Summer 2013

Generalized finite-difference time-domain schemes for solving nonlinear Schrödinger equations

Frederick Ira Moxley III

GENERALIZED FINITE-DIFFERENCE TIME-DOMAIN SCHEMES FOR SOLVING NONLINEAR SCHRÖDINGER EQUATIONS

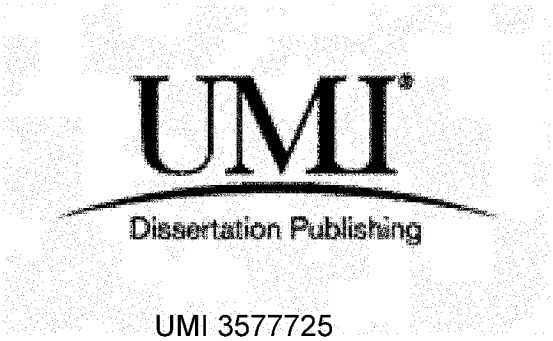
by

Frederick Ira Moxley III, B.A., M.S.

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

August 2013



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

LOUISIANA TECH UNIVERSITY

THE GRADUATE SCHOOL

June 12, 2013

Date

We hereby recommend that the dissertation prepared under our supervision by Frederick Ira Moxley III

entitled

Generalized Finite-Difference Time-Domain Schemes for Solving Nonlinear Schrödinger Equations

be accepted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Engineering (Physics)

Supervisor of Dissertation Research

Head of Department

Physics

Department

Recommendation concurred in:

Advisory Committee

Approved:

Director of Graduate Studies

Approved:

Dean of the Graduate School

Dean of the College

GS Form 13a
(6/07)

ABSTRACT

The nonlinear Schrödinger equation (NLSE) is one of the most widely applicable equations in physical science, and characterizes nonlinear dispersive waves, optics, water waves, and the dynamics of molecules. The NLSE satisfies many mathematical conservation laws. Moreover, due to the nonlinearity, the NLSE often requires a numerical solution, which also satisfies the conservation laws. Some of the more popular numerical methods for solving the NLSE include the finite difference, finite element, and spectral methods such as the pseudospectral, split-step with Fourier transform, and integrating factor coupled with a Fourier transform. With regard to the finite difference and finite element methods, higher-order accurate and stable schemes are often required to solve a large-scale linear system. Conversely, spectral methods via Fourier transforms for space discretization coupled with Runge-Kutta methods for time stepping become too complex when applied to multidimensional problems. One of the most prevalent challenges in developing these numerical schemes is that they satisfy the conservation laws.

The objective of this dissertation was to develop a higher-order accurate and simple finite difference scheme for solving the NLSE. First, the wave function was split into real and imaginary components and then substituted into the NLSE to obtain coupled equations. These components were then approximated using higher-order Taylor series expansions in time, where the derivatives in time were replaced by the derivatives in

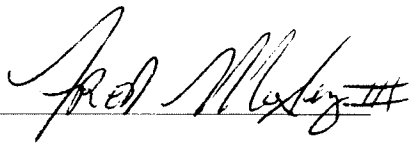
space via the coupled equations. Finally, the derivatives in space were approximated using higher-order accurate finite difference approximations. As such, an explicit and higher order accurate finite difference scheme for solving the NLSE was obtained. This scheme is called the explicit generalized finite-difference time-domain (explicit G-FDTD). For purposes of completeness, an implicit G-FDTD scheme for solving the NLSE was also developed.

In this dissertation, the discrete energy method is employed to prove that both the explicit and implicit G-FDTD scheme satisfy the discrete analogue form of the first conservation law. To verify the accuracy of the numerical solution and the applicability of the schemes, both schemes were tested by simulating bright and dark soliton propagation and collision in one and two dimensions. Compared with other popular existing methods (e.g., pseudospectral, split-step, integrating factor), numerical results showed that the G-FDTD method provides a more accurate solution, particularly when the time step is large. This solution is particularly important during the long-time period simulations. The explicit G-FDTD method proved to be advantageous in that it was simple and fast in computation. Furthermore, the G-FDTD showed that the solution propagates through the boundary with analytical solution continuation.

APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Dissertation. It is understood that "proper request" consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Dissertation. Further, any portions of the Dissertation used in books, papers, and other works must be appropriately referenced to this Dissertation.

Finally, the author of this Dissertation reserves the right to publish freely, in the literature, at any time, any or all portions of this Dissertation.

Author 

Date 8/1/13

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

Pursuing a Ph.D. degree is the ultimate goal of my educational career. Fortunately, this will finally become a reality.

I would like to express my sincere appreciation to my advisor, Dr. Weizhong Dai, for his constant encouragement and patient guidance throughout my years of study. He has helped me improve my research, and with writing and presentation skills. Without his help and support, this dissertation would not have been completed.

I am also very grateful for having an exceptional doctoral committee who have helped guide and direct me through this process. This includes Dr. Neven Simicevic. I also would like to thank Dr. Tim Byrnes who is an Assistant Professor at the National Institute of Informatics at Tokyo, Japan, Dr. David T. Chuss who is a NASA Astrophysicist, Dr. John C. Mather who is the 2006 Nobel Prize winner in Physics, and Dr. John P. Wefel who is a Professor of Physics and Director for the Louisiana Space Consortium at Louisiana State University, for being helpful and for their invaluable advice and guidance.

Most importantly, I would like to thank Jesus Christ. I owe thanks to my father, Dr. Frederick and my mother, Ines Moxley, for their faith in me and allowing me to pursue my dream. The love of everyone in my family is always the most powerful support for my life. This dissertation is dedicated to my daughter Vyera.

CHAPTER ONE

INTRODUCTION

1.1 General Overview

The nonlinear Schrödinger equation (NLSE) is one of the most widely used equations in physical science, and is used to characterize nonlinear dispersive waves, optics, water waves, and the dynamics of molecules. One example of how the NLSE is useful is how optical pulses propagate in fibers. These equations are also used to model telecommunications, hydrodynamics, nonlinear acoustics, and many other nonlinear systems [27]. Some of these other systems include nonlinear dispersive waves, plasmas, optics, water waves, and the dynamics of molecules [1, 47, 95, 100]. In terms of new and future applications, NLSE's will permit exploration of the superfluid effects of excition-polariton (or polariton) condensates. Polariton condensates are governed by a type of NLSE known as the Gross-Pitaevskii equation. Polariton condensates are quite promising because of the possibility of achieving room-temperature superfluidity, which opens the door for new technologies based on this system. Room-temperature superfluidity is an example of a promising new application for NLSE studies.

To solve the NLSE, the computational domain is typically isolated to a specific interval of interest that allows the non-zero behavior of the solution to be studied because solutions generally decay to zero as $|x| \to \infty$. These non-zero space discretized solutions

are achieved by various methods, but most commonly the finite difference [3, 24, 32, 44, 66, 85], spectral [11, 12, 85, 91] and the finite element methods [32] are used. Similarly, the discretization in time is usually solved via Crank-Nicolson [73], time splitting [10, 14, 105], semi-implicit, or fourth-order Runge-Kutta methods [2, 3, 52]. Analytically, the NLSE is typically solved using an inverse scattering transformation [102], or by using Lie group theory [39]. However, these analytical solutions are available only under certain particular initial conditions. In general, the NLSE requires a numerical solution. The two main types of numerical methods commonly employed for obtaining numerical solutions for the NLSE are the spectral and grid methods. These methods differ from each other in how they address the spatial derivative. The solution demonstrated by the spectral method will have a finite, linear combination of differentiable basis functions that satisfy boundary conditions. Conversely, finite element and finite difference (grid) methods approximate derivatives, using only a compact set of grid points [103]. Spectral and pseudospectral methods have been very popular choices for solving the NLSE where codes are obtained by the fast Fourier transform [35, 36, 58]. A split-step Fourier pseudospectral method was studied in [93]. Another spectral method with liberal stability restrictions is the integrating factor method [15, 61, 87]. Finite difference methods [17, 18, 20, 23, 28, 31, 32, 42, 50, 67, 85, 88, 89] are typically more flexible and easier to use than spectral methods, particularly for systems with complex boundary conditions [89]. The finite element method for the one dimensional NLSE was studied in [40]. An improved method is the quadrature discretization method [25, 59, 60, 75]. Eight numerical methods for the one-dimensional NLSE were compared in [24, 86]. Other popular numerical investigations of the NLSE include that of Karpman and

