NUMERICAL METHOD FOR TWO DIMENSIONAL NONLINEAR SCHRÖDIGER EQUATION

by

WEI YU

(Under the Direction of Thiab R. Taha)

ABSTRACT

The nonlinear Schrödinger equation is of tremendous importance in both theory and applications. The NLS type equation is the main governing equation in the area of optical solitons. Various regimes of pulse propagation in optical fibers are modeled by some form of the nonlinear Schrödinger equation.

In this thesis, we introduce sequential and parallel numerical methods for numerical simulations of two dimensional nonlinear Schrödinger equations. We implement the parallel methods on the pcluster multiprocessor system at UGA. The numerical results have shown that these methods give good results and considerable speedup.

INDEX WORDS: NLS, Split-step method, pseudo-spectral method, Finite difference method, Parallel algorithms, FFTW.

NUMERICAL METHOD FOR TWO DIMENSIONAL NONLINEAR SCHRÖDIGER EQUATION

by

WEI YU

B.E., Beijing University of Posts & Telecommunications, China, 2008

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2010

NUMERICAL METHOD FOR TWO DIMENSIONAL NONLINEAR SCHRÖDIGER EQUATION

by

WEI YU

Major Professor: Thiab R. Taha

Committee: Hamid R. Arabnia
Daniel M. Everett

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
December 2010

ACKNOWLEDGEMENTS

I would like to thank my major professor, Dr. Thiab R. Taha, for his efforts, guidance and support, which made my research better.

I am so deeply grateful to the members of my advisory committee, Dr. Hamid R. Arabnia and Dr. Daniel M. Everett, for their kind and valuable help.

I also want to thank all of the faculty, staff and my friends in the Department of Computer Science for discussing with them and learning a lot from them, especially Meng, talking with him helps me greatly.

Finally, I would like to thank my fiancée, Xiaobo Lu, for her continuous support and trust.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

The nonlinear Schrödinger equation (NLS) is being used to describe a wide class of physical phenomena (e.g., heat pulses in solids, modulation of deep water waves, and helical motion of a very thin vortex filament [1]).

The research on optical solitons has been going on for many decades and the two dimensional nonlinear Schrödinger (NLS) equation is one of the main governing equations in this area of study [9] [16] [17].

In this thesis, we will study the two dimensional NLS equation which is given by [9]

$$iq_t + \frac{1}{2}\left(q_{xx} + q_{yy}\right) + |q|^2 q = 0, \tag{1.1}$$

where q is a two dimensional complex-valued function. There are many popular numerical methods to solve the NLS equation. One of them is the split-step Fourier (SSF) method, proposed by R. H. Hardin and F. D. Tappert [5]. It is one of the most popular numerical methods for solving the NLS equation [1]. Various versions of the split-step method have been developed to solve the NLS equation [3].

Taha and Xu have developed split-step method for the NLS and CNLS equations as well as parallel implementations for these methods [3].

S. Zoldi et al [13] has implemented a parallel split-step Fourier method for large-scale simulations of the NLS equation, which are required for many physical problems. M. S. Ismail

and T. R. Taha have introduced a finite difference method to numerically simulate the CNLSE [12].

In this thesis, we implement three schemes of the two dimensional split-step Fourier method and two schemes of the pseudo-spectral method for the numerical simulation of the two dimensional NLS equation. We also implemented the parallel split-step Fourier method and pseudo-spectral method with the *Fastest Fourier Transform in the West* (FFTW), which is developed by M. Frigo and S. G. Johnson [14]. Moreover, we have implemented two schemes of the two dimensional finite difference methods using explicit and implicit methods.

The first chapter of this thesis is an introduction of the NLS equation. The second chapter describes the Fourier transform, the split-step method, the pseudo-spectral method and the finite difference methods. Chapter 3 presents the numerical methods to solve the two dimensional NLS equation and experiments results. The fourth chapter shows the rank for all the methods that have been utilized to solve the two dimensional NLS equation. Chapter 5 provides a summary of my current research work and gives the conclusion.

CHAPTER 2

PRELIMINARIES

2.1 THE DISCRETE TWO DIMENSIONAL FOURIER TRANSFORM

If $\{f[m, n]\}$ is a sequence of size $M \times N$, obtained by taking samples of a continuous function f with equal intervals at the direction of m and n, respectively, then its discrete Fourier transform (DFT) is given by

$$F[k,l] = \frac{1}{\sqrt{MN}} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[m,n] e^{-i2p\left(\frac{mk}{M}+\frac{nl}{N}\right)}, 0 \le k < M, 0 \le l < N \qquad (2.1.1)$$

where M and N are the numbers of samples in x and y directions in both spatial and frequency domains, respectively. And $F[k,l]$ is the two dimensional discrete spectrum of $f[m,n]$.

The inverse two dimensional DFT flips the sign of the exponent, which is defined as

$$f[m,n] = \frac{1}{\sqrt{MN}} \sum_{l=0}^{N-1} \sum_{k=0}^{M-1} F[k,l] e^{j2p\left(\frac{mk}{M}+\frac{nl}{N}\right)}, 0 \le m < M, 0 \le n < N \qquad (2.1.2)$$

It is the "inverse" of the forward two dimensional DFT, in the sense that computing the inverse transfer after the forward transform of a given sequence would yield the original sequence.

Both $F[k,l]$ and $f[m,n]$ could be considered as elements of two $M \times N$ matrices x and F, respectively.

2.2 THE TWO DIMENSIONAL FOURIER TRANSFORM

The Fast Fourier transform (FFT) is an efficient algorithm to compute the discrete Fourier transform (DFT) and its inverse. A DFT decomposes a sequence of values into components of different frequencies. It is useful and being used in many fields. However, calculating DFT directly from its definition is often too slow. Instead, FFT is a better way to calculate the same result more quickly.

Computing a DFT of N points requires $O(N^2)$. However, an FFT can calculate the same result in only $O(N\log_2 N)$ operations [19]. The difference in speed could be substantial, especially for large data sets where N may be very huge. In this case, FFTs are of great importantance to a large number of applications, like, digital signal processing and solving partial differential equations.

A two dimensional FFT is achieved by first transforming each row, replacing each row with its one dimensional transform FFT and then transforming each column, replacing each column with its transform. A two dimensional FFT of size $M \times N$ requires $M + N$ one dimensional FFT.

2.3 THE FASTEST FOURIER TRANSFORM IN THE WEST

The Fastest Fourier Transform in the West (FFTW) is a software library used to calculating DFTs, developed by M. Frigo and S. G. Johnson in MIT. FFTW is a comprehensive collection of fast C routines for calculating the DFT in one or more dimensions, of both real and complex data, and of arbitrary input size [3]. “It has gained a wide acceptance in both academia and industry, because it provides excellent performance on a variety of machines (even competitive with or faster than equivalent libraries supplied by vendors)” [20].

FFTW automatically adapts the DFT algorithm to details of the underlying hardware (cache size, memory size, registers, etc.). The inner loop of FFTW is generated automatically by a special-purpose compiler. The FFTW begins by generating codelets. A codelet is a fragment of C code that computes a Fourier transform of a fixed small size (e.g. 16 or 19). A composition of codelets is called a plan which depends on the size of the input and the underline hardware. At runtime, the FFTW's planner finds the optimal decomposition for transforms of a specified size on your machine and produces a plan that contains this information. The resulting plan can be reused as many times as needed. This makes the FFTW's relatively expensive initialization acceptable. FFTW also includes a shared-memory implementation on top of POSIX threads, and a distributed-memory implementation based on MPI (Message Passing Interface).

2.4 THE SPLIT-STEP METHOD

The split-step (Fourier) method is a pseudo-spectral numerical method used to solve nonlinear partial differential equations [21].

For example, consider the following equation

$$\begin{aligned} q_t &= (L+N)q, \\ q(x,y,0) &= q_0(x,y), \end{aligned} \tag{2.4.1}$$

where L and N are linear and nonlinear operators, respectively. In general, the operators L and N do not commute with each other.

The two dimensional NLS equation

$$q_t = iq_{xx} + iq_{yy} + ik|q|^2 q,$$

with k a real number, can be rewritten as

$$q_t = Lq + Nq,$$

where

$$Lq = iq_{xx} + iq_{yy}, Nq = ik|q|^2 q.$$

The solution of equation (2.4.1) could be advanced from one time-level to the next by using the following formula [3]

$$q(x, y, t + \Delta t) = \exp[\Delta t(L + N)]q(x, y, t), \tag{2.4.2}$$

where Δt denotes the time step. It is first order accurate. However, it would be exact if operators L and N are time-independent [8].

