Numerical Solutions to Partial Differential Equations

Abhishek Mukherjee, Pun Cheuk Nga Janice,
Mateusz Jakub Staniszewski and Seongguk Ryou

Supervised by Professor Grigoris Pavliotis

June 2016

Abstract

From the famous Navier-Stokes equation which finds a variety of applications in engineering whilst also being of interest to pure mathematicians, to the Black-Scholes equation which arises in finance and options pricing, partial differential equations (PDEs) govern our universe at every level. In most practical cases, it is not possible to solve these equations analytically. This project gives an introduction to some numerical methods which can be used to estimate solutions to PDEs and provides a more in-depth analysis of spectral methods with some specific linear and non-linear examples.

In the first part of the project, an introduction to partial differential equations and various examples are given according to their classification, followed by an overview of the Finite Differences and Finite Elements methods for solving PDEs.

In the next section, spectral method are introduced with examples including linear PDEs such as the Heat, Wave, and Poisson equations and non-linear ones such as Inviscid Burgers' and Korteweg-de Vries' equations.

In the final part, the advantages and disadvantages of using Spectral Methods are discussed. Additionally all results are summarised and we examine areas for further analysis of these partial differential equations.

Keywords: numerical solutions, spectral methods, basis functions, exact solutions

Contents

1 Background

Partial differential equations govern our universe. The study of partial differential equations (or PDEs) was formally started by the works of d'Alembert, Euler, Bernoulli, and Laplace in the 18^{th} century [12] who were developing models for vibrating strings, acoustic waves, and gravitational potential fields. To this day, the applications of PDEs to physics and engineering still exist, but they also serve an important role in the development of other branches of mathematics such as the study of abelian functions as well as differential geometry.

Mathematical models have been widely used in the form of ordinary and partial differential equations connected with problems in quantum mechanics, linear and non-linear waves, complex analysis, and finance. To use mathematics models in computers, numerical methods are required. Numerical methods are used to find exact solutions of the equations in these models. Numerical techniques used to approximate solutions to PDEs include the finite element method, the finite difference method, and spectral methods. These will be covered in further detail throughout the project.

After the finite difference and finite element methods had already been introduced, the creation of the Fast Fourier Transform in 1965 was the impetus for the discovery of spectral methods in the 1970s. However, the Fast Fourier Transform is not always used when it comes to implementing spectral methods. This is because explicit matrix multiplication is often less costly in terms of computing cost and gives sufficiently comparable solutions.

Spectral methods are usually the best tool when solving ODEs or PDEs given on a simple domain and with sufficiently smooth data. Fourier and Chebyshev spectral methods are the two most common types. Ten digits of accuracy can usually be achieved without too much effort. In comparison, finite difference and finite element methods can only achieve 2 to 3 digits of accuracy with an equivalent system of equations. Moreover, spectral methods require less computation cost compared to the finite element method, but the drawback is that they are often less accurate for problems with complex geometries and discontinuous coefficients.

Figure 1: Pierre-Simon Laplace

Figure 2: Gustav De-Vries

2 Introduction to Partial Differential Equations

2.1 Examples of PDEs

A partial differential equation is an equation involving functions of a particular variable and their partial derivatives. Partial differential equations are generally more difficult to solve analytically than ordinary differential equations. Thankfully, they can be solved by methods such as separation of variable, using a Green's Function, or numerical methods.

Example of PDEs:

1. $Linear\, equations$
   (a) Laplace's Equation:
   $\Delta u = \sum_{i=1}^{n} u_{x_i x_i} = 0$
   (b) Helmoholtz's equation:
   $-\Delta u = \lambda u$
   (c) Heat (or diffusion Equation)
   $u_t - \Delta u = 0$
   (d) Wave Equation
   $u_{tt} - \Delta u = 0$
   (e) Schrödinger's Equation
   $iu_t + \Delta u - 0$
   (f) Beam equation
   $u_{tt} + u_{xxxx} = 0$

2. $Non-linear\, Equations$
   (a) Scalar reaction-diffusion equation
   $u_t - \Delta u = f(u)$
   (b) Non-linear Poisson Equation
   $-\Delta u = f(u)$
   (c) Inviscid Burgers' Equation
   $u_t + uu_x = 0$
   (d) Non-linear Wave Equation
   $u_{tt} - \Delta u + f(u) = 0$
   (e) Korteweg-de Vries Equation
   $u_t + uu_x + u_{xxx} = 0$

Second order linear PDEs can be formally classified into three types including elliptic, parabolic and hyperbolic. We can write any second order PDE in the form $Au_{xx} + 2Bu_{xy} + Cu_{yy} +\, lower\, order\, terms = 0$ and define the classifications as follows.

1. $Elliptic$ (if $B^2 - AC < 0$)
   e.g Poisson Equation or Laplace equation (when$f(x, y) = 0$)
   $U_{xx} + U_{yy} = f(x, y)$
2. $Parabolic$ (if $B^2 - AC = 0$)
   e.g Diffusion equation
   $U_t = kU_{xx}$
3. $Hyperbolic$ (if $B^2 - AC > 0$)
   e.g Wave Equation
   $U_{tt} = c^2 U_{xx}$

2.2 Analytical Solution to Laplace's Equation

Though it is not the primary focus of this paper, it is important to note that it's possible to solve PDEs without even requiring the use of numerical methods. Take, for example, Laplace's equation

$$\nabla^2\Phi = 0.$$

Re-expressing in polar co-ordinates [8], this can be written as

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial \Phi}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2\Phi}{\partial \theta^2} = 0.$$

Consider solutions of the form $\Phi(r,\theta) = R(r)\Theta(\theta)$ to use separation of variables and solve it analytically. Substituting this in gives

$$\begin{aligned}\frac{\Theta}{r}\frac{d}{dr}\left(r\frac{dR}{dr}\right) + \frac{R}{r^2}\frac{d^2\Theta}{d\theta^2} &= 0\\ \implies \frac{r}{R}\frac{d}{dr}\left(r\frac{dR}{dr}\right) &= -\frac{\Theta''}{\Theta} = \lambda.\end{aligned}$$

Because we have separated the variables, both sides must be equal to a constant λ. Focusing on Θ

$$\begin{aligned}\Theta'' &= -\lambda\Theta\\ \implies \Theta &= \begin{cases} A\,cos\sqrt{\lambda}\theta + B\,sin\sqrt{\lambda}\theta & \text{if } \lambda \neq 0\\ A + B\,\theta & \text{if } \lambda = 0\end{cases}\end{aligned}$$

Given that $\nabla\Theta$ is 2π periodic, we can rewrite this as

$$\Theta = \begin{cases} A\,cos\,n\theta + B\,sin\,n\theta & \text{if } n \neq 0\\ A + B\theta & \text{if } n = 0\end{cases}$$

Switching our attention to R, we can use the substitution $u = ln\,r$ to get

$$R = \begin{cases} Cr^n + Dr^{-n} & \text{if } n \neq 0\\ C + D\,ln\,r & \text{if } n = 0\end{cases}$$

Combining the solutions for R and Θ gives

$$R\Theta = \begin{cases} (Cr^n + Dr^{-n})(A\,cos\,n\theta + B\,sin\,n\theta) & \text{if } n \neq 0\\ (C + D\,ln\,r)(A + B\theta) & \text{if } n = 0\end{cases}$$

As we are dealing with a linear Partial Differential Equation we can express a general solution as a combination of all possible solutions

$$\Phi = A_0 + B_0\theta + C_0\,ln\,r + \sum_{n=1}^{\infty}(A_n r^n + C_n r^{-n})\,cos\,n\theta + \sum_{n=1}^{\infty}(B_n r^n + D_n r^{-n})\,sin\,n\theta.$$

We can eliminate the $\theta\,ln\,r$ solution as it does not satisfy the requirement that $\nabla\Theta$ is 2π periodic. We also relabel the constants such that $AC = A_n$, $AD = C_n$, $BC = B_n$, and $BD = D_n$. This expression could be simplified = even further, but the above expression is the most practical form. Then, given boundary conditions, we can work our specific solutions.

3 Method I: Finite Differences

3.1 Deriving Formula using Taylor Expansion

This part of the project will give an introduction to the finite difference method of solving partial differential equations, and an example of solving the heat equation in one space dimension will be presented under Dirichlet boundary conditions. Error and stability will also be briefly discussed.

In this method, the partial differential equations are replaced by a set of equations, using an associated grid system [2]. The derivatives are approximated by algebraic equations - to perform the analysis we shall find such expressions for:

$$\frac{\partial u(x,y)}{\partial x}, \frac{\partial^2 u(x,y)}{\partial y^2}, \frac{\partial^2 u(x,y)}{\partial x^2}, \frac{\partial^2 u(x,y)}{\partial y^2}$$

First consider Taylor expansions of $u(x,y)$, keeping t constant:

$$u(x+h,y) = u(x,y) + h\frac{\partial u(x,y)}{\partial x} + \frac{h^2}{2!}\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{h^3}{3!}\frac{\partial^3 u(x,y)}{\partial x^3} + \dots \tag{3.1}$$

$$u(x-h,y) = u(x,y) - h\frac{\partial u(x,y)}{\partial x} + \frac{h^2}{2!}\frac{\partial^2 u(x,y)}{\partial x^2} - \frac{h^3}{3!}\frac{\partial^3 u(x,y)}{\partial x^3} + \dots \tag{3.2}$$

Adding equations (3.1) and (3.2) and rearranging for $\frac{\partial u(x,y)}{\partial x}$ we obtain:

$$\frac{\partial u(x,y)}{\partial x} = \frac{u(x+h,y) + u(x-h,y)}{2h} + \mathcal{O}(h^2) \tag{3.3}$$

This is known as the central difference approximation. Directly from equations (3.1) and (3.2) we can also obtain the forward and backward difference approximations:

$$\frac{\partial u(x,y)}{\partial x} = \frac{u(x+h,y) - u(x,y)}{h} + \mathcal{O}(h) \tag{3.4}$$

$$\frac{\partial u(x,y)}{\partial x} = \frac{u(x,y) - u(x-h,y)}{h} + \mathcal{O}(h) \tag{3.5}$$

The remainder $\mathcal{O}(h)$, is known as the local truncation error and will be discussed in a later part of this section. Similarly, it will be useful to derive the second order partial derivative being (summing and rearranging (3.1) and (3.2)):

$$\frac{\partial^2 u(x,y)}{\partial x^2} = \frac{u(x+h,y) + u(x-h,y) - 2u(x,y)}{h^2} + \mathcal{O}(h^2) \tag{3.6}$$

Respective approximations can be obtained for partial derivatives with respect to time (denoting the small increment of time $\delta(y)$ by k).

