A Review of the Viscous Burgers' Equation with a focus on Numerical Solutions

by

Jarren Ralf

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Bachelor of Science (Honours)

in the
Department of Mathematics
Faculty of Science and Horticulture

KWANTLEN POLYTECHNIC UNIVERSITY
Spring 2019

Approval

Name:	**Jarren Ralf**
Degree:	**Bachelor of Science (Honours) (Applications of Mathematics)**
Title:	**A Review of the Viscous Burgers' Equation with a focus on Numerical Solutions**
Supervisory Committee:	**Louis Saumier** Senior Supervisor Professor
Date Approved:	**August 8, 2020**

Abstract

The viscous Burgers' equation is a nonlinear Partial Differential Equation (PDE) used in the study of fluid dynamics, traffic flow, and shock theory. Applying the Cole-Hopf transformation converts it into the diffusion equation, which means an exact solution can be found under certain conditions. Nonlinear PDEs are often difficult to solve so that makes Burgers' equation a mathematically significant starting point when developing and testing new nonlinear numerical solvers. In this work, analytical solutions will be presented, with a focus on the Fourier series approximation. Multiple numerical approaches will be shared, including finite differences, finite elements, and spectral methods. A Crank-Nicolson scheme will be highlighted, including an implementation and numerical results. In addition, a finite element approach will be mapped out in detail, with results.

Keywords: Burgers' equation; Cole-Hopf transformation; diffusion equation; finite differences; finite elements; numerical solutions

Dedication

To my mom Brenda, my dad Jan, and my sister Kristen for their overwhelming support.

Acknowledgements

I would like to thank all of my instructors at Kwantlen Polytecnic University (KPU) over the years. I am grateful for those who have helped me to discover and cultivate my mathematical interests. A special thanks goes out to Louis Saumier for his passionate guidance with this project and many others.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Significant work has been done studying non-linear Partial Differential Equations (PDEs) in the context of fluid mechanics. One particular equation that has been examined in great depth is the famous Burgers' equation. This work will be a brief review of some literature, sharing both historical context and significance. Furthermore, analytical and numerical solutions, for a range of boundary and initial conditions, will be briefly described. In order to properly introduce the focus of the paper, a theoretical background of fluid mechanics and the Navier-Stokes equations will be presented in Chapter 2.

Burgers' equation was first written down by a man named Harry Bateman in 1915 [4], in the form

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \kappa\frac{\partial^2 u}{\partial x^2}, \quad 0 < x < L, \quad 0 < t < \tau, \tag{1.1a}$$

with the following initial and boundary conditions

$$u(x,0) = \psi(x), \quad 0 < x < L, \tag{1.1b}$$

$$u(0,t) = \zeta_1(t), \quad u(L,t) = \zeta_2(t), \quad 0 < t < \tau, \tag{1.1c}$$

where u is the velocity, x and t are the spacial and time coordinates, and κ is the coefficient of dissipation (assumed always to be positive). Notice that ψ, ζ_1, and ζ_2 are arbitrary functions which can be chosen based on the desired conditions. This equation will be elaborated on in Chapter 3.

Johannes Martinus Burgers was born and educated in the Netherlands, which included earning his PhD in 1918 under the supervision of Paul Ehrenfest [18]. Burgers' dissertation at the time included research on the quantum theory of the atom, motivated by the recent advances on the subject by Niels Bohr. It was many years later in 1948, when Burgers went on to use equation (1.1a) to model turbulence [7]. Due to his significant contribution to the subject of fluid mechanics, the equation now bears his name.

There are several ways to solve Burgers' equation analytically, as we will see in Chapter 4, but one such approach stands out. Julian David Cole and Eberhard Hopf independently discovered a transformation that linearizes Burgers' equation into the Diffusion equation. It is now called the Cole-Hopf transformation (more on this in Section 4.1.)

Figure 1.1: From left to right: Harry Bateman, Johannes Burgers, Paul Ehrenfest, Niels Bohr, Eberhard Hopf, and Julian Cole.

Most PDEs don't have an analytical solution, and hence a numerical approach has become a strong focus of the subject. This will be the core concept in Chapter 5. Finite difference schemes will be discussed, along with finite elements and spectral methods.

Chapter 6 will contain the results of the numerical solutions. They will be compared to each other and to the exact solution[1].

The final Chapter, Chapter 7, will give a very brief overview and summary of my senior project.

All of the matlab code used in the creation of this project is given in Appendix A.

[1]The exact solution will of course be an approximation using a finite Fourier series. The true solution requires infinitely many terms in the sum.

Chapter 2

Theoretical Background

Fluid mechanics is the study of moving gases and liquids. The term gas refers to a material that allows for substantial change in density given an arbitrary region of space within the substance. For clarification, this means the distances between molecules of a gas are, on average, orders of magnitude larger than that of a liquid, which therefore makes it easier to distinguish gases from liquids. The process of differentiating liquids from solids however, is a little more refined. The main consideration would be the tangential shear resistance. If an object can be subjected to this type of force and still maintain one piece, then it would be classified as a liquid; in contrast a solid would break into two or more pieces.

The second word in the classification of this area of study is mechanics, which in general refers to the study of movement. Motion of this type would be under the designation of Newtonian mechanics. This is because the classical laws of physics derived largely in part by Sir Isaac Newton, are the governing factors of fluids. There are several properties that are important when describing the movement of fluids. The most relevant for this literature review are viscosity, turbulence, and compressibility.

Viscosity describes the internal friction of a fluid. It manifests as a resistance to movement down an incline. If you analyze a slice of fluid (call it a layer) and observe the effects of layers with varying velocities moving past each other, the fast layer will try to curtail the speed of the slower layer with friction. This is a depiction of viscosity. This effect is described in equations using a dimensionless quantity called the Reynolds number, usually denoted Re. When the Reynolds number of a substance is low, the fluid would display signs of being highly viscous, observable as smooth laminar-type motion.

In contrast, high Reynolds numbers are reserved for turbulent phenomena when there is a lot of chaotic movement in the substance. Thus, turbulence is observed in gases and fast flowing liquids. This behavior spawns from tumultuous change in pressure. Observe in Figure 2.1 part (b) below, that the turbulent flow has layers of fluid flows that intersect with one another, which is an example of the chaotic behavior that is tough to model.

The effect of pressure to change the size, or region of a substance, is called compressibility. This can be directly described by a change in density, as we will see in the following

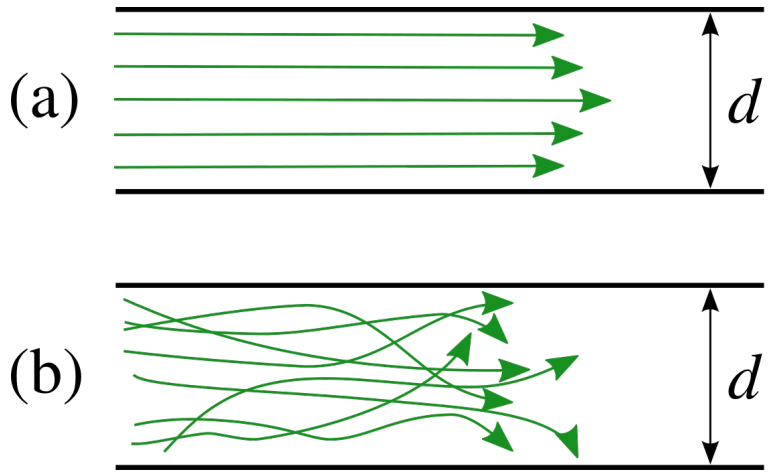

Figure 2.1: Top: Laminar Flow; Bottom: Turbulent Flow

derivation. An example of compressible fluid would be gases because the particles in a gas have the freedom to move around without being constrained to one another. An incompressible fluid on the other hand is something that keeps its density constant, which includes most liquids. In terms of calculus, compressibility can be described by a quantity called the divergence.

2.1 The Navier-Stokes Equations

This section will begin with a *brief* derivation of the Navier-Stokes equations by means of summarizing the work done by Alexandre Joel Chorin and Jerrold E. Marsden in *A Mathematical Introduction to Fluid Mechanics* [8].

The Navier-Stokes equations are derived from three fundamental principles of physics, namely the conservation of mass, balance of momentum (Newton's second law), and conservation of energy.

2.1.1 Conservation of Mass

Consider some region in space, we will call it $D \in \mathbb{R}^3$ and suppose $\mathbf{x}$ is some point in D. The velocity of a particle through position $\mathbf{x}$ at time t will be denoted by $\mathbf{u}(\mathbf{x}, t)$. Allow $\rho(\mathbf{x}, t)$ to denote the mass density of the fluid. If W is any subregion of D, then the mass of fluid in W can be expressed as

$$m(W, t) = \int_W \rho(\mathbf{x}, t)\, \mathrm{d}V \tag{2.1}$$

for some time t.

The rate of change of mass in W (2.1), given that we fix the subregion W in space, is

$$\frac{\mathrm{d}}{\mathrm{d}t}m(W,t) = \frac{\mathrm{d}}{\mathrm{d}t}\int_W \rho(\mathbf{x},t)\,\mathrm{d}V = \int_W \frac{\partial\rho}{\partial t}(\mathbf{x},t)\,\mathrm{d}V.$$

We will use the notation ∂W to describe the boundary of the region W. Suppose that $\mathbf{n}$ denotes the unit outward normal vector, then the mass flow rate per unit area is $\rho\mathbf{u}\cdot\mathbf{n}$ (see Figure 2.2).

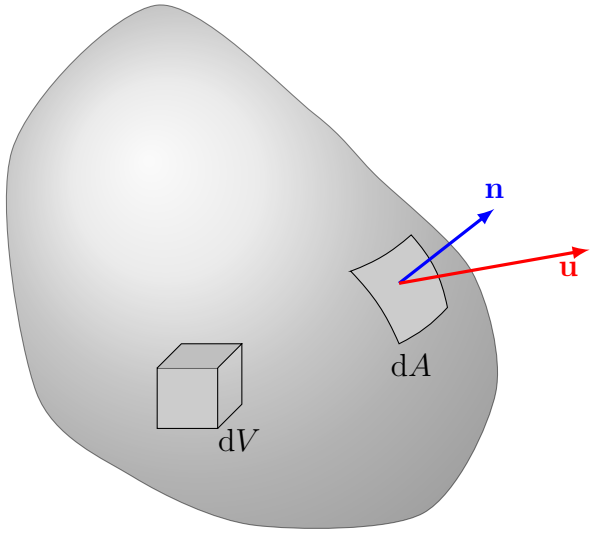

Figure 2.2: The normal vector with respect to the surface W relates to the flow rate of the mass crossing the boundary, ∂W, by $\rho\mathbf{u}\cdot\mathbf{n}$ per unit area.

This leads to the integral form of the law of conservation of mass,

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_W \rho\,\mathrm{d}V = -\int_{\partial W}\rho\mathbf{u}\cdot\mathbf{n}\,\mathrm{d}A. \tag{2.2}$$

Recall that the divergence of $\mathbf{u}$ describes the flux in a small region of space. Given this, we can see that equation (2.2) is equivalent to

$$\int_W \left[\frac{\partial\rho}{\partial t} + \mathrm{div}(\rho\mathbf{u})\right]\mathrm{d}V = 0$$

by the divergence theorem. Again this is also equivalent to

$$\frac{\partial\rho}{\partial t} + \mathrm{div}(\rho\mathbf{u}) = 0 \tag{2.3}$$

because it holds for all W. This is one of the forms that conservation of mass is expressed as, called the differential form.

2.1.2 Balance of Momentum

To formulate Newton's second law, we must first be able to describe the path of a particle in space. Naturally we start with the position as a function of time, denoted $\mathbf{x}(t) = (x(t), y(t), z(t))$. Obviously velocity is a time derivative of this quantity, as shown below

$$\mathbf{u}(x(t), y(t), z(t), t) = (\dot{x}(t), \dot{y}(t), \dot{z}(t)) = \mathbf{u}(\mathbf{x}(t), t).$$

Taking one more time derivative leads us to the acceleration of the particle, displayed in the form below

$$\mathbf{a}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{u}(x(t), y(t), z(t), t) = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{u}(\mathbf{x}(t), t).$$

Notice the above equation requires us to use the chain rule from calculus, so now the acceleration becomes

$$\mathbf{a}(t) = \frac{\partial \mathbf{u}}{\partial x}\dot{x} + \frac{\partial \mathbf{u}}{\partial y}\dot{y} + \frac{\partial \mathbf{u}}{\partial z}\dot{z} + \frac{\partial \mathbf{u}}{\partial t}.$$

Introducing a re-parameterization on $\mathbf{u}$ the following can be defined

$$\mathbf{u}(x, y, z, t) = (u(x, y, z, t), v(x, y, z, t), w(x, y, z, t)).$$

Now reformulating the derivatives using Euler's notation, as well as the equation above, leads us to

$$a(t) = u\mathbf{u}_x + v\mathbf{u}_y + w\mathbf{u}_z + \mathbf{u}_t.$$

Finally, the acceleration can be written in a more compact way using the gradient operator, ∇, as shown below

$$\mathbf{a}(t) = \partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u}$$

where

$$\partial_t \mathbf{u} = \frac{\partial \mathbf{u}}{\partial t}, \quad \text{and} \quad \mathbf{u} \cdot \nabla = u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} + w\frac{\partial}{\partial z}.$$

The material derivative is an operator defined by

$$\frac{D}{Dt} = \partial_t + \mathbf{u} \cdot \nabla. \tag{2.4}$$

To put it in basic terms, this operator is a way to describe the temporal rate of change with respect to some quantity of a particular fluid parcel moving with the fluid.

We now have a description of the acceleration, the next step is to derive an expression for the force. The first step in this process is to introduce pressure, $p(\mathbf{x}, t)$. Now imagine some smooth surface in our fluid, which we will call S. The force across S per unit area will be in terms of the pressure and unit normal, $p(\mathbf{x}, t)\mathbf{n}$ (see Figure 2.3).

Now imagine the pressure as a way of exerting a force on the boundary of the surface S, the notation for this will be $\mathbf{S}_{\partial W}$. In integral form, the total force exerted on the surface

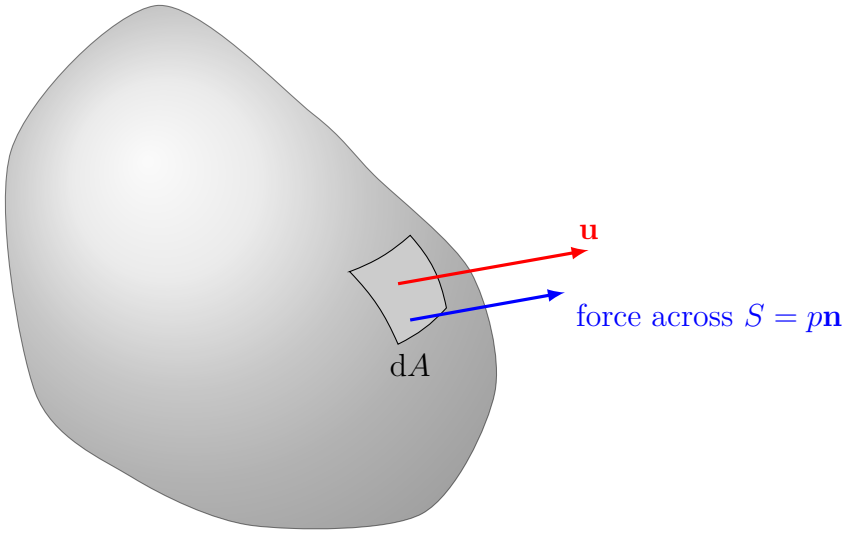

Figure 2.3: Pressure forces across a surface S.

would be

$$\mathbf{S}_{\partial W} = -\int_{\partial W} p\mathbf{n}\,\mathrm{d}A,$$

where the sign is negative because the direction is opposite compared to the outward facing normal. Next we apply the divergence theorem, but first we must consider any fixed vector in space, say $\mathbf{e}$, then we can express the force and the dot product of vector $\mathbf{e}$ as the following

$$\mathbf{e}\cdot\mathbf{S}_{\partial W} = -\int_{\partial W} p\mathbf{e}\cdot\mathbf{n}\,\mathrm{d}A = -\int_W \mathrm{div}(p\mathbf{e})\,\mathrm{d}V = -\int_W (\mathrm{grad}\,p)\cdot\mathbf{e}\,\mathrm{d}V$$

where grad is the gradient operator, similarly denoted ∇. This implies that

$$\mathbf{S}_{\partial W} = -\int_W \mathrm{grad}\,p\,\mathrm{d}V.$$

Allow $\mathbf{b}(\mathbf{x},t)$ to denote the given body force per unit mass. The total body force is

$$\mathbf{B} = \int_W \rho\mathbf{b}\,\mathrm{d}V.$$

We now have that

$$\text{force per unit volume} = -\mathrm{grad}\,p + \rho\mathbf{b}$$

on any piece of material, therefore by Newton's second law we arrive at

$$\rho\frac{D\mathbf{u}}{Dt} = -\mathrm{grad}\,p + \rho\mathbf{b},$$

the differential form of balance of momentum. It can be shown that this is equivalent to the following expression,

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{W_t}\rho\mathbf{U}\,\mathrm{d}V = S_{\partial W_t} + \int_{W_t}\rho\mathbf{b}\,\mathrm{d}V, \tag{2.5}$$

which is the law of balance of momentum in it's integral form.

2.1.3 Conservation of Energy

Since we are about to evoke the conservation of energy now, we start with the kinetic energy for fluid moving within a subregion W,

$$E_{\text{kinetic}} = \frac{1}{2}\int_W \rho\|\mathbf{U}\|^2\,\mathrm{d}V,$$

where $\|\mathbf{U}\|^2 = (u^2 + v^2 + w^2)$ is the squared length of the vector $\mathbf{u}$. Next, we are required to consider what the total energy of the system must be

$$E_{\text{total}} = E_{\text{kinetic}} + E_{\text{internal}}.$$

The internal energy is an acknowledgement of the internal molecular vibrations that exist. Whereas the kinetic energy is due to the motion of the fluid itself.

The following result follows from computing the rate of change of kinetic energy

$$\frac{\mathrm{d}}{\mathrm{d}t}E_{kinetic} = \int_{W_t}\rho\left(\mathbf{u}\cdot\left(\frac{\partial\mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right)\right).$$

From physics we also know that the rate of change of energy must be the same as the rate of which work is done. This leads to the result

$$\frac{\mathrm{d}}{\mathrm{d}t}E_{total} = \frac{\mathrm{d}}{\mathrm{d}t}\int_{W_t}\rho\mathbf{u}\cdot\mathbf{b}\,\mathrm{d}V - \int_{\partial W_t} p\mathbf{u}\cdot\mathbf{n}\,\mathrm{d}A.$$

From this, Euler's equations for isentropic flow, which is flow through a narrow opening without increase or decrease in entropy, can be written down

$$\begin{aligned}\frac{\partial\mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} &= -\nabla w + \mathbf{b},\\ \frac{\partial\rho}{\partial t} + \operatorname{div}(\rho\mathbf{u}) &= 0\end{aligned}$$

in D, and where w is the enthalpy per unit mass, which is related to the first law of thermodynamics. Enthalpy is a measure of the total heat content in a system.

At this stage, the force on a supposed surface S per unit area needs to be formulated as the expression

$$-p(\mathbf{x},\,\mathbf{t})\mathbf{n} + \boldsymbol{\sigma}(\mathbf{x},t)\cdot\mathbf{n},$$

where $\boldsymbol{\sigma}$ is a matrix called the stress tensor. Newton's second law brings us to

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{W_t}\rho\mathbf{u}\,\mathrm{d}V = -\int_{\partial W_t}(p\cdot\mathbf{n}-\boldsymbol{\sigma}\cdot\mathbf{n})\,\mathrm{d}A.$$

Without giving the derivation, two constants will be presented. Based on the system $\boldsymbol{\sigma}$, we get the following equation

$$\delta_i = \lambda(d_1+d_2+d_3)+2\mu d_i, \quad i=1,2,3,$$

where δ_i are the eigenvalues of $\boldsymbol{\sigma}$, and d_i are the eigenvalues of the symmetric part of $\nabla\mathbf{u}$. Notice this now gives us a relationship between the constants λ and μ. These scalars represent two different coefficients of viscosity. These quantities are related to the Reynolds number introduced in Section 2. They are a way of describing the internal friction in the fluid.

In preparation for the final aspect of this derivation, the transport theorem will be presented. For any function f of $\mathbf{x}$ and t,

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{W_t}\rho f\,\mathrm{d}V = \int_{W_t}\rho\frac{Df}{Dt}\,\mathrm{d}V \tag{2.6a}$$

or

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{W_t} f\,\mathrm{d}V = \int_{W_t}\left(\frac{\partial f}{\partial t}+\mathrm{div}(f\mathbf{u})\right)\mathrm{d}V. \tag{2.6b}$$

From (2.3), (2.5), and (2.6), and the divergence theorem, we arrive at the Navier-Stokes equations,

$$\rho\frac{D\mathbf{u}}{Dt} = -\nabla p+(\lambda+\mu)\nabla(\mathrm{div}\,\mathbf{u})+\mu\nabla^2\mathbf{u}, \tag{2.7}$$

where

$$\nabla^2\mathbf{u} = \left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\right)\mathbf{u}$$

is the Laplacian of $\mathbf{u}$.

