$ as simply $\phi_h^{k^{(2)}}$, with the understanding that for particular problems, some of the basis functions are used with a negative sign. This is made clear in the details of Chapter 8.

The space of harmonics are of interest because, under certain assumptions, they are invariant under both smoothing operators and corrections schemes; hence, they are invariant under the multigrid operator M_h^{2h}. We use this fact to obtain an eigenmatrix representation of M_h^{2h}. The eigenmatrix representation of a 2d model problem is derived in both [21] and [22] – mostly in the context of local Fourier analysis, although, [22] also explicitly derives it using rigorous Fourier analysis; and, [8] presents a 1d local Fourier analysis eigenmatrix representation. Here, we state a general rigorous Fourier analysis version of this eigenmatrix representation of M_h^{2h}.

Lemma 6.1. *For* $M_h^{2h} = S_h^{\nu_2}\left(I_h - J_{2h}^h A_{2h}^{-1} J_h^{2h} A_h\right) S_h^{\nu_1}$, *suppose we have that*

$$\begin{aligned}
&A_h : \mathcal{E}_h^{2h} \to \mathcal{E}_h^{2h} && \text{such that} && A_h\phi_h^{k^{(i)}} = \lambda_h^{k^{(i)}}\phi_h^{k^{(i)}}\\
&S_h : \mathcal{E}_h^{2h} \to \mathcal{E}_h^{2h} && && S_h\phi_h^{k^{(i)}} = \tilde{S}_h(k^{(i)})\phi_h^{k^{(i)}}\\
&A_{2h} : span\{\phi_{2h}^{k^{(1)}}\} \to span\{\phi_{2h}^{k^{(1)}}\} && && A_{2h}\phi_{2h}^{k^{(1)}} = \lambda_{2h}^{k^{(1)}}\phi_{2h}^{k^{(1)}}\\
&J_h^{2h} : \mathcal{E}_h^{2h} \to span\{\phi_{2h}^{k^{(1)}}\} && && J_h^{2h}\phi_h^{k^{(i)}} = \tilde{J}_h^{2h}(k^{(i)})\phi_{2h}^{k^{(1)}}
\end{aligned}$$

for $i = 1, 2$*; and further suppose that* $J_{2h}^h : span\{\phi_{2h}^{k^{(1)}}\} \to \mathcal{E}_h^{2h}$ *with*

$$J_{2h}^h\phi_{2h}^{k^{(1)}} = \begin{bmatrix}\tilde{J}_{2h}^h(k^{(1)}) & \tilde{J}_{2h}^h(k^{(2)})\end{bmatrix}\begin{bmatrix}\phi_h^{k^{(1)}}(x)\\ \phi_h^{k^{(2)}}(x)\end{bmatrix}.$$

Let $\psi_h^k(x) = \begin{bmatrix}\phi_h^{k^{(1)}}(x)\\ \phi_h^{k^{(2)}}(x)\end{bmatrix}$. *Then* $M_h^{2h} : \mathcal{E}_h^{2h} \to \mathcal{E}_h^{2h}$ *and*

$$M_h^{2h}\psi_h^k = M_h^{2h}(k)\psi_h^k$$

where $M_h^{2h}(k)$ *is the* 2×2 *matrix*

$$M_h^{2h}(k) = \begin{bmatrix}\tilde{S}_h(k^{(1)}) & \\ & \tilde{S}_h(k^{(2)})\end{bmatrix}^{\nu_2}\left\{I_2 - \frac{1}{\lambda_{2h}^{k^{(1)}}}\begin{bmatrix}\tilde{J}_{2h}^h(k^{(1)})\\ \tilde{J}_{2h}^h(k^{(2)})\end{bmatrix}\begin{bmatrix}\tilde{J}_h^{2h}(k^{(1)}) & \tilde{J}_h^{2h}(k^{(2)})\end{bmatrix}\begin{bmatrix}\lambda_h^{k^{(1)}} & \\ & \lambda_h^{k^{(2)}}\end{bmatrix}\right\}\begin{bmatrix}\tilde{S}_h(k^{(1)}) & \\ & \tilde{S}_h(k^{(2)})\end{bmatrix}^{\nu_1}. \tag{6.7}$$

Proof. Let the operators composing $M_h^{2h} = S_h^{\nu_2}\left(I_h - J_{2h}^h A_{2h}^{-1} J_h^{2h} A_h\right) S_h^{\nu_1}$ satisfy the assumptions in Lemma 6.1. Then the 2×2 matrix $M_h^{2h}(k)$ can be obtained directly by calculating $M_h^{2h}\phi_h^{k^{(i)}}$ for $i = 1, 2$ and using the assumptions on how each operator behaves on $\phi_h^{k^{(i)}}$. Indeed

$$S_h^{\nu_j}\phi_h^{k^{(i)}} = \tilde{S}_h^{\nu_j}(k^{(i)})\phi_h^{k^{(i)}} \quad \text{for} \quad i, j = 1, 2$$

together with

$$\left(I_h - J_{2h}^h A_{2h}^{-1} J_h^{2h} A_h\right)\phi_h^{k^{(i)}} = \phi_h^{k^{(i)}} - \frac{\lambda_h^{k^{(i)}}}{\lambda_{2h}^{k^{(1)}}} J_h^{2h}(k^{(i)})\left\{\tilde{J}_h^{2h}(k^{(1)})\phi_h^{k^{(1)}} + \tilde{J}_h^{2h}(k^{(2)})\phi_h^{k^{(2)}}\right\}$$

for $i = 1, 2$ yields the desired form eq. (6.7). □

The numbers $\lambda_h^{k^{(i)}}, \tilde{S}_h(k^{(i)}), \tilde{J_h^{2h}}(k^{(i)})$ and $\tilde{J_{2h}^h}(k^{(i)})$, which act like eigenvalues of their respective operators, are typically referred to in the literature as Fourier symbols. The entries of $M_h^{2h}(k)$, which can be found by simple matrix multiplication of eq. (6.7), can be written in terms of these Fourier symbols, and are given as:

$$\left(M_h^{2h}(k)\right)_{ij} = \begin{cases}\tilde{S}_h^{\nu}(k^{(i)})\left(1 - \frac{\lambda_h^{k^{(i)}}}{\lambda_{2h}^{k^1}}\tilde{J}_{2h}^h(k^{(i)})\tilde{J}_h^{2h}(k^{(i)})\right) & \text{for } i = j\\ -\tilde{S}_h^{\nu_1}(k^{(j)})\tilde{S}_h^{\nu_2}(k^{(i)})\frac{\lambda_h^{k^{(j)}}}{\lambda_{2h}^{k^1}}\tilde{J}_{2h}^h(k^{(i)})\tilde{J}_h^{2h}(k^{(j)}) & \text{for } i \neq j.\end{cases} \tag{6.8}$$

Furthermore, we note that if we let Q_h be the matrix of eigenvectors ϕ_h^k, ordered by the pairs $\phi_h^{k^{(1)}}, \phi_h^{k^{(2)}}$, then $Q_h^{-1} M_h^{2h} Q_h$ is a block diagonal matrix, where the k^{th} block is the 2×2 matrix $M_h^{2h}(k)$. This is the sense in which we say that M_h^{2h} is "nearly" diagonalizable.

We can similarly analyze a 3-grid cycle. Using, eq. (6.5) the 3-grid V-cycle iteration operator is ([8], [24]):

$$\begin{aligned} M_h^{4h} &= S_h^{\nu_2} \left\{ I_h - J_{2h}^h \left(I_{2h} - M_{2h}^{4h} \right) A_{2h}^{-1} J_h^{2h} A_h \right\} S_h^{\nu_1} \\ M_{2h}^{4h} &= S_{2h}^{\nu_2} \left(I_{2h} - J_{4h}^{2h} A_{4h}^{-1} J_{2h}^{4h} A_{2h} \right) S_{2h}^{\nu_1}. \end{aligned} \tag{6.9}$$

Where, instead of inverting A_{2h} in the 2-grid operator, the $2h$ equation is solved approximately by performing the 2-grid iteration M_{2h}^{4h} with zero initial approximation.

The Fourier analysis in this case, then, uses the space of $h - 4h$ harmonics, which is composed of two subspaces of $2h-$harmonics. Indeed, we simply incorporate the fact of eq. (6.6) in going from G_h to G_{2h}, then again going from G_{2h} to G_{4h}.

As before, let $k^{(1)}$ be an integer in some subset $\mathcal{N}$ of $[N]$ such that $\phi_h^{k^{(1)}}$ represent the low frequency grid functions. Then, for each $k^{(1)}$ we have an associated $k^{(2)}, k^{(3)}$, and $k^{(4)}$ where $\bigcup_{\mathcal{N}} \left\{ k^{(1)}, k^{(2)}, k^{(3)}, k^{(4)} \right\} = [N]$, and $\phi_h^{k^{(2)}}, \phi_h^{k^{(3)}}$, and $\phi_h^{k^{(4)}}$ are the high frequency grid functions such that all four functions agree up to sign on G_{4h}. These are used to define the $h - 4h$ space of harmonics. [24] defines the space of $h - 4h$ harmonics as well as an eigenmatrix representation for a 2d problem in terms of local Fourier analysis, while [8] states both for a 1d problem (still in terms of LFA). As before, we give a 1d, rigorous Fourier analysis presentation of the definition and eigenmatrix result.

Definition 6.3. *Let $\phi_h^k(x)$ denote the discrete eigenfunctions of $A_h \in \mathbb{R}^{N \times N}$ and let $\mathcal{E}_h = span \left\{ \phi_h^k(x) \right\}_{k=1}^N$. Let $k^{(1)} \in \mathcal{N} \subset [N]$ define a low frequency grid function $\phi_h^{k^{(1)}}$ as well as the associated high frequency grid functions $\phi_h^{k^{(i)}}$ for $i = 2, 3, 4$. Then we define the $h - 4h$* ***space of harmonics***

$$\mathcal{E}_h^{4h} = span \left\{ \phi_h^{k^{(1)}}, \phi_h^{k^{(2)}}, \phi_h^{k^{(3)}}, \phi_h^{k^{(4)}} \right\} = \mathcal{E}_h^{2h} \cup \mathcal{E}_{2h}^{4h}$$

for $x \in G_{4h}$, and $\bigcup_{k^{(i)}} \mathcal{E}_h^{4h} = \mathcal{E}_h$.

□

As before, under certain assumptions, $\mathcal{E}_h^{4h}$ is invariant under M_h^{4h}, and we can use this to obtain an eigenmatrix representation.

Lemma 6.2. *For M_h^{4h} as in (6.9), suppose we have that $A_h, S_h : \mathcal{E}_h^{4h} \to \mathcal{E}_h^{4h}$,*

$$\begin{aligned} J_h^{2h} &: \mathcal{E}_h^{4h} \to \mathcal{E}_{2h}^{4h}, \\ J_{2h}^h &: \mathcal{E}_{2h}^{4h} \to \mathcal{E}_h^{4h}, \\ \text{and } M_{2h}^{4h} &: \mathcal{E}_{2h}^{4h} \to \mathcal{E}_{2h}^{4h}, \end{aligned}$$

where $\mathcal{E}_{2h}^{4h} = span \left\{ \phi_h^{k^{(1)}}, \phi_h^{k^{(3)}} \right\}$. In particular, suppose we have $A_h \phi_h^{k^{(i)}} = \lambda_h^{k(i)} \phi_h^{k^{(i)}}$ and $S_h \phi_h^{k^{(i)}} = \tilde{S}_h(k^{(i)}) \phi_h^{k^{(i)}}$ for all $i = 1, \ldots, 4$. For the transfer operators, suppose

that $J_h^{2h}\phi_h^{k^{(i)}} = \tilde{J}_h^{2h}(k^{(i)})\phi_{2h}^{k^{(1)}}$ *for* $i = 1, 2$ *and* $J_h^{2h}\phi_h^{k^{(i)}} = \tilde{J}_h^{2h}(k^{(i)})\phi_{2h}^{k^{(3)}}$ *for* $i = 3, 4$ *and suppose*

$$J_{2h}^h\phi_{2h}^{k^{(1)}} = \begin{bmatrix}\tilde{J}_{2h}^h(k^{(1)}) & \tilde{J}_{2h}^h(k^{(2)})\end{bmatrix}\begin{bmatrix}\phi_h^{k^{(1)}} \\ \phi_h^{k^{(2)}}\end{bmatrix},$$

$$J_{2h}^h\phi_{2h}^{k^{(3)}} = \begin{bmatrix}\tilde{J}_{2h}^h(k^{(3)}) & \tilde{J}_{2h}^h(k^{(4)})\end{bmatrix}\begin{bmatrix}\phi_h^{k^{(3)}} \\ \phi_h^{k^{(4)}}\end{bmatrix}.$$

Let $\psi_h^k(x) = \begin{bmatrix}\phi_h^{k^{(1)}}(x) \\ \phi_h^{k^{(2)}}(x) \\ \phi_h^{k^{(3)}}(x) \\ \phi_h^{k^{(4)}}(x)\end{bmatrix}$, *then* $M_h^{4h} : \mathcal{E}_{2h}^{4h} \to \mathcal{E}_{2h}^{4h}$ *with*

$$M_h^{4h}\psi_h^k = M_h^{4h}(k)\psi_h^k$$

where

$$M_h^{4h}(k) = \begin{bmatrix}\tilde{S}_h(k) \\ \tilde{S}_h(k^{(2)}) \\ \tilde{S}_h(k^{(3)}) \\ \tilde{S}_h(k^{(4)})\end{bmatrix}^{D(\nu_2)}\left\{I_h - \begin{bmatrix}\tilde{J}_{2h}^h(k) & 0 \\ \tilde{J}_{2h}^h(k^{(2)}) & 0 \\ 0 & \tilde{J}_{2h}^h(k^{(3)}) \\ 0 & \tilde{J}_{2h}^h(k^{(4)})\end{bmatrix}\left[I_{2h} - M_{2h}^{4h}(k)\right]\begin{bmatrix}\frac{1}{\lambda_{2h}^k} \\ \frac{1}{\lambda_{2h}^{k^{(3)}}}\end{bmatrix}^D\begin{bmatrix}\tilde{J}_h^{2h}(k) & 0 \\ \tilde{J}_h^{2h}(k^{(2)}) & 0 \\ 0 & \tilde{J}_h^{2h}(k^{(3)}) \\ 0 & \tilde{J}_h^{2h}(k^{(4)})\end{bmatrix}^T\begin{bmatrix}\lambda_h^k \\ \lambda_h^{k^{(2)}} \\ \lambda_h^{k^{(3)}} \\ \lambda_h^{k^{(4)}}\end{bmatrix}^D\right\}\begin{bmatrix}\tilde{S}_h(k) \\ \tilde{S}_h(k^{(2)}) \\ \tilde{S}_h(k^{(3)}) \\ \tilde{S}_h(k^{(4)})\end{bmatrix}^{D(\nu_1)}.$$

