A Comparison Study of Chebyshev Spectral Collocation Based Methods for Solving Nonlinear Second Order Evolution Equations

By

Khayelihle A. Kheswa

Supervisor: Professor Sandile S. Motsa

UNIVERSITY OF KWAZULU - NATAL ™

INYUVESI YAKWAZULU-NATALI

School of Mathematics, Statistics and Computer Science

University of KwaZulu-Natal

Pietermaritzburg, South Africa

A dissertation submitted in the fulfillment of the requirements for

Masters in Science at the
School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal, Pietermaritzburg Campus

July 2015

Abstract

In this study Spectral Quasilinearisation Method (SQLM) coupled with finite difference and Bivariate Spectral Quasilinearisation Method (BSQLM) in solving second order nonlinear evolution partial differential equations are compared. Both methods use Newton-Raphson quasilinearisation method (QLM) and Chebyshev spectral collocation based on Lagrange interpolation to solve the governing equations. The Spectral Quasilinearisation Method coupled with finite difference is obtained by applying the spectral collocation method on space derivatives and finite difference of time derivatives while the BSQLM is a Bivariate Lagrange interpolation based scheme in which the spectral collocation method is applied independently to both time and space derivatives. The applicability of these methods is shown by solving a class of second order nonlinear evolution partial differential equations (NPDEs), namely Burgers equation, Burgers-Fisher, Fisher's equation, Newell-Whitehead-Segel equation and Zeldovich equation that arise in some fields of science and engineering. The numerical approximation results are validated for accuracy by comparing them with exact solutions. Tables for Explicit, Implicit and Crank-Nicolson SQLM and BSQLM with their computational times were generated for comparison; the order of accuracy for each method and error graphs are presented.

Preface

The work described in this research project was carried out under the supervision and direction of Professor Sandile Motsa, School of Mathematics, Statistics and Computer Sciences, University of KwaZulu-Natal, Pietermaritzburg, from August 2013 to July 2015.

It represents the original work of the author and has not been otherwise submitted in any form for any degree or diploma to any University. Where use has been made of the work of others it is duly acknowledged in the text.

Author: Mr Khayelihle A. Kheswa	Date
Supervisor: Prof Sandile S. Motsa	Date

Dedication

To Mr. MW Kheswa and Mrs. DN Kheswa

Acknowledgements

Firstly I would like to thank God, the Almighty for giving me strength and life to enable me to finish this work. I would like to give my special thanks Professor Sandile S. Motsa as my advisor, and a mentor. Without his support, guidance and endless patience, my dissertation would have been impossible. Also I wish to thank him for financial support he gave me to complete this study, Ndibulela andiqedi. I also thank to Professor Precious Sibanda my mentor during the internship I got from National Research Foundation, for his support and encouragement.

I would also like thank the faculty and my fellow students at the Department of Mathematical, Statistical and Computer Sciences for providing great working conditions. My special thanks to Mpendulo Magagula, Joseph Malinzi, Hloniphile Sithole and Oluwamuyiwa Otegbeye for their support. My special thanks to my family, in particular my parents and Vuyo. Lastly I would like to thank my friends Zimkitha Dlamini, Sphiwo Mtshali and Njabulo Makhathini for the support and courage they gave me.

Acknowledgement: *DST-NRF Centre of Excellence in Mathematical and Statistical Sciences (CoE-Mass). Option expressed and conclusions arrived at are those of the author and not necessarily to be attributed to the CoE-Mass.*

Table of Contents

List of Figures

List of Tables

April 22, 2016

Chapter 1

Introduction

This thesis discusses the comparison of Spectral Quasilinearisation Method coupled with finite difference and Bivariate Spectral Quasilinearisation Method in solving second order nonlinear evolution partial differential equations. Both methods are spectral based and different studies show that there is an increase in the use of spectral methods [6]. The methods to be used in this study have been used in solving boundary layer problems and evolution equations by Motsa et al. in [55, 57]. However, there has been no study which compared these methods in solving second order nonlinear evolution partial differential equations. This work discusses the question why it is important to solve nonlinear evolution partial differential equations, the importance of spectral methods, testing the methods and reaching a conclusion on which method is better than the other.

1.1 Background of the Problem

Partial differential equations as a field of research has been given attention by different authors for example [24, 28, 70]. However, research on finding solutions of nonlinear evolution partial differential equations using numerical methods continues. A nonlinear evolution partial differential equation is a system depending on continuous time variable t and the space variable y described by an equation of the form

$$\frac{\partial u}{\partial t} = F\left(u, \frac{\partial u}{\partial y}, \frac{\partial^2 u}{\partial y^2}\right), \tag{1.1}$$

where $u(y,t)$ is a function of space y and time t, $\dfrac{\partial u}{\partial y}$ and $\dfrac{\partial^2 u}{\partial y^2}$ are the first and second partial derivatives of the function $u(y,t)$ with respect to y. The function F can be expressed as the summation of linear and nonlinear functions L and N respectively, that is

$$\frac{\partial u}{\partial t} = L\left(u, \frac{\partial u}{\partial y}, \frac{\partial^2 u}{\partial y^2}\right) + N\left(u, \frac{\partial u}{\partial y}, \frac{\partial^2 u}{\partial y^2}\right),$$

and is discussed further in Chapter 2. For notational simplicity, $u(y,t)$ is written as u. It is still crucial to conduct more research on finding the solution of nonlinear evolution partial differential equations with the aim of improving accuracy. Agreeing with Chang Shu [24] in most of the science and engineering fields, partial differential equations (PDEs) might be encountered, singly or as a system. Examples are Burgers equation, Fisher's equation and Burgers-Fisher equation. This study focuses on second order nonlinear evolution partial differential equations which can effectively model the interaction between diffusion transport, reaction mechanisms and convection properties. Reaction-diffusion equations possess interesting properties that make them unique to study from both a numerical and an analytic point of view. The physical applications of reaction-diffusion equations are very broad; such equations can describe many of the dynamics found in nature, from chemical reactions, biological processes, ecological patterns, to geological events [70]. Thus, understanding the limitations of numerical solutions to these types of equations is of great importance to many scientists [24,70]. Generally, most of these problems may involve nonlinear evolution partial differential equations where exact solution is unattainable or difficult to get. This makes numerical scientists see the importance of developing alternative ways to estimate the solutions of the nonlinear evolution partial differential equations. After years of research, scientists, therefore, approximate the solution of the system of partial differential equations by using numerical discretisation techniques on some function values at certain distinct points, which are called grid points or mesh points. The most widely used numerical methods in engineering and in computational fluid dynamics are the finite difference, finite element and finite volume methods.

1.2 Review of Nonlinear Partial Differential Equations

Nonlinear partial differential equations are found in many different fields of science particularly in engineering, physics, chemistry and biology [35]. Other examples are the filtration of fluids, diffusion in the chemical reaction, population dynamics and the famous Black and Scholes equation in finance which are all modeled using evolution partial differential equations [72]. Thus, evolution partial differential equations are crucial in our societal life. Many nonlinear models of real-life problems are solved either numerically or analytically. As a result, many researchers have devoted their lives to investigating the solutions of evolution partial differential equations using different methods. It is noted in the literature that the time-dependent partial differential equations showed much development in application during 1945, immediately after the Second World War, when large-scale practical application became possible with the help of computers. During this evolution, many researchers played their role, which includes Von Neuman in 1951 [5], Crank and Nicolson in 1947 [54]. Finite difference method (FDM), finite element method (FEM) and finite volume method (FVM) are the methods that were used in the past. The FDM lost its attractiveness because of hitches in stability and inaccuracy. In the case of FVM and FEM mesh generation in higher dimensions became more cumbersome [26]. To overcome such problems the spectral method has taken over and become the most popular method in the last two decades because of the accuracy and efficiency that it brings in the computation. Recently, researchers have been using spectral methods which are more accurate and efficient compared to traditional methods [29].

1.3 Review of Spectral Methods

This section gives a brief introduction to spectral methods. Spectral methods and finite elements are closely connected and constructed from similar ideas; the main differentiating factor is that spectral methods use basis or test functions that are nonzero over the entire domain, while finite element methods use basis functions that are not zero only on small sub-domains. Hence, spectral methods take on a global approach whereas finite element methods (FEM) and finite difference methods (FDM) take a local approach. The computations at a given point does not depend only on information at the neighbouring points but also on information on the complete domain. For this reason, spectral methods have an exceptional error property which is known as exponential convergence, being the fastest possible, when the solution is smooth [6]. If the problem have disjoints, finite element methods (FEM) and finite volume methods (FVM) will be preferred than spectral methods since these two can handle the problems with disjoints much easily. The focal idea of spectral methods is to approximate the solution of the problem as a weighted sum of certain elementary functions and then choose the coefficients in the sum in order to minimise the difference between the exact solution and the estimated one as far as possible. The way in which test functions are selected leads to three well-known categories of spectral methods called Galerkin Method, Tau Method and the Collocation Method. Spectral collocation methods, also known as pseudo-spectral methods, are a subclass of spectral methods and are similar to Finite Difference methods due to direct use of a set of grid points, which are known as collocation points [25]. The differential equation need to be satisfied exactly at the collocation points. The Tau and Galerkin spectral methods are similar to each other due to the fact that the expanding basis is not obliged to satisfy boundary conditions, requiring extra equations to be applied in the boundary conditions [4]. Among the three methods, the spectral collocation method is measured to be the simplest with extraordinary precision and stability [3]. Doha et al. [28] also highlighted that the collocation method deals with nonlinear terms more easily than Galerkin and Tau Methods. During the last thirty years, the spectral collocation method has been considered a good candidate for solving nonlinear physical modeling problems and fractional differential equations because of its simplicity and accuracy as compared to finite difference methods [3]. The rate of convergence of spectral approximations depends only on the smoothness of the solution, yielding the ability to achieve high precision with a small amount of data. This fact is known in the literature as "spectral accuracy" [25]. The spectral collocation method is chosen considering the fact that it gives an exponential convergence rate, which is very useful in providing highly accurate solutions to nonlinear differential equations even if a small number of grid points are used. The spectral methods also work well in solving both linear and nonlinear equations. Generally the spectral methods are computationally less challenging compared to traditional methods but become inaccurate for problems with disjointed coefficients [6]. This increase in inaccuracy is the result of the Gibbs phenomenon. It is caused by oscillations arising from the discontinuity caused by the fact that a discontinuous solution is being approximated by an oscillatory set of smooth functions [6]. The Pseudospectral Collocation Method has been used together with several methods to solve evolution problems eg. with the Runge-Kutta method [34]. The Spectral Quasilinearisation Method coupled with finite difference in time was used for the first time in solving unsteady boundary layer flow problems by Motsa et al. [58]. This was the first paper in which both methods were applied in partial differential equations. Before that, these methods were applied to ordinary differential equations. Another method that has been employed which is based on spectral methods is the spectral relaxation method. The procedure in this method uses the idea of the Gauss-Seidel method to decouple the governing systems. From there, the arising equations form an iterative scheme which is developed by estimating linear terms in the current iteration level and nonlinear terms in the previous iteration level. Kameswaran et al. [60] used the Spectral Relaxation Method to solve boundary value problems that arise in fluid mechanics applications. Another method based on spectral methods is the Spectral Local Linearisation approach used by Motsa et al. [59] for natural convection boundary layer flow. This really shows how the spectral methods have gained the interest of different researchers. This study will use the pseudo-spectral methods known as Spectral Quasilinearisation Method coupled with finite difference in time (SQLM) and Bivariate Spectral Quasilinearisation Method (BSQLM). These methods have been used by Dlamini et al. [27] and Motsa et al. [53, 57], in solving Similarity Boundary Layer Problems and evolution problems respectively. It is important to note that both SQLM and BSQLM fall under collocation methods. There are aspects that need to be considered for efficient implementation of spectral methods. They are discussed in detail as:

- Evaluation of derivatives: The derivatives are approximated using differentiation matrices as fully discussed in [71]. The methods use different approaches to approximate or evaluate the derivatives. In the SQLM, time derivatives are evaluated using finite difference method (FDM) and space derivatives using the spectral method. In the BSQLM, both time and space derivatives are evaluated using the spectral method.

- Evaluation of nonlinear and non-constant coefficient terms: The most efficient way to evaluate nonlinear and general non-constant terms in spectral approximations is to apply transformation methods. For this study, nonlinear terms are linearised using the QLM which is discussed in Section 3.3. In general, it is easier to implement the collocation method than the other types of spectral methods and it deals quite well with nonlinear equations.

- Modeling error: These errors arise due to the difference between the real problem and the mathematical model. Modeling errors can be brought about both by time and spacing. Time discretisation errors in spectral methods are usually smaller than space discretisation errors. According to David in [25], there are two main reasons for this, which are: (i) change in time is commonly restricted in size by explicit stability conditions and stability of the time integration require time-differencing errors to be insignificant,(ii) many problems involve several space coordinates so any possible efficiency in the representation of the space disparity of the dependent variables is significant to the overall effectiveness of the method.

1.4 Aims and Research Objectives

This study will use Spectral Quasilinearisation Method (SQLM) and Bivariate Spectral Quasilinearisation Method (BSQLM) to compute the numerical solution of nonlinear evolution equations. The aim of the study is to investigate the applicability of these methods in solving second order nonlinear evolution equations, to examine the best method between SQLM and BSQLM by comparing the accuracy and computational speed of each method. Approximate solutions obtained are compared with the exact solutions which are available in the literature. The study will explore different forms of SQLM which use Explicit Method, Implicit Method and Crank-Nicolson finite difference schemes for time derivatives and compare them with results for BSQLM that uses spectral derivatives for all derivatives in time and space.

1.5 Problem Statement

In this study, the Spectral Quasilinearisation Method coupled with finite difference in time and Bivariate Spectral Quasilinearisation Method are employed in solving Burgers equation, Burgers-Fisher equation, Fisher's equation, Newell-Whitehead-Segel and Zeldovich equations. The results found using the Spectral Quasilinearisation Method coupled with finite difference in time and Bivariate Spectral Quasilinearisation Method are compared with the exact solutions to measure their accuracy and convergence. Finally, the governing equations will also be solved using the Matlab computing software. There are many ways to approximate solutions of nonlinear evolution partial differential equations and the development of both numerical and analytical methods for solving these equations continues to be an area of interest to many scientists, whose research aim is to improve the understanding of nonlinear problems. Different methods for obtaining analytical and approximate solutions to nonlinear evolution equations have been proposed and used successfully. However, some methods have limitations and drawbacks in approximating numerical solutions. These include slow convergence and poor accuracy, particularly for large time ($t > 1$) [57]. In this study, the effect of large time is examined using both methods. Spectral methods have been used effectively in numerous fields of sciences and engineering because of their ability to give accurate solutions of differential equations. Khater et al. [35] applied the Chebyshev spectral collocation method in space and finite differences to approximate the time derivative to solve Burger type equations. The SQLM uses a similar idea, that using explicit, implicit and Crank-Nicolson schemes on time derivatives will give an understanding of the effect that is brought about by the use of finite differences in time derivatives and spectral derivatives on space derivatives. There are problems that have been discovered in using finite differences, for example it requires many grid points to achieve good accuracy and, therefore, requires a lot of computer memory and computational time. To address this challenge, the BSQLM is used. It uses the spectral method on both space and time derivatives.

1.6 Significance of the Study

This study is undertaken to compare the effect of using the finite difference derivatives in time together with spectral derivatives in space and spectral derivatives on time and space derivatives. The proposed study will add value in the following ways:

1. Provide knowledge on how to use SQLM and BSQLM as a mathematical tool to solve nonlinear evolution equations.

2. Also act as the base for further research on numerical methods for Partial Differential Equations in numerical methods.

1.7 Plan of the Dissertation

Chapter 2 begins by giving the general form of the evolution problem to be investigated with defined initial and boundary conditions. Methods that have been used to solve nonlinear evolution partial differential equations in the past are considered, then a general introduction to spectral methods and the reasons which led to the use of spectral over other methods is given. Chapter 3 describes the SQLM by starting with linearisation of the problem using QLM and shows how to apply the spectral method to the linearised equation. Chapter 4, then defines the BSQLM and shows how this method works. In Chapter 5 the test problems are solved and results and discussion are presented. Concluding remarks follow in Chapter 6.

Chapter 2

Nonlinear Evolution Equations and Properties of Numerical Methods

Nonlinear evolution partial differential equations form a foundation of many models in mathematics, physical science, chemical and biological phenomena and recently their applications have extended to financial forecasting and economics. Since these equations cannot in general be solved analytically, it is, therefore, important to approximate their solution numerically. This study will use Explicit, Implicit and Crank-Nicolson Spectral Quasilinearisation Method and Bivariate Spectral Quasilinearisation Method to estimate solutions of nonlinear evolution partial differential equations. The next section will consider the general model which the study attempts to solve.

2.1 Mathematical Model

Consider the second order nonlinear parabolic evolution equation of this form:

$$\frac{\partial u}{\partial t} = L\left(u, \frac{\partial u}{\partial y}, \frac{\partial^2 u}{\partial y^2}\right) + N\left(u, \frac{\partial u}{\partial y}, \frac{\partial^2 u}{\partial y^2}\right), \quad 0 \leq y \leq l, t \in [0, T] \tag{2.1}$$

with the initial and boundary conditions

$$u(y,0) = u_0(y), \quad 0 \leq y \leq l, t = 0, \tag{2.2}$$

$$u(0,t) = g_0(t), \quad u(l,t) = g_1(t), t \in [0,T] \tag{2.3}$$

where $u(y,t)$ is the solution to be approximated and y and t are space and time variables respectively, and L and N are linear and nonlinear operators respectively. It is very important to note that $u_0(y)$ in (2.2) is the initial condition and (2.3) shows the left and right boundary conditions when $y = 0$ and $y = l$ respectively. In this study, the model represented by equation (2.1-2.3) is solved.

