ISTANBUL TECHNICAL UNIVERSITY
Faculty of Science and Letters – Mathematical Engineering

NUMERICAL SOLUTIONS OF (2+1) NONLINEAR SCHRÖDINGER EQUATION

GRADUATION PROJECT

Beycan KAHRAMAN
040020337

Department: Mathematical Engineering

Advisor: Ass. Prof. Nalan ANTAR

May 2008

ISTANBUL TECHNICAL UNIVERSITY
Faculty of Science and Letters – Mathematical Engineering

NUMERICAL SOLUTIONS OF (2+1) NONLINEAR SCHRÖDINGER EQUATION

GRADUATION PROJECT

Beycan KAHRAMAN
040020337

Department: Mathematical Engineering
Advisor: Ass. Prof. Nalan ANTAR

May 2008

NUMERICAL SOLUTIONS OF (2+1) NONLINEAR SCHRÖDINGER EQUATION (SUMMARY)

The aim of this project is to analyze Schrödinger's equation in three dimensional space. We are working on Yang's paper as a reference for our studies. In addition, we have studied renormalization methods and exponential time differencing for stiff systems. We have also studied programming spectral methods in matlab. During this analysis, we are dealing with spectral analysis, Fourier space and fast Fourier transform.

In the study, we could find a mode with renormalization method and using this mode as a initial condition for Schrödinger's equation we have observed its long term stability under some perturbation.

We have compared our results with a finite difference matlab program in order to sure about the results. As it predicted, we have seen that spectral methods are faster than other methods.

As a last test, we have observed both added-potential case and without potential case equations to analyze the effect of field intensity of the optical lattice. The results show that, without potential case, the system is more stable than the potential added case.

We do not study on penrose and vortex initial conditions, which have more than one solitons. We predict that these will give worse stability regions than one soliton give, from similar studies. They will be gratifying research areas for our next studies. In addition, we would study on eigenvalues, eigenmodes and eigenvectors of Schrödinger's equation in order to examine stability boundaries and finding the best initial constant values.

(2+1) NON-LİNEER SCHRÖDINGER DENKLEMİNİN SAYISAL ÇÖZÜMÜ

(ÖZET)

Bu projenin amacı (2+1) non-lineer Schrödinger denkleminin spectral yöntemlerle sayısal olarak çözülmesidir. Bu konuda Yang'ın makalesini referans olarak kullandık. Ayrıca renormalizasyon yöntemi [2-4] ve stiff sistemler için üstel zaman farklandırma yöntemi [5-7] üzerine çalıştık. Matlab'da bu konuda istediğimiz programları yapabilmek için Trefethen'in kaynakları [8-9] ile spectral yöntemlerin matlab'ta nasıl kodlanabileceğini öğrendik. Yapılan analiz sürecinde, spectral analizin yanı sıra, Fourier uzayı ve hızlı Fourier dönüşümü üzerine yoğun çalışmalarımız oldu.

In the study, we could find a mode with renormalization method and using this mode as a initial condition for Schrödinger's equation we have observed its long term stability under some perturbation.

We have compared our results with a finite difference matlab program in order to sure about the results. As it predicted, we have seen that spectral methods are faster than other methods.

As a last test, we have observed both added-potential case and without potential case equations to analyze the effect of field intensity of the optical lattice. The results show that, without potential case, the system is more stable than the potential added case.

We do not study on penrose and vortex initial conditions, which have more than one solitons. We predict that these will give worse stability regions than one soliton give, from similar studies. They will be gratifying research areas for our next studies. In addition, we would study on eigenvalues, eigenmodes and eigenvectors of Schrödinger's equation in order to examine stability boundaries and finding the best initial constant values.

Table of Contents

Table of Graphs

1. Introduction

The premise of this project is to analyze Schrödinger's equation in three dimensional space. We are interested in Yang's paper [1] as a reference for our studies. Additionally, we have benefited from [2-4] for renormalization methods and [5-7] for exponential time differencing for stiff systems. We have studied [8-9] in order to learn programming spectral methods in matlab. During this analysis, we are dealing with spectral analysis, Fourier space and fast Fourier transform.

We have confirmed the results in [1]; that is, finding a mode with renormalization method and using this mode as a initial condition for Schrödinger's equation to observe its long term stability with some perturbation.

In the next section, we will practice spectral analysis. Additionally, Fourier space and Fourier methods will be given in order to ensure the backward knowledge about this study. In the 3rd section, we will introduce renormalization method and exponential time differencing for stiff systems. In the 4th section, after studying necessary matlab commands, we will discuss finding a mode for Schrödinger's equation and examine the stability duration for this mode with a matlab program. In the last section, we will discuss the results and state the future works.

2. Spectral Analysis and Fourier Transform

2.1. Spectral Analysis

We are using spectral analysis for solving a wide variety of practical problems encountered by engineers and scientists in a very wide field of engineering and science. In most fields the functions in spectral analysis are temporal or spatial waveforms or discrete data. The purpose of spectral analysis is to represent a function by some of trigonometric functions called spectral components; that is, the purpose is decompose the function into these spectral components. The weighting function in the decomposition is a density of spectral components (spectrum). The reason we represent a function with its spectrum is that it can be an efficient, convenient and often revealing description of the given function.

For time-functions, the spectrum is obtained from an invertible transformation from a time-domain to a frequency-domain description. We are using sine wave components in this transformation because of our preoccupation with linear-time-invariant data sources and transformations, which we often call filters [10].

