Exploring the Efficacy of Fourier Analysis in Solving Ordinary and Partial Differential Equations

To what extent, can Fourier Analysis be used to solve Ordinary Differential Equations and Partial Differential Equations?

Subject: Mathematics

Word Count: 3272

May 2024 Examination Session

Contents

1 Introduction

The allure of differential equations and Fourier analysis for me lies in their ability to unravel the intricate patterns of nature and unlock the hidden harmonies within complex systems. The ubiquitous presence of these fields across diverse scientific disciplines from physics and engineering to biology and economics help us understand the phenomena of celestial orbits and fluid flow to disease spread and stock market trends using one uniform language (Brunton and Kutz, 2022).

Recently, I utilized differential equations to come up with the following accurate and applicable epidemic model for policymakers that accounts for disease incubation, quarantine, immunity wear-off, and mortality rates (Smith et al., 2004).

$$
\begin{aligned}
\frac{dS}{dt} &= -\frac{\beta SI}{N} + \mu R \\
\frac{dE}{dt} &= \frac{\beta SI}{N} - \phi E \\
\frac{dI}{dt} &= \phi E - \zeta I - \gamma I - \alpha I \\
\frac{dQ}{dt} &= \zeta I - \kappa Q - \epsilon Q \\
\frac{dR}{dt} &= \gamma I + \kappa Q - \mu R \\
\frac{dD}{dt} &= \alpha I + \epsilon Q
\end{aligned}
$$

where:

- N is the total population.
- β is the contact rate (1/days).
- ϕ is the incubation rate (1/days).
- ζ is the quarantine rate (1/days).
- γ is the recovery rate (non-quarantine) (1/days).

- κ is the recovery rate (quarantine) (1/days).
- μ is the immunity wearoff rate (1/days).
- α is the case fatality rate (non-quarantine) (1/days).
- ϵ is the case fatality rate (quarantine) (1/days).

I was astonished by how elegantly this model was described in the language of differential equations. Hence, through the use of differential equations, I was able to unveil the secrets of the behavior of a system and its change over time.

Fourier analysis, on the other hand, allowed me to deconstruct complex phenomena into fundamental sine and cosine components, revealing the underlying frequencies and patterns. During my research at my internship, I utilized Fourier analysis to analyze the frequency content of a complex, noisy, non-stationary vibration signal from an Inertial Measurement Unit (IMU) and perform order analysis to study and understand the fundamental orders and frequencies of the system. I was amazed how clear this coordinate transformation made the signal and how much information it revealed about the system's behavior. I became fascinated by their practical applications across scientific disciplines, enabling us to predict, simulate, and optimize the behavior of diverse systems. Thus, through this Extended Essay, I decided to investigate the question: **To what extent, can Fourier Analysis be used to solve Ordinary Differential Equations and Partial Differential Equations?**

Throughout this paper, I utilize Fourier Analysis to simplify the order of ODE systems and reduce PDE systems into ODE systems in a wide array of fields ranging from physics to biology to finance. **Overall, while Fourier Analysis has its limitations and thus cannot be applied to all differential equation systems, in the areas where it proves effective, it significantly simplifies and solves otherwise challenging ordinary differential equations (ODEs) and partial differential equations (PDEs), thus making it a valuable analysis tool.**

2 Background

2.1 Differential Equations

2.1.1 Why are Differential Equations important?

Calculus is the mathematics of change. Hence, differential equations, which relate the derivatives or integrals of a function to the function itself, can very elegantly summarize the behavior of otherwise complex, dynamic systems. As a result, differential equations are extremely powerful in their ability to model various systems in applied mathematics, physics, and engineering.

For example, the Lotka-Volterra equations describe the dynamics of populations of predators and prey. These equations are described below (Wangersky, 1978).

$$\frac{dx}{dt} = \alpha x - \beta xy \tag{1}$$

$$\frac{dy}{dt} = \delta xy - \gamma y \tag{2}$$

where:

- x is the population density of the prey.
- y is the population density of the predator.
- α is the exponential growth rate of the prey.
- γ is the exponential decay rate of the predators.
- β is the effect of the predators on the prey growth rate.
- δ is the effect of the presence of prey on the predator growth rate.

Equations 1 and 2 are a set of first-order, nonlinear ODEs. This is further explained in Section 2.1.2 Properties of Differential Equations.

2.1.2 Properties of Differential Equations

Definition 2.1. The order of a system of differential equations is defined as the highest-order derivative the system contains. Since the highest order derivative in the Lotka-Volterra Equations 1 and 2 is $\frac{d}{dt}$, the equations are first-order.

Definition 2.2. If $x_1(t)$ and $x_2(t)$ are both solutions to a linear system of differential equations, then any linear superposition is also a solution to the system, namely anything in the form:

$$x(t) = c_1 x_1(t) + c_2 x_2(t) \tag{3}$$

More formally, a system of differential equations is said to be linear if and only if the equations follows the form:

$$a_0 y + a_1 y' + a_2 y'' \cdots + a_n y^{(n)} = b(x) \tag{4}$$

where $a_0, a_1, a_2, ..., a_n$ are any differentiable functions (do not need to be linear).

2.1.3 Solution to Differential Equations

Note that the solution to a differential equation is not one function, but rather a set of functions that all satisfy the differential equation. Some initial conditions must be given to reduce the solution to a single function. For example, for the Lotka-Volterra equations, the phase space shown in Figure 1 plots the various function solutions given various different initial conditions. The smaller loops represent a small predator-prey environment, while the larger loops represent a large predator-prey environment. The solution for a specific initial condition over time is plotted in Figure 2.

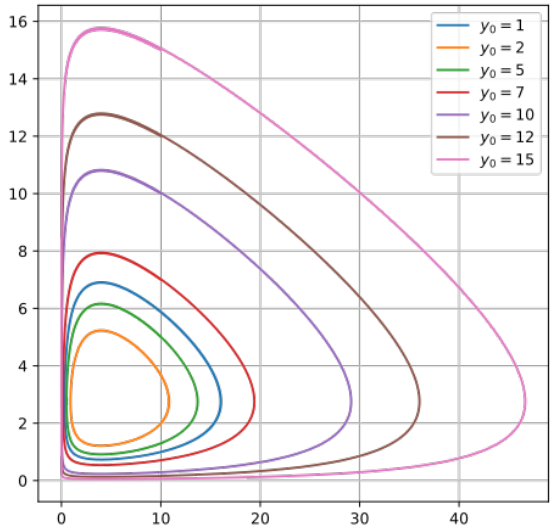

Figure 1: Solution to the Lotka-Volterra Equations given different predator initial conditions.

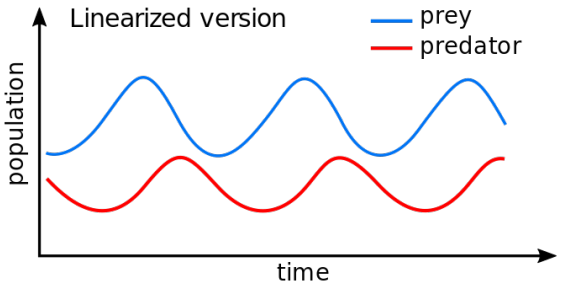

Figure 2: Solution to the Lotka-Volterra Equations given some initial conditions. The predator population function lags behind the prey population as an increase in the prey causes a proportional increase in the predator. Specifically, the predator function is shifted $\frac{\pi}{2}$ radians right of the prey function.

