ANALYSIS AND APPLICATION OF A LINEARIZATION TECHNIQUE FOR NONLINEAR PROBLEMS

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in Mathematics

by
Neslişah İMAMOĞLU KARABAŞ

December 2020

ACKNOWLEDGMENTS

There are several people I would like to express my gratitude because of their assistance and guidance during the all period of this thesis.

I first and foremost would like to offer my sincerest gratitude to my supervisor Prof. Dr. Gamze TANOĞLU for her kind support, endless patience, understanding and guidance during my search and overcome this dissertation.

I also thank The Scientific and Technological Research Council of Turkey (TÜBİTAK) for its financial support.

I warmly thank and appreciate to my parent for their confidence and support.

Finally, I wish to express my deepest acknowledgement to my husband and best friend Ömer KARABAŞ who encouraged me during the writing processes of this thesis. Without his patience and endless support this thesis would not have accomplished.

ABSTRACT

ANALYSIS AND APPLICATION OF A LINEARIZATION TECHNIQUE FOR NONLINEAR PROBLEMS

The purpose of this thesis is to investigate the implementation of linearization technique combining with the multiquadric radial basis function method to nonlinear problems which appears in engineering and physics. Presented linearization technique is formed by the Frèchet derivatives and Newton Raphson method. This technique is applied to Burgers' equation, Coupled Burgers' equation and 2-D cubic nonlinear Schrödinger equation. From the numerical results of the problems, it is expected that this technique can be used to solve other nonlinear and system of nonlinear partial differential equations numerically.

ÖZET

DOĞRUSAL OLMAYAN PROBLEMLER İÇİN BİR DOĞRUSALLAŞTIRMA TEKNİĞİNİN UYGULANMASI VE ANALİZİ

Bu tezin amacı mühendislikte ve fizikte görülen doğrusal olmayan problemlere multiquadric radyal baz fonksiyonları ile birlikte doğrusallaştırma tekniğini uygulanışını araştırmaktır. Sununulan doğrusallaştırma tekniği Frèchet türevi ve Newton Raphson metodu baz almaktadır. Bu teknik Burger denklemine, Coupled Burger denklemine ve 2-D kübik doğrusal olmayan Schrödinger denklemine uygunlanmıştır. Problemlerin sayısal sonuçlarından, bu tekniğin başka doğrusal olmayan denklemleri ve doğrusal olmayan kısmi türevli denklem sistemlerini sayısal olarak çözmek için kullanılabileceğine inanılmaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

CHAPTER 1

INTRODUCTION

In nature, many phenomena have nonlinear behavior. Generally, these phenomena are modelled by nonlinear partial differential equations (NPDE). Because of this, discovering a solution to a NPDE draws attention. Since every model does not have exact solution, numerical methods have importance to find the solution of a NPDE. The aim of this thesis is to propose a linearization technique combining with the multiquadric radial basis functions to find the numerical solutions of nonlinear problems.

1.1. Introduction

Nonlinear partial differential equations are important in several field of science and technology. In the study of nonlinear wave propagation problems, NPDEs have an important role. These problems arise in various areas of applied mathematics, fluid mechanics, thermodynamics, optics, real-world physical systems. In this thesis, three different NPDEs which are Burgers equation, Coupled Burgers equation and nonlinear cubic Schrödinder equation are considered.

One of the major nonlinear partial differential equation is Burgers equations which appear in fluid dynamics and in general engineering as a simplified model of turbulence, boundary layer behavior, mass transport and wave propagation in acoustic and model traffic flows. The Burgers' equation was presented in Bateman's paper (Bateman, 1915) and the steady-state solution of the problem was given in this work. Later this equation was introduced as mathematical model of turbulence by Burgers (Burgers, 1948). Many researchers have used various methods to seek analytical and numerical solutions to Burgers'-type equations for a wide range of initial and boundary conditions. To solve the Burgers' equation, Hopf and Cole developed a transformation (Ames, 1965). This transformation converts the equation (4.1) into a heat equation and known as a "Hopf-Cole Transformation". A several number of distinct solutions for the Burgers' equation were published by Benton and Platzman (Benton& Paltzman, 1972). Various numerical

methods have been used for solving Burgers' equation such as finite-difference methods (Kaysar, 2010), finite-element methods (Ozis, 2003), (Kutluay, 2004), differential quadrature methods (Jiwari, 2013) and collocation methods (Mittal & Jain, 2012).

Coupled Burgers' equation which is derived by Esipov (Esipov,1995) while studying the model of polydispersive sedimentation. Various analytical and numerical techniques have been used to solve the coupled Burgers' equation. Some of this techniques are: adomian decomposition method (Kaya, 2001), modified extended tanh-function method (Soliman, 2006), the MQ quasi-interpolation method (Chen & Wu, 2007), a chebyshev collocation method (Khater& Temsah& Hassan), a meshfree interpolation method (İslam& Haq& Uddin, 2009) and a collocation of modified cubic B-spline method (Mittal& Tripathi, 2014).

Cubic nonlinear Schrödinger(CNLSE) equation is an important partial differential equation both in physics and in engineering. Several phenomena in quantum mechanics, optics and water waves can be described by the CNLSE. Solving nonlinear Schrödinger equations analytically is a troublesome. Therefore, several researchers have worked on numerical solution of Schrödinger equations: Subaşı has applied finite difference schemes for the numerical solution of two-dimensional Schrödinger equation (Subaşı 2002). Bratsos has used a linearized Crank-Nicolson scheme for numerical solution of nonlinear cubic Schrödinger equation (Bratsos 2001). Wu has used Dufort-frankel-type methods for solving linear and nonlinear Schrödinger equations (Wu 1996). Dehghan and Taleei have applied a compact split-step finite difference method for solving the nonlinear Schrödinger equations (Dehghan & Taleei, 2010).

The main purpose of this thesis is to solve the above NPDEs by a linearization technique. The idea of this technique was presented by the Liu (Liu & Wu, 2000) to find a solution of ordinary differential equation of Duffing type nonlinearity. This technqiue was also used in application of Blasius and Onsiger equations in (Liu & Wu, 2001). In the study of (Fazel et al., 2013), the linearization technqiue also appears. They solve the nonlinear differential equations of motion by using this technique.

In all works (Liu & Wu, 2000) and (Fazel et al., 2013), differential quadrature method was used to find a numerical solution of nonlinear problems. In this study, we focus on combination of linearization technique and multiquadric radial basis function (MQ RBF). Methodology of radial basis function was introduced in 1971 (Hardy, 1971).

In 1990 MQ RBF was used to solve a PDE (Kansa I, 1990).

1.2. Outline of Thesis

The outline of this thesis is organized as follows;

In chapter 2, MQ RBF method is introduced. The idea of this method is explained briefly. Existence and uniqueness of the interpolation matrix is proven intuitively. Approximation of derivatives by using MQ RBF is given.

Linearization technique is proposed in Chapter 3. A conspectus of these concepts is given in Appendix A. The technique is introduced for both NPDEs and system of NPDEs.

Chapter 4 deals with the Burgers' equation. The proposed linearization technique is applied to Burgers' equation in this chapter. A convergence theorem about linearization technique is given. Numerical examples for this equation is illustrated to demonstrate efficiency of the method.

Chapter 5 focus on Coupled Burgers' equation. Linearization technique combining with the MQ RBF is utilized to solve Coupled Burgers' equation numerically. Numerical results are presented to show the accuracy of the method.

Application of the proposed method to 2-dimensional nonlinear cubic Schrödinger equation is presented in Chapter 6. Numerical results are revealed that the technique conserves the qualitative properties of the equations.

Finally, in Chapter 7 we summarize the results and give brief conclusion.

CHAPTER 2

MULTIQUADRIC RADIAL BASIS FUNCTION

The multiquadric radial basis function(MQ RBF) interpolation was introduced in the early 1970s by the Geodesist Rolland Hardy at Iowa State University (Hardy, 1971). Hardy developed this method with the motivation of a problem from cartography. In 1979, mathematician Richard Franke compared the MQ RBF method with different methods and he deuce that MQ RBF was the best to solve the scattered data interpolation problem (Franke, 1979).Then, Charles Michelli proved the invertibility of the system matrix for the MQ, and the development of the theory of the MQ method began in 1986 (Micchelli, 1986). Madych and Nelson in 1992 presented that MQ interpolation has spectral convergence rate (Madych& Nelson, 1992). To solve the differential equations, MQ method was firstly used by Edward Kansa in 1990 (Kansa I, 1990), (Kansa II, 1990). MQ method became populer after it was used to solve pdes.

2.1. Definiton of the Multiquadric Radial Basis Function

Hardy used the quadratic surfaces as the basis functions

$$\Phi(r; c) = \sqrt{c^2 + r^2} \tag{2.1}$$

where $r = \|x\|_2$, $x \in \mathcal{R}^d$ and c is a shape parameter. The function 2.1 is called the multiquadric radial basis function (MQ RBF). Figure 2.1 shows the shape of the surface on the unit circle

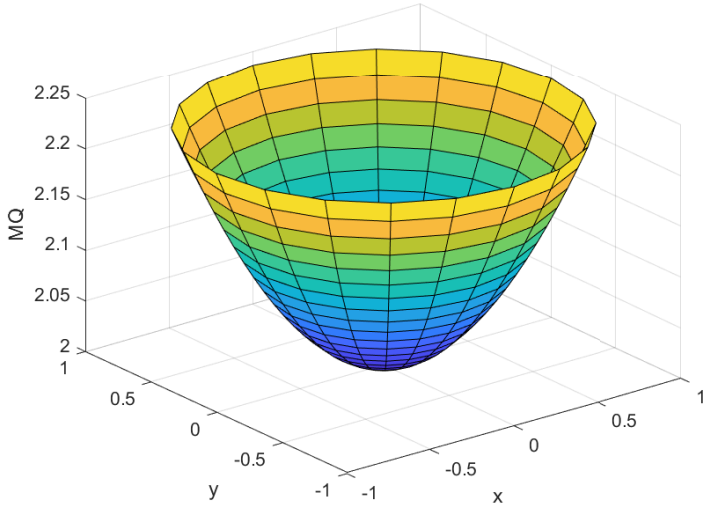


Figure 2.1. Shape of Multiquadric RBF (2.1) with parameter $c = 2$

To find the alternate definition of the MQ, let $c = 1/\varepsilon$ then MQ (2.1) becomes

$$\Phi(r, \varepsilon) = \varepsilon \sqrt{1 + \varepsilon^2 r^2}. \tag{2.2}$$

If the scaling factor ε is ignored, MQ can be redefined as

$$\Phi(r, \varepsilon) = \sqrt{1 + \varepsilon^2 r^2}. \tag{2.3}$$

Figure 2.2 displays the MQ with different values of shape parameter ϵ.

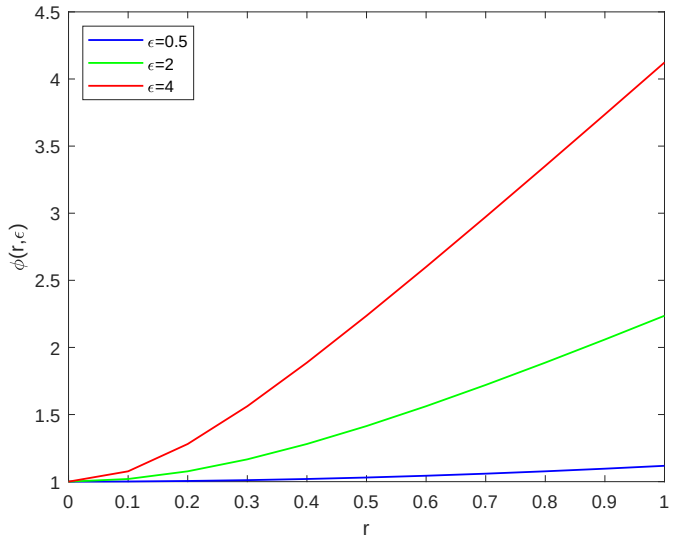


Figure 2.2. Multiquadric RBF with different parameters ε

2.2. RBF Interpolation

The RBF interpolation is given in the following form:

$$\psi(\mathbf{x}) = \sum_{k=1}^{N} \Phi(r, c)\lambda_k, \quad x \in \mathbb{R}^d, \tag{2.4}$$

where $\lambda_k, k = 1, 2, \ldots, N$ are the RBF coefficients and

$$r = \|x - x_k\|_2 = \sqrt{(x - x_1)^2 + \cdots + (x - x_N)^2}.$$

RBF coefficients, λ_k's, are obtained with the implementation of the interpolation condition

$$\psi(\mathbf{x}_i) = f(\mathbf{x}_i) \tag{2.5}$$

where $\mathbf{x}_i$, $i = 1, ..., N$ are the set of centers. Applying the interpolation conditions at N centers forms the $N \times N$ linear system

$$B\lambda = f. \tag{2.6}$$

Entries of the matrix B are

$$b_{jk} = \Phi(\|x_j - x_k\|_2, c), \quad j, k = 1, ..., N. \tag{2.7}$$

2.2.1. Invertibility of Interpolation Matrix

Equation (2.6) reveals that the solution of the MQ interpolation problem exists and is unique if and only if the matrix B is invertible. Invertibility of the MQ interpolation matrix is given with the theorem of Micchelli (Micchelli, 1986) as follows:

Theorem 2.1 *Let $\phi(r) = \Phi(\sqrt{r}) \in C[0, \infty)$ and $\phi(r) > 0$ for $r > 0$. Let $\phi'(r)$ be completely monotone and nonconstant on $(0, \infty)$. Then for any set of distinct centers $\{\mathbf{x}_j\}_{j=1}^N$, the $N \times N$ matrix B with entries $b_{jk} = \Phi(\|x_j - x_k\|_2)$ is invertible.*

To prove the theorem, definition of a completely monotone function is required.

Definition 2.1 *A function ϕ is completely monotone on $[0, \infty)$ if*

(i) $\phi \in C[0, \infty)$

(ii) $\phi \in C^\infty(0, \infty)$

(iii) $(-1)^\ell \phi^\ell(r)$ *where* $r > 0$ *and* $\ell = 0, 1, ...$

Proof 2.1 *For the MQ we have*

$$\phi(r) = \Phi(\sqrt{r}) = \sqrt{1 + \varepsilon^2 r}$$

and

$$\begin{aligned}
\phi'(r) &= \frac{\varepsilon^2}{2\sqrt{1+\varepsilon^2 r}} \\
\phi''(r) &= \frac{-\varepsilon^4}{4(1+\varepsilon^2 r)^{3/2}} \\
\phi^{(3)}(r) &= \frac{3\varepsilon^6}{8(1+\varepsilon^2 r)^{5/2}} \\
\phi^{(4)}(r) &= \frac{-15\varepsilon^8}{16(1+\varepsilon^2 r)^{7/2}}
\end{aligned}$$

Therefore

$$(-1)^{\ell}\phi^{(\ell)}(r) \geq 0 \tag{2.8}$$

and $\phi'(r)$ is completely monotone. Therefore, MQ interpolation matrix B is invertible.

2.2.2. Approximation of Derivatives

By using the following RBF expansion

$$u(x) = \sum_{i=1}^{N} \Phi(\|\mathbf{x} - \mathbf{x_i}\|_2; \varepsilon)\lambda_j \tag{2.9}$$

approximation of the derivatives of the function $u(x)$ can be expressed as

$$\frac{\partial}{\partial x_i}u(\mathbf{x}) = \sum_{j=1}^{N} \lambda_j \frac{\partial}{\partial x_i}\Phi(\|\mathbf{x} - \mathbf{x}_j^c\|; \varepsilon). \tag{2.10}$$

Higher order derivatives can be evaluated in a similar manner. By evaluating (2.10) at the centers $\{\mathbf{x}_j\}_{j=1}^{N}$, vector-matrix notation is obtained as

$$\frac{\partial}{\partial x_i}u(\mathbf{x}) = \frac{\partial}{\partial x_i}H\lambda \tag{2.11}$$

where the entries of $\frac{\partial}{\partial x_i}H$ are

$$h_{ij} = \frac{\partial}{\partial x_i}\Phi(\|\mathbf{x}_i - \mathbf{x}_j\|_2) \quad i, j = 1, ..., N$$

By substituting $\lambda = B^{-1}u$ into (2.11), the differentiation matrix can be defined as

$$D = \frac{\partial}{\partial x_i}HB^{-1}. \tag{2.12}$$

Since the system matrix B is invertible, the differentiation matrix is well-defined. Applying the chain rule to MQ RBF, derivatives are obtained as follows:

$$\frac{\partial \Phi}{\partial x_i} = \frac{d\Phi}{dr}\frac{\partial r}{\partial x_i} \tag{2.13}$$

and

$$\frac{\partial^2 \phi}{\partial x_i^2} = \frac{d\Phi}{dr}\frac{\partial^2 r}{\partial x_i^2} + \frac{d^2\Phi}{dr^2}\left(\frac{\partial r}{\partial x_i}\right)^2 \tag{2.14}$$

where

$$\frac{\partial r}{\partial x_i} = \frac{x_i}{r}, \ \frac{d\Phi}{dr} = \frac{\varepsilon^2 r}{\sqrt{1+\varepsilon^2 r^2}}, \ \frac{d^2\Phi}{dr^2} = \frac{\varepsilon^2}{(1+\varepsilon^2 r^2)^{3/2}}. \tag{2.15}$$

CHAPTER 3

ITERATIVE LINEARIZATION TECHNIQUE

In this chapter iterative linearization technique is presented for the numerical solution of non-linear and system of nonlinear differential equations. The idea of the method is formed by the Fréchet derivative and the Newton-Raphson method. In this method, firstly nonlinear equation is linearized by using the Frèchet derivative as in the works of (Liu & Wu, 2000) and (Fazel et al., 2013). After that, to obtain the numerical solution the meshless method with radial basis function is applied for spatial discretization and Crank-Nicolson method is applied for time discretization. Thus, this method is considered as a linearization technique.
A brief description about the Fréchet derivative and Newton Raphson method is given in the Appendix A.