Now the time-splitting procedure includes replacing the right-hand side of (2.4.2) by an appropriate combination of products of the exponential operator exp (ΔtL) and exp (ΔtN). We can find one answer by using the Baker-Campbell-Hausdorf (BCH) formula [22] for two operators A and B as following

$$\exp(lA)\exp(lB) = \exp\left(\sum_{n=1}^{\infty} l^n Z_n\right), \tag{2.4.3}$$

Where l is the coefficient of A and B, and

$$Z_1 = A + B$$

And the remaining operators Z_n are commutators of A and B, commutators of commutators of A and B, etc. The expressions for Z_n are actually rather complicated, e.g.

$$Z_2 = \frac{1}{2}[A, B],$$

Where $[A, B] = AB - BA$ is the commutator of A and B, and

$$Z_3 = \frac{1}{12}([A,[A, B]] + [[A, B], B]).$$

From this result, we can easily get the first-order approximation of the exponential operator in (2.4.2) as follows [3]

$$A_1(\Delta t) = \exp(\Delta tL)\exp(\Delta tN). \tag{2.4.4}$$

Note that this expression is exact whenever L and N commute.

It would be more convenient to view the scheme (2.4.4) as first solving the nonlinear part

$$q_t = Nq,$$

then advancing the solution by solving the linear part

$$q_t = Lq,$$

by employing the solution of the former as the initial condition of the latter. That is, the advancement in time is carried out in two steps, the so called split-step method.

The second-order approximation of the exponential operator in (2.4.2) is given by

$$A_2(\Delta t) = \exp\left(\frac{1}{2}\Delta tN\right)\exp(\Delta tL)\exp\left(\frac{1}{2}\Delta tN\right). \tag{2.4.5}$$

The fourth-order approximation of the exponential operator in (2.4.2) which preserves the symmetry could also be constructed [3] [23], e.g.

$$A_4(\Delta t) = A_2(w\Delta t)A_2[(1-2w)\Delta t]A_2(w\Delta t), \tag{2.4.6}$$

where

$$w = \frac{2 + \sqrt[3]{2} + \frac{1}{\sqrt[3]{2}}}{3}. \tag{2.4.7}$$

Note that the operators L and N in (2.4.4) – (2.4.6) may be interchanged without affecting the order of the method [24].

2.5 THE PSEUDO-SPECTRAL METHOD

Pseudo-spectral methods are a class of numerical methods which are used in applied mathematics and scientific computing for the solution of partial differential equations [25].

Given $q(x, y, t)$, we want to find $q(x, y, t+\Delta t)$ with a small Δt.

The first step we should take is to compute an intermediate value $u_1(x, y)$ by applying the rightmost operator in the symmetric decomposition

$$u_1(x, y) = e^{-iV(r)\Delta t/2h} q(x, y, t), \tag{2.5.1}$$

where h is the reduced Planck constant $V(r)$ depends only on position r.

To solve (2.5.1), it requires only a point wise multiplication.

The next step is to apply (2.5.1)

$$u_2(x, y) = e^{-iT\Delta t/h} u_1(x, y),$$

where T is the kinetic energy.

To simplify the above calculation, we can take the following equation to compute $u_2(x, y)$

$$\Phi_2(k) = e^{ihk^2\Delta t/2m} \Phi_1(k),$$

which also requires only a point wise multiplication. $\Phi_1(k)$ could be obtained from $u_1(x, y)$ by using the Fast Fourier transform (FFT). $u_2(x, y)$ could be obtained from $\Phi_2(k)$ by using the inverse FFT.

The final computation is

$$q(x, y, t + \Delta t) = e^{-iV(r)\Delta t/2h} u_2(x, y)$$

We can summarize the above sequence as

$$q(x, y, t + \Delta t) = e^{-iV(r)\Delta t/2h} F^{-1}\left[e^{ihk^2\Delta t/2m} F\left[e^{-iV(r)\Delta t/2h} q(x, y, t)\right]\right]$$

2.6 THE FINITE DIFFERENCE METHOD

Finite difference methods are numerical methods for approximating the solutions to differential equations using finite difference equations to approximate derivatives [28].

2.6.1 EXPLICIT METHOD

In explicit method, we calculate the state of a system at a later time from the state of the system at the current time.

Given $Y(t)$ as the current system state and $Y(t + \Delta t)$ as the state of the system as a later time, then we have

$$Y(t+\Delta t) = F(Y(t))$$

Using the classical explicit method with central difference in time, the finite difference representation of (1.1) would be

$$i(\frac{q_{k,j}^{n+1} - q_{k,j}^{n}}{\Delta t}) + \frac{1}{2}(\frac{q_{k+1,j}^{n} - 2q_{k,j}^{n} + q_{k-1,j}^{n}}{(\Delta x)^2} + \frac{q_{k,j+1}^{n} - 2q_{k,j}^{n} + q_{k,j-1}^{n}}{(\Delta y)^2}) + \left|q_{k,j}^{n}\right|^2 q_{k,j}^{n} = 0 \qquad (2.6.2)$$

where $0 \le k < N_x$ and $0 \le j < N_y$.

To compute $q(x, y, t)$, first we use $q(x, y, 0)$ as the initial condition and use it into (2.6.2) to get $q(x, y, \Delta t)$.

Then, for $2\Delta t \le t \le T$, we use the $q(x, y, t - \Delta t)$ which we get in the last computation to obtain $q(x, y, t)$.

2.6.2 IMPLICIT METHOD

In implicit method, we find the solution by solving an equation involving both the current state and the later one of the system.

Given $Y(t)$ as the current system state and $Y(t + \Delta t)$ as the state of the system at a later time, we solve the following equation to obtain $Y(t + \Delta t)$

$$G(Y(t), Y(t+\Delta t)) = 0$$

Using the alternating direction implicit method with central difference in time, the finite difference representation of (1.1) would be

$$i\left(\frac{q_{k,j}^{n+1/2}-q_{k,j}^{n}}{\Delta t/2}\right)+\frac{1}{2}\left(\frac{q_{k+1,j}^{n+1/2}-2q_{k,j}^{n+1/2}+q_{k-1,j}^{n+1/2}}{(\Delta x)^2}+\frac{q_{k,j+1}^{n}-2q_{k,j}^{n}+q_{k,j-1}^{n}}{(\Delta y)^2}\right)+\left|q_{k,j}^{n}\right|^2 q_{k,j}^{n}=0 \quad (2.6.3)$$

and

$$i\left(\frac{q_{k,j}^{n+1}-q_{k,j}^{n+1/2}}{\Delta t/2}\right)+\frac{1}{2}\left(\frac{q_{k+1,j}^{n+1/2}-2q_{k,j}^{n+1/2}+q_{k-1,j}^{n+1/2}}{(\Delta x)^2}+\frac{q_{k,j+1}^{n+1}-2q_{k,j}^{n+1}+q_{k,j-1}^{n+1}}{(\Delta y)^2}\right)+\left|q_{k,j}^{n+1/2}\right|^2 q_{k,j}^{n+1/2}=0 \quad (2.6.4)$$

where $0 \le k < N_x$ and $0 \le j < N_y$.

To compute $q(x, y, t)$, first we use $q(x, y, 0)$ as the initial condition and use it into (2.6.3) to get $q(x, y, \Delta t/2)$.

Second, we use the $q(x, y, \Delta t/2)$ which we get in the last computation into (2.6.4) to obtain $q(x, y, \Delta t)$.

Then, for $2\Delta t \le t \le T$, we repeat last two operations to obtain $q(x, y, t)$.

2.7 MESSAGE PASSING INTERFACE

Message Passing Interface (MPI) is a library specification for message-passing, proposed as a standard by a broadly based committee of vendors, implementations, and users.

Message passing is a paradigm that has been widely used on certain classes of parallel machines, especially those with distributed memory. Processes running on such machines communicate through messages [14].

CHAPTER 3

TWO DIMENSIONAL NONLINEAR SCHRÖDINGER EQUATION

The original two dimensional nonlinear Schrödinger equation is as follows

$$iq_t + \frac{1}{2}\left(q_{xx} + q_{yy}\right) + |q|^2 q = 0, \tag{3.0.1}$$

where q is a complex-valued function. The exact one-soliton solution of (3.0.1) is given by [7]

$$q(x, y, t) = \frac{A}{\cosh^p\left[\left(B_1 x + B_2 y - ut\right)\right]} ei\left(-k_1 x - k_2 y + wt + q\right), \tag{3.0.2}$$

In (3.0.2), A is the amplitude of the soliton, B_1 is the inverse width in the x-direction and B_2 is the inverse width in the y-direction. u represents the velocity of the soliton. k_1 and k_2 represents the soliton frequency in the x and y directions respectively, while w represents the solitary wave number and finally q is the phase constant of the soliton. The exponent p, which is unknown at this point, would be determined when finding the exact soliton solution [7].