Krushkal [54], Yajima and Outi [96], Satsuma and Yajima [71], and Hardin and Tappert [46]. More recently, Lanczos' Tau method was investigated in [19], and the Adomian decomposition method with variation iteration was examined in [92]. A beneficially in-depth and detailed summary of numerical methods for solving the NLSE can be seen in Yang's book [97].

One of the drawbacks of the current methodology used for solving the NLSE is the complicated manner in which space discretization schemes are combined with time stepping methods to perform the numerical evolution. Accordingly, boundary conditions are often necessary to offset any numerical inaccuracy that occurs when the solution approaches the edges of the computational domain. Regular boundary conditions include the homogenous Dirichlet [3, 24], periodic [85, 93], homogenous Neumann [42, 85], or higher order transport boundary conditions [4, 5, 72, 104]. Typically, if one desires the avoidance of implementing boundary conditions, the bounded computational domain must be large enough to avoid inaccuracies in the numerical approximation. Often, the space discretization is performed via Fourier transform, whereas time stepping is handled with Runge-Kutta (RK4) methods. This period is where the trade-off between accuracy and the complexity of the time stepping calculation becomes evident. Therefore, it is desirable to achieve a simple, yet accurate, numerical answer to avoid the difficulties of having to integrate advanced time stepping and space discretization schemes. Optimizing the manner by which the numerical solution is obtained for the NLSE allows for a higher-order accurate solution in both space and time while avoiding the trade-off between accuracy and complexity. Additionally, numerically solving the NLSE can be computationally expensive, particularly in multiple space dimensions. A typical problem

that arises when trying to solve the NLSE is a large requirement on computational resources, especially when the equation is in more than one space dimension. To offset this expense, it is desirable to lower the computational time by decreasing the number of grid points used to approximate the derivatives. Unfortunately, decreasing the number of grid points used in finite difference methods is not always feasible due to the analytical requirements on the ratio of grid steps. When these requirements are dismissed, an inaccurate, undesirable solution will occur. Generally, finite difference methods that employ an explicit computation are faster, however, this usually comes at the expense of offering a less precise solution [56]. Also, explicit computations allow for prototyping and testing of simulation programs due to their speed. When precision of computation becomes a concern, implicit methods are usually implemented to overcome these limitations.

The NLSE admits solutions that are commonly known as solitons, or self-reinforcing waves that maintain their shape and velocity during propagation. There are various types of solitons that form when the nonlinear term of the NLSE cancels with the dispersion terms. Specifically, we will study the two primary forms of solitons in this dissertation. One of the primary forms of solitons are bright solitons, formed when the nonlinear term of the NLSE is negative. This negative signifies a focused or attractive nonlinearity.

The objective of this dissertation study is to numerically solve the NLSE for soliton solutions in a fast, simple, and easy design for situations when the exact solution is known or unknown. The dark solitons of convex shaped density found in defocusing media will be simulated [8]. This computation will be performed in single and multiple

