PSEUDOSPECTRAL METHODS FOR A CLASS OF EVOLUTIONARY SYSTEMS

by

Sam Geldhof

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Mathematical and Computer Sciences).

Golden, Colorado

Date _____

Signed: _____
Sam Geldhof

Signed: _____
Dr. Mahadevan Ganesh
Thesis Advisor

Golden, Colorado

Date _____

Signed: _____
Dr. Willy Hereman
Professor and Head
Department of Applied Mathematics and Statistics

ABSTRACT

Mathematical and computer modeling of various physical processes typically involve evolutionary systems which are a class of time dependent partial differential equations. Exact solutions to many of these evolutionary systems are not known in general and a variety of numerical methods have been proposed to simulate the solutions of these problems. In this thesis, we approximate the solution of evolutionary systems using the method of lines in which we separately discretize our problem in space and in time. The spatial discretization is done using a class of high order Fourier spectral Galerkin methods with quadrature, also known as Fourier pseudospectral methods. Many evolutionary systems that we will be concerned with contain quadratic nonlinear terms, causing aliasing affects. We consider two techniques to resolve the affects of aliasing. The spatial discretization reduces the problem to a system of ordinary differential equations in time which we solve using standard numerical techniques. Many of these systems are stiff and therefore we also consider the stability of our numerical solvers and consider some implicit methods. In order to obtain accurate solutions for many problems, we must calculate a large number of Fourier coefficients, leading to impractical computation times. Therefore, we seek methods that allow us to calculate more Fourier modes in a computationally inexpensive manner. The algorithms developed and implemented in this thesis are based on the high order pseudospectral Galerkin method with quadrature in space and various implicit and explicit discretization techniques in time.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to begin by thanking my advisor, Mahadean Ganesh, for his countless hours in helping and advising. He has truly inspired me to work to constantly improve myself by not only focusing on my strengths, but more importantly, my weaknesses.

I would also like the thank the Department of Applied Math and Statistics at the Colorado School of Mines and the personnel therein. They have provided me both a fantastic education as well as the opportunity to support myself through my graduate education. In particular, I would like to thank my thesis committee Drs. Luis Tenorio and Cory Arhens for their high expectations and ensuring that I produce high quality work.

Finally, I would like to thank my friends and family for their support during the writing of this thesis. Their encouragement has given me the strength to continue working on this thesis when progress seemed impossible. Thank you so much.

CHAPTER 1

INTRODUCTION

Partial differential equations (PDEs) may be used to model a variety of natural phenomenon such as acoustics, elasticity, fluid dynamics, geophysics, heat transfer, optics, and quantum mechanics [8]. In this thesis, we will focus on evolutionary PDEs that occur typically in fluid dynamics modeling, among several applications. Many of these equations are related to the incompressible Navier-Stokes equations (NSE)

$$\underline{u}_t + \underline{u} \cdot \nabla \underline{u} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \underline{u} + \underline{f} \quad \text{in } \Omega \times (0, T] \tag{1.1}$$

$$\nabla \cdot \underline{u} = 0. \tag{1.2}$$

Here, we let $\underline{x}$ denote the vector of spatial variables with $\underline{x} \in \Omega \subset \mathbb{R}^n$, for $n = 1, 2$, and t is the temporal variable with $t \in [0, T] \subset \mathbb{R}$, where T is the final time. With these notations, we have $\underline{u} : \Omega \times [0, T] \to \mathbb{R}^n$, the fluid velocity, and $p : \Omega \times [0, T] \to \mathbb{R}$, the fluid pressure, are unknowns. ρ, the fluid density, and ν, the fluid's kinematic viscosity, are given positive constants and $\underline{f} : \Omega \times [0, T] \to \mathbb{R}^n$ is a given external force. We have used the notation $\underline{u}_t$ to denote the derivative of $\underline{u}$ with respect to t, ∇ is the gradient operator, and ∇^2 is the Laplacian. The equations (1.1)-(1.2) are often supplemented with additional conditions to give a unique solution for $\underline{u}$ and p, which we assume to be an initial condition defining $\underline{u}$ and p at $t = 0$ for all $\underline{x} \in \Omega$ and a boundary condition defining $\underline{u}$ and p on the boundary, $\partial\Omega$, of Ω, for all $t \in [0, T]$.

We consider PDEs of the form

$$\underline{u}_t + L[\underline{u}] = \underline{f}, \tag{1.3}$$

where $L[\cdot]$ is a spatial linear or nonlinear, first or second order differential operator and $\underline{f}$ is a given function. We refer to equations of the form (1.3) as *evolution systems*. The method of

lines (MOL) may be used to approximate the solutions of evolutionary systems [17]. In the MOL, the problem (1.3) is decomposed into two parts – first, the problem is discretized in space, then the problem is discretized in time. Many different methods have been developed to discretize PDEs in space. Some spatial discretization methods include: finite difference methods, finite element methods, and spectral methods. Each of these methods have their own advantages and disadvantages.

Finite difference methods approximate functions in space by discretizing the spatial domain as a grid of distinct points. Derivatives of these functions are then approximated using finite difference methods such as the forward difference

$$u'(x) \approx \frac{u(x+h) - u(x)}{h}, \tag{1.4}$$

where h is the grid spacing. While finite difference methods are simple to implement, they have some disadvantages. In order to use finite difference methods, the solution space must be discretized on a simple grid. Thus, finite difference methods can only be used on problems with simple geometries. Furthermore, the approximations of spatial derivatives in finite difference methods, such as the forward difference (1.4) tend to have a low order of convergence. While finite difference methods exist with more general domains and higher orders of convergence, these methods have increased computational costs. Therefore, in order to get accurate results with finite difference methods, we must either use low order methods with a very fine grid, or use costly high order methods. Both of these techniques lead to large amounts of computation time, making them impractical to give very accurate results.

In finite element methods, the spatial domain is partitioned into smaller subdomains called elements. Functions may then be approximated as linear combinations of basis functions on each of the element. A common finite element method is to choose the basis functions on each element to be low order polynomials so that functions are approximated as piecewise polynomials. Derivatives of these functions can then be computed analytically. While finite element methods get around the disadvantage of finite difference methods by allowing more

general domains, they also suffer form low orders of convergence. This makes finite elements also impractical for computing very accurate solutions.

Spectral methods approximate functions spatially by approximating functions as linear combinations of global basis functions. As with finite elements, this allows derivatives of the approximations to be calculated exactly. Since these basis functions are global, they must be defined on simple domains, losing the generality of spatial domains that finite element methods enjoy. However, unlike finite difference and finite element methods, which have orders of convergence based on the methods used, the order of convergence of spectral methods is based on the regularity of the function being approximated. That is, under certain conditions, the order of convergence of spectral methods may be very high, allowing a high degree of accuracy. In this thesis, we aim for high accuracy solutions for problems on simple domains and hence we will focus on spectral Galerkin methods with quadrature, known as pseudospectral methods. Furthermore, we will choose our basis functions to be trigonometric polynomials.

The rest of this thesis is organized as follows. In Chapter 2, we will discuss how to discretize functions in space with pseudospectral methods. We will also discuss the error made by this discretization, how this may be done in an efficient manner, and how to treat nonlinear problems. In Chapter 3, we will discuss how to discretize evolution systems in space. In this chapter, we will also discuss the errors made from the temporal discretization, what constraints must be satisfied to ensure good results, and how to estimate the order of convergence. In Chapter 4, we will analyze and apply pseudospectral methods to several evolutionary systems with a single spatial variable. In Chapter 5, we will discuss pseudospectral methods to solve evolutionary systems in two spatial dimensions. We will then demonstrate pseudospectral methods by simulating several model systems. We then end with Chapter 6, where we will discuss potential future work based on the topics discussed in the rest of the thesis.

CHAPTER 2

SPATIAL DISCRETIZATION AND FOURIER ANALYSIS

In this chapter, we examine how to discretize evolution equations in space using the Fourier pseudospectral method. To illustrate this spatial discretization, in this chapter, we consider a purely spatial problem in one dimension:

$$\begin{cases} L[u] = f, \quad \text{on } \Omega \\ \text{Periodic Boundary conditions} \end{cases} \tag{2.1}$$

where $L[\cdot]$ is a spatial linear or nonlinear, first or second order differential operator, $\Omega = (-\pi, \pi)$, and $u : \overline{\Omega} \to \mathbb{C}$. Here, $\overline{\Omega}$ denotes the closure of Ω.

To discretize (2.1), we assume that u is an element of a Hilbert space $V \subset L^2(\Omega)$, with an inner product, $\langle \cdot, \cdot \rangle$, which is the standard L^2 inner product on V:

$$\langle u, v \rangle = \int_\Omega u \overline{v} \, \mathrm{d}\mu \quad \text{for } u, v \in V, \tag{2.2}$$

where $\bar{\cdot}$ denotes the complex conjugate and $\mathrm{d}\mu$ denotes the standard Lebesgue measure on Ω.

A function, u, is discretized in space in spectral methods by an approximation, u_N, that is an element of an N-dimensional subspace, $V_N \subset V$, of the form

$$u_N = \sum_{k=1}^{N} \hat{u}_k \phi_k, \tag{2.3}$$

where $\{\phi_k\}_{k=1}^N$ is a set of global basis functions for V_N and the coefficients $\hat{u}_k$ are complex constants. The evolutionary systems that we will consider in this thesis will have periodic boundary conditions in space. Thus, we choose the global basis functions, ϕ_k, as being Fourier basis functions as they are periodic. That is, we will define ϕ_k as

$$\phi_k(x) = e^{ikx} \quad \text{for } k = -N, -N+1, \ldots, N-1. \tag{2.4}$$

In this chapter, we will show that this choice of basis functions leads to a high order of convergence for approximating smooth periodic functions.

We define the *residual* of the approximation u_{2N} as

$$R_{2N} = L[u_{2N}] - f. \tag{2.5}$$

Spectral Galerkin methods are: find $u_{2N} \in V_{2N}$ such that the residual, R_{2N} from (2.5), satisfies

$$\langle R_{2N}, \phi_k \rangle = 0 \quad \text{for } k = -N, -N+1, \ldots, N-1. \tag{2.6}$$

Thus, we find u_{2N} such that the residual is orthogonal to V_{2N}. We approximate the integral in (2.6) with quadrature to get a pseudospectral method. We let

$$\int_\Omega u \, \mathrm{d}\mu \approx \sum_{n=-N}^{N-1} w_n u(x_n) \tag{2.7}$$

denote a quadrature rule with grid points $\{x_n\}_{n=-N}^{N-1}$ and corresponding weights $\{w_n\}_{n=-N}^{N-1}$. Using the quadrature (2.7), we define a discrete inner product, $\langle \cdot, \cdot \rangle_{2N}$, on V_{2N} as

$$\langle u, v \rangle_{2N} = \sum_{n=-N}^{N-1} w_n u(x_n) \overline{v}(x_n) \quad \text{for } u, v \in V_{2N}. \tag{2.8}$$

Using this discrete inner product, we define pseudospectral methods as: find $u_{2N} \in V_{2N}$ such that the residual from (2.5) satisfies

$$\langle R_{2N}, \phi_k \rangle_{2N} = 0 \quad \text{for } k = -N, -N+1, \ldots, N-1. \tag{2.9}$$

For each set of basis functions, ϕ_k, there is a natural choice for the quadrature rule. We choose the quadrature rule so that the inner product of any two basis functions of V_{2N} is evaluated exactly. To this end, we choose the trapezoidal rule as our quadrature rule. Later in this chapter, we will show that the trapezoidal rule satisfies this condition. The $2N$-point trapezoidal rule on Ω is defined by letting $w_n = \frac{\pi}{N}$ and $x_n = \frac{\pi}{N} n$ for $n = -N, -N+1, \ldots N-1$. Explicitly, we have

$$\int_\Omega u \, \mathrm{d}\mu \approx \frac{\pi}{N} \sum_{n=-N}^{N-1} u\left(\frac{\pi}{N} n\right). \tag{2.10}$$

We continue this topic by discussing how to express functions as Fourier series and how to approximate these functions by truncating their Fourier series.

2.1 Fourier Series

The Fourier series of $u \in L^1(\Omega)$, is given by

$$u = \sum_{k=-\infty}^{\infty} \hat{u}_k \phi_k, \tag{2.11}$$

with ϕ_k as in (2.4), and where the Fourier coefficients, $\hat{u}_k$, are the complex constants

$$\hat{u}_k = \frac{1}{2\pi} \int_\Omega u \overline{\phi_k} \, \mathrm{d}\mu. \tag{2.12}$$

Writing a function in terms of its Fourier series allows derivatives to be computed easily. For example, if u_x exists, we have

$$u_x = \sum_{k=-\infty}^{\infty} ik\hat{u}_k \phi_k.$$

In Fourier pseudospectral methods, we compute only a finite number of terms in (2.12).

Definition 1. *$V_{2N} \subset L^2(\Omega)$ is a $2N$-dimensional Hilbert space with*

$$V_{2N} = span\{\phi_k\}_{k=-N}^{N-1} \subset L^2(\Omega),$$

where ϕ_k is as defined in (2.4).

Since $\langle \phi_j, \phi_k \rangle = 2\pi\delta_{j,k}$, $\{\phi_k\}_{k=-N}^{N-1}$ is an orthogonal basis for V_{2N}, where $\delta_{j,k}$ denotes the Kronecker delta function.

We may approximate a function by projecting it into V_{2N}. We denote this approximation of $u \in V$ as $u_{2N} \in V_{2N}$, where u_{2N} is given by truncating the Fourier series of u:

$$u_{2N}(x) = \sum_{k=-N}^{N-1} \hat{u}_k \phi_k, \tag{2.13}$$

where the Fourier coefficients, $\hat{u}_k$, are computed as in (2.12).

2.2 Spatial Error Analysis

As u_{2N} given in (2.13) only approximates the function u, it is important to understand the error made by this approximation. To analyze the error of our approximations, we use

the *big O notation*. We say that $f(x) = O(g(x))$ as $x \to x_0$, if there is a $C > 0$, such that $|f(x)| \leq C|g(x)|$ for all x in some neighborhood of x_0.

2.2.1 Convergence of Fourier Coefficients

In this section, we will discuss how a function's Fourier coefficients decay. For example, we have the following result from [3, Line 2.1.19] for continuously differentiable functions

Lemma 2.1. *If $u \in C^1(\Omega)$, then*

$$\hat{u}_k = O(k^{-1}) \text{ as } |k| \to \infty.$$

Proof. We prove this result using integration by parts.

$$\begin{aligned}
\hat{u}_k &= \frac{1}{2\pi} \int_\Omega u\overline{\phi}_k \, \mathrm{d}\mu = \frac{1}{2\pi} \int_{-\pi}^{\pi} u(x) e^{-ikx} \, \mathrm{d}x \\
&= \frac{1}{2\pi} \left[u(x) \frac{-1}{ik} e^{-ikx} \Big|_{x=-\pi}^{\pi} - \frac{1}{2\pi} \int_{-\pi}^{\pi} u'(x) \frac{-1}{ik} e^{-ikx} \, \mathrm{d}x \right. \\
&= \frac{i}{2\pi k} \left[u(\pi) - u(-\pi) - \int_{-\pi}^{\pi} u'(x) e^{-ikx} \, \mathrm{d}x \right] = O(k^{-1})
\end{aligned}$$

□

When u has more regularity than being just differentiable, we may strengthen this result. For example, if $u(-\pi) = u(\pi)$, then the boundary terms will cancel. Furthermore, if $u' \in C^1(\Omega)$, then we may repeat the integration by parts to get a faster rate of decay. Combining this process with induction, we strengthen Lemma 2.1 to the following result from [3, Line 2.1.20]

Theorem 2.2. *If $u \in C^s(\Omega)$ for some $s \in \mathbb{N}$ and if $u^{(r)}(-\pi) = u^{(r)}(\pi)$ for all $r \leq s - 2$, then*

$$\hat{u}_k = O(k^{-s}) \quad \text{as } |k| \to \infty.$$

For the case of an infinitely differentiable periodic function, we have the following result

Corollary 2.3. *If $u \in C^\infty(\Omega)$ and $u^{(r)}(-\pi) = u^{(r)}(\pi)$ for all $r \in \mathbb{N}$, then for all $s \in \mathbb{N}$,*

$$\hat{u}_k = O(|k|^{-s}) \quad \text{as } |k| \to \infty.$$

Using Lemma 2.1, Theorem 2.2, and Corollary 2.3, we see that the Fourier coefficients decay based on the regularity of the function u. We demonstrate this in the next example.

2.2.2 Example: The Convergence of Fourier Coefficients

We illustrate the results from Section 2.2.1 by examining the decay of the Fourier coefficients of several functions. We consider

$$u(x) = e^{\sin(x)}. \tag{2.14}$$

Here, $u(x)$ is infinitely differentiable, and all of its derivatives are 2π-periodic. In addition, we consider

$$v(x) = 2\pi^2 x^2 - x^4 \tag{2.15}$$

along with its first three derivatives $v'(x), v''(x)$, and $v'''(x)$. Here, the function $v(x)$ is infinitely differentiable and $v(-\pi) = v(\pi), v'(-\pi) = v'(\pi)$, and $v''(-\pi) = v''(\pi)$, but $v'''(-\pi) \neq v'''(\pi)$.

In Figure 2.1, we show the magnitude of the Fourier coefficients corresponding to $1 \leq k \leq 1000$, normalized by the magnitude of the Fourier coefficient corresponding to $k = 1$. Here, we have computed an approximation to the Fourier coefficients using the discrete Fourier transform as described later in this chapter. The Fourier coefficients decay as described in Theorem 2.2. That is, for the infinitely differentiable periodic function $u(x)$, we see that its Fourier coefficients decays like k^{-k}. For v, we see that $\hat{v}_k$ decays like k^{-4}, and similarly for its derivatives.

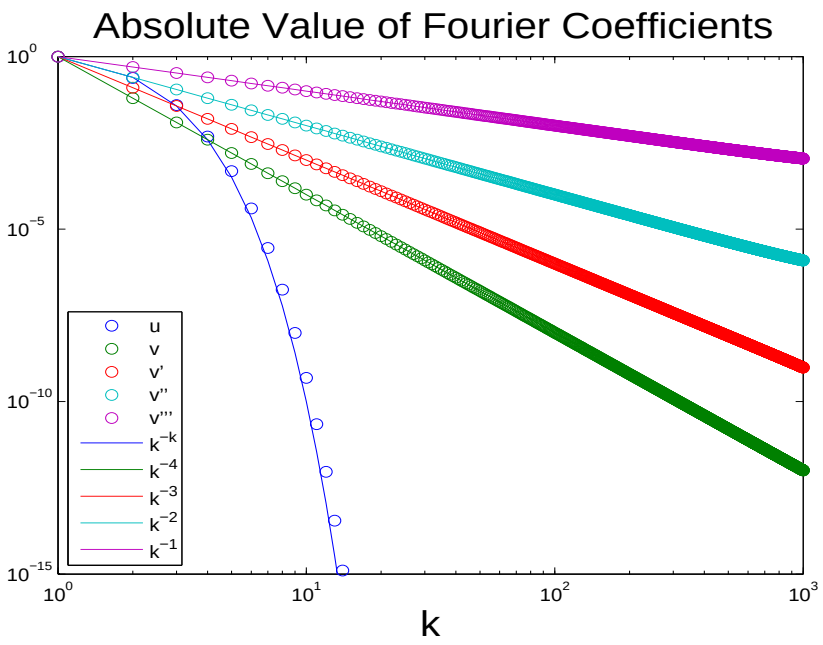

Figure 2.1: Plot of the absolute value of the Fourier coefficients of u from (2.14), v from (2.15), and first three derivatives of v: v', v'', and v''' for $k > 0$.

2.2.3 Convergence of Truncated Fourier Series Approximations

While the results from Section 2.2.1 tell us how the Fourier coefficients decay, they do not tell us how u_{2N} in (2.13) converges to u in (2.11) as $N \to \infty$ (or even if it does). Before we examine how u_{2N} converges to u, we first introduce the seminorm

$$|u|_m = \left(\sum_{k=-\infty}^{\infty} |k|^{2m} |\hat{u}_k|^2 \right)^{\frac{1}{2}}.$$

As we have results on how the Fourier coefficients decay from Section 2.2.1, we may examine how u_{2N} converges to u if we have a result that relates the size of a function, in some sense, to the size of its Fourier coefficients. This result is given by Parseval's theorem from [3, Line 2.1.14]

Theorem 2.4. *Let $u \in L^2(\Omega)$, then,*

$$\frac{1}{2\pi} \|u\| = |u|_0^2.$$

Proof.

$$\frac{1}{2\pi}\|u\| = \frac{1}{2\pi}\langle u, u\rangle = \frac{1}{2\pi}\left\langle u, \sum_{k=-\infty}^{\infty} \hat{u}_k \phi_k \right\rangle = \frac{1}{2\pi}\sum_{k=-\infty}^{\infty} \hat{u}_k \langle u, \phi_k\rangle = \sum_{k=-\infty}^{\infty} |\hat{u}_k|^2 = |u|_0^2.$$

□

Using Theorem 2.4 and the results from Section 2.2.1, we can determine the error made in approximation (2.13) using the following theorem from [15, Theorem 2.1]

Theorem 2.5. *If u has $s-1$ continuous periodic derivatives, then u_{2N} from* (2.13) *satisfies*

$$\|u - u_{2N}\| = O(N^{-s}|u|_s) \quad \text{as } N \to \infty.$$

Proof. By Theorem 2.4, we have

$$\frac{1}{2\pi}\|u - u_{2N}\|^2 = |u - u_{2N}|_0^2 = \sum_{k=-\infty}^{\infty} |\hat{u}_k - (\widehat{u_{2N}})_k|^2 = \sum_{k=-\infty}^{-N-1} |\hat{u}_k|^2 + \sum_{k=N}^{\infty} |\hat{u}_k|^2,$$

as $\hat{u}_k = (\widehat{u_{2N}})_k$ for $k = -N, -N+1, \ldots, N-1$ and $(\widehat{u_{2N}})_k = 0$ otherwise. By Theorem 2.2, both of these sums decay at the same rate, so that for some constant C, we have as $N \to \infty$

$$\|u - u_N\| \leq C \sum_{k=N}^{\infty} |\hat{u}_k|^2.$$

Now, by noting that for $k \geq N$ and for all $s > 0$, we have $N^{-2s}k^{2s} \geq 1$. Using this, we have

$$\|u - u_{2N}\|^2 \leq CN^{-2s} \sum_{k=N}^{\infty} k^{2s}|\hat{u}_k|^2 \leq CN^{-2s}|u|_s^2 = O(N^{-2s}|u|_s^2) \text{ as } N \to \infty$$

and the result follows. □

2.2.4 Example: The Convergence of Truncated Fourier Series Approximations

We consider the functions u and v from Section 2.2.2 defined in (2.14) and (2.15). To examine the errors made in the truncated Fourier series approximations of these functions, we define the discrete L^2 relative error, $RE_M(u, u_{2N})$, as

$$RE_M(u, u_{2N}) = \frac{\|u - u_{2N}\|_M}{\|u\|_M}, \tag{2.16}$$

where $\|\cdot\|_M$ denotes the approximation of $\|\cdot\|$ using the M-point trapezoidal rule (2.10).