Chapter 3

Burgers' Equation

Observing the system of equations (2.7), given at the end of Section 2.1.3, we make the simplifying assumption that our fluids will not have changing material density. Picture a spherical neighbourhood around a molecule such that other molecules do not ever cross the boundary of this neighbourhood. If this condition holds for all small regions within a fluid, then that fluid is incompressible. Therefore the divergence is zero. In addition, assuming that pressure is negligible our system simplifies to the following

$$\rho \frac{D\mathbf{u}}{Dt} = \mu \nabla^2 \mathbf{u}.$$

Now we divide both sides by density and make the substitution that $\kappa = \frac{\mu}{\rho}$ leaving us with

$$\frac{D\mathbf{u}}{Dt} = \kappa \nabla^2 \mathbf{u}.$$

Recall that $\frac{D}{Dt}$ denotes the material derivative operator (2.4) defined previously. Substituting it in brings us to

$$(\partial_t + \mathbf{u} \cdot \nabla)\mathbf{u} = \kappa \nabla^2 \mathbf{u}.$$

Before moving forward, we will consider only one spatial dimension. The implication of this is not only that the vector-valued function $\mathbf{u}$ becomes a real-valued function u, but the gradient and Laplacian become the first and second derivative respectively. Finally we arrived at the viscous Burgers' equation

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = \kappa \frac{\partial^2 u}{\partial x^2}$$

or expressed with different notation

$$u_t + u u_x = \kappa u_{xx}, \tag{3.1}$$

where κ is the dissipation constant and can be rewritten in terms as the Reynolds number like so

$$\kappa = \frac{1}{Re}.$$

The Reynolds number is defined in [8] by

$$Re = \frac{\mathcal{L}\mathcal{U}}{\nu}$$

where $\mathcal{L}$ is a characteristic length, $\mathcal{U}$ is a characteristic velocity, ν is the coefficient of kinetic viscosity defined by μ/ρ_0, and ρ_0 is a constant density for an incompressible flow (recall from Section 2.1.3 where μ is defined as a coefficient of viscosity). As specified in [8] the quantities $\mathcal{L}$ and $\mathcal{U}$ are "chosen in a somewhat arbitrary way." Chorin and Marsden derive a dimensionless version of the Navier-Stokes equation where the definition of the Reynolds number arises.

To clarify, in this paper whenever we refer to the viscous Burgers' equation or Burgers' equation, we are talking about (3.1).

3.1 Physical Significance

This section is inspired in part by the great work by Bronkile *et al* [6] in their research paper titled "A systematic literature review of Burgers' equation with recent advances." As suggested in their paper, the articles written on Burgers' equation seem to lack historical significance. Bronkile *et al* present many different areas that Burgers' equation has been used. Some of those examples will be presented below.

Burgers' equation is used often as a simplification of the Navier-Stokes equations where it is sometimes referred to as a *toy model* [20]. A toy model is a concept that is perhaps better formulated than is implied. It is a model that deliberately makes simplifications, often ignoring aspects of a problem, in order to observe certain characteristics. As seen in the derivation above, Burgers' equation completely lacks the ability to model pressure.

The first and most practical reason for the significance of Burgers' equation is because it is a nonlinear PDE with a well known analytical solution in certain cases. There are several numerical solvers for linear systems of PDEs, but nonlinearity still poses problems. In order to develop a nonlinear numerical solver for PDEs, the designer must be able to check that their technique works. In other words, the error needs to be quantifiable. The easiest way to do this is to compare your approximation to the true solution. In the case of Burgers' equation this becomes possible and is still to this day used for testing nonlinear solvers.

In many cases, in order to model real life situations more accurately, complicated initial conditions need to be invoked. In their unpublished work, Barth and Sukys [2] compute exact solutions to Burgers equation with initial conditions that contain discrete random variables. This research is an essential part of the development of numerical schemes to

stochastic PDEs with random initial conditions, of which Barth and Sukys are currently formulating.

3.1.1 Fluid Dynamics

Historically the equation has been used in the formulation of viscous and turbulent flows. Work on Burgers' equation started with Bateman [4] in 1915, where he proposed equation (1.1a) as a way to deal with small values of viscosity. The problem was that the motion of fluid becomes discontinuous as viscosity approaches zero. There is the presence of randomness while dealing with turbulence, so Bec and Kahanin [26] proposed the use of a multidimensional stochastic Burgers' equation to describe the phenomenon. Probability distribution of velocity gradients were studied by Weinan *et al* [11], and in particular the asymptotic behaviour.

3.1.2 Shock Theory

One of the most well known areas of study that Burgers' equation is often associated with is shock theory. As the viscosity of a fluid decreases and tends to zero, the velocity of the solution starts to become discontinuous (see Figure 3.1). Kreiss and Kreiss [19] studied the effects of shocks from the viscous Burgers' equation in 1985. They proved that all of the eigenvalues were negative, which helped lead to the uniqueness and existence of the steady state solution. They also investigated the speed of convergence. Inspired by the work of Kreiss and Kreiss, both Reyna and Ward [25] worked on a similar problem. With previous investigations focusing on positive values of κ, Reyna and Ward researched what happens in the limit as $\kappa \to 0$. Concluding that deviations in the boundary operator caused the equilibrium solution to destabilize, was their major contribution.

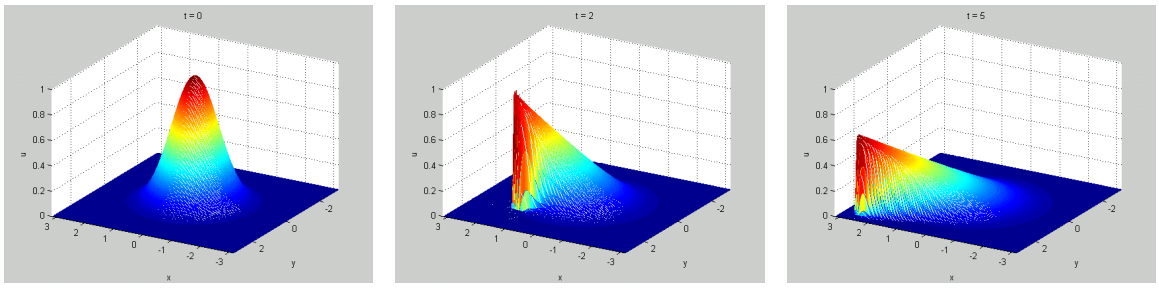

Figure 3.1: This is the two dimensional viscous Burgers' equation with a Gaussian initial condition solved numerically. At $t = 2$ there is shock formation, then at $t = 5$ the viscosity is causing the shock to dissipate as it travels.

3.1.3 Gas Dynamics

Panayotounakos and Drikakis [24] used a version of Burgers' with a source term in aerodynamics theory. The diffusion nature of the equation allowed for the heat exchange to be described effectively.

3.1.4 Cosmology

Gurbatov and Saichev [14] used the three dimensional version of Burgers equation to model the features of a random potential field in the context of cosmology. Solutions of the inviscid Burgers' equation in $\mathbb{R}^d$ was used to study Burgers' turbulence while modelling matter in the universe, researched by Molchanov *et al* [21]. Burgers' turbulence is "the study of the solutions to one- or multi-dimentional Burgers' equation with random initial conditions or random forcing," defined by Bec and Khanin [5] in their paper.

3.1.5 Traffic Flow

A model of traffic flow put forth by Greenshields [13] leads to the use of Burgers' equation. According to Musha and Higuchi [22], given that the concentration of cars is linearly related to drift speed, then Burgers' equation describes the concentration of cars on a highway.

3.1.6 Quantum Field

Jeffrey Yepez [29] presented a quantum algorithm for solving the Burgers' equation. He states that "the quantum algorithm exploiting superposition and entanglement is more efficient than the classical one because the quantum algorithm requires less memory." It is based on the quantum principle that qubits in a spin-$\frac{1}{2}$ system can be in a superposition state, rather than the classical bit system, that is either 1 or 0. Yepez's algorithm is able to compute solutions for arbitrarily high and low Reynolds numbers.

Chapter 4

Analytic Solutions

Within this section several versions of the analytical solution of Burgers' equation will be computed. The strategies different authors use to solve the equation will be reported and the apparent differences the choices of initial conditions make will be highlighted.

First consider the viscous Burgers' equation (3.1) with homogeneous Dirichlet boundary conditions

$$u(0,t) = u(1,t) = 0, \quad t > 0. \tag{4.1}$$

4.1 The Cole-Hopf Transformation

Nearly all authors I've read who share the analytical solution of Burgers' equation in their papers use the Cole-Hopf (or Hopf-Cole) transformation. The Cole-Hopf transformation [15, 9],

$$u = -2\kappa \frac{v_x}{v}, \tag{4.2}$$

was discovered independently by both Julian David Cole and Eberhard Hopf. What this transformation does is turn Burgers' equation, a nonlinear PDE, into a linear PDE, specifically the diffusion equation. The discovery of this technique inspired much research to explore this new avenue of dealing with nonlinear PDEs.

The transformation requires that the problem in question has periodic boundary conditions or equivalent (i.e. homogeneous). The transformation needs to be applied to all aspects of the equation, which obviously include the initial and boundary conditions. With our choice of boundary conditions above (4.1), the Cole-Hopf transformation changes them into the following

$$v_x(0,t) = v_x(1,t) = 0 \quad t > 0, \tag{4.3}$$

which you will notice as homogeneous Neumann conditions.

The next step to solving Burgers' equation is to compute the following derivatives

$$\begin{aligned} u_t &= -2\kappa\frac{v_{xt}}{v} + 2\kappa\frac{v_x v_t}{v^2}, \\ u_x &= -2\kappa\frac{v_{xx}}{v} + 2\kappa\frac{{v_x}^2}{v^2}, \\ u_{xx} &= -2\kappa\frac{v_{xxx}}{v} + 6\kappa\frac{v_x v_{xx}}{v^2} - 4\kappa\frac{{v_x}^3}{v^3}. \end{aligned}$$

If we take the above expressions and plug them into (3.1), after simplifying, we end up with

$$\frac{v_x}{v}(\kappa v_{xx} - v_t) - (\kappa v_{xx} - v_t)_x = 0.$$

4.2 Diffusion Equation

Notice that

$$v_t = \kappa v_{xx} \tag{4.4}$$

is the diffusion equation. The conclusion to draw from this is that solutions of (3.1) are composed of any solution $v(x,t)$ of (4.4).

Based on my exposure to the literature, most people tend to choose sinusoidal initial conditions to solve Burgers' equation exactly. Aside from the obvious smoothness factor that is evoked, this corresponds well with the periodic boundary conditions that are required in order to use the Cole-Hopf (4.2) transformation. The particular initial condition we will consider is

$$u(x,0) = \sin \pi x, \quad 0 < x < 1. \tag{4.5}$$

But, we still need to apply to the Cole-Hopf (4.2) transformation to (4.5). The result is

$$v(x,0) = \exp\left(-\frac{1-\cos(\pi x)}{2\kappa\pi}\right), \quad 0 < x < 1, \tag{4.6}$$

where exp is the exponential function with base e.

4.2.1 Separation of Variables

Öziş *et al* [23] use a traditional approach to solve the diffusion equation, in fact they use separation of variables [10] in order to later express the solution as an infinite sum. The important aspect of separation of variables is that it allows you to turn a PDE into several Ordinary Differential Equations (ODEs). Step one requires us to assume that the solution of the PDE will be a product solution of two functions, say $X(x)$ and $T(t)$. Now since we

have

$$v(x,t) = X(x)T(t),$$

we need to compute a few derivatives, namely

$$v_t = XT' \quad \text{and} \quad v_{xx} = X''T.$$

Now substitute these equations into (4.4),

$$\kappa X''T - XT' = 0$$

simplifying we get

$$\frac{T'}{T} = \kappa \frac{X''}{X}.$$

Since a function of x is equal to a function of t for all values in the domain, the functions X and T, must be constant functions. The expression now becomes

$$\frac{T'}{T} = \kappa \frac{X''}{X} = -\lambda,$$

for some unknown constant λ. This leaves us with

$$T' + \lambda T = 0 \quad \text{and} \quad \kappa X'' + \lambda X = 0.$$

Now our PDE has become two ODEs. The work that follows includes finding the associated eigenfunctions and eigenvalues for each ODE.

Starting with the spatial ODE, first we divide through by κ to get

$$X'' + \frac{\lambda}{\kappa} X = 0.$$

Since $\frac{\lambda}{\kappa}$ is a real number, by the trichotomy law, it is either negative, zero, or positive.

For the first case, when $\frac{\lambda}{\kappa} < 0$, the ODE becomes

$$X'' - \frac{\lambda}{\kappa} X = 0,$$

which leads us to solve the resulting characteristic equation

$$r^2 = \frac{\lambda}{\kappa} \implies r = \pm\sqrt{\frac{\lambda}{\kappa}}.$$

The solution to this equation is

$$X = c_1 \exp\left(\sqrt{\frac{\lambda}{\kappa}}\, x\right) + c_2 \exp\left(-\sqrt{\frac{\lambda}{\kappa}}\, x\right).$$

In order to apply the boundary conditions (4.3), we must first take a derivative, so

$$X' = c_3 \exp\left(\sqrt{\frac{\lambda}{\kappa}}\, x\right) - c_4 \exp\left(-\sqrt{\frac{\lambda}{\kappa}}\, x\right),$$

where new constants c_3 and c_4 are introduced to absorb the coefficients of x when they come down because of the chain rule. We now get the following

$$X'(0) = 0 \implies X'(0) = c_3 - c_4 = 0 \implies c_3 = c_4.$$

and then

$$X'(1) = 0 \implies X'(1) = c_3 \exp\left(\sqrt{\frac{\lambda}{\kappa}}\right) - c_3 \exp\left(\sqrt{\frac{\lambda}{\kappa}}\right) = 0 \implies c_3 = 0,$$

which means $c_4 = 0$ as well. That means the only solution for this case is the trivial solution $y = 0$.

Next, $\frac{\lambda}{\kappa} = 0$ yields the characteristic equation

$$r^2 = 0 \implies r = 0,$$

where the solution is

$$X(x) = c_5 + c_6 x.$$

After taking a derivative and applying the Neumann condition, we get $c_6 = 0$, and it follows that $c_5 = 0$ as well. Once again, the solution is thus $y = 0$.

Lastly, $\frac{\lambda}{\kappa} > 0$ leads us to solve

$$r^2 = -\frac{\lambda}{\kappa},$$

which means

$$r = \sqrt{\frac{\lambda}{\kappa}}\, i.$$

This leads to the general solution

$$X = c_7 \cos\left(\sqrt{\frac{\lambda}{\kappa}}\, x\right) + c_8 \sin\left(\sqrt{\frac{\lambda}{\kappa}}\, x\right).$$

After taking a derivative with respect to x, we have

$$X' = c_{10} \cos\left(\sqrt{\frac{\lambda}{\kappa}}\, x\right) - c_9 \sin\left(\sqrt{\frac{\lambda}{\kappa}}\, x\right).$$

After applying the boundary condition we have

$$X'(0) = c_{10} = 0.$$

With the other boundary condition leading to

$$X'(1) = -c_9 \sin\left(\sqrt{\frac{\lambda}{\kappa}}\right),$$

we know that either

$$c_9 = 0 \quad \text{or} \quad \sin\left(\sqrt{\frac{\lambda}{\kappa}}\right) = 0.$$

Therefore,

$$\sqrt{\frac{\lambda}{\kappa}} = n\pi \quad \text{for } n \in \mathbb{N} \cup \{0\},$$

which implies that the eigenvalues are

$$\lambda_n = n^2\pi^2\kappa \quad \text{for } n \in \mathbb{N} \cup \{0\},$$

with associated eigenfunctions

$$X_n = \cos(n\pi x).$$

Now the temporal ODE can be written as

$$T' + n^2\pi^2\kappa T = 0.$$

solving this separable equation gives us our second eigenfunction

$$T = a_n e^{-n^2\pi^2 kt} \quad \text{for } n \in \mathbb{N} \cup \{0\}.$$

The solution, u, will be any linear combination of the eigenfunctions. Therefore, based upon (4.3) and (4.6) we report that the solution to the heat equation (4.4), takes the following form

$$v(x,t) = a_0 + \sum_{n=1}^{\infty} a_n e^{-n^2\pi^2\kappa t}\cos(n\pi x) \tag{4.7}$$

where a_n, for $n \in \mathbb{N} \cup \{0\}$, are the Fourier coefficients, which are computed with the following two equations

$$a_0 = \int_0^1 \exp\left(-\frac{1-\cos(\pi x)}{2\pi\kappa}\right) \mathrm{d}x,$$
$$a_n = 2\int_0^1 \exp\left(-\frac{1-\cos(\pi x)}{2\pi\kappa}\right) \cos(n\pi x)\,\mathrm{d}x, \quad \text{for } n \in \mathbb{N}.$$

Therefore, given the initial condition (4.5) and boundary conditions (4.1), applying (4.2) to equation (4.7) gives us the exact solution to the viscous Burgers equation, namely

$$u(x,t) = \frac{2\pi\kappa \sum_{n=1}^{\infty} a_n e^{-n^2\pi^2\kappa t} n \sin(n\pi x)}{a_0 + \sum_{n=1}^{\infty} a_n e^{-n^2\pi^2\kappa t} \cos(n\pi x)}. \tag{4.8}$$

Similarly Öziş *et al* also found an exact solution to Burgers' with a different initial condition. Suppose the initial condition is

$$u(x,0) = 4x(1-x), \quad 0 < x < 1. \tag{4.9}$$

Of course (4.2) must be applied to (4.9). Using the same homogeneous boundary conditions (4.1) as above, the solution to Burgers' is (4.8); however, the Fourier coefficients change to

$$a_0 = \int_0^1 \exp\left(-\frac{x^2(3-2x)}{3\kappa}\right) \mathrm{d}x,$$
$$a_n = 2\int_0^1 \exp\left(-\frac{x^2(3-2x)}{3\kappa}\right) \cos(n\pi x)\,\mathrm{d}x, \quad \text{for } n \in \mathbb{N}.$$

Cole [9] reports an even more general solution to Burgers' equation in 1951. He starts by deploying the transformation (4.2) that would come to bear his name in the future. The initial condition he chose is

$$u(x,0) = u_0 \sin\left(\frac{\pi x}{L}\right), \quad 0 < x < L$$

where u_0 is the amplitude of the sinusoidal function, and L is the length of the domain in question. Notice that this does in fact change the boundary conditions (4.1) because instead of being on the interval $(0,1)$ we are now on $(0,L)$, but the difference between the two is trivial. Cole also expresses the solution in the form of a Fourier series,

$$u(x,t) = \frac{2\pi\kappa}{L} \frac{\sum_{n=1}^{\infty} a_n \exp\left(-\frac{n^2\pi^2\kappa}{L^2}\,t\right) n \sin\left(\frac{n\pi x}{L}\right)}{a_0 + \sum_{n=1}^{\infty} a_n \exp\left(-\frac{n^2\pi^2\kappa}{L^2}\,t\right) \cos\left(\frac{n\pi x}{L}\right)},$$

where

$$a_0 = \frac{1}{L}\int_0^L \exp\left(-\frac{u_0 L}{2\pi\kappa}\left[1-\cos\left(\frac{\pi x}{L}\right)\right]\right)\,\mathrm{d}x,$$
$$a_n = \frac{2}{L}\int_0^L \exp\left(-\frac{u_0 L}{2\pi\kappa}\left[1-\cos\left(\frac{\pi x}{L}\right)\right]\right)\cos\left(\frac{n\pi x}{L}\right)\,\mathrm{d}x, \quad \text{for } n \in \mathbb{N}.$$

The final formulation for the exact solution of Burgers' using Fourier analysis that I will present is done by Kadalbajoo and Awasthi [16]. The first thing to point out is instead of considering semi-infinite time, the problem is posed with finite time. Actually, Kadalbajoo and Awasthi start with the same conditions that Bateman [4] put forth in his 1915 paper, namely (1.1), except they make two small changes. The spatial domain goes from $(0, 1)$ instead of from $(0, L)$, and time $t = \tau$ is included. Hence, the domain is then referred to as

$$\Omega = (0, 1) \times (0, \tau]. \tag{4.10}$$

Unlike the solutions so far, the boundary is not homogeneous. Nonetheless, observe that applying (4.2) to (1.1c) forces the boundary to homogeneous conditions after the change of variables because the transform contains a spatial derivative. Transforming the initial condition (1.1b), results in the following formulation where $v(x, 0)$ cannot be written in closed-form for the case of an arbitrary initial condition and therefore becomes

$$v(x, 0) = \exp\left(-\frac{1}{2\kappa}\int_0^x \psi(s)\,\mathrm{d}s\right), \quad 0 < x < 1, \tag{4.11}$$

after the Cole-Hopf transformation. So for $(x, t) \in \Omega$ the solution is (4.8) with the Fourier coefficients as

$$a_0 = \int_0^1 \exp\left(-\frac{1}{2\kappa}\int_0^x \psi(s)\,\mathrm{d}s\right)\,\mathrm{d}x$$
$$a_n = 2\int_0^1 \exp\left(-\frac{1}{2\kappa}\int_0^x \psi(s)\,\mathrm{d}s\right)\cos(n\pi x)\,\mathrm{d}x, \quad \text{for } n \in \mathbb{N}.$$

