Summer 8-2016

Krylov Subspace Spectral Method with Multigrid for a Time-Dependent, Variable-Coefficient Partial Differential Equation

Haley Renee Dozier
University of Southern Mississippi

KRYLOV SUBSPACE SPECTRAL METHOD WITH MULTIGRID FOR A TIME-DEPENDENT, VARIABLE-COEFFICIENT PARTIAL DIFFERENTIAL EQUATION

by

Haley Renee Dozier

A Thesis
Submitted to the Graduate School
and the Department of Mathematics
at The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Master of Science

Approved:

_____
Dr. James Lambers, Committee Chair
Associate Professor, Mathematics

_____
Dr. Haiyan Tian, Committee Member
Associate Professor, Mathematics

_____
Dr. Huiqing Zhu, Committee Member
Associate Professor, Mathematics

_____
Dr. Karen S. Coats
Dean of the Graduate School

August 2016

Published by the Graduate School

ABSTRACT

KRYLOV SUBSPACE SPECTRAL METHOD WITH MULTIGRID FOR A TIME-DEPENDENT, VARIABLE-COEFFICIENT PARTIAL DIFFERENTIAL EQUATION

by Haley Renee Dozier

August 2016

Krylov Subspace Spectral (KSS) methods are traditionally used to solve time-dependent, variable-coefficient PDEs. They are high-order accurate, component-wise methods that are efficient with variable input sizes.

This thesis will demonstrate how one can make KSS methods even more efficient by using a Multigrid-like approach for low-frequency components. The essential ingredients of Multigrid, such as restriction, residual correction, and prolongation, are adapted to the time-dependent case. Then a comparison of KSS, KSS with Multigrid, KSS-EPI and standard Krylov projection methods will be demonstrated.

ACKNOWLEDGMENTS

I would like to thank all of the people who have helped me with writing this thesis, and for those who provided moral support.

First of all, I would like to thank my wonderful parents for their continuous encouragement and unfailing support no matter the cost.

Dr. James Lambers, for being with me every step of the way and pushing me to do my best (even when I was being particularly stubborn).

Most of all, I would like to thank my sister, Francie Sutherland. Without her influence, I wouldn't be the person I am today. Francie inspires me to be better everyday and I hope, one day, I can repay her for all of her guidance, love, and faith in her little sister.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure

LIST OF TABLES

Table

Chapter 1

BACKGROUND

1.1 Introduction

In this thesis, we will discuss certain time-stepping methods for time-dependent, variable coefficient partial differential equations (PDEs), known as Krylov Subspace Spectral methods. Krylov Subspace Spectral, or KSS, methods were made known in 2003 by Dr. James Lambers for his doctoral dissertation. Since then, many advances have been made to KSS methods, including the switch to one block Lanczos quadrature rule for each Fourier component instead of two "non-block" rules, which improved the accuracy of the method, and performing an asymptotic analysis of Lanczos iteration, which improved the efficiency. In this thesis, it will be shown that another advance, the combination of Multigrid methods with KSS methods, will further improve KSS.

First, we examine a time-dependent, variable coefficient PDE, such as

$$u_t - \frac{1}{r(x)m(t)}[(p(x)u_x)_x + q(x)u] = 0, \qquad a < x < b, \quad t > 0.$$

There are many existing methods to numerically solve this type of equation, including finite difference methods, finite element methods, and spectral methods. The problem with using any of these methods in combination with existing time-stepping methods, such as Runge-Kutta or multistep methods, is that efficiency of these methods doesn't scale well to larger input sizes (number of grid points). That is, unless the chosen time-step is sufficiently small, the computed solutions might exhibit unreasonable behavior with large input sizes [5]. Therefore, to improve accuracy as input sizes increases, the time step must be chosen to be even smaller, so not only is the expense per time step increasing, but the number of time steps needed to maintain accuracy is increasing as well.

KSS methods are used specifically to solve time-dependent, variable coefficient PDEs. Unlike the previously mentioned methods, KSS methods are explicit methods that have the stability of implicit methods, and are component-wise methods that are scalable — the method is reliable using a variety of input sizes. The difference between KSS and Krylov projection methods, as described in [7], is that KSS computes each Fourier coefficient of the solution using an approximation of the solution operator made for that specific component.

1.2 Multigrid Methods

Multigrid methods can be used to solve the linear system $Au = b$ according to the following framework: smoothing, restriction, then interpolation/prolongation. Multigrid methods are effective because many iterative techniques already contain a smoothing property that can eliminate high frequency (oscillatory) components of the error but have little to no effect on the low frequency components of the error. Multigrid methods can fix this effect by using residual correction on a coarser grid.

To integrate Multigrid into an existing method a Multigrid correction scheme is implemented. After the smoothing takes place in the existing method, a Multigrid correction scheme is added so that after convergence starts to deteriorate on the fine grid, the residual is restricted to a coarser grid to obtain an approximation of the error. Then the error is used to correct the initial approximation of the solution and is returned to the fine grid.

1.3 Combining KSS and Multigrid Methods

Due to the work of Cibotarica, Palchak, and Lambers in [2, 10], KSS methods are highly effective at eliminating the high frequency components of the error. In this thesis, it will be shown that incorporating Multigrid into KSS methods improves the accuracy of the method by applying a Multigrid-like approach on the low frequency components.

Since Multigrid methods are typically used to solve linear systems that arise from the spatial discretization of elliptic partial differential equations which are time-independent (such as Laplace's equation), we must first adapt the method to the time-dependent case. Therefore, to apply Multigrid to time-dependent problems, we must first generalize the method by redefining the residual, and then solve a non-homogeneous version of the PDE to obtain the error for the correction. Then we must create new functions to restrict the approximate solution, correct, and interpolate the corrected approximate solution.

1.4 Outline

In Chapter 2, we will review KSS methods in their current form and recent improvements made to these method. This includes the construction of KSS method, rapid node estimation and an asymptotic analysis of Lanczos iteration to improve the efficiency of the method.

Chapter 3 will describe traditional Multigrid methods and how they are used to solve linear systems of equations.

In Chapter 4, it will be shown that implementing Multigrid-like techniques with KSS methods increases the methods overall efficiency. It will also be shown how one can use

KSS with Multigrid, or KSS-MG, to solve parabolic and hyperbolic PDEs.

Chapter 5 will present numerical results from applying the Mutigrid modified KSS algorithm to both parabolic and hyperbolic problems. It will also compare Multigrid modified KSS to the previous KSS method as well as other standard methods.

Chapter 6 will conclude that KSS-MG is far more accurate than standard Krylov projection and more efficient than the current KSS methods. Further research will also be discussed.

