Spring 2019

Enhancement of Krylov Subspace Spectral Methods Through the Use of the Residual

Haley Dozier
University of Southern Mississippi

ENHANCEMENT OF KRYLOV SUBSPACE SPECTRAL METHODS THROUGH THE USE OF THE RESIDUAL

by

Haley Renee Dozier

A Dissertation
Submitted to the Graduate School,
the College of Arts and Sciences,
and the School of Mathematics and Natural Sciences
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Approved by:

Dr. James Lambers, Committee Chair
Dr. Haiyan Tian
Dr. Zhifu Xie
Dr. Huiqing Zhu

Dr. James Lambers	Dr. Bernd Schroeder	Dr. Karen S. Coats
Committee Chair	Director of School	Dean of the Graduate School

May 2019

ABSTRACT

Depending on the type of equation, finding the solution of a time-dependent partial differential equation can be quite challenging. Although modern time-stepping methods for solving these equations have become more accurate for a small number of grid points, in a lot of cases the scalability of those methods leaves much to be desired. That is, unless the time-step is chosen to be sufficiently small, the computed solutions might exhibit unreasonable behavior with large input sizes. Therefore, to improve accuracy as the number of grid points increases, the time-steps must be chosen to be even smaller to reach a reasonable solution.

Krylov subspace spectral (KSS) methods are componentwise, scalable, methods used to solve time-dependent, variable coefficient partial differential equations. The main idea behind KSS methods is to use an interpolating polynomial with frequency dependent interpolation points to approximate a solution operator for each Fourier coefficient.

This dissertation will discuss two techniques that were developed to eliminate error in the low frequency components of the solution computed using KSS methods. These two methods are a multigrid inspired technique (coarse grid residual correction) and developing a step size controller using the residual as an error approximation (adaptive time stepping).

ACKNOWLEDGMENTS

First and foremost I would like to thank my adviser Dr. James V. Lambers for his patience, guidance, and support throughout my years at the University of Southern Mississippi. As my teacher and mentor, Dr. Lambers has taught me more than I could ever give him credit for here. He has shown me, by his example, what a good person and mathematician should be.

I am additionally grateful to all of those whom I have had the pleasure of working with at USM. Each member of my dissertation committee (Dr. Lambers, Dr. Tian, Dr. Xie, and Dr. Zhu) has provided me with incredible insight and has led to my dissertation being a stronger academic work. Additionally I would like to thank my coworkers and friends Brianna Bingham, Corwin Stanford and Dr. Amber Sumner for being a constant support system throughout my academic career.

Lastly, I would like to thank the members of my family. I am so thankful to have the full support of my parents Jerry and Tami Dozier, my grandmother Frances Dozier, and my sister Francie Sutherland. This dissertation would not have been possible without their unconditional love, patience and support.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure

LIST OF TABLES

Table

LIST OF ABBREVIATIONS

ATS	-	Adaptive Time Stepping
CGRC	-	Coarse Grid Residual Correction
EPI	-	Exponential Propagation Iterative
FFT	-	Fast Fourier Transform
IFFT	-	Inverse Fast Fourier Transform
KP	-	Krylov Projection
KSS	-	Krylov Subspace Spectral
ODE	-	Ordinary Differential Equation
PDE	-	Partial Differential Equation

NOTATION AND GLOSSARY

General Usage and Terminology

The blackboard fonts are used to denote standard sets of numbers: $\mathbb{R}$ for the field of real numbers, $\mathbb{C}$ for the complex field. Bold font lower case letters or letters with an arrow above them are used to indicate vectors, i.e. $\vec{v}$ or $\mathbf{v}$. Functions are denoted by the letter f or greek letters. Norms are denoted using double pairs of lines, i.e. $||\cdot||$. The symbols $||\cdot||_\infty$, $||\cdot||_2$, $||\cdot||_F$ are the infinity-norm, the 2-norm, and the Frobenius-norm respectively.

Chapter 1

Background

Partial differential equations, or PDEs, are equations that contain unknown multivariate functions as well as their partial derivatives. A few well known partial differential equations are:

$u_t = u_{xx}$	1-D Heat Equation
$u_t = u_{xx} + u_{yy}$	2-D Heat Equation
$u_{tt} = u_{xx} + u_{yy} + u_{zz}$	3-D Wave Equation
$u_{rr} + \frac{1}{r}u_r + \frac{1}{r^2}u_{\theta\theta} = 0$	Laplace's Equation in Polar Coordinates

with the notation $u_t = \frac{\partial u}{\partial t}$, $u_{tt} = \frac{\partial^2 u}{\partial t^2}$, etc. [6]

There are many different ways to classify the types of partial differential equations. This is important because the way we solve these equations is often based on how a partial differential equation is classified. Some of the basic classifications as described in [6] are

1. *Order.* The order of any partial differential equation is based on the highest partial derivative in that equation. For example, all of the well-known partial differential equations listed above are second order because the highest partial derivative in those equations is the second partial derivative (i.e. u_{xx}, u_{tt},...).

2. *Number of Variables.* A PDE can be classified by the number of independent variables in the equation. For example, the 1-D heat equation listed above has two independent variables (x and t) while the 2-D heat equation has 3 independent variables (x, y, and t).

3. *Linearity.* Linear partial differential equations have dependent variables (and the dependent variable's partial derivatives) that occur linearly (e.g. they do not have a power greater than 1 and aren't multiplied together). If the partial differential equation is not linear, then it is classified as non-linear.

4. *Homogeneity.* The general form for a linear, second-order partial differential equation is

$$Au_{xx} + Bu_{xy} + Cu_{yy} + Du_x + Eu_y + Fu = G(x,y) \tag{1.1}$$

where A through G can be either constants or functions. The equation (1.1) is homogeneous if $G(x,y) = 0$. If $G(x,y) \neq 0$, then the equation is called non-homogeneous.

5. *Parabolic, Hyperbolic, and Elliptic.* There are 3 main classification of linear partial differential equations. We will use equation (1.1) to classify these types of equation.

   (a) Parabolic equations satisfy the property $B^2 - 4AC = 0$. An example of a parabolic partial differential equation is the 1-D heat equation: $u_t = u_{xx}$.

   (b) Hyperbolic equations satisfy the property $B^2 - 4AC > 0$. An example of a hyperbolic partial differential equation is the 1-D wave equation: $u_{tt} = u_{xx}$.

   (c) Elliptic equations satisfy the property $B^2 - 4AC < 0$. An example of a elliptic partial differential equation is Laplace's equation: $u_{xx} + u_{yy} = 0$.

6. *Type of Coefficient* There are two types of coefficients that a partial differential equation can have: constant or variable. For example if all of the coefficients A, B, C, D, E, or F in equation (1.1) are constant, then the partial differential equation is a constant coefficient equation. Otherwise the partial differential equation is classified as a variable coefficient equation.

7. *Time Dependence.* Time dependent partial differential equations involve the partial derivatives of the dependent variable with respect to time. Otherwise a partial differential equation is called time independent.

A partial differential equation can have multiple classifications. For example, the equation

$$u_{tt} = c^2 u_{xx} + s(x,t) \tag{1.2}$$

with appropriate boundary and initial conditions can be classified as a linear, constant coefficient, time-dependent, non-homogeneous, hyperbolic PDE. In this dissertation, time dependent variable coefficient partial differential equations will be discussed.

There are a multitude of phenomena in the world that are modeled by time dependent PDEs such as sound, heat, fluid flow, and electrodynamics among many others. For example, consider the 1-D parabolic PDE

$$u_t = \alpha^2 u_{xx}, \quad 0 < x < 1, \quad 0 < t < \infty \tag{1.3}$$

with boundary conditions

$$u(0,t) = 0 \tag{1.4}$$

$$u(1,t) = 0 \tag{1.5}$$

and initial condition

$$u(x,0) = \phi(x). \tag{1.6}$$

This PDE can be used to describe diffusion of heat through an iron rod.

To solve this type of PDE analytically typically the method of Separation of Variables is used. This method has the following algorithm:

- *Step 1:* Find all solutions of equation (1.3) of the form $u(x,t) = X(x)T(t)$, where $X(x)$ and $T(t)$ are functions that rely solely on x and t respectively, and then substituting $X(x)T(t)$ into the original equation. Then a general solution form can be found after solving for $X(x)$ and $T(t)$. For example, using the problem stated above where $\alpha = 1$ for simplicity:

$$u_t = uxx \tag{1.7}$$

$$X(x)T'(t) = X''(x)T(t) \tag{1.8}$$

$$\frac{T'(t)}{T(t)} = \frac{X''(x)}{X(x)}. \tag{1.9}$$

  Equation (1.9) implies that both $\frac{T'(t)}{T(t)}$ and $\frac{X''(x)}{X(x)}$ must be equal to a fixed constant k since x and t are independent of each other (the left side of the equation depends on t and the right depends on x). If we write

$$\frac{T'(t)}{T(t)} = \frac{X''(x)}{X(x)} = k \tag{1.10}$$

  then this implies that

$$T'(t) - kT(t) = 0 \tag{1.11}$$

$$X''(x) - kX(x) = 0 \tag{1.12}$$

  and if we let $k = -\lambda^2$ the general solutions to these two ordinary differential equations are

$$T(t) = Ae^{-\lambda^2 t} \tag{1.13}$$

$$X(x) = A\sin(\lambda x) + B\cos(\lambda x) \tag{1.14}$$

  where A and B are arbitrary constants. Therefore a solution of of the PDE $u_t = u_{xx}$ is any function of the form

$$u(x,t) = X(x)T(t) \tag{1.15}$$

$$= (e^{-\lambda^2 t})(A\sin(\lambda x) + B\cos(\lambda x)) \tag{1.16}$$

- *Step 2:* Since equation (1.16) describes general solutions to the PDE (1.3), the next step is to find a subset of solutions that satisfy the boundary conditions (in the example problem the boundary conditions are $u(0,t) = 0$ and $u(1,t) = 0$). [6]
  To find the subset of solutions that satisfy the boundary conditions, first the solutions must be substituted into the boundary conditions

$$\begin{aligned} u(0,t) &= (e^{-\lambda^2 t})(A\sin(0) + B\cos(0)) = 0 && (1.17)\\ &= Be^{-\lambda^2 t} = 0 && (1.18) \end{aligned}$$

  which implies that $B = 0$, and

$$\begin{aligned} u(1,t) &= (e^{-\lambda^2 t})(A\sin(\lambda) + B\cos(\lambda)) = 0 && (1.19)\\ &= (e^{-\lambda^2 t})(A\sin(\lambda)) = 0 && (1.20) \end{aligned}$$

  which implies $\sin(\lambda) = 0$ (if $A = 0$ there would be a zero solution). Therefore the subset of solutions to the PDE (1.3)that satisfy the boundary conditions is

$$u_n(x,t) = A_n e^{-\lambda_n^2 t}\sin(\lambda_n x) \tag{1.21}$$

  where λ_n is the solutions to $\sin(\lambda) = 0$ (i.e. $\lambda_n = \pm n\pi$ for $n = 1,2,3,...$). It should be noted that the solution can also be written as the sum

$$u(x,t) = \sum_{n=1}^{\infty} A_n e^{-\lambda_n^2 t}\sin(\lambda_n x). \tag{1.22}$$

- *Step 3:* The last step in this method of solving PDEs is to ensure that the solution found in step 2 also satisfies the initial condition of the PDE. For example, this can done by substituting the sum in equation (1.22) into the initial condition $u(x,0) = \phi(x)$ as follows

$$\begin{aligned} u(x,0) &= \phi(x) = \sum_{n=1}^{\infty} A_n e^0 \sin(\lambda_n x) && (1.23)\\ &= \phi(x) = \sum_{n=1}^{\infty} A_n \sin(\lambda_n x). && (1.24) \end{aligned}$$

  By expanding the sum and substituting $n\pi$ for λ, it can be shown that

$$\phi(x) = A_1\sin(\pi x) + A_2\sin(2\pi x) + A_3\sin(3\pi x) + ... \tag{1.25}$$

  Then multiplying each side of the equation by $\sin(m\pi x)$ (m being an arbitrary integer), integrating from 0 to 1, then solving for A_m, returns the formula

$$A_m = 2\int_0^1 \phi(x)\sin(m\pi x)dx. \tag{1.26}$$

This implies that the solution of the PDE given in (1.3) with the given initial and boundary conditions is

$$u(x,t) = \sum_{n=1}^{\infty} A_n e^{-\lambda_n^2 t} \sin(\lambda_n x) \tag{1.27}$$

where the coefficients are of the same form as equation (1.26).

In practice, the exact solution in not always so easy, or so fast, to find. For example, consider a PDE of the form

$$u_t = (p(x)u_x)_x \tag{1.28}$$

where $p(x)$ is a function. If we use the separation of variables technique here the problem becomes much more complex. If we substitute $X(x)T(t)$ into the equation for u then we can obtain

$$X(x)T'(t) = (p(x)X'(x)T(t))_x \tag{1.29}$$

and then use the product rule to get

$$X(x)T'(t) = p'(x)X'(x)T(t) + p(x)X''(x)T(t). \tag{1.30}$$

To separate the variables we can divide both sides by $X(x)T(t)$ to get

$$\frac{T'(t)}{T(t)} = \frac{p'(x)X'(x) + p(x)X''(x)}{X(x)} = k \tag{1.31}$$

where k is an arbitrary constant and we will set k to equal λ^2. From (1.31) we can create the two ODEs

$$T'(t) + \lambda^2 T(t) = 0 \tag{1.32}$$

$$p'(x)X'(x) + p(x)X''(x) + \lambda^2 X(x) = 0. \tag{1.33}$$

Equation (1.32) is a standard type ODE and so has a solution of $T(t) = Ae^{-\lambda^2 t}$ but equation (1.33) does not have an obvious solution. The solution to this dilemma is to use a numerical method to approximate the solution of the equation.

1.1 Numerical Methods for Solving Time Dependent Partial Differential Equations

Consider a time-dependent, variable coefficient PDE, such as

$$u_t + Lu = 0, \qquad 0 < x < 2\pi, \quad t > 0 \tag{1.34}$$

$$u(x,0) = f(x), \qquad 0 < x < 2\pi, \tag{1.35}$$

where L is a second order, self-adjoint, positive definite differential operator, such as a Sturm-Liouville differential operator. This type of problem often poses difficulties for both implicit and explicit time-stepping methods due to the lack of scalability of these methods caused by stiffness. That is, unless the chosen time-step is sufficiently small, the computed solutions might exhibit nonphysical behavior with large input sizes [8].

Although there are many numerical methods used to solve time-dependent PDEs, this section will focus on finite difference methods and a subclass of numerical methods called spectral methods. These two methods will be focused on because finite difference is one of the most well-known numerical methods used and the method that will be discussed in future chapters is a type of spectral method.

1.1.1 Finite Difference Methods

Given a time-dependent initial value problem, the main idea behind explicit finite difference methods is that a PDE can be replaced by its finite difference approximation and then can be solved for the solution at a specific time using the solution at previous times. This process of using the solution of an equation at a previous time to determine the solution of an equation at a later time is often referred to as "time-stepping" or "time-marching". Finite difference methods (both explicit and implicit) are very popular methods because they tend to be easy to implement with high accuracy.

Consider a PDE similar to the one in equation (1.34) with periodic boundary conditions where L is the second-order differential operator $-\frac{d^2}{dx^2}$. If a uniform grid is defined on $[0, 2\pi]$ with N grid points of equal spacing then the grid-points are located at $x_j = jh$ where $h = \frac{2\pi}{N}$. If we represent $u(x)$ as a vector $\vec{u}$ with N components, then we can approximate the action of L on $u(x_j)$ using the stencil

$$Lu \approx -\frac{u(x_{j+1}) - 2u(x_j) + u(x_{j-1})}{\triangle x^2}. \tag{1.36}$$

If a Crank-Nicholson method is used as the type stepping method to obtain the value of $\vec{u}$ at the next time step $(t + \triangle t)$, then the system that needs to be solved can be represented by

$$\left(I + \frac{\triangle t}{2}A\right)\vec{u}(t + \triangle t) = \left(I - \frac{\triangle t}{2}A\right)\vec{u}(t) \tag{1.37}$$

where

$$A = \frac{-1}{h^2}\begin{bmatrix} -2 & 1 & 0 & \cdots & \cdots & 0 & 1 \\ 1 & -2 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & \cdots \\ & & \ddots & \ddots & \ddots & & \\ 0 & 0 & \cdots & \cdots & 1 & -2 & 1 \\ 1 & 0 & \cdots & \cdots & 0 & 1 & -2 \end{bmatrix} \tag{1.38}$$

and $\triangle t$ is the length of the time-step [11].

Although this system has to be solved at each time step, the finite difference method is still computationally efficient for this problem because of the structure of the matrix A. However, with large grid sizes N, the finite difference method can be very expensive due to the large amount of time steps needed for the finite difference method to be accurate.

1.1.2 Spectral Methods

The main idea behind most spectral methods for time-dependent PDEs is that a solution of a PDE can be represented as a sum of basis functions (often Fourier basis functions) with time-dependent coefficients and can be substituted into the original equation to yield a system of ordinary differential equations that can be relatively easy to solve. Unlike finite difference methods that require a large number of computations (large number of time steps) for an accurate solution of a problem with significant spatial or temporal variation, spectral methods allow for less computational expense. If the exact solution is smooth, spectral methods can still be highly accurate with large time steps. If the exact solution is not smooth, or there are variable-coefficients, using large time steps becomes more difficult.

There are two types of approaches to Fourier spectral methods; Galerkin methods and collocation methods. In Galerkin spectral methods the residual is orthogonal to the space in which the approximation to the solution is in. If the approximate solution $\tilde{u}(x,t)$ to the simple PDE

$$u_t(x,t) = u_x(x,t) \tag{1.39}$$

is defined by

$$\tilde{u}(x,t) = \sum_{\omega=-N}^{N} \hat{u}(\omega,t)e^{i\omega x} \tag{1.40}$$

where $-\frac{N}{2}+1 \leq \omega \leq \frac{N}{2}$ if N is even and $-\frac{N-1}{2} \leq \omega \leq \frac{N-1}{2}$ if N is odd and the residual R is defined by

$$R = \tilde{u}_t(x,t) - \tilde{u}_x(x,t) \tag{1.41}$$

then in the Galerkin approach it is required that

$$\int_{-\pi}^{\pi} Re^{-i\omega x}dx = \int_{-\pi}^{\pi} (\tilde{u}_t(x,t) - \tilde{u}_x(x,t))e^{-i\omega x}dx = 0 \tag{1.42}$$

for all $-N \leq \omega \leq N$. This approach is very effective when solving a linear PDE, but becomes much more complex and difficult to solve when the PDE is very nonlinear [6].

Collocation spectral methods are a little more complex than Galerkin methods but can still be effective to solve highly nonlinear PDEs. In the collocation approach to solving

PDEs, the residual is required to vanish at a set of points rather than be orthogonal to the space that the approximate solution is in. Consider the 1-D wave equation

$$u_{tt}(x,t) = u_{xx}(x,t) \tag{1.43}$$

where the approximate solution has a similar form to (1.40). The residual R can be defined as

$$R = u_{tt}(x,t) - u_{xx}(x,t) \tag{1.44}$$

and for collocation methods, we require the residual to satisfy the condition

$$u_{tt}(x_j,t) - u_{xx}(x_j,t) = 0 \tag{1.45}$$

for some set of points x_j where $-N \leq j \leq N$.

A Fourier spectral method for solving the time-dependent PDE in (1.34) proceeds as follows. Assuming the solution $u(x,t)$ is sufficiently smooth (the solution and its derivative is piecewise continuous), an approximation to the solution $\tilde{u}(x,t)$ can be written as the Fourier series

$$\tilde{u}(x,t) = \frac{1}{\sqrt{2\pi}} \sum_{\omega=-\infty}^{\infty} e^{i\omega x} \hat{u}(\omega,t) \tag{1.46}$$

where ω is the wave number for each Fourier component and $\hat{u}(\omega,t)$ are the Fourier coefficients. The approximation to the solution can then be substituted into equation (1.34) to obtain

$$\sum_{\omega=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \hat{u}_t(\omega,t) e^{i\omega x} + L \frac{1}{\sqrt{2\pi}} e^{i\omega x} \hat{u}(\omega,t) = 0. \tag{1.47}$$

Let λ_ω be the eigenvalues of the operator L. Due to the orthogonality of the functions $e^{i\omega x}$, the decoupled system of ordinary differential equations

$$\hat{u}_t(\omega,t) - \lambda_\omega \hat{u}(\omega,t) = 0 \tag{1.48}$$

can be obtained. Equation (1.48) can then be solved using any numerical ordinary differential equation solver [6].

1.2 Krylov Subspace Spectral Methods

Krylov subspace spectral (KSS) methods were developed by James Lambers in 2003 for his doctoral dissertation [11]. KSS methods were created to solve time-dependent, variable-coefficient problems in an efficient and effective manner. Like some other spectral methods, the solution of a stiff PDE is found by spatially discretization the PDE and then solving

the system of ODEs that arises from the spatial discretization. This process involves the computation of matrix function vector products that are of the form

$$\vec{w} = \phi(At)\vec{b} \tag{1.49}$$

where t is dependent on the time step, $\vec{b}$ is a vector, ϕ is a smooth function and A is an ill-conditioned matrix.

The main idea behind Krylov subspace spectral methods is to use an interpolating polynomial with frequency-dependent interpolation points to approximate a solution operator (ϕ) for each Fourier coefficient. As a result, Krylov subspace spectral methods exhibit a high order of accuracy and stability, and in some cases Krylov subspace spectral methods have been recently proven to be unconditionally stable [1, 12]. A detailed description of Krylov subspace spectral methods can be found in Chapter 2.

1.3 Enhancement of Krylov Subspace Spectral Methods

Due to the increase of computing power of computers as well as the availability of data, scientists and engineers have been able to develop increasingly accurate mathematical models to describe much of the phenomena found in the natural world. Because of the progressively more complex mathematical problems, more and more sophisticated numerical methods are needed to accurately and efficiently solve these problems.

This dissertation will predominantly focus on the optimization of Krylov subspace spectral methods so that it can be implemented to solve today's numerical problems. This dissertation will provide two approaches to optimizing the results obtained from using Krylov subspace spectral methods through the use of the residual as an error estimation instead of using traditional error estimation techniques. These approaches will be described in detail in Chapter 5, but an overview can be found below.

1.3.1 Coarse Grid Residual Correction

Until recently, one of the downfalls of using Krylov subspace spectral methods was that the Lanczos algorithm had to be performed on each component. This dilemma resulted in a high computational cost until the work done by A. Cibotarica et al. [1] where it was shown that although all nodes needed for Gaussian quadrature were dependent on frequency, half depend primarily on the solution. This led to an asymptotic analysis of Lanczos on the frequency dependent nodes by using the coefficients of a differential operator which was shown to be beneficial to the high frequency components of the solution but not to the lower frequency components. This dilemma raised the question, what technique can be

implemented to eliminate error in the low frequency components of the approximate solution. This dissertation proposes that implementing a multigrid inspired technique, specifically coarse grid residual correction, can eliminate low frequency error in a cost effective way.

Although multigrid methods are generally used to solve linear systems that arise from the spatial discretization of elliptic PDEs which are time-independent, we propose a technique that is inspired by the multigrid algorithm and is generalized to a time dependent case.

1.4 Modified Adaptive Time-stepping

Many time-stepping methods use equally spaced time-steps to compute the solution of an equation. In some cases, this approach is ineffective. For example:

- when the solution is highly oscillatory and the chosen time step is too large to sufficiently resolve the solution,

- when the chosen time-step is too small for a relatively smooth solution resulting in computational effort being wasted.

This dilemma raises the question: how can an appropriate time step size ($\triangle t$) be chosen for a problem?

Adaptive time-stepping techniques are used frequently in numerical methods to ensure stability as well as decrease the computational expense of certain methods [16]. These adaptive methods usually use estimates of the local truncation error to determine whether a certain step size ($\triangle t$) should be modified. With many numerical methods, adaptive time stepping can be implemented simply by using the local truncation error of the method at a time step then adjusting the time step based on a scaling factor that is calculated from how large (or how small) the error is. For Krylov subspace spectral methods the approximate local truncation error is very difficult to approximate which means that the scaling factor for adaptive time stepping cannot be found. This dissertation will explore the effectiveness of using the residual as an error estimate for adaptive time stepping.

1.5 Outline

The outline of this dissertation is as follows: Chapters 2, 3, and 4 will present overviews of Krylov Subspace Spectral methods, coarse grid residual correction, and adaptive time-stepping, respectively. Chapter 5 will discuss residual correction techniques using coarse

grid residual correction and adaptive time-stepping. Numerical results will be examined in Chapter 6 and conclusions are presented in Chapter 7.

Chapter 2

Krylov Subspace Spectral Methods

We start by examining the initial value problem

$$\frac{\partial u}{\partial t}(x,t) + L(x,D)u(x,t) = 0, \quad 0 < x < 2\pi, \quad t > 0 \tag{2.1}$$

$$u(x,0) = f(x) \tag{2.2}$$

with periodic boundary conditions $u(0,t) = u(2\pi,t)$. Here, the operator $L(x,D)$ is the Sturm-Liouville differential operator defined by

$$Lu = -(p(x)u_x)_x + q(x)u, \tag{2.3}$$

where $p(x)$ and $q(x)$ are both functions of x. For simplicity $u(x,t)$ will be written as u, $L(x,D)$ will be written as L, and $p(x)$ as well as $q(x)$ will be written as p and q respectively.

A common way to solve this type of partial differential equation is to first approximate the solution with its Fourier series, i.e.

$$u(x,t) = \frac{1}{\sqrt{2\pi}} \sum_{\omega=-\infty}^{\infty} e^{i\omega x} \hat{u}(\omega,t) \tag{2.4}$$

where ω is the wave number for each Fourier component and $\hat{u}(\omega,t)$ is the Fourier coefficient of each component. Using the standard inner product,

$$\langle f,\ g \rangle = \int_0^{2\pi} \overline{f(x)} g(x) dx, \tag{2.5}$$

we can represent the Fourier coefficients, $\hat{u}(x,t)$, of the solution on $(0,2\pi)$ by

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

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

As stated in the previous chapter, the fundamental idea behind Krylov subspace spectral methods is to independently approximate all Fourier coefficients of the solution using an approximation of the exact solution operator that is tailored to each Fourier coefficient. Therefore, to approximate the exact solution operator, first (2.6) is spatially discretized to obtain

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

where L_N is a $N \times N$ symmetric positive definite matrix obtained from spatial discretization of L, and $\vec{x}$ is a vector of equally spaced points in $[0, 2\pi)$.

Since L is a second order, self-adjoint, symmetric positive definite operator, we know from [7] that L_N can be written as

$$L_N = Q\Lambda Q^T, \tag{2.8}$$

where Q is an orthonormal matrix with columns that are the normalized eigenvectors of L_N and Λ are the eigenvalues ($a = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n = b$). This implies that given a smooth function ϕ,

$$\phi(L_N) = Q\phi(\Lambda)Q^T. \tag{2.9}$$

It follows that

$$\begin{aligned} \vec{u}^H \phi(L_N)\vec{v} &= \vec{u}^H Q\phi(\Lambda)Q^T\vec{v} && (2.10)\\ &= \vec{u}^H \vec{q}_j \phi(\Lambda)\vec{q}_j^H \vec{v} && (2.11)\\ &= \sum_{j=1}^{n} \phi(\lambda_i)\vec{u}^H \vec{q}_j \vec{q}_j^H \vec{v} && (2.12) \end{aligned}$$

where $\vec{q}_j$ are the orthonormal eigenvectors of L_N [7, 1].

In their work *Matrices, Moments, and Quadrature*, Golub and Meurant showed that this type of bilinear form can be represented as a Riemann-Stieltjes integral

$$\vec{u}^H \phi(L_N)\vec{v} = \int_a^b \phi(\lambda)\, d\alpha(\lambda) \tag{2.13}$$

where $\alpha(\lambda)$ is defined by

$$\alpha(\lambda) = \begin{cases} 0 & \text{if } \lambda < a \\ \sum_{j=1}^{i} \vec{u}^H \vec{q}_j \vec{q}_j^H \vec{v} & \text{if } \lambda_i \leq \lambda \leq \lambda_{i-1} \\ \sum_{j=1}^{N} \vec{u}^H \vec{q}_j \vec{q}_j^H \vec{v} & \text{if } b \leq \lambda. \end{cases} \tag{2.14}$$

Although there are many ways to approximate the integral in equation (2.13), here Gaussian quadrature is used because the weights are guaranteed to be positive if the measure $\alpha(\lambda)$ is positive and increasing and it has a high degree of accuracy. In fact, an n-point quadrature rule can be constructed to obtain an exact result for polynomials of up to degree $2n-1$ with an appropriate choice of nodes and weights for the quadrature rule.

Case 1: $\vec{u} = \vec{v}$

As shown in [11], the nodes and weights for Gaussian quadrature can be found using the Lanczos algorithm applied to L_N with initial vectors $\vec{u}$ and $\vec{v}$. The Lanczos algorithm is as follows:

$\vec{r}_0 = \vec{u}$
$\vec{x}_0 = 0$
for $j = 1,2,...,n$ **do**
$\beta_{j-1} = \left\|\vec{r}_{j-1}\right\|_2$
$\vec{x}_j = \vec{r}_{j-1}/\beta_{j-1}$
$\alpha_j = \vec{x}_j^T A\vec{x}_j$
$\vec{r}_j = (A - \alpha_j I)\vec{x}_j - \beta_{j-1}\vec{x}_j - 1$
end

where vectors $x_1, x_2, ..., x_n$ are called Lanczos vectors. Then Gaussian quadrature can be used to approximate (2.13) as follows:

$$\int_a^b \phi(\lambda) d\alpha(\lambda) = \sum_{j=1}^{K} \phi(\lambda_j) w_j + error \tag{2.15}$$

where λ_j are the nodes and w_j are the weights. This quadrature rule is exact for polynomials of degree up to $2K-1$.

Case 2: $u \neq v$

In the case where $\vec{u} \neq \vec{v}$, the weights, w_j, are not always positive real numbers. This occurrence can destabilize the quadrature rule as shown in [2]. In this case, we consider a block approach:

$$[\vec{u}\ \vec{v}]^H \phi(L_N)[\vec{u}\ \vec{v}]. \tag{2.16}$$

We can represent this matrix as the Riemann-Stieltjes integral

$$[\vec{u}\ \vec{v}]^H \phi(L_N)[\vec{u}\ \vec{v}] = \int_a^b \phi(\lambda)\ d\mu(\lambda) = \begin{bmatrix} \vec{u^H}\phi(L_N)\vec{u} & \vec{u^H}\phi(L_N)\vec{v} \\ \vec{v}^H\phi(L_N)\vec{u} & \vec{v}^H\phi(L_N)\vec{v} \end{bmatrix}, \tag{2.17}$$

where $\mu(\lambda)$ is a 2×2 matrix with entries of the form $\alpha(\lambda)$ from (2.14) as demonstrated in [7]. Then a quadrature rule is used to approximate the integral (2.17). This leads to the approximation

$$\int_a^b \phi(\lambda)\ d\mu(\lambda) \approx \sum_{j=1}^{k} \phi(\lambda_j)\vec{v}_j\vec{v}_j^H + error \tag{2.18}$$

where λ_j is a scalar and each $\vec{v}_j$ is a 2-vector. Each node for Gaussian quadrature is an eigenvalue of the matrix T_k that is obtained from the block Lanczos algorithm.

The algorithm for block Lanczos, as described in [5], 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.

After applying block Lanczos to L_N using initial block $[\vec{u}\ \vec{v}]$, we obtain the block tridiagonal matrix with 2×2 blocks

$$T_K = \begin{bmatrix} M_1 & B_1^T & & & \\ B_1 & M_2 & B_2^T & & \\ & \ddots & \ddots & \ddots & \\ & & & B_{K-1} & M_K \end{bmatrix}, \tag{2.19}$$

where each B_j is upper triangular and both M_j and B_j are built using block Lanczos. As previously stated, the eigenvalues of the matrix T_K are used as the nodes for Gaussian quadrature, and $\vec{v}_j \vec{v}_j^H$ are the matrix-valued weights for Gaussian quadrature. It should be noted that $\vec{v}_j$ consists of the first two components of the normalized eigenvector corresponding to λ_j.

A time step of block Krylov subspace spectral methods proceeds as follows. We begin by defining

$$R_0(\omega) = [\vec{\hat{e}}_\omega\ \vec{u}^n] \tag{2.20}$$

where $\vec{\hat{e}}_\omega$ is the discretization of $\frac{1}{\sqrt{2\pi}} e^{i\omega\vec{x}}$ and $\vec{u}^n$ is the computed solution at time t_n (these are $\vec{u}$ and $\vec{v}$ in (2.16) above). The QR factorization of (2.20) leads to

$$R_0(\omega) = X_1(\omega) B_0(\omega) \tag{2.21}$$

with

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

and

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

where

$$\vec{u}_\omega^n = \vec{u}^n - \vec{\hat{e}}_\omega \vec{\hat{e}}^H \vec{u}^n = \vec{u}^n - \vec{\hat{e}}_\omega \vec{\hat{u}}(\omega, t_n). \tag{2.22}$$

Block Lanczos is then applied to the discretized operator, L_N, with initial 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 (2.19). 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

$$[\vec{\hat{u}}^{n+1}]_\omega = [B_0^H E_{12}^H e^{-T_K(\omega)\triangle t} E_{12} B_0]_{12}, \qquad E_{12} = [\ \vec{e_1}\ \vec{e_2}\ ]. \tag{2.23}$$

By applying an inverse Fast Fourier Transform (IFFT) to the vector of Fourier coefficients, we obtain the vector $\vec{u}^{n+1}$, which approximates the solution $u(x, t_{n+1})$ [13].