2.2. Fourier Transform

To understand the definition of Fourier transform we will begin with essential idea: representing a wave form in terms of frequency as opposed in time. Suppose our wave form is described by $4\cos 2\pi vt$, which has a frequency *v*. Using Euler's identity

$$e^{i\phi} = \cos\phi + i\sin\phi$$

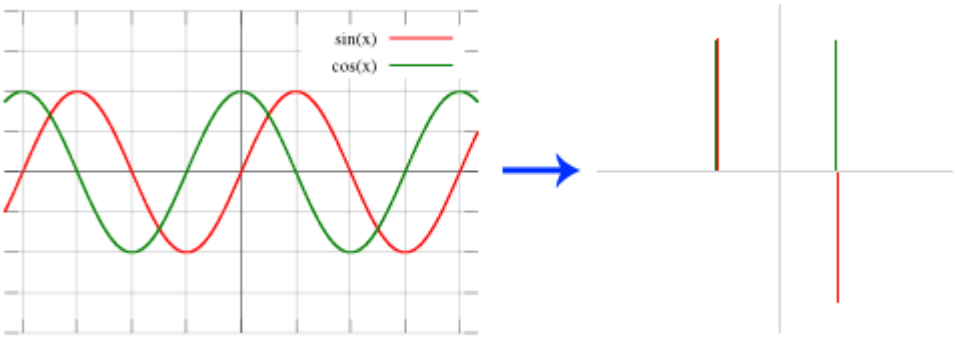
we can write $4\cos 2\pi vt$ in complex exponential form

$$4\cos 2\pi vt = 2e^{i2\pi vt} + 2e^{-i2\pi vt}$$

where the complex essentials have amplitudes of 2 and frequencies of *v* and –*v*. Or, another wave form

$$9\sin 2\pi vt = 3ie^{-i2\pi vt} - 3ie^{i2\pi vt}$$

which have complex amplitudes of *3i* and *-3i* and frequencies of –*v* and *v*.



Graph 1: Time Domain versus Frequency Domain

These two examples show how the waves $4\cos 2\pi vt$ and $9\sin 2\pi vt$ can be expressed in frequency terms and distinguished from each other using complex exponentials [11].

Now, we will introduce Fourier transform [8, 12-13]. The Fourier transform $\hat{u}(k)$ of a function $u(x)$ is defined as,

$$\hat{u}(k) = \int_{-\infty}^{\infty} e^{-ikx}.u(x).dx, \quad k \in R$$

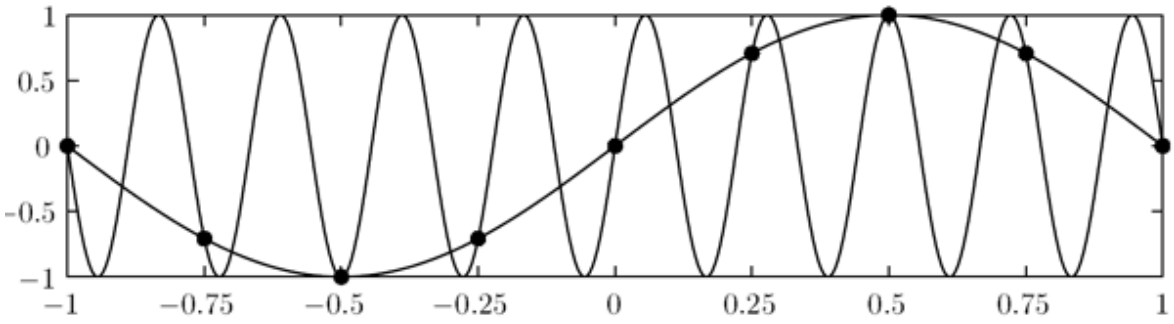
where $\hat{u}(k)$ is the amplitude density of $u(x)$ at wavenumber k, and this process of decomposing a function into its constituent waves is called Fourier analysis. Conversely, $u(x)$ could be reconstructed from $\hat{u}(k)$ by Fourier synthesis:

$$u(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{ikx}.\hat{u}(k).dk, \quad x \in R$$

2.2.1. The Semidiscrete Fourier Transform

We need to choose x from hZ rather than from R, because the spatial domain is discrete and the wavenumbers will no longer range over R. Instead, the appropriate wavenumber domain is bounded interval of $2\pi/h$ where a suitable choice is $[-\pi/h, \pi/h]$.

From discrete and unbounded physical space, we transform the function into bounded and continues Fourier space. Using the Fourier synthesis we turn back physical space. The reason for these connections is aliasing; that is, we have infinitely many complex essentials that match the functions on the grid hZ of k. That's why we choose the interval $[-\pi/h, \pi/h]$.



Graph 2: An Example of Aliasing on the Grid 0.25Z, sin(πx) is Identical to sin(9πx).

Now; for a function *v* defined on *hZ* with value v_j at x_j, the semidiscrete Fourier transform is defined by,

$$\hat{v}(k) = h\sum_{j=-\infty}^{\infty} e^{-ikx}.v_j, \qquad k \in [-\pi/h, \pi/h]$$

and the inverse semidiscrete Fourier transform is

$$v_j = \frac{1}{2\pi}\int_{-\pi/h}^{\pi/h} e^{ikx}.\hat{v}(k).dk, \qquad j \in Z\,.$$

If *u* is a differentiable function with Fourier transform $\hat{u}$, then:

$$\hat{u}'(k) = ik\hat{u}(k)\,.$$

2.2.2. The Discrete Fourier Transform (DFT)

In this part, we will examine spectral differentiation on a bounded, periodic grid. This process was stated in the form of an *NxN* matrix operation. The semi-discrete Fourier transform is replaced by discrete Fourier transform. In discrete case, the requirement of periodicity may suggest that this method has limited relevance for practical problems; however, periodic grids are surprisingly useful in practice. We will use a periodic grid of the interval [*0, 2π*]. In addition the grid will be divided into *N* grid points which is always even for our usage [12,13].

In discrete Fourier transformation case; the function *u* is transformed between discrete, bounded physical space and discrete, bounded Fourier space [8].

The discrete Fourier transform formula could be given as:

$$\hat{v}_k = h\sum_{j=1}^{N} e^{-ikx_j} v_j, \qquad k = -\frac{N}{2}+1, ..., \frac{N}{2}$$

On the other hand, the inverse discrete Fourier transform is given by:

$$v_j = \frac{1}{2\pi} \sum_{k=-N/2+1}^{N/2} e^{ikx_j} \hat{v}_k, \qquad j = 1, ..., N$$

Discrete Fourier transform has linearity, periodicity and inversion properties. The last property allows us to define the inverse discrete Fourier transform. There are additional information and theorems about these properties in [11].