By (3.0.2), we have the following pair of relations

$$u = -\left(k_1 B_1 + k_2 B_2\right)$$

$$p = 1$$

$$A = \sqrt{B_1^2 + B_2^2}$$

$$w = \frac{1}{2}\left[\left(B_1^2 + B_2^2\right) - \left(k_1^{\,2} + k_2^{\,2}\right)\right]$$

3.1 NUMERICAL METHOD

First, we study the two dimensional NLS equation (3.0.1) with the initial condition given by

$$q(x, y,0) = \frac{A}{\cosh[(B_1 x + B_2 y - ut)]} ei(-k_1 x - k_2 y + q), \tag{3.1.1}$$

where $t = 0.0$, $p = 1$, $B_1 = 1.0$, $B_2 = 0.1$, $k_1 = 0.6$, $k_2 = 0.8$. Here, we also assume that $q(x, y, t)$ satisfies periodic boundary condition with period $[-P, P]$.

After normalizing the spatial period to $[0, 2\pi]$, we have

$$iq_t = -\frac{p^2}{2P^2}(q_{xx} + q_{yy}) - |q|^2 q, \tag{3.1.2}$$

where $P = 10$, which is the half length of the period. $X = \pi(x + P) / P$ and $Y = \pi(y + P) / P$. Then, we divide the interval $[0, 2\pi]$ in the x-direction into N_x equal subintervals with grid spacing $\Delta X = 2p / N_x$, and denote $X_j = j\Delta X$, $j = 0,1,\cdots, N_x$ as the spatial grid points. We also divide the interval $[0, 2\pi]$ in the y-direction into N_y equal subintervals with grid spacing $\Delta Y = 2p / N_y$, and denote $Y_k = k\Delta Y$, $k = 0,1,\cdots, N_y$ as the spatial grid points.

Now we advance the solution of (3.0.1) from time t to the next time-level $t + \Delta t$ as follows:

First, we only focus on the nonlinear part to advance the solution [1]:

$$iq_t = -|q|^2 q, \tag{3.1.3}$$

which could be solved exactly with

$$\hat{q}(X_j, Y_k, t + \Delta t) = \exp\{\, i|q(X_j, Y_k, t)|^2 \Delta t \,\} q(X_j, Y_k, t). \tag{3.1.4}$$

Second, we focus on the linear part:

$$iq_t = -\frac{p^2}{2P^2}\left(q_{xx} + q_{yy}\right), \tag{3.1.5}$$

For the two dimensional discrete Fourier transform, we have

$$\hat{q}_{mn} = F_{m,n}\left(q_{jk}\right) = \frac{1}{N_x N_y}\sum_{j=0}^{N_x-1}\sum_{k=0}^{N_y-1} q_{jk}\exp\left[-i\left(mX_j + nY_k\right)\right], \tag{3.1.6}$$

$$-\frac{N_x}{2} \le m \le \frac{N_x}{2}-1, -\frac{N_y}{2} \le n \le \frac{N_y}{2}-1$$

and

$$q_{jk} = F^{-1}_{j,k}\left(\hat{q}_{mn}\right) = \sum_{m=-\frac{N_x}{2}}^{\frac{N_x}{2}-1}\sum_{n=-\frac{N_y}{2}}^{\frac{N_y}{2}-1}\hat{q}_{mn}\exp\left[i\left(mX_j + nY_k\right)\right], \tag{3.1.7}$$

$$j = 0,1,2,\cdots N_x - 1, k = 0,1,2,\cdots, N_y - 1$$

According to (3.1.6) and (3.1.7), we have

$$q_t = \frac{dq_{jk}}{dt} = \sum_{m=-\frac{N_x}{2}}^{\frac{N_x}{2}-1}\sum_{n=-\frac{N_y}{2}}^{\frac{N_y}{2}-1}\frac{d\hat{q}_{mn}}{dt}\exp\left[i\left(mX_j + nY_k\right)\right], \tag{3.1.8}$$

$$q_{xx} = \frac{d^2 q_{jk}}{dx^2} = \sum_{m=-\frac{N_x}{2}}^{\frac{N_x}{2}-1}\sum_{n=-\frac{N_y}{2}}^{\frac{N_y}{2}-1}\hat{q}_{mn}.\left(-m^2\right)\exp\left[i\left(mX_j + nY_k\right)\right], \tag{3.1.9}$$

$$q_{yy} = \frac{d^2 q_{jk}}{dy^2} = \sum_{m=-\frac{N_x}{2}}^{\frac{N_x}{2}-1}\sum_{n=-\frac{N_y}{2}}^{\frac{N_y}{2}-1}\hat{q}_{mn}.\left(-n^2\right)\exp\left[i\left(mX_j + nY_k\right)\right], \tag{3.1.10}$$

Substituting (3.1.8) – (3.1.10) into (3.1.5), and equating every pair of items yields the following result

$$i\frac{d\hat{q}_{mn}}{dt} = -\frac{p^2}{2P^2}\left(-m^2 - n^2\right)\hat{q}_{mn}, \tag{3.1.11}$$

Solving (3.1.11), we have

$$\hat{q}_{mn}(t+\Delta t)=\hat{q}_{mn}(t)\exp\left[-i\frac{p^2}{2p^2}\left(m^2+n^2\right)\Delta t\right], \tag{3.1.12}$$

Then applying (3.1.12) into the following

$$q\left(X_j,Y_k,t+\Delta t\right)=F^{-1}\left(F\left(\hat{q}\left(X_m,Y_n,t+\Delta t\right)\right)\right)$$

We have:

$$q\left(X_j,Y_k,t+\Delta t\right)=F^{-1}\left(\exp\left[-i\frac{p^2}{2p^2}\left(m^2+n^2\right)\Delta t\right]F\left(\hat{q}\left(X_m,Y_n,t\right)\right)\right), \tag{3.1.13}$$

Thus, (3.1.13) is the split-step Fourier method for the first-order splitting approximation (2.4.4), where Δt is the time step, F and F^{-1} are the forward and inverse discrete Fourier transforms respectively.

To advance in time from t to $t+\Delta t$ by the split-step Fourier method with the second-order splitting approximation (2.4.5), we should take the following steps [1]:

(1) Applying (3.1.4) to advance the solution using the nonlinear part

$$\hat{q}(X_m,Y_n,t+\frac{1}{2}\Delta t)=\exp\{\ i\left|q\left(X_m,Y_n,t\right)\right|^2\frac{1}{2}\Delta t\ \}q\left(X_m,Y_n,t\right).$$

(2) Applying (3.1.13) to advance the solution using the linear part

$$\tilde{q}\left(X_j,Y_k,t+\frac{1}{2}\Delta t\right)=F^{-1}\left(\exp\left[-i\frac{p^2}{2p^2}\left(m^2+n^2\right)\Delta t\right]F\left(\hat{q}\left(X_m,Y_n,t+\frac{1}{2}\Delta t\right)\right)\right).$$

(3) Applying (3.1.4) to advance the solution using the nonlinear part

$$q(X_m,Y_n,t)=\exp\{\ i\left|\tilde{q}\left(X_m,Y_n,t+\frac{1}{2}\Delta t\right)\right|^2\frac{1}{2}\Delta t\ \}\tilde{q}\left(X_m,Y_n,t\right).$$

Advancement in time from t to $t+\Delta t$ by the split-step Fourier method with the fourth-order splitting approximation (2.4.6) could be obtained with following steps [3]:

First, advance in time from t to $t + w\Delta t$ using the second-order split-step Fourier method, where

$$w = \frac{2 + \sqrt[3]{2} + \frac{1}{\sqrt[3]{2}}}{3}.$$

Second, advance in time from $t + w\Delta t$ to $t + (1-w)\Delta t$ using the second-order split-step Fourier method.

Finally, advance in time from $t + (1-w)\Delta t$ to $t + \Delta t$ using the second-order split-step Fourier method and we obtain approximation to $q(x, y, t + \Delta t)$.

3.2 NUMERICAL EXPERIMENTS

To test the numerical method, we computed the L_∞ norm and L_2 norm at the terminating time $T = 1$ [1]. Also, we compute the relative error for the following conserved quantity

$$I = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} |q|^2 \, dxdy, \tag{3.2.1}$$

Equation (3.2.1) is calculated using the two dimensional Simpson's rule, which is described as follows.