3.2 Introducing Mesh Grid Notation

For further analysis of partial differential equations, the following grid shall be introduced:

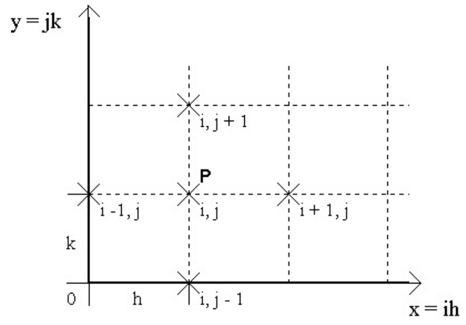


Figure 3: A grid with increments h in x direction and increment k in y direction at constant length

Figure 3 represents the mesh line grid notation which will be useful in dealing with finite difference approximation. Let $\delta(x)$ and $\delta(y)$ be constant and denoted by i and j respectively. Then equations (3.3) and (3.6) can be rewritten as:

$$\frac{\partial u_{i,j}}{\partial x} = \frac{u_{i+1,j} - u_{i-1,j}}{2h} + \mathcal{O}(h^2) \tag{3.7}$$

$$\frac{\partial^2 u_{i,j}}{\partial x^2} = \frac{u_{i+1,j} + u_{i-1,j} - 2u_{i,j}}{h^2} + \mathcal{O}(h^2) \tag{3.8}$$

For ease of representation, the Dirichlet boundary condition will be considered and thus problems arising with Neumann boundary conditions are omitted from the discussion.

3.3 Heat Equation

In the next part, a heat equation example shall be presented and numerically approximated with use of finite element method.

The canonical form of heat equation [2] is given as:

$$\frac{\partial U}{\partial T} = K\frac{\partial^2 U}{\partial X^2} \tag{3.9}$$

Let U be required in the interval $0 \leq X \leq L$ and let U_0 be particular value of U. Then by letting:

$$x = \frac{X}{L}, \qquad u = \frac{U}{U_0}, \qquad t = \frac{T}{T_0}$$

where T_0 is chosen s.t. $T_0 = \frac{L^2}{K}$ the eq. (3.9) becomes:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \qquad \text{where} \qquad 0 \le x \le 1 \tag{3.10}$$

By substituting the forward difference approximation (eq. (3.4)) for the LHS of equation (3.10) and the central difference approximation (eq. (3.8)) for the RHS, in the mesh grid notation we obtain:

$$\frac{u_{i+1,j} - u_{i-1,j}}{k} = \frac{u_{i+1,j} + u_{i-1,j} - 2u_{i,j}}{h^2} \tag{3.11}$$

Notice that that the truncation errors are of different orders and appears reasonable to set $k = \mathcal{O}(h^2)$. Also the equation is consistent (local truncation error tends to 0) as when:

$$\lim_{h\to\infty,k\to\infty} \mathcal{O}(k + h^2) = 0 \tag{3.12}$$

In practice we need to choose values for h and k. Here two types of errors shall be noted [3]:

1. Truncation error: due to approximation of derivative - reduced by choosing smaller values of h and k
2. Roundoff error: due to computer rounding at every calculation - reduced by choosing greater values of h and k

Additionally to ensure the solution of (3.11) to be stable we need [3]:

$$r = \frac{k}{h^2} \le \frac{1}{2} \tag{3.13}$$

With this knowledge, an attempt to solve a specific example of the heat equation will be made to explicitly show the finite difference method (this is done with the explicit method). Equation (3.11) can be rewritten as:

$$u_{i,j+1} = u_{i,j} + r(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}) \tag{3.14}$$

Consider equation (3.10) with conditions:

(i) $u(0,t) = u(0,L) = 0$ Boundary condition

(ii) $u(x,0) = sin(x)$ Initial condition

Using the fact that $L = 2\pi$ due to periodic solution, the Matlab code is used (can be found in Appendix), giving the following graphical solution:

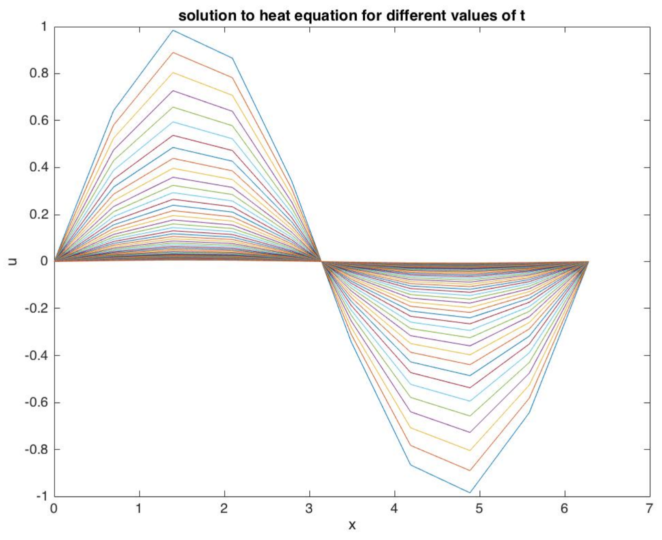


Figure 4: A plot of u vs. x at different times of t (corresponding to different lines)

The first plot (fig: 4) shows the 2-dimensional solution with each line corresponding to different solution at time t. For clarity, a ’waterfall’ plot is also shown giving a 3-dimensional plot of solution (fig: 5).

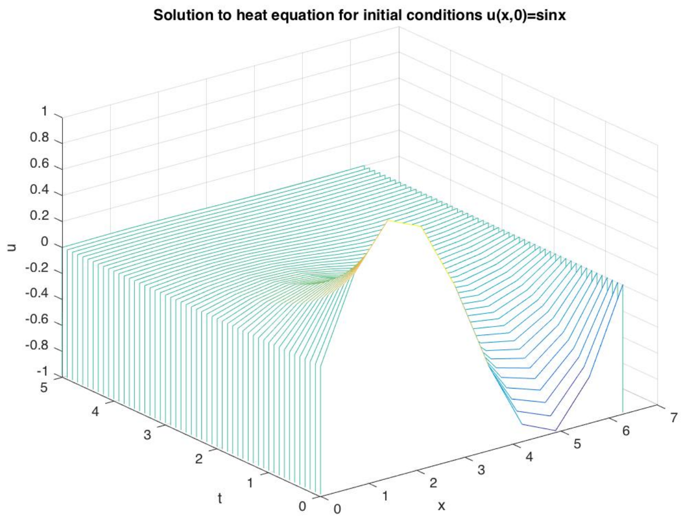


Figure 5: A plot of u vs. x at different times of t (corresponding to different lines)

The results will obtained above shall be used in discussion of comparison of methods in later parts of the paper. This part is closed with remark that the finite difference method has many other modifications and depending on boundary conditions other procedures need to be applied.

4 Method II: Finite Elements

4.1 Overview

The finite element method is a numerical method for solving partial differential equations which apply over complex shapes. In engineering science, it is difficult to solve a partial differential equation which applies over a complicated shape and they often cannot be solved directly. However the equations can be solved for very simple shapes like triangles and rectangles. The finite element method takes advantage of this fact. We replace the single complicated shape with an approximately equivalent network of simple elements. Finite element analysis can get you toward a solution when dealing with a very complicated shape undergoing complicated loads (temperature, vibration, earthquakes etc.) or trying to look at a part in great detail (this is more common in mechanical engineering).

The process is to solve the complex differential over each simpler shape and join all the simpler shapes together. This often results in thousands of simultaneous equations, which can be solved on a computer. The subdivision of a whole domain into simpler parts has several advantages: accurate representation of complex geometry, inclusion of dissimilar material properties, easy representation of the total solution, and capture of local effects. However there are limitations such as: models are time-consuming to create and verify, can be sensitive to boundary conditions (stress concentrations, local stresses) and sometimes the model needs to be refined repeatedly to give assurance that the results are reasonably accurate and valid. The method can be used for various fields of engineering such as dynamics, vibrations, fluid flow, structural analysis etc.

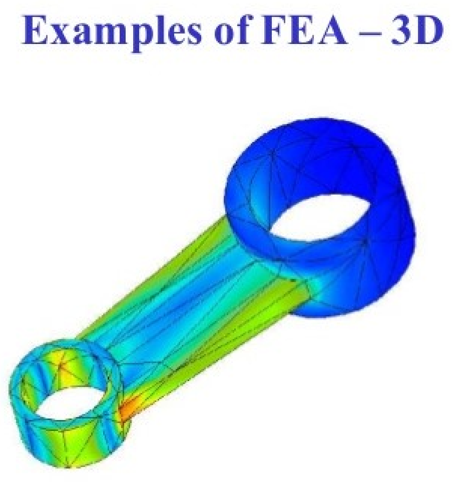


Figure 6: 3D complex structure sectioned by simpler finite elements

4.2 Historical background

The finite element method originated from the need to solve complex elasticity and structural analysis problems in civil engineering. It began in Russia with Galerkin who created a method to take the differential equation which is a continuous problem. As a computer can only solve finite(discrete) problem. In order to deal with the continuous problem with a computer, we need a way to discretize the continuous problem. One way is to take the derivative and replace by the finite differences. But Galerkin introduced a different idea. His idea was to choose some functions whose combination would be close to the right answer. He took some trial functions whose combination would be close to the right answer and tried to find how much of each functions go into the approximation. At first he was using a few functions to use this method. Now, a century later, thanks to the development of technology, it is possible to work with thousands of functions so these combinations of functions can give us a close approximation to the correct answer. A. Hrennikoff, R. Courant and K. Feng significantly contributed to development of the finite element method.

4.3 Finite Element Method Example

What follows is a simple 1D example of the finite element method.

First consider the following boundary value problem

$$\begin{cases} -u'' + u = f \quad \text{in } \Omega \\ u(0) = 0,\ u'(1) = b, \end{cases} \quad \text{where } f \in L^2(0,1),\ b \in \mathbb{R}. \tag{4.1}$$

The L^2 space is a Lebesgue measurable function space with L^2 norm.

The L^2 space is the set of square integrable; $(\int_{-\infty}^{\infty} |f(x)|^2\, dx < \infty)$ L^2- functions (which are the functions from the Lebesgue measure space which are square integrable) are defined on a Lebesgue measure space X.