In [13] it was shown this algorithm has local temporal accuracy of $O(\triangle t^{2K-1})$ for the parabolic problem and in [14] it was shown to have local temporal accuracy $O(\triangle t^{4K-2})$ for the second-order wave equation. To reduce the computational expense of performing Lanczos on each component, an asymptotic analysis of Lanczos can be performed. This was first shown in [5] by Palchak, et al. and then expanded on in [1] by Cibotarica, et al. The process is as follows.

Recall that $\vec{\hat{u}}$ is the discrete Fourier transform of $\vec{u}$ on a uniform N-point grid. Krylov subspace spectral methods use the block (2.20) for each ω as the initial block for block Lanczos to obtain T_K. It has been shown in [5] that for the PDE shown in (1.34), every non-zero, off diagonal entry of the blocks M_j and B_j converges to zero as $|\omega| \to \infty$. It can be shown that the matrix obtained from applying block Lanczos iteration to (2.20) 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.

This leads to a decoupling effect such that for a finite ω the Gaussian quadrature nodes can be estimated from the eigenvalues of these smaller matrices, which has been shown to be dramatically faster [5]. It should be noted that in the subsequent discussion, the nodes that are computed from applying “non-block" Lanczos to the spectral discretization of L, L_N, with initial vector $\vec{u}^n$ 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 Krylov subspace spectral method. The rest of the nodes that are computed from applying “non-block" Lanczos to $\vec{\hat{e}}_\omega$ will be referred to as “frequency-dependent nodes".

After obtaining the matrices from Lanczos iteration on the separate columns of R_0 and computing the frequency independent nodes, the frequency-dependent nodes must be computed for each ω. Here, an asymptotic analysis of Lanczos is used 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 [1] 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 $\hat{e}_\omega$, 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 approximate solution) and then use a Fast Fourier Transform to obtain the approximate solution $\vec{u}^{n+1}$.

Chapter 3

Coarse Grid Residual Correction

In numerical analysis, multigrid methods are a type of numerical method that are used to solve linear systems that can arise from the spatial discretization of elliptic PDEs. For example, a multigrid method in conjunction with an iterative method could be used to solve the linear equation

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

Iterative methods are often used along with multigrid methods because most iterative methods possess a "smoothing property" that effectively eliminates high frequency error but leaves low frequency error relatively unaffected.

The process in which multigrid methods solve linear equations like equation (3.1) is by first restricting some initial guess of the solution to a coarse grid (a grid with twice the spacing of the original), solving the equation on the coarse grid, and then finally interpolating the approximate solution back to the fine grid using restriction and interpolation operators. If a multigrid method is used with an existing iterative method, then the coarse grid can alternatively be used to correct the approximation to the solution found by implementing the iterative method. This process can be carried out by first obtaining an error approximation on the fine grid, restricting the error approximation to a coarse grid, using a relaxation method on the error approximation, then interpolating back to the fine grid to use the newly obtained error approximation to correct the approximate solution. This technique is called coarse grid correction. Coarse grid correction is often effective because the low frequency error on the fine grid that is largely unaffected by the iterative method "looks" more oscillatory (higher in frequency) on a 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.

3.1 Multigrid Algorithm

To implement multigrid after using an iterative method as a smoother, two operators are needed: a restriction operator and an interpolation operator. The restriction operator will transfer a vector from a fine grid to the next coarsest grid. The interpolation operator will

transfer a vector from a coarse grid to the next finest one. Generally the coarse grid has twice the spacing of the next finest grid [21].

Although there are many ways to restrict a vector to a coarse grid, the most obvious way is by a process called injection. This process involves defining the restriction operator, I_h^{2h}, as

$$I_h^{2h}\vec{v}^h = \vec{v}^{2h} \tag{3.2}$$

where $\vec{v}$ is the approximate solution of (3.1) found by the iterative method 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). This way the coarse grid vector is obtained directly from the fine grid.

Another process that is used to restrict a vector to the next coarsest grid is by full weighting. To implement full weighting the coarse grid vectors are obtained from the weighted averages of the fine grid vectors at neighboring points. Therefore we can define the restriction operator in a similar way to equation (3.2), but

$$\vec{v}_j^{2h} = \frac{1}{4}(\vec{v}_{2j-1}^h + 2\vec{v}_{2j}^h + \vec{v}_{2j+1}^h). \tag{3.3}$$

To interpolate back to a fine grid, an interpolation operator is needed. A general interpolation operator can be defined as

$$I_{2h}^h\vec{v}^{2h} = \vec{v}^h, \tag{3.4}$$

where

$$\vec{v}_{2j}^h = \vec{v}_j^{2h} \tag{3.5}$$

$$\vec{v}_{2j+1}^h = \frac{1}{2}(\vec{v}_j^{2h} + {}_{j+1}^{2h}). \tag{3.6}$$

Therefore, to interpolate to a fine grid the coarse grid vectors are mapped back to 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 the linear system 3.1, 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 $\vec{v}^{h} \leftarrow \vec{v}^{h} + I_{2h}^{h}\vec{v}^{2h}$.

In this algorithm the residual is used to obtain the approximation of the error.

Although in this discussion of multigrid we have concentrated on using a fine grid and then a coarse grid with twice the spacing of the fine grid, a user can actually choose how coarse of a grid to use. It is a standard practice to restrict to coarser grids with twice the spacing of the fine grid so to restrict to a grid with four times the spacing the restriction operator is applied twice. To ensure that the restriction method used is accurate, the residual on the finest grid can be compared to the residual on the coarsest grid. The restriction to a coarse grid and prolongation back to fine grid performed in multigrid is called a V-cycle.

Chapter 4

Adaptive Time Stepping

In general, algorithms for many time stepping methods use equally spaced time steps to numerically compute the solution for whichever type of equation that the method is trying to solve. This constant time step size is often chosen by the user and many texts have exercises that are based on comparing the outcome of the time stepping method using different time step sizes. It is often the case that a reader will find that a small time step size more accurately approximates the solution of the ordinary or partial differential equation listed in their exercise, but that accuracy comes with added computational expense (i.e. their program takes longer to run). In practice, using an arbitrarily selected constant step size is often an inappropriate approach to numerically solving differential equations. For explicit methods of solving partial differential equations, choosing a step size that is above the Courant-Friedrichs-Lewy (CFL) limit can cause the method not to converge to a solution. For this reason, many modern software packages and numerical solvers have a built in step size controller.

Step size controllers (adaptive time stepping algorithms) are often made to be dependent on the accuracy of the numerical method that it is being used with. A good adaptive time stepping algorithm in conjunction with a good error estimator can detect the when the solution is highly oscillatory and the chosen time step is too large to sufficiently resolve the solution or when the chosen time step is too small for a relatively smooth solution. For an adaptive time stepping algorithm to work properly, a good estimation of the error must be found. There are many approaches to estimating the error of a method. For simplicity, consider a first-order initial value problem of the form

$$y' = f(t,y) \tag{4.1}$$

$$y(t_0) = y_0 \tag{4.2}$$

where y is a real valued function in terms of t and $y' = \frac{dy}{dt}$. A simple method that can be used to solve an initial value problem of this type is a one step method (Euler's method for example). The main idea behind one step methods is that given the initial condition $y(t_0) = y_0$, the value of the solution y at the next step can be found using the following equation

$$y_{n+1} = y_n + h\phi(t_n, y_n; h), \quad n = 0, 1, ..., N-1 \tag{4.3}$$

where h is defined to be the step (or mesh) size and N is a positive integer that denotes the number of steps [16]. To determine the accuracy of this method, two common methods of error estimation in error analysis are

- **Global Error** $e_n = y(t_n) - y_n$
- **Local Truncation Error** $T_n = \frac{y(t_n+1)-y(t_n)}{h} - \phi(t_n, y(t_n), h)$.

It should be noted that $y(t_n)$ denotes the actual solution at t_n while y_n denotes the numerically approximated solution.

Since the methods listed above for computing global and local truncation error involve the exact solution, these methods are not useful when the exact solution is not known. One way to obtain the local truncation error of a one step method without using the exact solution is by using two one-step methods. Consider error estimation for one-step methods y_{n+1} and $\tilde{y}_{n+1}$ of order p and $p+1$ respectively. The local truncation error for each of these methods is

$$T_{n+1} = \frac{y(t_n+1)-y(t_n)}{h} - \phi_p(t_n, y(t_n), h) \tag{4.4}$$

$$\tilde{T}_{n+1} = \frac{\tilde{y}(t_n+1)-\tilde{y}(t_n)}{h} - \phi_{p+1}(t_n, \tilde{y}(t_n), h). \tag{4.5}$$

Assuming these methods are exact at time t_n, we can say that

$$T_{n+1} = \frac{y(t_n+1)-y(t_n)}{h} \tag{4.6}$$

$$\tilde{T}_{n+1} = \frac{y(t_n+1)-\tilde{y}(t_n)}{h} \tag{4.7}$$

and subtracting the two equations obtains

$$T_{n+1} - \tilde{T}_{n+1} = \frac{y(t_n+1)-y(t_n)}{h} - \frac{\tilde{y}(t_n+1)-\tilde{y}(t_n)}{h} \tag{4.8}$$

$$= \frac{(y(t_n+1)-y(t_n))-(\tilde{y}(t_n+1)-\tilde{y}(t_n))}{h} \tag{4.9}$$

$$= \frac{-y(t_n)+\tilde{y}(t_n)}{h}. \tag{4.10}$$

Since T_{n+1} is order p and $\tilde{T}_{n+1}$ is order $p+1$, the term of lower order can be neglected to obtain the simple error estimate of the pth-order accurate one-step method as seen in ([16]) is

$$T_{n+1} = \frac{\tilde{y}(t_n) - y(t_n)}{h}. \tag{4.11}$$

4.1 Derivation of an Adaptive Time Stepping Method

As stated before, adaptive time stepping techniques are used frequently in numerical methods to ensure stability as well as decrease the computational expense of certain methods. In general, these adaptive methods often use a user set tolerance as well as estimates of the local truncation error to determine if and how a certain step size (h) should be modified.

To modify the time step size h using the local trucation error of two one step methods (4.11), first the time step size h can be changed to qh for some scaling factor q. Then by multiplying the error by q^p we obtain the new local truncation error

$$|T_n(qh)| \approx |\frac{q^p}{h}(\tilde{y}_{n+1} - y_{n+1})|. \tag{4.12}$$

In adaptive time-stepping, the goal is to find a scaling factor q such that the local truncation error is less than some prescribed tolerance ε. To solve for q we can set 4.12 to be less than or equal to the prescribed tolerance and solve for q to obtain

$$q \leq \left(\frac{\varepsilon h}{\tilde{y}_{n+1} - y_{n+1}}\right)^{1/p}. \tag{4.13}$$

To avoid making the step size too large or too small a minimum and maximum step size should be set.

Chapter 5

Enhancement of Krylov Subspace Spectral Methods Through the Use of the Residual as an Error Estimator

Given the equation

$$f(\vec{x}) = \vec{b}, \tag{5.1}$$

in numerical analysis the residual, R is defined to be

$$R = \vec{b} - f(\vec{x}_0) \tag{5.2}$$

for some approximation $\vec{x}_0$ of the solution $\vec{x}$. The absolute error E is the difference between the value of the solution and the approximate solution, i.e.

$$E = \vec{x} - \vec{x}_0. \tag{5.3}$$

If the exact solution to equation (5.1) is unknown, the absolute error cannot be found but the residual can still be computed. This is a very simple example of a phenomenon in numerical analysis - how to measure the error of an approximate solution when the exact solution is unknown.

The residual is a useful tool for measuring error in cases when another error estimator is either too computationally expensive or hard to find. In this chapter, we explore the use of the residual as an error estimator with the purpose of (1) correcting the solution and (2) to adaptively change the time step size of the method. Although the residual is not equal to the local truncation error, the use of the residual for error approximation is useful tool for KSS methods. This is because KSS methods are used to solve the system of ODEs that arise from the spatial discretization of a PDE. This process involves matrix function vector products of the form

$$w = \phi(A\tau)b \tag{5.4}$$

where $\phi = e^{L\triangle t}$ is a smooth function, $b = u^n$ is solution from the previous time step, and $w = u^{n+1}$ is the solution at the current time step. If we define the residual at the current time step as

$$R_{n+1} = u_t^{n+1} - Lu^{n+1} \tag{5.5}$$

where

$$u^{n+1} = e^{L\triangle t}u^n + \mathcal{E}^n \tag{5.6}$$

where $\mathcal{E}^n$ is an error term, then it follows that

$$\begin{aligned} R_{n+1} &= \frac{d}{dt}(e^{L\triangle t}u^n + \mathcal{E}^n) - Lu^{n+1} && (5.7)\\ &= Le^{L\triangle t}u^n + \mathcal{E}_t^n - L(e^{L\triangle t}u^n + \mathcal{E}^n) && (5.8)\\ &= \mathcal{E}_t^n - L\mathcal{E}^n. && (5.9) \end{aligned}$$

This shows that the formula for the residual for KSS methods can be written in terms of the error. Therefore we are able to use the residual as an approximation of that error.

5.1 Frequency Analysis

Due to the work of Cibotarica, Lambers, and Palchak in [1], Krylov subspace spectral 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 [9]. To increase accuracy of the method, we first perform a frequency analysis to more accurately separate the high and low frequency components and to show that KSS can act as a smoother for the multigrid inspired technique - coarse grid residual correction.

In Chapter 2, Krylov space spectral methods were described for solving an equation of the form

$$u_t + Lu = 0 \tag{5.10}$$

for some differential operator L with some type of initial and boundary conditions. Also, recall from Chapter 2, the aim of Krylov subspace spectral methods is to approximate the solution of the partial differential equation using a Fourier series where the Fourier coefficients are found using an approximation to the exact solution operator that is tailored to each Fourier coefficient. If we define L_N as the discretization of the operator L and the approximation to the exact solution operator as

$$S(L_N;\triangle t) = e^{-L_N\triangle t} \tag{5.11}$$

then we can obtain the approximation by interpolation of the function $S(\lambda,\triangle t)$ at specified nodes where λ are the eigenvalues of L_N. The interpolation points can be found using block Lanczos iteration.

We begin our frequency analysis with analyzing the behavior of the block Lanczos algorithm. Recall, the algorithm for block Lanczos, as described in [5], 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.

Therefore to begin the first iteration of block Lanczos, we set $X_0 = 0$ and $R_0 = [e^{i\omega x}/\sqrt{2\pi}\ \ u^n]$. If we let $e^{i\omega x}/\sqrt{2\pi} = \hat{e}_\omega$ and u^n be a discretization of the solution at time t_n then the QR factorization of R_0 can be found by performing the following operations:

$$R_0 = X_1 B_0 \tag{5.12}$$

$$[a_1\ a_2] = [q_1\ q_2] \begin{bmatrix} r_{11}\ r_{12} \\ 0\ r_{22} \end{bmatrix} \tag{5.13}$$

where

$$r_{11} = ||a_1||_2 = ||\hat{e}_\omega||_2 = 1, \tag{5.14}$$

$$q_1 = \frac{a_1}{r_{11}} = \hat{e}_\omega, \tag{5.15}$$

$$r_{12} = \langle q_1, a_2 \rangle \approx \int_0^{2\pi} \frac{1}{\sqrt{2\pi}} e^{ß\omega x} u(x, t_n) dx = \hat{u}^n(\omega), \tag{5.16}$$

$$r_{22} = ||a_2 - r_{12} q_1||_2 = ||u^n - \hat{u}^n(\omega)\hat{e}_\omega||_2, \tag{5.17}$$

and

$$q_2 = \left|\left| \frac{a_2 - r_{12} q_1}{r_{22}} \right|\right|_2 = \frac{u^n - \hat{u}^n(\omega)\hat{e}_\omega}{||u^n - \hat{u}^n(\omega)\hat{e}_\omega||_2} = \frac{u_\omega^n}{||u_\omega^n||_2}. \tag{5.18}$$

It should be noted that $\hat{u}^n(\omega)$ is a coefficient of the Fourier interpolant of u^n and $u_\omega^n = u^n - \hat{u}^n(\omega)\hat{e}_\omega$. From the QR factorization of R_0 we obtain two matricies of the form

$$X_1 = [q_1\ \ q_2] = \left[\hat{e}_\omega\ \ \frac{u_\omega^n}{||u_\omega^n||_2} \right] \tag{5.19}$$

and

$$B_0 = \begin{bmatrix} r_{11}\ r_{12} \\ 0\ r_{22} \end{bmatrix} = \begin{bmatrix} 1 & \hat{u}^n(\omega) \\ 0 & ||\hat{u}^n(\omega)||_2 \end{bmatrix}. \tag{5.20}$$

Once X_1 and B_0 have been found, the matrices V and M_1 can be found.

$$V = L_N X_1 = \begin{bmatrix} L_N \widehat{e}_w & L_N \frac{u_w^n}{||u_w^n||} \end{bmatrix}$$

$$M_1 = X_1^H V = \begin{bmatrix} \widehat{e}_w^H \\ (\frac{u_w^n}{||u_w^n||})^H \end{bmatrix} \begin{bmatrix} L_N \widehat{e}_w & L_N \frac{u_w^n}{||u_w^n||} \end{bmatrix} = \begin{bmatrix} \widehat{e}_w^H L_N \widehat{e}_w & \widehat{e}_w^H L_N \frac{u_w^n}{||u_w^n||} \\ (\frac{u_w^n}{||u_w^n||})^H L_N \widehat{e}_w & (\frac{u_w^n}{||u_w^n||})^H L_N \frac{u_w^n}{||u_w^n||} \end{bmatrix}$$

Using $L_N = -p\triangle + Q$, where $Q = q(x)I$, $p = p(x)I$ and I is the identity matrix, M_1 can be simplified to

$$M_1 = \begin{bmatrix} -w^2 p + \overline{Q} & \widehat{\frac{L_N u_w^n}{||u_w^n||}} \\ \overline{\widehat{\frac{L_N u_w^n}{||u_w^n||}}} & (\frac{(u_w^n)^H L_N u_w^n}{||u_w^n||^2}) \end{bmatrix}$$

where $\overline{Q}$ is the average value of Q. From here, R_1, X_2, and B_1 can be found by performing the following operations:

$$R_1 = X_0 B_0^H - X_1 M_1 = \begin{bmatrix} L_N \widehat{e}_w & L_N \frac{u_w^n}{||u_w^n||} \end{bmatrix} - \begin{bmatrix} \widehat{e}_w * \frac{u_w^n}{||u_w^n||} \end{bmatrix} \begin{bmatrix} -w^2 p + \overline{Q} & \widehat{\frac{L_N u_w^n}{||u_w^n||}} \\ \overline{\widehat{\frac{L_N u_w^n}{||u_w^n||}}} & (\frac{(u_w^n)^H L_N u_w^n}{||u_w^n||^2}) \end{bmatrix}$$

$$= \begin{bmatrix} \widetilde{Q}\widehat{e}_w - \frac{u_w^n \overline{\widehat{L_N u_w^n}}}{||u_w^n||^2} & \frac{L_N u_w^n}{||u_w^n||} - \frac{\widehat{e}_w \widehat{L_N u_w^n}}{||u_w^n||} - \frac{u_w^n (u_w^n)^H L_N u_w^n}{||u_w^n||^3} \end{bmatrix}$$

Then from the QR factorization of R_1 we obtain X_2 and B_1.

$$R_1 = X_2 B_1 \begin{bmatrix} a_1 & a_2 \end{bmatrix} = \begin{bmatrix} q_1 & q_2 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}$$

where $a_1 = \widetilde{Q}\widehat{e}_w - \frac{u_w^n \overline{\widehat{L_N u_w^n}}}{||u_w^n||^2}$

$$a_2 = \frac{L_N u_w^n}{||u_w^n||} - \frac{\widehat{e}_w \widehat{L_N u_w^n}}{||u_w^n||} - \frac{u_w^n (u_w^n)^H L_N u_w^n}{||u_w^n||^3}$$

$$r_{11} = ||a_1|| = \left\| \widetilde{Q}\widehat{e}_w - \frac{u_w^n \overline{\widehat{L_N u_w^n}}}{||u_w^n||^2} \right\|$$

$$q_1 = \frac{a_1}{r_{11}} = \frac{\widetilde{Q}\widehat{e}_w - \frac{u_w^n \overline{\widehat{L_N u_w^n}}}{||u_w^n||^2}}{\left\| \widetilde{Q}\widehat{e}_w - \frac{u_w^n \overline{\widehat{L_N u_w^n}}}{||u_w^n||^2} \right\|}$$

$$r_{12} = \langle q_1, q_2 \rangle = \frac{\widetilde{Q}\widehat{e}_w - \frac{u_w^n \overline{\widehat{L_N u_w^n}}}{||u_w^n||^2}}{\left\| \widetilde{Q}\widehat{e}_w - \frac{u_w^n \overline{\widehat{L_N u_w^n}}}{||u_w^n||^2} \right\|} \cdot \frac{L_N u_w^n}{||u_w^n||} - \frac{\widehat{e}_w \widehat{L_N u_w^n}}{||u_w^n||} - \frac{u_w^n (u_w^n)^H L_N u_w^n}{||u_w^n||^3}$$

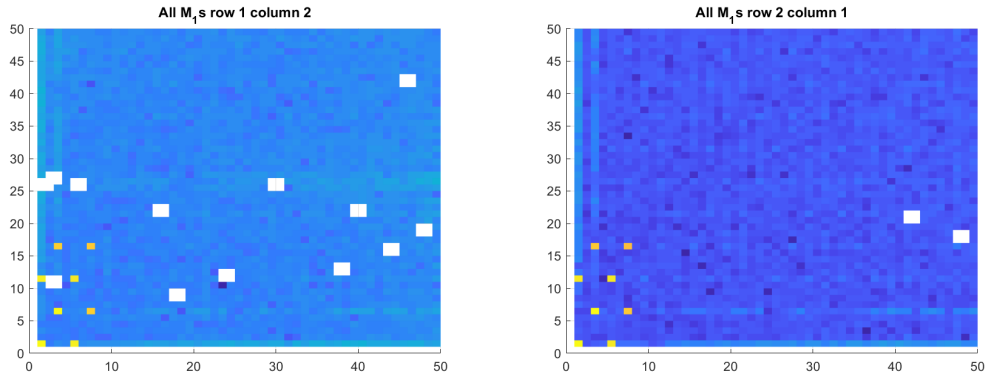

Figure 5.1: Entries of M_1 matrices obtained from Block Lanczos

$$r_{22} = \|a_2 - r_{12}q_1\| = \|a_2 - \langle q_1, a_2\rangle q_1\|$$

$$= \left\| \frac{L_N u_w^n}{||u_w^n||} - \frac{\widehat{e_w L_N u_w^n}}{||u_w^n||} - \frac{u_w^n (u_w^n)^H L_N u_w^n}{||u_w^n||^3} - \left(\frac{\widetilde{Q}\widehat{e}_w - \frac{u_w^n \widehat{L_N u_w^n}}{||u_w^n||^2}}{\left\| \widetilde{Q}\widehat{e}_w - \frac{u_w^n \widehat{L_N u_w^n}}{||u_w^n||^2} \right\|} \cdot \frac{L_N u_w^n}{||u_w^n||} - \frac{\widehat{e_w L_N u_w^n}}{||u_w^n||} - \frac{u_w^n (u_w^n)^H L_N u_w^n}{||u_w^n||^3} \right) \cdot \left(\frac{\widetilde{Q}\widehat{e}_w - \frac{u_w^n \widehat{L_N u_w^n}}{||u_w^n||^2}}{\left\| \widetilde{Q}\widehat{e}_w - \frac{u_w^n \widehat{L_N u_w^n}}{||u_w^n||^2} \right\|} \right) \right\|$$

and $q_2 = \frac{a_2 - r_{12}q_1}{r_{22}}$

$$= \frac{\frac{L_N u_w^n}{||u_w^n||} - \frac{\widehat{e_w L_N u_w^n}}{||u_w^n||} - \frac{u_w^n (u_w^n)^H L_N u_w^n}{||u_w^n||^3} - \left(\frac{\widetilde{Q}\widehat{e}_w - \frac{u_w^n \widehat{L_N u_w^n}}{||u_w^n||^2}}{\left\| \widetilde{Q}\widehat{e}_w - \frac{u_w^n \widehat{L_N u_w^n}}{||u_w^n||^2} \right\|} \cdot \frac{L_N u_w^n}{||u_w^n||} - \frac{\widehat{e_w L_N u_w^n}}{||u_w^n||} - \frac{u_w^n (u_w^n)^H L_N u_w^n}{||u_w^n||^3} \right) \cdot \left(\frac{\widetilde{Q}\widehat{e}_w - \frac{u_w^n \widehat{L_N u_w^n}}{||u_w^n||^2}}{\left\| \widetilde{Q}\widehat{e}_w - \frac{u_w^n \widehat{L_N u_w^n}}{||u_w^n||^2} \right\|} \right)}{\left\| \frac{L_N u_w^n}{||u_w^n||} - \frac{\widehat{e_w L_N u_w^n}}{||u_w^n||} - \frac{u_w^n (u_w^n)^H L_N u_w^n}{||u_w^n||^3} - \left(\frac{\widetilde{Q}\widehat{e}_w - \frac{u_w^n \widehat{L_N u_w^n}}{||u_w^n||^2}}{\left\| \widetilde{Q}\widehat{e}_w - \frac{u_w^n \widehat{L_N u_w^n}}{||u_w^n||^2} \right\|} \cdot \frac{L_N u_w^n}{||u_w^n||} - \frac{\widehat{e_w L_N u_w^n}}{||u_w^n||} - \frac{u_w^n (u_w^n)^H L_N u_w^n}{||u_w^n||^3} \right) \cdot \left(\frac{\widetilde{Q}\widehat{e}_w - \frac{u_w^n \widehat{L_N u_w^n}}{||u_w^n||^2}}{\left\| \widetilde{Q}\widehat{e}_w - \frac{u_w^n \widehat{L_N u_w^n}}{||u_w^n||^2} \right\|} \right) \right\|}.$$

For a third order method, this process would need to be repeated to obtain M_2.

Since the off diagonal entries of the matrix M_1 are essentially Fourier coefficients of a function, as the wave number increases these entries will approach 0. If we continue following the Block Lanczos algorithm, it can be shown that the off-diagonal entries of all M_j decay to 0. This can also be confirmed from numerical experiments as shown in the Figures 5.1 and 5.2. Since the off diagonal entries of M_j decay to zero as the frequency (ω) gets larger, we can define a cut off point for high and low frequency components by the point in which those entries become lower than a set tolerance.

Consider a 2 dimensional partial differential equation of the form (5.10) and a third order Krylov subspace spectral method. The frequencies for this type of problem come in pairs, we will call these ω_1 and ω_2. To search for the "cut off point" for high and low frequency, we are essentially searching for the point in which ω_1 and ω_2 are higher than some bound. To find this point we begin by making a basis function for the problem's given

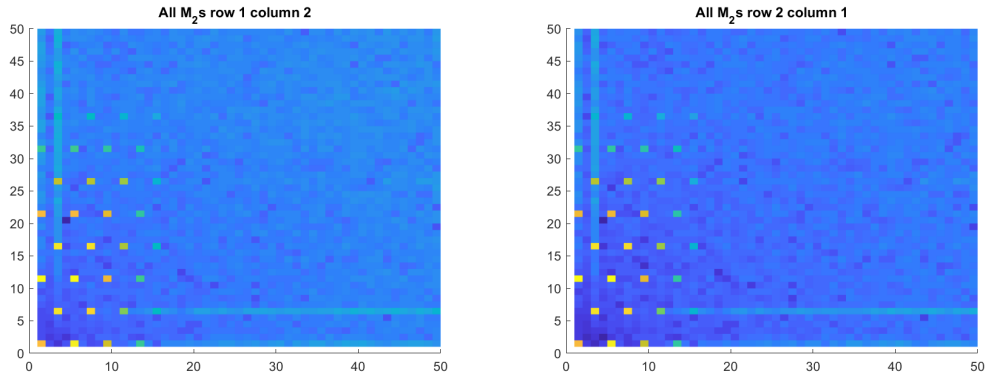

Figure 5.2: Entries of M_2 matrices obtained from Block Lanczos

boundary condition and given dimension. Then we can use that basis function along with the discretization of the solution at a given time. The basis function becomes the first column of R_0 for block Lanczos and the discretization of the solution is the second column. Once we perform black Lanczos using R_0 as the initial block; we obtain the matrices M_j and B_j for $j = 1,\ 2$ (since $K = 2$ for a third order method). If block Lanczos is performed for each frequency pair on an $N \times N$ grid, then we can obtain all the off-diagonal entries of each M_j. Since these off-diagonal entries are complex conjugates (because M is Hermitian), from here we will only consider one of these entries.

Now, we can define a matrix m_1 as a matrix containing all of the 1,2 entries of the M_1 matrices generated from performing block Lanczos using each frequency pair. We can perform the frequency analysis as follows:

$A = ||m_1||_F$
$n = 0$
while $B < \varepsilon * A$
    $n = n + 1$
    $B = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} |A_{i,j}|^2}$
end
$C = n * k$

where ε is a set tolerance, k is the bandwidth and $||\cdot||_F$ is the Frobenius norm. This algorithm can also be repeated on the 1,2 entry of M_2. The number C obtained from this algorithm is the number of non-negligable values for the 1,2 entry of M_1 (or M_2). This value can be used as the cut off point for high and low frequency. Future work will include a more efficient way to obtain this value.

5.2 Convergence

We start with the error estimate derived from block Gaussian quadrature for a third order KSS method from [1],

$$E_{\omega,k} = \tau^k \frac{\phi^{2k}(\xi)}{2k!} \vec{w}^H (L - \lambda_{1,\omega})(L - \lambda_{2,\omega}) \hat{e}_\omega \tag{5.21}$$

where $L_N = p\triangle + q(x)I$, $\mathbf{w} = (L_N - \lambda_1 I)(L_N - \lambda_{2,I})\mathbf{b}_H$, $\mathbf{b}_H = u^n$, $\lambda_{1,\omega} = p\omega^2 - ||q\tilde{(}x)$, and $\lambda_{1,\omega} = p\omega^2 + ||q\tilde{(}x)||_2$. From here we will say $q(x)I = q$. If we substitute $\vec{w}^H(L - \lambda_{1,\omega})(L - \lambda_{2,\omega})\hat{e}_\omega$ we obtain

$$\vec{w}^H (p\omega^2 q''(x) + q(x)^2 - pq(x)\omega^2 - ||\tilde{q}||_2^2)\hat{e}_\omega. \tag{5.22}$$

If we say that the decay rate of the solution is ω^{-p} then it follows that the decay rate of $\vec{w}^H$ is ω^{4-p}. If we let the decay rate of $q(x)$ be $\omega^{-\eta}$ then it follows that the decay rate of $\vec{w}^H(p\omega^2 q''(x) + q(x)^2 - pq(x)\omega^2 - ||\tilde{q}||_2^2)\hat{e}_\omega$ can be found in a similar manner to finding the decay rate of the convolution of two functions. That is given two functions f and g with decay rates $\omega^{-\alpha}$ and $\omega^{-\beta}$ respectively. The decay rate of the convolution of f and g, denoted $f * g$, is $\min(\omega^{-\alpha}, \omega^{-\beta})$ [20]. If we examine each term of $\vec{w}^H(p\omega^2 q''(x) + q(x)^2 - pq(x)\omega^2 - ||\tilde{q}||_2^2)\hat{e}_\omega$ as follows:

Term	Decay Rate of the Term ($\omega^{-\gamma}$)
$\mathbf{w}^H p\omega^2 q_{xx} e^{i\omega x}$	$\gamma = \min(p-4, \eta-2)$
$\mathbf{w}^H q^2 e^{i\omega x}$	$\gamma = \min(p-4, \eta)$
$\mathbf{w}^H pq\omega^2 e^{i\omega x}$	$\gamma = \min(p-4, \eta)$
$\mathbf{w}^H ||\tilde{q}||_2^2 e^{i\omega x}$	$\gamma = p-4$

then we can see that if $p > 4$ and $\eta > 2$ then $\vec{w}^H(L - \lambda_{1,\omega})(L - \lambda_{2,\omega})\hat{e}_\omega$ will approach 0 as the wave number ω approaches infinity. In cases where the function q is band-limited, then the decay rate of q is effectively $\omega^{-\infty}$ and so the error of the approximate solution will approach 0.

5.3 Coarse Grid Residual Correction - A Multigrid Inspired Technique

As stated in Chapter 3, 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 generally time independent like Laplace's equation) [3]. To be able to solve time dependent problems such as the ones considered in this dissertation we first need to generalize the method to the time dependent case. We do that by defining the residual as

$R = u_t + Lu$, then solving a non-homogeneous version of the partial differential equation to obtain the error for the correction.