2.1.4 ODE vs PDE

The Lotka-Volterra Equations are a system of Ordinary Differential Equations (ODEs). This means that the prey population x and the predator population y are solely functions of time t.

However, not all functions will be a function of solely one variable such as time or space. For example, $f(x,t)$ could represent a function dependent on both a position x and a time t. In this case, Partial Differential Equations (PDEs) could be written using partial derivatives to express the gradient for the multivariate function with respect to a specific variable. Equation 5 below is a trivial PDE.

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial t} = 0 \tag{5}$$

2.2 Fourier Analysis

2.2.1 What is Fourier Analysis?

Fourier Analysis is a field of mathematics that studies how complex function waveforms can be decomposed into a series of sinusoidal functions, whose frequencies form a harmonic series. In other words, the Fourier Series and Fourier Transform, the cornerstones of Fourier Analysis, turn a signal in time-space respectively into a signal in frequency space and a signal in real space into a signal in Fourier space (Stein and Shakarchi, 2011).

2.2.2 Difference between Fourier Series and Fourier Transform

Fourier series focuses on periodic functions and aims to represent them as a sum of sinusoidal functions (sine and cosine). It decomposes a periodic signal into a combination of harmonically related sine and cosine waves, revealing the frequency components present in the signal. On the other hand, Fourier transforms extend this idea to non-periodic signals or functions, allowing for the analysis of signals that do not repeat in a periodic manner. Fourier transforms provide a continuous spectrum representation of a signal in terms of frequency, capturing both the amplitude and phase information across all frequencies (Panigrahi, 2022).

2.2.3 Fourier Series

Let $f(x)$ be defined as the following periodic, yet discontinuous square waveform signal (Heaviside Step Function) (MathWorld, 2023):

$$f(x) = \begin{cases} 1 & \text{if } nT < x < \frac{T(2n+1)}{2}, n \in \mathbb{Z} \\ 0 & \text{if } x = \frac{nT}{2}, n \in \mathbb{Z} \\ -1 & \text{if } \frac{T(2n+1)}{2} < x < T(n+1), n \in \mathbb{Z} \end{cases} \tag{6}$$

In order to obtain the Fourier Series for $f(x)$, the Fourier Transform of $f(x)$ must be taken (Panigrahi, 2022).

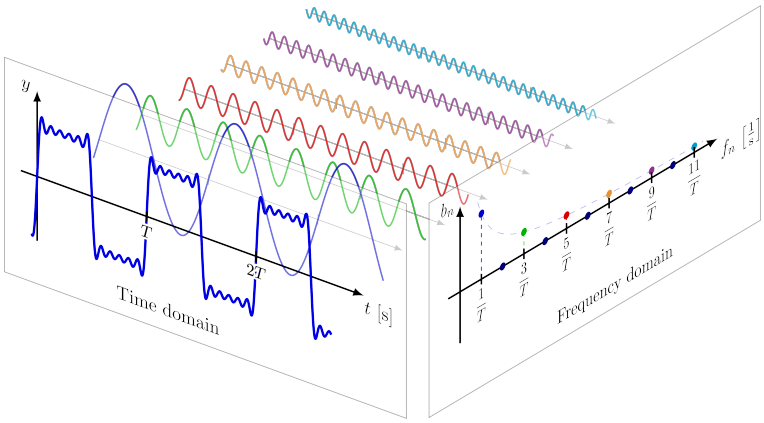

Figure 3: On the axis to the right, the Fourier Transform of $f(x)$ can be seen. If an infinite summation of all the frequencies with their corresponding amplitudes was performed as defined by the Fourier Transform, the solution would be exactly equal to $f(x)$.

Therefore, f(x) (Equation 6) can be defined by the infinite summation of sine functions through its Fourier Series as follows:

$$f(x) = \frac{4}{\pi} \sum_{n=1,3,5,\ldots}^{\infty} \frac{1}{n} \sin\left(\frac{2\pi n x}{T}\right) \tag{7}$$

2.2.4 Fourier and Inverse Fourier Transforms

Definition 2.3. For a continuous function $f(x)$, the Continuous Fourier Transform $\mathcal{F}\{f(x)\}$ (CFT) is defined as below (Grafakos et al., 2008). The transform returns the frequency space function $F(\omega)$.

$$F(\omega) = \mathcal{F}\{f(x)\} = \int_{-\infty}^{\infty} f(x)\, e^{-i\omega x}\, dx \tag{8}$$

Definition 2.4. In order to reverse the Fourier Transform, the Inverse Continuous Fourier

Transform $\mathcal{F}^{-1}\{F(\omega)\}$ (ICFT) can be applied as defined below (Grafakos et al., 2008).

$$f(x) = \mathcal{F}^{-1}\{F(\omega)\} = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)\, e^{i\omega x}\, d\omega \tag{9}$$

2.2.5 Properties of the Fourier Transform

The Fourier Transform, which is a coordinate transform, is important and useful because many mathematical operations in Fourier space are simplified compared to the original space. Below are some of the important properties of the Fourier Transform that will be useful in solving differential equations using Fourier Analysis (Bahouri, 2011; Sanderson, 2018).

Lemma 2.1. If a function $f(x)$ is scaled by a constant a, the Fourier Transform will also be scaled by a.

$$\mathcal{F}\{af(x)\} = a\mathcal{F}\{f(x)\} \tag{10}$$

Proof.

$$\begin{aligned}\mathcal{F}\{af(x)\} &= \int_{-\infty}^{\infty} af(x)\, e^{-i\omega x}\, dx \\ &= a\int_{-\infty}^{\infty} f(x)\, e^{-i\omega x}\, dx \\ &= a\mathcal{F}\{f(x)\}\end{aligned}$$

□

Theorem 2.1. The Fourier Transform is a linear transformation. This means that the transform of a linear combination of functions is equal to the linear combination of the transforms of the individual functions (Bahouri, 2011; Sanderson, 2018).

$$\mathcal{F}\{af(x) + bg(x)\} = a\mathcal{F}\{f(x)\} + b\mathcal{F}\{g(x)\} \tag{11}$$

Proof.

$$\mathcal{F}\{af(x) + bg(x)\} = \int_{-\infty}^{\infty} (af(x) + bg(x))\, e^{-i\omega x}\, dx$$

$$= \int_{-\infty}^{\infty} af(x)\, e^{-i\omega x}\, dx + \int_{-\infty}^{\infty} bg(x)\, e^{-i\omega x}\, dx$$

Using Lemma 2.1,

$$= a\int_{-\infty}^{\infty} f(x)\, e^{-i\omega x}\, dx + b\int_{-\infty}^{\infty} g(x)\, e^{-i\omega x}\, dx$$
$$= a\mathcal{F}\{f(x)\} + b\mathcal{F}\{g(x)\}$$

□

Theorem 2.2. The Fourier Transform can be shifted easily along the axis of the transform. When a function $f(x)$ along the x-axis by x_0 units, the transform of the shifted function is equal to the transform of the original function multiplied by $e^{-i\omega x_0}$ (Bahouri, 2011; Sanderson, 2018).

