TWO NUMERICAL APPROACHES FOR SOLVING NONLINEAR STIFF DIFFERENTIAL EQUATIONS

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

MASTER OF SCIENCE

in Mathematics

by
Neslişah İMAMOĞLU

December 2014
İZMİR

We approve the thesis of **Neslişah İMAMOĞLU**

Examining Committee Members:

Prof. Dr. Gamze TANOĞLU
Department of Mathematics, İzmir Institute of Technology

Prof. Dr. Emine MISIRLI
Department of Mathematics, Ege University

Assist. Prof. Dr. Olha IVANYSHYN YAMAN
Department of Mathematics, İzmir Institute of Technology

12 December 2014

Prof. Dr. Gamze TANOĞLU
Supervisor, Department of Mathematics
İzmir Institute of Technology

Prof. Dr. Oğuz YILMAZ
Head of the Department of
Mathematics

Prof. Dr. Bilge KARAÇALI
Dean of the Graduate School of
Engineering and Sciences

ACKNOWLEDGMENTS

This thesis appears in its current form due to the assistance and guidance of several people. I would like to offer my thanks to all of them.

I want to express my deep thanks to my advisor,Prof. Dr. Gamze Tanoğlu, for her guidance, caring and patience during my research to prepare my thesis.

I would like to thank Melek Sofyalıoğlu for her advices and helps. Many thanks to Yeşim Çiçek, Sıla Övgü Korkut, Gizem Kafkas and Damla Isıdıcı for providing a good atmosphere in our department and for useful discussions.

I owe the same amount of thanks to Ömer Karabaş for being constantly proud of me without any reason.

I warmly thank and appreciate my parents. They kept encouraging me when I encountered difficulty during the writing process of this thesis. Without their constant support, this accomplishment wouldn't have been made possible.

ABSTRACT

TWO NUMERICAL APPROACHES FOR SOLVING NONLINEAR STIFF DIFFERENTIAL EQUATIONS

This thesis presents two different numerical methods to solve non-linear stiff differential equations. The first method is exponential integrator, its error bounds are derived for the specific differential equations. Error analysis of exponential integrators is studied based on the Frèchet differentiation and Sobolev space. We obtain the error bounds in $H^s(\mathbb{R})$ norms under the certain assumptions. The second method is a new iterative linearizaton technique. For the second one, we first time applied to general Frèchet derivative as a linearization technique for the numerical solution of nonlinear partial differential equations. In computational part, in order to denote the effectiveness of the new proposed method, we compare our proposed method with the well-known techniques with respect to the errors.

ÖZET

DOĞRUSAL OLMAYAN SERT DİFERANSİYEL DENKLEMLERİ ÇÖZMEK İÇİN İKİ SAYISAL YAKLAŞIM

Bu tezde doğrusal olmayan sert diferansiyel denklemleri çözmek için iki farklı sayısal yöntem sunulmaktadır. İlk yöntem üstel integratördür, bu yöntemin hata sınırları özel diferansiyel denklemler için elde edilmiştir. Üstel integratörlerin hata analizi, Frèchet türeve ve Sobolev uzaylarına dayanmaktadır. Hata sınırlarını, gerekli kabuller altında $H^s(\mathbb{R})$ normunda elde ettik. İkinci yöntem yeni tekrarlı doğrusallaştırma tekniğidir. İkinci yöntemde, genel Frèchet türevini ilk kez doğrusal olmayan kısmi türevli diferansiyel denklemlerin sayısal çözümleri için doğrusallaştırma tekniği olarak uyguladık. Hesaba dayalı bölümde, yeni tasarlanan yöntemin etkililiğini göstermek için, kendi sunduğumuz yöntemi, iyi bilinen tekniklerle hatalarına göre kıyasladık.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

CHAPTER 1

INTRODUCTION

The aim of this thesis is to obtain approximated solutions of nonlinear stiff differential equations numerically. Stiff problems can be defined as: problems which can not be solved by the classical methods. These types of problems arise in various fields of science and engineering such as fluid mechanics, physics, chemical reactor theory, convection diffusion processes and other branches of applied mathematics.

We will deal with firstly the history of stiff differential equations. The earliest detection of stiffness in differential equations presented by two chemists Curtiss and Hirschfelder in 1952. They named the term of stiffness. They also gave the definiton of the stiffness as:" *Stiff equations are equations where certain implicit methods perform better, usually tremendous better, than explicit ones.*" (Curtis & Hirschfelder, 1950).

The second development was defined by Dahlquist in 1963. He dealed with the problems in stability. He said in Aiken (1985) that "... *around 1960 , thing became completely different and everyone became aware that the world was full of stiff problems.*" (Hairer & Wanner, 2000). Dahlquist is also said that the problem is stiff if "*Systems containing very fast components as well as very small component.* " (Dahlquist, 1963).

In 1968, Gear became one of the most important names in this area. Gear and Shampine presented an article in 1979. The aim of this article was to aid people who needs to solve stiff ordinary differential equations. They identified the problem area and described the characteristics shared by methods for the numerical solution of stiff problems (Shampine & Gear, 1976).

In 1970, Liniger designed efficient algorithms for solving stiff systems of ordinary differential equations (Liniger & Willoughby , 1970). In 1973, Lambert examined critically various qualitative statements including the notion of stiffness. One of them is 'A linear constant coefficient system is stiff if all of its eigenvalues have negative real part and the stiffness ratio is large.' This statement is adopted as a definition of stiffness. He selected the most satisfactory of these statements as a 'definition' of stiffness. This is: 'If a numerical method with a finite region of absolute stability , applied to a system with any initial condition, is forced to use in a certain interval of integration a step lentgh which is excessively small in relation to the smoothness of the exact solution in that interval, then the system is said to be stiff in that interval' (Lambert, 2000). Lambert thought that the rate of stiffness in real life problems would become more important year by year.

Brugnano and Trigiante in 1996 gave a definition of stiffness based on conditioning (Burgnano& Trigiante, 1996). In 1996, Spijker reviewed various aspects of stiffness in the numerical solution of initial value problems for systems of ordinary differential equation (Spijker, 1995).

The excellent text which is Solving Ordinary Differential Equations II of Hairer & Wanner has helped put this theory on a firm basis. They also gave a definition of stiffness as follows: *"Stiff equations are problems for which explicit methods don't work."* (Hairer & Wanner, 2000).

Numerical solutions of this type of equations are an important area in recent years. Various techniques are developed over the years. Implicit schemes have an advantage to solve the stiff differential equations. Because of the freedom of choice of the time step. When trying to solve nonlinear equations with an implicit methods, most of the methods have difficulties. To avoid these difficulties, generally combination of the explicit and implicit methods is used. The strategy of this combination is that, nonlinear part of the problem is solved by the explicit multi-step methods and linear part of the problem is solved by the implicit methods. The name of this strategy is Implicit-Explicit (IMEX) schemes. These schemes were presented to solve stiff PDEs in the late 1970's (Varah).

Another important scheme is to solve the stiff PDEs the method of lines. In this methods, first the spatial derivatives of a PDE are discretized with approximation method. Then any well-known numerical method is applied to obtain the numerical solution of the problem. These techniques include Finite Difference Formulas and Spectral Methods (Trefethen).

Beylkin constructed implicit and explicit schemes of arbitrary order, which they called Exact Linear Part (ELP) method in (Beylkin & Keiser & Vozovoi). Later, Cox and Matthews give a open derivation of the ELP schemes, which they called Exponential Time Differencing (ETD) methods (Cox& Matthews , 2002). Then, Hochbruck and Ostermann developed the Exponential Integrators (Hochbruck & Ostermann, 2010).

This thesis consists of two part. In the first part, we deal with the exponential integrators. To understand the idea of the exponential integrators, we give an historical background. The first idea of this methods was in the study of Hersch in 1958. He realized that numerical solutions of differential equations can't give the solution exactly, even if the equation can be solved with analytical methods. Thus he introduced a new exact approach for constant coefficient linear problems. The multistep exponential time difference methods was developed by Certaine in 1960. To obtain this methods, he used the variation of constants formula and algebraic approximation of the nonlinearity. The beginning of the exponential integrators started with this. In 1963, Pope offered that the nonlinear part of the problem is linearized in every time step. This is the main idea of the Rosenbrock methods (Pope, 1963). In

1967, Lawson formulated the exponential Runge-Kutta methods firstly. In this methods, exponential functions was used as Runge-Kutta coefficients. (Lawson, 1967). In 1978, Fredli proposed the higher order methods. In these methods, linear part is solved exactly but approximation of the nonlinear part is solved by explicit methods.
In 1998 Exponential integrators was introduced by Hochbruck,Lubich and Selhofer. Therefore this work was the first efficient implementation of an exponential integrator (Hochbruck & Lubich & Selhofer , 1998). Higher order exponential Runge-Kutta methods was developed by Hochbruck and Ostermann in 2005. (Hochbruck & Ostermann & Schweitzer, 2008).

In the second part, we develop a new linearization technique to solve non-linear partial differential equations which is based on Frèchet derivative and Newton-Raphson method. The idea of this technique is used in the study of Liu (Liu & Wu, 2000) to solve ordinary differential equations of Duffing-type non-linearity. Then this technique is appeared in the study of (Fazel & Moghadam & Poshtan, 2013). In this study, they applied the technique to solve non-linear ordinary differential equations of motion. In this thesis, we first time applied this linearization technique to find the numerical solutions of nonlinear partial differential equations. We give the procedure thoroughly to convert nonlinear partial differential equations into a set of linear algebraic equations using the Frèchet derivative in Newton-Raphson iteration.

For numerical implementation we choose Allen-Cahn and Burgers' equations. We applied both exponential integrators and a new iterative linearization technique to these problems.

The outline of the thesis as follows: Chapter 2 gives the definitions that we use the other chapters. These definitions are about stiff differential equations, Frèchet derivative and Sobolev space and norms. Chapter 3 focuses on exponential integrators.In this chapter derivation of the method is shown and error bounds for Allen-Cahn and Burgers' equation are obtained. Chapter 4 concentrates on the iterative linearization technique that we first develop to solve nonlinear partial differential equations. Linearizing the operators by using Frèchet derivative and combining this with the Newton-Rapson method are introduced in this chapter. In Chapter 5, various numerical examples are illustrated to show that the methods are worked. We summarize and give brief conclusion in Chapter 6.

CHAPTER 2

BASIC CONCEPTS

In this chapter, basic concepts that are used in the next chapters are introduced. Firstly, the idea of stiffness is given. Then, Frèchet differentiability is defined to use in the error analysis of exponential integrators and to use as a tool in the iterative linearization technique. Finally, Sobolev space and norms are defined for Chapter 3.

2.1. Stiff Differential Equations

Differential equations divide into stiff and non-stiff differential equations. We will deal with the stiff differential equations. The definition of stiffness can be formalized as follows:

Definition 2.1 *A linear differential system*

$$u'(t) = Au(t) + f(t) \quad , \quad u(0) = u_0$$

where $A \in \mathbb{R}^{n\times n}$ *and* $u, f, u_0 \in \mathbb{R}^n$.
This system is said to be stiff if and only if

i) *For all* i , $\mathbb{R}(\lambda_i) < 0$,

ii) $\frac{max|\mathbb{R}(\lambda_i)|}{min|\mathbb{R}(\lambda_i)|} \gg 1$, *where* λ_i *are eigenvalues of* A *for* $i = 1, 2, ..., n$.

We called $\frac{max|\mathbb{R}(\lambda_i)|}{min|\mathbb{R}(\lambda_i)|}$ as stiffness ration. We will check the stiffness of given any equation the aid of this definition.
Now, we focus our attention on two examples to clarify the stiff differential equation. We first consider the linear ODE system:

$$\begin{aligned} u_1' &= -u_1 + \sin t, \;\; u_1(0) = 1, \\ u_2' &= 2u_1 - 100u_2, \;\; u_2(0) = 0, \end{aligned} \tag{2.1}$$

where $t \in [0, 0.3]$. We can rewrite equation (2.1) following matrix form,

$$u'(t) = \begin{bmatrix} -1 & 0 \\ 2 & -100 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} \sin t \\ 0 \end{bmatrix}, \quad u_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{2.2}$$

This system is equivalent to following form

$$u'(t) = Au(t) + f(t), \quad u(0) = u_0, \tag{2.3}$$

where $A \in \mathbb{R}^{2\times 2}$ and $u, f, u_0 \in \mathbb{R}^2$. Then, we can check the stiffness according to Definition (2.2). We have to find eigenvalues of the coefficient matrix. The eigenvalues of the given matrix A are $\lambda_1 = -100$ and $\lambda_2 = -1$. Both of the eigenvalues are negative and stiffness ratio=$\frac{|\lambda_2|}{|\lambda_1|} = 100 \gg 1$. So, linear equation system (2.1) is said to be stiff.
Our next example is the detection of stiffness in PDE. Let us consider the heat equation:

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

with the initial condition and the boundary conditions

$$u(x,0) = f(x), \tag{2.4}$$
$$u(0,t) = u(1,t) = 0, \tag{2.5}$$

where $x \in (0, 1]$, $t \in (0, T]$.
We will solve the diffusion problem using the finite difference method. The basic idea of the method is to replace the spatial derivation in partial differential equation with algebraic approximation. We approximate the spatial partial derivative of u_{xx} using the central difference formula. The approximate solution of $u(x, t)$ at $x = x_n$ is denoted by $u_n(t)$

$$\begin{aligned} \frac{\partial u_n(t)}{\partial t} &= \frac{u_{n+1}(t) - 2u_n(t) + u_{n-1}(t)}{(\Delta x)^2}, \\ u_0(t) &= u_N(t) = 0 \ , \ t \in (0, T], \\ u_n(0) &= f(x_n) \ , \ n = 1, ..., N-1, \end{aligned} \tag{2.6}$$

where $\Delta x = \frac{1}{N}$ and $x_n = n\Delta x$, $n = 1, ..., N-1$. Suppose that the step size $\Delta x = h$. It is convenient to write (2.4) in matrix form

$$\begin{bmatrix} u_1' \\ u_2' \\ \vdots \\ u_{N-1}' \end{bmatrix} = \frac{1}{h^2} \begin{bmatrix} -2 & 1 & & & & & \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & 1 & \ddots & \ddots & & \\ & & & \ddots & \ddots & 1 & \\ & & & & & -2 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \end{bmatrix}.$$

This system can be written as

$$u'(t) = Au(t), \;\; u(0) = u_0, \tag{2.7}$$

where $u_0 = [f(x_1), ..., f(x_{N-1})]^T$ is the initial condition. Here we consider the periodic boundary conditions. Boundary conditions are embedded into the matrix. To examine the stiffness of the given diffusion problem, we need to find the eigenvalues of A. These eigenvalues are real and can be given by the following equality

$$\lambda_j = -\frac{4}{h^2} sin^2 \frac{j\pi}{2N}, \quad 1 \leq j \leq N-1.$$

The proof can be obtained by showing a relationship between the characteristic polynomial for A and Chebyshev polynomials

$$\lambda_{N-1} \leq \lambda_j \leq \lambda_1, \tag{2.8}$$

with the following results for λ_{N-1} and λ_1

$$\begin{aligned} \lambda_{N-1} &= -\frac{4}{h^2} sin^2 \frac{(N-1)\pi}{2N} \approx -\frac{4}{h^2}, \\ \lambda_1 &= -\frac{4}{h^2} sin^2 \frac{\pi}{2N} \approx -\pi^2, \end{aligned}$$

with the approximations available for larger N. Let r be an eigenvector with corresponding eigenvalue λ for A.
For a vector $r = (r_1, r_2, ..., r_{N-1})$ to be an eigenvector for A with corresponding eigenvalue λ we must have

$$\frac{1}{h^2}(r_{n-1} - 2r_n + r_{n+1}) = \lambda r_n, \quad n = 2, ..., N-2, \tag{2.9}$$

It is usually not easy to find eigenvalues and eigenvectors for such a matrix, but if we have a prediction then it is very easy to check whether it fits or not. A good suggestion for r can be to take

$$r_n = sin(n\theta), \quad n = 1, ..., N-1, \tag{2.10}$$

we work out

$$\begin{aligned} r_{n-1} + r_{n+1} &= sin(n-1)\theta + sin(n+1)\theta, \\ &= 2sin(n\theta)cos(\theta), \\ &= 2r_n cos\theta. \end{aligned} \tag{2.11}$$

When we replace (2.11) in (2.9), this gives

$$\begin{aligned} \lambda &= \frac{1}{h^2}(-2 + 2cos(\theta)), \\ &= \frac{-4}{h^2} sin^2(\theta/2). \end{aligned}$$

In addition to the $N-3$ equations we must also have the similar relations for $n = 1$ and $n = N-1$:

$$\begin{aligned} 2r_1 - r_2 &= \lambda r_1, \\ -r_{N-2} + 2r_{N-1} &= \lambda r_{N-1}. \end{aligned}$$

These are fulfilled automatically if we can manage to have $r_0 = r_N = 0$. For r_N we must require

$$r_N = sin(N\theta) = 0. \tag{2.12}$$

Note that the roots are

$$N\theta_j = j\pi, \quad j = 1, 2, ... \tag{2.13}$$

We therefore define

$$\theta_j = \frac{j\pi}{N}, \quad j = 1, 2, ..., N-1, \tag{2.14}$$

and with these $N-1$ values of θ we have a set of $N-1$ orthogonal eigenvectors and corresponding eigenvalues for A:

$$\lambda_j = -4sin^2\Big(\frac{j\pi}{2N}\Big). \tag{2.15}$$

Directly examining this formula,

$$\lambda_{N-1} \leq \lambda_j \leq \lambda_1. \tag{2.16}$$

The least and the largest eigenvalues are can be obtained by using (2.15)

$$\lambda_{N-1} = \Big(\frac{-4}{h^2}\Big)sin^2\Big(\frac{(N-1)\pi}{2N}\Big) = \Big(\frac{-4}{h^2}\Big), \tag{2.17}$$

$$\lambda_1 = \Big(\frac{-4}{h^2}\Big)sin^2\Big(\frac{\pi}{2N}\Big) = -\pi^2. \tag{2.18}$$

Finally, the proportion of equations is obtained

$$\frac{\lambda_{N-1}}{\lambda_1} \approx \frac{4}{(\pi h)^2}, \tag{2.19}$$

it can be seen that it is a stiff system if h is small. As a consequence, these two examples show that the stiffness of the differential equations can be identified by the Definition (2.2).