Chapter 2

Krylov Subspace Spectral Methods

In this chapter, we will discuss Krylov Subspace Spectral (KSS) Methods, a time-stepping method for solving time-dependent, variable coefficient PDEs. We start by examining the initial-value problem

$$u_t + Lu = 0, \quad 0 < x < 2\pi, \quad t > 0 \tag{2.1}$$

$$u(x,0) = f(x), \qquad 0 < x < 2\pi, \tag{2.2}$$

with periodic boundary conditions

$$u(0,t) = u(2\pi,t), \quad t > 0. \tag{2.3}$$

The operator L is the Sturm-Liouville differential operator defined by

$$Lu = -(pu_x)_x + qu, \tag{2.4}$$

where p and q are functions of x. To solve this PDE, we first use the boundary conditions to find a general solution, then use the initial conditions to find a unique solution. The solution of this particular PDE can be represented by the Fourier series

$$u(x,t) = \frac{1}{\sqrt{2\pi}} \sum_{\omega=-\infty}^{\infty} e^{i\omega x} \hat{u}(\omega,t), \tag{2.5}$$

where $\hat{u}(\omega,t)$ is the Fourier coefficient of each component, and ω is the wave number for each Fourier component. To find the Fourier coefficients of the solution, we can represent each Fourier coefficients as an inner product, denoted by $\langle \cdot, \cdot \rangle$, as

$$\hat{u}(\omega,t_{n+1}) = \left\langle \frac{1}{\sqrt{2\pi}} e^{i\omega x}, e^{-L\triangle t} u(x,t_n) \right\rangle, \tag{2.6}$$

where $e^{-L\triangle t}$ is the exact solution operator.

2.1 KSS Methods

The main idea behind KSS methods is to independently approximate all Fourier coefficients of the solution using an approximation of the exact solution operator that is tailored to each specific Fourier coefficient. To find the approximation of the exact solution operator, we can spatially discretize (2.6) and obtain the following

$$\left(\frac{1}{\sqrt{2\pi}}e^{i\omega\vec{x}}\right)^H \left(e^{-L_N t}\vec{u}(t_n)\right), \tag{2.7}$$

where L_N is an $n \times n$ symmetric positive definite matrix obtained from the spectral discretization of L, and $\vec{x}$ is a vector of equally spaced points in $[0, 2\pi)$. To simplify, we can set $\vec{u} = \frac{1}{\sqrt{2\pi}}e^{i\omega\vec{x}}$, $\vec{v}v = u(\vec{x}, t_n)$, and $\phi(L_N) = e^{-L_N t}$, where $\vec{u}$ and $\vec{v}$ are N-vectors, to obtain the simplified form

$$\vec{u}^H \phi(L_N)\vec{v}. \tag{2.8}$$

Because the matrix L_N is positive definite, we know that it must have positive, real eigenvalues, denoted by $b = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N = a$, and orthonormal eigenvectors, denoted by $\vec{q_j}$ where $j = 1, ..., N$. The spectral decomposition of a matrix is a way to factor a matrix into its canonical form so that you can represent that matrix in terms of its eigenvalues and eigenvectors. The general form of the spectral decomposition of some matrix M is

$$\begin{aligned} M &= Q\lambda Q^H \\ &= \sum_{j=1}^{N} \lambda_j \vec{q_j} \vec{q_j^H}, \end{aligned} \tag{2.9}$$

where λ is a diagonal matrix with the eigenvalues of M as its entries and the columns of Q are the orthonormal eigenvectors, $\vec{q_j}$. Therefore the spectral decomposition of (2.8) can be seen as

$$\vec{u}^H \phi(L_N)\vec{v} = \sum_{j=1}^{N} \phi(\lambda_j)\vec{u}^H \vec{q_j} \vec{q_j^H}\vec{v}. \tag{2.10}$$

As shown by Golub and Meurant in [3], we can represent this sum as a Riemann-Stieltjes integral

$$\vec{u}^H \phi(L_N)\vec{v} = \int_a^b \phi(\lambda)\, d\alpha(\lambda) \tag{2.11}$$

such that

$$\alpha(\lambda) = \begin{cases} 0 & \text{if } \lambda < a \\ \sum_{j=i}^{N} \vec{u}^H \vec{q_j} \vec{q_j}^H \vec{v} & \text{if } \lambda_i \leq \lambda \leq \lambda_{i-1} \\ \sum_{j=1}^{N} \vec{u}^H \vec{q_j} \vec{q_j}^H \vec{v} & \text{if } b \leq \lambda. \end{cases} \tag{2.12}$$

To approximate a definite integral, any quadrature rule can be used. In this case, Gaussian quadrature is used because it has a high degree of accuracy and the weights are guaranteed

to be positive. Generally, Gaussian quadrature approximates an integral by choosing suitable nodes, x_i, and weights, w_i, then computing the following sum

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{K} w_i f(x_i).$$

Applying Gaussian quadrature rules to (2.11) leads to the approximation

$$\int_a^b \phi(\lambda)\,d\alpha(\lambda) = \sum_{j=1}^{K} \phi(\lambda_j) w_j + error$$

where the nodes are λ_j from $j = 1, ..., K$ and weights are w_j from $j = 1, ..., K$. In [3], it was shown that this quadrature rule is exact for polynomials of degree up to $2k-1$.

When $u \neq v$, the weights, w_j, are not always positive real numbers. This occurrence can destabilize the quadrature rule. Therefore, in this case, the block approach is considered:

$$[\vec{u}\ \vec{v}]^H \phi(L_N)[\vec{u}\ \vec{v}]. \tag{2.13}$$

This matrix can be seen as a Riemann-Stieltjes integral

$$\int_a^b \phi(\lambda)\,d\mu(\lambda) = \begin{bmatrix} \vec{u^H}\phi(L_N)\vec{u} & \vec{u^H}\phi(L_N)\vec{v} \\ \vec{v}^H\phi(L_N)\vec{u} & \vec{v}^H\phi(L_N)\vec{v} \end{bmatrix}, \tag{2.14}$$

where $\mu(\lambda)$ is a 2×2 matrix with entries of the form $\alpha(\lambda)$ from (2.12). Then, we can use a quadrature rule to approximate the integral (2.14) as the sum

$$\sum_{j=1}^{k} W_j \phi(T_j) W_j + error, \tag{2.15}$$

with W_j and T_j as 2×2 matrices. Diagonalizing T_j leads to

$$\sum_{j=1}^{k} \phi(\lambda_j)\vec{v}_j\vec{v}_j^H + error. \tag{2.16}$$

where λ_j is a scalar and each $\vec{v}$ is a 2 vector and $\vec{v}_j\vec{v}_j^H$ is a 2×2 matrix.

To obtain the nodes and weights for KSS, we need an algorithm whose output will give the recurrence relation for the orthogonal polynomials required for Gaussian quadrature rules for the integral we need to evaluate.

As shown in [4, 10], the block Lanczos algorithm applied to L_N using initial vectors $\vec{u}$ and $\vec{v}$ will give us appropriate nodes and weights for Gaussian quadrature. The algorithm for block Lanczos, as described in [10], is

$X_0 = 0$, $R_0 = [\vec{u}\ \vec{v}]$, $R_0 = X_1 B_0$ (QR factorization)
for $j = 1, 2, ..., K$
    $V = L_N X_j$
    $M_j = X_j^H V$
    if $j < K$
        $R_j = V - X_{j-1} B_{j-1}^H - X_j M_j$
        $R_j = X_{j+1} B_j$ (QR factorization)
    end
end.