For a two dimensional function $z = f(x, y), a \le x \le b$ and $c \le y \le d$, with the interval $2m$ and $2n$, respectively, we denote the equally spaced sample points as

$$x_i = x_0 + ih, \; i = 0,1,2,\cdots,2m$$

and

$$y_j = y_0 + jk, \; j = 0,1,2,\cdots,2n$$

where $h = \frac{b-a}{2m}$ and $k = \frac{d-c}{2n}$.

Then composite Simpson's rule would be as

$$\iint_R f(x,y)dA = \int_a^b \int_c^d f(x,y)dydx \approx S2D(f,h,k).$$

where

$$\begin{aligned} S2D(f,h,k) = \frac{1}{9}hk\{ & f(a,c) + f(a,d) + f(b,c) + f(b,d) \\ & + 4\sum_{j=1}^{n} f(a, y_{2j-1}) + 2\sum_{j=1}^{n-1} f(a, y_{2j}) + 4\sum_{j=1}^{n} f(b, y_{2j-1}) + 2\sum_{j=1}^{n-1} f(b, y_{2j}) \\ & + 4\sum_{i=1}^{m} f(x_{2i-1}, c) + 2\sum_{i=1}^{m-1} f(x_{2i}, c) + 4\sum_{i=1}^{m} f(x_{2i-1}, d) + 2\sum_{i=1}^{m-1} f(x_{2i}, d) \\ & + 16\sum_{j=1}^{n}\left(\sum_{i=1}^{m} f(x_{2i-1}, y_{2j-1})\right) + 8\sum_{j=1}^{n-1}\left(\sum_{i=1}^{m} f(x_{2i-1}, y_{2j})\right) \\ & + 8\sum_{j=1}^{n}\left(\sum_{i=1}^{m-1} f(x_{2i}, y_{2j-1})\right) + 4\sum_{j=1}^{n-1}\left(\sum_{i=1}^{m-1} f(x_{2i}, y_{2j})\right)\}. \end{aligned}$$

And it could be shown that the error term is of the form

$$E_{S2D}(f,h,k) = o(h^4) + o(k^4),$$

that is

$$\int_a^b \int_c^d f(x,y)dydx = S2D(f,h,k) + o(h^4) + o(k^4)$$

Simpson's rule has the pattern of weights:

1, 4, 2, 4, 2, 4, 2 … 2, 4, 1

And two dimensional Simpson's rule has extended this pattern to:

1	4	2	4	2	4	2	…	2	4	1
4	16	8	16	8	16	8	…	8	16	4
2	8	4	8	4	8	4	…	4	8	2

4	16	8	16	8	16	8	…	8	16	4
2	8	4	8	4	8	4	…	4	8	2
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
2	8	4	8	4	8	4	…	4	8	2
4	16	8	16	8	16	8	…	8	16	4
1	4	2	4	2	4	2	…	2	4	1

In our numerical experiments we used $N = 256$ for different values of time steps Δt to test the accuracy of the first-order split-step schemes that are utilized in solving (1.1). The results are shown in Table 3.1.

Table 3.1: Convergence rates in time for the first-order splitting method ($N = 256, -10 \le x \le 10\,,\ -10 \le y \le 10\,,\ 0 \le t \le 1,\ T = 1$).

Δt	L_∞	L_2	i_1	cpu(s)
0.004	7.103022E-01	2.245761E-01	5.658099E-03	10.43
0.002	7.103286E-01	2.245781E-01	5.649435E-03	18.96
0.001	7.103115E-01	2.245769E-01	5.649756E-03	36.53
0.0005	7.103034E-01	2.245758E-01	5.649831E-03	70.35
0.00025	7.102995E-01	2.245752E-01	5.649850E-03	137.05
0.000125	7.102975E-01	2.245748E-01	5.649856E-03	277.04

In the above and following tables, L_∞ is the infinity norm, L_2 is the Euclidian norm, i_1 is the relative error, and $i_1 = \left| \frac{C_1 - C}{C} \right|$. C is the conserved quantity for the exact solution at $t = 0$, C_1 is the conserved quantity for the numerical solution at every time step.

Also, in our numerical experiments we used $\Delta t = 0.000125$ for different values of N to test the accuracy of the first-order split-step schemes that are utilized in solving (1.1). The results are shown in Table 3.2.

Table 3.2: Convergence rates in space for the first-order splitting method ($\Delta t = 0.000125$, $-10 \le x \le 10$, $-10 \le y \le 10$, $0 \le t \le 1$, $T = 1$).

N	L_∞	L_2	i_1	cpu(s)
32	5.124649E-01	3.034540E-01	1.840878E-03	2.47
64	5.397828E-01	3.034989E-01	7.862426E-03	10.80
128	5.638354E-01	3.040974E-01	4.169053E-03	47.53
256	5.693562E-01	3.044173E-01	2.047879E-03	261.42
512	5.773713E-01	3.045096E-01	1.019764E-03	1664.98

The numerical solutions of the two dimensional NLS equation (3.0.1) at $t = 0.1$with the initial condition (3.1.1) using the above split-step Fourier methods with $\Delta t = 0.000125$ and $N = 256$ are shown below:

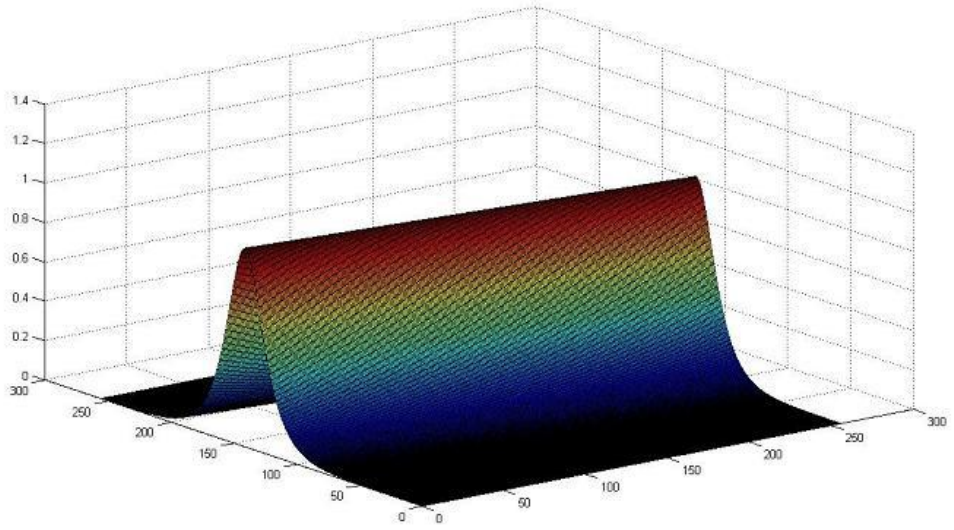

Figure 3.1: The initial conditions.

The modulus of the initial condition of equation (3.0.1) at $t = 0$.

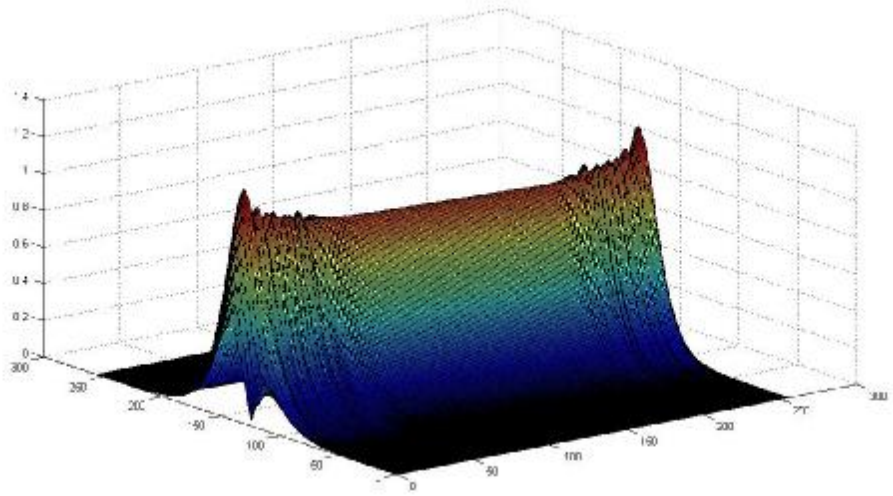

Figure 3.2: Two dimensional SSF1.

The modulus of equation (3.0.1) at $t = 0.1$ using the two dimensional first-order SSF.