From the error analysis performed in the previous section, we can see that smoothing already takes place in the Krylov subspace spectral method. To implement coarse grid residual correction, three functions are needed:

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).

It should be noted that the notation here is similar to the notation used by Briggs, Henson, and McCormick in their book *A Multigrid Tutorial*. That is, the superscript of the restriction and interpolation functions denotes the current grid (fine or coarse) and the subscript denotes the grid that is being interpolated or restricted to.

To implement restriction by full weighting the residual must first be reshaped from a vector to an $n \times n$ matrix B if the partial differential equation is on a 2-D Domain. Then, the residual can be restricted to the coarse grid matrix, C, as seen in the following equation:

$$C(i,j) = \frac{1}{4}(B(2i-1,2j-1) + B(2i-1,2j) + B(2i,2j-1) + B(2i,2j)). \tag{5.23}$$

To interpolate back to the fine grid, an interpolation function needs to be designed so that a coarse grid matrix entry is mapped to the corresponding fine grid matrix entry, as well as the entries surrounding it [4]. For our numerical experiments we perform this by padding the Fourier transform with zeros.

Krylov subspace spectral with coarse grid residual correction proceeds as follows:

1. Use Krylov subspace spectral as described in section 2 for all components of the solution (high and low frequency)
2. use coarse grid residual correction in a Fourier sense 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.

5.3.1 Case 1: Parabolic Partial Differential Equation

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

$$u_t = \alpha \triangle u + u - u^3, \tag{5.24}$$

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

$$u_t = \alpha \triangle u + (1 - 3u_0^2)u \tag{5.25}$$

where u_0 is the initial data

$$u_0 = 0.4 + 0.1\cos(2\pi x)\cos(5\pi y) \tag{5.26}$$

and $\alpha = 0.2$. 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 the Multigrid-like technique 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 restriction by full weighting can be implemented using (5.23).

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 5.3 represents both the coarse and fine grid residuals for (5.24) with Neumann boundary conditions. From this figure we observe that the fine and coarse grid residual are approximately the same.

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

$$L = \alpha \triangle + (1 - 3y_0^2). \tag{5.27}$$

Then we can solve the non-homogeneous equation

$$e_t = \alpha \triangle e + (1 - 3y_0^2)e + R(x,t) \tag{5.28}$$

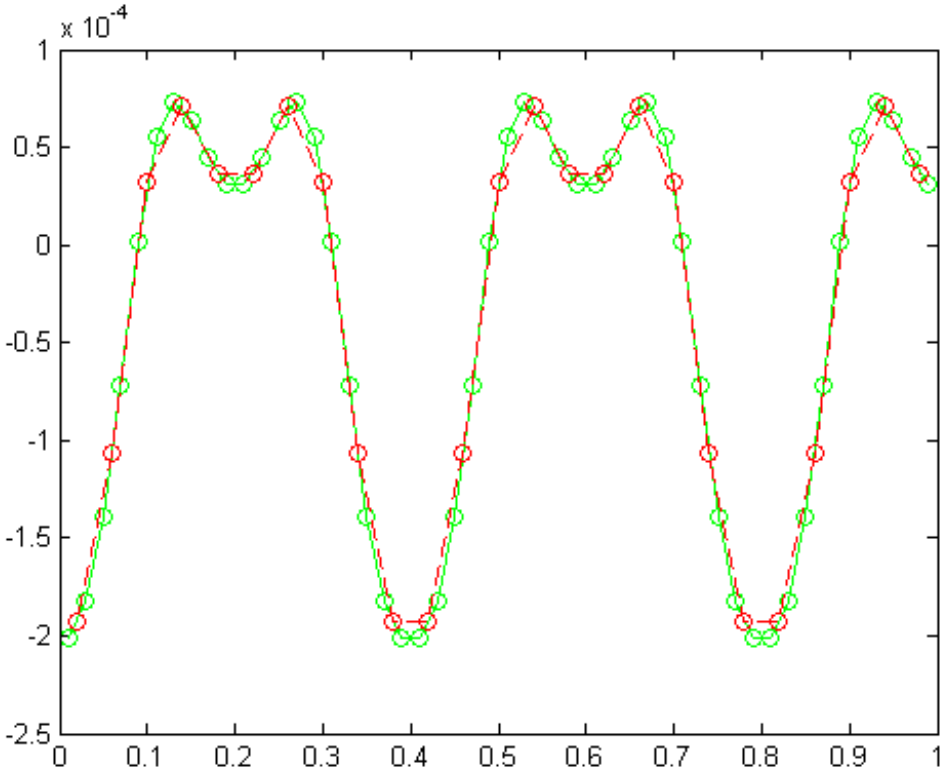

Figure 5.3: This figure shows the fine grid residual in red and the coarse grid residual in green.

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

$$e = e^{Lt}e_0 + \int_0^t e^{L(t-s)}R(x,s)\,ds. \tag{5.29}$$

Using Gaussian quadrature rules, we know the integral in (5.29) is approximately equal to

$$\int_0^{\Delta t} e^{L(\Delta t-s)}R(x,s)ds \approx \sum_k w_k e^{L(\Delta t-s_k)}R(x,s_k) \tag{5.30}$$

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, (**??**) is used, where a coarse grid matrix entry is mapped to the corresponding fine grid matrix entry as well as the entries surrounding it.

5.3.2 Case 2: Hyperbolic Partial Differential Equation

Consider the hyperbolic partial differential equation

$$u_{tt} = Lu, \qquad \text{on } (0,2\pi) \times (0,\infty), \tag{5.31}$$

$$u(x,0) = f(x), \qquad u_t(x,0) = g(x), \quad 0 < x < 2\pi. \tag{5.32}$$

To apply KSS with coarse grid residual correction 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:

$$\begin{bmatrix} \cos(\sqrt{-L}\triangle t) & \frac{1}{\sqrt{-L}}\sin(\sqrt{-L}\triangle t) \\ -\sqrt{-L}\sin(\sqrt{-L}\triangle t) & \cos(\sqrt{-L}\triangle t) \end{bmatrix}. \tag{5.33}$$

Therefore, we can use Krylov subspace spectral with coarse grid residual correction 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

$$e = \int_0^{\Delta t} \begin{bmatrix} \cos(\sqrt{-L}\triangle t - s) & \frac{1}{\sqrt{-L}}\sin(\sqrt{-L}\triangle t - s) \\ -\sqrt{-L}\sin(\sqrt{-L}\triangle t - s) & \cos(\sqrt{-L}\triangle t - s) \end{bmatrix} \begin{bmatrix} 0 \\ R(s) \end{bmatrix} ds. \tag{5.34}$$

The entries of (5.34) indicate which functions are the integrands in the Riemann-Stieltjes integrals that are used to compute the Fourier coefficients of the solution [15].

Consider an equation similar to the linearized form of the Allen-Cahn equation (6.6):

$$u_{tt} = \alpha\triangle u + (-3y_0^2)u \tag{5.35}$$

with Neumann boundary conditions as well. To implement KSS with coarse grid residual correction to solve this problem, a similar process is followed as when solving (6.3). First, the differential operator for any solution u becomes

$$Lu = \alpha\Delta u + (-3y_0^2)u. \tag{5.36}$$

Then a spatial discretization of the differential operator L allows us to obtain a representation of the solution operator, as seen by (5.33). Then the KSS method with coarse grid residual correction can be applied in a similar manner.

5.3.3 Adaptive Time Stepping

As described in Chapter 3, most adaptive time stepping methods use the local truncation error to control the step size of a time stepping method for solving differential equations. Adaptive methods for controlling the time step size of a method are very useful because they are able to concentrate the computational power of a method to areas that require it while expending less power to areas that don't require it.

Unlike the process described in Chapter 4 using two one step methods, finding a formula for the local truncation error of a time step in KSS methods can be quite difficult. Since an error approximation is necessary for adaptive methods and the global error would be inefficient to use, in this dissertation, we consider an alternate approach. Specifically, we propose using the residual as an error estimator to control the time step size in KSS methods. To do this, a very accurate residual must be obtained. Unlike other time stepping methods, by KSS we can obtain the time derivatives of the solution (u_t and u_{tt}) exactly.

Consider the time dependent partial differential equation of the form

$$u_t = -Lu \tag{5.37}$$

with periodic boundary conditions. To find the optimal step size, first numerical approximations to u_t and u must be found at the first time step. We do this by using a time step of the the KSS method (as described in Chapter 2) to find an approximate solution for the high frequency components and the low frequency components. Then, we define the residual for the high frequency components as

$$R^H = \tilde{u}_t^H + L\tilde{u}^H \tag{5.38}$$

and the residual for the low frequency components (found by using Krylov Projection) as

$$R^L = \tilde{u}_t^L + L\tilde{u}^L \tag{5.39}$$

where $\tilde{u}^H$ is an approximation of the high frequency components of the solution at that time step and $\tilde{u}^L$ is an approximation of the low frequency components of the solution at that time step.

The scaling factor q can be found similarly to equation (4.13) in Chapter 4:

$$q = \left(\frac{\varepsilon}{||R^H + R^L||_\infty} \right)^{1/p} \tag{5.40}$$

where ε is the preset tolerance and p is the order of the method. If q is greater than or equal to 1 then the solution can be accepted and the time step can be increased by a factor of $q * h$.

If q is less than 1 then the error is too large for that time step size so the step size must be reduced.

5.3.4 Case 1: Parabolic Problem

As mentioned in Section 5.2.1, the Allen-Cahn equation is a 2-D parabolic partial differential equation used to describe the reaction-diffusion of the separation of iron alloys. The linearized version of this equation is

$$u_t = \alpha \triangle u + (1 - 3u_0^2)u \tag{5.41}$$

where u_0 is the initial data is

$$u_0 = 0.4 + 0.1\cos(2\pi x)\cos(5\pi y) \tag{5.42}$$

and $\alpha = .1$.

To create a time step size controller for Krylov subspace spectral methods, the approximate solutions for the high frequency components and low frequency components at the starting time step must first be calculated. These approximate solutions are used to then find the residuals for the high frequency components R^H and low frequency components R^L. If we define the operator L as

$$L = \alpha \triangle + (1 - 3u_0) \tag{5.43}$$

then we can define the two residuals to be

$$R^H = u_t^H - Lu^H \tag{5.44}$$

$$R^L = u_t^L - Lu^L. \tag{5.45}$$

To use the residual as an error approximation for the step size controller we can use the maximum absolute value of the sum of the residuals (the infinity-norm) to calculate our scaling factor q:

$$q = \left(\frac{\varepsilon}{||R^H + R^L||_\infty} \right)^{\frac{1}{3}}. \tag{5.46}$$

For most of our numerical experiments, the tolerance is set to be $\varepsilon = 1.0 \times 10^{-4}$ and the tolerance is updated as the starting time step decreases using the formula

$$\varepsilon = \varepsilon/(2^p); \tag{5.47}$$

where $p = 3$ is the order of the original method.

5.3.5 Case 2: System of Equations

Consider a linearized version of the Brusselator problem

$$u_t = \alpha \triangle u + pu + \phi v \tag{5.48}$$

$$v_t = \alpha \triangle v + qv + \psi u \tag{5.49}$$

with homogeneous Dirichlet boundary conditions and appropriate initial data. Here we will let $\alpha,\ p$, and q be constant variables and ϕ and ψ are variable.

Similar to the parabolic problem, to create a step size controller for Krylov subspace spectral methods we first find the approximate solutions for the high and low frequency components and use these approximate solutions as well as the exact time derivative to find the residuals R^L and R^H. If we define the operator L as the matrix

$$L = \begin{bmatrix} \alpha \triangle + p & \phi \\ \psi & \alpha \triangle + q \end{bmatrix} \tag{5.50}$$

then we can define the low and high frequency residuals as

$$R^H = u_t^H - Lu^H \tag{5.51}$$

$$R^L = u_t^L - Lu^L. \tag{5.52}$$

Then adaptive time stepping can be implemented in a similar manner to the parabolic problem.

Chapter 6

Numerical Results

In this chapter, the effectiveness of Kylov subspace spectral methods with coarse grid residual correction and then with adaptive time stepping will be demonstrated. In the result sections for KSS with coarse grid residual correction, the following approaches will be compared:

- Krylov subspace spectral methods (KSS) as described in Chapter 2 with the high/low frequency split which will be represented in figures by a red solid line with star markers.
- Krylov subspace spectral methods with coarse grid residual correction (KSS-CGRC2, KSS-CGRC3, and KSS-CGRC5) as described in Chapter 3, which will be represented by a black solid or dotted line with square markers (depending on the problem).
- Krylov subspace spectral exponential propagation iterative methods (KSS-EPI) as described in [1] which will be represented by a magenta dash-dot line with star markers.
- Krylov Projection (KP), as described in [9] which will be represented by a blue dashed line with circle markers.

It should be noted that KSS-CGRC2 uses a 2-node Gaussian quadrature Rule, KSS-CGRC3 uses a 3-node Gaussian quadrature Rule, and KSS-CGRC5 uses a 5-node Gaussian quadrature Rule. Two types of problems will be considered in this section, a parabolic problem and a hyperbolic problem. The parabolic problem is the linearized version of the Allen Cahn Equation which describes the process of phase separation in multi-component alloy systems. The hyperbolic problem is a general wave equation.

In the result sections for Krylov subspace spectral methods with adaptive time stepping, the following approaches will be compared:

- Krylov subspace spectral methods as described in Chapter 2 with and without the high/low frequency split (KSS and KSS2 respectively.) KSS will always be represented in figures by a red solid line with circle markers. KSS2 will be represented by a red dash-dot line with circle markers.

- Krylov subspace spectral methods with adaptive time stepping (ATS) as described in Chapter 4 as well as modified adaptive time stepping methods:
  - ATS h_{min} (or just h_{min}) is KSS with adaptive time stepping and will be represented by a cyan, magenta, or green (depending on the minimum step size) dashed line with square markers.
  - ATS2 is KSS2 with adaptive time stepping and will be represented by a black dash-dot line with square markers.
  - ATS2 h_{min} is KSS2 with adaptive time stepping and will be represented by a cyan, magenta, or green (depending on the minimum step size) dash-dot line with square markers.
- Krylov Projection (KP), as described in [9], which will be represented by a blue dashed line with circle markers.
- Newton interpolation using Leja points (LEJA) which will be represented by a green dashed line with plus sign markers.

Two types of problems will be considered in this section, a parabolic problem and a system of equations. The parabolic problem is the same linearized Allen Cahn equation used in the experiments for KSS-CGRC. The system of equations is a linearized version of the Brusselator system of equations. The Brusselator system of equations describes a type of auto-catalytic reactions.

In all of these experiments the "exact" solution is found using MATLAB's ode15s solver with the smallest possible time step.

6.1 Krylov Subspace Spectral Methods with Coarse Grid Residual Correction Case 1: 2-D Linear Parabolic Problems

We first demonstrate the effectiveness of all four methods when solving a linearized version of the Allen Cahn equation:

$$u_t = \alpha \triangle u + (1 - 3u_0^2)u \tag{6.1}$$

where u_0 is the initial data

$$u_0 = 0.4 + 0.1\cos(9\pi x)\cos(18\pi y) \tag{6.2}$$

and alpha is a constant ($\alpha = 0.2$). For this numerical experiment there are homogeneous Neumann boundary conditions.

Table 6.1 contains the calculated relative error and Table 6.2 contains the execution time per time step size for grid resolution sizes $N = 50$, $N = 150$ and $N = 300$ points per dimension. When comparing the accuracy of all five methods (KSS-CGRC2, KSS-CGRC3, KSS-EPI, KSS, and KP), it is obvious that the KSS-CGRC methods have significantly lower error especially when using a small time step size ($\triangle t = 0.00125$). This advantage becomes more apparent when the grid size increases from $N = 50$ to $N = 150$ points per dimension as seen in Figure 6.1.

	KSS-CGRC2	KSS-CGRC3	KSS	KSS-EPI	KP
N=50	6.9647e-07	2.0368e-07	7.1375e-06	0.0001641	0.0001641
	6.1329e-09	2.0551e-09	8.2972e-07	8.5766e-06	2.9759e-05
	7.2752e-10	6.8869e-11	1.0589e-07	4.5585e-08	7.2182e-06
	9.4961e-11	3.2213e-12	1.1477e-08	5.4246e-09	6.6096e-08
	2.8871e-11	4.366e-11	1.2222e-09	6.5594e-09	4.0615e-09
N=150	5.0452e-07	1.5778e-07	7.364e-06	0.00016596	0.00016596
	4.2637e-09	2.463e-09	8.6024e-07	2.041e-05	7.4835e-05
	7.3494e-10	8.8751e-11	1.1013e-07	3.2691e-06	2.7004e-05
	9.8793e-11	4.0197e-12	1.1938e-08	3.8396e-07	2.7823e-06
	7.7294e-12	2.6783e-13	1.2666e-09	4.2296e-08	7.4641e-08
N=300	5.0351e-07	1.5909e-07	7.3879e-06	0.00016612	0.00016612
	4.2653e-09	2.5183e-09	8.6291e-07	4.8436e-05	0.00010625
	7.3494e-10	9.0492e-11	1.1052e-07	1.1786e-05	3.5554e-05
	9.913e-11	4.112e-12	1.1981e-08	2.4341e-06	6.2902e-06
	7.7676e-12	2.6502e-13	1.2709e-09	3.6416e-07	7.0351e-07

Table 6.1: Relative error calculated for time step sizes $\triangle t = 0.2, 0.1, 0.05, 0.025, 0.0125$ for the parabolic problem.

Also, as the grid size increases we can observe a decrease in accuracy for both KSS-EPI and Krylov projection. This phenomenon is not observed in the experimental results for KSS where the accuracy for this methods is relatively constant regardless of grid size. The results for KSS-CGRC get slightly more accurate on larger grid sizes.

Table 6.3 displays the Krylov subspace dimensions needed for the method. For the KSS method, the Krylov subspace dimension is 4 and so KSS-CGRC2 and KSS-CGRC3 also will have a Krylov subspace dimension of 4 since no more dimensions are required to do a correction. The number of iterations needed in a time step for Krylov Projection is much more and increases as the grid size (N) increases. The maximum average number of iterations needed for $N = 50, 150$, and 300 was 6.3125, 15.375, and 27 respectively. This result shows the scalability of the KSS and KSS-CGRC methods and that as the grid size increases the computational expense will not grow as fast as it will for Krylov Projection.

	KSS-CGRC2	KSS-CGRC3	KSS	KSS-EPI	KP
N=50	0.10938	0.03125	0	0.0625	0.015625
	0.17188	0.14063	0	0.03125	0
	0.125	0.17188	0.03125	0.125	0.015625
	0.29688	0.35938	0.078125	0.0625	0.078125
	0.34375	0.42188	0.125	0.21875	0.125
N=150	0.10938	0.125	0.046875	0.03125	0
	0.26563	0.46875	0.125	0.15625	0.09375
	0.42188	0.73438	0.20313	0.26563	0.35938
	1.1094	1.0313	0.375	0.79688	2
	1.8438	2.5625	0.73438	1.25	3.2188
N=300	0.5	0.57813	0.1875	0.10938	0.17188
	1.0938	1.0938	0.42188	0.42188	0.39063
	1.6875	2.1094	0.71875	1.0781	2.1719
	3.6875	5.0156	1.375	1.9688	13.813
	7.2344	10.25	2.8906	4.5469	30.953

Table 6.2: Execution times calculated for time step sizes $\triangle t = 0.2, 0.1, 0.05, 0.025, 0.0125$ for the parabolic problem

	KSS-CGRC2	KSS-CGRC3	KSS	KSS-EPI	KP
N=50	4	4	4	4	4
	4	4	4	6.5	5.5
	4	4	4	7	7
	4	4	4	6.75	7.875
	4	4	4	6.4375	6.3125
N=150	4	4	4	4	4
	4	4	4	6	5.5
	4	4	4	6	8.5
	4	4	4	6.25	16.625
	4	4	4	6.125	15.375
N=300	4	4	4	4	4
	4	4	4	5	5.5
	4	4	4	5	11.5
	4	4	4	5	24
	4	4	4	5.375	27

Table 6.3: Number of iterations for time step sizes $\triangle t = 0.2, 0.1, 0.05, 0.025, 0.0125$ for the parabolic problem.

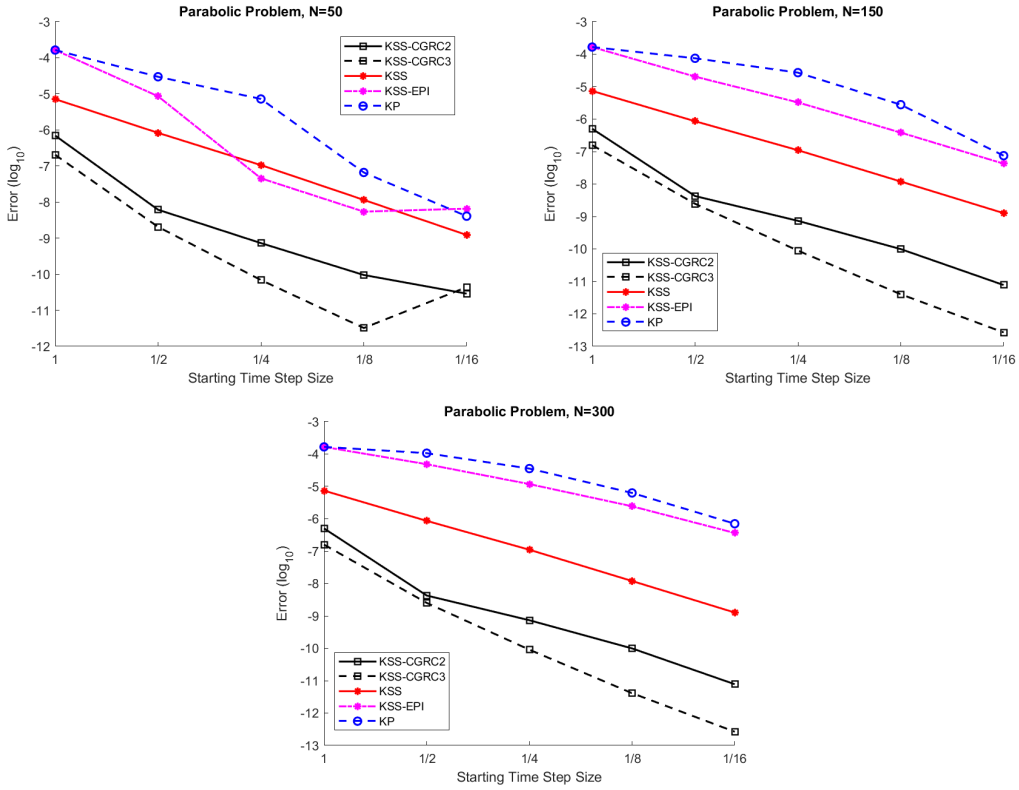

Figure 6.1: Relative Error (logarithmically scaled) for varying starting time step sizes ($\triangle t = 0.02, 0.01, 0.005, 0.0025, 0.00125$) for the parabolic problem.

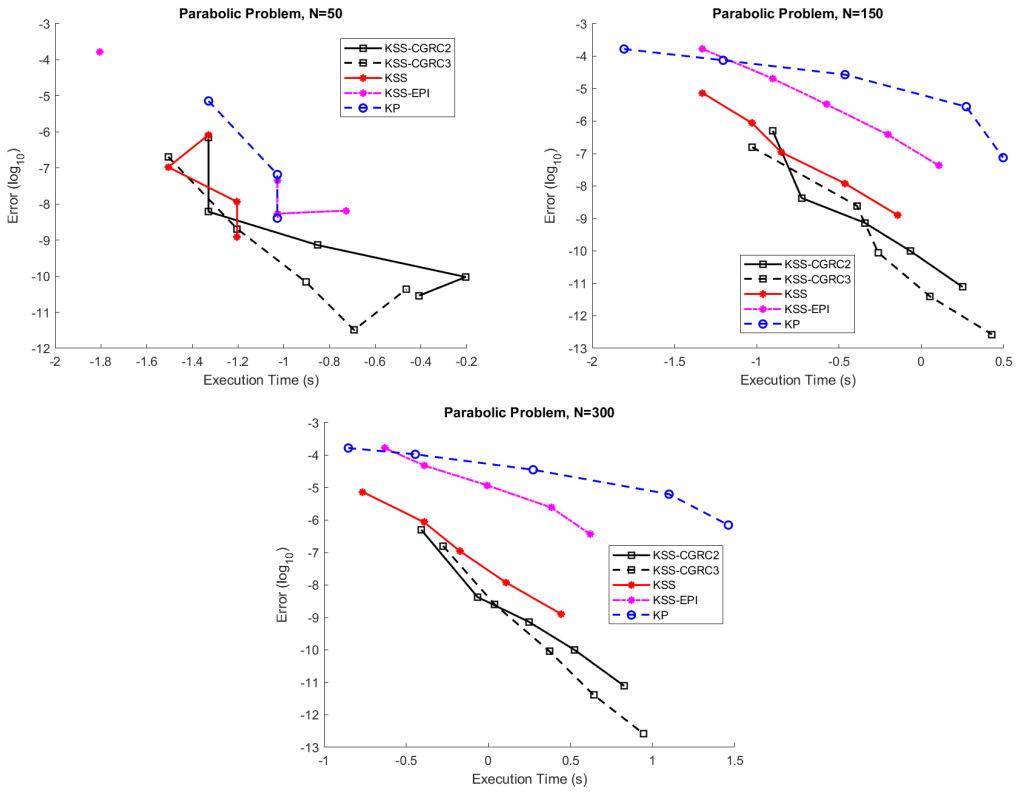

Figure 6.2: Relative error versus execution time in seconds (logarithmically scaled) for varying starting time step sizes ($\triangle t = 0.02, 0.01, 0.005, 0.0025, 0.00125$) for the parabolic problem.

6.2 Krylov Subspace Spectral Methods with Coarse Grid Residual Correction Case 2: 2-D Hyperbolic Problems

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

$$u_{tt} = \triangle u - (3u_0)u, \qquad \text{on } (0,2\pi) \times (0,\infty), \tag{6.3}$$

$$u(x,0) = f(x), \qquad u_t(x,0) = g(x), \quad 0 < x < 2\pi \tag{6.4}$$

where u_0 is the initial condition

$$u_0 = 0.04 + .1\cos(2\pi x)\cos(5\pi y). \tag{6.5}$$

Table 6.5 contains the time elapsed (per time step) is and Table 6.4 contains the relative error calculated per time step size for grid sizes $N = 50$ and $N = 150$ for each test method (KSS-CGRC3, KSS-CGRC5, KSS, KP, and KSS-EPI). For both KSS-CGRC methods, we observe a higher accuracy than other methods for both grid sizes. Also the KSS-CGRC methods improve as grid resolution increases whereas the results for KSS-EPI and Krylov Projection decrease in accuracy. From Figure 6.3 we can also clearly see the difference in accuracy of the methods at small time step sizes. When the grid resolution was $N = 50$, both KSS-CGRC methods significantly improved as the time step size went from the original size $\triangle t = 0.02$ to $\frac{1}{16}^{th}$ of the original step size. This phenomenon is even more apparent when the grid resolution increases to $N = 150$.

Similar to Section 6.1, KSS as well as both KSS-CGRC methods only needs a Krylov subspace dimension of 4. For Krylov Projection, the number of iterations needed in a time step is much more and increases as the grid size (N) increases. The maximum average number of iterations needed for $N = 50$ and 150, was 36.5 and 18.5 respectively. This result shows the scalability of the KSS and KSS-CGRC methods and that as the grid size increases the computational expense will not grow as fast as it will for Krylov Projection.

	KSS-CGRC3	KSS-CGRC5	KSS	KSS-EPI	KP
N=50	5.5349e-05	3.124e-05	0.0001967	9.54e-05	5.0232e-05
	3.1533e-05	4.9734e-06	8.8408e-05	4.8842e-06	7.7929e-06
	1.5991e-06	6.3904e-07	8.6061e-06	1.4539e-06	8.9385e-07
	9.8503e-09	6.2675e-09	2.2604e-07	1.5063e-07	2.0124e-07
	1.1217e-10	6.0229e-11	6.9945e-09	5.1051e-09	1.5384e-08
N=150	5.5914e-05	2.746e-05	0.00020072	2.8507e-05	0.0001427
	1.3768e-05	7.9052e-06	0.00010045	5.1973e-05	0.00013132
	2.0382e-06	7.0585e-07	3.3588e-06	7.2583e-05	0.00010472
	6.66e-09	4.7481e-09	8.3558e-08	9.2289e-06	0.00011705
	1.7555e-11	1.1384e-11	6.3856e-10	1.0968e-07	8.5461e-05

Table 6.4: Relative error calculated for time step sizes $\triangle t = 0.2, 0.1, 0.05, 0.025, 0.0125$ for the hyperbolic problem.

	KSS-CGRC3	KSS-CGRC5	KSS	KSS-EPI	KP
N=50	0.125	0.0625	0.03125	0.1875	0.1875
	0.10938	0.26563	0.046875	0.3125	0.8125
	0.25	0.51563	0.125	0.28125	0.48438
	0.46875	0.6875	0.20313	0.26563	0.35938
	0.95313	1.1406	0.42188	0.53125	0.375
N=150	0.3125	0.375	0.14063	7.3281	0.78125
	0.60938	0.79688	0.23438	32.344	1.8125
	1.3281	1.6094	0.42188	8.2344	2.2344
	2.4531	3.0781	0.98438	4.3438	2.8594
	4.4531	6.2813	1.8906	4.9844	4.4531

Table 6.5: Execution time in seconds for time step sizes $\triangle t = 0.2, 0.1, 0.05, 0.025, 0.0125$ for the hyperbolic problem.

	KSS-CGRC3	KSS-CGRC5	KSS	KSS-EPI	KP
N=50	4	4	4	23	23
	4	4	4	27	36.5
	4	4	4	15.75	23
	4	4	4	10.625	13.875
	4	4	4	9	9.5
N=150	4	4	4	60	17
	4	4	4	85	18.5
	4	4	4	31.75	13.25
	4	4	4	14.375	10.25
	4	4	4	9	8.3125

Table 6.6: Number of iterations for time step sizes $\triangle t = 0.2, 0.1, 0.05, 0.025, 0.0125$ for the hyperbolic problem.

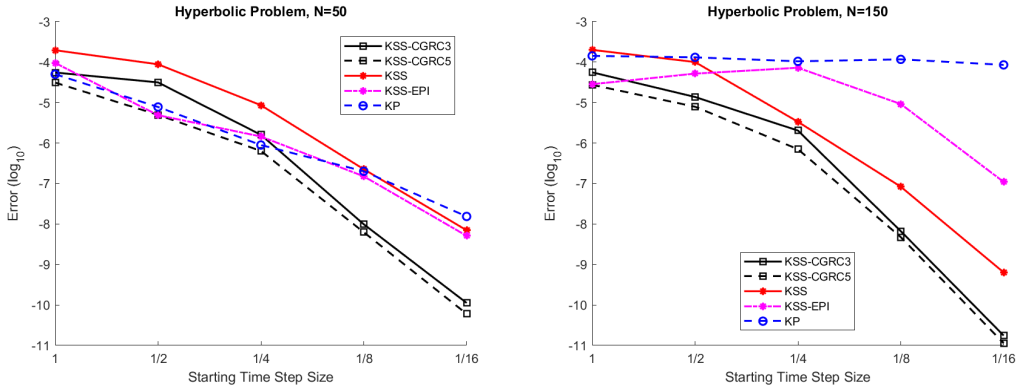

Figure 6.3: Relative Error (logarithmically scaled) for varying starting time step sizes for the hyperbolic problem.

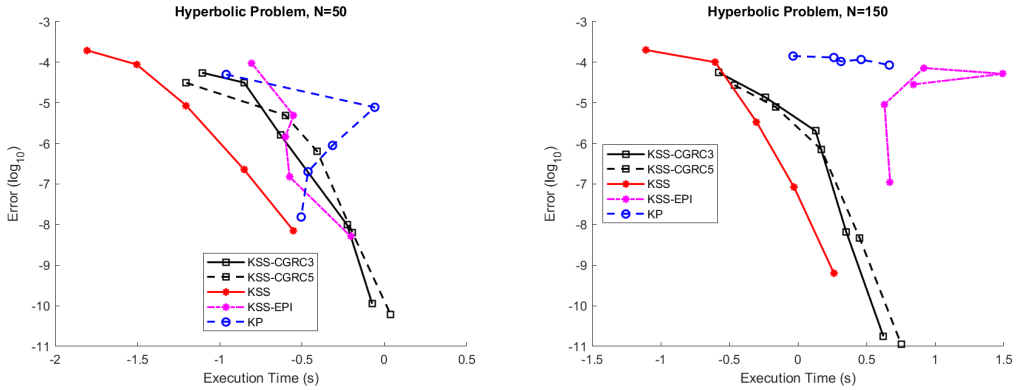

Figure 6.4: Relative error versus execution time in seconds (logarithmically scaled) for varying starting time step sizes for the hyperbolic problem.