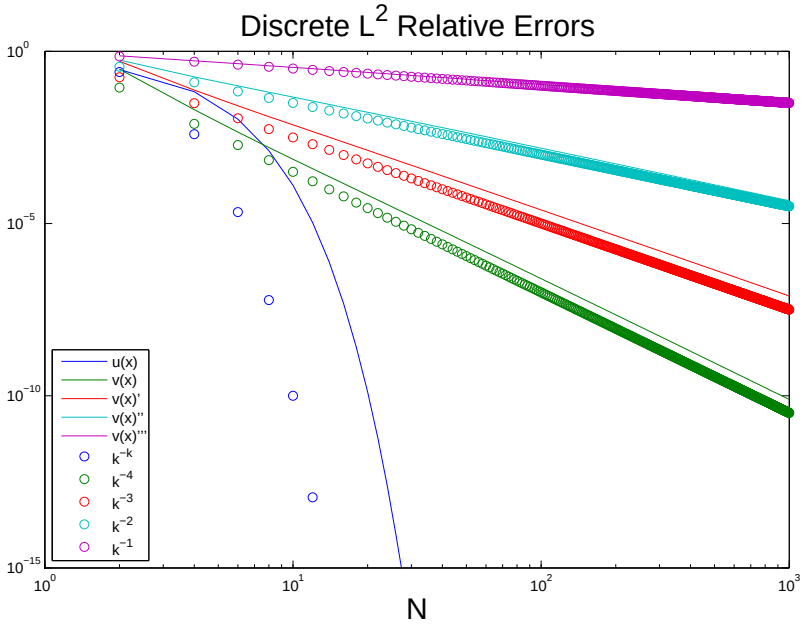

Figure 2.2: Plot of the discrete L^2 relative errors of the truncated Fourier series of the functions u_N from (2.14), v_N from (2.15), v_N', v_N'' and v_N''' for $N < 1000$.

For this example, we show in Figure 2.2 the discrete L^2 relative errors with $M = 10^6$. We see in Figure 2.2 that the L^2 relative errors decay as Theorem 2.5 states.

2.3 The Discrete Fourier Transform

We would like to be able to compute the Fourier coefficients from (2.12). However, in general, these integrals cannot be evaluated exactly. Thus, we must approximate them numerically. To do this, we use the trapezoidal rule (2.10). Substituting (2.12) into the $2N$-point trapezoidal rule, we have

$$\hat{u}_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} u(x)e^{-ikx}\,\mathrm{d}x \approx \frac{1}{2N}\sum_{n=-N}^{N-1} u\left(\frac{\pi}{N}n\right)e^{-ik\frac{\pi}{N}n}.$$

This sum is the discrete Fourier transform (DFT),

$$\hat{u}_k \approx \frac{1}{2N}\sum_{n=-N}^{N-1} u(x_n)e^{-ikx_n} \quad \text{for } k = -N, -N+1, \ldots, N-1, \tag{2.17}$$

where $x_n = \frac{\pi}{N}n$. u_N may then be computed from (2.13).

It is often convenient to think of the discrete Fourier transform as a linear transformation, $\mathcal{F}_{2N} : \mathbb{C}^{2N} \to \mathbb{C}^{2N}$, and use the notation

$$\underline{\hat{u}} = \mathcal{F}_{2N}(\underline{u}), \tag{2.18}$$

where $\underline{\cdot}$ denotes a vector and we use the nonstandard indexing of $-N \leq k, n \leq N-1$, $[\underline{\hat{u}}]_k = \hat{u}_k$ and $[\underline{u}]_n = u(x_n)$.

The trapezoidal rule does not compute integrals exactly. However, we now show that we may compute the Fourier coefficients exactly using (2.17), provided $u \in V_{2N}$. To show this result, we begin with a lemma.

Lemma 2.6. *If $-2N+1 \leq k \leq 2N-1$, then $\langle \phi_k, 1 \rangle = \langle \phi_k, 1 \rangle_{2N}$.*

Proof. We show through direct calculation that both give the same number. We begin by calculating $\langle \phi_k, 1 \rangle$. We recognize $1 = \phi_0$ and $\phi_k \in V_{4N}$, so that we have

$$\langle \phi_k, 1 \rangle = \langle \phi_k, \phi_0 \rangle = 2\pi\delta_{k,0}.$$

We now calculate $\langle \phi_k, 1 \rangle_{2N}$ for the cases where $k = 0$ and $k \neq 0$. If $k = 0$, we have

$$\langle \phi_k, 1 \rangle_{2N} = \langle 1, 1 \rangle_{2N} = \frac{\pi}{N} \sum_{n=-N}^{N-1} 1 = \frac{\pi}{N} 2N = 2\pi.$$

If $k \neq 0$, we have

$$\langle \phi_k, 1 \rangle_{2N} = \frac{\pi}{N} \sum_{n=-N}^{N-1} e^{ik\frac{\pi}{N}n} = \frac{\pi}{N} \sum_{n=0}^{2N-1} e^{ik\frac{\pi}{N}(n-N)} = \frac{\pi}{N}(-1)^k \sum_{n=0}^{2N-1} \left(e^{ik\frac{\pi}{N}}\right)^n.$$

Recognizing this as a geometric progression, we have

$$\langle \phi_k, 1 \rangle_{2N} = \frac{\pi}{N}(-1)^k \frac{1 - e^{i2\pi k}}{1 - e^{i\frac{\pi}{N}k}} = 0.$$

Therefore, the result follows. □

Theorem 2.7. *If $u \in V_{2N}$, then*

$$\hat{u}_k = \frac{1}{2N} \sum_{n=-N}^{N-1} u(x_n) e^{ikx_n}.$$

Proof. Let $u = \sum_{j=-N}^{N-1} \hat{u}_j \phi_j$. Then we have

$$\hat{u}_k = \frac{1}{2\pi}\langle u, \phi_k \rangle = \frac{1}{2\pi}\left\langle \sum_{j=-N}^{N-1} \hat{u}_j \phi_j, \phi_k \right\rangle = \frac{1}{2\pi}\sum_{j=-N}^{N-1} \hat{u}_j \langle \phi_j, \phi_k \rangle.$$

Thus, it will be sufficient to show $\langle \phi_j, \phi_k \rangle = 2\pi\delta_{j,k}$ for $-N \leq j, k \leq N-1$ using the $2N$-point trapezoidal rule. We note

$$\langle \phi_j, \phi_k \rangle = \int_\Omega \phi_j \overline{\phi}_k \, \mathrm{d}\mu = \int_\Omega \phi_{j-k} \, \mathrm{d}\mu = \langle \phi_{j-k}, 1 \rangle.$$

By Lemma 2.6, it will therefore suffice to show that $-2N+1 \leq j-k \leq 2N-1$. As $-N \leq j \leq N-1$ and $-N+1 \leq -k \leq N$, by adding these inequalities, we have $-2N+1 \leq j-k \leq 2N-1$ and the result follows. □

Often in computations we do not need the function $u_N(x)$, but only the values $u_N(x_n)$, for $n = -N, -N+1, \ldots, N-1$. To find these values, we substitute x_n for x in (2.13) and get the inverse discrete Fourier transform (IDFT),

$$u(x_n) \approx u_N(x_n) = \sum_{k=-N}^{N-1} \hat{u}_k e^{ikx_n}. \tag{2.19}$$

We define the IDFT as the transformation $\mathcal{F}_{2N}^{-1} : \mathbb{C}^{2N} \to \mathbb{C}^{2N}$ as

$$\underline{u} = \mathcal{F}_{2N}^{-1}(\underline{\hat{u}}). \tag{2.20}$$

We now continue by discussing fast methods for computing the DFT and IDFT.

2.4 The Fast Fourier Transform

To compute the DFT of a vector of length $2N$ as written in (2.17), we must compute $2N$ sums for each $\hat{u}_k$. Thus, using this naive approach to compute the DFT of a vector, we will need to use $O(N^2)$ operations. Therefore, if N is large, computing DFTs will be expensive. However, Cooley and Tukey [5], discovered a faster algorithm to compute the DFT. This algorithm is known as the fast Fourier transform (FFT). In Cooley and Tukey's seminal paper, they noted the following. If N is even, then

$$\begin{aligned}\mathcal{F}_{2N}(\underline{u})_k &= \frac{1}{2N}\sum_{n=-N}^{N-1} u_n e^{-ik\frac{\pi}{N}n} = \frac{1}{2N}\left[\sum_{n=-N/2}^{N/2-1} u_{2n}e^{-ik\frac{\pi}{N}2n} + \sum_{n=-N/2}^{N/2-1} u_{2n+1}e^{-ik\frac{\pi}{N}(2n+1)}\right] \\ &= \frac{1}{2N}\sum_{n=-N/2}^{N/2-1} u_{2n}e^{-ik\frac{2\pi}{N}n} + \frac{1}{2N}e^{-ik\frac{\pi}{N}}\sum_{n=-N/2}^{N/2-1} u_{2n+1}e^{-ik\frac{2\pi}{N}n} \\ &= \frac{1}{2}\mathcal{F}_N(\underline{u}^e)_k + \frac{1}{2}e^{-ik\frac{\pi}{N}}\mathcal{F}_N(\underline{u}^o)_k,\end{aligned}$$

where $\underline{u}^e_n = \underline{u}_{2n}$ and $\underline{u}^o_n = \underline{u}_{2n+1}$ for $-N/2 \le n \le N/2-1$. However, this is only valid for $-N/2 \le k \le N/2-1$. To compute $\mathcal{F}_{2N}(\underline{u})_k$ for values of k for the other values of k, we note

$$\hat{u}_{k\pm 2N} = \frac{1}{2N}\sum_{n=-N}^{N-1} u_n e^{-i(k\pm 2N)\frac{\pi}{N}n} = \frac{1}{2N}\sum_{n=-N}^{N-1} u_n e^{-ik\frac{2\pi}{N}n} = \hat{u}_k,$$

and

$$e^{-i(k-N)\frac{\pi}{N}} = e^{i\pi}e^{-ik\frac{\pi}{N}} = -e^{-ik\frac{\pi}{N}}.$$

So we can calculate $\hat{u}_k$ with

$$\hat{u}_k = \begin{cases} \frac{1}{2}\mathcal{F}_N(\underline{u}^e)_{k+N} - \frac{1}{2}e^{-i\frac{\pi}{N}}\mathcal{F}_N(\underline{u}^o)_{k+N}, & \text{for } k < -N/2 \\ \frac{1}{2}\mathcal{F}_N(\underline{u}^e)_k + \frac{1}{2}e^{-i\frac{\pi}{N}}\mathcal{F}_N(\underline{u}^o)_k, & \text{for } -N/2 \le k \le N/2-1 \\ \frac{1}{2}\mathcal{F}_N(\underline{u}^e)_{k-N} - \frac{1}{2}e^{-i\frac{\pi}{N}}\mathcal{F}_N(\underline{u}^o)_{k-N}, & \text{for } N/2-1 < k. \end{cases}$$

Therefore, if N is even, rather than calculating one DFT of length $2N$, which takes CN^2 operations to compute, we compute two DFTs of length N, which takes $C(N/2)^2$ operations to compute. Furthermore, if N is a large power of two, this approach may be repeated recursively, giving a computation time of $O(N\log(N))$. Since Cooley and Tukey's paper appeared in 1965, the FFT has been improved (for example in the case when $N = pq$ for $p, q \in \mathbb{Z}$ and for N prime using Rader's algorithm) to compute DFTs using $O(N\log(N))$ operations for any N. These cases will not be discussed here, and instead we refer to [7] and the references therein.

2.5 Nonlinearity and Aliasing

We may solve nonlinear PDEs using the pseudospectral method. However, when we have a nonlinear product, for example uv for $u, v \in V_{2N}$, we have [2]

$$uv = \left(\sum_{p=-N}^{N-1} \hat{u}_p \phi_p \right) \left(\sum_{q=-N}^{N-1} \hat{v}_q \phi_q \right) = \sum_{k=-2N}^{2N-2} \hat{w}_k \phi_k \in V_{4N}, \tag{2.21}$$

where $\hat{w}_k$ is given by

$$\hat{w}_k = \sum_{p+q=k} \hat{u}_p \hat{v}_q.$$

Thus, we can see that the space V_{2N} is not closed under products.

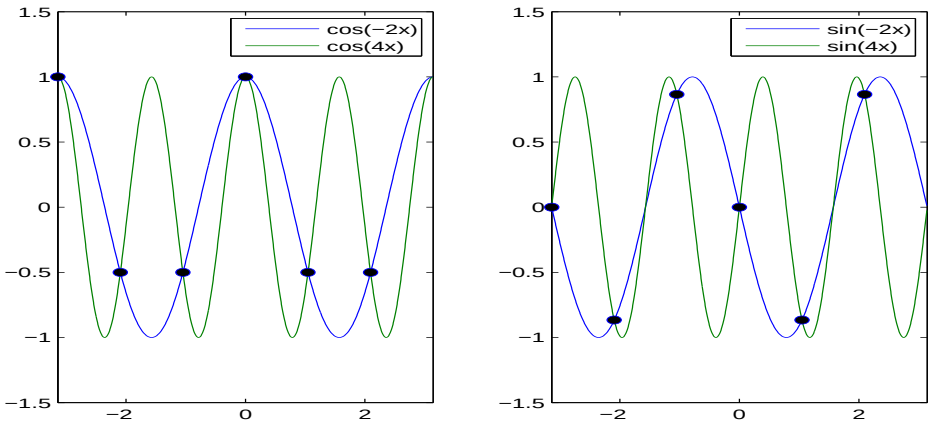

Figure 2.3: Plot of the real and imaginary parts of the functions e^{-i2x} and e^{i4x}. These functions are identical on the six quadrature points.

Figure 2.3 illustrates the grave nature of these nonlinear products. We have plotted the real and imaginary parts of $e^{-i2x} \in V_6$ and $e^{i4x} \in V_{12}$, the square of $e^{i2x} \in V_6$. We see that while these are formally different functions, at the quadrature points, these functions are equal. We say that the function e^{i4x} has been *aliased* onto the function e^{-i2x}. Thus, when we compute products of functions, we must take caution.

A simple solution to prevent aliasing is to set the upper half of the Fourier coefficients to zero, making the sum in (2.21) go over the original range of k. However, Orszag [14], found a better solution, now known as Orszag's two-thirds rule: only the upper one-third of

the modes need to zeroed. While aliasing will still occur, it will only happen to the modes that will be zeroed out anyways by this rule. Thus we may compute products using the two-thirds dealiasing rule with

$$\underline{uv} = \mathcal{F}_{2N}^{-1}[\rho_{2/3}(\underline{k}/N) \circ \mathcal{F}_{2N}(\underline{u})] \circ \mathcal{F}_{2N}^{-1}[\rho_{2/3}(\underline{k}/N) \circ \mathcal{F}_{2N}(\underline{v})],$$

where $\circ$ represents the Hadamard (entrywise) product, $[\underline{k}]_k = k$, and $\rho_{2/3}(k/N)$ is the two-thirds dealiasing rule

$$\rho_{2/3}(k/N) = \begin{cases} 1 & \text{if } |k|/N \leq \frac{2}{3} \\ 0 & \text{if } |k|/N > \frac{2}{3}. \end{cases} \tag{2.22}$$

In addition to the two-thirds rule to prevent aliasing, we also consider a Fourier smoothing method proposed by Hou and Li [12]. Instead of applying an all or nothing approach to the Fourier coefficients as the two-thirds rule does, we smoothly decay the higher Fourier coefficients with

$$\rho_{FSM}(k/N) = \exp(-\alpha(|k|/N)^m). \tag{2.23}$$

Here, we choose $\alpha = 36$ so that $\rho_{FSM}(1)$ is near the machine epsilon for double precision floating-points (approximately 10^{-16}) and $m = 36$ so that approximately four-fifths of the Fourier coefficients are retained. We then compute nonlinear terms with

$$\underline{uv} = \mathcal{F}_{2N}^{-1}[\rho_{FSM}(\underline{k}/N) \circ \mathcal{F}_{2N}(\underline{u})] \circ \mathcal{F}_{2N}^{-1}[\rho_{FSM}(\underline{k}/N) \circ \mathcal{F}_{2N}(\underline{v})].$$

Figure 2.4 shows the difference between the two methods. As we can see from this figure, many of the higher modes which are lost in the two-thirds rule are retained in the Fourier smoothing method. Since most of the aliasing occurs in only the highest modes, this will create only a small amount of aliasing error. The advantage of the Fourier smoothing method is that instead of only keeping two-thirds of the modes, we keep nearly four-fifths of the modes. This effect is much stronger in higher dimensions. For example, in three dimensions, one-third of the coefficients are lost in each dimension, giving only 8/27 (less than 1/3) of the original number of modes, however, with the Fourier smoothing method,

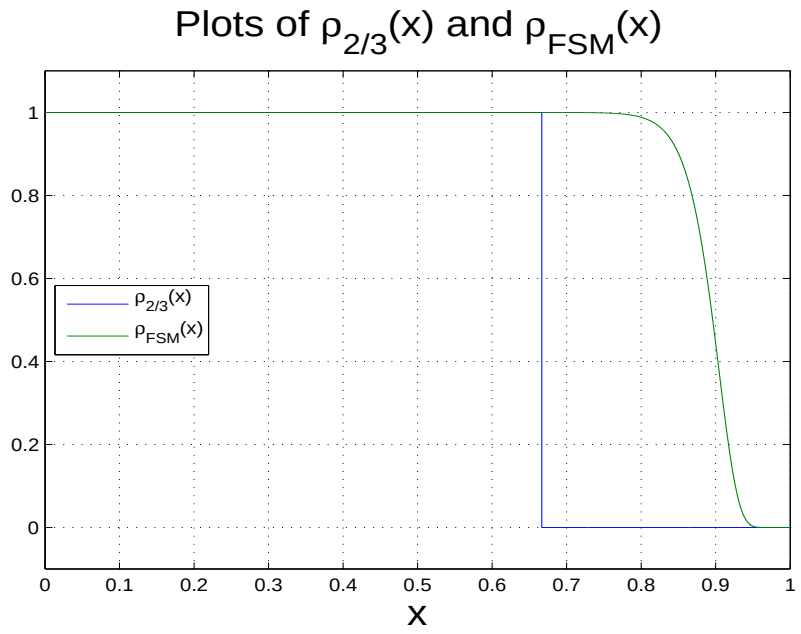

Figure 2.4: Comparison between the two dealiasing methods. The functions $\rho_{2/3}(x)$ and $\rho_{FSM}(x)$ are plotted.

over half of the coefficients are retained. The Fourier smoothing method also avoids aphysical oscillations, called the Gibbs phenomenon, as the Fourier smoothing method ensures that the Fourier coefficients decay gently, creating a smooth function.

We saw in this chapter how the Fourier pseudospectral method may be used to discretize a function along with its derivatives in space. By using Fourier series to represent functions, we saw that the errors from this spatial discretization is based on the periodic regularity of the functions. We then explored how to efficiently compute discrete Fourier transforms using the FFT. We then ended by examining issues with the pseudospectral method when dealing with nonlinear equations as well as two techniques for correcting this problem. We now turn our attention to how we may discretize our evolutionary systems in time.

CHAPTER 3

TEMPORAL DISCRETIZATION AND ANALYSIS

In the last chapter, we discretized a typical model problem in space by approximating unknown functions by truncated Fourier series. This also allowed us to compute spatial derivative easily. Once we have done this, our PDEs, which depended on both space and time, can be converted into a system of ODEs that depends on time alone. This is known as the method of lines [2]. In the rest of this chapter, we focus on how to solve these systems of ODEs numerically. The ODE solvers that we consider may be found in [9].

3.1 Time Integrators

Consider the evolutionary system (1.3) in one spatial dimension. Using the techniques discussed in Chapter 2, we may approximate this equation at the quadrature points $\{x_n\}_{n=-N}^{N-1}$. Thus, we will have $u(x_n,t) \approx [\underline{u}]_n(t), L[u(x_n,t)] \approx [\underline{L}]_n(t)$, and $f(x_n,t) \approx [\underline{f}]_n(t)$. Therefore, we may approximate (1.3) as

$$\underline{u}'(t) = -\underline{L}(t) + \underline{f}(t), \tag{3.1}$$

a system of $2N$ ODEs. In this chapter, we consider a typical model ODE

$$\begin{cases} y'(t) &= f(t,y(t)) \quad \text{for } t \in (0,T] \\ y(0) &= y_0. \end{cases} \tag{3.2}$$

We may solve this ODE on a time grid $0 = t^0 < t^1 < \ldots < t^M = T$ for some $M \in \mathbb{N}$. In this thesis, we consider only uniform time grids. That is, we consider a time grid with points $t^m = m\Delta t$ for $m = 0, 1, 2, \ldots, M$, with $\Delta t = T/M$. Using the Fundamental Theorem of Calculus, we may solve (3.2) with

$$\begin{cases} y(t^{m+1}) &= y(t^m) + \int_{t^m}^{t^{m+1}} f(t,y(t))\,\mathrm{d}t \quad \text{for } 0 \le m \le M-1, \\ y(t^0) &= y_0. \end{cases} \tag{3.3}$$

The integral in (3.3) may not be evaluated exactly in general. Thus, we approximate this integral to get an approximation $y^m \approx y(t^m)$, where y^m is calculated with a time integrator

of the form

$$\begin{cases} y^{m+1} & = y^m + \Phi(t^m, t^{m+1}, y^m, y^{m+1}, \Delta t, f), \quad \text{for } 0 \le m \le M-1 \\ y^0 & = y_0. \end{cases} \tag{3.4}$$

Here, $\Phi(t^m, t^{m+1}, y^m, y^{m+1}, \Delta t, f)$ is called the *increment function* [9].