3.1. Derivation of the Method

Process starts with the considering the general form of the non-linear differential equation

$$L(u) = 0 \tag{3.1}$$

where L is a differential operator. Iterative solution of (3.1) that is obtained by applying the Newton-Raphson method is

$$u^{(n+1)} = u^{(n)} + \theta^{(n)} \tag{3.2}$$

where $u^{(n)}$ is the approximate solution of (), $\theta^{(n)}$ corresponds the refinement variable and n is the iteration number. To find the refinement variable $\theta^{(n)}$ following equation is con-

sidered

$$\theta^{(n)} L'(u^{(n)}) + L(u^{(n)}) = 0. \tag{3.3}$$

Applying the Fréchet derivative, the term $\theta L'(\theta)$ in equation (3.3) is

$$\theta^{(n)} L'(u^{(n)}) = \frac{\partial}{\partial \varepsilon} L(u^{(n)} + \varepsilon \theta^{(n)}) \Big|_{\varepsilon=0}. \tag{3.4}$$

Here the refinement variable θ goes to zero, thus the equation (3.3) turned to the equation (3.1).

3.2. Linearization Process for Systems

Consider the system containing two nonlinear partial differential equations

$$\begin{aligned} L_1(U, V) &= 0, \\ L_2(U, V) &= 0. \end{aligned} \tag{3.5}$$

Due to the Newton-Raphson method, the solution of equation (3.5) is stated as

$$\begin{aligned} U^{n+1} &= U^n + {\theta_1}^n \\ V^{n+1} &= V^n + {\theta_2}^n. \end{aligned} \tag{3.6}$$

where θ_1^n and θ_2^n are refinements.

$$\begin{aligned} {\theta_1}^n {L_1}'(U^n, V^n) + {\theta_2}^n {L_1}'(U^n, V^n) + L_1(U^n, V^n) &= 0, \\ {\theta_1}^n {L_2}'(U^n, V^n) + {\theta_2}^n {L_2}'(U^n, V^n) + L_2(U^n, V^n) &= 0. \end{aligned} \tag{3.7}$$

After applying Fréchet derivatives to get ${\theta_i}^n L_j{}'(U^n, V^n),\ i, j = 1, 2,$

$$
\begin{aligned}
{\theta_1}^n L_i{}'(U^n, V^n) &= \frac{\partial}{\partial \varepsilon} L_i(U^n + \varepsilon {\theta_1}^n, V^n)\Big|_{\varepsilon=0}, \\
{\theta_2}^n L_i{}'(U^n, V^n) &= \frac{\partial}{\partial \varepsilon} L_i(U^n, V^n + \varepsilon {\theta_2}^n)\Big|_{\varepsilon=0}.
\end{aligned} \tag{3.8}
$$

Figure 3.1 describes the iterative procedure.

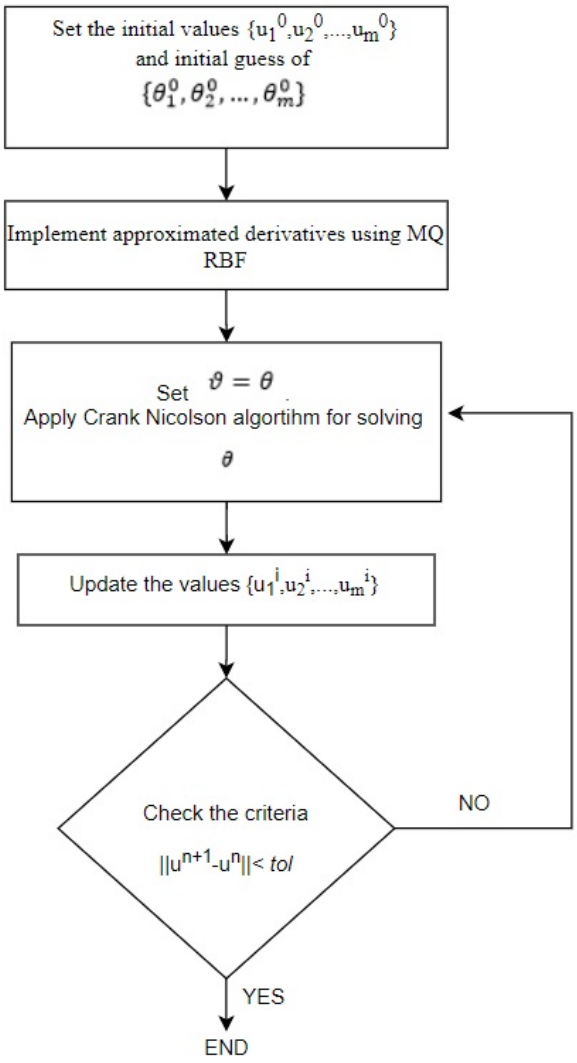


Figure 3.1. Diagram for the iterative solution procedure.

CHAPTER 4

BURGERS' EQUATION

The Burgers' equation is given as follows

$$u_t + uu_x = \kappa u_{xx}. \tag{4.1}$$

In this chapter, to solve the Burgers' equation numerically first linearized technique which is proposed in Chapter 3 is applied. Then to obtain the linear system of equations, for space discretization MQ RBF method is used and as a time discretization Crank-Nicolson rule is applied.

4.1. Linearization of the Burgers' Equation

Consider the Burgers' equation in the differential operator form as

$$L(u) = u_t + uu_x - \kappa u_{xx}. \tag{4.2}$$

Applying the Fréchet derivative to the operator (4.2),

$$\begin{aligned} \theta L'(u) &= \left.\frac{\partial L}{\partial \varepsilon}\right|_{\varepsilon=0} && (4.3)\\ &= \theta_t + u\theta_x + \theta u_x - \kappa\theta_{xx}, && (4.4) \end{aligned}$$

is obtained. Here the derivative (4.3) is

$$\frac{\partial L(u+\varepsilon\theta)}{\partial \varepsilon} = \frac{\partial}{\partial \varepsilon}\Big((u+\varepsilon\theta)_t + (u+\varepsilon\theta)(u+\varepsilon\theta)_x - \kappa(u+\varepsilon\theta)_{xx}\Big). \tag{4.5}$$

Combination of the equation (4.4) and equation (3.3) gives the following equation:

$$\theta L'(u) + L(u) = \theta_t + u\theta_x + \theta u_x - \kappa\theta_{xx} + u_t + uu_x - \kappa u_{xx} = 0. \tag{4.6}$$

Here equation (4.6) is linear with respect to θ. This equation is expressed as

$$\theta_t = \kappa\theta_{xx} - u\theta_x - \theta u_x + \alpha$$

where

$$\alpha = u_t + uu_x - \kappa u_{xx}. \tag{4.7}$$

4.1.1. Convergence Theorem

Theorem 4.1 *Let X and Y be normed space and $L : X \to Y$ be a differential operator such that $L(u) = u_t + uu_x - \kappa u_{xx}0$. Assume that L is Frèchet differentiable has bounded inverse and L is twice Frèchet differentiable. Then the iteration*

$$u^{(n+1)} = u^{(n)} + \theta^{(n)}$$

where $\theta = -(L')^{-1}L$ converges.

Proof 4.1 *Consider the differential operator form of Burgers' equation (4.2), to show that the operator is Frèchet differentiable. We need to apply the definition of the Frèchet*

derivative in the Appendix A.

$$\lim_{\|\theta\|_X \to 0} \frac{\| L(u+\theta) - L(u) - dL(u)[\theta] \|_Y}{\| \theta \|_X} = 0$$
$$\lim_{\|\theta\|_X \to 0} \frac{\| \theta_t + \theta u_x + \theta\theta_x + u\theta_x - \kappa\theta_{xx} - dL(u)[\theta] \|_Y}{\| \theta \|_X} = 0$$
$$dL(u)[\theta] = \theta_t + \theta u_x + u\theta_x - \kappa\theta_{xx}$$

Hence operator (4.2) is Frèchet differentiable. We need to show that $(dL(u)[\theta]) + L(u) = 0$ *has a unique solution, for existence and boundedness of* L^{-1}.

$$\theta_t + \theta u_x + u\theta_x - \kappa\theta_{xx} + u_t + uu_x - \kappa u_{xx} = 0 \tag{4.8}$$

Rewriting 4.9, we obtain

$$\theta_t + \kappa\theta_+ \alpha\theta_x + \beta\theta + \gamma = 0 \tag{4.9}$$

where $\alpha = u$, $\beta = u_x$ *and* $\gamma = u_t + uu_x - \kappa u_{xx}$. *Consider the equation*

$$\theta_t - \kappa\theta_{xx} + \alpha\theta_x + \beta\theta = -\gamma. \tag{4.10}$$

To solve equation (4.10) let $y = x - at$ *and set*

$$v(t, y) = e^{\beta t}\theta(t, y + \alpha t).$$

Then

$$v_t = e^{\beta t}(\theta_t + \alpha\theta_x + \beta\theta) = \kappa\theta_{xx}$$

and

$$v_y = \theta_x, \quad v_{yy} = \theta_{xx},$$

the function $v(t, y)$ satisfies the following differential equation

$$v_t - \kappa v_{yy} = -\gamma \tag{4.11}$$

Therefore solution of equation (4.10) is

$$\theta(t, x) = e^{-\beta t} v(t, x - \alpha t)$$

Integral representation of the solution of the nonhomogoneous heat equation (4.11) in $\mathbb{R}$ is

$$v(t, y) = \int_{-\infty}^{\infty} S(t, y - z)\phi(z)dz + \int_0^t \int_{-\infty}^{\infty} S(t - s, y - z)(-\gamma)dzds \tag{4.12}$$

where $S = \frac{1}{2\sqrt{\pi\kappa t}} e^{-y^2/4\kappa t}$ for $t > 0$ and $\phi(y)$ is the initial condition of the equation (4.11). Hence the solution of equation (4.10) becomes

$$\begin{aligned} \theta(t, x) &= e^{-\beta t} \int_{-\infty}^{\infty} S(t, x - \alpha t - z,)\theta_0(x - \alpha t)dz \\ &+ \int_0^t \int_{-\infty}^{\infty} S(t - s, x - \alpha t - z)e^{\beta(s-t)}(-\gamma)dzds \end{aligned}$$

where $S = \frac{1}{2\sqrt{\pi\kappa t}} e^{-x^2/4\kappa t}$ for $t > 0$ and θ_0 is the initial condition of the equation (4.10). Finally, we need to show that $dL(u)[\theta]$ is Frechet differentiable.

$$\lim_{\|\tau\|_X \to 0} sup_{\|\tau\|_{X=1}} \frac{\| dL(u + \tau)[\theta] - dL(u)[\theta] - d^2L(u)[\theta, \tau] \|_Y}{\| \tau \|_X} = 0$$

$$\lim_{\|\tau\|_X \to 0} sup_{\|\tau\|_X = 1} \frac{\| \theta\tau_x + \tau\theta_x - d^2L(u)[\theta,\tau] \|_Y}{\| \tau \|_X} = 0$$

$$d^2L(u)[\theta,\tau] = \theta\tau_x + \tau\theta_x$$

This means that operator L is twice Frèchet differentiable. Thus, the iteration converges.

4.2. Numerical Results

Linearization of the Burgers equation (4.1) is given in section 4.1. The linearized equation of the Burgers' equation is given with the following equation

$$\theta_t - \kappa A\theta + u(B\theta) + u_t - \kappa Au + u(Bu) = 0 \tag{4.13}$$

where A and B are the differentiation matrix that is obtained by the MQ RBF method. To solve the linear equation (4.13), Crank-Nicolson method is used and the following linear system of equations is obtained

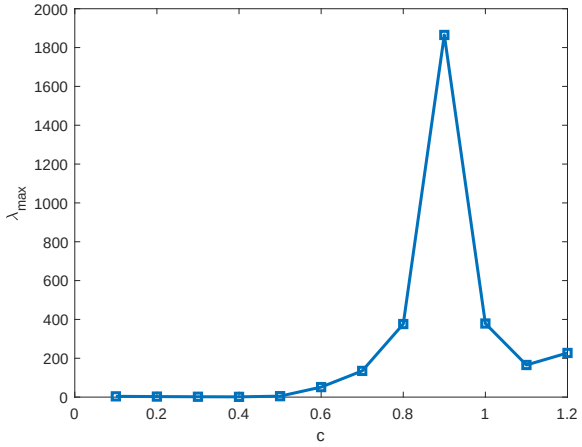
$$\begin{aligned} \frac{\theta_{n+1} - \theta_n}{\Delta t} + \frac{u_{n+1} - u_n}{\Delta t} &= \kappa A \frac{u_{n+1} + u_n}{2} - \frac{u_{n+1} + u_n}{2}\left(B\frac{u_{n+1} + u_n}{2}\right) + \kappa A \frac{\theta_{n+1}\theta_n}{2} \\ &- \frac{u_{n+1} + u_n}{2}\left(B\frac{\theta_{n+1} + \theta_n}{2}\right) - \frac{\theta_{n+1} + \theta_n}{2}\left(B\frac{u_{n+1} + u_n}{2}\right). \end{aligned} \tag{4.14}$$

We have two different test problem about the Burgers' equation as follows.

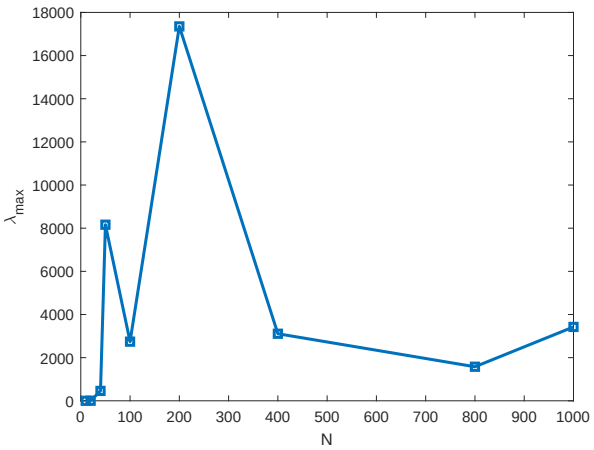
Test Problem 4.1 *Burgers' equation (4.1) is given with the following initial and boundary conditions*

$$\begin{aligned} u(x,0) &= \sin \pi x, x \in [0,1] \\ u(0,t) &= u(1,t) = 0, \quad t > 0. \end{aligned}$$

Numerical experiments mention that selections of the shape parameter c and the number of nodes N are very effective to the condition numbers of the derivative matrices. It is observed from the Figures 4.1a - 4.1b , matrix A in equation (4.13) becomes ill-conditioned according the this selections. Figures 4.1a - 4.1b show the maximum eigenvalue of matrix A versus the shape parameter and number of nodes.



(a)λ_{max} versus c



(b)λ_{max} versus N

Figure 4.2. Stability of differentiation matrix

Numerical results are obtained by the proposed method with the shape parameter $c = 0.5$, $N = 20$ and $\Delta t = 0.0001$. To show the effects of the viscosity, different $\kappa = 0.1, 0.01$ values are taken. Table 4.1 and Table 4.2 are formed for $\kappa = 0.1, 0.01$ respectively. These tables give the numerical solutions of the problem for different time and space values. In Table 4.1 and Table 4.2 we compare the proposed method with the least squares quadratic B-spline finite element method (Kutluay, 2004), a finite element method (Ozis, 2003) and weighted average differential quadrature method (Jiwari, 2013). Comparison reveals that the present method gives the better results than other methods.

