NUMERICAL SOLUTIONS TO PARAXIAL WAVE EQUATIONS

by

COLBY MROSS

A THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in the
Mathematics Graduate Program of
Delaware State University

DOVER, DELAWARE
August 2018

This thesis is approved by the following members of the Final Oral Review Committee:
Dr. Jinjie Liu, Committee Chairperson, Division of Physical and Computational Science, Delaware State University
Dr. Mathew Tanzy, Committee Member, Division of Physical and Computational Science, Delaware State University
Dr. Sokratis Makrogiannis, Committee Member, Division of Physical and Computational Science, Delaware State University
Dr. Aristides Marcano, External Committee Member, Division of Physical and Computational Science, Delaware State University

DEDICATION

This thesis is dedicated to my family who supported me throughout this process. It is especially dedicated to my wife, Chelsea, for her encouragement and support. She believed in me every step of the way. It is also dedicated to my two loving children, Colby and Emily.

ACKNOWLEDGEMENTS

First of all I would like to thank my advisor Dr. Jinjie Liu who supported me in every problem that occurred during the process of writing this report. I have high regard for Dr. Liu as a person and as a thinker, and thank him for all of his time. Lastly, I would like to thank all the professor at Delaware State University that I encountered though my graduate studies, especially my defense committee for putting their valuable time and effort on my thesis.

ABSTRACT

In this thesis, we apply various numerical methods to solve ordinary differential equations and the paraxial wave equation. The numerical methods we applied to solving paraxial wave equation are the 4th order Runge Kutta method, the Crank-Nicolson method, the Leapfrog Crank-Nicolson method, and the splitting spectrum method. The advantage of the explicit RK4 method is the high order accuracy in time. We perform detailed comparison between these numerical methods. The paraxial wave equation is derived from Maxwell's equation and we focus on the case of cubic Kerr nonlinearity presents, which is applied to study optical pulse propagation in nonlinear Kerr media. The Leapfrog Crank-Nicolson method, being an implicit method, is the most cost efficient method and when choosing small step sizes can be the most accurate when applied to paraxial wave equations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES AND ILLUSTRATIONS

Chapter I: INTRODUCTION

The nonlinear Schrodinger Equation (NLSE) is an important equation for studying the dynamics of optical pulses [3, 6] and Bose-Einstein condensation [9]. Recent investigation on the propagation of light pulses through optical fibers have lead to the interest in the study of nonlinear Schrodinger Equation (NLSE) [2]. Some applications of NLSE are wave interaction between electrons and ions, chemical interaction based on the first principle, nonlinear optics, protein folding, and the flow of superfluids without friction. In optics NLSE mimic the pulses which preserve their shape inside fibers despite their dispersive effects. The equation could describe the wave group envelope of such pulses in material, and it could also describe the solutions propagating in fluid [17]. NLSE describes the propagation of short optical pulses as fundamental and higher order solutions inside optical fibers. When two or more optical waves at different wavelengths are launched inside a fiber simultaneously, the several new nonlinear effects become important [1]. In dispersive nonlinear media, optical pulse has self-focusing and self-steepening effect [12, 15]. Numerical tools are key to understand these phenomenons [8, 13, 10].

In this thesis, we first review basic numerical Runge Kutta methods for solving ordinary differential equation (ODE) initial value problem, then we focus on the paraxial wave equation. The numerical method we applied to solving paraxial wave equation are the 4th order Runge Kutta method, the Crank-Nicolson method, the Leapfrog Crank-Nicolson method, and the splitting spectrum method. We performed detailed comparison between these numerical methods. The paraxial wave equation is derived from Maxwell's equation and we focus on the case where the cubic Kerr nonlinearity presents. This paraxial wave equation is applied to investigate the self focusing of optical pulse in nonlinear media. The paraxial wave equation can serve as a tool to help us to study the nonlinear optical pulse propagation in nonlinear media using Maxwell solvers such as the Finite-Difference Time-Domain (FDTD)

method [16, 14]. Using FDTD method to solve the Maxwell equations for long range pulse propagation (millions of wavelength or longer) is a very challenging task. This mainly due to the small time step from the explicit finite difference algorithm. Progress was made to archive this goal, such as the moving frame FDTD method [7, 11]. The solution of the paraxial wave equation can be used as a reference solution to help us on developing feasible FDTD based pulse propagator.

The thesis is organized as follows. In chapter 2, we will review the basic Runge-Kutta method for solving ordinary differential equations. In chapter 3, we will first derive the paraxial wave equation from Maxwell's equations under the paraxial assumption. Numerical examples on solving paraxial wave equation will be presented in chapter 4. We will comparatively show the difference between the various numerical methods and their accuracy discrepancies. Conclusion and future work is discussed in Chapter 5.

Chapter II: NUMERICAL METHODS FOR DIFFERENTIAL EQUATIONS

In this chapter we will describe the various numerical methods used throughout this paper. Along with a description we will derive, explain, and give an algorithm for each method. These numerical methods will be used to solve the following Ordinary Differential Equation (ODE) initial value problem:

$$y' = f(t, y), \qquad a \leq t \leq b, \qquad y(a) = \alpha. \tag{2.1}$$

2.1 Euler's Method

When it comes to solving ODE initial value problem the Euler's Method is the most elementary approach. Euler's Method is a first order method and estimates the next approximations based on the rate of change at the current point. Euler's Method is derived directly from Taylor's Series Expansion in the from:

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2!}y''(t_i) + \frac{h^3}{3!}y'''(t_i) + \cdots + \frac{h^n}{n!}y^{(n)} + \frac{h^{n+1}}{(n+1)!}y^{(n)}(\tau), \tag{2.2}$$

where $h = t_{i+1} - t_i$ and τ is some value between (t_i, t_{i+1}). For instance, if $n = 1$, we get:

$$y(t_{i+1}) = y(t_i) + hy\prime(t_i) + \frac{1}{2!}h^2y''(\tau). \tag{2.3}$$

We can truncate the last term of the right hand side to obtain a first order approximation of the numerical update equation of the differential equation. Because $y(t)$ satisfies the ODE (2.1), we obtain the Euler's algorithm [5, 4],

$$y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)). \tag{2.4}$$

Let $w_i \approx y(t_i)$, for each i=1, 2,. . . ,N. Using the Euler Method an approximation of the numerical solutions of at t_i, i=1, 2,. . . ,N, can be obtained using the numerical method of:

$$w_0 = \alpha,\ w_{i+1} = w_i + hf(t_i, w_i), \qquad \text{for each} \quad i = 0, 1, \ldots, N-1. \tag{2.5}$$

The algorithm of Euler Method can be written as follows:

INPUT: endpoints a, b; integer N; initial condition α.

OUTPUT: approximation w to y at the $(N+1)$ values of t.

Step 1 Set $h = (b-a)/N$.

$t = a$;

$w = \alpha$;

OUTPUT (t, w). (The initial condition)

Step 2 For $i = 1, 2, ..., N$ do Steps 3,4.

Step 3 Set $w = w + hf(t, w)$;

$t = a + ih$.

Step 4 OUTPUT (t, w).

Step 5 STOP.