When block Lanczos is applied to L_N using initial block $[\vec{u}\ \vec{v}]$, we obtain the 2×2 matrices M_j and B_j. These matrices are the entries of the block tridiagonal matrix

$$T_K = \begin{bmatrix} M_1 & B_1^T & & & \\ B_1 & M_2 & B_2^T & & \\ & \ddots & \ddots & \ddots & \\ & & & B_{K-1} & M_K \end{bmatrix} \tag{2.17}$$

where each entry is a function of ω and all B_j are upper triangular. Therefore, for Gaussian quadrature rules, the nodes λ_j are the eigenvalues of T_K, $\vec{v}_j \vec{v}_j^H$ are the matrix-valued weights and $\vec{v}_j$ consists of the first two components of the eigenvector corresponding to λ_j.

2.2 Block KSS

Block KSS as seen in [8] begins by defining

$$R_0(\omega) = [\hat{e}_\omega\ u^n] \tag{2.18}$$

where $\hat{e}_\omega$ is the discretization of $\frac{1}{\sqrt{2\pi}} e^{i\omega\vec{x}}$, which is $\vec{u}$ in the previous section, and $\vec{u}^n$ is the computed solution at the given time, which is $\vec{v}$ from the previous section. From [10], we know that the QR factorization of (2.18) leads to

$$R_0(\omega) = X_1(\omega) B_0(\omega) \tag{2.19}$$

with

$$X_1(\omega) = \begin{bmatrix} \hat{e}_w & \frac{u_w^n}{||u_w^n||^2} \end{bmatrix}$$

and

$$B_0(\omega) = \begin{bmatrix} 1 & \hat{e}_w^H u^n \\ 0 & ||u_w^n||^2 \end{bmatrix},$$

where

$$u^n_\omega = u^n - \hat{e}_\omega \hat{e}^H u^n = u^n - \hat{e}_\omega \hat{u}(\omega, t_n). \tag{2.20}$$

Next, block Lanczos can be applied to the discretized operator, L_N, with inital block $X_1(\omega)$. From block Lanczos, we obtain our M_j and B_j so that we can produce the matrix T_K with the same form as (2.17). The eigenvalues of this matrix are the nodes we will use for Gaussian quadrature and the Fourier coefficients at time t_{n+1} can be represented as

$$[\hat{u}^{n+1}]_\omega = [B_0^H E_{12}^H e^{-T_K(\omega)\triangle t} E_{12} B_0]_{12}, \qquad E_{12} = [\ \vec{e_1}\ \vec{e_2}\ ] \tag{2.21}$$

By applying an inverse Fast Fourier Transform (FFT) to the vector of Fourier coefficients, we obtain the vector $\vec{u}^{n+1}$, from which we can retrieve the values of the solution, $u(x, t_{n+1})$. According to Cibotarica, Palchak and Lambers in [10], this algorithm has local temporal accuracy of $O(\triangle t^{2K-1})$ for the parabolic problem and $O(\triangle t^{Kk-2})$ for the second-order wave equation.

The problem with Krylov Subspace Spectral Methods is that they need to perform Lanczos or Arnoldi on each component, which makes the method computationally expensive, but recently, research by Cibotarica, et al. in [2] led to the creation of an asymptotic analysis of Lanczos. This modification of the method will be discussed in the next section.

2.3 Asymptotic Analysis of Lanczos

Recall from the previous sections that $\hat{u}$ is the discrete Fourier transform of u on a uniform N-point grid. KSS methods use initial block $R_0(\omega) = [\hat{e}_\omega\ u^n]$ for each ω, then block Lanczos iteration is used with initial block of $R_0(\omega)$ to get T_K.

In [10], Palchak, Cibotarica and Lambers showed that for the PDE shown in (2.1), every non-zero, off diagonal entry of the subsequent matrices M_j and B_j converges to zero as the limit of the wave number, ω, approaches infinity. This is significant because then it can be shown that the matrix obtained from applying block Lanczos iteration to $R_0 = [\hat{e}_\omega\ u^n]$ actually converges to the matrix obtained when "non-block" Lanczos is applied to the columns of R_0 separately and then alternating rows and columns of the tridiagonal matrices produced by the "non-block" Lanczos iterations.

Since the block Lanczos case converges to the "non-block" case, there is a decoupling effect, and so for a finite ω, it was shown in [10] the Gaussian quadrature notes can be estimated from the eigenvalues of these smaller matrices, which has been shown to be hundreds of times faster. It should be noted that in the subsequent pages, the nodes that are computed from applying "non-block" Lanczos to the spectral discretization of L, L_N, with initial block u^n will be referred to as the "frequency independent nodes" since they are

mainly dependent on the computed solution. These nodes will only have to be calculated once in the KSS method. The rest of the nodes that are computed from applying "non-block" Lanczos to $\hat{e}_\omega$ will be referred to as "frequency dependent nodes".

After obtaining the matrices from Lanczos iteration on the separate entries of R_0 and computing the frequency independent nodes, the frequency dependent nodes must be computed for each ω. This process can be computationally expensive, but in [2], it was shown that an asymptotic analysis of Lanczos can be used to decrease the expense by estimating the elements of $T_K$in terms of the coefficients of a differential operator (neglecting the lower-order terms in the polynomial ω). This implementation of the asymptotic analysis of Lanczos was extremely beneficial in eliminating the high frequency error of the solution, but did nothing to eliminate the low frequency error. In [2] Cibotarica et al. found that after using a Fast Fourier Transform to split the solution into high and low frequency parts, the asymptotic analysis of Lanczos could be applied to eliminate high frequency error, while standard Krylov projection could be used to eliminate the low frequency error.

In summary, adding the asymptotic analysis of Lanczos as well as rapid node estimation for Gaussian quadrature produces the following framework for KSS to compute u^{n+1} from u_n:

- Apply the Lanczos algorithm to L_N using initial vector $\vec{u}^n$ and compute the eigenvalues of T_K (the matrix obtained from Lanczos).
- Apply the Lanczos algorithm to L_N using initial vector e_ω, then use the asymptotic analysis of Lanczos to estimate the eigenvalues of the resulting matrix $T_K(\omega)$.
- Combine the eigenvalues obtained from the previous steps to obtain the block Gaussian quadrature nodes.
- We can compute $\hat{u}^{n+1}$ (the Fourier coefficients of the approximte solution) and then use a Fast Fourier Transform to obtain the approximate solution $\vec{u}^{n+1}$.

Chapter 3
Multigrid Methods

Multigrid methods are an effective tool for solving a linear system that arise from the spatial discretization of elliptic PDEs, such as

$$A\vec{x} = \vec{b}, \tag{3.1}$$

in conjunction with any iterative method. This is because most iterative methods possess a "smoothing property" that is highly useful for eliminating high frequency error, but leave the low frequency error relatively unaffected. The implementation of Multigrid with an iterative method is a simple, yet effective, solution to these types of problems.

Multigrid methods solve linear equations like (3.1) by restricting an initial guess to a coarser grid (a grid with twice the spacing of the original), solving on the coarse grid, then interpolating the approximate solution back to the fine grid using appropriate restriction and interpolation operators. When combining Multigrid-like methods with an existing iterative method, then coarse grids can also be used to correct the approximate solution obtained from the iterative method. This is done by obtaining the residual on the fine grid, restricting the residual to a coarser grid, using a relaxation method on the residual, then interpolating back to the fine grid to use the newly obtained residual to correct the approximate solution. This is called coarse grid correction. Coarse grid correction is so effective because the low frequency error that the iterative method does not eliminate on a fine grid "looks" more oscillatory (higher in frequency) on a coarse grid. Briggs, Henson, and McCormick demonstrated this very clearly in [1] by using the following example.