2.2. Differentiation in Banach Space

In this section, we take the differentiability of general operators in Banach spaces into consideration. We will give the definition of the Fréchet derivative and the Taylor's formula.

2.2.1. Fréchet Derivative

Recall the definition of derivative for real valued functions:

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

if the limit exists. Writing

$$y = \begin{cases} \frac{f(x+h)-f(x)}{h} - f'(x) & h \neq 0 \\ 0 & h = 0 \end{cases}$$

we see that the definition of derivative implies that $\psi(h)$ is a continuous function at 0, while it is clearly a continuous function elsewhere (as the differentiable function f is continuous). Moreover we have the equation

$$f(x+h) = f(x) + f'(x)h + h\psi(h) \tag{2.20}$$

We can now generalize this idea to obtain a more general definition of derivative. Notice that we need that domain and range of f are normed vector spaces, otherwise we can't add (if we don't have a vector space) or talk about continuity (if we don't have norms). The derivative defined in this way, the usual definition on general vector spaces, is called Fréchet derivative.

Definition 2.2 *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*

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

for all $h \in V$.(note V must be chosen such that $x + V = x + v|v \in V \subset U$.)

2.2.2. Gateaux Derivative

The Gateaux differential generalizes the idea of a directional derivative. If f is Frèchet differentiable, then it is also Gateaux differentiable. Gateaux derivative definition is given with the following lemma.

Lemma 2.1 *If $f : U \to Y$ differentiable at x, then for all $h \in X$ we have*

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

where t is chosen in $\mathbb{R}$.

Proof Let us assume $t > 0$ (for $t < 0$ the argument is the same). By definition of derivative we have for t small enough (so $th \in V$)

$$f(x + th) = f(x) + (Df(x))(th) + ||th||\psi(th),$$

or by rearranging and using linearity

$$(Df(x))h = \frac{f(x + th) - f(x)}{t} - ||h||\psi(th)$$

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

Now we can take the limit as $t \to 0$ on both sides (as the left hand side is constant it has a limit) to get the desired result (note $\lim_{t\to 0} ||h||\psi(th) = ||h||\psi(0h) = 0$ as ψ is continuous.) □

2.2.3. 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 2.3 *Let $u \in U$: f is twice (Fréchet) differentiable at u. The second (Fréchet) differential of f at u is defined as*

$$d^2 f(u) = df'(u). \tag{2.21}$$

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 $L(x, Y)$:

$$d^2 f(u) \in L(X, L(X, Y)). \tag{2.22}$$

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 $X \to L(X, Y)$,

$$\Phi : h \to \Phi(h, .) \in L(X, Y) \tag{2.23}$$

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

$$||\Phi||_{L(X,L(X,Y))} = \sup_{||h||\leq 1} ||\Phi(h)||_{L(X,Y)} \tag{2.24}$$

$$= \sup_{||h||\leq 1} \sup_{||k||\leq 1} ||\Phi(h, k)|| = ||\Phi(h, k)||_{L_2(X,Y)} \tag{2.25}$$

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

$$d^2 f(u)[h, k]. \tag{2.26}$$

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

$$f'' : u \to d^2 f(u). \tag{2.27}$$

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

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

If $h = h_1 = ... = h_n$ we will write for short $d^n f(u)[h]^n$.

2.2.4. Taylor's Formula

Let $f \in C^n(Q, Y)$ and let $u, u+v \in Q$ be such that the interval $[u, u+v] \subset Q$.Then, Taylor's formula for Fréchet differentiable maps is that

$$f(u+v) = f(u) + df(u)[v] + ... + \frac{1}{(n-1)!}\int_0^1 (1-t)^{n-1} d^{(n)} f(u+tv)\ dt[v]^n.$$

The last integral can be written as

$$\frac{1}{(n-1)!}\int_0^1 (1-t)^{n-1} d^{(n)} f(u+tv)\ dt[v]^n = \frac{1}{n!} d^{(n)} f(u)[v]^n + \varepsilon(u,v)[v]^n$$

where

$$\varepsilon(u,v) = \frac{1}{(n-1)!}\int_0^1 (1-t)^{n-1}[d^n f(u+tv) - d^n f(u)]\ dt \to 0 \ \ as \ \ v \to 0.$$

2.3. Sobolev Spaces

We interested in the Sobolev spaces which forms a Hilbert space.These spaces are denoted as $H^s(\mathbb{R}) = W^{s,2}(\mathbb{R})$, where s is integer. The inner product and norm are defined as

$$(u,v)_{H^s} = \sum_{j=0}^{s} \int_{\mathbb{R}} \partial_x^j u(x) \partial_x^j v(x) dx \text{ and } \|u\|_{H^s} = \sqrt{(u,u)_{H^s}}. \tag{2.28}$$

We see that $H^s(\mathbb{R})$ contains all functions which has weak derivatives up to order s in $L^2(\mathbb{R})$, and we remark that $L^2(\mathbb{R}) = H^0(\mathbb{R})$.

2.3.1. Standard Sobolev Spaces

Consider $H^s(\mathbb{R})$ defined when s is a positive integer, with inner product and norm as in (2.28). From the definition, we observe that $H^r(\mathbb{R})$ is continously imbedded in $H^s(\mathbb{R})$ for

$r > s$, which results in that the respective norms are comparable in the following way

$$\|u\|_{H^s} \leq C\|u\|_{H^r}, \tag{2.29}$$

for u in $H^r(\mathbb{R})$. We first show that $H^s(\mathbb{R})$ is imbedded in $L^\infty(\mathbb{R})$ for $s \geq 1$.

Lemma 2.2 *The space $H^s(\mathbb{R})$ is a Banach algebra for $s \geq 1$. In particular, if u, v are in $H^s(\mathbb{R})$ for $s \geq 1$, then*

$$\|uv\|_{H^s} \leq C_s\|u\|_{H^s}\|u\|_{H^s},$$

where where C_s depends only on s.

Proof Since the Sobolev norm is a sum of (weak) derivatives of u and v, it is sufficient to show that for all $r \leq s$

$$\|\partial_x^r(uv)\|_{L^2} \leq C_s\|u\|_{H^s}\|u\|_{H^s}.$$

Consider $\partial_x^r(uv)$ and expand it using Leibniz rule

$$\partial_x^r(uv) = \sum_{j=0}^{r} \binom{r}{j} \partial_x^j u \partial_x^{r-j} v.$$

By the triangle inequality it is sufficient to look at one term in the above sum. Moreover, we need to be careful in the estimation of the term, since when we vary j and s we get different orders of the derivatives on u and v, which is not necessarily bounded in $H^s(\mathbb{R})$. However, we get for $r < s$ and $0 \leq j \leq r$

$$\begin{aligned} \|\partial_x^j u \partial_x^{r-j} v\|_{L^2}^2 &= \int_{-\infty}^{\infty} (\partial_x^j u)^2 (\partial_x^{r-j} v)^2 dx \leq \|\partial_x^j u\|_{L^\infty}^2 \int_{-\infty}^{\infty} (\partial_x^{r-j} v)^2 dx \\ &\leq C_s\|u\|_{H^{j+1}}^2 \|\partial_x^{r-j} v\|_{L^2}^2 \leq C_s\|u\|_{H^s}^2\|v\|_{H^s}^2 \end{aligned}$$

since $j+1 \leq r+1 \leq s$ and $r-j \leq s$. For $r=s$ and $0 \leq j < r$ we get, using same technique as above

$$\|\partial_x^j u \partial_x^{s-j} v\|_{L^2}^2 \leq C_s \|u\|_{H^s}^2 \|v\|_{H^s}^2. \tag{2.30}$$

we are left with one case; when $r=s=j$,

$$\|\partial_x^s uv\|_{L^2}^2 = \int_{-\infty}^{\infty} (\partial_x^s u)^2 (v)^2 dx \leq \|v\|_{L^\infty}^2 \|\partial_x^s u\|_{L^2}^2 \leq C_s \|u\|_{H^s}^2 \|v\|_{H^s}^2.$$

By taking the square root of the above estimates, and summing up all the derivatives, we get

$$\|uv\|_{H^s} \leq C_s \|u\|_{H^s} \|v\|_{H^s},$$

and the lemma is proven. □

CHAPTER 3

EXPONENTIAL INTEGRATORS

In this chapter, we take a look a class of numerical integrators for time integration stiff systems which called Exponential Integrators. A reason to use exponential integrators is to overcome the problem of the stiffness in applications. Application of exponential integrators can be seen in different areas. More common usage of exponential integrators is in applied mathematics and physics. For example, reaction-diffusion equations, Schrödinger equations, Maxwell equations can be solved by exponential integrators (Kandolf, 2011).

3.1. Derivation of the Method

In this section, we will give the brief survey about what an exponential integrator look like. Consider the equation of the form,

$$u'(t) = F(t, u(t)), \quad u(0) = u_0. \tag{3.1}$$

Linearizing the equation (3.1) at time t, gives the semilinear problem

$$u'(t) + Au(t) = B(t, u(t)) \tag{3.2}$$

where $A = -DF(t, u(t))$ is the Jacobian of F and $B(t, u(t)) = F(t, u(t)) + Au(t)$ is the reminder. The linear part of the equation (3.2),

$$u'(t) + Au(t) = 0, \quad u(0) = u_0 \tag{3.3}$$

can be solved exactly by

$$u(t) = e^{-tA}u_0. \tag{3.4}$$

To obtain the exact solution of (3.2), multiply the equation (3.2) by integrating factor e^{tA}. This gives,

$$\begin{aligned} e^{tA}u' + e^{tA}Au &= e^{tA}B(t, u(t)), \\ (e^{tA}u)' &= e^{tA}B(t, u(t)). \end{aligned} \tag{3.5}$$

Integrating both sides of the equation (3.5) gives us following integral representation.

$$\begin{aligned} \int_0^t \frac{d}{d\tau}(e^{\tau A}u(\tau))d\tau &= \int_0^t e^{\tau A}B(\tau, u(\tau))d\tau \\ e^{tA}u(t) - u(0) &= \int_0^t e^{\tau A}B(\tau, u(\tau))d\tau \\ u(t) &= e^{-tA}u_0 + \int_0^t e^{-tA}e^{\tau A}B(\tau, u(\tau))d\tau \end{aligned} \tag{3.6}$$

The exact solution of the evaluation equation (3.6) is called variation of constants formula. For time steps $t_{n+1} = t_n + h$ the variation of constants formula becomes

$$u(t_{n+1}) = e^{-hA}u(t_n) + \int_0^h e^{-(h-\tau)A}B(t_n + \tau, u(t_n + \tau)). \tag{3.7}$$

with step size $h > 0$. Equation (3.7) represents the recursive exact solution. To obtain different numerical schemes, different quadrature formulas can be used for approximating the integral. Let choose $B(t) = c_1$ where c_1 is a constant. Then the solution becomes

$$\begin{aligned} u(t) &= e^{-tA}u_0 + \int_0^t e^{-(t-\tau)A}c_1 d\tau = e^{-tA}u_0 + e^{-tA}\frac{1}{A}e^{\tau A}c_1\Big|_0^t, \\ &= e^{-tA}u_0 + e^{-tA}\frac{e^{tA} - 1}{A}c_1 = e^{-tA}u_0 + t\frac{e^{-tA} - 1}{-tA}c_1. \end{aligned} \tag{3.8}$$

Then, take the $B(t) = c_1 + c_2 t$ as a first degree polynomial. Now the solution is

$$\begin{aligned} u(t) &= e^{-tA}u_0 + \int_0^t e^{-(t-\tau)A}(c_1 + c_2\tau)d\tau, \\ &= e^{-tA}u_0 + t\frac{e^{-tA} - 1}{-tA}c_1 + t^2\frac{e^{-tA} + tA - 1}{t^2A^2}c_2. \end{aligned} \tag{3.9}$$

As a rule, if the $B(t)$ is a n-th degree polynomial, the solution will be

$$\begin{aligned} u(t) &= e^{-tA}u_0 + \int_0^t e^{-(t-\tau)A}\left(c_1 + c_2 t + c_3\frac{t^2}{2!} + ... + c_n\frac{t^{n-1}}{(n-1)!}\right)d\tau \\ &= e^{-tA}u_0 + t\frac{e^{-tA}-1}{-tA}c_1 + t^2\frac{e^{-tA}-1+tA}{t^2A^2}c_2 + t^3\frac{e^{-tA}-1+tA-\frac{1}{2}tA^2}{-t^3A^3}c_3 \\ &+ ... + \int_0^t e^{-(t-\tau)A}c_n\frac{t^{n-1}}{(n-1)!}d\tau. \end{aligned} \tag{3.10}$$

In the next section, we will introduce the φ functions in order to generalize the result found in (3.10).

3.1.1. Derivation of the φ Functions

The integral representation of the φ function is

$$\begin{aligned} \varphi_0(z) &= e^z, \\ \varphi_k(z) &= \int_0^1 e^{(1-\theta)z}\frac{\theta^{k-1}}{(k-1)!}d\theta. \end{aligned} \tag{3.11}$$

The argument z can be a scalar or a matrix. The φ_i- functions are defined recursively by

$$\begin{aligned} \varphi_0(z) &= e^z, \\ \varphi_{k+1}(z) &= \frac{\varphi_k(z)-\frac{1}{k!}}{z}, \quad k \geq 0. \end{aligned} \tag{3.12}$$

In recursive formula, φ_0 is the matrix exponential. The first few φ-functions are

$$\varphi_0(z) = e^z = 1 + z + \frac{1}{2}z^2 + \frac{1}{3!}z^3 + \ldots \tag{3.13}$$

$$\varphi_1(z) = \frac{e^z-1}{z} = 1 + \frac{1}{2}z + \frac{1}{3!}z^2 + \frac{1}{4!}z^3 + \ldots \tag{3.14}$$

$$\varphi_2(z) = \frac{e^z-1-z}{z^2} = \frac{1}{2} + \frac{1}{3!}z + \frac{1}{4!}z^2 + \frac{1}{5!}z^3 + \ldots \tag{3.15}$$

$$\varphi_3(z) = \frac{e^z-1-z-\frac{1}{2}z^2}{z^3} = \frac{1}{3!} + \frac{1}{4!}z + \frac{1}{5!}z^2 + \frac{1}{6!}z^3 + \ldots \tag{3.16}$$

We further obtain following results with aid of the (3.13)-(3.16).

$$u(t) = e^{-tA}u_0 + t\varphi_1(-tA)c_1 \tag{3.17}$$

$$u(t) = e^{-tA}u_0 + t\varphi_1(-tA)c_1 + t^2\varphi_2(-tA)c_2 \tag{3.18}$$

$$u(t) = e^{-tA}u_0 + t\varphi_1(-tA)c_1 + t^2\varphi_2(-tA)c_2 + t^3\varphi_3(-tA)c_3 + ... + t^n\varphi_n(-tA)c_n \tag{3.19}$$

As a result, the solution (3.10) can be written in the following form

$$u(t) = e^{-tA}u_0 + \sum_{i=1}^{n} t^i\varphi_i(-tA)c_i. \tag{3.20}$$

3.1.2. Exponential Euler Method

In this section, we will use the representation (3.6) to obtain first order exponential integrator which is exponential Euler method. We first use the Taylor series expansion of $B(t_n+\tau, u(t_n+\tau))$ up to the order τ in equation (3.7). Just to be easy we take $B(t_n, u(t_n)) = B(t_n)$. Taylor polynomial of $B(t_n + \tau)$ at the point t_n is

$$B(t_n + \tau) = B(t_n) + O(\tau). \tag{3.21}$$

Inserting (3.21) into equation (3.7), and using the φ_1- function, the numerical solution will be

$$u(t_{n+1}) = e^{-hA}u(t_n) + h\varphi_1(-hA)B(t_n). \tag{3.22}$$

The equation (3.60) is called exponential Euler method.

3.1.3. Exponential Rosenbrock Euler Method

In this section, we will show the derivation of the exponential Rosenbrock Euler method. Consider the time discretization of differential equations in autonomous form

$$u'(t) = F(u(t)), \quad u(t_0) = u_0. \tag{3.23}$$

The method is based on a continuous linearization of (3.23). For a given point u_n, this linearization is

$$u'(t) \quad = \quad J_n u(t) + B_n(u(t)) \tag{3.24}$$

where

$$J_n = DF(u_n) = \frac{\partial F}{\partial u}(u_n), \quad B_n(u(t)) = F(u(t)) - J_n(u(t)). \tag{3.25}$$

Applying the exponential Euler method to (3.24), we procure

$$u_{n+1} \quad = \quad e^{hJ_n} u_n + h\varphi_1(hJ_n) B_n(u_n). \tag{3.26}$$

When we regulate the equation, we obtain

$$u_{n+1} \quad = \quad u_n + h\varphi_1(hJ_n) F(u_n).$$

The numerical scheme (3.26) gives the exponential Rosenbrock-Euler method. Exponential Rosenbrock-Euler method is explicit time stepping scheme. This method is computationally attractive since it involves just one matrix function in each step. To implement exponential Rosenbrock-Euler method it is important to approximate the application of matrix functions to vectors efficiently.

3.2. Convergence Analysis

Convergence analysis of exponential integrators is done by using semi-group theory in (Hochbruck & Ostermann, 2010) and (Hochbruck & Ostermann & Schweitzer, 2008). In this section, convergence analysis of exponential Euler method will be worked on problem-based in Sobolev space. We will use the same technique given in the strudy of (Holden & Lubich & Risebro). We will work on Allen-Cahn equation and Burgers' equation to find the bounds.
General form of the problem that we focus on is in the following form

$$u_t = P(\partial_x) + B(u), \quad u(t_0) = u_0 \tag{3.27}$$

with a polynomial P of degree $l \geq 2$ satisfying

$$\mathbb{R}(P(i\xi)) \leq 0 \quad \text{for all} \quad \xi \in \mathbb{R}. \tag{3.28}$$

To obtain error bounds for this type equation, some hypothesis and lemmas help us. We start with the following hypothesis which is related to well-posedness of the solutions.