$$\mathcal{F}\{f(x - x_0)\} = e^{-i\omega x_0} F(\omega) \tag{12}$$

Proof.

$$\mathcal{F}\{f(x - x_0)\} = \int_{-\infty}^{\infty} f(x - x_0)\, e^{-i\omega x}\, dx$$

Let $u = x - x_0$. Thus, $x = u + x_0$ and $dx = du$.

$$= \int_{-\infty}^{\infty} f(u)\, e^{-i\omega(u+x_0)}\, du$$
$$= e^{-i\omega x_0} \int_{-\infty}^{\infty} f(u)\, e^{-i\omega u}\, du$$
$$= e^{-i\omega x_0} F(\omega)$$

□

Theorem 2.3. The Fourier Transform of a function's derivative is equal to the transform of the original function multiplied by $i\omega$ (Bahouri, 2011; Sanderson, 2018).

$$\mathcal{F}\{\frac{df(x)}{dx}\} = i\omega\mathcal{F}\{f(x)\} \tag{13}$$

Proof.

$$\mathcal{F}\{\frac{df(x)}{dx}\} = \int_{-\infty}^{\infty} \frac{df(x)}{dx}\, e^{-i\omega x}\, dx$$

Using integration by parts, let $u = f(x)$ and $dv = e^{-i\omega x}dx$.

$$= \left[f(x)e^{-i\omega x}\right]_{-\infty}^{\infty} - \int_{-\infty}^{\infty} f(x) \left(\frac{d}{dx}e^{-i\omega x}\right)\, dx$$

Since $f(x)$ must be continuous and integrable to be differentiable and take the Fourier Transform on $\mathbb{R}$, $\lim_{x\to-\infty} f(x) = \lim_{x\to\infty} f(x) = 0$ must be true. Thus, the first term is equal to zero.

$$\begin{aligned}
&= -\int_{-\infty}^{\infty} f(x) \left(-i\omega e^{-i\omega x}\right)\, dx \\
&= i\omega \int_{-\infty}^{\infty} f(x)e^{-i\omega x}\, dx \\
&= i\omega\mathcal{F}\{f(x)\}
\end{aligned}$$

□

Theorem 2.4. The Fourier Transform of the product of two functions is equal to the convolution of the transforms of the individual functions over 2π (Bahouri, 2011; Sanderson, 2018).

$$\mathcal{F}\{f(x)g(x)\} = \frac{1}{2\pi}(F(\omega) * G(\omega)) \tag{14}$$

Proof.

$$\mathcal{F}\{f(x)g(x)\} = \int_{-\infty}^{\infty} f(x)g(x)\, e^{-i\omega x}\, dx$$

From the definition of the inverse Fourier Transform in Equation 9,

$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} F(\kappa)e^{i\kappa x}\, d\kappa\right) g(x)\, e^{-i\omega x}\, dx$$

Using Fubini's Theorem, we can switch the order of integration.

$$
\begin{aligned}
&= \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\kappa) \left(\int_{-\infty}^{\infty} g(x) e^{-i\omega x} e^{i\kappa x} \, dx \right) d\kappa \\
&= \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\kappa) \left(\int_{-\infty}^{\infty} g(x) e^{-i(\omega-\kappa)x} \, dx \right) d\kappa \\
&= \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\kappa) G(\omega - \kappa) \, d\kappa
\end{aligned}
$$

Given the definition of convolution of two functions $f(x)$ and $g(x)$ as:

$$
f(x) * g(x) = \int_{-\infty}^{\infty} f(\kappa) g(x - \kappa) \, d\kappa
$$

The Fourier Transform of the product of two functions is equal to the convolution of the transforms of the individual functions over 2π.

$$
\mathcal{F}\{f(x)g(x)\} = \frac{1}{2\pi}(F(\omega) * G(\omega))
$$

□

3 ODE Simplification

A primary application of Fourier Analysis is to simplify higher-order ODE systems into lower-order ODE systems. This can greatly reduce the complexity of the problem and can allow for either simpler analytical solutions or numerical solutions using Euler's Method or Runge-Kutta methods. The following examples aim to illustrate this application of Fourier Analysis (Libretexts, 2021a).

3.1 Driven Simple Harmonic Oscillator with Damping

The following ODE for a Driven Simple Harmonic Oscillator with Damping represents the dynamic equation governing the motion of a mass subjected to both a restoring force and an external driving force while dissipating energy due to damping.

3.1.1 Definition

Let us define a Simple Harmonic Oscillator (SHO) of mass m and is attached to a spring with spring constant k. The SHO is subject to a damping force with damping coefficient γ and an additional driving force $f(t)$.

Define $\omega_0 = \sqrt{\frac{k}{m}}$ as the natural frequency of the SHO and the damping force as $F_d = -2m\gamma v$, where v is the velocity of the SHO.
Using Newton's Second Law,

$$\Sigma F = ma = -m\omega_0^2 x - 2m\gamma v + f(t) \tag{15}$$

$$m\frac{d^2x}{dt^2} = -m\omega_0^2 x - 2m\gamma\frac{dx}{dt} + f(t) \tag{16}$$

Dividing by m and rearranging terms, we get the following differential equation for a driven SHO with damping (Libretexts, 2021b):

$$\frac{d^2x}{dt^2} + 2\gamma\frac{dx}{dt} + \omega_0^2 x(t) = \frac{f(t)}{m} \tag{17}$$

3.1.2 Application of the Fourier Transform

Let us take the Fourier Transform of Equation 17 with respect to t. Thus, let $X(\omega)$ and $F(\omega)$ be the respective Fourier Transforms of $x(t)$ and $f(t)$ with respect to t.

$$\mathcal{F}_t\left\{\frac{d^2x}{dt^2} + 2\gamma\frac{dx}{dt} + \omega_0^2 x(t)\right\} = \mathcal{F}_t\left\{\frac{f(t)}{m}\right\} \tag{18}$$

Using Theorem 2.1,

$$\mathcal{F}_t\left\{\frac{d^2x}{dt^2}\right\} + \mathcal{F}_t\left\{2\gamma\frac{dx}{dt}\right\} + \mathcal{F}_t\left\{\omega_0^2 x(t)\right\} = \mathcal{F}_t\left\{\frac{f(t)}{m}\right\} \tag{19}$$

Using Lemma 2.1,

$$\mathcal{F}_t\left\{\frac{d^2x}{dt^2}\right\} + 2\gamma\mathcal{F}_t\left\{\frac{dx}{dt}\right\} + \omega_0^2\mathcal{F}_t\left\{x(t)\right\} = \frac{\mathcal{F}_t\left\{f(t)\right\}}{m} \tag{20}$$

Using Theorem 2.3,

$$\mathcal{F}_t\left\{\frac{dx}{dt}\right\} = i\omega\mathcal{F}_t\left\{x(t)\right\} \tag{21}$$

$$= i\omega X(\omega) + C_1 \tag{22}$$

$$\mathcal{F}_t\left\{\frac{d^2x}{dt^2}\right\} = i\omega\mathcal{F}_t\left\{\frac{dx}{dt}\right\} \tag{23}$$

$$= -\omega^2 X(\omega) + i\omega C_1 + C_2 \tag{24}$$

Therefore,

$$-\omega^2 X(\omega) + i\omega C_1 + C_2 + 2\gamma i\omega X(\omega) + 2\gamma C_1 + \omega_0^2 X(\omega) = \frac{F(\omega)}{m} \tag{25}$$

$$X(\omega)\left(\omega_0^2 - \omega^2 + 2\gamma i\omega\right) = \frac{F(\omega) - mC_1\left(i\omega + 2\gamma\right) - mC_2}{m} \tag{26}$$

Thus,

$$X(\omega) = \frac{F(\omega) - mC_1\left(i\omega + 2\gamma\right) - mC_2}{m\left(\omega_0^2 - \omega^2 + 2\gamma i\omega\right)} \tag{27}$$

By applying the Fourier Transform to the driven SHO equation, we have reduced the problem from solving a second order ODE (Equation 17) to solving a simple algebraic equation (Equation 27). This is a significant reduction in complexity.