Let k be the wave number and n be the number of grid points with spacing h between them. The components can be shown as

$$\omega^h_{k,2j} = \sin\left(\frac{2jk\pi}{n}\right) = \sin\left(\frac{jk\pi}{\frac{n}{2}}\right) = \omega^{2h}_{k,j}. \tag{3.2}$$

It should be noted that the superscripts indicate the grid that each vector is defined on. Therefore ω^h is defined on the finest grid and ω^{2h} is defined on the next coarsest grid.

If $k = 4$, $n = 12$, and $1 \leq k < \frac{n}{2}$, then Figure 3.1 from [1] shows a representation of (3.2) on the fine grid and the corresponding projection of (3.2) onto the coarse grid. Even on the next coarsest grid, the frequency of the wave appears higher than in the fine grid, which allows the iterative method to be more effective at eliminating the error.

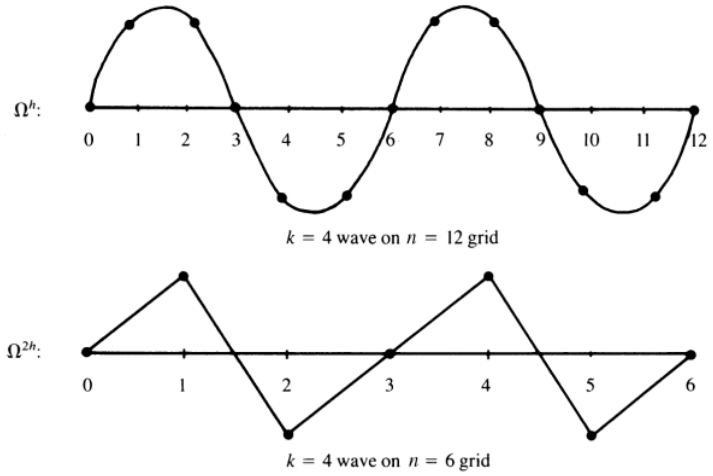

Figure 3.1: *This figure shows a comparisson between (3.2) on the fine grid (top) and on the coarse grid (bottom).*

3.1 Multigrid Algorithm

To implement Multigrid with an iterative method that has the "smoothing property", we need two operators: a restriction operator and an interpolation operator. The restriction operator will transfer a vector from a grid to the next coarsest grid and the interpolation operator will transfer a vector from a grid to the next finest one. It should be noted that the coarse grid has twice the spacing of the next finest grid. According to [1], this is the standard practice because in most cases there is not indication of any benefit to using any other grid spacing.

The most obvious way to "restrict" a vector to a coarser grid is by injection. This method defines the restriction operator, I_h^{2h}, as

$$I_h^{2h}\vec{v}^h = \vec{v}^{2h} \tag{3.3}$$

where $\vec{v}$ is the approximate solution of (3.1) and $\vec{v}_j^{2h} = \vec{v}_{2j}^h$. This indicates that the j^{th} node on the coarse grid is the same as the $2j^{th}$ node on the fine grid (e.g. the 3^{rd} node on the coarse grid is equal to the 6^{th} node on the next finest grid). Therefore, this method obtains the coarse grid vector directly from the fine grid vector. Another way to 'restrict' a vector to the next coarsest grid is by a method called full weighting. In this method, the coarse grid vectors are obtained from the weighted averages of the vectors at neighboring fine grid points. Therefore, the restriction operator is defined the same as in (3.3), but

$$\vec{v}_j^{2h} = \frac{1}{4}(\vec{v}_{2j-1}^h + 2\vec{v}_{2j}^h + \vec{v}_{2j+1}^h). \tag{3.4}$$

To interpolate, or prolong, back to a fine grid, an interpolation operator is needed. The interpolation operator is defined as

$$I_{2h}^{h}\vec{v}^{2h} = \vec{v}^{h}, \tag{3.5}$$

where

$$\vec{v}_{2j}^{h} = \vec{v}_{j}^{2h} \tag{3.6}$$

$$\vec{v}_{2j+1}^{h} = \frac{1}{2}(\vec{v}_{j}^{2h} + {}_{j+1}^{2h}). \tag{3.7}$$

Therefore, to interpolate back to a fine grid from a coarse grid, the coarse grid vectors are mapped back to the even fine grid nodes and the odd fine grid nodes are the averages of the coarse grid vectors on either side.

The multigrid algorithm for a linear system $A\vec{x} = \vec{b}$, where $\vec{v}$ is an approximation of $\vec{x}$ and A is an $n \times n$ matrix, is

Smooth $A^h\vec{v}^h = \vec{b}^h$ on the finest grid using initial approximation $\vec{v}^h$.
Compute $\vec{b}^{2h} = I_h^{2h}\vec{r}^h$ where I_h^{2h} is the restriction operator and $\vec{r}^h$ is the residual
found by $\vec{r}^h = \vec{b}^h - A^h\vec{v}^h$.
Smooth $A^{2h}\vec{v}^{2h} = \vec{b}^{2h}$ using initial guess $\vec{v}^{2h}$.
Compute $\vec{b}^{4h} = I_{2h}^{4h}\vec{r}^{2h}$.
$\vdots$
Solve $A^{kh}\vec{v}^{kh} = \vec{b}^{kh}$
$\vdots$
Correct $\vec{v}^{2h} \leftarrow \vec{v}^{2h} + I_{4h}^{2h}\vec{v}^{4h}$
Correct $\vec{v}^h \leftarrow \vec{v}^h + I_{2h}^h\vec{v}^{2h}$.

As one can see from the Multigrid algorithm, a user can choose how coarse of a grid to use. To ensure that the restriction method used is accurate, the residual on the fine grid should be compared to the residual on the coarse grid. The restriction to a coarse grid and prolongation back to fine grid performed in Multigrid is called a V-cycle, and Figure 3.2 from [6] visualizes this process.

3.2 Implementation of Multigrid

To implement a Multigrid algorithm like the one in the previous Section, a computer code must be written that can be tailored to the relaxation (or iterative) method that a user is implementing Multigrid with. Many Multigrid experts suggest that writing a highly modular

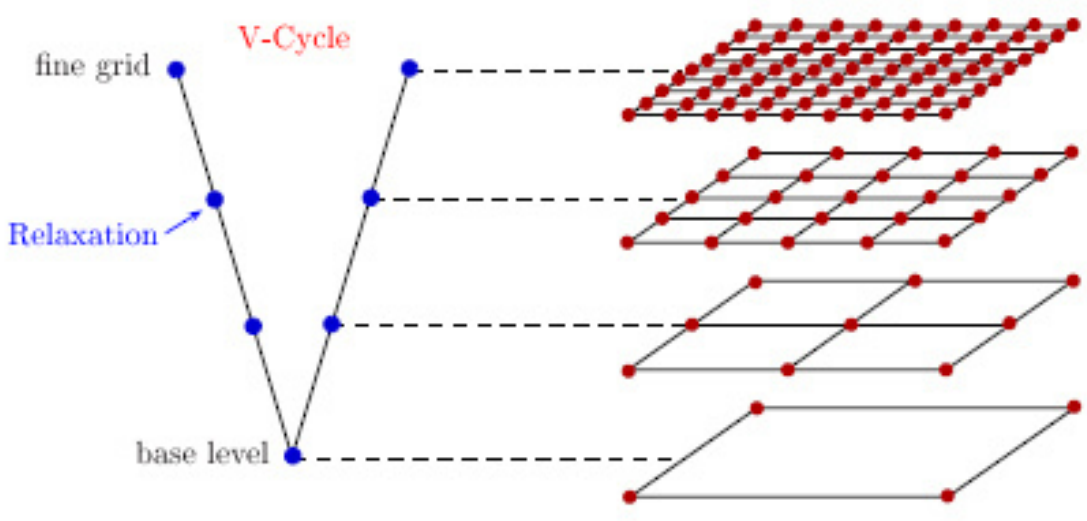

Figure 3.2: *The v-cycle as seen in [6]*

program is the most effective [1]. This allows the programs to evolve from simple relaxation methods to powerful tools as well as allowing the subroutines (restriction, interpolation, etc.) to be easily changed and corrected.