This method is very basic, direct and first order accurate, but can be less accurate than higher order methods. An approximation of the error is proportional to the step size h. Hence, a good approximation is obtained with a very small h which can take a long time and be computational expensive.

2.2 Runge Kutta method of order two

To improve the order of accuracy, the Euler's method can be extended to higher order. In this section, we review two second order methods, the modified Euler method and the midpoint method.

The Modified Euler Method approximates $f(t, y)$ at t_i using the average value of $f(t, y_i)$ and $f(t, \tilde{y})$ where $\tilde{y}$ is the solution of the Euler method. The modified Euler method can be written as:

$$w_0 = \alpha$$

$$w_{i+1} = w_i + \frac{h}{2}[f(t_i, w_i) + f(t_{i+1}, w_i + hf(t_i, w_i))], \quad \text{for } i = 0, 1, \ldots, N-1. \tag{2.6}$$

The algorithm for the modified Euler method follows:

INPUT: endpoints a, b; integer N; initial condition α.

OUTPUT: approximation w to y at the $(N+1)$ values of t.

Step 1 Set $h = (b-a)/N$.

$t = a$;

$w = \alpha$;

OUTPUT (t, w).

Step 2 For $i = 1, 2, ..., N$ do Steps 3,4.

Step 3 Set $w = w + \frac{h}{2}[f(t, w) + f(t + h, w + hf(t, w))]$;

$t = a + ih$.

Step 4 OUTPUT (t, w).

Step 5 STOP.

The midpoint method approximates the right hand of the differential equation $f(t, y)$ using $f(t, \tilde{y})$ where $\tilde{y}$ is the solution at $t_i + h/2$ using the Euler method. The midpoint method can be written as:

$$w_0 = \alpha$$

$$w_{i+1} = w_i + hf\Big(t_i + \frac{h}{2}, w_i + \frac{h}{2}f(t_i, w_i)\Big), \quad \text{for } i = 0, 1, \ldots, N-1. \tag{2.7}$$

The algorithm for the midpoint method follows:

INPUT: endpoints a, b; integer N; initial condition α.

OUTPUT: approximation w to y at the $(N+1)$ values of t.

Step 1 Set $h = (b-a)/N$.

$t = a$;

$w = \alpha$;

OUTPUT (t, w).

Step 2 For $i = 1, 2, ..., N$ do Steps 3,4.

Step 3 Set $w = w + hf\Big(t + \frac{h}{2}, w + \frac{h}{2}f(t,w)\Big)$;

$t = a + ih$.

Step 4 OUTPUT (t, w).

Step 5 STOP.

Both methods are the Runge Kutta methods of order two (RK2).

2.3 Runge Kutta method of order 3

A Runge Kutta method of order 3 (RK3) is given below.

Starting with:

$$w_0 = \alpha,$$

we get

$$w_{i+1} = w_i + \frac{1}{6}(k_1 + 4k_2 + k_3)h, \quad \text{for } i = 0, 1, \ldots, N-1, \tag{2.8}$$

where

$$k_1 = f(t_i, w_i),$$

$$k_2 = f(t_i + \frac{1}{2}h,\ w_i + \frac{1}{2}k_1 h),$$

$$k_3 = f(t_i + h,\ w_i - k_1 h + 2k_2 h).$$

The introduction of k_1, k_2, and k_3 in the code eliminates the need for successive nesting. This made the code very easy to comprehend and debug.

Below is the RK3 Algorithm:

INPUT: endpoints a, b; integer N; initial condition α.

OUTPUT: approximation w to y at the $(N+1)$ values of t.

Step 1 Set $h = (b-a)/N$.

$t = a$;

$w = \alpha$;

OUTPUT (t, w).

Step 2 For $i = 1, 2, ..., N$ do Steps 3,4.

Step 3 Set $k1 = f(t, w)$;

$k2 = f(t + h/2, w + hk1/2)$;

$k3 = f(t + h, w - hk1 + 2k2h)$;

$w = w + h * (k1 + 4k2 + k3)/6$;

$t = a + ih$.

Step 4 OUTPUT (t, w).

Step 5 STOP.

2.4 Fourth order Runge Kutta method (RK4)

The fourth order Runge Kutta method (RK4) is a very popular Runge-Kutta numerical method. This method is the fourth order method which generally produces an approximation with enough accuracy and larger stability region than RK2 and RK3. Similarly to the RK3 method we introduce k_1, k_2, k_3, and k_4. Here is the method:

$$w_0 = \alpha,$$

$$k_1 = hf(t_i, w_i),$$

$$k_2 = hf\Big(t_i + \frac{h}{2},\ w_i + \frac{1}{2}k_1\Big),$$

$$k_3 = hf\Big(t_i + \frac{h}{2},\ w_i + \frac{1}{2}k_2\Big),$$

$$k_4 = hf(t_{t+1},\ w_i + k_3),$$

$$w_{i+1} = w_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \quad \text{for } i = 0, 1, \ldots, N-1. \tag{2.9}$$

The higher order of numerical method generally give less error than lower order methods, because the truncation error is proportional to h^n and the smaller h is the more accurate the method is. If the exact solution of the ODE is a polynomial of order n or less, it would be solved exactly by RKn. For example, Euler's Method will solve a first degree polynomial (linear line) exactly and RK4 will solve a fourth degree polynomial (or less) exactly. Below is the coding algorithm for RK4.

Below is the RK4 Algorithm:

INPUT: endpoints a, b; integer N; initial condition α.

OUTPUT: approximation w to y at the $(N+1)$ values of t.

Step 1 Set $h = (b-a)/N$.

$t = a$;

$w = \alpha$;

OUTPUT (t, w).

Step 2 For $i = 1, 2, \ldots, N$ do Steps 3,4.

Step 3 Set $k1 = hf(t, w)$;

$k2 = hf(t + h/2, w + k1/2)$;

$k3 = hf(t + h/2, w + k2/2)$;

$k4 = hf(t + h, w + k3)$;

$w = w + (k1 + 2k2 + 2k3 + k4)/6$;

$t = a + ih$.

Step 4 OUTPUT (t, w).

Step 5 STOP.

In the chapter IV, we will show some numerical examples on the solution to ODE using Runge Kutta method, followed by the solution to paraxial wave equation using Runge Kutta and the comparison with some other commonly used numerical methods.

Chapter III: PARAXIAL WAVE EQUATIONS AND THEIR NUMERICAL SOLUTIONS

Recent investigation on the propagation of light pulses through optical fibers have lead to the interest in the study of nonlinear Schrodinger Equation (NLSE). An example of a NLSE is:

$$\partial_t \psi(\vec{x}, t) = -\frac{\varepsilon^2}{2}\nabla^2\psi + V(\vec{x})\psi + \beta|\psi|^2\psi, \tag{3.1}$$

where: t represents time, $\vec{x}(\in \Re^d)$ is spatial coordinate, $\psi(\vec{x}, t)$ is the complex-valued wave function, $V(\vec{x})$ is the real-valued external potential, β is a given interaction constant, $0 < \varepsilon \leq 1$ is the scaled Planck Constant [2].

In this chapter, we first derive the paraxial wave equation from Maxwell's equations and then review some of the commonly used numerical methods for solving paraxial wave equation, including the explicit Runge Kutta method, the implicit Crank-Nicolson method, the leapfrog Crank-Nicolson method, and the splitting spectrum method.