6.3 Krylov Subspace Spectral Methods with Adaptive Time Stepping Case 1: 2-D Linear Parabolic Problems

Here we consider a similar linearized version of the Allen Cahn equation mentioned in Section 6.1.

$$u_t = \alpha \triangle u + (1 - 3u_0^2)u \tag{6.6}$$

where u_0 is the initial data and alpha is a constant ($\alpha = 0.1$). The initial data is

$$u_0 = 0.4 + 0.1\cos(2\pi x)\cos(5\pi y). \tag{6.7}$$

We begin by examining the approximate solution of the equation as shown in Figure 6.5. Since the Allen Cahn equation (even the linearized version) is a diffusive partial differential equation, the solution becomes less oscillatory over time. This implies that an accurate step size controller would modify the time step size to be smaller at lower time steps then allow the step size to increase as the solution becomes less oscillatory. Figures 6.6 and 6.6 show this result - at higher frequencies the step size controller decreases the time step and at low frequencies less variation can be seen between solutions at different time steps and so the step size controller allows the step size to become large again. We can see this phenomenon in Figures 6.10 and 6.11. From these figures we can see that around $t = 0.08$ seconds the solution diffuses enough for the adaptive time stepping algorithm to increase the time step size. This is particularly apparent in Figure 6.11, the slope of the line measuring time versus the number of time steps for the adaptive algorithm increases dramatically after approximately $t = 0.08$ seconds.

When $N = 25$ or $N = 50$ points per dimension, the total number of time steps required to reach the set error tolerance is much lower than using traditional Krylov subpace spectral methods. Comparing Figures 6.7 and 6.6, we can see that when using adaptive time stepping with a starting time step of $\triangle t = .0025$ seconds the method requires an eighth of the amount of time steps used in the traditional method.

When the resolution of the problem increases to $N = 150$, adaptive time stepping becomes less efficient to use because of the computational power require to reach the set error tolerance on a high resolution grid. This phenomenon can be clearly seen when comparing figures 6.9 and 6.8. The number of time steps in the version of Krylov subspace spectral methods without adaptive time stepping are much less than the version with adaptive time stepping implemented. This result implies that implementing a step size controller for a diffusive type problem is most effective on problems with lower resolution (points per dimension) unless an optimal minimum step size is set (see later experiments).

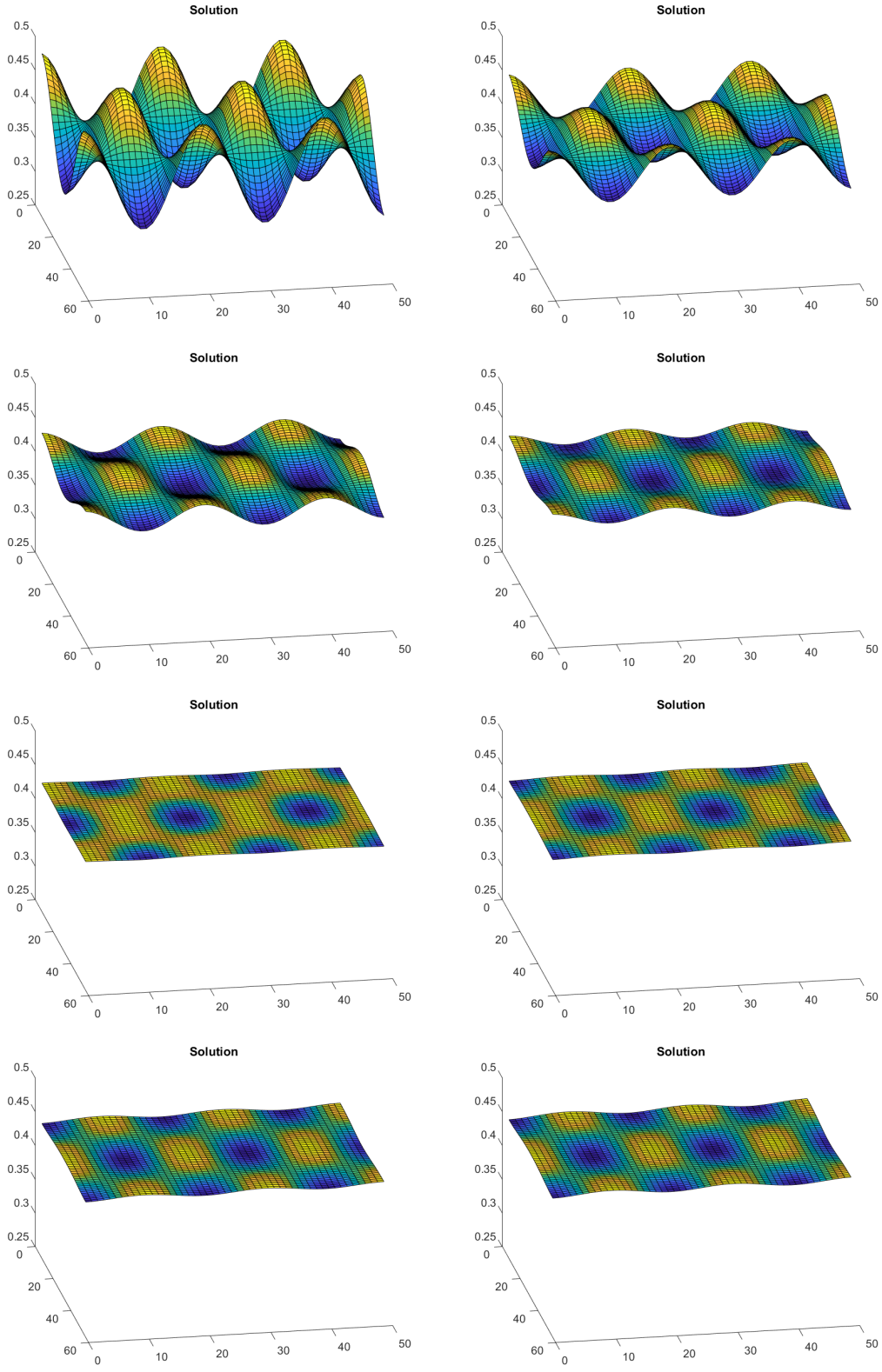

Figure 6.5: The solution of the linearized Allen Cahn equation at the times $t = 0.01,\ 0.03,\ 0.05,\ 0.07,\ 0.11,\ 0.13,\ 0.17,\ .2$, with $N = 50$ points per dimension.

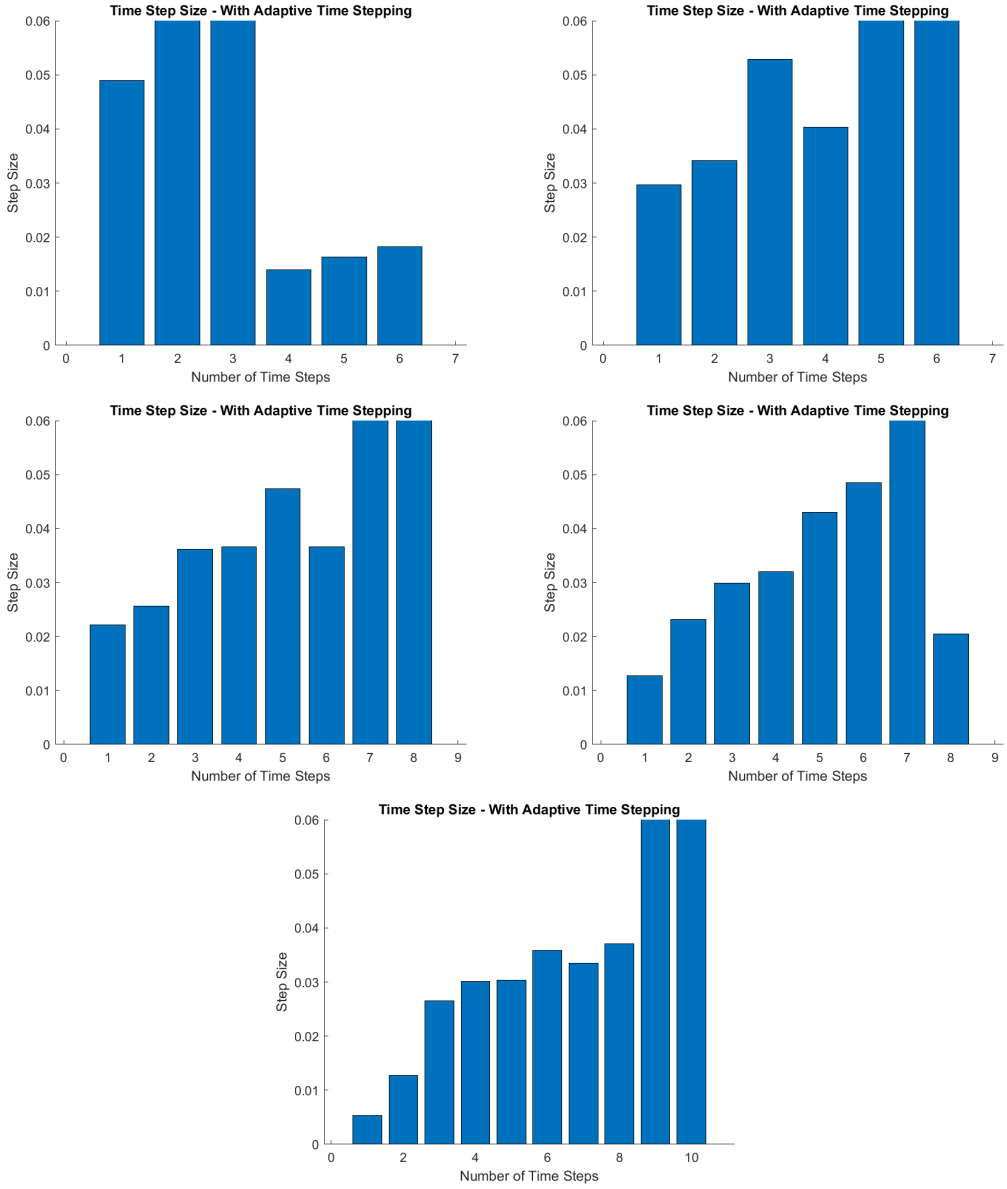

Figure 6.6: Time step size ($\triangle t$) for each time step for using KSS with adaptive time stepping with $N = 50$ points per dimension on the linearized Allen Cahn equation. From first figure (top left) to last (bottom right) the starting time steps are $\triangle t = 0.04, 0.02, 0.01, 0.005, 0.0025$ seconds.

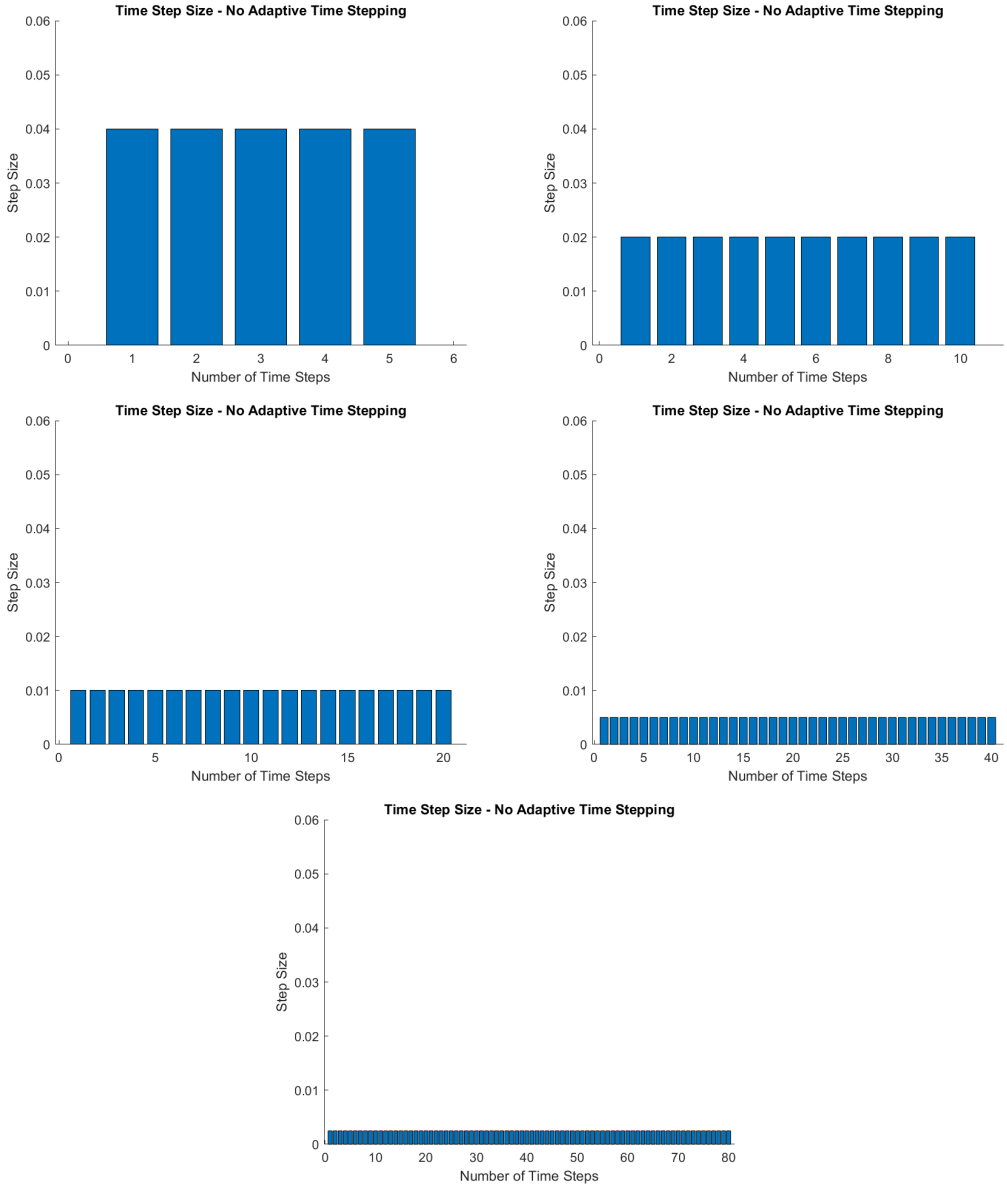

Figure 6.7: Time step size ($\triangle t$) for each time step for using KSS without adaptive time stepping with $N = 50$ points per dimension on the linearized Allen Cahn equation. From first figure (top left) to last (bottom right) the time step sizes are $\triangle t = 0.04, 0.02, 0.01, 0.005, 0.0025$ seconds.

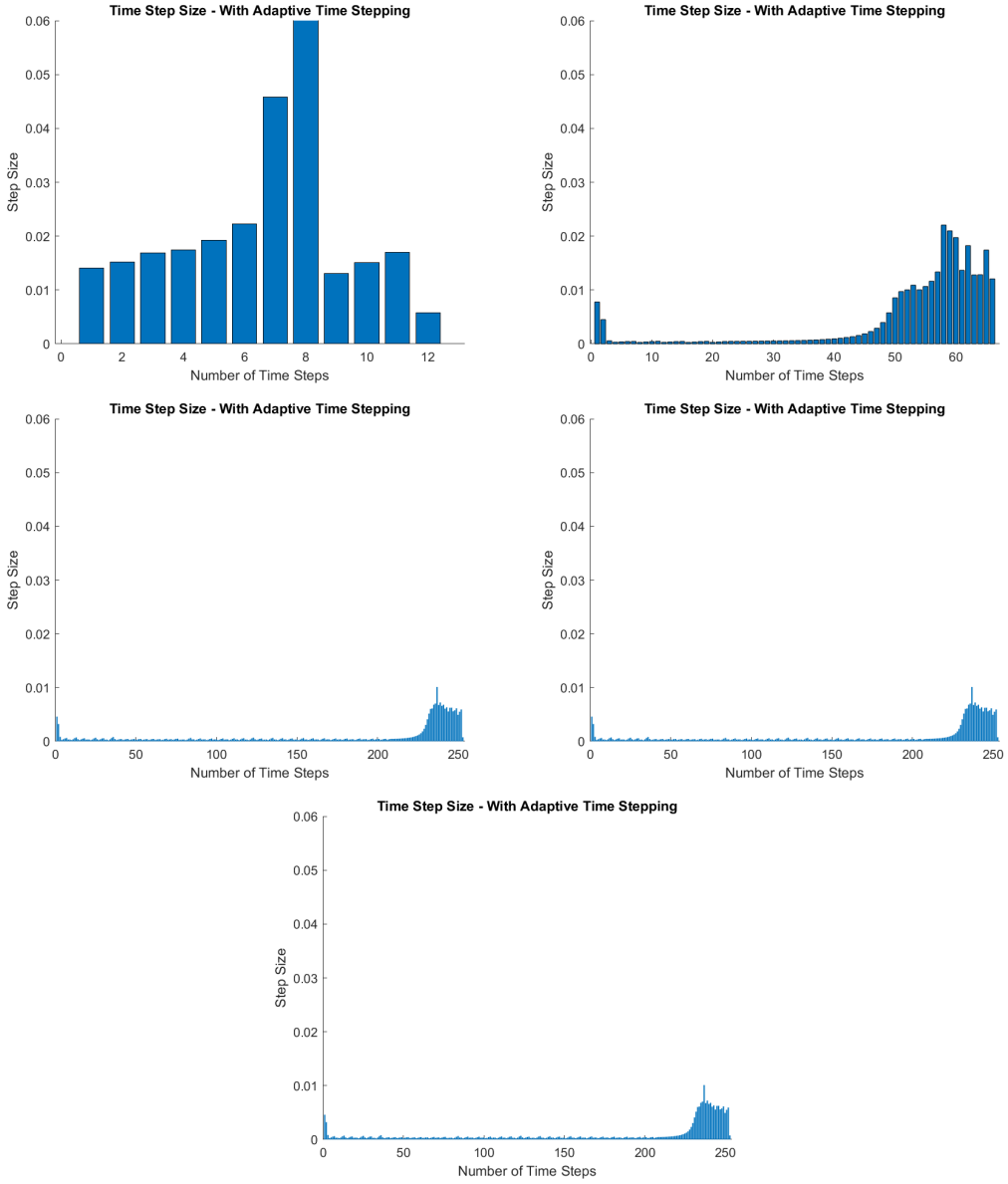

Figure 6.8: Time step size ($\triangle t$) for each time step for using KSS with adaptive time stepping with $N = 150$ points per dimension on the linearized Allen Cahn equation. From first figure (top left) to last (bottom right) the starting time steps are $\triangle t = 0.04, 0.02, 0.01, 0.005, 0.0025$ seconds.

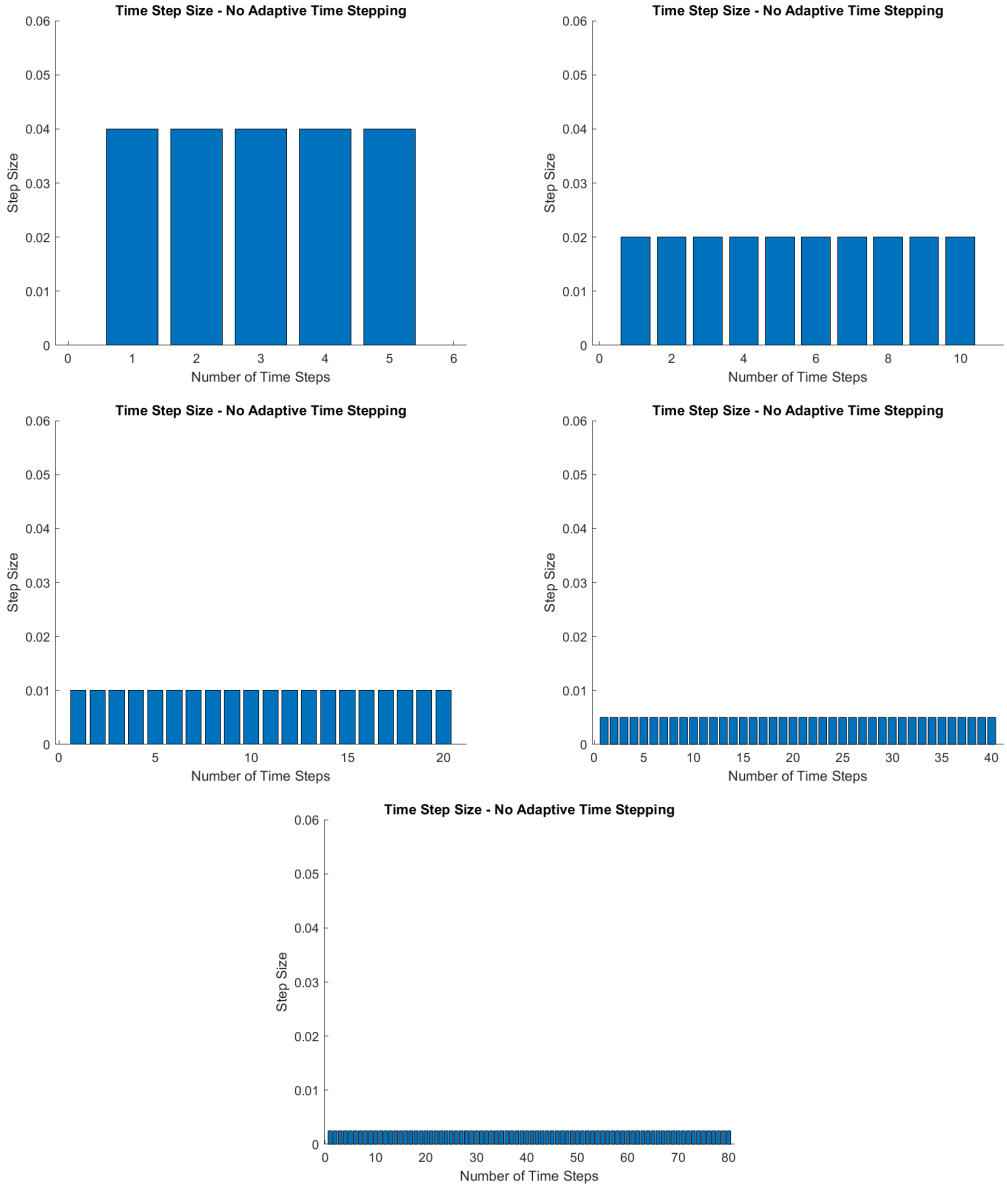

Figure 6.9: Time step size ($\triangle t$) for each time step for using KSS without adaptive time stepping with $N = 150$ points per dimension on the linearized Allen Cahn equation. From first figure (top left) to last (bottom right) the time step sizes are $\triangle t = 0.04, 0.02, 0.01, 0.005, 0.0025$ seconds.

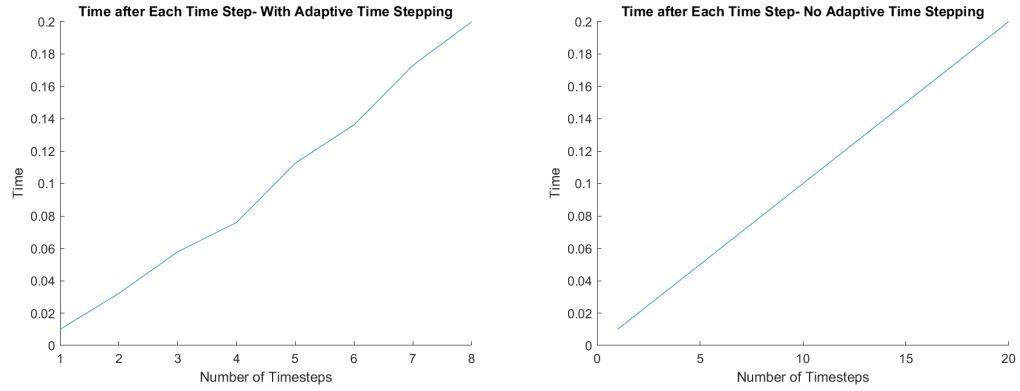

Figure 6.10: A comparison of the final time after each time step for KSS with adaptive time stepping on the left and KSS without adaptive time stepping on the right. The points per dimension in these experiments were $N = 50$. The starting time step for both is $\triangle t = 0.01$.

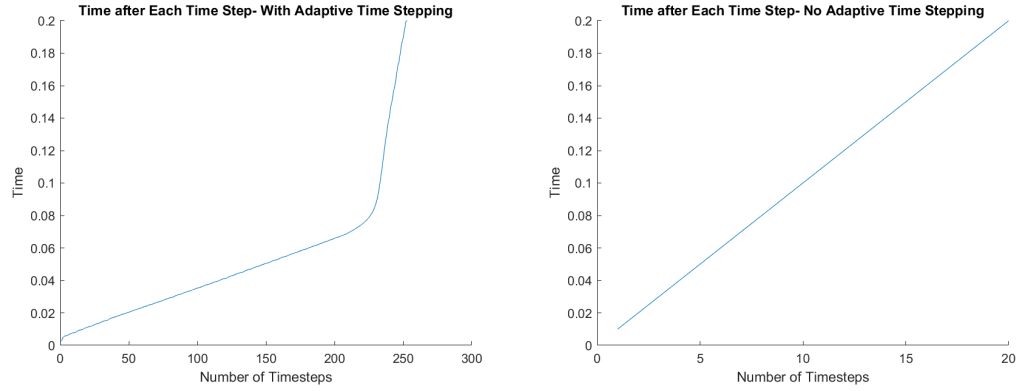

Figure 6.11: A comparison of the final time after each time step for KSS with adaptive time stepping on the left and KSS without adaptive time stepping on the right. The starting time step for both is $\triangle t = 0.01$ and there are $N = 150$ points per dimension.

	Starting Time-step	KSS-ATS	KSS
$N = 25$	.04	3	5
	.02	5	10
	.01	6	20
	.005	5	40
	.0025	8	80
$N = 50$	.04	6	5
	.02	6	10
	.01	8	20
	.005	8	40
	.0025	10	80
$N = 150$	.04	12	5
	.02	66	10
	.01	253	20
	.005	253	40
	.0025	253	80

Table 6.7: A comparison of the final time step count for Krylov subspace spectral methods with and without adaptive time stepping (KSS-ATS and KSS respectively).

6.4 Adaptive Time Stepping Case 1: Comparison of Residuals

As stated in the previous chapter, we define the residual R for the linearized Allen Cahn equation to be

$$R = u_t - Lu \tag{6.8}$$

where u_t is the first derivative of u with respect to time, and $L = \alpha\triangle + (1 - 3u_0^2)$. Since KSS methods split high and low frequency components of the solution as explained in the frequency analysis section of Chapter 5, we will use R^H to denote the residual obtained from the high frequency components of the solution and R^L to denote the residual obtained from the low frequency components of the solution. The high frequency components of the solution are obtained from using KSS as described in Chapter 2, and the low frequency components of the solution are obtained from Krylov Projection (KP) [9].

Figures 6.12 and 6.13 show a plots of the high and low frequency residuals at times $t = 0.01,\ 0.032196,\ 0.057856,\ 0.075929,\ 0.11258,\ 0.1363,\ 0.17295,\ 0.2$. The residual for the high frequency components is much smaller than the residual computed for the low frequency components. This is to be expected since it has been shown that KSS is generally more accurate than Krylov Projection for these types of problems [1].The residuals computed for the high frequency components range from -1×10^{-9} to 1×10^{-9} and the residuals computed for the low frequency components range from -1×10^{-5} to 1×10^{-5}. This behavior coincides with previous work by Cibotarica et al [1]. Since the residual for the low frequency components of the solution is much larger than those of the high frequency components, the error estimation for the step size controller will depend primarily on the size of the low frequency residual.

Tables 6.8, 6.9, 6.10, and 6.11 contain different local error estimates for KSS with and without adaptive time stepping using starting step size $\triangle t = 0.04$.The equations used to calculate each error estimate are:

$$\text{Low Frequency Error} = \frac{||R^L||_\infty}{||b_L + b_H||_\infty} \tag{6.9}$$

$$\text{High Frequency Error} = \frac{||R^H||_\infty}{||b_L + b_H||_\infty} \tag{6.10}$$

$$\text{Relative Residual} = \frac{||R^L + R^H||_\infty}{||b_L + b_H||_\infty} \tag{6.11}$$

$$\text{Residual} = R^H + R^L. \tag{6.12}$$

In Tables 6.8 and 6.10, the residual column contains the error approximation for the accepted step sizes. The error approximations for KSS-ATS are very uniform as expected. This is

due to the fact that if the error approximation calculated using any given step size was too high, the step size controller shortened the step size to increase accuracy. Without adaptive time-stepping, the error approximation can be more than ten times higher especially at the beginning and the end of the time domain.

Number of Time Steps	$\triangle t$	t	High Frequency Error	Low Frequency Error	Relative Residual	Residual
1	0.048916	0.04	0	0.00010967	0.00010967	5.468e-05
2	0.071225	0.088916	1.2959e-10	7.3988e-05	7.3988e-05	3.2393e-05
3	0.10778	0.16014	4.0123e-12	6.8197e-05	6.8197e-05	2.8861e-05
4	0.013918	0.17011	1.1197e-08	8.414e-05	8.4144e-05	3.6703e-05
5	0.016311	0.18402	1.5781e-09	0.00014161	0.00014161	6.2126e-05
6	0.018203	0.2	6.4656e-10	0.00015292	0.00015292	6.7616e-05

Table 6.8: Estimates of local error for the linearized Allen Cahn equation with $N = 50$ points per dimension computed using the adaptive time stepping algorithm. The starting time step size estimate used was $\triangle t = 0.04$ seconds.

Number of Time Steps	$\triangle t$	t	High Frequency Error	Low Frequency Error	Relative Residual	Residual
1	0.04	0.04	0	0.00010967	0.00010967	5.468e-05
2	0.04	0.08	8.013e-10	4.5163e-05	4.5164e-05	1.9773e-05
3	0.04	0.12	1.1396e-10	2.2638e-05	2.2638e-05	9.592e-06
4	0.04	0.16	1.2766e-11	3.5129e-05	3.5129e-05	1.4944e-05
5	0.04	0.2	2.6902e-11	0.00066233	0.00066233	0.00028887

Table 6.9: Estimates of local error for the linearized Allen Cahn equation with $N = 50$ points per dimension computed without the use of adaptive time stepping.

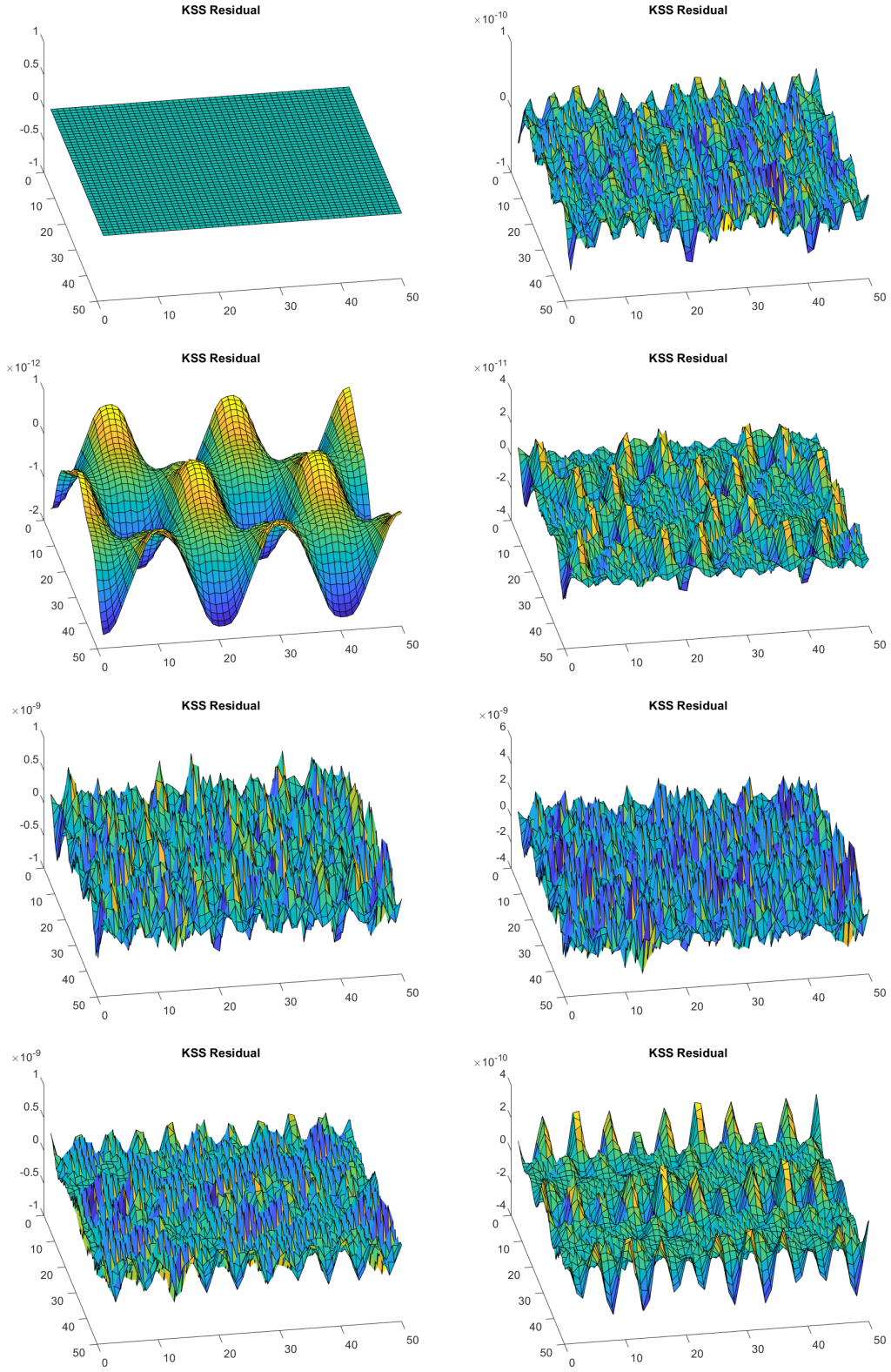

Figure 6.12: The residual from using KSS on the high frequency components at each time step when adaptive time stepping was used. Here $N = 50$ and $t = 0.01, 0.032196, 0.057856, 0.075929, 0.11258, 0.1363, 0.17295, 0.2$.