As a last step, the system should behave symmetrically. Therefore, by using $\hat{v}_{-N/2} = \hat{v}_{N/2}$,

$$v_j = \frac{1}{2\pi} \sum_{k=-N/2}^{N/2} e^{ikx_j} \hat{v}_k, \qquad j = 1, ..., N.$$

The source code of discrete Fourier transform could be found in [14].

2.2.3. The Fast Fourier Transform (FFT)

A direct calculation of an *N*-point DFT requires $(N-1)^2$ multiplications and $N(N-1)$ additions. For large *N*, say $N > 1000$, this operation requires too much CPU time. In 1965 Cooley and Tukey discovered an new method that calculates the discrete Fourier transform in $O(N \log N)$ floating point operations, which will reduce the calculation time by a factor of 200 when $N = 1024$ [8,11].

Today, there are many algorithms for calculating DFT faster. One of the most widely used FFT algorithm is radix 2, where $N = 2^R$ $(R \in Z^+)$ will increase the performance.

Radix-2 FFT

In radix 2 algorithm, we start by halving the *N*-point DFT into two sums, each of which is *N/2*-point DFT. By using Buneman's bit reversal algorithm with using weights and crosswise operations on each step the DFT is redacted in stages [11]. Additionally, rotations in FFT will increase the performance.

Let us examine fast Fourier transform step by step. We should calculate the N-point DFT

$$H_k = \sum_{j=0}^{N-1} h_j W^{jk}$$

where *W* stands for either weight $e^{i2\pi/N}$ or $e^{-i2\pi/N}$. The base 2, for $N = 2^R$ is often called radix 2. We will focus on the radix 2 method, because others could be determined from radix 2.

Let us begin with halving the N-point DFT into two sums as

$$H_k = \sum_{j=0}^{\frac{1}{2}N-1} h_{2j}(W^2)^{jk} + \sum_{j=0}^{\frac{1}{2}N-1} h_{2j+1}(W^2)^{jk}W^k = H_k^0 + W^k H_k^1$$

with even and odd sub parameters. Notice that N/2-point DFTs $\{H_k^0\}$ and $\{H_k^1\}$ use weight W^2 (not W). The periods of $\{H_k^0\}$ and $\{H_k^1\}$ are N/2, and the elements are $\{h_0, h_2, ..., h_{N-2}\}$ and $\{h_1, h_3, ..., h_{N-1}\}$ respectively.

Since $N = 2^R$ we could divide N/2 evenly by 2, and using $W^{N/2} = -1$ we could write

$$H_k = H_k^0 + W^k H_k^1 \qquad \Rightarrow \qquad H_{k+N/2} = H_k^0 - W^k H_k^1$$

The calculations can be diagrammed as

$$\begin{array}{ccc} H_k^0 & \rightarrow & H_k^0 + W^k H_k^1 \\ & \times & \\ H_k^1 & \rightarrow & H_k^0 - W^k H_k^1 \end{array} \qquad \left(k = 0,1,...,\frac{1}{2}N - 1\right)$$

where the diagram is called the butterfly.

The splitting of $\{H_k\}$ into two half-size DFTs, could be repeated $R=\log_2 N$ stages where we are performing N one-point DTFs. Another important step is requirement for reversal of the last two bits (digits). Let's examine N/4-point DFT.

$$H_k^0 = H_k^{00} + (W^2)^k H_k^{01} \qquad H^0_{k+\frac{1}{4}N} = H_k^{00} - (W^2)^k H_k^{01}$$

$$H_k^1 = H_k^{10} + (W^2)^k H_k^{11} \qquad H^1_{k+\frac{1}{4}N} = H_k^{10} - (W^2)^k H_k^{11}$$

Here, the sets could be given as:

$$\{H_k^{00}\} = DFT\{...00\}$$
$$\{H_k^{01}\} = DFT\{...10\}$$
$$\{H_k^{10}\} = DFT\{...01\}$$
$$\{H_k^{11}\} = DFT\{...11\}$$

Therefore, to begin the FFT calculation, one must first rearrange $\{H_k\}$ so it is listed in *bit reverse order*.

Bit Reversal

Buneman's algorithm is the simplest method for performing the bit reversal permutation. It is based on a simple pattern. If we have permuted the N numbers {0, 1, ..., N-1} by bit reversing their binary expansions, then the permutation of the 2N numbers {0, 1, ..., 2N-1} is obtained by doubling the numbers in the permutation of {0, 1, ..., N-1} to get the first N numbers and then adding 1 to these doubled numbers to get the last N numbers.

0 1

0 2 $\xrightarrow{+1}$ 1 3

0 4 2 6 $\xrightarrow{+1}$ 1 5 3 7

...

Let us prove the algorithm. Suppose m has the following binary expansion

$$m = (a_1\, a_2 \ldots a_R)_{base2}$$

where each element is either 0 or 1. The number m is mapped to $P_N(m)$, its bit reversed image, hence

$$P_N(m) = (a_1\, a_2 \ldots a_R)_{base2}$$

If we double $P_N(m)$, then

$$2P_N(m) = (a_1\, a_2 \ldots a_R\, 0)_{base2}$$

And we see that $2P_N(m)$ is the bit reversed image of

$$m = (0\, a_1\, a_2 \ldots a_R)_{base2}$$

where m is considered as an element of {0, 1, ..., 2N-1} for the first N numbers. Furthermore,

$$2P_N(m) + 1 = (a_1\, a_2 \ldots a_R\, 1)_{base2}$$

is the bit reversal of

$$m = (1\, a_1\, a_2 \ldots a_R)_{base2}$$

which accounts for the last N numbers in the list. While Buneman's method is admirably simple, it has the defect that it performs unnecessary swap operations. It can be improved by more efficient algorithms that perform only those swaps that are absolutely necessary.

FAS Algorithm

This algorithm splits P_N into two bit reversal permutations of square root size, that is:

$$m = (a_1\, a_2 \ldots a_Q\, b_1\, b_2 \ldots b_Q)_{base2}$$

where m could be written as

$$m = KM + L.$$

Therefore,

$$P_N(m) = MP_N(L) + P_N(K)$$

could be calculated faster than Buneman's method [11].