3.1 From Maxwell Equations to Paraxial Wave Equations

In this section our goal is to derive the paraxial wave equation from Maxwell equations. First, the Maxwell equations in time domain given by:

$$\nabla \times E = -\mu_0 \partial_t H, \tag{3.2}$$

$$\nabla \times H = \epsilon_0 n^2 \partial_t E, \tag{3.3}$$

where E and H are the electric and magnetic fields respectively, n is the refractive index, and ϵ_0 and μ_0 are vacuum permittivity and permeability respectively.

Using substitution on the system of equations from equation (3.2) and (3.3) we get:

$$\nabla \times \nabla \times E = -\mu_0 \epsilon_0 n^2 \partial_t^2 E. \tag{3.4}$$

Now using the paraxial approximation, we can simplify $\nabla \times \nabla \times E = \nabla(\nabla \cdot E) - \nabla^2 E \approx -\nabla^2 E$. Also applying a Fourier transform to $\partial_t^2 E$ to get $-\omega^2 E$ and substituting $\mu_0 \epsilon_0 = \frac{1}{c^2}$, we get:

$$\nabla^2 E + \frac{\omega^2}{c^2} n^2 E = 0, \tag{3.5}$$

Focusing on Equation (3.5) we split $\nabla^2 E$ to assist simplifying

$$\nabla^2 E = \nabla_\perp^2 E + \partial_{zz} E. \tag{3.6}$$

Consider the plane wave solution:

$$E = A e^{ikz}, \tag{3.7}$$

where $A = A(x, y, z)$ is the amplitude of the wave.

Evaluating for $\partial_{zz} E$ we get:

$$\partial_z E = A_z e^{ikz} + Aike^{ikz}, \tag{3.8}$$

and by the product rule we have:

$$\partial_{zz} E = A_{zz} e^{ikz} + 2A_z ike^{ikz} + A(ik)^2 e^{ikz}. \tag{3.9}$$

Using paraxial approximation, $A_{zz} = 0$, so the above equation is simplified to

$$\partial_{zz} E = (2ikA_z - k^2 A) e^{ikz}, \tag{3.10}$$

where $k = \frac{\omega}{c} n_0$.

For nonlinear media, using the fact that

$$n = n_0 + \Delta n, \tag{3.11}$$

where n_0 is the refractive index for linear part and Δn is the index for nonlinearity.

We have

$$n^2 = n_0^2 + 2n_0\Delta n + \Delta n^2. \tag{3.12}$$

In general, the nonlinear wave is much smaller than the linear wave, so $\Delta n << n_0$. When we square a small number it becomes ever smaller implying $\Delta n^2 \approx 0$ therefore

$$n^2 = n_0^2 + 2n_0\Delta n. \tag{3.13}$$

Substituting equations (3.6), (3.10), and (3.13) into equation (3.5), we get:

$$2ik\partial_z E + \nabla_\perp^2 E + 2k^2 \frac{\Delta n}{n_0} E = 0. \tag{3.14}$$

Equation (3.14) is a partial differential equation (PDE) in (x, y, z) space. In the next few sections, we focus on the numerical solution to the paraxial wave equation in (x, z) to study the wave propagation in z direction in Kerr nonlinear media where $\Delta n = n_2|E|^2$. We let $u = E$, $k = k_0 = n_0\omega_0/c$ and rewrite the paraxial wave equation as:

$$u_z = \frac{i}{2k_0} u_{xx} + \frac{i\omega_0}{c} n_2 |u|^2 u. \tag{3.15}$$

Let $c_1 = \frac{1}{2k_0}$ and $c_3 = \frac{\omega_0 n_2}{c}$, the equation (3.15) can be simply written as

$$u_t = ic_1 u_{xx} + ic_3 |u|^2 u. \tag{3.16}$$

3.2 RK4 for Paraxial Wave Equations

In Chapter 2 we introduced the RK4 algorithm for ODEs, in this chapter we will modify it to accommodate the paraxial wave equation (3.16).

Let $u(z,x) = u(n\Delta z, j\Delta x) = u_j^n$. We use centered finite difference to approximate u_{xx}, we get:

$$u_{xx} = \frac{u(n\Delta z,(j+1)\Delta x) - 2u(n\Delta z, j\Delta x) + u(n\Delta z,(j-1)\Delta x)}{\Delta x^2} \tag{3.17}$$

$$= \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2}. \tag{3.18}$$

Plugging it into the right hand side of equation (3.16) we get

$$f(u_j^n) = ic_1 \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2} + ic_3|u_j^n|^2 u_j^n, \tag{3.19}$$

For the left hand side of equation (3.15), we have

$$u_z = \frac{u((n+1)\Delta z, j\Delta x) - u(n\Delta z, j\Delta x)}{\Delta z} = \frac{u_j^{n+1} - u_j^n}{\Delta z}. \tag{3.20}$$

The incremental in z direction is decided according to the CFL condition for parabolic equation: $\Delta z = h = \Delta x^2/(2c_1)$. This is to guarantee stability.

The boundary condition is the Dirichlet boundary where $u(x = x_L, z) = u(x = x_R, z) = 0$.

The RK4 Algorithm for paraxial wave equation (3.15) is given below:

INPUT: computational domain $x_L < x < x_R$, $a < z < b$ and initial condition.

OUTPUT: solution $u(x)$ at each $z = b$.

Step 1

Set $\Delta x = (x_R - x_L)/N_x$, $h = \frac{\Delta x^2}{2c_1}$, $N = (b-a)/h$.

Initial value $w = w(z = a, x)$.

Step 2 For $i = 1, 2, \ldots, N$ do Steps 3.

Step 3 Set $k1 = hf(w)$;

$k2 = hf(w + k1/2)$;

$k3 = hf(w + k2/2)$;

$k4 = hf(w + k3)$;

$w = w + (k1 + 2k2 + 2k3 + k4)/6$;

Apply boundary condition.

Step 4 OUTPUT (x, w).

Step 5 STOP.

3.3 The Crank-Nicholson method

The Crank-Nicholson method is another popular method for NLSE or paraxial wave equation. Starting with the same paraxial equation

$$u_z = ic_1 u_{xx} + ic_3 |u|^2 u. \tag{3.21}$$

Using the similar finite difference approximation we get:

$$\frac{u_j^{n+1} - u_j^n}{\Delta z} = \frac{ic_1}{2}\left(\frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{\Delta x^2} + \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2}\right) + ic_3 |u_j^n|^2 \cdot \frac{u_j^{n+1} + u_j^n}{2}. \tag{3.22}$$

Here the approximation of u_{xx} uses u at both time levels n and $n + 1$. Rearranging terms and simplifying gives us:

$$u_j^{n+1} - u_j^n = \frac{ic_1 \Delta z}{2\Delta x^2}\left(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1} + u_{j+1}^n - 2u_j^n + u_{j-1}^n\right) + \frac{ic_3 \Delta z |u_j^n|^2}{2} u_j^{n+1} + \frac{ic_3 \Delta z |u_j^n|^2}{2} u_j^n, \tag{3.23}$$

$$= R\big(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1} + u_{j+1}^n - 2u_j^n + u_{j-1}^n\big) + b_j u_j^{n+1} + b_j u_j^n, \tag{3.24}$$

where $R = ic_1\Delta z/(2\Delta x^2)$ and $h_j = ic_3\Delta z|u_j^n|^2/2$.