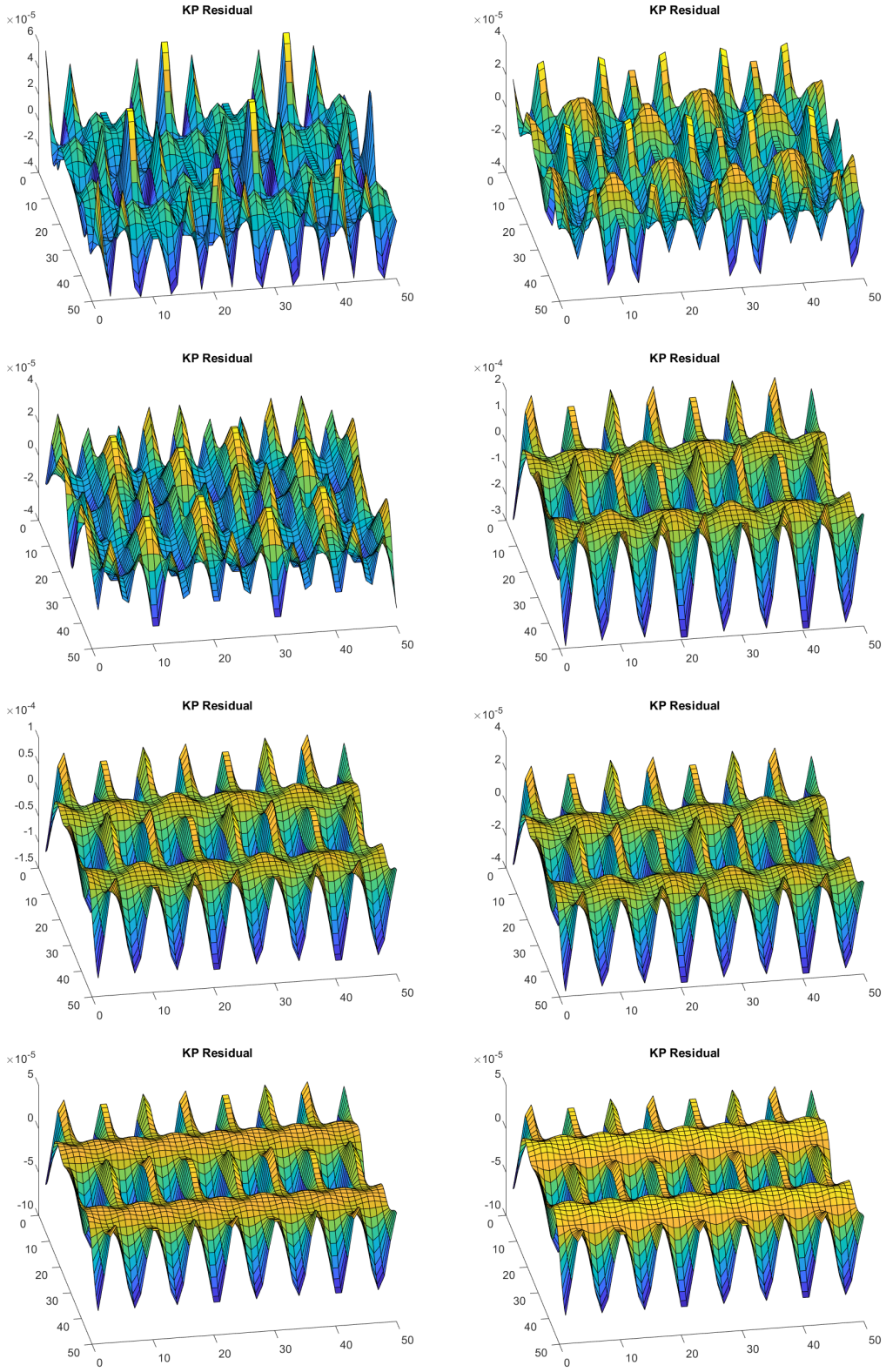

Figure 6.13: The computed residual from Kylov Projection on the the low frequency components at each time step when adaptive time stepping was used. Here $N = 50$ and $t = 0.01,\ 0.032196,\ 0.057856,\ 0.075929,\ 0.11258,\ 0.1363,\ 0.17295,\ 0.2$.

Number of Time Steps	$\triangle t$	t	High Frequency Error	Low Frequency Error	Relative Residual	Residual
1	0.014032	0.01	0	7.2417e-05	7.2417e-05	3.6197e-05
2	0.015222	0.024032	1.9321e-06	0.0001642	0.00016441	7.8321e-05
3	0.01691	0.039254	2.4758e-07	0.00016087	0.00016078	7.2951e-05
4	0.017466	0.056164	9.6755e-08	0.00020694	0.00020693	9.0742e-05
5	0.01926	0.07363	8.4663e-08	0.00017383	0.00017382	7.4578e-05
6	0.022229	0.092891	5.0941e-08	0.00015327	0.00015325	6.5049e-05
7	0.045867	0.11512	2.5124e-08	2.6917e-05	2.6904e-05	1.1383e-05
8	0.088932	0.16099	8.2834e-12	3.2321e-05	3.2321e-05	1.3719e-05
9	0.013073	0.17074	1.0453e-07	9.5143e-05	9.5162e-05	4.1533e-05
10	0.015087	0.18381	1.2874e-07	0.00014819	0.00014822	6.5057e-05
11	0.016986	0.1989	9.8652e-08	0.00015845	0.00015847	7.0065e-05
12	0.0058312	0.2	3.5864e-07	1.3146e-06	1.5105e-06	6.7334e-07

Table 6.10: Estimates of local error for the linearized Allen Cahn equation with $N = 150$ points per dimension computed using the adaptive time stepping algorithm. The starting timestep size estimate used was $\triangle t = 0.04$ seconds.

Number of Time Steps	$\triangle t$	t	High Frequency Error	Low Frequency Error	Relative Residual	Residual
1	0.04	0.04	0	0.010403	0.010403	0.0052
2	0.04	0.08	1.5325e-08	0.00063215	0.00063215	0.00027686
3	0.04	0.12	1.1245e-09	0.00099291	0.0009929	0.00042066
4	0.04	0.16	3.3118e-11	0.00011567	0.00011567	4.9205e-05
5	0.04	0.2	4.9632e-10	0.00073846	0.00073846	0.00032211

Table 6.11: Estimates of local error for the linearized Allen Cahn equation with $N = 150$ points per dimension computed without the use of adaptive time stepping.

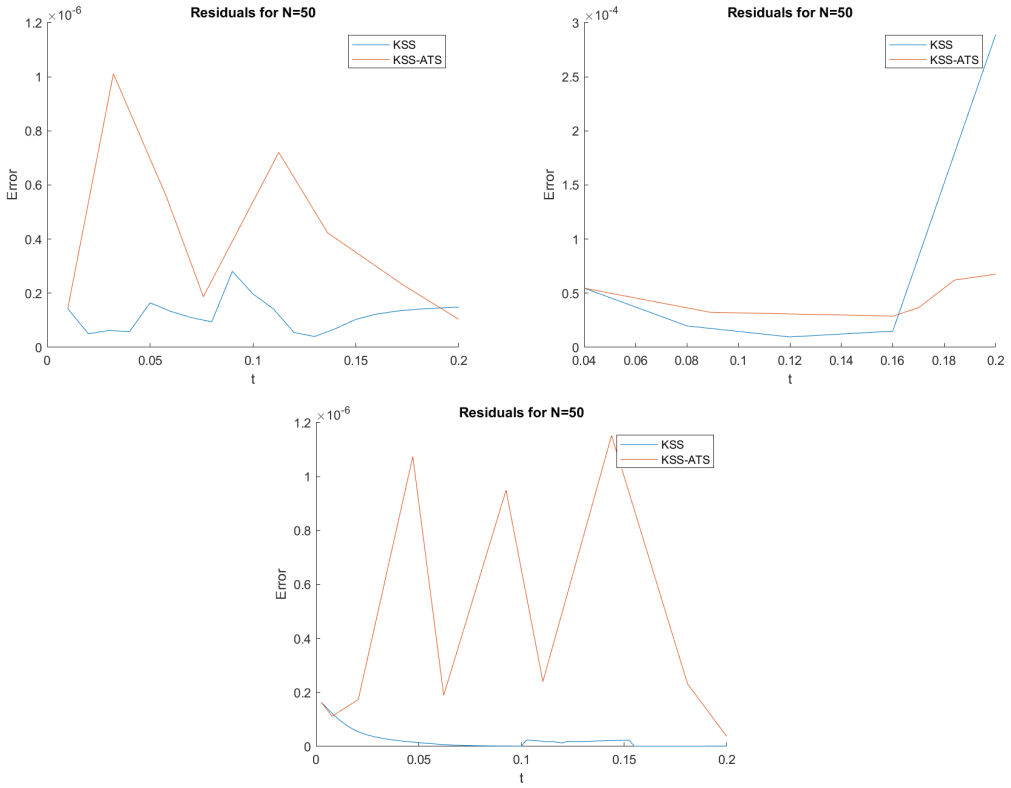

Figure 6.14: A comparison of execution time (t) and the size of the residual at that time. The top left figure contains the residual for the initial starting step size, the next figure is for $1/4$ of initial starting step size, and the last figure is for $1/16$ of the initial starting step size.

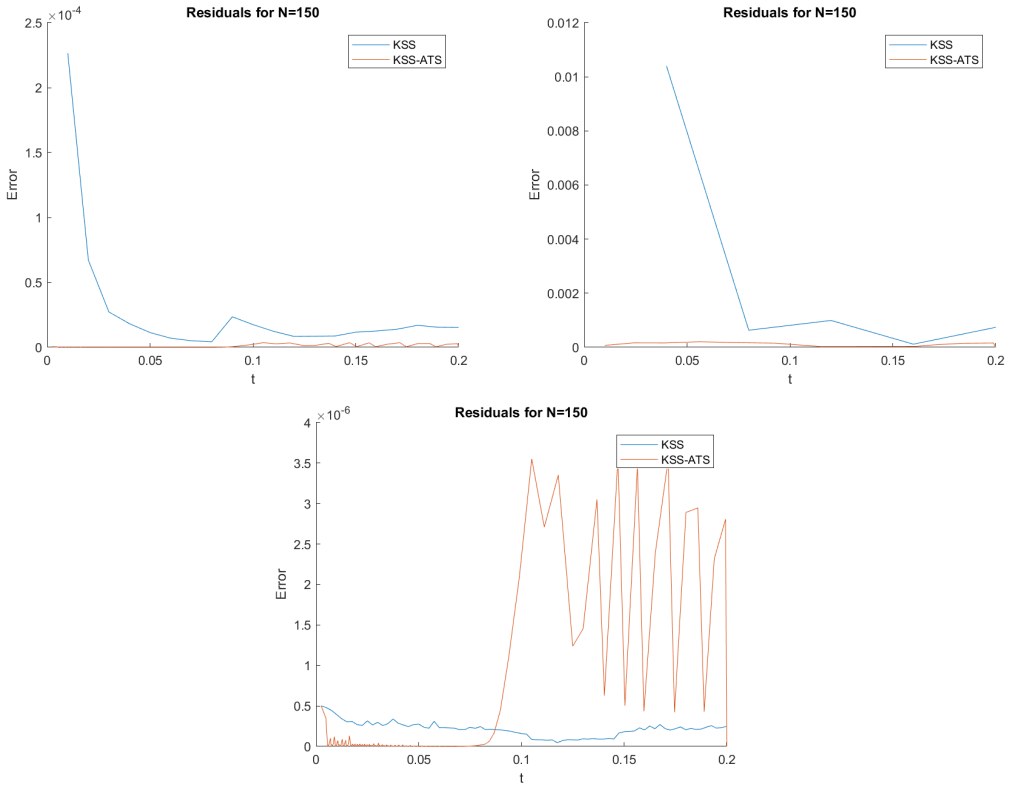

Figure 6.15: A comparison of execution time (t) and the size of the residual at that time. The top left figure contains the residual for the initial starting step size, the next figure is for $1/4$ of initial starting step size, and the last figure is for $1/16$ of the initial starting step size.

6.5 Adaptive Time Stepping Case 1: Performance

In this section we compare the performance of KSS-ATS with KSS as described in Chapter 2, Krylov Projection (KP), and Leja Interpolation (LEJA). Figure 6.16 depicts the relative error of the methods using varying starting time step sizes. Because of the chosen tolerance, KSS-ATS is not as accurate as KSS but is significantly more accurate than Krylov Projection (in most cases) and Leja interpolation. This phenomenon can also be seen in Figure 6.17 which depicts the relative error of the methods with the execution time. Although Krylov Projection does relatively well on low resolution grids, it does not continue to do well as the resolution increases. KSS without adaptive time stepping performs the best at the smallest time step sizes, but when the first time step size is large, KSS with adaptive time stepping has a much smaller relative error.

It should be noted that in previous sections KSS methods only required a Krylov subspace of dimension 4 for all time step sizes and on any grid. Since the results from this section are obtained from using KSS only on the high frequency components of the solution and Krylov

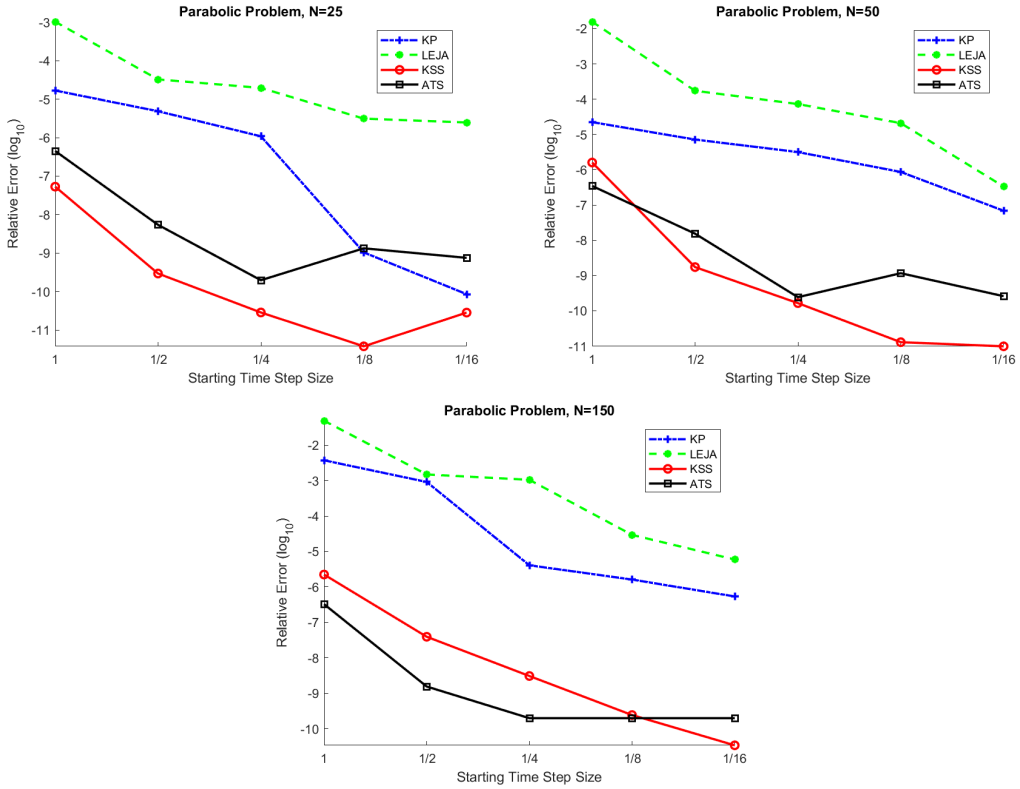

Figure 6.16: Relative Error (logarithmically scaled for varying starting time step sizes for the linearized Allen Cahn equation.)

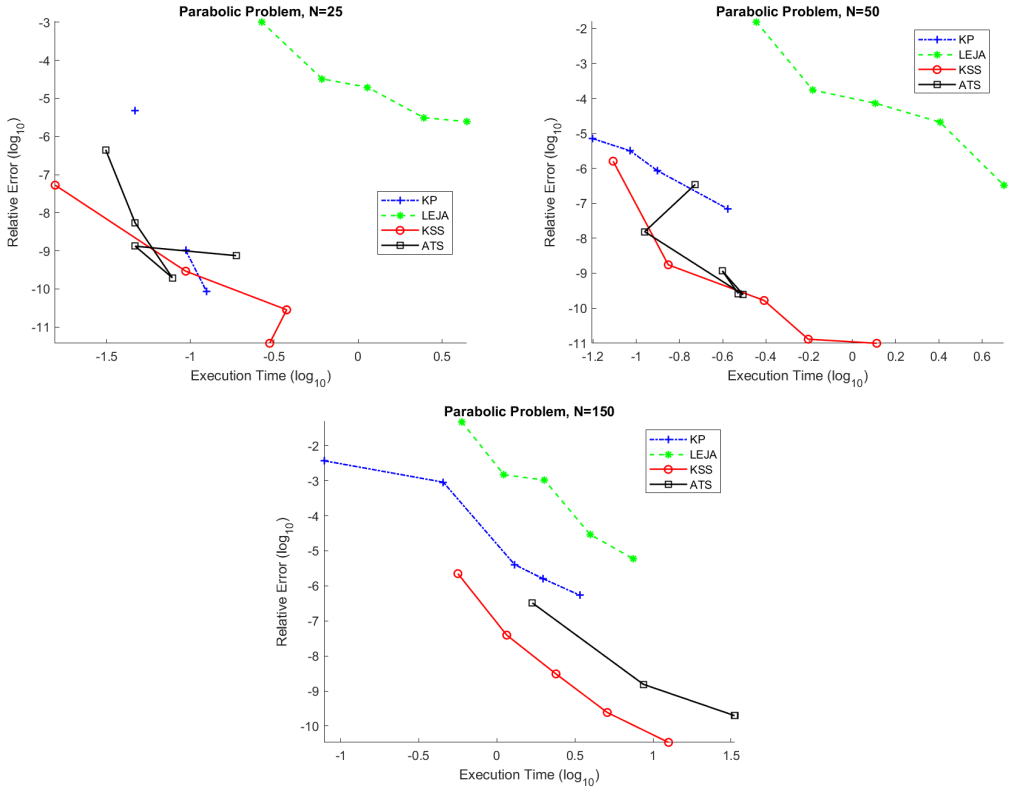

Figure 6.17: Relative Error versus execution time (logarithmically scaled) for varying starting time step sizes for the linearized Allen Cahn equation.)

Projection on the low frequency components, the average number of Lanczos iterations required will increase due to this high/low frequency split.

	KSS-ATS	KSS	KP	LEJA
	4.5001e-07	5.3121e-08	1.6791e-05	0.0010
$N = 25$	5.4086e-09	2.9308e-10	4.8585e-06	3.2654e-05
	1.9623e-10	2.8443e-11	1.0834e-06	1.9609e-05
	1.3268e-09	3.7545e-12	1.0376e-09	3.1185e-06
	7.4963e-10	2.8244e-11	8.5101e-11	2.4773e-06
	3.4654e-07	1.6025e-06	2.238e-05	0.015756
$N = 50$	1.5395e-08	1.7359e-09	7.1928e-06	0.00017377
	2.4321e-10	1.6514e-10	3.1733e-06	7.2937e-05
	1.1643e-09	1.2885e-11	8.6563e-07	2.0942e-05
	2.5983e-10	9.8412e-12	6.8766e-08	3.3349e-07
	3.2607e-07	2.2329e-06	0.0037182	0.048547
$N = 150$	1.551e-09	3.9276e-08	0.00090998	0.0014814
	1.9885e-10	3.0713e-09	4.0716e-06	0.0010469
	1.9885e-10	2.4359e-10	1.6116e-06	2.9298e-05
	1.9885e-10	3.4122e-11	5.3783e-07	5.9134e-06

Table 6.12: Relative Error for the linearized Allen Cahn equation for starting time step sizes $\triangle t = 0.04, 0.02, 0.01, 0.005, 0.0024$ seconds.

	KSS-ATS	KSS	KP	LEJA
	0.0156	0.0313	0	0.9219
N=25	0.0313	0.0156	0	0.5625
	0.0625	0.2500	0.03125	0.7969
	0.0313	0.2813	0.0781	1.8125
	0.0625	0.6250	0.0313	3.7344
	0.1875	0.09375	0	0.78125
N=50	0.09375	0.20313	0.10938	0.6875
	0.5	0.46875	0.0625	1.25
	0.17188	0.67188	0.25	2.625
	0.375	1.3281	0.67188	5.1563
	1.6563	0.60938	0.10938	0.6875
N=150	9.3906	1.1406	0.65625	1.1875
	33.516	2.1875	1.2344	2.0156
	33.031	4.5938	1.9688	3.8438
	33.047	8.625	3.4531	7.1875

Table 6.13: Execution time for linearized Allen Cahn Equation for starting time step sizes $\triangle t = 0.04, 0.02, 0.01, 0.005, 0.0024$ seconds.

	KSS-ATS	KSS	KP	LEJA
N=25	5.67	5.2	3.2	10
	5.7	5.5	3.2	10
	6.11	5.45	3.45	9
	7.17	5.15	3.65	10
	6.3	4.475	3.89	10
N=50	4.125	5	3.4	13.6
	5.8571	5.4	4.2	16
	6.6	5.6	3.75	14.3
	6.625	5.2	3.8	10.95
	6.3333	4.7375	3.8875	10
N=150	4.25	4.8	3	58
	4.3765	5.2	5.9	54.4
	4.135	5.4	7.55	30.2
	4.1292	5.2	6.55	27.1
	4.1235	4.7875	6.0125	24.163

Table 6.14: Average number of iterations for the linearized Allen Cahn Equation for starting time step sizes $\triangle t = 0.04, 0.02, 0.01, 0.005, 0.0024$ seconds.

6.6 Adaptive Time Stepping Case 1: Inclusion of a Minimum Step Size

In previous sections, the step size controller used for adaptive time stepping allowed the time steps to become as big or as small as needed to reach the set error tolerance. This caused the number of time steps for problems with a large grid size ($N = 150$) to grow very large. Allowing the step size controller to decrease the time step size to a very small number was very helpful in terms of accuracy but it became extremely inefficient.

Figures 6.18 and 6.19 compare the relative error against the total execution time and the relative error against the starting time step size. The minimum step sizes in these figures are h_{min}= 0.02, 0.01 and 0.005. From these figures, we can see that although there is a slight disadvantage to choosing a minimum step size in terms of accuracy, there is a large advantage in terms of efficiency of the method.

From Figure 6.18 it can be observed that choosing a minimum step size reduces the execution time of the algorithm significantly, especially on high resolution grids ($N = 150$). This is due to the fact that setting a minimum step size reduces the amount of time steps

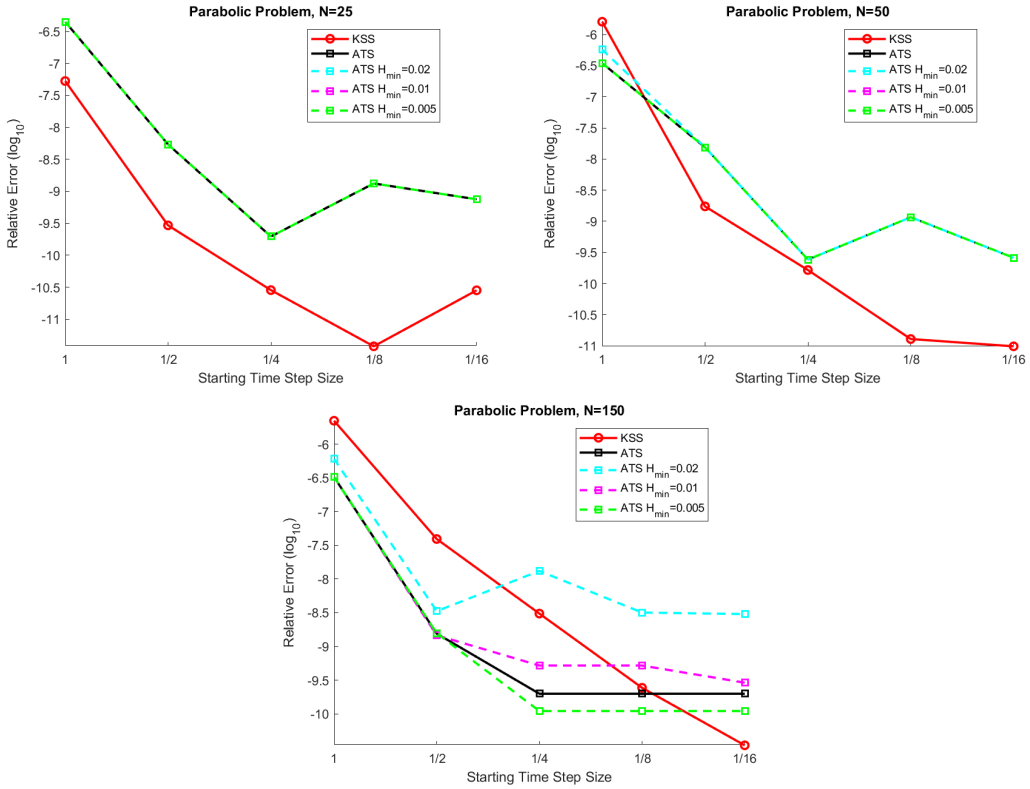

Figure 6.18: Relative Error (logarithmically scaled) for varying starting time step sizes for the linearized Allen Cahn equation.

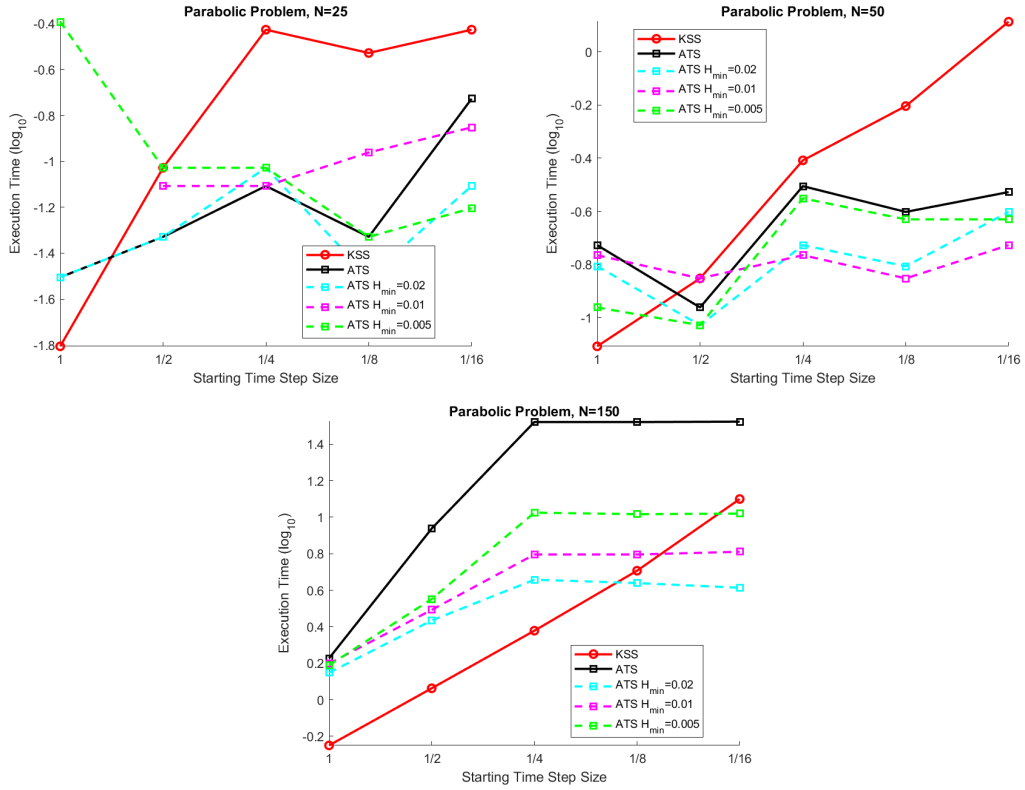

Figure 6.19: Execution time (logarithmically scaled) for varying starting time step sizes for the linearized Allen Cahn equation.

used. In general, as the minimum step size increases, the total number of time steps decrease but this also increases the relative error slightly.

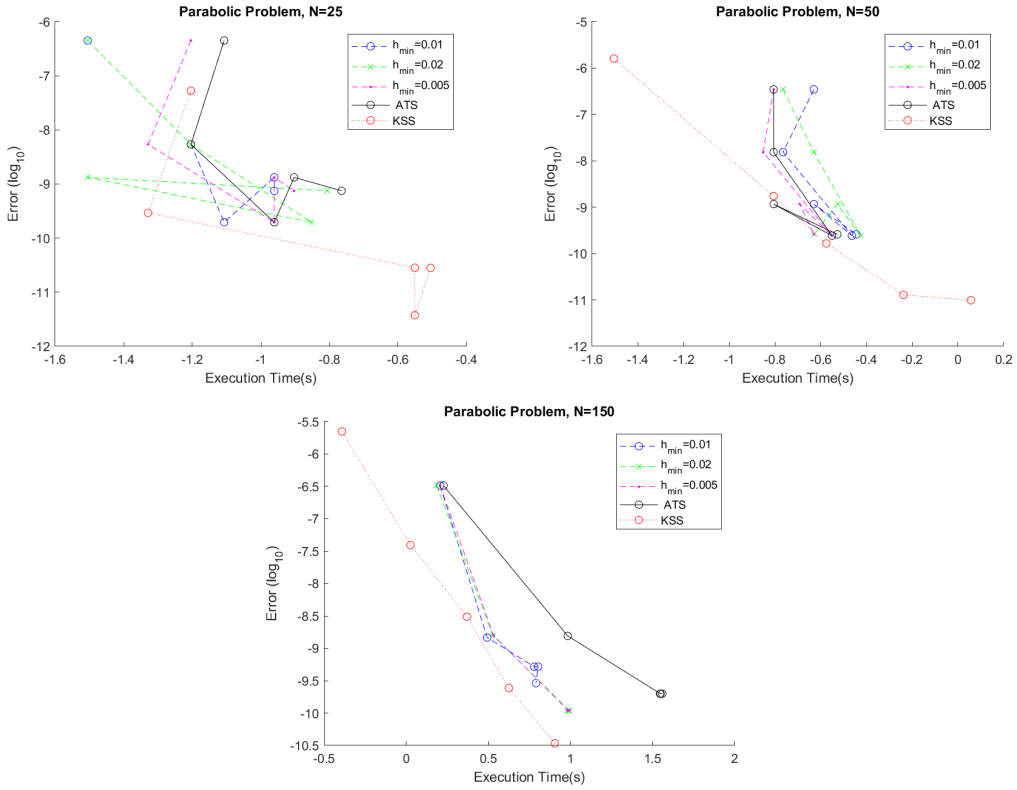

Figure 6.20: Relative error vs execution time (logarithmically scaled) for varying starting time step sizes for the linearized Allen Cahn equation.

6.7 Adaptive Time Stepping Case 1: Without the High and Low Frequency Split

In this section we consider KSS methods without the high/low frequency split described in [1]. Like in the previous section we will also consider the grid resolution sizes $N = 25,\ 50,$ and 150. In the graphs the label KSS-ATS2 and KSS2 will depict the experimental results obtained from the KSS method with and without adaptive time stepping method and without the frequency split.

Recall from Section 6.4, when the high and low frequency components are split, most of the error per time step comes from performing Krylov Projection on the low frequency components of the solution. When we no longer perform this split, in general, KSS is less accurate which can be observed when comparing Tables 6.15 and 6.12. The accuracy results of implementing adaptive time stepping are approximately the same but the execution times of adaptive time stepping without splitting the high and low frequency components is significantly less as shown in 6.23.