Hypothesis 3.1 *(Local well-posedness). For a fixed time T, there exist $R \leq 0$ such that for all u_0 in $H^m(\mathbb{R})$ with $\|u_0\|_{H^k} \leq R$, there exist a unique strong solution u in $C([0,T], H^k)$ of (3.27). In addition, for the initial data u_0 there exists a constant $K(R,T) < \infty$, such that*

$$\|\tilde{u}(t) - u(t)\|_{H^k} \leq K(R,T)\|\tilde{u}_0 - u_0\|_{H^k} \tag{3.29}$$

for two arbitrary solutions u and $\tilde{u}$, corresponding to two different initial data u_0 and $\tilde{u}_0$.

Next hypothesis is concerned about the boundedness of the solutions.

Hypothesis 3.2 *(Boundedness). The solution $u(t)$ and the initial data u_0 of (3.27) are both in $H^k(\mathbb{R})$, and are bounded as*

$$\|u(t)\|_{H^k} \leq R < \rho \quad \text{and} \quad \|u_0\|_{H^k} \leq C < \infty$$

for $0 \leq t \leq T$.

Last hypothesis is associated with the differentiability of the solutions.

Hypothesis 3.3 *(Differentiable). Assume that sufficiently smooth solution $u(t)$ with derivatives in $H^k(\mathbb{R})$ and $B(u)$ is sufficiently often Fréchet-differentiable. All occuring derivatives are assumed to be bounded.*

In order to show the linear part of the problem is bounded, we need to following lemma.

Lemma 3.1 *Let P be a linear polynomial of degree $l \geq 2$ with constant coefficients, which satisfies*

$$\mathbb{R}(P(i\xi)) \geq 0 \quad for \ \ all \quad \xi \in \mathbb{R}. \tag{3.30}$$

In addition, let m be a integer such that $m \geq l$, and assume v_0 is in $H^{k+l}(\mathbb{R})$ and the solution $v(t) = \Phi_A^t(v_0) = e^{At}v_0$ of linear part is in $H^k\mathbb{R}$ and satisfies

$$\int_{\mathbb{R}} (\partial_x^{j+l/2} v)^2 dx < \infty$$

for all $j \leq m$ and l even. Then $\Phi_A^t(v_0)$ has a non-increasing norm in $H^k(\mathbb{R})$, in particular

$$\|\Phi_P^t(v_0)\|_{H^k} \leq \|v_0\|_{H^{k+l}}. \tag{3.31}$$

Proof P is given as $P(x) = \sum_{\alpha=2}^{l} a_\alpha x^\alpha$, where α is in $\mathbb{R}$ for all α. Let consider the following equation for the linear part,

$$v_t = P(\partial x)v. \tag{3.32}$$

Substituting polynomial into equation (3.32), (3.32) becomes

$$v_t = a_l \partial_x^l v + a_{l-1}\partial_x^{l-1} v + ... + a_2 \partial_x^2 v.$$

The time evolution of $\Phi_A^t(v_0)$ is given as

$$\frac{1}{2}\|\Phi_A^t(v_0)\|_{H^m}^2 = (v, v_t)_{H^m} = \sum_{j=0}^{m} \int_{\mathbb{R}} \partial_x^j v (a_l \partial_x^{j+l} + ... + a_2 \partial_x^{j+2}) v dx. \tag{3.33}$$

It is sufficient to estimate one general term in the above sum, say

$$\sum_{j=0}^{m} \int_{\mathbb{R}} \partial_x^j v a_l \partial_x^{j+l} v = a_l \sum_{j=0}^{m} \int_{\mathbb{R}} \partial_x^j v \partial_x^{j+l} v.$$

By partial integration the above equation turns into

$$\begin{aligned} a_l \sum_{j=0}^{m} \int_{\mathbb{R}} \partial_x^j v \partial_x^{j+l} v dx &= a_l \sum_{j=0}^{m} \Big([\partial_x^j v \partial_x^{j+l-1} v]_{-\infty}^{\infty} - \int_{\mathbb{R}} \partial_x^{j+1} v \partial_x^{j+l-1} v dx \Big) \\ &= -a_l \sum_{j=0}^{m} \int_{\mathbb{R}} \partial_x^{j+1} v \partial_x^{j+l-1} v dx \Big), \end{aligned}$$

where we have used that the derivatives on v of order up to m decay to zero when $x \to \pm\infty$. Performing partial integration together with the decay property for the derivatives of v subsequently, we get if l even

$$a_l \sum_{j=0}^{m} \int_{\mathbb{R}} \partial_x^j v \partial_x^{j+l} v dx = a_l \sum_{j=0}^{m} (-1)^{l-1} \int_{\mathbb{R}} \partial_x (\partial_x^{j+l/2} v)^2 dx = -a_l \sum_{j=0}^{m} \int_{\mathbb{R}} (\partial_x^{j+l/2} v)^2 dx$$

By the property given in (3.30), the coefficient a_l is such that the right-hand-side of the above equation is negative, that is $a_l > 0$. We write this for simplicity as

$$a_l \sum_{j=0}^{m} \int_{\mathbb{R}} \partial_x^j v \partial_x^{j+l} v dx = -a_l \sum_{j=0}^{m} \int_{\mathbb{R}} (\partial_x^{j+l/2} v)^2 dx = -a_l \|\partial_x^{l/2} v\|_{H^m}. \tag{3.34}$$

If l is odd, we obtain by partial integration

$$\begin{aligned} a_l \sum_{j=0}^{m} \int_{\mathbb{R}} \partial_x^j v \partial_x^{j+l} v dx &= a^l \sum_{j=0}^{m} (-1)^l \int_{\mathbb{R}} \partial_x (\partial_x^{j+(l-1)/2} v)^2 dx \\ &= -a_l \sum_{j=0}^{m} [(\partial_x^{j+(l-1)/2} v)^2]_{-\infty}^{\infty} = 0. \end{aligned} \tag{3.35}$$

By using the estimates in (3.34) and (3.35), we get for (3.33),

$$\frac{1}{2} \|\Phi_A^t(v_0)\|_{H^m}^2 = -C \|\partial_x^{l/2} v\|_{H^m} \leq 0,$$

where C is a constant. Solving the differential equation gives

$$\|\Phi_A^t(v_0)\|_{H^m} \leq \|v_0\|_{H^{m+l}}. \tag{3.36}$$

(Nilsen, 2011) □

3.2.1. Convergence Analysis of Allen-Cahn Equation

Our focus in this section is Allen-Cahn equation which is

$$u_t - u + u^3 = \gamma u_{xx} \ , \ u(t_0) = u_0 \tag{3.37}$$

If we separate the equation (3.37) as in the general form (3.27), this gives us

$$P(\partial_x)u = u_{xx} \tag{3.38}$$

and

$$B(u) = u - u^3 \tag{3.39}$$

where (3.38) is linear operator and (3.39) is the non-linear operator. Applying the exponential Euler method, we obtain the approximate solution as in the following form.

$$u_{n+1} = e^{hA}u_n + h\varphi_1(hA)B(u_n) \tag{3.40}$$

where φ_1 is given in equation (3.14). To obtain error bounds, hypothesis (3.1), hypothesis (3.2), hypothesis (3.3) and lemma (3.1) are used.

3.2.2. Local Error

In this section, we estimate the local error for exponential Euler method under the certain assumptions which are presented in previous section. We summarize the result with

the following lemma.

Lemma 3.2 *Let $s \geq 1$ be an integer and assume Hypothesis (3.2) and Hypothesis (3.3) hold for $k = s + l$ for the solution $u(t) = \Phi^t(u_0)$ of (3.37). If the initial data u_0 is in $H^{s+l}(\mathbb{R})$, then the local error of the exponential Euler method is bounded in H^s by*

$$\|\Psi^h(u_0) - \Phi^h(u_0)\|_{H^s} \leq c_1 h^2, \tag{3.41}$$

where c_1 depends on $\|u_0\|_{H^{s+l}}$ and where h is small time step.

Proof Let $\Phi_A^t(v) = e^{tA}v$ where A is the linear flow. We start from the variation-of-constant formula for $\Phi^h(u_0)$

$$u(t) = e^{tA}u_0 + \int_0^t e^{(t-s)A}B(u(s))\ ds. \tag{3.42}$$

Taylor expansion of $B(u(s))$ is

$$B(u(s)) = B(u(0)) + \int_0^s dB(u(\sigma))[B(u(\sigma))]\ d\sigma. \tag{3.43}$$

Take $t = h$ and insert (3.43) into (3.42),

$$u(h) = e^{hA}u_0 + \int_0^h e^{(h-s)A}B(u_0)\ ds + \delta, \tag{3.44}$$

where

$$\delta = \int_0^h \int_0^s e^{(h-s)A}dB(u(\sigma))[B(u(\sigma))]\ d\sigma ds. \tag{3.45}$$

One step exponential-Euler (3.60) is as follows

$$u_1 = e^{hA}u_0 + h\varphi_1(hA)B(u_0). \tag{3.46}$$

The error between the exact and the exponential-Euler solution is,

$$\begin{aligned} u_1 - u(h) &= h\varphi_1(hA)B(u_0) - (-\frac{I - e^{hA}}{A}B(u_0)) + \delta, \\ &= \delta. \end{aligned} \tag{3.47}$$

We continue with the error bound for δ in (3.47). We use the Banach algebra propety of $H^s(\mathbb{R})$ at each step to obtain the estimation.

$$\begin{aligned} \|\delta\|_{H^s} &\leq \int_0^h \int_0^s \|e^{(h-s)A} dB(u(\sigma))[B(u(\sigma))]\|_{H^s} d\sigma ds \\ &\leq \int_0^h \int_0^s \|dB(u(\sigma))[B(u(\sigma))]\|_{H^s} d\sigma ds \\ &\leq \int_0^h \int_0^s \|((u(\sigma))(B(u(\sigma))))_x\|_{H^s} d\sigma ds \\ &\leq \int_0^h \int_0^s \|(u(\sigma))(B(u(\sigma)))\|_{H^{s+1}} d\sigma ds \\ &\leq C \int_0^h \int_0^s \|(u(\sigma)\|_{H^{s+1}} \|B(u(\sigma))\|_{H^{s+1}} d\sigma ds. \end{aligned}$$

Using the definition of B

$$\begin{aligned} \|\delta\|_{H^s} &\leq C \int_0^h \int_0^s \|(u(\sigma)\|_{H^{s+1}} \|(u(\sigma)(1 - u^2(\sigma))\|_{H^{s+1}} d\sigma ds \\ &\leq C \int_0^h \int_0^s \|(u(\sigma)\|_{H^{s+1}} (\|(u(\sigma)\|_{H^{s+1}} + \|u^3(\sigma)\|_{H^{s+1}} d\sigma ds \\ &\leq C \int_0^h \int_0^s \|(u(\sigma)\|_{H^{s+1}} (\|u^3(\sigma)\|_{H^{s+1}} d\sigma ds. \end{aligned}$$

Using the *Hypothesis 2* for $H^{s+l}(\mathbb{R})$, which gives that

$$\|u(\sigma)\|_{H^{s+1}} \leq \|u(\sigma)\|_{H^{s+l}} \leq R,$$

when $l \geq 2$ which results in

$$\|\delta\| \leq C \int_0^h \int_0^s R^4 d\sigma ds = CR^4 \int_0^h s ds = CR^3 h^2 = c_1 h^2. \tag{3.48}$$

This completes the proof. □

3.2.3. Global Error

To estimate the global error in $H^s(\mathbb{R})$ and obtain the convergence rate for exponential Euler method, we use the local error estimate in Lemma (3.2). We need to show that exponential Euler solution is bounded at each time step, so that local error estimate is valid. The global error estimation is given with the following theorem.

Theorem 3.1 *Assume there exists a solution of (3.37) and let $s \geq 1$ be an integer. If Hypothesis (3.1) and Hypothesis (3.3) hold for $k = s$ and Hypothesis (3.2) holds for $k = s + l$ for $l \geq 2$, then there exists $\overline{h} > 0$ such that for all $h \leq \overline{h}$ and $t_n = nh \leq T$,*

$$\|u_n - u(t_n)\|_{H^s} \leq Ch,$$

where u_n is the exponential Euler solution.

Proof We denote the notational convention as in (Holden & Lubich & Risebro)

$$u_k^n = \Phi^{n-kh}(u_k)$$

which is the exact solution of 3.37. With this notation, we note

$$u_n = u_n^n \quad \text{and} \quad u(t_n) = u_n^0.$$

We make the induction by assuming that

$$\begin{aligned} \|u_k\|_{H^s} &\leq R, \\ \|u_k - u(t_k)\|_{H^s} &\leq \gamma h. \end{aligned}$$

The global error can be obtained by using Telescope sum and the triangle inequality as follows

$$\begin{aligned} \|u_n - u(t_n)\|_{H^s} &= \|u_n^n - u_n^0\|_{H^s} = \|u_n^n - u_n^{n-1} + u_n^{n-1} - u_n^{n-2} + u_n^{n-2} - ... - u_n^0\|_{H^s}, \\ &= \Big\| \sum_{k=0}^{k+1} u_n^{k+1} - u_k^n \Big\|_{H^s} \leq \sum_{k=0}^{k+1} \|u_n^{k+1} - u_k^n\|_{H^s}. \end{aligned} \tag{3.49}$$

By using notational convention (3.49) becomes

$$\begin{aligned} \|u_n - u(t_n)\|_{H^s} &\leq \sum_{k=0}^{n-1} \|\Phi^{(n-k-1)h}(u_{k+1}) - \Phi^{(n-k)h}(u_k)\|_{H^s}, \\ &= \sum_{k=0}^{n-1} \|\Phi^{(n-k-1)h}(\Pi^h u_k) - \Phi^{(n-k-1)h}(\Phi^h(u_k))\|_{H^s}. \end{aligned} \tag{3.50}$$

For $k \leq n-2$ we get

$$\left\|\Pi^h u_k\right\|_{H^s} = \|u_{k+1}\|_{H^s} \leq R, \tag{3.51}$$

and

$$\begin{aligned} \left\|\Phi^h(u_k)\right\|_{H^s} &= \left\|\Phi^h(u_k) - \Phi^h(u(t_k)) + \Phi^h(u(t)_k)\right\|_{H^s}, \\ &\leq \left\|\Phi^h(u_k) - \Phi^h(u(t_k))\right\|_{H^s} + \left\|\Phi^h(u(t)_k)\right\|_{H^s}. \end{aligned} \tag{3.52}$$

By using the Lipschitz continuity, (3.52) turns to

$$\left\|\Phi^h(u_k)\right\|_{H^s} \leq K(R,h)\|u_k - u(t_k)\|_{H^s} + \|u(t_{k+1})\|_{H^s} \leq K(R,h)\gamma h + \rho, \tag{3.53}$$

from the assumption of the induction. Let choose $K(R,h)\gamma h = R - \rho$,

$$\left\|\Phi^h(u_k)\right\|_{H^s} \leq R. \tag{3.54}$$

Now, by using *Hypothesis* (3.1) and *Lemma* (3.2), we get, for $k \leq n-1$ and $nh \leq T$,

$$\begin{aligned} \left\|\Phi^{(n-k-1)h}(\Pi^h u_k) - \Phi^{(n-k-1)h}(\Phi^h(u_k))\right\|_{H^s} &\leq K(R,T)\left\|\Pi^h(u_k) - \Phi^h(u_k)\right\|_{H^s}, \\ &\leq K(R,T)c_1 h^2. \end{aligned} \tag{3.55}$$

Substituting this result into (3.50), we obtain

$$\|u_n - u(t_n)\|_{H^s} \leq nK(R,T)c_1 h^2 \leq \gamma h \tag{3.56}$$

where c_1 depends on $\|u_0\|_{H^{s+l}}\|$. This completes the proof. □

3.2.4. Convergence Analysis of Burgers' Equation

In this section we study Burgers' equation as follows:

$$u_t + uu_x = \kappa \ , \ \ u(t_0) = u_0. \tag{3.57}$$

If we separate the equation (3.57) as in the general form (3.27), this gives us

$$Au = P(\partial_x)u = u_{xx} \tag{3.58}$$

$$B(u) = -uu_x \tag{3.59}$$

where (4.11) is linear operator and (3.59) is the non-linear operator. Approximated solution is given by the exponentail Euler method as:

$$u_{n+1} = e^{hA}u_n + h\varphi_1(hA)B(u_n) \tag{3.60}$$

where φ_1 is given in the equation (3.14). To prove the convergence results for Burgers' equation by using exponential Euler method, we use the same framework as in Allen-Cahn

equation's error estimation. The major difference between two equation is the nonlinear parts of the problems.

3.2.5. Local Error

In this section, to obtain local error estimation for exponential Euler method, we use the hypotheses that are defined in the previous section. The following lemma gives the local error result.

Lemma 3.3 *Let $s \geq 1$ be an integer and assume Hypothesis (3.2) and Hypothesis (3.3) hold for $k = s + l$ for the solution $u(t) = \Phi^t(u_0)$ of (3.57). If the initial data u_0 is in $H^{s+l}(\mathbb{R})$, then the local error of the exponential-Euler method is bounded in H^s by*

$$\|\Psi^h(u_0) - \Phi^h(u_0)\|_{H^s} \leq c_1 h^2, \tag{3.61}$$

where c_1 depends on $\|u_0\|_{H^{s+l}}$ and where h is small time step.