Distributing and rearranging terms we get

$$-Ru_{j-1}^{n+1} + (1 + 2R - h_j)u_j^{n+1} - Ru_{j+1}^{n+1} = Ru_{j+1}^n + (1 - 2R + h_j)u_j^n + Ru_{j+1}^n. \tag{3.25}$$

The above equation is a tridiagonal system and we use Thomas Method [5] to solve it

$$a_j u_{j-1}^{n+1} + b_j u_j^{n+1} + c_j u_{j+1}^{n+1} = d_j(Thomas) \Rightarrow u_j^{n+1}. \tag{3.26}$$

The initial condition and boundary conditions are the same as for RK4 method. Crank Nicolson method is an implicit method so there is no CFL restriction, i.e., Δz can be chosen to be much larger than the one we used in RK4. However, usually larger Δz will result in larger numerical error.

3.4 Leapfrog Crank-Nicholson

The next numerical method is the Leapfrog Crank-Nicholson method which is very similar to the Crank-Nicholson method. The difference is instead of $\frac{u_j^{n+1} - u_j^n}{\Delta z}$ this method is going to skip over the u_j^n and go from u_j^{n-1} to u_j^{n+1} hence the name is leapfrog. Here is the Leapfrog Crank-Nicholson method

$$\frac{u_j^{n+1} - u_j^{n-1}}{2\Delta z} = \frac{ic_1}{2}\left(\frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{\Delta x^2} + \frac{u_{j+1}^{n-1} - 2u_j^{n-1} + u_{j-1}^{n-1}}{\Delta x^2}\right) + ic_3|u_j^n|^2 \cdot \frac{u_j^{n-1} + u_j^{n+1}}{2}. \tag{3.27}$$

Simplifying above equation similarly as we did for the Crank-Nicholson method we get

$$u_j^{n+1} - u_j^{n-1} = \frac{ic_1\Delta z}{\Delta x^2}\big(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1} + u_{j+1}^{n-1} - 2u_j^{n-1} + u_{j-1}^{n-1}\big) + ic_3\Delta z|u_j^n|^2(u_j^{n-1} + u_j^{n+1}), \tag{3.28}$$

which can be written as

$$-Ru_{j-1}^{n+1} + (1 + 2R - h_j)u_j^{n+1} - Ru_{j+1}^{n+1} = Ru_{j-1}^{n-1} + (1 - 2R + h_j)u_j^{n-1} + Ru_{j+1}^{n-1}, \tag{3.29}$$

where $R = ic_1\Delta z/\Delta x^2$ and $h_j = ic_3\Delta z|u_j^n|^2$. Using the Thomas Method as in the Crank-Nicholson method we can solve from here.

3.5 Splitting Spectral Method

The splitting spectral method is the next method used to solve the paraxial wave equation.

$$u_z = ic_1u_{xx} + ic_3|u|^2u. \tag{3.30}$$

The highlights of this method is to split the paraxial wave equation into two equations as such:

$$u_z = ic_3|u|^2u, \tag{3.31}$$

and

$$u_z = ic_1u_{xx}. \tag{3.32}$$

Equation (3.31) is solved in (x, z) domain

$$\frac{u_z}{u} = ic_3|u|^2, \tag{3.33}$$

so

$$\partial_z log(u) = ic_3|u|^2, \tag{3.34}$$

which leads to the solution

$$u^* = u^n e^{\Delta z i c_3 |u^n|^2}. \tag{3.35}$$

After solving equation (3.31), the updated value of u^* is used as the initial condition to solve equation (3.32) in spectrum domain. First, we perform a Fourier transform from x to ξ:

$$\hat{u}_z = -ic_1\xi^2\hat{u}, \tag{3.36}$$

where $\hat{u}$ is the solution in spectrum domain $\hat{u}^n = \mathcal{F}(u^*)$. It can be solved analytically:

$$\hat{u}^{n+1} = \hat{u}^n e^{-\Delta z i c_1 \xi^2}. \tag{3.37}$$

Then the solution is obtain by inverse Fourier transform:

$$u^{n+1} = \mathcal{F}^{-1}(\hat{u}^{n+1}). \tag{3.38}$$

Chapter IV: NUMERICAL EXAMPLES

In this chapter we will first show some numerical tests on solving ODEs, followed by the study of paraxial wave equation using the various numerical methods described in chapter 2 and 3.

4.1 Numerical Example on ODE

The first method we attempted was the first order Euler's Method for solving ODEs initial value problem

$$y' = y,\ 0 \leq t \leq 5,\ y(0) = 1. \tag{4.1}$$

We test convergence of the numerical results by adjusting the number of grid points (N value) because if we reference the algorithm from chapter 2 we see that the N is integrated into the "step size", h, where $h = \frac{5}{N}$. The step size is the common difference between each point. The smaller the common difference, h, ideally the less error in the approximation, so to make h smaller we simply increase N consequently giving us less error.

4.1.1 Euler Method

Reviewing some of these results of the numerical solutions, we know that the results should closely mimic the exact solution $u = u(0)e^t$. Looking at Figure 4.1 we can see that the Euler's Method with small N values result in larger error. Figure 4.1 shows how all five curves look with $0 \leq t \leq 5$. Figure 4.2 zooms in at $t = 4$ and shows the error of each curve and how when N is increase the error of each curve will decrease by getting closer to the exact solution. Table 4.1 shows the order of Euler's Method and we can see this method is of order one. In summary Euler's Method is not a very accurate method but will approach the correct answer very slowly when N is set to be a larger number.

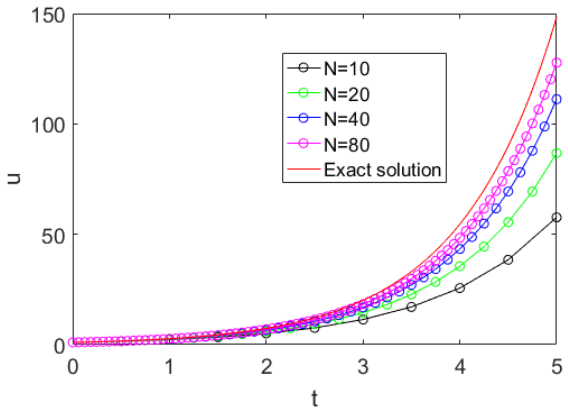

Figure 4.1: Euler's Method with N=10, 20, 40, and 80

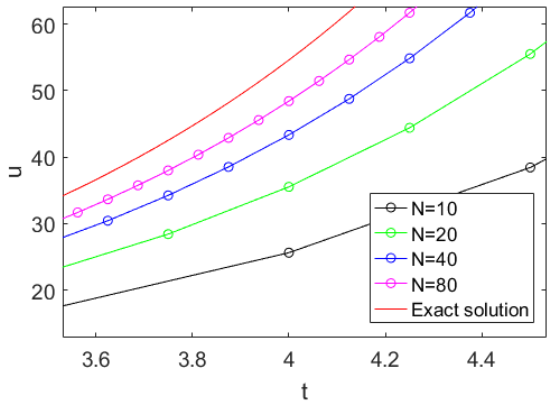

Figure 4.2: Euler's Method with N=10, 20, 40, and 80

4.1.2 RK4

RK4 was the next numerical method we tried. Similar to previous method, N values are chosen to be 10, 20, 40, and 80. We can see in Figure 4.3 that all four curves are fairly accurate. After adjusting the scale we can see the discrepancies of all four curves in Figure 4.4. As we expected when $N = 10$ the most error occurs which is shown in Table 4.2. Figures 4.5 and 4.6 are displayed to show how close the RK4 method can get to the exact solution depending how large we make N. Comparing these results to the Euler's Method we can see that with N equaling 10, 20, and 40 RK4 method was more accurate and approaching the exact solution quicker.