Among the evolution equations that are considered in this study is the Burgers equation. The Burgers equation is a quasi-linear parabolic partial differential equation that defines the time evolution of the function $u(y,t)$ under nonlinear convection and linear dissipation that takes the following form:

$$u_t = \varepsilon u_{yy} - uu_y + f(y,t), \quad 0 \leq y \leq l, t \geq 0, \tag{2.4}$$

where $\varepsilon > 0$ is the quantity that represents the kinematic viscosity in the equation. When the viscosity is equal to zero, the evolution of the function $u(y,t)$ may develop tremors and if viscosity is small, sharp gradients can develop and disperse as $t \to \infty$ [67]. The Burgers equation is an essential partial differential equation from fluid mechanics. It arises in several areas of everyday life sciences such as gas dynamics and traffic flow [43]. The Burgers equation seems to have applications even in economics according to Schumpeter in [30]. In the period 1911 to 1939, economic development of industry was a periodical process with a period of order half-century ("business cycle"). It consisted of cascades of creation, processes of formation and cascades of destruction. Creative and destructive cascades can be described by Lotka-Volterra type equations. The mechanism of technological changes in the industry during processes of formation can be divided into two components: creation of new technologies by a firm (innovation process) and adoption of technologies, created by other firms (imitation process). For an industry with many firms, its development can be described as an evolution of its efficiency distribution [30]. The Burger type equation first appeared in 1915 in a paper written by Batman according to Nguyen [67]. Yet, the equation gets its name from far-ranging research by Burgers at the beginning of 1939 [64]. Burgers equation appears frequently as the description of a more complex and sophisticated models. Hence, it is usually thought of as a "toy model", namely, a tool that is used to understand some of the inside behavior of the general problem. The results confirm that it is correspondent to the Navier-Stokes equation for incompressible flow with the pressure term uninvolved [47]. Burgers equation has been used as a modest model for many physically exciting problems for convection-diffusion phenomena such as shock waves, turbulence, decaying free turbulence, traffic flows, flow-related problems, gas dynamics, number theory, forest fire, population growth models etc. [35]. Khater et al. [35] used the method which is called Chebyshev spectral collocation (ChSC) method to solve Burgers equation. The ChSC method is obtained through starting with Chebyshev approximation for the approximate solution and creating approximations for the higher-order derivatives through successive differentiation of the approximate solution. Reducing the equation to a system of ordinary differential equations (ODEs) that are solved by the Runge–Kutta method of fourth order. Biazar et al. [22] used the Variation Iteration Method (VIM) to find the solution to Burgers equation where they compared their results to the ones obtained using Adomian's Decomposition Method (ADM). The Burgers equation has also been solved by Nguyen [67] using the Finite Element Method. The equation is first transformed using the Hopf-Cole transformation [67]. A modified Adomian's method was used by Darvishi et al. [1]. Their method was based on decomposing the equation using the Hofp-Cole Transformation. Tamer et al. [64] highlighted that evolution equations have not been solved exactly, but with many combinations of initial and boundary conditions, using specified conditions an approximate solution close to exact solution can be found certainly.

The Fisher's equation is also considered as another example of the nonlinear evolution equation. The Fisher's equation was first introduced by R.A. Fisher as a model of the wave propagation of a favoured gene in a population in 1937 [32]. According to Zarebnia and Jalili [70], the Fisher's equation plays a crucial role in neutron flux in a nuclear reactor. It has a wide application in ecology and plasma physics. It is also used to describe the interaction of diffusion and reaction processes in biology, chemistry and in engineering [19]. Recently Mittal and Jain [49] used the Modified Cubic Spline Collocation method to solve Fisher's equation. This method is applied without any transformation and linearisation process. Jalili et al. [70] used the spectral collocation method to solve Fisher's equation. Their method combines the Crank-Nicolson scheme operating on the diffusive terms and a second-order Adams-Bashforth scheme acting on the advective terms [70]. Fisher's equation is also encountered in various applications such as tissue engineering, autocatalytic and other chemical reactions, combustion, and neurophysiology [9]. The Fisher's equation has the form

$$u_t = vu_{yy} + \rho f(u) \tag{2.5}$$

[19]. There are two cases that are most popular, firstly $f(u) = u(1-u)$ and secondly $f(u) = \rho u(1-u)$, where $\rho > 0$. The first case is commonly used to describe the kinetic advancing rate of an advantageous gene and the second case arises in large number of biological and chemical phenomena [48]. The coefficient v and ρ are the diffusion coefficient and reactive factor respectively, t is the time, y is the distance and $u(y, t)$ is the population density. The solution of Fisher's equation has been studied using

many different computational approaches. This includes the numerical solutions that were presented in [13] with a pseudo-spectral approach. Gazdag and Canosa [25] were the first to study numerical solutions of Fisher's equation using pseudo-spectral method. Later, many researchers studied numerical solutions of the Fisher's equation. Hagstrom and Keller [12] presented asymptotic boundary conditions by using a centred finite-difference algorithm. The numerical approach which was named Accurate Space Derivatives (ASD) method was carried out efficiently by the use of the Fast Fourier Transform (FFT) algorithm in [10, 13]. Mittal and Arora [48] used B-spline scheme to find the solution of the Fisher's equation.

Thirdly, the Burgers-Fisher equation is considered as another example of the nonlinear evolution equation. The common form of the Burgers-Fisher equation can be expressed as:

$$u_t = u_{yy} - \alpha u^{\gamma} u_y - \beta u(u^{\gamma} - 1), \tag{2.6}$$

where α, γ and β are non-zero parameters [38]. The Burgers-Fisher equation occurs in many areas of sciences and physical applications, for example in modeling of gas dynamics, financial mathematics and fluid mechanics. In this study, Equation 2.6 has been solved using the numerical approaches BSQLM and the SQLM and the convergence of the method proved to be rapid. Many authors have investigated the use and application of the Burgers-Fisher equation. Javidi and Golbabai [34] introduced a spectral collocation method for the solution of Burgers-Fisher equation. Dhawan et al. [26] solved the same equation using a Multi Quadratic Scheme. They mentioned that this type of equation has many applications in gas dynamics. Kaya and Sayed [36] used Adomian Decomposition Method (ADM) to solve nonlinear evolution partial differential equations of this nature. They found that solutions are very rapidly convergent by utilising the ADM. Furthermore, the Adomian Decomposition

Method does not require discretisation of the variables, (in time and space), as it is not affected by calculation rounding off errors and the need for large computer memory and time. Rashid and Abbas [14] solved the Burgers-Fisher equation by first reducing the problem to a system of ordinary differential equations which can be solved by the fourth order Runge-Kutta method. The application of the Burgers-Fisher nonlinear second-order evolution equations describes numerous processes in science and biology, e.g. heat and mass transfer, filtration of liquids, diffusion in chemical reactions, population dynamics etc [38]. Therefore, it is important to study the solution profiles of this nature to handle a large range of problems occurring in day-to-day life

Another type of equation that is investigated in this study is Newell-Whitehead-Segel equation that describes the dynamic behaviour near the bifurcation point of the Rayleigh-Bernard convection of binary fluid mixtures, nonlinear optics, chemical reactions and biological systems [50]. The Newell-Whitehead-Segel equation takes the following form:

$$u_t = u_{yy} - u(1-u)(a-u). \tag{2.7}$$

The Newell-Whitehead-Segel equation has been given considerable attention in recent years by introducing various methods and techniques to solve it. For example, Saravanan and Magesh [63] used the reduced differential transform method and the Adomian Decomposition Method. Among other researchers that have solved Newell-Whitehead-Segel equation is Pue-on [50]. Pue-on used Laplace Adomian Decomposition Method to solve Newell-Whitehead-Segel equation.

Lastly, the Zeldovich equation is investigated. The equation appears in combustion theory. The Zeldovich equation takes the following form:

$$u_t = u_{yy} + u^2 - u^3. \tag{2.8}$$

The function $u(y,t)$ is unknown and it represents the temperature while other terms are concerned with generating heat combustion [45]. The Zeldovich and Newell-Whitehead-Segel equations arise from the well-known Fitzhugh-Nagumo equation. These nonlinear partial differential equations are extensively used as models to describe complex physical occurrences in various fields of science, especially in fluid mechanics, solid-state physics, plasma physics, plasma wave and chemical physics [66]. The Newell-Whitehead-Segel and Zeldovich equations are special cases of the classical Fitzhugh-Nagumo (FN) equation of the form:

$$u_t = u_{yy} - u(1-u)(a-u), \tag{2.9}$$

where a is constant and $u(y,t)$ is the unknown function depending on the time-based variable t and y is the space variable. When $a = -1$ the equation reduces to the Newell-Whitehead-Segel equation and when $a = 0$ reduces to the Zeldovich equation [46, 66]. Many authors have solved this equation using different approaches, among others Motsa [55] using the Homotopy Analysis Method. Jiwari et al. [66] used the polynomial differential quadrature method (PDQM) to solve the Zeldovich equation.

Its clear from the above discussion that many researchers have investigated and solved nonlinear evolution partial differential equations using different methods, but that does not prohibit new research on solving same the equations. For this study, the Spectral Quasilinearisation Method coupled with finite difference in time and Bivariate Spectral Quasilinearisation Method will be used to solve the equations (2.1-2.3). The approximated results are then compared to exact solutions of each type of equation available in the literature to show the accuracy of the method. The Zeldovich and Newell-Whitehead-Segel equations describe the interaction between diffusion, convection reaction and diffusion transports. It has been highlighted earlier that there is no general procedure for finding analytic solutions of the nonlinear diffusion equation to date. Numerical solutions are of great importance in approximating the solutions of many physical problems. There are many researchers who used various numerical procedures to obtain the numerical solution of nonlinear evolution partial differential equations of the type given by equations (2.1- 2.3). In the past few years, several powerful mathematical methods such as Adomian Decomposition Method, Homotopy Analysis Method have been used in attempting to solve the equation. It is still demanding to solve these equations, either numerically or analytically. As a result several assumptions have to be made unnecessarily to make nonlinear models solvable [21, 69]. Evolution problems like other problems in mathematics have some numerical properties that need to be taken into consideration when solving them. In the next section, the properties of numerical methods are discussed.

2.2 Properties of Numerical Methods

In scientific computation, the numerical methods used to solve problems should be robust. A numerical method is said to be robust if the conclusions remain true, even though the model is not perfect. The robustness in the numerical approximation is an important property since in real life there is no perfect model even if the perfect information is available to construct the model [44]. There are several properties that need to be considered in numerical analysis which include; accuracy, stability, consistency and convergence. It is important to choose a method considering these concepts. The summary of these numerical properties is discussed below.

2.2.1 Accuracy and Consistency

Numerical schemes are used to approximate the solution of evolution equations. In the algorithm development, when approximating the function $u(y, t)$ there are errors that arise. The errors that are associated with the algorithm development of the numerical scheme are:

- Convergence error: it is the difference in numerical solution and exact solutions of the given equations. It is also called iteration error.
- Modeling error: Modeling errors arise due to the difference between the real problem and its formulation as a mathematical model.
- Truncation error: Truncation error refers to the error in a method, which occurs because some series (finite or infinite) is truncated to a fewer number of terms. Such errors are essentially algorithmic errors and can predict the extent of the error that will occur in the method.
- Round off error: Round off error occur because of the computing machine inability to deal with certain numbers. Such numbers need to be rounded off to some near approximation which is dependent on the word size used to represent numbers of the device.

A scheme is consistent if the operator reduces to the original differential equation as the increases in the independent variables vanish. Consistency requires that the original equations can be recovered from the arithmetical equations. Clearly this should be a minimum requirement for any discretisation. Consistency is necessary for convergence, but not every consistent scheme is convergent [18].

2.2.2 Convergence and Stability

For a numerical scheme to be useful it needs to correspond to the partial differential equation that is approximated. A numerical method is said to be convergent if the solution of the discrete equations tends to exact solution of the differential equation as the distance between the computational grid is defined. The significance of spectral methods is that they can achieve high accuracy with little more resolution than is required to achieve moderate accuracy [25]. The fundamental problem of the numerical analysis in the boundary value problems is to find the approximate solution $u(y,t)$ which converges to exact solution as N_y increases for some given time interval $[0,T]$. To estimate the error, the estimated solution is subtracted from the exact solution. The primary result is the Lax-Richtmyer equivalence theorem which states that stability is equivalent to convergence for consistent approximations to well-posed linear problems [25]. It is important to note that the theorem is applicable to any discretisation; real fluid dynamics is usually nonlinear and a typical problem is usually boundary value or mixed initial and boundary value problems. Let the infinity norm error is approximated as

$$E_i = \|u_i^n - u_i^*\|_\infty, \quad 0 \le i \le N_y,$$

where u_i^n is the approximated solution, ${u_i}^*$ is the exact solution at time level (t) and N_y represent collocation points in the space direction. The scheme is consistent if as N_y tends to infinity, infinity norm error goes to zero and then the scheme is said to be convergent [29]. For nonlinear evolution problems which are influenced by boundary conditions, convergence and stability are difficult to prove. Convergence can be proved by repeating the experiments many times. The study to be conducted solves the nonlinear evolution equations which are linearised using QLM (see Section 3.3). Taylor series is well known to converge in the largest circle around the expansion point that does not contain any singularities. This result generalises in a straightforward manner to interpolating polynomials, where the nodes are scattered over an interval rather than all taken at one point. If the method is stable the solution obtained converges to the exact solution [58]. In this chapter both the mathematical models of interest and the numerical properties were discussed. The next chapter will introduce the Spectral Quasilinearisation Method coupled with finite difference in time (SQLM) and show how it is formulated and applied to solve mathematical problems.

Chapter 3

Spectral Quasilinearisation Method Coupled with Finite Difference

3.1 Introduction

In this chapter, spectral quasilinearisation method (SQLM) for solving the partial differential equation (2.1) is presented. The quasilinearisation technique is essentially a generalized Newton-Raphson Method that was originally used by Bellman and Kalaba [16] for solving functional equations. The SQLM uses Chebyshev spectral method combined with quasilinearisation method (QLM). The governing nonlinear equations are linearised using the Newton-Raphson based quasilinearisation method (QLM), then integrated using Chebyshev spectral collocation method [52]. The method has been used successfully in solving nonlinear boundary layer problems by Motsa [55]. Our focus in this study is to apply this method to second order nonlinear evolution partial differential equations. The SQLM uses the finite difference method for time derivatives and spectral method on the space derivatives. According to David et al. [25] finite difference methods refers to method of descritisation which are acquired by estimating a function $u(y,t)$ and its derivative $u'(y,t)$ approximation using the

Taylor series. In this context local refers to the use of neighboring grid points to estimate the function or its corresponding derivative at a specified point. He also defines Spectral methods as a great methods used for the solution of partial differential equations. In contrast with finite difference methods, spectral methods are global methods, which means the computation at any given point depends not only on information at neighboring points, but on information from the whole domain. Spectral methods converge exponentially, which gives them superiority in accuracy over local methods. Global methods are better than local methods when the solution differs significantly in time or in space, when very high spatial resolution is required, and also when long time integration is needed [25]. It is then important to note that the finite difference methods are local methods while the spectral methods are global methods. The SQLM takes advantage of both methods to try to improve the accuracy of the local methods. The SQLM is suitable for both ordinary differential equations (ODEs) and partial differential equations (PDEs). For both cases, if the equation is nonlinear it is always better to split the equation into linear and nonlinear components then linearise the nonlinear part using the QLM. The difference between ODE and PDE is that for PDEs, there are two or more independent variables while in ODEs there is only one independent variable. For the purpose of this study, our focus is on PDEs which are discussed in the next section.

3.2 Describing the Spectral Quasilinearisation Method with Finite Difference

This section describe the Specral Quasilinearisation Method coupled with finite difference (SQLM) for solving partial differential equations. The spectral quasilinearisation method uses a quasilinearisation technique which is basically a generalised Newton-

Raphson Method that was originally used by Bellman and Kalaba [16] for solving functional equations. The idea used in the construction of iterative techniques for nonlinear evolution equations is a critical task in numerical analysis. The SQLM is based on the Newton–Raphson method which is often used as a starting point in the development of iterative methods with higher order convergence since it converges to the root quadratically [56]. The SQLM has been used to solve boundary layer problems [58]. In this study the method is used to solve second nonlinear evolution problems. The SQLM is used to approximate $u(y,t)$, the solution of equation (2.1-2.3). Since this method is spectral-based, the domain is defined globally on $[-1,1]$. The governing equation is a nonlinear evolution partial differential equation defined on $0 \leq y \leq l$. Before using the spectral method, the domain needs to be transformed from $0 \leq y \leq l$ to $-1 \leq x \leq 1$ and the transformation equation $y = l(x+1)/2$ is used. An application of the chain rule gives

$$\frac{\partial u}{\partial x} = \frac{2}{l}\frac{\partial u}{\partial y} \qquad \textit{and} \qquad \frac{\partial^2 u}{\partial x^2} = \left(\frac{2}{l}\right)^2 \frac{\partial^2 u}{\partial y^2}. \tag{3.1}$$

Thus, the governing partial differential equation (2.1) is expressed as:

$$\frac{\partial u}{\partial t} = L\left(u, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}\right) + N\left(u, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}\right), \quad -1 \leq x \leq 1, t \in [0,T] \tag{3.2}$$

with the initial and boundary conditions

$$u(x,0) = u_0(x), \quad -1 \leq x \leq 1, t = 0, \tag{3.3}$$

$$u(-1,t) = g_0(t), \quad u(1,t) = g_1(t), t \in [0,T]. \tag{3.4}$$

The spectral method uses all available function values to build the necessary approximation, thus rendering them global methods [31]. According to David et al. [25], the spectral methods have become progressively common since the development of fast transformed methods. As a result, it has been used in weather prediction, numerical simulation of turbulence flow and other problems where high accuracy is desired for complicated solutions [72]. The method involves presenting the solution to a given problem as a truncated series of known functions of independent variables. The spectral method gives results of remarkable accuracy with the efficient use of computer resources. It is chosen considering the following:

(a) Accuracy: To achieve the usefulness of the spectral method it is crucial to design it to give greater accuracy than can be obtained using other methods like finite difference methods. The choice of spectral method representation depends on the kind of boundary conditions involved in the problem.

(b) Efficiency: The spectral method must produce more accurate results than the other methods that were traditionally used in the past, for example, finite difference method [25].

The nonlinear part of the equation for which the study seeks to approximate is linearised using Quasilinearisation Method (QLM). The linearisation is discussed in the next section.