The L^2 norm is defined as following: $\phi \in L^2(0,1)$ then $|\phi|^2 \equiv \int_0^1 |\phi|^2\, dx$

Let $V = H^1(0,1) = \{v \in H^1(0,1), \mid v(0) = 0\}$.

The Sobolev space H^1 is defined based on the inner product

$$< f, g >_{H^1} = \int_{\Omega} f(x)g(x) + \nabla f(x) \cdot \nabla g(x) dV \quad \text{where } \Omega \subset R^n \text{ and } f, g : \Omega \to R \tag{4.2}$$

Note that for $n = 1$, the inner product becomes simply $< f, g >_{H^1} = \int_a^b f(x)g(x) + f'(x)g'(x)dx$.

Based on the inner product, the H^1-norm

$$\|f\|_{H^1} = \sqrt{(f, f)_{H^1}} \tag{4.3}$$

is defined and we obtain the Sobolev space.

$$H^1(\Omega) = f : \Omega \to R \mid \|f\|_{H^1} < \infty \tag{4.4}$$

of all functions on Ω for which the H^1 -norm is finite. Therefore both the L^2 space and H^1 space form a Hilbert space (as they are both a Banach space and have an inner product).

The weak formulation is

$$u \in V, \int_0^1 (u'v' + uv)dx = \int_0^1 fvdx + bv(1), \quad \forall v \in V. \tag{4.5}$$

Existence of an unique solution is guaranteed by the Lax-Milgram Lemma which will be followed shortly but before we go into that, there are two things which need to be defined: bilinears form and coercive functions.
The bilinear form on a vector space V is defined as a function $B : V \times V \to K$ which is linear in each argument separately:

$$\begin{aligned} B(u+v,w) &= B(u,w) + B(v,w) \\ B(u,v+w) &= B(u,v) + B(u,w) \\ B(\lambda u,v) &= B(u,\lambda v) = \lambda B(u,v). \end{aligned}$$

A bilinear function ϕ on a normed space E is called coercive if there exists a positive constant K such that $\phi(x,x) \geq K\,\|x\|^2 \;\; \forall x \in E$.

Let ϕ be a bounded coercive bilinear form on a Hilbert space H. The Lax-Milgram theorem states that, for every bounded linear functional f on H, there exists a unique $x_j \in H$ such that $\; f(x) = \phi(x, x_f)$.

Now partition the domain $I = [0,1]$ into N parts as $0 = x_0 < x_1 < \cdots < x_N = 1$. We call the points $x_i, 0 \leqslant i \leqslant N$ nodes. The sub-intervals $I_i = [x_{i-1}, x_i]$, $1 \leqslant i \leqslant N$ are called elements. Denote $h_i = x_i - x_{i-1}$ and the mesh parameter $h = max_{1 \leq i \leq N} h_i$

An approximate solution will be sought in the space $V_h = \{v_h \in V \mid v_h \mid I_i \in P_1(I_i), 1 \leq i \leq N\}$ Given the properties of the Sobolev space $H^1(I)$, we also have $v_h \in C(\bar{I})$

Now we choose the basis functions as following:

For $i = 1, \cdots, N-1$:

$$\phi_i(x) = \begin{cases} (x - x_{i-1})/h_i, & x_{i-1} \leq x \leq x_i, \\ (x_{i+1} - x)/h_{i+1}, & x_i \leq x \leq x_{i+1}, \\ 0, & \text{otherwise} \end{cases} \tag{4.6}$$

For $i = N$:

$$\phi_N(x) = \begin{cases} (x - x_{N-1})/h_N, & x_{N-1} \leq x \leq x_N, \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

The defined basis functions are linearly independent and we have V_h =span$\{\phi_i,\ 1 \leq i \leq N\}$, so their weak derivatives exist. They are defined almost everywhere and are equal to constants. Hence our finite element method becomes:

$$u_h \in V_h,\ \int_0^1 (u_h' v_h' + u_h v_h)dx = \int_0^1 f v_h dx + b v_h(1),\ \ \forall v_h \in V_h \tag{4.8}$$

which, using the representation $u_h = \sum_{i=1}^{N} u_j \phi_j$, can be transformed to the linear system

$$\sum_{i=1}^{N} u_j \int_0^1 (\phi_i' \phi_j' + \phi_i \phi_j)dx = \int_0^1 f \phi_i dx + b\phi_i(1),\ \ 1 \leq i \leq N \tag{4.9}$$

This system can be written as $Au = b$, where

$$\begin{cases} u = (u_1, \dots, u_N)^T \text{is the vector of unknown coefficients} \\ b = (\int_0^1 f\phi_i dx, \dots, \int_0^1 f\phi_{N-1} dx, \int_0^1 f\phi_N dx + b)^T \text{is the load vector} \end{cases}$$

We can form a stiffness matrix which represents the system of linear equations that must be solved in order to calculate an approximate solution to the differential equation. Entries of the stiffness matrix can be calculated as $A_{ij} = \int_0^1 (\phi_i' \phi_j' + \phi_i \phi_j) dx$ using the formulae

$$\begin{aligned} &\int_0^1 \phi_i' \phi_{i-1}' dx = -\frac{1}{h}, \ \ 2 \le i \le N, && \int_0^1 (\phi_i')^2 dx = \frac{2}{h}, \quad 1 \le i \le N-1, \\ &\int_0^1 \phi_i \phi_{i-1} dx = \frac{h}{6}, \quad 2 \le i \le N, && \int_0^1 (\phi_i)^2 dx = \frac{2h}{3}, \ \ 1 \le i \le N-1 \\ &\int_0^1 (\phi_N')^2 dx = \frac{1}{h}, && \int_0^1 (\phi_N)^2 dx = \frac{h}{3} \end{aligned}$$

From above, we can obtain the stiffness matrix

$$A = \begin{bmatrix} (\frac{2h}{3} + \frac{2}{h}) & (\frac{h}{6} + \frac{1}{h}) & & & \\ (\frac{h}{6} + \frac{1}{h}) & (\frac{2h}{3} + \frac{2}{h}) & (\frac{h}{6} + \frac{1}{h}) & & \\ \ddots & \ddots & \ddots & & \\ & (\frac{h}{6} + \frac{1}{h}) & (\frac{2h}{3} + \frac{2}{h}) & (\frac{h}{6} + \frac{1}{h}) \\ & & (\frac{h}{6} + \frac{1}{h}) & (\frac{h}{3} + \frac{1}{h}) \end{bmatrix} \tag{4.10}$$

By solving $Au = b$,where A is the stiffness matrix and b is the load vector, the process ends.

In summary, the finite element method can be derived in such steps.

1. Discretise domain
2. Derive (simpler) finite element equations
3. Assemble and combine element equations
4. Apply boundary constraints
5. Solve
6. Post-processing (visualisation)

5 Method III: Spectral Methods

By taking the differentiation matrix produced through a finite difference model and taking the process to the limit by using a differentiation formula of infinite order and bandwidth, we end up with a spectral method. In practice, we cannot work with infinite matrices, so the approach is to:

1. Define a function f s.t. $f(x_j) = u_j$ for every j
2. Let $w_j = f'(x_j)$

In general we can split simple problems by examining the domain type. For periodic domains, it is usually most appropriate to create a basis using trigonometric polynomials on an equispaced grid. For non-periodic domains, we use algebraic polynomials e.g Chebyshev polynomial on irregular grids.

5.1 Fourier Spectral Method

For a periodic domain, we usually use the Fourier spectral method to calculate the numerical solution of PDEs. First we use our fundamental spatial domain $[-\pi, \pi)$ and let n be a positive even integer,

$$h = \frac{2\pi}{n}$$

Define $x_j = jh$. The grid points in this domain are

$$x_{-\frac{n}{2}} = -\pi, ..., x_0 = 0, ..., x_{\frac{n}{2}-1} = \pi - h$$

Then take the discrete Fourier transform:

$$\hat{v}_k = h \sum_{j=\frac{-n}{2}}^{\frac{n}{2}-1} e^{-ikjh} v_j$$

Take the Inverse discrete Fourier Transform:

$$v_j = \frac{1}{2\pi} \sum_{j=\frac{-n}{2}}^{\frac{n}{2}-1} e^{-ikjh} \hat{v}_k$$

As a result, we can calculate a numerical solution for the partial equation. However the implementation of the spectral method does not always use the fast Fourier transform in the process. Instead, explicit matrix multiplication may be used to calculate numerical solution of PDEs in order to reduce computational cost. It can be done by setting up matrix interpolating using the sinc function, Chebyshev function etc. Then we can derive a set of linear equations to find the numerical result. An example using such a method will also be shown later on.

5.2 Chebyshev Fourier Spectral Method

The Chebyshev spectral method can be implemented by using the fast Fourier transform method. In some cases, the Chebyshev spectral method can be used to speed up the calculation.

Domain of Chebyshev series:

$$\text{Chebyshev series in } x \in [-1, 1]$$

The Chebyshev polynomial denoted T_n is defined by

$$T_n(x) = cos(ncos^{-1}x) = cosn\theta \qquad \text{for} \quad x = cos\theta$$

In general

$$T_{n+1}(x) = \frac{1}{2}(z^{n+1} + z^{-n+1}) = \frac{1}{2}(z^n + z^{-n})(z + z^{-1}) - \frac{1}{2}(z^{n-1} + z^{1-n})$$

Recurrence relations can be set up

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

As $T_n(x)$ has exactly n degrees for every n and any polynomial with degree n can be denoted as

$$p(x) = \sum_{n=0}^{N} a_n T_n(x) \quad x \in [-1, 1]$$

$$p(x) = \sum_{n=0}^{N} a_n cosn\theta \quad \theta \in \mathbb{R}$$

Steps for Chebyshev spectral method:

1. Given data $v_0, ..., v_n$ at Chebyshev points $x_0 = 1, ..x_n = -1$. Represent the data by a vector V of length 2N where $V_{2n-j} = v_j, j = 1, 2, ..., n-1$

2. Using the fast Fourier transform,

   $$\hat{V}_k = \tfrac{\pi}{n} \textstyle\sum_{j=1}^{2n} e^{-ik\theta j}, \quad k = -n+1, ..., n$$

3. Define $\hat{W}_k = ik\hat{V}_k$ except $\hat{W}_n = 0$

4. Compute the derivative of the trigonometric interpolant P on the equispaced grid by the inverse FFT

   $$W_j = \tfrac{1}{2\pi} \textstyle\sum_{-n+1}^{n} e^{ik\theta_j} \hat{W}_k \quad j = 1, ..., 2n$$

5. Calculate the derivative of the algebraic polynomial p on the interior grid points

   $$p(x) = P(\theta) \quad \text{where } x = cos\theta$$

   $$p'(x) = \tfrac{P'(\theta)}{dx/d\theta} = \tfrac{-\sum_{n=0}^{n} na_n sinn\theta}{-(1-x^2)^{\frac{1}{2}}} = \tfrac{-P'(\theta)}{\sqrt{1-x^2}}$$

$$w_j = -\frac{W_j}{\sqrt{1-x^2}} \quad j = 1, ..., n-1$$

with the special formulas for the end points

$$w_0 = \frac{1}{2\pi}\sum_{i=0}^{n\prime} i^2\hat{v}_n \quad w_n = \frac{1}{2\pi}\sum_{i=0}^{n\prime}(-1)^{i+1}i^2\hat{v}_n$$

The Chebyshev spectral method can be used to calculate PDEs with boundary conditions. In some case, using a Chebyshev grid can reach higher accuracy than an equispaced grid.