The first method we will consider is constructed by approximating the integral in (3.3) using the left-hand rectangular rule, giving the forward Euler method

$$\begin{cases} y^{m+1} & = y^m + \Delta t f(t^m, y^m) \quad \text{for } 0 \le m \le M-1 \\ y^0 & = y_0, \end{cases} \tag{3.5}$$

where the increment function for the forward Euler method is

$$\Phi_{FE}(t^m, t^{m+1}, y^m, y^{m+1}, \Delta t, f) = \Delta t f(t^m, y^m).$$

In Section 3.2, we will examine the error of this method. We will see that the forward Euler method has a first-order of convergence in time. In Chapter 2, we saw that smooth periodic functions enjoyed a high order of convergence in space. Thus, we will want to use time integrators that also have a high order of convergence. To this end, the next time integrator we will consider is the classic fourth-order Runge-Kutta method. The Runge-Kutta method can be thought of as a generalization of approximating the integral in (3.3) with Simpson's rule. The Runge-Kutta method is given by

$$\begin{cases} y^{m+1} & = y^m + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad \text{for } 0 \le m \le M-1 \\ y^0 & = y_0, \end{cases} \tag{3.6}$$

where

$$\begin{aligned} k_1 &= f\left(t^m, y^m\right), \\ k_2 &= f\left(t^{m+\frac{1}{2}}, y^m + \frac{\Delta t}{2} k_1\right), \\ k_3 &= f\left(t^{m+\frac{1}{2}}, y^m + \frac{\Delta t}{2} k_2\right), \\ k_4 &= f\left(t^{m+1}, y^m + \Delta t k_3\right). \end{aligned}$$

The increment function for the Runge-Kutta method is

$$\Phi_{RK}(t^m, t^{m+1}, y^m, y^{m+1}, \Delta t, f) = \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

where k_1, k_2, k_3, and k_4 are defined as before.

For both of these methods, the increment functions do not depend on y^{m+1}. We call these methods *explicit* as y^{m+1} depends on an explicit function of y^m. We will see in Section 3.3 that explicit methods, in certain cases, will require severe restrictions on Δt to give meaningful results. Such restrictions may be avoided by using increment functions that depend on y^{m+1}. For these method, y^{m+1} will depend on an implicit equation of y^m and y^{m+1}, and thus, these methods are referred to as *implicit* methods.

The first implicit method that we consider is given by approximating the integral in (3.3) using the right-hand rectangular rule, called the backward Euler method

$$\begin{cases} y^{m+1} & = y^m + \Delta t f(t^{m+1}, y^{m+1}) \quad \text{for } 0 \leq m \leq M-1 \\ y^0 & = y_0. \end{cases} \tag{3.7}$$

The increment function of the backwards Euler method is then given by

$$\Phi_{BE}(t^m, t^{m+1}, y^m, y^{m+1}, \Delta t, f) = \Delta t f(t^{m+1}, y^{m+1}).$$

This method, like its explicit counterpart, also has first-order convergence in time. Therefore, as we did with our explicit methods, we seek an implicit method with a higher order of convergence. To this end, the final time integrator that we will consider is the trapezoidal rule, which approximates the integral in (3.3) with the trapezoidal rule. The trapezoidal rule is given by

$$\begin{cases} y^{m+1} & = y^m + \frac{\Delta t}{2}\left(f(t^m, y^m) + f(t^{m+1}, y^{m+1})\right) \quad \text{for } 0 \leq m \leq M-1 \\ y^0 & = y_0 \end{cases} \tag{3.8}$$

The increment function of the trapezoidal rule is then given by

$$\Phi_{TR}(t^m, t^{m+1}, y^m, y^{m+1}, \Delta t, f) = \frac{\Delta t}{2}\left(f(t^m, y^m) + f(t^{m+1}, y^{m+1})\right).$$

These implicit methods, which we will show in Section 3.3, have good stability properties allowing us to use large time steps. The implicit methods, however, suffer by defining y^{m+1}

in terms of an implicit equation. Therefore, we will need a method to solve these equations. To solve implicit equations, we use Newton's method

Theorem 3.8. *[1, 3.9.5] If x_n is an approximation to the solution of $F(x_*) = 0$, then the sequence*

$$x_{n+1} = x_n - \frac{F(x_n)}{F'(x_n)}$$

will converge quadratically to x_* quadratically.

Using Newton's method, we can now solve our implicit methods for y^{m+1}. To do this, we need three things: x_0, $F(x)$, and $F'(x)$. To get our starting point, x_0, we need a good guess of y^{m+1}. We will use the solution for y^{m+1} given by using the forward Euler method. That is, we let

$$x_0 = y^m + \Delta t f(t^m, y^m). \tag{3.9}$$

To get $F(x)$, we need a function such that $F(y^{m+1}) = 0$. Therefore, we define $F(x)$ as

$$F(x) = y^m - x + \Phi(t^m, t^{m+1}, y^m, x, \Delta t, f), \tag{3.10}$$

where $\Phi(t^m, t^{m+1}, y^m, x, \Delta t, f)$ is the increment function of either the backward Euler method (3.7) or the trapezoidal rule (3.8). Finally, we need $F'(x)$. To compute $F'(x)$, however, will require $\Phi_x(t^m, t^{m+1}, y^m, x, \Delta t, f)$, which may not be able to be evaluated exactly. Thus, we approximate $\Phi_x(t^m, t^{m+1}, y^m, x, \Delta t, f)$ numerically using the forward difference scheme. The forward difference scheme to approximate $g'(x)$ is given by

$$g'(x) \approx \frac{g(x+h) - g(x)}{h}, \tag{3.11}$$

where $h > 0$ is a constant. Using (3.11), we may then approximate $F'(x)$ by

$$F'(x) \approx -1 + \frac{\Phi(t^m, t^{m+1}, y^m, x+h, \Delta t, f) - \Phi(t^m, t^{m+1}, y^m, x, \Delta t, f)}{h}. \tag{3.12}$$

Inserting (3.9), (3.10), and (3.12) into Theorem 3.8, we update our solution y^{m+1} with our implicit methods with

$$\begin{cases} x_{n+1} & = x_n - \frac{y^m - x_n + \Phi(t^m, t^{m+1}, y^m, x_n, \Delta t, f)}{-1 + \frac{\Phi(t^m, t^{m+1}, y^m, x_n + h, \Delta t, f) - \Phi(t^m, t^{m+1}, y^m, x_n, \Delta t, f)}{h}} \quad \text{for } n \in \mathbb{N}, \\ x_0 & = y^m + \Delta t f(t^m, y^m), \end{cases} \tag{3.13}$$

with $h > 0$. We stop iterating after $|x_{n+1} - x_n|$ is less than some tolerance, τ.

3.2 Order of Convergence of Time Integrators

In this section, we will examine the order of convergence of our time integrators. We begin by examining the error in our time integrators in one time step. The local truncation error, e^{m+1}, is the error accumulated in one time step, assuming that the solution is known exactly at the previous time [9]. The local truncation error is given formally as

$$e^{m+1} = y(t^{m+1}) - y(t^m) - \Phi(t^m, t^{m+1}, y(t^m), y(t^{m+1}), \Delta t, f), \tag{3.14}$$

where $\Phi(t^m, t^{m+1}, y(t^m), y(t^{m+1}), \Delta t, f)$ is the increment function associated with one of the methods.

In order to compute the local truncation error of a method, Taylor's theorem will play a critical role:

Theorem 3.9. *[16] Let $\Omega \subset \mathbb{R}, r \in \mathbb{N}$. If $f \in C^{r+1}(\Omega)$, for any $a \in \Omega$, we have*

$$f(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2}f''(a) + \ldots + \frac{(x-a)^r}{r!}f^{(r)} + O((x-a)^{r+1}).$$

Using Taylor's theorem, we now give the order of convergence for the four time integrators introduced in Section 3.1.

Theorem 3.10. *If $y \in C^3([0,T])$, then the local truncation error, e^{m+1}, of the forward Euler method* (3.5) *satisfies*

$$e^{m+1} = \frac{\Delta t^2}{2}y''(t^m) + O(\Delta t^3).$$

Proof.

$$\begin{aligned} e^{m+1} &= y(t^{m+1}) - y(t^m) - \Phi_{FE}(t^m, t^{m+1}, y(t^m), y(t^{m+1}), \Delta t, f) \\ &= y(t^{m+1}) - y(t^m) - \Delta t f(t^m, y(t^m)) = y(t^{m+1}) - y(t^m) - \Delta t y'(t^m) \\ &= y(t^{m+1}) - y(t^m) - \left[y(t^{m+1}) - y(t^m) - \frac{\Delta t^2}{2} y''(t^m) + O(\Delta t^3) \right] \\ &= \frac{\Delta t^2}{2} y''(t^m) + O(\Delta t^3). \end{aligned}$$

□

Theorem 3.11. *If $y \in C^5([0,T])$, then the local truncation error, e^{m+1}, of the Runge-Kutta method* (3.6) *satisfies*

$$e^{m+1} = O(\Delta t^5).$$

The proof of this theorem is omitted, as we instead refer to Section 9.5 of [13] for a derivation of the local truncation error of the Runge-Kutta method.

Theorem 3.12. *If $y \in C^3([0,T])$, then the local truncation error, e^{m+1}, of the backward Euler method* (3.7) *satisfies*

$$e^{m+1} = -\frac{\Delta t^2}{2} y''(t^{m+1}) + O(\Delta t^3).$$

Proof.

$$\begin{aligned} e^{m+1} &= y(t^{m+1}) - y(t^m) - \Phi_{BE}(t^m, t^{m+1}, y(t^m), y(t^{m+1}), \Delta t, f) \\ &= y(t^{m+1}) - y(t^m) - \Delta t f(t^{m+1}, y(t^{m+1})) = y(t^{m+1}) - y(t^m) - \Delta t y'(t^{m+1}) \\ &= y(t^{m+1}) - y(t^m) - \left[-y(t^m) + y(t^{m+1}) + \frac{\Delta t^2}{2} y''(t^{m+1}) + O(\Delta t^3) \right] \\ &= -\frac{\Delta t^2}{2} y''(t^{m+1}) + O(\Delta t^3). \end{aligned}$$

□

Theorem 3.13. *If $\in C^4([0,T])$, then the local truncation error, e^{m+1}, of the trapezoidal rule* (3.8) *satisfies*

$$e^{m+1} = -\frac{\Delta t^3}{12} y'''(t^{m+\frac{1}{2}}) + O(\Delta t^4).$$

Proof. We begin by noting that by Taylor's theorem, we have

$$y(t^{m+1}) = y(t^{m+\frac{1}{2}}) + \frac{\Delta t}{2}y'(t^{m+\frac{1}{2}}) + \frac{\Delta t^2}{8}y''(t^{m+\frac{1}{2}}) + \frac{\Delta t^3}{48}y'''(t^{m+\frac{1}{2}}) + O(\Delta t^4)$$
$$y(t^m) = y(t^{m+\frac{1}{2}}) - \frac{\Delta t}{2}y'(t^{m+\frac{1}{2}}) + \frac{\Delta t^2}{8}y''(t^{m+\frac{1}{2}}) - \frac{\Delta t^3}{48}y'''(t^{m+\frac{1}{2}}) + O(\Delta t^4).$$

Subtracting the two equations above yields

$$y(t^{m+1}) - y(t^m) = \Delta t y'(t^{m+\frac{1}{2}}) + \frac{\Delta t^3}{24}y'''(t^{m+\frac{1}{2}}) + O(\Delta t^4). \tag{3.15}$$

Taylor's theorem also gives

$$y'(t^{m+1}) = y'(t^{m+\frac{1}{2}}) + \frac{\Delta t}{2}y''(t^{m+\frac{1}{2}}) + \frac{\Delta t^2}{8}y'''(t^{m+\frac{1}{2}}) + O(\Delta t^3)$$
$$y'(t^m) = y'(t^{m+\frac{1}{2}}) - \frac{\Delta t}{2}y''(t^{m+\frac{1}{2}}) + \frac{\Delta t^2}{8}y'''(t^{m+\frac{1}{2}}) + O(\Delta t^3).$$

Adding these equations gives

$$y'(t^{m+1}) + y'(t^m) = 2y'(t^{m+\frac{1}{2}}) + \frac{\Delta t^2}{4}y'''(t^{m+\frac{1}{2}}) + O(\Delta t^3)$$
$$\Rightarrow \Delta t y'(t^{m+\frac{1}{2}}) = \frac{\Delta t}{2}(y'(t^{m+1}) + y(t^m)) - \frac{\Delta t^3}{8}y'''(t^{m+\frac{1}{2}}) + O(\Delta t^4). \tag{3.16}$$

Inserting (3.16) into (3.15), we have

$$y(t^{m+1}) - y(t^m) - \frac{\Delta t}{2}(y'(t^{m+1}) + y'(t^m)) = -\frac{\Delta t^3}{12}y'''(t^{m+\frac{1}{2}}) + O(\Delta t^4). \tag{3.17}$$

Now, using (3.17), we compute the local truncation error of the trapezoidal rule

$$\begin{aligned} e^{m+1} &= y(t^{m+1}) - y(t^m) - \Phi_{CN}(t^m, t^{m+1}, y(t^m), y(t^{m+1}), \Delta t, f) \\ &= y(t^{m+1}) - y(t^m) - \frac{\Delta t}{2}\left[(f(t^{m+1}, y(t^{m+1})) + f(t^m, y(t^m)))\right] \\ &= y(t^{m+1}) - y(t^m) - \frac{\Delta t}{2}[y'(t^{m+1}) + y'(t^m)] \\ &= -\frac{\Delta t^3}{12}y'''(t^{m+\frac{1}{2}}) + O(\Delta t^4). \end{aligned}$$

□

While the local truncation error tells us the error in each time step of a method, it does not tell us what the total accumulated error of the method is. To this end, we now define the *global truncation error*, E^{m+1}, of a method as

$$E^{m+1} = y(t^{m+1}) - y^{m+1}. \tag{3.18}$$

The global truncation error tells us the total accumulation of errors after many time steps. Intuitively, we may think of the global truncation error as the sum of each of the local truncation errors. Therefore, if we have a method with a local truncation error of $O(\Delta t^{r+1})$, at the final time $T = M\Delta t = O(1)$ (so that $M = O(\Delta t^{-1})$), we would have a global truncation error of $MO(\Delta t^{r+1}) = O(\Delta t^r)$. Thus, we would have the global truncation error is one order less than the local truncation error. However, for this to be the case, we must be able to bound the propagation of the errors throughout the computations. To this end, we now define *Lipschitz continuous* functions. A function $f : \mathbb{C}^n \to \mathbb{C}$ is said to be Lipschitz continuous if there is a $K \geq 0$ such that for any $\underline{x}, \underline{y} \in \mathbb{C}^n$,

$$|f(\underline{x}) - f(\underline{y})| \leq K|\underline{x} - \underline{y}|. \tag{3.19}$$

The smallest positive K to satifiy (3.19) is called the *Lipschitz constant* of f.

Using the definition of Lipschitz continuity, we now state a theorem relating a method's local truncation error to its global truncation error.

Theorem 3.14. *[9] A time integrator with an increment function* $\Phi(t^m, t^{m+1}, y^m, y^{m+1}, \Delta t, f) = B_{t^m,t^{m+1},\Delta t,f}(y^m, y^{m+1})$ *with a local truncation error of order* $O(\Delta t^{r+1})$ *has a global truncation error at it final time step,* E^M*, of order* $O(\Delta t^r)$ *if* $B_{t^m,t^{m+1},\Delta t,f}(y^m, y^{m+1})$ *is Lipschitz continuous with a Lipschitz constant* $K < 1$.

Therefore, by 3.14, we have that the global truncation errors of the forward and backward Euler methods are of first order, the Runge-Kutta method is of forth order, and the trapezoidal rule is of second order, provided they satisfy the above conditions. We now continue on the topic of ODE s solvers by examining when these methods give meaningful results.

3.3 Stability of Time Integrators

In this section, we will discuss stability issues that arise from solving differential equations with the numerical procedures discussed earlier in this chapter. We will give only a brief introduction to this subject and refer to [10] for a more detailed analysis.

We begin by considering the following ODE

$$\begin{cases} y'(t) &= \lambda y(t) \text{ for } t > 0 \\ y(0) &= 1, \end{cases} \tag{3.20}$$

where $\lambda \in \mathbb{C}^- = \{z \in \mathbb{C} : Re(z) \leq 0\}$. The exact solution of (3.20) is

$$y(t) = e^{\lambda t}. \tag{3.21}$$

By solving (3.20) with the forward Euler method (3.5) with $f(t, y) = \lambda y$, we have

$$y^{m+1} = (1 + \lambda \Delta t) y^m = R(\lambda \Delta t) y^m, \tag{3.22}$$

where $R(z) = (1 + z)$ is called the *stability function* of the forward Euler method. Therefore, we have

$$y^m = [R(\lambda \Delta t)]^m. \tag{3.23}$$

Since, $\lambda \in \mathbb{C}^-$, the exact solution from (3.21) remains bounded as $t \to \infty$. We want our numerical solution to share this same property. The numerical solution from (3.23) will stay bounded as $t = m\Delta t \to \infty$ if

$$|R(\lambda \Delta t)| = |1 + \lambda \Delta t| \leq 1. \tag{3.24}$$

We call the set of points $\lambda \Delta t \in \mathbb{C}$ that satisfy (3.24) the *stability region* of the forward Euler method and refer to the forward Euler method as being *stable* if $\lambda \Delta t$ lies within its stability region.

We continue our stability analysis to the other time integrators introduced in Section 3.1. Solving (3.20) with the Runge-Kutta method (3.6), we have

$$y^{m+1} = y^m + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

where k_1, k_2, k_3 and k_4 are given by

$$
\begin{aligned}
k_1 &= \lambda y^m, \\
k_2 &= \lambda y^m + \frac{\lambda^2 \Delta t}{2} y^m = \left(\lambda + \frac{\lambda^2 \Delta t}{2}\right) y^m, \\
k_3 &= \lambda y^m + \frac{\lambda \Delta t}{2}\left(\lambda + \frac{\lambda^2 \Delta t}{2}\right) y^m = \left(\lambda + \frac{\lambda^2 \Delta t}{2} + \frac{\lambda^3 \Delta t^2}{4}\right) y^m, \\
k_4 &= \lambda y^m + \lambda \Delta t \left(\lambda + \frac{\lambda^2 \Delta t}{2} + \frac{\lambda^3 \Delta t^2}{4}\right) y^m = \left(\lambda + \lambda^2 \Delta t + \frac{\lambda^3 \Delta t^2}{2} + \frac{\lambda^4 \Delta t^3}{4}\right) y^m.
\end{aligned}
$$

Thus

$$
k_1 + 2k_2 + 2k_3 + k_4 = \left(6\lambda + 3\lambda^2 \Delta t + \lambda^3 \Delta t^2 + \frac{\lambda^4 \Delta t^3}{4}\right) y^m,
$$

and therefore

$$
\begin{aligned}
y_{m+1} &= y^m + \frac{\Delta t}{6}\left(6\lambda + 3\lambda^2 \Delta t + \lambda^3 \Delta t^2 + \frac{\lambda^4 \Delta t^3}{4}\right) y^m \\
&= \left(1 + \lambda \Delta t + \frac{(\lambda \Delta t)^2}{2} + \frac{(\lambda \Delta t)^3}{6} + \frac{(\lambda \Delta t)^4}{24}\right) y^m = R(\lambda \Delta t) y^m.
\end{aligned}
$$

So that the Runge-Kutta method will be stable if

$$
|R(\lambda \Delta t)| = \left|1 + \lambda \Delta t + \frac{(\lambda \Delta t)^2}{2} + \frac{(\lambda \Delta t)^3}{6} + \frac{(\lambda \Delta t)^4}{24}\right| \leq 1. \tag{3.25}
$$

We now analyze the stability of the implicit methods and will see that they have good stability properties. Solving (3.20) with the backward method (3.7), we have

$$
y^{m+1} = y^m + \lambda \Delta t y^{m+1}
$$

so that

$$
y^{m+1} = \frac{1}{1 - \lambda \Delta t} y^m = R(\lambda \Delta t) y^m. \tag{3.26}
$$

Thus, the backward Euler method will be stable if

$$
|R(\lambda \Delta t)| = \left|\frac{1}{1 - \lambda \Delta t}\right| \leq 1 \Rightarrow |1 - \lambda \Delta t| \geq 1. \tag{3.27}
$$

Here, we see that unlike the explicit methods, which had bounded stability regions, the implicit backward Euler method has an unbounded stability region. Furthermore, the stability region for the backwards Euler method contains $\mathbb{C}^-$ so that the numerical solution stays bounded when the exact solution remains bounded. We refer to methods that have this property as being *A-stable*, a property that will be important for the discretization of several of the space-time models presented later.