Here, $\left[\cdot\right]^D$ *denotes transforming the vector into a diagonal matrix, and* $M_{2h}^{4h}(k)$ *is analogous to that of Lemma 6.1.*

Proof. As in Lemma 6.1, this lemma can be seen by using the assumptions and calculating $M_h^{4h}\phi_h^{k^{(i)}}$ for all $i = 1, \ldots, 4$. □

We note that the entries of $M_h^{4h}(k)$ are given by

$$(M_h^{4h})_{ij} = \begin{cases}\tilde{S}_h^{\nu}(k^{(i)})P(i,i) & i = j \\ \tilde{S}_h^{\nu_1}(k^{(i)})\tilde{S}_h^{\nu_2}(k^{(j)})P(i,j) & i \neq j\end{cases} \tag{6.10}$$

with

$$P(i,j) = \begin{cases}1 - \dfrac{\lambda_h^{k^{(i)}}}{\lambda_{2h}^{k^{(m)}}}\tilde{J}_{2h}^h(k^{(i)})\tilde{J}_h^{2h}(k^{(i)})[1 - (M_{2h}^{4h})_{nn}] & i = j \\ -\dfrac{\lambda_h^{k^{(j)}}}{\lambda_{2h}^{k^{(m)}}}\tilde{J}_{2h}^h(k^{(i)})\tilde{J}_h^{2h}(k^{(j)})[1 - (M_{2h}^{4h})_{nn}] & i + j = 7, 3 \\ \dfrac{\lambda_h^{k^{(j)}}}{\lambda_{2h}^{k^{(m)}}}\tilde{J}_{2h}^h(k^{(i)})\tilde{J}_h^{2h}(k^{(j)})[(M_{2h}^{4h})_{mn^T}] & \textit{otherwise.}\end{cases} \tag{6.11}$$

where

$$m = \begin{cases}1 & i = 1, 2 \\ 3 & i = 3, 4\end{cases} \qquad n = \begin{cases}1 & i = 1, 2 \\ 2 & i = 3, 4\end{cases} \qquad n^T = \begin{cases}1 & i = 3, 4 \\ 2 & i = 1, 2.\end{cases} \tag{6.12}$$

We can see that the 3-grid operator is defined recursively in terms of 2-grid restriction and smoothing operators and M_{2h}^{4h}; it was this representation, together with what we know about 2-grid operators, which allowed us to determine the 3-grid eigenmatrix representation. And in fact, we can represent any l-grid operator recursively in this way. We note that the representation eq. (6.13) of Lemma 6.3 is given in [22], but their concluding final bound on $\|M_l\|$ is slightly different. The author is unaware of the bound eq. (6.14) stated elsewhere in the literature, and it was developed here explicitly for future use in Chapter 8 and the analysis of the Model Problem.

Lemma 6.3. *The l-grid V-cycle operator can be written recursively in terms of 2-grid operators in the following way:*

$$\begin{aligned} M_l &= M_1^l = M_{l-1}^l + B_l^{l-1} M_{l-1} B_{l-1}^l \\ \text{where } B_l^{l-1} &= S_l^{\nu_2} J_l^{l-1} \\ B_{l-1}^l &= A_{l-1}^{-1} J_{l-1}^l A_l S_l^{\nu_1}. \end{aligned} \tag{6.13}$$

Suppose that $\|M_{l-1}^l\| \leq C_1$ and $\|B_l^{l-1}\| \cdot \|B_{l-1}^l\| \leq C_2$ then we have

$$\|M_l\| \leq \frac{C_1 \left(1 - C_2^{l-1}\right)}{1 - C_2}. \tag{6.14}$$

Proof. The equalities of eq. (6.13) follow directly from eq. (6.5). For eq. (6.14), we assume, as in the statement of the lemma, that $\|M_{l-1}^l\| \leq C_1$ and $\|B_l^{l-1}\| \cdot \|B_{l-1}^l\| \leq C_2$, then

$$\begin{aligned} &\|M_l\| \leq \|M_{l-1}^l\| + \|B_{l-1}^l\| \|B_l^{l-1}\| \|M_{l-1}\| \leq C_1 + C_2 \left(C_1 + C_2 \|M_{l-2}\|\right) \\ &\leq C_1 \left(1 + C_2 + C_2^2 (C_1 + C_2 \|M_{l-3}\|)\right) \leq \ldots \leq C_1 \left(\sum_{j=0}^{l-2} C_2^j \right) \\ &= C_1 \frac{1 - C_2^{l-1}}{1 - C_2}. \end{aligned}$$

□

The results of these sections allow us to determine floating point error bounds on multigrid operators. In this same manner, Chapter 7 analyses the error accumulation of two-grid, three-grid, and finally l-grid operators in matrix-vector multiplication with the respective multigrid iteration operators.

Chapter 7 Error Analysis of Computed Multigrid Solutions

As discussed in Section 5.2, the attention paid to error analysis in the literature seems to be wholly focused on discretization error. [19] formalizes what it means for an FMG algorithm to achieve discretization-level accuracy. They measure the error of exact multigrid solutions against the inherent discretization error, and use numerical experiments to validate this so-called worst-case relative accuracy measure. Further, [17] explores the use of multigrid methods in computing truncation error, which can then be used in extrapolation to higher order accuracy. But again, the spotlight remains on truncation error. In fact, most standard introductory texts (and historic contributions) contain sections addressing the accuracy of multigrid algorithms only in the context of convergence factors and the discretization-level accuracy. We want to understand the role of roundoff errors in multigrid algorithms, and the relative errors of *computed* discrete solutions as compared with exact discrete solutions.

Thus, with an ultimate goal of understanding how roundoff error accumulates in an l-grid V-cycle, we take the standard approach of begining with the $l = 2$ case and look at $fl\left(M_h^{2h}\psi_h\right)$. We then look at $fl\left(M_h^{4h}\psi_h\right)$ for the $l = 3$ case in the hope of extracting a pattern which indicates how error accumulates in moving from l grids to $(l + 1)$ grids. This, together with Lemma 6.3 will allow us to determine a bound on the relative error of $fl\left(M_l\psi_h\right)$, for a general l-grid V-cycle.

While some assumptions are necessary, we dedicate this chapter to stating results with the most possible generality. We will return to both the Model Problem and the Model Algorithm in Chapter 8.

7.1 2-Grid Analysis

As indicated by Lemma 6.3 (and Chapter 6 in general), two-grid operators are the building blocks of any multigrid algorithm, and, in particular of a V-cycle. Thus, we begin our analysis by supposing a linear system has been solved by a 2-grid V-cycle, and bounding the relative error on the computed solution. Recall that the 2-grid iteration operator is written as

$$M_h^{2h} = S_h^{\nu_2}\left(I_h - J_{2h}^h A_{2h}^{-1} J_h^{2h} A_h\right) S_h^{\nu_1}.$$

Theorem 7.1. *Suppose that the 2-grid iteration operator $M_2 = M_h^{2h}$ has an eigenmatrix representation as in Lemma 6.1, and assume that A_{2h}^{-1} exists explicitly, or can be solved for exactly. Let $k^{(1)} \in \mathcal{N} \subset [N]$ and $k^{(2)}$ define high and low frequency grid functions on G_h. Let $\tilde{\psi}_k = a_{k_1}\phi_h^{k^{(1)}} + a_{k_2}\phi_h^{k^{(2)}}$ for some coefficients a_{k_1} and b_{k_2} in $\mathbb{R}$ and let $M_2(k) = M_h^{2h}(k)$ for $k \in \mathcal{N}$. Then*

$$\|fl\left(M_2(k)\tilde{\psi}_k\right) - M_2(k)\tilde{\psi}_k\| \leq \|M_2(k)\tilde{\psi}_k\| \cdot |\delta_k^2| \tag{7.1}$$

where $|\delta_k^2| \leq (3\eta_h^{\max}(k) + 6 + \nu)\,\epsilon$ and

$$\eta_h^{\max}(k) = \max\left\{\left|\eta_h^{k^{(1)}}\right|, \left|\eta_h^{k^{(2)}}\right|\right\}$$

with

$$\eta_h^{k^{(i)}} = \frac{\lambda_h^{k^{(i)}}/\lambda_{2h}^{k^{(1)}}\ \tilde{J}_{2h}^h(k^{(i)})\tilde{J}_h^{2h}(k^{(i)})}{1-\lambda_h^{k^{(i)}}/\lambda_{2h}^{k^{(1)}}\ \tilde{J}_{2h}^h(k^{(i)})\tilde{J}_h^{2h}(k^{(i)})}. \tag{7.2}$$

Furthermore, the bound on $M_2\psi_h$ is given by

$$\frac{\|fl(M_2\psi_h) - M_2\psi_h\|}{\|M_2\psi_h\|} \leq \left(|\mathcal{N}| + 4 + \nu + 3\cdot\max_k\left\{\eta_h^{\max}(k)\right\}\right)\epsilon. \tag{7.3}$$

Proof. Let M_2 and $M_2(k)$ satisfy the desired assumptions and be as in Lemma 6.1; let $k^{(1)}$ and $k^{(2)}$ be as in the problem statement and recall that $\bigcup_{\mathcal{N}}\left\{k^{(1)}, k^{(2)}\right\} = [N]$. Now, since A_h is diagonalizable, for any $\psi_h \in \mathbb{R}^N$ we can write $\psi_h = \sum_{\mathcal{N}} a_{k_1}\phi_h^{k^{(1)}} + a_{k_2}\phi_h^{k^{(2)}}$ for some constants $a_{k_1}, a_{k_2} \in \mathbb{R}$, and thus we have

$$M_2\psi_h = \sum_{\mathcal{N}}\left(a_{k_1}M_2\phi_h^{k^{(1)}} + a_{k_2}M_2\phi_h^{k^{(2)}}\right) = \sum_{\mathcal{N}} M_2(k)\tilde{\psi}_k. \tag{7.4}$$

We first look at the computed $M_2(k)\tilde{\psi}_k$. From eq. (6.8) we see that

$$\begin{aligned} M_2(k)\tilde{\psi}_k &= \left\{a_{k_1}(M_2(k))_{11} + a_{k_2}(M_2(k))_{12}\right\}\phi_h^{k^{(1)}} + \left\{a_{k_1}(M_2(k))_{21} + a_{k_2}(M_2(k))_{22}\right\}\phi_h^{k^{(2)}} \\ &\coloneqq A_1\phi_h^{k^{(1)}} + A_2\phi_h^{k^{(2)}} \end{aligned}$$

For the i^{th} entry $(M_2(k)\tilde{\psi}_k)_i$ we have

$$\begin{aligned} \Rightarrow fl\left((M_2(k)\tilde{\psi}_k)_i\right) &= fl\left(\hat{A}_1(\phi_h^{k^{(1)}})_i + \hat{A}_2(\phi_h^{k^{(2)}})_i\right) \\ &= \left[A_1(\phi_h^{k^{(1)}})_i(1+\beta_{A_1})(1+\beta_1) + A_2(\phi_h^{k^{(2)}})_i(1+\beta_{A_2})(1+\beta_2)\right](1+\beta_3) \end{aligned} \tag{7.5}$$

where $|\beta_i| \leq \epsilon$ for $i = 1,2,3$ (for multiplication and addition) and we must determine bounds on $|\beta_{A_1}|$ and $|\beta_{A_2}|$, the errors accumulated in computing A_1 and A_2 respectively. Since $(M_2(k))_{11}, (M_2(k))_{22}$ and $(M_2(k))_{12}, (M_2(k))_{21}$ have analogous structures, we provide details only for bounding $|\beta_{A_1}|$, and conclude an analogous bound for $|\beta_{A_2}|$.

$$\begin{aligned} \hat{A}_1 &= fl(A_1) \\ &= \left(\left(a_{k_1}(M_2(k))_{11}\right)(1+\alpha_1) - \left(a_{k_2}(M_2(k))_{12}\right)(1+\alpha_2)\right)(1+\alpha_3). \end{aligned} \tag{7.6}$$

For α_1 we have:

$$\begin{aligned} fl(a_{k_1}(M_2(k))_{11}) &= fl\left(a_{k_1}\tilde{S}_h^{\nu}(k^{(1)})\left(1 - \frac{\lambda_h^{k^{(1)}}}{\lambda_{2h}^{k^{(1)}}}\tilde{J}_{2h}^h(k^{(1)})\tilde{J}_h^{2h}(k^{(1)})\right)\right) \\ &= a_{k_1}\tilde{S}_h^{\nu}(k^{(1)})\left(1 - \frac{\lambda_h^{k^{(1)}}}{\lambda_{2h}^{k^{(1)}}}\tilde{J}_{2h}^h(k^{(1)})\tilde{J}_h^{2h}(k^{(1)})(1+\delta_1)\right)(1+\delta_2) \end{aligned}$$

where $|\delta_1| \leq 3\epsilon$ and $|\delta_2| \leq (2+\nu)\epsilon$,

$$
\begin{aligned}
&= \left(a_{k_1} \left(M_2(k)\right)_{11} - a_{k_1} \tilde{S}_h^{\nu}(k^{(1)}) \frac{\lambda_h^{k^{(1)}}}{\lambda_{2h}^{k^{(1)}}} \tilde{J}_{2h}^h(k^{(1)}) \tilde{J}_h^{2h}(k^{(1)}) \delta_1 \right) (1+\delta_2) \\
&= a_{k_1} \left(M_2(k)\right)_{11} \left(1 + \frac{\frac{-\lambda_h^{k^{(1)}}}{\lambda_{2h}^{k^{(1)}}} \tilde{J}_{2h}^h(k^{(1)}) \tilde{J}_h^{2h}(k^{(1)})}{1 - \frac{\lambda_h^{k^{(1)}}}{\lambda_{2h}^{k1}} \tilde{J}_{2h}^h(k^{(1)}) \tilde{J}_h^{2h}(k^{(1)})} \delta_1 \right) (1+\delta_2) \\
&= a_{k_1} \left(M_2(k)\right)_{11} (1+\alpha_1)
\end{aligned}
$$

$$
\text{where } \ |\alpha_1| \leq \left(3 \left| \frac{\frac{-\lambda_h^{k^{(1)}}}{\lambda_{2h}^{k^{(1)}}} \tilde{J}_{2h}^h(k^{(1)}) \tilde{J}_h^{2h}(k^{(1)})}{1 - \frac{\lambda_h^{k^{(1)}}}{\lambda_{2h}^{k^{(1)}}} \tilde{J}_{2h}^h(k^{(1)}) \tilde{J}_h^{2h}(k^{(1)})} \right| + 2 + \nu \right) \epsilon := \left(3 \left| \eta_h^{k^{(1)}} \right| + 2 + \nu \right) \epsilon.
$$