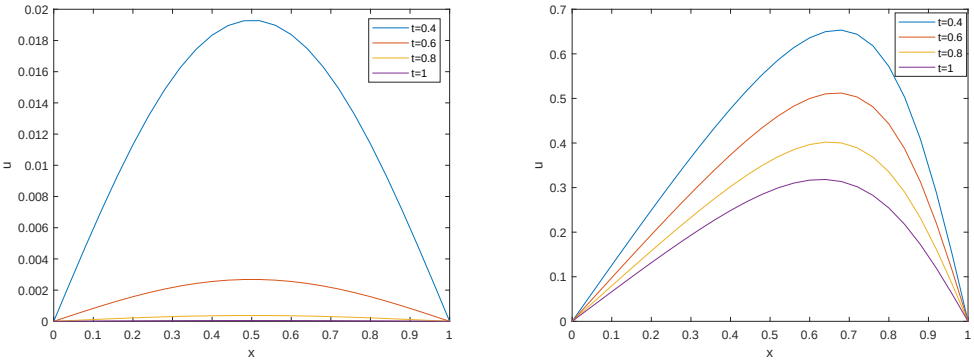
x	$\kappa = 0.1$					
	t	(Kutluay, 2004) $\Delta_t = 0.0001$ $\Delta_x = 0.0125$	(Ozis, 2003) $\Delta_t = 0.0001$ $\Delta_x = 0.0125$	(Jiwari, 2013) $\Delta_t = 0.0001$ $\Delta_x = 0.04$	Present scheme $\Delta_t = 0.0001$ $\Delta_x = 0.05$	Exact Solution
0.25	0.4	0.31429	0.31420	0.30880	0.30889	0.30889
	0.8	0.19758	0.19756	0.19565	0.19567	0.19568
	1.0	0.16391	0.16391	0.16251	0.16256	0.16256
	3.0	0.02743	0.02742	0.02720	0.02720	0.02720
0.50	0.4	0.57636	0.57629	0.56953	0.56963	0.56963
	0.8	0.36245	0.36243	0.35922	0.35924	0.35924
	1.0	0.29437	0.29437	0.29190	0.29192	0.29192
	3.0	0.04057	0.04051	0.04020	0.04021	0.04021
0.75	0.4	0.62592	0.62603	0.62554	0.62545	0.62544
	0.8	0.37713	0.37716	0.37409	0.37394	0.37392
	1.0	0.29016	0.29019	0.28746	0.28749	0.28747
	3.0	0.01334	0.03000	0.02977	0.02977	0.02977

Table 4.1. Numerical and exact solution of test problem 4.1 when $\Delta_x = 0.05, \Delta_t = 0.0001$ and $\kappa = 0.1$.

x	$\kappa = 0.01$				
	t	(Kutluay, 2004) $\Delta_t = 0.0001$ $\Delta_x = 0.0125$	(Jiwari, 2013) $\Delta_t = 0.0001$ $\Delta_x = 0.0125$	Present scheme $\Delta_t = 0.0001$ $\Delta_x = 0.05$	Exact Solution
0.25	0.4	0.34819	0.34191	0.34196	0.34191
	0.8	0.22752	0.22151	0.22204	0.22148
	1.0	0.19375	0.18814	0.18996	0.18819
	3.0	0.07754	0.07537	0.07724	0.07511
0.50	0.4	0.66543	0.66070	0.66073	0.66071
	0.8	0.44526	0.43913	0.43914	0.43914
	1.0	0.38047	0.37434	0.37445	0.37442
	3.0	0.15362	0.15008	0.15202	0.15018
0.75	0.4	0.91201	0.91027	0.91038	0.91026
	0.8	0.65254	0.64739	0.64752	0.64740
	1.0	0.56157	0.55599	0.55613	0.55605
	3.0	0.22874	0.22481	0.22576	0.22481

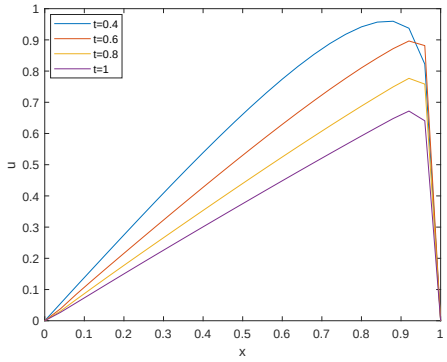
Table 4.2. Numerical and exact solution of test problem 4.1 when $\Delta_x = 0.05, \Delta_t = 0.0001$ and $\kappa = 0.01$.

Behaviour of the solution for different values of viscosity $\kappa = 1, 0.1, 0.01, 0.005$ at the fixed times $t = 0.4,\ 0.8,\ 1$ is given in Figure 4.4.

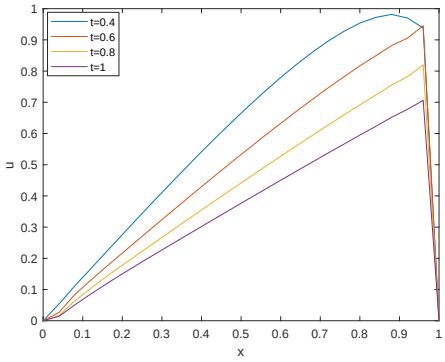


(a) $\kappa = 1$

(b) $\kappa = 0.1$



(c) $\kappa = 0.01$



(d) $\kappa = 0.005$

Figure 4.4. Numerical solutions of test problem 4.1 for different κ values

Figure 4.5 shows layer behaviour of the computed solutions. It is observed from this figure, proposed method gives good results for small value of viscosity.

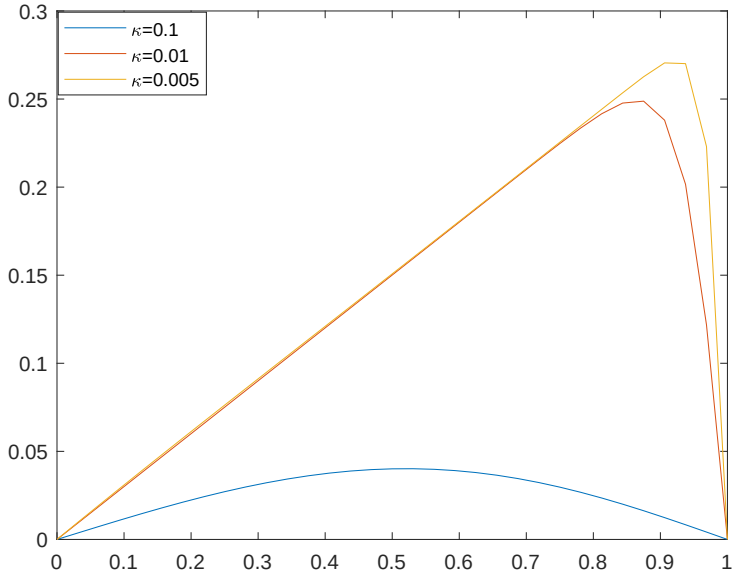


Figure 4.5. Numerical solutions of test problem 4.1 at the end time $T = 3$

Test Problem 4.2 *Consider the Burgers' equation (4.1) with periodic conditions and* $\kappa = 0.03$ *on domain* $[-\pi, \pi]$ *(Kassam & Trefethen, 2005). Initial condition of the problem is*

$$u(x, 0) = e^{-10 \sin^2(x/2)}$$

Figure 4.6 shows the numerical solution of the test problem 4.2.

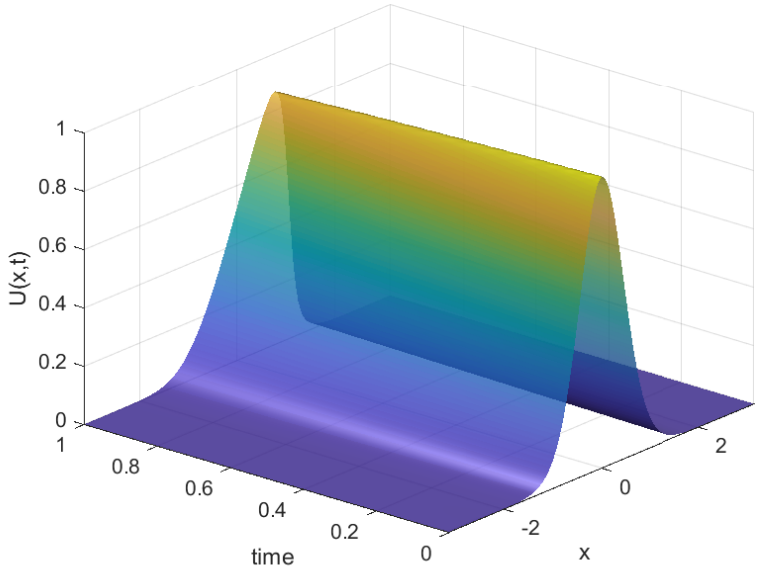


Figure 4.6. Numerical solution of test problem 4.2

Plot of the numerical solutions of the test problem 4.2 is presented at different times in Figure 4.7. From the Figure 4.7,it is observed that numerical solutions conserves the shape of the initial condition.

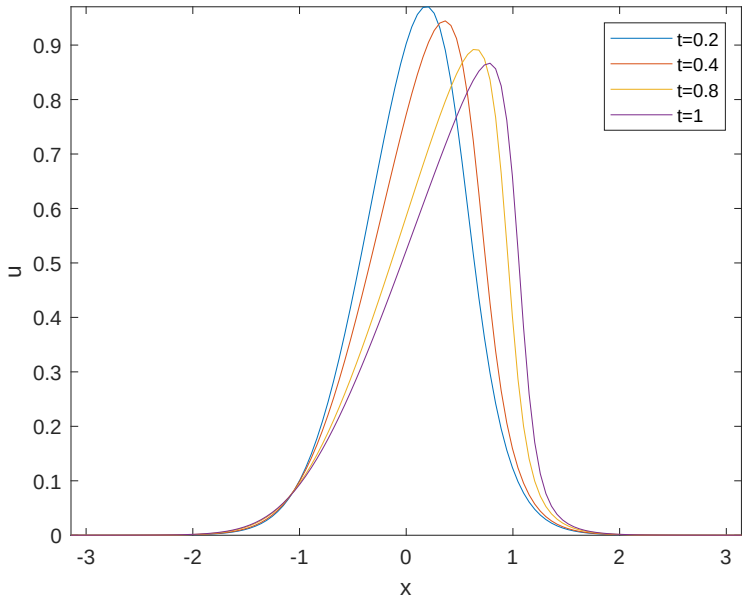


Figure 4.7. Numerical solution of test problem 4.2 at different times

CHAPTER 5

COUPLED BURGERS' EQUATION

Coupled Burgers' equation which is the system of nonlinear partial differential equations are described as the following form

$$\begin{aligned} u_t &= u_{xx} - \eta u u_x - p(uv)_x, \quad a < x < b, \quad 0 \le t \le T \\ v_t &= v_{xx} - \xi v v_x - q(uv)_x, \quad a < x < b, \quad 0 \le t \le T \end{aligned} \tag{5.1}$$

with the initial conditions

$$u(x,0) = f_1(x), \quad v(x,0) = f_2(x), \quad a < x < b \tag{5.2}$$

and the boundary conditions

$$u(a,t) = g_0(t), \quad u(b,t) = g_1(t), 0 \le t \le T \tag{5.3}$$

$$v(a,t) = h_0(t), \quad v(b,t) = h_1(t), \le t \le T \tag{5.4}$$

where η and ξ are real constants, p and q are arbitrary constant depending on the system parameters.

5.1. Linerization of the Coupled Burgers' Equation

To see the efficiency of the technique on system of nonlinear partial differential equations we consider the coupled Burgers' equation (5.1. Writing the equations (5.1)

following operator form is obtained

$$\begin{aligned} L_1(u,v) &= u_t - u_{xx} + \eta u u_x + p(uv)_x \\ L_2(u,v) &= v_t - v_{xx} + \xi v v_x + q(uv)_x \end{aligned} \tag{5.5}$$

By applying the procedure given in section (3.2), we get

$$\begin{aligned} &\theta_{1_t} - \theta_{1_{xx}} + (\eta u + \alpha v)\theta_{1_x} + (\eta u_x + \alpha v_x)\theta_1 + \alpha u_x \theta_2 + \alpha u \theta_{2_x} + L_1(u,v) = 0 \\ &\theta_{2_t} - \theta_{2_{xx}} + (\xi v + \beta u)\theta_{2_x} + (\xi v_x + \beta u_x)\theta_2 + \beta v_x \theta_1 + \beta v \theta_{1_x} + L_2(u,v) = 0. \end{aligned} \tag{5.6}$$

Matrix notation of the system (5.6) is

$$\begin{aligned} \Theta_t - A\Theta + C(u,v)B\Theta &+ D(u,v) + \Theta E(u,v)\tilde{B}\Theta = \\ &- \phi_t + A\phi + F(u,v)B\phi \end{aligned}$$

where $\Theta = [\theta_1, \theta_2]^T$ and $\phi = [u, v]^T$

$$\begin{aligned} A &= \begin{pmatrix} \frac{d^2}{dx^2} & \mathbf{0} \\ \mathbf{0} & \frac{d^2}{dx^2} \end{pmatrix}, \; B = \begin{pmatrix} \frac{d}{dx} & \mathbf{0} \\ \mathbf{0} & \frac{d}{dx} \end{pmatrix} \\ C(u,v) &= \begin{pmatrix} \eta u + pv & \mathbf{0} \\ \mathbf{0} & \xi v + qu \end{pmatrix}, \; D(u,v) = \begin{pmatrix} \eta u_x + p v_x & p u_x \\ q v_x & \xi v_x + q u_x \end{pmatrix} \\ E(u,v) &= \begin{pmatrix} \mathbf{0} & pu \\ qv & \mathbf{0} \end{pmatrix}, \; \tilde{B} = \begin{pmatrix} \mathbf{0} & \frac{d}{dx} \\ \frac{d}{dx} & \mathbf{0} \end{pmatrix} \\ F(u,v) &= \begin{pmatrix} \eta u + pv & pu \\ qv & \xi v + qu \end{pmatrix} \end{aligned}$$

In order to solve the system given in Equation (5.6), we apply Crank-Nicolson method as follows:

$$\frac{\Theta^{n+1}-\Theta^{n}}{\Delta t} - A\frac{\Theta^{n+1}+\Theta^{n}}{2} + C(u^{n+1},v^{n+1})B\frac{\Theta^{n+1}+\Theta^{n}}{2} \tag{5.7}$$

$$+ D(u^{n+1},v^{n+1})\frac{\Theta^{n+1}+\Theta^{n}}{2} + E(u^{n+1},v^{n+1})\tilde{B}\frac{\Theta^{n+1}+\Theta^{n}}{2} = \tag{5.8}$$

$$- \frac{\phi^{n+1}-\phi^{n}}{\Delta t} + A\frac{\phi^{n+1}+\phi^{n}}{2} + F(u^{n+1},v^{n+1})B\frac{\phi^{n+1}+\phi^{n}}{2}. \tag{5.9}$$

5.2. Numerical Results

In this section, we applied the proposed method to different coupled Burgers' equations. L_2 and L_∞ norms are used to calculate the difference between the exact and numerical solutions.

Test Problem 5.1 *Numerical solution of coupled Burgers' equation (5.1) is obtained for* $\alpha = 1$, $\beta = 1$ *and* $\eta = \xi = -2$ *which corresponds to following equations*

$$\begin{aligned} u_t &= u_{xx} + 2uu_x - 1(uv)_x \\ v_t &= v_{xx} + 2vv_x - 1(uv)_x \end{aligned}$$

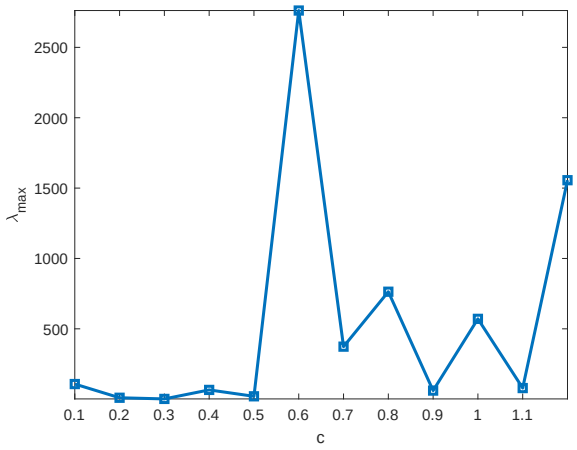
with the given initial conditions

$$u(x,0) = v(x,0) = \sin(x), \quad -\pi \le x \le \pi.$$

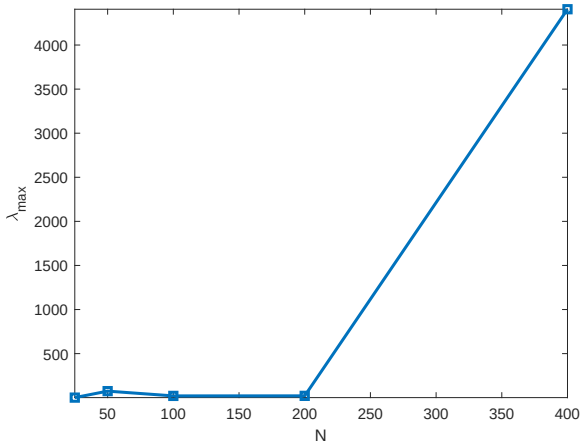
Boundary conditions are taken from the exact solution (Kaya, 2001) which is

$$u(x,t) = v(x,t) = e^{-t}\sin(x) \tag{5.10}$$

Numerical results are obtained by taking the domain $x \in [-\pi, \pi]$ with $\Delta_t = 0.001$. To find the numerical results Crank-Nicolson schemen is used as in equation (5.9). Although Crank-Nicolson is unconditionally stable for equation (5.9), MQ RBF also effects the stability. Figures 5.1a-5.1b show that maximum eigenvalue of differentiation matrix A in equation (5.9) versus shape parameter c and number of nodes N respectively. Matrix A gets ill-conditioned by changing the shape parameter and number of nodes.