We now analyze the stability of the trapezoidal rule (3.8). Solving (3.20) with the trapezoidal rule, we have

$$y^{m+1} = y^m + \frac{\Delta t}{2}(\lambda y^m + \lambda y^{m+1})$$

so that

$$y^{m+1} = \left(\frac{1 + \frac{\lambda \Delta t}{2}}{1 - \frac{\lambda \Delta t}{2}}\right) y^m = R(\lambda \Delta t) y^m. \tag{3.28}$$

We see that the trapezoidal rule will be stable if

$$|R(\lambda \Delta t)| = \left|\frac{1 + \frac{\lambda \Delta t}{2}}{1 - \frac{\lambda \Delta t}{2}}\right| \leq 1 \Rightarrow Re(\lambda \Delta t) \leq 0 \Rightarrow \lambda \Delta t \in \mathbb{C}^-, \tag{3.29}$$

and therefore the trapezoidal rule is also A-stable.

3.4 Approximation of the Rate of Convergence

As we are using the method of lines, the solutions of PDEs will be discretized in both space and time. Thus, we will have both spatial and temporal errors, giving a total error in our approximation of the form

$$|u(x,T) - u_N^{\Delta t}(x,T)| = C\Delta t^r + O(N^{-s} + \Delta t^{r+1}).$$

If the spatial approximation is computed with sufficient accuracy, then the error in our solution will be dominated by the term $C\Delta t^r$. Therefore, if the exact solution is known, we can confirm our order of convergence in time by comparing two solutions with different time steps, say Δt and $\Delta t/2$. We have

$$u(x,t) - u_N^{\Delta t}(x,t) \approx C\Delta t^r, \quad u(x,t) - u_N^{\Delta t/2}(x,t) \approx C(\Delta t/2)^r.$$

Dividing these equations gives

$$\frac{u(x,t) - u_N^{\Delta t}(x,t)}{u(x,t) - u_N^{\Delta t/2}(x,t)} \approx 2^r.$$

So, by taking the logarithm of base 2, we may compute the estimated the order of convergence when $u(x,t)$ is known, EOC_k, with

$$EOC_k(x,t) = \log_2 \left[\frac{u(x,t) - u_N^{\Delta t}(x,t)}{u(x,t) - u_N^{\Delta t/2}(x,t)} \right] \approx r. \tag{3.30}$$

If the exact solution is unknown, we may still get an estimated order of convergence by examining the solutions found with three different time steps:

$$\begin{aligned} u(x,t) - u_N^{\Delta t}(x,t) &\approx C\Delta t^r \\ u(x,t) - u_N^{\Delta t/2}(x,t) &\approx C(\Delta t/2)^r \\ u(x,t) - u_N^{\Delta t/4}(x,t) &\approx C(\Delta t/4)^r. \end{aligned}$$

By subtracting the first two equations and similarly the last equations, we have

$$u_N^{\Delta t/2}(x,t) - u_N^{\Delta t}(x,t) \approx C(\Delta t/2)^r \quad u_N^{\Delta t/4}(x,t) - u_N^{\Delta t/2}(x,t) \approx C(\Delta t/4)^r.$$

By dividing these two equations and taking the logarithm of base two, we may compute the estimated order of convergence when the exact solution is unknown, EOC_u, with

$$EOC_u(x,t) = \log_2 \left[\frac{u_N^{\Delta t/2}(x,t) - u_N^{\Delta t}(x,t)}{u_N^{\Delta t/4}(x,t) - u_N^{\Delta t/2}(x,t)} \right] \approx r. \tag{3.31}$$

In this chapter, we introduced four numerical methods for solving ordinary differential equations. We examined the errors of these methods in a single time step which we used to get an estimate of the total error accumulated over all of the time steps. We continued by exploring under which circumstances the ODE solvers will give meaningful results by looking at the stability of the methods. The chapter then concluded with techniques that may be used to estimate how the numerical solutions converge in both the case when the exact solution is known and when it is unknown. We now put the theory developed in these last two chapters into practice by applying the Fourier pseudospectral method of lines to several one-dimensional evolutionary problems.

CHAPTER 4

PSEUDOSPECTRAL METHODS FOR LINEAR AND NONLINEAR ONE-DIMENSIONAL EVOLUTION SYSTEMS

In this chapter, we develop and demonstrate pseudospectral methods to numerically solve a class of linear and nonlinear evolutionary systems. The pseudospectral methods are obtained by combining the Fourier pseudospectral method introduced in Chapter 2 with time discretization techniques described in Chapter 3. As we are seeking to solve equations related to fluid dynamics, we will want to study one-dimensional problems that share characteristics of the Navier-Stokes equations (1.1)-(1.2). We see that (1.1) contains both a quadratically nonlinear advection term and a linear diffusive term. Therefore, for our one-dimensional problems, we will want to study equations that have these types of terms. The first of these equations we will study is the forced heat equation which contains a linear diffusion term

$$\begin{cases} u_t - \nu u_{xx} = f \text{ in } \Omega \times (0,T] \\ u(x,0) = u_0(x) \\ \text{Periodic boundary conditions,} \end{cases} \tag{4.1}$$

where ν is a given positive constant.

The second equation we will study is the inviscid Burger's equation which has a quadratically nonlinear advection term

$$\begin{cases} u_t + \left(\frac{u^2}{2}\right)_x = f \text{ in } \Omega \times (0,T] \\ u(x,0) = u_0(x) \\ \text{Periodic boundary conditions.} \end{cases} \tag{4.2}$$

Finally, for the third equation that we will study is Burgers' equation which contains both a linear diffusive term as well as a quadratically nonlinear advection term

$$\begin{cases} u_t - \nu u_{xx} + \left(\frac{u^2}{2}\right)_x = f \text{ in } \Omega \times (0,T] \\ u(x,0) = u_0(x) \\ \text{Periodic boundary conditions,} \end{cases} \tag{4.3}$$

where ν is a given positive constant.

In this chapter, we solve each of these three equations using approaches introduced in Chapters 2 and 3.

4.1 The Heat Equation

In this section, we will study the first of our one-dimensional PDEs, the heat equation (4.1). We begin by first finding the exact solution for the unforced heat equation to create a benchmark to compare our numerical results to.

4.1.1 Solution of the Heat Equation

We may solve the unforced heat equation, (4.1) with $f = 0$, by the method of separation of variables. We assume the ansatz that the solution $u(x,t)$ may be written in the form:

$$u(x,t) = u_1(x)u_2(t). \tag{4.4}$$

This converts the partial derivatives of u in the original equation into ordinary derivatives of the functions u_1 and u_2. Substituting (4.4) into (4.1) gives

$$u_1 u_2' = \nu u_1'' u_2.$$

Dividing both sides by $\nu u_1 u_2$ yields

$$\frac{u_2'}{\nu u_2} = \frac{u_1''}{u_1} = c. \tag{4.5}$$

Here, we can see that c may depend on neither x nor t and is therefore a constant. We now separate our PDE into two ODEs:

$$u_2' = c\nu u_2, \quad u_1'' = cu_1.$$

Solving the time problem for u_2, we have

$$u_2(t) = e^{c\nu t}. \tag{4.6}$$

We now solve the spatial problem for u_1. If $c \geq 0$, enforcing the periodic boundary conditions will yield only the trivial solution. Therefore, we let $c = -\lambda^2 < 0$. Then we have

$$u_{1,\lambda}(x) = Ae^{i\lambda x} + Be^{-i\lambda x}, \tag{4.7}$$

where A and B are constants to be determined. As $e^{\pm i\lambda x}$ is $\frac{2\pi}{\lambda}$-periodic, our solution will be 2π-periodic if and only if $\lambda \in \mathbb{N}$. Imposing the initial condition

$$u(x,0) = u_0(x) = u_2(0)u_1(x),$$

and noting that $u_2(0) = 1$, we must have $u_0(x) = u_1(x)$. We see in (4.7) that as λ is a positive integer, $u_1(x)$ is of the form of a partial Fourier series. Therefore, if we let

$$u_0(x) = \sum_{k=-\infty}^{\infty} \hat{u}_k e^{ikx}$$

be the Fourier series of $u_0(x)$, we may solve (4.7) for each term

$$(u_1)_k(x) = \begin{cases} \hat{u}_0 & \text{if } k = 0 \\ \hat{u}_k e^{ikx} + \hat{u}_{-k} e^{-ikx} & \text{if } k > 0. \end{cases}$$

Adding these terms together, we get

$$u_1(x) = \sum_{k=-\infty}^{\infty} \hat{u}_k e^{ikx}. \tag{4.8}$$

The solution $u(x,t)$ is given by plugging (4.6) and (4.8) into (4.4), yielding

$$u(x,t) = \sum_{k=-\infty}^{\infty} \hat{u}_k e^{ikx - k^2\nu t}, \tag{4.9}$$

as $c = -k^2$.

4.1.2 Pseudospectral Methods for the Heat Equation

In this section, we will derive the pseudospectral method for the heat equation. We begin by discretizing the spatial derivatives as discussed in Chapter 2. We approximate the functions $u(\cdot,t), u_0$, and $f(\cdot,t)$ with their $2N$ Fourier approximations. This yields

$$\begin{cases} (u_{2N}(x,t))_t - \nu\,(u_{2N}(x,t))_{xx} = f_{2N}(x,t) \\ u_{2N}(x,t) = (u_0(x))_{2N}\,. \end{cases}$$

Evaluating the spatial derivatives, we have

$$-\nu \left(u_{2N}(x,t)\right)_{xx} = -\nu \sum_{k=-N}^{N-1} \hat{u}_k(t) \frac{\partial^2}{\partial x^2} e^{ikx} = -\nu \sum_{k=-N}^{N-1} \hat{u}_k(t)(-k^2) e^{ikx}.$$

We may then evaluate $-\nu \left(u_{2N}(x,t)\right)_{xx}$ at the quadrature points, $\{x_n\}_{n=-N}^{N-1}$, by using the IDFT (2.19), giving

$$\left(\underline{u}_{2N}\right)_{xx}(t) = \mathcal{F}_{2N}^{-1}[-\underline{k}^2 \circ \mathcal{F}_{2N}[\underline{u}_{2N}]],$$

where $\underline{k}^2 = [(-N)^2, (-N+1)^2, \ldots, (N-1)^2]^T$. Substituting and rearranging yields

$$\begin{cases} \left(\underline{u}_{2N}(t)\right)_t = \nu \mathcal{F}_{2N}^{-1}[-\underline{k}^2 \circ \mathcal{F}_{2N}[\underline{u}_{2N}(t)]] + \underline{f}_{2N}(t) \\ \underline{u}_{2N}(0) = \underline{u}_0, \end{cases} \tag{4.10}$$

which is in the form of (3.2).

We may then solve (4.10) with one the of the time integrators discussed in Section 3.1. For example, using the forward Euler method (3.5) we have

$$\begin{cases} \underline{u}_{2N}(t^{m+1}) = \underline{u}_{2N}(t^m) + \Delta t \left(\nu \mathcal{F}_{2N}^{-1}[-\underline{k}^2 \circ \mathcal{F}_{2N}[\underline{u}_{2N}(t^m)]] + \underline{f}_{2N}(t^m)\right) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{for } m = 0, 1, \ldots M-1 \\ \underline{u}_{2N}(t^0) = \underline{u}_0. \end{cases}$$

4.1.3 Simulation of the Heat Equation

In this section, we will compute the numerical solution of the heat equation (4.1) in several examples using pseudospectral methods. When solving the heat equation, it is important to consider stability conditions discussed in Section 3.3. We can see in (4.5) that the temporal ODE is of the form of (3.20) with $\lambda = c\nu = -k^2\nu$. Thus, for the forward Euler method to be stable, by (3.24), we need $|1 - \Delta t k^2 \nu| \leq 1$ for $k = -N, -N+1, \ldots, N-1$. As $\Delta t k^2 \nu$ is positive for all k and k^2 is maximized at $k = -N$, we need

$$\Delta t \leq \frac{2}{\nu N^2} \tag{4.11}$$

to ensure stability.

4.1.3.1 Example 1

In this first example, we show the necessity of this stability condition. For this example, we let $N = 16, \nu = 1, f = 0, u_0(x) = 2\pi^2 x^2 + x^4$, and $T = 1$. With this choice of N and ν, (4.11) becomes $\Delta t \leq 1/128$.

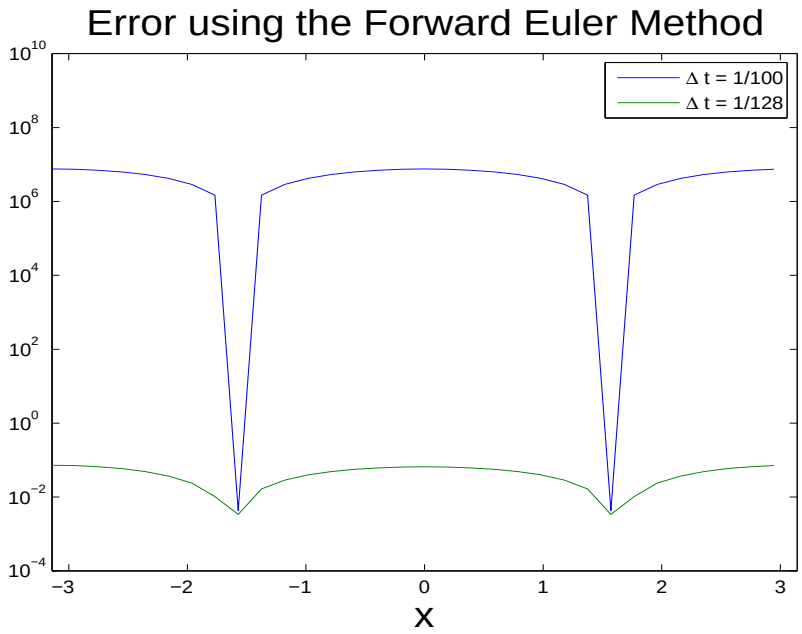

Figure 4.1: Plot of the error of the numerical solution of the heat equation obtained by the pseudospectral forward Euler method using two different time steps $\Delta t = 1/100$ (unstable) and $\Delta t = 128$ (stable).

In Figure 4.1, we show the error of the computed solution using the pseudospectral forward Euler method. We see that if our time step Δt satisfies (4.11), then the error in the solution is on the order of 10^{-1}. However, if this condition is not met, then the computed solution is very poor with a pointwise error on the order of 10^7. In Figure 4.2 we show the absolute value of the Fourier coefficients of the numerical solutions. We see the lower Fourier coefficients of the numerical solutions closely matches to the exact solution's Fourier coefficients. The error is caused by the instability of the highest modes, where the Fourier coefficient corresponding to $k = 15$ is on the order of 10^8.

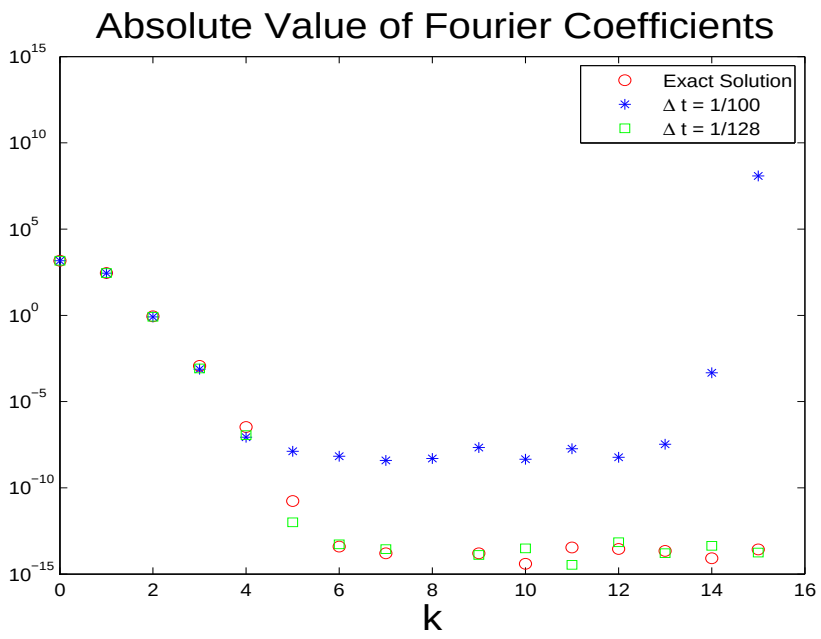

Figure 4.2: Plot of the absolute value of the Fourier coefficients for the numerical solution using the pseudospectral forward Euler method using two different time steps $\Delta t = 1/100$ (unstable) and $\Delta t = 1/128$ (stable).

We now compare these results using our implicit methods which do not have this stability requirement due to their A-stability. In Figure 4.3, we plot the numerical solution of the heat equation using both the backward Euler method as well as the trapezoidal rule. We see using a much larger time step $\Delta t = 1/N = 1/16$ gives reasonable results for both of these methods. We can also see that the error of the trapezoidal rule is about two orders of magnitude smaller than the error of the backward Euler method. This is expected as the trapezoidal method has a higher order of convergence, as discussed in Section 3.2.

4.1.3.2 Example 2

In our first example, the exact solution was known. However, the exact solution to many of the PDEs that we will be solving will be unknown. Thus, we will want a method that will allow us to test our code for problems that we may not solve exactly. To this end, we will exploit the freedom of the forcing function. We choose a function $u(x,t)$ that is spatially periodic to be the exact solution of our problem. Then, by plugging $u(x,t)$ into

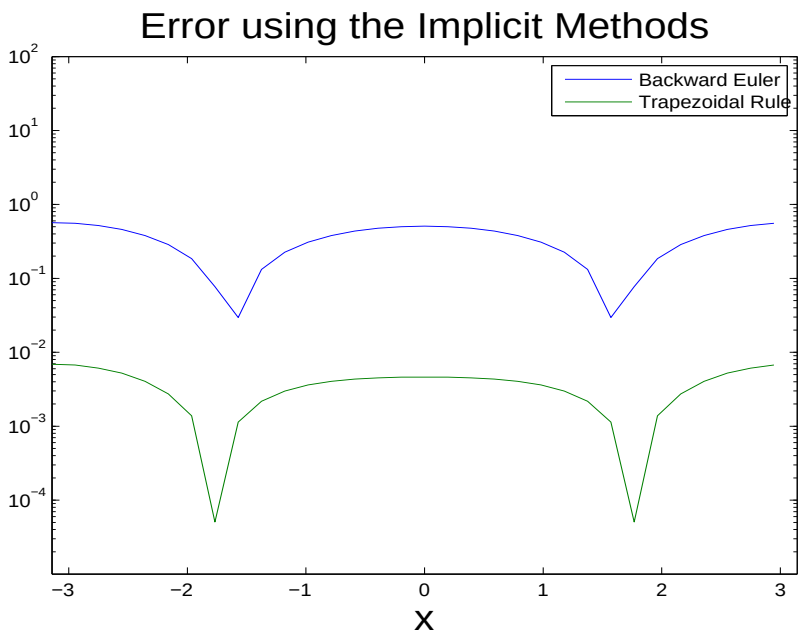

Figure 4.3: Plot of the numerical solution of the heat equation using the backward Euler method and trapezoidal rule using a time step of $\Delta t = 1/N = 1/16$.

the right-hand side of (4.1), we may find $f(x,t)$ such that this $u(x,t)$ solves the differential equation.

For this example, we will consider the solution $u(x,t) = e^{\sin(x)-\nu t}$. With this choice of an exact solution, we have $u_t(x,t) = -\nu e^{\sin(x)-\nu t}$ and $u_{xx}(x,t) = (\cos^2(x) - \sin(x))e^{\sin(x)-\nu t}$ so that $f(x,t) = -\nu(1 + \cos^2(t) - \sin(t))e^{\sin(x)-\nu t}$.

In Figure 4.4, we have plotted the numerical solution of the problem with $\nu = 1, N = 16, \Delta t = 1/2$, and $T = 1$ for both the backward Euler method and the trapezoidal rule. We now examine the order of convergence in time of these methods.

In Table 4.1 we have given, the value of of $u_{2N}^{\Delta t}(\pi/2, 1)$ computed using the backward Euler method for several different choices of Δt. We also show the estimated order of convergence of these methods calculated from (3.30) using that the exact solution is known ($u(\pi/2, 1) = 1$) as well as the estimated order of convergence from (3.31), where we do not assume to know the exact solution. Here, we see that for the backward Euler method, both estimated orders of convergence tend to one as expected from the error analysis covered in

Table 4.1: Estimated Order of Convergence using the Backward Euler method

Δt	$u_{2N}^{\Delta t}(\pi/2, 1)$	$EOC_k(\pi/2, 1)$	$EOC_u(\pi/2, 1)$
1/2	1.2783	0.9182	0.8750
1/4	1.1473	0.9578	0.9362
1/8	1.0758	0.9785	0.9677
1/16	1.0385	0.9892	0.9837
1/32	1.0194	0.9946	0.9918
1/64	1.0097	0.9973	0.9959
1/128	1.0049	0.9986	–
1/256	1.0024	–	–

Table 4.2: Estimated Order of Convergence using the trapezoidal rule

Δt	$u_{2N}^{\Delta t}(\pi/2, 1)$	$EOC_k(\pi/2, 1)$	$EOC_u(\pi/2, 1)$
1/2	0.9738	2.0066	2.0081
1/4	0.9935	2.0020	2.0025
1/8	0.9984	2.0005	2.0006
1/16	0.9996	2.0001	2.0002
1/32	0.9999	2.0000	2.0000
1/64	1.0000	2.0000	2.0000
1/128	1.0000	2.0000	–
1/256	1.0000	–	–

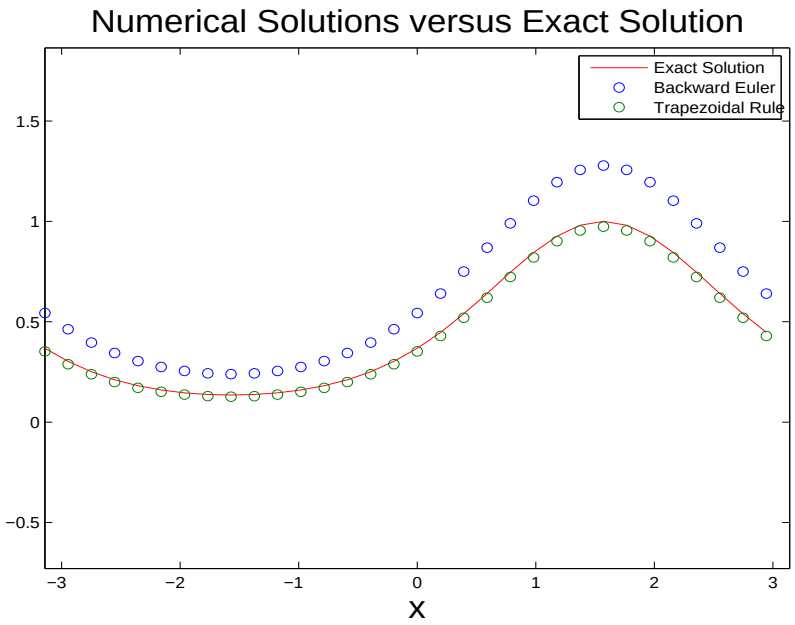

Figure 4.4: Plot of numerical solutions versus the exact solution using the backward Euler method and the trapezoidal rule with $\Delta t = 1/2$

Section 3.2. Similarly, in Table 4.2, we repeat the above analysis for the trapezoidal rule. Here, we see that both estimated orders of convergence tend to two as also expected.