For α_2 we have:

$$
\begin{aligned}
fl(a_{k_2}(M_2(k))_{12}) &= fl\left(a_{k_2} \tilde{S}_h^{\nu_1}(k^{(2)}) \tilde{S}_h^{\nu_2}(k^{(1)}) \frac{\lambda_h^{k^{(2)}}}{\lambda_{2h}^{k^{(1)}}} \tilde{J}_{2h}^h(k^{(1)}) \tilde{J}_h^{2h}(k^{(2)}) \right) \\
&= \left(a_{k_2} \tilde{S}_h^{\nu_1}(k^{(2)}) \tilde{S}_h^{\nu_2}(k^{(1)}) \frac{\lambda_h^{k^{(2)}}}{\lambda_{2h}^{k^{(1)}}} \tilde{J}_{2h}^h(k^{(1)}) \tilde{J}_h^{2h}(k^{(2)}) \right) (1+\delta_1) \\
&= a_{k_2} \left(M_2(k)\right)_{12} (1+\delta_1)
\end{aligned}
$$

where $|\delta_1| \leq (4+\nu)\epsilon$.

Therefore, going back to eq. (7.6) we have

$$
\hat{A}_1 = fl(A_1) = A_1(1+\alpha_{12})(1+\alpha_3)
$$

where $|\alpha_{12}| \leq \max\{|\alpha_1|, |\alpha_2|\}$, and $|\alpha_3| \leq \epsilon$ implies

$$
\hat{A}_1 = A_1(1+\beta_{A_1}) \ \text{ where } \ |\beta_{A_1}| \leq \left(3 \left| \eta_h^{k^{(1)}} \right| + 3 + \nu \right) \epsilon.
$$

Similarly, we have that

$$
\hat{A}_2 = A_2(1+\beta_{A_2}) \ \text{ where } \ |\beta_{A_2}| \leq \left(3 \left| \eta_h^{k^{(2)}} \right| + 3 + \nu \right) \epsilon.
$$

We can now use eq. (7.5) to conclude that

$$
\begin{aligned}
fl\left((M_2(k)\tilde{\psi}_k)_i \right) &= (M_2(k)\tilde{\psi}_k)_i (1+\delta_k^2) \\
\Rightarrow \| fl\left(M_2(k)\tilde{\psi}_k \right) - M_2(k)\tilde{\psi}_k \| &\leq \| M_2(k)\tilde{\psi}_k \| |\delta_k^2|
\end{aligned}
$$

where $|\delta_k^2| \leq \left(3 \max\left\{ \left| \eta_h^{k^{(1)}} \right|, \left| \eta_h^{k^{(2)}} \right| \right\} + 6 + \nu \right) \epsilon$.

Finally, the bound on $M_2\psi_h$ is determined from eq. (7.4) and summing over all $k \in \mathcal{N}$. We have,

$$\begin{aligned} fl\left((M_2\psi_h)_i\right) &= \sum_{\mathcal{N}} fl\left((M_2(k)\tilde{\psi}_k)_i\right)(1+\tilde{\delta}) = \sum_{\mathcal{N}}(M_2(k)\tilde{\psi}_k)_i(1+\delta_k^2)(1+\tilde{\delta}) \\ &= (M_2\psi_h)_i + \sum_{\mathcal{N}}(M_2(k)\tilde{\psi}_k)_i\tilde{\delta}_k^2 \end{aligned}$$

where $\quad |\tilde{\delta}| \leq (|\mathcal{N}|-1)\,\epsilon \;$ and $\; \tilde{\delta}_k^2 = \delta_k^2 + \tilde{\delta}$. This implies

$$\|fl\left(M_2\psi_h\right) - M_2\psi_h\| \leq \left\| \max_{k\in\mathcal{N}} \tilde{\delta}_k^2 \cdot \sum_{\mathcal{N}} M_2(k)\tilde{\psi}_k \right\| \leq \max_{k\in\mathcal{N}} |\tilde{\delta}_k^2| \|M_2\psi_h\|$$

$$\Rightarrow \frac{\|\text{fl}\left(M_2\psi_h\right) - M_2\psi_h\|}{\|M_2\psi_h\|} \leq \max_{k\in\mathcal{N}} \left|\tilde{\delta}_k^2\right|$$

where

$$\begin{aligned} |\tilde{\delta}_k^2| &\leq \left(|\mathcal{N}| - 2 + 6 + \nu + 3\max\left\{\left|\eta_h^{k^{(1)}}\right|, \left|\eta_h^{k^{(2)}}\right|\right\}\right)\epsilon \\ &= \left(|\mathcal{N}| + 4 + \nu + 3\max\left\{\left|\eta_h^{k^{(1)}}\right|, \left|\eta_h^{k^{(2)}}\right|\right\}\right)\epsilon. \end{aligned}$$

This yields the desired eq. (7.3). □

We note that the bound of eq. (7.3) is quite general, and may not be particularly useful in practice. Indeed, if $\eta_h^{\max}(k)$ is unbounded as $h \to 0$, then eq. (7.3) is of no use; in this case, one may be required to determine a tighter bound based on the entries of $M_2(k)$, as in Chapter 8. In the following section, we derive an analogous result on the 3-grid operator.

7.2 3-Grid Analysis

Analysing the error of the 3-grid iteration operator will yield insight into the general relationship between any l-grid operator with the 2-grid operator. Recall that the 3-grid operator is given by

$$\begin{aligned} M_h^{4h} &= S_h^{\nu_2}\left\{I_h - J_{2h}^h\left(I_{2h} - M_{2h}^{4h}\right)A_{2h}^{-1}J_h^{2h}A_h\right\}S_h^{\nu_1} \\ M_{2h}^{4h} &= S_{2h}^{\nu_2}\left(I_{2h} - J_{4h}^{2h}A_{4h}^{-1}J_{2h}^{4h}A_{2h}\right)S_{2h}^{\nu_1}, \end{aligned}$$

and, as before, we use the eigenmatrix representation $M_h^{4h}(k)$ of M_h^{4h} to derive a bound on the computed $M_h^{4h}\psi_h$ for any $\psi_h \in \mathbb{R}^N$.

Theorem 7.2. *Suppose that the 3-grid iteration operator $M_3 = M_h^{4h}$ can be represented as in Lemma 6.2, and assume that A_{4h}^{-1} exists explicitly, or can be solved for exactly. Let $k^{(1)} \in \mathcal{N}$ and $k^{(n)}$ for $n = 2, 3, 4$, define low and (associated) high frequency grid functions, respectively, on G_h. Let $\tilde{\psi}_k = \sum_{n=1}^4 a_{k_n}\phi_h^{k^{(n)}}$ for $a_{k_n} \in \mathbb{R}$ and let $M_3(k) = M_h^{4h}(k)$ for $k \in \mathcal{N}$. Then,*

$$\|fl\left(M_k\tilde{\psi}_k\right) - M_k\tilde{\psi}_k\| \leq \|M_k\tilde{\psi}_k\| \cdot |\delta_k^3|$$

where $\delta_k^3 = \max_n \delta_n + \delta_5$, $|\delta_5| \le 7\epsilon$ *and* $|\delta_n| \le (3 + \max\{\beta_n^1, \beta_n^2, \beta_n^3\})\,\epsilon$ *where*

$$
\begin{aligned}
&|\beta_n^3| \le 10 + 2\nu,\\
&|\beta_n^2| \le 8 + \nu + \left|\frac{(M_{2h}^{4h})_{mm}}{1 - (M_{2h}^{4h})_{mm}}\right| \left(3|\eta_{2h}^{k^{(m)}}| + 2 + \nu\right),\\
&|\beta_n^1| \le 2 + \nu + \left|\frac{\lambda_h^{k^{(n)}}/\lambda_{2h}^{k^{(m)}}\,\tilde{J}_{2h}^h(k^{(n)})\tilde{J}_h^{2h}(k^{(n)})\cdot(1 - 2(M_{2h}^{4h})_{mm})}{1 - \lambda_h^{k^{(n)}}/\lambda_{2h}^{k^{(m)}}\,\tilde{J}_{2h}^h(k^{(n)})\tilde{J}_h^{2h}(k^{(n)})\cdot(1 - (M_{2h}^{4h})_{mm})}\right| \left(3|\eta_{2h}^{k^{(m)}}| + 2 + \nu\right),
\end{aligned}
\tag{7.7}
$$

where η_{2h}^k *is as in eq.* (7.2), $n = 1, \dots, 4$, *and* $m = \begin{cases} 1 & n = 1, 2 \\ 3 & n = 3, 4 \end{cases}$.

Let us further assume that

$$
|\beta_n^2| \le 8 + \nu + c_n^2 \left(3|\eta_{2h}^{k^{(m)}}| + 2 + \nu\right) \quad \textit{and} \quad |\beta_n^3| \le 2 + \nu + c_n^3 \left(3|\eta_{2h}^{k^{(m)}}| + 2 + \nu\right)
\tag{7.8}
$$

where c_n^2 *and* c_n^3 *are some constants such that*

$$
|\delta_k^3| \le \left[18 + \nu + c\,(3\eta_{2h}^{\max}(k) + 2 + \nu)\right]\epsilon
$$

where $c = \max_n\{c_n^2, c_n^3\}$ *and* $\eta_{2h}^{\max}(k) = \max\left\{|\eta_{2h}^{k^{(1)}}|, |\eta_{2h}^{k^{(3)}}|\right\}$. *Then the bound on* $M_3\psi_h$ *is given by*

$$
\frac{\|fl(M_3\psi_h) - M_3\psi_h\|}{\|M_3\psi_h\|} \le \left(|\mathcal{N}| + 16 + \nu + c(2 + \nu) + 3 \cdot \max_k\left(\eta_{2h}^{\max}(k)\right)\right)\epsilon. \tag{7.9}
$$

Proof. Let M_3 and $M_3(k)$ satisfy the desired assumptions and be as in Lemma 6.2; and let $k^{(n)}$ for $n = 1 \dots 4$ be as in the problem statement and recall that $\bigcup_{\mathcal{N}} \left\{k^{(1)}, k^{(2)}, k^{(3)}, k^{(4)}\right\} = [N]$. Now, since A_h is diagonalizable, for any $\psi_h \in \mathbb{R}^N$ we can write

$$
\psi_h = \sum_{\mathcal{N}} \left(a_{k_1}\phi_h^{k^{(1)}} + a_{k_2}\phi_h^{k^{(2)}} + a_{k_3}\phi_h^{k^{(3)}} + a_{k_4}\phi_h^{k^{(4)}}\right)
$$

for some constants $a_{k_n} \in \mathbb{R}$, $n = 1, \dots, 4$. Then

$$
M_3\psi_h = \textstyle\sum_{\mathcal{N}} \left(a_{k_1}M_3\phi_h^{k^{(1)}} + a_{k_2}M_3\phi_h^{k^{(2)}} + a_{k_3}M_3\phi_h^{k^{(3)}} + a_{k_4}M_3\phi_h^{k^{(4)}}\right) = \textstyle\sum_{\mathcal{N}} M_3(k)\tilde{\psi}_k.
\tag{7.10}
$$

We look first entry-wise, for $i = 1, \dots, N$, at $fl\left((M_3(k)\tilde{\psi}_k)_i\right)$. From eq. (6.10) and (6.11) we can see that

$$
M_3(k)\tilde{\psi}_k = A_1\phi_h^{k(1)} + A_2\phi_h^{k(2)} + A_3\phi_h^{k(3)} + A_4\phi_h^{k(4)}
$$

where

$$
A_n = \left(a_{k_1}\left(M_3(k)\right)_{1n} + a_{k_2}\left(M_3(k)\right)_{2n} + a_{k_3}\left(M_3(k)\right)_{3n} + a_{k_4}\left(M_3(k)\right)_{4n}\right)\phi_h^{k^{(n)}}.
$$

Hence

$$fl\left((M_3(k)\tilde{\psi}_k)_i\right) = fl\left(\hat{A}_1(\phi_h^{k^{(1)}})_i + \hat{A}_2(\phi_h^{k^{(2)}})_i + \hat{A}_3(\phi_h^{k^{(3)}})_i + \hat{A}_4(\phi_h^{k^{(4)}})_i\right)$$
$$= \left[\sum_{n=1}^{4} A_n(\phi_h^{k^{(n)}})_i(1+\delta_n)\right](1+\delta_5) \tag{7.11}$$

where $|\delta_5| \leq 7\epsilon$, and we must determine bounds on $|\delta_n|$–the round off error in computing A_n. We provide details for $|\delta_1|$ and use this to draw conclusions about the others.

The computed coefficient A_1 satisfies

$$\hat{A}_1 = \left(\sum_{j=1}^{4} a_{k_j}\left(M_3(k)\right)_{j1}(1+\beta_1^j)\right)(1+\beta_5) \tag{7.12}$$

where $|\beta_5| \leq 3\epsilon$ and we must determine bounds on the remaining β_1^j.