5.3 Spectral Accuracy

One of the primary reasons for using spectral methods is due to their accuracy when compared to the finite methods. This property is known as spectral accuracy, and is due to the fast that smooth functions have rapidly decaying Fourier transforms. As a result, the aliasing errors that arise as a result of discretisation are relatively small. In general, as N increases, convergence at the rate $O(N^{-m})$ is achieved as long as the solution is infinitely differentiable.

We can see a basic example [5] below. Consider $u(x) = \exp(sin^2(x))$.

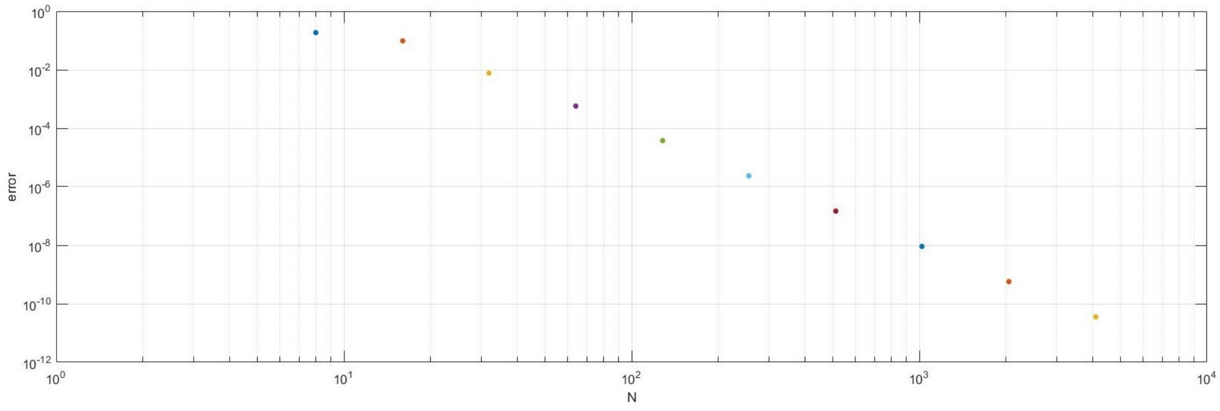


Figure 7: Convergence of fourth-order finite differences for $\exp(sin^2(x))$

We can apply both a finite difference and a spectral method over a range of N (the number of discrete points we take) and compare the error from the true value of the derivative of u. If we want to reduce the error to 10^{-10} we can see that using the fourth-order finite differences we need between 10^3 and 10^4 grid points, while for the spectral method it is between 10 and 10^2.

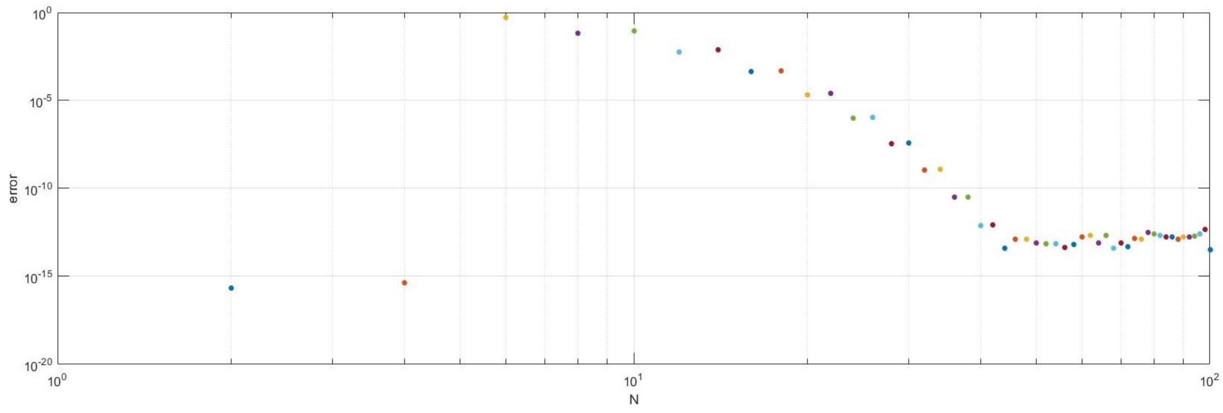


Figure 8: Convergence of spectral method for $\exp(sin^2(x))$

6 Solving Partial Differential Equations

In the following section, linear and non-linear examples will be solved using spectral methods. For the time dependence, finite difference methods such as the leap-frog and Euler time-step method will be used.

6.1 Linear

6.1.1 Wave Equation

Waves are everywhere. Most of the information that we receive comes to us in the form of waves. Waves transfer energy in different forms and the discovery of the wave equation by d'Alembert has had a significant contribution to the engineering field.

The wave equation is the important partial differential equation that describes propagation of waves with speed v.

$$\nabla^2\psi = \frac{1}{v^2}\frac{\partial^2\psi}{\partial t^2} \tag{6.1}$$

The form above gives the wave equation in three-dimensional space where ∇^2 is the Laplacian, which can also be written $v^2\nabla^2\psi = \psi_{tt}$. Now we will show how the solution of the wave equation would look using spectral method on MATLAB.

Consider the variable coefficient wave equation

$$u_t + c(x)u_x = 0, \;\; c(x) = \frac{1}{5} + sin^2(x-1) \tag{6.2}$$

for $x \in [0, 2\pi], t > 0$, with periodic boundary conditions. As an initial condition we take $u(x, 0) = exp(-100(x-1)^2)$. This function is not mathematically periodic, but it is so close to zero at the ends of the interval that it can be regarded as periodic in practice.

To construct our numerical scheme, we proceed just as we might with a finite difference approximation of a PDE. Solve the time part with the leap frog method and then use spectral methods to deal with the space dependence. A graphical solution can be created as follows.

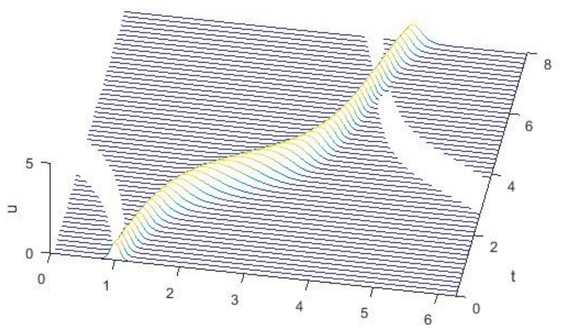


Figure 9: The numerical solution is plotted in 3D with the initial condition above.

6.1.2 Heat Equation

The $Heat$ equation describes the distribution of temperature in a period of time for some object. It is one of the most common and discussed partial differential equations. An example of the heat equation would include a rod of metal bar and the distribution of temperature inside it. Various assumptions can be made but most commonly it's said that the bar is insulated on the lateral surface and temperature can only escape on either of the ends of the bar. A heat equation belongs to class of parabolic linear equations and is often written in the form:

$$\frac{\partial U}{\partial T} = K \Delta U \tag{6.3}$$

As shown in section 3 the equation (6.3) can be transformed to simplified form, which will be dealt with using spectral methods:

$$\frac{\partial u}{\partial t} = \Delta u \tag{6.4}$$

with pre-defined initial condition:

$$u(x,0) = u_0(x) \tag{6.5}$$

Then there are 3 distinct types of boundary conditions which are considered for partial differential equations [9]. Let Ω denote the bounded domain in $\mathbb{R}^n$ (note that we are dealing with heat equation in n dimensions). Then the following conditions may be considered:

1. Dirichlet Boundary Condition:

$$u = 0 \qquad \text{on} \qquad \delta\Omega$$

2. Neumann Boundary Condition:

$$\frac{\partial u}{\partial v} = 0 \qquad \text{on} \qquad \delta\Omega$$

3. Robin Boundary Condition:

$$\alpha u + \beta \frac{\partial u}{\partial v} = 0 \qquad \alpha, \beta > 0$$

To show how to tackle the heat equation in this section we will use 1-dimensional heat equation with Dirichlet Boundary Condition and initial condition $u(x,0) = sin(x)$ - thus the problem we shall consider takes form:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \tag{6.6}$$

$$\begin{cases} u(x,0) = sin(x) \qquad \text{initial condition} \\ u(0,t) = u(0,L) = 0 \qquad \text{boundary condition} \end{cases}$$

By taking the Fourier transform in x we can rewrite above equation by:

$$\frac{\partial \hat{u}}{\partial t} = k\frac{\partial^2 \hat{u}}{\partial^2 t}$$

We use the hat symbol to denote a partial derivative's Fourier transform. For the Fourier transform of u in x following equation is obtained:

$$\frac{\partial^m \hat{u}(\omega, t)}{\partial x^m} = (i\omega)^m \hat{u}(\omega, t)$$

Thus:

$$\frac{\partial \hat{u}}{\partial t} = k(i\omega)^2 \hat{u}(\omega, t) \tag{6.7}$$

Using the Forward Euler-Method on equation (6.7) it can be written in the form:

$$\hat{u}_{i,j+1} = \hat{u}_{i,j} + kh(i\omega)^2 \hat{u}_{i,j} \tag{6.8}$$

Here h denotes time-step and the reader can see the resemblance to the finite difference method for solving the heat equation. All that is left is the visual representation which will be done by MATLAB code [13]. Upon running the code following visual representation of numerical solution is obtained:

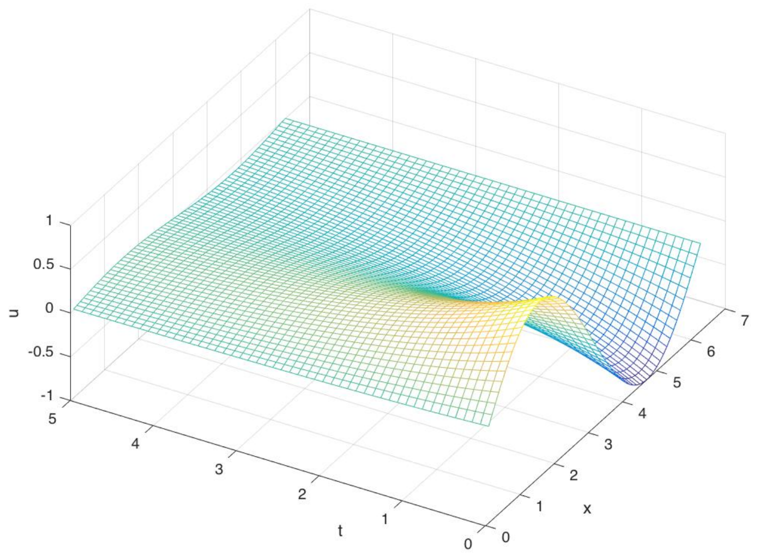


Figure 10: The numerical solution of $u(x,t)$ is plotted in 3D

The resemblance to the solution in section 3 is straightforward. By modifying the code it's easy to check solutions for other initial conditions, with periodic boundary problem. i.e. for:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \tag{6.9}$$

$$\begin{cases} u(x,0) = 5cos(x) + 3x & \text{initial condition} \\ u(0,t) = u(0,L) = 0 & \text{boundary condition} \end{cases}$$

By running the code, the visual representation is given (we just substitute $u = sinx$ by $u = 5\cos(x) + 3x$ into the initial conditions).

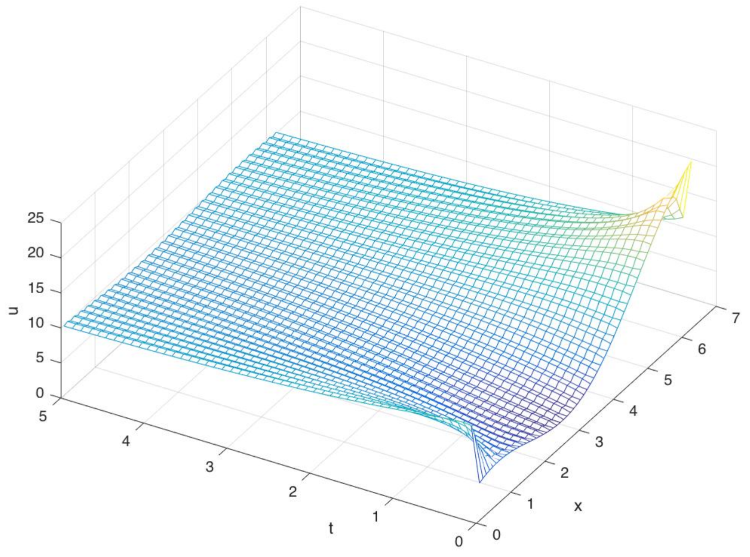


Figure 11: The numerical solution $u(x,t)$ is plotted in 3D with different initial conditions

6.1.3 Poisson Equation

The $Poisson$ equation is one of the most important Partial Differential Equations and arises in a variety of physical situations.
It can be expressed in the form:

$$\nabla^2 u = f(x,y)$$

where

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

and $f(x,y)$ is a known continuous function on a bounded, open, and connected domain Ω with a piecewise-smooth boundary. The form given above is for two dimensions, but the Poisson equation can be extended to higher dimensions [6].

One example of where the Poisson equation arises is the steady-state diffusion equation where the solution u is time-independent. This reduces it to the form

$$\nabla^2 u = 0 \tag{6.10}$$

which is Laplace's equation, a special case of the Poisson equation that was solved analytically in Section 2. Similar examples also exist in electrostatics and gravitation.

To be able to solve the Poisson equation we require boundary conditions. Having an initial condition does not make sense. For the purposes of computation, we will work with Dirichlet conditions, i.e.

$$u(x, y) = g(x, y), \qquad (x, y) \in \partial\Omega$$

where g is a function defined on the boundary of Ω.

A simple example would be the one dimensional case of the Poisson equation

$$u_{xx} = e^{5x}, \qquad -1 < x < 1, \quad u(\pm 1) = 0$$

We can solve this equation analytically to get the solution $u = 1/25\,[e^{5x} + x\sinh(5) + \cosh(5)]$. Given a suitable implementation for calculating the Chebyshev differentiation matrix D_n we can calculate the second derivative by taking the square D_N^2 [5]. It is possible to use other methods which require less computational power (such as recurrences) to calculate this matrix but for basic implementations, the square can be taken without too much concern.

The next step is to deal with the boundary conditions. We take our interior Chebyshev points $x_1, ..., x_{N-1}$ as our grid and $v = (v_1, ..., v_{N-1})^T$ as our vector of unknowns. We then proceed as follows:

1. Define $p(x)$ as the unique polynomial with $p(\pm 1) = 0$ and $p(x_j) = v_j$, $1 \le j \le N-1$.
2. Let $w_j = p''(x_j)$, $1 \le j \le N-1$

The procedure above effectively fixes v_0 and v_N at 0 and ignores w_0 and w_N. This means that we can ignore the first and last columns and rows of D_N^2. So we can now define a new matrix $\tilde{D}_N^2$ which ignores these columns and rows. Now we have reduced the problem to simply solving a system of linear equations: $\tilde{D}_N^2 v = f$.

Re-creating this in MATLAB allows us to solve this system, and we can also look at the error when comparing our numerical solution to the analytical solution. In this example, for $N = 25$ we have a maximum error of the order 10^{-14}.

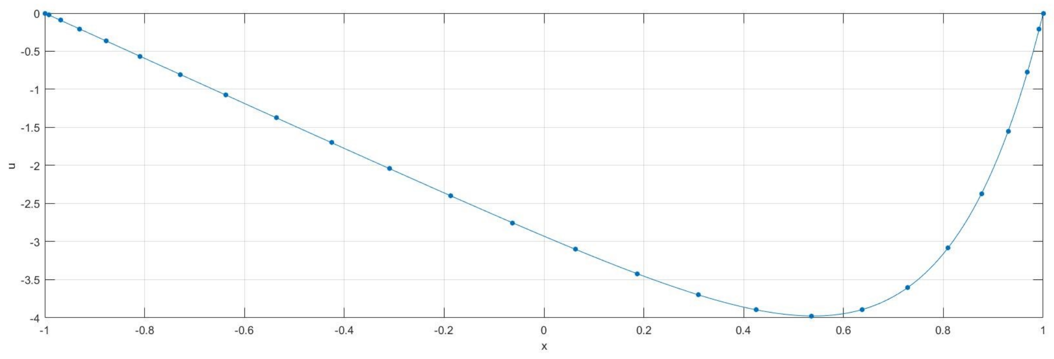


Figure 12: The numerical solution u is plotted in the interval $x \in [-1, 1]$

We can extend this method to solve the Poisson equation in more dimensions. Take, for example

$$u_{xx} + u_{yy} = 10sin(8x(y-1)), \quad -1 < x, y < 1.$$

with $u = 0$ on the boundary. We have to set up a Chebyshev grid in each direction, which is known as a tensor product grid, and use Kronecker products to solve the resulting problem. The resulting solution can be plotted as follows.

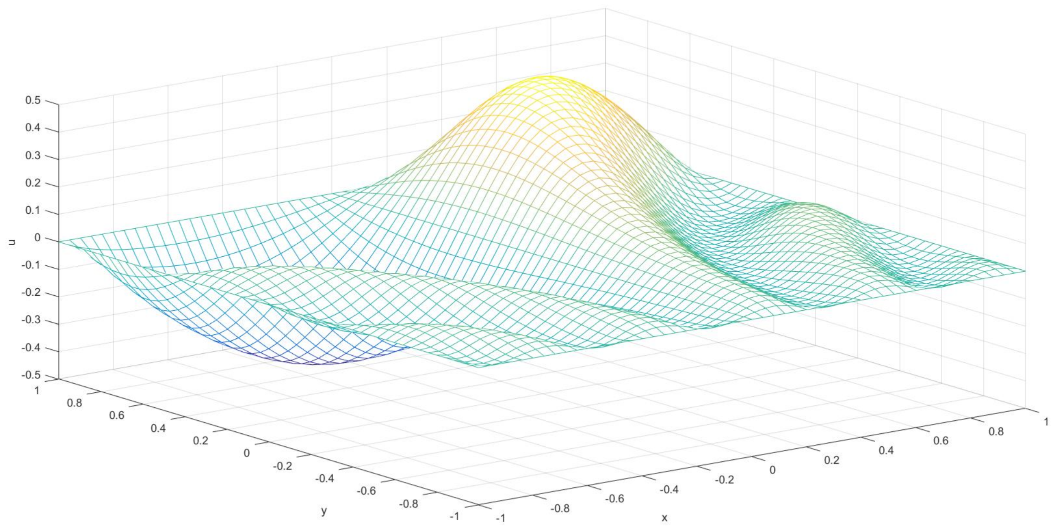


Figure 13: The numerical solution u is plotted in the interval $x, y \in [-1, 1]$

6.2 Non-Linear Equations

In this section we calculate some numerical solutions to partial differential equations with non-linear terms.

6.2.1 Inviscid Burgers' Equation

A partial differential equation of the form $u_t + (f(u))_x = 0$ is called a conservation law in which u represent the density of some quantity and f(u) associate rightward flux. Integrating with respect to t,

$$\frac{d}{dt}\int_a^b u(x)dx = f(u(a,t)) - f(u(b,t))$$

A non-linear example of the conservation law is the Inviscid Burgers' equation. It is a first order quasilinear hyperbolic equation. The equation has the form:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0$$

This equation can be used in gas dynamics and traffic flows. One crucial phenomenon that arises by the Inviscid Burgers equation is the formation shock which is discontinuity happens after a certain finite time propagating in a regular manner.