3.3 Quasilinearisation Method (QLM) and Differentiation Matrix

The QLM is a very efficient method for constructing approximate solutions to nonlinear problems. The method is a Taylor series numerical approach in which the truncation error is chosen so that the convergence of the iterations is quadratic [51]. The origin of the method lies in the theory of dynamic programming and was first initiated by Bellman and Kabala in 1965 [16]. In this study, QLM is used to linearise the nonlinear terms in equation (3.2). The nonlinear part is $N\left(u, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}\right)$ but for convenience in notation $N(u, u'u'')$ will be used, where u' and u'' are first and second partial derivatives with respect to x. The nonlinear part $N(u, u'u'')$ is then linearised as follows:

$$N(u, u'u'') \approx N\left(u_i, u_i', u_i''\right) + \frac{\partial N}{\partial u_i}\left(u_{i+1} - u_i\right) + \frac{\partial N}{\partial u'_i}\left(u_{i+1}' - u_i'\right) \tag{3.5}$$

$$+\frac{\partial N}{\partial u_i''}\left(u_{i+1}'' - u_i''\right),$$

where the subscripts i and $i+1$ represent the current and next iteration respectively. Equation (3.5) can be written as:

$$N\left(u, u', u''\right) \approx N\left(u_i, u_i', u_i''\right) - \sum_{p=0}^{2} \frac{\partial N}{\partial u_i^{(p)}} u_i^{(p)} + \sum_{p=0}^{2} \frac{\partial N}{\partial u_i^{(p)}} u_{i+1}^{(p)}. \tag{3.6}$$

In equation (3.6) the index $0, 1$ and 2 represents derivatives of u, u' and u'' with respect to x when the sum is expanded. As a result, equation (3.2) can be written as follows:

$$\frac{\partial u_{i+1}}{\partial t} = L(u_{i+1}, u_{i+1}', u_{i+1}'') + \sum_{p=0}^{2} \frac{\partial N}{\partial u_i^{(p)}} u_{i+1}^{(p)} + N\left(u_i, u_i', u_i''\right) \tag{3.7}$$

$$-\sum_{p=0}^{2} \frac{\partial N}{\partial u_i^{(p)}} u_i^{(p)}, \quad x \in [-1, 1], \quad t \geq 0$$

which is now linear. Since our time is defined on $[0, T]$, the time derivative at the next time interval is approximated by $\frac{\partial u_{i+1}}{\partial t}$. More on time decritisation is discussed in Section 3.4. It is now possible to apply Chebyshev spectral collocation method in equation (3.7). The method is defined on a global domain $[-1, 1]$ in the x-direction as stated earlier. The grid points called collocation points are the Chebyshev -Gauss-Lobatto points defined by

$$x_r = \cos\left(\frac{r\pi}{N_x}\right), \quad r = 0, 1, 2, \ldots, N_x$$

[71].The underlying idea of the collocation method is to approximate the unknown solution of $u(x,t)$ in the entire domain by an interpolating higher order Lagrange polynomial at the given collocation points. The partial derivatives in the x-variable are approximated by the derivatives of the Lagrange polynomial. This N_x^{th} order polynomial is chosen such that it satisfies the boundary conditions of the given nonlinear partial differential equation. Consider $u_{N_x}(x,t)$ to be the approximated solution and residual to be defined as

$$R_{N_x}(x,t) = \frac{\partial u_{i+1}}{\partial t} - L(u_{i+1}, u'_{i+1}, u''_{i+1}) - \sum_{p=0}^{2} \frac{\partial N}{\partial u_i^{(p)}} u_{i+1}^{(p)} - N\left(u_i, u'_i, u''_i\right) + \sum_{p=0}^{2} \frac{\partial N}{\partial u_i^{(p)}} u_i^{(p)}.$$

The residual vanishes at the interior grid points. Therefore $R_{N_x}(x_r,t) = 0$ for $r \in$ {0,1,2,...,N$_x$}, leading to $N_x + 1$ equations. Equation (3.7) is evaluated at x_r, $r = 0, 1, 2, \ldots, N_x$. Let $u(x,t)$ be an interpolating polynomial given by

$$u_{N_x}(x,t) = \sum_{r=0}^{u_x} l_r(x) u(x_r,t) \quad r = 0, 1, 2, 3, \ldots, N_x. \tag{3.8}$$

The functions $l_r(x)$ are Lagrange cardinal polynomial that take the form:

$$l_r(x) = \prod_{k=0, k\neq r}^{N_x} \frac{x - x_k}{x_r - x_k}.$$

At each x_k, the values of l_r is either 0 or 1; these indices are given as follows

$$l_r(x_k) = \begin{cases} 1, & k = r \\ 0, & k \neq r \end{cases}. \tag{3.9}$$

Differentiating equation (3.8) with respect to x, once, twice,... m times gives

$$\frac{\partial u_{N_x}(x,t)}{\partial x} = \sum_{k=0}^{N_x} l'_r(x) u(x_r,t), x = x_s; s = 0, 1, 2, \ldots, N_x$$

$$\frac{\partial^2 u_{N_x}(x,t)}{\partial x^2} = \sum_{k=0}^{N_x} l''_r(x) u(x_r,t),$$

$$\vdots$$

$$\frac{\partial^m u_{N_x}(x,t)}{\partial x^m} = \sum_{k=0}^{N_x} l_r^m(x)u(x_r,t).$$

Which can be written as

$$u' = D\mathbf{u},$$

$$u'' = (D\mathbf{u})' = Du' = D(D\mathbf{u}) = D^2\mathbf{u},$$

$$\vdots$$

$$u^{(m)} = D^{(m)}\mathbf{u},$$

where $\mathbf{u} = [u(x_0,t), u(x_1,t), \ldots, u(x_{N_x},t)]^T$ and $D = D_{sr} = l_r'(x_s)$. The entries of matrix D are computed using Theorem 3.3. Chebyshev spectral differentiation matrix: For each $N_x \geq 1$, let rows and columns of the $(N_x+1)\times(N_x+1)$ Chebyshev spectral differentiation matrix D be indexed from 0 to N_x. The entries of this matrix are

$$(D)_{00} = \frac{2N_x^2+1}{6}, \quad (D)_{N_xN_x} = -\frac{2N_x^2+1}{6}, \tag{3.10}$$

$$(D)_{jj} = \frac{-x_j}{2(1-x_j^2)}, \quad j = 1,2,\ldots,N_x-1, \tag{3.11}$$

$$(D)_{rj} = \frac{c_r(-1)^{r+j}}{c_j(x_r-x_j)}, r \neq j, r,j = 1,2,\ldots,N_x-1, \tag{3.12}$$

where

$$c_r = \begin{cases} 2 & r = 0 \quad or \quad N_x, \\ 1 & otherwise. \end{cases}$$

The application of the spectral method on the linearised PDE in equation (3.7) gives

$$\frac{\partial u_{i+1}}{\partial t} = L(\mathbf{u}_{i+1}, D\mathbf{u}_{i+1}, D^2\mathbf{u}_{i+1}) + \sum_{p=0}^{2} a_{1,p}D^p\mathbf{u}_{i+1} + N\left(\mathbf{u}_i, D\mathbf{u}_i, D^2\mathbf{u}_i\right) - \sum_{p=0}^{2} a_{1,p}D^p\mathbf{u}_i, \tag{3.13}$$

where $a_{1,0} = \dfrac{\partial N}{\partial u}, a_{1,1} = \dfrac{\partial N}{\partial u'}$ and $a_{1,2} = \dfrac{\partial N}{\partial u''}$.

According to Trefethen [71], the purpose of all these spectral differentiation matrices is to replace the derivatives by differentiation matrices. Space derivatives are approximated by differentiation matrices and time derivatives are approximated by finite difference methods which are discussed in the next section.

3.4 Time Descritisation

This section discusses the time discretisation of the parabolic nonlinear evolution equations. The time stepping method is used to compute the solution. Consider the partial differential equation of this form

$$\frac{\partial u_{i+1}}{\partial t} = \left(L(I, D, D^2) + \sum_{p=0}^{2} a_{1,p} D^p \right) \mathbf{u}_{i+1} + N\left(\mathbf{u}_i, D\mathbf{u}_i, D^2\mathbf{u}_i\right) - \sum_{p=0}^{2} a_{1,p} D^p \mathbf{u}_i, \quad (3.14)$$

where I is an $(N_x+1) \times (N_x+1)$ identity matrix. Suppose $0 \leq t < T$ and let $0 = t^0 < t^1 < t^2 < \cdots < t^{n+1} = T$. Considering the short time interval $[t^n, t^{n+1}]$, where $t^{n+1} = t^n + \Delta t$ and Δt is the time step t, the initial condition $u(x, t^n) = u^n$ is used to compute $u^{n+1} \approx u(x, t^{n+1})$ at the next time interval. The simplest time stepping method involves differencing of u between time level n and $n+1$. Hence at time level n

$$\frac{\partial u^n}{\partial t} = \frac{u^{n+1} - u^n}{\Delta t}. \quad (3.15)$$

The method of time stepping affects both efficiency and accuracy of the approximate solution to transient problems [42].

For this work, three types are considered, namely: Explicit, Implicit and Crank-Nicolson Methods. The time-stepping approaches are described in the next section.

3.4.1 Explicit Spectral Quasilinearisation Method (ESQLM)

In an Explicit Spectral Quasilinearisation Method (ESQLM), the solution is approximated at the present time interval n to get the solution at $n+1$. The terms with lower script i are assumed to be known from the previous iteration and those with $i+1$ are the terms at next iterations. Evaluating equation (3.14) at the time level n using the forward difference approximation

$$\frac{\partial u^n_{i+1}}{\partial t} = \frac{u^{n+1}_{i+1} - u^n_{i+1}}{\Delta t} \tag{3.16}$$

that give rise to

$$\mathbf{u}^{n+1}_{i+1} = \mathbf{u}^n_{i+1} + \Delta t \left(L(I, D, D^2) + \sum_{p=0}^{2} a^n_{1,p} D^p \right) \mathbf{u}^n_{i+1} \tag{3.17}$$

$$+\Delta t \left(N \left(\mathbf{u}^n_i, D\mathbf{u}^n_i, D^2\mathbf{u}^n_i \right) \right) - \Delta t \left(\sum_{p=0}^{2} a^n_{1,p} D^p \mathbf{u}^n_i \right).$$

If we let

$$A_1 = \left(I + \Delta t \left(L(I, D, D^2) + \sum_{p=0}^{2} a^n_{1,p} D^p \right) \right) \tag{3.18}$$

and the vector

$$K_1 = \Delta t \left(N \left(\mathbf{u}^n_i, D\mathbf{u}^n_i, D^2\mathbf{u}^n_i \right) - \sum_{p=0}^{2} a^n_{1,p} D^p \mathbf{u}^n_i \right), \tag{3.19}$$

then re-arrange equation (3.17) gives

$$\left[\begin{array}{c} u^{n+1}_{i+1}(x_0) \\ \hline u^{n+1}_{i+1}(x_1) \\ \vdots \\ u^{n+1}_{i+1}(x_{N_x-1}) \\ \hline u^{n+1}_{i+1}(x_{N_x}) \end{array}\right] = \left[\begin{array}{ccccccc} 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \hline & & & & & & \\ & & & \mathbf{A}_1 & & & \\ & & & & & & \\ \hline 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} u^{n}_{i+1}(x_0) \\ \hline u^{n}_{i+1}(x_1) \\ \vdots \\ u^{n}_{i+1}(x_{N_x-1}) \\ \hline u^{n}_{i+1}(x_{N_x}) \end{array}\right] + \left[\begin{array}{c} u(1,t) \\ \hline \\ \mathbf{K}_1 \\ \\ \hline u(-1,t) \end{array}\right],$$

which is same as;

$$\mathbf{u}^{n+1}_{i+1} = A_1 \mathbf{u}^n_{i+1} + K_1, \tag{3.20}$$

where $\mathbf{u}^{n+1}_{i+1}$ is a column vector $[u(x_0, t^{n+1}), u(x_1, t^{n+1}), \ldots, u(x_{N_x}, t^{n+1})]^T$ at a next iteration $i+1$. Given $\mathbf{u}^n_i$, we compute $\mathbf{u}^n_{i+1}$ using equation (3.20) and applying the boundary conditions on K_1 as shown above. The scheme represented by equation (3.20) is called fully explicit since it computes $\mathbf{u}_{i+1}^{n+1}$ from known quantities at time t^n. When $n = 0$, equation (3.20) can be used to calculate u_{i+1}^1 since u_{i+1}^0 is known from initial conditions. When $n = 1$, equation (3.20) can be used to compute u_{i+1}^2 since u_{i+1}^1 is now known from the previous iteration. Similarly for $n = 2, 3, \ldots\ldots$ and using equation (3.20) gives $u_{i+1}^3, u_{i+1}^4, \ldots\ldots$ This subsection we discussed and showed how the explicit method for SQLM is obtained. The model has been represented in Matrix form as shown by equation (3.20) which will be solved using Matlab. The next subsection will introduce the implicit scheme for SQLM.

3.4.2 Implicit Spectral Quasilinearisation Method (ISQLM)

We now consider the Implicit Spectral Quasilinearisation Method (ISQLM). This scheme is very similar to Explicit Spectral Quasilinearisation Method except that equation (3.14) is evaluated at $n+1$. In $\dfrac{\partial u_{i+1}^{n+1}}{\partial t} = \left(L(I, D, D^2) + \sum_{p=0}^{2} a_{1,p} D^p \right) \mathbf{u}_{i+1}^{n+1} + N\left(\mathbf{u}_i, D\mathbf{u}_i^{n+1}, D^2\mathbf{u}_i^{n+1}\right) - \sum_{p=0}^{2} a_{1,p} D^p \mathbf{u}_i^{n+1}$, we replace time derivative with backward difference approximation;

$$\frac{\partial u_{i+1}^{n+1}}{\partial t} = \frac{u_{i+1}^{n+1} - u_{i+1}^{n}}{\Delta t}$$

to get

$$\mathbf{u}_{i+1}^{n+1} = \mathbf{u}_{i+1}^{n} + \Delta t \left(L(I, D, D^2) + \sum_{p=0}^{2} a_{1,p}^{n+1} D^p \right) \mathbf{u}_{i+1}^{n+1} \tag{3.21}$$

$$+\Delta t \left(N\left(\mathbf{u}_i^{n+1}, D\mathbf{u}_i^{n+1}, D^2\mathbf{u}_i^{n+1}\right) - \sum_{p=0}^{2} a_{1,p}^{n+1} D^p \mathbf{u}_i^{n+1} \right).$$

Rearranging equation (3.21) leads to

$$\left[I - \Delta t \left(L(I, D, D^2) + \sum_{p=0}^{2} a_{1,p}^{n+1} D^p \right) \right] \mathbf{u}_{i+1}^{n+1} = \mathbf{u}_{i+1}^{n} \tag{3.22}$$

$$+\Delta t \left(N\left(\mathbf{u}_i^{n+1}, D\mathbf{u}_i^{n+1}, D^2\mathbf{u}_i^{n+1}\right) - \sum_{p=0}^{2} a_{1,p}^{n+1} D^p \mathbf{u}_i^{n+1} \right).$$

The above equation (3.22) can be written as

$$A_2\mathbf{u}_{i+1}^{n+1} = B_2\mathbf{u}_{i+1}^{n} + K_2, \tag{3.23}$$

where

$$A_2 = \left[I - \Delta t\left(L(I, D, D^2) + \sum_{p=0}^{2} a_{1,p}^{n+1} D^p\right)\right],$$

$$B_2 = I, \quad K_2 = \Delta t\left(N\left(I, D, D^2\right) - \sum_{p=0}^{2} a_{1,p}^{n+1} D^p\right),$$

I is the $(N_x+1)\times(N_x+1)$ identity matrix and $B = I$.

Before we solve equation (3.23) for $\mathbf{u}_{i+1}^{i+1}$, we impose boundary conditions as follows

$$\left[\begin{array}{ccccc} 1 & 0 & \cdots & 0 & 0 \\ \hline & & & & \\ & & \mathbf{A}_2 & & \\ & & & & \\ \hline 0 & 0 & \cdots & 0 & 1 \end{array}\right]\left[\begin{array}{c} u_{i+1}^{n+1}(x_0) \\ \hline u_{i+1}^{n+1}(x_1) \\ \vdots \\ u_{i+1}^{n+1}(x_{N_x-1}) \\ \hline u_{i+1}^{n+1}(x_{N_x}) \end{array}\right] = \left[\begin{array}{ccccc} 0 & 0 & \cdots & 0 & 0 \\ \hline & & & & \\ & & \mathbf{B}_2 & & \\ & & & & \\ \hline 0 & 0 & \cdots & 0 & 0 \end{array}\right]\left[\begin{array}{c} u_{i+1}^{n}(x_0) \\ \hline u_{i+1}^{n}(x_1) \\ \vdots \\ u_{i+1}^{n}(x_{N_x-1}) \\ \hline u_{i+1}^{n}(x_{N_x}) \end{array}\right] + \left[\begin{array}{c} u(1,t) \\ \hline \\ \mathbf{K}_2 \\ \\ \hline u(-1,t) \end{array}\right].$$

Solving the above system represented by equation (3.4.2) by inverting matrix A_2 gives

$$\mathbf{u}_{i+1}^{n+1} = A_2^{-1}(B_2\mathbf{u}_{i+1}^{n} + K_2). \tag{3.24}$$

When $n = 0$, equation (3.23) can be used to compute u_{i+1}^1 since u_{i+1}^0 is known from initial conditions. When $n = 1$, equation (3.23) can be used to compute u_{i+1}^2 since u_{i+1}^1 is now known from the previous iteration. Similarly for $n = 2, 3, \ldots\ldots$ and using equation (3.23) gives $u_{i+1}^3, u_{i+1}^4, \ldots\ldots$

3.4.3 Crank-Nicolson Spectral Quasilinearisation Method (CN-SQLM)

Another way used to discretise time step is the Crank-Nicolson method. In this method, equation (3.14) is evaluated at time level $n + 1/2$ to get

$$\frac{\partial u_{i+1}^{n+1/2}}{\partial t} = \left(L(I,D,D^2) + \sum_{p=0}^{2} a_{1,p} D^p \right) \mathbf{u}_{i+1}^{n+1/2} + N\left(\mathbf{u}_i, D\mathbf{u}_i^{n+1/2}, D^2\mathbf{u}_i^{n+1/2}\right) - \sum_{p=0}^{2} a_{1,p} D^p \mathbf{u}_i^{n+1/2}. \tag{3.25}$$

Then approximate $\frac{\partial u_{i+1}^{n+1/2}}{\partial t}$ with central difference approximation;

$$\frac{\partial u_{i+1}^{n+1/2}}{\partial t} = \frac{u_{i+1}^{n+1} - u_{i+1}^{n}}{\Delta t},$$

and approximate $\mathbf{u}_{i+1}^{n+1/2}$ with

$$u_{i+1}^{n+1/2} = \frac{u_{i+1}^{n+1} + u_{i+1}^{n}}{2}.$$

This changes equation (3.25) above to equation (3.26)

$$\mathbf{u}_{i+1}^{n+1} = \mathbf{u}_{i+1}^{n} + \Delta t \left(L(I,D,D^2) + \sum_{p=0}^{2} a_{1,p}^{n+1/2} D^p \right) \mathbf{u}_{i+1}^{n+1/2} \tag{3.26}$$

$$+\Delta t \left(N\left(\mathbf{u}_i^{n+1/2}, D\mathbf{u}_i^{n+1/2}, D^2\mathbf{u}_i^{n+1/2}\right) - \sum_{p=0}^{2} a_{1,p}^{n+1/2} D^p \mathbf{u}_i^{n+1/2} \right).$$

The above equation (3.26) can be written as

$$A_3 \mathbf{u}_{i+1}^{n+1} = B_3 \mathbf{u}_{i+1}^{n} + K_3, \tag{3.27}$$

where

$$A_3 = I - \frac{1}{2}\left[\Delta t \left(L(I,D,D^2) + \sum_{p=0}^{2} a_{1,p}^{n+1} D^p \right) \right],$$

$$B_3 = I + \frac{1}{2}\left[\Delta t \left(L(I,D,D^2) + \sum_{p=0}^{2} a_{1,p}^{n} D^p \right) \right],$$

$$K_3 = \Delta t \left(N \left(\mathbf{u}_i^{n+1/2}, D\mathbf{u}_i^{n+1/2}, D^2\mathbf{u}_i^{n+1/2} \right) - \sum_{p=0}^{2} a_{1,p}^{n+1/2} D^p \mathbf{u}_i^{n+1/2} \right)$$

and I is the $(N_x{+}1{\times}N_x{+}1)$ identity matrix. Imposing the boundary conditions we get

$$\begin{bmatrix} 1 \quad 0 \quad \cdots \quad 0 \quad 0 \\ \hline \\ \mathbf{A}_3 \\ \\ \hline 0 \quad 0 \quad \cdots \quad 0 \quad 1 \end{bmatrix} \begin{bmatrix} u_{i+1}^{n+1}(x_0) \\ \hline u_{i+1}^{n+1}(x_1) \\ \vdots \\ u_{i+1}^{n+1}(x_{N_x-1}) \\ \hline u_{i+1}^{n+1}(x_{N_x}) \end{bmatrix} = \begin{bmatrix} 0 \quad 0 \quad \cdots \quad 0 \quad 0 \\ \hline \\ \mathbf{B}_3 \\ \\ \hline 0 \quad 0 \quad \cdots \quad 0 \quad 0 \end{bmatrix} \begin{bmatrix} u_{i+1}^{n}(x_0) \\ \hline u_{i+1}^{n}(x_1) \\ \vdots \\ u_{i+1}^{n}(x_{N_x-1}) \\ \hline u_{i+1}^{n}(x_{N_x}) \end{bmatrix} + \begin{bmatrix} u(1,t) \\ \hline \\ \mathbf{K}_3 \\ \\ \hline u(-1,t) \end{bmatrix}.$$

Solving equation (3.27) by inverting matrix A_3, gives

$$\mathbf{u}_{i+1}^{n+1} = A_3^{-1}(B_3\mathbf{u}_{i+1}^{n} + K_3). \tag{3.28}$$

When $n = 0$, equation (3.28) can be used to compute u_{i+1}^1 since u_{i+1}^0 is known from initial conditions. When $n = 1$, equation (3.28) can be used to compute u_{i+1}^2 since u_{i+1}^1 is now known from the previous iteration. Similarly for $n = 2, 3, \ldots\ldots$ and using equation (3.28) gives $u_{i+1}^3, u_{i+1}^4, \ldots\ldots$

3.5 Summary

This chapter discussed the SQLM and show how it can be applied to nonlinear evolution problems, first by describing the method, and by explaining how QLM works on nonlinear evolution partial differential equations as first used by Bellman and Kabala [16]. Time dicretisation was discussed which a crucial step is since the SQLM uses the finite difference in time derivatives. Applying finite difference method on time derivative gives rise to different forms of SQLM which are ESQLM, ISQLM and CN-SQLM. In Chapter 5 we shall show how this method is applied to specific examples.