As in DFT part, the source code of fast Fourier transform could be found in [14].

3. Renormalization and Exponential Time Differencing

We are dealing with the Schrödinger's equation

$$iU_z + U_{xx} + U_{yy} - \frac{E_0}{1 + I_0 \sin^2 x \sin^2 y + |U|^2} U = 0 .$$

Using the renormalization idea proposed in [15], we should find a non-linear equation where the initial condition u(x, y) satisfies it.

3.1. Renormalization

Lattice vortices of equation

$$iU_z + U_{xx} + U_{yy} - \frac{E_0}{1 + V + \varepsilon |U|^2} U = 0$$

are sought in the form of

$$u(x, y, z) = U(x, y) e^{-i\mu z} .$$

Therefore, the equation becomes

$$\mu u + u_{xx} + u_{yy} - \frac{E_0 u}{1 + V + \varepsilon |u|^2} = 0$$

Now, lets pass to Fourier space:

$$\mu \hat{u} - k^2 \hat{u} - F\left(\frac{E_0 u}{1 + V + \varepsilon |u|^2} \right) = 0$$

Changing parameter

$$\hat{u} = \lambda \hat{w}$$

the equation will be

$$\mu \hat{w} - k^2 \hat{w} - F\left(\frac{E_0 w}{1 + V + \varepsilon |\lambda|^2 |w|^2} \right) = 0 .$$

Now, adding $r\hat{w}$ to both sides:

$$\mu \hat{w} - k^2 \hat{w} + r\hat{w} - r\hat{w} - F\left(\frac{E_0 w}{1 + V + \varepsilon |\lambda|^2 |w|^2} \right) = 0$$

$\hat{w}$ could be calculated as:

$$\hat{w} = \frac{1}{k^2 + r} \left((\mu + r)\hat{w} - F\left(\frac{E_0 w}{1 + V + \varepsilon |\lambda|^2 |w|^2} \right) \right) .$$

Multiplying by complex conjugate $\hat{w}*$

$$\hat{w}\hat{w}^* = \frac{1}{k^2 + r}\left((\mu + r)\hat{w}\hat{w}^* - \hat{w}^* F\left(\frac{E_0 w}{1 + V + \varepsilon\,|\,\lambda\,|^2 |\,w\,|^2}\right)\right)$$

And lastly integrating over k we will get:

$$\int\int \hat{w}\hat{w}^*\,dk = \int\int \frac{1}{k^2 + r}\left((\mu + r)\hat{w}\hat{w}^* - \hat{w}^* F\left(\frac{E_0 w}{1 + V + \varepsilon\,|\,\lambda\,|^2 |\,w\,|^2}\right)\right)dk$$

Defining objective function $G(\mu, r)$ as

$$G(\mu, r) = \int\int \hat{w}\hat{w}^*\,dk - \int\int \frac{1}{k^2 + r}\left((\mu + r)\hat{w}\hat{w}^* - \hat{w}^* F\left(\frac{E_0 w}{1 + V + \varepsilon\,|\,\lambda\,|^2 |\,w\,|^2}\right)\right)dk$$

we could approximate to the exact $\hat{w}$ with an initial U. Then we could calculate $\hat{u}$, u and U respectively.

r could be used as a constant value, $\varepsilon = \pm 1$; μ and E_0 system dependent variables and lastly k will be used as:

$$\hat{\dot{u}} = ik\hat{u} \quad (1D)$$
$$k = k_x \hat{i} + k_y \hat{j} \quad (2D)$$
$$k^2 = k_x^2 + k_y^2$$

3.2. Exponential Time Differencing

Cox and Matthews has developed a class of numerical methods for stiff systems, based on the method of exponential time differencing [5]. We will describe these schemes with second and higher order accuracy, and introduce new Runge Kutta versions of these schemes, and lastly extending the method to show how may it be applied to systems whose linear part is nondiagonal.

Although our primary interest lies in solving PDEs, it is clearer and more instructive first to describe ETD methods in the context of simple model ODE for the evolution of a single Fourier mode. The model ODE is

$$\dot{u} = cu + F(u, t), \tag{1}$$

where *c* is a constant and *F(u,t)* represents nonlinear and forcing terms. For the high-order Fourier modes, *c* is large and negative (for dissipative PDEs) or large and imaginary (for dispersive PDEs).

3.2.1. Exponential Time Differencing Method

To derive the exponential time differencing (ETD) methods, we begin by multiplying the equation with e^{-ct}, then integrating the equation over a single time step from $t = t_n$ from $t = t_{n+1} = t_n + h$ to give

$$u(t_{n+1}) = u(t_n)e^{ch} + e^{ch}\int_0^h e^{-c\tau}F(u(t_n+\tau), t_n+\tau)d\tau. \tag{2}$$

This formula is exact, and the essence of the ETD methods is in deriving approximations to the integral in this expression. The simplest exponential time differencing method, ETD1, could be achieved by choosing *F* constant. Therefore, the method become,

$$u_{n+1} = u_n e^{ch} + F_n(e^{ch} - 1)/c, \qquad F_n = F(u_n, t_n), \tag{3}$$

which has a local truncation error $h^2\dot{F}/2$. This version of exponential time differencing method has been applied in computational electrodynamics, but is rarely mentioned outside of this field in the numerical analysis literature [5].

If instead of assuming that *F* is constant over the interval $t_n \le t \le t_{n+1}$, and use higher approximations we arrive at higher order numerical schemes. For choosing,

$$F = F_n + \tau(F_n - F_{n-1})/h + O(h^2), \tag{4}$$

We arrive at the numerical scheme ETD2 given by,

$$u_{n+1} = u_n e^{ch} + F_n\left((1+hc)e^{ch} - 1 - 2hc\right)/hc^2 + F_{n-1}\left(-e^{ch} + 1 + hc\right)/hc^2, \tag{5}$$

which has a local truncation error of $5h^3\ddot{F}/12$.