Example of inviscid Burger's equation:
Shock Wave with periodic domain

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0 \qquad 0 \leq x \leq 2\pi$$

with initial conditions:

$$u(x,0) = f(x) = \exp(-10(4x-1)^2)$$
$$u(x,t) = f(x-ut)$$

Note: Constant wave speed =1 (wave travel at a maximum wave speed u=1) m, wave speed depends on $u(x,t)$.

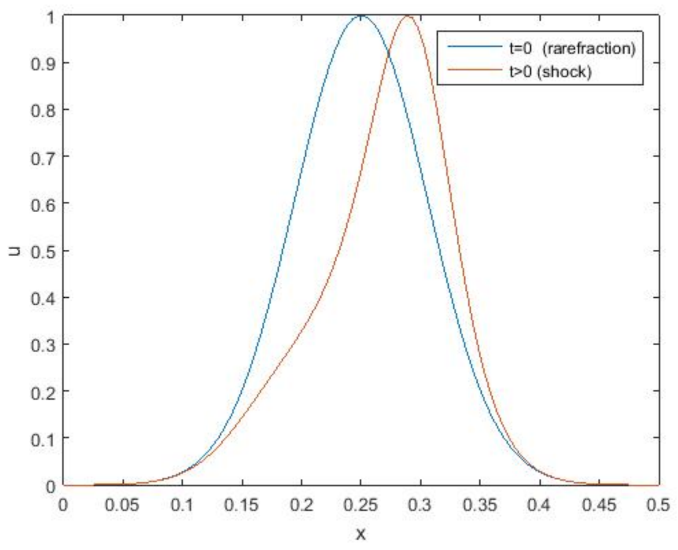


Figure 14: The analytical solution $u(x,t) = f(x - ut)$ is plotted

Let $u(x_j, t_k) = u_{j,k}\ x_j = j\Delta x,\ j = 0, 1, ..., 2n-1,\ t_k = k\Delta t,\ k = 0, 1, ..., m,$ and $\Delta t = \frac{T}{m}$
First we deal with the x dependent part by taking the discrete Fourier transform of u and the inverse Fourier transform to $\hat{u}$. First using the discrete Fourier transform

$$\hat{u}_v = \sum_{j=0}^{2n-1} u(x_j, t)\exp(-ix_j v) \text{for } v = -n+1, ..., n$$

$$\text{where } x_j = j\Delta x = \frac{j\pi}{n}$$

Then using the inverse Fourier transform:

$$u_j = \frac{1}{2n}\sum_{-n+1}^{n} \hat{u}_v \exp(ix_j v)$$

$$\text{for} \quad j = 0, 1, ..., 2n-1$$

and differentiate $u_{j,k}$with respect to x to find the numerical solution for $\hat{u_x}x$

$$\begin{aligned}\frac{\partial u_{j,k}}{\partial x} &= \frac{1}{2n}\sum_{v=-n+1}^{n} \hat{u}_v iv \exp(ix_j v)\\ &= F^{-1}(iv\hat{u}_v)\\ &= F^{-1}(ivF(u_{j,k}))\end{aligned}$$

Second we solve the time dependent part using the leap frog method

$$u_t = \frac{u_{j,k+1} - u_{j,k-1}}{2\Delta t}$$

From the equation,

$$u_t + uu_x = 0$$
$$u_{j,k+1} = u_{j,k-1} - 2\Delta t\, u_{j,k}\, F^{-1}(ivF(u_{j,k}))$$

Assume wave speed $u \approx 1$, set $t = -\Delta t$

$$u_{j,-1} = u(x, -\Delta t) = u(x + 1{*}\Delta t) = f(x - ut)$$
$$\approx f(x + \Delta t) = \exp(-10(4(x + \Delta t - 1)^2)$$

The solution can be calculated through iteration by the time independent part. The following graphs were obtained upon running the code listed in Appendix. First the 2-dimension solutions are shown.

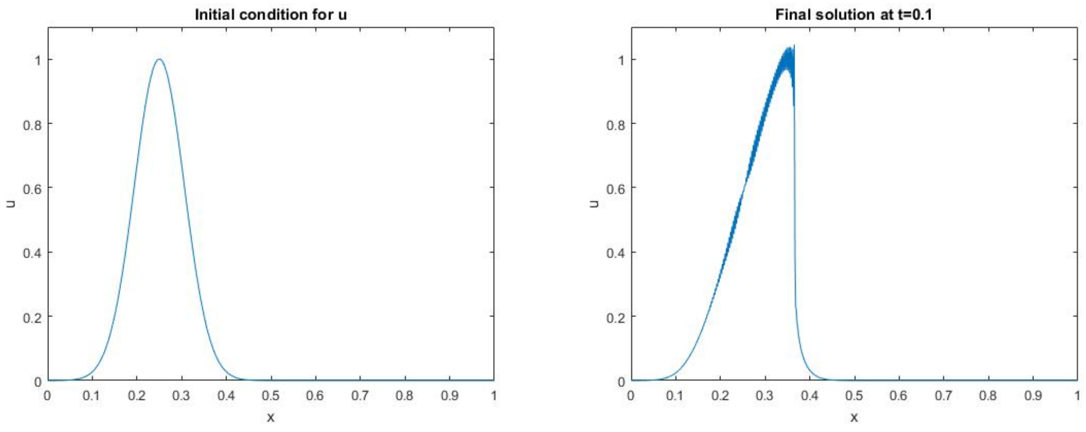


Figure 15: Initial and final condition solution for the Inviscid Burger's Equation

In the last graph, a 3D solution is shown and how it changes for different values of time. It is plotted using MATLAB with the code reference given in the Appendix.

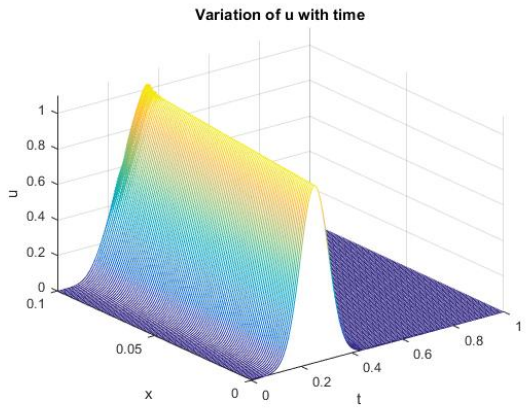


Figure 16: Non linear solution for the Inviscid Burger's Equation

6.2.2 Korteweg-de Vries Equation

The Korteweg-de Vries Equation was discovered by a phenomenon observed by a young Scottish engineer named John Scott Russell (1808-1882). He observed solitary waves (waves of translation) in the Union Canal at Hermiston which is closed to Riccarton campus of Heriot-Watt University Edinburgh.
His ideas was not recognised until the mid 1960s when applied scientists began to use modern digital computers to calculate nonlinear wave propagation. He viewed the solitary wave as a self-sufficient dynamic entity, a ”thing” displaying many properties of a particle.
The phenomenon described by Russell can be expressed by a non-linear partial differentiation equation of third order as shown below:

$$\tfrac{\partial u}{\partial t} + \epsilon u \tfrac{\partial u}{\partial x} + \tfrac{\partial^3 u}{\partial x^3} = 0$$

The Korteweg-de Vries Equation was derived by Korteweg-de Vries (1985) which describe a non-linear shallow water wave. It is a hyperbolic partial differentiation equation as illustrated in section 2.

1. Solitons waves are result from solution of kdV
2. Non linear term causes waves to steepen: u_x
3. Dispersive term cause waves to disperse u_{xxx}

Example of Korteweg-de Vries Equation: We will calculate a solution using a Fourier spectral method:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} = 0 \quad 0 \le x \le 2\pi \tag{6.11}$$

with periodic boundary conditions $u(0) = u(2\pi)$
Initial condition:

$$u(x,0) = A\,\text{sech}^2\Big(\sqrt{\frac{A}{12}}\Big(x - \pi - \frac{A}{3}t\Big)\Big) \quad A = 100$$

First, we deal with the x dependent part of u using the fast Fourier transform. Take the discrete Fourier transform of u

$$\hat{u}_v = \sum_{j=0}^{2n-1} u(x_j,t)\exp(-ix_jv) \text{ for } v = -n+1,...,n$$

$$\text{where } x_j = j\Delta x = \frac{j\pi}{n}$$

Then we take the inverse Fourier Transform:

$$u_j = \frac{1}{2n}\sum_{-n+1}^{n} \hat{u}_v \exp(ix_jv)$$

$$\text{for} \quad j = 0,1,...,2n-1$$

and differentiate $u_{j,k}$with respect to x in order to find the numerical solution u_x

$$\frac{\partial u_{j,k}}{\partial x} = \frac{1}{2n}\sum_{v=-n+1}^{n} \hat{u}_v iv\exp(ix_jv)$$

$$= F^{-1}(ivF(u_{j,k}))$$

Find u_{xxx} using the same method as above

$$u_{xxx} = \frac{\partial^3 u_{j,k}}{\partial x^3} = \frac{1}{2n} \sum_{v=-n+1}^{n} \hat{u}_v \exp(ix_j v)$$
$$= F^{-1}(iv^3 F(u_{j,k}))$$

Second, we deal with the time dependent part of u using the leap frog method

$$u_t = \frac{u_{j,k+1} - u_{j,k-1}}{2\Delta t}$$

$$u_{j,k+1} = u_{j,k-1} + 2\Delta t(\ u_{j,k}\ F^{-1}(ivF(u_{j,k})) + F^{-1}((iv)^3 F(u_{j,k}))$$

In order to remove instability for high wave number, u_{xxx} is approximate as below:

$$\sin(v^3\Delta t) \approx v^3\Delta t + \mathcal{O}(\Delta t^3)$$

as $\Delta t \to 0$
Therefore it is a good approximation Using $\sin x = x - \frac{x^3}{3!} + ...$ we get

$$u_{j,k+1} = u_{j,k-1} + 2\Delta t(\ u_{j,k}\ F^{-1}(ivF(u_{j,k})) + F^{-1}(-i\sin(v^3\Delta t)F(u_{j,k}))$$

which is a stable solution.