Figures 6.25, 6.26, and 6.27 depict the time step sizes with different starting time step

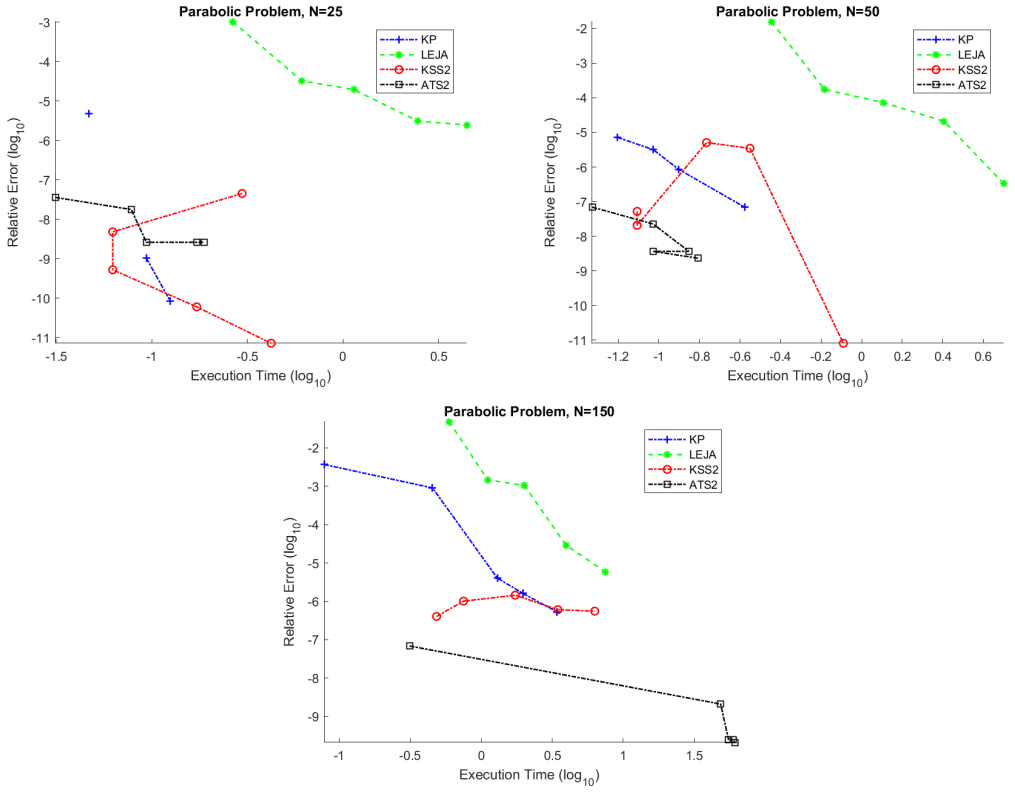

Figure 6.21: A logarithmically scaled (log_{10}) comparison of execution time and relative error between KSS-ATS, KSS, KP, and LEJA.

values with grid sizes $N = 25,\ 50$, and 150 respectively. As with previous experiments, for grid sizes $N = 25$ and $N = 50$ the total number of time steps taken when using adaptive time stepping are relatively small compared to KSS without adaptive time stepping but when the grid size increases to $N = 150$ KSS with adaptive time stepping loses its advantage.

When comparing the number of time steps for KSS-ATS2 for each starting time step size in Figure 6.26 to the number of time steps for KSS-ATS in Figure 6.6, we can see that the total number of time steps required is less when the high/low frequency split is performed. This implies that the execution time for KSS-ATS2 is less than that of KSS-ATS.

Next we will consider using a minimum step size to counteract the significant increase in number of time steps observed in high resolution grid sizes. Table 6.15 contains the relative error on grid sizes $N = 25,\ 50$, and 150 for starting time steps $\triangle t = 0.04$, 0.02, 0.01, 0.005, and 0.0025. The relative error KSS-ATS2 with a minimum step size is approximately the same as KSS-ATS2 without a minimum step size except at the largest grid size. The benefit of setting a minimum step size is particularly apparent when examining the execution times of all 3 methods at a large grid size ($N = 150$). When adaptive time stepping with a

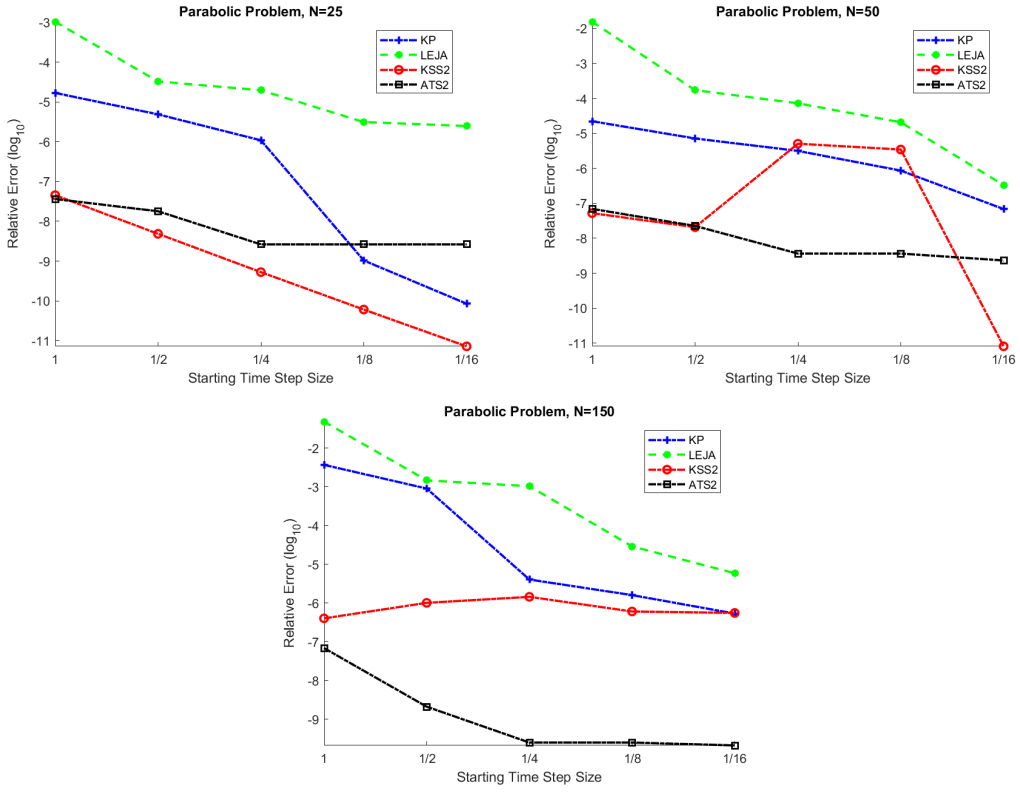

Figure 6.22: A logarithmically scaled (log_{10}) comparison of starting time step size and relative error between KSS-ATS, KSS, KP, and LEJA.

minimum step size is implemented, the execution time at smaller starting time step sizes is ten times faster.

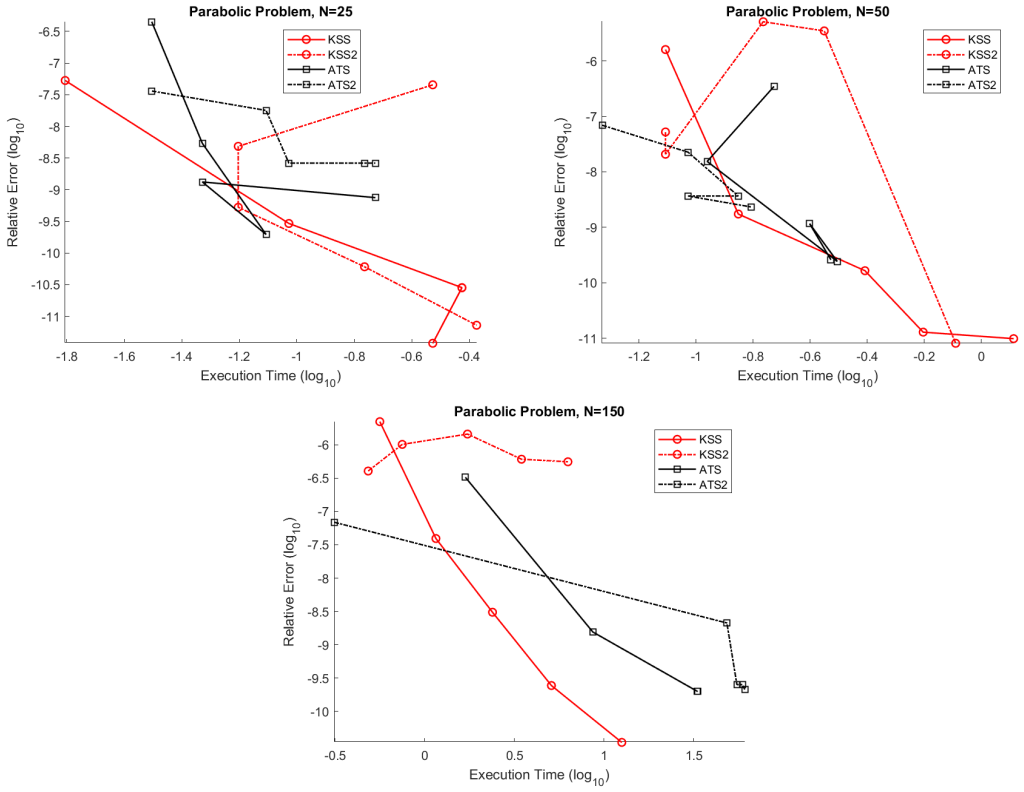

Figure 6.23: A logarithmically scaled (log_{10}) comparison between execution time and relative error for KSS-ATS and KSS with and without the frequency split.

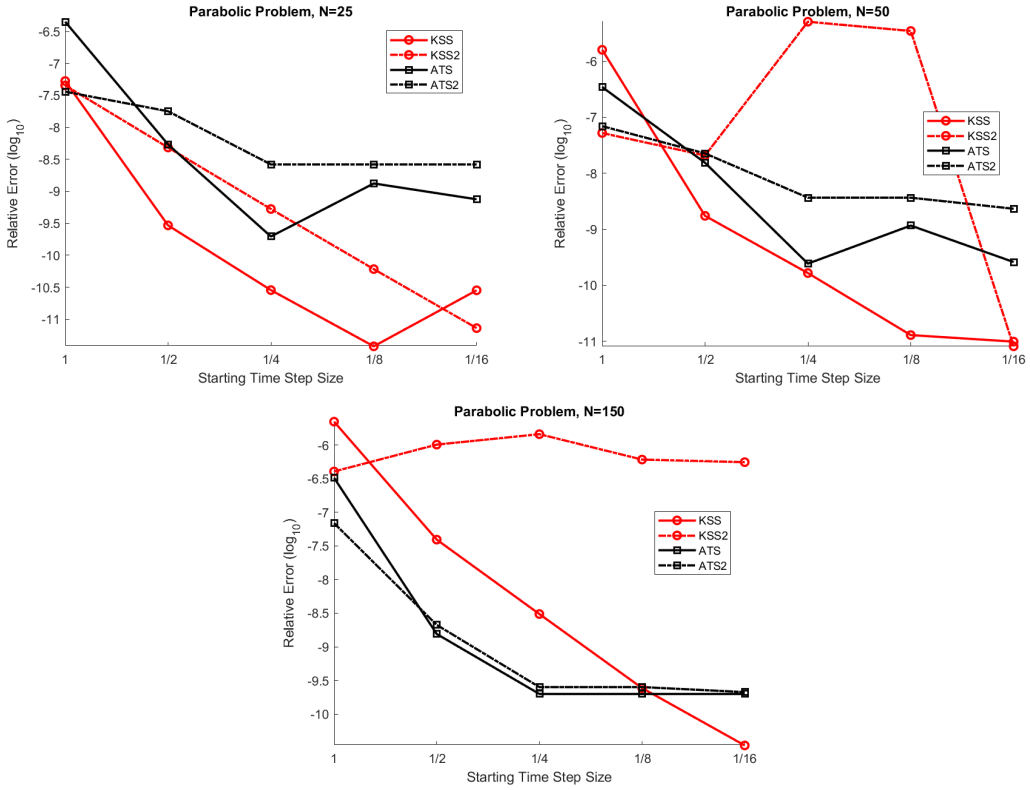

Figure 6.24: A logarithmically scaled (log_{10}) comparison between starting time step size and relative error for KSS-ATS and KSS with and without the frequency split.

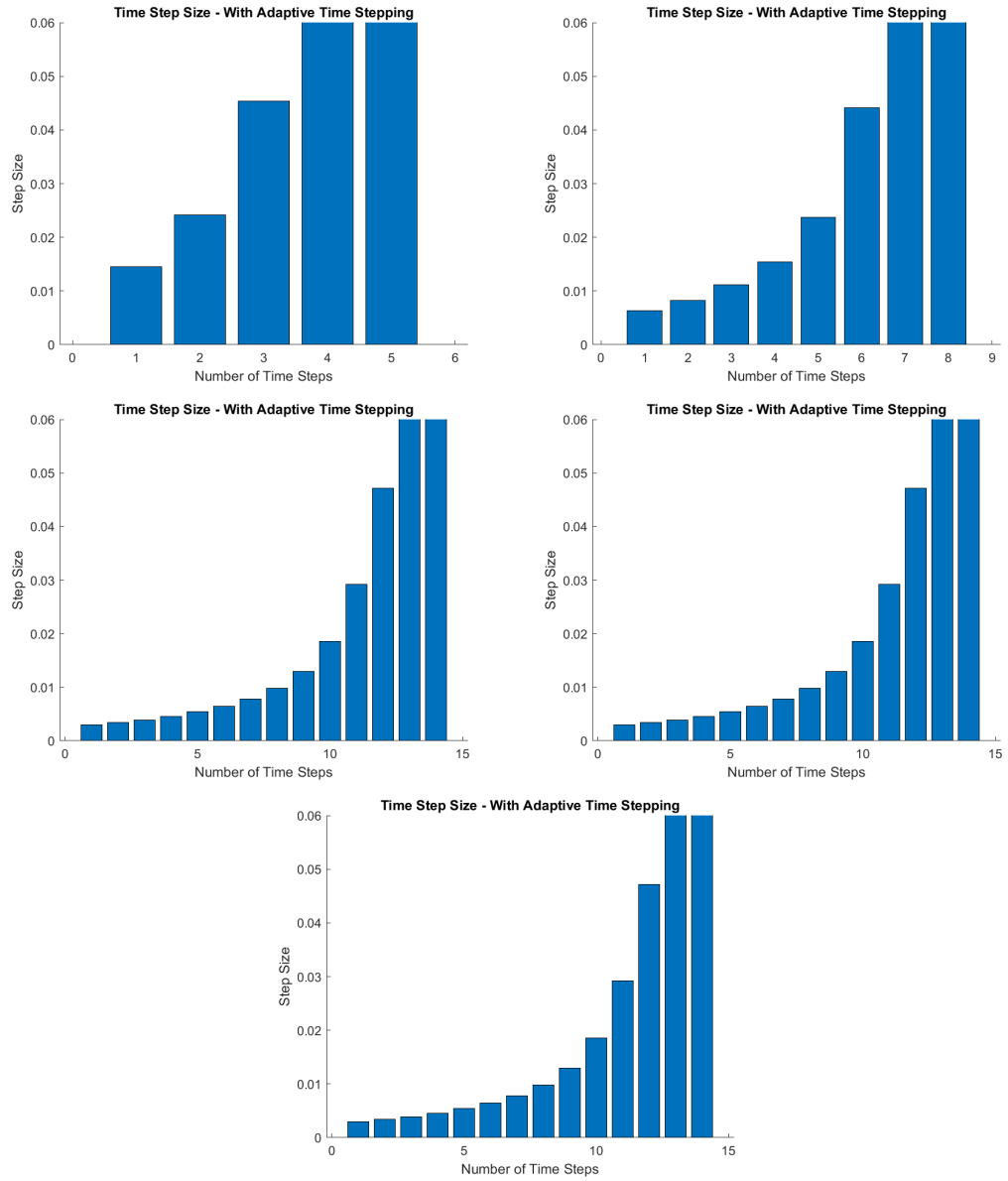

Figure 6.25: Time step size ($\triangle t$) for each time step for using KSS using adaptive time stepping without splitting the high and low frequency components (KSS-ATS2). $N = 25$ points per dimension. From first figure (top left) to last (bottom right) the starting time step sizes are $\triangle t = 0.04, 0.02, 0.01, 0.005, 0.0025$ seconds.

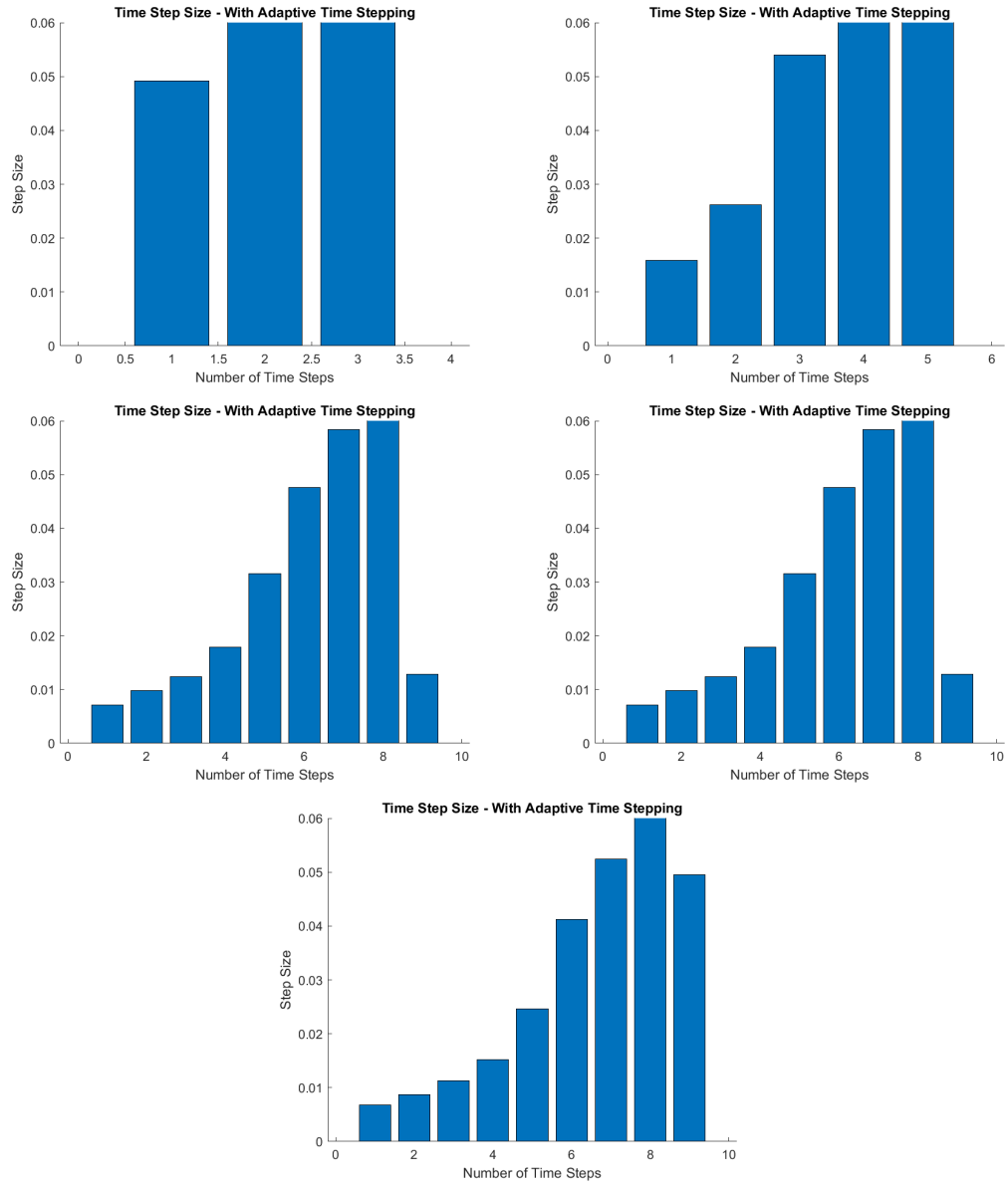

Figure 6.26: Time step size ($\triangle t$) for each time step for using KSS using adaptive time stepping without splitting the high and low frequency componenets. $N = 50$ points per dimension.From first figure (top left) to last (bottom right) the starting time step sizes are $\triangle t = 0.04, 0.02, 0.01, 0.005, 0.0025$ seconds.

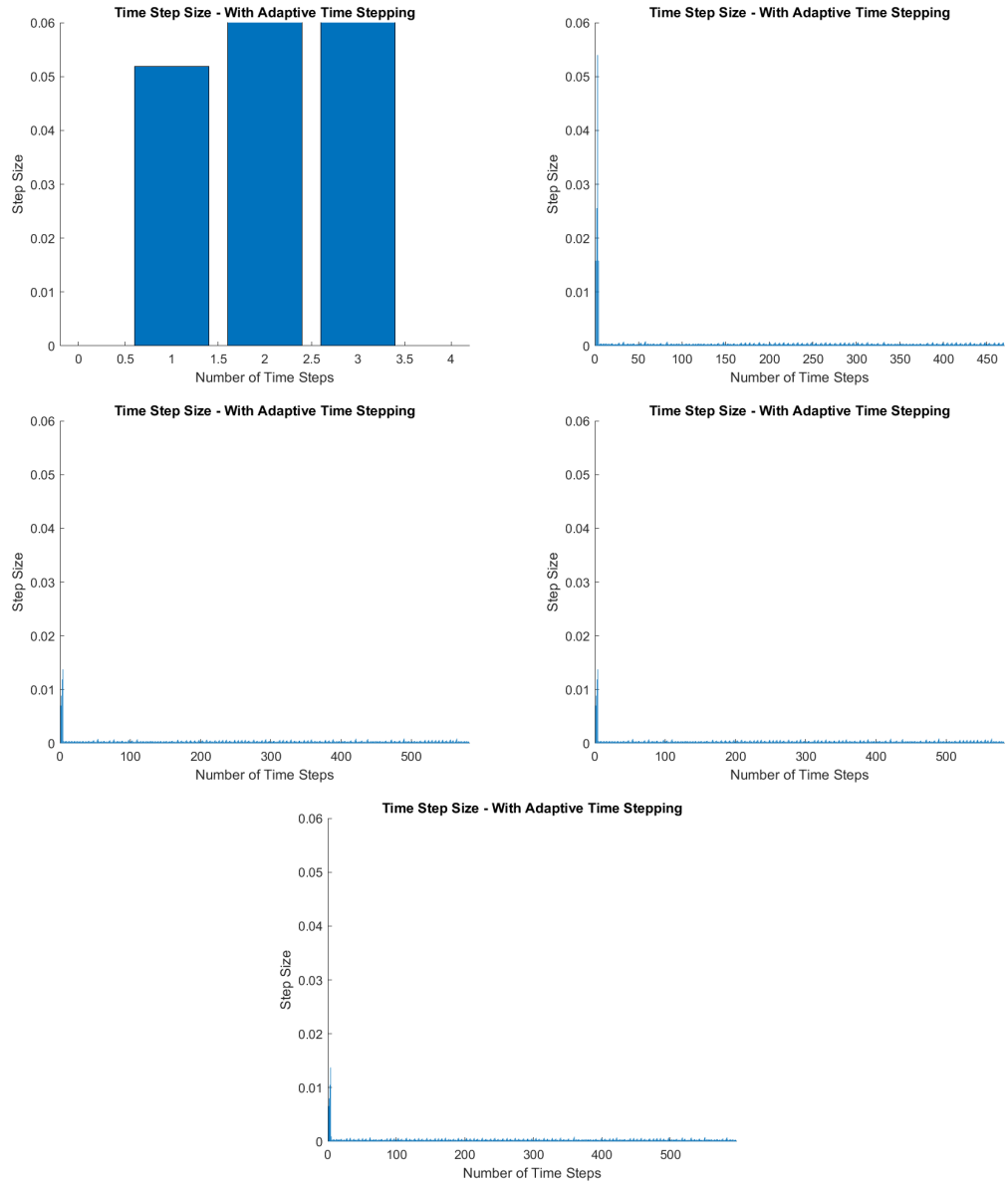

Figure 6.27: Time step size ($\triangle t$) for each time step for using KSS using adaptive time stepping without splitting the high and low frequency componenets. $N = 150$ points per dimension.From first figure (top left) to last (bottom right) the starting time step sizes are $\triangle t = 0.04, 0.02, 0.01, 0.005, 0.0025$ seconds.

	KSS-ATS2	KSS2	KSS-ATS2 with h_{min}=0.01
N=25	3.6242e-08	4.537e-08	3.6242e-08
	1.7882e-08	4.8474e-09	1.7882e-08
	2.6315e-09	5.2853e-10	3.5713e-09
	2.6315e-09	6.0639e-11	3.5713e-09
	2.6315e-09	7.2363e-12	2.6315e-09
N=50	6.9356e-08	5.2352e-08	6.9356e-08
	2.2615e-08	2.0787e-08	2.2615e-08
	3.6632e-09	5.0852e-06	3.6632e-09
	3.6632e-09	3.4751e-06	3.6632e-09
	2.3256e-09	8.1189e-12	2.3256e-09
N=150	6.9074e-08	4.05e-07	6.9074e-08
	2.1176e-09	1.017e-06	8.6758e-07
	2.5167e-10	1.454e-06	8.4545e-07
	2.5167e-10	6.0926e-07	8.4545e-07
	2.1183e-10	5.5772e-07	3.8163e-07

Table 6.15: Relative Error for KSS with and without adaptive time stepping without the high/low frequency split for starting time step sizes $\triangle t = 0.04, 0.02, 0.01, 0.005, 0.0024$ seconds.

	KSS-ATS2	KSS2	KSS-ATS2 with h_{min}=0.01
N=25	4	4	4
	4	4	4
	4	4	4
	4	4	4
	4	4	4
N=50	4	4	4
	4	4	4
	4	4	4
	4	4	4
	4	4	4
N=150	4	4	4
	4	4	4
	4	4	4
	4	4	4
	4	4	4

Table 6.16: Number of Iterations for KSS with and without adaptive time stepping without the high/low frequency split for starting time step sizes $\triangle t = 0.04, 0.02, 0.01, 0.005, 0.0024$ seconds.

	KSS-ATS2	KSS2	KSS-ATS2 with h_{min}=0.01
N=25	0.0625	0	0.078125
	0.046875	0.078125	0.078125
	0.0625	0.10938	0.17188
	0.125	0.28125	0.14063
	0.078125	0.40625	0.09375
N=50	0.046875	0.03125	0.046875
	0.046875	0.10938	0.078125
	0.125	0.3125	0.078125
	0.10938	0.34375	0.09375
	0.10938	0.70313	0.125
N=150	0.20313	0.42188	0.1875
	36.313	0.67188	3.2188
	45.516	1.125	3.6406
	44.344	2.4219	3.625
	46.453	5.5938	3.6719

Table 6.17: Execution time per time step size for KSS with and without adaptive time stepping without the high/low frequency split for starting time step sizes $\triangle t = 0.04$, 0.02, 0.01, 0.005, 0.0024 seconds.

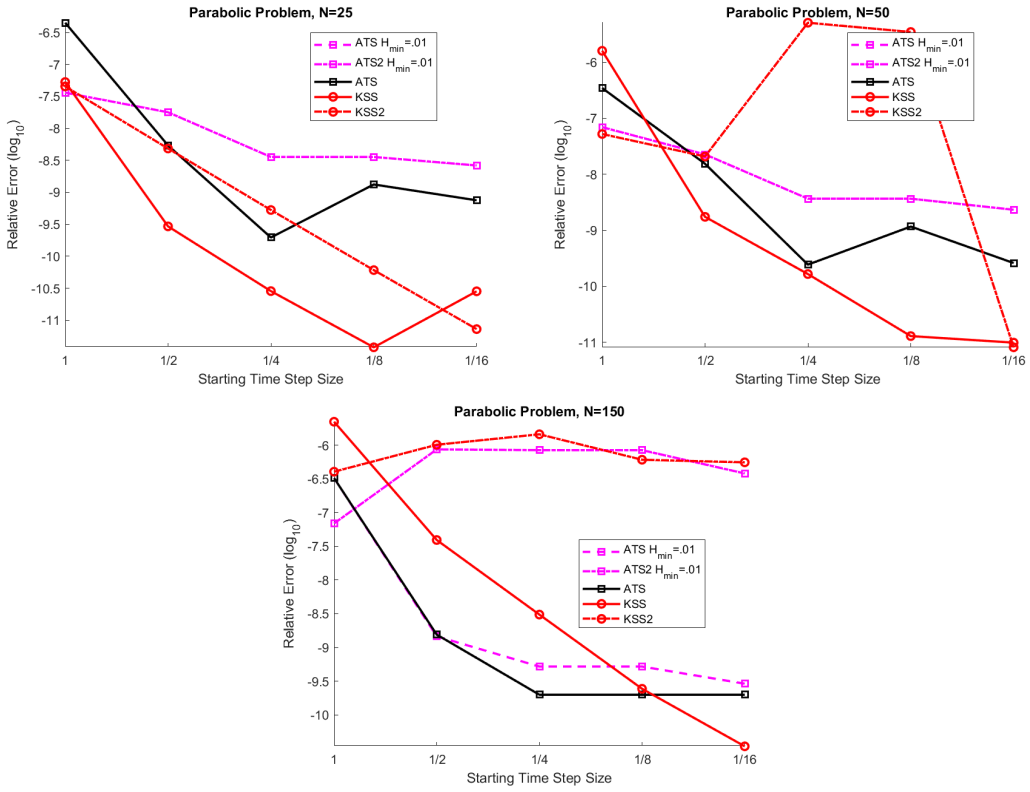

Figure 6.28: A logarithmically scaled (log_{10}) comparison of starting time step size and relative error between ATS including a minimum step size, ATS2 including a minimum step size, ATS, KSS2, and ATS2

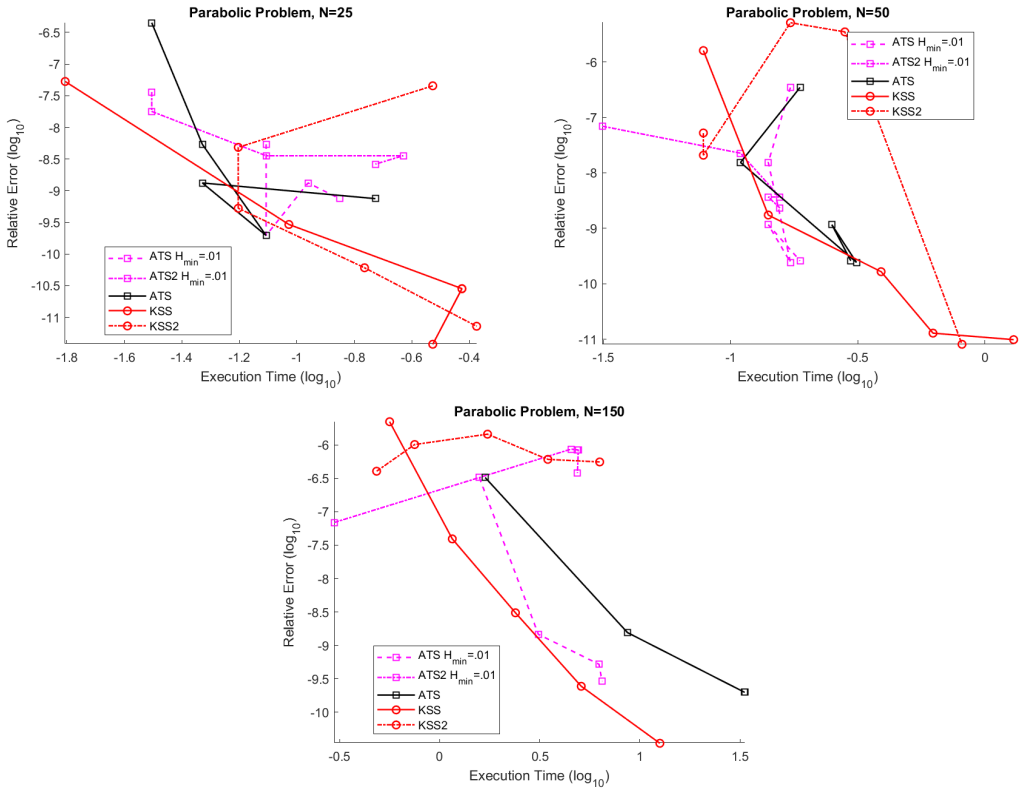

Figure 6.29: A logarithmically scaled (log_{10}) comparison of execution time and relative error between ATS including a minimum step size, ATS2 including a minimum step size, ATS, KSS2, and ATS2

6.8 Krylov Subspace Spectral Methods with Adaptive Time Stepping Case 2: 2-D Linear Systems of Equations

In this section we consider a linearized version of the Brusselator problem:

$$u_t = 1 + uv^2 - 4u + \alpha \nabla^2 u, \quad x, y \varepsilon [0,1],\ t \varepsilon [0,.1], \tag{6.13}$$

$$v_t = 3u - u^2 v + \alpha \nabla^2 u, \tag{6.14}$$

with $\alpha = 0.2$ and homogeneous Dirichlet boundary conditions. The initial data is

$$u_0 = \quad \sin(5\pi x)\sin(3\pi y) \tag{6.15}$$

$$v_0 = \quad \sin(7\pi x)\sin(4\pi y). \tag{6.16}$$

We begin by examining the computed solution of the linearized Brusselator system using the time step sizes specified in the first plot in Figure 6.33. Figure 6.30 are plots of the solution u (labeled component 1) and Figure 6.31 are plots of the solution v (labeled component 2). Since the Brusselator system of equations is a reaction diffusion type of problem (as is the linearized version) the solutions for both components become less oscillatory over time. Like the case with the linearized Allen Cahn equation discussed in the last section, a diffusive problem implies that an effective step size controller would decrease the time step size for the first few time steps then slowly allow the time step sizes to grow larger as the solutions become less oscillatory. This is the case, as seen in Figures 6.32, 6.33, 6.35 and 6.37. Figures 6.32 and 6.37 depict the final time after each time step for the grid resolution sizes $N = 50$ and $N = 150$ using KSS with and without adaptive time stepping. In both figures, between $t = 0.05$ and $t = 0.06$ seconds, the slope of the curve becomes steeper (athough it is more pronounced in Figure 6.32 due to the lower count of time steps). This indicates that the solution smooths out enough for the error estimation to be low enough to increase the time step by a large amount.