Proof Let $\Phi_A^t(v) = e^{tA}v$ where A is the linear flow. We start from the variation-of-constant formula for $\Phi^h(u_0)$

$$u(t) = e^{tA}u_0 + \int_0^t e^{(t-s)A}B(u(s))\ ds. \tag{3.62}$$

Taylor expansion of $B(u(s))$ is

$$B(u(s)) = B(u(0)) + \int_0^s dB(u(\sigma))[B(u(\sigma))]\ d\sigma. \tag{3.63}$$

Take $t = h$ and insert (3.63) into (3.62),

$$u(h) = e^{hA}u_0 + \int_0^h e^{(h-s)A}B(u_0)\ ds + \delta, \tag{3.64}$$

where

$$\delta = \int_0^h \int_0^s e^{(h-s)A}dB(u(\sigma))[B(u(\sigma))]\ d\sigma ds. \tag{3.65}$$

One step exponential-Euler (3.60) is as follows

$$u_1 = e^{hA}u_0 + h\varphi_1(hA)B(u_0). \tag{3.66}$$

The error between the exact and the exponential-Euler solution is,

$$\begin{aligned} u_1 - u(h) &= h\varphi_1(hA)B(u_0) - (-\frac{I - e^{hA}}{A}B(u_0)) + \delta, \\ &= \delta. \end{aligned} \tag{3.67}$$

We continue with the error bound for δ. This estimation gives us

$$\begin{aligned} \|\delta\|_{H^s} &\leq \int_0^h \int_0^s \|e^{(h-s)A} dB(u(\sigma))[B(u(\sigma))]\|_{H^s} d\sigma ds \\ &\leq \int_0^h \int_0^s \|dB(u(\sigma))[B(u(\sigma))]\|_{H^s} d\sigma ds \\ &\leq \int_0^h \int_0^s \|((u(\sigma))(B(u(\sigma))))_x\|_{H^s} d\sigma ds \\ &\leq \int_0^h \int_0^s \|(u(\sigma))(B(u(\sigma)))\|_{H^{s+1}} d\sigma ds \\ &\leq C \int_0^h \int_0^s \|(u(\sigma)\|_{H^{s+1}} \|B(u(\sigma))\|_{H^{s+1}} d\sigma ds. \end{aligned}$$

Using the definition of B

$$\begin{aligned} \|\delta\|_{H^s} &\leq C \int_0^h \int_0^s \|(u(\sigma)\|_{H^{s+1}} \|(u(\sigma)u_x(\sigma)\|_{H^{s+1}} d\sigma ds \\ &\leq C \int_0^h \int_0^s \|(u(\sigma)\|_{H^{s+1}} \|(u(\sigma)\|_{H^{s+1}} \|u_x(\sigma)\|_{H^{s+1}} d\sigma ds \\ &\leq C \int_0^h \int_0^s \|(u(\sigma)\|_{H^{s+1}} \|(u(\sigma)\|_{H^{s+1}} \|u(\sigma)\|_{H^{s+2}} d\sigma ds. \end{aligned}$$

Using the *Hypothesis 2* for $H^{s+l}(\mathbb{R})$, which gives that

$$\|u(\sigma)\|_{H^{s+1}} \leq \|u(\sigma)\|_{H^{s+2}} \leq \|u(\sigma)\|_{H^{s+l}} \leq R,$$

when $l \geq 2$ which results in

$$\|\delta\| \leq C \int_0^h \int_0^s R^3 d\sigma ds = CR^3 \int_0^h s ds = CR^3 h^2 = c_1 h^2 \tag{3.68}$$

This completes the proof. □

3.2.6. Global Error

To obtain the global error estimation and the first order convergence rate, we need to show that the exponential Euler solution at each step is bounded. Thus we need to use Lemma (3.3) which gives the local error estimation. The proof is done with the same way as we did in the proof of Theorem (3.1).

Theorem 3.2 *Assume there exists a solution of (3.57) and let $s \geq 1$ be an integer. If Hypothesis (3.1) and Hypothesis (3.3) hold for $k = s$ and Hypothesis (3.2) holds for $k = s + l$ for $l \geq 2$, then there exists $\overline{h} > 0$ such that for all $h \leq \overline{h}$ and $t_n = nh \leq T$,*

$$\|u_n - u(t_n)\|_{H^s} \leq Ch,$$

where u_n is the exponential Euler solution.

Proof The notational convention can be written as in (Holden & Lubich & Risebro)

$$u_k^n = \Phi^{n-kh}(u_k)$$

which is the exact solution of 3.57. With this notation, we note

$$u_n = u_n^n \quad \text{and} \quad u(t_n) = u_n^0.$$

The induction is started by assuming that

$$\begin{aligned} \|u_k\|_{H^s} &\leq R \\ \|u_k - u(t_k)\|_{H^s} &\leq \gamma h. \end{aligned}$$

Convergence result can be obtained by using Telescope sum and the triangle inequality as follows

$$\begin{aligned} \|u_n - u(t_n)\|_{H^s} &= \|u_n^n - u_n^0\|_{H^s} = \|u_n^n - u_n^{n-1} + u_n^{n-1} - u_n^{n-2} + u_n^{n-2} - \ldots - u_n^0\|_{H^s} \\ &= \Big\| \sum_{k=0}^{k+1} u_n^{k+1} - u_k^n \Big\|_{H^s} \leq \sum_{k=0}^{k+1} \|u_n^{k+1} - u_k^n\|_{H^s}. \end{aligned} \tag{3.69}$$

Using notational convention 3.69 becomes

$$\begin{aligned} \|u_n - u(t_n)\|_{H^s} &\leq \sum_{k=0}^{n-1} \|\Phi^{(n-k-1)h}(u_{k+1}) - \Phi^{(n-k)h}(u_k)\|_{H^s} \\ &= \sum_{k=0}^{n-1} \|\Phi^{(n-k-1)h}(\Pi^h u_k) - \Phi^{(n-k-1)h}(\Phi^h(u_k))\|_{H^s}. \end{aligned} \tag{3.70}$$

We obtain for $k \leq n-2$

$$\left\|\Pi^h u_k\right\|_{H^s} = \|u_{k+1}\|_{H^s} \leq R, \tag{3.71}$$

and

$$\begin{aligned} \left\|\Phi^h(u_k)\right\|_{H^s} &= \left\|\Phi^h(u_k) - \Phi^h(u(t_k)) + \Phi^h(u(t)_k)\right\|_{H^s} \\ &\leq \left\|\Phi^h(u_k) - \Phi^h(u(t_k))\right\|_{H^s} + \left\|\Phi^h(u(t)_k)\right\|_{H^s}. \end{aligned} \tag{3.72}$$

Using the Lipschitz continuity, (3.72) turns into

$$\left\|\Phi^h(u_k)\right\|_{H^s} \leq K(R,h)\|u_k - u(t_k)\|_{H^s} + \|u(t_{k+1})\|_{H^s} \leq K(R,h)\gamma h + \rho, \tag{3.73}$$

from the assumption of the induction argument. We choose $\leq K(R,h)\gamma h = R - \rho$,

$$\left\|\Phi^h(u_k)\right\|_{H^s} \leq R. \tag{3.74}$$

Therefore, by using *Hypothesis* (3.1) and *Lemma* (3.3), we get, for $k \leq n-1$ and $nh \leq T$,

$$\begin{aligned} \left\|\Phi^{(n-k-1)h}(\Pi^h u_k) - \Phi^{(n-k-1)h}(\Phi^h(u_k))\right\|_{H^s} &\leq K(R,T)\left\|\Pi^h(u_k) - \Phi^h(u_k)\right\|_{H^s} \\ &\leq K(R,T)c_1 h^2. \end{aligned} \tag{3.75}$$

Substituting result (3.75) into (3.70), we obtain

$$\|u_n - u(t_n)\|_{H^s} \leq nK(R,T)c_1 h^2 \leq \gamma h. \tag{3.76}$$

This completes the proof. □

CHAPTER 4

ITERATIVE LINEARIZATION TECHNIQUE

In this chapter a new numerical technique is proposed for the numerical solution of non-linear differential equations. This technique is based on the Fréchet derivative and Newton-Raphson method. In thsi technique, firstly we linearize the equation by using the Frèchet derivative to overcome the nonlinearity. Then in space discretization, localized differential quadrature method is used. In the time direction we apply the Crank-Nicolson rule. In this process, we convert to nonlinear differential equation to set of linear equations which are solved by a Newton-Raphson iterative method (Liu & Wu, 2000) and (Fazel & Moghadam & Poshtan, 2013).

4.1. Linearization Processes

To linearize the non-linear differential equation, we use the Fréchet derivative. A brief description about the Fréchet derivative was given in the Chapter 2. To start the process, consider the general form of the non-linear differential equation

$$L(u) = 0 \tag{4.1}$$

where L is the differential operator. First, Newton-Raphson iteration is applied to solve the equation (4.1) as follows

$$u^{(n+1)} = u^{(n)} + \theta^{(n)} \tag{4.2}$$

where $u^{(n)}$ is the approximated function of u, $\theta^{(n)}$ is the refinement function and n is the iteration number. The refinement variable $\theta^{(n)}$ is obtained by solving the following differential equation

$$\theta L'(\theta) + L(u) = 0. \tag{4.3}$$

The term $\theta L'(\theta)$ in equation (4.3) is the Fréchet derivative which is defined as

$$\theta L'(\theta) = \frac{\partial}{\partial \varepsilon} L(u + \varepsilon\theta)\Big|_{\varepsilon=0}. \tag{4.4}$$

Note that the refinement function θ goes to zero, thus the equation (4.3) reduced to the equation (4.1).

Now with this process we obtain the linear differential equation. To convert to linear differential equation into a system of algebraic equation, LDQ method is used. The idea of the LDQ method will explain in the following section.

4.2. Localized Differential Equation

The initial step of the LDQM is to determine neighboring grids of the point of interest and order of the approximation of the first derivative. For example if the function is discretized at the beginning boundary of the physical domain with respect to space variable x (at $x = 0$) or nearby it and a sixth-order first derivative approximation is used that means 7 neighboring grid points should be forward type in the direction of the space variable, and if the function is discretized at the end boundary of the physical domain with respect to space variable x (at $x = L$) or nearby it, the selection of the neighboring grid points should be backward type. At the interior reference points central type scheme is used. Then, the discretization of the first-order derivative of a function $u(x)$ with respect to space variable, x, at any discrete point x_i can be approximated using a weighted linear combination of the function values at some of the neighboring reference points within the computational domain as

$$\frac{\partial u(x_i, t)}{\partial x} = \sum_{j \in S_i} a_{ij}^{(1)} u(x_j, t), \quad i = 1, 2, ..., N, \tag{4.5}$$

where S_i represents the corresponding set of the neighboring nodes for the discrete grid point x_i in the domain or at the boundaries, N is the total amount of grid points in the direction of x. Weighting coefficients of the first-order derivative can be evaluated as follows

$$a_{ij}^{(1)} = \frac{\prod_{k \in S_i, k \neq i} (x_i - x_k)}{(x_i - x_j) \prod_{k \in S_i, k \neq j} (x_j - x_k)}, \quad i = 1, 2, ..., N, \quad j \in S_i, \quad i \neq j, \tag{4.6}$$

$$a_{ii}^{(1)} = -\sum_{j \in S_i, j \neq i} a_{ij}^{(1)}, \quad i = 1, 2, ..., N \tag{4.7}$$

Similarly,the higher order derivative can be expressed as:

$$\frac{\partial^r u(x_i,t)}{\partial x^r} = \sum_{j\in S_i} a_{ij}^{(r)} u(x_j,t), \quad i = 1,2,...,N, \quad r \geq 2 \tag{4.8}$$

where

$$a_{ij}^{(r)} = r.\Big(a_{ij}a_{ii}^{(r-1)} - \frac{a_{ij}^{(r-1)}}{(x_i - x_j)}\Big), \quad i = 1,2,...,N, \quad j \in S_i, \quad i \neq j, \quad r \geq 2, \tag{4.9}$$

$$a_{ii}^{(r)} = -\sum_{j\in S_i, j\neq i} a_{ij}^{(r)}, \quad i = 1,2,...,N, \quad r \geq 2. \tag{4.10}$$

Convenient choice for the sampling points is that of the equally spaced sampling points. (Zong & Zhang), (Zong & Lam) and (Yilmaz & Girgin & Evran)
After linearizing and applying the LDQ method in space, we have a set of linear equations as follows

$$\theta_t = A\theta \tag{4.11}$$

To solve equation (4.11), we apply the Crank-Nicolson rule. The algorithm of this processes is in the following form:

- Step 1: Fixed the initial condition as $u_1^0, u_2^0, ..., u_m^0$.
- Step 2: Predict the initial guess of $u_1^1, u_2^1, ..., u_m^1$.
- Step 3: Set the initial condition of θ as $0, 0, ..., 0$.
- Step 4: Calculate the approximated derivatives using LDQ method.
- Step 5: Calculate the refinement function's values $\theta_1, \theta_2, ..., \theta_m$.
- Step 6: Update the values $u_1^i, u_2^i, ..., u_m^i$ from the linear equation.
- Step 7: Continue solving the equation (4.11) by using Crank-Nicolson rule until the solutions approach to desired tolerance. Here we are also updated the approximated solution for u, i.e, $u^{(0)}, u^{(1)}....u^{(n)}$.

In order to elaborate the iterative procedure, we give a block diagram which is in Figure 4.1.

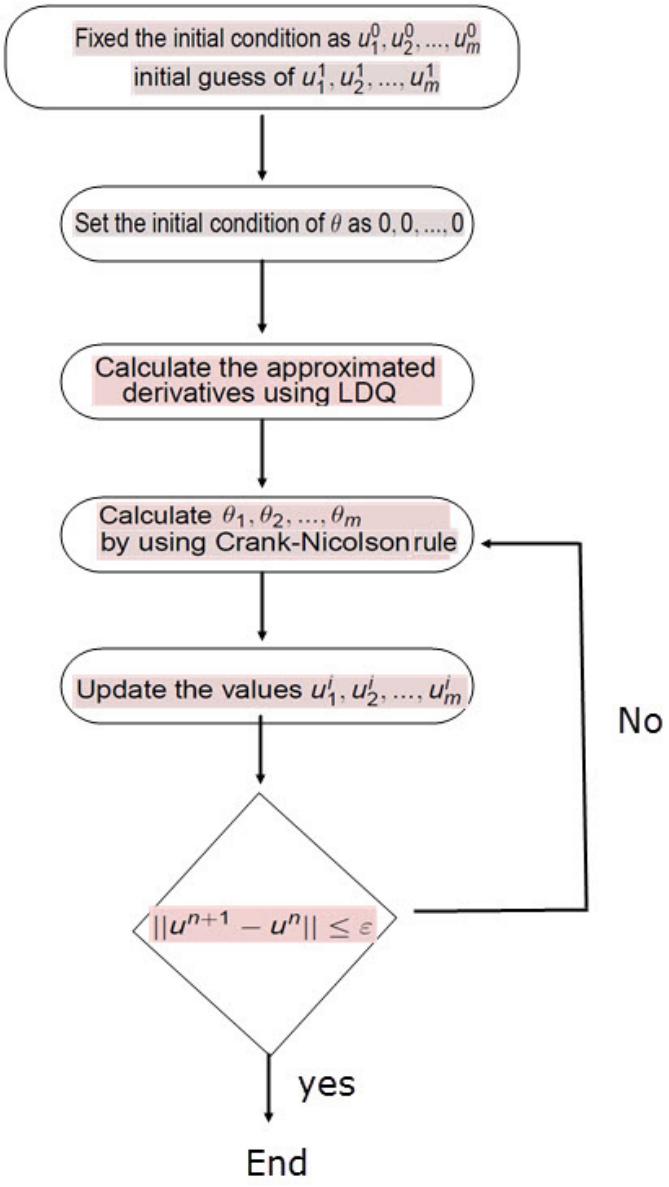

Figure 4.1. Diagram for the iterative solution procedure.

4.3. Linearization of the Allen-Cahn Equation

As an example of nonlinear differential equation, we consider Allen-Cahn equation. This equation is given by

$$u_t = \gamma u_{xx} + u - u^3, \quad x \in [-1, 1]. \tag{4.12}$$

Differential operator L is defined according to equation (4.12) as

$$L(u) = u_t - \gamma u_{xx} - u + u^3. \tag{4.13}$$

To obtain the Frèchet derivative of the operator (4.13),

$$L(u + \varepsilon\theta) = (u + \varepsilon\theta)_t - \gamma(u + \varepsilon\theta)_{xx} - (u + \varepsilon\theta) + (u + \varepsilon\theta)^3. \tag{4.14}$$

Then, definiton of the derivative is applied to equation (4.14)

$$\left.\frac{\partial L}{\partial \varepsilon}\right|_{\varepsilon=0} = \theta_t - \gamma\theta_{xx} - \theta + 3u^2\theta. \tag{4.15}$$

By substituting equation (4.15) into (4.3), we get

$$u_t - \gamma u_{xx} - u + u^3 + \theta_t - \gamma\theta_{xx} - \theta + 3u^2\theta = 0. \tag{4.16}$$

The equation (4.16) is linear with respect to θ. After this linearization, LDQ method is applied to space variable x and calculation of the u values starts.

4.4. Linearization of the Burgers' Equation

The second example is Burgers' equation which is given by

$$u_t + uu_x = \kappa u_{xx}. \tag{4.17}$$

Differential operator of the Burgers' equation can be defined as

$$L(u) = u_t + uu_x - \kappa u_{xx}. \tag{4.18}$$

Fréchet derivative of equation (4.18) is expressed in the following form

$$\begin{aligned} L(u+\varepsilon\theta) &= (u+\varepsilon\theta)_t + (u+\varepsilon\theta)(u+\varepsilon\theta)_x - \kappa(u+\varepsilon\theta)_{xx}, \\ &= u_t + \varepsilon\theta_t + uu_x + u\varepsilon\theta_x + \varepsilon\theta u_x + \varepsilon^2\theta\theta_x - \kappa u_{xx} - \kappa\varepsilon\theta_{xx}. \end{aligned} \tag{4.19}$$

Applying the definition of Fréchet derivative, we obtain

$$\begin{aligned} \left.\frac{\partial L}{\partial \varepsilon}\right|_{\varepsilon=0} &= \theta_t + u\theta_x + \theta u_x - \kappa\theta_{xx}, \\ &= \theta L'(\theta). \end{aligned} \tag{4.20}$$

Now, combining the equation (4.20) with equation (4.3), we get

$$\theta L'(\theta) + L(u) = \theta_t + u\theta_x + \theta u_x - \kappa\theta_{xx} + u_t + uu_x - \kappa u_{xx} = 0. \tag{4.21}$$

Here the last equation (4.21) is linear with respect to θ and can be written as

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

where

$$\alpha = u_t + uu_x - \kappa u_{xx}. \tag{4.22}$$

Derivative of u and θ with respect to x is obtained from the *LDQ* method. Then we evaluate the values of θ by using the Crank-Nicolson rule and update the u values.