For β_1^1 we have:

$$fl\left(a_{k_1}(M_3(k))_{11}\right) =$$
$$fl\left(a_{k_1}\tilde{S}_h^{\nu}(k^{(1)})\left(1 - \frac{\lambda_h^{k^{(1)}}}{\lambda_{2h}^{k^{(1)}}}\tilde{J}_{2h}^{h}(k^{(1)})\tilde{J}_h^{2h}(k^{(1)})\left[1 - fl\left((M_{2h}^{4h}(k))_{11}\right)\right]\right)\right)$$
$$= a_{k_1}\tilde{S}_h^{\nu}(k^{(1)})\left(1 - \frac{\lambda_h^{k^{(1)}}}{\lambda_{2h}^{k^{(1)}}}\tilde{J}_{2h}^{h}(k^{(1)})\tilde{J}_h^{2h}(k^{(1)})\left[1 - \left(M_{2h}^{4h}(k)\right)_{11}(1+\alpha_1)\right](1+\alpha_2)\right)(1+\alpha_3)$$
where $\quad |\alpha_1| \leq (3\eta_{2h}^{k^{(1)}} + 3 + \nu)\epsilon,\ |\alpha_2| \leq 5 \ \text{ and } \ |\alpha_3| \leq (2+\nu)\epsilon$
$$= a_{k_1}\tilde{S}_h^{\nu}(k^{(1)})\left(1 - \frac{\lambda_h^{k^{(1)}}}{\lambda_{2h}^{k^{(1)}}}\tilde{J}_{2h}^{h}(k^{(1)})\tilde{J}_h^{2h}(k^{(1)})\left[1 - \left(M_{2h}^{4h}(k)\right)_{11} + \left(1 - 2\left(M_{2h}^{4h}(k)\right)_{11}\right)\alpha_1\right]\right)(1+\alpha_3)$$
$$= a_{k_1}\left(M_3(k)\right)_{11}\left(1 + \frac{\lambda_h^{k^{(1)}}/\lambda_{2h}^{k^{(1)}}\tilde{J}_{2h}^{h}(k^{(1)})\tilde{J}_h^{2h}(k^{(1)})\left(2\left(M_{2h}^{4h}(k)\right)_{11} - 1\right)}{1 - \lambda_h^{k^{(1)}}/\lambda_{2h}^{k^{(1)}}\tilde{J}_{2h}^{h}(k^{(1)})\tilde{J}_h^{2h}(k^{(1)})\left(1 - \left(M_{2h}^{4h}(k)\right)_{11}\right)}\alpha_1\right)(1+\alpha_3)$$
$$= a_{k_1}\left(M_3(k)\right)_{11}\left(1+\beta_1^1\right),$$

where

$$|\beta_1^1| \leq \left(2 + \nu + \left|\frac{\lambda_h^{k^{(1)}}/\lambda_{2h}^{k^{(1)}}\tilde{J}_{2h}^{h}(k^{(1)})\tilde{J}_h^{2h}(k^{(1)})\cdot(1 - 2(M_{2h}^{4h})_{11})}{1 - \lambda_h^{k^{(1)}}/\lambda_{2h}^{k^{(1)}}\tilde{J}_{2h}^{h}(k^{(1)})\tilde{J}_h^{2h}(k^{(1)})\cdot(1 - (M_{2h}^{4h})_{11})}\right|\left(3|\eta_{2h}^{k^{(1)}}| + 2 + \nu\right)\right)\epsilon.$$

For β_1^2 we have:

$$
\begin{aligned}
&fl\left(a_{k_2}\left(M_3(k)\right)_{21}\right) = \\
&= fl\left(a_{k_2}\tilde{S}_h^{\nu_1}(k^{(2)})\tilde{S}_h^{\nu_2}(k^{(1)})\frac{-\lambda_h^{k^{(1)}}}{\lambda_{2h}^{k^{(1)}}}\tilde{J}_{2h}^{h}(k^{(2)})\tilde{J}_h^{2h}(k^{(1)})\left[1 - fl\left(M_{2h}^{4h}(k)\right)_{11}\right]\right) \\
&= \left(a_{k_2}\tilde{S}_h^{\nu_1}(k^{(2)})\tilde{S}_h^{\nu_2}(k^{(1)})\frac{-\lambda_h^{k^{(1)}}}{\lambda_{2h}^{k^{(1)}}}\tilde{J}_{2h}^{h}(k^{(2)})\tilde{J}_h^{2h}(k^{(1)})\left[1 - \left(M_{2h}^{4h}(k)\right)_{11}(1+\alpha_1)\right]\right)(1+\alpha_2) \\
&\text{where} \quad |\alpha_1| \le (3\eta_{2h}^{k^{(1)}} + 3 + \nu)\epsilon \text{ and } |\alpha_2| \le (8+\nu)\epsilon \\
&= a_{k_2}\left(M_3(k)\right)_{21}\left(1 + \frac{\left(M_{2h}^{4h}(k)\right)}{1 - \left(M_{2h}^{4h}(k)\right)}\alpha_1\right)(1+\alpha_2) \\
&= a_{k_2}\left(M_3(k)\right)_{21}(1+\beta_1^2)
\end{aligned}
$$

where

$$\left|\beta_1^2\right| \le \left(8 + \nu + \left|\frac{(M_{2h}^{4h})_{11}}{1 - (M_{2h}^{4h})_{11}}\right|\left(3|\eta_{2h}^{k^{(1)}}| + 2 + \nu\right)\right)\epsilon.$$

And β_1^3 we have:

$$
\begin{aligned}
&fl\left(a_{k_3}\left(M_3(k)\right)_{31}\right) = \\
&fl\left(a_{k_3}\tilde{S}_h^{\nu_1}(k^{(3)})\tilde{S}_h^{\nu_2}(k^{(1)})fl\left(M_{2h}^{4h}(k)\right)_{21}\right) \\
&= \left(a_{k_3}\tilde{S}_h^{\nu_1}(k^{(3)})\tilde{S}_h^{\nu_2}(k^{(1)})\left(M_{2h}^{4h}(k)\right)_{21}(1+\alpha_1)\right)(1+\alpha_2) \\
&\text{where} \quad |\alpha_1| \le (4+\nu)\epsilon \text{ and } |\alpha_2| \le (6+\nu)\epsilon \\
&= a_{k_3}\left(M_3(k)\right)_{31}\left(1+\beta_1^3\right)
\end{aligned}
$$

where

$$\left|\beta_1^3\right| \le (10+\nu)\epsilon.$$

Finally, since $\left(M_3(k)\right)_{41}$ has the same structure at $\left(M_3(k)\right)_{31}$, β_1^4 also satisfies

$$\left|\beta_1^4\right| \le (10+\nu)\epsilon.$$

Now, returning to eq. (7.12) we can see that

$$\hat{A}_1 = A_1(1+\delta_1)$$

where $|\delta_1| \le 3\epsilon + \max\{\beta_1^1, \beta_1^2, \beta_1^3\}$.

Through similar calculations we can determine analogous bounds on $\beta_n^1, \beta_n^2, \beta_n^3$, and hence on δ_n for $n = 2, \ldots, 4$. Using eq. (7.11) we can conclude the desired:

$$
\begin{aligned}
&fl\left((M_3(k)\tilde{\psi}_k)_i\right) = (M_3(k)\tilde{\psi}_k)_i(1+\delta_k^3) \\
&\Rightarrow \|fl\left(M_3(k)\tilde{\psi}_k\right) - M_3(k)\tilde{\psi}_k\| \le \|M_3(k)\tilde{\psi}_k\| \cdot |\delta_k^3| \\
&\text{where} \quad |\delta_k^3| \le 7\epsilon + \max_n \delta_n.
\end{aligned}
$$

Finally, suppose that the β_n^j are bounded as in eq. (7.8) such that

$$|\delta_k^3| \leq [18 + \nu + c\,(3\eta_{2h}^{\max}(k) + 2 + \nu)]\,\epsilon$$

where c, and $\eta_{2h}^{\max}(k)$ are as in the theorem statement. Then the bound on $M_3\psi_h$ is determined from eq. (7.10) and summing over all $k \in \mathcal{N}$. As before, entry-wise, for $i = 1, \ldots, N$, we have

$$\begin{aligned} fl\,((M_3\psi_h)_i) &= \sum_{\mathcal{N}} fl\left((M_3(k)\tilde{\psi}_k)_i\right)(1+\tilde{\delta}) = \sum_{\mathcal{N}} (M_3(k)\tilde{\psi}_k)_i(1+\delta_k^3)(1+\tilde{\delta}) \\ &= (M_3\psi_h)_i + \sum_{\mathcal{N}} (M_3(k)\tilde{\psi}_k)_i\tilde{\delta_k^3} \end{aligned}$$

where $\quad |\tilde{\delta}| \leq (|\mathcal{N}| - 1)\,\epsilon \quad$ and $\quad \tilde{\delta_k^3} = \delta_k^3 + \tilde{\delta}.$ This implies

$$\|fl\,(M_3\psi_h) - M_3\psi_h\| \leq \left\| \max_{k\in\mathcal{N}} \tilde{\delta_k^3} \cdot \sum_{\mathcal{N}} M_3(k)\tilde{\psi}_k \right\| \leq \max_{k\in\mathcal{N}} \left|\tilde{\delta_k^3}\right| \|M_3\psi_h\|$$

$$\Rightarrow \frac{\|\mathrm{fl}\,(M_3\psi_h) - M_3\psi_h\|}{\|M_3\psi_h\|} \leq \max_{k\in\mathcal{N}} \left|\tilde{\delta_k^3}\right|$$

where

$$|\tilde{\delta_k^3}| \leq (|\mathcal{N}| + 16 + \nu + c\,(3\eta_{2h}^{\max}(k) + 2 + \nu))\,\epsilon.$$

This yields the desired eq. (7.9).

□

We note here, as before, that the bound of Theorem 7.2 depends on both the β_n^j and $\eta_h^{\max}(k)$ being bounded for $h \to 0$. But together with Theorem 7.1 these general statements are useful in the following sense. We can see from eq. (7.7) that in going from employing a 2-grid operator to a 3-grid operator, the floating point error bound picks up error terms recursively. In particular, the bound of the β_n^1 term looks *almost* like a 3-grid version of $\eta_h^{k^{(n)}}$ from Theorem 7.1. And while we pick up some new modest constants in the 3-grid case of Theorem 7.2, it seems that the bulk of the accumulation comes from an $\eta_h^{k^{(n)}}$ term. What's more, should everything be bounded nicely as in Theorem 7.2, this suggests that all we really need to know is how a general 2-grid operator M_l^{l+1} accumulates error. We explore this idea further in the Section 7.3 and make generalizations of these results.

7.3 l-Grid Analysis

To understand the roundoff errors from an l-grid operator M_l, recall from Lemma 6.3 that M_l can be written recursively in terms of 2-grid operators. We use this to bound the relative error of the computed $M_l\psi_h$ for any $\psi_h \in \mathbb{R}^N$.

Theorem 7.3. *Let $\psi_h \in \mathbb{R}^N$ with $\|\psi_h\| \leq C$ and let M_l be the the l-grid multigrid operator as defined in Equation* (6.5)*. Suppose that $M_l = M_1^l = M_{l-1}^l + B_l^{l-1}M_{l-1}B_{l-1}^l$*

is such that $\|M_{l-1}^l\| \le C_1$ *and* $\|B_l^{l-1}\| \cdot \|B_{l-1}^l\| \le C_2$. *Let* δ_2 *be the error bound of the 2-grid operator as in Theorem 7.1. Namely, suppose that* $\|fl\left(M_{l-1}^l\psi_h\right) - M_{l-1}^l\psi_h\| \le \|M_{l-1}^l\psi_h\| \cdot |\delta_2|$ *where*

$$|\delta_2| \le \left(4 + \nu + |\mathcal{N}| + 3\max_{k\in N}\eta_h^{\max}(k)\right)\epsilon. \tag{7.13}$$

Then

$$\frac{\|fl\left(M_l\psi_h\right) - M_l\psi_h\|}{\|M_l\psi_h\|} \le |\delta_l|$$

where

$$|\delta_l| \le (l-1)|\delta_2| + (l-2)\epsilon - \frac{CC_1}{(1-C_2)^2}\left((1-C_2)^l - C_2^2 + 2C_2 - 1\right)\epsilon.$$

Proof. Let δ_2 be such that entry-wise, for $i = 1\ldots, N$, we have $fl\left((M_{l-1}^l\psi_h)_i\right) = (M_{l-1}^l\psi_h)_i(1+\delta_2)$ and $|\delta_2|$ satisfies eq. (7.13). Then for $M_l = M_1^l = M_{l-1}^l + B_l^{l-1}M_{l-1}B_{l-1}^l$ we have

$$\begin{aligned} fl\left((M_l\psi_h)_i\right) &= \left(fl\left((M_{l-1}^l\psi_h)_i\right) + fl\left((B_l^{l-1}M_{l-1}B_{l-1}^l\psi_h)_i\right)\right)(1+\alpha_1) \\ &= \left((M_{l-1}^l\psi_h)_i(1+\delta_2) + (B_l^{l-1}M_{l-1}B_{l-1}^l\psi_h)_i(1+\alpha_2^l)\right)(1+\alpha_1) \\ &= (M_l\psi_h)_i(1+\delta_l) \end{aligned}$$

where $\delta_l = \delta_2 + \alpha_2^l + \alpha_1$ with

$$|\alpha_1| \le \epsilon \quad \text{and}$$

$$|\alpha_2^l| \le \|B_l^{l-1}\|\|M_{l-1}\|\|B_{l-1}^l\|\|\psi_h\|\epsilon + \delta_{l-1} \le CC_2 \cdot \frac{C_1(1-C_2^{l-2})}{(1-C_2)}\epsilon + \delta_{l-1}.$$

Analogous to δ_l, δ_{l-1} satisfies $fl\left((M_{l-1}\psi_h)_i\right) = (M_{l-1}\psi_h)_i\left(1+\delta_{l-1}\right)$ where $\delta_{l-1} = \delta_2 + \alpha_2^{l-1} + \alpha_1$, etc. Continuing this recursion we have

$$\begin{aligned} |\delta_l| &\le |\delta_2| + \epsilon + CC_2 \cdot \frac{C_1(1-C_2^{l-2})}{(1-C_2)}\epsilon + \delta_{l-1} \\ &\le |\delta_2| + \epsilon + CC_2 \cdot \frac{C_1(1-C_2^{l-2})}{(1-C_2)}\epsilon + |\delta_2| + \epsilon + CC_2 \cdot \frac{C_1(1-C_2^{l-3})}{(1-C_2)}\epsilon + \delta_{l-3} \\ &\le \ldots \le (l-2)(|\delta_2| + \epsilon) + \frac{CC_2C_1}{1-C_2}\sum_{j=2}^{l-1}(1-C_2)^{l-j} + |\delta_2| \\ &= (l-1)|\delta_2| + (l-2)\epsilon + CC_2C_1(1-C_2)^{l-1}\sum_{j=2}^{l-1}(1-C_2)^{-j}\epsilon \\ &= (l-1)|\delta_2| + (l-2)\epsilon + CC_2C_1(1-C_2)^{l-1} \cdot \frac{-\left((1-C_2)^l - C_2^2 + 2C_2 - 1\right)}{C_2(1-C_2)^{l+1}}\epsilon \\ &= (l-1)|\delta_2| + (l-2)\epsilon - \frac{CC_1}{(1-C_2)^2}\left((1-C_2)^l - C_2^2 + 2C_2 - 1\right)\epsilon. \end{aligned}$$

□

In Chapter 8 we apply the ideas from this chapter to the Model Problem. We demonstrate first how to employ Theorem 7.1 and Theorem 7.2 to derive a meaningful error bound on the computed $M_2\psi_h$ and $M_3\psi_h$ in Section 8.1 and Section 8.2 respectively. Finally, in Section 8.3 we discuss the general case of using an l-grid operator, and how these results might be able to be interpreted in terms of the motivating Example 5.1. Finally, we discuss future directions for this work.