Another method to remove instability is to use an integrating factor. It is based on the idea that the non-linear part can be transformed to the linear part of the partial equation. From (6.11) Take Fourier Transform:

$$\hat{u_t} + \frac{i}{2}k\hat{u^2} - ik^3\hat{u} = 0$$

Multiply it with e^{ik^3t}

$$e^{ik^3t}\hat{u_t} + e^{ik^3t}\frac{i}{2}k\hat{u^2} - ik^3e^{ik^3t}\hat{u} = 0$$

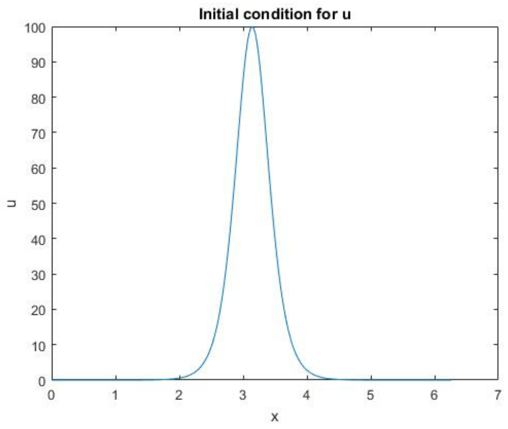
Define $\hat{U} = e^{ik^3t}\hat{u}$ with $\hat{U}_t = ik^3\hat{U} + e^{ik^3t}\hat{u}_t$ It becomes

$$\hat{U}_t + ik^3\hat{U} + \frac{i}{2}e^{ik^3t}k\hat{u^2} - ik^3\hat{U} = 0$$
$$\hat{U}_t + \frac{i}{2}e^{ik^3t}k\hat{u^2} = 0$$

which is a linear partial differentiation equation.

Continue using the first method by approximate using Taylor series
Given wave speed $u = \frac{A}{3}$, set $t = -\Delta t$

$$u_{j,-1} = u(x, -\Delta t) = u(x + \frac{A}{3}\Delta t) = f(x - ut)$$
$$= A\,\mathrm{sech}^2(\sqrt{\frac{A}{12}}(x + \frac{A}{3}\Delta t - \pi))$$



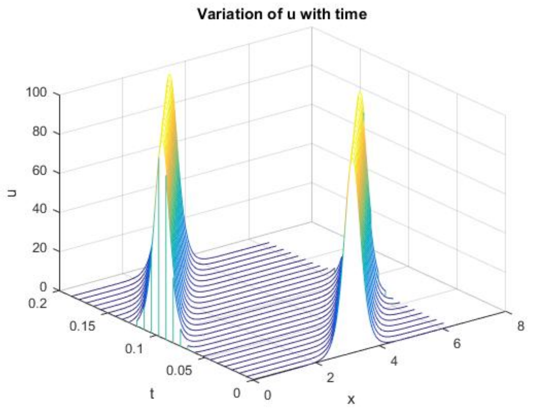


Figure 17: Initial and final condition solution for the Kdv Equation and variation of u against t

7 Conclusion

Partial differential equations can be used to describe a wide variety of phenomena such as sound, heat, electrostatics, electrodynamics, fluid flow, elasticity, quantum mechanics and even the stock market.

Numerical methods have made solving partial differential equations significantly easier. Sometimes, partial differential equations are difficult to solve analytically. The introduction of numerical methods is a significant achievement that allows people to find numerical solution of partial differential equation using computers.

In the above, we have shown three major numerical methods to solve partial differentiation equations including finite element, finite difference and spectral methods. We also have shown several famous examples of solving partial differentiation equations using the Fourier spectral method and Chebyshev differentiation matrix.

To compare between the three methods, spectral methods have a lot of advantages over the finite difference and finite element methods. Firstly, spectral methods have higher accuracy on smooth functions, achieving 10 digits of accuracy for a relatively small number of grid points. Secondly, spectral methods usually have a higher efficiency. They require less computation time, hence saving cost in calculating problems. Finally, they can achieve exponential convergence/spectral convergence as shown in previous sections.

However, spectral methods also suffers drawbacks. Firstly, programming spectral models is relatively more difficult when compared to using the finite difference method. It has to be divided into two methods to solve the time dependent and space dependent equations. Secondly, it is more costly per degree of freedom than the finite difference method. In geometries for especially complicated domains, the spectral method heavily loses its accuracy.

To continue our study, solving numerical solution using spectral method in rectangular grid and polar coordinate can be done. Moreover, instead of solving equation using the Fourier spectral method, we can try to solving equation using collocation and Tau spectral methods. More complicated equations can also be computed using spectral methods. In addition, we could also look at modern implementations of the finite difference method and finite element method. Last, a modified spectral method could also be investigated to try and overcome disadvantages of spectral methods.

8 Appendix

The following code was used for solution of heat equation in section 3. The code has all explanatory details in the comments and was done with help from the website - details in references [4].

```
function [u,err,x,t] = finite_heat(L, t_0,t_f,t_steps,x_steps)
%function takes input arguments:
%L - the domain of x, specified by initial conditions
%t_0, t_f - times between which the solution is to be found
%t_steps, x_steps - number of grid points for computation

%create the domain for x and t - the mesh grid which will be used
x = linspace(0,L,x_steps)';
t = linspace(t_0,t_f,t_steps+1);

%r (must be smaller than 1/2 for stability)
h = L/(x_steps-1);
k = (t_f-t_0)/t_steps;
r = k/h^2;

%initialize x values in the matrix x-t, column 1
for i=1:x_steps
  u(i,1) = sin(x(i));
end


%analytic solution to the initial conditions given, initialize error
err(:,1) = u(:,1) -
exp(t_0-t(1))*sin(x);

%for every initialized value of x compute it at time t
for j=1:t_steps
  %initialize row of values t, row 1
  u(1,j+1) = u(1,j) + r*(u(2,j)-2*u(1,j)+u(x_steps-1,j));

  %for each value of x add it's value at time t
  for i=2:x_steps-1
    u(i,j+1) = u(i,j) + r*(u(i+1,j)-2*u(i,j)+u(i-1,j));
  end

  %add last row of x values - for the matrix dimensions to agree
  u(x_steps,j+1) = u(x_steps,j) + r*(u(2,j)-2*u(x_steps,j)+u(x_steps-1,
      j));

  %compute error
  err(:,j+1) = u(:,j+1) - exp(t_0-t(j+1))*sin(x);
```

```
end

>> m=finite_heat(2*pi,0,5,50,10)
```

The following is coding for the Wave equation image refernce[14]

```
% p6.m - variable coefficient wave equation

% Grid, variable coefficient, and initial data:
  N = 128; h = 2*pi/N; x = h*(1:N); t = 0; dt = h/4;
  c = .2 + sin(x-1).^2;
  v = exp(-100*(x-1).^2); vold = exp(-100*(x-.2*dt-1).^2);

% Time-stepping by leap frog formula:
  tmax = 8; tplot = .15; clf, drawnow, set(gcf,'renderer','zbuffer')
  plotgap = round(tplot/dt); dt = tplot/plotgap;
  nplots = round(tmax/tplot);
  data = [v; zeros(nplots,N)]; tdata = t;
  for i = 1:nplots
    for n = 1:plotgap
      t = t+dt;
      v_hat = fft(v);
      w_hat = 1i*[0:N/2-1 0 -N/2+1:-1] .* v_hat;
      w = real(ifft(w_hat));
      vnew = vold - 2*dt*c.*w; vold = v; v = vnew;
    end
    data(i+1,:) = v; tdata = [tdata; t];
  end
  waterfall(x,tdata,data), view(10,70), colormap(1e-6*[1 1 1]);
  axis([0 2*pi 0 tmax 0 5]), ylabel t, zlabel u, grid off.
```

The following is adapt from [10] and modification and explanations have been added

```
%t=0
hold off
x = linspace(0,0.5);
u = exp(-10*(4*x - 1).^2);
%t>0 take delta t =0.05
u1 = exp(-10*(4*(x-u*0.05)-1).^2);
plot (x,u,x,u1)
legend('t=0 (rarefraction)','t>0 (shock)')
xlabel('x')
ylabel('u')

x1=0;
a = 2.*pi; % length in x-direction
x2=a;

```

```
T = 0.1; % length of time for solution

n = 2048;   % no of grid points Xn

dx = a/(2.*n); % grid spacing in x direction

dt = dx/4.;

m = round(T/dt);

t=0.; % initial time = 0

% set up mesh in x-direction:
x = linspace(x1,x2-dx, 2*n)';

% matlab stores wave numbers differently from us:
nu = [0:n  -n+1:-1]';

% define U^0 at t_k = 0
U_tk_1 = exp(-10*(4.*x-1).^2);

% define U^(-1) at t_k = -1
% using U^(-1) = U^0(x-dt/5) since c~1/5 near x=1:
U_tk_2 = exp(-10*(4.*(x + dt) -1).^2);

figure(1)
plot (x,U_tk_1,'-')
axis ([0 1 0 1.1])
title ('Initial condition for U')
xlabel ('x')
ylabel ('U')

% store solution every 10th timestep in matrix U(nxm):

U=zeros(2*n,round(m/10));
tvec=zeros(round(m/10),1);

U(:,1) = U_tk_1;
tvec(1) = t;

U_tk = zeros(2*n,1);

kk=1;
% now march solution forward in time using U_tk+1 = U_tk-1 - 2*i*dt*(c(
    x)*ifft(nu*fft(U_tk))) :
```

```

for k = 1:m-1
     t = t+dt;

     uhat = fft(U_tk_1);
     what = 1i*nu.*(uhat);
     w = real(ifft(what));
     U_tk = U_tk_2 - 2*dt*U_tk_1.*w ;
     % for next time step:
     U_tk_2 = U_tk_1;
     U_tk_1 = U_tk;

     if mod(k,10)==0
        U(:,kk+1) = U_tk;
        tvec(kk+1) = t;
        kk=kk+1;
     end

end



figure(2)
waterfall(x,tvec,U')
axis ([0 1 0 0.1 0 1.1])
title ('Variation of U with time')
xlabel ('t')
ylabel ('x')
zlabel ('U')


figure(3)
plot (x,U_tk)
axis ([0 1 0 1.1])
title('Final solution at t=0.1')
xlabel ('x')
ylabel ('U')
```