Consider a four level V-cycle applied to a one dimensional problem with 16 grid points. Figure 3.3 shows the changes to the program's data structure in which the V-cycle progresses. Each grid in the V-cycle needs two arrays, one for the approximation to the solution ($\vec{v}$) and one for the right side vectors ($\vec{f}$). Since boundary values are stored on each grid, then the k^{th} coarsest grid involves $2k+1$ points (the 1^{st} coarsest grid has 3 points, the 2^{nd} courses has 5, etc.).

As the program "descends" through the V-cycle to coarser and coarser grids, the program will fill the segments for the solution array and the residual vectors ($\vec{f}$) fill the right side array corresponding to the next coarsest grid. As the program "ascends" the V-cycle, only the solution array is overwritten (see, for example [1]).

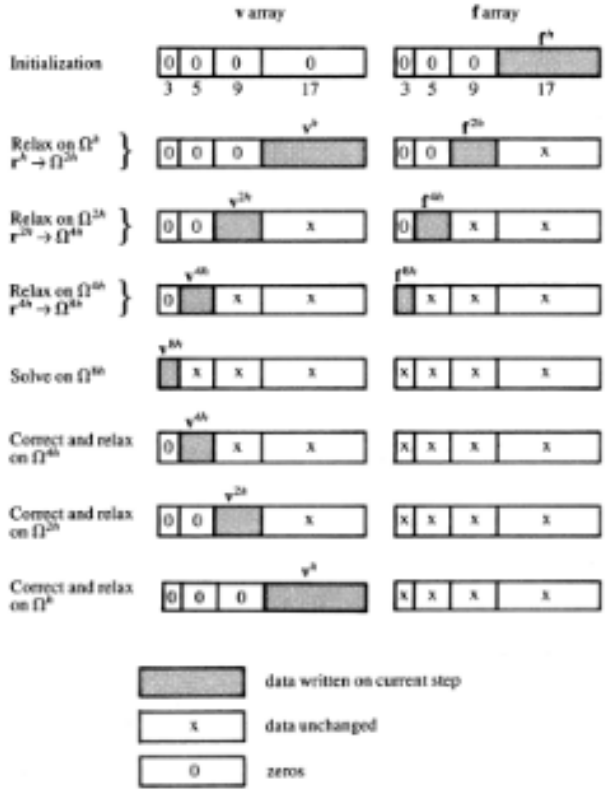

Figure 3.3: *An illustration from [1] of a four level V-cycle showing the changes in the data arrays. The $\vec{v}$ arrays hold the solutions vectors and the $\vec{f}$ arrays hold the right-side vectors.*

Chapter 4

Combining KSS and Multigrid Methods

Due to the work of Cibotarica, Lambers, and Palchak in [2], KSS methods are already highly effective at computing the high frequency components of the error. The low frequency components of the error though are still computed using standard Krylov projection as seen in [7].

As stated in the previous chapter, Multigrid is an effective way of eliminating low frequency error in linear systems that arise from the spatial discretization on elliptic partial differential equations which are time-independent (such as Laplace's equation). KSS methods are used to solve time dependent, variable coefficient PDEs. Therefore, to apply Multigrid to time-dependent problems, we must first generalize the method by defining the residual as $R = u_t + Lu$, then solving a non-homogeneous version of the PDE to obtain the error for the correction.

Since smoothing already takes place in the KSS method, from here we need three functions to implement Multigrid generalized for a time-dependent PDE:

1. a function to restrict the problem to a coarser grid (eg I_h^{2h})
2. a function to make a new Jacobian on the coarse grid
3. and a function to Interpolate back to the fine grid (eg I_{2h}^h).

To make a restriction function for a time-dependent PDE, the residual must first be reshaped from a vector to an $n \times n$ matrix B if the PDE is on a 2-D Domain. Then, the residual can be restricted to the coarse grid matrix, C, as seen in the following equation:

$$C(i,j) = \frac{1}{4}(B(2i-1,2j-1) + B(2i-1,2j) + B(2i,2j-1) + B(2i,2j)). \tag{4.1}$$

This is a method similar to the full weighting method as described in (3.4).

To make an interpolation function where a coarse grid matrix entry is mapped to the corresponding fine grid matrix entry, as well as the entries surrounding it, the following

algorithm can be followed:

$$B(2i-1,2j-1) = C(i,j) \tag{4.2}$$
$$B(2i-1,2j) = C(i,j) \tag{4.3}$$
$$B(2i,2j-1) = C(i,j) \tag{4.4}$$
$$B(2i,2j) = C(i,j). \tag{4.5}$$

KSS-MG proceeds as follows:

1. Use KSS as described in section 2 for all components of the solution (high and low frequency)

2. use Multigrid to eliminate low frequency error,

3. then combine the results of the previous two steps to obtain an approximate solution.

It should be noted that the Multigrid v-cycle will only descend to the next coarsest grid. Coarser grids could be used though, and further research could explore the effectiveness of this option.

4.1 Using KSS-MG to solve a Parabolic PDE

The Allen-Cahn equation is a parabolic, two dimensional PDE often used in mathematical physics. It is used to describe the reaction-diffusion of the separation of iron alloys. The Allen-Cahn equation is

$$u_t = \alpha \triangle u + u - u^3. \tag{4.6}$$

where α is a constant and for our problem $\alpha = 0.2$. Since the Allen-Cahn equation is nonlinear, then to use KSS with Multigrid (KSS-MG) to solve this problem, first we must linearize the equation. The linearized form is as follows,

$$u_t = \alpha \triangle u + (1 - 3y_0^2)u \tag{4.7}$$

where y_0 is the initial data. Also, there are homogeneous Neumann boundary conditions.

As stated in Section 3.1, Multigrid can be used to improve the accuracy of iterative methods that have the smoothing property. After KSS is applied, we are left with a relatively smooth error. To use Multigrid for residual correction, first the solution computed from KSS is used for the initial approximation and therefore used to find the residual, $R(x,t)$. To restrict the residual to a coarse grid, the residual is reshaped to an $n \times n$ matrix and restiction by full weighting can be implemented using (4.1).