4.2.2 The Heat Kernel

There is one more common approach to solving this problem and it involves a convolution with the heat kernel. One example of this approach is performed by Kevorkian [17]. He uses an arbitrary spatial function (1.1b) for his initial conditions. For the first time in this paper, the solution is considered on an infinite domain, and hence, boundary conditions are unnecessary. Kevorkian applies (4.2) to the initial condition, and we have seen that the equation becomes (4.11). In order to make the equations less crowded, the following substitution is carried out

$$v(x, 0) = \alpha \exp\left(-\frac{1}{2\kappa}\int_0^x \psi(s)\mathrm{d}s\right) \equiv \alpha g(x), \quad \alpha \in \mathbb{R}. \tag{4.12}$$

The problem is now to solve (4.4) on $-\infty < x < \infty$ given the initial condition of $v(x,0) = \alpha g(x)$. Using Fourier transforms it can be shown that the heat kernel is

$$H(x,t) = \frac{1}{\sqrt{4\pi t}} e^{-x^2/4t}. \tag{4.13}$$

A convolution is an operation of the form

$$(\mathcal{F} * \mathcal{G})(x) = \int_{-\infty}^{\infty} \mathcal{F}(x-\xi)\mathcal{G}(\xi)\,\mathrm{d}\xi, \tag{4.14}$$

giving that both $\mathcal{F}$ and $\mathcal{G}$ are integrable. Now a solution to the heat equation can be expressed as a convolution (4.14) of the initial condition (4.12) and the heat kernel (4.13) as follows,

$$v(x,t) = (H * g)(x) = \frac{\alpha}{2\sqrt{\pi\kappa t}} \int_{-\infty}^{\infty} g(\xi) e^{-(x-\xi)^2/4\kappa t}\,\mathrm{d}\xi. \tag{4.15}$$

After computing one spatial derivative of (4.15) we are left with

$$v_x(x,t) = -\frac{\alpha}{2\sqrt{\pi\kappa t}} \int_{-\infty}^{\infty} \frac{g(\xi)g(x-\xi)}{2\kappa t} e^{-(x-\xi)^2/4\kappa t}\,\mathrm{d}\xi.$$

Now transforming back to the original problem using (4.2) we get the final solution to Burgers' equation

$$u(x,t) = \frac{\int_{-\infty}^{\infty} g(\xi)\frac{(x-\xi)}{t} e^{-(x-\xi)^2/4\kappa t}\,\mathrm{d}\xi}{\int_{-\infty}^{\infty} g(\xi) e^{-(x-\xi)^2/4\kappa t}\,\mathrm{d}\xi}.$$

Kevorkian [17] also solves Burgers' equation on a semi-infinite domain. He sets the original initial condition (1.1b) to be constant. He chooses the function on the left boundary to be constant as well

$$u(0,t) = h, \tag{4.16}$$

where $h \in \mathbb{R}$. Apply (4.2) to (4.16) and the result is

$$hv(0,t) + 2\kappa v_x(0,t) = 0$$

for the boundary and

$$v(x,0) = \alpha$$

for the initial condition. From this information, Kevorkian claims that the solution to the diffusion equation is well know and presents it in the form

$$v(x,t) = \alpha \left[1 - \operatorname{erfc}\left(\frac{x}{2\sqrt{\kappa t}} \right) + \exp\left(\frac{h^2 t}{4\kappa} - \frac{hx}{2\kappa} \right) \operatorname{erfc}\left(\frac{x - ht}{2\sqrt{\kappa t}} \right) \right], \tag{4.17}$$

where $\operatorname{erfc}(x)$ denotes the complementary error function, given by

$$\operatorname{erfc}(x) = 1 - \operatorname{erf}(x) = 1 - \frac{2}{\sqrt{\pi}} \int_0^x e^{-\eta^2} \, \mathrm{d}\eta = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-\eta^2} \, \mathrm{d}\eta.$$

Thus by the Cole-Hopf transform, (4.17) becomes

$$u(x,t) = h \frac{\operatorname{erfc}\left(\frac{x-ht}{2\sqrt{\kappa t}} \right)}{\exp\left(\frac{hx}{2\kappa} - \frac{h^2 t}{4} \right) \operatorname{erf}\left(\frac{x}{2\sqrt{\kappa t}} \right) + \operatorname{erfc}\left(\frac{x-t}{2\sqrt{\kappa t}} \right)}.$$

Chapter 5

Numerical Solutions

Numerical methods are important because in practice, most PDEs do not have an analytical solution. With PDEs being one of the most powerful tools we have to describe the natural world, the importance of numerical solutions to these types of problems cannot be overstated. Many areas of mathematics contribute to the development of numerical methods, most prominent among them are numerical analysis, calculus, and scientific computing. The numerical solution approaches focused upon in this section will include finite differences, finite elements, and spectral methods.

5.1 Finite Differences

The main idea behind finite differences is to approximate the derivative using nearby points, which is called a stencil. The most basic example of a stencil would be the points $x_0 + h$ and x_0 when used in the difference quotient, or forward difference, known by most as an approximation for the derivative in one dimension

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h}. \tag{5.1}$$

More points could be considered in order to converge to the true solution more quickly as h gets small, such as in the following

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0 - h)}{2h} \tag{5.2}$$

which is called the centred difference. An additional example would be the following approximation based on a five point stencil for the first derivative

$$f'(x_0) \approx \frac{f(x_0 - 2h) - 8f(x_0 - h) + 8f(x_0 + h) - f(x_0 + 2h)}{12h}$$

seen below in 5.1.

Figure 5.1: This is a five point stencil.

When your function is based on more than one variable, which is the case with PDEs, then you need an array, cube, or hypercube of points. In the case of the one-dimensional Burgers' equation, we will need to use a two dimensional stencil.

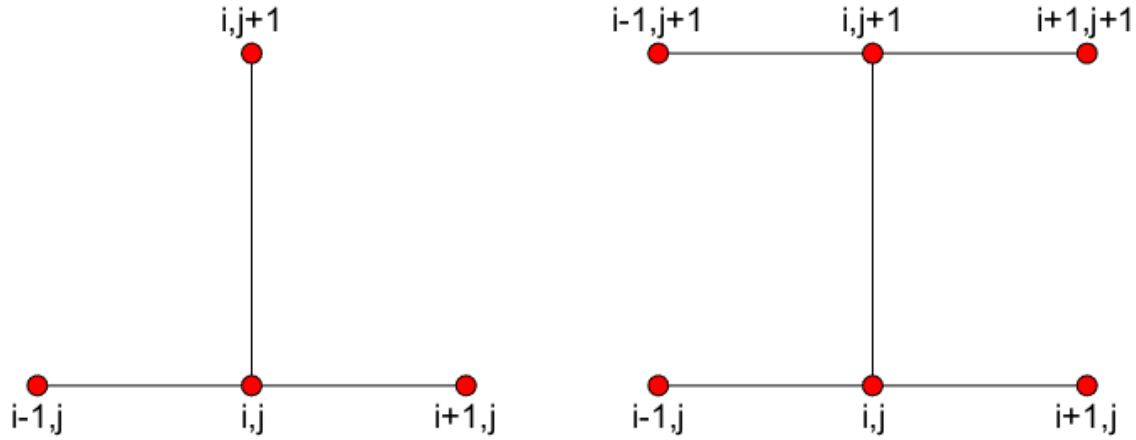

Figure 5.2: This is the Lax-Wendroff stencil (*left*) and the Crank-Nicolson stencil (*right*) where i, j represents space and time, respectively.

Although the finite difference method is probably the most conceptually simple, it still has it's faults. The two main problems include challenges with computing the boundary values occasionally and roundoff errors.

Some finite difference stencils might not be able to evaluate points on the boundary. The one-dimensional example is if you were computing a derivative for a function at $x_0 = 1$ given the domain $[0, 1]$, if you are using (5.1), then you are attempting to evaluate a point outside of the domain, namely $x_0 + h$. Hence, an approximation at $x = 1$ is not possible with this formula. One alternative is to use another difference formula for this case, such as

$$f'(x_0) \approx \frac{f(x_0) - f(x_0 - h)}{h}.$$

Now notice that $x_0 = 1$ implies that $x_0 - h \in [0, 1]$. Although this sounds like a simple fix, continually changing the difference formula while trying to produce an approximation on a complicated geometry could prove to be cumbersome. Another way around this issue is to use adaptive meshes. Adaptive mesh is a technique that refines the area on which the numerical differentiation is being done. It would be like making the grid of points smaller until a desired accuracy is met. This approach in theory would allow you to get arbitrarily

close to the boundary, however the more the mesh gets refined, the more computationally expensive the method becomes.

Another fault of finite difference methods generally is that they are usually prone to roundoff errors. Take the simple forward difference formula (5.1) for instance. Observe that the h in the denominator is intended to be aa small number. This suggest that the desired value is a quantity close to zero. In a floating point system, approaching particularly small values eventually causes the approximation to blow up. Essentially the trick is to choose a value that is sufficiently small, but not too small. Sometimes finding this balance proves difficult for certain questions.

Mohan. K. Kadalbajoo and Ashish Awasthi [16] take an approach of a Crank-Nicolson scheme for solving Burgers' numerically using finite differences. The conditions that they are setting up is similar to (1.1a), except with $L = 1$ for the spatial domain. Recall their formulation of the domain from earlier (4.10). The solution domain will be denoted with $\bar{\Omega}$, and will be a discretized uniform mesh $\bar{\Omega}_{i,j} = \{(x_i, t_j) |\ i = 0, 1, \ldots, N,\ \ j = 0, 1, \ldots, M\}$. They divide the space interval $[0, 1]$ and the time interval $[0, \tau]$ into N and M equal subintervals respectively. The discretized version of the domain includes the endpoints which will make implementation easier. The mesh width in space will be $h = 1/N$ and in time will be $k = \tau/M$. This leads to the point representations of $x_i = ih$ for $i = 0, 1, \ldots, N$ and $t_j = 0, 1, \ldots, M$ for space and time respectively. Finally, the equations for the Crank-Nicholson scheme equations can be written down as follows

$$-rv_{i+1,j+1} + (1+r)v_{i,j+1} = rv_{i+1,j} + (1-r)v_{i,j}, \qquad i = 0 \tag{5.3a}$$

$$-\frac{r}{2}v_{i+1,j+1} + (1+r)v_{i,j+1} - \frac{r}{2}v_{i-1,j+1} = \frac{r}{2}v_{i+1,j} + (1-r)v_{i,j} + \frac{r}{2}v_{i-1,j}, \quad i = 1, 2, \ldots, N-1 \tag{5.3b}$$

$$-rv_{i-1,j+1} + (1+r)v_{i,j+1} = rv_{i-1,j} + (1-r)v_{i,j}, \qquad i = N \tag{5.3c}$$

where $r = k\kappa/h^2$, $j = 0, 1, \ldots, M$, and $v_{i,j}$ is the discrete approximation to $v(x_i, t_i)$ at the point (i, j). This is the approximation to the heat equation (5.3), which leads to the approximate solution of Burgers' equation by using this discretized version of the Cole-Hopf transformation (4.2) by means of a centred difference formula (see Equation 5.2)

$$u_{i,j} = \kappa \left(\frac{v_{i+1,j} - v_{i-1,j}}{hv_{i,j}} \right). \tag{5.4}$$

In the computational component of their paper, Mohan. K. Kadalbajoo and Ashish Awasthi tackled two problems. For both they used homogeneous Dirichlet conditions (4.1), and for the initial conditions they used (4.5) and (4.9). They verified that the theoretical order of accuracy $\mathcal{O}(h^2 + k^2)$ was correct, and also reported that the method is unconditionally stable for these problems.

A more classical finite difference approach to solving Burgers' equation was taken by Aref and Daripa [1]. They started with an initial condition of an arbitrary function, such as (1.1b) in addition to true periodic boundary conditions formulated as

$$u(x+L,t) = u(x,t), \tag{5.5}$$

where L is the length of the interval. Aref and Daripa prescribe grid values $u_k(t) = u(kL/N, t)$ for $k = 0, 1, \ldots, N-1$. The spatial derivatives then get described by the following discretized system

$$\begin{aligned} u_{xx} &= \left(\frac{N}{L}\right)^2 (u_{k+1} + u_{k-1} - 2u_k), \\ uu_x &= \left(\frac{N}{2L}\right) u_k(u_{k+1} - u_{k-1}). \end{aligned}$$

The next step that is a process called nondimensionalization. It is a way of performing certain variable substitutions so that physical quantities get partially or sometimes entirely removed. The expression used to compute this is

$$v_k(s) = \left(\frac{L}{N\kappa} u_k\left(\frac{L^2 s}{N^2 \kappa}\right)\right), \quad k = 0, 1, \ldots N-1.$$

Finally, this all leads to

$$\frac{\mathrm{d}v_k}{\mathrm{d}s} = -\left(\frac{v_k}{2}\right)(v_{k+1} - k_{k-1}) + v_{k+1} + k_{k-1} - 2v_k,$$

which is the discretization of Burgers equation. The periodic boundary conditions (5.5) are now interpreted as $v_{k+N} = v_k$. The important thing to point out about this formulation is that it has created a system of N coupled ODEs.

5.1.1 My Implementation

In my implementation of a finite difference approach to Burgers' equation I used the Crank-Nicolson scheme. I believe it would be worth while to describe the algorithm.

The first thing to realize is that the method is both implicit and explicit. It is explicit in space, however it is implicit in time. The scheme uses a 6-point stencil (5.3) in general, except for the first and last spatial position, wherein it uses a 4-point stencil. Observe from (5.3) that the second index, the j-th index, is $j+1$ on the left-hand side (LHS), but simply j on the right hand side (RHS). The physical meaning of this is that the RHS represents the current time step, and the LHS represents the subsequent time step. Each $v_{_{i,j}}$ on the LHS can then be treated as unknowns. This translates into two knowns for each of the $i = 0$ and $i = N$ equations, whereas the rest have three. By paying close attention to the indexing, you realize that their will be $N+1$ unknowns. Conveniently, (5.3) provides us with $N+1$

equations. We can therefore construct the coefficient matrix for the LHS. The RHS can be computed because the initial condition gives us the value of $v_{i,0}$ for all i. Thus this becomes the following matrix system

$$\begin{bmatrix} 1+r & -r & & & & & \\ -\frac{r}{2} & 1+r & -\frac{r}{2} & & & & \\ & & \ddots & & & & \\ & & & -\frac{r}{2} & 1+r & -\frac{r}{2} & \\ & & & & -r & 1+r \end{bmatrix} \begin{bmatrix} v_{0,j+1} \\ v_{1,j+1} \\ \vdots \\ v_{N-1,j+1} \\ v_{N,j+1} \end{bmatrix} = \begin{bmatrix} rv_{1,j} + (1-r)v_{0,j} \\ \frac{r}{2}v_{2,j} + (1-r)v_{1,j} + \frac{r}{2}v_{0,j} \\ \vdots \\ \frac{r}{2}v_{N,j} + (1-r)v_{N-1,j} + \frac{r}{2}v_{N-2,j} \\ (1-r)v_{N,j} + rv_{N-1,j} \end{bmatrix}$$

which will be referred to in this paper as the system $A\mathbf{v}_{j+1} = \mathbf{b}_j$ for $j = 0, 1, \ldots, M-1$. See Algorithm 1 below, which is the Crank-Nicolson scheme I deployed to solve Burgers' equation.

Algorithm 1: Crank-Nicolson Finite Difference Scheme for the Diffusion Equation

```
function CrankNicolson (f, κ, L, T, N, M);
  Input  : The initial condition, f, the viscosity coefficient, κ, the length of the
           space interval, L, the length of the time interval, T, the number of
           sub-intervals in space, N, and the number of sub-intervals in time, M
  Output: The solution to the diffusion equation, v, as a matrix of points
h = L/N; // Set the mesh width for space
k = T/M; // Set the mesh width for time
r = kκ/h^2; // Define the constant r
for i = 0 to N do
    v_{i,0} = f(ih); // Compute the solution for the first time step, j = 0
end
Build the tridiagonal matrix A, (see above);
/* Deploy the Crank-Nicolson stencil.                                    */
for j = 0 to M − 1 do
    b_j(0) = r v_{1,j} + (1 − r) v_{0,j};
    for i = 1 to N − 1 do
        b_j(i) = (r/2) v_{i+1,j} + (1 − r) v_{i,j} + (r/2) v_{i−1,j};
    end
    b_j(N) = r v_{i−1,j} + (1 − r) v_{i,j};
    /* Solve the system of equations for the next time step.             */
    v_{:,j+1} = A\b_j // Fill the j + 1-th column of the solution matrix v
end
```

The initial condition is evoked when $j = 0$,

$$v_{i,0} = \exp\left(-\frac{1}{2\kappa}\int_0^x \psi(s)\,\mathrm{d}s\right).$$

5.2 Finite Elements

The finite elements approach is to break up the problem into a finite set of smaller more intelligible problems. Usually this means that these smaller problems are based on shapes with really simple geometry in order to simplify calculation. This is done by formulating an approximation for the problem into a linear combinations of basis functions. Basis functions will usually have a little bit of overlap, refer to Figure 5.4. Usually piece-wise linear basis functions are chosen (hat functions). A system of equations is then derived based on these smaller elements, and thus the problem at hand can be modelled (see Figure 5.3).

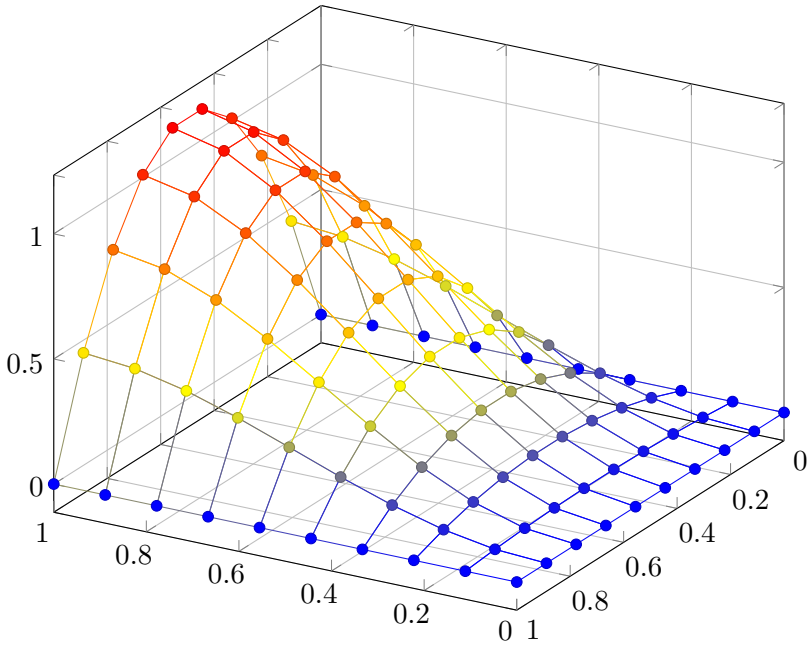

Figure 5.3: An example finite elements mesh for the diffusion equation.

Öziş *et al* [23] take an approach of solving Burgers' by first applying the Hopf-Cole transformation (4.2), then deploying the technique of finite elements to the diffusion equation (4.4). The initial condition is (4.5) and boundary conditions are homogeneous (4.1). First, the linear space of all test functions will be represented by $H[0,1]$, known as a Sobolev space. A Sobolev space means that the given test function, $\Phi(x) \in L^2[0,1]$, and it's derivative exists in $L^2[0,1]$ as well. Now integrate the product of the heat equation (4.4) and some test function like so

$$\int_0^1 \Phi(x)(v_t - \kappa v_{xx})\,\mathrm{d}x = 0.$$

Next we end up with

$$\int_0^1 \left(\Phi(x)v_t + \kappa v_x \Phi_x\right)\,\mathrm{d}x = \kappa\left(\Phi(1)v_x(1,t) - \Phi(0)v_x(0,t)\right), \tag{5.6}$$

after integration by parts. Equation (5.6) will be considered the weak form of the heat equation (4.4). In order to approximate it, a Galerkin method will be applied.

The approximate solution will be of the form

$$\widetilde{v}(x,t) = \sum_{i=1}^{N+1} b_i(t)\Psi_i(x) \tag{5.7}$$

where $\Psi_i(x) \in H[0,1](1 \leq i \leq N+1)$ are linearly independent trial basis functions and $b_i(t)(1 \leq i \leq N+1)$ are the yet undetermined differentiable functions of time. According to the Galerkin method, the trial basis functions and the test basis functions are equal, so they may be swapped in (5.7). Now,

$$\Phi(x) = \sum_{i=1}^{N+1} a_i\Phi_i(x),$$

for every test function $\Phi(x)$, where $a_i(1 \leq i \leq N+1)$ are arbitrary real numbers.