The following is adapt from [10] and modification and explanations have been added

```
x1=0;
a = 2.*pi; % length in x-direction
x2=a;

A = 100; % A = maximum amplitude of the soliton
V=A/3.; % V = speed at which the solition travels
T = 2.*pi/V; % length of time for solution
```

```
n = 128;    % no of grid points Xn
dx = a/(2.*n); % grid spacing in x direction
dt = (dx^3)/(2*pi^2)
m = round(T/dt)
t=0.; % initial time = 0

% set up mesh in x-direction:
x = linspace(x1,x2-dx, 2*n)';

% matlab stores wave numbers differently from us:
nu = [0:n  -n+1:-1]';

% define U^0 at t_k = 0
U_tk_1  = A*sech(sqrt(A/12.)*(x-pi)).^2;

initial_U  = U_tk_1;

% define U^(-1) at t_k = -1
% using U^(-1) = U^0(x+V*dt) since soliton moves at speed V and t=-dt :
% general solution: U(x,t) = A*sech(sqrt(A/12.)*(x-pi - Vt))^2
U_tk_2 =  A*sech(sqrt(A/12.)*(x + V*dt -pi)).^2;

figure(1)
plot (x,U_tk_1,'-')
%axis ([0 2*pi 0 2])
title ('Initial condition for U')
xlabel ('x')
ylabel ('U')

% store solution every 10000th timestep in matrix U(nxm):
U=zeros(2*n,round(m/10000));
tvec=zeros(round(m/10000),1);

U(:,1) = U_tk_1;
tvec(1) = t;

U_tk = zeros(2*n,1);

kk=1;
% now march solution forward in time using U_tk+1 = U_tk-1 - 2*i*dt*(c(
    x)*ifft(nu*fft(U_tk))) :

for k = 1:m-1
     t = t+dt;
    w = real(ifft(1i*nu.*(fft(U_tk_1))));
    w3 = real(ifft(-1i*sin(nu.^3*dt).*(fft(U_tk_1))));
    U_tk = U_tk_2 - 2*dt*U_tk_1.*w - 2.*w3 ;
    % for next time step:
    U_tk_2 = U_tk_1;
    U_tk_1 = U_tk;
%record every 10000 steps
    if mod(k,10000)==0
       U(:,kk+1) = U_tk;
       tvec(kk+1) = t;
       kk=kk+1;
    end

end
kk


figure(2)
waterfall(x,tvec,U')
%axis ([0 1 0 2 0 1.1])
title ('Variation of U with time')
xlabel ('x')
ylabel ('t')
zlabel ('U')


figure(3)
plot (x,U_tk,x,initial_U)
%axis ([0 2 0 1.1])
legend('Final solution at t=one period','Initial Condition at t=0')
xlabel ('x')
ylabel ('U')
```

The following program has been adapted from [5] with a different function to illustrate the differences in accuracy between finite order and spectral methods.

```
% Convergence of fourth-order finite differences
% Set up grid in [-pi,pi] and function u(x):
Nvec = 2.^(3:12);
clf, subplot('position',[.1 .4 .8 .5])
for N = Nvec
h = 2*pi/N; x = -pi + (1:N)'*h;
u = exp((sin(x)).^2);
uprime = 2*sin(x).*\(x).*u;
% Construct sparse fourth-order differentiation matrix:
e = ones(N,1);
D = sparse(1:N,[2:N 1],2*e/3,N,N)...
- sparse(1:N,[3:N 1 2],e/12,N,N);
```

```
D = (D-D')/h;
% Plot max(abs(D*u-uprime)):
error = norm(D*u-uprime,inf);
loglog(N,error,'.','markersize',15), hold on
end
grid on, xlabel N, ylabel error

% Convergence of periodic spectral method
% Set up grid:
clf, subplot('position',[.1 .4 .8 .5])
for N = 2:2:100;
h = 2*pi/N;
x = -pi + (1:N)'*h;
u = exp((sin(x)).^2);
uprime = 2*sin(x).*(x).*u;
% Construct spectral differentiation matrix:
column = [0 .5*(-1).^(1:N-1).*cot((1:N-1)*h/2)];
D = toeplitz(column,column([1 N:-1:2]));
% Plot max(abs(D*u-uprime)):
error = norm(D*u-uprime,inf);
loglog(N,error,'.','markersize',15), hold on
end
grid on, xlabel N, ylabel error
```

Following code was used for solving heat equation with spectral method [13]:

```

%Solving Heat Equation using pseudo-spectral and Forward Euler
%u_t= \alpha*u_xx
%BC= u(0)=0, u(2*pi)=0
%IC=sin(x)
clear all; clc;
%Grid
N = 64;          %Number of steps
h = 2*pi/N;      %step size
x = h*(1:N);     %discretize x-direction
alpha = .5;       %Thermal Diffusivity constant
t = 0;
dt = .001;
%Initial conditions
v = sin(x);
k=(1i*[0:N/2-1 0 -N/2+1:-1]);
k2=k.^2;
%Setting up Plot
tmax = 5; tplot = .1;
plotgap= round(tplot/dt);
nplots = round(tmax/tplot);
```

```
data = [v; zeros(nplots,N)]; tdata = t;
for i = 1:nplots
    v_hat = fft(v);  %Fourier Space
    for n = 1:plotgap
        v_hat = v_hat+dt*alpha*k2.*v_hat; %FE timestepping
    end
    v = real(ifft(v_hat)); %Back to real space
    data(i+1,:) = v;
    t=t+plotgap*dt;
    tdata = [tdata; t]; %Time vector
end
%Plot using mesh
mesh(x,tdata,data), grid on,
view(-60,55), xlabel x, ylabel t, zlabel u, zlabel u
```

The following program has been adapted from [5] with a different function to solve the 1D and 2D Poisson equations.

```
  N = 25;
  [D,x] = cheb(N);
  D2 = D^2;
  D2 = D2(2:N,2:N);                         % boundary conditions
  f = exp(5*x(2:N));
  u = D2\f;                                 % Poisson eq. solved here
  u = [0;u;0];
  clf, subplot('position',[.1 .4 .8 .5])
  plot(x,u,'.','markersize',16)
  xx = -1:.01:1;
  uu = polyval(polyfit(x,u,N),xx);          % interpolate grid data
  line(xx,uu)
  grid on
  exact = ( exp(5*xx) - sinh(5)*xx - h(5) )/25;
  max_error = norm(uu-exact,inf)
```

The cheb function above is defined as follows.

```
% CHEB  compute D = differentiation matrix, x = Chebyshev grid
  function [D,x] = cheb(N)
  if N==0, D=0; x=1; return, end
  x = cos(pi*(0:N)/N)';
  c = [2; ones(N-1,1); 2].*(-1).^(0:N)';
  X = repmat(x,1,N+1);
  dX = X-X';
  D  = (c*(1./c)')./(dX+(eye(N+1)));          % off-diagonal entries
  D  = D - diag(sum(D'));                      % diagonal entries
```

References

[1] Lawrence C. Evans *Partial Differential Equations*(2010)American Mathematical Society P.3-6

[2] Evans G., Blackledge and J. and Yardley. P. (2000), *Numerical Methods for Partial Differential Equations*. Leicester, UK: Springer, p. 29, 34-35

[3] Coleman M. (2004), *An Introduction to Partial Differential Equations with Matlab*. Connecticut, US: Chapman & Hall/Crc, p. 540, 553

[4] Math at University of Toronto. 2014. *matlab *.m files to solve the heat equation*. [Online]. Available at: http://www.math.toronto.edu/mpugh/Teaching/AppliedMath/Spring99/heat.html. [Accessed 1 June 2016].

[5] Trefethen L. (2000). *Spectral Methods in MATLAB*. [Online]. Available at: http://epubs.siam.org/doi/book/10.1137/1.9780898719598 [Accessed 28 May 2016].

[6] Iserles A. (1996). *A First Course in the Numerical Analysis of Differential Equations*. Cambridge, UK: Cambridge University Press, p. 112-115

[7] Trefethen L. *Inviscid Burger's Equation* Available at: https://people.maths.ox.ac.uk/trefethen/pdectb/burgers2.pdf [Accessed 6 June 2016]

[8] Hunt R. E. (2002). *Poisson's Equation* Available at: http://www.damtp.cam.ac.uk/user/reh10/lectures/nst-mmii-chapter2.pdf [Accessed 6 June 2016]

[9] Wei-Ming N. (2011). *The Mathematics of Diffusion* Available at: http://epubs.siam.org/doi/pdf/10.1137/1.9781611971972.ch1

[10] Dr. Louise Olsen-Kettle. *Non Linear Partial Differential Equations*. [Online]. Available at: https://espace.library.uq.edu.au/view/UQ:239427/Lectures_Book.pdf [Accessed 7 June 2016]

[11] Klaus B. (2000). *The Korteweg-de Vries Equation: History, exact Solutions, and graphical Representation* [Online]. Available at: http://people.seas.harvard.edu/ jones/solitons/pdf/025.pdf

[12] Brezis H. and Browder F. (1998). Partial Differential Equations in the 20th Century. *Advances in Mathematics*. [Online]. 135, p.76-144. [Accessed 8 June 2016] Available at: http://www.math.ualberta.ca/ fazly/files/HistoryPDE.pdf

[13] University of Michigan. (2014). *Parallel Spectral Methods*. [Online] [Accessed 9 June 2016] Available at: https://github.com/openmichigan/PSNM/blob/master/ExamplesInMatlab /Programs/Heat_Eq_1D_Spectral_FE.m

[14] Lloyd N. Trefethen. (2000) *Spectral Methods in MATLAB*. SIAM, Philadelphia [Online] Available at: https://people.maths.ox.ac.uk/trefethen/spectral.html [Accessed 9 June 2016]

[15] Vietnamese-German University *Sobolev space* [Online]. Available at: http://www.vgu.edu.vn/fileadmin/pictures/studies/master/compeng/study_subjects/modules/math/notes/chapter-03.pdf [Accessed 7 June 2016]

[16] Rowland, Todd *Bilinear Forms* [Online] Available at: http://mathworld.wolfram.com/BilinearForm.html [Accessed 7 June 2016]

[17] Stover, Christopher and Weisstein, Eric W *Lax-Milgram Theorem*. [Online]. Available at: http://mathworld.wolfram.com/Lax-MilgramTheorem.html [Accessed 7 June 2016]

[18] Gilbert Strang. (2012) *Serious Science*. [Online]. Available at : https://www.youtube.com/watch?v=WwgrAH-IMOk [Accessed 7 June 2016]

[19] Ross, C.T.F. (1996) *Finite Element Techniques in Structural Mechanics*, Woodhead Publishers [Online]. Available at: https://www.youtube.com/watch?v=lrpj3cZrKn4 [Accessed 8 June 2016]

[20] McMaster University *Finite Elements* [Online]. Available at: http://ms.mcmaster.ca/∼bprotas/MATH745b/fem_01_4.pdf [Accessed 8 June 2016]

Special thanks to Prof. Grigoris Pavliotis for his assistance and guidance through the project.