4.2 The Inviscid Burgers' Equation

We now examine our second one-dimensional problem, the inviscid Burgers' equation (4.2). We begin by studying the exact solution of the inviscid Burgers' equation.

4.2.1 Solution of the Inviscid Burgers' Equation

To solve the unforced inviscid Burgers' equation, (4.2) with $f = 0$, we use the method of characteristics. In the method of characteristics, we find curves in the $x - t$ plane, called *characteristics*, such that the unknown function $u(x, t)$ is constant on. We write these curves by expressing x as a function of t so that $u(x(t), t)$ is constant. Taking the total derivative of u with respect to t, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}u(x(t), t) = u_t + x'(t)u_x = 0.$$

This will then hold, provided

$$x'(t) = u(x(t), t).$$

As u is constant, integrating yields

$$\begin{aligned} x(t) - x(0) &= tu(x(t), t) \\ \Rightarrow x(0) &= x(t) - tu(x(t), t) \\ \Rightarrow u_0(x(0)) &= u_0(x(t) - tu(x(t), t)) \end{aligned}$$

As $u_0(x(0)) = u(x(0), 0) = u(x(t), t)$, we have

$$u(x, t) = u_0(x - tu(x, t)), \tag{4.12}$$

which is an implicit equation of $u(x, t)$.

4.2.2 Pseudospectral Methods for the Inviscid Burgers' Equation

In this section, we derive pseudospectral methods for the inviscid Burgers' equation (4.2). We begin by discretizing the spatial derivatives as discussed in Chapter 2. We approximate the functions $u(\cdot, t), u_0$, and $f(\cdot, t)$ with their $2N$ Fourier approximations. This yields

$$\begin{cases} \left(u_{2N}(x,t)\right)_t + \left(\frac{u_{2N}^2(x,t)}{2}\right)_x = f_{2N}(x,t) \\ u_{2N}(x,t) = \left(u_0(x)\right)_{2N}. \end{cases} \tag{4.13}$$

The first thing we note is the nonlinear product u_{2N}^2. To compute this term, we will use a dealiasing rule from Section 2.5. Therefore we compute $\left(\frac{u_{2N}^2}{2}\right)_x$ by

$$\left(\frac{u_{2N}^2}{2}\right)_x = \mathcal{F}_{2N}^{-1}\left[\frac{i}{2}\underline{k} \circ \rho(\underline{k}/N) \circ \mathcal{F}_{2N}[\underline{u}_{2N} \circ \underline{u}_{2N}]\right],$$

where ρ is either the two-thirds dealiasing rule (2.22) or the Fourier smoothing method (2.23). Substituting this into (4.13) and rearranging terms gives

$$\begin{cases} \left(\underline{u}_{2N}(t)\right)_t = -\mathcal{F}_{2N}^{-1}\left[\frac{i}{2}\underline{k} \circ \rho(\underline{k}/N) \circ \mathcal{F}_{2N}[\underline{u}_{2N} \circ \underline{u}_{2N}]\right] + \underline{f}_{2N}(t) \\ \underline{u}_{2N}(0) = \underline{u}_0, \end{cases} \tag{4.14}$$

which is in the form of (3.2).

We may then solve (4.14) with one the of the time integrators discussed in Section 3.1. For example, using the Forward Euler method (3.5) we have

$$\begin{cases} \underline{u}_{2N}(t^{m+1}) = \underline{u}_{2N}(t^m) + \Delta t \left(\mathcal{F}_{2N}^{-1} \left[\frac{i}{2} \underline{k} \circ \rho(\underline{k}/N) \circ \mathcal{F}_{2N}[\underline{u}_{2N} \circ \underline{u}_{2N}] \right] + \underline{f}_{2N}(t^m) \right) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{for } m = 0, 1, \dots M-1 \\ \underline{u}_{2N}(t^0) = \underline{u}_0. \end{cases}$$

4.2.3 Simulation of the Inviscid Burgers' Equation

In this section, we will compute the numerical solution of the inviscid Burgers' equation using pseudospectral methods in a variety of examples. As was the case with the heat equation, for the inviscid Burgers' equation, we must still satisfy the stability conditions from Section 3.3. From [2, Line (12.8)], we have that $\lambda = -iku$. Thus, for a method to be stable, we require $\lambda \Delta t = -iku\Delta t$ to lie in the method's stability region for all $k = -N, -N+1, \dots, N-1$ and for all $u(x,t)$ for $x \in \Omega$ and $t \in [0,T]$. As $\lambda \Delta t$ is imaginary, we will consider the intersection of a method's stability with the imaginary axis. These intervals may be found in [3, Table D.1]. For the forward Euler method, the intersection of its stability region and the imaginary axis is the origin. Thus, the forward Euler method is unconditionally unstable for solving the inviscid Burgers' equation. For the Runge-Kutta method, the intersection of the stability region and the imaginary axis is the interval $i[-2.83, 2.83]$. Thus, for the Runge-Kutta method to be stable, we require $|ku\Delta t| \leq 2.83$ for all $k = -N, -N+1, \dots, N-1$ and $u(x,t) \in \Omega \times [0,T]$. Therefore, the stability condition for the Runge-Kutta method for solving the inviscid Burgers' equation is given by

$$\Delta t \leq \frac{2.83}{N u_{max}}, \tag{4.15}$$

where $u_{max} = \max\limits_{\substack{x \in \Omega \\ t \in [0,T]}} \{|u(x,t)|\}$.

4.2.3.1 Example 1

In our first example in simulating the inviscid Burgers' equation using pseudospectral methods, we will show how the stability condition (4.15) must be satisfied. For this example, we let $N = 256, f = 0, u_0(x) = \sin(x)$, and $T = 1/2$. For this choice of N and $u_0(x)$, we have a stability condition of $\Delta t \leq \frac{256}{2.83} \approx 1/90$. As this equation is nonlinear, we use the two-thirds dealiasing technique.

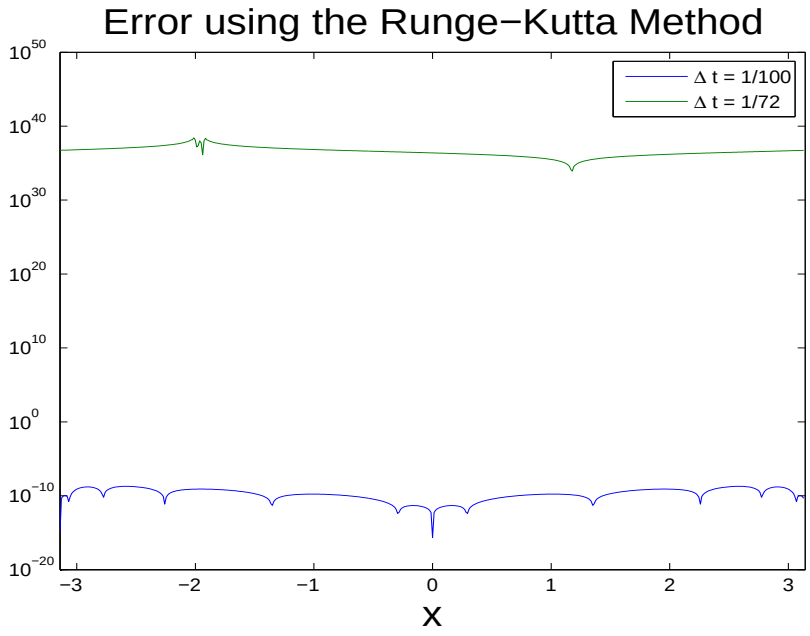

Figure 4.5: Error plot of numerical solutions obtained using the Runge-Kutta method with $\Delta t = 1/100$ (stable) and $\Delta t = 1/72$ (unstable)

In Figure 4.5, we plot the pointwise errors of the numerical solution obtained using the Runge-Kutta method for $\Delta t = 1/100$ and $\Delta t = 1/72$. For $\Delta t = 1/100$, the method is stable and the pointwise errors are on the order of 10^{-10}. In contrast, for $\Delta t = 1/72$ and the method is unstable, the pointwise errors are on the order of 10^{38}. Thus, to ensure good results, we must ensure that (4.15) is satisfied.

4.2.3.2 Example 2

In our second example for the inviscid Burgers' equation, we will pick the exact solution $u(x,t)$. Here, we consider the exact solution $u(x,t) = \sin(x-t)$ so that the initial condition is given by $u_0(x) = \sin(x)$. We also have $u_t(x,t) = -\cos(x-t)$ and $\left(\frac{u^2(x,t)}{2}\right)_x = \sin(x-t)\cos(x-t)$ so that $f(x,t) = \cos(x-t)[\sin(x-t)-1]$. For this example, we also have $N = 16$ and $T = 1$. For these choices of N and $u_0(x)$, we have the stability condition $\Delta t \leq \frac{16}{2.83} \approx \frac{1}{6}$. For this simulation, we use the two-thirds dealiasing technique.

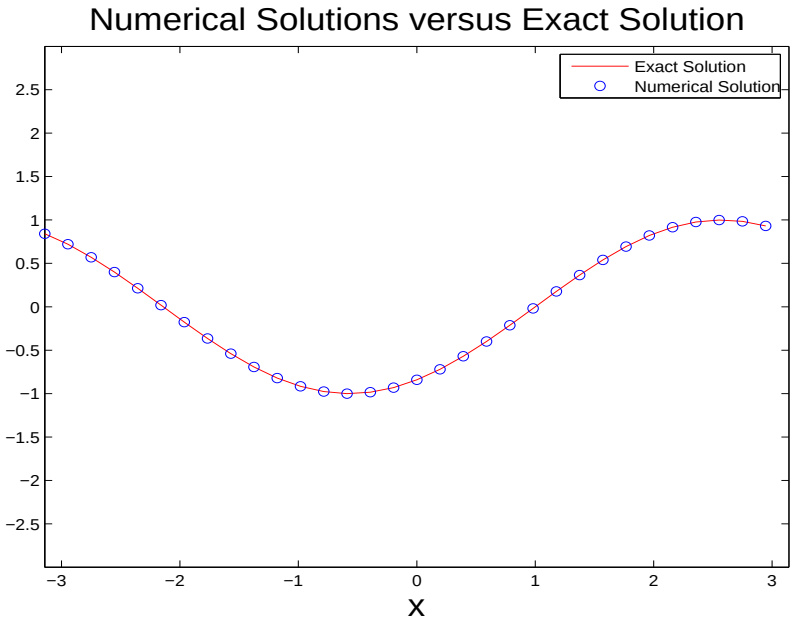

Figure 4.6: Plot of numerical solutions versus the exact solution using the Runge-Kutta method with $\Delta t = 1/8$

In Figure 4.6, we have plotted the numerical solution obtained using the Runge-Kutta method with the a time step of 1/8. We see that even with the large step size, the forth order Runge-Kutta method is able to give accurate results. We now examine the order of convergence in time of this method.

In Table 4.3 we give the value of $u_{2N}^{\Delta t}(\pi/2, 1)$ for various time steps. We also show the estimated order of convergence of these methods calculated from (3.30) and (3.31). Here we see that both estimated orders of convergence for the Runge-Kutta method approach the

Table 4.3: Estimated Order of Convergence using the Runge-Kutta method

Δt	$u_{2N}^{\Delta t}(\pi/2, 1)$	$EOC_k(\pi/2, 1)$	$EOC_u(\pi/2, 1)$
1/8	0.5403	4.2185	4.2241
1/16	0.5403	4.1227	4.1263
1/32	0.5403	4.0655	4.0675
1/64	0.5403	4.0338	4.0349
1/128	0.5403	4.0167	4.0172
1/256	0.5403	4.0090	–
1/512	0.5403	–	–

value of four as expected from the analysis done in Section 3.2.

4.2.4 A Non-smooth Solution and Comparison of Two Dealiasing Techniques

In our last example for the inviscid Burgers' equation, we would like to do some in-depth analysis between two dealiasing techniques, namely, the two-thirds dealiasing rule (2.22) and the Fourier smoothing method (2.23). We follow [12] and consider the unforced inviscid Burgers' equation (4.2) ($f = 0$). The initial condition used is $u_0(x) = \sin(x)$. For time integration, a third order Runge-Kutta method was used in [12]. In these simulations, we will use our fourth order Runge-Kutta method (3.6). While the solution remains smooth, we will expect to have higher accuracy, but as a shock singularity begins to develop, which occurs at $t = 1$ [12], the solution begins to lose regularity and therefore near $t = 1$, the spatial error will begin to dominate, giving similar errors.

In [12], the time step is determined using a Courant number $C = \pi/4$. The Courant number is defined as

$$C = \frac{u_{max}\Delta t}{\Delta x}, \tag{4.16}$$

where $u_{max} = \max_{\substack{x\in\Omega \\ t\in[0,T]}} \{|u|\}$. Thus, we let $\Delta t = \frac{\pi\Delta x}{4u_{max}}$. For an "exact solution", in [12], the implicit solution of the inviscid Burger's equation (4.12) is computed using Newton's method (3.8) with a tolerance $\tau = 10^{-13}$. We follow the same approach for our exact solution.

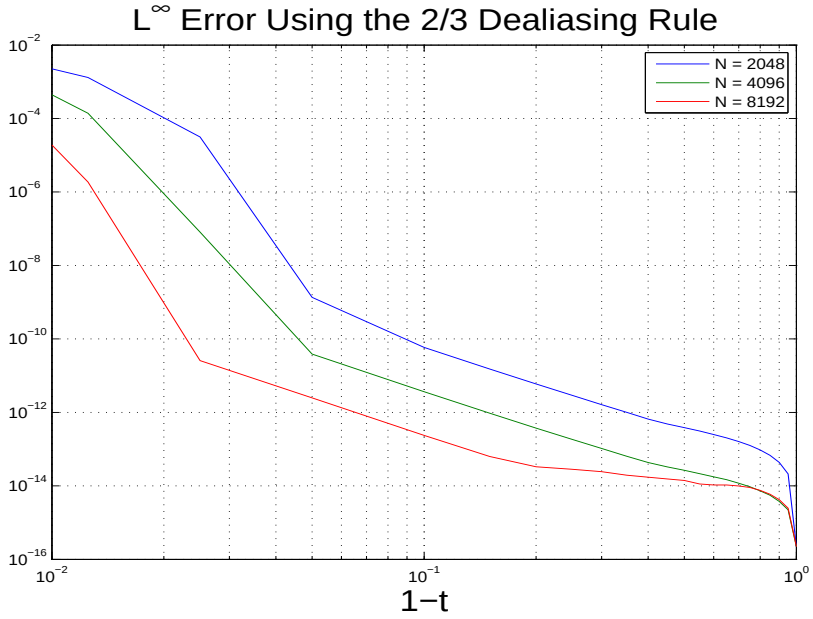

Figure 4.7: Plot of L^∞ error of the two-thirds dealiasing rule.

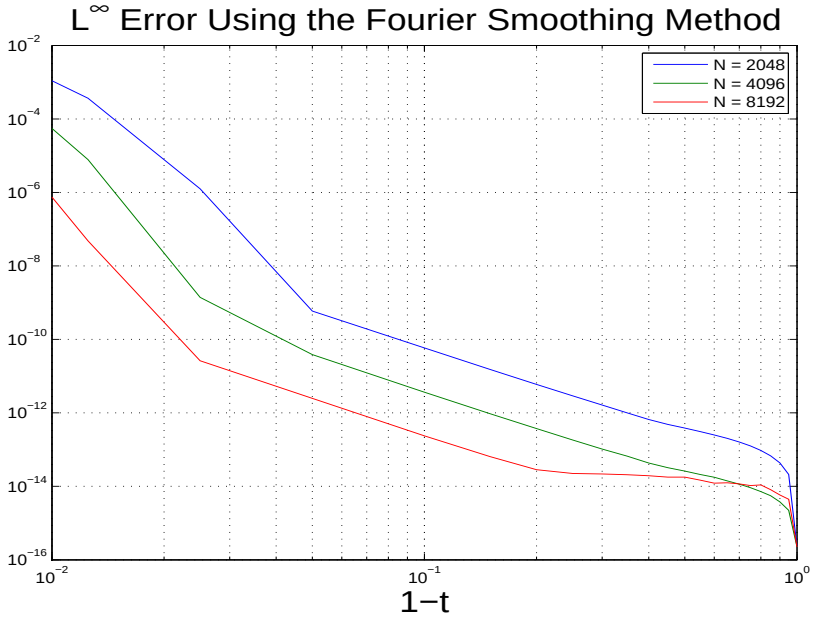

Figure 4.8: Plot of L^∞ error of the Fourier smoothing method.

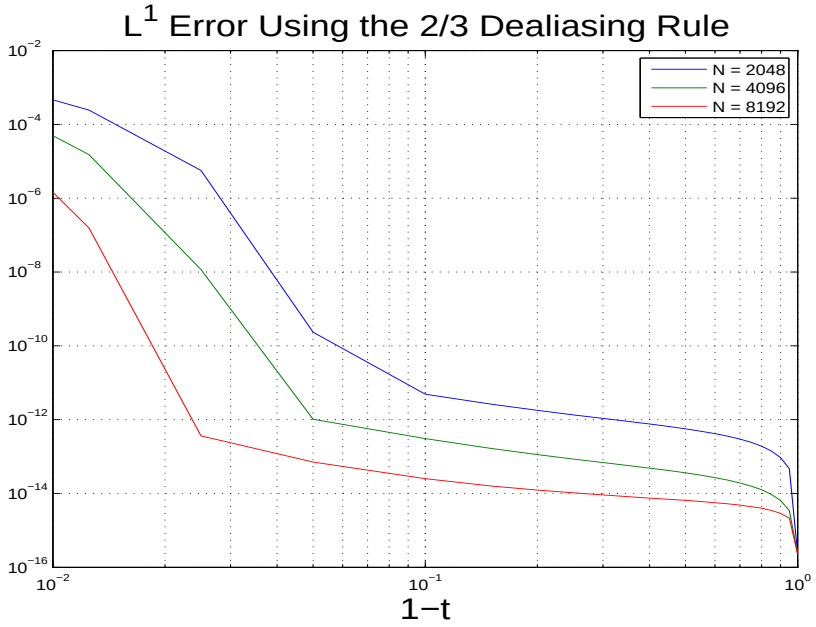

Figure 4.9: Plot of L^1 error of the two-thirds dealiasing rule.

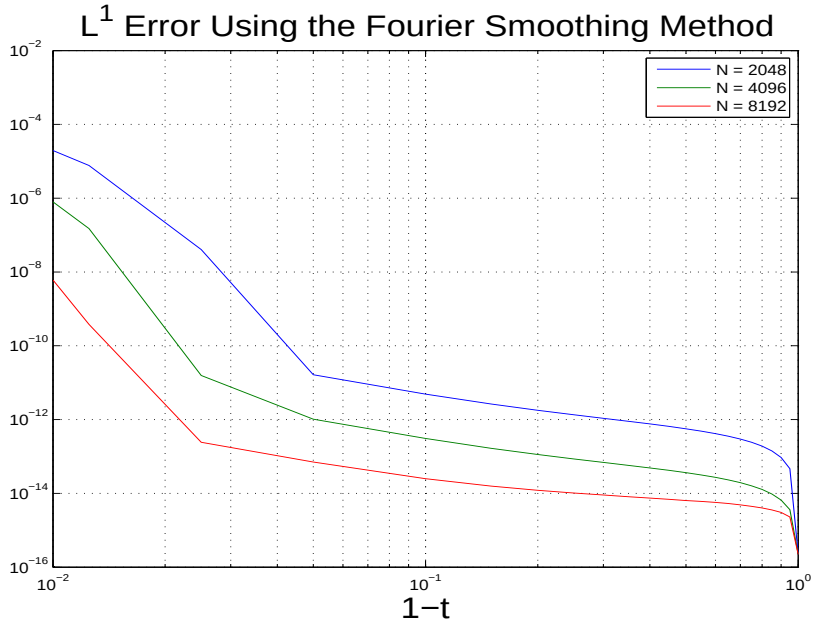

Figure 4.10: Plot of L^1 error of the Fourier smoothing method.

In Figure 4.7 and Figure 4.8, we plot the L^∞ errors for both the two-thirds dealiasing rule and the Fourier smoothing method for several different grid sizes respectively (cf. [12] Fig. 2). In Figure 4.9 and Figure 4.10, we repeat the above for the L^1 errors (cf. [12] Fig. 3). Note the axes. Comparing with the corresponding figures, we see that before the singularity develops ($t \leq .95$), the errors in these figures are smaller than as in [12]. However, as the solution loses regularity and the spatial error begins to dominate near $t = .95$, the errors match that in [12]. In these figures, we have the same conclusions as described in [12]. That is, we can see that near the time of the singularity, the Fourier smoothing method has a smaller error than the two-thirds dealiasing rule for both the L^∞ and L^1 errors. Next, as in [12], we examine the pointwise errors in these methods.