dimensions, without the necessity of implementing absorbing boundary conditions, due to analytical solution continuation. In this dissertation, we present both an explicit and implicit generalized finite-difference time-domain (G-FDTD) method for solving the time-dependent NLSE in single and multiple space dimensions. In this method, the function $\psi(x,t)$ is first split into real and imaginary components, resulting in two coupled equations. The real and imaginary components are then approximated using higher-order Taylor series expansions in time. Then the derivatives in time are substituted into the derivatives in space via the coupled equations. Finally, the derivatives in space are approximated using higher-order finite difference methods. The obtained scheme will be shown to satisfy the discrete analogous form of the conservation law, Eq. (2.3). The new scheme is then tested by various examples of soliton propagation and collision. In addition, comparisons are made between the popular spectral methods for convenience. The temporal accuracy of the spectral methods, which are the focus of this study are limited to fourth-order accuracy. This limitation is due to the RK4 and split-step time stepping schemes the spectral methods obey. The G-FDTD method will be adapted to suit a higher order accuracy that out-performs the spectral methods. The limitation of temporal accuracy given by spectral methods leads to the small space grid and time step that can be employed. As a result, the number of the iterations needed for simulation may be considerably large for a certain time period, which reduces the computational efficiency. Larger time steps used for soliton simulations will reduce the running time of the associated computer program, which is an objective to be achieved with G-FDTD.

1.2 Organization of the Dissertation

Chapter 1 proposes the research objective of this dissertation based on a general literature review.

Chapter 2 reviews the background and previous works related to this research. First, the conservation laws for the NLSE are discussed. Then, the basic formulations and stability conditions of the conventional spectral methods (pseudospectral, split-step, integrating-factor) are described, respectively. Finally, the G-FDTD algorithm for the linear Schrödinger equation, stability, and absorbing boundary conditions are described at the end of the chapter.

Chapter 3 discusses the explicit scheme for solving the nonlinear Schrödinger equation. The discrete energy method is used to prove the discrete analogue of the conservation law in detail. Numerical examples for testing the explicit scheme are described. Next, these numerical examples are shown to satisfy the explicit scheme for the NLSE. Numerical accuracy is then discussed based on the numerical results obtained by both the G-FDTD method and the various spectral methods.

Chapter 4 examines the implicit scheme for solving the nonlinear Schrödinger equation. The discrete energy method is used to prove the discrete analogue of the conservation law in detail. Numerical examples for testing the explicit scheme are described. Next, the numerical examples are shown to satisfy the implicit scheme for the NLSE. The numerical accuracy is discussed based on the numerical results obtained by both the G-FDTD method and the various spectral methods.

Chapter 5 presents the conclusion of the research and provides some suggestions for future work.

CHAPTER TWO

BACKGROUND AND PREVIOUS WORK

2.1 Nonlinear Schrödinger Equation

The nonlinear Schrödinger equation (NLSE) is one of the most widely applicable equations in physical science, and is used to characterize nonlinear dispersive waves, plasmas, optics, water waves, and the dynamics of molecules [1, 47, 95, 100]. The NLSE can be expressed as

$$i\frac{\partial\psi(x,t)}{\partial t}-\frac{\partial^2\psi(x,t)}{\partial x^2}+\lambda|\psi(x,t)|^{p-1}\psi(x,t)=0 \tag{2.1}$$

where $\psi(x,t)$ is a complex valued function that governs the evolution of a weakly nonlinear, strongly dispersive, almost monochromatic wave [17]. The integer $p \geq 3$ determines the nature of the nonlinear term, and $i=\sqrt{-1}$. In addition, λ is a positive or negative constant corresponding to the focusing, or defocusing, NLSE, respectively [9]. The NLSE permits single and multiple hyperbolic secant solutions known as bright solitons, or hyperbolic tangent solutions known as dark solitons [76]. The behavior of solutions depends considerably on the sign of λ, the parameter p, and the spatial dimension n. For $\lambda<0$, and $n=1$ there exists a four-parameter family of solitary wave solutions [32]