Chapter 8 Application to Model Problem

For clarity, we apply Theorem 7.1 and Theorem 7.2 to the Model Problem using the Model Algorithm and explicitly calculate error bounds. For convenience, we restate both.

We consider Poisson's equation with Dirichlet boundary conditions

$$\begin{cases} -\Delta v = f \text{ in } \Omega \\ v = 0 \text{ on } \partial\Omega, \end{cases} \tag{8.1}$$

with $\Omega = [0, 1]$. Is discretized using the center difference to obtain the matrix equation

$$A_h \mathbf{v}_h = \mathbf{f}_h$$

where $A_h = \frac{1}{h^2} T_N$ with $h = \frac{1}{N}$ and T_N is the tridagonal matrix with 2's on the diagonal and -1 on the super and sub diagonal. A_h has eigenvalues and (orthonormal) eigenvectors, respectively:

$$\lambda_h^k = \frac{4}{h^2} \sin^2\left(\frac{k\pi h}{2}\right) \quad \text{for} \quad k = 1 \dots N,$$
$$\phi_h^k(x) = \sqrt{2h} \sin(k\pi x) \quad \text{for} \quad k = 1, \dots, N \quad \text{and} \quad x \in G_h.$$

Moreover, recall our Model Algorithm specifies J_h^{2h} as full-weight restriction, J_{2h}^h as linear interpolation, $S_h = I_h - \frac{\omega h^2}{2} A_h$ with $\omega = 2/3$, and $\nu_1 = 2$ and $\nu_2 = 1$. See Example (6.1) for details. In the following sections, we will show explicitly that these operators define multigrid iteration operators M_h^{2h} and M_h^{4h} that are invariant under the spaces $\mathcal{E}_h^{2h}$ and $\mathcal{E}_h^{4h}$, respectively.

In Section 8.1 we look at a single iteration of a 2-grid V-cycle and provide a floating point error bound on matrix-vector multiplication $M_h^{2h}\psi$ for any $\psi \in \mathbb{R}^N$. Likewise, in Section 8.2, we look at a single iteration of a 3-grid V-cycle and provide a floating point error bound on the matrix-vector multiplication $M_h^{4h}\psi$. Finally, in Section 8.3 we look at the general l-grid operator.

8.1 Model Problem 2-Grid Analysis

Consider solving the Model Problem using a single iteration of a 2-grid V-cycle using the Model Algorithm, and assume that we can solve the residual equation exactly on G_{2h}. Under the assumption that the multigrid algorithm converges (which the Model Algorithm does) we need that the computed multigrid solution converges to the multigrid solution in exact arithmetic. That is, we look at $\left\| fl\left(M_h^{2h}\psi_h\right) - M_h^{2h}\psi_h \right\|$ for any $\psi_h \in \mathbb{R}^N$.

We first need to define $\mathcal{E}_h^{2h}$, the space of $h - 2h$ harmonics for the model problem. We then walk through the assumptions of Lemma 6.1 and calculate the entries of $M_h^{2h}(k)$.

Space of Harmonics

For the Model Problem, the low-frequency grid functions are those for which $k < \frac{N}{2}$, and the high-frequency components are those for which $k \geq \frac{N}{2}$. Therefore, in Definition 6.2 we take $\mathcal{N} = \{1, \ldots, \frac{N}{2} - 1\}$, and for any $k = k^{(1)} \in \mathcal{N}$, $k^{(2)} = N - k$. Then,

$$\mathcal{E}_h^{2h} = \text{span}\left\{\phi_h^k, -\phi_h^{N-k}\right\} \quad \text{for} \quad k = 1, \ldots, \frac{N}{2} - 1.$$

Indeed, we can see here that for $x \in G_{2h}$ we have $x = x_j = jh$ for $j = 1, \ldots, N$ and j even, and therefore,

$$\begin{aligned} -\phi_h^{N-k} &= -\sin\left((N-k)\pi x\right) \\ &= -\left(\sin(N\pi x)\cos(k\pi x) - \cos(N\pi x)\sin(k\pi x)\right) \\ &= -\left(\sin(j\pi)\cos(k\pi x) - \cos(j\pi)\sin(k\pi x)\right) \\ &= \sin(k\pi x) \\ &= \phi_h^k(x). \end{aligned}$$

Transfer Operators

The transfer operators from the Model Algorithm satisfy

$$\begin{aligned} J_h^{2h} &: \mathcal{E}_h^{2h} \to \text{span}\left\{\phi_{2h}^k\right\} \\ J_{2h}^h &: \text{span}\left\{\phi_{2h}^k\right\} \to \mathcal{E}_h^{2h}, \end{aligned}$$

which we can see by using the details from Example 6.1 and explicitly calculating $\tilde{J}_h^{2h}$ and $\tilde{J}_{2h}^h$:

$$\begin{aligned} J_h^{2h}\phi_h^k(x) &= \frac{1}{4}\sqrt{2h}\left(\phi_h^k(x-h) + 2\phi_h^k(x) + \phi_h^k(x+h)\right) \\ &= \frac{\sqrt{2h}}{4}[\sin(k\pi x)\cos(k\pi h) - \cos(k\pi x)\sin(k\pi h) + 2\sin(k\pi x) \\ &\quad + \sin(k\pi x)\cos(k\pi h) + \cos(k\pi x)\sin(k\pi h)] \\ &= \frac{1}{2}\sqrt{2h}(\cos(k\pi h) + 1)\sin(k\pi x) = \frac{\sqrt{2}}{2}\sqrt{4h}\frac{1}{2}(\cos(k\pi h) + 1)\sin(k\pi x) \\ &= \frac{\sqrt{2}}{2}\cos^2\left(\frac{k\pi h}{2}\right)\phi_{2h}^k(x) \\ &\Rightarrow \tilde{J}_h^{2h}(k) = \frac{\sqrt{2}}{2}\cos^2\left(\frac{k\pi h}{2}\right), \text{ and} \\ &\quad \tilde{J}_h^{2h}(N-k) = \frac{\sqrt{2}}{2}\cos^2\left(\frac{(N-k)\pi h}{2}\right) = \frac{\sqrt{2}}{2}\sin^2\left(\frac{k\pi h}{2}\right). \end{aligned}$$

Similarly,

$$J_{2h}^h \phi_{2h}^k(x) = \begin{cases} \phi_{2h}^k(x) & \text{if} \quad x \in G_{2h} \\ \frac{1}{2}\left(\phi_{2h}^k(x-h) + \phi_{2h}^k(x+h)\right) & \text{if} \quad x \in G_h \setminus G_{2h} \end{cases}$$
$$= a\phi_h^k(x) + b(-\phi_h^{N-k}(x)).$$

If $x \in G_{2h}$, then $\phi_{2h}^k(x) = a\phi_h^k(x) + b(-\phi_h^{N-k}(x))$ yields $\sqrt{2} = a + b$ since $\phi_h^k = -\phi_h^{N-k}$ on G_{2h}. If $x \in G_h \setminus G_{2h}$ then we have:

$$\frac{1}{2}\phi_{2h}^k(x-h) + \frac{1}{2}\phi_{2h}^k(x+h) = a\phi_h^k(x) + b(-\phi_h^{N-k}(x)) \Rightarrow$$
$$\sqrt{4h}\cos(k\pi h)\sin(k\pi x) = (\sqrt{2}-b)\sqrt{2h}\sin(k\pi x) - b\sqrt{2h}\sin((N-k)\pi x)$$
$$= (\sqrt{2}-b)\sqrt{2h}\sin(k\pi x) - b\sqrt{2h}\left[\sin(j\pi)\cos(k\pi x) - \cos(j\pi)\sin(k\pi x)\right]$$
$$\text{for } j \text{ odd}$$
$$= (\sqrt{2}-2b)\sqrt{2h}\sin(k\pi x)$$
$$\Rightarrow b = \sqrt{2}\cdot\frac{1}{2}(1-\cos(k\pi h)) = \sqrt{2}\sin^2\left(\frac{k\pi h}{2}\right) \text{ and therefore}$$
$$a = \sqrt{2}\cos^2\left(\frac{k\pi h}{2}\right)$$
$$\Rightarrow \tilde{J}_{2h}^h(k) = \sqrt{2}\cos^2\left(\frac{k\pi h}{2}\right).$$

Smoothing Operator

It follows directly from $S_h = I_h - \frac{\omega}{2}T_N$ that $S_h : \mathcal{E}_h^{2h} \to \mathcal{E}_h^{2h}$ with

$$\tilde{S}_h(k) = 1 - \frac{4}{3}\sin^2\left(\frac{k\pi h}{2}\right).$$

Now, in light of Theorem 7.1, we have the following

Theorem 8.1. *For any $\psi_h \in \mathbb{R}^N$ and $k = 1, \ldots, \frac{N}{2} - 1$, let $\tilde{\psi}_k = \begin{bmatrix} a_k & b_k \end{bmatrix} \psi_h^k$ where $a_k, b_k \in \mathbb{R}$ and $\psi_h^k(x) = \begin{bmatrix} \phi_h^k(x) & -\phi_h^{N-k}(x) \end{bmatrix}^T$; further, let $M_2 = M_h^{2h}$ and $M_2(k) = M_h^{2h}(k)$. Then we have*

$$\|fl\left(M_2(k)\tilde{\psi}_k\right) - M_2(k)\tilde{\psi}_k\| \leq \|M_2(k)\tilde{\psi}_k\|\left(6 + \nu + 3\eta_h^k\right)\epsilon \tag{8.2}$$

where $\eta_h^k = \cot^2\left(\frac{k\pi h}{2}\right)$.

Moreover, ψ_h satisfies

$$\|fl(M_2\psi_h) - M_2\psi_h\| \leq C\|M_2\|p_2(N)\epsilon,$$

where $p_2(N)$ is a degree 2 polynomial in $\frac{N}{2}$, $\|M_2\| \leq 0.360559$, and C is a constant depending on coefficients a_k and b_k.

Proof. In light of Theorem 7.1 and eq. (7.2), we first calculate η_h^k and η_h^{N-k} to prove eq. (8.2). For η_h^k, note that

$$\begin{aligned}
\lambda_h^{k^{(1)}}/\lambda_{2h}^{k^{(1)}}\ \tilde{J}_{2h}^h(k^{(1)})\tilde{J}_h^{2h}(k^{(1)}) &= \frac{4\sin^2(k\pi h/2)}{\sin^2(k\pi h)}\cos^4(k\pi h/2) \\
&= \frac{4\sin^2(k\pi h/2)\cos^4(k\pi h/2)}{\left(2\sin(k\pi h/2)\cos(k\pi h/2)\right)^2} \\
&= \cos^2(k\pi h/2) \\
\Rightarrow \eta_h^k = \frac{\cos^2(k\pi h/2)}{1-\cos^2(k\pi h/2)} &= \cot^2\left(\frac{k\pi h}{2}\right).
\end{aligned}$$

Similarly, for η_h^{N-k} we have

$$\begin{aligned}
\lambda_h^{k^{(2)}}/\lambda_{2h}^{k^{(1)}}\ \tilde{J}_{2h}^h(k^{(2)})\tilde{J}_h^{2h}(k^{(2)}) &= \frac{4\cos^2(k\pi h/2)}{\sin^2(k\pi h)}\sin^4(k\pi h/2) \\
&= \frac{4\cos^2(k\pi h/2)\sin^4(k\pi h/2)}{\left(2\sin(k\pi h/2)\cos(k\pi h/2)\right)^2} \\
&= \sin^2(k\pi h/2) \\
\Rightarrow \eta_h^{N-k} = \frac{\sin^2(k\pi h/2)}{1-\sin^2(k\pi h/2)} &= \tan^2\left(\frac{k\pi h}{2}\right).
\end{aligned}$$

Thus, by Theorem 7.1, and since $\cot^2(k\pi h/2) \geq \tan^2(k\pi h/2)$ for $k = 1, \ldots, \frac{N}{2}-1$, we have

$$\left|\delta_k^2\right| \leq \left(3\eta_h^{\max}(k) + 6 + \nu\right)\epsilon = \left(3\eta_h^k + 6 + \nu\right)\epsilon = \left(3\cot^2\left(\tfrac{k\pi h}{2}\right) + 6 + \nu\right)\epsilon$$

for $k = 1, \ldots, \dfrac{N}{2} - 1$.