(a) λ_{max} versus c



(b) λ_{max} versus N

Figure 5.2. Stability of differentiation matrix

L_2 and L_∞ error norms are computed at different time levels. Numerical errors for u and v are presented in Table 5.1. In Table 5.2, order of convergence of MQ RBF is calculated.

t	N=25		N=50	
	L_2	L_∞	L_2	L_∞
0.1	0.0025	0.0005	2.5383e-04	5.6247e-04
0.5	0.0046	0.0048	3.6715e-04	3.7689e-04
1	0.0066	0.0032	4.8722e-04	2.2854e-04

Table 5.1. L_2 and L_{infty} errors for u(x,t) with $\Delta t = 0.001$

$\varepsilon = 0.5$	t=1	$\Delta t = 0.001$		
N	L_∞	Order of conv.	L_2	Order of conv.
5	0.21774	-	0.59188	-
10	0.04490	2.27776	0.13024	2.18410
20	0.00603	2.89773	0.01265	3.36365
40	0.00063	3.24190	0.00134	3.23428
80	0.00006	3.39779	0.00004	5.17879

Table 5.2. L_2 and L_∞ errors and order of convergence of MQ RBF.

In Figure (5.3), $u(x,t)$ is considered for the comparison of numerical and analytical solutions, because the results are the same for $u(x,t)$ and $v(x,t)$. From this figure it is observed that numerical solutions are in a perfect agreement with the analytical solutions.

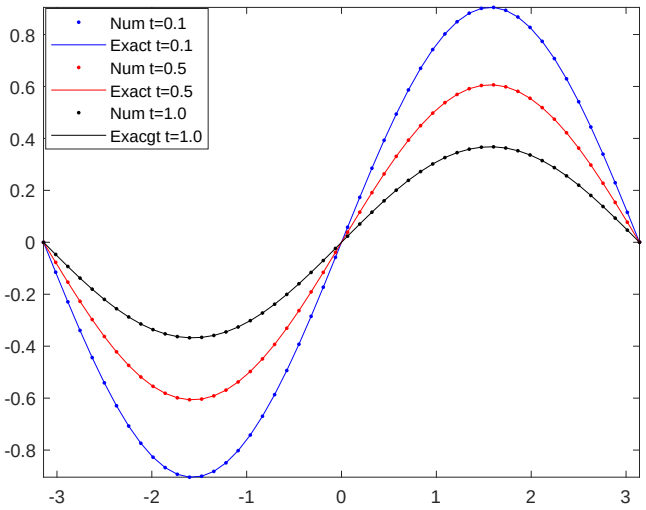


Figure 5.3. A comparison between numerical and analytical results for coupled Burgers' equation

Test Problem 5.2 *In this test problem, coupled Burgers equation is considered with* $\eta = \xi = 2$ *for different values of p and q as follows;*

$$\begin{aligned} u_t &= u_{xx} - 2uu_x - p(uv)_x \\ v_t &= v_{xx} - 2vv_x - q(uv)_x. \end{aligned} \tag{5.11}$$

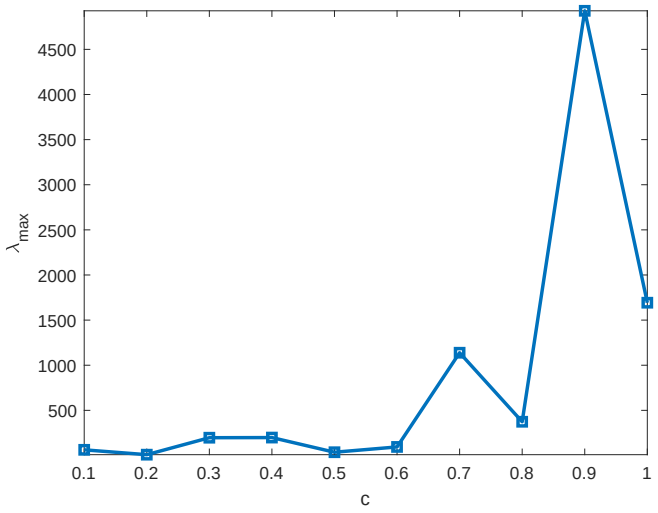
Exact solution of the problem is given in (Soliman, 2006) as

$$\begin{aligned} u(x,t) &= a_0(1 - tanh(A(x - 2At))) \\ v(x,t) &= a_0(\frac{2q-1}{2p-1} - tanh(A(x - 2At))) \end{aligned}$$

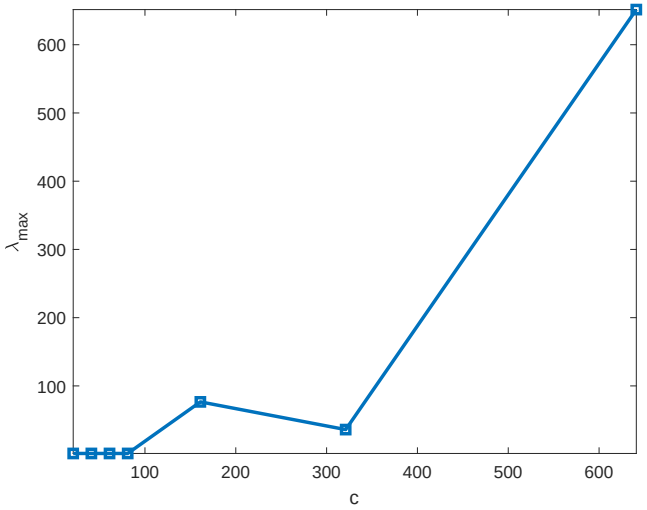
where $A = 0.5a_0\frac{4pq-1}{2p-1}$.

The initial and boundary conditions are taken form the exact solution. The numerical results are obtained for the domain $x \in [-10, 10]$ with $\Delta t = 0.001$.

For the numerical stability, eigenvalues of differentiation matrix A are considered. Figure (5.4a) shows the maximum eigenvalue of matrix A for changing shape parameter c. Effect of the number of nodes N on maximum eigenvalue of A is shown in Figure (5.4b).



(a) λ_{max} versus c



(b) λ_{max} versus N

Figure 5.5. Stability of differentiation matrix

The parameters in equation (5.11) are $a_0 = 0.05, p = 0.1$ and $q = 0.3$. MQ shape parameter is $c = 0.5$ for this problem. L_2 and L_∞ error norms are computed for the solutions of u and v in Table 5.3 and 5.4. From these tables it is observed that numerical solutions has good agreement with the exact solutions.

N	L_2	L_∞
10	4.4483e-04	3.6254e-04
20	5.2476e-04	3.8948e-04
40	4.4128e-04	1.7262e-04
80	5.8670e-04	8.2323e-04

Table 5.3. L_2 and L_{infty} errors for u(x,t) with $\Delta t = 0.001$ at $t = 1$

N	L_2	L_∞
10	2.5483e-045	2.3966e-04
20	2.9526e-04	2.5624e-04
40	2.0773e-04	1.1871e-04
80	2.2936e-04	4.1527e-05

Table 5.4. Errors at $t = 1$ for v(x,t) with $\Delta t = 0.001$

In Figures 5.6 and 5.7 show the numerical and exact solutions of u and v at time $t = 1$ for the parameters $a_0 = 0.1, p = 1$ and $q = 2$.

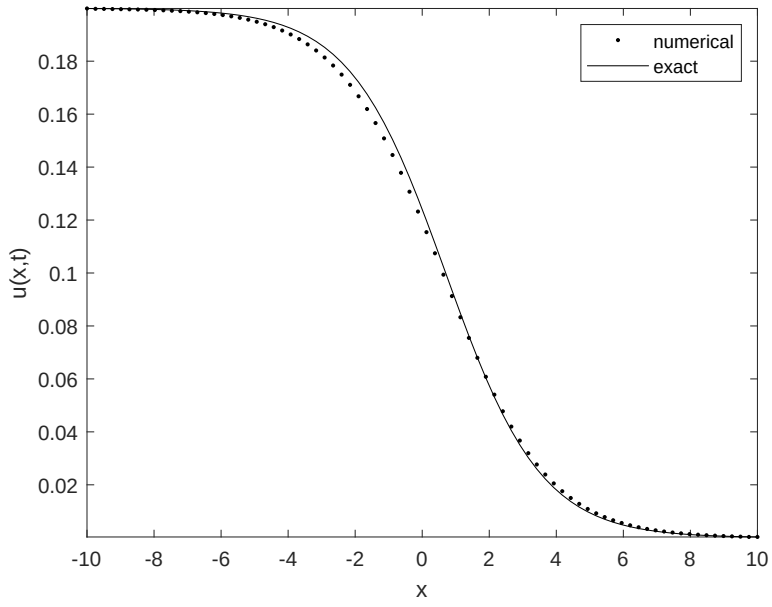


Figure 5.6. Exact and Numerical solutions for u(x,t) at time t=1 for $p = 1,\ q = 2,\ a_0 = 0.1$

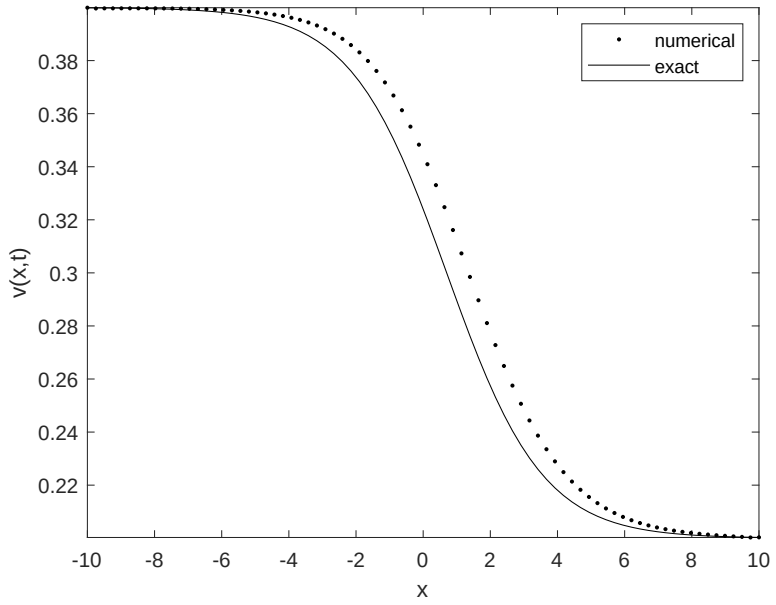


Figure 5.7. Exact and Numerical Solutions for v(x,t) at time t=1 for $p = 1,\ q = 2,\ a_0 = 0.1$

CHAPTER 6

NONLINEAR SCHRÖDINGER EQUATION

We will consider following the general form of the cubic nonlinear Schrödinger equation(CNLSE)

$$i\frac{\partial\psi(t,\mathbf{x})}{\partial t} = [-\alpha\Delta + \beta|\psi(t,\mathbf{x})|^2 + \omega(\mathbf{x})]\psi(t,\mathbf{x}),\ (t,\mathbf{x}) \in (0,T]\times\Omega \tag{6.1}$$

with an initial condition

$$\psi(0,\mathbf{x}) = \psi_0(\mathbf{x}),\ x \in \Omega$$

where $\Delta = \nabla^2$ is Laplace operator, t is time variable, α is a positive constant $\Omega \subset \mathbb{R}^d$ is a bounded domain. In this chapter we consider two dimensional (d=2) CNLSE, and we set the $\mathbf{x} = (x, y)$.

The CNLSE (6.1) conserves many quantitites.In this work, conservation of mass and energy is considered. For CNLSE (6.1), mass functional $N(\psi(t,\mathbf{x}))$ and energy fuctional $E(\psi(t,\mathbf{x}))$ are given as follows;

$$N(\psi(t,\mathbf{x})) = \int_\Omega |\psi(t,\mathbf{x})|^2 d\mathbf{x} \tag{6.2}$$

$$E(\psi(t,\mathbf{x})) = \frac{1}{2}\int_\Omega \left(\alpha|\nabla\psi(t,\mathbf{x})|^2 + \omega(\mathbf{x})|\psi(t,\mathbf{x})|^2 + \frac{\beta}{2}|\psi(t,\mathbf{x})|^4\right)d\mathbf{x} \tag{6.3}$$

6.1. Linerization of the Nonlinear Schrödinger Equation

The nonlinear cubic Schrödinger equation can be expressed as a system of two partial differential equations by separating the real and imaginary parts

$$\psi(t, \mathbf{x}) = U(t, \mathbf{x}) + iV(t, \mathbf{x}) \tag{6.4}$$

$$\begin{aligned} L_1(U, V) &= U_t + \alpha\Delta V - \beta(U^2 + V^2)V - \omega V = 0 \\ L_2(U, V) &= V_t - \alpha\Delta U + \beta(U^2 + V^2)U + \omega U = 0 \end{aligned} \tag{6.5}$$

Applying the linearization technique to the system (6.5) the following

$$\theta_{1_t} + \alpha\Delta\theta_2 - 2\beta UV\theta_1 - \beta(U^2 + 3V^2)\theta_2 - \omega\theta_2 + L_1(U, V) = 0 \tag{6.6}$$

$$\theta_{2_t} - \alpha\Delta\theta_1 + \beta(3U^2 + V^2)\theta_1 + 2\beta UV\theta_2 + \omega\theta_1 + L_2(U, V) = 0 \tag{6.7}$$

is obtained. Matrix notation of the equations (6.6) and (6.7) can be written as follows:

$$\Theta_t + \alpha A\Theta + B\Theta + \beta C(U, V)\Theta = -\Psi_t - \alpha A\Psi - B\Psi - \beta D(|\Psi|)\Psi \tag{6.8}$$

where $\Theta = [\theta_1, \theta_2]^T$ and $\Psi = [U, V]^T$. In addition,

$$A = \begin{bmatrix} \mathbf{0} & \sum_{i=1}^{d} \frac{\partial^2}{\partial x_i^2} \\ -\sum_{i=1}^{d} \frac{\partial^2}{\partial x_i^2} & \mathbf{0} \end{bmatrix}, \quad B = \begin{bmatrix} \mathbf{0} & -\omega \\ \omega & \mathbf{0} \end{bmatrix}$$

$$C(U,V) = \begin{bmatrix} -2UV & -(U^2+3V^2) \\ (3U^2+V^2) & 2UV \end{bmatrix}, \quad D(|\psi|) = \begin{bmatrix} \mathbf{0} & -(U^2+V^2) \\ (U^2+V^2) & \mathbf{0} \end{bmatrix}$$

Here, the operator $\sum_{i=1}^{d} \frac{\partial^2}{\partial x_i^2}$ corresponds to a matrix that is obtained by the MQ RBF method and $\mathbf{0}$ denotes the zero matrix. In order to solve the system given in equation (6.8), Crank-Nicolson method is applied as follows:

$$\begin{aligned} \frac{\Theta^{n+1}-\Theta^n}{\Delta t} + \alpha A\frac{\Theta^{n+1}+\Theta^n}{2} \quad &+ \quad B\frac{\Theta^{n+1}+\Theta^n}{2} + \beta C(U^{n+1},V^{n+1})\frac{\Theta^{n+1}+\Theta^n}{2} = -\frac{\Psi^{n+1}-\Psi^n}{\Delta t} \\ &- \quad \alpha A\frac{\Psi^{n+1}+\Psi^n}{2} - B\frac{\Psi^{n+1}+\Psi^n}{2} - \beta D(|\frac{\Psi^{n+1}+\Psi^n}{2}|)\frac{\Psi^{n+1}+\Psi^n}{2}. \end{aligned}$$

6.2. Numerical Results

In discrete space, mass and energy conservations are given with the following summations

$$DN = \Delta\mathbf{x}\sum_{i=1}^{Nx}|N_i^n - N_i^0|, \tag{6.9}$$

$$DE = \Delta\mathbf{x}\sum_{i=1}^{N_x}(E_i^n - E_i^0). \tag{6.10}$$

where N^0 is initial mass, E^0 is initial energy and

$$\begin{aligned} N_i^n &= |\psi(t_n,\mathbf{x}_i)|^2|(U_i^n)^2+(V_i^n)^2| \\ E_i^n &= \frac{\alpha}{2}|(a_i^n)^2+(b_i^n)^2| + \frac{1}{2}\omega(\mathbf{x}_i)|(U_i^n)^2+(V_i^n)^2| + \frac{\beta}{4}((U_i^n)^2+(V_i^n)^2)^2 \end{aligned}$$

$a = U_x,\ b = V_x$ for $\Psi = U + iV$.