Divide the interval $[0,1]$ up into N subintervals $\Omega_1, \Omega_2, \ldots, \Omega_N$ of $h_1, h_2, \ldots, h_N$ respectively. The following is how the ith element (the line segment Ω_i), will be denoted

$$\Omega_i = [x_i, x_{i+1}]; \qquad h_i = x_{i+1} - x_i, \qquad \text{for } i = 1, 2, \ldots, N+1.$$

Now we can construct the test functions (see Figure 5.4)

$$\Phi_i(x) = \begin{cases} \frac{x-x_{i-1}}{h_{i-1}}, & x \in \Omega_{i-1} \\ 1 - \frac{x-x_i}{h_i}, & x \in \Omega_i \\ 0, & x \notin \Omega_{i-1} \cup \Omega_i \end{cases} \qquad \text{for } i = 1, 2, \ldots, N+1\,. \tag{5.8}$$

Plugging in the boundary conditions (4.1) into (5.6) we get

$$\int_0^1 \left(\Phi(x)v_t + \kappa v_x\Phi_x\right)\,\mathrm{d}x = 0. \tag{5.9}$$

Next

$$v_h(x,t) = \sum_{i=1}^{N+1} b_i(t)\Phi_i(x), \tag{5.10}$$

where $v_h(x,t)$ is the approximate solution of the heat equation. Now plugging (5.10) into (5.9) leaves us with

$$\int_0^1 \left(\Phi(x)\frac{\partial v_h}{\partial t}(x,t) + \kappa\frac{\mathrm{d}\Phi}{\mathrm{d}x}\frac{\partial v_h}{\partial x}(x,t)\right)\,\mathrm{d}x = 0. \tag{5.11}$$

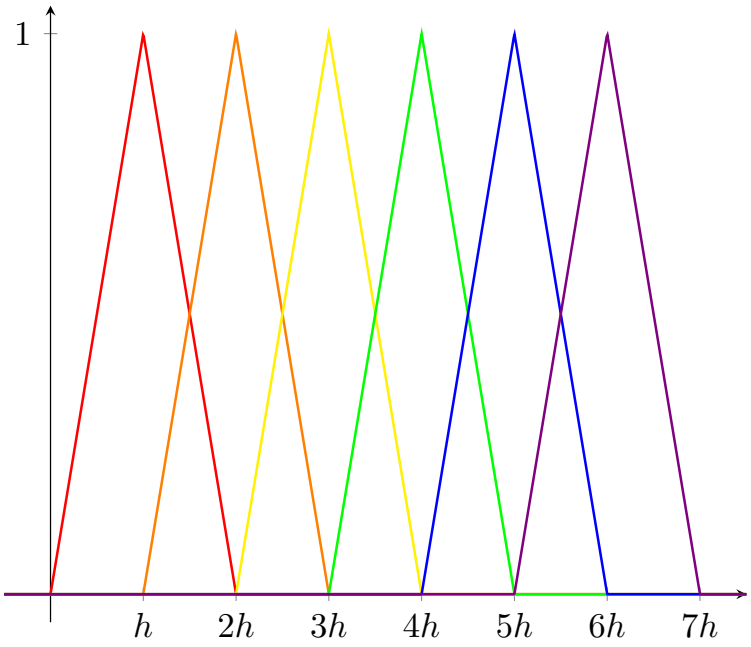

Figure 5.4: Piecewise linear functions

Next substitute (5.10) into (5.11) and after some simplification, the system becomes

$$\sum_{N+1}^{i=1}\sum_{N+1}^{j=1} a_i \left(\frac{\mathrm{d}b_j}{\mathrm{d}t}\int_0^1 \Phi_i(x)\Phi_j(x)\,\mathrm{d}x\right) = -\sum_{N+1}^{i=1}\sum_{N+1}^{j=1} a_i \left(b_j \int_0^1 \Phi_i'(x)\Phi_j'(x)\,\mathrm{d}x.\right)$$

Now we get a system of discrete linear first-order differential equations

$$\sum_{j=1}^{n+1} c_{ij}\frac{\mathrm{d}b_j}{\mathrm{d}t} = -\kappa \sum_{j=1}^{N+1} b_j k_{ij} \tag{5.12}$$

where c_{ij} and k_{ij} represent the following integrals

$$c_{ij} = \int_0^1 \Phi_i(x)\Phi_j(x)\,\mathrm{d}x \tag{5.13}$$

$$k_{ij} = \int_0^1 \Phi_i'(x)\Phi_j'(x)\,\mathrm{d}x \tag{5.14}$$

for $1 \le i, j \le N+1$. The matrix form of the system (5.12) is

$$C\frac{\mathrm{d}}{\mathrm{d}t}\underline{b}(t) + \kappa K\underline{b}(t) = 0 \tag{5.15}$$

where C and K are $N+1 \times N+1$ tridiagonal positive definite symmetric matrices and $\underline{b}^j$ denotes the finite difference approximation to the vector b. We can solve the ODE system

(5.15) using finite differences. Similarly, the time derivative is approximated by

$$\frac{\mathrm{d}}{\mathrm{d}t}\underline{b} = \frac{1}{\Delta t}(\underline{b}^{j+1} - \underline{b}^{j}).$$

Introducing a weight factor $\theta \in [0,1]$ allows Öziş *et al* to rewrite the system in its most general form

$$\frac{1}{\Delta t}C\left(\underline{b}^{j+1} - \underline{b}^{j}\right) + \kappa K\left(\theta\underline{b}^{j+1} + (1-\theta)\underline{b}^{j}\right) = 0. \tag{5.16}$$

A rearrangement of the system then takes the form

$$D\underline{b}^{j+1} = E\underline{b}^{j} \tag{5.17}$$

where

$$D = C + \kappa\theta\Delta tK \qquad \text{and} \qquad E = C - \kappa(1-\theta)\Delta tK.$$

Using the initial condition, this system can be solved.

Once the system has been solved, the discretized Cole-Hopf transformation (5.4) is used to establish an approximation to Burgers' equation.

5.2.1 My Implementation

I coded the finite elements approach by following the work of Öziş *et al* [23]. The following will contain some simplifications as well as some more detailed information regarding the approach.

First notice that the definition of the chosen piecewise linear functions (5.8), or hat functions, are made *very* general. This is usually a strength of the finite element method because if gives the user a choice of picking the most appropriate function to cover a given subregion. For our purposes, since we are solving Burgers' equation on a rectangular domain, we can make simplifications to these hat functions. We will use equidistant points in the space domain, which implies the h will no longer require a subscript, because every subinterval will be the same length. The corollary of this choice is that for each case of hat functions Φ_i and Φ_j such that $i = j$ and $|i-j| = 1$, these will all be equal.

Take the first hat function, Φ_1, where (5.8) implies that

$$\Phi_1(x) = \begin{cases} \frac{x-x_0}{h}, & x \in \Omega_0 \\ 1 - \frac{x-x_1}{h}, & x \in \Omega_1 \\ 0, & x \notin \Omega_0 \cup \Omega_1. \end{cases} \tag{5.18}$$

Notice that on the interval $[0, L]$ we can find the values x_0 and x_1 quite easily. Since $x_i = ih$ this suggests that $x_0 = 0$ and $x_1 = h$. Next we have two integrals to compute, namely for the cases $i = j$ and $|i - j| = 1$. Because the hat functions have compact support, they are zero on most of the domain and hence the products of them for $|i - j| \geq 2$ will all be 0. Now let us take the case of $i = j$ for the first piecewise linear function in the domain. The first step is to compute the product of $\Phi_i \Phi_i$ on the first region, Ω_0, which is

$$\Phi_1 \Phi_1 = \frac{x^2}{h^2}.$$

Now computing the integral leaves us with

$$\int_0^h \Phi_1 \Phi_1 \, \mathrm{d}x = \frac{x^3}{3h^2}\bigg|_0^h = \frac{h}{3}.$$

As you can see in Figure 5.5, due to symmetry, the integral on the subsequent region, Ω_1, will be $h/3$ also. Therefore,

$$\int_0^1 \Phi_1 \Phi_1 \, \mathrm{d}x = \int_0^{2h} \Phi_1 \Phi_1 \, \mathrm{d}x = \frac{2h}{3}.$$

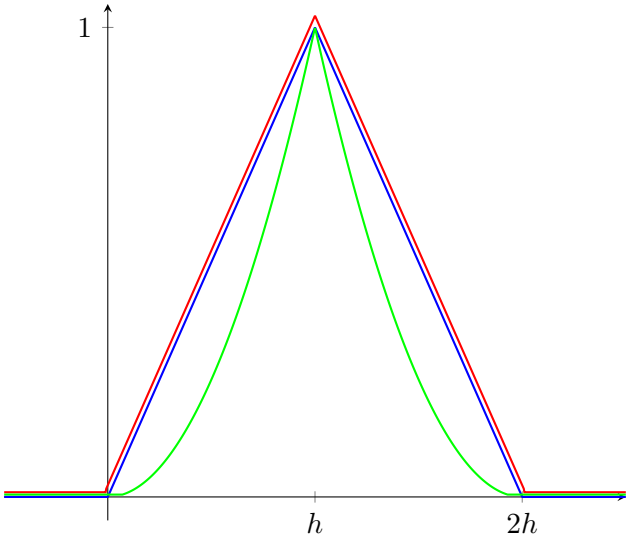

Figure 5.5: The area under the curve of $\Phi_1 \Phi_1$

Next for the $|i - j| = 1$ case, we introduce the following hat function in the sequence

$$\Phi_2(x) = \begin{cases} \frac{x - x_1}{h}, & x \in \Omega_1 \\ 1 - \frac{x - x_2}{h}, & x \in \Omega_2 \\ 0, & x \notin \Omega_1 \cup \Omega_2, \end{cases} \tag{5.19}$$

and we know that $x_1 = h$ and $x_2 = 2h$. The product of $\Phi_1\Phi_2$ on Ω_1 is

$$\Phi_1\Phi_2 = \frac{1}{h^2}(x-h)(2h-x) = \frac{1}{h^2}(-x^2+3hx-2h^2).$$

Next we compute the integral over the same region, displayed in Figure 5.6

$$\int_h^{2h} \Phi_1\Phi_2 \,\mathrm{d}x = \frac{1}{h^2}\left(-\frac{x^3}{3}+\frac{3hx^2}{2}-2h^2x\right)\Bigg|_h^{2h} = -\frac{8h}{3}+6h-4h-\left(-\frac{h}{3}+\frac{3h}{2}-2h\right) = \frac{h}{6}.$$

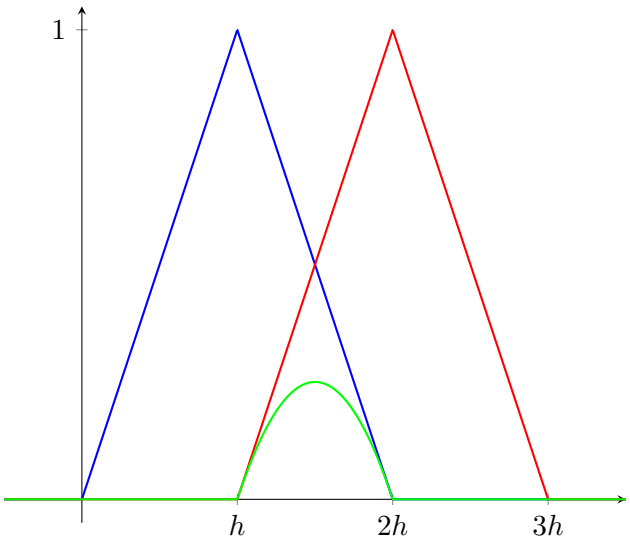

Figure 5.6: The area under the curve of $\Phi_1\Phi_2$

It follows from this result that we can calculate all the constants, c_{ij}, for $1 \leq i, j \leq N+1$. Therefore we can construct the following tridiagonal matrix

$$C = \begin{bmatrix} \frac{2h}{3} & \frac{h}{6} & & & & & \\ \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & & & & \\ & & \ddots & & & & \\ & & & \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & \\ & & & & \frac{h}{6} & \frac{2h}{3} \end{bmatrix}. \tag{5.20}$$

Next we must compute the k_{ij} constants. In order to do this, we require derivatives of (5.18) and (5.19) as follows

$$\Phi_i'(x) = \begin{cases} \frac{1}{h}, & x \in \Omega_{i-1} \\ -\frac{1}{h}, & x \in \Omega_i \\ 0, & x \notin \Omega_{i-1} \cup \Omega_i \end{cases}$$

for each $i = 1, 2, \ldots, N+1$. The calculation for k_{ij} in the cases $i = j$ and $|i - j| = 1$ are both given below (see Figure 5.7)

$$\int_0^1 \Phi_1' \Phi_1' \, \mathrm{d}x = 2 \int_0^h \frac{1}{h^2} \, \mathrm{d}x = \frac{2}{h}$$
$$\int_0^1 \Phi_1' \Phi_2' \, \mathrm{d}x = \int_h^{2h} -\frac{1}{h^2} \, \mathrm{d}x = -\frac{1}{h}.$$

Now we can construct our next matrix,

$$K = \begin{bmatrix} \frac{2}{h} & -\frac{1}{h} & & & & & \\ -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & & & & \\ & & \ddots & & & & \\ & & & -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & \\ & & & & -\frac{1}{h} & \frac{2}{h} & \end{bmatrix}. \tag{5.21}$$

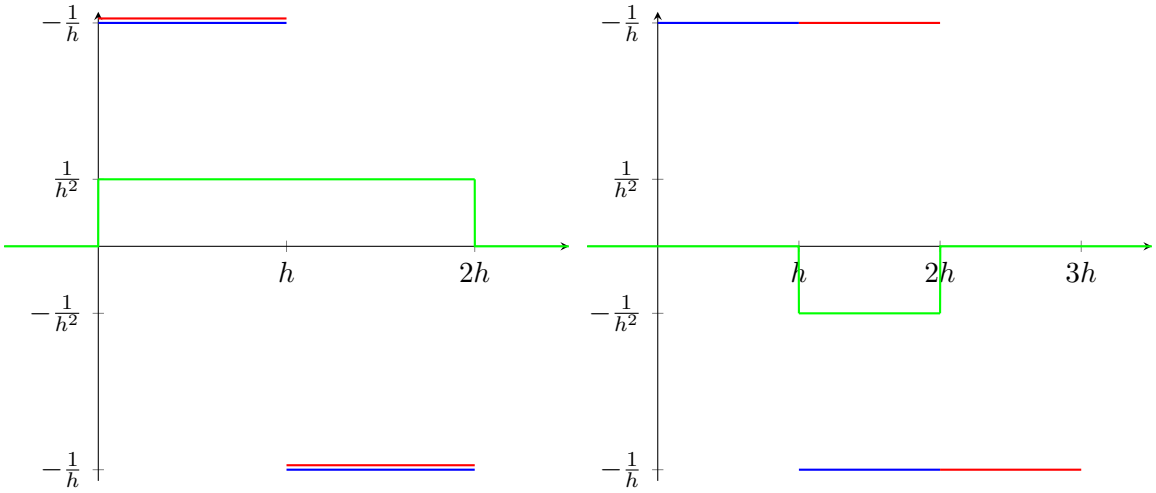

Figure 5.7: The area under the curve of $\Phi_1' \Phi_1'$ (*left*) and $\Phi_1' \Phi_2'$ (*right*).

At this point we have now established (5.15). With an introduction of the finite difference approximation to the time derivatives and the weight factor θ, the system becomes (5.16). The system can undergo another change into (5.17). From this form a solution to the Diffusion equation can be established. Then, the discrete Cole-Hopf transformation (5.4) is used to arrive at the solution we seek.

Algorithm 2 is the pseudocode I used to deploy the finite element method. The initial condition will be $j = 0$,

$$v_{i,0} = \exp\left(-\frac{1}{2\kappa} \int_0^x \psi(s) \, \mathrm{d}s\right).$$

Algorithm 2: Finite Elements Scheme for the Diffusion Equation

```
function FiniteElements (f, κ, L, T, N, M, θ);
  Input  : The initial condition, f, the viscosity coefficient, κ, the length of the
           space interval, L, the length of the time interval, T, the number of
           sub-intervals in space, N, the number of sub-intervals in time, M, and
           the weight factor, θ
  Output: The solution to the diffusion equation, v, as a matrix of points
h = L/N; // Set the mesh width for space
k = T/M; // Set the mesh width for time
for i = 0 to N do
    v_{i,0} = f(ih); // Compute the solution for the first time step, j = 0
end
Build the tridiagonal matrix C, (see above);
Build the tridiagonal matrix K, (see above);
D = C + κθkK; // Construct the matrix D
E = C − κ(1 − θ)kK; // Construct the matrix E
for j = 0 to M − 1 do
    /* Solve the system of equations for the next time step.        */
    v_{:,j+1} = D\(E v_{:,j}) // Fill the j + 1-th column of the solution matrix v
end
```

5.3 Spectral Methods

Spectral methods are a technique that incorporate Fourier transforms to move the given functions into the frequency-domain, which in turn allows for easier computation. In particular the use of the Fast Fourier Transform (FFT) creates a solution in terms of a sum of sinusoidal functions. The final step is to choose the coefficients of these basis functions which satisfy the PDE.

Basdevant *et al* [3] worked on several spectral methods in their paper. First to set up the question, they chose the following initial and boundary conditions

$$u(x,0) = -\sin \pi x, \quad |x| \leq 1 \tag{5.22a}$$

$$u(\pm 1, 0) = 0. \tag{5.22b}$$

Basdevant *et al* make a interesting change of variables, namely they substitute $x' = \pi x$ and $t' = \pi t$, then drop the primes, which means x' and t' will be displayed as x and t, respectively. The problem (5.22) changes slightly to the following

$$\begin{aligned} u_t + uu_x &= \kappa' u_{xx}, \quad -\pi \leq x \leq \pi, \quad t > 0 \\ u(x,0) &= -\sin x, \end{aligned}$$

where $\kappa' = \pi\kappa$. A truncated Fourier series can now approximate the solution because it is 2π-periodic. This is represented as

$$u_n(x,t) = \sum_{k=-N}^{N} u_k(t) e^{ikx}.$$

Basdevant *et al* cite Gottlieb and Orszag [12] with coming up with the following spectral equations

$$\frac{\mathrm{d}u_k}{\mathrm{d}t} = -\frac{ik}{2} N_k(\mathbf{u},\mathbf{u}) - \kappa' k^2 u_k, \quad -N \le k \le N, \tag{5.23}$$

with the initial conditions

$$\begin{aligned} u_k(0) &= 0 && \text{for } k \neq 1 \\ u_t(0) &= i/2 && u_{-1}(0) = -i/2, \end{aligned}$$

where $\mathbf{u} = [u_k]$ and $N_k(\mathbf{u},\mathbf{u})$ represents nonlinear term which follows from the convolution product.

Now assume $M \ge N$ and set the following parameters

$$\begin{aligned} \widetilde{u_k} &= u_k && |k| \le N \\ \widetilde{u_k} &= 0 && N < |k| \le M. \end{aligned}$$

Next, Basdevant *et al* set out the following algorithm:

Algorithm 3: Computing the Discrete Fourier Transform (DFT)

1. Perform an in inverse Discrete Fourier Transform (DFT) using Fast Fourier algorithm (FFT) of length M on $\widetilde{u_k}$ to obtain $u(x_j), x_j = 2\pi j/2M, 0 \le j \le 2M-1$;
2. Compute $w(x_j) = u(x_j)^2, 0 \le j \le 2M-1$;
3. Perform the direct DFT of length M on the $w(x_j)$ values to produce $u * u(k) = w_k, -M \le k \le M$;

The DFT will convert the problem into the frequency domain where algebraic manipulations can be performed.

The scheme is the Galerkin method if $M > \frac{3}{2}N$ and

$$N_k(\mathbf{u},\mathbf{u}) = \sum_{p=-N}^{N} u_p u_{k-p}.$$

On the other hand, the pseudo-spectral scheme is

$$\begin{aligned} N_k(\mathbf{u},\mathbf{u}) = &\sum u_p u_q, \\ &|p|,|q| \le N \\ &k = p+q+2Me, \end{aligned}$$

when $\frac{3}{2}N \ge M \ge N$, where e may be 0 or ± 1.

Next, rewrite (5.23) as

$$\frac{\mathrm{d}u_k}{\mathrm{d}t} = \mathcal{N}_k(\mathbf{u},\mathbf{u}) + \lambda u_k, \qquad \mathcal{N}_k = -\frac{ik}{2}N_k,$$

$$\lambda = -\kappa' k^2. \tag{5.24}$$

Integrating (5.24) with respect to time yields

$$u_k(t+\Delta t) = u_k(t-\Delta t)e^{2\lambda\Delta t} + \int_{t-\Delta t}^{t+\Delta t} \mathcal{N}_k(\mathbf{u},\mathbf{u})(\tau)e^{\lambda(t+\Delta t-\tau)}\,\mathrm{d}\tau.$$

Finally, applying the trapezoidal rule gives the final explicit equation for the spectral method

$$u_k(t+\Delta t) = u_k(t-\Delta t)e^{2\lambda\Delta t} + 2\Delta t e\lambda\Delta t\mathcal{N}_k(\mathbf{u},\mathbf{u})(t).$$

Chapter 6

Results

This section will be comprised of a quantitative and qualitative review of the analytical solution and numerical methods developed as part of this project. First the graphs of the analytic solution to the viscous Burgers' equation will be presented and commented on. Then the rest of the section will focus on the numerical methods, in this case the finite differences and finite elements schemes.

The computer code that I used to produce the following images and tables is presented in Appendix A. An example of how one would implement each function is given in Listing A.11.

6.1 Analytical Solution

Graphs of the analytical solution are produced using the code from Listing A.1 and the code for applying the Cole-Hopf transformation is provided in Listing A.3.

The graphs and tables throughout this section will be based on one of two initial conditions. For clarity we will define the following sinusoidal initial condition as initial condition 1

$$u(x,0) = \sin(\pi x), \quad 0 \leq x \leq 1.$$

The remaining condition is

$$u(x,0) = 4x(1-x), \quad 0 \leq x \leq 1,$$

and this parabolic equation will be referred to as initial condition 2. The spatial domain will be of unit length, which means $L = 1$, so therefore $x \in [0,1]$. Usually the temporal domain is given by $[0,\tau]$. For most of the following cases, a choice of τ is arbitrary and just done so the graphs are displayed nicely. Different values of τ will be used in the data portion of this chapter however.