Furthermore, since $\cot(k\pi h/2)$ is unbounded as $h \to 0$, it is of no practical use for us to consider $\max_k \cot^2(k\pi h/2)$ as in Theorem 7.1. We take a slightly different apporach in the following.

$$\begin{aligned}
\|fl\left(M_2\psi_h\right) - M_2\psi_h\| &\leq \left\| \sum_{k=1}^{N/2-1} M_2\tilde{\psi}_k\tilde{\delta}_k^2 \right\| \quad \text{where} \quad \left|\tilde{\delta}_k^2\right| \leq (N/2-1)\epsilon + \left|\delta_k^2\right| \\
&\leq \sum_{k=1}^{N/2-1} \|M_2\|\|\begin{bmatrix} a_k & b_k \end{bmatrix}\|\|\psi_h^k\| \left|\tilde{\delta}_k^2\right| \leq \|M_2\| C \sum_{k=1}^{N/2-1} \left|\tilde{\delta}_k^2\right|,
\end{aligned}$$

where $C = \max_k \{a_k, b_k\}$, and we are left to determine a bound on $\|M_2\|$ and $\sum \left|\tilde{\delta}_k^2\right|$. We address the latter first.

$$
\begin{aligned}
&\sum_{k=1}^{N/2-1} \left|\tilde{\delta}_k^2\right| \leq \sum_{k=1}^{N/2-1} \left(\frac{N}{2} + 5 + \nu + 3\cot^2\left(\frac{k\pi h}{2}\right)\right)\epsilon \\
&= \left(\frac{N}{2} + 5 + \nu\right)\left(\frac{N}{2} - 1\right)\epsilon + 3\sum_{k=1}^{N/2-1} \cot^2\left(\frac{k\pi h}{2}\right)\epsilon \\
&= \left(\left(\frac{N}{2}\right)^2 + (4+\nu)\frac{N}{2} - (5+\nu) + 3\sum_{k=1}^{N/2-1} \left(\csc(k\pi h) + \cot(k\pi h)\right)^2\right)\epsilon \\
&\leq \left(\left(\frac{N}{2}\right)^2 + (4+\nu)\frac{N}{2} - (5+\nu) + 3\sum_{k=1}^{N/2-1} 3\csc^2(k\pi h) + \cot^2(k\pi h)\right)\epsilon \\
&= \left(\left(\frac{N}{2}\right)^2 + (4+\nu)\frac{N}{2} - (5+\nu) + 9\sum_{k=1}^{\lfloor\frac{N-1}{2}\rfloor} \csc^2\left(\frac{k\pi}{N}\right) + 3\sum_{k=1}^{\lfloor\frac{N-1}{2}\rfloor} \cot^2\left(\frac{k\pi}{N}\right)\right)\epsilon \\
&= \left(\left(\frac{N}{2}\right)^2 + (4+\nu)\frac{N}{2} - (5+\nu) + 9\cdot\frac{1}{12}(2N^2 - 3(-1)^N - 5) + 3\cdot\frac{1}{6}(N-1)(N-2)\right)\epsilon \\
&= \left(9\left(\frac{N}{2}\right)^2 + (1+\nu)\frac{N}{2} + (-10+\nu)\right)\epsilon \\
&= p_2(N)\epsilon.
\end{aligned}
$$

Now, recall from Chapter 7 that if Q is a matrix of eigenvectors ϕ_h^k, ordered by the pairs ϕ_h^k and ϕ_h^{N-k}, then $QM_2Q^{-1} = \text{diag}\,(M_2(k))$ is a block diagonal matrix where the k^{th} block is given by

$$
M_2(k) = \begin{bmatrix} 1 - \frac{4}{3}\sin^2\left(\frac{k\pi h}{2}\right) & \\ & 1 - \frac{4}{3}\cos^2\left(\frac{k\pi h}{2}\right) \end{bmatrix} \begin{bmatrix} \sin^2\left(\frac{k\pi h}{2}\right) & -\cos^2\left(\frac{k\pi h}{2}\right) \\ -\sin^2\left(\frac{k\pi h}{2}\right) & \cos^2\left(\frac{k\pi h}{2}\right) \end{bmatrix} \begin{bmatrix} 1 - \frac{4}{3}\sin^2\left(\frac{k\pi h}{2}\right) & \\ & 1 - \frac{4}{3}\cos^2\left(\frac{k\pi h}{2}\right) \end{bmatrix}^2. \tag{8.3}
$$

Therefore, $\|M_2\|_2^2 = \max_{k=1,\ldots N/2-1} \left\{\mu_{\max}\left(M_2(k)^T M_2(k)\right)\right\}$, is the maximum over all k of the largest eigenvalue of $M_2(k)^T M_2(k)$. $M_2(k)^T M_2(k)$ is a 4×4 matrix whose entries can be computed by eq. (8.3). Specifically, $M_2(k)^T M_2(k)$ has characteristic polynomial

$$
\mu\left[\mu - \left(\tilde{S}_h^{2\nu_1}(k)\sin^4(k\pi h/2) - \tilde{S}_h^{\nu_1}(N-k)\cos^4(k\pi h/2)\right)\left(\tilde{S}_h^{2\nu_2}(k) + \tilde{S}_h^{2\nu_2}(N-k)\right)\right] = 0
$$

which implies $\|M_2\|_2^2 =$

$$
\begin{aligned}
&\max_k\{\left((1 - \tfrac{4}{3}\sin^2(k\pi h/2))^{2\nu_1}\sin^4(k\pi h/2) - (1 - \tfrac{4}{3}\cos^2(k\pi h/2))^{\nu_1}\cos^4(k\pi h/2)\right)\cdot \\
&\qquad\qquad \cdot\left((1 - \tfrac{4}{3}\sin^2(k\pi h/2))^{2\nu_2} + (1 - \tfrac{4}{3}\cos^2(k\pi h/2))^{2\nu_2}\right)\} \\
&= \max_k\left\{\frac{-37}{324}\cos(k\pi h) - \frac{64}{324}\cos(3k\pi h) - \frac{7}{108}\cos(5k\pi h)\right\} \\
&\leq 0.130003.
\end{aligned}
$$

The final inequality is determined by maximizing the continuous function $\frac{-37}{324}\cos(\theta) - \frac{64}{324}\cos(3\theta) - \frac{7}{108}\cos(5\theta)$ on $\left(0, \frac{\pi}{2}\right)$. And we thus conclude

$$\|M_2\| \leq \sqrt{0.130003} \approx 0.360559.$$

□

8.2 Model Problem 3-Grid Analysis

Consider solving the Model Problem using a single iteration a 3-grid V-cycle using the Model Algorithm, and assume that we can solve the residual equation exactly on grid G_{4h}. As before, we first define the low- and high-frequency grid functions (and hence $\mathcal{E}_h^{4h}$), then use this to show that $M_h^{4h} : \mathcal{E}_h^{4h} \to \mathcal{E}_h^{4h}$, and calculate the entries of $M_h^{4h}(k)$.

Space of Harmonics

For the 3-grid iteration matrix on the Model Problem we have $\mathcal{N} = \{1, \ldots, \frac{N}{4} - 1\}$ and we take $k = k^{(1)} \in \mathcal{N}$. Specifically, the low-frequency grid functions ϕ_h^k are those for which $k < \dfrac{N}{4}$, and the high-frequency grid functions $\phi_h^{k^{(2)}}$, $\phi_h^{k^{(3)}}$, and $\phi_h^{k^{(4)}}$ are defined by

$$k^{(2)} = N - k, \;\; k^{(3)} = \frac{N}{2} - k, \text{ and } k^{(4)} = \frac{N}{2} + k \text{ for } k = 1, \ldots, \frac{N}{4} - 1.$$

We thus can define

$$\mathcal{E}_h^{4h} = \text{span}\left\{\phi_h^k, -\phi_h^{N-k}, -\phi_h^{N/2-k}, \phi_h^{N/2+k}\right\} \text{ for } k = 1 \ldots, \frac{N}{4} - 1.$$

Where, indeed, for $x \in G_{4h}$ we have

$$\phi_h^k(x) = -\phi_h^{N-k}(x) = -\phi_h^{N/2-k}(x) = \phi_h^{N/2+k}(x)$$

.

We note that M_h^{4h} is composed of operators that are defined on (or translate between) pairs of successive grids: G_h and G_{2h} or G_{2h} and G_{4h}. That is, the operators behave analogously to those in Section 8.1.

Transfer Operators

We note that on G_{2h}, $\phi_h^k = -\phi_h^{N-k}$ and $\phi_h^{N/2+k} = -\phi_h^{N/2+k}$. Therefore, the $h - 2h$ transfer operators take vectors to and from $\mathcal{E}_h^{4h}$ and $\mathcal{E}_{2h}^{4h} = \text{span}\left\{\phi_{2h}^k, -\phi_{2h}^{N/2-k}\right\}$. Specifically we have, as before,

$$J_h^{2h} : \mathcal{E}_h^{4h} \to \mathcal{E}_h^{4h} \text{ with } \tilde{J}_h^{2h}(k) = \frac{\sqrt{2}}{2}\cos^2(k\pi h/2).$$
$$J_{2h}^{h} : \mathcal{E}_h^{4h} \to \mathcal{E}_h^{4h} \text{ with } \tilde{J}_{2h}^{h}(k) = \sqrt{2}\cos^2(k\pi h/2).$$

Smoothing Operator

The weighted Jacobi operator on the Model Problem is defined as $S_h = I_h - \frac{\omega}{2} T_N$ hence

$$S_h : \mathcal{E}_h^{4h} \to \mathcal{E}_h^{4h} \text{ with } \tilde{S}_h(k) = 1 - \frac{4}{3}\sin^2\left(\frac{k\pi h}{2}\right).$$

M_{2h}^{4h} Operators

Similarly, for the $2h - 4h$ operators we have:

$$\begin{aligned}
&A_{2h} : \mathcal{E}_{2h}^{4h} \to \mathcal{E}_{2h}^{4h} \text{ with } \lambda_{2h}^k = \frac{1}{h^2}\sin^2(k\pi h) \\
&S_{2h} : \mathcal{E}_{2h}^{4h} \to \mathcal{E}_{2h}^{4h} \text{ with } \tilde{S}_{2h}(k) = 1 - \frac{\omega}{2h^2}\sin^2(k\pi h) \\
&J_{2h}^{4h} : \mathcal{E}_{2h}^{4h} \to \operatorname{span}\left\{\phi_{4h}^k\right\} \text{ with } \tilde{J}_{2h}^{4h}(k) = \frac{\sqrt{2}}{2}\cos^2(k\pi h) \\
&J_{4h}^{2h} : \operatorname{span}\left\{\phi_{4h}^k\right\} \to \mathcal{E}_{2h}^{4h} \text{ with } \tilde{J}_{4h}^{2h}(k) = \sqrt{2}\cos^2(k\pi h).
\end{aligned}$$

Now, in light of Theorem 7.2 we have the following.

Theorem 8.2. *For any $\psi_h \in \mathbb{R}^N$, let $\tilde{\psi}_k = \begin{bmatrix} a_k & b_k & c_k & d_k \end{bmatrix} \psi_h^k$ where $a_k, b_k, c_k, d_k \in \mathbb{R}$ and $\psi_h^k = \begin{bmatrix} \phi_h^k & -\phi_h^{N-k} & -\phi_h^{N/2-k} & \phi_h^{N/2+k} \end{bmatrix}^T$; further suppose that $M_3 = M_h^{4h}$ and $M_3(k) = M_h^{4h}(k)$ are as in Lemma 6.2. Then we have*

$$\|fl(M_3(k)\tilde{\psi}_k) - M_3(k)\tilde{\psi}_k\| \leq \|M_3(k)\tilde{\psi}_k\| \cdot |\delta_k^3| \tag{8.4}$$

where $|\delta_k^3| \leq \left(12 + \nu + \eta_h^k \cdot \left[3\eta_{2h}^k + 5\right]\right)\epsilon$ with $\eta_h^k = \cot^2\left(\frac{k\pi h}{2}\right)$.

Moreover, ψ_h satisfies

$$\begin{aligned}
&\|fl(M_3\psi_h) - M_3\psi_h\| \\
&\leq C\|M_3\|\left(\left(\tfrac{N}{4}\right)^2 + (10+\nu)\tfrac{N}{4} - (11+\nu) + \textstyle\sum_{k=1}^{N-1}\tfrac{1}{4}\cot^4\left(\tfrac{k\pi h}{2}\right) + \left(\tfrac{3}{2}+\nu\right)\cot^2\left(\tfrac{k\pi h}{2}\right)\right)\epsilon \\
&\leq C\|M_3\|p_3(N)\epsilon
\end{aligned}$$

where $p_3(N)$ is a degree 4 polynomial in $\frac{N}{4}$ and C is a constant depending on the coefficients $a_k, b_k, c_k,$ and d_k.

Proof. Let $\psi_h \in \mathbb{R}^N$ and $\tilde{\psi}_k$, M_3 and $M_3(k)$ be as in the statement. Then we prove eq. (8.4) by explicitly calculating δ_n for $n = 1, \ldots, 4$ as defined in Theorem 7.2. Recall that each δ_n requires us to determine and bound $\beta_n^1, \beta_n^2, \beta_n^3$. For all n, $\beta_n^3 = 10 + 2\nu$, we are thus left to calculate β_n^2 and β_n^3. Recall that $k = 1, \ldots, \frac{N}{4} - 1$.