6.2.1. Two-Dimensional Nonlinear Cubic Schrödinger Equation

In equation (6.1), with the $\alpha = 1$, $\omega = 0$, $T = 3$ and $\Omega = [0, \pi]^2$, following equation is considered

$$i\frac{\partial \psi}{\partial t} + \frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} + \beta|\psi|^2\psi = 0 \tag{6.11}$$

with the exact solution of the given model (Abbasbandy et al., 2013)

$$\psi(x, y, t) = A\exp(i(k_1 x + k_2 y - \rho t)), \tag{6.12}$$

where

$$\rho = {k_1}^2 + {k_2}^2 - \beta|A|^2, \tag{6.13}$$

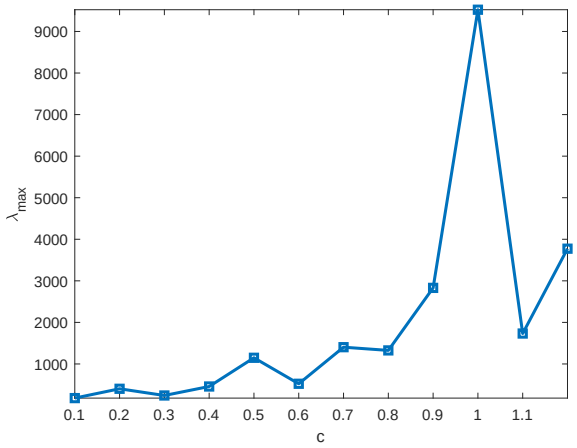
subject to initial and boundary conditions

$$\psi(x, y, 0) = \psi_0(x, y), \quad (x, y) \in \Omega, \tag{6.14}$$

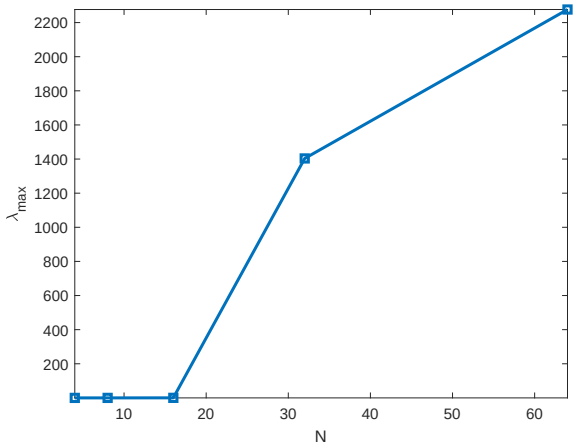
$$\psi(x, y, t) = g(x, y, t), \quad (x, y) \in \partial\Omega,\ t > 0, \tag{6.15}$$

where $\Omega = [a, b] \times [a, b]$ is a rectangular region and $\partial\Omega$ denotes the boundary of Ω.

Figures (6.1a)-(6.1b) show that maximum eigenvalue of differentiation matrix A in equation (6.8) versus shape parameter c and number of nodes N respectively. Varying the shape parameter and number of nodes effect the condition number of the differentiation matrix and matrix becomes ill-conditioned rapidly. Optimal shape parameter c and number of nodes N can be chosen with these graphs.



(a) λ_{max} versus c



(b) λ_{max} versus N

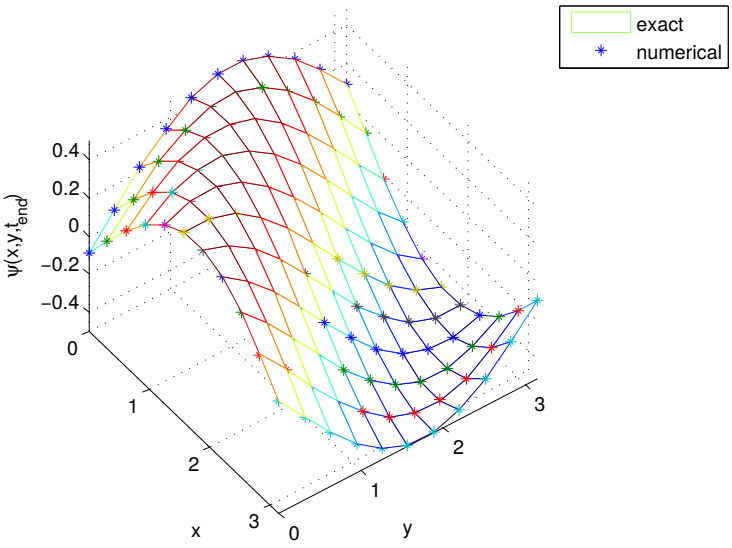
Figure 6.2. Stability of differentiation matrix

The proposed method is applied to this example and the numerical results are shown in Table 6.1. The error norms L_1 and L_2 for the real and imaginary parts are presented with fixed number of collocation points $N = 10 \times 10$ at different times up to t = 1 with fixed $\Delta t = 0.001$.

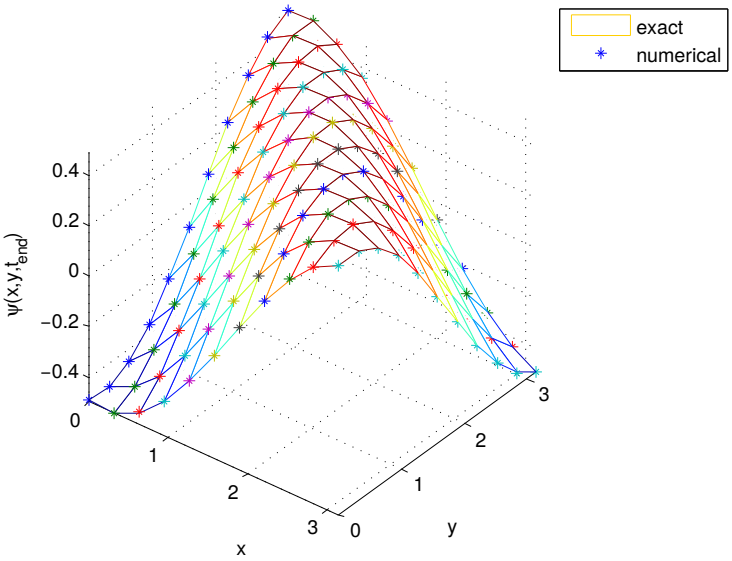
t	Real Part		Imaginary part	
	L_1	L_2	L_1	L_2
0.1	1.2916e-02	6.1962e-03	4.1602e-03	2.1514e-03
0.2	1.8770e-02	9.6928e-03	4.7991e-03	2.4357e-03
0.3	2.2654e-02	1.2328e-02	4.2126e-03	2.3923e-03
0.4	1.9275e-02	1.1574e-02	4.8929e-03	3.3571e-03
0.5	2.0306e-02	1.4381e-02	7.4393e-03	4.8017e-03
0.6	2.4049e-02	1.5879e-02	7.4599e-03	5.6786e-03
0.7	2.8174e-02	1.8491e-02	8.6283e-03	5.4560e-03
0.8	3.0907e-02	2.3580e-02	6.7824e-03	4.6352e-03
0.9	3.4673e-02	2.7562e-02	4.8147e-03	3.4125e-03
1	3.8240e-02	2.8358e-02	4.2647e-03	2.9836e-03

Table 6.1. Numerical errors for $N = 10 \times 10$ with $\Delta t = 0.001$ for equation (6.11)

Numerical results are shown in Figure 6.4 (a)-(b) and from this figure it is observed that the numerical solution is in a great numbness with the exact solution. Here, MQ RBF method is used for discretizing the space derivatives where $N_x = 10 = N_y$ and the shape of the radial basis is 0.5. Moreover, initial and boundary conditions are chosen from the exact solution.



(a)Real part



(b)Imaginary part

Figure 6.4. Surface plot of numerical solution and analytic solution at time $t = 1$.

In Table 6.2 error norms for different number of collocation points with $\Delta t = 0.001$ are shown. It can be observed that the accuracy of the method improves by increasing the number of collocation points. Table 6.3 gives the error at $t = 3$. This emphasize that the

t	$N = 8$		$N = 16$	
	L_1	L_2	L_1	L_2
0.1	8.2572e-03	5.0463e-03	1.6233e-03	8.4697e-04
0.2	1.4362e-02	8.0032e-03	1.4246e-03	8.9173e-04
0.3	1.9362e-02	1.0506e-02	2.1029e-03	1.3948e-03
0.4	1.9564e-02	1.3560e-02	2.4511e-03	1.4347e-03
0.5	2.3993e-02	1.7731e-02	1.9402e-03	1.2821e-03
0.6	3.0219e-02	2.2057e-02	1.8258e-03	1.1608e-03
0.7	3.5083e-02	2.4959e-02	1.8729e-03	1.0928e-03
0.8	3.7903e-02	2.6853e-02	1.6465e-03	1.0567e-03
0.9	4.1190e-02	2.8474e-02	1.9067e-03	1.1634e-03
1	4.0474e-02	2.9549e-02	2.4241e-03	1.5675e-03

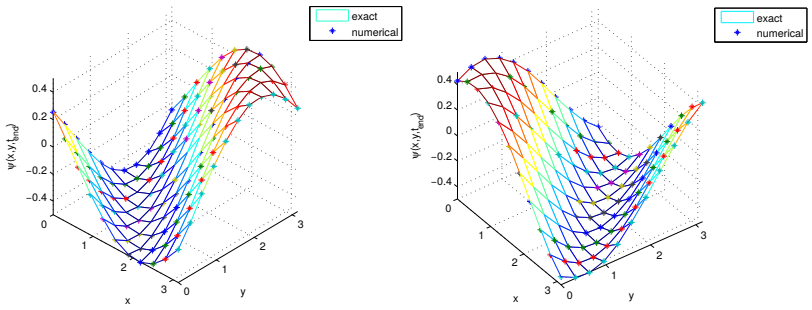
Table 6.2. Numerical errors for $\Delta t = 0.001$ for equation 6.11

proposed method is numerically stable.

N	Δt	L_1	L_2
16×16	0.1	0.2305	0.1505
16×16	0.01	0.0175	0.0122
16×16	0.001	0.0027	0.0016

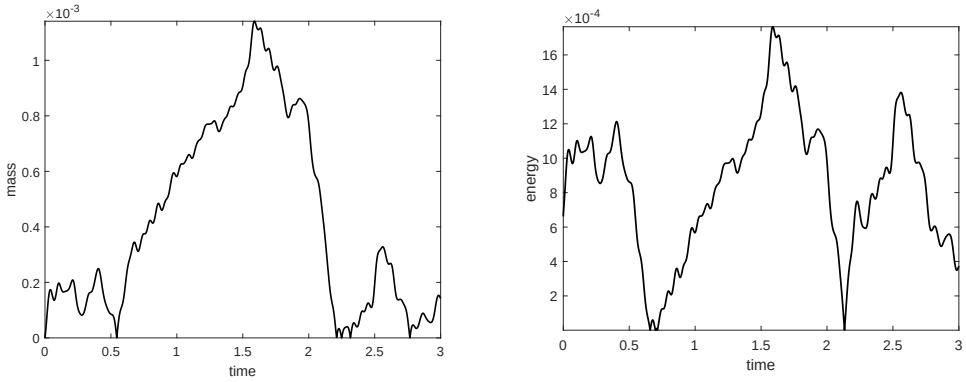
Table 6.3. Numerical errors for different Δt where $x, y \in [0, \pi]^2$ and $t = 3$

In order to check the behaviour of the proposed method for long time, the numerical results at time $t = 3$ are shown in Figure 6.6. From this figure, it is clear that the proposed method preserves the behaviour of the analytic solution for a long time. Figure 6.8 show that the discrete energy and the discrete mass are well preserved.



(a)Real part (b)Imaginary part

Figure 6.6. Surface plot of numerical solution and analytic solution at time $t = 3$.



(a)Mass error (b)Energy error

Figure 6.8. Conservation of mass and energy

6.2.2. Two-Dimensional Nonlinear Cubic Schrödinger Equation with External Potential

In second test problem, CNLSE (6.1) is considered with the external potential as follows $\alpha = \frac{1}{2}, \beta = 1$, $T = 1$ and $\Omega = [0, 2\pi]^2$

$$i\frac{\partial \psi}{\partial t} + \frac{1}{2}\left(\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2}\right) - |\psi|^2\psi + \omega(x, y)\psi = 0, \tag{6.16}$$

with potential

$$\omega(x, y) = -(1 - \sin^2 x \sin^2 y). \tag{6.17}$$

(Xu & Zhang, 2012) The exact solution is given as

$$\psi(x, y) = \sin x \sin y e^{-2it}. \tag{6.18}$$

The boundary and initial values are easily obtained from the equation (6.18). The numerical results about L_1 and L_2 error norms are given in Table 6.4 these results also shows that the presented method remains stable.

$N_x = N_y$	Δt	L_1	L_2
16	$\Delta t = 0.01$	0.0477	0.0377
16	$\Delta t = 0.001$	0.0047	0.0038
16	$\Delta t = 0.0001$	4.7352e-04	3.7539e-04

Table 6.4. Numerical errors for different Δt where $x, y \in [0, 2\pi]^2$

In Figure 6.9, the plot of the obtained approximate solutions and analytical solution at $t = 1$ is given. Error of mass is given in Figure 6.10. From this figure it is observed that the proposed method conserves the mass.

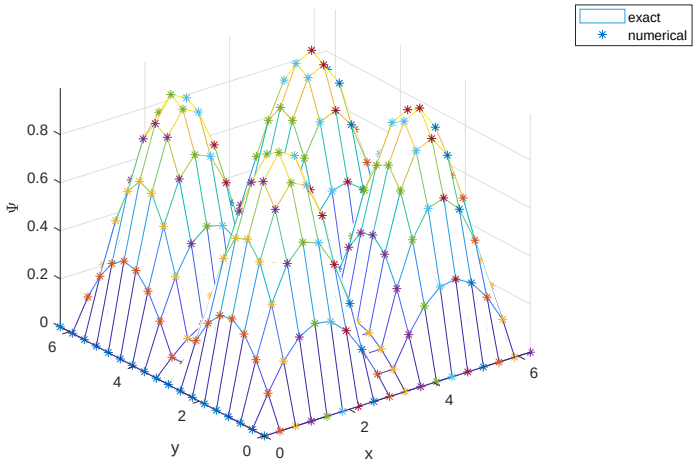


Figure 6.9. Exact and numerical solutions of Equation (6.16) at $t = 1$.

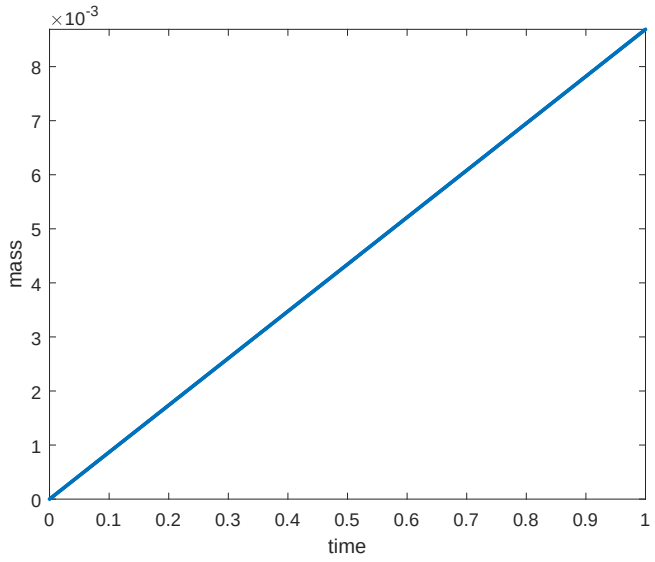


Figure 6.10. Mass error of Equation (6.16) .

CHAPTER 7

CONCLUSION

Nonlinear problems have an important role in science and engineering. Thus the solutions of nonlinear partial differential equations are so important. The aim of this thesis is to propose a numerical method to solve the NPDEs.

The procedure of the proposed method consist of a linearization technique and multiquadric radial basis function method. This linearization technique is formed by Newton-Raphson method and the Frèchet derivatives. MQ RBF method is chosen for the space disceritization. The reason is that the MQ RBF method is easy to implement the multi dimensional problems and MQ RBF method is efficient solver for the linear problems. In addition to solve the problems, Crank Nicolson method is used.