CHAPTER 5

NUMERICAL EXPERIMENTATION

In this chapter, we numerically examine the exponential Euler method and new linearization technique which are given in the previous chapters. We study two equations in details; that is, we use the Allen-Cahn equation and viscous Burgers' equation. Since we have two different numerical scheme, we separate this section into two part. In the first part, we give the numerical results for the exponential Euler method to support the theoretical results. In the second part, we present the numerical results for the new linearization technique.
We consider the two test problem with the following initial and boundary conditions. First equation is the Allen-Cahn equation.

Allen-Cahn Equation: Allen-Cahn equation is a well-known reaction diffusion equation of mathematical physics. The equation is given by

$$u_t = \gamma u_{xx} + u(1 - u^2), \quad x \in [-1, 1], \quad t \in [0, T] \tag{5.1}$$

where $\gamma = 0.001$ with initial and boundary conditions

$$u(0, x) = 0.53x + 0.47sin(-1.5\pi x) \tag{5.2}$$

$$u(t, -1) = -1 \tag{5.3}$$

$$u(t, 1) = 1. \tag{5.4}$$

This equation has a stable equilibria at $u = 1$ and $u = -1$ also has an unstable equilibrium at $u = 0$. One of the interesting features of this equation is the phenomenon of metastability. Regions of the solution that are near ± 1 will be flat, and the interface between such areas can remain unchanged over a very long timescale before changing suddenly (Trefethen & Kassam).

Burgers' Equation: Burgers' equation is a partial differential equation that is used in fluid mechanics. It take place in various areas of applied mathematics, such as modeling

of gas dynamics and traffic flow. For the Burgers' equation, we consider two different initial and boundary conditions. The first equation with periodic boundary conditions is given by

$$u_t = \kappa u_x x - u u_x, \quad x \in [-\pi, \pi], \quad t \in [0, 1] \tag{5.5}$$

where $\kappa = 0.03$. Initial and boundary conditions are

$$\begin{aligned} u(x, 0) &= e^{-10 \sin^2(x/2)}, & (5.6)\\ u(-\pi, t) &= 0, & (5.7)\\ u(\pi, t) &= 0. & (5.8) \end{aligned}$$

The second equation with initial and boundary conditions is given by

$$\begin{aligned} u_t &= \kappa u_x x - u u_x, \quad x \in [0, 1], \quad t \in [0, 3], & (5.9)\\ u(x, 0) &= 4x(1 - x), & (5.10)\\ u(0, t) &= u(1, t) = 0, & (5.11) \end{aligned}$$

where $\kappa = 0.1, 0.01$.
For the Allen-Cahn equation and Burgers' equation, the term u_{xx} is caused the stiffness .

5.1. Numerical Results for Exponential Euler Method

In this section, we investigate the exponential Euler method numerically. Since we have obtained theoretical results for this method, we now turn our attention to illustrate these results by dealing with numerical experiments. In the numerical experimentation for the Allen-Cahn equation and Burgers' equation, we consider the accuracy of the errors and the convergence rates.

5.1.1. Numerical Results for Allen-Cahn Equation

To obtain the results, we consider the Allen-Cahn equation with the initial and boundary conditions which are given by equations (5.1)-(5.4).

To solve this equation numerically firstly, we need to apply a space discretization technique. For the space discretization, we use the central finite difference method. The central difference approximation of u_{xx} is

$$u_{xx}\Big|_{(t,x_i)} \approx \frac{u(t, x_{i+1}) - 2u(t, x_i) + u(t, x_{i-1})}{h_x^2} \tag{5.12}$$

where h_x is the spatial stepping in space that is divided the interval into N equal part and $i = 1, ..., N + 1$. Now, the equation (5.1) turns to

$$u_t = \gamma A u + u(1 - u^2) \tag{5.13}$$

where A central difference coefficients matrix.
For time discretization, exponential Euler method is used. Thus, an approximation is given as

$$u_{n+1} = e^{h_t \gamma A} + h_t \varphi_1 h_t \gamma A(u - u^3). \tag{5.14}$$

We obtain the convergence rate for h_t. Allen-Cahn equation is solved by using $N_t = 50, 100, 200, 400, 800$. We take $N_x = 50$ which gives $h_x = 0.04$ for the calculations. Since the exact solution does not exist, the solution that is obtained by using higher order exponential method is used as an exact solution. The errors are computed using L^1-, L^2- and $L^\infty-$ norms. Standard linear regression on logarithmic scales is used to obtain convergence rates. The convergence rates for h_t for the exponential Euler method are given in Table 5.1. Error plot is provided in Figure 5.1. We observe that the method is converge with the expected rate.

h_x	h_t	L^∞	$order$
0.04	0.2	0.01251	
			0.6494
	0.1	0.007984	
			0.8243
	0.05	0.00450	
			0.9120
	0.025	0.00239	
			0.9560
	0.0125	0.00123	

Table 5.1. Error table of Allen-Cahn equation via exponential Euler method.

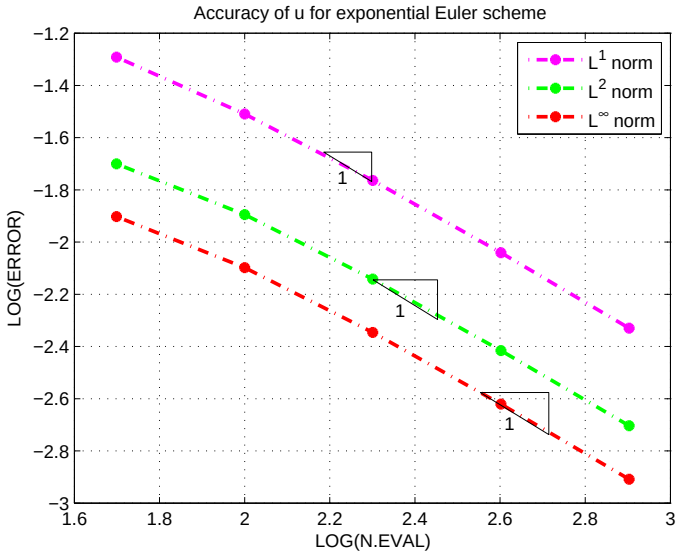

Figure 5.1. Order of exponential Euler method for Allen-Cahn equation.

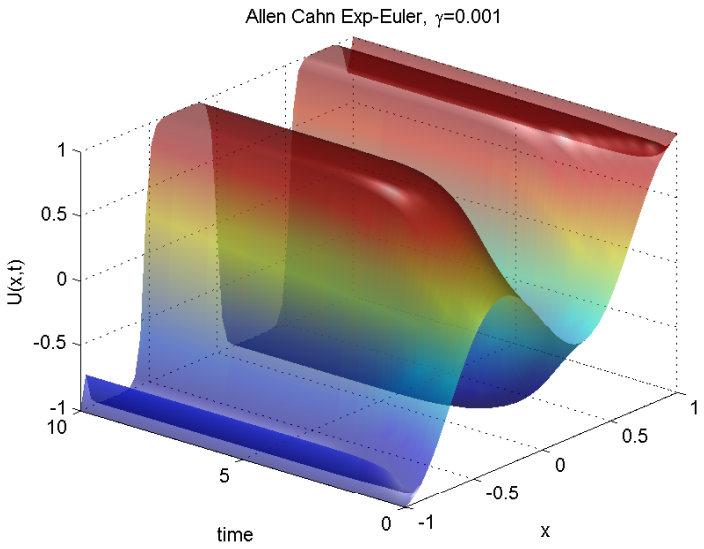

Figure 5.2. Exponential Euler solution of Allen-Cahn equation when $h_x = 0.04, h_t = 0.01$, T=10

5.1.2. Numerical Results for Burgers' Equation

We consider the Burgers' equations (5.5) and (5.9) to derive the numerical results. To solve these two equations, exponential Euler method is used. For spatial discretization of the first and the second derivative in space, central finite difference scheme is used. Central finite difference scheme for the first derivative is

$$u_x\Big|_{(t,x_i)} \approx \frac{u(t, x_{i+1}) - u(t, x_{i-1})}{2h}.$$

The second derivative approximation is

$$u_{xx}\Big|_{(t,x_i)} \approx \frac{u(t, x_{i+1}) - 2u(t, x_i) + u(t, x_{i-1})}{(h)^2}.$$

The equation (5.5) turns into

$$u_t = \kappa Au + u(Bu) \tag{5.15}$$

where A and B are the central difference coefficients matrix which comes from the second derivative and the first derivative.
Implementation of exponential Euler method for the Burgers' equation is given as follows:

$$u_{n+1} = e^{h_t \kappa A} + h_t \varphi_1 h_t \kappa A(u(Bu)). \tag{5.16}$$

Since we have two different initial conditions for the Burgers' equation, we consider the equations (5.5)-(5.8) as a first example. For the first example, to obtain the convergence rate for h_t we solve Burgers' equation using $N_t = 50, 100, 200, 400, 800$. We take $N_x = 64$ which gives $h_x = 2\pi/64$ for the calculations. Since the exact solution does not exist, the solution that is obtained by using higher order exponential method is used as an exact solution. The errors are computed using L^1-, L^2- and $L^\infty-$ norms. Standard linear regression on logarithmic scales is used to obtain convergence rates. The convergence rates and errors for h_t for the exponential Euler method are given in Table 5.2. Error plot is presented in Figure 5.3. We observe that the method is converge with the expected rate. A plot of the solution for exponential Euler method is givenin Figure 5.4. We note that solution conserve the shape of initial condition.

h_x	h_t	L^∞	$order$
$\frac{2\pi}{64}$	0.02	0.09749	
			1.0081
	0.01	0.04847	
			1.0017
	0.005	0.02420	
			1.0003
	0.0025	0.01210	
			1.0000
	0.00125	0.00605	

Table 5.2. Error table of Burgers' equation via exponential Euler method.

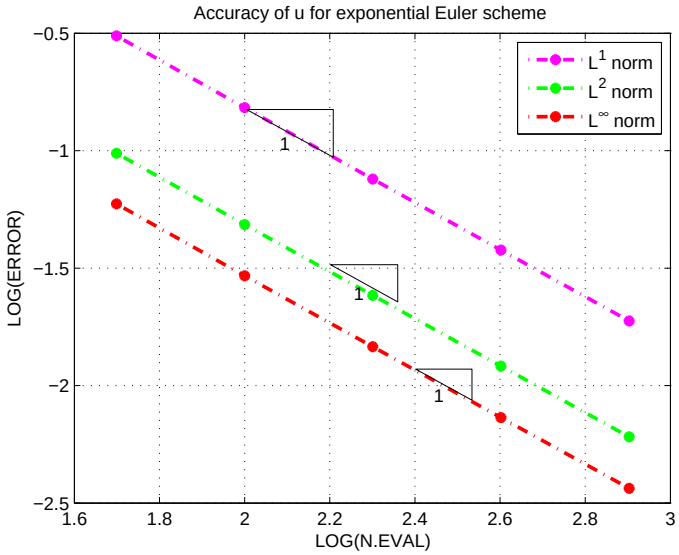

Figure 5.3. Order of exponential Euler method for Burgers' equation

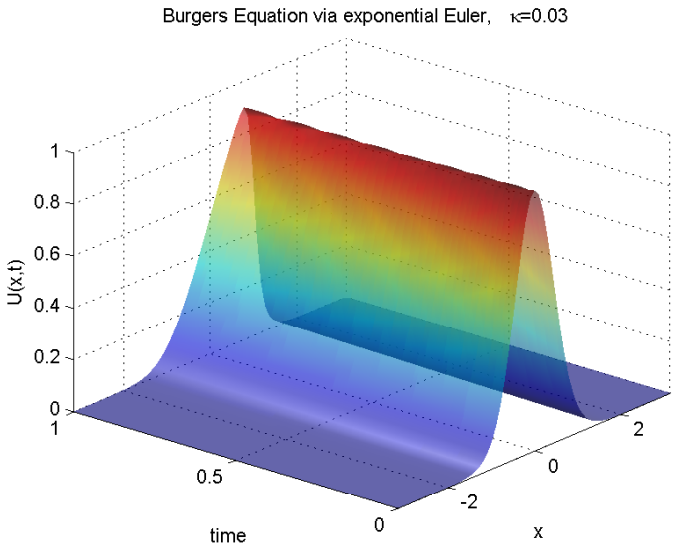

Figure 5.4. Numerical solution of Burgers' equation via exponential Euler method .

We now consider the equations which are given by the equations (5.9)-(5.11) as a second example of Burgers' equation when $\kappa = 0.1$. We solve the equation at different h_t values. We obtain this values using $N_t = 300, 6000, 1200, 2400, 4800$ grid nodes. We take $N_x = 40$ which gives $h_x = 1/40$ for the calculations. We take the same norms with the previous examples to obtain the convergence rates. Exponential Euler solution and the exact solution of the equation at different time and space nodes is given in Table 5.3. From the table we observe that numerical solution is close to exact solution. A standard error plot is presented in Figure 5.5. This figure shows that expected order is achieved. Numerical solutions at different times is given in Figure 5.6. A plot of the solution for exponential Euler method is given in Figure 5.7. We observe that solutions conserve the shape of initial condition.

x	$\kappa = 0.1$		
	t	Exp Euler	Exact Solution
0.25	0.4	0.31748	0.31752
	0.8	0.19956	0.19956
	1.0	0.16563	0.16560
	3.0	0.02781	0.02775
0.50	0.4	0.58471	0.58454
	0.8	0.36763	0.36740
	1.0	0.29860	0.29834
	3.0	0.04116	0.04106
0.75	0.4	0.64672	0.64562
	0.8	0.38628	0.38534
	1.0	0.29660	0.29586
	3.0	0.03051	0.03044

Table 5.3. Exponential Euler solution and exact solution of example 2 when $h_x = 0.05, h_t = 0.00625$ and $\kappa = 0.1$.

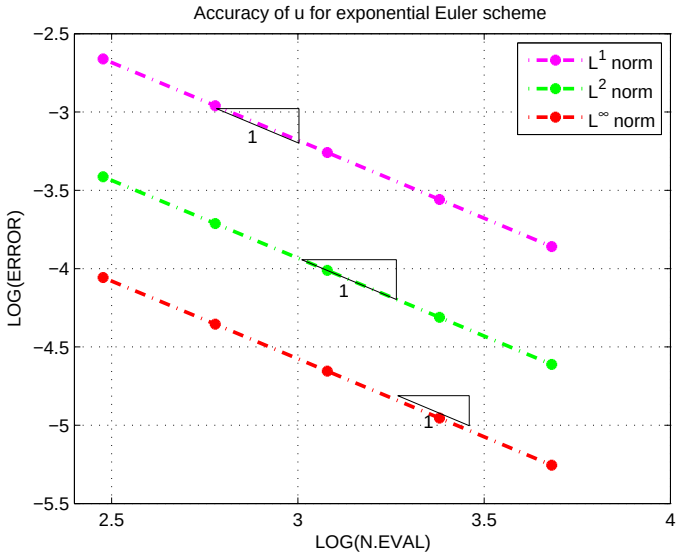

Figure 5.5. Order of Exponential Euler solution of example 2 when $\kappa = 0.1$.

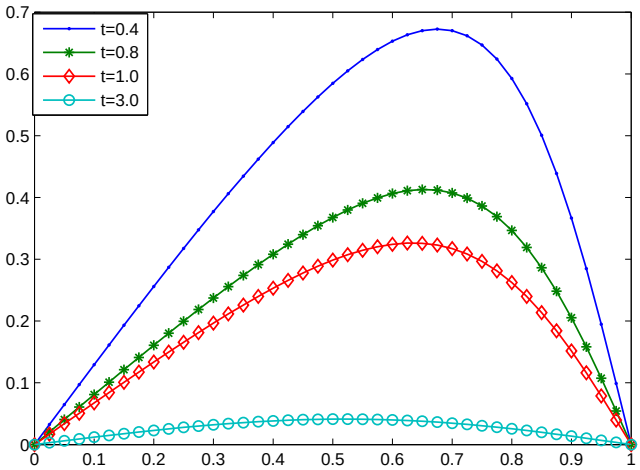

Figure 5.6. Exponential Euler solution of example 2 at different times when $h_x = 0.05, h_t = 0.00625$ and $\kappa = 0.1$.

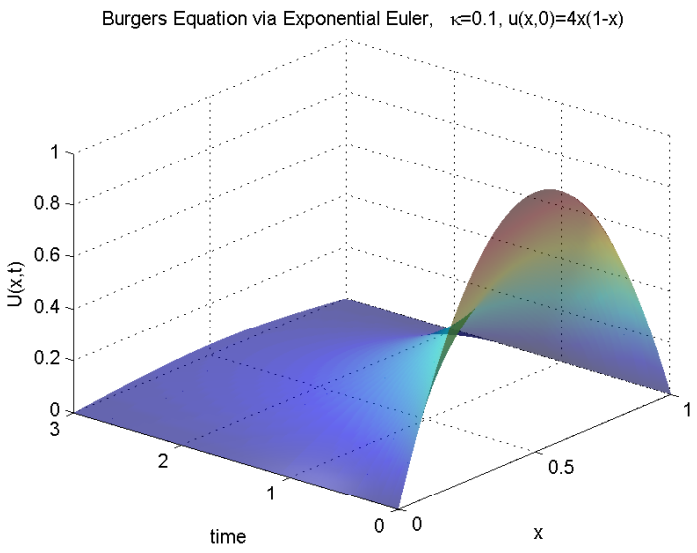

Figure 5.7. Exponential Euler solutions of example 2 when $h_x = 0.05, h_t = 0.01$ and $\kappa = 0.1$.