3.2.1. Exponential Time Differencing Method with Runge-Kutta Time Stepping

The ETD methods described above are of multistep type, requiring previous evaluations of the nonlinear term *F*. Such methods are often inconvenient to use, since initially only one value is available. This problem could be avoided by Runge-Kutta methods, which also typically have the advantages of smaller error constants and larger stability regions than multistep methods. We will obtain ETD methods of RK type of orders 2, 3, and 4.

3.2.1.1. Second-Order Runge-Kutta ETD Method

A second order ETD method of RK type, analogous to the "improved Euler" method, is as follows. First, the step is taken to give

$$a_n = u_n e^{ch} + F_n(e^{ch} - 1)/c\,. \tag{6}$$

Then the approximation

$$F = F(u_n, t_n) + (t - t_n)\big(F(a_n, t_n + h) - F(u_n, t_n)\big)/h + O(h^2) \tag{7}$$

is applied on the interval $t_n \le t \le t_{n+1}$, and is substituted into (2) to yield the scheme ETD2RK given by

$$u_{n+1} = a_n + \big(F(a_n, t_n + h) - F_n\big)(e^{ch} - 1 - hc)/hc^2\,, \tag{8}$$

The truncation error per step for this method is $-h^3\ddot{F}/12$; smaller by a factor of 5 than that of EDT2.

3.2.1.2. Third-Order Runge-Kutta ETD Method

A third order EDT RK scheme can be constructed in a similar way, analogous to the classical third-order RK method: ETD3RK is given by

$$\begin{aligned}
a_n &= u_n e^{ch/2} + (e^{ch/2} - 1)F(u_n, t_n)/c,\\
b_n &= u_n e^{ch} + (e^{ch} - 1)\big(F(a_n, t_n + h/2) - F(u_n, t_n)\big)/c,\\
u_{n+1} &= u_n e^{ch} + \{F(u_n, t_n)\big[-4 - hc + e^{ch}(4 - 3hc + h^2c^2)\big]\\
&\quad + 4F(a_n, t_n + h/2)\big[2 + hc + e^{ch}(-2 + hc)\big]\\
&\quad + F(b_n, t_n + h)\big[-4 - 3hc - h^2c^2 + e^{ch}(4 - hc)\big]\big\}/h^2c^3
\end{aligned} \tag{9}$$

The terms a_n and b_n approximate the values of u at $t_n + h/2$ and $t_n + h$ respectively.

3.2.1.3. Fourth-Order Runge-Kutta ETD Method

A straightforward extension of the standard fourth-order RK method yields a scheme which is only third order. However, by varying the scheme and introducing further parameters, a fourth-order scheme ETD4RK is obtained:

$$\begin{aligned}
a_n &= u_n e^{ch/2} + (e^{ch/2} - 1)F(u_n, t_n)/c,\\
b_n &= u_n e^{ch/2} + (e^{ch} - 1)F(a_n, t_n + h/2)/c,\\
c_n &= a_n e^{ch/2} + (e^{ch/2} - 1)\big(2F(b_n, t_n + h/2) - F(u_n, t_n)\big)/c\\
u_{n+1} &= u_n e^{ch} + \{F(u_n, t_n)\big[-4 - hc + e^{ch}(4 - 3hc + h^2c^2)\big]\\
&\quad + 2\big(F(a_n, t_n + h/2) + F(b_n, t_n + h/2)\big)\big[2 + hc + e^{ch}(-2 + hc)\big]\\
&\quad + F(c_n, t_n + h)\big[-4 - 3hc - h^2c^2 + e^{ch}(4 - hc)\big]\big\}/h^2c^3
\end{aligned} \tag{10}$$

The computer algebra package Maple was used to confirm that this method is indeed fourth order [5].

4. Finding Mode and Numerical Stability Analysis

In the first part, we will introduce some frequently used functions in MATLAB. After defining these functions we will give some examples. In the second part we will explain how to find modes for the Schrödinger's equation in three dimensional space. After finding the mode, we will analyze numerical stability analysis for the calculated mode.

4.1 MATLAB Background

meshgrid: Used to generate two-dimensional X and Y matrices for three-dimensional plots by using one-dimensional x and y arrays.

Syntax:

```
[X,Y] = meshgrid(x,y)
[X,Y] = meshgrid(x)
[X,Y,Z] = meshgrid(x,y,z)
```

Example:

```
[X,Y] = meshgrid(1:3,10:14)

X =

     1     2     3
     1     2     3
     1     2     3
     1     2     3
     1     2     3

Y =

    10    10    10
    11    11    11
    12    12    12
    13    13    13
    14    14    14
```

conj: Used to calculate complex conjugate of x.

Syntax:

```
ZC = conj(Z)
```

Algorithm:

```
If Z is a complex array

conj(Z) = real(Z) - i*imag(Z)
```

fft2: Calculates two-dimensional Fourier transform

Syntax:

```
Y = fft2(X)
Y = fft2(X,m,n)
```

Algorithm:

```
fft2(X) can be simply computed as

fft(fft(X).').'
```

This computes the one-dimensional FFT of each column X, then of each row of the result. The time required to compute fft2(X) depends strongly on the number of prime factors in [m,n] = size(X). It is fastest when m and n are powers of 2.

ifft2: Calculates two-dimensional inverse Fourier transform

Syntax:

```
Y = ifft2(X)
Y = ifft2(X,m,n)
```

The algorithm for ifft2(X) is the same as the algorithm for fft2(X), except for a sign change and scale factors of [m,n] = size(X). The execution time is fastest when m and n are powers of 2 and slowest when they are large primes [16].

4.2 Finding Mode

In renormalization part, we have defined the objective function $G(\mu, r)$ as:

$$G(\mu,r) = \int\int \hat{w}\hat{w}^* dk - \int\int \frac{1}{k^2 + r}\left((\mu + r)\hat{w}\hat{w}^* - \hat{w}^* F\left(\frac{E_0 w}{1 + V + \varepsilon \mid \lambda \mid^2 \mid w \mid^2} \right) \right) dk \qquad (11)$$

So, we should find the roots of $G(\mu, r)$ in order to make it zero and equate the two integrals. We will use secant method for calculating the roots. So lets describe secant method.