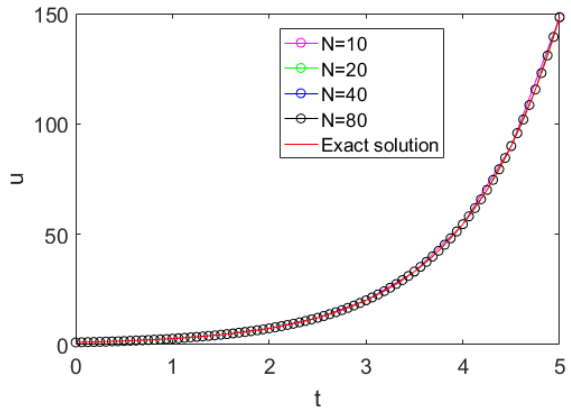

Figure 4.3: RK4 with N=10, 20, 40, and 80

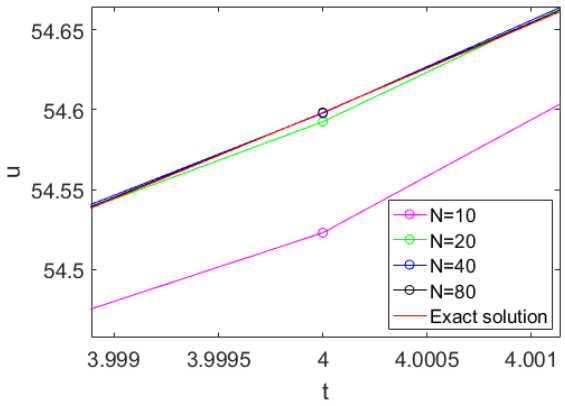

Figure 4.4: RK4 with N=10, 20, 40, and 80 zoomed in

4.1.3 Numerical Method Comparison

Next we compared the common numerical methods of Euler's Method, RK2, and RK4 in Figures 4.7, 4.8, and 4.9. We let N=40 in all four algorithms and plotted them all on Figure 4.7 with the exact solution. At first glance you will see the outlier of Euler's Method which is the purple curve and it is much lower than the other four curves. I zoomed in slightly on Figure 4.8 and you will see that the curve of Euler's Method is about 15 units lower than the other four. In Figure 4.9 we adjusted the scaling further and you can see the difference of RK2 and RK4. This error is attributed to the order of each method. Euler's being only order one has the most error, RK2 is a second order method, and RK4 is fourth

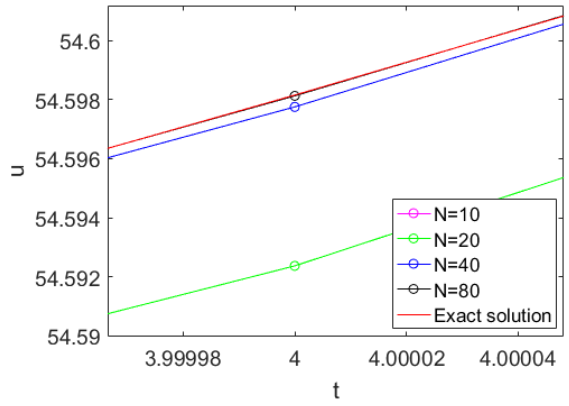

Figure 4.5: RK4 with N=10, 20, 40, and 80 zoomed in

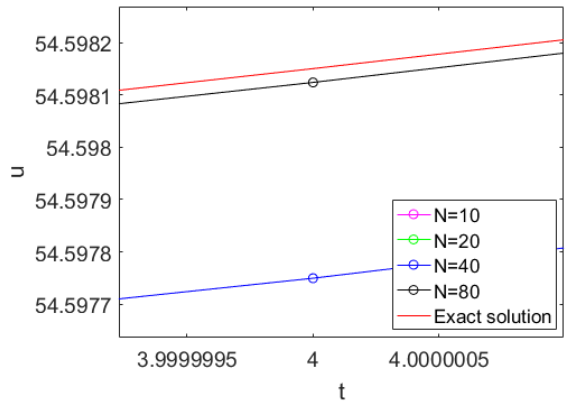

Figure 4.6: RK4 with N=10, 20, 40, and 80 zoomed in

order making it the most accurate.

4.2 Numerical Examples of Paraxial Wave Equations

In this section, we solve the paraxial wave equation. The paraxial wave equation is in the form:

$$u_z = \frac{i}{2k_0}u_{xx} + \frac{i\omega_0}{c}n_2|u|^2u,$$

where $k_0 = \frac{n_0\omega_0}{c}$, $0 \le z \le d$, $x_L \le x \le x_R$. Initial condition is a Gaussian pulse $u(0,x) = A_0e^{-\frac{(x-x_0)^2}{2b^2}}$. Boundary condition is Dirichlet boundary condition $u(z,x_L) = u(z,x_R) = 0$. Other parameters are $c = 299792458$ is the speed of light in vacuum, $k_0 = 1$, $\omega_0 = 2.35 \times$

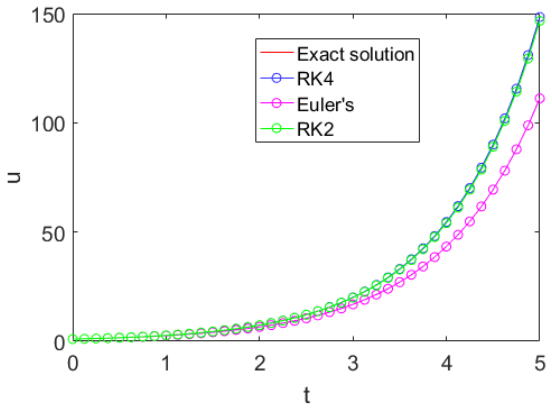

Figure 4.7: Comparison of all methods with N=40 with $0 \leq t \leq 5$

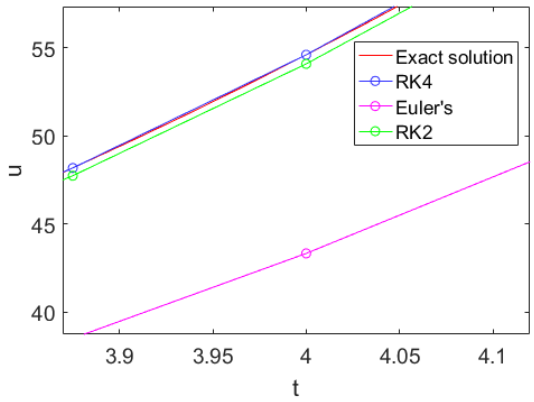

Figure 4.8: Comparison of all methods with N=40 with $3.9 \leq t \leq 4.1$

$10^{15},\ n_2 = 63.66,\ b = 8.5 \times 10^{-5},\ A_0 = 1, x_0 = 0,\ x_L = -10^{-3},\ x_R = 10^{-3},\ d = 10^{-7},\ \triangle x = (x_R - x_L)/N,\ N = 400,\quad \triangle z = \frac{\triangle x^2}{2}$.