In this thesis three main nonlinear problems are taken into consideration. These are Burgers' equation, Coupled Burgers' equation and cubic nonlinear Schrödinger equation. With these problems, the efficiency and the numerical stability of the proposed method for both nonlinear and the system of nonlinear PDEs are shown. In chapter 4, proposed method is implemented to Burgers' equation. The results are demonstrated that the linearization technique combined with the MQ RBF gives the better solutions then the known techniques in the literature. Figure 4.5 reveals that the method is convenient for small viscosity values. Exact solutions are known for the test problems of coupled Burgers' equation. Thus errors are presented in L_2 and L_∞ norms. Order of convergence of MQ RBF is obtained for the coupled Burgers' equation and one can see the spectral convergence of MQ RBF with this example. Moreover, 2-D cubic nonlinear Schrödinger eqaution is considered for two cases: without and with external potential. For the first case, numerical solution has the similar behaviour with the exact solution. Additionally, proposed method conserves the some densities which are energy and mass. For the CNLSE with external potential, probability density is illustrated in Figure 6.9. Proposed method also conserves the mass for this example but energy is not conserved.

As a result, presented method is applicable for several nonlinear partial differential equations. Numerical results and simulations reveal that proposed method is an efficient solver and one can easily adapt the method for the similar problems.

REFERENCES

Abbasbandy, S., Roohani Ghehsareh, H. and Hashim, I. 2013: A meshfree method for the solution of two-dimensional cubic nonlinear Schrödinger equation, *Eng. Anal. Bound. Elem.,* **37,** 885-898.

Ames, W.F. 1965: Non-linear Partial Differential Equations in Engineering. *Academic Press, New York*.

Bateman, H. 1915: Some Recent Researches on The Motion of Fluids. *Mon. Weather Rev.*, **43,** 163-170.

Benton, E. and Paltzman, G.W. 1972: A table of Solutions of the One Dimensional Burgers' Equations. *Quart. Appl. Math*, **30,**195-212.

Bratsos, A.G. 2001: A linearized finite-difference method for the solution of the nonlinear cubic Schrödinger equation. *Kor. J. Comp. Appl. Math.*, **8,**459-467.

Burgers, J.M. 1948: A Mathematical Model Illustrating the Theory of Turbulance. *Adv. Appl. Mech.*, **1,** 171-199.

Chen, R. and Wu, Z. 2007: Solving partial differential equation by using multiquadric quasi-interpolation. *Appl Math Comput.*, **186,** 1502.

Dehghan, M. and Taleei, A. 2010: A compact split-step finite difference method for solving the nonlinear Schrödinger equations with constant and variable coefficients. *Comp. Phy. Comm.*, **181(1),** 43-51.

Esipov, S. E. 1995: Coupled Burgers Equations: A Model of Polydispersive Sedimentation *Physical Review E*, **52,** 3711.

Fazel, M.R. , Moghadam, M.M. and Poshtan, J. 2013: Application of The GDQ Method in Nonlinear Analysis of a Flexible Manipulator Undergoing Large Deformation. *J. Mech. Eng. Science,* **227(12),** 2671û2685.

Franke, R. 1979: A critical comparison of some methods for the interpolation of scattered data. *Naval Postgraduate School Technical Report* **NPS-53-79-03**.

Hardy, R. 1971: Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysical Research,* **76(8)**.

İslam, S. , Haq, S. and Uddin, M. 2009: A meshfree interpolation method for the numerical solution of the coupled nonlinear partial differential equations. *Eng. Anal. Bound. Elm.*, **33,** 399-499.

Jiwari, R., Mittal, R.C. and Sharma, K.K. 2013: A Numerical Scheme Based On Weighted Average Differential Quadrature Method for The Numerical Solution of Burgers' Equation. *Appl. Math. Comput.*, **219**, 6680-6691.

Khater A.H., Temsah R.S. and Hassan M.M. 2007: A Chebyshev spectral collocation method for solving Burgers' type equations. *J. Appl. Math. Comput*, **222(2),** 333-350.

Kansa, E. J. 1990: Multiquadrics - a scattered data approximation scheme with applications to computational fluid dynamics I: Surface approximations and partial derivative estimates. *Computers and Mathematics with Applications,* **19(8/9),** 127û145.

Kansa, E. J. 1990: Multiquadrics - a scattered data approximation scheme with applications to computational fluid dynamics II: Solutions to parabolic, hyperbolic, and elliptic partial differential equations. *Computers and Mathematics with Applications,* **19(8/9),**147û161.

Kassam, A. K. and Trefethen, L. N. 2005: Fourth-order time-stepping for stiff PDEs. *SIAM J. Sci. Comput.*, **26(4),**1214û1233.

Kaya, D. 2001: An explicit solution of coupled viscous Burgers equations by the Decomposition method. *Int. J. Math. Math. Sci,* **27(11),**675-380.

Kaysar R., Nurmamat H. and Rahmatjan, Y. 2010: Some new semi-implicit finite diffrence scheme for numerical solution of Burgers' equation, In: *International Conference on Computer Application and System Modelling (ICCASM 2010), IEEE, 978-1-4244-7237,***14,** 451-455.

Kutluay, S., Esen, A. and Dag, I. 2004:Numerical Solution of The Burgers' Equation by The Least-Squares Quadratic B-spline Finite Element Method. *J. Comput. Appl. Math.,* **167**, 251-261.

Liu, G.R. and Wu, T.Y. 2000: Numerical Solution for Differential Equations of Duffing-Type Non-Linearity Using the Generalized Differential Quadrature Rule *J.Sound and Vibration ,* **237,** 805-817.

Liu, G. R. and Wu T. Y., 2001: An application of the generalized differential quadrature rule in Blasius and Onsager equations *International Journal for Numerical Methods in Engineering,* **52(9),** 1013-1027.

Madych, W. R. and Nelson, S. A. 1992: Bounds on multivariate interpolation and exponential error estimates for multiquadric interpolation. *Journal of Approximation Theory,* **70,** 94û114.

Mathews J.H. and Fink K.D. 2004: Numerical Methods usin MATLAB. Pearson; 4 edition , ISBN-13:978-0130652485

Micchelli, C. 1986: Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation,* **2**11û22.

Mittal, R.C. and Tripathi, A. 2014: A Collocation Method for Numerical Solutions of Coupled Burgers' Equations. *Int. J. Comp. Meth. Eng. Sci. Mech.*, **15,** 457-471

Mittal, R.C. and Jain, R.K. 2012: Numerical solutions of nonlinear Burgers' equation with modified cubic B-splines collocation method. *Appl. Math. Comput.* , **218,**7839-7855.

Ozis, T., Aksan, E.N. and Ozdes,A. 2003:A Finite Element Approach for Solution of Burgers' Equation. *Appl. Math. Comput.*, **139**, 417-428.

Soliman, A. A. 2006: The modified extended tanh-function method for solving Burgers type equations. *Journal of Physics A: Mathematical and Theoretical*, **361(2),**394û404.

Subaşı, M. 2002: On the finite-differences schemes for the numerical solution of two dimensional Schrödinger equation. *Num. Meth. Part. Diff. Eq.*, **18(6),**752-758.

Wu, L. 1996: Dufortûfrankel-type methods for linear and nonlinear Schrödinger equations. *SIAM J. Numer. Anal.* **33,**1526-1533.

Xu, Y. and Zhang, L. 2012: Alternating direction implicit method for solving two dimensional cubic nonlinear Schrödinger equation. *Comp. Phy. Comm.*, **183,**1082-1093.

APPENDIX A

ANALYTICAL FRAMEWORK

A.1. Derivative in Banach Space

Aim of this appendix is to present the definitions that are used throughout this thesis.

A.1.1. Gateaux Derivative

Definition A.1 *If $f : U \to V$ differentiable at x, then for all $h \in U$ we have*

$$Df(x)h = \lim_{\varepsilon \to 0} \frac{f(x + \varepsilon h) - f(x)}{\varepsilon},$$

where ε is chosen in $\mathbb{R}$.

A.1.2. Fréchet Derivative

The usual definition on general vector spaces, is called Fréchet derivative.

Definition A.2 *Let X and Y be normed vector spaces, and $U \subset X$ open, $f : U \to Y$. We say f is differentiable at $x \in U$ if there exists a bounded linear map $Df(x) \in L(X, Y)$[1] and a continuous function $\psi : V \to Y$, where V is an open neighbourhood of $0 \in X$, with $\psi(0) = 0$, such that*

$$f(x + h) = f(x) + (Df(x))h + ||h||\psi(h)$$

[1] *the space of linear continuous maps from X to Y*

for all $h \in V$.

A.1.3. Higher Derivative

Let $f \in C(U, Y)$ be differentiable in the open set $U \subset X$ and consider $f' : U \to L(X, Y)$

Definition A.3 *Let* $u \in U$*:* f *is twice (Fréchet) differentiable at* u*. The second (Fréchet) differential of* f *at* u *is defined as*

$$d^2 f(u) = df'(u). \tag{A.1}$$

If f is twice differentiable at all points of U we say that f is twice differentiable in U According to the above definition $d^2 f(u)$ is a linear continuous map from X to $L(x, Y)$:

$$d^2 f(u) \in L(X, L(X, Y)). \tag{A.2}$$

It is convenient to see $d^2 f(u)$ as a bilinear map on X. For this, let $L_2(X, Y)$ denote the space of continuous bilinear maps from $X \times X \to Y$. To any $A \in L(X, L(X, Y))$ we can combine $\Phi_A \in L_2(X, Y)$ given by $\Phi_A(u_1, u_2) = [A(u_1)](u_2)$. Conversely, given $\Phi \in L_2(X, Y)$ and $h \in X, \Phi(h, .) : k \to \Phi(h, k)$ is a continuous linear map from X to Y; hence to any $\Phi \in L_2(X, Y)$ is associated the linear application $X \to L(X, Y)$,

$$\Phi : h \to \Phi(h, .) \in L(X, Y) \tag{A.3}$$

It is easy to see that in this way we define an isomorphism between $L(X, L(X, Y))$ and $L_2(X, Y)$. Actually, such an isomorphism is an isometry because there results

$$\|\Phi\|_{L(X,L(X,Y))} = \sup_{\|h\| \le 1} \|\Phi(h)\|_{L(X,Y)} \tag{A.4}$$

$$= \sup_{\|h\| \le 1} \sup_{\|k\| \le 1} \|\Phi(h, k)\| = \|\Phi(h, k)\|_{L_2(X,Y)} \tag{A.5}$$

In the following we will use the same symbol $d^2 f(u)$ to denote the continuous bilinear map obtained by the preceding isometry. The value of $d^2 f(u)$ at a pair (h, k) will be denoted by

$$d^2 f(u)[h, k]. \tag{A.6}$$

If f is twice differentiable in U, the second (Fréchet) derivative of f is the map $f'' : U \to L_2(X, Y)$,

$$f'' : u \to d^2 f(u). \tag{A.7}$$

If f'' is continuous from U to $L_2(X, Y)$ we say that $f \in C^2(U, Y)$. To define $(n + 1)-$th derivatives $(n \geq 2)$ we can proceed by induction. Given $f : U \to Y$, let f be n times differentiable in U. The nth differential at a point $x \in U$ will be identified with a continuous n-linear map from $X \times X \times X \times ... \times X$ ($n$times) to Y (recall that, as before, there is an isometry between $L(X, ..., L(X, Y)$ and $L_n(X, Y)$). Let $f^{(n)} : U \to L_n(X, Y)$

$$f^{(n)} : u \to d^n f(u).$$

The $(n + 1)-$th differential at u will be defined as the differential of $f^{(n)}$, namely

$$d^{n+1} f(u) = df^n(u) \in L(X, L_n(X, Y)) \approx L_n(X, Y).$$

We will say that $f \in C^n(U, Y)$ if f is n times (Fréchet) differentiable in U and the nth derivative f^n is continuous from U to $L_n(X, Y)$. The value of $d^n f(u)$ at $(h_1, ..., h_n)$ will be denoted by

$$d^n f(u)[h_1, ..., h_n].$$

$h = h_1 = ... = h_n$ is written in short form as $d^n f(u)[h]^n$.

A.2. Newton-Raphson Method

Newton-Raphson method is the one of the well-known algorithm which developed for finding root of real-valued functions in the 17th century.

Theorem A.1 *Assume that* $f \in C^2[a, b]$ *and there exists a number* $p \in [a, b]$*, where* $f(p) = 0$*. If* $f'(p) \neq 0$*, the there exists a* $\delta > 0$ *such that the sequence* $p_{n_{n=0}}^{\infty}$ *defined by the iteration*

$$p_n = pn - 1 - \frac{p_{n-1}}{f'(p_{n-1})}; \ \ for; \ \ n = 1, 2, ...$$

will converge to p for any initial approximation $p_0 \in [p - \delta, p + \delta]$.

For more details, see (Matthews& Fink, 2004)

APPENDIX B

MATLAB CODES FOR NUMERICAL EXPERIMENTS

B.1. Codes for Burgers' Equation

```
      %%BURGERS EQUATION%%
%%Ut+UUx=kappaUxx%%
%%U(x,0)=sin(pi*x) , 0<x<1 %%
%%U(0,t)=U(1,t)=0 , 0<t<1%%
clear all;close all;clc
tic
for k=1:3
 %% Space discretization
 x0=0;
 xn=1;
 nx=33;
 dx=(xn-x0)/(nx-1);
 x=x0:dx:xn;
 x=x';
 %%
 gama=[0.1 0.01 0.005]
 %% time
  t0=0;
  tn=3;
  nt=3*10^4+1;
  dt=(tn-t0)/(nt-1);
  t=t0:dt:tn;
```

```
 %%
cr=dt/dx
pe=dx/gama(k)
 %%
 itermax=1000;
 %%
 temp=zeros(nx,1);
 teta=zeros(nx,1);
 cy=zeros(nx,1);
  %%
 ce=sin(pi*x);
 u(:,1)=ce;
  %%
 temp(1)=0;temp(nx)=0;
 teta(1)=0;teta(nx)=0;
 cy(2:nx-1,1)=0;
%%
 c=0.;
for j=1:nx
    for i=1:nx
        A(j,i)=sqrt((x(j)-x(i))^2+c^2);
    end
end
for j=1:nx
    for i=1:nx
        B(j,i)=(x(j)-x(i))/A(j,i);
        C(j,i)=c^2/A(j,i)^3;
    end
end
I=eye(nx);
invA=A\I;
a=B*invA;
```

```
b=C*invA;
"
z=eig(b);
  %% Loop
 for i=1:nt-1
 ce(1)=0;ce(nx)=0;
 cy(1)=0;cy(nx)=0;
 cy(2:nx-1)=cy(2:nx-1,1);
for j=1:itermax
     amat=(1/dt)*eye(nx)+(0.5)*diag((cy+ce)/2)*a
     +(0.5)*diag(a*((cy+ce)/2))-(0.5*gama(k))*b;
     bmat=-(1/dt)*(cy-ce)-(0.25)*(cy+ce).*(a*(cy+ce))
     +(0.5*gama(k))*b*(cy+ce);
"
     teta=amat(2:nx-1,2:nx-1)\bmat(2:nx-1);
     temp(2:nx-1,1)=cy(2:nx-1,1)+teta;
     err=norm(temp-cy,2);
     cy=temp;
     if err<1e-10
         break
     end
 end
k1(i)=j;
     u(:,i+1)=cy;
     ce=cy;
 end

 plot(x,u(:,end))
 hold all
end
figure(2)
plot(x,u(:,4001))
```

```
hold all
plot(x,u(:,6001))
plot(x,u(:,8001))
hold all
plot(x,u(:,10001))
legend('t=0.4','t=0.6','t=0.8','t=1')
   xlabel('x');
   ylabel('u');
figure(3)
[X ,T]=ndgrid(x,t);
surf(X, T,u, ...
    'FaceColor','interp',...
    'EdgeColor','none',...
    'FaceLighting','phong')
axis tight
 view(-50,30)
   camlight left
   alpha(0.6);
   xlabel('x');
ylabel('time');
zlabel('U(x,t)');
title('Burgers Equation');
toc
```