In Figure 4.11 and Figure 4.12, we plot the pointwise errors of the two methods for two different sized grid at $t = 0.9875$ (cf. [12] Fig. 4). Here, we have similar results as in [12]. We see that the error of the Fourier smoothing method is isolated to near the singularity (located at the boundary of the domain), while the two-thirds dealiasing rule has a nearly uniform error across the entirety of the domain. Finally, we examine the difference in the Fourier spectra of the two methods.

In Figure 4.13 and Figure 4.14, we plot the Fourier spectra (absolute value of the Fourier coefficients) using both dealiasing methods as well as the exact solution obtained using Newton's method at several different times for two different sized grids (cf. [12] Fig.5). Here, we see that as the singularity begins to develop, the Fourier spectra increases. We see that the Fourier smoothing method retains almost all of the Fourier coefficients, while the two-thirds dealiasing rule captures significantly fewer. We also see that near $k = 2N/3$, the two-thirds rule has a large spike in its Fourier coefficients caused by the harsh cut-off function, while the Fourier smoothing method does not as the Fourier smoothing method decays the Fourier coefficients gently.

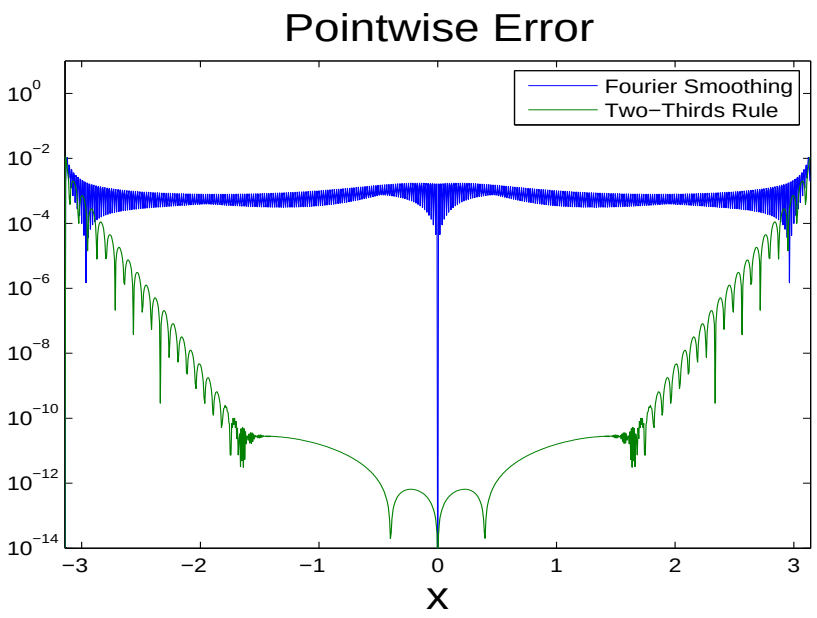

Figure 4.11: Plot of the pointwise errors for the two dealiasing method for $2N = 1024$. Here, the two-thirds dealiasing rule is shown in blue, while the Fourier smoothing method is given in green.

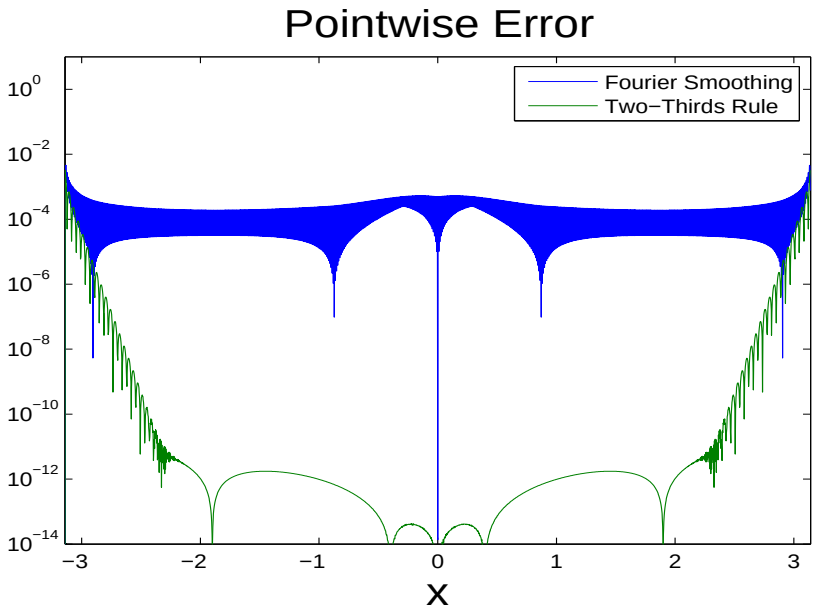

Figure 4.12: Plot of the pointwise errors for the two dealiasing method for $2N = 2048$. Here, the two-thirds dealiasing rule is shown in blue, while the Fourier smoothing method is given in green.

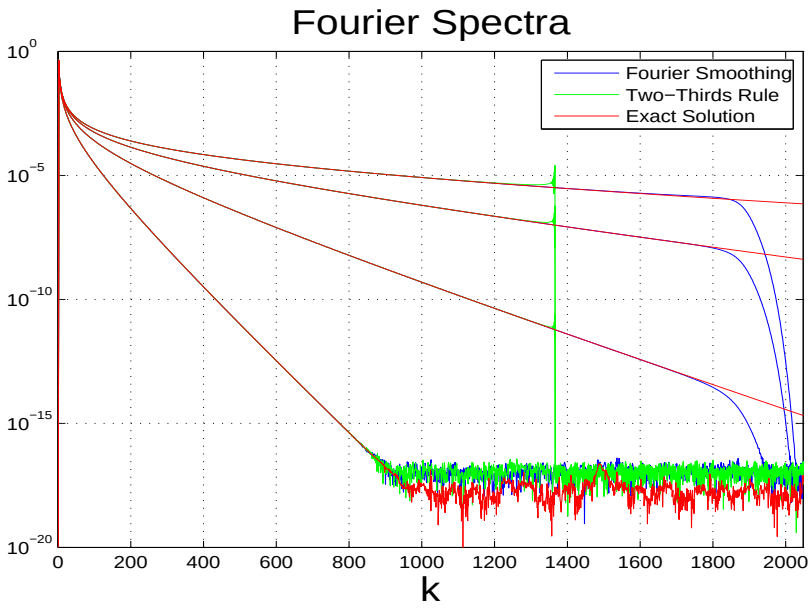

Figure 4.13: Plot of the Fourier spectra of the numerical solutions using both dealiasing methods along with the exact solution for $2N = 4096$. Here, we have plotted the Fourier spectra for times $t = 0.9, 0.95, 0.975, 0.9875$, going from bottom to top.

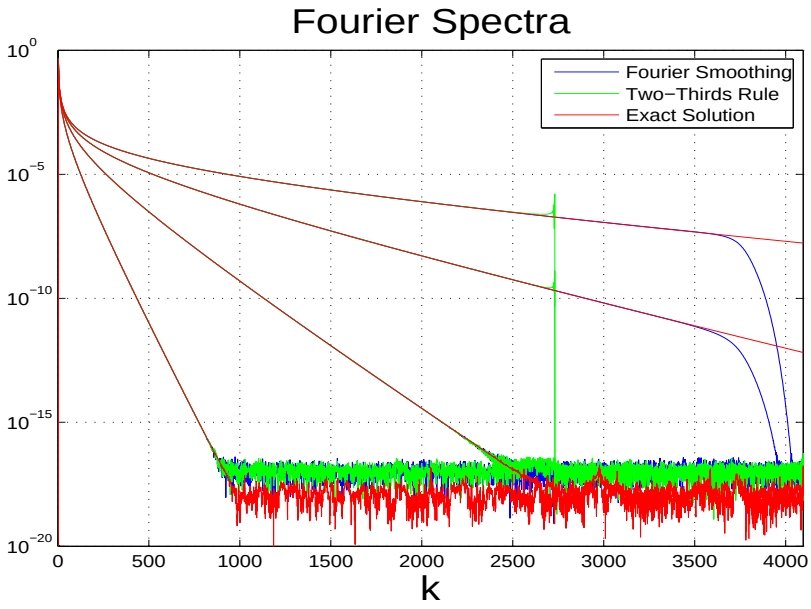

Figure 4.14: Plot of the Fourier spectra of the numerical solutions using both dealiasing methods along with the exact solution for $2N = 8192$. Here, we have plotted the Fourier spectra for times $t = 0.9, 0.95, 0.975, 0.9875$, going from bottom to top.

4.3 Burgers' Equation

We now to examine the last of our one-dimensional models, Burgers' Equation (4.3). As we have done for the other equations, we begin with the exact solution.

4.3.1 Solution of Burgers' Equation

To solve the unforced Burgers' equation (4.3), we will use the Cole-Hopf transform [4,11]. For the Cole-Hopf transform, we make the following substitution of variables:

$$u = -2\nu \frac{v_x}{v}. \tag{4.17}$$

We then have

$$\begin{aligned}
u_t &= -2\nu \frac{v v_{xt} - v_x v_t}{v^2} = -2\nu \left[\frac{v_{xt}}{v} - \frac{v_x v_t}{v^2} \right] \\
u_x &= -2\nu \frac{v v_{xx} - v_x^2}{v^2} = -2\nu \left[\frac{v_{xx}}{v} - \left(\frac{v_x}{v} \right)^2 \right] \\
u_{xx} &= -2\nu \left[\frac{v v_{xxx} - v_x v_{xx}}{v^2} - 2\frac{v_x}{v} \frac{v v_{xx} - v_x^2}{v^2} \right] = -2\nu \left[\frac{v_{xxx}}{v} - 3\frac{v_x v_{xx}}{v^2} + 2\left(\frac{v_x}{v} \right)^3 \right].
\end{aligned}$$

Plugging these into (4.3) with $f = 0$ and dividing by -2ν gives

$$\frac{v_{xt}}{v} - \frac{v_x v_t}{v^2} - \nu \left[\frac{v_{xxx}}{v} - 3\frac{v_x v_x x}{v^2} + 2\left(\frac{v_x}{v} \right)^3 \right] - 2\nu \left[\frac{v_x v_{xx}}{v^2} - \left(\frac{v_x}{v} \right)^3 \right] = 0.$$

Canceling terms, multiplying by v^2, and rearranging gives

$$\nu v_x v_{xx} - v_x v_t + v v_{xt} - \nu v v_{xxx} = 0.$$

Finally, by factoring, we have

$$v_x(\nu v_{xx} - v_t) + v(v_t - \nu v_{xx})_x = 0,$$

which will hold, provided v solves the unforced heat equation (4.1). To solve the heat equation for v, we need an initial condition. Noting that $u(x,0) = u_0(x)$, we may find $v(x,0)$ by solving the ODE

$$u_0(x) = -2\nu \frac{v_x(x,0)}{v(x,0)}.$$

Recognizing that $v_x/v = \log(v)_x$, we may solve this ODE by integrating both sides and exponentiating, giving

$$v(x,0) = \exp\left(-\frac{1}{2\nu}\int u_0(x)\,\mathrm{d}x\right).$$

Once we have $v(x,0)$, we may solve for $v(x,t)$ using (4.9). Finally, u is given by reversing the transform (4.17).

4.3.2 Pseudospectral Methods for Burgers' Equation

We begin by discretizing the functions $u(\cdot,t), u_0$, and $f(\cdot,t)$ with their $2N$ Fourier approximations. This gives

$$\begin{cases} \left(u_{2N}(x,t)\right)_t - \nu\left(u_{2N}\right)_{xx} + \left(\frac{u_{2N}^2}{2}\right)_x = f_{2N}(x,t) \\ u_{2N}(x,t) = \left(u_0(x)\right)_{2N}. \end{cases} \tag{4.18}$$

As we have already discretized the spatial derivatives in (4.18) in Sections 4.1.2 and 4.2.2, these give

$$\begin{cases} \left(\underline{u}_{2N}(t)\right)_t = \nu\mathcal{F}_{2N}^{-1}\left[-\underline{k}^2 \circ \mathcal{F}_{2N}(\underline{u}_{2N})\right] - \mathcal{F}_{2N}^{-1}\left[\frac{i}{2}\underline{k} \circ \rho(\underline{k}/N)\mathcal{F}_{2N}[\underline{u}_{2N} \circ \underline{u}_{2N}]\right] + \underline{f}_{2N}(t) \\ \underline{u}_{2N}(t) = \underline{u}_0, \end{cases} \tag{4.19}$$

which is of the form of (3.2).

We then may solve (4.19) by the time integrators from Section 3.1.

4.3.3 Simulation of Burgers' Equation

Using details from Section 4.3.1, we may construct a solution to Burgers' equation by picking a $2\pi-$periodic $v(x,t)$ that solves the heat equation. In this section, we consider $v(x,t) = \cos(x)e^{-\nu t} + c$, for some constant c. Therefore, substituting $v(x,t)$ into the Cole-Hopf transform (4.17), we choose the exact solution of Burgers' equation

$$u(x,t) = 2\nu\frac{\sin(x)e^{-\nu t}}{\cos(x)e^{-\nu t} + c}, \tag{4.20}$$

where we choose $c = \sqrt{1+4\nu^2}$, so that the exact solution remains bounded, but is less regular for small ν.

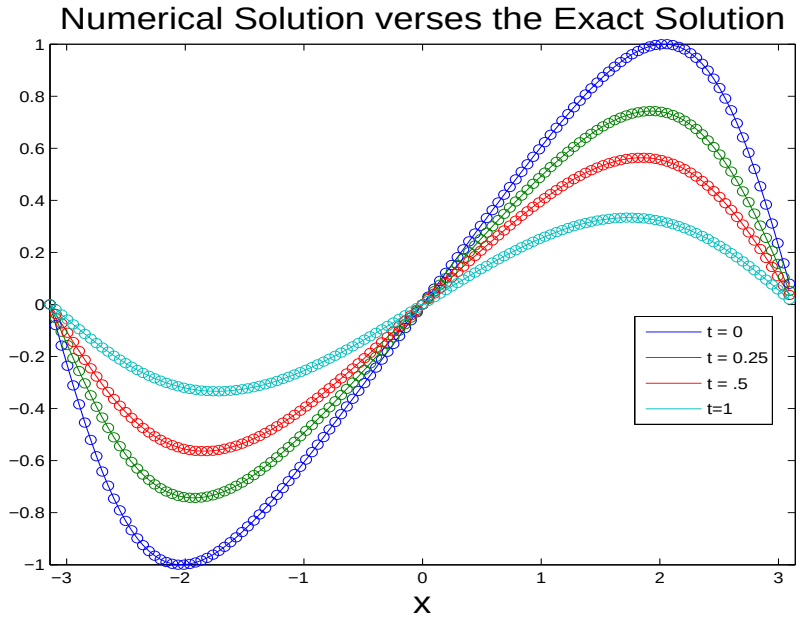

Figure 4.15: Plot of the numerical and exact solution at several different times for $\nu = 1$. The exact solutions are the solid lines while the numerical solutions are circles

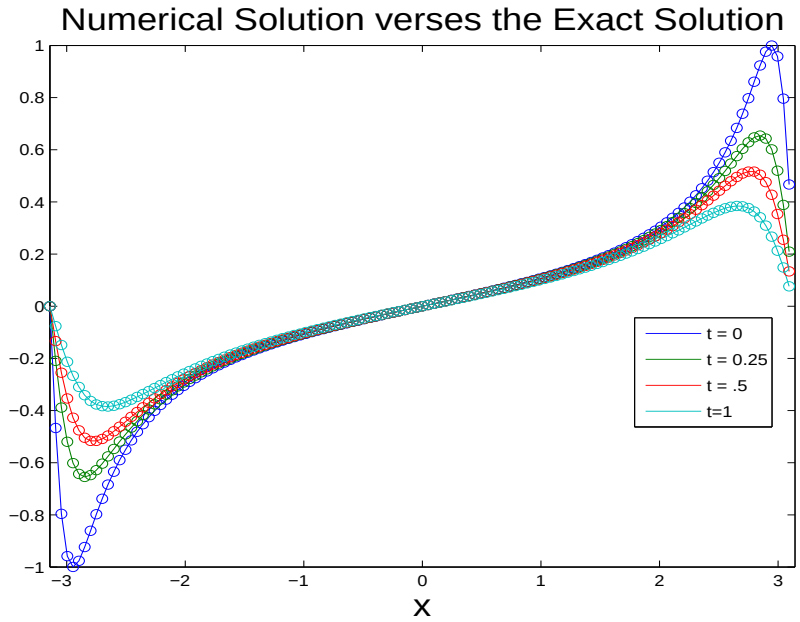

Figure 4.16: Plot of the numerical and exact solution at several different times for $\nu = 1/10$. The exact solutions are the solid lines while the numerical solutions are circles

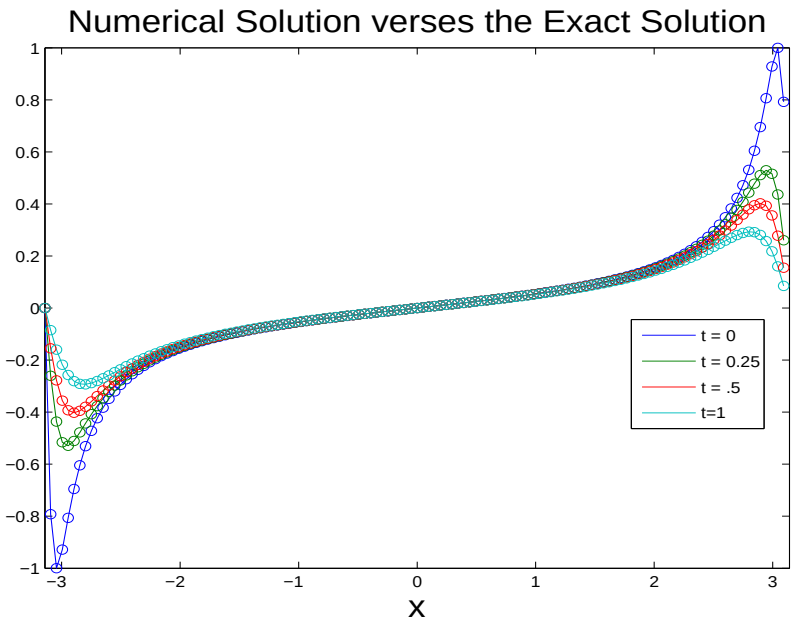

Figure 4.17: Plot of the numerical and exact solution at several different time for $\nu = 1/20$. The exact solutions are the solid lines while the numerical solutions are circles

For this example, we will use a grid of 512 points and a time step of $\Delta t = 1/100$ using the trapezoidal method. As these solutions are nearly singular, we use the Fourier smoothing method dealiasing technique.

In Figure 4.15, we plot the numerical solution verses the exact solution for $\nu = 1$ at times $t = 0, 0.25, 0.5, 1$. In Figure 4.16 and Figure 4.17, we repeat the above for $\nu = 1/10$ and $\nu = 1/20$ respectively. We see that as ν decreases, the solution becomes more singular, however, we are still able to get good results.

CHAPTER 5

PSEUDOSPECTRAL METHODS FOR NONLINEAR TWO-DIMENSIONAL EVOLUTIONARY SYSTEMS

In many applications, physical processes modeled by PDEs depend on more than one spatial variable. In this chapter, we discuss how to simulate the solutions of evolutionary systems in two spatial dimensions. To this end, we begin this chapter by generalizing spatial discretization techniques discussed in Chapter 2 to higher dimensions. We then introduce two model problems in fluid dynamics: the incompressible Navier-Stokes and Euler equations. We then manipulate these equations into vorticity forms and then apply the pseudospectral method of lines. We then finish this chapter by demonstrating pseudospectral methods for solving these systems with several examples.

5.1 Spatial Discretization in Two Dimensions

In this section, we generalize the spatial discretization discussed in Chapter 2. To this end, consider a typical two-dimensional periodic model function, $u : \Omega^2 \to \mathbb{R}$, where $\Omega^2 = (-\pi, \pi) \times (-\pi, \pi)$. The Fourier series of u is then given by

$$u(x, y) = \sum_{k=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \hat{u}_{\underline{k}} \phi_k(x) \phi_j(y), \tag{5.1}$$

where $\underline{k} = [k, j]^T$ and the Fourier coefficients are given by

$$\hat{u}_{\underline{k}} = \frac{1}{4\pi^2} \iint_{\Omega^2} u(x, y) \phi_k(x) \phi_j(y) \, \mathrm{d}x \, \mathrm{d}y. \tag{5.2}$$

Using the Fourier series in two dimensions we may compute derivatives with respect to x and y easily. For example, we may compute $u_x(x, y)$ with

$$u_x(x, y) = \sum_{k=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (ik) \hat{u}_{\underline{k}} \phi_k(x) \phi_j(y),$$

and similarly, compute $u_y(x, y)$ with

$$u_y(x,y) = \sum_{k=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (ij)\hat{u}_{\underline{k}}\phi_k(x)\phi_j(y).$$

In Chapter 2, we truncated the Fourier series to get an approximation of u, u_N, that is an element of a finite dimensional Hilbert space V_N. In higher dimensions, we truncate both sums in (5.1) to get an approximation. That is, we have
$u(x,y) \approx u_{NM}(x,y)$, with

$$u_{NM} = \sum_{k=-N}^{N-1} \sum_{j=-M}^{M-1} \hat{u}_{\underline{k}}\phi_k(x)\phi_j(y), \tag{5.3}$$

where $N, M \in \mathbb{N}$. Thus, $u_{NM} \in V_{2N} \otimes V_{2M}$, the *tensor product* of the spaces V_{2N} and V_{2M}.