Comparing this to the NLSE (3.1) in we can see that the "$V(\vec{x})$" equals zero ultimately ignoring the real valued external potential.

4.2.1 Solving a Paraxial Equation Using RK4 method

Since RK4 is one of the most accurate numerical method due to its fourth order in time. The result is shown in Figure 4.10 and Figure 4.11. We can clearly see the features of the higher accuracy when N is set to a higher value displayed in the Table 4.3. The accuracy for

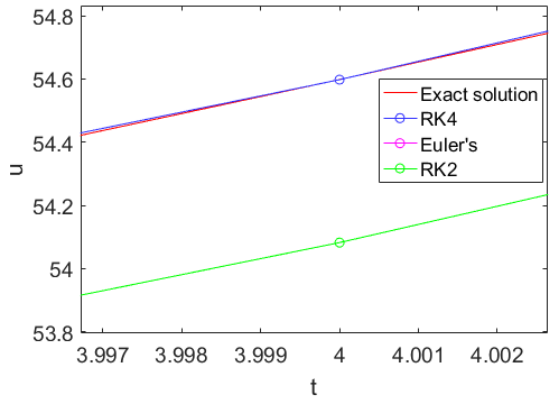

Figure 4.9: Comparison of all methods with N=40 with $3.997 \leq t \leq 4.002$

the first run of $N = 200$ we consider to be very accurate and we can see the RK4 method is coming in at 2nd order in space and we know from before RK4 is fourth order with respect to z. Looking at Figures 4.10, 4.11, and 4.12 we can see the differences of the curves in Figure 4.11 when zoomed in at the peak. Note the convergence rate from $N = 200$ to $N = 400$ to $N = 800$ and $N = 1600$.

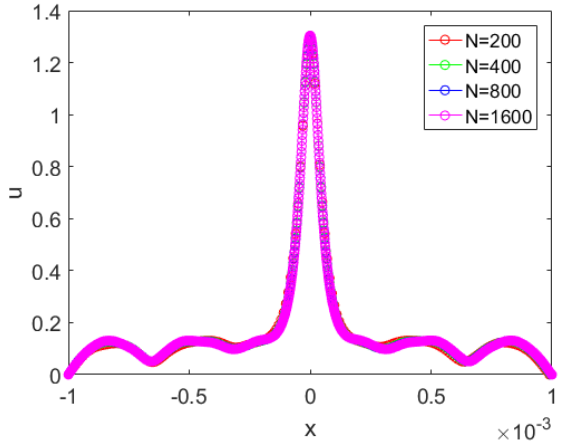

Figure 4.10: RK4 Method of Paraxial Equation

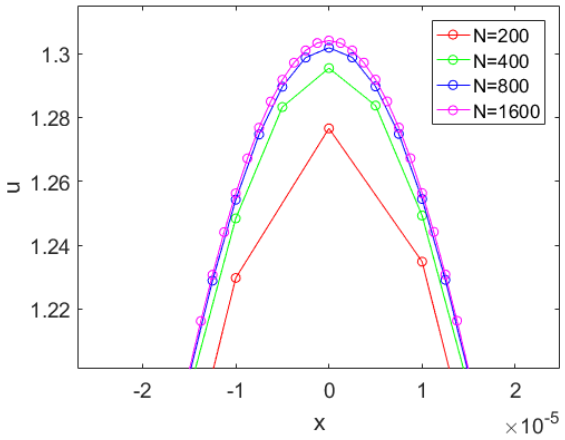

Figure 4.11: RK4 Method of Paraxial Equation zoomed in

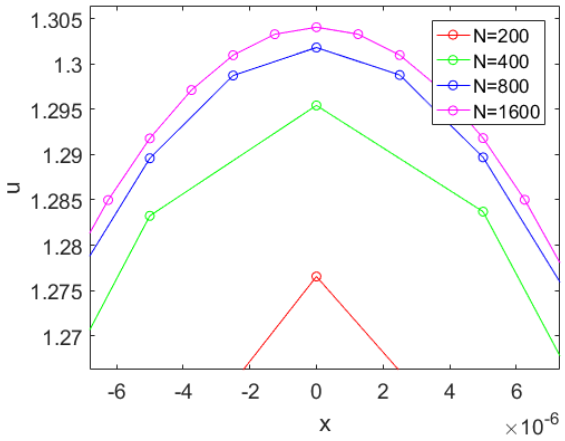

Figure 4.12: RK4 Method of Paraxial Equation zoomed in further

4.2.2 Solving a Paraxial Equation Using Crank-Nicholson Method

The next applied method is the Crank-Nicholson Method. This method is second order with respect to time and space so we expected the RK4 method to be more accurate. We can see from Figures 4.13 and 4.14 the results. For the Crank-Nicholson method we chose to use N values of 100, 200, 400, and 800. Table 4.4 shows the error and order of the Crank-Nicholson method. One thing that we found interesting is that with the RK4 method the lower N value solutions were below the exact solution and as N increase the curves built up to the exact solution. With the Crank-Nicholson method the exact opposite happens

with the smaller N values the curves are above the exact solution. It seems the RK4 method undershoots the exact solution and the Crank-Nicholson method overshoots the exact solution. In a future section when the two methods are graphed on the same graph we notice that the RK4 method is always below the Crank-Nicholson method.

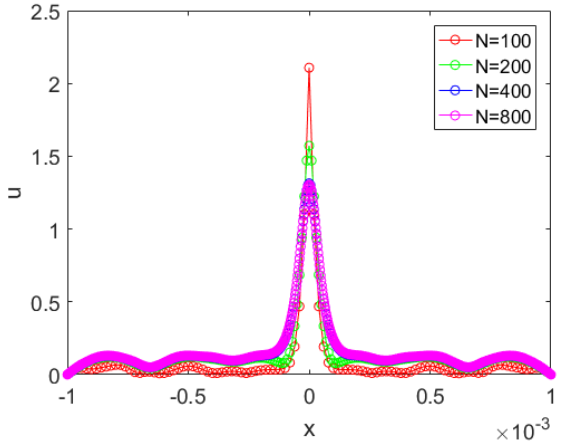

Figure 4.13: Crank-Nicholson Method of Paraxial Equation

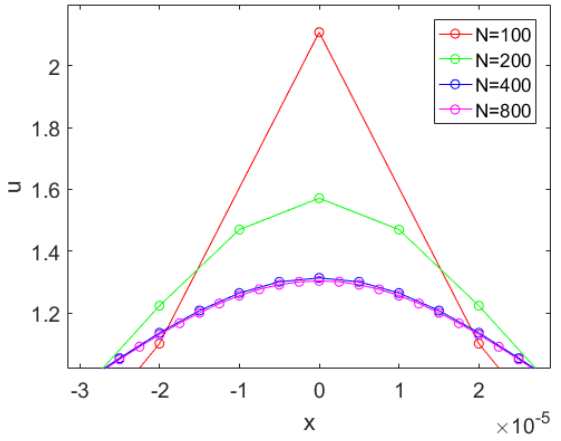

Figure 4.14: Crank-Nicholson Method of Paraxial Equation zoomed in

4.2.3 Solving a Paraxial Equation Using Leapfrog Crank-Nicholson Method

Next method is the Leapfrog Crank-Nicholson Method. Because it is an implicit method, so we choose Δz to be 10 times larger than the one used in RK4. As a result, the CPU

runtime is much less than the RK4 method. The results are shown in Figure 4.15 and 4.16. We used N values of 100, 200, 400, 800, and 1600. Table 4.5 shows the order which came in at the highest of all the methods we tried at more than second order.