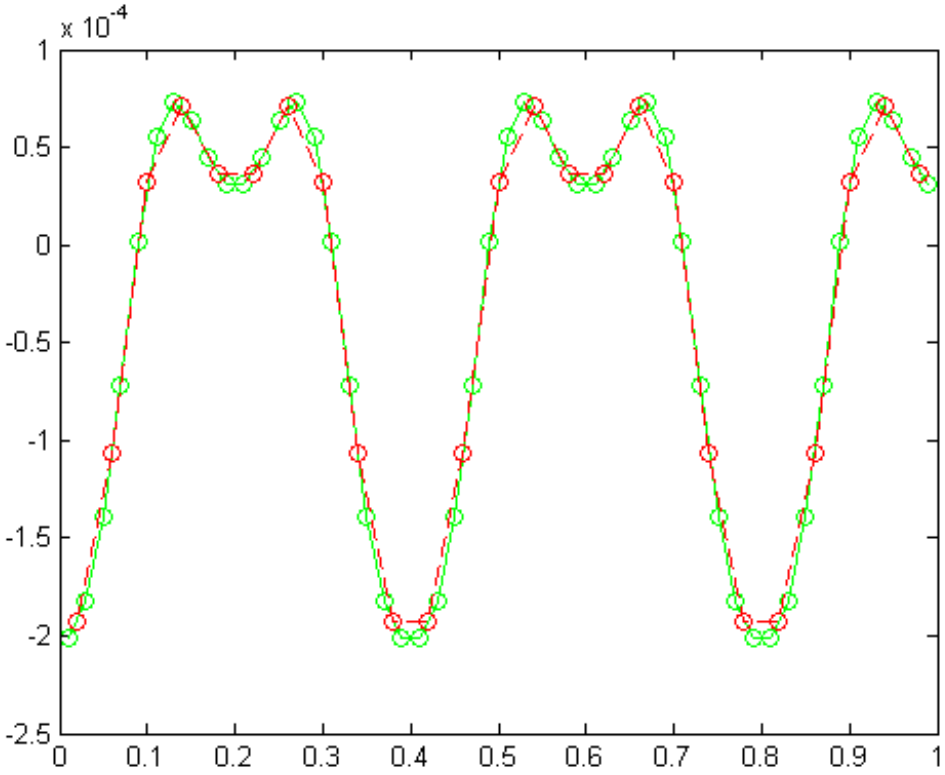

Figure 4.1: *This figure shows the fine grid residual in red and the coarse grid residual in green.*

To verify if the restriction operator appropriately maps the residual to the coarse grid, the residual on the coarse grid and the residual on the fine grid can be compared by visualizing both on the same grid. Figure 4.1 represents both the coarse and fine grid residuals for (4.6) with Neumann boundary conditions.

Once the residual is restricted to the coarse grid the differential operator L must be restricted to the coarse grid. The operator L is defined as

$$L = \alpha\triangle + (1 - 3y_0^2). \tag{4.8}$$

Then we can solve the non-homogeneous equation

$$e_t = \alpha\triangle e + (1 - 3y_0^2)e + R(x,t) \tag{4.9}$$

where e_t is the error, $e(x,0)$ is the initial condition and $e(x,0) = e_0 = 0$. Also the residual is $R(x,t) = u_t - Lu$. It follows that

$$e = e^{Lt}e_0 + \int_0^t e^{L(t-s)}R(x,s)\,ds. \tag{4.10}$$

Using Gaussian quadrature rules, we know the integral in (4.10) is approximately equal to

$$\int_0^{\Delta t} e^{L(\Delta t - s)} R(x,s)ds \approx \sum_k w_k e^{L(\Delta t - s_k)} R(x,s_k) \tag{4.11}$$

where s_k are the Gauss-Legendre points, transformed to the interval $[0,\Delta t]$ and w_k are the weights transformed to the same interval.

The newly obtained approximation can then be interpolated back to the fine grid where it will be used to correct the solution. To interpolate back to the fine grid, (4.5) is used, where a coarse grid matrix entry is mapped to the corresponding fine grid matrix entry as well as the entries surrounding it.

4.2 Using KSS-MG to solve a Hyperbolic Problem

Consider the hyperbolic PDE

$$u_{tt} = Lu, \qquad \text{on } (0,2\pi) \times (0,\infty), \tag{4.12}$$

$$u(x,0) = f(x), \qquad u_t(x,0) = g(x), \quad 0 < x < 2\pi. \tag{4.13}$$

To apply KSS-MG to solve a hyperbolic problem such as this one, first a spatial discretization of the differential operator L must be obtained so that a representation of the solution operator can be obtained. The solution operator can be expressed as a matrix of functions of the operator L:

$$\begin{bmatrix} \cos(\sqrt{-L}\triangle t) & \frac{1}{\sqrt{-L}}\sin(\sqrt{-L}\triangle t) \\ -\sqrt{-L}\sin(\sqrt{-L}\triangle t) & \cos(\sqrt{-L}\triangle t) \end{bmatrix}. \tag{4.14}$$

Therefore, we can use KSS-MG to solve for the solution as well as the first derivative with respect to t as follows:

$$\begin{bmatrix} u \\ u_t \end{bmatrix}_{n+1} = \begin{bmatrix} \cos(\sqrt{-L}\triangle t) & \frac{1}{\sqrt{-L}}\sin(\sqrt{-L}\triangle t) \\ -\sqrt{-L}\sin(\sqrt{-L}\triangle t) & \cos(\sqrt{-L}\triangle t) \end{bmatrix} \begin{bmatrix} u \\ u_t \end{bmatrix}_n.$$

The residual, R, computed at various times is

$$R = u_{tt} - Lu,$$

then the error used to update the solution is

$$e = \int_0^{\Delta t} \begin{bmatrix} \cos(\sqrt{-L}\triangle t - s) & \frac{1}{\sqrt{-L}}\sin(\sqrt{-L}\triangle t - s) \\ -\sqrt{-L}\sin(\sqrt{-L}\triangle t - s) & \cos(\sqrt{-L}\triangle t - s) \end{bmatrix} \begin{bmatrix} 0 \\ R(s) \end{bmatrix} ds. \tag{4.15}$$

The entries of (4.15) indicate which functions are the integrands in the Riemann-Stieltjes integrals that are used to compute the Fourier coefficients of the solution [9].

Consider an equation similar to the linearized form of the Allen-Cahn equation (4.7):

$$u_{tt} = \alpha \triangle u + (-3y_0^2)u \tag{4.16}$$

with Neumann boundary conditions as well. To implement KSS-MG to solve this problem, a similar process is followed as when solving (5.2). First, the differential operator for any solution u becomes

$$Lu = \alpha \Delta u + (-3y_0^2)u. \tag{4.17}$$

Then a spatial discretization of the differential operator L allows us to obtain a representation of the solution operator, as seen by (4.14). Then KSS-MG can be applied in a similar manner.

Chapter 5

Numerical Results

5.1 Numerical Results

In this section, the effectiveness of KSS with Multigrid (KSSMG) will be demonstrated. The follwing approaches will be compared:

- KSS, as described in Chapter 2
- KSSMG, as described in Chapter 4
- Krylov Projection (KP), as described in [7]
- KSS-EPI, as described in [2].