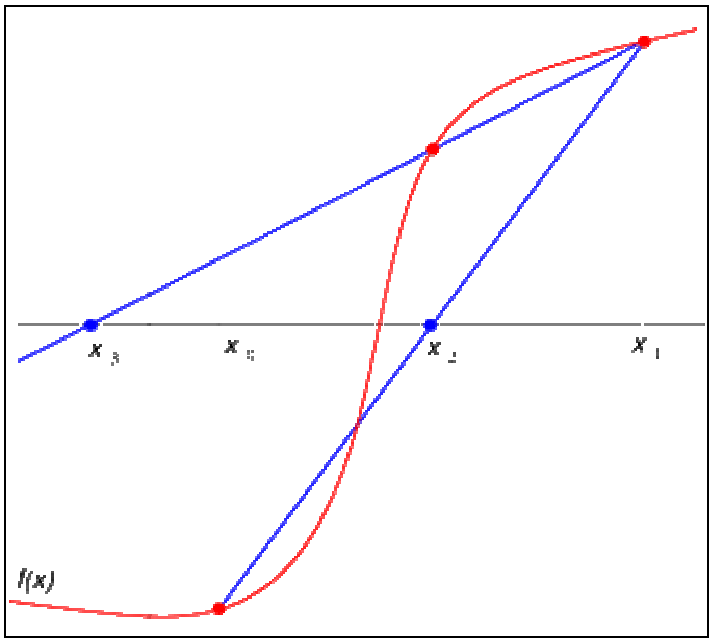
Secant Method

In numerical analysis, the secant method is a root-finding algorithm that uses a succession of roots of secant lines to better approximate a root of a function f.

The secant method is defined by the recurrence relation

$$x_{n+1} = x_n + \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n).$$

As can be seen from the recurrence relation, the secant method requires two initial values, x0 and x1, which should ideally be chosen to lie close to the root. The graph 3, shows the iterations of the method.



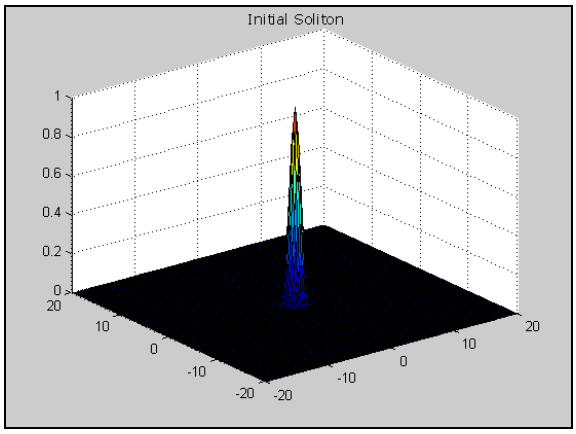
Graph 3: Secant Method

We have start programming with defining uniform grid points x_n and y_n (n: 1, 2, ..., 128) with the interval $(-\pi, \pi)$. By using meshgrid function we achieve, two-dimensional x and y matrices.

We have chosen the constant variables and perturbation function as:

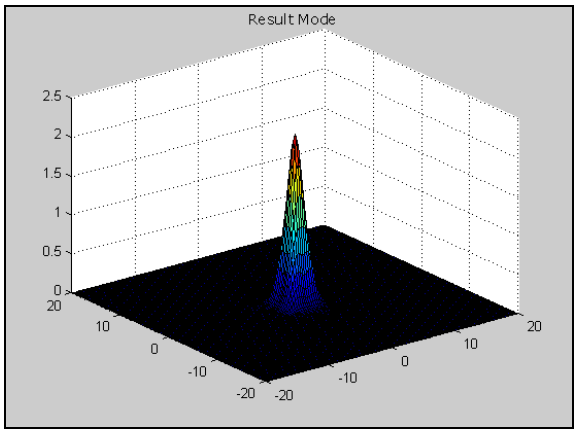
$$r = 10 \qquad \mu = 3.2$$
$$E_0 = 7.5 \qquad V(x, y) = 2\sin^2 x \sin^2 y$$

by using the values from [1]. Then we have applied the secant method for finding the roots of the equation (11) with the initial soliton of $u = e^{-x^2-y^2}$ (shown in graph 4) until a tolerance point.

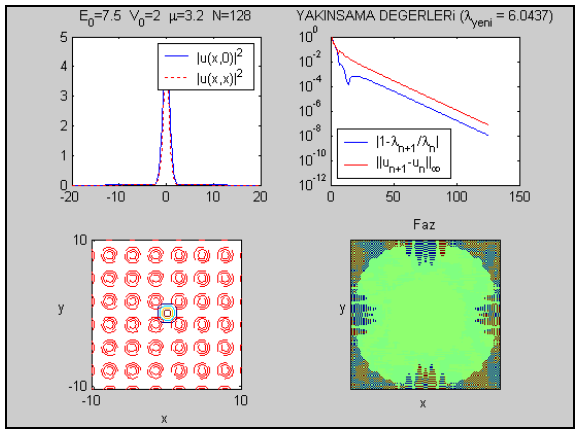


Graph 4: Initial Soliton

Since we have chosen the function depends on *w*, as a result depends on λ; we will find a λ_{best} for this step. Here we will calculate new w^1_{best} by λ_{best}. Then, we are continuing this process until a tolerance value where $\left|w^{n+1}_{best} - w^n_{best}\right|$ is used for comparing. At the end of this process, we will get the best-fit mode for our equation (shown in the graph 5), and the complex analysis and comparisons are shown in graph 6.



Graph 5: Best-Fit Mode



Graph 6: Analysis

4.3 Numerical Stability Analysis

We need to use fourth order Runge-Kutta method for advancing in time in calculating numerical stability analysis of the function. So lets start with describing fourth order Runge-Kutta method.

Fourth Order Runge-Kutta Method

In numerical analysis, the Runge–Kutta methods are an important family of implicit and explicit iterative methods for the approximation of solutions of ordinary differential equations. These techniques were developed around 1900 by the German mathematicians C. Runge and M.W. Kutta.