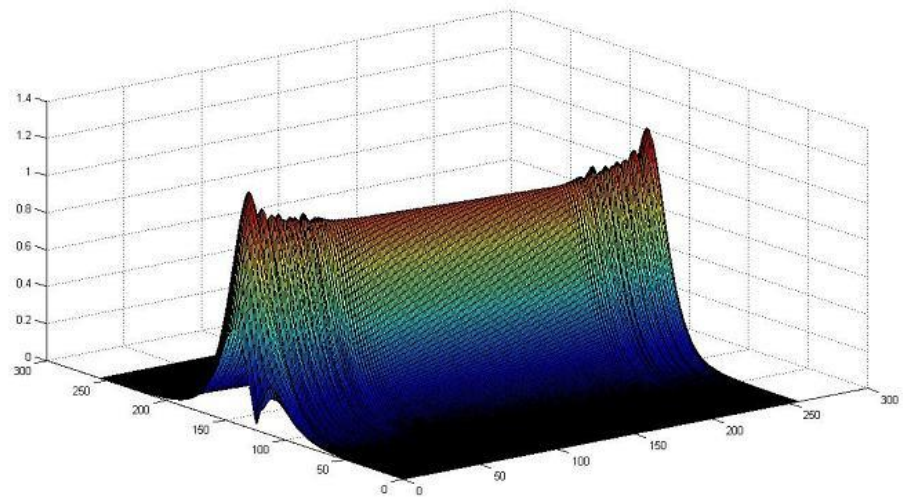

Figure 3.3: Two dimensional SSF2.

The modulus of equation (3.0.1) at $t = 0.1$ using the two dimensional second-order SSF.

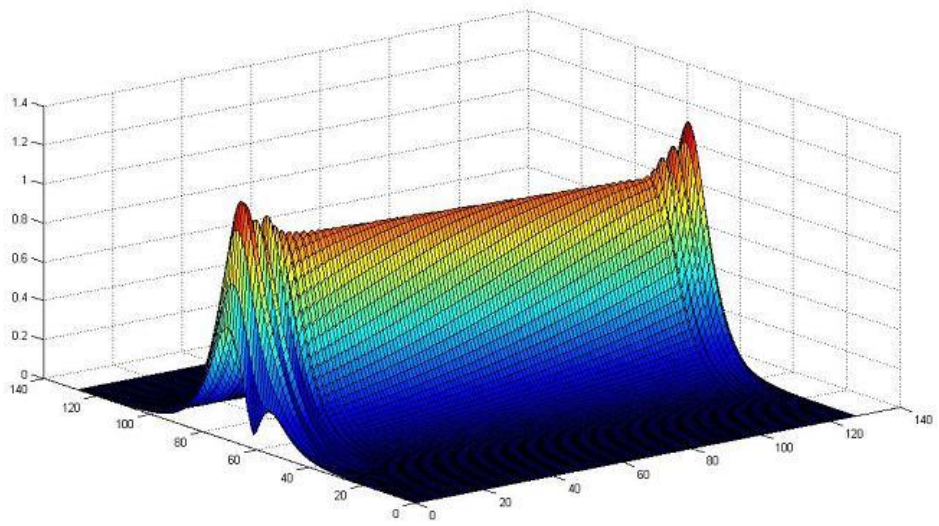

Figure 3.4: Two dimensional SSF4.

The modulus of equation (3.0.1) at $t = 0.1$ using the two dimensional fourth-order SSF, where $N = 128$.

3.3 PSEUDO-SPECTRAL METHOD AND EXPERIMENTS

Pseudo-spectral method is a Fourier method in which q (x, y, t) is transformed into Fourier space with respect to x, y, and derivatives (or other operators) with respect to x, y are then made algebraic in the transformed variable. Again we normalize the spatial period to [0, 2π] for convenience [1].

With this scheme, $q_{xx} + q_{yy}$ could be computed as

$$F^{-1}\left(\left(m^2 + n^2\right)F\left(q\left(X,Y,t\right)\right)\right)$$

Then, combined with a leap frog time step the two dimensional NLS equation (3.1.2) is approximated by

$$q\left(X,Y,t+\Delta t\right) = q\left(X,Y,t-\Delta t\right) - 2i\Delta t\frac{p^2}{2p^2}F^{-1}\left(\left(m^2+n^2\right)F\left(q\left(X,Y,t\right)\right)\right)$$

$$-4i\Delta t\left|q\right|^2 q\,. \qquad (3.3.1)$$

Following the ideas of Fornberg and Whitham we make a modification in approximating the two dimensional NLS equation (3.1.2)

$$q\left(X,Y,t+\Delta t\right) = q\left(X,Y,t-\Delta t\right) + 2iF^{-1}\left(\sin\left(-\Delta t\frac{p^2}{2p^2}\left(m^2+n^2\right)\right)F\left(q\left(X,Y,t\right)\right)\right)$$

$$-4i\Delta t\left|q\right|^2 q\,. \qquad (3.3.2)$$

The algorithm to implement (3.3.1) and (3.3.2) is described as below:

First, we denote the initial condition as u_1, which is equal to q (x, y, 0) at the very beginning.

Second, we use u_2 to represent q (x, y, t), where $t = \Delta t$ at first.

Third, for $t = 2\Delta t$ to numsteps $\times\ \Delta t$ (which is 1), we do the following:

(1) $u_3 = -\Delta t \frac{p^2}{2p^2} F^{-1}\left(\left(m^2+n^2\right)F\left(q\left(X,Y,t\right)\right)\right)$ (for (3.3.1))

(or $u_3 = F^{-1}\left(\sin\left(-\Delta t \frac{p^2}{2p^2}\left(m^2+n^2\right)\right)F\left(u_2\right)\right)$ for (3.3.2))

(2) $u_2 = q(x,y,t)$

(3) $u = u_1 - 2i\left(2\Delta t\left|u_2\right|^2 u_2 - u_3\right)$

(4) $u_1 = u_2$

$u_2 = u$

Finally, we obtain $q(x,y,t)$.

The approximation of the linear part of (3.1.2) is the difference between (3.3.1) and (3.3.2). Any solution of (3.1.5) would exactly satisfy the linear part of (3.3.2). And (3.3.1) is linearly stable for $\Delta t/(\Delta x)^2 < 1/p^2$. However, according to linear analysis (3.3.2) is unconditionally stable [1].

The solutions of the two dimensional NLS equation (3.0.1) at t = 0.1with the initial condition (3.1.1) using the pseudo-spectral methods with Δt = 0.000125 and N = 256 are shown below:

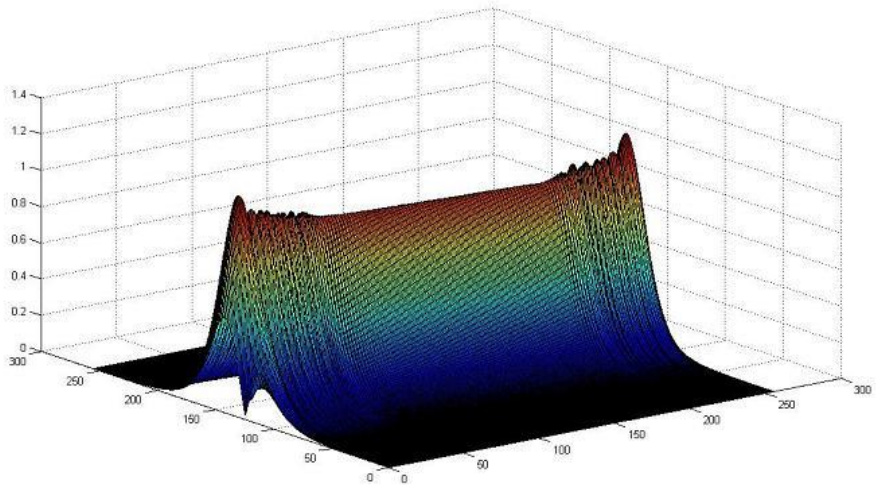

Figure 3.5: Two dimensional pseudo-spectral methods.

The modulus of equation (3.0.1) at $t = 0.1$ using the two dimensional pseudo-spectral method.