Following the last image will be explanations and a discussion (see Figures 6.1 - 6.10).

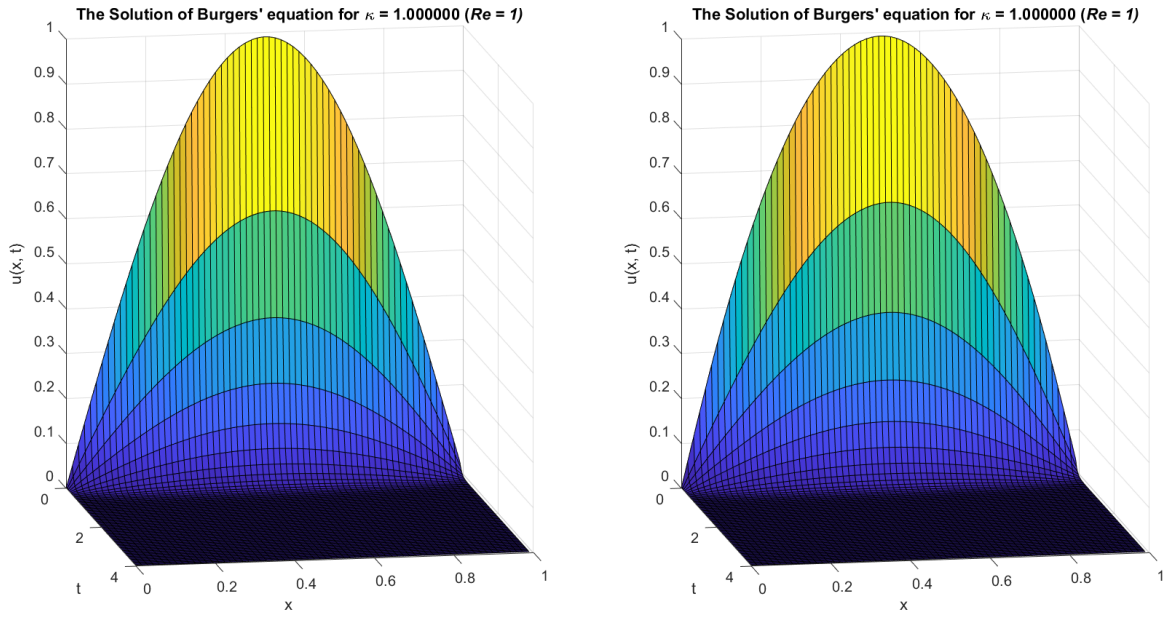

Figure 6.1: The approximated analytical solution of Burgers' equation with $Re = 1$ for initial conditions 1 and 2 on the left and right respectively.

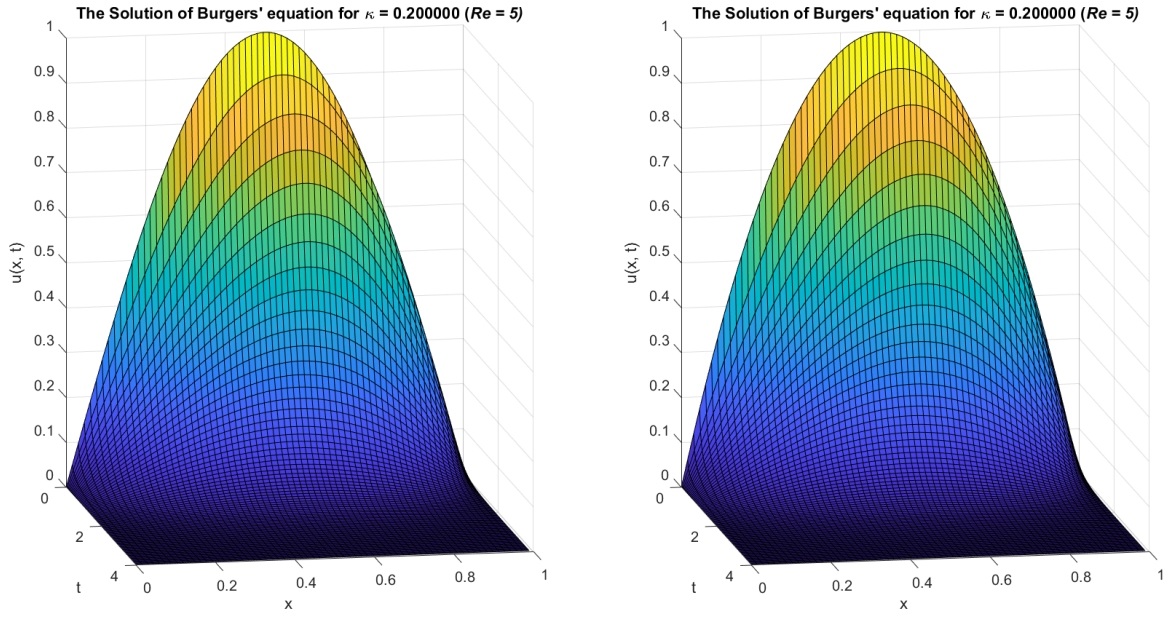

Figure 6.2: The approximated analytical solution of Burgers' equation with $Re = 5$ for initial conditions 1 and 2 on the left and right respectively.

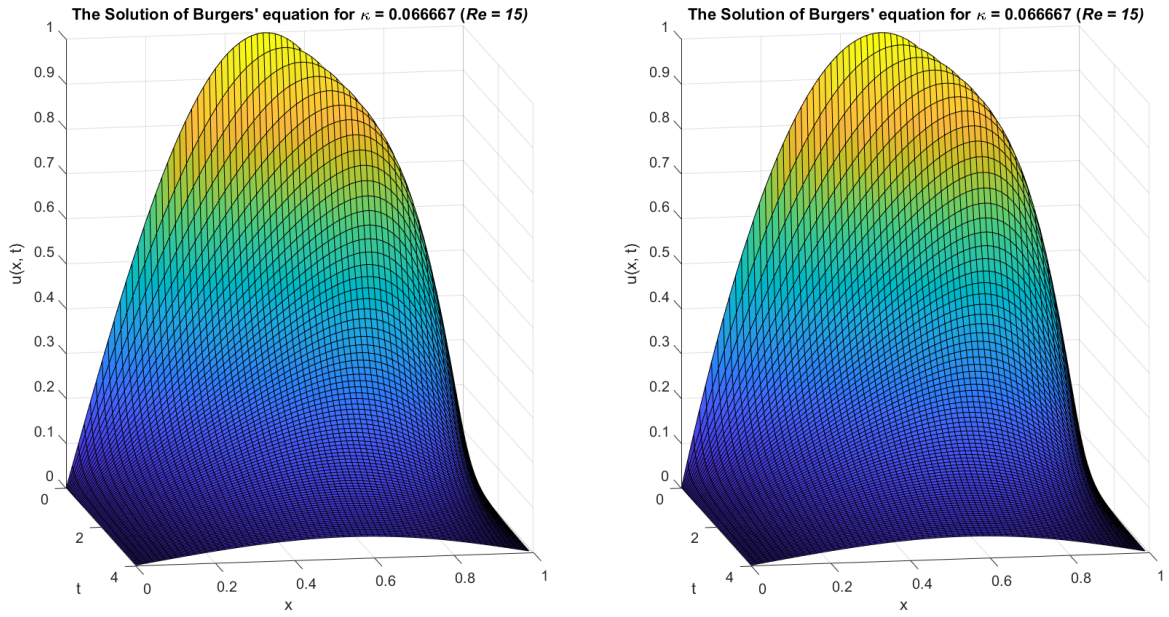

Figure 6.3: The approximated analytical solution of Burgers' equation with $Re = 15$ for initial conditions 1 and 2 on the left and right respectively.

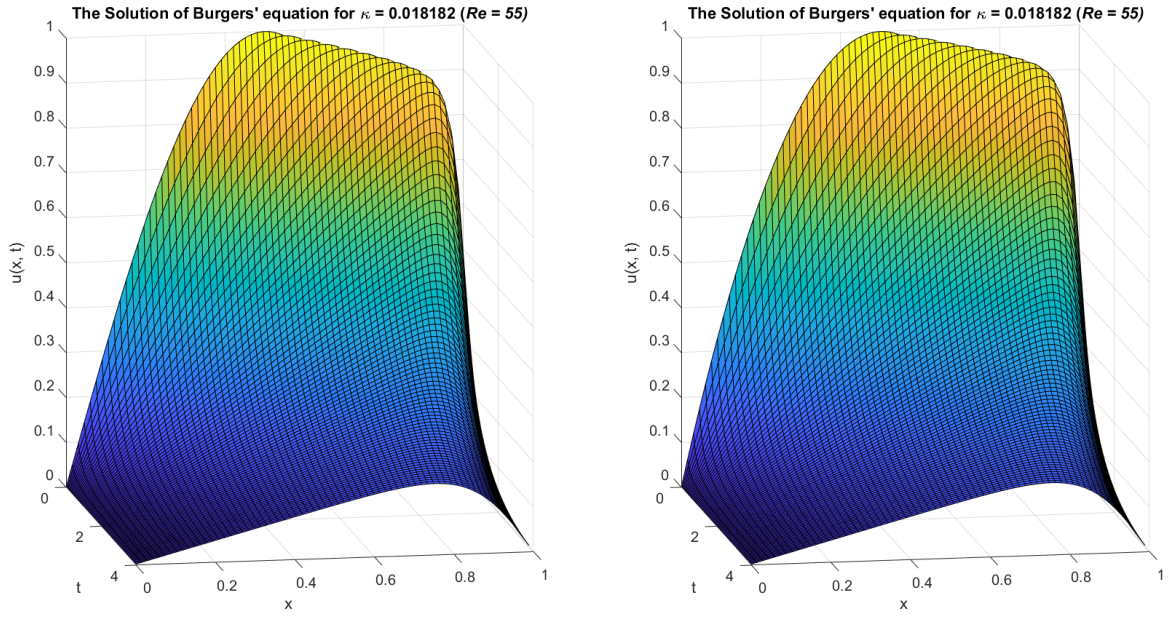

Figure 6.4: The approximated analytical solution of Burgers' equation with $Re = 55$ for initial conditions 1 and 2 on the left and right respectively.

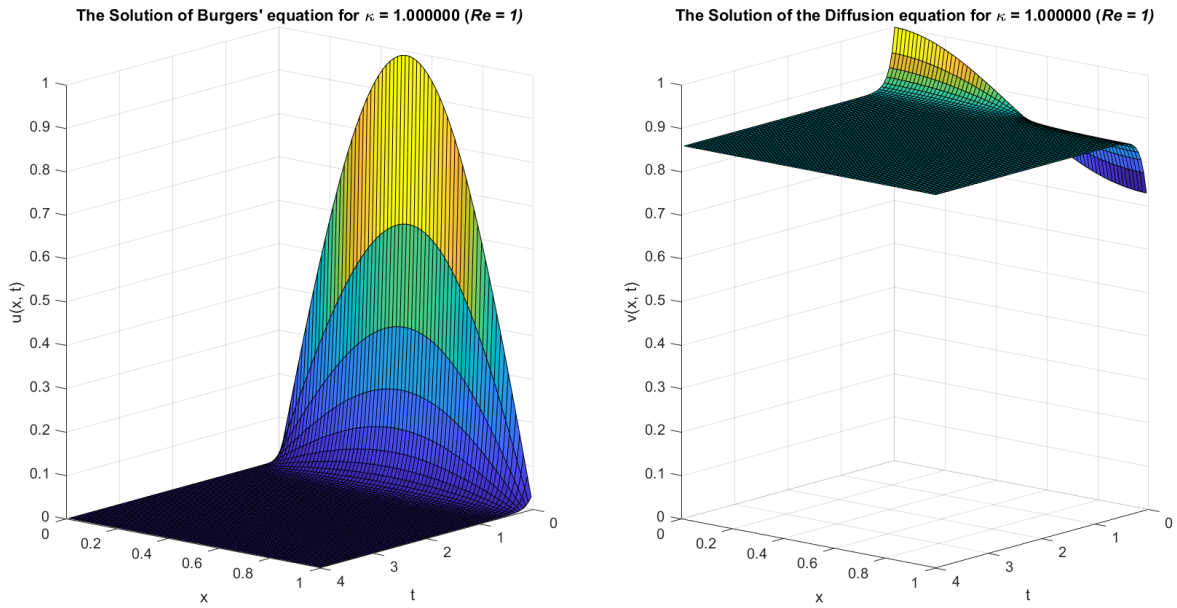

Figure 6.5: For a $Re = 1$, on the left is the approximated analytical solution of Burgers' equation with initial conditions 1 and on the right is the corresponding approximated analytical solution of the diffusion equation with $v(x,0) = \exp\left(\frac{\cos(\pi x)-1}{2\kappa\pi}\right)$.

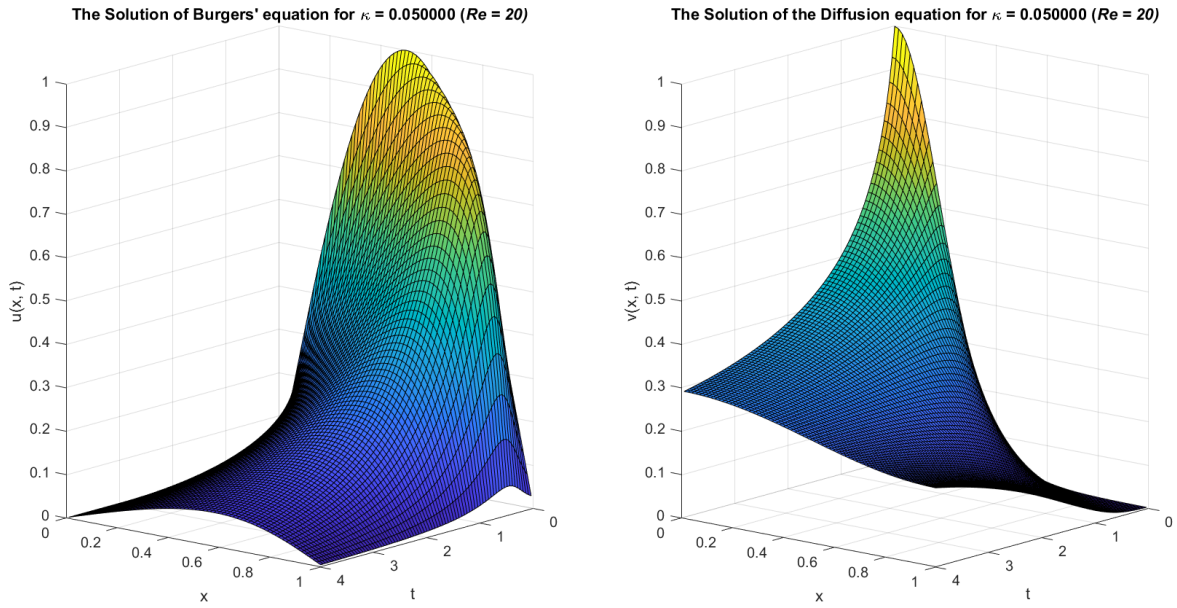

Figure 6.6: For a $Re = 20$, on the left is the approximated analytical solution of Burgers' equation with initial conditions 1 and on the right is the corresponding approximated analytical solution of the diffusion equation with $v(x,0) = \exp\left(\frac{\cos(\pi x)-1}{2\kappa\pi}\right)$.

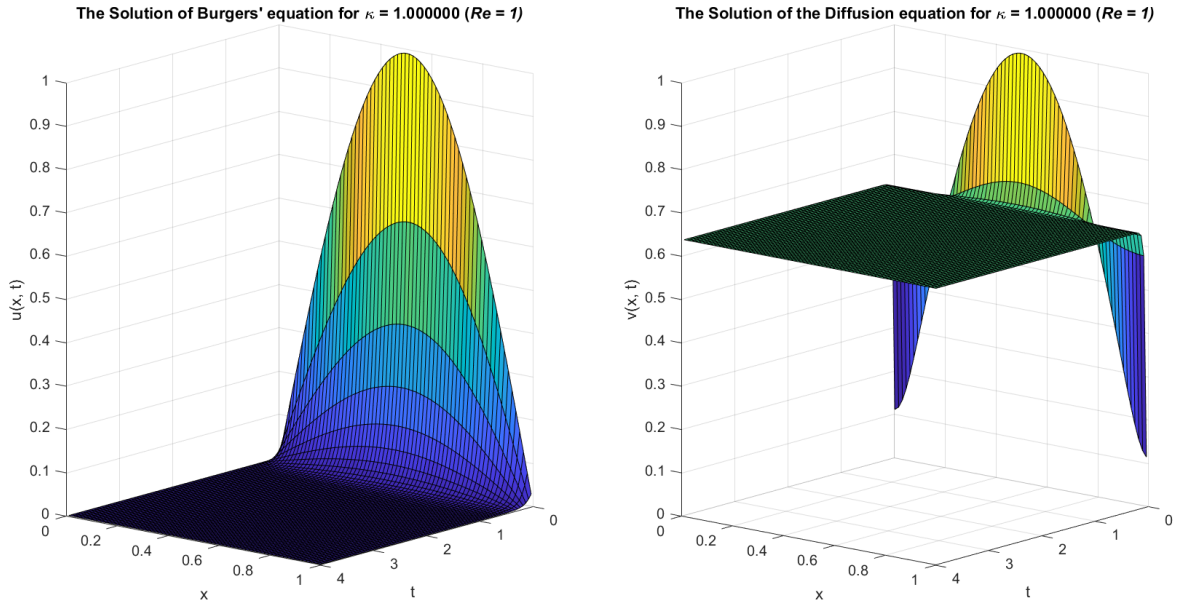

Figure 6.7: For a $Re = 1$, on the left is the approximated analytical solution of Burgers' equation with initial conditions 1 and on the right is the approximated analytical solution of the diffusion equation with initial condition 1 as well.

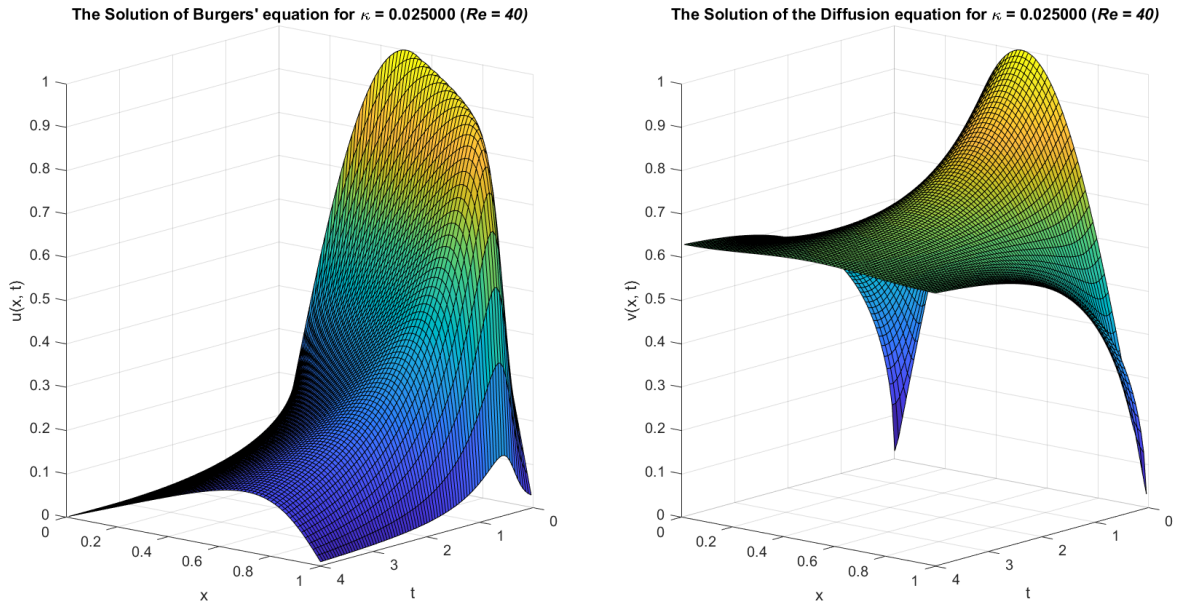

Figure 6.8: For a $Re = 40$, on the left is the approximated analytical solution of Burgers' equation with initial conditions 1 and on the right is the approximated analytical solution of the diffusion equation with initial condition 1 as well.