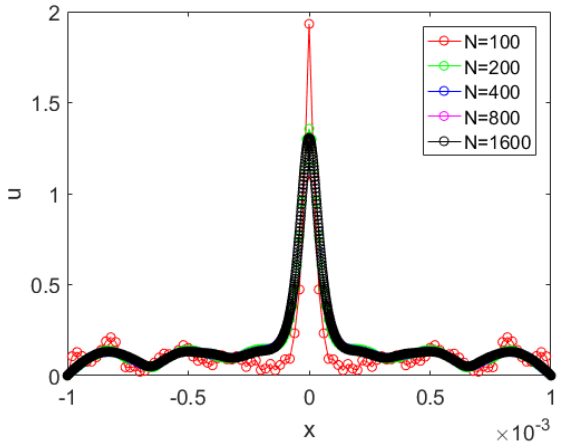

Figure 4.15: Leapfrog Crank-NicholsonMethod of Paraxial Equation

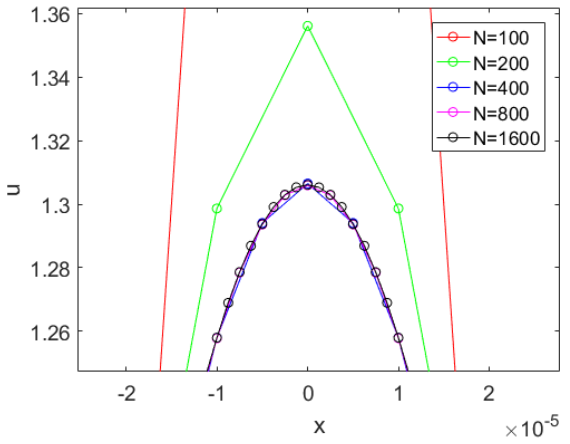

Figure 4.16: Leapfrog Crank-NicholsonMethod of Paraxial Equation zoomed in

4.2.4 Solving a Paraxial Equation Using Splitting Spectral Method

Lastly we used the Splitting Spectral Method. We chose to use N values of 50, 100, 200, and 400. The results are shown in Figures 4.17 and 4.18. The error and order is recored in Table 4.6.

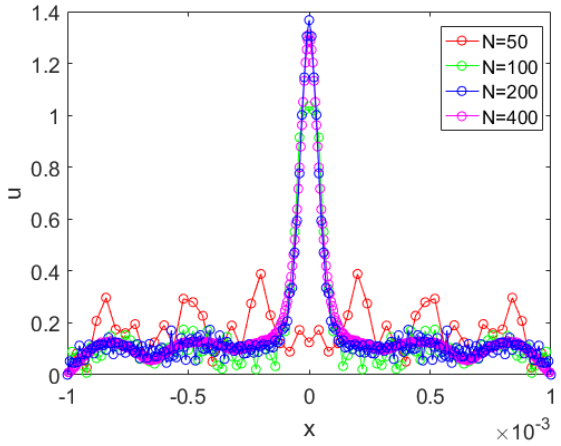

Figure 4.17: Time Splitting Spectral Method of Paraxial Equation

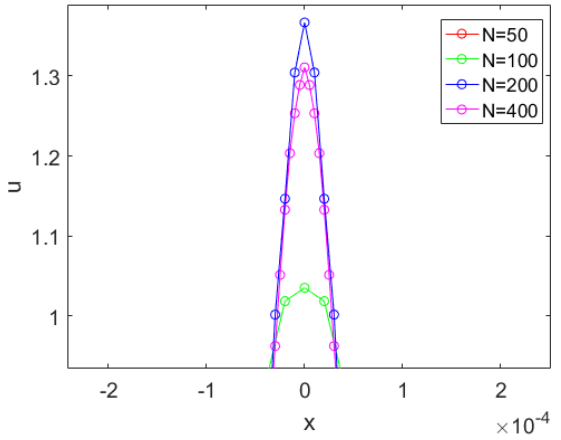

Figure 4.18: Time Splitting Spectral Method of Paraxial Equation zoomed in

4.2.5 Comparison of the Methods used to solve Paraxial Wave Equation

For this section we chose to record all the methods of the paraxial equation on the same graph. Figure 4.19 shows the results. Each curve has an N value of 200 to make it a fair comparison. At first glance we can see that the RK4 and the Leapfrog Crank-Nicholson methods are very close to each other and the Crank-Nicholson and the Time splitting method are very close. Our numerical results show that the Leapfrog Crank Nicolson (LF-CN) method is the most accurate method for paraxial wave equation. For $N = 800$, we see that the Error for LF-CN is 1.6×10^{-5}, smaller than all other methods, including the RK4

(5×10^{-5}), the CN (2×10^{-5}) , and the splitting method (7×10^{-5}). Table 4.7 show the time comparison of each method and how cost efficient the LFCN method is with slightly more error than RK4. For the LFCN and the splitting method we took the dz and made it ten times larger than RK4 and the CN methods. Making the dz ten times larger made the code run in a tenth of the time, while only giving up a small amount of error.