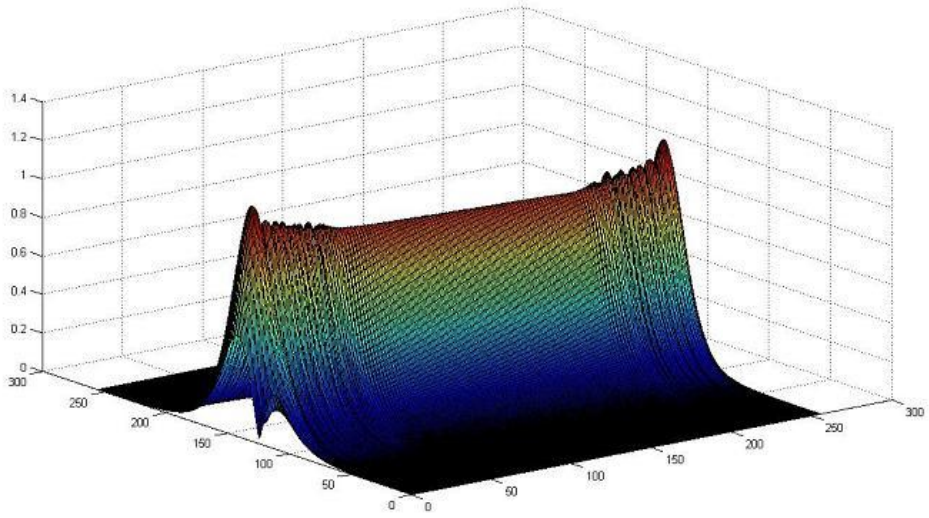

Figure 3.6: Modified two dimensional pseudo-spectral methods.

The modulus of equation (3.0.1) at $t = 0.1$ using the modified two dimensional pseudo-spectral method.

3.4 FINITE DIFFERENCE METHOD AND EXPERIMENTS

3.4.1 EXPLICIT METHOD

By using the explicit method, at time t_{n+1} and a certain point (X, Y), where $X = i * x$ and $Y = j * y$, we have

$$i\left(\frac{q_{i,j}^{n+1} - q_{i,j}^{n}}{\Delta t}\right) = -\frac{1}{2}\left(\frac{q_{i+1,j}^{n} - 2q_{i,j}^{n} + q_{i-1,j}^{n}}{(\Delta x)^2} + \frac{q_{i,j+1}^{n} - 2q_{i,j}^{n} + q_{i,j-1}^{n}}{(\Delta y)^2}\right) - \left|q_{i,j}^{n}\right|^2 q_{i,j}^{n} \qquad (3.4.1)$$

To obtain q (x, y, t), we take the following steps:

First, we denote the initial condition as u_1, which is equal to q (x, y, 0) at the very beginning.

Second, we use u_2 to represent q (x, y, t), where $t = \Delta t$ at first.

Third, for $t = \Delta t$ to numsteps $\times \Delta t$ (which is equal to 1), we do the following:

(1) $u_2(i\Delta x, j\Delta y) = \dfrac{u_1((i+1)\Delta x, j\Delta y) - 2u_1(i\Delta x, j\Delta y) + u_1((i-1)\Delta x, j\Delta y)}{2(\Delta x)^2}$

(2) $u_2(i\Delta x, j\Delta y) = u_2(i\Delta x, j\Delta y) + \dfrac{u_1(i\Delta x, (j+1)\Delta y) - 2u_1(i\Delta x, j\Delta y) + u_1(i\Delta x, (j-1)\Delta y)}{2(\Delta y)^2}$

(3) $u_2(i\Delta x, j\Delta y) = u_2(i\Delta x, j\Delta y) + u_1(i\Delta x, j\Delta y) * \left|u_1(i\Delta x, j\Delta y)\right|^2$

(4) $u_2(i\Delta x, j\Delta y) = u_2(i\Delta x, j\Delta y) * i\Delta t$

(5) $u_2(i\Delta x, j\Delta y) = u_2(i\Delta x, j\Delta y) + u_1(i\Delta x, j\Delta y)$

And finally, we obtain $q(x, y, t)$.

To do the numerical experiments, we used $N = 256$ for different values of time steps Δt to test the accuracy of the explicit method that are utilized in solving (1.1). The results are shown in Table 3.3.

Table 3.3: Convergence rates in time for the explicit method

($N = 256, -10 \le x \le 10\ ,\ -10 \le y \le 10\ ,\ 0 \le t \le 1,\ T = 0.5$) .

Δt	L_∞	L_2	i_1	cpu(s)
0.000125	1.545538E-01	2.127007 E-01	6.675301E-04	46.87
0.0001	1.545503E-01	2.127015E-01	6.655717E-04	57.62
0.00005	1.545478E-01	2.127060E-01	6.617672E-04	119.05
0.000025	1.545475E-01	2.127087E-01	6.598842E-04	218.77

We also used $\Delta t = 0.000125$ for different values of N to test the accuracy of the explicit method that are utilized in solving (1.1). The results are shown in Table 3.4.

Table 3.4: Convergence rates in space for the explicit method

($\Delta t = 0.000125\ , -10 \le x \le 10\ , -10 \le y \le 10\ ,\ 0 \le t \le 1,\ T = 0.5$) .

N	L_∞	L_2	i_1	cpu(s)
32	7.622602 E-02	2.339236 E-01	1.240102E-03	0.61
64	1.108153E-01	1.854285E-01	6.788211E-04	2.51
128	1.375294E-01	2.010176E-01	6.884926E-04	10.06
256	1.545538E-01	2.127007E-01	6.675301E-04	46.08
512	2.833494E-01	3.786824E-01	2.875514E-02	4227.40

The solutions of the two dimensional NLS equation (3.0.1) at $t = 0.1$with the initial condition (3.1.1) using the explicit method with $\Delta t = 0.000125$ and $N = 256$ are shown below:

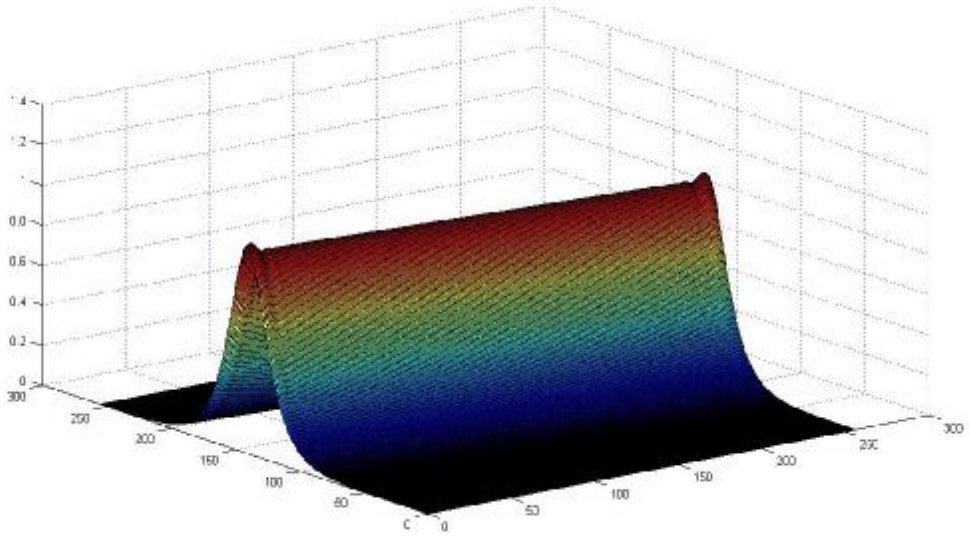

Figure 3.7: Two dimensional explicit method.

The modulus of equation (3.0.1) at $t = 0.1$ using the two dimensional explicit method.

3.4.2 IMPLICIT METHOD

For the implicit part, we use alternating direction implicit (ADI) method.

With ADI method, at time t_{n+1} and a certain point (X, Y), where $X = i * \Delta x$ and $Y = j * \Delta y$, we have

$$\frac{i(q_{k,j}^{n+1/2} - q_{k,j}^{n})}{\Delta t/2} = -\left|q_{k,j}^{n}\right|^2 q_{k,j}^{n} - \frac{1}{2}\left(\frac{q_{k+1,j}^{n+1/2} - 2q_{k,j}^{n+1/2} + q_{k-1,j}^{n+1/2}}{(\Delta x)^2} + \frac{q_{k,j+1}^{n} - 2q_{k,j}^{n} + q_{k,j-1}^{n}}{(\Delta y)^2}\right) \tag{3.4.2}$$

and

$$\frac{i(q_{k,j}^{n+1} - q_{k,j}^{n+1/2})}{\Delta t/2} = -\left|q_{k,j}^{n+1/2}\right|^2 q_{k,j}^{n+1/2} - \frac{1}{2}\left(\frac{q_{k+1,j}^{n+1/2} - 2q_{k,j}^{n+1/2} + q_{k-1,j}^{n+1/2}}{(\Delta x)^2} + \frac{q_{k,j+1}^{n+1} - 2q_{k,j}^{n+1} + q_{k,j-1}^{n+1}}{(\Delta y)^2}\right) \tag{3.4.3}$$

Then with (3.4.2) and (3.4.3), we do the following to obtain $q(x, y, t)$ [29]:

First, we implement the ADI method in a loop over the *y*-direction:

for $j = 1: N_y$

for $k = 1: N_x$

$$g(k) = (k + 2r)q_{k,j} - rq_{k,j+1} - rq_{k,j-1} - \frac{\Delta t}{2}\left|q_{k,j}\right|^2 q_{k,j}$$

end

solve $Aq^{new}(:, j) = g$

end

where,

$$\mathrm{A} = \begin{bmatrix} i-2r & r & 0 & 0 & 0 & 0 & 0 \\ r & i-2r & r & 0 & 0 & 0 & 0 \\ 0 & r & i-2r & r & 0 & 0 & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 0 & 0 & r & i-2r & r & 0 \\ 0 & 0 & 0 & 0 & r & i-2r & r \\ 0 & 0 & 0 & 0 & 0 & r & i-2r \end{bmatrix}$$

And q^{new} is an intermediate stage.