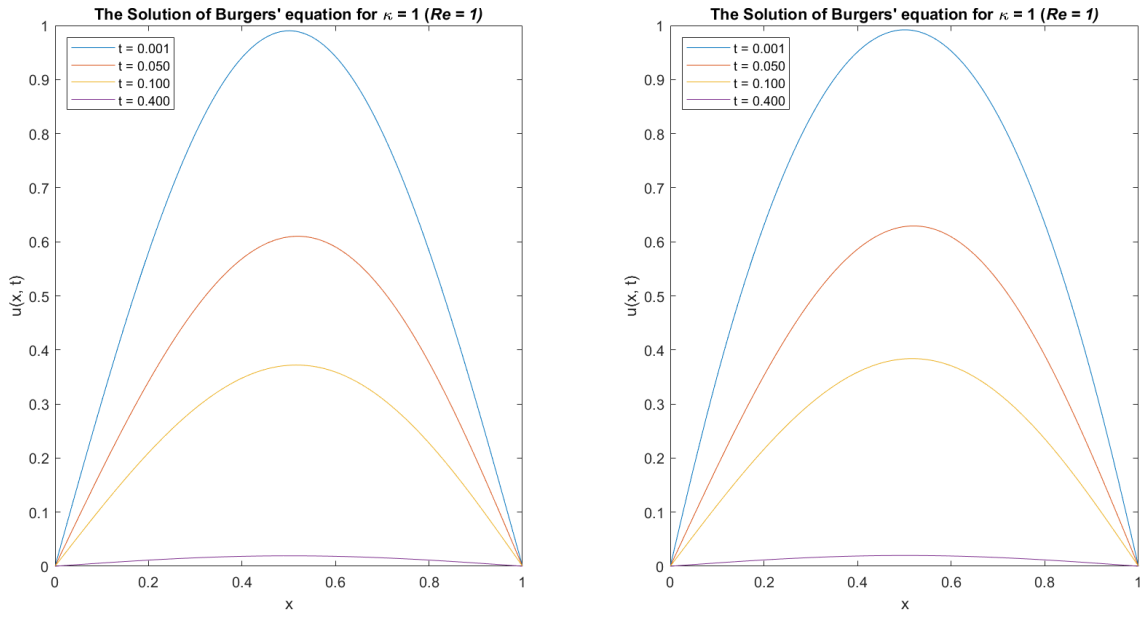

Figure 6.9: The approximated analytical solution of Burgers' equation at various times t, with $Re = 1$ for initial conditions 1 and 2 on the left and right respectively.

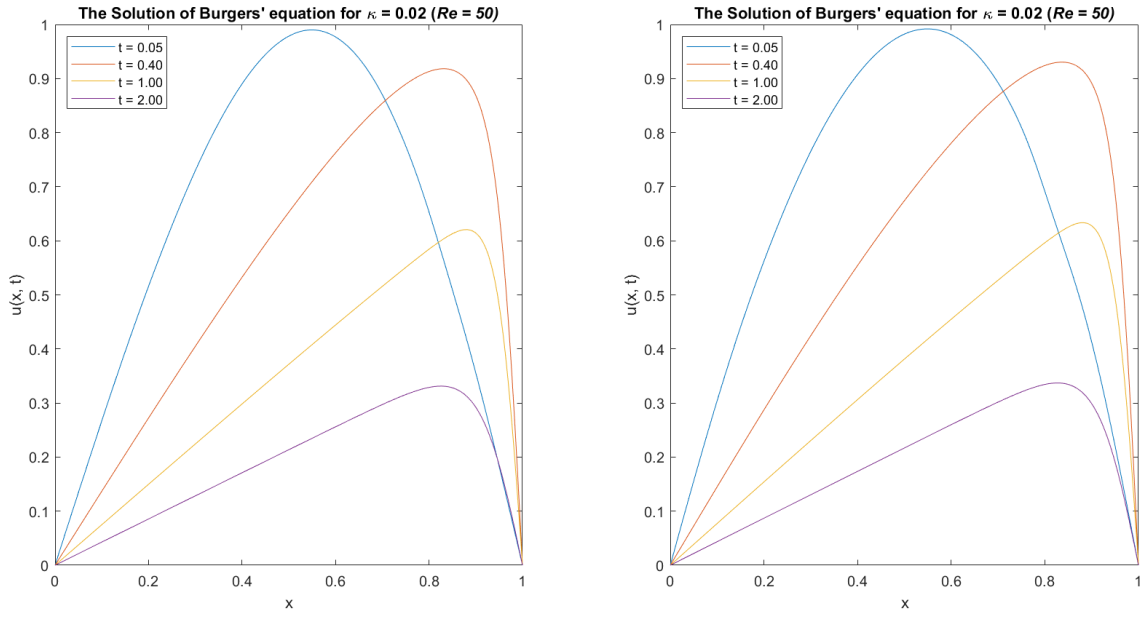

Figure 6.10: The approximated analytical solution of Burgers' equation at various times t, with $Re = 55$ for initial conditions 1 and 2 on the left and right respectively.

First we see in Figures 6.1 through to 6.4, the surface plot of the analytical solution of the Viscous Burgers' equation. In order, these plots are computed with a Reynolds number of $Re = 1, 5, 15$, and 55. For this level of viscosity, we notice that the sinusoidal and parabolic initial conditions are indistinguishable from one another. Another common behaviour for each of the plots is that the solution converges to 0 as time increases. An increase of the Reynolds number represents more turbulent flows, and as expected, the solution dissipates more slowly. The curious behavior is the fact that the crest of the wave travels away from the initial position. This characteristic is a primitive feature of shock formation; it is primitive because the Reynolds number is still quite small, but nonetheless, the solution still dissipates and converges to 0. This behaviour is due to the nature of Burgers' equation as it represents a fluid that has negligible change in pressure internally within the fluid. This basically means that the particles of fluid are not showing signs of applying forces to one another that might manifest as sporadic behaviour. Instead, Burgers' equation models a fluid that is, let us say, "well behaved," and flows smoothly.

Recall that the initial condition of the diffusion equation is a Cole-Hopf transformation of the original sine initial condition of Burgers' equation (See Figures 6.5 & 6.6). Clearly the curve at $t = 0$ behaves as an exponential function, which is to be expected. A big difference between the two plots is that the diffusion equation does not approach 0 as time increases, which is what Burgers' equation does. Instead, the solution to the diffusion equation seems to approach 8.5 approximately; this seems like it could be the average of the initial condition. The physical meaning of the diffusion equation is that given some initial profile of heat, the heat should diffuse into the rest of the domain as time gets larger. In other words, the amount of heat energy at the initial time will be conserved (because energy is conserved), such that as time gets large, the temperature in every position is uniform. Notice in Figure 6.6 that the increase in the Reynolds number shifts the initial condition vertically downwards, which effects the average function value. In addition, the diffusion process is occurring more slowly.

The final two surface plots depict the solution of Burgers' equation and the heat equation given initial condition 1. Both Figure 6.7 & 6.8 clearly have exactly the same initial condition. Despite the different values for the dissipation coefficient in each of the plots, the heat equation converges to the same value for both graphs. The explanation is that the initial condition for both cases is the same, and thus the average function value is the same. The average function value is computed as follows

$$f_{avg} = \frac{1}{b-a}\int_a^b f(x)\,\mathrm{d}x = \frac{1}{1-0}\int_0^1 \sin(\pi x)\,\mathrm{d}x = \frac{2}{\pi} \approx 0.636619.$$

This calculation matches the value of which the solution to the diffusion equation is approaching as time increases.

The last two graphs regarding the analytical solution of Burgers' equation are Figures 6.9 & 6.10. Instead of a surface plot, the graphs depict univariate functions. In our case this shows the solution of Burgers' equation for initial conditions 1 and 2, plotted against the position. On the same graph, several different instances of time are shown so that we can more clearly see how the solution evolves over time. In Figure 6.9 the Reynolds number is 1, so the solution just dissipates while the centre is fixed in space. Next, Figure 6.10 shows us how the crest of the wave is travelling away from the initial position. This is the beginning of a shock formation. Due to the shape of this wave at $t = 0.40$, it is commonly refereed to as a triangle wave. It is a rather famous plot associated with Burgers' equation.

Before moving on to numerical methods, an important question to ask is "when would shock formation occur in a real life scenario?" To answer this question, we must first introduce a concept called the Mach number.

The Mach number (Ma) is defined by the following

$$Ma = \frac{u}{c}$$

where c is the speed of sound in some medium and u is the flow velocity. This quantity is a ratio of some velocity to the speed of sound. For instance, if u were to equal the speed of sound, then u would be describe as travelling at Mach 1 because $Ma = 1$. To quantify this value, the speed of sound through air at 20 °C is approximately 343 m/s. Supersonic speeds would refer to velocities that exceed Mach 1.

Shock formation occurs when the flow speed of some fluid is supersonic. Realistically in fluid dynamics, this behaviour would be exhibited by gases (as opposed to liquids—but in principle they are technically not excluded).

When a given initial condition is discontinuous, then shock formation may also occur. Similar characteristics could be observed in a liquid, if discontinuous initial conditions could be mimicked. This could plausibly be done by a container, or even an object obstructing the path of the fluid.

6.2 Numerical Methods

We now turn our focus to a discussion of numerical methods, and how well they compared to the approximated analytical solution. This section begins with the finite differences method and will be concluded with finite elements.

6.2.1 Finite Differences

First, Tables 6.1 - 6.6 will be used to report differences in the computation of the numerical method. Images of the absolute error with respect to the solution will follow in Figures 6.11 - 6.14. The discussion will be presented after the images.

Table 6.1: Absolute error in the finite difference approximation for initial condition 1, with $\kappa = 1$ ($Re = 1$), $\tau = 0.1$, and $\Delta t = 0.001$

x	Computed error for different values of N				
	$N = 10$	$N = 20$	$N = 40$	$N = 80$	$N = 160$
0.1	0.000906	0.000226	0.000057	0.000015	0.000005
0.2	0.001738	0.000434	0.000110	0.000029	0.000009
0.3	0.002425	0.000606	0.000154	0.000041	0.000012
0.4	0.002900	0.000725	0.000183	0.000048	0.000014
0.5	0.003110	0.000777	0.000196	0.000051	0.000015
0.6	0.003018	0.000753	0.000190	0.000049	0.000014
0.7	0.002615	0.000652	0.000164	0.000042	0.000012
0.8	0.001928	0.000481	0.000121	0.000031	0.000009
0.9	0.001024	0.000255	0.000064	0.000016	0.000005

Table 6.2: Absolute error in the finite difference approximation for initial condition 1, with $\kappa = 0.066666$ ($Re = 15$), $\tau = 0.1$, and $\Delta t = 0.001$

x	Computed error for different values of N				
	$N = 10$	$N = 20$	$N = 40$	$N = 80$	$N = 160$
0.1	0.014794	0.003655	0.000911	0.000228	0.000057
0.2	0.022725	0.005690	0.001423	0.000356	0.000089
0.3	0.019162	0.004898	0.001231	0.000308	0.000077
0.4	0.003048	0.000852	0.000219	0.000055	0.000013
0.5	0.022658	0.005799	0.001458	0.000365	0.000092
0.6	0.050970	0.013174	0.003322	0.000832	0.000208
0.7	0.071358	0.018352	0.004622	0.001157	0.000289
0.8	0.072042	0.018270	0.004583	0.001147	0.000286
0.9	0.046133	0.011520	0.002878	0.000719	0.000180

Table 6.3: Absolute error in the finite difference approximation for initial condition 1, with $\kappa = 0.066666$ ($Re = 15$), $\tau = 0.01$, and $\Delta t = 0.001$

x	Computed error for different values of N				
	$N = 10$	$N = 20$	$N = 40$	$N = 80$	$N = 160$
0.1	0.173972	0.007713	0.002060	0.000516	0.000129
0.2	0.344758	0.009959	0.002699	0.000676	0.000169
0.3	0.502989	0.004394	0.001324	0.000330	0.000082
0.4	0.630404	0.006816	0.001497	0.000376	0.000095
0.5	0.702953	0.018272	0.004377	0.001097	0.000275
0.6	0.699441	0.024665	0.005976	0.001497	0.000374
0.7	0.611716	0.023865	0.005780	0.001446	0.000361
0.8	0.449751	0.017505	0.004225	0.001055	0.000264
0.9	0.236977	0.008852	0.002130	0.000531	0.000133

Table 6.4: Absolute error in the finite difference approximation for initial condition 2, with $\kappa = 1$ ($Re = 1$), $\tau = 0.1$, and $\Delta t = 0.001$

x	Computed error for different values of N				
	$N = 10$	$N = 20$	$N = 40$	$N = 80$	$N = 160$
0.1	0.000929	0.000232	0.000059	0.000016	0.000005
0.2	0.001785	0.000446	0.000113	0.000030	0.000009
0.3	0.002496	0.000624	0.000158	0.000042	0.000013
0.4	0.002991	0.000747	0.000189	0.000050	0.000015
0.5	0.003212	0.000802	0.000203	0.000053	0.000016
0.6	0.003119	0.000778	0.000196	0.000051	0.000015
0.7	0.002703	0.000674	0.000170	0.000044	0.000012
0.8	0.001992	0.000497	0.000125	0.000032	0.000009
0.9	0.001057	0.000264	0.000066	0.000017	0.000005

Table 6.5: Absolute error in the finite difference approximation for initial condition 2, with $\kappa = 0.066666$ ($Re = 15$), $\tau = 0.1$, and $\Delta t = 0.001$

x	Computed error for different values of N				
	$N = 10$	$N = 20$	$N = 40$	$N = 80$	$N = 160$
0.1	0.017010	0.004141	0.001029	0.000257	0.000065
0.2	0.024796	0.006236	0.001561	0.000390	0.000098
0.3	0.018704	0.004847	0.001223	0.000306	0.000076
0.4	0.000417	0.000045	0.000007	0.000002	0.000001
0.5	0.027630	0.007142	0.001801	0.000451	0.000113
0.6	0.055343	0.014321	0.003613	0.000905	0.000226
0.7	0.074181	0.019016	0.004784	0.001198	0.000299
0.8	0.074236	0.018751	0.004699	0.001175	0.000294
0.9	0.048359	0.012055	0.003010	0.000752	0.000188

Table 6.6: Absolute error in the finite difference approximation for initial condition 2, with $\kappa = 0.066666$ ($Re = 15$), $\tau = 0.01$, and $\Delta t = 0.001$

x	Computed error for different values of N				
	$N = 10$	$N = 20$	$N = 40$	$N = 80$	$N = 160$
0.1	0.195240	0.010842	0.002885	0.000724	0.000181
0.2	0.375962	0.010284	0.002786	0.000697	0.000174
0.3	0.530202	0.002078	0.000735	0.000182	0.000045
0.4	0.645502	0.009358	0.002137	0.000537	0.000135
0.5	0.707802	0.019364	0.004643	0.001163	0.000291
0.6	0.705190	0.024600	0.005950	0.001490	0.000372
0.7	0.631172	0.023945	0.005791	0.001449	0.000362
0.8	0.485441	0.018302	0.004414	0.001103	0.000275
0.9	0.271055	0.009275	0.002216	0.000552	0.000138

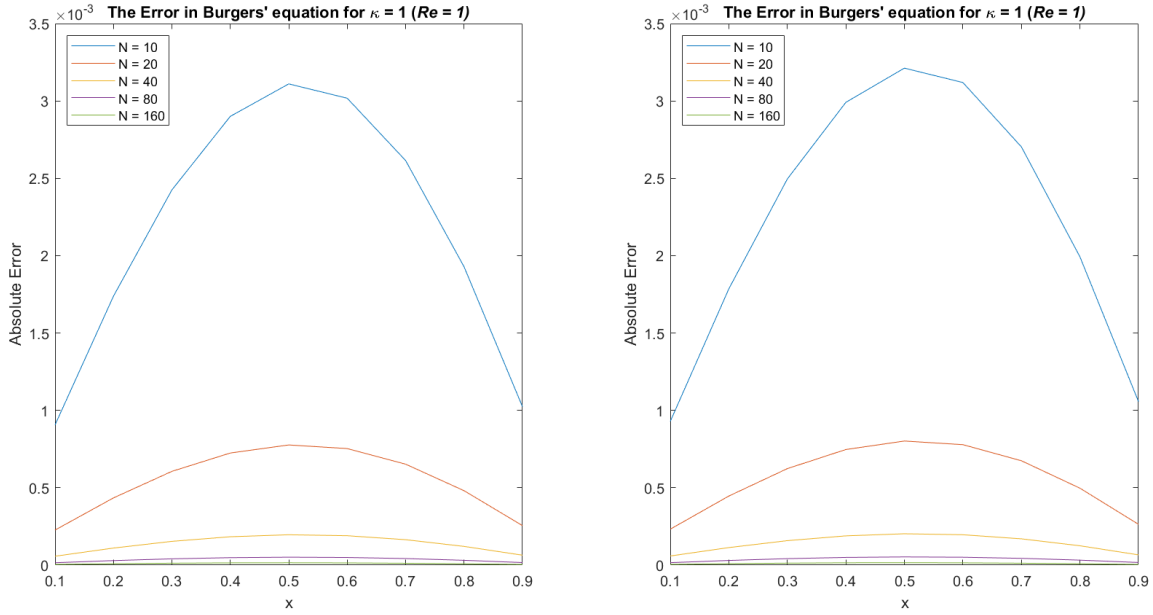

Figure 6.11: Absolute error in the finite difference approximation with $\kappa = 1$ ($Re = 1$), $\tau = 0.1$, and $\Delta t = 0.001$, for initial condition 1 and 2 on the left and right, respectively.

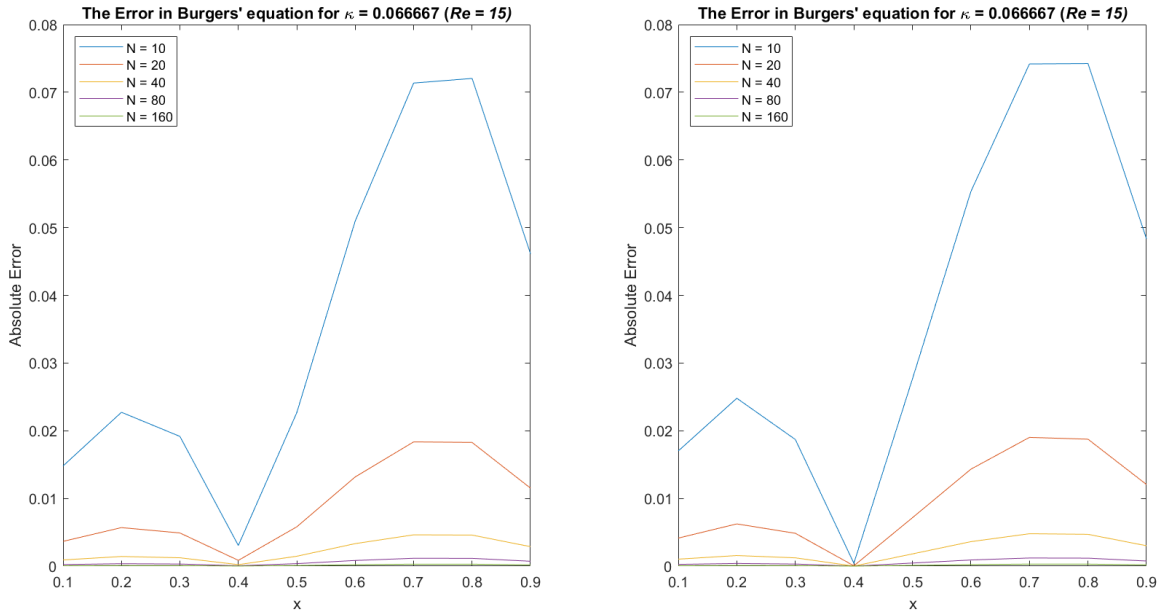

Figure 6.12: Absolute error in the finite difference approximation with $\kappa = 0.066666$ ($Re = 15$), $\tau = 0.1$, and $\Delta t = 0.001$, for initial condition 1 and 2 on the left and right, respectively.

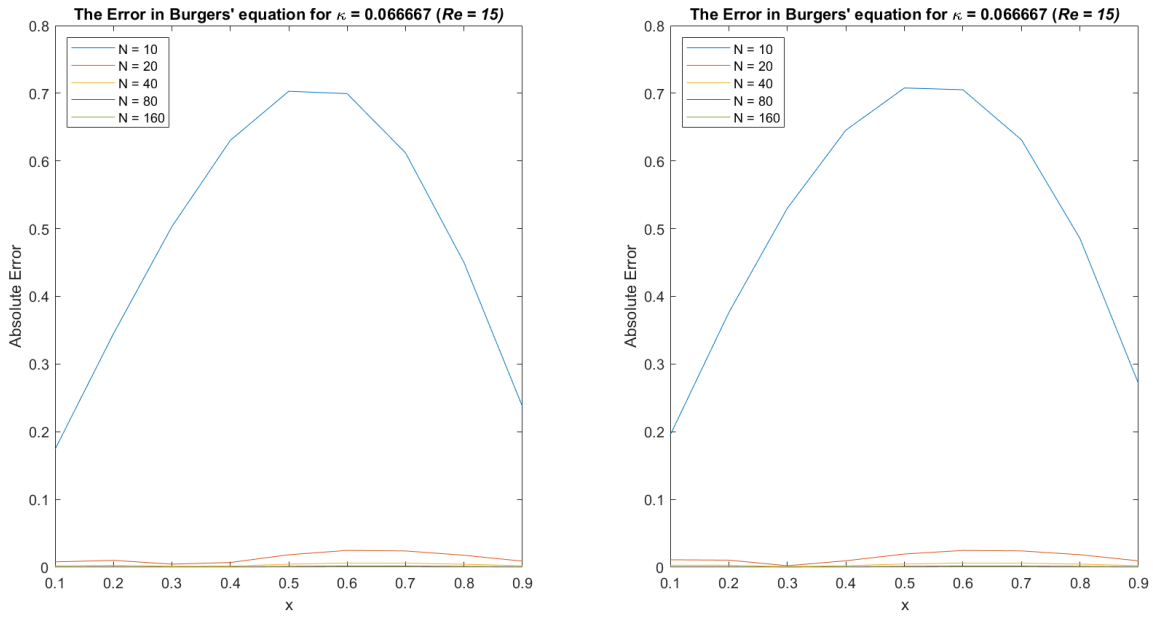

Figure 6.13: Absolute error in the finite difference approximation with $\kappa = 0.066666$ ($Re = 15$), $\tau = 0.01$, and $\Delta t = 0.001$, for initial condition 1 and 2 on the left and right, respectively.