For δ_1:

$$\left|\frac{(M_{2h}^{4h})_{11}}{1-(M_{2h}^{4h})_{11}}\right| = \frac{|\sin^2(k\pi h)(1-(4/3)\sin^2(k\pi h))^3|}{|1-\sin^2(k\pi h)(1-(4/3)\sin^2(k\pi h))^3|} \leq \frac{81}{943}$$
$$\Rightarrow |\beta_1^2| \leq 8+\nu+(.08589)\left[3|\eta_{2h}^k|+2+\nu\right]$$

$$\left|\frac{\lambda_h^k/\lambda_{2h}^k \tilde{J}_{2h}^h(k)\tilde{J}_h^{2h}(k)\cdot(1-2(M_{2h}^{4h})_{11})}{1-\lambda_h^k/\lambda_{2h}^k \tilde{J}_{2h}^h(k)\tilde{J}_h^{2h}(k)\cdot(1-(M_{2h}^{4h})_{11})}\right| = \frac{\cos^2(k\pi h/2)(1-2(M_{2h}^{4h})_{11})}{1-\cos^2(k\pi h/2)(1-(M_{2h}^{4h})_{11})}$$
$$\leq \frac{\cos^2(k\pi h/2)}{1-\cos^2(k\pi h/2)} = \eta_h^k$$
$$\Rightarrow |\beta_1^1| \leq 2+\nu+\eta_h^k\left[3|\eta_{2h}^k|+2+\nu\right]$$

$$\Rightarrow |\delta_1| \leq (3+2+\nu+\eta_h^k\left[2\eta_{2h}^k+2+\nu\right])\epsilon$$

For δ_2:

$$|\beta_2^2| = |\beta_1^2| \leq 8+\nu+(.08589)\left[3|\eta_{2h}^k|+2+\nu\right]$$

$$\left|\frac{\lambda_h^{N-k}/\lambda_{2h}^k \tilde{J}_{2h}^h(N-k)\tilde{J}_h^{2h}(N-k)\cdot(1-2(M_{2h}^{4h})_{11})}{1-\lambda_h^{N-k}/\lambda_{2h}^k \tilde{J}_{2h}^h(N-k)\tilde{J}_h^{2h}(N-k)\cdot(1-(M_{2h}^{4h})_{11})}\right|$$
$$= \frac{\sin^2(k\pi h/2)(1-2(M_{2h}^{4h})_{11})}{1-\sin^2(k\pi h/2)(1-(M_{2h}^{4h})_{11})} \leq \frac{\sin^2(k\pi h/2)}{1-\sin^2(k\pi h/2)} = \eta_h^{N-k} \leq \frac{2-\sqrt{2}}{2+\sqrt{2}}$$

$$\Rightarrow |\beta_2^1| \leq 2+\nu+(0.17157)\left[2\eta_{2h}^k+2+\nu\right])\epsilon$$

$$\Rightarrow |\delta_2| \leq (3+2+\nu+(.017157)\left[2\eta_{2h}^k+2+\nu\right])\epsilon$$

For δ_3:

$$\left|\frac{(M_{2h}^{4h})_{22}}{1-(M_{2h}^{4h})_{22}}\right| = \frac{|\cos^2(k\pi h)(1-(4/3)\cos^2(k\pi h))^3|}{|1-\cos^2(k\pi h)(1-(4/3)\cos^2(k\pi h))^3|} \leq \frac{2}{53}$$
$$\Rightarrow |\beta_3^2| \leq 8+\nu+(.037735)\left[3|\eta_{2h}^{N/2-k}|+2+\nu\right]$$

$$\left|\frac{\lambda_h^{N/2-k}/\lambda_{2h}^{N/2-k} \tilde{J}_{2h}^h(N/2-k)\tilde{J}_h^{2h}(N/2-k)\cdot(1-2(M_{2h}^{4h})_{22})}{1-\lambda_h^{N/2-k}/\lambda_{2h}^{N/2-k} \tilde{J}_{2h}^h(N/2-k)\tilde{J}_h^{2h}(N/2-k)\cdot(1-(M_{2h}^{4h})_{22})}\right|$$
$$= \frac{(1/16)(1+\sin(k\pi h))(1-2(M_{2h}^{4h})_{22})}{1-(1/16)(1+\sin(k\pi h))(1-(M_{2h}^{4h})_{22})} \leq \frac{(1/32)(2+\sqrt{2})(29/27)}{1-(28/27)(1/32)(2+\sqrt{2})}$$
$$\leq 0.128906$$

$$\Rightarrow |\beta_3^1| \leq (2+\nu+(0.128906)\left[2\eta_{2h}^k+2+\nu\right])\epsilon$$

$$\Rightarrow |\delta_3| \leq (2+\nu+(.1289)\left[3|\eta_{2h}^{N/2-k}|+2+\nu\right])\epsilon$$

For δ_4:

$$|\beta_4^2| = |\beta_3^2| \leq 8+\nu+(.037735)\left[3|\eta_{2h}^{N/2-k}|+2+\nu\right]$$

$$\left|\frac{\lambda_h^{N/2+k}/\lambda_{2h}^{N/2-k}\tilde{J}_{2h}^h(N/2+k)\tilde{J}_h^{2h}(N/2+k)\cdot(1-2(M_{2h}^{4h})_{22})}{1-\lambda_h^{N/2+k}/\lambda_{2h}^{N/2-k}\tilde{J}_{2h}^h(N/2+k)\tilde{J}_h^{2h}(N/2+k)\cdot(1-(M_{2h}^{4h})_{22})}\right|$$
$$= \frac{(1/16)(1-\sin(k\pi h))(1-2(M_{2h}^{4h})_{22})}{1-(1/16)(1-\sin(k\pi h))(1-(M_{2h}^{4h})_{22})} \leq \frac{(1/16)(29/27)}{1-(28/27)(1/16)} \leq .0717$$

$$\Rightarrow |\beta_4^1| \leq (2+\nu+(0.0717)\left[2\eta_{2h}^k+2+\nu\right])\epsilon$$

$$\Rightarrow |\delta_4| \leq (2+\nu+(.0717)\left[3|\eta_{2h}^{N/2-k}|+2+\nu\right])\epsilon$$

Since η_h^k is unbounded as $h \to 0$, we have $\max_n \delta_n = \delta_1$ and therefore, entry-wise, for $i = 1, \ldots, N$, we have (again, by Theorem 7.2),

$$fl\left((M_3(k)\tilde{\psi}_k)_i\right) = (M_3(k)\tilde{\psi}_k)_i(1+\delta_k^3)$$

where $|\delta_k^3| \leq |\delta_1|+|\delta_5| \leq (3+2+\nu+\eta_h^k\left[3\eta_{2h}^k+2+\nu\right])\epsilon+7\epsilon = (12+\nu+\eta_h^k\left[3\eta_{2h}^k+2+\nu\right])\epsilon$.

Now, for

$$\psi_h = \sum_{k=1}^{N/4-1}\begin{bmatrix}a_k & b_k & c_k & d_k\end{bmatrix}\psi_h^k = \sum_{k=1}^{N/4-1}\tilde{\psi}_k.$$

we have

$$\|fl\left(M_3\psi_h\right) - M_3\psi_h\| \leq \left\|\sum_{k=1}^{N/4-1} M_2\tilde{\psi}_k\tilde{\delta}_k^3\right\| \quad \text{where} \quad \left|\tilde{\delta}_k^3\right| \leq (N/4-1)\epsilon + |\delta_k^3|$$
$$\leq C\|M_3\|\sum_{k=1}^{N/4-1}\left|\tilde{\delta}_k^3\right|.$$

Similar to Theorem 8.1 we can bound this sum as follows.

$$\sum_{k=1}^{N/4-1}\left|\tilde{\delta}_k^3\right| = (N/4-1+12+\nu)(N/4-1)\epsilon + \sum_{k=1}^{N/4-1} 3\eta_h^k\eta_{2h}^k + (2+\nu)\eta_{2h}^k$$
$$= (N/4)^2 + (10+\nu)(N/4) - (11+\nu) + \textstyle\sum_{k=1}^{N/4-1} 3\cot^2\left(\frac{k\pi h}{2}\right)\cot^2\left(k\pi h\right) + (2+\nu)\cot^2\left(\frac{k\pi h}{2}\right)$$
$$= (N/4)^2 + (10+\nu)(N/4) - (11+\nu) + \textstyle\sum_{k=1}^{N/4-1} \frac{1}{4}\cot^4\left(\frac{k\pi h}{2}\right) + \left(\frac{3}{2}+\nu\right)\cot^2\left(\frac{k\pi h}{2}\right) + \frac{1}{4}.$$

At worst, we can bound this sum above by running k from 1 to N, in which case summing $\cot^4(k\pi h/2)$ and $\cot^2(k\pi h/2)$ yield polynomials in N of degree 4 and 2 respectively. In fact, we can explicitly bound these terms as

$$\sum_{k=1}^{N/4-1} \frac{1}{4}\cot^4\left(\frac{k\pi h}{2}\right) + \left(\frac{3}{2}+\nu\right)\cot^2\left(\frac{k\pi h}{2}\right) + \frac{1}{4}$$
$$\leq \frac{64}{45}\left(\frac{N}{4}\right)^4 + \left(\frac{92}{9}+8\nu\right)\left(\frac{N}{4}\right)^2 + \frac{5}{4}\left(\frac{N}{4}\right) - \left(\frac{11}{8}+2\nu\right).$$

which implies that

$$\textstyle\sum_{k=1}^{N/4-1}\left|\tilde{\delta}_k^3\right| \leq p_3(N) = \frac{64}{45}\left(\frac{N}{4}\right)^4 + \left(\frac{101}{9}+8\nu\right)\left(\frac{N}{4}\right)^2 + \left(\frac{37}{4}+\nu\right)\left(\frac{N}{4}\right) - \left(\frac{83}{8}+3\nu\right).$$

Numerically calculating the actual sum $\sum_{k=1}^{N/4-1}\frac{1}{4}\cot^4\left(\frac{k\pi h}{2}\right) + \left(\frac{3}{2}+\nu\right)\cot^2\left(\frac{k\pi h}{2}\right)$ for $N = 2^L$ and $L = 2,\ldots,20$ reveals that these sums are indeed of the order N^4. In particular, they are of the order $\cot^4\left(\frac{\pi h}{2}\right)$.

□

Theorem 8.2 is enlightening in the sense that it highlights the recursive pattern of error accumulation in going from an l-grid to an $(l+1)$-grid operator. The unfortunate part is the large bound. An order N^4 bound is clearly less than ideal, and in light of Example 5.1 does not reflect realities. Instead, we bound the roundoff errors of a 3-grid operator using the results of Lemma 6.3. We first calculate the bounds C_1 and C_2 of Lemma 6.3 for the Model Problem and the Model Algorithm.

Corollary 8.1. *Recall from Lemma 6.3 that the l-grid V-cycle operator can be written as*

$$M_l = M_1^l = M_{l-1}^l + B_l^{l-1}M_{l-1}B_{l-1}^l. \tag{8.5}$$

For the Model Algorithm applied to the Model Problem, we have

$$\|M_{l-1}^l\| \leq C_1 = 0.360559 \;\; \textit{and} \;\; \|B_{l-1}^l\|\cdot\|B_l^{l-1}\| \leq C_2 = \frac{\sqrt{82}}{9},$$

hence

$$\|M_l\| \leq (-58.5889)\left(1-(1.00615)^{l-1}\right).$$

Proof. The 2-grid operator bound $\|M_{l-1}^l\| \leq C_1 = 0.360559$ is proved in Theorem 8.1. For $\|B_l^{l-1}\|\|B_{l-1}^l\| \leq \frac{\sqrt{82}}{9}$ we take a similar approach. Indeed, let us consider

$$B_2^1 = S_h^{\nu_2}J_{2h}^h \quad \text{and} \quad B_1^2 = A_{2h}^{-1}J_h^{2h}A_hS_h^{\nu_1}.$$

Then $\|B_2^1\|_2^2 = \max_k\left\{\mu_{\max}\left(B_2^1(k)^T\right)\left(B_2^1(k)\right)\right\}$ where, for $k = 1,\ldots\frac{N}{2}-1$,

$$B_2^1(k) = S_h^{\nu_2}(k)J_{2h}^h(k) = \begin{bmatrix} 1-\frac{4}{3}\sin^2\left(\frac{k\pi h}{2}\right) & \\ & 1-\frac{4}{3}\cos^2\left(\frac{k\pi h}{2}\right)\end{bmatrix}^{\nu_2}\begin{bmatrix}\sqrt{2}\cos^2(k\pi h/2)\\ \sqrt{2}\sin^2(k\pi h/2)\end{bmatrix}.$$

This yields,

$$\|B_2^1\| = \max_k \left\{ \cos^4(k\pi h/2)(1 - \frac{4}{3}\sin^2(k\pi h/2)^{2\nu_2} + 2\sin^2(k\pi h/2)(1 - \frac{4}{3}\cos^2(k\pi h/2))^{2\nu_2} \right\}$$
$$= \max_k \left\{ \frac{1}{9}\sin^4(k\pi h) - \frac{7}{3}\sin^2(k\pi h) + 2 \right\} \leq 2$$
$$\Rightarrow \|B_2^1\| \leq \sqrt{2}.$$

Similarly, $\|B_1^2\|_2^2 = \max_k \left\{ \mu_{\max}(B_1^2(k))^T (B_1^2) \right\}$, where, for $k = 1, \ldots, \frac{N}{2} - 1$

$$B_1^2(k) = A_{2h}^{-1}(k) J_h^{2h}(k) A_h(k) S_h^{\nu_1}(k)$$
$$= \frac{4}{\sin^2(k\pi h)} \begin{bmatrix} \frac{\sqrt{2}}{2}\cos^2\left(\frac{k\pi h}{2}\right) \\ \frac{\sqrt{2}}{2}\sin^2\left(\frac{k\pi h}{2}\right) \end{bmatrix}^T \begin{bmatrix} \sin^2\left(\frac{k\pi h}{2}\right) & \\ & \cos^2\left(\frac{k\pi h}{2}\right) \end{bmatrix} \begin{bmatrix} 1 - \frac{4}{3}\sin^2\left(\frac{k\pi h}{2}\right) & \\ & 1 - \frac{4}{3}\cos^2\left(\frac{k\pi h}{2}\right) \end{bmatrix}^{\nu_1}.$$

This yields

$$\|B_1^2\| = \max_k \frac{1}{2} \left\{ \left(1 - \frac{4}{3}\sin^2\left(\frac{k\pi h}{2}\right)\right)^4 + \left(1 - \frac{4}{3}\cos^2\left(\frac{k\pi h}{2}\right)\right)^4 \right\}$$
$$= \max_k \frac{1}{162} \left\{ 4\cos(4k\pi h) + 40\cos(2k\pi h) + 38 \right\} \leq \frac{41}{81}$$
$$\Rightarrow \|B_1^2\| \leq \frac{\sqrt{41}}{9}$$
$$\Rightarrow \|B_l^{l-1}\| \|B_{l-1}^l\| \leq C_2 = \sqrt{2} \cdot \frac{\sqrt{41}}{9} = \frac{\sqrt{82}}{9}.$$

□

This allows us to derive a more practical bound on the relative error of the computed $M_3\psi_h$.