To compute the Fourier coefficients in (5.2), we approximate using the DFT described in Chapter 2. To do this, we note that (5.2) may be rewritten as

$$\hat{u}_{\underline{k}} = \frac{1}{2\pi} \int_\Omega \underbrace{\left[\frac{1}{2\pi} \int_\Omega u(x,y)\phi_k(x)\,\mathrm{d}x\right]}_{\tilde{u}_k(y)} \phi_j(y)\,\mathrm{d}y.$$

We can see that the term $\tilde{u}_k(y)$ is the Fourier transform of $u(x,y)$ treating y as fixed. $\hat{u}_{\underline{k}}$ is then computed by taking the Fourier transform of $\tilde{u}_k(y)$ with respect to y for each k. Thus, we may approximate $\hat{u}_{\underline{k}}$ by approximating these Fourier transforms with the discrete Fourier transform. Therefore, we approximate $\hat{u}_{\underline{k}}$ with

$$\hat{u}_{\underline{k}} \approx \frac{1}{4NM} \sum_{n=-N}^{N-1} \sum_{m=-M}^{M-1} u(x_n, y_m) e^{-ik\frac{\pi}{N}n - ij\frac{\pi}{M}m}. \tag{5.4}$$

5.2 The Vorticity Equation

We now study the incompressible 2-D Navier-Stokes equations (NSE). Letting $n = 2$ in equations (1.1)-(1.2), we have

$$\underline{u}_t + (\underline{u}\cdot\nabla)\underline{u} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\underline{u} + \underline{f} \tag{5.5}$$

$$\nabla\cdot\underline{u} = 0. \tag{5.6}$$

When $\nu = 0$, we refer to these equations as the incompressible Euler equations. We now manipulate equations (5.5)-(5.6) into vorticity form. Here, we treat $\underline{u}$ and $\underline{f}$ as a vectors with three components: $\underline{u} = [u, v, 0]^T$ and $\underline{f} = [f_1, f_2, 0]^T$.

Recall that the curl is the annihilator of the gradient, that is, for a sufficiently smooth scalar function f, we have

$$\nabla \times \nabla f = \nabla \times [f_x, f_y, f_z] = [f_{zy} - f_{yz}, f_{xz} - f_{zx}, f_{yx} - f_{xy}] = \underline{0}. \tag{5.7}$$

Thus if we take the curl of (5.5), we may obtain a model without a pressure term. Using the identity $(\underline{u} \cdot \nabla)\underline{u} = \nabla\left(\frac{\|\underline{u}\|^2}{2}\right) + (\nabla \times \underline{u}) \times \underline{u}$, (5.5) becomes

$$\underline{u}_t + \nabla\left(\frac{\|\underline{u}\|^2}{2}\right) + (\nabla \times \underline{u}) \times \underline{u} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\underline{u} + \underline{f}.$$

Now, by taking the curl and using (5.7), we have

$$\nabla \times \underline{u}_t + \nabla \times ((\nabla \times \underline{u}) \times \underline{u}) = \nu\nabla \times (\nabla^2\underline{u}) + \nabla \times \underline{f}.$$

By letting $\underline{\omega} = \nabla \times \underline{u} = [0, 0, v_x - u_y]^T = [0, 0, \omega]^T$ be the vorticity and $\nabla \times \underline{f} = \underline{g} = [0, 0, g]^T$, we have

$$\underline{\omega}_t + \nabla \times (\underline{\omega} \times \underline{u}) = \nu\nabla^2\underline{\omega} + \underline{g}.$$

By using the identity $\nabla \times (\underline{a} \times \underline{b}) = \underline{a}(\nabla \cdot \underline{b}) - \underline{b}(\nabla \cdot \underline{a}) + (\underline{b} \cdot \nabla)\underline{a} - (\underline{a} \cdot \nabla)\underline{b}$, we have

$$\underline{\omega}_t + \nabla \times (\underline{\omega} \times \underline{u}) = \underline{\omega}_t + \underline{\omega}(\nabla \cdot \underline{u}) - \underline{u}(\nabla \cdot \underline{\omega}) + (\underline{u} \cdot \nabla)\underline{\omega} - (\underline{\omega} \cdot \nabla)\underline{u} = \nu\nabla^2\underline{\omega} + \underline{g}.$$

Using (5.6), $\nabla \cdot \omega = \nabla \cdot \nabla \times \underline{u} = 0$, and $(\underline{\omega} \cdot \nabla)\underline{u} = [0 * u_x, 0 * v_y, \omega * 0_z]^T = \underline{0}$, this equation reduces to the scalar PDE

$$\begin{cases} \omega_t + u\omega_x + v\omega_y = \nu\nabla^2\omega + g, \\ \omega(x, y, 0) = \omega_0(x, y) \\ \text{Periodic boundary conditions,} \end{cases} \tag{5.8}$$

where now $\underline{u} = \nabla \times \underline{\omega}$ is treated as a function of ω and g is divergent free. We refer to (5.8) as the Navier-Stokes vorticity equation when $\nu > 0$ and as the Euler vorticity equation for $\nu = 0$.

This derivation leaves us with some questions. The first is how to find $\underline{u}$. The second is whether the $\underline{u}$ we find is unique. As $\underline{\omega}$ is defined with a differential operator of $\underline{u}$, $\underline{u}$ may be defined with an integral operator of $\underline{u}$ and therefore may not be unique. Finally, since we have eliminated (5.6), we should question whether the $\underline{u}$ that we compute using $\underline{\omega}$ is divergent free. We now answer these questions.

We begin by considering the stream function $\underline{\psi} = [0, 0, \psi]^T$, which we define as

$$\psi_x = -v, \qquad \psi_y = u. \tag{5.9}$$

Then we have

$$\underline{\omega} = \nabla \times \underline{u} = v_x - u_y = -\psi_{xx} - \psi_{yy} = -\nabla^2\psi.$$

So, we may compute ψ by solving Poisson's equation

$$-\nabla^2\psi = \omega. \tag{5.10}$$

We shall now explore the uniqueness of (5.10). Consider ψ_1 and ψ_2, two solutions of (5.10). Then we have

$$-\nabla^2\psi_1 = \omega \qquad -\nabla^2\psi_2 = \omega.$$

Subtracting the two gives

$$\nabla^2(\underbrace{\psi_1 - \psi_2}_{h}) = 0.$$

So, (5.10) is unique up to a harmonic function – a solution of Laplace's equation

$$\nabla^2 h = 0. \tag{5.11}$$

Now, since we are using periodic boundary conditions, we consider the solutions to (5.11) with periodic boundary conditions. Let $h(x, y)$ be a solution to (5.11) on Ω^2 with periodic boundary conditions and let

$$h(x, y) = \sum_{k=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \hat{h}_{\underline{k}} \phi_k(x) \phi_j(y)$$

be the Fourier series of h. Then we have

$$h_{xx} = \sum_{k=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} -k^2 \hat{h}_{\underline{k}} \phi_k(x)\phi_j(y), \qquad h_{yy} = \sum_{k=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} -j^2 \hat{h}_{\underline{k}} \phi_k(x)\phi_j(y).$$

Substituting this into (5.11) we have

$$\nabla^2 h = \sum_{k=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (-k^2 - j^2) \hat{h}_{\underline{k}} \phi_k(x)\phi_j(y) = 0.$$

By orthogonality, we have

$$-(j^2 + k^2) h_{\underline{k}} \phi_k(x)\phi_j(y) = 0 \text{ for all } j, k \in \mathbb{Z}.$$

Since $\phi_k(x) \neq 0$ for any x, k and similarly for $\phi_j(y)$, we have that if $j^2 + k^2 \neq 0$, then $h_{\underline{k}} = 0$, yielding

$$h(x, y) = h_{\underline{0}},$$

a constant. So, we have that the solution of (5.11) is a constant and thus the solution of (5.10) is unique up to a constant. Therefore, using (5.9), we may still recover $\underline{u}$ uniquely. Finally, we show that (5.6) is satisfied using $\underline{u}$ as found above

$$\nabla \cdot \underline{u} = u_x + v_y = \psi_{yx} - \psi_{xy} = 0,$$

as desired.

5.3 Simulation of the Euler Vorticity Equation

In this section, we demonstrate pseudospectral methods for the numerical solution of the Euler vorticity equation (5.8) ($\nu = 0$). In our first simulation, we pick ω in order to examine the convergence of the pseudospectral methods. To this end, we choose the vorticity

$$\omega(x, y, t) = \sin(x - 2t) + \sin(y - 3t).$$

Differentiating with respect to time gives

$$\omega_t(x, y, t) = -2\cos(x - 2t) - 3\cos(y - 3t).$$

Similarly, differentiating the vorticity with respect to the spatial variables gives

$$\omega_x(x, y, t) = \cos(x - 2t)$$
$$\omega_y(x, y, t) = \cos(y - 3t).$$

By solving Poisson's equation (5.10), we may get the stream function

$$\psi(x, y, t) = \sin(x - 2t) + \sin(y - 3t).$$

Differentiating the stream function with respect to the spatial variables gives

$$u(x, y, t) = \cos(y - 3t)$$
$$v(x, y, t) = -\cos(x - 2t).$$

Finally, by plugging the above into (5.8) with $\nu = 0$, we have

$$g(x, y, t) = -2\sin(x - 2t) - 3\sin(y - 3t).$$

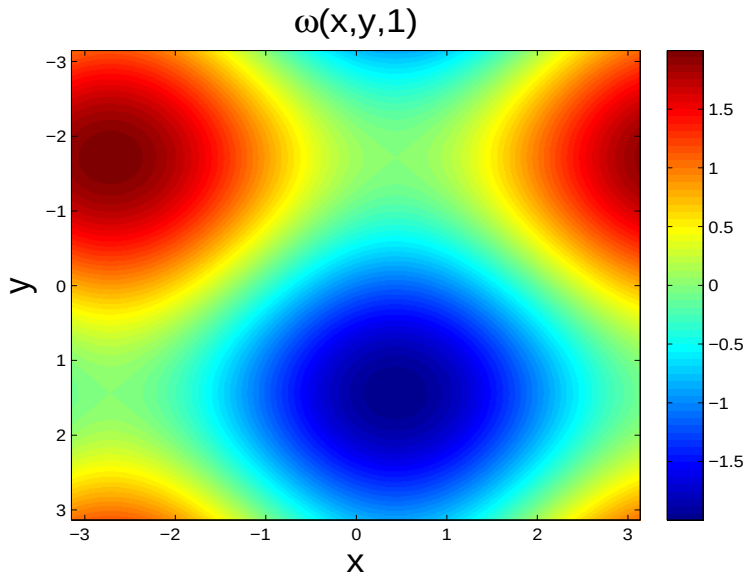

Figure 5.1: Plot of exact solution of the Euler vorticity at $t = 1$.

In Figure 5.1 we have plotted the exact vorticity at $t = 1$. In Figure 5.2, we have plotted the numerical solution of the vorticity at $t = 1$. Here, we have used a spatial grid of $N = M = 256$. For time stepping, we have used the Runge-Kutta method with a time step

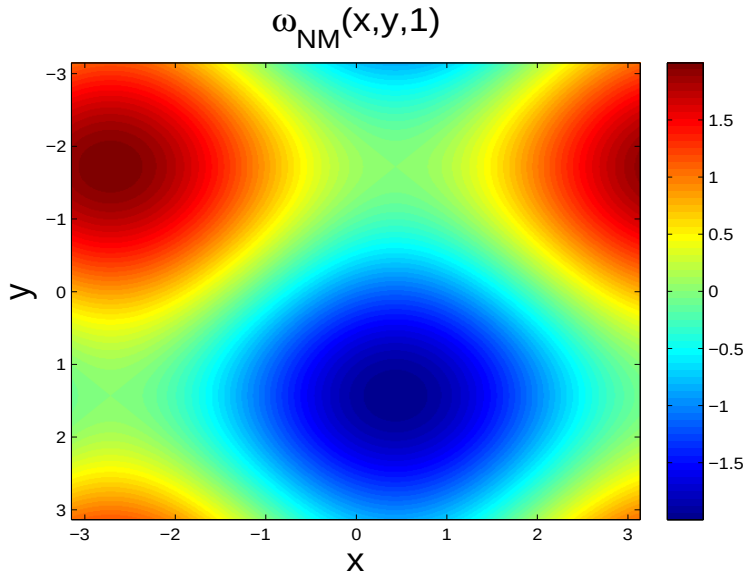

Figure 5.2: Plot of numerical solution of the Euler vorticity equation at $t = 1$ with $N = M = 256$, using the fourth oder Runge-Kutta method with $\Delta t = 1/256$.

$\Delta t = 1/256$. For a dealiasing technique, we have used the Fourier smoothing method. We can see visually that the two appear to be the same. We now verify the rate of convergence by computing the error of the method in time.

Table 5.1: Estimated Order of Convergence using the Runge-Kutta method

Δt	$u^{\Delta t}_{2N,2M}(\pi/2,\pi/2,1)$	$EOC_k(\pi/2,\pi/2,1)$	$EOC_u(\pi/2,\pi/2,1)$
1/2	−1.4104	4.0698	4.0732
1/4	−1.4064	4.0170	4.0179
1/8	−1.4062	4.0042	4.0044
1/16	−1.4061	4.0011	4.0011
1/32	−1.4061	4.0003	4.0003
1/64	−1.4061	4.0001	4.0001
1/128	−1.4061	4.0000	–
1/256	−1.4061	–	–

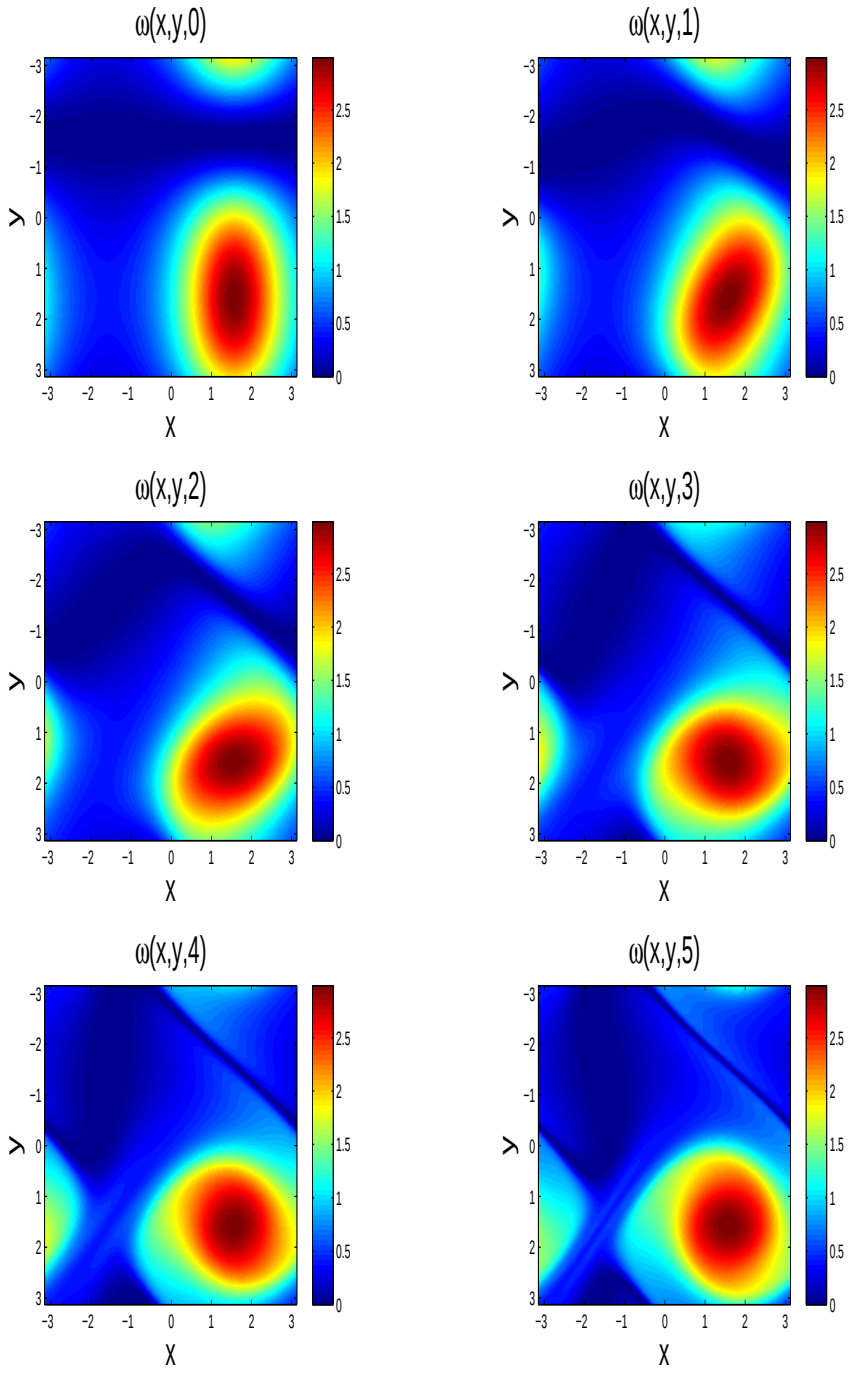

Figure 5.3: Plot of numerical solution of the Euler vorticity with $N = M = 128$ at $t = 0, 1, 2, 3, 4, 5$ using the Runge-Kutta 4 method and $\Delta t = 1/100$.

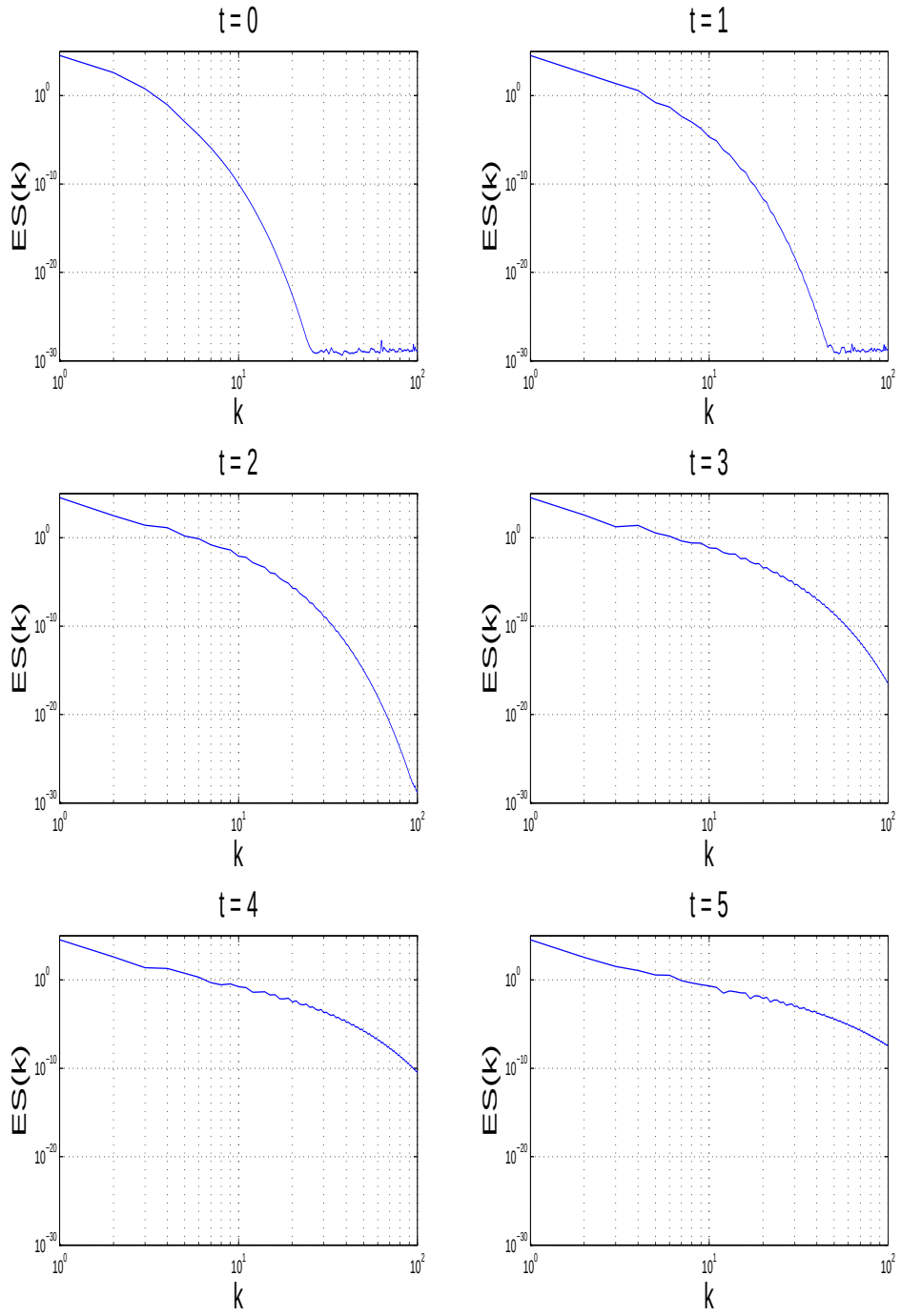

Figure 5.4: Plot of the energy spectrum of the numerical solution of the Euler vorticity with $N = M = 128$ at $t = 0, 1, 2, 3, 4, 5$ using the Runge-Kutta 4 method and $\Delta t = 1/100$.

In Table 5.1, we give the numerical solution computed at the point $(\pi/2, \pi/2)$ at the final time $T = 1$ with $N = M = 16$ using the Runge-Kutta method and the same equations as the previous example. We have also given the estimated orders of convergence for both when the exact solution is known and when it is unknown. We can see that they tend to 4, confirming validity of the code.

We now simulate the Euler vorticity equations for when the exact solution is unknown. To this end, we consider initial vorticity

$$\omega_0(x, y) = e^{\sin(x)} \log(2 + \sin(y)).$$

Using this initial condition, we solve the unforced Euler vorticity equation.

In Figure 5.3 we plot the vorticity of the numerical solution obtained with $N = M = 64$ using the Runge-Kutta 4 method with $\Delta t = 1/100$. Here, we have also used the Fourier smoothing method for dealiasing. We can see that as the time increases, the vorticity becomes less regular. To see how the solution loses regularity, in Figure 5.4, we plot the energy spectrum of the velocity field. We define the energy spectrum of the velocity field as

$$ES(k) = \sum_{k-\frac{1}{2} \le |\underline{k}| < k+\frac{1}{2}} |\hat{u}_{\underline{k}}|^2. \tag{5.12}$$

We see that as the solution loses regularity, the energy spectrum decays less slowly.