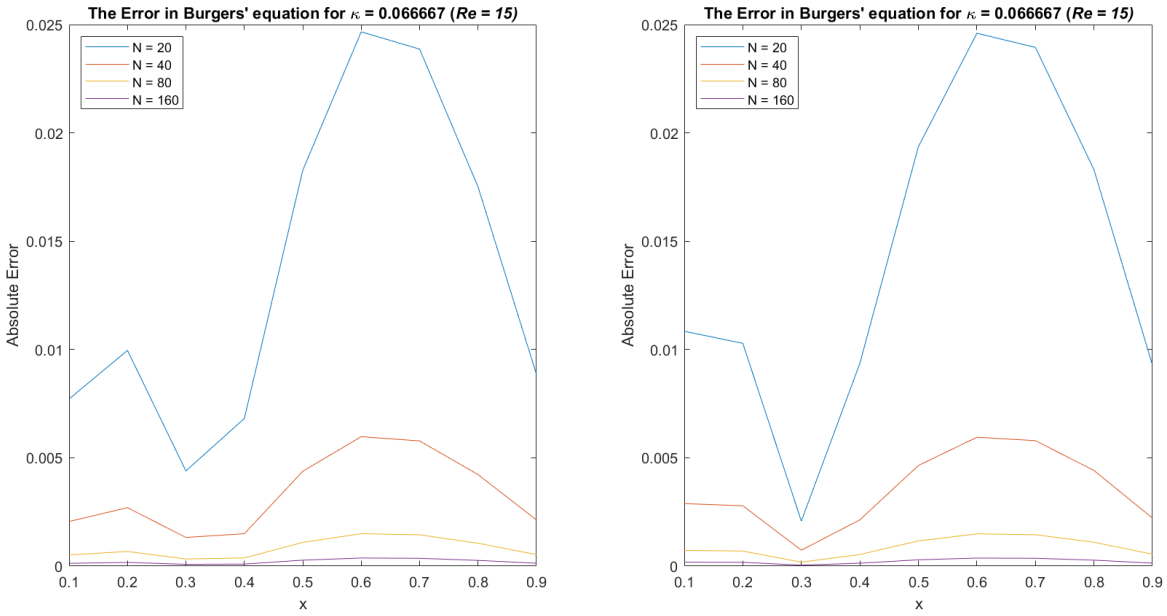

Figure 6.14: Absolute error in the finite difference approximation with $\kappa = 0.066666$ ($Re = 15$), $\tau = 0.01$, and $\Delta t = 0.001$, for initial condition 1 and 2 on the left and right, respectively** ($N = 10$ omitted).

All of the data and figures in Section 6.2.1 are computed using Listing A.7. The version of the Cole-Hopf transformation computed via the numerical ODE solver (`ode45`) was used (see Listing A.4). The matlab code in question yields a solution to the diffusion equation, so thus a discretized Cole-Hopf transformation must be applied, the code for which is given in Listing A.5.

Tables 6.1 - 6.6 all exhibit similar behavior. They all tell us that the Crank-Nicolson finite difference scheme does a fine job approximating the solution of Burgers' equation. When the space step, Δx, is large, (N is small), then the approximation isn't too accurate. But as Δx decreases, so to does the error. Observe that in some cases by halving the space step, we are almost picking up two digits of accuracy. This is behaviour of quadratic convergence, which is exactly what we expect for the method. Even the discretized Cole-Hopf transformation (5.4) has a quadratic order of accuracy (excluding the boundary) because it uses a centred difference approximation. We also notice that as the dissipation coefficient, κ, gets small, the approximation is less accurate. But similarly, as Δx decreases, the absolute error decreases as well.

The graphs that follow (see Figures 6.11 - 6.14) model the data in the tables. I will reiterate what was said above, but clearly from the graphs we see that as N increases, the absolute error decreases. For the $\kappa = 1$ case seen in Figure 6.11 has a particularly small error. As expected for a larger Reynolds number, the error starts off larger. Take a look at Figure 6.13. The first value of the error for $N = 10$ is quite a lot larger than the rest. In order to get a better understanding of the other error, the plot is recomputed without the first curve, given by Figure 6.14. The value for the absolute error is completely reasonable in this diagram. It can be concluded in general that the finite difference approximation to Burgers' equation via a Crank-Nicolson scheme is quite effective.

When solving PDEs numerically the Courant-Friedrichs-Lewy (CFL) condition is often important to check. It is a quantity that is a necessary condition for convergence. It usually arises when solving a PDE using an explicit time-step type scheme (such as finite differences). As reported in [16], the Crank-Nicolson scheme in this case is unconditionally stable. Therefore, for any choice of positive Δx, the approximation will converge to the true solution.

6.2.2 Finite Elements

The finite elements approach gave inconclusive results. When graphed, the results seem to converge to a strange shape and we are not sure why. At first choosing $\theta = 0$ (explicit finite difference) yields a discontinuous graph with large orders of magnitude. Upon further review of [23], we found a stability condition for the method. In other words, the CFL number played a role in convergence for weights of $0 \leq \theta < 1/2$. Recall that r is defined as

$$r = \frac{\kappa \Delta t}{\Delta x^2}.$$

Then the finite element equation (5.17) is stable for

$$0 < r < \frac{1}{6(1-2\theta)} \qquad \text{if} \qquad 0 \leq \theta < 1/2.$$

Otherwise when $1/2 \leq \theta \leq 1$, the finite element equation is stable for positive r.

After creating a condition that checks for stability in the code, it was found that for any choice of θ, the solution was the same, however, still incorrect. The code I used to attempt to compute the solution for the finite elements approach is given in Listing A.8 (the numerical Cole-Hopf transformtion was used, namely Listing A.4).

Within the [23], Öziş *et al* mention a method for solving the system of equations called the Crout method. In an attempt to troubleshoot, I implemented this method as an alternative (see both Listing A.9 & A.10). The Crout method faired no better, and alas, the finite element method still did not converge to the true solution. It is worth mentioning though that the Crout method implementation did yield similar results compared to the original method. This suggests that it is quite unlikely that the source of our errors occur in the solving of the linear system (since we solved it two different ways and they both match).

The problem with the implementation of this method occurs at the second time step. When the matrix which is suppose to represent the solution to the heat equation is graphed, the initial condition is clearly correct, but the next time step shows signs of the answer already beginning to diverge. The problem seems to be with the system of equations that is being solved.

Chapter 7

Conclusion

The viscous Burgers' equation has been comprehensively derived and solved. Physical and mathematical significances have been briefly touched upon. The Cole-Hopf transformation was introduced and implemented as the primary tool for solving Burgers' equation. Multiple versions of an analytical solution have been given, which include consideration on an infinite domain leading to a convolution with the heat kernel, as well as finite domains using Fourier series. The latter was used as a benchmark for analysis and comparison to the numerical solutions, by which graphs were presented along with a collection of data. Explanations for several numerical techniques were talked about. These include finite differences, finite elements, and spectral methods. My implementation of a Crank-Nicolson finite difference scheme was given in detail, as well graphs and numerical results. My approach for applying a finite elements scheme was shared, however a solution was not found via this technique. A brief discussion about the failures to implement this method is given.

One area I would have liked to explore in this project is investigation into the different matrix norms. For instance I calculated the ℓ_1 and ℓ_∞ norm of matrix C (5.20), K (5.21), D, and E for the finite element approach (see Table 7.1 below). First of all, it turns out that

Table 7.1: Matrix norms for Finite Element Scheme

Matrix	ℓ_1 norm (ℓ_∞ norm)
C	h
K	$\frac{4}{h}$
D	$\frac{4(h^2+3\kappa\theta\Delta t)}{3h}$
E	$\frac{4(h^2-3\kappa(1-\theta)\Delta t)}{3h}$

in each case, $\ell_1 = \ell_\infty$. These quantities were calculated by finding the maximum absolute

column sum (ℓ_1 norm) and the maximum absolute row sum (ℓ_∞ norm). Unfortunately the impact and significance of these values were not explored. Computing the matrix norms of the tridiagonal matrix in the finite difference scheme could have been done as well. In terms of algorithm development, if would have been useful to have convergence to the analytical solution of Burgers equation (Listing A.6 & A.2) based on some tolerance level based on checking the matrix norm of the difference between two iterates.

This project could have explored the significance of Burgers' equation more. There were a number of instances in the literature that contained information on the physical significance of Burgers' equation, but I deemed that it would seem like mindless regurgitation to write about every instance. Even with this being shared, I believe that research papers centred on this topic in the future should do a better job of establishing physical significance whenever possible, because the current literature seems to lack it.

A project with an additional semester of time would have certainly seen the implementation of several more numerical methods. The hope was that upon completion of the project that we could compare the performance of numerical methods with one another. Unfortunately this goal did not come to fruition. In light of this fact, the discussion was as thoughtful and compressive as possible.

In conclusion, the viscous Burgers' equation is a well documented mathematical jewel. There is not doubt that with the rise of computers, there will be a increase in the development of nonlinear numerical solvers. The presence of Burgers' equation will rise with it.

Bibliography

[1] H Aref and PK Daripa. Note on finite difference approximations to burgersâĂŹ equation. *SIAM journal on scientific and statistical computing*, 5(4):856–864, 1984.

[2] Timothy Barth and Jonas Sukys. On the calculation of exact cumulative distribution statistics for burgers equation. 2018.

[3] Cea Basdevant, M Deville, P Haldenwang, JM Lacroix, J Ouazzani, R Peyret, Paolo Orlandi, and AT Patera. Spectral and finite difference solutions of the burgers equation. *Computers & fluids*, 14(1):23–41, 1986.

[4] Harry Bateman. Some recent researches on the motion of fluids. *Monthly Weather Review*, 43(4):163–170, 1915.

[5] Jérémie Bec and Konstantin Khanin. Burgers turbulence. *Physics Reports*, 447(1-2):1–66, 2007.

[6] Mayur P Bonkile, Ashish Awasthi, C Lakshmi, Vijitha Mukundan, and VS Aswin. A systematic literature review of burgersâĂŹ equation with recent advances. *Pramana*, 90(6):69, 2018.

[7] Johannes Martinus Burgers. Mathematical examples illustrating relations occurring in the theory of turbulent fluid motion. In *Selected Papers of JM Burgers*, pages 281–334. Springer, 1995.

[8] Alexandre Joel Chorin and Jerrold E Marsden. *A mathematical introduction to fluid mechanics*, volume 3. Springer, 1990.

[9] Julian D Cole. On a quasi-linear parabolic equation occurring in aerodynamics. *Quarterly of applied mathematics*, 9(3):225–236, 1951.

[10] Matthew P Coleman. *An introduction to partial differential equations with MATLAB*. Chapman and Hall/CRC, 2016.

[11] Weinan E, Konstantin Khanin, Alexandre Mazel, and Yakov Sinai. Probability distribution functions for the random forced burgers equation. *Phys. Rev. Lett.*, 78:1904–1907, Mar 1997.

[12] David Gottlieb, (joint author.) Orszag, Steven A, and National Science Foundation (U.S.). *Numerical analysis of spectral methods : theory and applications*. Philadelphia : Society for Industrial and Applied Mathematics, 1977. "Based on a series of lectures presented ... at the NSF-CBMS regional conference held at Old Dominion University from August 2-6, 1976.".

[13] BD Greenshields, Ws Channing, Hh Miller, et al. A study of traffic capacity. In *Highway research board proceedings*, volume 1935. National Research Council (USA), Highway Research Board, 1935.

[14] SN Gurbatov and AI Saichev. Probability distributions and spectra of potential hydrodynamic turbulence. *Radiofizika*, 27:456–468, 1984.

[15] Eberhard Hopf. The partial differential equation ut+ uux= μxx. *Communications on Pure and Applied mathematics*, 3(3):201–230, 1950.

[16] Mohan K Kadalbajoo and Ashish Awasthi. A numerical method based on crank-nicolson scheme for burgersâĂŹ equation. *Applied mathematics and computation*, 182(2):1430–1442, 2006.

[17] Jirair Kevorkian. *Partial Differential Equations: Analytical Solution*. Springer, New York, NY, 1990.

[18] Martin J Klein. Paul ehrenfest, niels bohr, and albert einstein: Colleagues and friends. *Physics in Perspective*, 12(3):307–337, 2010.

[19] Gunilla Kreiss and Heinz-Otto Kreiss. Convergence to steady state of solutions of burgers' equation. *Applied Numerical Mathematics*, 2(3-5):161–179, 1986.

[20] Mikel Landajuela. Burgers equation. *BCAM Internship-summer*, 2011.

[21] SA Molchanov, D Surgailis, and WA Woyczynski. The large-scale structure of the universe and quasi-voronoi tessellation of shock fronts in forced burgers turbulence in rd. *The Annals of Applied Probability*, pages 200–228, 1997.

[22] Toshimitsu Musha and Hideyo Higuchi. Traffic current fluctuation and the burgers equation. *Japanese journal of applied physics*, 17(5):811, 1978.

[23] T Öziş, EN Aksan, and A Özdeş. A finite element approach for solution of burgersâĂŹ equation. *Applied Mathematics and Computation*, 139(2-3):417–428, 2003.

[24] DE Panayotounakos and D Drikakis. On the closed-form solutions of the wave, diffusion and burgers equations in fluid mechanics. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 75(6):437–447, 1995.

[25] Luis G. Reyna and Michael J. Ward. On the exponentially slow motion of a viscous shock. *Communications on Pure and Applied Mathematics*, 48(2):79–120, 1995.

[26] Guolian Wang, Ming Zeng, and Boling Guo. Stochastic burgers' equation driven by fractional brownian motion. *Journal of Mathematical Analysis and Applications*, 371(1):210–222, 2010.

[27] Mingliang Wang. Exact solutions for a compound kdv-burgers equation. *Physics Letters A*, 213(5-6):279–287, 1996.

[28] Wei Wang and AJ Roberts. Diffusion approximation for self-similarity of stochastic advection in burgersâĂŹ equation. *Communications in Mathematical Physics*, 333(3):1287–1316, 2015.

[29] Jeffrey Yepez. An efficient quantum algorithm for the one-dimensional burgers equation. *arXiv preprint quant-ph/0210092*, 2002.

Appendix A

Matlab Code

Listing A.1: Function that approximates the analytical solution of Burgers' Equation

```
function [u, x, t, denominator] = BurgersEq1D(f, c, L, T, dx, dt, N)
% This function solnBurgersEq1D, solves the one-dimentional viscous Burgers' equation using a finite
% Fourier series approximatation to the diffusion equation. This function outputs the solution to the heat
% equation as a matrix of points. In addition the user can also receive the x and t-values that the
% solution is evaluated at. Lastly, the user could be passed the solution to the diffusion equation,
% denoted here by 'denominator'.
%
%         f  = The initial condition passed as an anonymous function
%         c  = The diffusion/viscosity constant       -- positive real number
%         L  = The end point of the interval in space -- positive real number -- [0, L]
%         T  = The end point of the interval in time  -- positive real number -- [0, T]
%         dx = The small change in x on the grid -- Delta x -- positive real number *small i.e. < L
%         dt = The small change in t on the grid -- Delta t -- positive real number *small i.e. < T
%         N  = The number of terms used in the Fourier series approximation -- large positive real number
%
% @author Jarren Ralf

t = 0.001:dt:T; % t-grid points
x = 0.001:dx:L; % x-grid points
x = x';         % transpose to make column vector

a0 = integral(@(x) f(x), 0, L)/L; % Calculate the a0 constant

% Initialize the coefficient, numerator, and denominator matrices
          a =   zeros(1, N);
  numerator =   zeros(length(x), length(t));
denominator = a0*ones(length(x), length(t));

for n = 1:N

    % Calculate the appropriate Fourier coefficient at step n
    a(n) = 2*integral(@(x) f(x).*cos(n*pi*x/L), 0, L)/L;

    % Update the grid points, each node in the mesh, of the numerator/denominator
    % matrix with the next function evaluation of the Fourier series at step n
      numerator =   numerator + n*a(n)*exp(-1*n^2*pi^2*c*t/L^2).*sin(n*pi*x/L);
    denominator = denominator +   a(n)*exp(-1*n^2*pi^2*c*t/L^2).*cos(n*pi*x/L);
end

% Construct the approximation of the analytical solution of Burgers' equation
u = 2*pi*c*numerator/L./denominator;
```

Listing A.2: Function that creates an animation of the solution of Burgers' Equation

```
function [u, x, t, denominator] = BurgersEq1D_Numerical(x0, c, L, T, dx, dt, N)
% This function solnBurgersEq1D, solves the one-dimentional viscous Burgers' equation using a finite
% Fourier series approximatation to the diffusion equation. This function outputs the solution to the heat
% equation as a matrix of points. In addition the user can also receive the x and t-values that the
% solution is evaluated at. Lastly, the user could be passed the solution to the diffusion equation,
% denoted here by 'denominator'.
%
%         x0 = The initial condition passed as vector -- positive real numbers
%         c  = The diffusion/viscosity constant       -- positive real number
%         L  = The end point of the interval in space -- positive real number -- [0, L]
%         T  = The end point of the interval in time  -- positive real number -- [0, T]
%         dx = The small change in x on the grid -- Delta x -- positive real number *small i.e. < L
%         dt = The small change in t on the grid -- Delta t -- positive real number *small i.e. < T
%         N  = The number of terms used in the Fourier series approximation -- large positive real number
%
% @author Jarren Ralf

t = 0:dt:T; % t-grid points
x = 0:dx:L; % x-grid points
x = x';         % transpose to make column vector

a0 = trapz(x, x0)/L; % Calculate the a0 constant

% Initialize the coefficient, numerator, and denominator matrices
          a =   zeros(1, N);
  numerator =   zeros(length(x), length(t));
denominator = a0*ones(length(x), length(t));

for n = 1:N

    % Calculate the appropriate Fourier coefficient at step n
    a(n) = 2*trapz(x, x0.*cos(n*pi*x/L))/L;

    % Update the grid points, each node in the mesh, of the numerator/denominator
    % matrix with the next function evaluation of the Fourier series at step n
      numerator =   numerator + n*a(n)*exp(-1*n^2*pi^2*c*t/L^2).*sin(n*pi*x/L);
    denominator = denominator +   a(n)*exp(-1*n^2*pi^2*c*t/L^2).*cos(n*pi*x/L);
end

% Construct the approximation of the analytical solution of Burgers' equation
u = 2*pi*c*numerator/L./denominator;
```

Listing A.3: Function that applies the Cole-Hopf transformation to an anonymous function and outputs an anonymous fucntion

```
function f = ColeHopfTransformation(initialCond, c)
% This function coleHopfTransformation, applies the Cole-Hopf transformation to a given anonymous function
% for a chosen value of the dissipation coefficient. This function receives the initial condition for
% Burgers' equation and converts it into the corresponding initial condition for the Heat equation. The
% given initial condition must be an integrable function i.e. finding a symbolic anti-derivative must be
% posible.
%
%         initialCond = The chosen initial condition for Burgers' equation given as an anonymous function
%                   c = The diffusion/viscosity constant (positive real number)
%
% @author Jarren Ralf

syms v(x) % Declare the symbolic variable of v as a function of x

% The Cole-Hopf tranformation and a condition to solve for the constant that arises after integration
ode = diff(v, x) == -1/2*initialCond(x)/c*v;
cond = v(0) == 1;

% Solve the ODE and output the solution as a function handle (anonymous function)
f = matlabFunction(dsolve(ode, cond));
```

Listing A.4: Function that applies the Cole-Hopf transformation to an anonymous function and outputs a vector of points

```
function x0 = ColeHopfTransformation_Numerical(u0, c, L, dx)
% This function coleHopfTransformation_Numerical, applies the Cole-Hopf transformation to a given
% anonymous function for a chosen value of the dissipation coefficient, interval length and space step.
% This function receives the initial condition for Burgers' equation and converts it into the
% corresponding initial condition for the Heat equation. This method uses ode45 which is a high order
% numerical ode solver.
%
%               u0 = The initial condition passed as an anonymous function
%               c  = The diffusion/viscosity constant       -- positive real number
%               L  = The end point of the interval in space -- positive real number -- [0, L]
%               dx = The small change in x on the grid -- Delta x -- positive real number *small i.e. < L
%
% @author Jarren Ralf

x = 0:dx:L; % x-grid points
y0 = 1;     % The iniital condition used to solve the ODE

% Rearrange the ODE in the form y' = f(z, y(z)) and set up the right hand side as an anonymous function
f = @(z, y) -(y*u0(z))/(2*c);

% Solve the ODE and output the solution as a column vector of of real numbers
[z, x0] = ode45(f, x, y0);
```