Once we have solved for $X(\omega)$, we can take the inverse Fourier Transform of Equation 27 to obtain $x(t)$ (Libretexts, 2021b).

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{i\omega t}d\omega \tag{28}$$

$$= \frac{1}{2\pi}\int_{-\infty}^{\infty} \frac{F(\omega) - mC_1\left(i\omega + 2\gamma\right) - mC_2}{m\left(\omega_0^2 - \omega^2 + 2\gamma i\omega\right)}d\omega \tag{29}$$

3.1.3 Initial and Boundary Conditions

To constrain the solution of our ODE, we must provide initial and boundary conditions for the ODE. First, we will set the position of the SHO initially at 0.5 units above the

equilibrium point. The following condition represents this:

$$x(0) = 0.5 \tag{30}$$

For sake of example, a non-periodic driving force as defined below will be utilized:

$$f(t) = e^{-\frac{t}{500}} \tag{31}$$

Let us also apply a boundary condition to allow the SHO to start at rest.

$$x'(0) = 0 \tag{32}$$

3.1.4 Numerical Solution to Driven SHO

Equation 28 can be easily solved numerically using the Fast Fourier Transform (FFT) and the Inverse Fast Fourier Transform (IFFT) algorithms. The FFT and IFFT algorithms are optimized algorithms for computing the Discrete Fourier Transform (DFT) and the Inverse Discrete Fourier Transform (IDFT) of a sequence. The Python code can be found in appendix A.1.

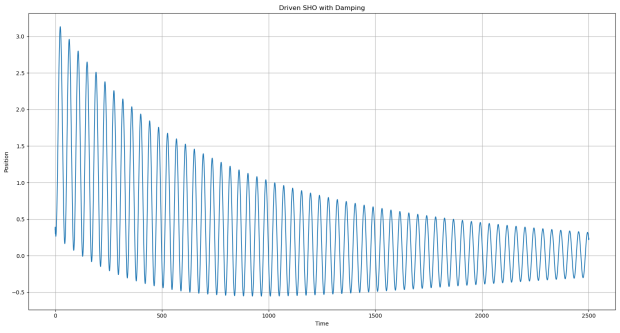

Figure 4: Defining Equation 30 as the initial condition, above is the solution to the driven SHO equation with damping using the FFT and IFFT algorithms in Python.

4 PDE to ODE Reduction

Furthermore, the Fourier Transform serves as a powerful mathematical tool to solve partial differential equations (PDEs) by enabling the reduction of complex PDE systems to simpler ordinary differential equations (ODEs). Consequently, the system's complexity diminishes, paving the way for more manageable ODEs. This reduction not only facilitates the exploration of analytical solutions but also enhances the feasibility of employing numerical techniques like Euler's Method or Runge-Kutta methods for efficient computation. The utilization of Fourier Transform in this context offers a valuable approach to gaining deeper insights into the behavior and dynamics of various systems, making it a fundamental tool in the study of differential equations as shown by the following examples (Danchin, 2005).

4.1 The Heat Equation

Fourier Analysis was first discovered when Joseph Fourier first developed the Heat Equation. The Heat Equation models the flow of heat along a certain heat profile over time. The Heat Equation is more generally known as the Diffusion Equation and is used to model the diffusion of varying concentrations of a substance in space and time. Thus, this equation has applications in physics, chemistry, biology, engineering, and beyond. In this example, a 1D solution will be derived and solved (Sanderson, 2019).

4.1.1 Definition

Let us define a rod with the following assumptions:

- The rod is of length L and is composed of a homogeneous material with a heat diffusion coefficient α^2.
- The rod is perfectly insulated along the Y and Z axes. Thus, heat can only flow along the X-axis of the rod.

- The rod is thin enough such that the temperature of the rod at any cross-section is uniform.
- The rod is initially at a uniform temperature $u(x,0) = f(x)$. Thus, the rod temperature at position x at time t is $u(x,t)$.

Thus, the heat equation in one dimension is defined as follows (Crawford, 2022):

$$\frac{\partial u}{\partial t} = \alpha^2 \nabla^2 u = \alpha^2 \frac{\partial^2 u}{\partial x^2} \tag{33}$$

Or in subscript notation,

$$u_t = \alpha^2 u_{xx} \tag{34}$$

4.1.2 Application of the Fourier Transform

Since $u(x,t)$ is defined as the temperature of the rod at position x and time t, we can apply the Fourier Transform to $u(x,t)$ with respect to position x to reduce the Equation 34 PDE into an ODE. Thus, let $U(\kappa,t)$ be the Fourier Transform of $u(x,t)$ with respect to x.

Taking the Fourier Transform of Equation 34 with respect to x,

$$\mathcal{F}_x \{u_t\} = \mathcal{F}_x \left\{\alpha^2 u_{xx}\right\} \tag{35}$$

Using Lemma 2.1,

$$\mathcal{F}_x \{u_t\} = \alpha^2 \mathcal{F}_x \{u_{xx}\} \tag{36}$$

Using Theorem 2.3,

$$\mathcal{F}_x \left\{ \frac{\partial^2 u(x,t)}{\partial x^2} \right\} = i\kappa \mathcal{F}_x \left\{ \frac{\partial u(x,t)}{\partial x} \right\} \tag{37}$$

$$= -\kappa^2 \mathcal{F}_x \{u(x,t)\} \tag{38}$$

$$= -\kappa^2 U \tag{39}$$

Therefore,

$$\mathcal{F}_x\{u_t\} = -\alpha^2\kappa^2 U \tag{40}$$

$$\frac{dU}{dt} = -\alpha^2\kappa^2 U \tag{41}$$

4.1.3 Initial and Boundary Conditions

A simple square waveform will be used as the initial rod temperature for the ODE:

$$u(x,0) = \begin{cases} 0 & \text{if } 0 \leq x < \frac{2L}{5} \\ 1 & \text{if } \frac{2L}{5} \leq x \leq \frac{3L}{5} \\ 0 & \text{if } \frac{3L}{5} < x \leq L \end{cases} \tag{42}$$

Since no heat can diffuse out of the ends of the rod, the following boundary conditions must be true.

$$u_x(0,t) = u_x(L,t) = 0 \tag{43}$$

4.1.4 Numerical Solution to Heat Equation

Equation 41 is a first-order ODE that can be easily numerically integrated. A numerical solution to Equation 41 using a fifth-order Runge-Kutta approximation in Python is presented below. The Python code can be found in appendix A.2.