Chapter 4

Bivariate Spectral Quasilinearisation Method

This chapter presents the Bivariate Spectral Quasilinearisation Method (BSQLM) for solving the nonlinear evolution problems. Nonlinear evolution problems such as partial differential equations with time t as one of the independent variables arise in many fields of mathematics and other divisions of science. In physics, biology, mechanics and material science. The examples include but are not limited to the Naiver-Stokes from fluid mechanics, the nonlinear diffusion equation from heat transfer and biological science [73]. Consider the second order nonlinear evolution parabolic partial differential equation discussed in Chapter 2:

$$\frac{\partial u}{\partial \tau} = L\left(u, \frac{\partial u}{\partial y}, \frac{\partial^2 u}{\partial y^2}\right) + N\left(u, \frac{\partial u}{\partial y}, \frac{\partial^2 u}{\partial y^2}\right), \quad 0 < y < l, \tau \in (0, T], \tag{4.1}$$

with the initial and boundary conditions

$$u(y, 0) = u_0(y), \quad 0 \leq y < l, \tau = 0, \tag{4.2}$$

$$u(0, \tau) = g_0(\tau), \quad u(l, \tau) = g_1(\tau), \quad \tau \in (0, T], \tag{4.3}$$

where $u(\tau, y)$ is the solution to be approximated. For consistency in notation, in Chapter 2 t was used as the time variable, and in this chapter τ will be used to denote the time variable. Both independent variables τ and y need to be transformed to $[-1, 1]$ independently. Consider the y-variable. The region $0 \leq y \leq l$ is converted to $-1 \leq x \leq 1$ using linear transformation $y = l(x+1)/2$, the τ takes the positive values since it measures time variable. The τ-variable is transformed using $\tau = T(t+1)/2$ to $t \in [-1, 1]$ so that spectral method can be applied. After transformation equation (4.1) can be written as:

$$\frac{\partial u}{\partial t} = L\left(u, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial y^2}\right) + N\left(u, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}\right), \quad -1 \leq x \leq 1, \quad -1 \leq t \leq 1. \tag{4.4}$$

The solution $u(x,t)$ of equation (4.4) is approximated by a bivariate Lagrange interpolation polynomial which is the same as 2-D Lagrange Interpolation polynomial. The method of the 2-D Lagrange interpolation is based on the 1-D Lagrange interpolation. According to Brezinski et al. [20], bivariate interpolation can be found in the tensor product of univariate interpolation functions. The variables are treated separately and this method is called the classical approach to multivariate interpolation. Since the Bivariate Lagrange interpolation is being applied, one of the variables is kept as a constant while the other is varying. The Lagrange interpolation method is convenient for obtaining the function in explicit form. The Bivariate Lagrange interpolation can be described as

$$u(x,t) = \sum_{r=0}^{N_x} \sum_{s=0}^{N_t} u(x_r, t_s) l_{rs}(x,t) \tag{4.5}$$

with

$$l_{rs} = l_r(x) l_s(t), \; 0 \leq r \leq N_x, \; 0 \leq s \leq N_t, \tag{4.6}$$

and $l_r(x)$ and $l_s(t)$ defined as

$$l_r(x) = \prod_{k=0, k\neq r}^{N_x} \frac{x - x_k}{x_r - x_k}, \;\; l_s(t) = \prod_{k=0, k\neq s}^{N_t} \frac{t - t_k}{t_s - t_k}, \tag{4.7}$$

and

$$l_{rs}(x_k, t_n) = \begin{cases} 1, \ r = k \ , s = n \\ \\ 0, \ \text{otherwise} \end{cases} . \tag{4.8}$$

Where $u(x,t)$ is a polynomial of degree $\leq N_x \times N_t$ interpolating $(N_x+1) \times (N_t+1)$ points [20]. We choose collocation points in both x and t directions which are defined by

$$x_r = \cos\left(\frac{r\pi}{N_x}\right), \ t_s = \cos\left(\frac{s\pi}{N_t}\right), \quad r = 0,1,2,\ldots,N_x, \quad s = 0,1,2,\ldots,N_t. \tag{4.9}$$

The grid points defined above in equation (4.9) are called Chebyshev-Gauss-Labatto points. The grid points in equation (4.9) make it easier to evaluate equation (4.1) on the points in $[-1,1] \times [-1,1]$ in both x and t variables. Since equation (4.4) contains linear and nonlinear terms, nonlinear terms are linearised using the same QLM procedure discussed in Section 3.3. After applying the quasilinearisation method, the right-hand side of equation (4.4) takes the same form as equation (3.7) except for the fact that for BSQLM transformation is also applied in t. The linearised terms are as follows

$$N(u,u',u'') \approx N\left(u_i, u_i', u_i''\right) + \sum_{p=0}^{2} \frac{\partial N}{\partial u_i^{(p)}} \left(u_{i+1}^{(p)} - u_i^{(p)}\right), \tag{4.10}$$

where i and $i+1$ represent the previous and current iterations respectively and primes represent the space derivatives. Let $a_{1,p} = \dfrac{\partial N}{\partial u^{(p)}}$, then

equation (4.4) can be written as

$$\frac{\partial u_{i+1}}{\partial t} - L(u_{i+1}, u_{i+1}', u_{i+1}'') - \sum_{p=0}^{2} a_{1,p} u_{i+1}' = K_i(u_i, u_i', u_i''), \tag{4.11}$$

where

$$K_i(u_i, u_i', u_i'') = N(u_i, u_i', u_i'') - \sum_{p=0}^{2} a_{1,p} u_i'. \tag{4.12}$$

The most crucial step in the implementation of the solution is the evaluating of the time derivative at the grid points t_s for $s = 0, 1, 2, \ldots, N_t$ and the space derivative at the grid point x_r for $r = 0, 1, 2, \ldots, N_x$. Using the Lagrange polynomial in equation (4.5) the values of the time derivatives at collocation points (x_r, t_s) are computed for $s = 0, 1, 2, \ldots, N_t$ as

$$\left.\frac{\partial u}{\partial t}\right|_{x=x_r, t=t_s} = \sum_{g=0}^{N_x} \sum_{h=0}^{N_t} u(x_g, t_p) l_g(x_r) \frac{dl_p(t_s)}{dt} \tag{4.13}$$

which is same as

$$\left.\frac{\partial u}{\partial t}\right|_{x=x_r, t=t_s} = \sum_{p=0}^{N_t} d_{sp} u(x_r, t_p), \tag{4.14}$$

where $d_{sp} = \dfrac{dl_p(t_s)}{dt}$ is the first derivative Chebyshev differentiation matrix of size $(N_t + 1) \times (N_t + 1)$ which is defined in [71]. The space derivatives at the collocation points are worked out as

$$\left.\frac{\partial u}{\partial x}\right|_{x=x_r, t=t_s} = \sum_{g=0}^{N_x} \sum_{p=0}^{N_t} u(x_g, t_p) l_h(t_s) \frac{dl_g(x_r)}{dx} \tag{4.15}$$

which is similar to

$$\left.\frac{\partial u}{\partial x}\right|_{x=x_r, t=t_s} = \sum_{p=0}^{N_x} D_{rg} u(x_g, t_s) = \mathbf{D}\mathbf{u}(x_g, t_s), \tag{4.16}$$

where $D_{rg} = \dfrac{dl_g(x_r)}{dx}$ is the first derivative Chebyshev differentiation matrix of size $(N_x + 1) \times (N_x + 1)$. The second,third, $\ldots\ldots$, m^{th} derivatives are approximated by

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=x_r, t=t_s} = \sum_{p=0}^{N_x} D_{rg}^2 u(x_g, t_s) = \mathbf{D}^2\mathbf{u}(x_g, t_s), \tag{4.17}$$

,

$$\left.\frac{\partial^3 u}{\partial x^3}\right|_{x=x_r, t=t_s} = \sum_{p=0}^{N_x} D_{rg}^3 u(x_g, t_s) = \mathbf{D}^3\mathbf{u}(x_g, t_s), \tag{4.18}$$

$$\vdots$$

$$\left.\frac{\partial^{(m)}u}{\partial x^{(m)}}\right|_{x=x_r,t=t_s} = \sum_{h=0}^{N_x} D_{rg}^{(m)} u(x_g,t_s) = \mathbf{D}^{(m)}\mathbf{u}(x_g,t_s), \quad r = 0,1,2,\ldots,N_x, \tag{4.19}$$

where $D_{rg}^2 = \dfrac{dl_g^2(x_r)}{dx^2}$, $D_{rg}^3 = \dfrac{dl_g^3(x_r)}{dx^3}, \ldots\ldots,$ $D_{rg}^{(m)} = \dfrac{dl_g^{(m)}(x_r)}{dx^{(m)}}$ and $\mathbf{u}(x_g,t_s)$ is defined as

$$\mathbf{u}(x_r,t_s) = [u_s(x_0,t_s), u_s(x_1,t_s), \ldots, u_s(x_{N_x,t_s})]^T. \tag{4.20}$$

The superscript T denotes the transpose. Substituting equation(4.14) and the space derivatives discussed in equation (4.16-4.19) into equation (4.11) gives

$$L(\mathbf{u}_{i+1,s}, \mathbf{D}\mathbf{u}_{i+1,s}, \mathbf{D}^2\mathbf{u}_{i+1,s}) - \sum_{p=0}^{N_t} d_{sp}\mathbf{u}_{i+1,s} + \sum_{p=0}^{2} \mathbf{a}_{1,p}\mathbf{D}\mathbf{u}_{i+1,s} = K_i(\mathbf{u}_{i,s}, \mathbf{D}\mathbf{u}_{i,s}, \mathbf{D}^2\mathbf{u}_{i,s}) \tag{4.21}$$

for $s = 0,1,2,\ldots,N_t$, with

$$\mathbf{a}_{1,p} = \begin{bmatrix} a_{1,p}(x_0,t_s) & & & & \\ & a_{1,p}(x_1,t_s) & & & \\ & & a_{1,p}(x_2,t_s) & & \\ & & & \ddots & \\ & & & & a_{1,p}(x_{N_x},t_s) \end{bmatrix}.$$

Upon including initial conditions for equation (4.11) on equation (4.21), we get

$$L(\mathbf{u}_{i+1,s}, \mathbf{D}\mathbf{u}_{i+1,s}, \mathbf{D}^2\mathbf{u}_{i+1,s}) - \sum_{p=0}^{N_t-1} d_{sp}\mathbf{u}_{i+1,s} + \sum_{p=0}^{2} \mathbf{a}_{1,p}\mathbf{D}\mathbf{u}_{i+1,s} = K_s \tag{4.22}$$

where

$$K_s = K_i(\mathbf{u}_{i,s}, \mathbf{D}\mathbf{u}_{i,s}, \mathbf{D}^2\mathbf{u}_{i,s}) + d_{sN_t}\mathbf{u}_{N_t}, \quad s = 0,1,2,\ldots,N_t-1. \tag{4.23}$$

Equation (4.22) may be written as $N_t(N_x+1) \times N_t(N_x+1)$

$$\begin{bmatrix} A_{0,0} & A_{0,1} & \cdots & A_{0,N_t-1} \\ A_{1,0} & A_{1,1} & \cdots & A_{1,N_t-1} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N_t-1,0} & A_{N_t-1,1} & \cdots & A_{N_t-1,N_t-1} \end{bmatrix} \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_{N_t-1} \end{bmatrix} = \begin{bmatrix} \mathbf{K}_0 \\ \mathbf{K}_1 \\ \vdots \\ \mathbf{K}_{N_t-1} \end{bmatrix}, \text{ where}$$

$$A_{r,r} = L[\mathbf{I}, \mathbf{D}, \mathbf{D}^2] + \sum_{p=0}^{2} \mathbf{a}_{1,p}\mathbf{D}^p - d_{r,r}\mathbf{I}$$

and

$$A_{r,s} = -d_{r,s}\mathbf{I},$$

when $r \neq s$, (4.24)and $\mathbf{I}$ represents the identity matrix of size $(N_x+1) \times (N_x+1)$.

4.1 Summary and Conclusion

This chapter investigated and showed how the BSQLM is applied. This new method of solution uses Bivariate spectral method for solving nonlinear evolution partial differential equations has been proposed and this study will use it to solve the second order evolution partial differential equations. In this chapter, it was shown how this method is formulated and how it works. In the next chapter, this method is applied to different types of equations which are Burgers, Burgers-Fisher, Fisher's, Newell-Whitehead-Segel and Zeldovich equations. The following chapter will focus on problem-solving using this method, the BSQLM discussed in this chapter and the SQLM discussed in Chapter 3.

Chapter 5

Numerical Experiments for Nonlinear Evolution Equations and Discussion

This chapter discusses the approximation of the solutions of nonlinear evolution equations using spectral quasilinearisation method coupled with finite differences in time and the bivariate spectral quasilinearisation method which uses spectral method in both time and space derivatives. It will demonstrate the applications of the methods by solving second order nonlinear evolution equations. These equations include the Burgers equation, Burgers-Fisher equation, Fisher's equation, Newell-Whitehead-Segel equation and Zeldovich equation. All numerical solutions are obtained using Matlab.

5.1 Numerical Convergence Analysis of the Methods

To illustrate the convergence and accuracy of the schemes, the infinity norm error is considered. Since the equations considered have exact solutions, the infinity norm can easily be determined at each iteration. The approximate solution at a particular time level and the corresponding exact solution are used to determine the level of accuracy of the methods. The infinity norm error for any bounded function is defined as

$$E_i = \|u_i^n - u_i^*\|_\infty, \quad 0 \leq i \leq N_y \tag{5.1}$$

where u_i^n is the approximated solution and ${u_i}^*$ is the exact solution at time level t. To determine the level of accuracy for SQLM and BSQLM approximate solution, at a particular time level, comparing it with the exact solution, we use the maximum error or infinity norm error which is defined as

$$E_i = max_i\{|u(y_i, t) - u^*(y_i, t)|, 0 \leq i \leq N_y\} \tag{5.2}$$

where $u^*(y_i, t)$ is the approximate solution and $u(y_i, t)$ is the exact solution at the time level t as reported by Motsa et al. in [57]. The methods will converge if infinity norm errors goes to zero as the number of collocation points increases. It also converges if an increase in collocation points results in a decrease in the infinity norm errors. The infinity norm errors will be presented in a tabular form and graphically. Parameters used to obtain results will be stated explicitly.

5.2 Numerical Experiments

This section present the results obtained from the SQLM and BSQLM for the six examples considered. These methods were discussed in Chapter 3 and Chapter 4 respectively. The main aim of this study is to demonstrate the accuracy and show the applicability of the methods used. Comparison of the infinity norm error results is obtained from the numerical approximation and the exact solutions. A comparison of the methods can be made. Before moving to the numerical solution examples, it is important to note that all the results were obtained using ten iterations.

5.2.1 Burgers Equation

The general Burgers equation is given by:

$$\frac{\partial u}{\partial t} = \varepsilon \frac{\partial^2 u}{\partial y^2} - u\frac{\partial u}{\partial y} + f(y,t), \quad 0 \leq y \leq l, t \geq 0, \tag{5.3}$$

where $\varepsilon > 0$ is the coefficient of the kinematic viscosity [67]. The initial and boundary conditions respectively, are given by

$$u(y,0) = u_0(y), 0 \leq y \leq l, \tag{5.4}$$

$$u(0,t) = g_0(y), \quad u(l,t) = g_1(y). \tag{5.5}$$

The nonlinear term is given by $u\dfrac{\partial u}{\partial y}$ in equation (5.3). According to Nguyen [67], the boundary condition must be well specified in order to achieve a well posed solution. The exact solution of Burgers equation can only be found for restricted values of ε [43]. For the purpose of comparing the numerical methods, the following particular Burgers Equation is considered.

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial y^2} - u\frac{\partial u}{\partial y}, \quad 0 \leq y \leq 2, \quad 0 \leq t \leq 2, \tag{5.6}$$

subject to boundary conditions

$$u(0,t) = \frac{1}{2}\left[1 - \tanh\left\{\frac{1}{4}\left(-15 - \frac{t}{2}\right)\right\}\right] \tag{5.7}$$

and

$$u(l,t) = \frac{1}{2}\left[1 - \tanh\left\{\frac{1}{4}\left(l - 15 - \frac{t}{2}\right)\right\}\right]. \tag{5.8}$$

The initial condition is given by

$$u(y,0) = \frac{1}{2}\left[1 - \tanh\left\{\frac{1}{4}(y - 15)\right\}\right], \quad 0 \le y \le 2, \tag{5.9}$$

and the exact solution of the initial boundary value problem (5.6-5.9) is

$$u(y,t) = \frac{1}{2}\left[1 - \tanh\left\{\frac{1}{4}\left(y - 15 - \frac{t}{2}\right)\right\}\right]$$

which is given in [65].