Listing A.5: Function that applies the discretized Cole-Hopf transformation to a solution of the heat equation represented by a matrix of points

```
function u = ColeHopfTransformation_Discrete(v, c, L, T, dx, dt)
% This function coleHopfTransformation_Discrete, applies the discrete Cole-Hopf transformation to the
% solution of the Diffusion equation, v. The output is the approximate solution to Burgers' equation.
%
%         v  = The solution to the Diffusion equation -- matrix of real numbers
%         c  = The diffusion/viscosity constant       -- positive real number
%         L  = The end point of the interval in space -- positive real number -- [0, L]
%         T  = The end point of the interval in time  -- positive real number -- [0, T]
%         dx = The small change in x on the grid -- Delta x -- positive real number *small i.e. < L
%         dt = The small change in t on the grid -- Delta t -- positive real number *small i.e. < T
%
% @author Jarren Ralf

M  = floor(T/dt);  % The number of subintervals in time
N  = floor(L/dx);  % The number of subintervals in space

u = zeros(N + 1, M + 1); % Initialize the solution matrix for Burgers' equation

% Apply the cole-hopf transformation via a forward difference approximation for the first derivative
u(0 + 1, :) = -2*c*((v(1 + 1, :) - v(0 + 1, :))./(dx*v(0 + 1, :)));

% Apply the cole-hopf transformation via a centered difference approximation for the first derivative
for i = 1:N - 1
    u(i + 1, :) = -1*c*((v((i + 1) + 1, :) - v((i + 1) - 1, :))./(dx*v(i + 1, :)));
end

% Apply the cole-hopf transformation via a backward difference approximation for the first derivative
u(N + 1, :) = -2*c*((v(N + 1, :) - v((N - 1) + 1, :))./(dx*v(N + 1, :)));
```

Listing A.6: Function that creates an animation of the solution of Burgers' Equation

```
function AnimatedPDESoln(u, u0, c, x, t, L, a, filename)
% This function animatedPDESoln, animates the solution to the given PDE. No objects get passed
% back to the user by this method, rather it simply creates the animation, closes it when it is finished
% running (i.e. max time is reached —— all of the vector t has been traversed), and finally saves a gif of
% the solution. This function needs to be tailored to the chosen PDE.
%
%                   u = The solution to the given PDE          —— matrix of real numbers
%                  x0 = The initial condition passed as vector —— positive real numbers
%                   c = The diffusion/viscosity constant       —— positive real number
%                   x = The x–values that the PDE is solved at —— vector of real numbers
%                   t = The t–values that the PDE is solved at —— vector of real numbers
%                   L = The end point of the interval in space —— [0, L] (positive real number)
%                   A = The amplitude of the sin initial condition (if chosen) —— real number
%            filename = The chosen filename passed to the function as a string
%
% @author Jarren Ralf

% Set the initial condition, instantiate a plot object, and then set the graph axis
  y = u0(x);
fig = figure;
  h = plot(x, y);
axis([0 L 0 a]);

% Set the X–Data and Y–Data source for the plot object to be x and y respectively
h.XDataSource = 'x';
h.YDataSource = 'y';

% Retreive the number of columns of the solution matrix
numColumns = size(u, 2);

% Initialize the cell array that will store frame structures corresponding to the figure
images = cell(1, numColumns);

% Animate and create a gif!
for i = 1:numColumns

    % Pass the updated function values to y, i.e. the solution changing in time
    y = u(:, i);

    % Updates the data source, i.e. the y–values from above and label the axis
    plot(x, y);
    axis([0 L 0 a]);
    refreshdata
    title({['The Solution of Burgers'' equation for \nu = ', num2str(c), ' (\itRe = ', num2str(1/c), ...
            ')']; ['t = ', num2str(t(i), '%5.3f')]})
    xlabel('x')
    ylabel('u(x, t)')

    % Draws the updated graphics on the figure immediately, then store the frames in the cell array
    drawnow
    frame = getframe(fig);
    images{i} = frame2im(frame);
end

% Close the figure
close;

% Create the gif and save as the chosen filename in the current directory
for i = 1:numColumns
    [A, map] = rgb2ind(images{i}, 256);
    if i == 1
        imwrite(A, map, filename, 'gif', 'LoopCount', Inf, 'DelayTime', 0.1);
    else
        imwrite(A, map, filename, 'gif', 'WriteMode', 'append', 'DelayTime', 0.1);
    end
end
```

Listing A.7: Function that solves the Heat Equation via the Crank-Nicolson finite difference approach

```
function [v, x, t, N, M] = HeatEq1D_CrankNicolson(x0, c, L, T, dx, dt)
% This function solveHeatEq1D_CrankNicolson, solves the one dimensional Heat equation using the finite
% difference technique via the Crank–Nicolson scheme. This function outputs the solution to the heat
% equation as a matrix of points. In addition the user can also receive the x and t–values that the
% solution is evaluated at. Lastly, the number of sub–intervals in the x and t dimensions can also be
% passed to the user.
%
%         x0 = The initial condition passed as vector —— positive real numbers
%         c  = The diffusion/viscosity constant       —— positive real number
%         L  = The end point of the interval in space —— positive real number —— [0, L]
%         T  = The end point of the interval in time  —— positive real number —— [0, T]
%         dx = The small change in x on the grid —— Delta x —— positive real number *small i.e. < L
%         dt = The small change in t on the grid —— Delta t —— positive real number *small i.e. < T
%
% @author Jarren Ralf

t = 0:dt:T;      % t–grid points
x = 0:dx:L;      % x–grid points
M = floor(T/dt); % The number of subintervals in time
N = floor(L/dx); % The number of subintervals in space
r = dt*c/dx^2;   % Defined constant used in the Crank–Nicolson scheme

% – MATLAB indices start at 1, NOT 0. Hence the code will have many '+1's for the purpose of re–indexing –
% Initialize the solution matrix
v = zeros(N + 1, M + 1);

% Build the tridiagonal matrix
mtx = full(gallery('tridiag', N + 1, –1/2*r, 1 + r, –1/2*r));
mtx(1    , 2) = –1*r;
mtx(N + 1, N) = –1*r;

% Use the initial condition to populate the first column of the solution matrix for t = 0
v(:, 0 + 1) = x0;

% Initialize the vector b
b = zeros(N + 1, 1);

% Apply the Crank–Nicolson Scheme
for j = 0:M – 1

    % The i = 0 case
    b(0 + 1) = r*v((0 + 1) + 1, j + 1) + (1 – r)*v(0 + 1, j + 1);

    % The i = 1, 2, ..., N – 1 case
    b(2:end–1) = r/2*v((2:end–1) + 1, j + 1) + (1 – r)*v(2:end–1, j + 1) + r/2*v((2:end–1) – 1, j + 1);

    % The i = N case
    b(N + 1) = (1 – r)*v(N + 1, j + 1) + r*v((N + 1) – 1, j + 1);

    % Solve the system for the next time step
    v(:, (j + 1) + 1) = mtx\b;
end
```

Listing A.8: Function that solves Burgers' Equation a finite elements approach

```
function [v, x, t, N, M] = HeatEq1D_FiniteElements(x0, c, L, T, dx, dt, theta)
% This function solveHeatEq1D_FiniteElements, solves the one dimensional Heat equation using a finite
% elements technique. This function outputs the solution to Burgers' equation as a matrix of points.
% In addition the user can also receive the x and t-values that the solution is evaluated at. Lastly,
% the number of sub-intervals in the x and t dimensions can also be passed to the user.
%
%         x0 = The initial condition passed as vector --- positive real numbers
%         c  = The diffusion/viscosity constant       --- positive real number
%         L  = The end point of the interval in space --- positive real number --- [0, L]
%         T  = The end point of the interval in time  --- positive real number --- [0, T]
%         dx = The small change in x on the grid --- Delta x --- positive real number *small i.e. < L
%         dt = The small change in t on the grid --- Delta t --- positive real number *small i.e. < T
%      theta = The choice of weight factor --- positive real number in [0, 1]
%
% @author Jarren Ralf

if theta < 0.5
    if (c*dt)/dx^2 > (1 - 2*theta)/6
        disp('The finite element equation is unstable!');
    end
end

x = 0:dx:L;      % x-grid points
t = 0:dt:T;      % x-grid points
N = floor(L/dx); % The number of subintervals in space
M = floor(T/dt); % The number of subintervals in time

% Construct the matrices C and K (see my report for derivation of these quantities)
cii =  2*dx/3;
cij =    dx/6;
kii =  2/dx;
kij = -1/dx;
  C = full(gallery('tridiag', N + 1, cij, cii, cij));
  K = full(gallery('tridiag', N + 1, kij, kii, kij));

% Construct the matrix D and E matrix
D = C + c*     theta *dt*K;
E = C - c*(1 - theta)*dt*K;

% Initialize the solution matrix
v = zeros(N + 1, M + 1);

% Use the initial condition to populate the first column of the solution matrix for t = 0
v(:, 1) = x0;

% Solve the system for each time step
for j = 1:M
    v(:, j + 1) = D\(E*v(:, j));
end
```

Listing A.9: Function that solves the Heat Equation via the finite elements approach using the Crout method to solve the system of linear equations

```
function [v, x, t, N, M] = HeatEq1D_FiniteElements_Crout(x0, c, L, T, dx, dt, theta)
% This function solveHeatEq1D_FiniteElements_Crout, solves the one dimensional Heat equation using
% a finite elements technique which uses the Crout method for solving the system of linear equations.
% This function outputs the solution to the heat equation as a matrix of points. In addition the user
% can also receive the x and t-values that the solution is evaluated at. Lastly, the number of
% sub-intervals in the x and t dimensions can also be passed to the user.
%
%         x0 = The initial condition passed as vector --- positive real numbers
%         c  = The diffusion/viscosity constant       --- positive real number
%         L  = The end point of the interval in space --- positive real number --- [0, L]
%         T  = The end point of the interval in time  --- positive real number --- [0, T]
%         dx = The small change in x on the grid --- Delta x --- positive real number *small i.e. < L
%         dt = The small change in t on the grid --- Delta t --- positive real number *small i.e. < T
%      theta = The choice of weight factor --- positive real number in [0, 1]
%
% @author Jarren Ralf

if theta < 0.5
    if (c*dt)/dx^2 > (1 - 2*theta)/6
        disp('The finite element equation is unstable!');
    end
end

x = 0:dx:L;      % x-grid points
t = 0:dt:T;      % x-grid points
N = floor(L/dx); % The number of subintervals in space
M = floor(T/dt); % The number of subintervals in time

% Construct the matrices C and K (see my report for derivation of these quantities)
cii =  2*dx/3;
cij =    dx/6;
kii =  2/dx;
kij = -1/dx;
  C = full(gallery('tridiag', N + 1, cij, cii, cij));
  K = full(gallery('tridiag', N + 1, kij, kii, kij));

% Construct the matrix D and E matrix
D = C + c*     theta *dt*K;
E = C - c*(1 - theta)*dt*K;

[L, U] = LUdecompCrout(D); % Build the Lower and Upper tridiagonal matrices

v = zeros(N + 1, M + 1); % Initialize the solution matrix
y = zeros(N + 1,     1); % Initialize the y vector

v(:, 1) = x0; % Use the initial condition to populate the first column of the solution matrix for t = 0

% Loop through each time step
for j = 1:M
      b  = E*v(:, j);    % Initialize the b matrix
    y(1) = b(1)/L(1, 1); % Set the first y-value i.e. for n = 1

    % Loop through the rest of the space steps to build the y vector
    for i = 2:N + 1
        y(i) = (b(i) - L(i, i - 1)*y(i - 1))/L(i, i);
    end

    v(N + 1, j + 1) = y(N + 1); % Set the last x-value i.e. for n = N + 1

    % Loop backwards through the rest of the space steps to build the x vector
    for i = N:-1:1
        v(i, j + 1) = y(i) - U(i, i + 1)*v(i + 1, j + 1);
    end
end
```

Listing A.10: Breaks down a matrix A into it's LU decomposion in preparatuion for the crout method of solving systems of linear equations

```
function [L, U] = LUdecompCrout(A)
% This function LUdecompCrout, decomposes a given matrix, A, into a lower triangular matrix, L, and an
% upper triangular matrix, U. This function was modified and vectorized by Jarren Ralf.
%
%            A = Any N x M matrix with real number entries
%
% @author https://en.wikipedia.org/wiki/Crout_matrix_decomposition

[R, C] = size(A); % Retrieve the number of rows and columns of the given matrix
L = zeros(R, C);  % Initialize the matrix L with the same dimensions as matrix A
U = eye(R, C);    % Initialize the matrix U with the identity matrix

L(:, 1) = A(:, 1); % Extract the first column of A and set it as the first column of L
U(1, 2:end) = A(1, 2:end)/L(1, 1); % Set the fist row of the U matrix starting from the second column

for i = 2:R

    % Construct the lower triangular matrix, L
    for j = 2:i
        L(i, j) = A(i, j) - L(i, 1:j - 1)*U(1:j - 1, j);
    end

    % Construct the upper triangular matrix, U
    for j = i + 1:R
        U(i, j) = (A(i, j) - L(i, 1:i - 1)*U(1:i - 1, j))/L(i, i);
    end
end
```

Listing A.11: Examples of the implementation of each of the functions above

```
%%%%%%%%%%%%%%%%%%%%%%
%% Analytical Solution
%%%%%%%%%%%%%%%%%%%%%%
clear; clc; close all;

a  = 1;    % Amplitude of the sin initial condition
L  = 1;    % Length of the x-interval -- [0, L]
T  = 1;    % Length of the t-intercal -- [0, T]
c  = 1;    % Diffusion/Viscosity Constant
dt = 0.02; % This is delta t -- The size of the sub-interval in time
dx = 0.01; % This is delta x -- The size of the sub-interval in space
numTerms  = 100; % Number of terms in the finite Fourier series

% Toggle the chosen initial condition
initialCond = @(x) a*sin(pi*x/L);
%initialCond = @(x) 4*x.*(1 - x);

% Apply the Cole-Hopf Transformation to the initial condition
f = ColeHopfTransformation(initialCond, c);

% Solve the equation
[u, x, t] = BurgersEq1D(f, c, L, T, dx, dt, numTerms);

% Plot the surface
surf(t, x', u);
title(['The Solution of Burgers'' equation for \nu = ', num2str(c), ' (\itRe = ', num2str(1/c), ')'])
xlabel('t')
ylabel('x')
zlabel('u(x, t)')



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Analytical Solution - Using Numerical Integration while performing the Cole-Hopf transform
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear; clc; close all;

a  = 1;    % Amplitude of the sin initial condition
L  = 1;    % Length of the x-interval -- [0, L]
T  = 1;    % Length of the t-intercal -- [0, T]
c  = 1;    % Diffusion/Viscosity Constant
dt = 0.02; % This is delta t -- The size of the sub-interval in time
dx = 0.01; % This is delta x -- The size of the sub-interval in space

% Number of terms in the finite Fourier series -- Errors occur when too many terms are used
numTerms  = 100;

% Toggle the chosen initial condition (feel free to define your own!)
u0 = @(z) a*sin(pi*z/L);
%u0 = @(z) 4*z.*(1 - z);

% Apply the Cole-Hopf Transformation to the initial condition
x0 = ColeHopfTransformation_Numerical(u0, c, L, dx);

% Solve the equation
[u, x, t] = BurgersEq1D_Numerical(x0, c, L, T, dx, dt, numTerms);

% Plot the surface
surf(t, x', u);
title(['The Solution of Burgers'' equation for \nu = ', num2str(c), ' (\itRe = ', num2str(1/c), ')'])
xlabel('t')
ylabel('x')
zlabel('u(x, t)')
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Animation of the Analytical Solution
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear; clc; close all;

a  = 1;     % Amplitude of the sin initial condition
L  = 1;     % Length of the x-interval -- [0, L]
T  = 1.5;   % Length of the t-intercal -- [0, T]
c  = 1;     % Diffusion/Viscosity Constant
dt = 0.002; % This is delta t -- The size of the sub-interval in time
dx = 0.001; % This is delta x -- The size of the sub-interval in space
numTerms  = 100; % Number of terms in the finite Fourier series

% Toggle the chosen initial condition (feel free to define your own!)
u0 = @(z) a*sin(pi*z/L);
%u0 = @(z) 4*z.*(1 - z);

% Apply the Cole-Hopf Transformation to the initial condition
x0 = ColeHopfTransformation_Numerical(u0, c, L, dx);

% Solve the equation
[u, x, t] = BurgersEq1D_Numerical(x0, c, L, T, dx, dt, numTerms);

% Created an animated solution and save it as .gif
AnimatedPDESoln(u, u0, c, x, t, L, a, 'BurgersEquation.gif');




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Finite Elements Scheme -- Crout method
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear; clc; close all;

a  = 1;      % Amplitude of the sin initial condition
L  = 1;      % Length of the x-interval -- [0, L]
T  = 1;      % Length of the t-intercal -- [0, T]
c  = 1;      % Diffusion/Viscosity Constant
dt = 0.0016; % This is delta t -- The size of the sub-interval in time
dx = 0.001;  % This is delta x -- The size of the sub-interval in space

% Chosoe the method     0 -> Explicit;     .5 -> Crank-Nicolson;     1 -> Implicit
theta = 0;

% Toggle the chosen initial condition (feel free to define your own!)
u0 = @(z) a*sin(pi*z/L);
%u0 = @(z) 4*z.*(1 - z);

% Apply the Cole-Hopf Transformation to the initial condition
x0 = ColeHopfTransformation_Numerical(u0, c, L, dx);

% Solve the Heat equation via a finite elements scheme using the Crout method
[v, x, t] = HeatEq1D_FiniteElements_Crout(x0, c, L, T, dx, dt, theta);

% Compute the solution to Burgers' equation by applying the Cole-Hopf transformation
u = ColeHopfTransformation_Discrete(v, c, L, T, dx, dt);

% Plot the surface
surf(t, x', u);
title(['The Solution of Burgers'' equation for \nu = ', num2str(c), ' (\itRe = ', num2str(1/c), ')'])
xlabel('t')
ylabel('x')
zlabel('u(x, t)')
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Finite Elements Scheme
%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear; clc; close all;

a  = 1;      % Amplitude of the sin initial condition
L  = 1;      % Length of the x-interval -- [0, L]
T  = 1;      % Length of the t-intercal -- [0, T]
c  = 1;      % Diffusion/Viscosity Constant
dt = 0.0016; % This is delta t -- The size of the sub-interval in time
dx = 0.01;   % This is delta x -- The size of the sub-interval in space

% Chosoe the method     0 -> Explicit;     .5 -> Crank-Nicolson;     1 -> Implicit
theta = 1;

% Toggle the chosen initial condition (feel free to define your own!)
u0 = @(z) a*sin(pi*z/L);
%u0 = @(z) 4*z.*(1 - z);

% Apply the Cole-Hopf Transformation to the initial condition
x0 = ColeHopfTransformation_Numerical(u0, c, L, dx);

% Solve the Heat equation via a finite elements scheme
[v, x, t] = HeatEq1D_FiniteElements(x0, c, L, T, dx, dt, theta);

% Compute the solution to Burgers' equation by applying the Cole-Hopf transformation
u = ColeHopfTransformation_Discrete(v, c, L, T, dx, dt);

% Plot the surface
surf(t, x', u);
title(['The Solution of Burgers'' equation for \nu = ', num2str(c), ' (\itRe = ', num2str(1/c), ')'])
xlabel('t')
ylabel('x')
zlabel('u(x, t)')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Crank-Nicolson Finite Difference Scheme
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear; clc; close all;

a  = 1;    % Amplitude of the sin initial condition
L  = 1;    % Length of the x-interval -- [0, L]
T  = 2;    % Length of the t-intercal -- [0, T]
c  = 1/5;  % Diffusion/Viscosity Constant
dt = 0.05; % This is delta t -- The size of the sub-interval in time
dx = 0.02; % This is delta x -- The size of the sub-interval in space

% Toggle the chosen initial condition (feel free to define your own!)
u0 = @(z) a*sin(pi*z/L);
%u0 = @(z) 4*z.*(1 - z);

% Apply the Cole-Hopf Transformation to the initial condition
x0 = ColeHopfTransformation_Numerical(u0, c, L, dx);

% Solve the heat equation using finite differences via the Crank-Nicolson scheme
[v, x, t] = HeatEq1D_CrankNicolson(x0, c, L, T, dx, dt);

% Compute the solution to Burgers' equation via the discrete Cole-Hopf transformation
u = ColeHopfTransformation_Discrete(v, c, L, T, dx, dt);

% Plot the surface
surf(t, x', u);
title(['The Solution of Burgers'' equation for \nu = ', num2str(c), ' (\itRe = ', num2str(1/c), ')'])
xlabel('t')
ylabel('x')
zlabel('u(x, t)')
```

Appendix B

Various Forms of Burgers' Equations

This appendix contains several different versions of Burger's equation, inspired by Appendix D in Coleman's PDE textbook [10].

First, we present the most generalized form of Burger's equation,

Generalized Burger's equation: $u_t + f(u)u_x = \kappa u_{xx}$.

If $f(u) = u$, we have

Burger's equation (with dissipation): $u_t + uu_x = \kappa u_{xx}$.

If we add a space-time noise term $\eta(x,t)$, we end up with [28]

Stochastic Burger's equation: $u_t + uu_x = \kappa u_{xx} - \lambda\eta_x$.

If we set the dissipation coefficient, ϵ, equal to zero, we get

Burger's equation: $u_t + uu_x = 0$.

A rather nasty version of the equation comes from including the Korteweg-de Vries equation [27] with it, as well as a forcing term,

KdV-Burgers equation: $u_t + puu_x + qu^2u_x + ru_{xx} - su_{xxx} = f(t)$.