Theorem 8.3. *Let $\psi_h \in \mathbb{R}^N$ and $M_3 = M_h^{4h}$ be as in Lemma 6.2. Then we have*

$$\frac{\|fl\left(M_3\psi_h\right) - M_3\psi_h\|}{\|M_3\psi_h\|} \leq C\|M_2\| \left\{ p_2(N) + \frac{\sqrt{82}}{9} \right\} \epsilon$$

where C, $\|M_2\|$, and $p_2(N)$ are as in Theorem 8.1.

Proof. Let $\psi_h \in \mathbb{R}^N$. If

$$M_h^{4h} = M_h^{2h} + \left(S_h^{\nu_2} J_{2h}^h\right) M_{2h}^{4h} \left(A_{2h}^{-1} J_h^{2h} A_h S_h^{\nu_1}\right) = M_h^{2h} + B_{2h}^h M_{2h}^{4h} B_h^{2h}$$

then for $i = 1, \ldots, N$ we have

$$fl\left((M_h^{4h}\psi_h)_i\right) = \left(fl\left((M_h^{2h}\psi_h)_i\right)(1+\alpha_1) + fl\left((B_{2h}^h M_{2h}^{4h} B_h^{2h}\psi_h)_i\right)(1+\alpha_2)\right)(1+\alpha_3)$$
where $|\alpha_1| \leq C\|M_h^{2h}\| |p_2(N)\epsilon$,
$|\alpha_2| \leq \left(\|B_{2h}^h\| \|M_{2h}^{4h}\| \|B_{2h}^h\| \|\psi_h\| + C\|M_{2h}^{4h}\| p_2(N)\right)\epsilon$,
and $|\alpha_3| \leq \epsilon$.

From the proof of Theorem 8.1, $p_2(N) = 9\left(\frac{N}{2}\right)^2 + (1+\nu)\frac{N}{2} + (-10+\nu)$, $C = \max_k \{a_k, b_k\}$ comes from the coefficients in $\tilde{\psi}_k$, and both $\|M_h^{2h}\|, \|M_{2h}^{4h}\| \leq 0.360559$.

This implies that

$$\begin{aligned}
&fl\left(M_h^{4h}\psi_h\right) = M_h^{4h}\psi_h(1+\delta) \\
&\text{where} \quad |\delta| \leq \left(\|B_{2h}^h\|\|M_{2h}^{4h}\|\|B_{2h}^h\|\|\psi\| + C\|M_{2h}^{4h}\|p_2(N)\right) \\
&\Rightarrow \frac{\|fl\left(M_h^{4h}\psi_h\right) - M_h^{4h}\psi_h\|}{\|M_h^{4h}\psi_h\|} \leq C(0.360559)\left(p_2(N) + \frac{\sqrt{82}}{9}\right)\epsilon.
\end{aligned}$$

□

Theorem 8.2 thus tell us that the relative error of the 3-grid V-cycle is of the order N^2. In the next section, we extend this result to a general l-grid V-cycle, which will allow for conclusions about an L-grid V-cycle to solve a linear system of size $2^L \times 2^L$, as well as future work in drawing conclusions about FMG algorithms.

8.3 Model Problem Generalizations

The previous section demonstrated the recursive-like dependencies that calculations concerning l-grid operators have on $(l+1)$-grid operators. In this section, we will make some generalizations.

First, we can directly apply the results of Theorem 8.1 on $|\delta_2|$ together with those of Theorem 7.3 and Corollary 8.1 to bound $fl\left(M_l\psi_h\right)$.

Corollary 8.2. *Let $\psi_h \in \mathbb{R}^N$ with $\|\psi_h\| \leq C$ and let M_l be the the l-grid multigrid operator as defined by Equation* (6.5) *and the Model Algorithm. Then*

$$M_l = M_1^l = M_{l-1}^l + B_l^{l-1}M_{l-1}B_{l-1}^l$$

is such that $\|M_{l-1}^l\| \leq C_1 = 0.360559$ and $\|B_l^{l-1}\| \cdot \|B_{l-1}^l\| \leq C_2 = \frac{\sqrt{82}}{9}$. Let δ_2 be the error bound of the 2-grid operators as in Theorem 8.1. Namely, suppose that $\|fl\left(M_{l-1}^l\psi_h\right) - M_{l-1}^l\psi_h\| = |\delta_2|$ where

$$|\delta_2| \leq C(0.360559)p_2(N)\epsilon \tag{8.6}$$

where $p_2(N)$ is a degree 2 polynomial in $\frac{N}{2}$. Then

$$\frac{\|fl\left(M_l\psi_h\right) - M_l\psi_h\|}{\|M_l\psi_h\|} \leq |\delta_l|$$

where

$$|\delta_l| \leq C(l-1)(0.360559)p_2(N)\epsilon + (l-2)\epsilon - C\frac{0.360559}{(.0061539)^2}\left((-.0061539)^l + 1.012269\right)\epsilon. \tag{8.7}$$

Proof. The bound eq. (8.7) follows directly from Theorem 7.3. □

Thus, Corollary 8.2 shows that the relative error in floating point arithmetic of an l-grid V-cycle defined by both the Model Problem and the Model algorithm, while it does not depend explicitly on $\kappa(A_h)$, is at least of order N^2. Because the V-cycle is the building block of FMG, this result is a starting point in understanding the effects of roundoff errors in a full multigrid algorithm. Independently, this result suggests that a standard multigrid algorithm should behave *at least* as good as a backward stable algorithm, and our analysis does not reflect the lore that multigrid algorithms are unaffected by the extreme ill-conditioning.

8.4 Conclusions and Future Work

Recall that our motivating Example 5.1 suggested that a full multigrid algorithm can compute solutions with relative accuracy slightly better than that of a backward stable algorithm, but slightly worse than that of an inverse-equivalent algorithm. Our conjecture was that the condition number might play a small roll in this relative error, but it doesn't seem that it depends on it in the same sense that backward stability does. The theory of Chapter 7 does not explicitly convey any dependence on $\kappa(A_h)$, but the computations and application of these bounds to the Model Problem in this chapter provides a bound that is the same order of the condition number. Indeed, the conclusions in Section 8.3 indicate that the relative errors are of the order N^2, including some constants obtained from two-grid operators.

Immediate future work includes an additional numerical example where solutions are computed by iterating an L-grid V-cycle. These errors can then be compared with the bound δ_L determined in eq. (8.7), as well as with $\epsilon\kappa(A_h)$. Moreover, since the l-grid V-cycles are the building blocks of the FMG, a natural next step is extending Theorem 7.3 and Corollary 8.2 to a result on a full multigrid algorithm. These results could then be compared directly with our original motivating example.

The author is also interested in applying/extending these results to the fourth order biharmonic operator to compare these methods directly with the work of Part I. In particular, it is of interest, whether or not these order N^2 errors bounds will hold on a operator with condition number on the order of N^4. Such an example would provide compelling evidence, together with an initial theoretical framework, for the accuracy of multigrid methods. And finally, we would be remiss to not address the extension of this theoretical framework outside of the Model Algorithm and the various assumptions of Chapter 7.

Bibliography

[1] Louis Bauer and Edward L. Reiss. Block five diagonal matrices and the fast numerical solution of the biharmonic equation. *Math. Comp.*, 26:311–326, 1972.

[2] P. E. Bjorstad and B. P. Tjostheim. High precision solutions of two fourth order eigenvalue problems. *Computing*, 63(2):97–107, 1999.

[3] P.E. Bjorstad and B.P. Tjostheim. Efficient algorithms for solving a fourth-order equation with the spectral-galerkin method. *Siam Journal On Scientific Computing*, 18(2):621–632, March 1997.

[4] Achi Brandt. Rigorous local mode analysis of multigrid. *Preliminary Proceeding of the Fourth Copper Mountain Conference on Multigrid Methods*, April 1989.

[5] Achi Brandt. Rigorous quantitative analysis of multigrid. I. Constant coefficients two-level cycle with L_2-norm. *SIAM J. Numer. Anal.*, 31(6):1695–1730, 1994.

[6] William L. Briggs, Van Emden Henson, and Steve F. McCormick. *A multigrid tutorial*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2000.

[7] Goong Chen, Matthew P. Coleman, and Jianxin Zhou. Analysis of vibration eigenfrequencies of a thin plate by the Keller-Rubinow wave method. I. Clamped boundary conditions with rectangular or circular geometry. *SIAM J. Appl. Math.*, 51(4):967–983, 1991.

[8] Siegfried Cools and Wim Vanroose. Local Fourier analysis of the complex shifted Laplacian preconditioner for Helmholtz problems. *Numer. Linear Algebra Appl.*, 20(4):575–597, 2013.

[9] R. Courant and D. Hilbert. *Methods of mathematical physics. Vol. II.* Wiley Classics Library. John Wiley & Sons, Inc., New York, 1989. Partial differential equations, Reprint of the 1962 original, A Wiley-Interscience Publication.

[10] Carl W. David. The laplacian in sphereical polar coordinates. Department of Chemistry at DigitalCommons@UConn, 2007. Chemistry Education Materials, Paper 34.

[11] James Demmel, Ming Gu, Stanley Eisenstat, Ivan Slapnivcar, Krevsimir Veselić, and Zlatko Drmavc. Computing the singular value decomposition with high relative accuracy. *Linear Algebra Appl.*, 299(1-3):21–80, 1999.

[12] James W. Demmel. *Applied numerical linear algebra*. Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[13] Froilán M. Dopico and Plamen Koev. Perturbation theory for the LDU factorization and accurate computations for diagonally dominant matrices. *Numer. Math.*, 119(2):337–371, 2011.

[14] Froilán M. Dopico and Juan M. Molera. Accurate solution of structured linear systems via rank-revealing decompositions. *IMA J. Numer. Anal.*, 32(3):1096–1116, 2012.

[15] Louis Ehrlich. Solving the biharmonic equation as coupled finite difference equations. *SIAM Journal on Numerical Analysis*.

[16] W. N. Everitt, B. T. Johansson, L. L. Littlejohn, and C. Markett. Quasi-separation of the biharmonic partial differential equation. *IMA J. Appl. Math.*, 74(5):685–709, 2009.

[17] Scott R. Fulton. On the accuracy of multigrid truncation error estimates. *Electron. Trans. Numer. Anal.*, 15:29–37, 2003. Tenth Copper Mountain Conference on Multigrid Methods (Copper Mountain, CO, 2001).

[18] Zhendong Luo. A high accuracy numerical method based on spectral theory of compact operator for biharmonic eigenvalue equations. *J. Inequal. Appl.*, pages 2016:77, 11, 2016.

[19] Carmen Rodrigo, Francisco J. Gaspar, Cornelis W. Oosterlee, and Irad Yavneh. Accuracy measures and Fourier analysis for the full multigrid algorithm. *SIAM J. Sci. Comput.*, 32(5):3108–3129, 2010.

[20] Jie Shen. Efficient spectral-galerkin method i. direct solvers of second- and fourth-order equations using legendre polynomials. *SIAM Journal on Scientific Computing*, 15(6), November 1994.

[21] Klaus Stüben and Ulrich Trottenberg. Multigrid methods: fundamental algorithms, model problem analysis and applications. In *Multigrid methods (Cologne, 1981)*, volume 960 of *Lecture Notes in Math.*, pages 1–176. Springer, Berlin-New York, 1982.

[22] U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, Inc., San Diego, CA, 2001. With contributions by A. Brandt, P. Oswald and K. Stüben.

[23] Roman Wienands and Wolfgang Joppich. *Practical Fourier analysis for multigrid methods*, volume 4 of *Numerical Insights*. Chapman & Hall/CRC, Boca Raton, FL, 2005. With 1 CD-ROM (Windows and UNIX).

[24] Roman Wienands and Cornelis W. Oosterlee. On three-grid Fourier analysis for multigrid. *SIAM J. Sci. Comput.*, 23(2):651–671, 2001. Copper Mountain Conference (2000).

[25] C. Wieners. Bounds for the n lowest eigenvalues of fourth-order boundary value problems. *Computing*, 59(1):29–41, March 1997.

[26] Qiang Ye. Preconditioning for accurate solutions of linear systems and eigenvalue problems. *arXiv ID:1705.04340[math.NA].*

[27] Qiang Ye. Computing singular values of diagonally dominant matrices to high relative accuracy. *Math. Comp.*, 77(264):2195–2230, 2008.

[28] Qiang Ye. Accurate inverses for computing eigenvalues of extremely ill-conditioned matrices and differential operators. *Math. Comp.*, 87(309):237–259, 2018.

Vita

Education

- *Texas Tech University,* Lubbock, Texas
  MS in Mathematics, May 2013
  BS in Mathematics, May 2011
  with honors, cum laude

Professional Positions

- *Teaching Assistant,* University of Kentucky, Lexington, Kentucky
  August 2014-May 2019
- *NSF Mathematical Sciences Graduate Internship,* Nevada National Security Site, Las Vegas, Nevada
  Summer 2017
- *Lecturer*, Texas Tech University, Lubbock, Texas
  August 2013 - May 2014
- *Research Assistant*, Texas Tech University, Lubbock, Texas

Awards and Honors

- *Mathematics Department Fellowship*, University of Kentucky, Lexington, Kentucky
  2018-2019 Academic Year
- *Royster Outstanding Teaching Award*, University of Kentucky, Lexington, Kentucky
  2016-2017 Academic year
- *Max Steckler Fellowship*, University of Kentucky, Lexington, Kentucky
  2016-2017 Academic Year