Now we consider the same equation for $\kappa = 0.01$. Exponential Euler solution and exact solution for this example are given in Table 5.4. This table shows that numerical solution converges to exact solution. The convergence rates for the exponential Euler method are given in Figure 5.8. From this figure we observe that expected order is achieved. The solutions of Burgers' equation at different times are provided in Figure 5.9. This figure reveals that numerical solutions conserve the shape of initial condition.

x	$\kappa = 0.01$		
	t	Exp Euler	Exact Solution
0.25	0.4	0.36209	0.36226
	0.8	0.23033	0.23045
	1.0	0.19459	0.19469
	3.0	0.07610	0.07613
0.50	0.4	0.68362	0.68368
	0.8	0.45352	0.45371
	1.0	0.38550	0.38568
	3.0	0.15212	0.15218
0.75	0.4	0.92134	0.92050
	0.8	0.66264	0.66272
	1.0	0.56915	0.56932
	3.0	0.22776	0.22774

Table 5.4. Exponential Euler solution and exact solution of example 2 when $h_x = 0.025, h_t = 0.00625$ and $\kappa = 0.01$.

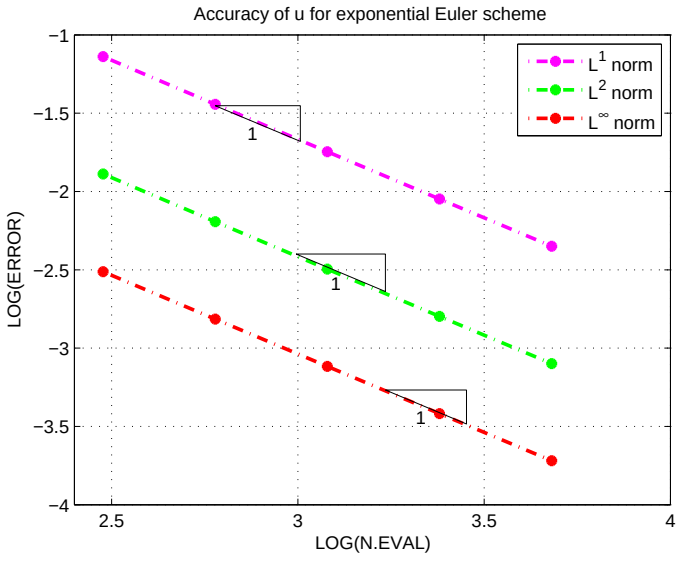

Figure 5.8. Order of Exponential Euler solution of example 2 when $\kappa = 0.01$.

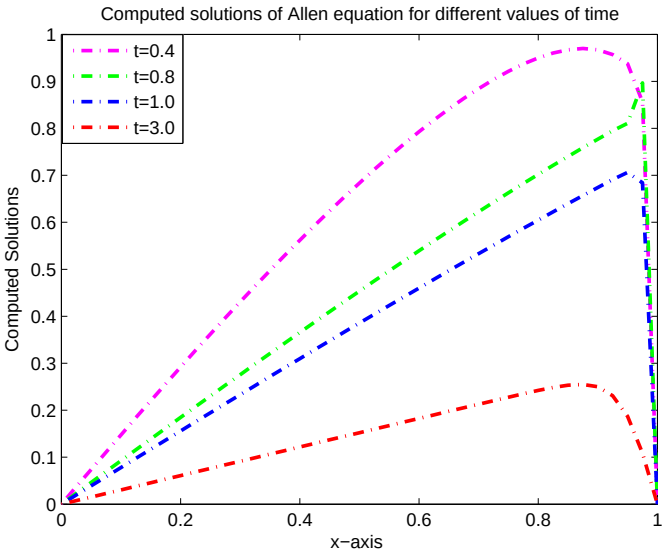

Figure 5.9. Exponential Euler solutions of example 2 when $h_x = 0.025, h_t = 0.00625$ and $\kappa = 0.01$.

5.2. Numerical Results for New Linearization Technique

In this section, we present the numerical results for new linearization technique. Linearization of the Allen-Cahn equation and Burgers' equation were given in Chapter 4 with the equations (4.16) and (4.21). We now focus on the application of this process in numerical sense.

5.2.1. Numerical Results for Allen-Cahn Equation

In this process, as a space discretization technique we use the LDQ method which given in Chapter 4. After linearizing and applying the LDQ method, we obtain the Allen-Cahn equation (5.1) in the following form.

$$u_t - \gamma A u - u + u^3 + \theta_t - \gamma A\theta - \theta + 3u^2\theta = 0 \tag{5.17}$$

where A is the coefficient matrix that comes from LDQ method. In equation (5.17), to solve the system, Crank-Nicolson scheme is used. Then, the system returns to

$$\begin{aligned}\frac{u_{n+1}-u_n}{\Delta t}+\frac{\theta_{n+1}-\theta_n}{\Delta t} &= \gamma A\frac{u_{n+1}+u_n}{2}+\frac{u_{n+1}+u_n}{2}-\left(\frac{u_{n+1}+u_n}{2}\right)^3+\gamma A\frac{\theta_{n+1}\theta_n}{2},\\ &+ \frac{\theta_{n+1}+\theta_n}{2}-3\left(\frac{u_{n+1}+u_n}{2}\right)^2\frac{\theta_{n+1}+\theta_n}{2}. \end{aligned} \tag{5.18}$$

To solve the system (5.18), u_{n+1}, u_n and θ_n have to be known. For the first step, u_0 which is the initial condition is known, u_1 is guessed and $\theta_0 = 0$. Since we do not have the exact solution of Allen-Cahn equation, we show the efficiency of the methods with the stability. To show that the method is stable, we take different time steps as $h_t = 0.1, 0.01, 0.001$. Numerical solutions of this equation at different times and spaces values is given in Table 5.5. From this table, We observe that the new technique is stable. A plot of numerical solution is presented in Figure 5.10. This figure shows that the numerical solutions conserve the shape of initial condition.

x	t	$h_t = 0.1$	$h_t = 0.01$	$h_t = 0.001$
	0.1	0.073564	0.073561	0.073561
-0.5	0.5	0.104196	0.104175	0.104175
	0.9	0.145777	0.145730	0.145729
	0.1	6.10968e-20	-1.76754e-20	6.71595e-21
0	0.5	-5.91510e-19	-3.30568e-20	-7.50574e-21
	0.9	-5.06166e-19	2.64537e-19	-1.78773e-19
	0.1	-0.073564	-0.073561	-0.073561
0.5	0.5	-0.104186	-0.104175	-0.104175
	0.9	-0.145777	-0.145730	-0.145729

Table 5.5. Numerical solution of Allen-Cahn equation via iterative linearization technique at different times and different Δt values.

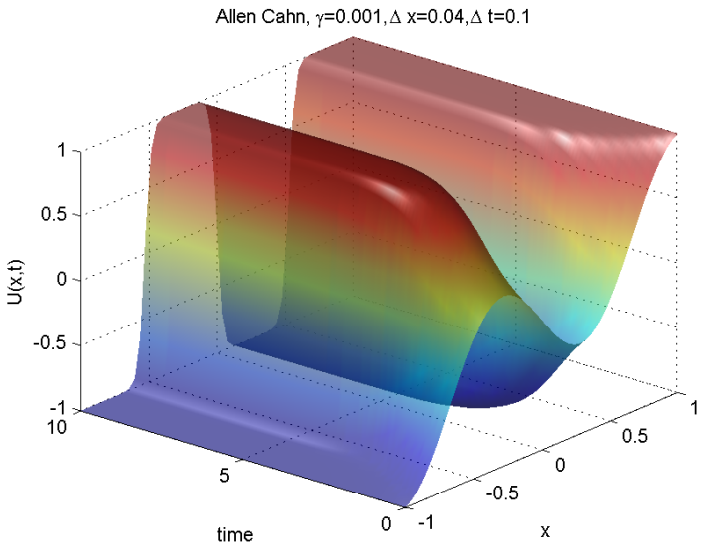

Figure 5.10. Numerical solution of Allen-Cahn equation via iterative linearization technique when hx=0.04, ht=0.01, T=10.

5.2.2. Numerical Results for Burgers' Equation

Equation (5.5) was linearized in the previous chapter. After linearizing the Burgers' equation, for the spatial discretization in space we use the LDQ method. The first and the second order derivatives are given with the equations (4.5) and (4.8). The linear equation for the Burgers' equation is given by

$$\theta_t + u_t - \kappa A u - \kappa A \theta + u(B\theta) + \theta(Bu) + u(Bu) = 0 \tag{5.19}$$

where A and B are the coefficient matrixes that comes from LDQ method.
To solve the system, Crank-Nicolson rule is used and the system (5.19) returns to

$$\begin{aligned} \frac{u_{n+1}-u_n}{\Delta t} + \frac{\theta_{n+1}-\theta_n}{\Delta t} &= \kappa A \frac{u_{n+1}+u_n}{2} - \frac{u_{n+1}+u_n}{2}\left(B\frac{u_{n+1}+u_n}{2}\right) + \kappa A \frac{\theta_{n+1}\theta_n}{2} \\ &- \frac{u_{n+1}+u_n}{2}\left(B\frac{\theta_{n+1}+\theta_n}{2}\right) - \frac{\theta_{n+1}+\theta_n}{2}\left(B\frac{u_{n+1}+u_n}{2}\right). \end{aligned} \tag{5.20}$$

To solve the system (5.20), u_{n+1}, u_n and θ_n have to be known. For the first step, u_0 which is the initial condition is known, u_1 is guessed and $\theta_0 = 0$. Firstly, as an example of Burgers' equation we consider the equation (5.5)-(5.8). A plot of the numerical solution is given in

Figure 5.11. Numerical solutions of the Burgers equation at different times are shown in Fgiure 5.12. From these figures, we observe that the numerical solutions conserve the shape of initial condition.

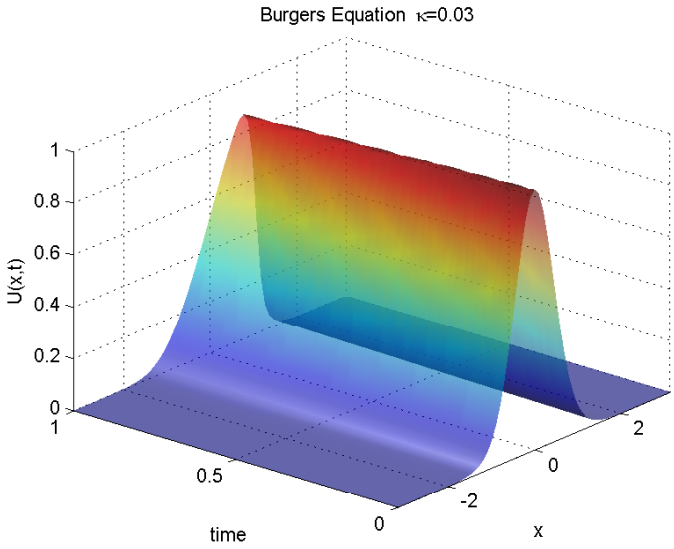

Figure 5.11. Numerical solution of Burgers' equation via iterative linearization technique

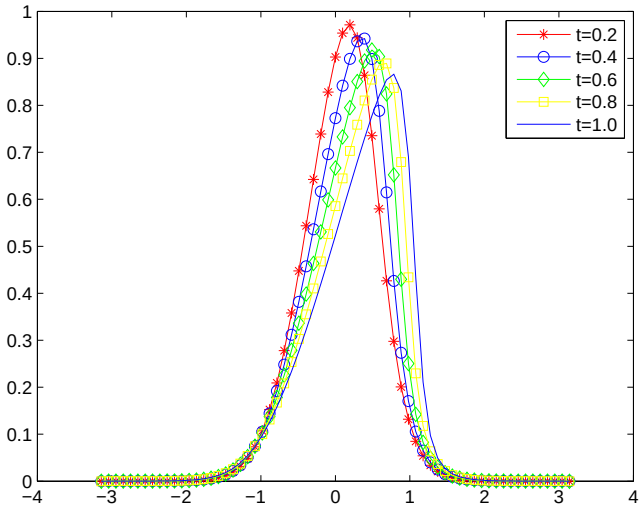

Figure 5.12. Numerical solution of Burgers' equation via iterative linearization technique at different time

As a second example, we consider the equations (5.9)-(5.11). For this equation we use the same processes as the previous example. After linearizing the equation, LDQ method is applied on the space. Finally, the system is solved by Crank-Nicolson rule. At the first we give the numerical results for $\kappa = 0.1$. In Table 5.2.2, the new technique is compared with the well-known published schemes and the exact solution. Comparison shows that our presented scheme is better than the others. A plot of the numerical solutions at different times is given in Figure 5.13. The iterative linearization solution is provided in Figure 5.14 .

x	$\kappa = 0.1$					
	t	(Kutluay, 2004) $h_t = 0.0001$ $h_x = 0.0125$	(Ozis, 2003) $h_t = 0.0001$ $h_x = 0.0125$	(Jiwari, 2013) $h_t = 0.0001$ $h_x = 0.04$	Present scheme $h_t = 0.0001$ $h_x = 0.05$	Exact Solution
0.25	0.4	0.32091	0.32678	0.31744	0.31752	0.31752
	0.8	0.20211	0.20274	0.19952	0.19955	0.19956
	1.0	0.16782	0.16786	0.16557	0.16560	0.16560
	3.0	0.02828	0.02814	0.02775	0.02775	0.02775
0.50	0.4	0.58788	0.59660	0.58443	0.58453	0.58454
	0.8	0.37111	0.37293	0.36733	0.36740	0.36740
	1.0	0.30183	0.30255	0.29830	0.29835	0.29834
	3.0	0.04185	0.04161	0.04106	0.04106	0.04106
0.75	0.4	0.65054	0.64691	0.64556	0.64564	0.64562
	0.8	0.39068	0.39120	0.38526	0.38536	0.38534
	1.0	0.30057	0.30067	0.29582	0.29587	0.29586
	3.0	0.03106	0.03084	0.03043	0.03044	0.03044

Table 5.6. Numerical and exact solutions of example 2 when $h_x = 0.05, h_t = 0.0001$ and $\kappa = 0.1$.

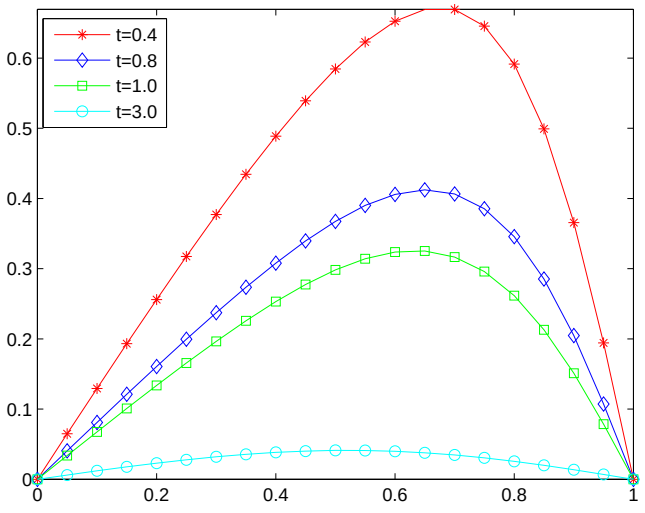

Figure 5.13. Iterative linearization solution of example 2 at different times when $h_x = 0.05, h_t = 0.01$ and $\kappa = 0.1$.

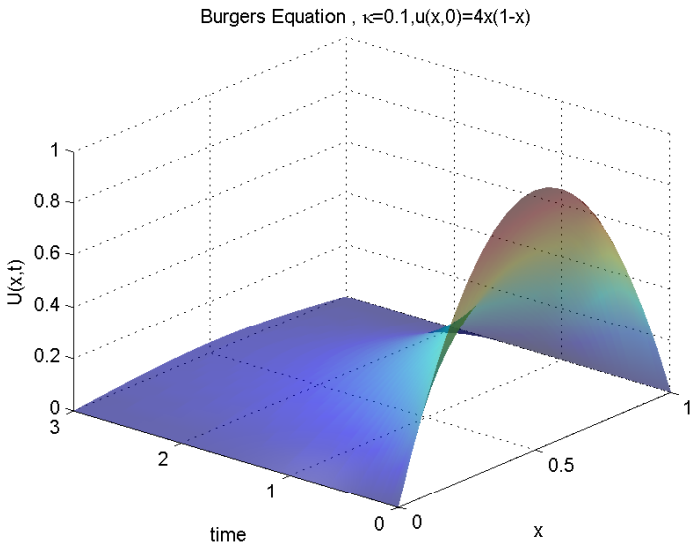

Figure 5.14. Iterative linearization solution of example 2 when $h_x = 0.05, h_t = 0.01$ and $\kappa = 0.1$.

We now give the iterative linearization technique results for $\kappa = 0.01$. The comparison of our presented scheme and the well-known published schemes is given in Table 5.2.2. From this table, we observe that this new technique is better than the others. The numerical solution at different times is presented in Figure 5.15. This figure show that the numerical solutions conserve the shape of initial condition. Finally, the iterative linearization solutions for $\kappa = 0.1, 0.01, 0.001$ are provided in Figure 5.16.

x	$\kappa = 0.01$				
	t	(Kutluay, 2004) $h_t = 0.0001$ $h_x = 0.0125$	(Jiwari, 2013) $h_t = 0.0001$ $h_x = 0.0125$	Present scheme $h_t = 0.0001$ $h_x = 0.05$	Exact Solution
0.25	0.4	0.36911	0.36213	0.36226	0.36226
	0.8	0.23703	0.23066	0.23045	0.23045
	1.0	0.20069	0.19468	0.19469	0.19469
	3.0	0.07865	0.07613	0.07613	0.07613
0.50	0.4	0.68818	0.68357	0.68367	0.68368
	0.8	0.46011	0.45412	0.45371	0.45371
	1.0	0.39206	0.38563	0.38567	0.38568
	3.0	0.15576	0.15217	0.15218	0.15218
0.75	0.4	0.92194	0.92064	0.92084	0.92050
	0.8	0.66777	0.66303	0.66314	0.66272
	1.0	0.57491	0.56929	0.56961	0.56932
	3.0	0.23183	0.22774	0.22778	0.22774

Table 5.7. Numerical and exact solution of example 2 when $h_x = 0.05, h_t = 0.0001$ and $\kappa = 0.01$.