To get A, first, from (3.4.2), we have

$$(i-2r)q_{k,j}^{n+1/2} + rq_{k+1,j}^{n+1/2} + rq_{k-1,j}^{n+1/2} = (i+2r)q_{k,j} - rq_{k,j+1} - rq_{k,j-1} - \frac{\Delta t}{2}\left|q_{k,j}\right|^2 q_{k,j} \qquad (3.4.4)$$

where $r = \dfrac{\Delta t}{4 * \Delta x^2}$.

Then, for $0 \le k < N_x$ and $0 \le j < N_y$ we put the coefficients of the left side of (3.4.4) into a matrix, which is *A*.

Second, we implement the ADI method in a loop over the *x*-direction:

for k = 1: N_x

for j = 1: N_y

$$g(j) = (i + 2r)q_{k,j} - rq_{k,j+1} - rq_{k,j-1} - \frac{\Delta t}{2}\left|q_{k,j}\right|^2 q_{k,j}$$

end

solve $Aq\ (k,:) = g$

end

where A could be obtained using the same method in the first stage, but here, $r = \frac{\Delta t}{4 * \Delta y^2}$

And finally, we obtain $q(x, y, t)$.

The execution of the first and second steps advances the solution with a Δt step in time, overwriting *q*.

The solutions of the two dimensional NLS equation (3.0.1) at t = 0.1with the initial condition (3.1.1) using the ADI method with Δt = 0.000125 and N = 256 are shown below:

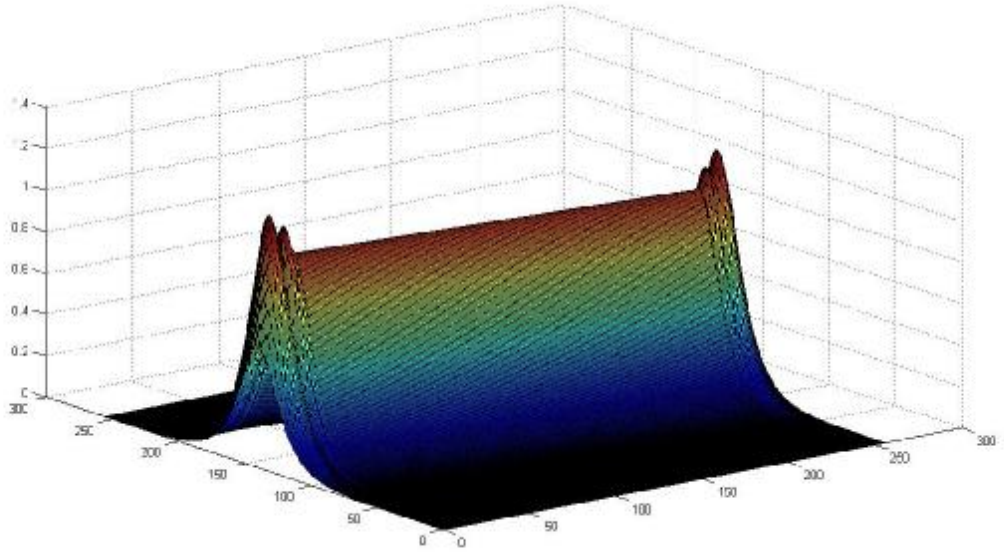

Figure 3.7: Two dimensional implicit methods.

The modulus of equation (3.0.1) at $t = 0.1$ using the two dimensional implicit method.

3.5 PARALLEL IMPLEMENTATION AND EXPERIMENTS

For the parallelization of the first-order split-step Fourier method, we parallelize it as following [3]:

Denote A, of size $N_x \times N_y$, as the approximation solution to q at time t. Assume that there are n processors in a distributed-memory parallel computer. The parallelization of (3.1.4) is straightforward. Then we distribute A among n processors. Each processor l with array elements $A[lN_x / n]$ to $A[(l+1)N_x / n-1]$, where $0 \le l \le n-1$, works on its own subarrays independently without communicating with others. After that, we employ FFTW's MPI routines to parallelize the calculation of $F\left(\hat{q}(X_j, Y_k, t+\Delta t)\right)$. For $\exp\left[-i\frac{p^2}{2p^2}\left(m^2+n^2\right)\Delta t\right]F\left(\hat{q}(X_m, Y_n, t)\right)$, its

parallelization is also straightforward. Finally, we use FFTW's MPI routines again to parallelize

$$q(X_j, Y_k, t+\Delta t) = F^{-1}\left(\exp\left[-i\frac{p^2}{2p^2}(m^2+n^2)\Delta t \right] F(\hat{q}(X_m, Y_n, t)) \right).$$

Similarly, we could parallelize for the second-order, fourth-order split-step Fourier methods and the pseudo-spectral method.

We implement the parallel algorithms of the split-step Fourier methods and the pseudo-spectral method on pcluster of UGA. And we optimize all the codes at the same optimization level.

In our simulations, speedup S_p is defined as

$$S_p = \frac{t_1}{t_n}.$$

where

t_1 = Time spent to run the MPI code on single processor

and

t_n = Time spent to run the MPI code on n processors

The results for parallel implementation of split-step method and pseudo-spectral method are shown in Table 3.5 to Table 3.9.

From the numerical experiments results we could observe clearly that S_p increases as the problem size $N_x \times N_y$ increases with a fixed number of processors *n* and when N is large, S_p obtained on the multiprocessor computer running the parallel codes is considerable.

Table 3.5: Results for parallel implementation of first-order splitting method (Δt = 0.000125). t_p is the time on p processors, S_p is the speedup on p processors.

	N = 128	N = 256	N = 512
t1 (sec)	98.53	411.44	2083.26
t2 (sec)	70.06	287.36	1361.45
t4 (sec)	35.51	139.81	602.16
t8 (sec)	18.20	71.36	320.14
S_2 = t1 / t2	1.41	1.43	1.53
S_4 = t1 / t4	2.77	2.94	3.46
S_8 = t1 / t8	5.41	5.77	6.51

Table 3.6: Results for parallel implementation of second-order splitting method (Δt = 0.000125). t_p is the time on p processors, S_p is the speedup on p processors.

	N = 128	N = 256	N = 512
t1 (sec)	116.87	486.84	2339.22
t2 (sec)	77.95	320.85	1447.12
t4 (sec)	40.31	155.12	663.23
t8 (sec)	21.40	78.92	348.62
S_2 = t1 / t2	1.50	1.52	1.62
S_4 = t1 / t4	2.90	3.14	3.53
S_8 = t1 / t8	5.46	6.17	6.71

Table 3.7: Results for parallel implementation of fourth-order splitting method (Δt = 0.000125). Array size is 128 for both x and y, t_p is the time on p processors, S_p is the speedup on p processors.

	$N = 128$
t1 (sec)	344.65
t2 (sec)	234.62
t4 (sec)	118.17
t8 (sec)	62.58
S_2 = t1 / t2	1.47
S_4 = t1 / t4	2.92
S_8 = t1 / t8	5.51

Table 3.8: Results for parallel implementation of the pseudo-spectral method (Δt = 0.000125). t_p is the time on p processors, S_p is the speedup on p processors.

	$N = 128$	$N = 256$	$N = 512$
t1 (sec)	90.31	378.95	1771.33
t2 (sec)	65.77	273.25	1204.99
t4 (sec)	33.72	132.39	574.37
t8 (sec)	16.95	68.88	292.64
S_2 = t1 / t2	1.37	1.39	1.47
S_4 = t1 / t4	2.68	2.86	3.08
S_8 = t1 / t8	5.33	5.50	6.05

Table 3.9: Results for parallel implementation of the modified pseudo-spectral splitting method ($\Delta t = 0.000125$). t_p is the time on p processors, S_p is the speedup on p processors.