5.2 The Parabolic Problem

We first demonstrate the effectiveness of all four methods when solving a the parabolic problem from Chapter 4:

$$u_t = \alpha \triangle u + (1 - 3y_0^2)u. \tag{5.1}$$

In Table 5.1 the time elapsed (per time step) and the relative error calculated for that time step is shown for both grid sizes $N = 50$ and $N = 150$ for each test method (KSS-MG, KSS, KP, and KSS-EPI). In terms of accuracy, KSS-MG is far superior to the other test methods for both grid sizes. As the grid size increases from 50 to 150 in Figure 5.1 to 5.2, we can see a significant decrease in accuracy for both KSS-EPI and Krylov projection whereas the accuracy for KSS and KSS-MG stay relatively the same. Comparatively, the percent increase in computational time between grid sizes for KSS is much larger than the percent increase for KSS-MG. This implies that for even larger grid sizes, KSS-MG may be more efficient than the other test methods, though further numerical experiments would have to be performed to validate this theory.

Method	$\triangle t$	50 Grid Points		150 Grid Points	
		Time Elapsed (in seconds)	Relative Error	Time Elapsed (in seconds)	Relative Error
KSS-MG	0.2	0.0625	6.9647E-07	0.3437	5.0451E-07
	0.1	0.1093	6.1328E-09	0.5156	4.2637E-09
	0.05	0.1875	7.2752E-10	1.5625	7.3492E-10
	0.025	0.21875	9.4966E-11	2.8593	9.8773E-11
	0.0125	0.5312	2.8411E-11	5.8281	7.718 E-11
KSS	0.2	0.0312	7.1374E-06	0.1093	7.3640E-06
	0.1	0.0312	8.2972E-07	0.2187	8.6024E-07
	0.05	0.0468	1.0589E-07	0.3437	1.1012E-07
	0.025	0.0937	1.1476E-08	0.7968	1.1937E-08
	0.0125	0.0937	1.2216E-09	1.4218	1.2666E-09
KP	0.2	0.0156	1.6409E-04	0.0937	1.6595E-04
	0.1	0.0781	2.8841E-05	0.2968	1.9227E-06
	0.05	0.0937	6.1232E-07	0.6093	6.1411E-06
	0.025	0.1562	8.7526E-07	1.0312	6.6878E-06
	0.0125	0.2500	5.7278E-07	2.1093	5.5827E-06
KSS-EPI	0.2	0.0625	1.6409E-04	0.0312	1.6595E-04
	0.1	0.0625	8.5766E-06	0.234375	2.0453E-05
	0.05	0.109375	4.5584E-07	0.5312	3.2762E-05
	0.025	0.15625	5.4245E-09	1.125	4.5564E-07
	0.0125	0.3125	6.4457E-09	2.2812	4.6628E-08

Table 5.1: *Comparisson of the efficiency and accuracy of each method for the parabolic case.*

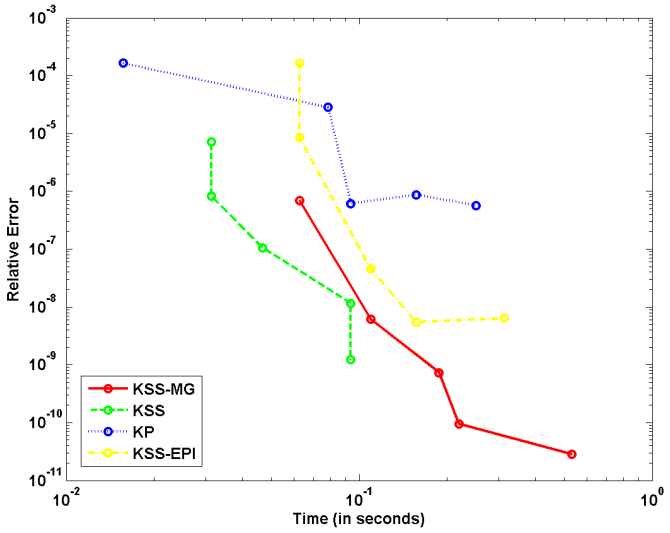

Figure 5.1: *Time of each timestep for each method vs error with grid size N=50. The blue dotted curve represents Krylov Projection, the red solid curve represents KSS-MG, the yellow dash-dot curve represents KSS-EPI and the green dashed curve is KSS.*

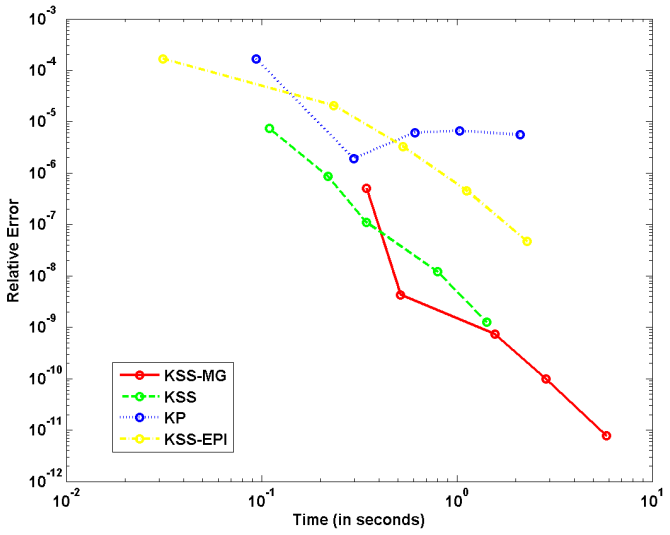

Figure 5.2: *Time of each timestep for each methodvs error with grid size N=150. The blue dotted curve represents Krylov Projection, the red solid curve represents KSS-MG, the yellow dash-dot curve represents KSS-EPI and the green dashed curve is KSS.*

5.3 The Hyperbolic Problem

We now demonstrate the effectiveness of all four methods when solving a the hyperbolic problem from Chapter 4:

$$u_{tt} = Lu, \qquad \text{on } (0,2\pi) \times (0,\infty), \tag{5.2}$$

$$u(x,0) = f(x), \qquad u_t(x,0) = g(x), \quad 0 < x < 2\pi. \tag{5.3}$$

In Table 5.1 the time elapsed (per time step) and the relative error calculated for that time step is sown for both grid sizes $N = 50$ and $N = 150$ for each test method (KSS-MG, KSS, KP, and KSS-EPI). We can see from Figure 5.3 and Figure 5.4 that all four methods yield similar efficiency (computation time), yet KSS-MG yields much higher accuracy, especially at the larger grid size. As the grid size increases from $N = 50$ per dimension to $N = 150$ the accuracy of Krylov projection and KSS-EPI decreased while the accuracy of KSS and KSS-MG increased. Comparitively, the percent increase of computational time was less for KSS-MG than for KSS. As with the parabolic case, this implies that for even larger grid sizes, KSS-MG may be more efficient than the various test approaches seen in Table 5.2. Further numerical experiments would have to be performed to confirm this.