Similar to the linearized Allen Cahn equation, when the resolution of the problem increases from $N = 50$ to $N = 150$, adaptive time stepping becomes less efficient due to the need of smaller time steps but in this case this effect is not as impactful. By comparing figures 6.33 and 6.35, we can see that for $N = 50$ as the starting time step size changes from $\triangle t = 0.01$ to $\triangle t = 0.000625$ the number of time steps taken are 32, 42, 37, 45, and 61 . When $N = 150$ points per dimension, the number of time steps taken are 39, 67, 91, 194, and 270 (see table 6.18). Although there is a significant increase between the number of time steps required on the lower resolution grid and a higher resolution grid, this increase is not as pronounced as it was for the linearized Allen Cahn problem.

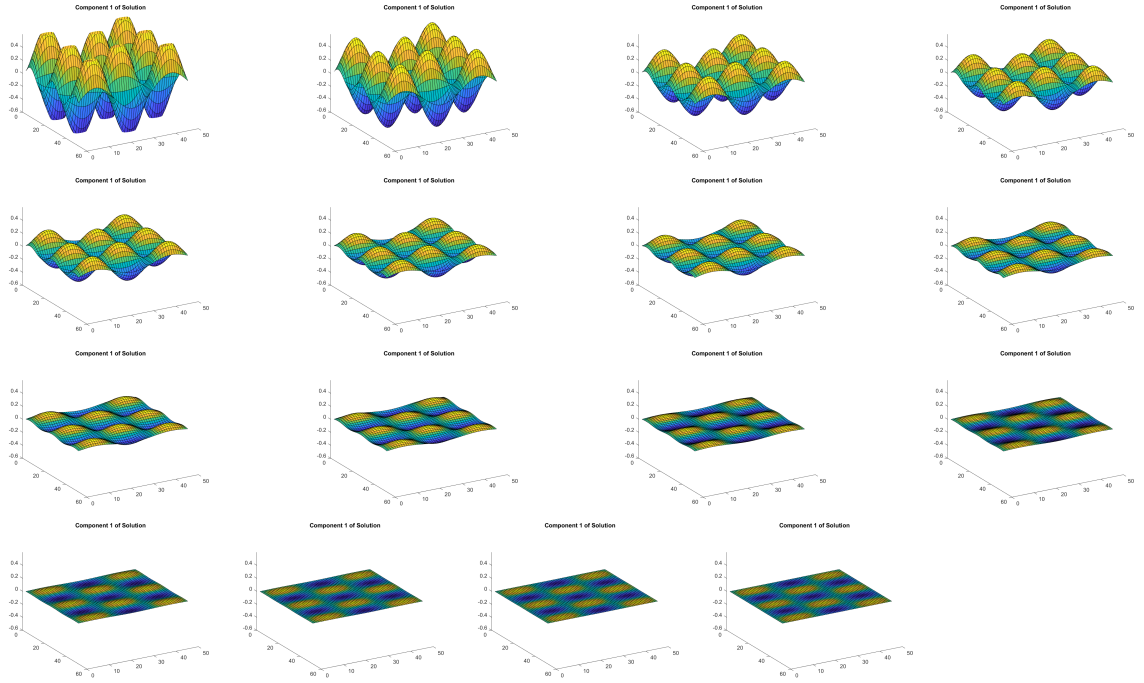

Figure 6.30: The first component of the solution to the Brusselator system of equations with $N = 50$ points per dimension.

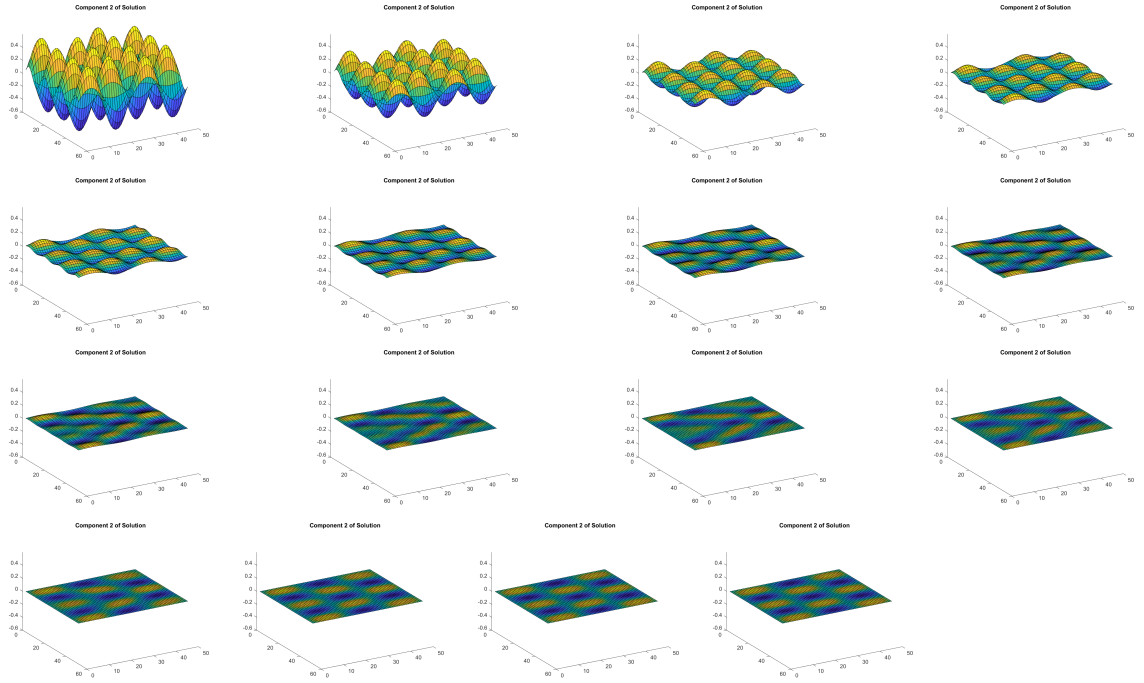

Figure 6.31: The second component of the solution to the Brusselator system of equations with $N = 50$ points per dimension.

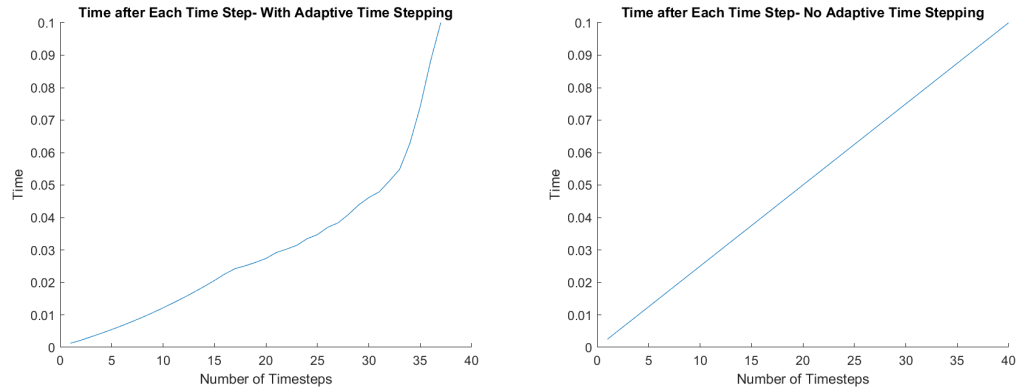

Figure 6.32: A comparison of the final time after each time step for KSS with adaptive time stepping on the left and KSS without adaptive time stepping on the right. The starting time step for both is $\triangle t = 0.0025$ and there are $N = 50$ points per dimension.

	Starting Time Step Size	$N = 50$	$N = 150$
Linearized Brusselator	.01	32	39
	.005	42	67
	.0025	37	91
	.00125	45	194
	.000625	61	270
Linearized Allen Cahn	.04	6	12
	.02	6	66
	.01	8	253
	.005	8	253
	.0025	10	253

Table 6.18: Total number of time steps taken for the grid sizes $N = 50$ and $N = 150$ for both the linearized Allen Cahn problem and the linearized Brusselator problem

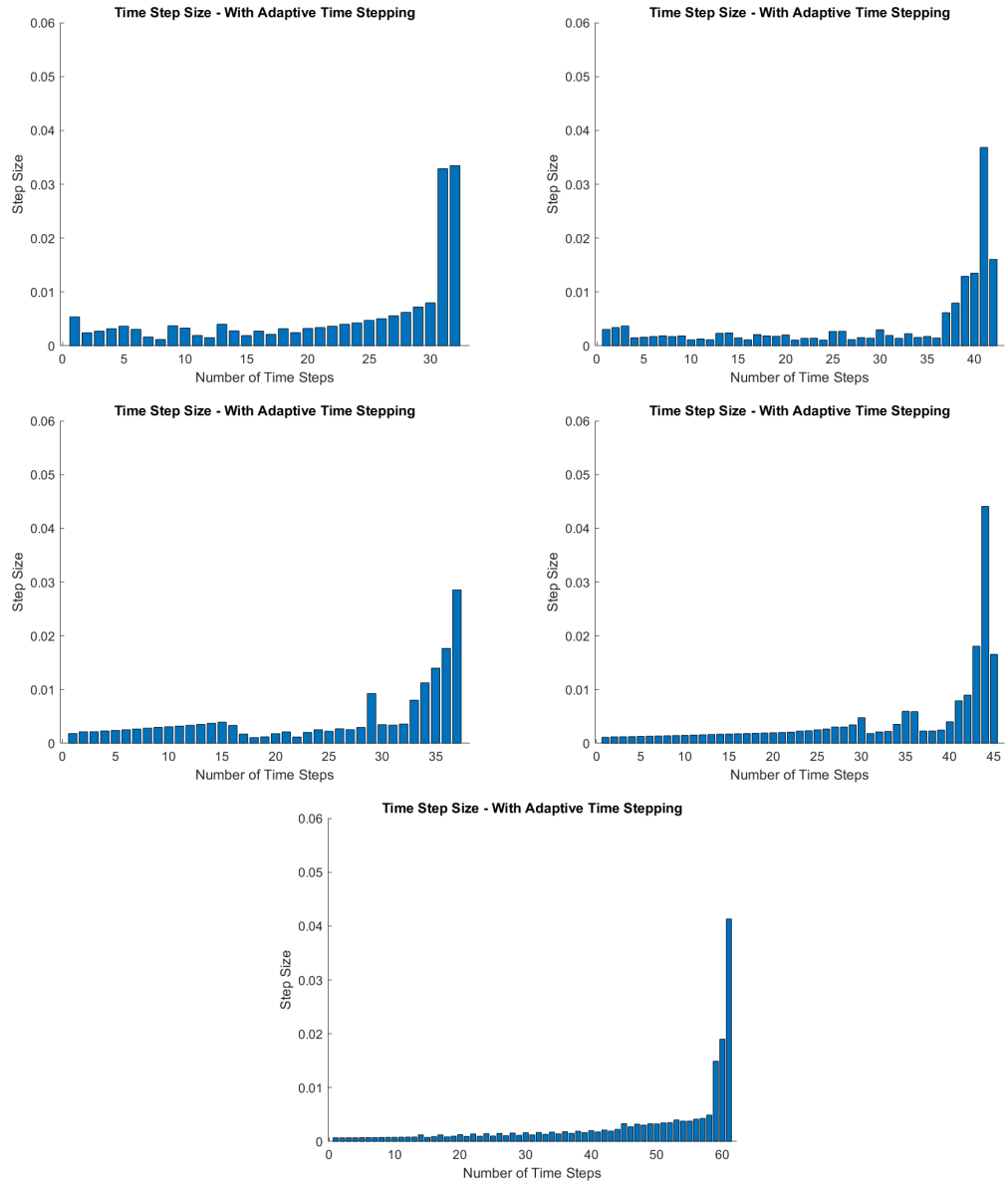

Figure 6.33: Time step size ($\triangle t$) for each time step for using KSS with adaptive time stepping with $N = 50$ points per dimension on the linearized Brusselator system of equations. From first figure (top left) to last (bottom right) the starting time steps are $\triangle t = 0.01, 0.005, 0.0025, 0.00125, 0.000625$ seconds.

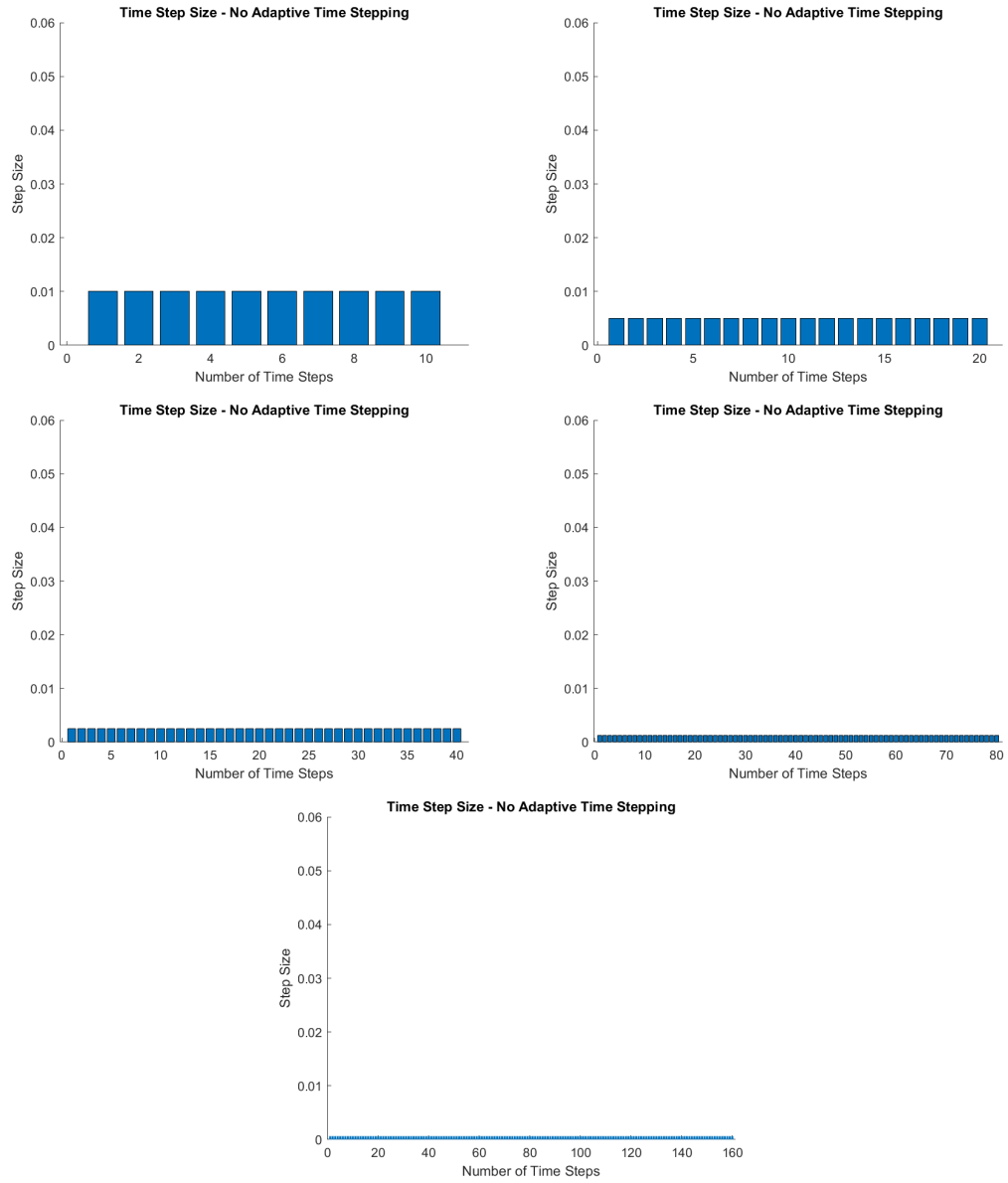

Figure 6.34: Time step size ($\triangle t$) for each time step for using KSS without adaptive time stepping with $N = 50$ points per dimension on the linearized Brusselator system of equations. From first figure (top left) to last (bottom right) the time step sizes are $\triangle t = 0.01, 0.005, 0.0025, 0.00125, 0.000625$ seconds.

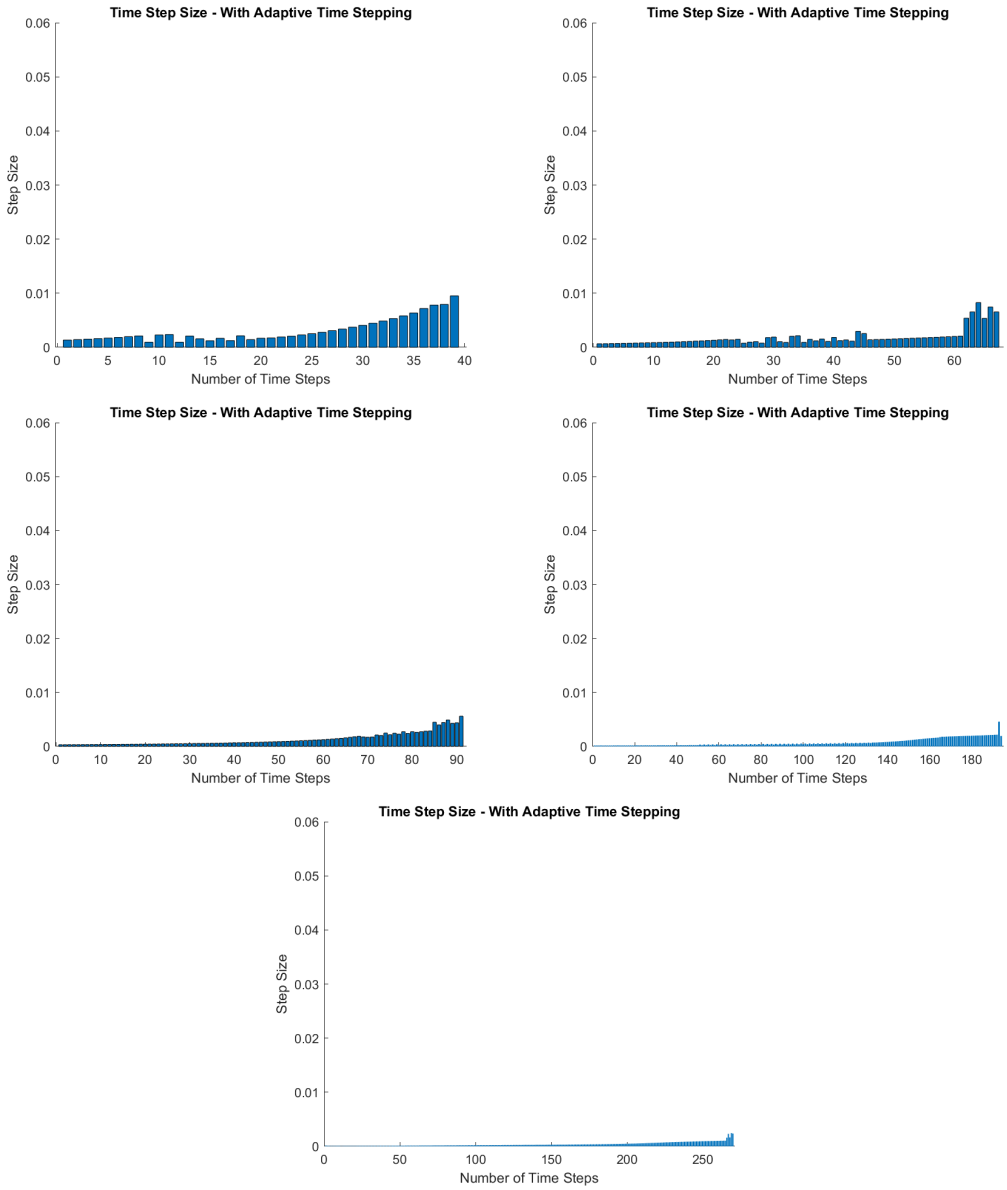

Figure 6.35: Time step size ($\triangle t$) for each time step for using KSS with adaptive time stepping with $N = 150$ points per dimension on the linearized Brusselator system of equations. From first figure (top left) to last (bottom right) the starting time steps are $\triangle t = 0.01, 0.005, 0.0025, 0.00125, 0.000625$ seconds.

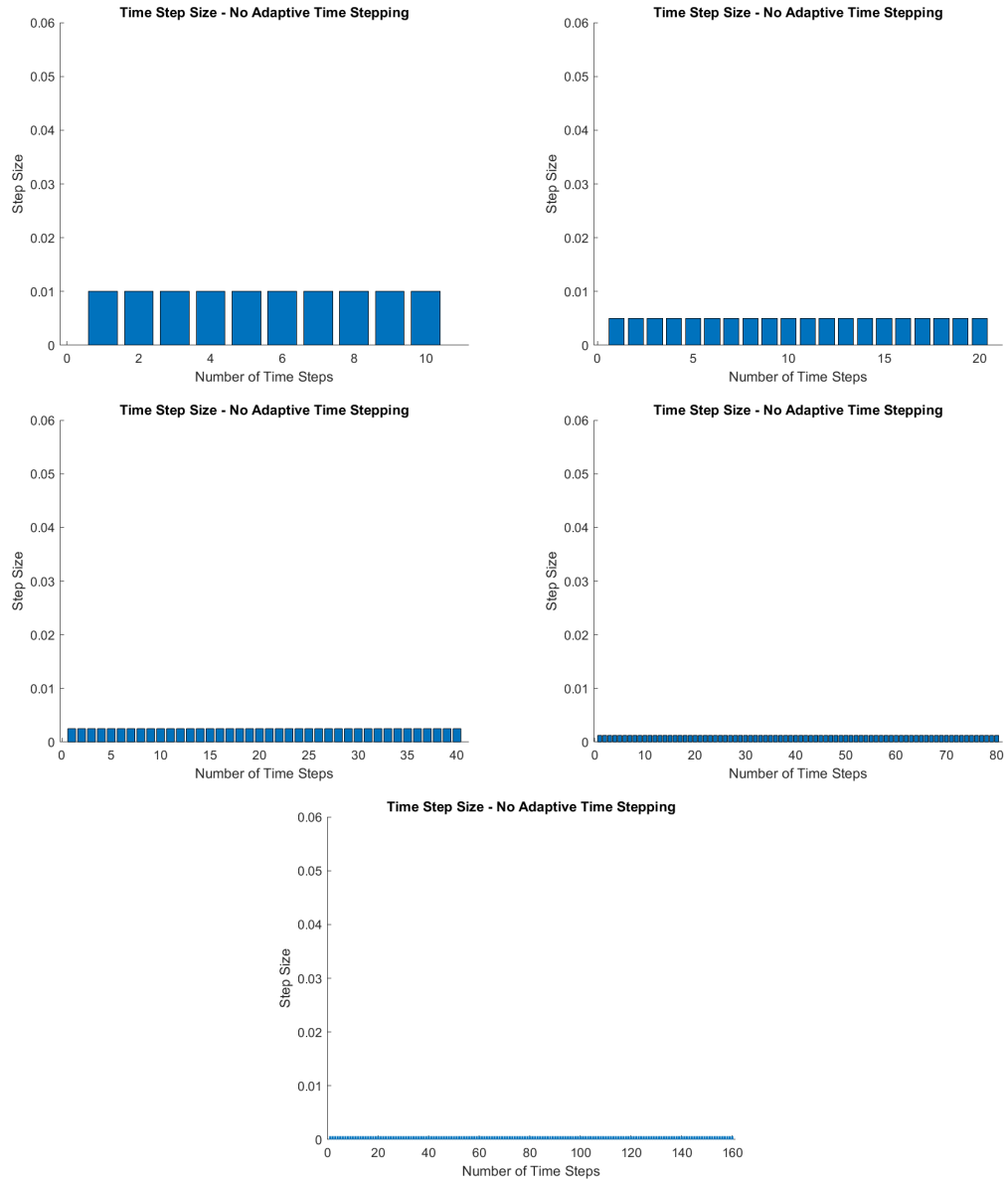

Figure 6.36: Time step size ($\triangle t$) for each time step for using KSS without adaptive time stepping with $N = 150$ points per dimension on the linearized Brusselator system of equations. From first figure (top left) to last (bottom right) the time step sizes are $\triangle t = 0.01, 0.005, 0.0025, 0.00125, 0.000625$ seconds.

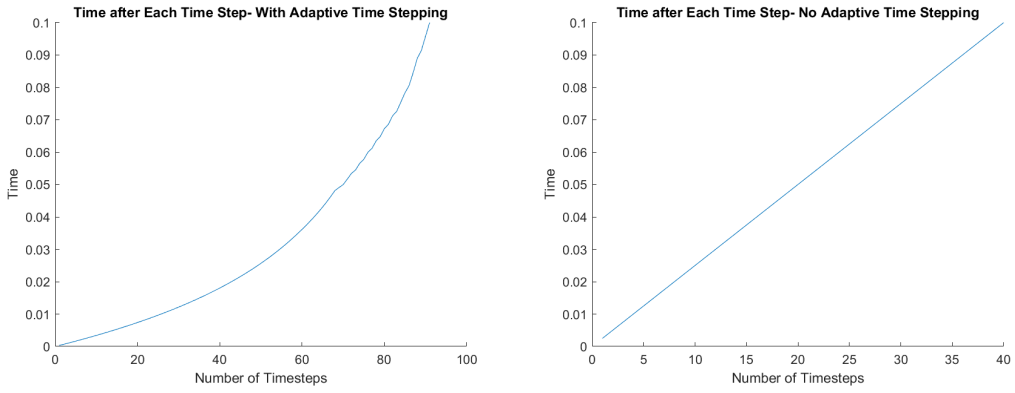

Figure 6.37: A comparison of the final time after each time step for KSS with adaptive time stepping on the left and KSS without adaptive time stepping on the right. The starting time step for both is $\triangle t = 0.0025$ and there are $N = 150$ points per dimension.

6.9 Adaptive Time Stepping Case 2: Comparison of Residuals

Similar to the linearized Allen Cahn problem, we can define the residual for the linearized Brusselator problem to be

$$R = y_t - Ly \tag{6.17}$$

where L is a matrix of differential operators and y is the matrix containing the solutions u and v.

Tables 6.19, 6.20 contain error estimates for grid resolution $N = 50$ points per dimension obtained from equations (6.9) through (6.12). At the bottom of each table, the average error estimate for each column is calculated using the following formula

$$Avg = \frac{\sum_{j=1}^{n} err_j}{n} \tag{6.18}$$

where Avg is the average, err_j is the error estimate of the j^{th} time step, and n is the total number of time steps. By comparing the average low frequency error and average relative error, we observe that when using either KSS with or without adaptive time stepping these values are the same. This is due to the use of Krylov Projection on the low frequencies. Since the low frequency error is much larger than the high frequency error (which the error obtained by using KSS on high frequencies), the low frequency error term influences the residual size the most. This implies that the adaptive step sizes are most influenced by the error obtained on the low frequencies. From Tables 6.19 and 6.20, we can also see that although the average high frequency error for KSS without adaptive time stepping is smaller than with adaptive time stepping, the average low frequency error for KSS without adaptive time stepping is much higher. This is due to the fact that the low frequency error is the dominating error in the error approximation that is used for the step size controller. Therefore the adaptive time stepping algorithm for KSS is essentially minimizing the low frequency error.

Tables 6.21, 6.22 contain error estimates for grid resolution $N = 150$ points per dimension obtained from equations equations (6.9) through (6.12). Similar to when $N = 50$, we can observe from Tables 6.21 and 6.22 that the low error for low frequencies is much higher than the error for high frequencies. Therefore, even on a grid with higher resolution, the low frequency error dominates the error approximation for the step size controller. Therefore the step size for adaptive time stepping increases or decreases the time step based on low frequency error.

Number of Time Steps	$\triangle t$	t	High Frequency Error	Low Frequency Error	Relative Residual	Residual
1	0.005383	0.005	0	8.0216e-05	8.0216e-05	8.014e-05
2	0.0024354	0.0063457	1.7143e-08	2.4087e-05	2.4086e-05	1.6872e-05
3	0.0027282	0.0087811	2.3591e-09	0.00011163	0.00011163	7.1133e-05
4	0.0031773	0.011509	1.1061e-09	0.00011794	0.00011794	6.3306e-05
5	0.0036657	0.014687	9.9021e-10	0.00014704	0.00014704	6.512e-05
6	0.0030715	0.01652	2.6907e-09	6.0029e-05	6.0029e-05	2.1248e-05
7	0.0016577	0.018055	4.4567e-10	0.00025566	0.00025566	7.9515e-05
8	0.0011923	0.018884	4.7895e-09	0.00012038	0.00012038	3.3595e-05
9	0.0037236	0.020077	9.5342e-11	1.2473e-05	1.2473e-05	3.2831e-06
10	0.0033127	0.021938	1.8653e-10	7.3367e-05	7.3367e-05	1.7752e-05
11	0.0019387	0.023595	2.1215e-09	0.00029393	0.00029393	6.2359e-05
12	0.0015348	0.024564	1.6773e-10	0.00013349	0.00013349	2.5195e-05
13	0.004028	0.026099	7.1555e-10	3.1389e-05	3.1389e-05	5.5323e-06
14	0.0027929	0.028113	2.0886e-10	0.00023711	0.00023711	3.7497e-05
15	0.0019239	0.029509	3.3973e-09	0.00027876	0.00027876	3.8239e-05
16	0.0027467	0.031433	3.3298e-10	0.00027652	0.00027652	3.4369e-05
17	0.002139	0.032807	3.6342e-09	0.00024394	0.00024394	2.6467e-05
18	0.0031928	0.034946	2.7279e-10	0.00030535	0.00030535	3.0068e-05
19	0.0024669	0.036542	3.8284e-09	0.00032008	0.00032008	2.7099e-05
20	0.0032407	0.039009	1.9433e-10	0.00058317	0.00058317	4.4109e-05
21	0.0034114	0.04225	6.273e-10	0.0013491	0.0013491	8.5731e-05
22	0.0036482	0.045661	1.6483e-10	0.0016176	0.0016176	8.1766e-05
23	0.0040052	0.049309	1.0675e-10	0.001902	0.001902	7.5568e-05
24	0.0042689	0.053314	1.2645e-10	0.0026893	0.0026893	8.2592e-05
25	0.0047453	0.057583	1.0304e-10	0.0031448	0.0031448	7.2803e-05
26	0.0050494	0.062329	9.1454e-11	0.0048451	0.0048451	8.3002e-05
27	0.0055872	0.067378	7.0785e-11	0.0060208	0.0060208	7.3811e-05
28	0.0062184	0.072965	5.7828e-11	0.0084456	0.0084456	7.2535e-05
29	0.0072213	0.079184	7.6489e-13	0.01102	0.01102	6.3856e-05
30	0.007987	0.086405	4.9475e-13	0.019759	0.019759	7.3908e-05
31	0.032902	0.094392	6.6318e-13	0.00063548	0.00063548	1.4305e-06
32	0.033467	0.1	7.8963e-11	0.0003662	0.0003662	4.7051e-07
Average	.005464	0.039349	1.44e-09	0.002047	0.002047	4.84e-05

Table 6.19: Estimates of local error for the linearized Brusselator system of equations with $N = 50$ points per dimension computed using the adaptive time stepping algorithm. The starting timestep size estimate used was $\triangle t = 0.01$ seconds.

Number of Time Steps	$\triangle t$	t	High Frequency Error	Low Frequency Error	Relative Residual	Residual
1	0.01	0.01	0	0.00069523	0.00069523	0.00069457
2	0.01	0.02	3.664e-11	0.0020302	0.0020302	0.0010001
3	0.01	0.03	1.6405e-12	0.016124	0.016124	0.0039226
4	0.01	0.04	1.7709e-12	0.013857	0.013857	0.0016635
5	0.01	0.05	1.9582e-12	0.010965	0.010965	0.00064962
6	0.01	0.06	4.9312e-13	0.013701	0.013701	0.00040069
7	0.01	0.07	9.8791e-13	0.022048	0.022048	0.00031847
8	0.01	0.08	9.085e-13	0.013962	0.013962	9.9669e-05
9	0.01	0.09	1.9351e-12	0.031082	0.031082	0.00010974
10	0.01	0.1	3.9761e-13	0.0017904	0.0017904	3.1296e-06
Average	0.01	0.055	4.67e-12	0.012625	0.012625	0.000886

Table 6.20: Estimates of local error for the linearized Brusselator system of equations with $N = 50$ points per dimension computed without using the adaptive time stepping algorithm. The starting time step size used was $\triangle t = 0.01$ seconds.