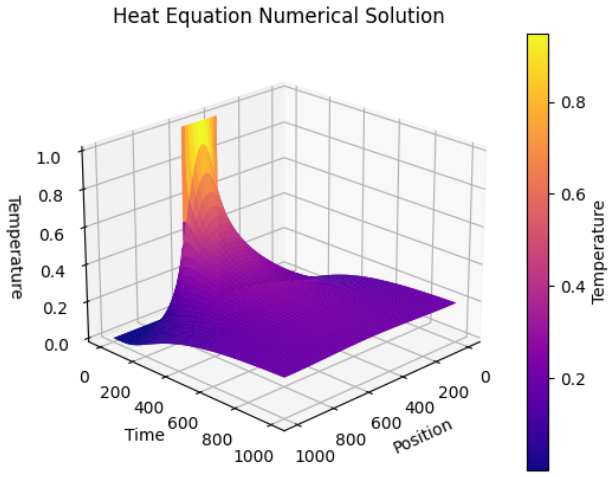

Figure 5: Using Equation 42 as the initial temperature function of the rod, this plot shows the temperature change along x and t by performing a numerical integration of Equation 41.

4.2 The Wave Equation

The wave equation is a PDE that describes how waves, such as sound or electromagnetic waves, propagate through a medium by relating the second derivative of the wave function with respect to both time and position, illustrating the wave's behavior and propagation characteristics.

4.2.1 Definition

Let us define a loop of string with the following assumptions:

- The string is of length L. A wave can travel through this medium at speed c.
- The wave energy does not dissipate as it travels through the string.
- The initial wave propagation is defined as $u(x, 0) = f(x)$. Thus, the amplitude of the wave from the equilibrium position at position x at time t is $u(x, t)$.

Thus, the wave equation in one dimension is defined as follows (Libretexts, 2022):

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u = c^2 \frac{\partial^2 u}{\partial x^2} \tag{44}$$

Or in subscript notation,

$$u_{tt} = c^2 u_{xx} \tag{45}$$

4.2.2 Application of the Fourier Transform

Since $u(x,t)$ is defined as the displacement of the wave at position x and time t, we can apply the Fourier Transform to $u(x,t)$ with respect to position x to reduce the Equation 45 PDE into an ODE. Thus, let $U(\kappa,t)$ be the Fourier Transform of $u(x,t)$ with respect to x.

Taking the Fourier Transform of Equation 45 with respect to x,

$$\mathcal{F}_x\left\{u_{tt}\right\} = \mathcal{F}_x\left\{\alpha^2 u_{xx}\right\} \tag{46}$$

Using Lemma 2.1,

$$\mathcal{F}_x\left\{u_{tt}\right\} = \alpha^2 \mathcal{F}_x\left\{u_{xx}\right\} \tag{47}$$

Using Theorem 2.3,

$$\mathcal{F}_x\left\{\frac{\partial^2 u(x,t)}{\partial x^2}\right\} = i\kappa \mathcal{F}_x\left\{\frac{\partial u(x,t)}{\partial x}\right\} \tag{48}$$

$$= -\kappa^2 \mathcal{F}_x\{u(x,t)\} \tag{49}$$

$$= -\kappa^2 U \tag{50}$$

Therefore,

$$\mathcal{F}_x\left\{u_{tt}\right\} = -\alpha^2\kappa^2 U \tag{51}$$

$$\frac{d^2U}{dt^2} = -c^2\kappa^2 U \tag{52}$$

4.2.3 Initial and Boundary Conditions

A hyperbolic secant distribution function will be used as the initial wave in the string for the ODE:

$$u(x, 0) = \text{sech}(x) \tag{53}$$

Though an open boundary condition could be employed, this would result in the wave propagating in one direction indefinitely. Instead of utilizing a large domain to represent this, the following periodic boundary condition will be employed such that the wave propagation travels out of the left of the domain and then feeds back into the right of the domain:

$$u(0, t) = u(L, t) \tag{54}$$

4.2.4 Numerical Solution to Wave Equation

Equation 52 is a first-order ODE that can be easily numerically integrated. A numerical solution to Equation 52 using a fifth-order Runge-Kutta approximation in Python is presented below. The Python code can be found in appendix A.3.

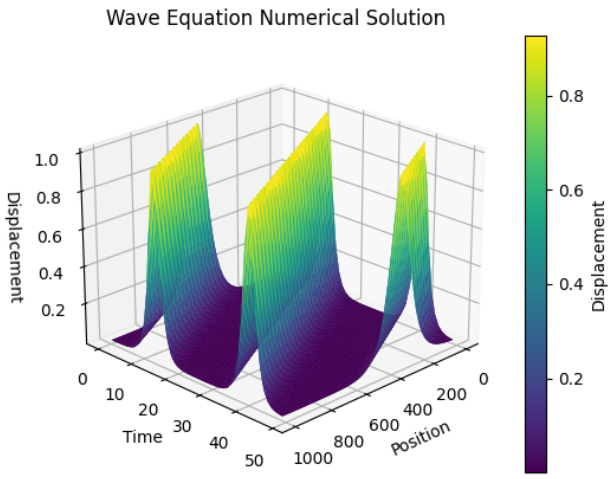

Figure 6: Defining Equation 53 the initial wave displacement, this plot shows the temperature change along x and t by performing a numerical integration of Equation 52.

4.3 Black-Scholes Equation

The Black-Scholes equation is a PDE that describes the depreciation in the price of a European call or put option at a given stock price and a given time.

4.3.1 Definition

Let us define a European call or put option with the following assumptions:

- Define σ to be the volatility of the underlying asset and r be the risk-free interest rate.
- The initial valuation of an option at a fixed stock price is $V(S, 0) = f(t)$. Thus, the option at a given underlying stock price S and a given option expiry time t is $V(S, t)$.

Thus, the Black-Scholes equation is defined as follows (Buchanan, 2014):

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0 \tag{55}$$

4.3.2 Application of the Fourier Transform

Let us take the Fourier Transform of Equation 55 with respect to S. Thus, let $\hat{V}(\kappa, t)$ be the Fourier Transform of $V(S, t)$ with respect to S.

$$\mathcal{F}_S\left\{\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV\right\} = \mathcal{F}_S\{0\} \tag{56}$$

Using Theorem 2.1 and the fact that the integral of the zero function is zero,

$$\mathcal{F}_S\left\{\frac{\partial V}{\partial t}\right\} + \mathcal{F}_S\left\{\frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}\right\} + \mathcal{F}_S\left\{rS\frac{\partial V}{\partial S}\right\} - \mathcal{F}_S\{rV\} = 0 \tag{57}$$

Using Lemma 2.1,

$$\mathcal{F}_S\left\{\frac{\partial V}{\partial t}\right\} + \frac{1}{2}\sigma^2 \mathcal{F}_S\left\{S^2 \frac{\partial^2 V}{\partial S^2}\right\} + r\mathcal{F}_S\left\{S\frac{\partial V}{\partial S}\right\} - r\mathcal{F}_S\{V\} = 0 \tag{58}$$