After applying the transformation in equation (5.6), as discussed in Chapter 2, the term $u\dfrac{\partial u}{\partial y}$ which is nonlinear is linearised using QLM to become

$$N \approx u_i u'_{i+1} + u'_i u_{i+1} - u_i u'_i.$$

Note that $f(y,t) = 0$ in equation (5.3). Equation (5.6) is transformed using $y = l(x+1)/2$ to map the domain on interval $[0,l]$ to $[-1,1]$. It is important to note that $l = 2$ in this case. The BSQLM requires the transformation as discussed in Chapter 4. The infinity norm errors for both SQLM and BSQLM are displayed in Tables 5.1-5.4 with their corresponding central processing times (CPU) in seconds. Figure 5.1 shows the infinity norm error graph using the SQLM and BSQLM for the Burgers equation and plotted in the same axes.

Table 5.1: Infinity Norm Errors for ISQLM in solving Burgers Equation using $N_t = 10001$

$t\backslash N_y$	6	8	10
0.2	3.04413e-004	1.50819e-004	1.00016e-009
0.4	2.89587e-004	1.43474e-004	1.58076e-009
0.6	2.75482e-004	1.36487e-004	1.86990e-009
0.8	2.62063e-004	1.29839e-004	1.99145e-009
1.0	2.49297e-004	1.23515e-004	2.01789e-009
1.2	2.37152e-004	1.17498e-004	1.99124e-009
1.4	2.25598e-004	1.11775e-004	1.93581e-009
1.6	2.14607e-004	1.06329e-004	1.86561e-009
1.8	2.04150e-004	1.01149e-004	1.78867e-009
2.0	1.94203e-004	9.62211e-005	1.70960e-009
CPU Time	3.745158 seconds	4.736429 seconds	5.118239 seconds

Table 5.2: Infinity Norm Errors for ESQLM in solving Burgers Equation using $N_t = 10001$.

$t\backslash N_y$	6	8	10
0.2	3.04414e-004	1.50821e-004	1.00068e-009
0.4	2.89588e-004	1.43477e-004	1.58157e-009
0.6	2.75482e-004	1.36489e-004	1.87090e-009
0.8	2.62064e-004	1.29842e-004	1.99261e-009
1.0	2.49298e-004	1.23518e-004	2.01922e-009
1.2	2.37153e-004	1.17501e-004	1.99273e-009
1.4	2.25599e-004	1.11777e-004	1.93745e-009
1.6	2.14607e-004	1.06332e-004	1.86738e-009
1.8	2.04151e-004	1.01152e-004	1.79058e-009
2.0	1.94203e-004	9.62235e-005	1.71161e-009
CPU Time	1.716948 seconds	1.730464 seconds	1.770547 seconds

Table 5.1 and Table 5.2 shows the infinity error norms of the Burgers equation. The results were obtained from the ISQLM and ESQLM. The results were obtained using $t \in [0, 2]$ in time variable and $y \in [0, 2]$ in the space variable, and printed between $t \in [0.2, 2]$. The numbers of collocation points that were used in both methods are $N_t = 10001$ and $N_y = 6, 8, 10$ in time and space respectively. The central processing time is also given and it increases up to 5.118239 for Implicit SQLM while in Explicit SQLM it goes up to 1.770547. The time can be seen to increase with the increase in the collocation points in the y-variable in both tables while the error norm decreases as N_y increases.

Table 5.3: Infinity Norm Errors for CN-SQLM in solving Burgers Equation using $N_t = 100$.

$t\backslash N_y$	60	80	100
0.2	3.04413e-004	1.50820e-004	8.34626e-011
0.4	2.89587e-004	1.43476e-004	1.31918e-010
0.6	2.75482e-004	1.36488e-004	1.56018e-010
0.8	2.62063e-004	1.29841e-004	1.66169e-010
1.0	2.49297e-004	1.23517e-004	1.68394e-010
1.2	2.37152e-004	1.17500e-004	1.66185e-010
1.4	2.25599e-004	1.11776e-004	1.61570e-010
1.6	2.14607e-004	1.06331e-004	1.55681e-010
1.8	2.04151e-004	1.01151e-004	1.49255e-010
2.0	1.94203e-004	9.62224e-005	1.42663e-010
CPU Time	2.374973 seconds	2.651092 seconds	3.225580 seconds

Table 5.3 shows the infinity norm errors between exact and approximate solution for Burgers equation. The equation was solved using the CN-SQLM. The results were obtained using $N_y = 60, 80$ and 100 and $N_t = 100$. Keeping all other things the same as in ISQLM and ESQLM presented in Table 5.1 and Table 5.2, $t \in [0, 2]$ was used in

the time variable and $y \in [0, 2]$ in the space variable, and printed the results between $t \in [0.2, 2]$. The collocation points for CN-SQLM are $N_y = 60, 80$ and 100 for space variable and $N_t = 100$ in time variable. It can be seen that the CPU time increases as N_y increases. It is also clear that as N_y increases, the infinity norm error decreases.

Table 5.4: Infinity Norm Errors for BSQLM in solving Burgers Equation using $N_t = 10$.

$t \backslash N_y$	6	8	10
0.2	1.32511e-011	1.52101e-014	3.60822e-014
0.4	1.28206e-011	3.50830e-014	5.16254e-014
0.6	1.29469e-011	3.15303e-014	3.23075e-014
0.8	1.18751e-011	2.26485e-014	4.14113e-014
1.0	1.15774e-011	1.74305e-014	7.46070e-014
1.2	1.13397e-011	3.58402e-014	1.00475e-013
1.4	1.05408e-011	3.78586e-014	1.13243e-013
1.6	1.01664e-011	1.48770e-014	4.87388e-014
1.8	9.76508e-012	1.46449e-014	6.04039e-014
2.0	9.30122e-012	1.94289e-014	6.48370e-014
CPU Time	0.137524 seconds	0.147500 seconds	0.148133 seconds

The same example for Burgers equation is also solved using the Bivariate Spectral Quasilinearization Method. The infinity norm error is given in Table 5.4 with corresponding CPU time above. To obtain Table 5.4 an equal number of collocation points in t and y-variables were used which equals to ten. For the BSQLM, the CPU time is less than a second. As N_y increases the time taken also increases. Few grid points were used but better results were obtained than with the other methods. The infinity norm error decreases as N_y increases, which implies the convergence. This is actually what was also observed by Motsa et al. [57].

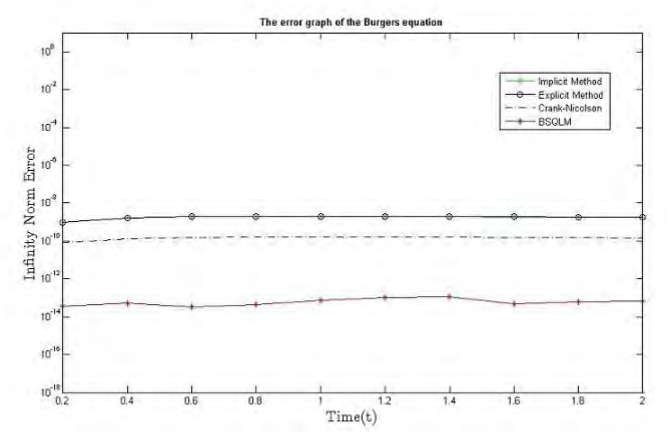

Figure 5.1: Infinity norm error for Burgers equation problem at $t = 2$ for SQLM and BQSLM.

Figure 5.1 shows the comparison of the infinity error norms of the ISQLM, ESQLM, CN-SQLM and the BSQLM. The graphs are plotted in the same set of axes between $t \in [0.2, 2]$. ISQLM and ESQLM on the graph do not show much difference between them. CN-SQLM and BSQLM can be seen to give a small infinity norm error with the smallest in BSQLM.

In this example it is observed that the accuracy for both methods, the SQLM and BSQLM increases as the number of collocation N_y increases. In Tables 5.1, 5.2 and 5.4 as the number of collocation points increases from $N_y = 6, 8$ to $N_y = 10$, the infinity error norm diminishes while keeping N_t at 10 for BSQLM and $N_t = 10001$ for ISQLM and ESQLM. Although the CN-SQLM uses more collocation which is $N_y = 100$ in the y-variable, from the results it is clear that it is much better than ISQLM and ESQLM as expected from the theory. The results in Tables 5.1 - 5.4 indicate that BSQLM is much superior to SQLM in solving the Burgers equation. Figure 5.1 is also in agreement with the tables. During the computation, it is observed that explicit and implicit methods work perfectly for large N_t while BSQLM gives more accurate results with small N_y and N_t. It was observed that the BSQLM gives a highly accurate solution with infinity norm error of up to 10^{-14}, CN-SQLM with an infinity norm error of up to 10^{-11} while the ESQLM and ISQLM have errors of 10^{-9}. Figure 5.1 is in agreement with the observation from the three tables. For both methods SQLM and BSQLM, the infinity norm error presented in tables are consistent with the graph.

5.2.2 Fisher's Equation

This subsection will consider a nonlinear reaction-diffusion Fisher's equation. The equation given by

$$\frac{\partial u}{\partial t} = v\frac{\partial^2 u}{\partial y^2} + \rho f(u), \; y \; \in (0,l), \; t \geq 0, \tag{5.10}$$

where t is the time, y is the spatial coordinate, v is the diffusion coefficient, ρ the reaction factor and $f(u)$ the nonlinear reaction term [19]. The initial and boundary conditions for Fisher's Equation defined in equation (5.10) are as follows:

$$u(y,0) = u_0(y), 0 \leq y \leq l, \tag{5.11}$$

$$u(0,t) = g_0(y), \quad u(l,t) = g_1(y), \tag{5.12}$$

.

Equation (5.10) with $f(u) = u(1-u)$, $v = 1$ becomes

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial y^2} + u(1-u), \; y \; \in (0,2), \; t \geq 0, \tag{5.13}$$

subject to initial condition

$$u(y,0) = \frac{1}{4}\left\{-1 + \tanh\left(\frac{y}{2\sqrt{6}}\right)\right\}^2. \tag{5.14}$$

It has exact solution

$$u(y,t) = \frac{1}{4}\left\{-1 + \tanh\left(\frac{1}{12}\left(-5t + \sqrt{6}y\right)\right)\right\}^2 \tag{5.15}$$

[49, 70]. The boundary conditions are obtained from the exact solution by first setting $y = 0$ for the left boundary condition and $y = 2$ for the right boundary condition. Equation (5.13) is transformed using linear transformation equation $y = l(x+1)/2$, in the examples, using $l = 2$, to convert $y \in [0, 2]$ to $x \in [-1, 1]$. While still discussing transformation for spectral methods, it is important to note that on the time derivatives the finite difference for SQLM will be applied while BSQLM employs spectral method on both variables. To use the bivariate spectral method both variables are transformed to $[-1, 1]$ using the linear transformation equation discussed above. The nonlinear part $-u^2$ is linearised to get

$$N \approx 2u_i u_{i+1} + u_i' u_{i+1} - u_i.$$

The infinity norm errors are presented in Tables 5.5 - 5.8 for both methods and their corresponding CPU times. Figure 5.2 shows the infinity error norms for both methods, the SQLM and BSQLM. The following results were obtained:

Table 5.5: Infinity Norm Errors for ISQLM in solving Fisher's Equation using $N_t = 10001$.

$t \backslash N_y$	6	8	10
0.2	8.52360e-002	3.56192e-002	1.32149e-006
0.4	9.23712e-002	3.90970e-002	2.83271e-006
0.6	9.80688e-002	4.20325e-002	4.43236e-006
0.8	1.02033e-001	4.42663e-002	6.07514e-006
1.0	1.04091e-001	4.56870e-002	7.68970e-006
1.2	1.04206e-001	4.62422e-002	9.19009e-006
1.4	1.02473e-001	4.59410e-002	1.04907e-005
1.6	2.19475e-001	4.48483e-002	1.15204e-005
1.8	9.43538e-002	4.30724e-002	1.22325e-005
2.0	8.85701e-002	4.07490e-002	1.26094e-005
CPU Time	3.516405 seconds	3.699973 seconds	4.058191 seconds

Table 5.6: Infinity Norm Errors for ESQLM solving Fisher's Equation using $N_t = 10001$

$t\backslash N_y$	6	8	10
0.2	8.52349e-002	3.56157e-002	5.37636e-006
0.4	9.23697e-002	3.90915e-002	9.55656e-006
0.6	9.80672e-002	4.20262e-002	1.23969e-005
0.8	1.02032e-001	4.42601e-002	1.40921e-005
1.0	1.04090e-001	4.56816e-002	1.48251e-005
1.2	1.04206e-001	4.62382e-002	1.47757e-005
1.4	1.02473e-001	4.59386e-002	1.41247e-005
1.6	9.90956e-002	4.48477e-002	1.30490e-005
1.8	9.43544e-002	4.30734e-002	1.17124e-005
2.0	8.85710e-002	4.07515e-002	1.02562e-005
CPU Time	0.822218 seconds	0.856920 seconds	0.886251 seconds

Table 5.5 and Table 5.6 show the infinity error norms between exact and approximate solutions for Fisher's equation using ISQLM and ESQLM. The results were obtained using $t \in [0, 2]$ in the time variable and $y \in [0, 2]$ in the space variable, and printed between $t \in [0.2, 2]$. The collocation points that were used in both methods are $N_t = 10001$ and $N_y = 6, 8, 10$ in time and space respectively. The CPU time increased up to 4.058191 for ISQLM while in ESQLM, increased to 0.886251, a value lower than the first one. It is clear that the time increases with the increase in the collocation points in the y-variable in both tables. The ESQLM is fast but it gives poor results or large infinity error norms compared to ISQLM. In both methods ESQLM and ISQLM, the infinity norm error decreases as N_y increases.

Table 5.7: Infinity Norm Errors for CN-SQLM in solving Fisher's Equation using $N_t = 200$.

$t\backslash N_y$	60	80	100
0.2	8.52367e-002	3.56208e-002	1.07549e-007
0.4	9.23723e-002	3.91000e-002	1.79777e-007
0.6	9.80704e-002	4.20369e-002	2.21720e-007
0.8	1.02035e-001	4.42722e-002	2.45049e-007
1.0	1.04093e-001	4.56942e-002	2.59007e-007
1.2	1.04209e-001	4.62506e-002	2.69868e-007
1.4	1.02476e-001	4.59504e-002	2.80913e-007
1.6	9.90986e-002	4.48586e-002	2.92872e-007
1.8	9.43570e-002	4.30832e-002	3.04723e-007
2.0	8.85733e-002	4.07600e-002	3.14621e-007
CPU Time	0.699829 seconds	1.015784 seconds	1.721804 seconds

Table 5.7 shows the infinity norm errors for the Fisher's equation which was solved using CN-SQLM. The results were obtained using the same t and y-variables that were used to obtain Table 5.5 and Table 5.6 which are $t \in [0, 2]$ and $y \in [0, 2]$ respectively, and printed at $t \in [0.2, 2]$. For this example the collocation points used for CN-SQLM are $N_y = 60, 80$ and 100 for the space variable and $N_t = 200$ for the time variable. The increase of N_t brought some changes in the results. The CPU time increased as N_y increased. An improvement of the results when N_y was increased from 80 to 100 (see Table 5.7) was observed.

Table 5.8: Infinity Norm Errors for BSQLM in solving Fisher's Equation using $N_t = 10$.

$t\backslash N_y$	6	8	10
0.2	4.72638e-009	7.31253e-010	7.20778e-010
0.4	6.03092e-009	3.49795e-009	3.52016e-009
0.6	8.00753e-009	8.81872e-010	9.25764e-010
0.8	1.06166e-008	4.49431e-010	4.24011e-010
1.0	1.11248e-008	2.00848e-009	2.02626e-009
1.2	1.08120e-008	9.62241e-010	9.67102e-010
1.4	7.29772e-009	5.76628e-010	5.76644e-010
1.6	3.49390e-009	9.45206e-010	9.24232e-010
1.8	2.84109e-009	5.42070e-010	5.76087e-010
2.0	4.11841e-009	9.46436e-011	9.62009e-011
CPU Time	0.000472 seconds	0.000464 seconds	0.001619 seconds

Equation (5.13) was also solved using the Bivariate Spectral Quasilinearisation Method. The infinity norm error is given in Table 5.8 which was obtained using an equal number of collocation points in t and y-variables which equals to ten. For this method, the CPU time is far less than a second even considering the fact that the collocation points were varied. As the number of collocation points increased from $N_y = 6$ to $N_y = 8$, the CPU time decreased, and then when $N_y = 10$ was used, the CPU time increased. The infinity norm error decreased as the number of collocation points increased more significantly compared to SQLM.

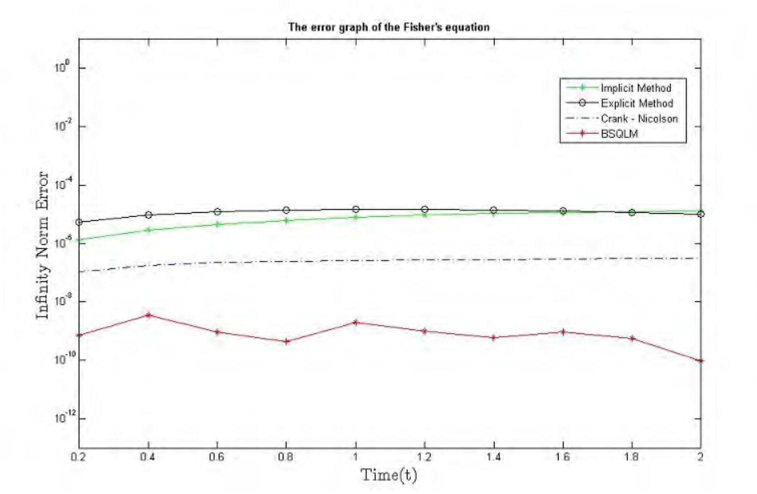

Figure 5.2: Infinity norm error for Fisher's equation problem at $t = 2$ for SQLM and BQSLM.

Figure 5.2 shows the comparison of the infinity error norms of the ISQLM, ESQLM, CN-SQLM and the BSQLM. The graphs are plotted in the same set of axes $\forall$ t $\in [0.2, 2]$. Initially ESQLM gives large infinity norm errors compared to ISQLM from $t = 0$ to $t < 1.4$, but after that ESQLM matches with ISQLM slightly from $t \geq 1.6$. CN-SQLM on the graph appears to be better than ESQLM and ISQLM. Lastly, the BSQLM gives the smallest infinity norm errors out of all the methods.