One member of the family of Runge–Kutta methods is so commonly used that it is often referred to as "RK4" or simply as "the Runge–Kutta method". Let an initial value problem be specified as follows.

$$y' = f(t, y) \qquad y(t_0) = y_0 .$$

Then, the RK4 method for this problem is given by the following equations:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \qquad t_{n+1} = t_n + h$$

where y_{n+1} is the RK4 approximation of $y(t_{n+1})$ and,

$$k_1 = f(t_n, y_n)$$
$$k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1)$$
$$k_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2)$$
$$k_4 = f(t_n + h, y_n + hk_3)$$

This method is reasonably simple and robust and is a good general candidate for numerical solution of differential equations when combined with an intelligent adaptive step-size routine.

In order to advance in time, we have used the initial mode found in previous part with RK4 in Fourier Space. In equation (11), $V(x, y)$ represents the field intensity of the optical lattice [1]. Here the intensities of the probe beam and the lattice have been normalized with respect to the dark irradiance of the crystal. The dark irradiance is the background illumination used in experiments to fine-tune the non-linearity. Material damping of the

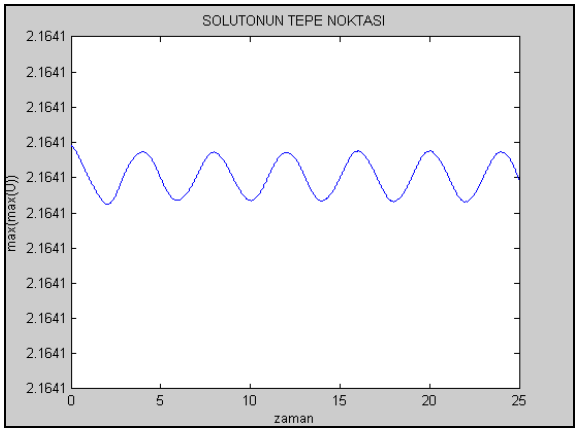
probe beam is very weak in typical experiments since the crystals are fairly short (up to 2 cm), hence neglected in equation (1). If the lattice is periodic along the x and y directions (rectangular lattice), then $V(x, y)$ can be expressed as

$$V(x, y) = I_0 \sin^2 x \sin^2 y$$

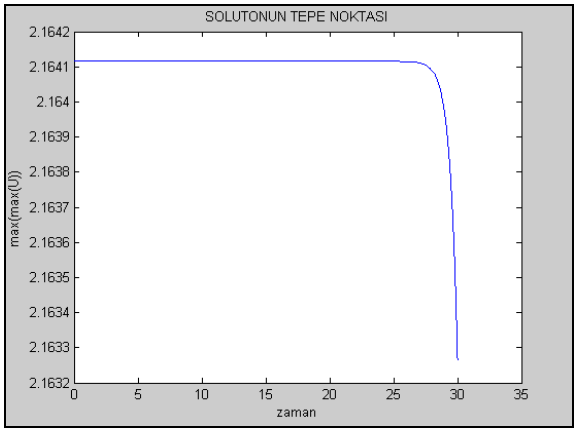
where I_0 is its peak intensity and D its spacing. In order to analyze the importance of field intensity in the equation, we have studied on numerical solution in two case: added-potential and without potential ceses.

4.3.1 Added-Potential

In this case, we deal with equation (11). Studying on this part, we have noticed that; after a wavy stablility period (graph 7), the soliton starts decreasing (shown in graph 8)

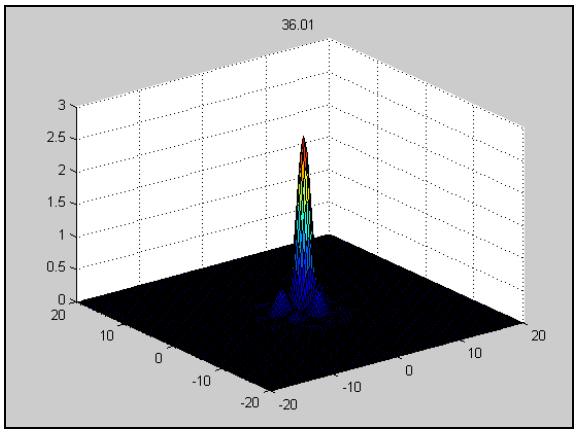


Graph 7: Wavy Stability

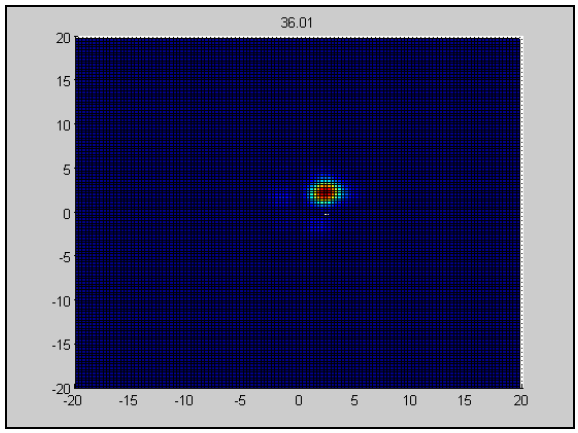


Graph 8: Decreasing Period

and some small solitons starts to occur near the original soliton (shown with an angle in graph 9 and from top in graph 10).