Using Theorem 2.3,

$$\mathcal{F}_S\left\{\frac{\partial V}{\partial S}\right\} = i\kappa\mathcal{F}_S\left\{V(S,t)\right\} \tag{59}$$

$$= i\kappa\hat{V}(\kappa,t) \tag{60}$$

$$\mathcal{F}_S\left\{\frac{\partial^2 V}{\partial S^2}\right\} = i\kappa\mathcal{F}_S\left\{\frac{\partial V}{\partial S}\right\} \tag{61}$$

$$= -\kappa^2\hat{V}(\kappa,t) \tag{62}$$

Using Theorem 2.4,

$$\mathcal{F}_S\left\{S\frac{\partial V}{\partial S}\right\} = \frac{1}{2\pi}(\mathcal{F}_S\left\{S\right\} * \mathcal{F}_S\left\{\frac{\partial V}{\partial S}\right\}) \tag{63}$$

$$= \frac{1}{2\pi}(\mathcal{F}_S\left\{S\right\} * i\kappa\hat{V}) \tag{64}$$

$$\mathcal{F}_S\left\{S^2\frac{\partial^2 V}{\partial S^2}\right\} = \frac{1}{2\pi}(\mathcal{F}_S\left\{S^2\right\} * \mathcal{F}_S\left\{\frac{\partial^2 V}{\partial S^2}\right\}) \tag{65}$$

$$= \frac{1}{2\pi}(\mathcal{F}_S\left\{S^2\right\} * -\kappa^2\hat{V}) \tag{66}$$

$$\tag{67}$$

Therefore,

$$\frac{\partial \hat{V}}{\partial t} + \frac{1}{2}\sigma^2\mathcal{F}_S\left\{S^2\frac{\partial^2 V}{\partial S^2}\right\} + r\mathcal{F}_S\left\{S\frac{\partial V}{\partial S}\right\} - r\hat{V} = 0 \tag{68}$$

$$\frac{\partial \hat{V}}{\partial t} + \frac{1}{2}\sigma^2\frac{1}{2\pi}(\mathcal{F}_S\left\{S^2\right\} * -\kappa^2\hat{V}) + r\frac{1}{2\pi}(\mathcal{F}_S\left\{S\right\} * i\kappa\hat{V}) - r\hat{V} = 0 \tag{69}$$

$$\frac{\partial \hat{V}}{\partial t} - \frac{1}{4\pi}\sigma^2\kappa^2(\mathcal{F}_S\left\{S^2\right\} * \hat{V}) + \frac{1}{2\pi}ri\kappa(\mathcal{F}_S\left\{S\right\} * \hat{V}) - r\hat{V} = 0 \tag{70}$$

Thus,

$$\frac{d\hat{V}}{dt} = \frac{1}{4\pi}\sigma^2\kappa^2(\mathcal{F}_S\left\{S^2\right\} * \hat{V}) - \frac{1}{2\pi}ri\kappa(\mathcal{F}_S\left\{S\right\} * \hat{V}) + r\hat{V} \tag{71}$$

4.3.3 Initial and Boundary Conditions

A simple linear function will be used as the initial option price for the ODE:

$$V(S_{max}, t) = 0.15t + 148.5 \tag{72}$$

If the underlying stock of an option is worth nothing, then the option itself is worth nothing. This boundary condition can expressed as follows:

$$V(0, t) = 0 \tag{73}$$

Furthermore, since a call option is advantageous when the stock price S is greater than the exercise price E and a put option is only advantageous when the exercise price E is greater than the stock price S, the following boundary conditions must be true.

$$V(S, t) = \begin{cases} max(S - E, 0) & \text{if call option} \\ max(E - S, 0) & \text{if put option} \end{cases} \tag{74}$$

4.3.4 Numerical Solution to Black-Scholes Equation

Equation 55 is a first-order ODE that can be easily numerically integrated. A numerical solution to Equation 71 using a fifth-order Runge-Kutta approximation in Python is presented below. The Python code can be found in appendix A.4.

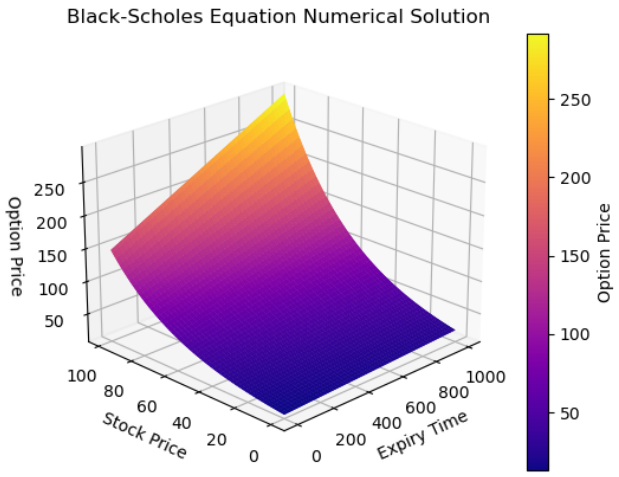

Figure 7: Defining the initial option price as Equation 72, this plot shows the temperature change along S and t by performing a numerical integration of Equation 71.

5 Limitations of Fourier Analysis

As shown by the various examples, Fourier transforms are a powerful tool for solving differential equations. However, there are certain situations where Fourier transforms may not be applicable or effective. Here are a few cases (Howell, 2016):

1. Nonlinear Equations: Fourier transforms are generally not applicable to nonlinear differential equations. The transforms rely on linearity properties, such as superposition and scaling, which do not hold for nonlinear equations. In such cases, other techniques like numerical methods or perturbation methods may be more suitable.

2. Variable Coefficients: Fourier transforms are most commonly used for differential equations with constant coefficients. When the coefficients of the differential equation are functions of the independent variable or have a complicated dependence, the application of Fourier transforms becomes more challenging. In such cases, specialized techniques like Laplace transforms or numerical methods may be employed.

3. Finite Boundaries: Fourier transforms are well-suited for solving differential equations on unbounded domains or for periodic problems. However, when dealing with

differential equations on finite intervals or with non-periodic boundary conditions, additional techniques like separation of variables, finite difference methods, or numerical techniques may be required.

4. Discontinuous Functions: Fourier transforms rely on the assumption that the functions involved are well-behaved and continuous. If the functions or their derivatives exhibit discontinuities, the Fourier transform may not be directly applicable. Techniques like generalized functions (e.g., distributions) or other specialized methods may be employed in these cases.

5. Stochastic or Random Processes: Fourier transforms are primarily used for deterministic differential equations. When dealing with stochastic or random processes, such as in the field of stochastic differential equations, other tools like stochastic calculus, probability theory, or numerical simulation methods may be more appropriate.

6 Conclusion

If a differential equation is linear and has constant coefficients, then Fourier analysis is especially useful to solve the differential equation by representing all solutions as a linear combination of sines and cosines. Furthermore, it is especially effective to reduce multidimensional, linear PDEs into a series of ODEs due to the simple interpretation of the derivative in Fourier space (Howell, 2016).

With that said, Fourier Analysis is not always the best method for solving differential equations. For example, the Airy ODE equation $y'' - xy = 0$ is not easily solvable using Fourier analysis, due to the non-constant coefficient x. Though Fourier analysis can still be used to solve the differential equation by representing the solution in terms of the Fourier Transform of x, the power series approach is a much easier method for solving this differential equation (Craig and Goodman, 1990).