The Fisher's equation was solved in this subsection and the infinity norm errors are shown in Tables 5.5-5.7 for ISQLM, ESQLM and CN-SQLM. The BSQLM is also used to solve the Fisher's equation and the infinity norm errors are shown in Table 5.8. The results are calculated using the following collocation points: $N_y = 100$ and $N_t = 200$ for CN-SQLM, $N_t = 10001$ and $N_y = 10$ for ISQLM and ESQLM and $N_y = N_t = 10$ for BSQLM. When ISQLM was used, the infinity norm errors show an increase as the collocation points N_y increase giving an infinity norm error of up to 10^{-6}. When ESQLM was used, the results show similar behaviour, but the ESQLM gives large infinity error norms compared to the ISQLM. Initially $N_x = 6$ were used as the collocation points for ISQLM and ESQLM, and the results clearly showed that ISQLM is better than ESQLM. As collocation points increase to $N_y = 8$ and 10 there is much improvement on the ESQLM. Figure 5.2 is also in agreement with what is observed in Table 5.5 and Table 5.6. Although the ISQLM seems to be better when comparing the time taken to execute the code, the ESQLM is much faster since it took less than a second to give all the results. Turning to the results of the CN-SQLM, where the same example was solved but using different collocation points, namely $N_y = 100$ and $N_t = 200$. In the results shown in Table 5.7, as the collocation points increase from $N_y = 60, 80$ and $N_y = 100$, the results get better with an infinity error of up to 10^{-7} taking approximately 1.7 seconds to execute the code. Moving to the second method for our study in solving the Fisher's equation, the results obtained here seem to be more accurate and take less than a second. Even Figure 5.2 is in agreement with our observations. Comparing the two methods, the BSQLM uses fewer collocation points. In the results of SQLM an infinity error of up to 10^{-9} is not observed for all collocation points used, but for BSQLM that error is obtained using collocation points $N_t = 10$ in the time direction and $N_y = 6$ in the space direction. From the Tables 5.5-5.7 it is clear that in the SQLM, the CN-SQLM gives the better results with an infinity norm error of up to 10^{-7}. Comparing the CN-SQLM with the BSQLM, it is clear that BSQLM produces much better results with an infinity norm errors of up to 10^{-11}. The infinity norm errors for both methods are plotted and shown in Figure 5.2 which agrees with the results in Tables 5.5 - 5.8. Considering the CPU time for the Fisher's equation example, the BSQLM takes much less time than SQLM. The BSQLM gives more accurate results compared to SQLM in solving the Fisher's equation. The results were obtained $\forall$t $\in [0.2, 2]$ in the time variable and $y \in [0, 2]$ in the space variable.

5.2.3 The Burgers-Fisher Equation

The Burgers-Fisher equations occur in various areas of science and physical applications, such as modeling of gas dynamics, financial mathematics, population dynamics and others [38]. The boundary conditions are taken from the exact solution. The standard form of Burgers-Fisher equation can be written as:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial y^2} - \alpha u^\gamma \frac{\partial u}{\partial y} - \beta u(u^\gamma - 1), \ y \ \in (0, l), \ t \geq 0, \tag{5.16}$$

with initial condition

$$u(y, 0) = \left\{ \frac{1}{2} - \frac{1}{2} \tanh \left(\frac{\alpha\gamma}{2(1+\gamma)} x \right) \right\}^{1/\gamma}, \tag{5.17}$$

and the exact solution given as:

$$u(y, t) = \left\{ \frac{1}{2} - \frac{1}{2} \tanh \left(\frac{\alpha\gamma}{2(1+\gamma)} \left[x - \left(\frac{\alpha^2 + \beta(1+\gamma)^2}{\alpha(1+\gamma)} \right) t \right] \right) \right\}^{1/\gamma}. \tag{5.18}$$

Consider the Burgers-Fisher equation [70]

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial y^2} - u \frac{\partial u}{\partial y} - u(u - 1), \ 0 \leq y \leq 2, \ t \geq 0, \tag{5.19}$$

with initial condition

$$u(y,0) = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{y}{4}\right), \tag{5.20}$$

and the exact solution is

$$u(y,t) = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{y}{4} + \frac{5t}{8}\right). \tag{5.21}$$

From the standard Burgers-Fisher equation in equation (5.16), it is noticeable that $\alpha = \beta = \gamma = 1$. The numerical solutions for equation (5.19) are shown in Tables 5.9 - 5.12. Figure 5.3 shows the corresponding infinity norm error graph in which both methods are plotted on the same set of axes.

Table 5.9: Infinity Norm Errors for ISQLM solving Burgers-Fisher Equation using $N_t = 10001$

$t \backslash N_y$	6	8	10
0.2	1.32713e-001	1.67515e-001	1.78506e-001
0.4	1.23191e-001	1.25183e-001	1.08292e-001
0.6	1.12158e-001	9.27447e-002	6.41620e-002
0.8	1.00902e-001	7.05505e-002	3.71844e-002
1.0	8.94654e-002	5.52157e-002	2.10432e-002
1.2	7.80244e-002	4.41585e-002	1.16256e-002
1.4	6.69131e-002	3.57767e-002	6.27567e-003
1.6	5.64822e-002	2.91435e-002	3.31556e-003
1.8	4.70036e-002	2.37408e-002	1.71770e-003
2.0	3.86355e-002	1.92773e-002	8.74278e-004
CPU Time	4.151901 seconds	4.688118 seconds	5.216620 seconds

Table 5.10: Infinity Norm Errors for ESQLM in solving Burgers-Fisher Equation using $N_t = 10001$.

$t\backslash N_y$	6	8	10
0.2	1.32719e-001	1.67547e-001	1.78532e-001
0.4	1.23192e-001	1.25180e-001	1.08272e-001
0.6	1.12155e-001	9.27293e-002	6.41361e-002
0.8	1.00900e-001	7.05373e-002	3.71637e-002
1.0	8.94646e-002	5.52083e-002	2.10311e-002
1.2	7.80248e-002	4.41570e-002	1.16218e-002
1.4	6.69144e-002	3.57796e-002	6.27830e-003
1.6	5.64839e-002	2.91492e-002	3.32239e-003
1.8	4.70055e-002	2.37479e-002	1.72675e-003
2.0	3.86374e-002	1.92846e-002	8.84053e-004
CPU Time	2.276083 seconds	2.593988 seconds	2.687586 seconds

Table 5.9 and Table 5.10 show the infinity norm errors for Burger-Fisher equation which was solved using ISQLM and ESQLM. The results were obtained using $t \in [0, 2]$ in the time variable and $y \in [0, 2]$ in the space variable, and printed at $t \in [0.2, 2]$. The collocation points that were used in both methods are $N_t = 10001$ and $N_y = 6, 8, 10$ in time and space respectively. The CPU time increases up to 5.216620 for Implicit SQLM while in Explicit SQLM it goes up to 2.687586. It clear from the tables that time increases with the increase in the collocation points in the y-variable in both tables. It can also be observed that as $t \to 2$ the infinity norm errors decreases.

Table 5.11: Infinity Norm Errors for CN-SQLM in solving Burgers-Fisher Equation using $N_t = 100$.

$t \backslash N_y$	60	80	100
0.2	1.32785e-001	1.68664e-001	1.11562e-001
0.4	1.23436e-001	1.26388e-001	1.10165e-001
0.6	1.12322e-001	3.32207e-002	6.52414e-002
0.8	1.00997e-001	9.35036e-002	3.77973e-002
1.0	8.95187e-002	5.54602e-002	2.13862e-002
1.2	3.18125e-002	4.42936e-002	3.44256e-002
1.4	6.69295e-002	3.58514e-002	6.38106e-003
1.6	5.64914e-002	2.91856e-002	3.37492e-003
1.8	4.70091e-002	2.37655e-002	1.75237e-003
2.0	3.86390e-002	1.92925e-002	8.95679e-004
CPU Time	1.410424 seconds	2.090944 seconds	2.417226 seconds

Table 5.11 shows the infinity error norms between exact and approximate solution. The Burger-Fisher equation was solved using the Crank-Nicolson Spectral Quasilinearisation Method. The results were obtained using the same t and y-variables that were used in Table 5.5 and Table 5.6 above which are $t \in [0, 2]$ and $y \in [0, 2]$ respectively, and printed at time $t \in [0.2, 2]$. The collocation points used for Crank-Nicolson are $N_y = 60, 80$ and 100 for the space variable and $N_t = 100$ for the time variable. The infinity norm error decreases as $t \to 2$ and CPU increases as N_y increases.

Table 5.12: Infinity Norm Errors for BSQLM in solving Burgers-Fisher Equation using $N_t = 10$.

$t\backslash N_y$	6	8	10
0.2	1.87740e-008	1.24961e-008	1.26302e-008
0.4	4.56477e-008	4.08365e-008	4.07989e-008
0.6	2.73151e-008	1.72630e-008	1.72630e-008
0.8	1.87368e-008	1.87368e-008	1.87368e-008
1.0	2.06177e-008	2.47657e-008	2.47913e-008
1.2	1.64798e-008	1.52373e-008	1.52373e-008
1.4	1.61543e-008	1.13923e-008	1.13923e-008
1.6	1.20451e-008	1.10165e-008	1.10224e-008
1.8	5.80282e-009	5.68583e-009	5.68959e-009
2.0	1.34142e-009	5.09331e-010	5.11942e-010
CPU Time	0.007001 seconds	0.033598 seconds	0.013124 seconds

The Bivariate Spectral Quasilinearisation Method has been used to solve the Burgers-Fisher equation. The infinity norm error given in Table 5.12 was obtained using 10 collocation points in both the t and the y-variables. The CPU time is also given. For the BSQLM, the CPU time is less than a second even if the collocation points were varied. Initially, when 6 collocation points were used in thee space variable the CPU time was the smallest with a value of 0.007001 seconds. As the number of collocation points increases from 6 to 8 the CPU time initially increases then when $N_y = 10$ was used the CPU time decreases to 0.013124.

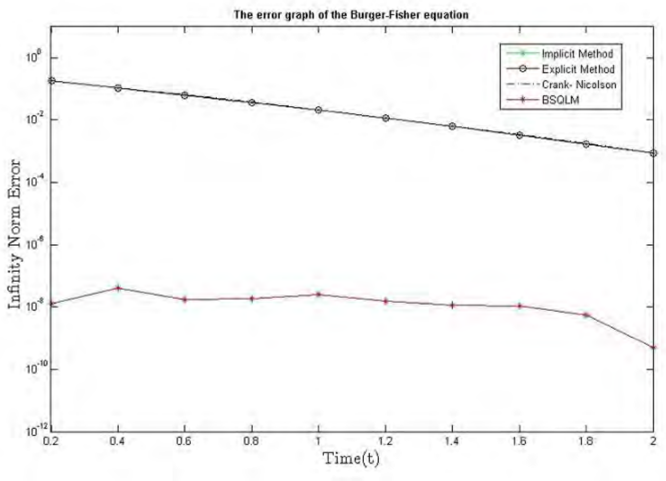

Figure 5.3: Infinity error norm for Burgers-Fisher equation problem at $t = 2$ for SQLM and BQSLM.

Figure 5.3 shows the comparison of the infinity norm errors of the ISQLM, ESQLM, CN-SQLM and the BSQLM. The graphs are plotted in the same set of axes at time $t \in [0.2, 2]$. ISQLM, ESQLM and CN-SQLM on the graph does not show any difference between them. BSQLM can be seen to give a small infinity error norm compared to SQLM.

In this subsection, the Burgers-Fisher equation has been solved using SQLM and BSQLM. Using the same parameters, explicit and implicit methods were implemented (i.e. $N_t = 10001$, $N_y = 10$) , $N_y = N_t = 100$ for Crank-Nicolson and $N_y = N_t = 10$ for BSQLM were used. To solve the Burgers-Fisher equation the following parameters were used: $\alpha = 1, \beta = 1, \gamma = 1$. The infinity norm error for SQLM are shown in Tables 5.9 - 5.11. Correspondingly Table 5.12 shows the infinity norm errors for the Burgers-Fisher equation using the BSQLM. The infinity error norms are also plotted on the same set of axes for both methods and displayed in Figure 5.3. As the number of collocations N_y increases, the infinity norm error decreases. The

CPU time increases as the number of collocation points increases and as $t \to 2$, the infinity norm error decreases.

In the results for SQLM, using an explicit method, implicit method and Crank-Nicolson and the BSQLM, it is observed that the infinity norm errors improve as $t \to 2$. Secondly, it is noted that as the number of collocation points increase, the infinity norm errors decrease significantly. Comparing both methods, it is clear that the BSQLM gives much better results even taking into consideration that different collocation points have been employed. The infinity norm errors in general for this example are poor and give only up to 10^{-4} for SQLM and are better for BSQLM since it increases up to 10^{-10}. Having solved this equation using both the SQLM and BSQLM, it is clear from the results that BSQLM gives far better results than the SQLM. The infinity norm error graph is shown in Figure 5.3 which also agrees that the BSQLM is a better method than SQLM. In the same figure, the SQLM infinity norm error seem to match each other and with BSQLM after $t = 1.8$, the infinity norm error declines significantly.

5.2.4 Newell-Whitehead-Segel and Zeldovich Equations

In this subsection the Newell-Whitehead-Segel and Zeldovich equations will be discussed. Newell-Whitehead-Segel (NWS) equation is an important nonlinear reaction-diffusion equation and usually is used to model the transmission of nerve impulse, also used in circuit theory, biology and the area of population genetics as mathematical models while the Zeldovich has been reported to arise in combustion theory [45]. These equations arise if the Fitzhugh-Nagumo (FN) equation is reduced. The Fitzhugh-Nagumo (FN) equation takes this form:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial y^2} - u(1-u)(a-u). \tag{5.22}$$

If $a = -1$ in equation (5.22), then the NWS is formed to be $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial y^2} + u(1-u^2)$ and if $a = 0$ the Zeldovich equation is formed. The Zeldovich equation is represented as

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial y^2} + u^2 - u^3. \tag{5.23}$$

The Newell-Whitehead-Segel (NWS) equation is a well known global equation to govern evolution of nearly one-dimensional (1D) nonlinear patterns produced by a finite-wavelength instability in isotropic two-dimensional media, a classical example being the Rayleigh-Bernard convection [17]. The Rayleigh-Bernard convection according to Zahra et al. [68] is a natural convection, that occurs in a horizontal plane layer of fluids from below, in which the fluid develops a regular pattern of convection cells called Bernard cells. When heating is sufficiently intensive, the convective motion of the fluid is developed spontaneously, the hot fluid moves upward, and the cold fluid moves downward. Figure 5.4 and Figure 5.5 show the Rayleigh-Bernard convection phenomenon and the convection cells in a gravity field respectively.

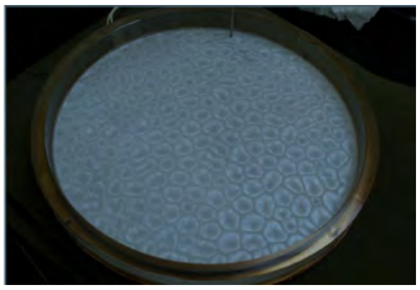

Figure 5.4: Rayleigh-Bernard convection phenomenon.

by John Matsson [7]

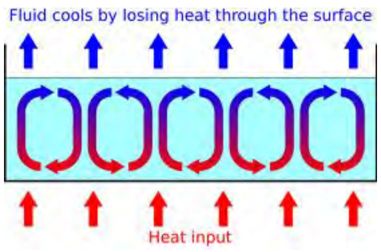

Figure 5.5: Convection cells in a gravity field.

by Zahra et al. [68]

The Newell-Whitehead-Segel equation has been given considerable attention in recent years and various methods and techniques have been introduced to solve it. These methods includes Reduced Differential Transform Method (RDTM), and the Adomian Decomposition Method [17, 63]. The Newell-Whitehead-Segel equation is used to model the interaction of the effect of the diffusion term with the nonlinear effects of the reaction term and has been applied in nonlinear optics, biological systems as well as in chemical reaction [50].

This section is concerned by the approximate solutions of the Newell-Whitehead-Segel equation of the form:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial y^2} + u - u^4 \tag{5.24}$$

with the exact solution given by

$$u(y,t) = \left\{ \frac{1}{2} + \frac{1}{2} \tanh\left(-\frac{3}{2\sqrt{10}} \left(y - \frac{7t}{\sqrt{10}} \right) \right) \right\}^{2/3}. \tag{5.25}$$

The boundary conditions are taken from the exact solution [33]. Using SQLM and BSQLM and by implementing them in Matlab, the following results were obtained:

Table 5.13: Infinity Norm Errors for ISQLM solving Newell-Whitehead-Segel Equation using $N_t = 10001$

$t\backslash N_y$	6	8	10
0.2	1.73485e-001	7.86088e-002	2.01504e-006
0.4	1.59528e-001	7.50749e-002	6.53105e-007
0.6	1.36904e-001	6.66936e-002	2.82604e-006
0.8	1.10308e-001	5.53293e-002	6.49133e-006
1.0	8.42379e-002	4.32396e-002	8.81558e-006
1.2	6.16321e-002	3.21885e-002	9.31855e-006
1.4	4.36523e-002	2.30845e-002	8.41785e-006
1.6	3.01961e-002	1.61086e-002	6.84161e-006
1.8	2.05429e-002	6.36882e-002	5.17549e-006
2.0	1.38161e-002	7.44483e-003	3.73055e-006
CPU Time	1.174178 seconds	1.124924 seconds	2.688793 seconds

Table 5.14: Infinity Norm Errors for ESQLM in solving Newell-Whitehead-Segel Equation using $N_t = 10001$.

$t \backslash N_y$	6	8	10
0.2	1.73486e-001	7.86083e-002	2.01458e-006
0.4	1.59530e-001	7.50775e-002	6.50252e-007
0.6	1.36908e-001	6.67015e-002	2.83193e-006
0.8	1.10312e-001	5.53419e-002	6.49909e-006
1.0	8.42425e-002	4.32547e-002	8.82338e-006
1.2	6.16363e-002	3.22035e-002	9.32473e-006
1.4	4.36559e-002	2.30976e-002	8.42164e-006
1.6	3.01989e-002	1.61191e-002	6.84328e-006
1.8	2.05450e-002	1.10326e-002	5.17579e-006
2.0	1.38176e-002	7.45046e-003	3.73018e-006
CPU Time	2.062504 seconds	2.293936 seconds	2.439014 seconds

Table 5.13 and Table 5.14 shows the infinity norm error for the Newell-Whitehead-Segel Equation. The ISQLM and ESQLM were used to solve this equation. For both tables $N_t = 10001$ and $N_y = 6, 8, 10$ were used for the time and space variables respectively. From Table 5.13 and Table 5.14, it can be observed that the infinity norm error decreases as the number of collocation points N_y increases and the CPU time increases. In terms of the CPU time, the Explicit SQLM takes fewer seconds than Implicit SQLM.

Table 5.15: Infinity Norm Errors for CN-SQLM in solving Newell-Whitehead-Segel Equation using $N_t = 100$.