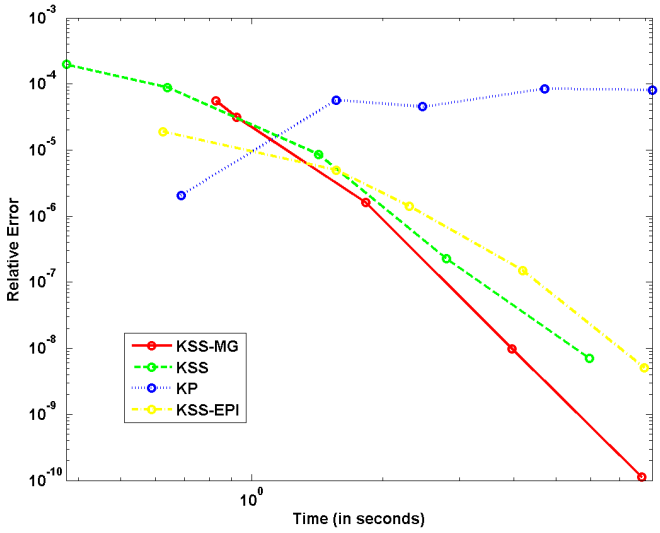

Figure 5.3: *Time of each timestep vs error with grid size N=50 (per dimension). The blue dotted curve represents Krylov Projection, the red solid curve represents KSS-MG, the yellow dash-dot curve represents KSS-EPI and the green dashed curve is KSS.*

Method	$\triangle t$	50 Grid Points		150 Grid Points	
		Time Elapsed (in seconds)	Relative Error	Time Elapsed (in seconds)	Relative Error
KSS-MG	2	0.8281	5.5349E-05	35.4218	5.5914E-05
	1	0.9218	3.1532E-05	71.7343	1.37679E-05
	0.5	1.8281	1.5991E-06	136.5156	2.03821E-06
	0.25	3.9531	9.8831E-09	278.9218	6.66E-09
	0.125	7.8437	1.121E-10	634.9062	1.76E-11
KSS	2	0.375	1.9670E-04	32.6093	2.0072E-04
	1	0.6406	8.8408E-05	64.9531	1.0044E-04
	0.5	1.4218	8.6060E-06	135.6875	3.3587E-06
	0.25	2.7968	2.2603E-07	274.0781	8.3558E-08
	0.125	5.9531	6.9945E-09	511.7968	6.385E-10
KP	2	0.6875	2.0655E-06	32.5	1.4269E-04
	1	1.5625	5.6772E-05	61.8906	1.3131E-04
	0.5	2.4687	4.5361E-05	128.0625	1.0472E-04
	0.25	4.7031	8.4548E-05	262.5781	1.1705E-04
	0.125	8.2968	8.1277E-05	534.4843	8.5461E-05
KSS-EPI	2	0.625	1.8973E-05	55.0468	4.0456E-05
	1	1.5625	4.9768E-06	173.5625	3.9417E-05
	0.5	2.2968	1.4233E-06	151.8281	8.4689E-05
	0.25	4.1875	1.5063E-07	317.1093	1.0330E-05
	0.125	7.9531	5.1046E-09	666.4531	1.1814E-07

Table 5.2: *Comparisson of the efficiency and accuracy of each method for the hyperbolic case.*

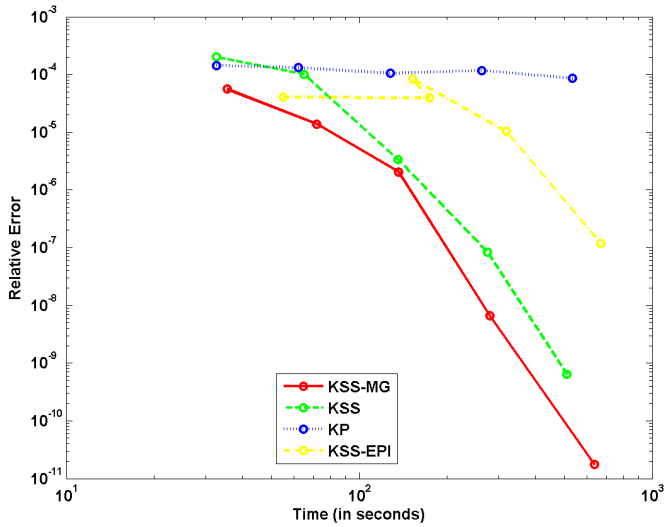

Figure 5.4: *Time of each timestep vs error with grid size N=150 (per dimension). The blue dotted curve represents Krylov Projection, the red solid curve represents KSS-MG, the yellow dash-dot curve represents KSS-EPI and the green dashed curve is KSS.*

Chapter 6

Conclusion

It has been shown that the combination of KSS and Multigrid in the manner described in chapter 4 is incredibly efficient at solving time-dependent variable coefficient PDEs. The effectiveness of KSS at eliminating high frequency error and the speed of Multigrid at eliminating low frequency error combined makes KSS-MG, by far, superior to other similar methods.

By implementing Multigrid techniques with KSS methods, we have demonstrated that the resulting KSS-MG methods have much higher accuracy than standard KSS methods (and KSS-EPI methods) as well as Krylov Projection in solving parabolic PDEs. It has also been shown that in the case of hyperbolic problems, KSS-MG is more efficient than the other methods as well.

Future work on the combination of Multigrid and KSS is needed to fully explore the effectiveness of this method. Topics for further reasearch are varied and include generalizing KSS-MG to solve a wider variety of problems as well as examining the effectiveness of grid coarseness. In this thesis, only the next coarsest grid was used. An efficient way to use any number of corrections must be developed to fully understand the effect of coarse grid correction in KSS-MG. In the future, the Multigrid method in general will be examined and we hope to examine the effectiveness of implementing KSS-like ideas into a standard Multigrid method to develop more scalable methods for solving time-dependent PDE on non-rectangular domains, using Algebraic Multigrid (AMG).

With these future advancements, it is hopeful that more efficient and accurate methods will be developed for time-dependent variable-coefficient PDEs.

BIBLIOGRAPHY

[1] William Briggs, Van Emden Henson, and Steve F. McCormick. *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics, Philidelphia, PA 19104, 2000.

[2] A. Cibotarica, J. V. Lambers, and E. M. Palchak. Solution of nonlinear time-dependent pde through componentwise approximation of matrix functions. *Journal of Computational Physics*, 2016-to appear.

[3] Gene H. Golub and Gerard Meurant. *Matrices, Moments and Quadrature with Applications*. Princeton University Press, 2010.

[4] G.H. Golub and R. Underwood. The block lanczos method for computing eigenvalues. *Mathematical Software III*, pages 361–377, 1977.

[5] B. Gustafsson, H.O. Kreiss, and J. Oliger. *Time-Dependent Problems and Difference Methods*. Wiley, New York, 1995.

[6] Daniel Hermann, Kurt Busch Meikel Frank, and Peter WÃűlfle. Photonic band structure computations. *Optics Express*, 8:167–172, 2001.

[7] Marlis Hochbruck and Christian Lubich. Approximations to the matrix exponential operator. *Numer. Anal.*, 34:1911–1925, 1996.

[8] J. V. Lambers. Enhancement of krylov subspace spectral methods by block lanczos iteration. *Electronic Transactions on Numerical Analysis*, 31:86–109, 2008.

[9] J. V. Lambers. A spectral time-domain method for computational electrodynamics. *Adv. Appl. Math. Mech.*, 1:781–798, 2009.

[10] E. M. Palchak, A. Cibotarica, and J. V. Lambers. Solution of time-dependent pde through rapid estimation of block gaussian quadrature nodes. *Linear Algebra and its Applications*, 468:233–259, 2015.