Furthermore, there are trivial differential equations that Fourier analysis cannot solve due to its requirement needing an absolutely integrable and bounded function. For ex-

ample, take the differential equation $y' = 1$. It is trivial to find that the solution to this differential equation is $y = x + C$. However, since the integration of function diverges to infinity and the function is unbounded, Fourier Analysis cannot be used to solve this differential equation.

Overall, though the application of Fourier Analysis on differential equations has limitations, where it can be utilized it reduces and solves otherwise difficult-to-solve ordinary differential equations (ODEs) and partial differential equations (PDEs) to a significant extent. Hence, Fourier Analysis is an incredibly powerful and valuable tool for solving differential equations.

7 References

Bahouri, H. (2011). *Fourier analysis and nonlinear partial differential equations*. Springer.

Brunton, S. L. and Kutz, J. N. (2022). *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press.

Buchanan, R. (2014). *Solving the black-scholes equation*. https://sites.millersville.edu/rbuchanan/book/BlackScholes.pdf.

Craig, W. and Goodman, J. (1990). *Linear dispersive equations of Airy type. Journal of differential equations*, 87(1):38–61.

Crawford, T. (2022). *Oxford calculus: Heat equation derivation*. https://tomrocksmaths.com/2022/06/25/oxford-calculus-heat-equation-derivation/.

Danchin, R. (2005). *Fourier analysis methods for PDEs. UPEC Lecture notes*, 14(1).

Grafakos, L. et al. (2008). *Classical fourier analysis*, volume 2. Springer.

Howell, K. B. (2016). *Principles of Fourier analysis*. CRC Press.

Libretexts (2021a). *10.5: Fourier transforms of differential equations*. https://phys.libretexts.org/Bookshelves/Mathematical_Physics_and_Pedagogy/Complex_Methods_for_the_Sciences_(Chong)/10%3A_Fourier_Series_and_Fourier_Transforms/10.05%3A_Fourier_Transforms_of_Differential_Equations.

Libretexts (2021b). *11.1: The Driven Harmonic Oscillator. Physics LibreTexts*.

Libretexts (2022). *29: Solving the wave equation with Fourier transforms*. https://phys.libretexts.org/Courses/University_of_California_Davis/UCD%3A_Physics_156_-_A_Cosmology_Workbook/Workbook/29%3A_Solving_the_Wave_Equation_with_Fourier_Transforms.

MathWorld (2023). *Fourier Series - Square Wave*. https://mathworld.wolfram.com/FourierSeriesSquareWave.html.

Panigrahi, K. (2022). *Difference between Fourier Series and Fourier Transform*. https://www.tutorialspoint.com/difference-between-fourier-series-and-fourier-transform.

Sanderson, G. (2018). *But what is the Fourier Transform? A visual introduction.* https://www.3blue1brown.com/lessons/fourier-transforms.

Sanderson, G. (2019). *Solving the heat equation*. https://www.3blue1brown.com/lessons/heat-equation.

Smith, D., Moore, L., et al. (2004). *The SIR model for spread of disease-the differential equation model. Convergence*.

Stein, E. M. and Shakarchi, R. (2011). *Fourier analysis: an introduction*, volume 1. Princeton University Press.

Wangersky, P. J. (1978). *Lotka-Volterra population models. Annual Review of Ecology and Systematics*, 9(1):189–218.

A Appendix

A.1 Driven SHO with Damping Numerical Solution

```
# Simulate a simple SHO with damping and a driving force in the time domain using fourier transforms
#
# The equation of motion is:
# x'' + 2*gamma*x' + omega_0^2*x = f(t)/m
# where x is the position of the oscillator, gamma is the damping coefficient, omega_0 is the natural frequency of the oscillator without damping, and f(t) is the driving force.
#
# Taking the Fourier transform of both sides of the equation of motion gives:
# -omega^2*X + 2*gamma*i*omega*X + omega_0^2*X = F/m
# where X is the Fourier transform of x, omega is the frequency, and F is the Fourier transform of f(t).
#
# Solving for X gives:
# X = F/(m*(omega_0^2 - omega^2 + 2*i*gamma*omega))
#
# Taking the inverse Fourier transform of X gives the position of the oscillator in the time domain.

import numpy as np
import matplotlib.pyplot as plt

# Define the driving force
```

```
20 def f(t):
21     return np.exp(-t/500)
22
23 # Define parameters
24 omega_0 = 0.15 / (2 * np.pi)     # Natural frequency of the oscillator
   ↪  without damping
25 gamma = 0.0001 # Damping coefficient.
26 m = 1000    # Mass of the oscillator
27
28 # Define time
29 T = 2500.0
30 N = 100000 # Number of discretization points
31 dt = T/N
32 t = np.linspace(0,T,N)
33
34 # Make sure to satify the x_0 and v_0 initial conditions
35 # Take the Fourier transform of f(t)
36 F = np.fft.fft(f(t))
37
38 # Define omega
39 omega = np.fft.fftfreq(N, d=dt)
40
41 # Solve for X
42 x_0 = 0.5 # Initial position of the oscillator
43 v_0 = 0 # Initial velocity of the oscillator
44 C_1 = (-82.385 * x_0 * omega_0**2 - 31.91 * v_0 * omega_0)/dt
45 C_2 = (4.78 * x_0 * omega_0**2 - 0.3137 * v_0 * omega_0)/dt
46 X = (F - m*C_1*1j*omega - m*(2*gamma*C_1 + C_2))/(m*(omega_0**2 -
   ↪  np.power(omega, 2) + 2*1j*gamma*omega))
```

```
47
48 # Plot the Fourier transform of f(t) and X
49 plt.figure("Driven SHO with Damping")
50 plt.plot(omega,F.real, label='Real f(t)')
51 plt.plot(omega,F.imag, label='Imaginary f(t)')
52 plt.plot(omega,X.real, label='Real X')
53 plt.plot(omega,X.imag, label='Imaginary X')
54 plt.xlabel('Frequency')
55 plt.ylabel('Amplitude')
56 plt.title('Driven SHO with Damping')
57 plt.legend(loc=1)
58 plt.grid()
59
60 # Take the inverse Fourier transform of X
61 x = np.fft.ifft(X)
62
63 # Plot the position of the oscillator
64 plt.figure('Driven SHO with Damping Position')
65 plt.plot(t,x.real, label='Fourier Solution')
66 plt.xlabel('Time')
67 plt.ylabel('Position')
68 plt.title('Driven SHO with Damping')
69 plt.grid()
70 plt.show()
```