$t\backslash N_y$	60	80	100
0.2	1.73486e-001	7.86103e-002	1.11870e-006
0.4	1.59530e-001	7.50804e-002	3.41998e-006
0.6	1.36908e-001	6.67041e-002	6.42435e-006
0.8	1.10313e-001	5.53437e-002	9.03575e-006
1.0	8.42431e-002	4.32556e-002	1.02231e-005
1.2	6.16365e-002	3.22037e-002	9.76489e-006
1.4	4.36557e-002	2.30973e-002	8.19722e-006
1.6	3.01987e-002	1.61185e-002	6.26248e-006
1.8	2.05448e-002	1.10321e-002	4.48419e-006
2.0	1.38174e-002	7.44995e-003	3.08003e-006
CPU Time	1.626536 seconds	2.181471 seconds	2.998783 seconds

Table 5.15 represent the infinity norm error table for the Newell-Whitehead-Segel solved by CN-SQLM using $N_y = N_t = 100$ and $N_y = 200$ as our collocation points. Different collocation points have been used for N_y varied between $60, 80$ and 100. It can clearly be seen from Table 5.15 that as N_y increases, the infinity norm error decreases and CPU time increases as the number of collocation points increases.

Table 5.16: Infinity Norm Errors for BSQLM in solving Newell-Whitehead-Segel Equation using $N_t = 10$.

$t \backslash N_y$	6	8	10
0.2	8.71737e-007	7.04671e-007	7.31647e-007
0.4	1.85014e-006	2.27974e-006	2.35822e-006
0.6	1.31310e-006	1.41643e-006	1.36411e-006
0.8	1.14975e-006	1.14975e-006	1.14975e-006
1.0	1.47879e-006	1.37367e-006	1.31643e-006
1.2	1.52254e-006	1.31428e-006	1.37807e-006
1.4	3.50655e-007	3.50655e-007	3.50655e-007
1.6	6.49222e-007	6.48695e-007	6.82745e-007
1.8	6.36843e-007	5.82270e-007	6.45927e-007
2.0	6.70400e-008	5.93894e-008	5.54443e-008
CPU Time	0.029039 seconds	0.030137 seconds	0.103207 seconds

The infinity norm error results when using the Bivariate Spectral Quasilinearisation Method for the Newell-Whitehead-Segel equation are given in Table 5.16 which was obtained using an equal number of collocation points in t and y-variables, equal to 10. The CPU time is also given. For the BSQLM, the CPU time is far less than a second even if the collocation points were varied. As the number of collocation points increases the CPU time initially decreases then when $N_y = 10$ is used the time increases.

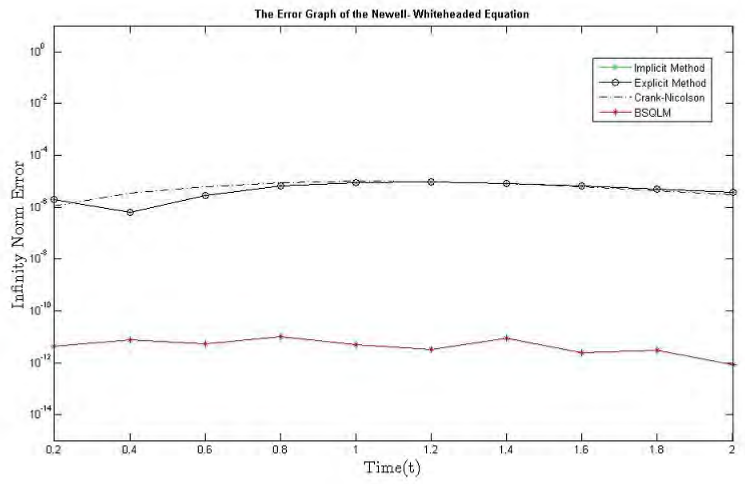

Figure 5.6: Infinity norm error for Newell-Whitehead-Segel equation problem at $t = 2$ for SQLM and BQSLM.

The infinity norm error graph is shown in Figure 5.6. It shows the comparison of the two methods the SQLM and BSQLM infinity norm error results plotted on the same set of axes, at time $t \in [0.2, 2]$. When $t = 0.4$ the ESQLM and ISQLM gives the minimal infinity norm error compared to CN-SQLM. But as t increases, the observation changes. For ESQLM and ISQLM from $t > 0.3$, these methods are better than CN-SQLM until $t \leq 1$ (see Figure 5.6). In the same figure, from $t \geq 1.4$ to $t = 2$, the infinity norm error of the CN-SQLM decreases. The BSQLM on the graph appeared to give an infinity norm error better than the SQLM.

The second part of this subsection considers the Zeldovich equation given by the following PDE:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial y^2} + u^2(1-u) \tag{5.26}$$

with the exact solution given by:

$$u(y,t) = \left\{ \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{1}{2\sqrt{2}}\left(y + \frac{t}{\sqrt{2}}\right)\right)\right\}, \tag{5.27}$$

with the following initial condition

$$u(y,0) = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{y}{2\sqrt{2}}\right), \tag{5.28}$$

cited in [66]. Solving this equation gives the results presented and discussed below:

Table 5.17: Infinity Norm Errors for ISQLM solving Zeldovich Equation using $N_t = 10001$

$t\backslash N_y$	6	8	10
0.2	1.42962e-001	6.23347e-002	3.44329e-007
0.4	1.39253e-001	6.04930e-002	5.88798e-007
0.6	1.35032e-001	5.84444e-002	7.53944e-007
0.8	1.30368e-001	5.62222e-002	8.63953e-007
1.0	1.25336e-001	5.38606e-002	9.34596e-007
1.2	1.20009e-001	5.13935e-002	9.76287e-007
1.4	1.14462e-001	4.88536e-002	9.96109e-007
1.6	1.08766e-001	4.62720e-002	9.99069e-007
1.8	1.02987e-001	4.10950e-002	9.88877e-007
2.0	1.71887e-002	3.85481e-002	9.68402e-007
CPU Time	3.252149 seconds	3.875101 seconds	4.295436 seconds

Table 5.18: Infinity Norm Errors for ESQLM in solving Zeldovich Equation using $N_t = 10001$.

$t\backslash N_y$	6	8	10
0.2	1.42962e-001	6.04925e-002	3.44541e-007
0.4	1.39253e-001	5.84435e-002	5.89105e-007
0.6	1.35031e-001	5.62210e-002	7.54275e-007
0.8	1.30368e-001	5.38592e-002	8.64274e-007
1.0	1.25336e-001	5.13919e-002	9.34893e-007
1.2	1.20009e-001	4.88520e-002	9.76555e-007
1.4	1.14462e-001	4.62703e-002	9.96346e-007
1.6	1.08765e-001	4.36754e-002	9.99277e-007
1.8	1.02987e-001	4.10932e-002	9.89056e-007
2.0	9.71881e-002	3.85464e-002	9.68555e-007
CPU Time	1.073123 seconds	1.140519 seconds	1.223779 seconds

Table 5.17 and Table 5.18 show the infinity norm errors when the Zeldovich equation was solved using ISQLM and ESQLM. The results were obtained using $t \in [0, 2]$ in the time variable and $y \in [0, 2]$ in the space variable, and printed at time $t \in [0.2, 2]$. The collocation points that were used in both methods are $N_t = 10001$ and $N_y = 6, 8, 10$ in time and space respectively. The CPU time increases to 4.295436 seconds for ISQLM while in ESQLM it goes up to 1.223779 seconds. It is noticeable that the time increases with the increase in the collocation points in the x-variable in both tables. The ESQLM and ISQLM gave results which are similar when rounded off to two significant numbers at $N_y = 10$.

Table 5.19: Infinity Norm Errors for CN-SQLM in solving Zeldovich Equation using $N_t = 200$.

$t\backslash N_y$	60	80	100
0.2	1.42962e-001	6.04927e-002	4.17104e-008
0.4	1.39253e-001	5.84439e-002	6.95383e-008
0.6	1.35032e-001	5.62215e-002	8.66680e-008
0.8	1.30368e-001	5.38598e-002	9.65290e-008
1.0	1.25336e-001	5.13926e-002	1.01375e-007
1.2	1.20009e-001	4.88527e-002	1.02707e-007
1.4	1.14462e-001	4.62711e-002	1.01550e-007
1.6	1.08766e-001	4.36762e-002	9.86311e-008
1.8	1.02987e-001	4.10940e-002	9.44817e-008
2.0	9.71884e-002	3.85471e-002	8.95037e-008
CPU Time	1.969425 seconds	2.138365 seconds	2.665391 seconds

Table 5.19 above shows the infinity norm errors. The Zeldovich equation was solved using Crank-Nicolson Spectral Quasilinearisation. The collocation points used for CN-SQLM are $N_y = 60, 80$ and 100 for the space variable and $N_t = 200$ for the time variable. The infinity norm error decreases as $t \to 2$ and CPU time increases as N_y increases.

Table 5.20: Infinity Norm Errors for BSQLM in solving Zeldovich Equation using $N_t = 10$.

$t \backslash N_y$	6	8	10
0.2	7.683e-008	1.040e-009	1.107e-011
0.4	9.370e-008	1.060e-009	1.091e-011
0.6	1.051e-007	1.104e-009	1.109e-011
0.8	1.049e-007	1.033e-009	6.895e-012
1.0	1.013e-007	9.479e-010	4.840e-012
1.2	1.036e-007	7.950e-010	3.388e-012
1.4	9.908e-008	5.692e-010	2.713e-012
1.6	9.061e-008	3.776e-010	3.166e-012
1.8	7.863e-008	2.352e-010	3.552e-012
2.0	6.487e-008	1.872e-010	3.441e-012
CPU Time	0.000519 seconds	0.000627 seconds	0.000734 seconds

The Bivariate Spectral Quasilinearisation Method was also used to solve the Zeldovich equation. The infinity norm error is given in Table 5.20 and its corresponding CPU time. Table 5.20 was obtained using an equal number of collocation points in t and y-variables, equal to 10. The BSQLM CPU time is far less than a second even if the collocation points were varied. As the number of collocation points increases the CPU time initially decreases but when $N_y = 10$ were used the time increases. It can also be observed that as $t \to 2$, the infinity norm error decreases for all collocation points used, which is in agreement with the observation of Motsa et al. [57] when BQSLM is used.

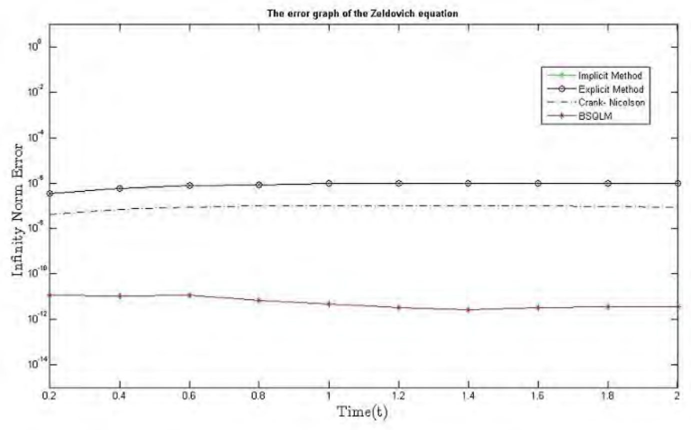

Figure 5.7: Infinity norm error for Zeldovich equation problem at $t = 2$ for SQLM and BQSLM.

Figure 5.7 shows the comparison of the infinity norm error using the ISQLM, EQSLM, CN-SQLM and the BSQLM. The graphs are plotted in the same set of axes at time $t \in [0.2, 2]$. From the figure, the ESQLM and IBSQLM seem to match, the CN-SQLM appeared just below the two. The BSQLM appeared far below the three.

In this subsection, the Newell-Whitehead-Segel and Zeldovich equations were solved using $N_y = N_t = 100$ and $N_y = 200$, $N_y = 100$ for CN-SQLM, and $N_t = 10001$, $N_y = 10$ were used for ESQLM and ISQLM. Lastly $N_y = N_t = 10$ were used for BSQLM. The exact solution from the literature [33] has been used to validate the results for Newell-Whitehead-Segel equation. The infinity norm error results are shown in Tables 5.13 - 5.15 and their CPU time for SQLM. Table 5.16 displays the infinity norm error results for BSQLM.

The SQLM results for Newell-Whitehead-Segel equation were obtained, and it was noted that the Crank-Nicolson SQLM gives smaller infinity norm errors using much larger collocation points in the y-variable. The Explicit and Implicit SQLM uses large number of collocation points in the t- variable which is due to the fact that on time the finite differences are used. The BSQLM is further investigated, using fewer collocation points to $N_y = 10$ in space and $N_t = 10$. The BSQLM gave striking results of up to 10^{-8} while in the SQLM up to 10^{-6} was observed with much higher collocation points. The BSQLM is clearly much more efficient than SQLM.

Lastly the Zeldovich equation was solved, and the infinity norm errors are shown in Tables 5.17- 5.19 for SQLM and Table 5.20 for BSQLM. The maximum errors are improving as the number of collocation points increases. This shows that BSQLM is still superior to SQLM in solving the Zelvovich equation. The infinity norm error graph is shown in Figure 5.7. This figure shows that the BSQLM gives better results than SQLM. In the Zeldovich, the SQLM gives an infinity norm error of up to 10^{-7} while the BSQLM gives an infinity norm error of up to 10^{-12}.

5.2.5 Comparative Discussion

Six examples were solved using SQLM and BSQLM and the results were described in Chapter 5. Numerical simulations were carried out to find approximate numerical solutions of the infinity norm error which is the quantity of interest for this study. In all examples for Explicit and Implicit spectral quasilinearisation method in the time domain $N_t = 10001$ was used. Through all numerical experimentation using the SQLM, this value was found to give accurate results. In the space direction which is a spectral discretisation $N_y = 6, 8, 10$ were used. For Crank-Nicolson SQLM, $N_t = 100$ and 200 were used in the time variable and $N_y = 60, 80, 100$. This was found to give accurate results. When the value of N_y was increased and keeping N_t constant the results did not change to a significant extent. For the BSQLM the number of collocation points used in both time and space directions (t, y) were $N_y = N_t = 10$ in all cases. It was found to give accurate results. It was discovered that both the SQLM and BSQLM algorithms are based on the computation of the value of some quantity, say u_{i+1}^{n+1}, at each time step. This was achieved by iterating using the known value at the previous time step i obtained from initial conditions to find the solution at the next iteration $i + 1$. The solutions were obtained using 10 iterations for both methods. The correctness of the computed SQLM and BSQLM approximate results was confirmed against the results obtained using the exact solution for each example. The equations that were solved were linearised using the QLM since they are nonlinear. In the SQLM the time derivatives were approximated by finite difference method and in space derivative spectral method is applied while in BSQLM both the y-variables and t-variables spectral method was applied. Tables 5.1, 5.2, 5.5, 5.6, 5.9, 5.10, 5.13, 5.14, 5.17, 5.18 show the infinity error norms for SQLM. The tables show results for all examples solved using ISQLM and ESQLM. The tables also give the CPU time for each corresponding table. It was observed that the results were computed using the same number of collocation points in the x and t -variables. The Crank-Nicolson SQLM results are given in Tables 5.3, 5.7, 5.11, 5.15, 5.19. These tables were obtained using an equal number of collocation points in the space direction N_y and in the time variable, $N_t = 100$ for all other examples except for the Fisher's equation and Zeldovich equation. It is noted that for Fisher's and Zeldovich equations an increase in the collocation points in the time variable decreases the infinity norm errors, while in the other examples there is no significant change in the results if N_t is increased. All the examples solved by SQLM were also solved by the BSQLM, the second method employed in this study. Tables 5.4, 5.8, 5.8, 5.16, 5.20 show the infinity error norms. The results in these tables were obtained using ten collocations in both x and t-directions. Corresponding CPU times for each example are also given. All the results have been printed between $0.2 \leq t \leq 2$. It is clear from the comparison of the computational run times (CPU time) that the BSQLM takes less computer time than the SQLM. The results furthermore indicate in both methods, as the number of collocation points increases in the space variable the computational time also increases. It is revealing that from all examples, the BQSLM takes less than a second to execute the code while with the SQLM, the maximum time that was observed is 5.623292 seconds. The actual superiority of the BSQLM in terms of computational proficiency and accuracy when compared to the SQLM may be elucidated by the fact that the SQLM scheme used the finite difference in time derivative which has been proven in literature to be less accurate than the spectral methods. Figures 5.1, 5.2, 5.3, 5.6, 5.7 show the infinity error norms each example solved using the SQLM and BSQLM and plotted on the same set of axes. These figures are in agreement with our observations. Even though different collocation points were used, the BSQLM appeared to give much less infinity error norm, and moreover required fewer collocation points. The apparent computational speed and smaller error showed by the BSQLM over the SQLM prevailed in all the types of the equations which were considered. Hence, the BSQLM is a better method that can be used to obtain numerical solutions of nonlinear partial differential equations than the SQLM.

5.3 Overview

In this chapter the spectral quasilinearisation method (SQLM) and bivariate spectral quasilinearisation method (BSQLM) were applied in solving the nonlinear evolution problem of second order. The following examples were solved: Burgers equation, Burgers-Fisher equation, Fisher's equation, Newell-Whitehead-Segel equation and Zeldovich equation. The results for the approximate solution have been computed. To be able to compare the SQLM and BSQLM, the exact and approximate solutions have been used to find the infinity norm error for each example. Finally, the infinity norm error were plotted on the same set of axes for all examples. In all the examples solved, $y \in [0, 2]$ were used in the space domain and different values of collocation points N_y. The tables show the infinity error norm at $t \in [0.2, 2]$. Figures 5.1, 5.2, 5.3, 5.6, 5.7 show the the infinity error norm at a fixed time $t = 2$ that were used for error analysis. The purpose of this chapter was to do comparisons and investigate the applicability of the SQLM and BSQLM for solving second order evolution problems. The BSQLM gives much smaller infinity norm error than SQLM using fewer collocation points in the space direction and time direction. The numerical results presented in this study clearly prove that the BSQLM is much better than SQLM.

Chapter 6

Conclusion

6.1 Concluding Remarks

This study was conducted to investigate the applicability of two methods, the SQLM and the BSQLM. Both methods that are employed are Chebyshev collocation spectral methods. The main aim was to do a comparative study and apply both methods to the same problems. For this study, nonlinear partial differential equations of the second order were considered. Firstly, the discussion began by studying the background of nonlinear evolution partial differential equations trying to gain insight into what actually made researchers try to find solutions to these equations. The spectral methods were then discussed in detail, investigating the advantages of using them and the reasons that make them the method of choice for some researchers and for this study.

In Chapter 2 our main focus was to explain the numerical properties of nonlinear evolution partial differential equations. Since many researchers had studied the evolution problems using different approaches, traditional methods were visited, namely finite difference methods, finite elements and finite volume [26]. The limitations that brought by these methods were also highlighted. Some methods which are spectral-based were studied and have been used to solve evolution problems (see for example [21], [22]).

Chapter 3 and Chapter 4 discussed the main focuses of the study, the SQLM, and BSQLM respectively. The SQLM was developed by combining the quasilinearisation method, Chebyshev collocation spectral method with Lagrange interpolation and applying the finite difference method to time derivatives. That gave rise to Explicit, Implicit and Crank-Nicolson SQLM, while bivariate spectral quasilinearisation method (BSQLM) was developed using the quasilinearisation together with Chebyshev collocation spectral method with bivariate Lagrange interpolation. The main aim of this study was to conduct a comparative study of these methods in solving the nonlinear partial differential equations, investigating the accuracy, robustness and effectiveness of each method.