5.4 Simulation of the Navier-Stokes Vorticity Equation

In this section, demonstrate the pseudospectral method for the numerical solution of the Navier-Stokes vorticity equation (5.8) with $\nu > 0$. In this simulation, we pick ω in order to examine the convergence of the pseudospectral method. To this end, we choose the vorticity

$$\omega(x, y, t) = [\sin(x - 2t) + \sin(y - 3t)]e^{-\nu t}.$$

Differentiating with respect to time gives

$$\omega_t(x, y, t) = [-\nu \sin(x - 2t) - 2\cos(x - 2t) - \nu \sin(y - 3t) - 3\cos(y - 3t)]e^{-\nu t}.$$

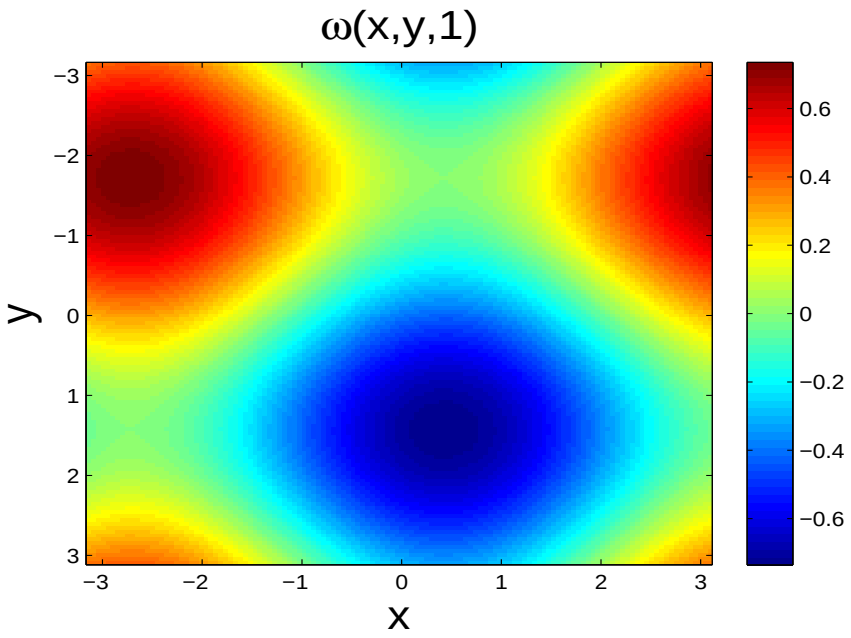

Figure 5.5: Plot of exact solution of the Navier-Stokes vorticity at $t = 1$.

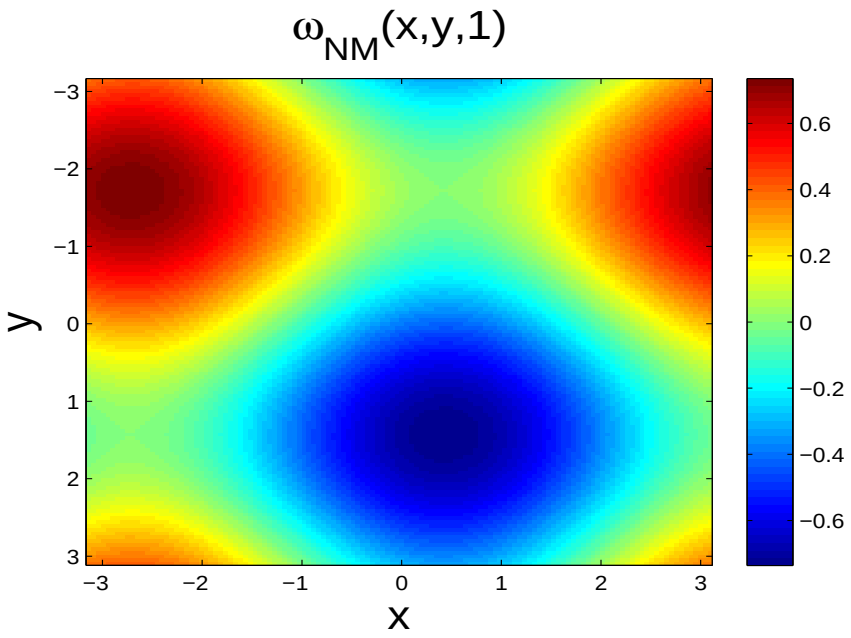

Figure 5.6: Plot of numerical solution of the Navier-Stokes vorticity equation at $t = 1$ with $N = M = 256$, using the Runge-Kutta method with $\Delta t = 1/256$.

Similarly, differentiating the vorticity with respect to the spatial variables gives

$$\omega_x(x,y,t) = \cos(x-2t)e^{-\nu t}, \qquad \omega_y(x,y,t) = \cos(y-3t)e^{-\nu t}.$$

By solving Poisson's equation (5.10), we may get the stream function

$$\psi(x,y,t) = [\sin(x-2t) + \sin(y-3t)]e^{-\nu t}.$$

Differentiating the stream function with respect to the spatial variables gives

$$u(x,y,t) = \cos(y-3t)e^{-\nu t}, \qquad v(x,y,t) = -\cos(x-2t)e^{-\nu t},$$

where we can see that the velocity field is divergent free. By taking the Laplacian of the vorticity and multiplying by ν, we have

$$\nu\nabla^2(x,y,t)\omega = -\nu[\sin(x-2t) - \sin(y-3t)]e^{-\nu t}.$$

Finally, by plugging the above into (5.8), we have

$$f(x,y,t) = [-2\sin(x-2t) - 3\sin(y-3t)]e^{-\nu t}.$$

In Figure 5.5 we have plotted the exact vorticity at $t = 1$. In Figure 5.6, we have plotted the numerical solution of the vorticity at $t = 1$. Here, we have used a spatial grid of $N = M = 64$. For time stepping, we have used the trapezoidal method with a time step $\Delta t = 1/2000$. For dealiasing. we have used the Fourier smoothing method. We can see visually that the two appear to be the same.

5.5 Long-term Behavior of the 2-D Incompressible Navier-Stokes Equations

In our last 2-dimensional simulation, we would like to study the long term behavior of the 2-dimensional incompressible Navier-Stokes equations. To this end, we consider a numerical experiment suggested in [6]. In [6], the model problem begins with the initial condition of the vorticity in terms of its Fourier coefficients with

$$(\hat{\omega}_0)_{\underline{k}} = |\,(\hat{\omega}_0)_{\underline{k}}\,|e^{i\theta_{\underline{k}}}, \tag{5.13}$$

where $\theta_{\underline{k}}$ is generated randomly on $[0, 2\pi]$ with a uniform distribution and

$$(\hat{\omega}_0)_{\underline{k}} = \begin{cases} \frac{c}{\left(|\underline{k}|+\left(\sqrt{\nu}|\underline{k}|\right)^5\right)^{\frac{1}{2}}} & \text{if } |\underline{k}| \leq k_\alpha \\ 0 & \text{otherwise,} \end{cases}$$

where c is such that $|\omega_0|_\infty = 2$. In [6], $k_\alpha = 60$ and $\nu = 10^{-3}$.

The forcing function in [6] is defined $\underline{f}$ by

$$\hat{\underline{f}} = \left[\left(\hat{f}_1\right)_{\underline{k}}, \left(\hat{f}_2\right)_{\underline{k}}, 0\right]^T, \tag{5.14}$$

where

$$\left|\left(\hat{f}_n\right)_{\underline{k}}\right| = \begin{cases} c & \text{if } |\underline{k}| = 3 \\ 0 & \text{otherwise,} \end{cases}$$

for $n = 1, 2$ and where c is chosen so that $|\underline{f}|_2 = 0.225$. The Fourier coefficients are then given by

$$\left(\hat{f}_n\right)_{\underline{k}} = \left|\left(\hat{f}_n\right)_{\underline{k}}\right| e^{i\theta_{\underline{k}}},$$

where, $\theta_{\underline{k}}$ are also chosen randomly on $[0, 2\pi]$ with a uniform distribution. $\underline{g}$ is then given by $\underline{g} = \nabla \times \underline{f}$.

We simulate this problem using the pseudospectral method. The grid size is chosen with $N = M = 256$. The time integration method used is the trapezoidal method with a time step $\Delta t = 10^{-3}$ and a final time of $T = 100$. This is the same time step and final time used in [6]. However, in [6], a third order method was used in time. We now present the results of the simulations.

In Figure 5.7, we plot the vorticity profile of the numerical solution of the Navier-Stokes vorticity equation with $N = M = 256$ using the trapezoidal method and $\Delta t = 10^{-3}$. In the top left, we show the initial condition, where we note that there is no apparent structure. In the top right, we plot the vorticity at $t = 5$, where we see that vorticity has gained some structure. In the lower four plots, we show the vorticity at $t = 30, 50, 85$, and 100. We see that the solution does not approach a steady solution.

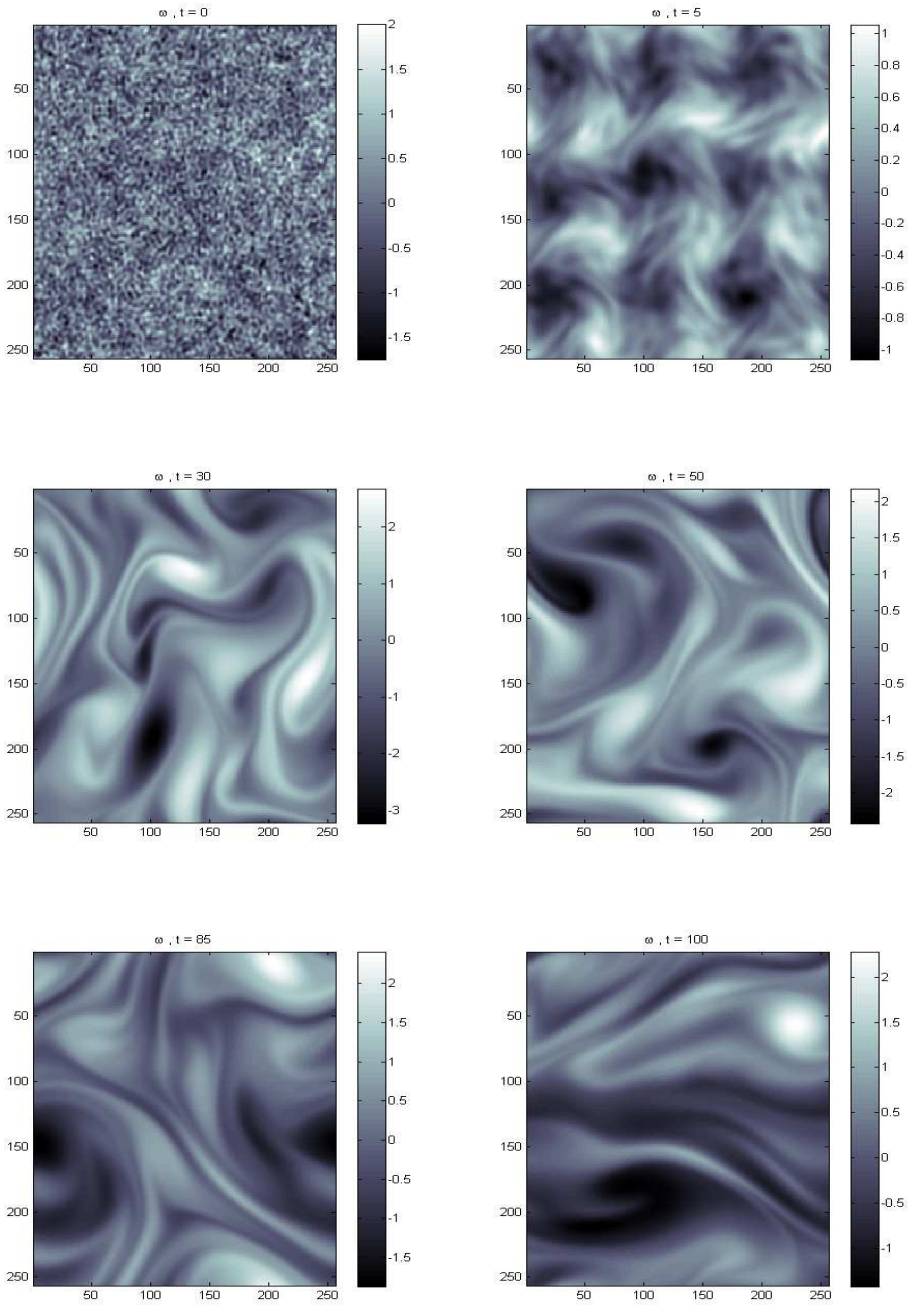

Figure 5.7: Plot of vorticity profile of the numerical solution of the Navier-Stokes vorticity equation with $N = M = 256$ at $t = 0, 5, 30, 50, 85, 100$ using the trapezoidal method and $\Delta t = 10^{-3}$.

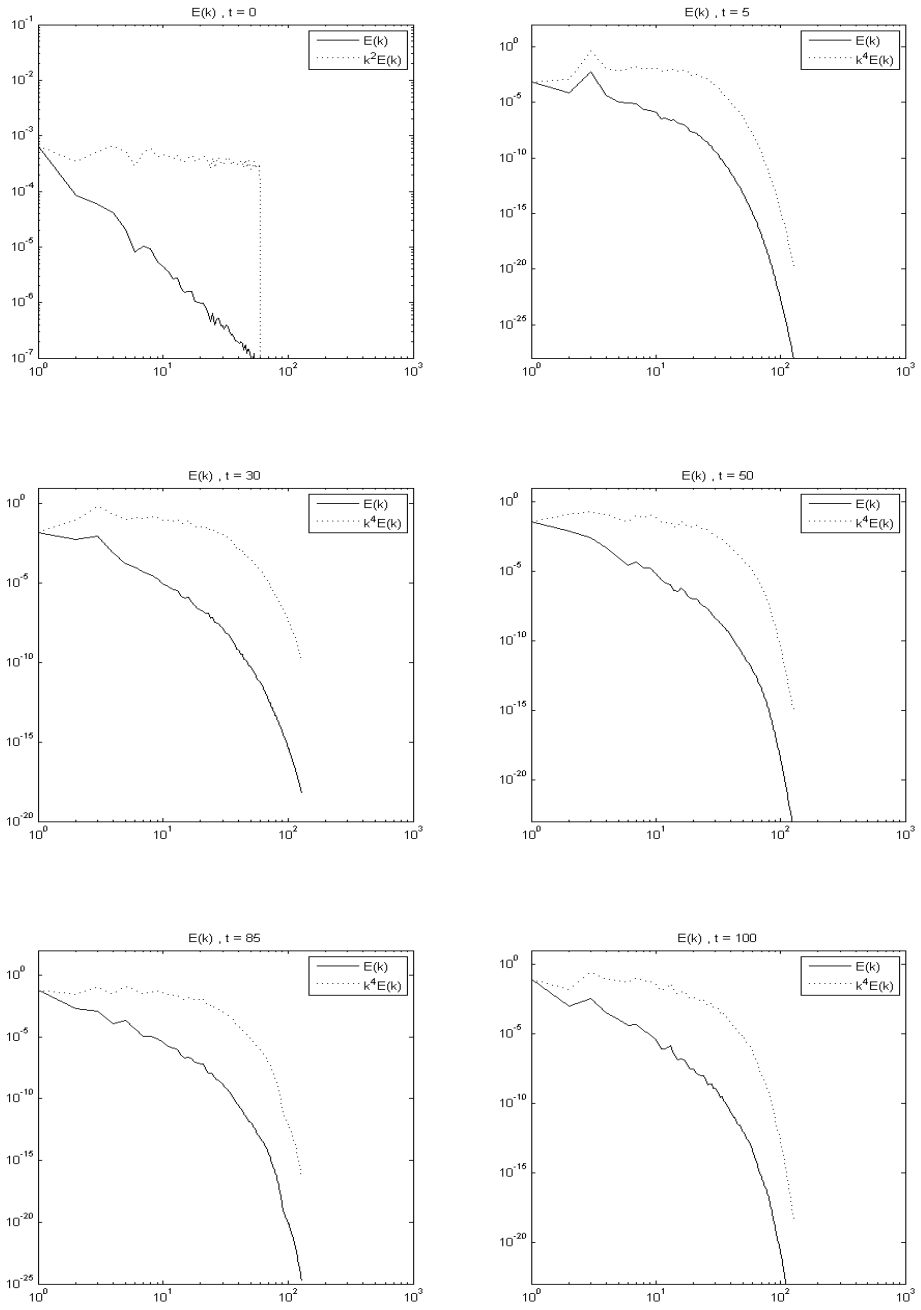

Figure 5.8: Plot of the energy spectrum of the numerical solution of the Navier-Stokes vorticity equation with $N = M = 256$ at $t = 0, 5, 30, 50, 85, 100$ using the trapezoidal method and $\Delta t = 10^{-3}$.

In Figure 5.8 we plot the energy spectrum of the numerical solution. We see that the initial energy spectrum decays like k^{-2}. The solution at the other times has an energy spectrum that decays like k^{-4}, showing that the solution gains some regularity as time progresses.

In this chapter, we generalized the techniques discussed in previous chapters to higher dimensional problems. This was done by considering a solution space that is the tensor product of spaces used to solve one-dimensional problems. We then considered two model evolutionary systems: the incompressible Navier-Stokes equations and the Euler equations, which we manipulated into scalar vorticity forms. The Fourier pseudospectral method in higher dimensions was then demonstrated in a series of examples. We finish this thesis with a summary and discussion of possible future work that may follow the research contained in this thesis.

CHAPTER 6
CONCLUSIONS AND FUTURE RESEARCH

In this thesis, we discussed how to approximate the solution of evolutionary systems using Galerkin spectral methods with quadrature, which are also known as pseudospectral methods. To use a pseudospectral method to approximate solutions of these PDEs, we used the method of lines by first discretizing the problem in space, then by discretizing in time. To approximate our system in space, we approximated unknown functions with truncated Fourier series. The Fourier coefficients of these series, which are defined as integrals, could not be evaluated exactly and we instead approximated these integrals using the trapezoidal rule. Once functions were approximated in the form of truncated Fourier series, we were able to differentiate our functions in space. By plugging these approximations back into the original system, we were left with a system of ODEs which depended only on time. To approximate the solution of these systems of ODEs, we used standard numerical methods.

We then demonstrated this technique on several evolutionary equations which depended on only one spatial dimension. In these examples, we also considered nonlinear systems and had to account for aliasing that would occur. To remedy the affects of aliasing, we used two separate techniques, which we analyzed fully in an example problem. The techniques for solving evolutionary systems in a single spatial dimension was then generalized into higher dimensions by using tensor product basis. We then demonstrated pseudospectral methods in more than one spatial dimension using both the viscid and inviscid vorticity equation.

This thesis only scratches the surface of solving evolutionary systems with pseudospectral methods. This being the case, this research can be continued in many different directions. This thesis only discusses solving systems in only one and two spatial dimensions, however, many physical problems exist in three spatial dimensions. To this end, we may continue this research by solving problems in higher spatial dimensions such as the Navier-Stokes

equations in three spatial dimensions. The evolutionary systems we discussed in this thesis all used periodic boundary conditions, and while many physical systems occur in a periodic manner, not all physical systems do. Thus, this research may be continued by considering evolutionary systems that have spatial boundary conditions that are not periodic such as Dirichlet or Neumann boundary conditions. Many of the PDE's discussed in this thesis depended on a parameter, such as the kinematic viscosity in the vorticity equation. In many applications, the evolutionary systems must be solved for a large number of different parameter values. Using the methods discussed in this thesis, for each parameter value, we would have to compute a new solution from scratch, leading to an infeasible amount of computation time. Instead, we could explore method such as the empirical interpolation method that would allow solutions with different parameter values to be computed very efficiently.

REFERENCES CITED

[1] M. Abramowitz and I. Stegun. Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables. Applied mathematics series. Dover Publ., 1964.

[2] J. Boyd. Chebyshev and Fourier Spectral Methods. Dover Publications, second edition, 2001.

[3] C. Canuto, M. Hussaini, A. Quarteroni, and T. Zang. Spectral Methods: Fundamentals in Single Domains. Springer-Verlag, 2006.

[4] J. Cole. On a quasilinear parabolic equation occuring in aerodynamics. Q. Appl. Math., 9:225–236, 1951.

[5] J. Cooley and J. Tukey. An algorithm for the machine calculation of complex fourier series. Math. Comput., pages 297–301, 1965.

[6] A. Debussche, T. Dubois, and R. Temam. The nonlinear Galerkin method: A multiscale method applied to the simulation of homogeneous turbulent flows. Theoretical and Computational Fluid Dynamics, 7:279–315, 1995.

[7] M. Frigo and S. Johnson. The design and implementation of FFTW3. Proceedings of the IEEE, 93(2):216–231, 2005.

[8] R. Haberman. Applied Partial Differential Equations (with Fourier Series and Bondary Value Problems). Prentice-Hall, fourth edition, 2004.

[9] E. Hairer, S. Nørsett, and G. Wanner. Solving Ordinary Differential Equations I: Nonstiff Problems. Springer-Verlag, second edition, 1993.

[10] E. Hairer and G. Wanner. Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems. Springer-Verlag, second edition, 1996.

[11] E. Hopf. The partial differential equation $u_t + uu_x = \mu u_{xx}$. Commun. Pure Appl. Math., 3:201–230, 1950.

[12] T. Hou and R. Li. Computing nearly singular solutions using pseudo-spectral methods. J. Comput. Physics, 226:379–397, 2007.

[13] J. Mathews and K. Fink. Numerical Methods Using MATLAB. Prentice Hall, third edition, 1999.

[14] S. Orszag. On the elimination of aliasing in finite difference schemes by filtering high-wavenumber components. Journal of the Atmospheric Sciences, 28:1074, 1971.

[15] J. Shen, T. Tang, and L. Wang. Spectral Methods: Algorithms, Analysis and Applications. Springer-Berlag, 2011.

[16] M. Spivak. Calculus. Publish or Perish, third edition, 1994.

[17] L. Trefethen. Spectral Methods in Matlab. SIAM, 2000.