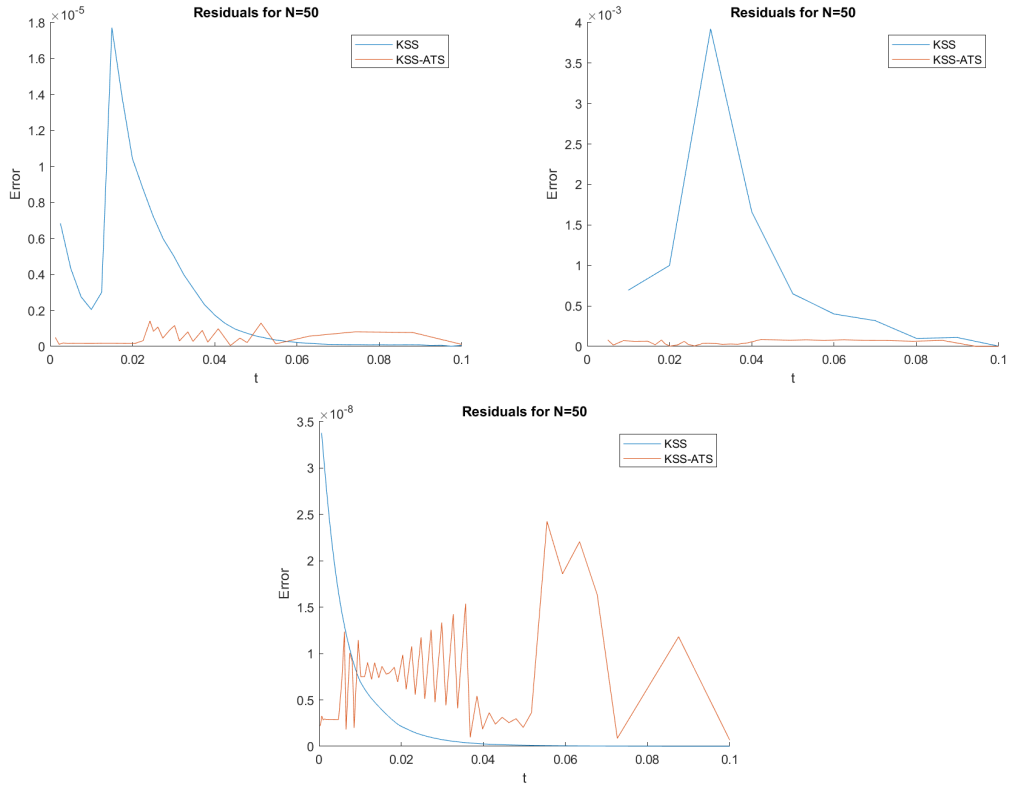

Figure 6.38: A comparison of execution time (t) and the size of the residual at that time. The top left figure contains the residual for the initial starting step size, the next figure is for $1/4$ of initial starting step size, and the last figure is for $1/16$ of the initial starting step size.

Number of Time Steps	$\triangle t$	t	High Frequency Error	Low Frequency Error	Relative Residual	Residual
1	0.0013443	0.00125	0	8.0398e-05	8.0398e-05	8.0389e-05
2	0.0014235	0.0025943	6.8483e-08	9.1958e-05	9.2003e-05	8.4223e-05
3	0.0015094	0.0040179	6.5758e-08	0.00010072	0.00010077	8.3887e-05
4	0.0016074	0.0055273	5.7363e-08	0.00010996	0.00010999	8.28e-05
5	0.0017167	0.0071347	4.8535e-08	0.00012131	0.00012134	8.2087e-05
6	0.0018415	0.0088514	4.0426e-08	0.00013416	0.00013419	8.1014e-05
7	0.0019814	0.010693	3.2845e-08	0.0001502	0.00015018	8.0289e-05
8	0.0021082	0.012674	2.5149e-08	0.00017691	0.00017692	8.3012e-05
9	0.00095752	0.013201	1.0743e-06	4.0913e-05	4.0906e-05	1.6677e-05
10	0.0022806	0.014159	3.3472e-08	1.8855e-05	1.8844e-05	7.4007e-06
11	0.0023693	0.01644	3.0028e-09	0.00024306	0.00024306	8.9188e-05
12	0.00095649	0.017032	3.2404e-06	7.6162e-05	7.6089e-05	2.3749e-05
13	0.0020911	0.017988	2.3762e-08	3.198e-05	3.1978e-05	9.5699e-06
14	0.0015892	0.019034	2.5842e-08	0.00010184	0.00010183	2.8476e-05
15	0.0012156	0.019829	1.3932e-07	0.00010759	0.00010758	2.7931e-05
16	0.0016646	0.021044	7.8131e-09	0.00015872	0.00015872	3.8948e-05
17	0.0012413	0.021876	1.1394e-07	0.00013392	0.00013391	3.0143e-05
18	0.0021271	0.023118	5.4357e-09	9.3663e-05	9.3663e-05	1.9873e-05
19	0.0014277	0.024181	4.8448e-08	0.00021278	0.00021277	4.1337e-05
20	0.0016817	0.025609	4.6353e-09	0.00033969	0.00033969	6.1194e-05
21	0.0017363	0.027291	5.5063e-09	0.00055812	0.00055812	9.0851e-05
22	0.0019159	0.029027	8.6356e-10	0.00051527	0.00051527	7.4434e-05
23	0.0020636	0.030943	1.0179e-09	0.00062673	0.00062673	8.0033e-05
24	0.0022956	0.033006	1.9073e-10	0.00065175	0.00065175	7.2641e-05
25	0.0025205	0.035302	1.8753e-10	0.00078475	0.00078475	7.5542e-05
26	0.002778	0.037823	3.6824e-10	0.00091325	0.00091325	7.4689e-05
27	0.0030787	0.040601	3.0659e-10	0.0010744	0.0010744	7.3471e-05
28	0.0033835	0.043679	2.5472e-10	0.0013418	0.0013418	7.5334e-05
29	0.0037222	0.047063	1.8128e-10	0.0016647	0.0016647	7.5112e-05
30	0.0040696	0.050785	1.2232e-10	0.0021561	0.0021561	7.6518e-05
31	0.004465	0.054855	7.6271e-11	0.0027784	0.0027784	7.5713e-05
32	0.0048601	0.05932	4.4177e-11	0.0037977	0.0037977	7.7541e-05
33	0.0053129	0.06418	2.5283e-11	0.0051457	0.0051457	7.6549e-05
34	0.0058181	0.069493	1.3122e-11	0.0072239	0.0072239	7.6148e-05
35	0.006353	0.075311	6.2264e-12	0.010617	0.010617	7.6809e-05
36	0.0071789	0.081664	3.1576e-12	0.014462	0.014462	6.9303e-05
37	0.0078128	0.088843	1.1634e-12	0.025376	0.025376	7.7581e-05
38	0.00793	0.092749	1.4126e-09	0.0064936	0.0064936	1.1954e-05
39	0.0095199	0.1	3.3383e-12	0.031623	0.031623	4.4185e-05
Average	0.0030706	0.034569	1.3e-07	0.003085	0.003085	6.17e-05

Table 6.21: Estimates of local error for the linearized Brusselator system of equations with $N = 150$ points per dimension computed using the adaptive time stepping algorithm. The starting timestep size estimate used was $\triangle t$ 0.01 seconds.

Number of Time Steps	$\triangle t$	t	High Frequency Error	Low Frequency Error	Relative Residual	Residual
1	0.01	0.01	0	0.047307	0.047307	0.047302
2	0.01	0.02	1.4129e-13	0.040492	0.040492	0.019955
3	0.01	0.03	3.5218e-14	0.014575	0.014575	0.003533
4	0.01	0.04	1.2402e-13	0.02552	0.02552	0.0030411
5	0.01	0.05	5.0207e-14	0.040526	0.040526	0.0023741
6	0.01	0.06	7.1807e-14	0.012181	0.012181	0.00035093
7	0.01	0.07	2.7325e-14	0.023805	0.023805	0.00033741
8	0.01	0.08	7.8023e-14	0.071964	0.071964	0.00050214
9	0.01	0.09	7.3612e-14	0.099621	0.099621	0.00034249
10	0.01	0.1	4.9479e-14	0.27109	0.27109	0.00045967
Average	0.01	.055	6.5e-14	0.064708	0.064708	0.00782

Table 6.22: Estimates of local error for the linearized Brusselator system of equations with $N = 150$ points per dimension computed without using the adaptive time stepping algorithm. The starting timestep size estimate used was $\triangle t = 0.01$ seconds.

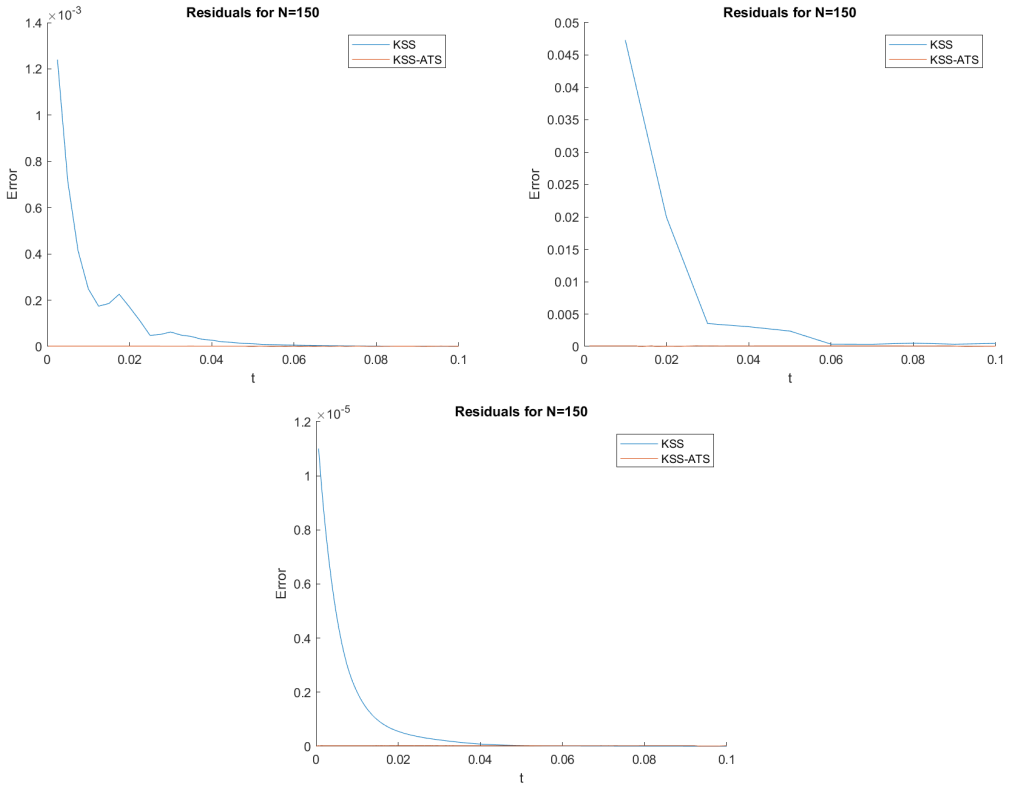

Figure 6.39: A comparison of execution time (t) and the size of the residual at that time. The top left figure contains the residual for the initial starting step size, the next figure is for $1/4$ of initial starting step size, and the last figure is for $1/16$ of the initial starting step size.

6.10 Adaptive Time Stepping Case 2: Performance

In this section, we will compare the performance of KSS and KSS-ATS with Krylov Projection (KP) as described in [1] and Leja interpolation (LEJA). Figure 6.40 shows the relative error for different starting time step sizes. The first time step size was $\triangle t = .01$ and then was decreased by a factor of $\frac{1}{2}$ four times. In terms of accuracy, KSS-ATS was the most accurate method at larger time steps for both of the coarser grid resolutions ($N = 25$ and $N = 50$) and the most accurate at every time step size for the highest resolution. From Figure 6.40 we can see that as the grid size resolution gets higher, KSS-ATS becomes the most appealing method accuracy wise. Figure 6.41 shows the relative error of each method versus the execution time of the method for each time step size. Here we see that at the lowest resolutions, the execution time for Krylov Projection was, in general, smaller than for other methods while Leja interpolation had the largest execution times. At $N = 25$ points per dimension, KSS and KSS-ATS has very similar execution times but at $N = 50$ and $N = 150$ KSS-ATS had slightly larger execution times. This is most likely due to the fact that when decreasing a time step size there is some repetition of computation that adds to KSS-ATS's computational expense.

Table 6.10 contains the average number of arnoldi iterations for each method. Leja interpolation used by far the most Arnoldi iterations while KSS and KSS-ATS used the least on average. This is especially true at high grid size resolution ($N = 150$).

	KSS-ATS	KSS	KP	LEJA
	0.00023845	0.00093887	0.0011416	2.9889
$N = 50$	1.7004e-05	8.4691e-05	0.00032831	0.093217
	1.1607e-06	2.2402e-06	7.1968e-05	0.0042728
	5.6335e-08	4.0861e-08	1.6787e-05	0.0022783
	6.3369e-09	1.1379e-09	6.6769e-08	0.00030902
	0.0003207	0.0010772	0.001791	20.401
$N = 150$	1.7503e-05	0.0001113	0.00036163	4.3314
	9.023e-07	3.7286e-06	9.4574e-05	0.1001
	6.2342e-08	4.5857e-07	2.4945e-05	0.0066592
	1.4338e-08	4.0675e-08	6.5058e-06	0.00049425

Table 6.23: Relative error for linearized Brusselator equation for starting time step size $\triangle t = 0.02, 0.01, 0.005, 0.0025, 0.00125, 0.000625$

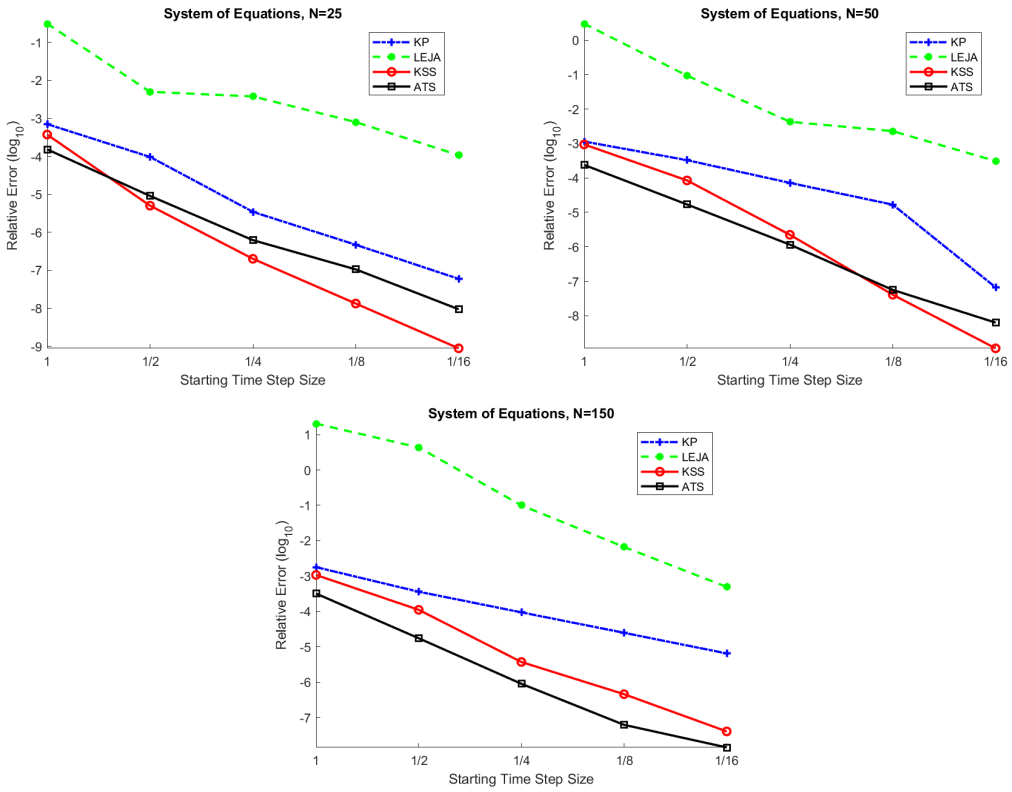

Figure 6.40: Relative error (logarithmically scaled) for varying starting time step sizes for the Linearized Brusselator Equation with N=25, N=50, then N=150 points per dimension.

	KSS-ATS	KSS	KP	LEJA
$N = 50$	1.9063	0.67188	0.14063	0.82813
	2.1094	0.89063	0.23438	1.2188
	2.25	1.0938	0.51563	2.2813
	3.8125	2.2188	0.6875	4.9531
	4.0469	4.4063	1.3906	9.7031
$N = 150$	12.781	3.6094	4.6406	1.7031
	22.125	7.3594	6.0625	2.7969
	27.141	14	8.5625	5.4844
	67.547	28.25	11.719	11.859
	75.125	43.281	17.609	20.547

Table 6.24: Times for linearized Brusselator equation for starting time step sizes $\triangle t = 0.02$, 0.01, 0.005, 0.0025, 0.00125, 0.000625.

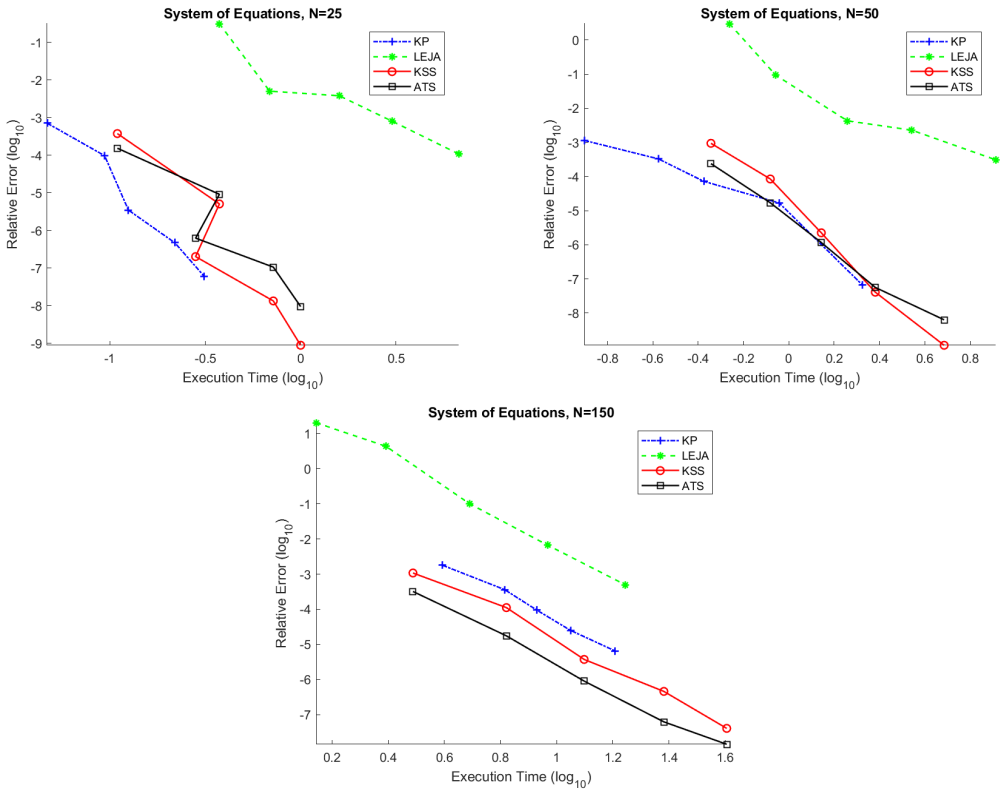

Figure 6.41: Relative error (logarithmically scaled) and the execution time for the Linearized Brusselator Equation with N=25, N=50, then N=150 points per dimension.

	KSS-ATS	KSS	KP	LEJA
$N = 50$	4.2444	5.9	6.4	11
	4.2787	5.7	5.55	11.45
	5.2459	5.05	4.925	10.425
	5.5333	5	4.375	9.35
	5.77	5	4.3625	9.5813
$N = 150$	4.2353	6.2	15.4	35.5
	4.1667	6.55	13.5	23
	4.5288	5.775	10.425	23.725
	4.6654	5	7.8625	21.05
	5.0253	5	6.025	17.281

Table 6.25: Number of iterations for linearized Brusselator equation for starting time step sizes $\triangle t = 0.02, 0.01, 0.005, 0.0025, 0.00125, 0.000625$.

6.11 Adaptive Time Stepping Case 2: Inclusion of a Minimum Step Size

In the previous experiments, no minimum step size was included for adaptive time stepping. The purpose of not including a minimum step size was to analyze the step size controller's effectiveness on the problem and to ensure that a specific error tolerance was reached. Here we will consider minimum step sizes h_{min} =0.02, 0.01, and 0.05.

Figures 6.42, 6.43, and 6.44 display a comparison of KSS-ATS with and without a minimum time step (ATS h_{min} and ATS respectively). Although the error is slightly higher when a minimum step size is used, the execution time is less. This is due to the fact that once a step size is deemed too small by the algorithm, a "small enough" step size is used. This eliminates the possibility of too much repetitive computation in the algorithm.

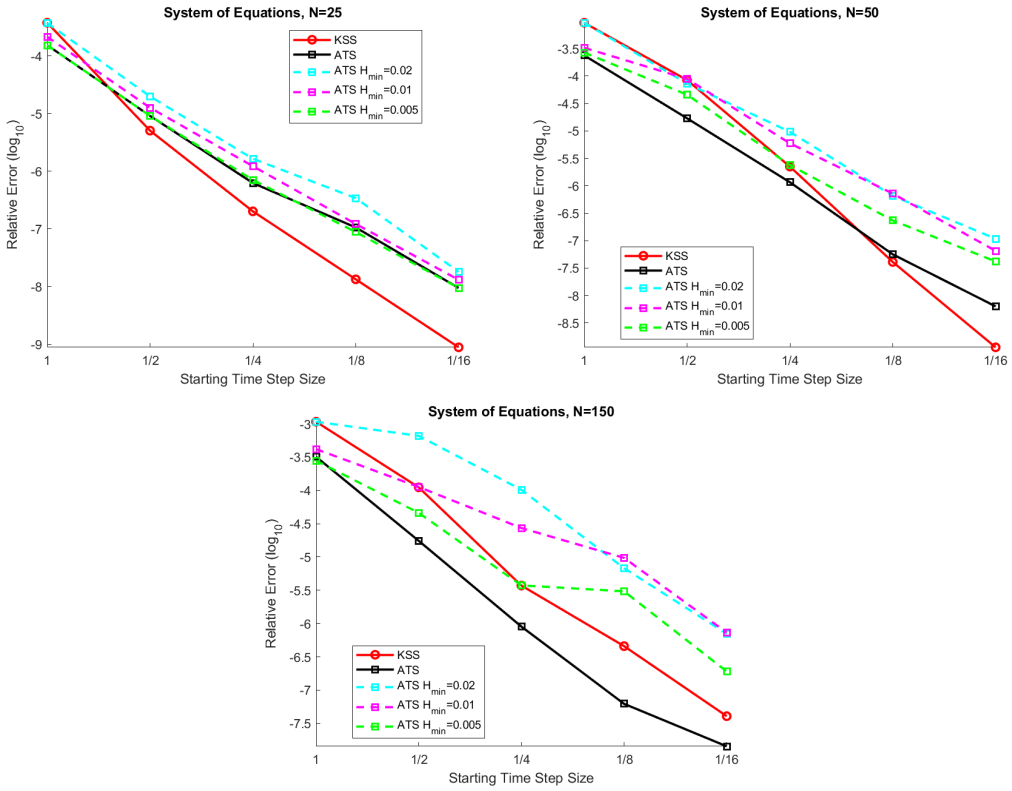

Figure 6.42: Comparison of the relative error of KSS (logarithmically scaled with and without adaptive time stepping with and without the minimum step size with respect to each starting time step size. $h_{min} = 0.01$ has a minimum step size of .01, $h_{min} = 0.02$ has a minimum step size of .02 and $h_{min} = 0.005$ has a minimum step size of .005.

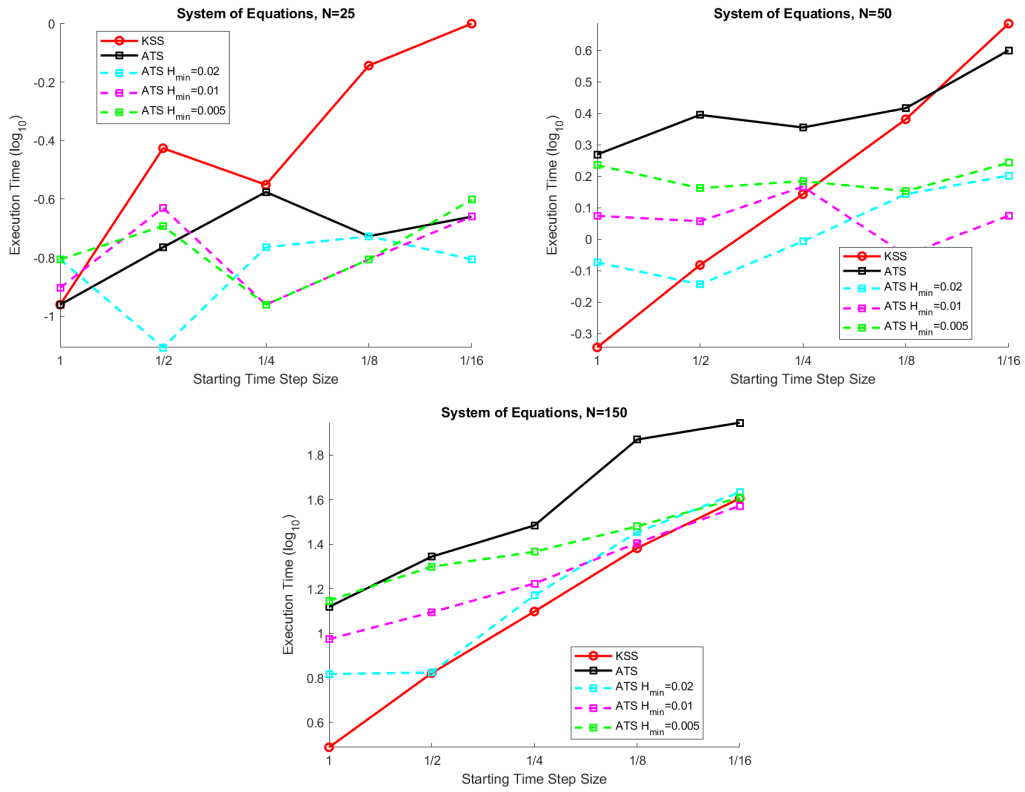

Figure 6.43: Comparison of the execution times of KSS with and without adaptive time stepping with and without the minimum step size with respect to each starting time step size. $h_{min} = 0.01$ has a minimum step size of .01, $h_{min} = 0.02$ has a minimum step size of .02 and $h_{min} = 0.005$ has a minimum step size of .005.

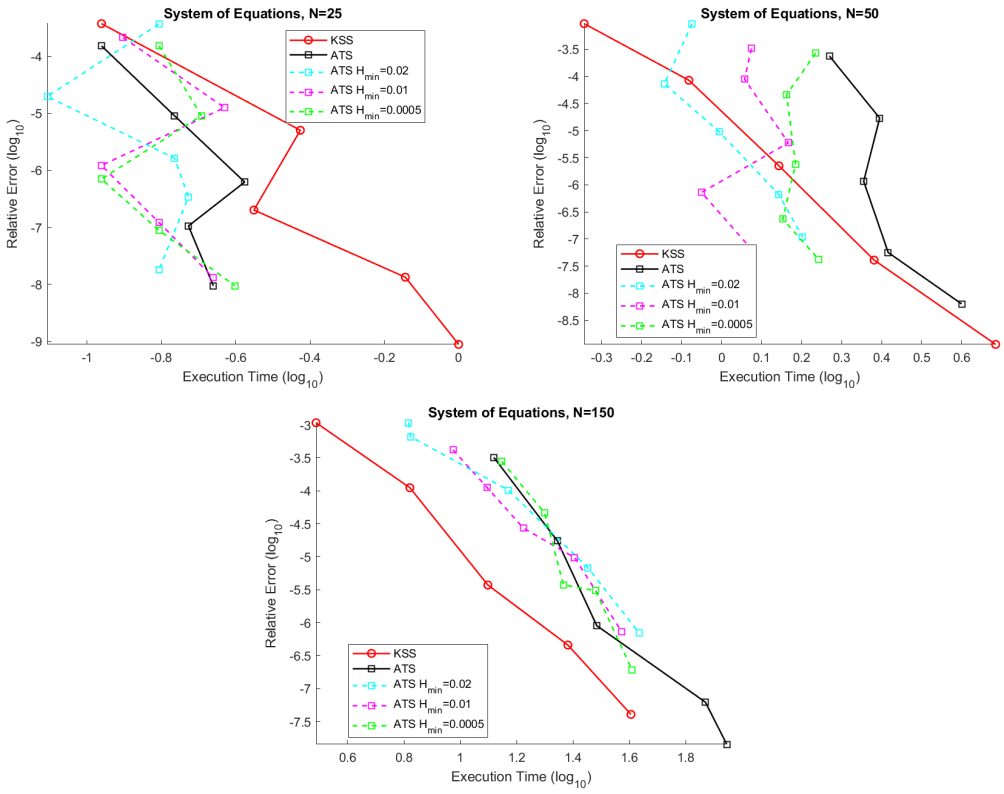

Figure 6.44: Comparison of the execution times versus the relative error of KSS with and without adaptive time stepping with and without the minimum step size with respect to each starting time step size. $h_{min} = 0.01$ has a minimum step size of .01, $h_{min} = 0.02$ has a minimum step size of .02 and $h_{min} = 0.005$ has a minimum step size of .005.

Chapter 7

Conclusion

When solving time dependent variable coefficient partial differential equations Krylov Subspace Spectral Methods are extremely effective, especially on high frequency components of the solution. This dissertation focused on eliminating error caused by the low frequency components of the solution. These two approaches concentrate on the use of the residual as a local error approximation to improve the accuracy of the method and each has its own advantages and disadvantages.

It has been shown that the implementation of coarse grid residual correction along with KSS is both effective and efficient. The effectiveness of KSS at eliminating high frequency error combined with the effectiveness of coarse grid residual correction to eliminate low frequency error makes KSS-CGRC far superior to other similar methods. In addition to this, KSS-CGRC retains KSS methods scalability and so as grid size increases, the benefits of using KSS-CGRC increase.

It has also been shown that the addition of adaptive time stepping is effective at ensuring the accuracy of KSS when the starting step size of the method is chosen arbitrarily. When a step size is chosen that is too large, adaptive time stepping will shrink that step size to ensure accuracy. Although KSS with adaptive time (KSS-ATS) stepping is not as scalable as KSS-CGRC, the inclusion of a minimum step size does decrease computational expense.

Future work on this topic may include expanding these methods to solve a wider variety of problems, developing a threshold to judge when to use adaptive time stepping or coarse grid residual correction, and using these methods to solve problems on non-rectangular domains.

BIBLIOGRAPHY

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

[2] K. Atkinson. *An Introduction to Numerical Analysis*. Wiley, 1989.

[3] H. Dozier and J. V. Lambers. Krylov subspace spectral methods with coarse-grid residual correction for solving time-dependent, variable-coefficient pdes. In *Proceedings, International Conference on Spectral and High Order Methods, 2016*.

[4] Haley Dozier. Krylov subspace spectral method with multigrid for a time-dependent, variable-coefficient partial differential equation, 2016.

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

[6] Stanley J. Farlow. *Partial DIfferential Equations for Scientists and Engineers,*.

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

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

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

[10] Marlis Hochbruck and Christian Lubich. A gautschi-type method for oscillatory second-order differential equations. *Numerische Mathematik*, 83:403–426, 1999.

[11] J. V. Lambers. *Krylov Subspace Methods for Variable-Coefficient Initial-Boundary Value Problems*. PhD thesis, Stanford University, 2003.

[12] J. V. Lambers. Derivation of high-order spectral methods for time-dependent pde using modified moments. *Electronic Transactions on Numerical Analysis*, 28:114–135, 2008.

[13] J. V. Lambers. Enhancement of krylov subspace spectral methods by block lanczos iteration. *Electronic Transactions on Numerical Analysis*, 2008.

[14] J. V. Lambers. An explicit, stable, high-order spectral method for the wave equation based on block gaussian quadrature. *IAENG Journal of Applied Mathematics*, 38, 2008.

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

[16] James V. Lambers and Amber C. Sumner. *Explorations in Numerical Analysis*. World Scientific, 2018.

[17] K. Solna P. Guidotti, J. V. Lambers. Analysis of 1-d wave propagation in inhomogeneous media. *Numerical Functional Analysis and Optimization*, 27, 2006.

[18] Endre Suli and David Meyers. *An Introduction to Numerical Analysis*. Cambridge University Press, 2003.

[19] M. Tokaman. A new class of exponential propagation iterative methods of runge-kutta type (epirk). *Journal of Computational Physics*, 230, 2011.

[20] user147263. Decay of a convolution. Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/813127 (version: 2014-05-28).

[21] Van Emden Henson William Briggs and Steve F. McCormick. *A Multigrid Tutorial,*.