	$N = 128$	$N = 256$	$N = 512$
t1 (sec)	98.83	411.52	2100.01
t2 (sec)	70.08	290.61	1093.76
t4 (sec)	35.40	141.17	610.10
t8 (sec)	20.64	74.73	307.96
S_2 = t1 / t2	1.41	1.42	1.92
S_4 = t1 / t4	2.79	2.92	3.44
S_8 = t1 / t8	4.79	5.51	6.82

For the explicit and implicit method, we do the parallelization for the calculation of q_{xx} and q_{yy} in each for loop. Results are shown in the following tables.

Table 3.10: Results for parallel implementation of the explicit method ($\Delta t = 0.000125$). t_p is the time on p processors, S_p is the speedup on p processors.

	$N = 128$	$N = 256$	$N = 512$
t1 (sec)	18.02	89.01	268.75
t2 (sec)	11.09	44.61	134.66
t4 (sec)	6.15	25.74	69.08
t8 (sec)	3.88	15.02	44.45
S_2 = t1 / t2	1.62	1.99	2.00
S_4 = t1 / t4	2.93	3.45	3.90
S_8 = t1 / t8	4.64	5.93	6.05

Table 3.11: Results for parallel implementation of the implicit method ($\Delta t = 0.000125$). t_p is the time on p processors, S_p is the speedup on p processors.

	$N = 128$	$N = 256$	$N = 512$
t1 (sec)	35.37	176.71	547.51
t2 (sec)	21.48	86.41	268.39
t4 (sec)	11.58	48.47	138.96
t8 (sec)	7.26	29.06	84.89
S_2 = t1 / t2	1.65	2.05	2.04
S_4 = t1 / t4	3.05	3.65	3.94
S_8 = t1 / t8	4.87	6.08	6.45

CHAPTER 4

RANKING

We have described seven methods to solve the two dimensional nonlinear Schrödinger equation. To test their performances and get their ranking based on their infinite norm, Euclidian norm and relative error, we have done the following experiments:

Let N equals to 128 and 256, respectively. For each N, we use the initial conditions:

$$k_1 = 0.6, \qquad k_2 = 0.8, \qquad B_1 = 1.0, \qquad B_2 = 0.1$$

for all these seven methods, record their results when $T = 0.5$.

Then for each result, we find out its infinite norm, Euclidian norm and relative error.

According to these criteria, we get the ranking:

1. Two dimensional explicit method.
2. Two dimensional split-step Fourier method using the first-order splitting method.
3. Two dimensional split-step Fourier method using the second-order splitting method.
4. Modified two dimensional pseudo-spectral method.
5. Two dimensional pseudo-spectral method.
6. Two dimensional split-step Fourier method using the fourth-order splitting method.
7. Two dimensional implicit method.

CHAPTER 5

CONCLUSION

In this thesis, we have applied the well-known split-step Fourier method, pseudo-spectral method and finite difference method for solving the two dimensional nonlinear Schrödinger equation. We have implemented three split-step Fourier method schemes, two pseudo-spectral method schemes and two finite difference method schemes. We find that the higher-order split-step scheme needs more computational time than the lower-order split-step methods.

For the parallel implementation of those schemes with fixed number of processors, we found out that as the problem size becomes larger, the speedup becomes larger. We also could achieve considerable speedups on the multiprocessor computer by running the parallel codes for large problem sizes.

REFERENCES

1. T. R. Taha, M. J. Ablowitz (1984). *Analytical and Numerical Aspects of Certain Nonlinear Evolution Equations. II. Numerical Nonlinear Schrödinger Equation*. Journal of Computational Physics, vol. 55, No. 2, pp. 203-230.
2. L. F. Mollenauer, R. H. Stolen, J. P. Gordon (1980). *Experimental observation of picosecond pulse narrowing and solitons in optical fibers*. Physical Review Letters, vol. 45, No. 13, pp. 1095 – 1098.
3. X. Xu, T. R. Taha (2004). *Parallel Split-step Fourier Methods for Nonlinear Schrödinger-Type Equations*. Journal of Mathematical modelling and Algorithms, vol. 2, No. 3, pp. 185 – 201.
4. E. Bouchbinder (2003). *The Nonlinear Schrödinger Equation*.
5. R. H. Hardin, F. D. Tappert (1973). *Applications of the split-step Fourier method to the numerical solution of nonlinear and variable coefficient wave equations*. SIAM Review Chronicle, vol. 15, pp. 423.
6. E. Knobloch, J. D. Gibbon (1991). *Coupled NLS equations for counter propagating waves in systems with reflection symmetry*. Physics Letters A, vol. 154, Issues 7 – 8, pp. 353 – 356.
7. A. Biswas (2009). *1-Soliton Solution of 1 + 2 Dimensional Nonlinear Schrödinger's Equation in Kerr Law Media*. International Journal of Theoretical Physics, vol. 48, No. 3, pp. 689 – 692.

8. J. A. C. Weideman, B. M. Herbst (1986). *Split-step methods for the solution of the nonlinear Schrödinger equation*. SIAM Journal on Numerical Analysis, vol. 23, Issue, pp. 485 – 507.
9. M. J. Ablowitz, H. Segur (1981). *Solitons and the Inverse Scattering Transform*. SIAM, Philadelphia.
10. G. M. Muslu, H. A. Erbay. *Numerical Simulation of Blow-up Solutions for the Generalized Davey-Stewartson System*. International Journal of Computer Mathematics.
11. D. G. Fox, S. A. Orszag (1973). *Pseudospectral Approximation to Two-Dimensional Turbulence*. Journal of Computational Physics, vol. 11, Issue 4, pp. 612 – 619.
12. M. S. Ismail, T. R. Taha (2001). *Numerical simulation of coupled nonlinear Schrödinger equation*. Mathematics and Computers in Simulation, Special Issue on "Optical Solitons".
13. S. Zoldi, V. Ruban, A. Zenchuk, S. Burtsev (1999). *Parallel implementation of the split-step Fourier method for solving nonlinear Schrödinger systems*. SIAM News, vol. 32, No. 1, pp. 8 – 9.
14. M. Frigo, S. G. Johnson (1997). *The Fastest Fourier Transform in the West*. Technical report, MIT – LCS – TR – 728, MIT Laboratory for Computer Science.
15. N. N. Akhmediev, A. Ankiewicz, R. Grimshaw (1999). *Hamiltonian-versus-energy diagrams in soliton theory*. Physical Review E, vol. 59, Issue 5, pp. 6088 – 6096.
16. A. Biswas, A. B. Aceves (2001). *Dynamics of solitons in optical fibers*. Journal of Modern Optics, vol. 48, Issue 7, pp. 1135 – 1150.
17. P. E. Zhidkov (2001). *Korteweg-de Vries and Nonlinear Schrödinger's Equation: Qualitative Theory*. Springer, New York.

18. Wikipedia. *Fourier Transform*.

Cited: Available at http://en.wikipedia.org/wiki/Fourier_transform.

19. J. W. Cooley, J. W. Tukey (1965). *An algorithm for the machine computation of complex Fourier series*. Mathematics of Computation, vol. 19, pp. 297 – 301.

20. M. Frigo (1999). *A fast Fourier transform compiler*. MIT Laboratory for Computer Science.

21. Wikipedia. *Split-step method*. Cited: Available at http://en.wikipedia.org/wiki/Split-step_method.

22. J. M. Sanz-Serna, M. P. Calvo (1994). *Numerical Hamiltonian problems*. Chapman & Hall, London.

23. R. McLachlan (1994). *Symplectic integration of Hamiltonian wave equations*. Numerical Mathematics, vol. 66, pp. 465 – 492.

24. G. M. Muslu, H. A. Erbay (2003). *A split-step Fourier method for the complex modified Korteweg-de Vries equation*. Computer & Mathematics with Applications, vol. 45, Issues 1 – 3, pp. 503 – 514.

25. S. A. Orszag (1972). *Comparison of Pseudospectral and Spectral Approximation*. Studies in Applied Mathematics, vol. 51, pp. 253 – 259.

26. Cited: Available at http://en.wikipedia.org/wiki/Message_Passing_Interface.

27. N. Asif, Shwetanshumala, S. Konar (2008). *Photovoltaic spatial soliton pairs in two-photon photorefractive materials*. Physics Letters A, vol. 372, Issue 5, pp. 735 – 740.

28. Wikipedia. *Finite difference method*.

Cited: Available at http://en.wikipedia.org/wiki/Finite_difference_method.

29. Cited: http://www.mth.pdx.edu/~daescu/mth410_510s/notes_week8.pdf.