In Chapter 5 and Chapter 6 numerical simulations were carried out and the results discussed. The types of problems that were solved are Burgers, Burgers-Fisher, Fisher's, Newell-Whitehead-Segel and Zeldovich equations. Comparing the results obtained using both methods, it is clear that the BSQLM gives accurate results compared to the SQLM. The limitation of the SQLM is that on the time derivative the finite difference method is applied, which requires many collocation points which require more computing power to be used to solve these equations. From the results it was also noted that the BSQLM gives accurate results even with small intervals both in space and time variables, with CPU times which are less than a second, while in SQLM, with some examples like Newell-Whitehead-Segel, results are obtained with CPU time of more than five seconds. It is clear from the results presented in this study that BSQLM works much better that the SQLM. The collocation points were even increased to 200 for the Newell-Whitehead-Segel example and observe that the SQLM results were giving large infinity error norms and BSQLM results continued to give use small infinity error norms.

In thesis, the SQLM and BSQLM was used for second order nonlinear evolution equations with exact solutions. It may be concluded that the BSQLM is a very powerful and efficient technique in finding solutions for wide classes of problems like the ones considered. With regard to the applications, the BSQLM outlined in the previous sections was found to be quite efficient than SQLM achieving less computational time and small infinity norm errors. It is also crucial to note out that the advantage of the BSQLM over the SQLM is that BSQLM does not use finite difference which requires more collocation points. Both methods required linearisation and we used QLM. BSQLM showed to be much better than SQLM in solving the examples that were discussed in this study.

6.2 Further Studies or Research

In this study, our focus was to compare the two methods of solving nonlinear parabolic evolution problems. From the results, it is clear that the BSQLM gives better results than the SQLM. To take this study further it will be useful to investigate coupled systems and to discover whether the findings made here also apply to solving coupled systems. Khater et al. [75] used the spectral collocation methods based on Lagrange polynomials for spatial derivatives to obtain numerical solutions of some coupled equations. The problem is reduced to a system of ODEs that are solved by the fourth order Runge–Kutta method. They tested their method on coupled KdV equations, coupled modified KdV equations, coupled KdV system and Boussinesq system. To take this study further coupled system that fall under same class that was considered in this study will examined. Another researcher that also looked at coupled system is Kaya [74] in solving viscous Burgers equations. He considered a coupled system of viscous Burgers' equations with appropriate initial values using the decomposition method. In his method, the solution is calculated in the form of a convergent power series with easily computable components. The method does not need linearization, weak nonlinearity assumptions or perturbation theory. I also wish to study the effect of linearization in the problems which are nonlinear.

Bibliography

[1] S. Abbasbandy, M.T. Darvishi, “A numerical solution of Burgers’ equation by modified Adomian method”, *Applied Mathematics and Computation*, vol. 163, no. 3, pp. 1265-1272, 2005.

[2] A. Majid, K. Somayeh, “Solution for Partial Differential Equations Involving Logarithmic Nonlinearities”, *Australian Journal of Basic and Applied Sciences*, vol. 5, no. 4, pp. 60-66, 2011.

[3] A.H. Bhrawy, L.M Assas and M.A Alghamdi,“An Efficient Spectral Collocation Algorithm for Nonlinear Phi-four Equations”, *Boundary Value Problem*, vol. 1, no. 87, 2013.

[4] C. Canuto, M.Y. Hussaini, A. Quarteroni , T.A. Zang “Spectral Methods: Fundamentals in Single Domains”, *Springer-Verlag* , 2006.

[5] J. Noye“Computational Techniques for Differential Equations”, *Elsevier*, 2000.

[6] B. Costa, “Spectral Methods for Partial Differential Equations”, *A Mathematical Journal*, vol. 6, no. 4, pp. 1-32, 2004.

[7] J. Matsson, “Oral Roberts University: A Student Project on Rayleigh-Bénard Convection”, *American Society for Engineering Education*, 2008

[8] P. G. Ciarlet, A. Iserles, R. V. Kohn, M. H. Wright,“ A Practical Guide to Pseudospectral Methods”, *Cambridge University Press*, 1996.

[9] M.Bastani and D. K. Salkuyeh,“A Highly Accurate Method to Solve Fisher’s Equation”, *Journal of Physics*, vol. 78, no. 3, pp. 335-346, 2012.

[10] R.C. Mittal and G. Arora, “Efficient Numerical Solution of Fisher’s Equation by Using B-Spline Method”, *International Journal of Computer Mathematics*, vol. 87, no. 13, pp. 3039-3051, 2010.

[11] R. C. Mittal and R. K. Jain, “Numerical Solutions of Nonlinear Fisher’s Reaction–Diffusion Equation with Modified Cubic B-Spline Collocation Method”, *Mathematical Sciences*, vol. 7, no. 12, pp. 1-12, 2013.

[12] A. Verma, R. Jiwari and M. E. Koksal, “Analytic and Numerical Solutions of Nonlinear Diffusion Equations Via Symmetry Reductions”, *Advances in Difference Equation* , vol. 2014, no. 1, pp. 229, 2014.

[13] J. Gazdag and J. Canosa, “Numerical Solution of Fisher’s Equation”, *Journal of Applied Probability*, vol. 11, no. 3, pp. 445-457, 1974.

[14] A. Rashid and M. Abbas, “Numerical Solution of Generalized Burgers-Fisher Equation by Fourier Pseudospectral Method”, *Department of Mathematics, Gomal University, D.I.Khan, Pakistan.School of Mathematical Sciences, Universiti Sains Malaysia, Penang, Malaysia..*

[15] A.J. Khattak, “A Computational Meshless Method for the Generalized Burger’s- Huxley”, *Applied Mathematical Modelling*, vol. 33, no. 9, pp. 3718-3729, 2009.

[16] R.E Bellman, R.E Kabala , “Quasilinearization and Nonlinear Boundary Value-Problems”, *American Elsevier Publishing*, New York, USA, 1965.

[17] B.A. Malomed, “Stability and Grain Boundaries in the Dispersive Newell–Whitehead–Segel Equation” *Physica Scripta*, vol. 57, no. 1, pp. 115, 1998.

[18] R.L. Burden and J.D Faires,“ Numerical Analysis, Ninth Edition.” *Brooks/Cole, Inc*, 2011.

[19] M. Bastani and D.K Salkuyeh, “A Highly Accurate Method to Solve Fisher’s Equation”, *Journal of Physics*,vol. 78, no. 3, pp. 335-346, 2012.

[20] C. Brezinski, L.Wuytack, “Numerical Analysis:Historical Development in the 20th Century”, *Elsevier Science B.V*, 2001.

[21] B. Batiha, M.S.M Noorani, I. Hashim, “Application of Variational Iteration Method to The Generalized Burgers–Huxley Equation”, *Chaos, Solitons and Fractals*, vol. 36, no. 3, pp. 660-663, 2006.

[22] J.Biazar, H.Aminikhahb,“Exact and Numerical Solutions for Non-Linear Burger’s Equation By VIM”, *Mathematical and Computer Modelling*, vol. 49, no. 7 pp. 1394-1400, 2009.

[23] B Killer and E Isaacson, “ Anaylsis of Numerical Methods”, Courier Corporation, 2012.

[24] C. Shu, “Differential Quadrature and Its Application in Engineering”, *Springer-Verlag London*, 2000.

[25] D.Gottlieb, S.A. Orszag, “Numerical Analysis of Spectral Methods:Theory and Applications”, *SIAM*, 1977.

[26] S. Dhawan, R. Kumar and S. Kumar, "Multi-quadratic Quasi Interpolation for Burger-Fisher Equation", *Int. J. of Appl. Math. and Mech*, vol. 9, no. 11, pp. 41-50, 2013.

[27] P. G. Dlamini, S. S. Motsa, and M. Khumalo, "On the Comparison between Compact Finite Difference and Pseudospectral Approaches for Solving Similarity Boundary Layer Problems," *Mathematical Problems in Engineering*, vol. 2013, Article ID 746489, 15 pages, 2013. doi:10.1155/2013/746489

[28] E. H. Doha, D. Baleanu, A.H Bhrawy, and M.A. Abdelkawy, "A Jacobi Collocation Method for Solving Nonlinear Burgers- Type Equations" *Abstract and Applied Analysis*, vol. 2013, Article ID 760542, 12 pages, 2013. doi:10.1155/2013/760542

[29] G. Ben-Yu, "Spectral Methods and Their Applications", *Shanghai University, P R China*, 2008.

[30] G. Khenkine, "Burgers Type Equations, Gelfand's problem and Schumpeterian Dynamics", *Journal of Fixed Point Theory and Applications* -Springer, vol. 11, no. 2, pp. 199-223, 2012.

[31] J. S. Hesthaven, S.Gottlieb, D. D. Gottlieb, "Spectral Methods for Time-Dependent Problems", *Cambridge University Press*, vol. 21, 2007.

[32] R.A. Fisher, " The wave of advance of advantageous genes," *Wiley Online Library- Annals of Eugenics*, vol. 7, no. 4, pp. 335-369, 1937

[33] G. Hariharan, "An efficient Legrendre Wavalet-Based Approximation Method for Newell-Whiteheaded and Allen-Cahn Equation ", *The Journal of membrane biology* , vol. 247, no. 5, pp. 371-380, 2014.

[34] M. Javidi and A. Golbabaib, "Spectral Collocation Method for Parabolic Partial Differential Equations with Neumann Boundary Conditions", *Applied Mathematical Sciences*, vol. 1, no. 5, pp. 211-218, 2007.

[35] A.H. Khater, R.S. Temsaha, M.M. Hassanb, "A Chebyshev Spectral Collocation Method For Solving Burgers'-Type Equations", *Journal of Computational and Applied Mathematics*, vol. 222, no. 2, pp. 333-350, 2008.

[36] D.A Kaya, El-Sayed and M. Salah, "A Numerical Simulation and Explicit Solutions of the Generalized Burgers–Fisher Equation, " *Applied Mathematics and Computation*,vol. 152, no. 2, pp. 403-413, 2004.

[37] H.F. Easif, S.A Manaa, A.Mekaeel, "Adomain Decomposition Method for Klein Gordon Equation", *International Journal of Engineering Inventions*, vol. 13 ,no. 12, pp.37-42, 2013.

[38] D. Kocacoban, A.B. Koc, A. Kurnaz, and Y. Keskin, "A Better Approximation to the Solution of Burger-Fisher Equation, " *Proceedings of the World Congress on Engineering*, vol. 1, 2011

[39] H. N.A. Ismail, K. Raslan, A.A Abd Rabboh,"Adomian Decomposition Method for Burger's–Huxley and Burger's–Fisher Equations", *Applied Mathematics and Computation*, vol. 159, no. 1, pp. 291-301, 2004.

[40] A. Khattak, "A Computational Meshless Method for the Generalized Burger's–Huxley Equation", *Applied Mathematical Modelling*, vol. 33, no. 9, pp.3718-3729, 2008.

[41] C.C.W. Leentvaar, "Numerical Solution of the Black-Scholes Equation with a Small Number of Grid Points", *Delft University of Technology Faculty of Elec-*

trical Engineering,Mathematics and Computer Science, Section Applied Mathematics, Numerical Analysis Group, 2003.

[42] J. C. Strikwerda, Second Edition, "Finite Difference Schemes and Partial Differential Equations", *SIAM*, 2004.

[43] K. Maleknejad, E. Babolian, A.J Shaerlar, M Jahangir, "Operational Matrices for Solving Burgers' Equation by Using Block-Pulse Functions with Error Analysis", *Australian Journal of Basic and Applied Sciences*, vol. 5, no. 12, pp. 602-608, 2011.

[44] M.M. Meerschaert, "Mathematical Modelling", *Academic Press*, 2013.

[45] M. Nadjafikhah and A. Mahdavi, "Approximate Symmetry Analysis of a Class of Perturbed Nonlinear Reaction-Diffusion Equations," *Abstract and Applied Analysis*, vol. 2013, Article ID 395847, 7 pages, 2013. doi:10.1155/2013/395847

[46] B.H. Gilding and R. Kersner, "Memorandum No. 1585: Travelling Waves in nonlinear Diffusion-Convection-Reaction", *Travelling waves in nonlinear diffusion-convection-reaction. Lecture Notes (Faculty of Mathematical Sciences. Memorandum 1585.)-University of Twente*, 2001.

[47] M. Landajuela, "Burgers Equation ", *Basque Center for Applied Mathematics, Summer Internship*, 2011.

[48] R.C. Mittal and G. Arora,"Efficient Numerical Solution of Fisher's Equation by Using B-Spline Method", *International Journal of Computer Mathematics*, vol. 87, no. 13, pp. 3039-3051, 2010.

[49] R.C. Mittal and R.K.Jain, "Numerical Solutions of Nonlinear Fisher's Reaction–Diffusion Equation with Modified Cubic B-Spline Collocation Method", *Mathematical Sciences*, vol. 7, no. 1, pp. 1-10, 2013.

[50] P. Pue-on, "Laplace Adomian Decomposition Method for Solving Newell-Whitehead-Segel Equation", *Applied Mathematical Sciences*, vol. 7, 2013, no. 132, pp. 6593 – 6600, 2013.

[51] P. A. Ravi, S. Hristova, "Quasilinearization for initial Value Problems Involving Differential Equations with "Maxima"", *Mathematical and Computer Modelling*, vol. 55, no. 9, pp. 2096-2105, 2012.

[52] S.S. Motsa, "Spectral Method Based Numerical Solutions of Partial Differential Equations", *Sixth Annual Workshop On Computational Applied Mathematics And Mathematical Modelling In Fluid Flow*,University of KwaZulu-Natal, 8-12 July 2013.

[53] S.S. Motsa, "New Developments in Spectral Method Based Numerical Solutions Of PDEs: Bi-Variate Spectral Collocation Approach", *Seventh Annual Workshop On Computational Applied Mathematics And Mathematical Modelling In Fluid Flow*,University of KwaZulu-Natal, 7-11 July 2014.

[54] S.A. Manaa, F.M. Saleem, "Numerical Solution and Stability Analysis for Burger's -Huxley Equation" *Raf.J. of Comp. Math's.*, vol. 6, no. 2, 2009.

[55] S.S. Motsa, "On the Bivariate Spectral Homotopy Analysis Method Approach for Solving Nonlinear Evolution Partial Differential Equations," *Abstract and Applied Analysis*, vol. 2014, Article ID 350529, 8 pages, 2014. doi:10.1155/2014/350529

[56] S.S. Motsa and P. Sibanda, "Some Modifications of the Quasilinearization Method with Higher-Order Convergence for Solving Nonlinear BVPs" *Numerical Algorithms*, vol. 63, no. 3, pp. 399-417, 2013.

[57] S.S. Motsa, V. M. Magagula, and P. Sibanda, "A Bivariate Chebyshev Spectral Collocation Quasilinearization Method for Nonlinear Evolution Parabolic Equations," *The Scientific World Journal*, vol. 2014, Article ID 581987, 13 pages, 2014. doi: 10.1155/2014/581987

[58] S.S. Motsa, P. G. Dlamini, and M. Khumalo, "Spectral Relaxation Method and Spectral Quasilinearization Method for Solving Unsteady Boundary Layer Flow Problems," *Advances in Mathematical Physics*, vol. 2014, Article ID 341964, 12 pages, 2014. doi:10.1155/2014/341964

[59] S.S. Motsa, Z. G. Makukula, and S. Shateyi, "Spectral Local Linearisation Approach for Natural Convection Boundary Layer Flow," *Mathematical Problems in Engineering*, vol. 2013, Article ID 765013, 7 pages, 2013. doi:10.1155/2013/765013

[60] P. K Kameswaran, P. Sibanda and S.S Motsa, "A Spectral Relaxation Method for Thermal Dispersion and Radiation Effects in a Nanofluid Flow", *Springer-Boundary Value Problems* , vol. 2013, no. 1, pp. 1-18, 2013.

[61] S.S Motsa, "On The Practical Use of the Spectral Homotopy Analysis Method and Local Linearisation Method for Unsteady Boundary-Layer Flows Caused by an Impulsively Stretching Plate", *Numerical Algorithm*, vol. 66, no. 4, pp. 865-883, 2013.

[62] J. Shen, T. Tang, L. Wang, "Spectral Methods: Algorithms, Analysis and Applications", *Springer Science & Business Media* , vol. 41, 2011.

[63] A. Saravanan, N. Magesh, "A Comparison between the Reduced Differential Transform Method and the Adomian Decomposition Method for The

Newell–Whitehead–Segel Equation", *Journal of the Egyptian Mathematical Society*, vol. 21, no. 3, pp. 259-265, 2013.

[64] T.A. Abassy, M.A. El-Tawil, H.K. Saleh, "The Solution Of Burgers' And Good Boussinesq Equations Using ADM-Pade Technique, " *Chaos Solitons and Fractals*, vol. 32, no. 3, pp. 1008-1026, 2007.

[65] T.S. El-Danaf, M.A. Ramadan, "On the Analytical and Numerical Solutions of the One-Dimensional Non-linear Buurgers' Equation", *The Open Applied Mathematics Journal*, vol. 1, pp. 1-8, 2007.

[66] R.Jiwari, R.K Gupta, V. Kumar, "Polynomial differential quadrature method for numerical solutions of the generalized Fitzhugh–Nagumo equation with time-dependent coefficients", *Ain Shams Engineering Journal*, vol. 5, no. 4, pp. 1343-1350, 2014.

[67] VG. Nguyen,"A Numerical Study Of Burgers' Equation With Robin Boundary Conditions". Masters Thesis. Virginia Polytechnic Institute and State University, 2001. Print.

[68] W. K. Zahra, M. S. El-Azab, "Cubic B-Spline Collocation Algorithm for The Numerical Solution Of Newell Whiteheaded Segel Type Equations" *Electronic Journal of Mathematical Analysis and Applications*, vol. 2, no. 2, pp. 81-100, 2014.

[69] Z.K.Jabar, "Numerical Solution of the Generalized Burger-Huxley by Finite Difference Method ", *J.Thi-Qar Sci.*, vol. 2 , 2010.

[70] M. Zarebnia, S.Jalili, "Application Of Spectral Collocation Method to a Class of Nonlinear PDEs", *Communications in Numerical Analysis*, 2013.

[71] L.N Trefethen," Spectral Methods in Matlab", *SIAM, Philadephia*, 2000.

[72] J.Zahao, "Highly Accurate Compact Finite Difference Method and its Applications", *VDM Puplishing House Ltd*, 2006.

[73] S. Zheng, "Nonlinear Evolution Equations", *Chapman Hall/CRC*, 2004.

[74] D. Kaya, "An explicit solution of coupled viscous Burgers' equation by the decomposition method", *International Journal of Mathematics and Mathematical Sciences*, vol. 27, no. 11, pp. 675-680, 2001. doi:10.1155/S0161171201010249.

[75] A.H. Khater, R.S. Temsaha, D.K. Callebaut, "Numerical solutions for some coupled nonlinear evolution equations by using spectral collocation method", *Mathematical and Computer Modelling* , vol.48, pp.1237–1253, 2008.