B.2. Codes for Coupled Burgers' Equation

```
      t0=0;
 tn=3;
 dt=0.001
N=(tn-t0)/dt;
 t=t0:dt:tn;
```

```
itermax=1000;
 eta=-2;
 ksi=-2;
 p=1;
 q=1;
%  eta=2;
%  ksi=2;
%  p=0.1;
%  q=0.3;
%  a0=0.5;
%  K=0.5*a0*(4*p*q-1)/(2*p-1)
 for m=1:1
     N1=50*m;
     M=N1-1;
     step(m)=N1;
 %  X = linspace(-10,10,N1);
 X = linspace(-pi,pi,N1);
 nx=N1;
  u=zeros(N1,1);
  v=zeros(N1,1);
  uex=zeros(N1,1);
  vex=zeros(N1,1);
  Z=zeros(2*N1,1);
          for i=1:M+1
     for j=1:N
           uex(i,j)=exp(-t(j))*sin(X(i));
           vex(i,j)=exp(-t(j))*sin(X(i));
%  uex(i,j)=a0*(1-tanh(K*(X(i)-2*K*t(j))));
%  vex(i,j)=a0*((2*q-1)/(2*p-1)-tanh(K*(X(i)-2*K*t(j))));
     end
          end
  Zex=[uex;vex];
```

```
 ce=[uex(:,1);vex(:,1)];
 Z(:,1)=ce;
temp=zeros(2*N1,1);
 teta=zeros(2*N1,1);
 cy=zeros(2*N1,1);
  %%
    temp(1)=uex(1,1);temp(M+1)=uex(end,1);
    temp(M+2)=vex(1,1);temp(end)=vex(end,1);
 %  temp(1)=0;temp(M+1)=0;temp(M+2)=0;temp(end)=0;
 %  teta(1)=0;teta(M+1)=0;teta(M+2)=0;teta(end)=0;
    cy(2:2*M+1,1)=0;
 %  cy(:,1)=Zex(:,2);
    c=0.5;
 %  c=0.1:.1:0.9;
for l=1:length(c)
 ce=[uex(:,1);vex(:,1)];
 Z(:,1)=ce;
 temp=zeros(2*M+1,1);
 teta=zeros(2*M+2,1);
 cy=zeros(2*M+2,1);
 %%
  temp(1)=uex(1,1);temp(M+1)=uex(end,1);
  temp(M+2)=vex(1,1);temp(end)=vex(end,1);
%  temp(1)=0;temp(M+1)=0;temp(M+2)=0;temp(end)=0;
%  teta(1)=0;teta(M+1)=0;teta(M+2)=0;teta(end)=0;
  cy(2:2*M+1,1)=0;
% cy(:,1)=Zex(:,2);
 for j=1:nx
    for i=1:nx
        A(j,i)=sqrt((X(j)-X(i))^2+c(l)^2);
    end
 end
```

```
for j=1:nx
    for i=1:nx
        B(j,i)=(X(j)-X(i))/A(j,i);
        C(j,i)=c(l)^2/A(j,i)^3;
    end
end
I=eye(nx);
invA=A\I;
cond(A)
a=B*invA;
b=C*invA;
AA=[b zeros(nx,nx); zeros(nx,nx) b];
BB=[a a; a a];
for i=1:N-1
 ce(1)=uex(1,i+1);ce(M+1)=uex(end,i+1);
ce(M+2)=vex(1,i+1); ce(2*M+2)=vex(end,i+1);
 cy(1)=uex(1,i+1);cy(M+1)=uex(end,i+1);
cy(M+2)=vex(1,i+1); cy(2*M+2)=vex(end,i+1);
%   ce(1)=0;ce(M+1)=0; ce(M+2)=0; ce(2*M+2)=0;
%   cy(1)=0;cy(M+1)=0; cy(M+2)=0; cy(2*M+2)=0;
for j=1:itermaxlog(dt(e)/dt(e-1))));
   vec11=eta*(ce(1:M+1)+cy(1:M+1))/2
          +p*(ce(M+2:end)+cy(M+2:end))/2;
   vec12=p*(ce(1:M+1)+cy(1:M+1))/2;
   vec21=q*(ce(M+2:end)+cy(M+2:end))/2;
   vec22=ksi*(ce(M+2:end)+cy(M+2:end))/2+q*(ce(1:M+1)+cy(1:M+1))/2;
   ADM=[diag(vec11), diag(vec12); diag(vec21), diag(vec11)];
   cvec11=eta*a*((ce(1:M+1)+cy(1:M+1))/2)
              +p*a*((ce(M+2:end)+cy(M+2:end))/2);
   cvec12=p*a*((ce(1:M+1)+cy(1:M+1))/2);
   cvec21=q*a*((ce(M+2:end)+cy(M+2:end))/2);
   cvec22=ksi*a*((ce(M+2:end)+cy(M+2:end))/2)
```

```
                +q*a*((ce(M+2:end)+cy(M+2:end))/2);
   AUM=[diag(cvec11), diag(cvec12); diag(cvec21), diag(cvec22)];
   DMAT=(((1/dt)*eye(size(AA)))-(AA*0.5)+(ADM*BB*0.5))+(AUM*0.5);
   rvec1=eta*((ce(1:M+1)+cy(1:M+1))/2).*a*((ce(1:M+1)+cy(1:M+1))/2)
   +p*a*((ce(1:M+1)+cy(1:M+1))/2).*((ce(M+2:end)+cy(M+2:end))/2)
   +p*((ce(1:M+1)+cy(1:M+1))/2).*a*((ce(M+2:end)+cy(M+2:end))/2);
   rvec2=ksi*((ce(M+2:end)+cy(M+2:end))/2).*
   a*((ce(M+2:end)+cy(M+2:end))/2)
   +q*a*((ce(1:M+1)+cy(1:M+1))/2).*((ce(M+2:end)+cy(M+2:end))/2)
   +q*((ce(1:M+1)+cy(1:M+1))/2).*a*((ce(M+2:end)+cy(M+2:end))/2);
   DVEC=vertcat(rvec1,rvec2);
   rmat1=((-(1/dt)*eye(size(AA)))+(AA*0.5))*cy;
   rmat2=(((1/dt)*eye(size(AA)))+(AA*0.5))*ce;
   RS=rmat1+rmat2-DVEC;
   teta=DMAT\RS;
   temp=cy+teta;
   err=norm((temp-cy),2);
   cy=temp;
   if err<1e-10
       break
   end
 end
     Z(:,i+1)=cy;
     ce=cy;
end
 e1(l)=max(abs(Z(1:N1,end)-Zex(1:N1,end)));
 e2(l)=norm(abs(Z(1:N1,end)-Zex(1:N1,end)),2);
 ei(l)=norm(abs(Z(1:N1,end)-Zex(1:N1,end)),inf);
e11(m,l)=e1(l);
e21(m,l)=e2(l);
e3(l)=max(abs(Z(N1+1:end,end)-Zex(N1+1:end,end)));
e4(l)=norm(abs(Z(N1+1:end,end)-Zex(N1+1:end,end)),2);
```

```
e12(m,l)=e3(l);
e22(m,l)=e4(l);
end
dx(m)=2*pi/N1;
if m>1
   order1(m)=abs((log(e11(m)/e11(m-1)))/(log(dx(m)/dx(m-1))));
   order2(m)=abs((log(e21(m)/e21(m-1)))/(log(dx(m)/dx(m-1))));
   else
order1(m)=0;
order2(m)=0;
end
end
% figure(1)
% plot(X,Z(1:N1,101),'.b',X,Zex(1:N1,101),'-b')
% hold on
% plot(X,Z(1:N1,501),'.r',X,Zex(1:N1,501),'-r')
% plot(X,Z(1:N1,end),'.k',X,Zex(1:N1,end),'-k')
% axis tight
% title('u(x,t)')
% figure(2)
% plot(X,Z(N1+1:end,101),'.b',X,Zex(N1+1:end,101),'-b')
% hold on
% plot(X,Z(N1+1:end,501),'.r',X,Zex(N1+1:end,501),'-r')
% plot(X,Z(N1+1:end,end),'.k',X,Zex(N1+1:end,end),'-k')
% axis tight
% title('v(x,t)')
e1
e2
ei
figure(3)
semilogy(c,e11(1,:))
xlabel c
```

```
ylabel error
title('MQ-RBF')
figure(4)
semilogy(c,e11(2,:))
xlabel c
ylabel error
\title('MQ-RBF')
```

B.3. Codes for Cubic Nonlinear Schrödinger Equation

```
      clear all;close all
tic
M =15; %30; % Number of intervals in x-direction
N = M; % Number of intervals in z-direction
N1 = N+1 ;
Np = (N+1)^2;
shape = .7;
X = linspace(0,pi,M+1);
Y = linspace(0,pi,N+1);
[x1,y1] = meshgrid(X,Y);
x = reshape(x1',N1^2,1);
y = reshape(y1',N1^2,1);
dx=pi/N1;
dy=dx;
%% Initial
C0=zeros(2*Np,1);
C0(1:Np)=0.5*cos(x+y);
C0(Np+1:end)=0.5*sin(x+y);
ce=C0;
cy=zeros(2*Np,1); %Psi_(n+1)
temp=zeros(2*Np,1);
```

```
teta=zeros(2*Np,1);
%% Time
t0=0; tend=3;
dt=0.001; Nt=(tend-t0)/dt;
t=t0:dt:tend;
for i=1:Nt+1
    ex1(:,i)=0.5*(cos(x+y-7*t(i)/4));
    ex2(:,i)=0.5*(sin(x+y-7*t(i)/4));
end
exa1=permute(reshape(ex1,N1,N1,Nt+1),[1 2 3]);
exa2=permute(reshape(ex2,N1,N1,Nt+1),[1 2 3]);
solut1=zeros(N1,N1,Nt+1);
solut2=zeros(N1,N1,Nt+1);
H = zeros(Np,Np);  rx = zeros(Np,Np);  ry = zeros(Np,Np);
r = zeros(Np,Np);
      o = ones(1,length(x));
      %% Boundaries
    dirichletBCLR1 = find( x==0 );
    dirichletBCLR2 = find( x==pi );
    dirichletBCL = find( (y == 0) & ~(x==0 | x== pi));
    dirichletBCU = find( (y == pi) & ~(x==0 | x== pi));
    interior = find( x~=0 & x~=pi & y~= 0 & y~=pi );
    c1=ce(1:Np);
    c2=ce(Np+1:end);
    cy1=cy(1:Np);
    cy2=cy(Np+1:end);
    c1(dirichletBCLR1)=.5*cos(y(dirichletBCLR1)-7*t0/4);
    c1(dirichletBCLR2)=.5*cos(pi+y(dirichletBCLR2)-7*t0/4);
    c1(dirichletBCL)=.5*cos(x(dirichletBCL)-7*t0/4);
    c1(dirichletBCU)=.5*cos(pi+x(dirichletBCU)-7*t0/4);
    c2(dirichletBCLR1)=.5*sin(y(dirichletBCLR1)-7*t0/4);
    c2(dirichletBCLR2)=.5*sin(pi+y(dirichletBCLR2)-7*t0/4);
```

```
    c2(dirichletBCL)=.5*sin(x(dirichletBCL)-7*t0/4);
    c2(dirichletBCU)=.5*sin(pi+x(dirichletBCU)-7*t0/4);
    ce=vertcat(c1,c2);
    solut1(:,:,1)=permute(reshape(c1,N1,N1),[1,2]);
    solut2(:,:,1)=permute(reshape(c2,N1,N1),[1 2]);
    %% RBF matrix
    rx = (x*o - (x*o)');
  ry = (y*o - (y*o)');
   r = sqrt( rx.^2 + ry.^2 );
   mr=mq(r,shape,2);
  Hxx = mqDerivatives(r,rx,shape,2);
  Hyy = mqDerivatives(r,ry,shape,2);
  H=Hxx+Hyy;
  H(dirichletBCLR1,:) = mq(r(dirichletBCLR1,:),shape);
  H(dirichletBCLR2,:) = mq(r(dirichletBCLR2,:),shape);
  H(dirichletBCL,:) = mq(r(dirichletBCL,:),shape);
  H(dirichletBCU,:) = mq(r(dirichletBCU,:),shape);
  I=eye(Np);
invA=mr\I;
cninv=cond(invA)
b=(H)*invA;
BA=[zeros(Np,Np), b; -b, zeros(Np,Np)];
cnba=cond(BA)


%% Time Loop
itermax=100;
tol=1e-10;
temp1=temp(1:Np);
temp2=temp(Np+1:end);
  for j=2:Nt+1
  c1=ce(1:Np);
```

```
c2=ce(Np+1:end);
cy1=cy(1:Np);
cy2=cy(Np+1:end);
c1(dirichletBCLR1)=.5*cos(y(dirichletBCLR1)-7*t(j)/4) ;
c1(dirichletBCLR2)=.5*cos(pi+y(dirichletBCLR2)-7*t(j)/4);
c1(dirichletBCL)=.5*cos(x(dirichletBCL)-7*t(j)/4);
c1(dirichletBCU)=.5*cos(pi+x(dirichletBCU)-7*t(j)/4);
c2(dirichletBCLR1)=.5*sin(y(dirichletBCLR1)-7*t(j)/4);
c2(dirichletBCLR2)=.5*sin(pi+y(dirichletBCLR2)-7*t(j)/4);
c2(dirichletBCL)=.5*sin(x(dirichletBCL)-7*t(j)/4);
c2(dirichletBCU)=.5*sin(pi+x(dirichletBCU)-7*t(j)/4);
ce=vertcat(c1,c2);
cy1(dirichletBCLR1)=.5*cos(y(dirichletBCLR1)-7*t(j)/4);
cy1(dirichletBCLR2)=.5*cos(pi+y(dirichletBCLR2)-7*t(j)/4);
cy1(dirichletBCL)=.5*cos(x(dirichletBCL)-7*t(j)/4);
cy1(dirichletBCU)=.5*cos(pi+x(dirichletBCU)-7*t(j)/4);
cy2(dirichletBCLR1)=.5*sin(y(dirichletBCLR1)-7*t(j)/4) ;
cy2(dirichletBCLR2)=.5*sin(pi+y(dirichletBCLR2)-7*t(j)/4);
cy2(dirichletBCL)=.5*sin(x(dirichletBCL)-7*t(j)/4);
cy2(dirichletBCU)=.5*sin(pi+x(dirichletBCU)-7*t(j)/4);
cy=vertcat(cy1,cy2);
temp1(dirichletBCLR1)=.5*cos(y(dirichletBCLR1)-7*t(j)/4);
temp1(dirichletBCLR2)=.5*cos(pi+y(dirichletBCLR2)-7*t(j)/4);
temp1(dirichletBCL)=.5*cos(x(dirichletBCL)-7*t(j)/4);
temp1(dirichletBCU)=.5*cos(pi+x(dirichletBCU)-7*t(j)/4);
temp2(dirichletBCLR1)=.5*sin(y(dirichletBCLR1)-7*t(j)/4);
temp2(dirichletBCLR2)=.5*sin(pi+y(dirichletBCLR2)-7*t(j)/4);
temp2(dirichletBCL)=.5*sin(x(dirichletBCL)-7*t(j)/4);
temp2(dirichletBCU)=.5*sin(pi+x(dirichletBCU)-7*t(j)/4);
temp=vertcat(temp1,temp2);
for k=1:itermax
    % left side
```

```
 cvec11=2*(((ce(1:Np)+cy(1:Np))/2).*((ce(Np+1:end)+cy(Np+1:end))/2))
 cvec12=((((ce(1:Np)+cy(1:Np))/2).^2)
 +(3*(((ce(Np+1:end)+cy(Np+1:end))/2).^2)));
 cvec21=((3*(((ce(1:Np)+cy(1:Np))/2).^2))
 +((((ce(Np+1:end)+cy(Np+1:end))/2).^2)));
 ADM=[diag(cvec11), diag(cvec12); -diag(cvec21), -diag(cvec11)];
 DMAT=(((1/dt)*eye(size(BA)))+(BA*0.5)+(ADM*0.5));
     % rigth side
rvec1=((((0.5)*(ce(1:Np)+cy(1:Np))).^2)
+(((0.5)*(ce(Np+1:end)+cy(Np+1:end))).^2))
.*((0.5)*(ce(Np+1:end)+cy(Np+1:end)));
rvec2=-((((0.5)*(ce(1:Np)+cy(1:Np))).^2)
+(((0.5)*(ce(Np+1:end)+cy(Np+1:end))).^2)).*((0.5)*(ce(1:Np)+cy(1:Np)));
DVEC=vertcat(rvec1,rvec2);
rmat1=((-(1/dt)*eye(size(BA)))-(BA*0.5))*cy;
rmat2=(((1/dt)*eye(size(BA)))-(BA*0.5))*ce;
RS=rmat1+rmat2-DVEC;
teta=DMAT\RS;
   temp=cy+teta;
 err=norm((temp-cy),2);
 cy=temp;
 if err<=tol
%       point(i)=k;
     break
 end
 end
 ce=cy; %%%sol(x,t)
 sol=ce;
sol1=sol(1:Np); sol2=sol(Np+1:end);
solut1(:,:,j)=permute(reshape(sol1,N1,N1),[1 2 ]);
solut2(:,:,j)=permute(reshape(sol2,N1,N1),[1 2 ]);
 end
```

```
exact=zeros(2*N1,1);
exact1=0.5*cos(x1+y1-((7/4)*tend));
exact2=0.5*sin(x1+y1-((7/4)*tend));
xnew=x;
ynew=y;
[temp,li]=sort(xnew);
xnew=xnew(li);
ynew=ynew(li);
appnew1=sol1(li);
appnew2=sol2(li);
for i=1:N1
    [temp,li]=sort(ynew((i-1)*N1+1:i*N1));
    li=(i-1)*N1+li;
    xnew(((i-1)*N1+1:i*N1))=xnew(li);
    ynew(((i-1)*N1+1:i*N1))=ynew(li);
    appnew1(((i-1)*N1+1:i*N1))=appnew1(li);
    appnew2(((i-1)*N1+1:i*N1))=appnew2(li);
end
xnew=reshape(xnew,N1,N1);
ynew=reshape(ynew,N1,N1);
appnew1=reshape(appnew1,N1,N1);
appnew2=reshape(appnew2,N1,N1);
err1=max(max(abs(appnew1-exact1))) %real part
err2=max(max(abs(appnew2-exact2)))%imaginary part
ENEEXA= sqrt(((exact1.^2 +exact2.^2)));
PROB=sqrt(((appnew1.^2)+(appnew2.^2)));
errprob=max(max(abs(ENEEXA-PROB)))
err=max(abs(ENEEXA-PROB))
errorl1=norm(ENEEXA-PROB,1)
errorl2=norm(ENEEXA-PROB,2)
errorli=norm(ENEEXA-PROB,inf)
N0=exa1(:,:,1).^2+exa2(:,:,1).^2;
```

```
for j=1:Nt+1 %%t counter
top=0;
e0=0;
for i=1:N %%x counter
 for k=1:N
 der=((((solut1(i+1,k,j)-solut1(i,k,j))/(dx)
 +(solut1(i,k+1,j)-solut1(i,k,j))/(dy)).^2)
 +(((solut2(i+1,k,j)-solut2(i,k,j))/(dx)
 +(solut2(i,k+1,j)-solut2(i,k,j))/(dy)).^2));
 energy0(i,k)=31/64;
 ham=((-(1/4)*(((((solut1(i,k,j).^2))+((solut2(i,k,j)).^2))).^2))
 +(0.5*der));
 energson(i,k)=(-(1/4)*((((solut1(i,k,j).^2))+((solut2(i,k,j)).^2))).^2)
 +(0.5*der);
 top=top+ham;
 e0=e0+energy0(i,k);
 end
end
 E(j)=dx*dy*top;
 EE(j)=dx*dy*e0;
 GE(j)=abs(E(j)-(31/64)*pi^2);
end
enerbas=dy*dx*norm(energy0,1);
enerson=dy*x*norm(energson,1);
for k=1:Nt+1
N=solut1(:,:,k).^2+solut2(:,:,k).^2;
mass0=0;
mass=0;
for i=2:N1
    for j=2:N1
       mass0=mass0+N0(i,j)*dy*dx;
       mass=mass+N(i,j)*dy*dx;
```

```
    end
end
m0(k)=mass0;
m(k)=mass;
end
%R=GE/enerrbas;
  %DE=abs(en-eee);
  DN=abs(m-m0);
  figure(1)
  plot(t,DN)
  axis tight
  xlabel('time')
  ylabel('mass error')
  figure(2)
  plot(t,GE)
  axis tight
  xlabel('time')
  ylabel('energy error')
figure(3)
mesh(x1,y1,exa1(:,:,end))
hold on
plot3(xnew,ynew,solut1(:,:,end),'*')
axis tight
```