Graph 9: Small Solitons (Unstable)



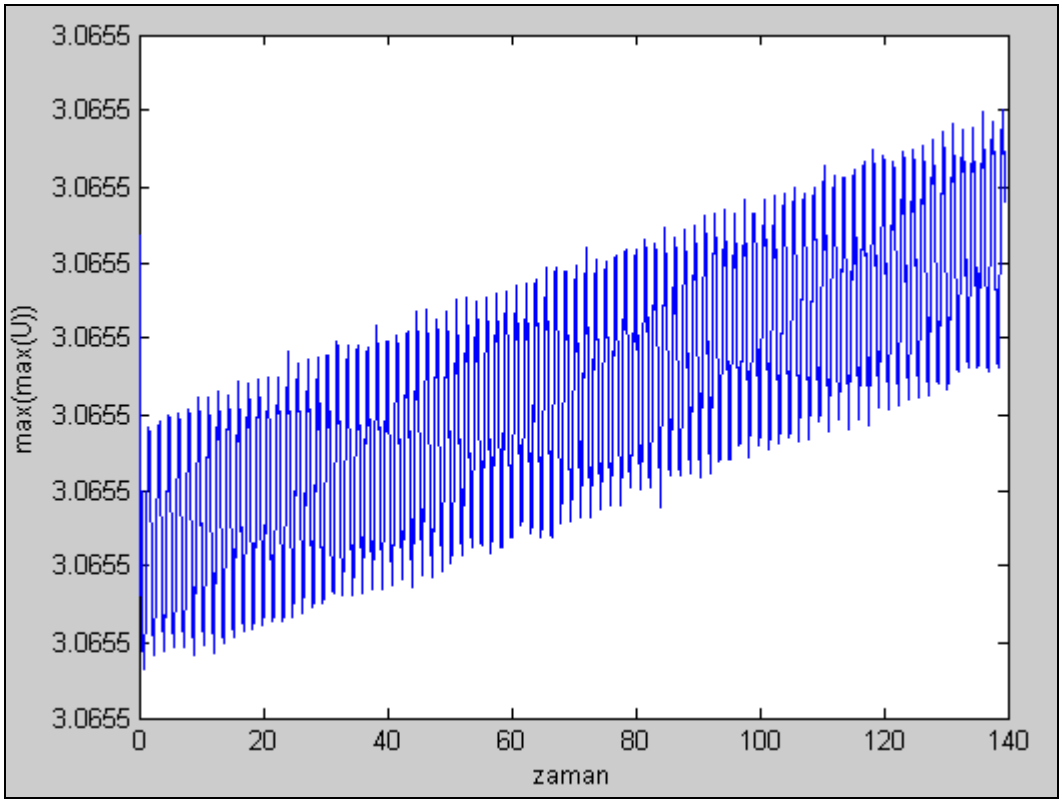
Graph 10: Top View - Unstability

4.3.1 Without Potential

In the second case, we have studied on

$$G(\mu,r)=\int\int \hat{w}\hat{w}^{*}\,dk-\int\int \frac{1}{k^{2}+r}\left((\mu+r)\hat{w}\hat{w}^{*}-\hat{w}^{*}F\left(\frac{E_{0}w}{1+\varepsilon\,|\,\lambda\,|^{2}|\,w\,|^{2}}\right)\right)dk$$

where $V(x,y)=0$, the potential is chosen to be zero. We have noticed that the solitons become more stable than the first case, and advancing with very small increases in a wavy form (graph 11).



Graph 11: Without Potential

5. Conclusion

We have analyzed Schrödinger's equation in three dimensional spaces by using Yang's paper [1] as a reference for our studies. In this analysis, we have studied renormalization methods, exponential time differencing for stiff systems, and programming spectral methods in matlab. During this analysis, we are dealing with spectral analysis, Fourier space and fast Fourier transform.

In the end, we have found a mode with renormalization method and use this mode as a initial condition for Schrödinger's equation to observe its long term stability with some perturbation. Therefore, we have confirmed the results in [1].

We have compared our results with a finite difference matlab program in order to sure about the results. As it predicted, we have seen that spectral methods are faster than other methods.

In our study we have observed both added-potential case and without potential case equations to analyze the effect of field intensity of the optical lattice. The results show that, without potential case, the system is more stable than the first case.

We do not study on penrose and vortex initial conditions, which have more than one solitons. We predict that these will give worse stability regions than one soliton give, from similar studies. They will be gratifying research areas for our next studies. In addition, we would study on eigenvalues, eigenmodes and eigenvectors of Schrödinger's equation in order to examine stability boundaries and finding the best initial constant values.

6. Bibliography

[1] J. Yang, "Stability of vortex solitons in a photorefractive optical lattice", New J. Phys. 6, 47 (2004).

[2] G. Fibich, Y. Sivan, M. I. Weinstein, "Bound states of nonlinear Schrödinger equations with a periodic nonlinear microstructure" Science Direct. 217, 31-57 (2006).

[3] Z. H. Musslimani, J. Yang, "Self-trapping of light in a two-dimensional photonic lattice," J. Opt. Soc. Am. B 21, 973-981 (2004).

[4] M. J. Ablowitz, Z. H. Musslimani, "Spectral renormalization method for computing self-localized solutions to nonlinear systems," Opt. Lett. 30, 2140-2142 (2005).

[5] S. M. Cox, P. C. Matthews, "Exponential Time Differencing for Stiff Systems", Journal of Computational Physics vol.176, iss.2, March 2002, pp.430-455.

[6] A. Kassam, L. N. Trefethen, "Fourth Order Time Stepping for Stiff PDEs", vol.26, iss.4, April 2005, pp.1214-1233.

[7] A. Kassam, "Solving reaction-diffusion equations 10 times faster", Oxford University, Numerical Analysis Group Research Report No. 16, 2003.

[8] L. N. Trefethen, Spectral Methods in Matlab, SIAM. 2000.

[9] L. N. Trefethen, "M-files – Spectral Methods in Matlab", 2000, http://www.comlab.ox.ac.uk/people/nick.trefethen/spectral.html

[10] W. A. Gardner, Statistical Spectral Analysis, Prentice Hall. 1988.

[11] J. S. Walker, Fast Fourier Transforms, CRC Press. 1996

[12] H. J. Weaver, Applications of Discrete and Continuos Fourier Analysis, Krieger Publishing Comp. 1992.

[13] R. V. Churchill, Fourier Series and Boundary Value Problems, McGraw Hill Book Comp. 1941.

[14] P. Bourke, "Discrete Fourier Transform – Fast Fourier Transform", 1993, http://local.wasp.uwa.edu.au/~pbourke/other/dft/

[15] Petviashvili V I 1976 Plasma Phys. 2 469

[16] MATLAB, “Index of MATLAB Documentation”, April 2001, http://www-eleves-isia.cma.fr/documentation/matlab/techdoc/ref/