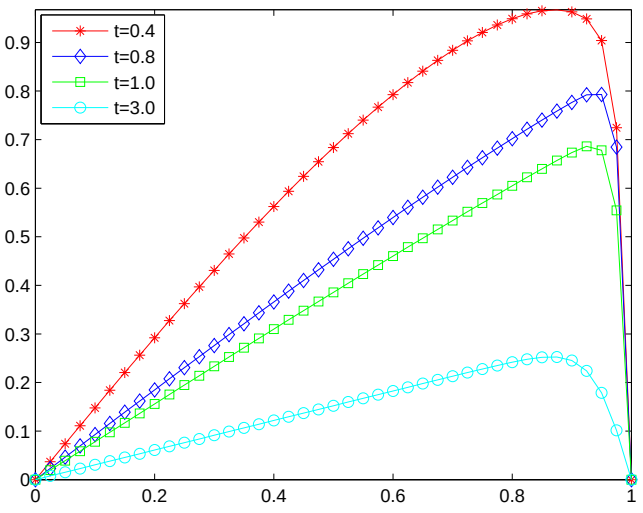

Figure 5.15. Iterative linearization solutions of example 2 when $h_x = 0.025, h_t = 0.01$ and $\kappa = 0.01$.

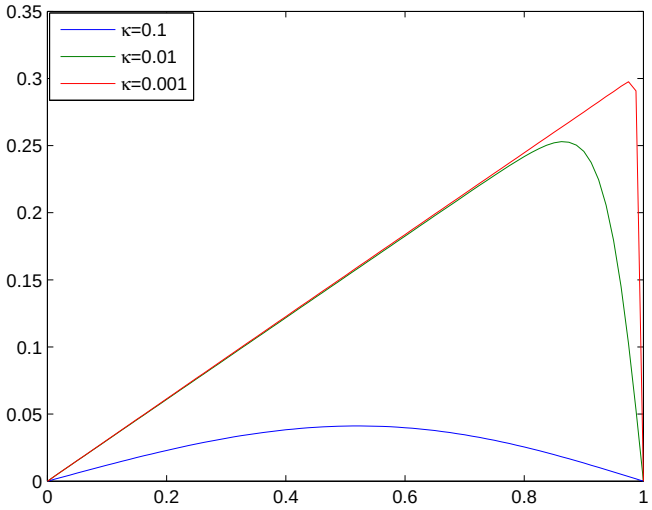

Figure 5.16. Numerical solutions at the end time $T = 3$ with different κ values.

CHAPTER 6

CONCLUSION

In this thesis, we present two numerical methods to solve the nonlinear stiff problems,namely exponential integrators and a new iterative linearization technique. We only study the error analysis of exponential- Euler method by using the Sobolev space norms and Frechet derivative on problem-based. After introducing well known time integration methods, exponential Euler method and exponential Rosenbrock-Euler method, this analysis technique are used for the Allen-Cahn equation and Burgers' equation in order to estimate the local and global errors. Then, we develop a new iterative linearization technique based on the Frèchet derivative and Newton-Raphson method. For the space disceritization localized differential quadrature rule is used, in addition in time discretization Crank-Nicolson scheme is used. Several examples are illustrated in order to show how these methods are work. We found that exponential Euler method solutions are preserved the convergence rates. In addition, we illustrate the theoretical results by conducting numerical solutions for the Allen-Cahn and Burgers' equations. For the new linearization technique, we compared our results with the well known techniques. This technique seems to be better than the other techniques.

REFERENCES

Beylkin, G. , Keiser J.M. and Vozovoi L. 1998: A New Class of Time Discretization Schemes for the Solution of Nonlinear PDEs. *J. Comput. Phys.,* **147** 362-387.

Brugnano, L. and Trigiante, D. 1996: On the characterization of stiffness for ODEs. *Dynam. Contin. Discrete Impuls. Systems.,* **2(3)** 317û335.

Caliari ,M. and Ostermann, A. 2009: Implementation of Exponential Rosenbrock-Type Integrators. *Elsevier Applied Numerical Mathematics,* **59(3),** 568-581 .

Cox ,S.M. and Matthews,P. C. 2002: Exponential Time Differencing for Stiff Systems . *Journal Compt. Phys,* **176** 430-455.

Curtis,C.F. and Hirschfelder,J.O. 1950: Integration of Stiff Equations. *Proc. Natl. Acad. Sci. USA ,* **38(3),** 235-243.

Dahlquist, G. 1963: A Special Stability Problem for Linear Multi-Step. *BIT Numer. Math.,* **3,** 27û43.

El-Azab,T.M.A. 2012: Exponential Peer Methods. *Martin-Luther-Universität Halle-Wittenberg* .

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.

Hairer, E. and Wanner,G. 2000: Solving Ordinary Differential Equations II. *Springer,* **Second Edition**.

Hochbruck ,M. and Ostermann, A. 2010: Exponential Integrators. *Applied Numerical Mathematics,* **Cambridge University Press,** 209-286 .

Hochbruck ,M. and Ostermann, A. 2005: Explicit Exponential Runge-Kutta Methods for Semilinear Parabolic Problems. *SIAM Journal on Numerical Analysis,* **43(3),** 1069-1090 .

Hochbruck ,M. , Ostermann, A. and Schweitzer,J. 2008: Explicit Exponential Runge-Kutta Methods for Semilinear Parabolic Problems. *SIAM Journal on Numerical Analysis,* **47(1),** 786-803.

Hochbruck ,M. and Ostermann, A. 2005: Exponential Runge-Kutta Methods for Parabolic Problems. *Applied Numerical Mathematics,* **53(2-4),** 323-339 .

Hochbruck,M. , Lubich,C. and Selhofer, H. 1998: Exponential Integrators for Large Systems of Differential Equations . *SIAM J. Sci. Comput. ,* **19(5),** 1552-1574.

Holden, H. , Lubich, C. and Risebro, H. 2011: Operator Splitting for Partial Differential Equations with Burgers Nonlinearity. *Math. Comp. ,* **82(2013),** 173-185.

Huang,P. and Abduwali,A. 2011: A Numerical Method For Solving Allen-Cahn Equation . *J. Appl. Math. Informatics ,* **29,** 1477-1487.

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.

Kandolf, P. 2011: Exponential Integrators. *McMaster University* .

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.

Lambert, J.D 1991: Numerical Methods for Ordinary Differential Systems. *John Wiley & Sons.*

Lawson,J. D. 1967: Generalized Runge-Kutta Processes for Stable Systems with Large Lipschitz Constants. *SIAM Journal on Numerical Analysis,* **4(3),** 372-380 .

Liniger ,W. and Willoughby,R. A. 1970: Efficient Integration Methods for Stiff Systems of Ordinary Differential Equations . *SIAM Journal on Numerical Analysis,* **7(1)** .

Luan ,V.T. and Ostermann, A. 2014: Explicit Exponential Runge-Kutta Methods of High Order for Parabolic Problems. *Journal of Compt. and Applied Maths.* .

Minchev,B.V. ,2004: Exponential Integrators for Semilinear Problems. *University of Bergen ,PhD Thesis* **2004**.

Minchev,B.V. , Wright,W.M. ,2005: A review of exponential integrators for first order semi-linear problems. *NTNU* **2005(2)**.

Nilsen,E.B. ,2011: On Operator Splitting for the Viscous Burgers' and the Korteweg-de Vries Equations. *NTNU* **201**

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

Pope,D. A. 1963: An Exponential Method of Numerical Integration of Ordinary Differential Equations. *Communications of the ACM,* **6(8),** 491-493 .

Schmelzer,T. , Trefethen,L.N. 2007: Evaluating Matrix Functions for Exponential Integrators via Caratheodory-Fejer Approximation and Contour Integrals. *Electronic Transactions on Numerical Analysis* **29,**1-18.

Shampine , L.F. and Gear, C.W. 1976: A User's View of Solving Ordinary Differential Equations . *Department of Computer Science University of Illinois at Urbana.*

Spijker, M.N 1995: Stiffness in numerical initial-value problems. *Journal of Computational and Applied Mathematics,***72(1996),** 393-406.

Kassam, A.K.and Trefethen, L.N. 2005: Fourth-order time stepping for stiff PDEs . *SIAM J. Sci. Comput.,* **26(4),** 1214-1233.

Trefethen, L.N. : Finite Difference and Spectral Methods for Ordinary and Partial Differential Equations.

Varah, J.M, 1980: Stability Restrictions on Second Order, Three Level Finite Difference Schemes for Parabolic Equations. *SIAM J. Numer. Anal.,* **17,** 300-309.

Yilmaz, Y., Girgin, Z. and Evran, S. 2013: Buckling Analyses of Axially Functionally Graded Nonuniform Columns with Elastic Restraint Using a Localized Differential Quadrature Method. *Math. Prob. Eng.,* **Vol. 2013** .

Zong, Z. and Lam, K.Y 2002: A localized differential differential quadrature(LDQ) method and its application to the 2D wave equation. *Comput. Mech.,* **382,**29-39.

Zong, Z. and Zhang, Y. 2009: Advanced Differential Quadrature Methods. *Chapmann & Hall,* **applied mathematics and nonlinear science series**