B.4. Codes for Schrödinger Equation with External Potential

```
     clear all;
close all
tic
M =15; %30; % Number of intervals in x-direction
N = M; % Number of intervals in z-direction
```

```
N1 = N+1 ;
Np = (N+1)^2;
shape =0.5;
%coefficients
alfa=0.5;
beta=-1;
%%points
X = linspace(0,2*pi,M+1); %
Y = linspace(0,2*pi,N+1); %
[x1,y1] = meshgrid(X,Y);
x = reshape(x1',N1^2,1);
y = reshape(y1',N1^2,1);
dx=2*pi/N1;
dy=dx;
%% Initial
C0=zeros(2*Np,1);
C0(1:Np)=sin(x).*sin(y);
ce=C0;
cy=zeros(2*Np,1); %Psi_(n+1)
temp=zeros(2*Np,1);
teta=zeros(2*Np,1);
%% Time
t0=0; tend=1;
dt=0.001; Nt=(tend-t0)/dt;
t=t0:dt:tend;
%% EXACT
for i=1:Nt+1
    ex1(:,i)=sin(x).*sin(y)*cos(2*t(i));
    ex2(:,i)=-sin(x).*sin(y)*sin(2*t(i));
end
exa1=permute(reshape(ex1,N1,N1,Nt+1),[1 2 3]);
exa2=permute(reshape(ex2,N1,N1,Nt+1),[1 2 3]);
```

```
solut1=zeros(N1,N1,Nt+1);
solut2=zeros(N1,N1,Nt+1);

H = zeros(Np,Np);  rx = zeros(Np,Np);  ry = zeros(Np,Np);
r = zeros(Np,Np);
      o = ones(1,length(x));
      %% Boundaries
    dirichletBCLR1 = find( x==0 );
    dirichletBCLR2 = find( x==2*pi );
    dirichletBCL = find( (y == 0) & ~(x==0 | x== 2*pi));
    dirichletBCU = find( (y ==2* pi) & ~(x==0 | x== 2*pi));
    interior = find( x~=0 & x~=2*pi & y~= 0 & y~=2*pi );
    c1=ce(1:Np);
    c2=ce(Np+1:end);
    cy1=cy(1:Np);
    cy2=cy(Np+1:end);
    c1(dirichletBCLR1)=0;c1(dirichletBCLR2)=0;
    c1(dirichletBCL)=0; c1(dirichletBCU)=0;
    c2(dirichletBCLR1)=0;c2(dirichletBCLR2)=0;
    c2(dirichletBCL)=0; c2(dirichletBCU)=0;
    ce=vertcat(c1,c2);

solut1(:,:,1)=permute(reshape(c1,N1,N1),[1,2]);
solut2(:,:,1)=permute(reshape(c2,N1,N1),[1 2]);
%% RBF matrix
    rx = (x*o - (x*o)');
  ry = (y*o - (y*o)');
   r = sqrt( rx.^2 + ry.^2 );
   mr=mq(r,shape,2);
  Hxx = mqDerivatives(r,rx,shape,2);
  Hyy = mqDerivatives(r,ry,shape,2);
```

```
  H=Hxx+Hyy;
  H(dirichletBCLR1,:) = mq(r(dirichletBCLR1,:),shape);
  H(dirichletBCLR2,:) = mq(r(dirichletBCLR2,:),shape);
  H(dirichletBCL,:) = mq(r(dirichletBCL,:),shape);
  H(dirichletBCU,:) = mq(r(dirichletBCU,:),shape);
  I=eye(Np);
invA=mr\I;
cninv=cond(invA)
b=(H)*invA;
BA=[zeros(Np,Np), b; -b, zeros(Np,Np)];
cnba=cond(BA)
%% Potential
v=-(1-(sin(x).^2).*(sin(y).^2));
VV=[zeros(Np,Np), diag(v); -diag(v),zeros(Np,Np)];
%% Time Loop
itermax=100;
tol=1e-10;
temp1=temp(1:Np);
temp2=temp(Np+1:end);
  for j=2:Nt+1
  c1=ce(1:Np);
  c2=ce(Np+1:end);
  cy1=cy(1:Np);
  cy2=cy(Np+1:end);
  c1(dirichletBCLR1)=0;c1(dirichletBCLR2)=0;
  c1(dirichletBCL)=0; c1(dirichletBCU)=0;
  c2(dirichletBCLR1)=0;c2(dirichletBCLR2)=0;
  c2(dirichletBCL)=0; c2(dirichletBCU)=0;
  ce=vertcat(c1,c2);
  cy1(dirichletBCLR1)=0;cy1(dirichletBCLR2)=0;
  cy1(dirichletBCL)=0; cy1(dirichletBCU)=0;
  cy2(dirichletBCLR1)=0;cy2(dirichletBCLR2)=0;
```

```
 cy2(dirichletBCL)=0; cy2(dirichletBCU)=0;
 cy=vertcat(cy1,cy2);
 temp1(dirichletBCLR1)=0;temp1(dirichletBCLR2)=0;
 temp1(dirichletBCL)=0; temp1(dirichletBCU)=0;
 temp2(dirichletBCLR1)=0;temp2(dirichletBCLR2)=0;
 temp2(dirichletBCL)=0; temp2(dirichletBCU)=0;
 temp=vertcat(temp1,temp2);
 for k=1:itermax
     % left side
 cvec11=2*(((ce(1:Np)+cy(1:Np))/2).*((ce(Np+1:end)+cy(Np+1:end))/2
 cvec12=((((ce(1:Np)+cy(1:Np))/2).^2)
 +(3*(((ce(Np+1:end)+cy(Np+1:e))
 cvec21=((3*(((ce(1:Np)+cy(1:Np))/2).^2))
 +((((ce(Np+1:end)+cy(Np+1:end))/2).^2)));
 ADM=beta*[diag(cvec11), diag(cvec12); -diag(cvec21), -diag(cvec11)];
 DMAT=(((1/dt)*eye(size(BA)))+alfa*(BA*0.5)+(ADM*0.5))+VV*.5;

     % rigth side
rvec1=((((0.5)*(ce(1:Np)+cy(1:Np))).^2)
+(((0.5)*(ce(Np+1:end)+cy(Np+1:end))).^2))
.*((0.5)*(ce(Np+1:end)+cy(Np+1:end)));
rvec2=-((((0.5)*(ce(1:Np)+cy(1:Np))).^2)
+(((0.5)*(ce(Np+1:end)+cy(Np+1:end))).^2))
.*((0.5)*(ce(1:Np)+cy(1:Np)));
DVEC=vertcat(rvec1,rvec2);
rmat1=-(((1/dt)*eye(size(BA)))-alfa*(BA*0.5))*cy;
rmat2=(((1/dt)*eye(size(BA)))-alfa*(BA*0.5))*ce;
RS=rmat1+rmat2-beta*DVEC-VV*.5*(cy+ce);
teta=DMAT\RS;
   temp=cy+teta;
 err=norm((temp-cy),2);
 cy=temp;
```

```
  if err<=tol
%         point(i)=k;
         break
    end
    end
    ce=cy; %%%sol(x,t)
    sol=ce;
 sol1=sol(1:Np); sol2=sol(Np+1:end);
 solut1(:,:,j)=permute(reshape(sol1,N1,N1),[1 2 ]);
 solut2(:,:,j)=permute(reshape(sol2,N1,N1),[1 2 ]);
    end
  toc
  exact=zeros(2*N1,1);
exact1=sin(x1).*sin(y1)*cos(2*tend);
exact2=sin(x1).*sin(y1)*sin(-2*tend);
xnew=x;
ynew=y;
[temp,li]=sort(xnew);
xnew=xnew(li);
ynew=ynew(li);
appnew1=sol1(li);
appnew2=sol2(li);
for i=1:N1
    [temp,li]=sort(ynew((i-1)*N1+1:i*N1));
    li=(i-1)*N1+li;
    xnew(((i-1)*N1+1:i*N1))=xnew(li);
    ynew(((i-1)*N1+1:i*N1))=ynew(li);
    appnew1(((i-1)*N1+1:i*N1))=appnew1(li);
    appnew2(((i-1)*N1+1:i*N1))=appnew2(li);
end
xnew=reshape(xnew,N1,N1);
ynew=reshape(ynew,N1,N1);
```

```
appnew1=reshape(appnew1,N1,N1);
appnew2=reshape(appnew2,N1,N1);
err1=max(max(abs(appnew1-exact1))) %real part
err2=max(max(abs(appnew2-exact2)))%imaginary part
ENEEXA= sqrt(((exact1.^2 +exact2.^2)));
PROB=sqrt(((appnew1.^2)+(appnew2.^2)));
errprob=max(max(abs(ENEEXA-PROB)))
errorl1=norm(ENEEXA-PROB,1)
errorl2=norm(ENEEXA-PROB,2)
errorli=norm(ENEEXA-PROB,inf)
N01=exa1(:,:,1).^2+exa2(:,:,1).^2;
C01=permute(reshape(C0(1:Np),N1,N1),[1 2]);
C02=permute(reshape(C0(Np+1:end),N1,N1),[1,2]);
N0=solut1(:,:,1).^2+solut2(:,:,1).^2;

for j=1:Nt+1 %%t counter
top=0;
e0=0;
for i=1:N1-1 %%x counter
    en0=0;
    for k=1:N1-1
der=((((solut1(i+1,k,j)-solut1(i,k,j))/(dx)
+(solut1(i,k+1,j)-solut1(i,k,j))/(dy)).^2)
 +(((solut2(i+1,k,j)-solut2(i,k,j))/(dx)
 +(solut2(i,k+1,j)-solut2(i,k,j))/(dy)).^2));
energy0(i,k)=0.5*alfa*((cos(x1(i,k)).*sin(y1(i,k))).^2
+(sin(x1(i,k).*cos(y1(i,k)))).^2)
+0.5*((1-((sin(x1(i,k)).*sin(y1(i,k))).^2)))
*((sin(x1(i,k)).*sin(y1(i,k))).^2)
-0.25*beta*(sin(x1(i,k)).*sin(y1(i,k))).^4;
energson(i)=((0.5*alfa)*((((solut1(i,k,j).^2))+((solut2(i,k,j)).^2))).^2)
-(0.25*beta*der)
```

```
(0.5*(((solut1(i,k,j).^2))+((solut2(i,k,j)).^2))
*((1-(sin(x(i)).^2).*(sin(y(k)).^2))));
  top=top+ham;
   e0=e0+energy0(i,k);
    end
end
top1(j)=dx*dy*top;
EE(j)=dx*dy*e0;
  GE(j)=abs(dx*dy*top-8.48169);
end
% EE=dx*dy*sum(sum(energy0));
% GE=abs(top1-EE);
enerbas=dy*dx*norm(energy0,1);
%enerson=dy*x*norm(energson,1);
for k=1:Nt+1
N=solut1(:,:,k).^2+solut2(:,:,k).^2;
mass0=0;
mass=0;
for i=2:N1
    for j=2:N1
       mass0=mass0+N0(i,j)*dy*dx;
       mass=mass+N(i,j)*dy*dx;
    end
end
m0(k)=mass0;
m(k)=mass;
end
  DN=abs(m-m0);
  figure(1)
  plot(t,DN,'.-')
  axis tight
  xlabel('time')
```

```
ylabel('mass')
figure(2)
plot(t,GE,'.-')
axis tight
xlabel('time')
ylabel('energy')

 figure(5)
 mesh(x1,y1,ENEEXA)
 hold on
 plot3(xnew,ynew,PROB,'*')
  axis tight
 xlabel('x')
 ylabel('y')
 zlabel('\Psi')
```

VITA

Date and Place of Birth: 19.04.1989, İzmir-TURKEY

EDUCATION

2015 - 2020 Doctor of Philosophy in Mathematics

Graduate School of Engineering and Sciences, İzmir Institute of Technology,

İzmir -Turkey

Thesis Title: ANALYSIS AND APPLICATION OF LINEARIZATION TECHNIQUE FOR NONLINEAR PROBLEMS

Supervisor: Prof. Dr. Gamze Tanoğlu

2012 - 2014 Master of Science in Mathematics

Graduate School of Engineering and Sciences, İzmir Institute of Technology,

İzmir -Turkey

Thesis Title: TWO NUMERICAL APPROACHES FOR SOLVING NONLINEAR STIFF DIFFERENTIAL EQUATIONS

Supervisor:Prof. Dr. Gamze Tanoğlu

2011-2012 Pedagogical Formation Program

Faculty of Education, Ege University, İzmir- Turkey

2007-2011 Bachelor of Mathematics

Department of Mathematics, Faculty of Science, Ege University,

İzmir - Turkey

PROFESSIONAL EXPERIENCE

2013 - 2021 Research and Teaching Assistant

Department of Mathematics, İzmir Institute of Technology,

İzmir -Turkey

PUBLICATIONS

İmamoğlu N., Gürarslan G. and Tanoğlu G., 2014: "Numerical Solution of Burger Equation by Using General Frechet Derivatives Combined with Differential Quadrature." Abstract Book of ICRAPAM 2014, International Conference on Recent Advances in Pure and Applied Mathematics, ISBN: 978-975-00211-1-4, Pp. 135

İmamoğlu N., Gürarslan G., Tanoğlu G. and Yılmaz, Y., 2017: "Frechet Derivative Based Linearization Method for Burgers-Type Equations." Abstract Book of MME 2017, International Workshop on Mathematical Methods in Engineering, ISBN: 978-975-6734-19-3, Pp. 125

İmamoğlu Karabaş N., Korkut Uysal, S. Ö., Erdoğan U. and Tanoğlu G., 2017: "On the weak convergence of exponential ıntegrators for Stochastic ordinary differential equations." Abstract Book of WDEA 2017, The 8.th International Workshop on Differential Equations and Applications, Pp. 14

İmamoğlu Karabaş N., Korkut Uysal, S.,Tanoğlu G., Aziz, İ. and İslam, S., 2020 : "An Efficient Approach for Solving Nonlinear Multidimensional Schrödinger Equations", Engineering with Boundary Elements (submitted).