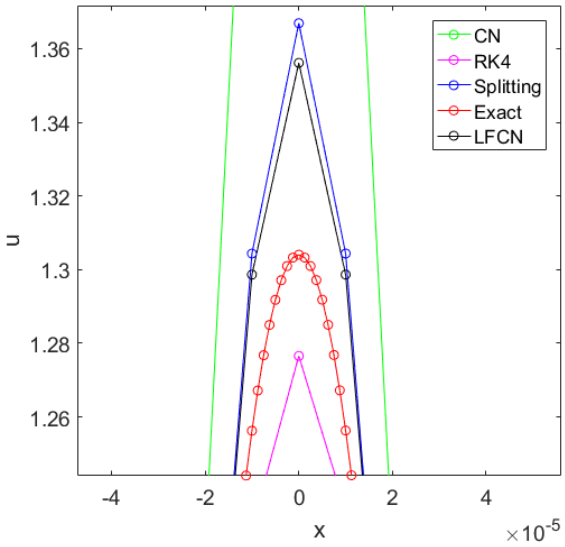

Figure 4.19: Comparison of the four Methods with N=200

Table 4.1: Errors and convergence rate of Euler's method for ODE

Nt	Error	Order
10	9.9978	-
20	4.4108	1.1806
40	1.7876	1.3030
80	0.6826	1.3889

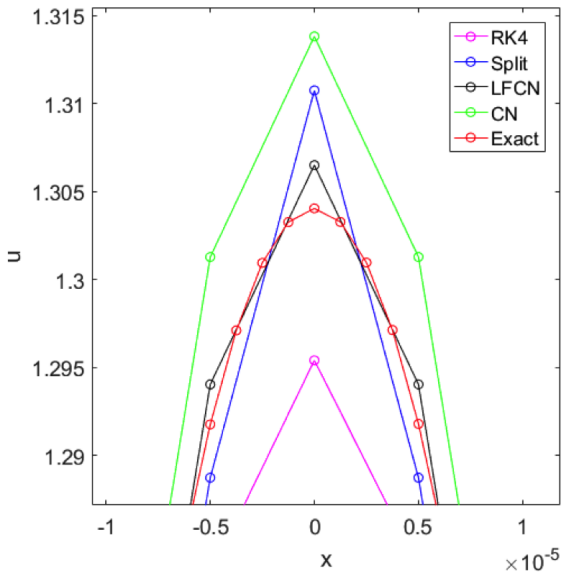

Figure 4.20: Comparison of the four Methods with N=400

Table 4.2: Errors and convergence rate of RK4 for ODE

Nt	Error	Order
10	0.0276	-
20	0.0014	4.3206
40	6.4732e-05	4.4170
80	2.9409e-06	4.4602

Table 4.3: Errors and convergence rate of RK4 for Paraxial Wave Equation

Nx	Error	Order
200	8.0056e-04	-
400	1.9442e-04	2.0418
800	5.0863e-05	1.9345
1600	1.1208e-05	2.1820

Table 4.4: Errors and convergence rate for Crank-Nicholson Method for Paraxial Wave Equation

Nx	Error	Order
100	0.0135	-
200	.0042	1.7010
400	1.4695e-04	4.8230
800	2.0097e-05	2.8702

Table 4.5: Errors and convergence rate for Leapfrog Crank-Nicholson Method for Paraxial Wave Equation

Nx	Error	Order
100	0.0108	-
200	8.5139e-04	3.6651
400	9.3498e-05	3.1868
800	1.6637e-05	2.4905

Table 4.6: Errors and convergence rate for Time Splitting Spectral Method for Paraxial Wave Equation

Nx	Error	Order
50	0.0328	-
100	0.0057	2.5275
200	0.0023	1.3091
400	2.0264e-04	3.5006

Table 4.7: Comparison of time and error of each method

Method	Time (sec)	Error ($Nx = 200$)	Error ($Nx = 400$)
RK4	.57	8.01e-4	1.94e-4
CN	1.26	4.20e-3	1.46e-4
Splitting	.11	2.30e-3	2.02e-4
LFCN	.15	8.87e-4	9.35e-5

Chapter V: CONCLUSION

In this thesis, we applied various numerical methods for solving ODE initial value problem and the paraxial wave equation with cubic nonlinearity. The numerical method we applied to solving paraxial wave equation are the 4th order Runge Kutta method, the Crank-Nicolson method, the Leapfrog Crank-Nicolson method, and the splitting spectrum method. The advantage of the explicit RK4 method is the high order accuracy in time. Our numerical results show that the leapfrog Crank-Nicolson method is the most cost efficient method as (1) it has smaller than when comparing with all other methods with same spatial resolution; (2) the step size can be chosen much larger than the explicit methods since it is implicit so it saves CPU time. The paraxial wave equation is derived from Maxwell's equation and we focus on the case where the cubic Kerr nonlinearity presents, which is applied to study optical pulse propagation in nonlinear Kerr media.

Further work will be on the extension of current work to NLSE including the potential term and more complex media. Another future work will be the comparison with the numerical solution to Maxwell's equation using Finite-Differene Time-Domain (FDTD) method.

REFERENCE LIST

[1] G. Agrawal. Nonlinear fiber optics: its history and recent progress. *J. Opt. Soc. AM. B*, 28:1—7, 2011.

[2] W. Bao. Numerical methods for the dynamics of the nonlinear schrodinger/gross-pitaevskii equations. *American Institute of Mathematical Sciences*, pages 1–39, 2013.

[3] Robert W Boyd. *Nonlinear optics*. Elsevier, 2003.

[4] Brown. *Euler's Method, Taylor Series Method, Runge Kutta Methods, Multi-Step Methods and Stability*. Brown Lectures in Mathematics. University of Brown, 2000.

[5] Richard L. Burden and J. Douglas Faires. *Numerical Analysis*. Cengage Learning, 9 edition, 2011.

[6] Gadi Fibich. *The nonlinear Schrödinger equation*. Springer, 2015.

[7] B. Fidel, E. Heyman, R. Kastner, and R.W. Ziolkowski. Hybrid ray-FDTD moving frame approach to pulse propagation. *J. Comput. Phys.*, 138:480–500, 1997.

[8] C. V. Hile and W. L. Kath. Numerical solutions of Maxwell's equations for nonlinear optical pulse propagation. *J. Opt. Soc. Am. B*, 13(6):1135–1146, 1996.

[9] Jacek Kasprzak, M Richard, S Kundermann, A Baas, P Jeambrun, JMJ Keeling, FM Marchetti, MH Szymańska, R Andre, JL Staehli, et al. Bose–einstein condensation of exciton polaritons. *Nature*, 443(7110):409, 2006.

[10] M. Kolesik, J. V. Moloney, and M. Mlejnek. Unidirectional optical pulse propagation equation. *Phys. Rev. Lett.*, 89:283902, 2002.

[11] Y. Pemper, V. Lomakin, E. Heyman, R. Kastner, and R. W. Ziolkowski. Moving coordinate frame FDTD analysis of long range tracking of pulsed fields in graded index waveguides. *J. Electromagn. Waves and Appl.*, 14:493–496, 2000.

[12] J. E. Rothenberg. Space-time focusing: breakdown of the slowly varying envelope approximation in the sel-focusing of femtosecond pulses. *Opt. Lett.*, 17(19):1340–1342, 1992.

[13] Mads Peter Sørensen, Moysey Brio, Garry M Webb, and Jerome V Moloney. Solitary waves, steepening and initial collapse in the maxwell–lorentz system. *Physica D: Nonlinear Phenomena*, 170(3-4):287–303, 2002.

[14] A. Taflove and S. Hagness. *Computational Electrodynamics: The Finite-Difference Time-Domain Method*. Artech House, Norwood, MA, 3rd edition, 2005.

[15] M. Trippenbach and Y. B. Band. Effects of self-steepening and self-frequency shifting on short-pulse splitting in dispersive nonlinear media. *Phys. Rev. A*, 57(6):4791–4803, 1998.

[16] K. S. Yee. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. *IEEE Trans. Antennas Propag.*, 14:302–307, 1966.

[17] Vladimir E Zakharov. Stability of periodic waves of finite amplitude on the surface of a deep fluid. *Journal of Applied Mechanics and Technical Physics*, 9(2):190–194, 1968.