A.2 Heat Equation Numerical Solution

```
1 # Implementation of the heat equation in 1D
2 import numpy as np
3 import matplotlib.pyplot as plt
```

```
from scipy.integrate import odeint
from mpl_toolkits.mplot3d import axes3d

a = 1    # Thermal diffusivity constant
L = 100  # Length of domain
N = 1000 # Number of discretization points
dx = L/N
x = np.arange(-L/2,L/2,dx) # Define x domain

# Define discrete wavenumbers
kappa = 2*np.pi*np.fft.fftfreq(N, d=dx)

# Initial condition
u0 = np.zeros_like(x)
u0[int((L/2 - L/10)/dx):int((L/2 + L/10)/dx)] = 1
u0hat = np.fft.fft(u0)

# SciPy's odeint function doesn't play well with complex numbers, so
↪ we recast
# the state u0hat from an N-element complex vector to a 2N-element
↪ real vector
u0hat_ri = np.concatenate((u0hat.real,u0hat.imag))

# Simulate in Fourier frequency domain
dt = 0.1
t = np.arange(0,1000,dt)

def rhsHeat(uhat_ri,t,kappa,a):
    uhat = uhat_ri[:N] + (1j) * uhat_ri[N:]
```

```
    d_uhat = -a**2 * (np.power(kappa,2)) * uhat
    d_uhat_ri =
    ↪  np.concatenate((d_uhat.real,d_uhat.imag)).astype('float64')
    return d_uhat_ri

uhat_ri = odeint(rhsHeat, u0hat_ri, t, args=(kappa,a))

uhat = uhat_ri[:,:N] + (1j) * uhat_ri[:,N:]

u = np.zeros_like(uhat)

for k in range(len(t)):
    u[k,:] = np.fft.ifft(uhat[k,:])

u = u.real

# Mesh plot
u_plot = u[0:-1:10,:]
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.view_init(elev=22.5, azim=45, roll=0)

x = np.arange(u_plot.shape[1])
y = np.arange(u_plot.shape[0])
X, Y = np.meshgrid(x, y)

cmap = 'plasma'
ax.plot_surface(X, Y, u_plot, cmap=cmap)
cbar = plt.colorbar(ax.plot_surface(X, Y, u_plot, cmap=cmap), ax=ax)
```

```
cbar.set_label('Temperature')

ax.set_xlabel('Position')
ax.set_ylabel('Time')
ax.set_zlabel('Temperature')
ax.set_title('Heat Equation Numerical Solution')

# Image plot
# plt.figure()
# plt.imshow(np.flipud(u), aspect=8)
# plt.axis('off')
plt.show()
```

A.3 Wave Equation Numerical Solution

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from mpl_toolkits.mplot3d import axes3d

c = 3     # Wave speed
L = 20    # Length of domain
N = 1000 # Number of discretization points
dx = L/N
x = np.arange(-L/2,L/2,dx) # Define x domain

# Define discrete wavenumbers
kappa = 2*np.pi*np.fft.fftfreq(N, d=dx)

# Initial condition
```

```
16 u0 = 1/np.cosh(x)
17 u0hat = np.fft.fft(u0)
18
19 # SciPy's odeint function doesn't play well with complex numbers, so
   ↪ we recast
20 # the state u0hat from an N-element complex vector to a 2N-element
   ↪ real vector
21 u0hat_ri = np.concatenate((u0hat.real,u0hat.imag))
22
23 # Simulate in Fourier frequency domain
24 dt = 0.025
25 t = np.arange(0,500*dt,dt)
26
27 def rhsWave(uhat_ri,t,kappa,c):
28     uhat = uhat_ri[:N] + (1j) * uhat_ri[N:]
29     d_uhat = -c*(1j)*kappa*uhat
30     d_uhat_ri =
       ↪ np.concatenate((d_uhat.real,d_uhat.imag)).astype('float64')
31     return d_uhat_ri
32
33 uhat_ri = odeint(rhsWave, u0hat_ri, t, args=(kappa,c))
34 uhat = uhat_ri[:,:N] + (1j) * uhat_ri[:,N:]
35
36 # Inverse FFT to bring back to spatial domain
37 u = np.zeros_like(uhat)
38
39 for k in range(len(t)):
40     u[k,:] = np.fft.ifft(uhat[k,:])
41
```

```
u = u.real


# Waterfall plot
u_plot = u[0:-1:10,:]
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.view_init(elev=22.5, azim=45, roll=0)

x = np.arange(u_plot.shape[1])
y = np.arange(u_plot.shape[0])
X, Y = np.meshgrid(x, y)

cmap = 'viridis'
ax.plot_surface(X, Y, u_plot, cmap=cmap)
cbar = plt.colorbar(ax.plot_surface(X, Y, u_plot, cmap=cmap), ax=ax)
cbar.set_label('Displacement')

ax.set_xlabel('Position')
ax.set_ylabel('Time')
ax.set_zlabel('Displacement')
ax.set_title('Wave Equation Numerical Solution')

# Image plot
# plt.figure()
# plt.imshow(np.flipud(u), aspect=8)
# plt.axis('off')
plt.show()
```

A.4 Black-Scholes Equation Numerical Solution

```
# Implementation of the Black-Scholes equation
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from mpl_toolkits.mplot3d import axes3d

sigma = 10    # Volatility of underlying asset
r = 0.05      # Risk-free interest rate
L = 50  # Length of domain
N = 1000 # Number of discretization points
ds = L/N
s = np.arange(0,L,ds) # Define x domain

# Take the Fourier transform of S(t)
# Perform one-sided FFT
fourier_unit = np.fft.fft(np.ones_like(s)) / len(s)
fourier_S = -1j * fourier_unit
fourier_S_squared = -1 * fourier_unit

# Define discrete wavenumbers
kappa = 2*np.pi*np.fft.fftfreq(N, d=ds)

# Initial condition
v0 = s/4 + L/4
v0hat = np.fft.fft(v0)

```

```
# SciPy's odeint function doesn't play well with complex numbers, so we recast
# the state v0hat from an N-element complex vector to a 2N-element real vector
v0hat_ri = np.concatenate((v0hat.real,v0hat.imag))

# Simulate in Fourier frequency domain
def black_scholes(vhat_ri,t,kappa,sigma,r, fourier_S, fourier_S_squared):
    vhat = vhat_ri[:N] + (1j) * vhat_ri[N:]
    d_vhat = (sigma**2 / (4 * np.pi)) * (np.power(kappa,2)) * fourier_S_squared * vhat - (r / (2 * np.pi)) * (1j) * kappa * fourier_S * vhat + r * vhat
    d_vhat_ri = np.concatenate((d_vhat.real,d_vhat.imag)).astype('float64')
    return d_vhat_ri

vhat_ri = odeint(black_scholes, v0hat_ri, s, args=(kappa, sigma, r, fourier_S, fourier_S_squared))

vhat = vhat_ri[:,:N] + (1j) * vhat_ri[:,N:]

v = np.zeros_like(vhat)

for k in range(len(s)):
    v[k,:] = np.fft.ifft(vhat[k,:])

v = v.real

```

```
49 # Mesh plot
50 v_plot = v[0:-1:10,:]
51 fig = plt.figure()
52 ax = fig.add_subplot(111, projection='3d')
53 ax.view_init(elev=22.5, azim=-135, roll=0)
54 x = np.arange(v_plot.shape[1])
55 y = np.arange(v_plot.shape[0])
56 X, Y = np.meshgrid(x, y)
57 cmap = 'plasma'
58 ax.plot_surface(X, Y, v_plot, cmap=cmap)
59 cbar = plt.colorbar(ax.plot_surface(X, Y, v_plot, cmap=cmap), ax=ax)
60 cbar.set_label('Option Price')
61
62 ax.set_xlabel('Expiry Time')
63 ax.set_ylabel('Stock Price')
64 ax.set_zlabel('Option Price')
65 ax.set_title('Black-Scholes Equation Numerical Solution')
66
67 # Image plot
68 # plt.figure()
69 # plt.imshow(np.flipud(v_plot), aspect=8)
70 # plt.axis('off')
71 plt.show()
```