APPENDIX A

MATLAB CODES FOR NUMERICAL EXPERIMENTS

```
      %% ALLEN-CAHN EQUATION BY EXPONENTIAL INTEGRATOR
%%Ut=eps*Uxx+U-U^3%%
%%U(x,0)=0.53*x+0.47*sin(-1.5*pi*x)%%
clear all;close all;clc
for e=1:5
N=50;
hx=2/N;
x=-1:hx:1;
l=0.001;
Nt=50*2^(e-1);
step(e)=Nt;
ht=10/Nt;
t=0:ht:10;
 A=l*((1/hx)^2)*fin(N);
y(1,:)=0.53*x+0.47*sin(-1.5*pi*x);
u(:,1)=y(1,2:N)';
for i=1:Nt
      jac=A;
      [V,D]=eig(ht*jac);
      d=diag(D);
      g(:,i)=u(:,i) - (u(:,i)).^3;
      u(:,i+1)=expm(jac*ht)*u(:,i)
      +ht*V*diag(phi1(d,ht,1))*inv(V)*(g(:,i));
end
v1(:,1:Nt+1)=-1;
v2(:,1:Nt+1)=1;
ua=vertcat(v1,u,v2);
if e==3
figure;
[X,Y]=ndgrid(x,t);
surf(X, Y,ua, ...
```

```
    '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('Allen Cahn Exp-Euler,\gamma=0.001');
end
urk(:,1)=y(1,2:N)';
for j=1:Nt
    jac=A;
    [V,D]=eig(ht*jac);
    d=diag(D);
    U1(:,1)=urk(:,j);
    U2(:,1)=expm((.5)*jac*ht)*urk(:,j)+
    ht*V*((.5)*diag(phi1(d,ht,2)))*inv(V)*G_tez(U1);
    urk(:,j+1)=expm(jac*ht)*urk(:,j)+
    ht*(V*diag(phi1(d,ht,1)-2*phi2(d,ht,1))*inv(V)*G_tez(U1)+
    V*diag(2*phi2(d,ht,1))*inv(V)*G_tez(U2));
end
v1(:,1:Nt+1)=-1;
v2(:,1:Nt+1)=1;
u1=vertcat(v1,urk,v2);
er1=norm(abs(u1(:,Nt+1)-ua(:,Nt+1)),1)
er2=norm(abs(u1(:,Nt+1)-ua(:,Nt+1)),2)
erinf=norm(abs(u1(:,Nt+1)-ua(:,Nt+1)),inf)
u11(e)=max(er1);
u22(e)=max(er2);
u33(e)=max(erinf);
dt(e)=ht;
if e>1
order1(e)=abs((log(u11(e)/u11(e-1)))/(log(dt(e)/dt(e-1))));
order2(e)=abs((log(u22(e)/u22(e-1)))/(log(dt(e)/dt(e-1))));
```

```
order3(e)=abs((log(u33(e)/u33(e-1)))/(log(dt(e)/dt(e-1))));
else
order1(e)=0;
order2(e)=0;
order3(e)=0;
end
ht
end
figure
plot(log10(step),log10(u11),'-.m*',...
'LineWidth',2)
hold all
plot(log10(step),log10(u22),'-.g*',...
'LineWidth',2)
hold all
plot(log10(step),log10(u33),'-.r*',...
'LineWidth',2)
grid on
xlabel('LOG(N.EVAL)')
ylabel('LOG(ERROR)')
legend('L-1 norm','L-2 norm','L-inf norm')
title('Accuracy of u for exponential Euler scheme ')
figure
plot(x,ua(:,81),'-.m',...
 'LineWidth',2)
hold all
plot(x,ua(:,161),'-.g',...
 'LineWidth',2)
hold all
plot(x,ua(:,481),'-.b',...
 'LineWidth',2)
hold all
plot(x,ua(:,end),'-.r',...
 'LineWidth',2)
xlabel('x-axis ')
ylabel('Computed Solutions')
hold off
```

```
legend('t=1','t=2','t=6','t=10')
erroreuler1=u11
erroreuler2=u22
erroreuler3=u33
ordereuler1=order1
ordereuler2=order2
ordereuler3=order3
%% BURGERS EQUATION BY EXPONENTIAL INTEGRATOR
%%Ut=kappa*Uxx-UUx%%
 %%U(x,0)=exp(-10*((sin(x/2)).^2))%%
 for e=1:5
 eps=0.03;
 N=64;
 hx=2*pi/N;
 x=-pi:hx:pi;
 Nt=50*2^(e-1);
 ht=1/Nt;
 step(e)=Nt;
 t=0:ht:1;
 AA=fin(N);
 A=eps*AA/(hx^2);
 CC=fin2(N);
 C=CC/(hx*2);
 f(1,1:N+1)=exp(-10*((sin(x/2)).^2));
 u(1:N-1,1)=(f(1,2:N))';
 for i=1:Nt
   jac=A;
   [V,D]=eig(ht*jac);
    d=diag(D);
    g(:,i)=-u(:,i).*(C*u(:,i)) ;
    u(1:N-1,i+1)=expm(ht*jac)*u(:,i)+
    ht*V*diag(phi1(d,ht,1))*inv(V)*(g(:,i));%%%%exp.eulerr
    end
v1(:,1:Nt+1)=0;
v2(:,1:Nt+1)=0;
k3=[v1;u;v2];
usol=real(k3);
```

```
figure ;
[X ,T]=ndgrid(x,t);
surf(X, T,usol, ...
        '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 via exponential Euler, \kappa=0.03');
figure;
plot(x,usol(:,41),'r*-')
hold all,
plot(x,usol(:,81),'bd-')
hold all
plot(x,usol(:,101),'gs-')
hold all
plot(x,usol(:,301),'co-')
urk(:,1)=u(1:N-1,1);
for j=1:Nt
       jac=A;
       [V,D]=eig(ht*jac);
       d=diag(D);
       U1(:,1)=urk(:,j);
       U2(:,1)=expm((.5)*jac*ht)*urk(:,j)+
       ht*V*diag(phi1(d,ht,2))*inv(V)*FT(U1,C);
       urk(:,j+1)=expm(jac*ht)*urk(:,j)+
       ht*(V*diag(phi1(d,ht,1)-2*phi2(d,ht,1))*inv(V)*FT(U1,C)+
       V*diag(2*phi2(d,ht,1))*inv(V)*FT(U2,C));
end
v1(:,1:Nt+1)=0;
v2(:,1:Nt+1)=0;
u1=vertcat(v1,urk,v2);
```

```
usol2=real(u1);
er1=norm(abs(usol2(:,Nt+1)-usol(:,Nt+1)),1)
er2=norm(abs(usol2(:,Nt+1)-usol(:,Nt+1)),2)
erinf=norm(abs(usol2(:,Nt+1)-usol(:,Nt+1)),inf)
u11(e)=max(er1);
u22(e)=max(er2);
u33(e)=max(erinf);
dt(e)=ht;
if e>1
order1(e)=abs((log(u11(e)/u11(e-1)))/(log(dt(e)/dt(e-1))));
order2(e)=abs((log(u22(e)/u22(e-1)))/(log(dt(e)/dt(e-1))));
order3(e)=abs((log(u33(e)/u33(e-1)))/(log(dt(e)/dt(e-1))));
else
order1(e)=0;
order2(e)=0;
order3(e)=0;
end
ht
end
figure
plot(log10(step),log10(u11),'-.m*',...
'LineWidth',2)
hold all
plot(log10(step),log10(u22),'-.g*',...
'LineWidth',2)
hold all
plot(log10(step),log10(u33),'-.r*',...
'LineWidth',2)
grid on
xlabel('LOG(N.EVAL)')
ylabel('LOG(ERROR)')
legend('L-1 norm','L-2 norm','L-inf norm')
title('Accuracy of u for exponential Euler scheme ')
figure
plot(x,usol(:,11),'-.m',...
 'LineWidth',2)
hold all
```

```
plot(x,usol(:,21),'-.g',...
 'LineWidth',2)
hold all
plot(x,usol(:,61),'-.b',...
 'LineWidth',2)
hold all
plot(x,usol(:,end),'-.r',...
 'LineWidth',2)
xlabel('x-axis ')
ylabel('Computed Solutions')
hold off
legend('t=1','t=2','t=6','t=10')
erroreuler1=u11
erroreuler2=u22
erroreuler3=u33
ordereuler1=order1
ordereuler2=order2
ordereuler3=order3
%% BURGERS EQUATION BY EXPONENTIAL INTEGRATOR
%%Ut=kappa*Uxx-UUx%%
%%U(x,0)=4*x.*(1-x)%%
for e=1:5
    eps=0.1;
    N=40;
    hx=2/N;
    x=0:hx:1;
    Nt=50*2^(e-1);
    ht=1/Nt;
    step(e)=Nt;
    t=0:ht:1;
    AA=fin(N);
    A=eps*AA/(hx^2);
    CC=fin2(N);
    C=CC/(hx*2);
    f(1,1:N+1)=4*x.*(1-x);
    u(1:N-1,1)=(f(1,2:N))';
    for i=1:Nt
```

```
    jac=A;
    [V,D]=eig(ht*jac);
    d=diag(D);
    g(:,i)=-u(:,i).*(C*u(:,i)) ;
    u(1:N-1,i+1)=expm(ht*jac)*u(:,i)+
    ht*V*diag(phi1(d,ht,1))*inv(V)*(g(:,i));%%%%exp.eulerr
end
v1(:,1:Nt+1)=0;
v2(:,1:Nt+1)=0;
k3=[v1;u;v2];
usol=real(k3);
figure ;
[X ,T]=ndgrid(x,t);
surf(X, T,usol, ...
     '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 via exponential Euler, \kappa=0.1');
save('burger_exp.mat','usol')
figure;
plot(x,usol(:,41),'r*-')
hold all,
plot(x,usol(:,81),'bd-')
hold all
plot(x,usol(:,101),'gs-')
hold all
plot(x,usol(:,301),'co-')
urk(:,1)=u(1:N-1,1);
for j=1:Nt
        jac=A;
```

```
         [V,D]=eig(ht*jac);
         d=diag(D);
         U1(:,1)=urk(:,j);
         U2(:,1)=expm((.5)*jac*ht)*urk(:,j)+
        ht*V*diag(phi1(d,ht,2))*inv(V)*FT(U1,C);
        urk(:,j+1)=expm(jac*ht)*urk(:,j)+
        ht*(V*diag(phi1(d,ht,1)-2*phi2(d,ht,1))*inv(V)*FT(U1,C)+
        V*diag(2*phi2(d,ht,1))*inv(V)*FT(U2,C));
end
v1(:,1:Nt+1)=0;
v2(:,1:Nt+1)=0;
u1=vertcat(v1,urk,v2);
usol2=real(u1);
 er1=norm(abs(usol2(:,Nt+1)-usol(:,Nt+1)),1)
er2=norm(abs(usol2(:,Nt+1)-usol(:,Nt+1)),2)
erinf=norm(abs(usol2(:,Nt+1)-usol(:,Nt+1)),inf)
u11(e)=max(er1);
u22(e)=max(er2);
u33(e)=max(erinf);
dt(e)=ht;
if e>1
order1(e)=abs((log(u11(e)/u11(e-1)))/(log(dt(e)/dt(e-1))));
order2(e)=abs((log(u22(e)/u22(e-1)))/(log(dt(e)/dt(e-1))));
order3(e)=abs((log(u33(e)/u33(e-1)))/(log(dt(e)/dt(e-1))));
else
    order1(e)=0;
    order2(e)=0;
    order3(e)=0;
    end
    ht
end
figure
plot(log10(step),log10(u11),'-.m*',...
'LineWidth',2)
hold all
plot(log10(step),log10(u22),'-.g*',...
'LineWidth',2)
```

```
hold all
plot(log10(step),log10(u33),'-.r*',...
'LineWidth',2)
grid on
xlabel('LOG(N.EVAL)')
ylabel('LOG(ERROR)')
legend('L-1 norm','L-2 norm','L-inf norm')
title('Accuracy of u for exponential Euler scheme ')
figure
plot(x,usol(:,11),'-.m',...
 'LineWidth',2)
hold all
plot(x,usol(:,21),'-.g',...
 'LineWidth',2)
hold all
plot(x,usol(:,61),'-.b',...
 'LineWidth',2)
hold all
plot(x,usol(:,end),'-.r',...
 'LineWidth',2)
xlabel('x-axis ')
ylabel('Computed Solutions')
hold off
legend('t=1','t=2','t=6','t=10')
erroreuler1=u11
erroreuler2=u22
erroreuler3=u33
ordereuler1=order1
ordereuler2=order2
ordereuler3=order3
FINITE DIFFERENCE FUNCTIONS
function A=fin(N)
A=zeros(N-1,N-1);
for i=1:N-1
    for j=1:N-1
        if i==j
            A(i,j)=-2;
```

```
        end
        if (i-j)==1
            A(i,j)=1;
        end
        if (i-j)==-1
            A(i,j)=1;
        end
    end
end
A;
function B=fin2(N)
%A=zeros(N-1,N-1);
for i=1:N-1
    for j=1:N-1
    if (i-j)==1
            B(i,j)=-1;
        end
        if (i-j)==-1
            B(i,j)=1;
        end
    end
end
B;
ORDER TRIANGLE
function ordertriangle(order, varargin)
     if(nargin==2)
         b_loglog = varargin{1};
     else
         b_loglog = false;
     end
     if(nargin==3)
         color = varargin{2};
     else
         color = 'k';
     end
     [x y] = ginput(2);
     posinit = struct('x', x(1), 'y', y(1));
```

```
    width   = x(2)-x(1);

    if(b_loglog)
        a = y(1)/( x(1)^order);
        posy = a* x(2)^order;
        posxt= sqrt(x(2)*x(1));
        posyt= a* (posxt)^order;

    else
        posy = (posinit.y+width*order);
        posxt = posinit.x+width/2;
        posyt = posinit.y+width/2*order;
    end

    if(order>0)
        text(posxt, posyt, sprintf('%i', order),...
                'VerticalAlignment','bottom',...
                'HorizontalAlignment','right');
    else
        text(posxt, posyt, sprintf('%i', -order),..
.            'VerticalAlignment','top',...
            'HorizontalAlignment','right');
    end
    line([posinit.x (posinit.x+width)
    (posinit.x+width) posinit.x],...
         [posinit.y posy (posinit.y) posinit.y],...
         'Color', color);

end
ALLEN-CAHN EQUATION VIA ITERATIVE
Ut=eps*Uxx+U-U^3%%
U(x,0)=0.53*x+0.47*sin(-1.5*pi*x)%%
clear all;
close all;
clc
itermax=1000;
x0=-1;
```

```
xn=1;
nx=101;
dx=(xn-x0)/(nx-1);
x=x0:dx:xn;
x=x';
%%
t0=0;
tn=1;
nt=101;
dt=(tn-t0)/(nt-1);
t=t0:dt:tn;
%%
gama=0.001;
cr=dt/dx;
pe=dx/gama;
%%
temp=zeros(nx,1);
teta=zeros(nx,1);
cy=zeros(nx,1);
ce=0.53.*x+0.47.*sin(-1.5.*pi.*x);
u(:,1)=ce;
%%
temp(1)=-1;temp(nx)=1;
teta(1)=0;teta(nx)=0;
cy(2:nx-1,1)=0;
%%
[a,b]= LDQ10(nx,dx);
for i=1:nt-1
ce(1)=-1;ce(nx)=1;
cy(1)=-1;cy(nx)=1;
for j=1:itermax
      amat=(1/dt)*eye(nx)-(0.5*gama)*b-(0.5)*eye(nx)
      -(1.5)*diag(((cy+ce)/2).^2);
      bmat=-(1/dt)*(cy-ce)+(0.5*gama)*b*(cy+ce)
      +(0.5)*(cy+ce)-((cy+ce)/2).^3;
      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
      u(:,i+1)=cy;
      ce=cy;
end
%% plot
v1(:,1:nt)=-1;
v2(:,1:nt)=1;
usol=vertcat(v1,u(2:nx-1,:),v2);
[T X]=meshgrid(t,x);
surf(X, T,usol, ...
     '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('Allen Cahn,\gamma=0.001,\Delta x=0.04,\Delta t=0.1');
BURGERS EQUATION VIA LDQ
Ut+UU_x=epsU_{xx}%%
U(x,0)=exp(-10*(\sin(x/2))^2) , -\pi<x<\pi %%
U(0,t)=U(1,t)=0 , 0<t<1%%
clear all
close all
clc
%% Space discretization
x0=-pi;
xn=pi;
nx=65;
```

```
dx=(xn-x0)/(nx-1);
x=x0:dx:xn;
x=x';
%%
gama=0.03;
%% time
t0=0;
tn=1;
nt=101;
dt=(tn-t0)/(nt-1);
t=t0:dt:tn;
%%
cr=dt/dx
pe=dx/gama
%%
itermax=1000;
%%
temp=zeros(nx,1);
teta=zeros(nx,1);
cy=zeros(nx,1);
 %%
ce=exp(-10*(sin(x/2)).^2);
u(:,1)=ce;
 %%
temp(1)=0;temp(nx)=0;
teta(1)=0;teta(nx)=0;
cy(2:nx-1,1)=0;
 %%
[a,b]= LDQ10(nx,dx);
 %% Loop
for i=1:nt-1
  ce(1)=0;ce(nx)=0;
  cy(1)=0;cy(nx)=0;
  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)*b;
```

```
      bmat=-(1/dt)*(cy-ce)-(0.25)*(cy+ce).*(a*(cy+ce))
      +(0.5*gama)*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
          u(:,i+1)=cy;
      ce=cy;
end
%% Plot
figure
plot(x,u(:,21),'r*-')
hold all
plot(x,u(:,41),'bo-')
hold all
plot(x,u(:,61),'gd-')
hold all
plot(x,u(:,81),'ys-')
hold all
plot(x,u(:,101))
figure ;
[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 \kappa=0.03');
%%BURGERS EQUATION 2 LDQ%%
%%Ut+UUx=epsUxx%%
%%U(x,0)=4x(1-x) , 0<x<1 %%
%%U(0,t)=U(1,t)=0 , 0<t<1%%
clear all
close all
clc
%% Space discretization
x0=0;
xn=1;
nx=41;
dx=(xn-x0)/(nx-1);
x=x0:dx:xn;
x=x';
%%
gama=0.01;
%% time
t0=0;
tn=3;
nt=3*10^2+1;
dt=(tn-t0)/(nt-1);
t=t0:dt:tn;
%%
cr=dt/dx
pe=dx/gama
 %%
itermax=1000;
 %%
temp=zeros(nx,1);
teta=zeros(nx,1);
cy=zeros(nx,1);
 %%
ce=4*x.*(1-x);
u(:,1)=ce;
 %%
temp(1)=0;temp(nx)=0;
```

```
teta(1)=0;teta(nx)=0;
cy(2:nx-1,1)=0;
 %%
[a,b]= LDQ10(nx,dx);
 %% Loop
for i=1:nt-1
ce(1)=0;ce(nx)=0;
cy(1)=0;cy(nx)=0;
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)*b;
      bmat=-(1/dt)*(cy-ce)-(0.25)*(cy+ce).*(a*(cy+ce))
      +(0.5*gama)*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
 u(:,i+1)=cy;
 ce=cy;
  end
 %% Plot
 figure;
 plot(x,u(:,41),'r*-')
 hold all,
 plot(x,u(:,81),'bd-')
 hold all
 plot(x,u(:,101),'gs-')
 hold all
 plot(x,u(:,301),'co-')
 axis tight
 figure ;
 [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 ,\kappa=0.1,u(x,0)=4x(1-x)');
%%LDQ FUNCTION
function [a,b]= LDQ10(n,dx)
a=zeros(n,n);
b=zeros(n,n);
i=1;
a(i,i)=-7381;a(i,i+1)=25200;a(i,i+2)=-56700;
a(i,i+3)=100800;a(i,i+4)=-132300;
a(i,i+5)=127008;a(i,i+6)=-88200;
a(i,i+7)=43200;
a(i,i+8)=-14175;a(i,i+9)=2800;
a(i,i+10)=-252;
i=2;
a(i,i-1)=-252;a(i,i)=-4609;a(i,i+1)=11340;
a(i,i+2)=-15120;a(i,i+3)=17640;
a(i,i+4)=-15876;a(i,i+5)=10584;
a(i,i+6)=-5040;
a(i,i+7)=1620;a(i,i+8)=-315;
a(i,i+9)=28;
i=3;
a(i,i-2)=28; a(i,i-1)=-560;
a(i,i)=-3069;a(i,i+1)=6720;a(i,i+2)=-5880;
a(i,i+3)=4704;a(i,i+4)=-2940;
a(i,i+5)=1344;
a(i,i+6)=-420;a(i,i+7)=80;a(i,i+8)=-7;
i=4;
```

```
a(i,i-3)=-7; a(i,i-2)=105;
a(i,i-1)=-945;a(i,i)=-1914;a(i,i+1)=4410;
a(i,i+2)=-2646;a(i,i+3)=1470;
a(i,i+4)=-630;
a(i,i+5)=189;a(i,i+6)=-35;
a(i,i+7)=3;
i=5;
a(i,i-4)=3; a(i,i-3)=-40;
a(i,i-2)=270;a(i,i-1)=-1440;a(i,i)=-924;
a(i,i+1)=3024;a(i,i+2)=-1260;
a(i,i+3)=480;
a(i,i+4)=-135;a(i,i+5)=24;
a(i,i+6)=-2;
for i=6:n-5
a(i,i-5)=-2; a(i,i-4)=25;
a(i,i-3)=-150;a(i,i-2)=600;a(i,i-1)=-2100;
a(i,i)=0;a(i,i+1)=2100;
a(i,i+2)=-600;
a(i,i+3)=150;a(i,i+4)=-25;
a(i,i+5)=2;
end
i=n-4;
a(i,i+4)=-3; a(i,i+3)=40;
a(i,i+2)=-270;a(i,i+1)=1440;a(i,i)=924;
a(i,i-1)=-3024;a(i,i-2)=1260;
a(i,i-3)=-480;
a(i,i-4)=135;a(i,i-5)=-24;a(i,i-6)=2;
i=n-3;
a(i,i+3)=7; a(i,i+2)=-105;
a(i,i+1)=945;a(i,i)=1914;a(i,i-1)=-4410;
a(i,i-2)=2646;a(i,i-3)=-1470;
a(i,i-4)=630;
a(i,i-5)=-189;a(i,i-6)=35;
a(i,i-7)=-3;
i=n-2;
a(i,i+2)=-28; a(i,i+1)=560;
a(i,i)=3069;a(i,i-1)=-6720;a(i,i-2)=5880;
```

```
a(i,i-3)=-4704;a(i,i-4)=2940;
a(i,i-5)=-1344;
a(i,i-6)=420;a(i,i-7)=-80;a(i,i-8)=7;
i=n-1;
a(i,i+1)=252;a(i,i)=4609;a(i,i-1)=-11340;
a(i,i-2)=15120;a(i,i-3)=-17640;
a(i,i-4)=15876;a(i,i-5)=-10584;
a(i,i-6)=5040;
a(i,i-7)=-1620;a(i,i-8)=315;
a(i,i-9)=-28;
i=n;
a(i,i)=7381;a(i,i-1)=-25200;
a(i,i-2)=56700;a(i,i-3)=-100800;a(i,i-4)=132300;
a(i,i-5)=-127008;a(i,i-6)=88200;
a(i,i-7)=-43200;
a(i,i-8)=14175;a(i,i-9)=-2800;
a(i,i-10)=252;
a=a./(2520*dx);
i=1;
b(i,i)=177133;b(i,i+1)=-972200;
b(i,i+2)=2754450;b(i,i+3)=-5232800;b(i,i+4)=7088550;
b(i,i+5)=-6932016;b(i,i+6)=4872700;
b(i,i+7)=-2407200;
b(i,i+8)=794925;b(i,i+9)=-157800;
b(i,i+10)=14258
i=2;
b(i,i-1)=14258;b(i,i)=20295;
b(i,i+1)=-188010;b(i,i+2)=401880;b(i,i+3)=-527660;
b(i,i+4)=501354;b(i,i+5)=-344820;
b(i,i+6)=167560;
b(i,i+7)=-54630;b(i,i+8)=10735;b(i,i+9)=-962;
i=3;
b(i,i-2)=-962;b(i,i-1)=24840;
b(i,i)=-32615;b(i,i+1)=-29280;b(i,i+2)=84420;
b(i,i+3)=-83216;b(i,i+4)=56910;
b(i,i+5)=-27360;
b(i,i+6)=8830;b(i,i+7)=-1720;
```

```
b(i,i+8)=153;
i=4;
b(i,i-3)=153;b(i,i-2)=-2645;
b(i,i-1)=33255;b(i,i)=-57860;b(i,i+1)=21210;
b(i,i+2)=13734;b(i,i+3)=-12530;
b(i,i+4)=6420;
b(i,i+5)=-2115;b(i,i+6)=415;
b(i,i+7)=-37;
i=5;
b(i,i-4)=-37; b(i,i-3)=560;
b(i,i-2)=-4680;b(i,i-1)=39360;b(i,i)=-70070;
b(i,i+1)=38304;b(i,i+2)=-3360;
b(i,i+3)=-320;
b(i,i+4)=315;b(i,i+5)=-80;b(i,i+6)=8;
for i=6:n-5
b(i,i-5)=8;b(i,i-4)=-125;
b(i,i-3)=1000;b(i,i-2)=-6000;b(i,i-1)=42000;
b(i,i)=-73766;b(i,i+1)=42000;b(i,i+2)=-6000;
b(i,i+3)=1000;b(i,i+4)=-125;b(i,i+5)=8;
end
i=n-4;
b(i,i+4)=-37; b(i,i+3)=560;
b(i,i+2)=-4680;b(i,i+1)=39360;b(i,i)=-70070;
b(i,i-1)=38304;b(i,i-2)=-3360;
b(i,i-3)=-320;
b(i,i-4)=315;b(i,i-5)=-80;b(i,i-6)=8;
i=n-3;
b(i,i+3)=153;b(i,i+2)=-2645;
b(i,i+1)=33255;b(i,i)=-57860;
b(i,i-1)=21210;b(i,i-2)=13734;
b(i,i-3)=-12530;b(i,i-4)=6420;
b(i,i-5)=-2115;b(i,i-6)=415;
b(i,i-7)=-37;
i=n-2;
b(i,i+2)=-962;b(i,i+1)=24840;
b(i,i)=-32615;b(i,i-1)=-29280;
b(i,i-2)=84420;b(i,i-3)=-83216;
```

```
b(i,i-4)=56910;
b(i,i-5)=-27360;b(i,i-6)=8830;
b(i,i-7)=-1720;b(i,i-8)=153;
i=n-1;
b(i,i+1)=14258;b(i,i)=20295;
b(i,i-1)=-188010;b(i,i-2)=401880;
b(i,i-3)=-527660;b(i,i-4)=501354;b(i,i-5)=-344820;
b(i,i-6)=167560;b(i,i-7)=-54630;b(i,i-8)=10735;b(i,i-9)=-962;
i=n;
b(i,i)=177133;b(i,i-1)=-972200;
b(i,i-2)=2754450;b(i,i-3)=-5232800;
b(i,i-4)=7088550;b(i,i-5)=-6932016;
b(i,i-6)=4872700;
b(i,i-7)=-2407200;b(i,i-8)=794925;
b(i,i-9)=-157800;b(i,i-10)=14258;
b=b./(25200*dx*dx);
end
```