$$\psi(x,t) = f(x-ct)e^{ig(x-bt)} \tag{2.2}$$

where $f^{p-1}(x) = -\frac{(p+1)\alpha}{2\lambda}\text{sech}^2[\frac{p-1}{2}\sqrt{\alpha}(x-x_c)]$, and $g(x) = \frac{c}{2}x + \phi$, $\alpha = \frac{c}{2}[\frac{c}{2} - b] > 0$. Specifically, the case of $n = 1$, $p = 3$, and $\lambda < 0$ has been studied in detail [102]. Solitary wave solutions of Eq. (2.2) can be written as $\psi(x,t) = 2\eta f e^{ig}$, where $f = \text{sech}(2\eta(x-x_c) + 8\eta\xi t)$, and $g = -2\xi x - 4(\xi^2 - \eta^2)t + \phi$. Here, η, ξ, ϕ, and x_c are parameters. Generally, Eq. (2.1) becomes more difficult to solve as $p \to \infty$, because there is a spreading of active wave numbers in Fourier space. This is related to the spatial gradients of physical space [48, 69].

Multiplying Eq. (2.1) by the conjugate function, $\bar{\psi}(x,t)$, and taking the imaginary part, then multiplying Eq. (2.1) by $\frac{\partial\bar{\psi}(x,t)}{\partial t}$ and taking the real part, one may see that the NLSE satisfies two conservation laws,

$$N(\psi) = \int_{-\infty}^{\infty} |\psi(x,t)|^2\, dx = constant, \tag{2.3}$$

and

$$E(\psi) = \int_{-\infty}^{\infty} \left\{\frac{1}{2}|\nabla\psi(x,t)|^2 + \frac{\lambda}{p+1}|\psi(x,t)|^{p+1}\right\} dx = constant. \tag{2.4}$$

Indeed, if we multiply Eq. (2.1) by the conjugate function $\bar{\psi}(x,t)$, and integrate it over $(-\infty,\infty)$, we obtain

$$\int_{-\infty}^{\infty} i\frac{\partial\psi}{\partial t}\bar{\psi}dx - \int_{-\infty}^{\infty} \frac{\partial^2\psi}{\partial x^2}\bar{\psi}dx + \lambda\int_{-\infty}^{\infty} |\psi|^{p-1}\psi\bar{\psi}\, dx = 0. \tag{2.5}$$

Using integration by parts, we have

$$\int_{-\infty}^{\infty} \frac{\partial^2 \psi}{\partial x^2} \bar{\psi} dx = \frac{\partial \psi}{\partial t} \bar{\psi}|_{-\infty}^{\infty} - \int_{-\infty}^{\infty} \frac{\partial \psi}{\partial x} \frac{\partial \bar{\psi}}{\partial x} dx = - \int_{-\infty}^{\infty} \left| \frac{\partial \psi}{\partial x} \right|^2 dx. \tag{2.6}$$

Here, we assume $\psi(x,t)$ is zero at $\pm\infty$. Furthermore, taking the real part

$$Re \int_{-\infty}^{\infty} \frac{\partial \psi}{\partial t} \bar{\psi} dx = \int_{-\infty}^{\infty} \frac{1}{2} \frac{\partial |\psi|^2}{\partial t} dx = \frac{1}{2} \frac{d}{dt} \int_{-\infty}^{\infty} |\psi|^2 dx. \tag{2.7}$$

and

$$\int_{-\infty}^{\infty} |\psi|^{p-1} \psi \bar{\psi} dx = \int_{-\infty}^{\infty} |\psi|^{p+2} dx. \tag{2.8}$$

Substituting Eqn.'s (2.7) and (2.8) into Eq. (2.5) and chosing the imaginary part gives $\frac{d}{dt} \int_{-\infty}^{\infty} |\psi|^2 dx = 0$, meaning Eq. (2.3) is satisfied. On the other hand, we multiply the NLSE by $\partial_t \bar{\psi}$ and integrate the resulting equation over $(-\infty, \infty)$. This gives

$$i \int_{-\infty}^{\infty} \frac{\partial \psi}{\partial t} \frac{\partial \bar{\psi}}{\partial t} dx - \int_{-\infty}^{\infty} \frac{\partial^2 \psi}{\partial x^2} \frac{\partial \bar{\psi}}{\partial t} dx + \lambda \int_{-\infty}^{\infty} |\psi|^{p-1} \psi \frac{\partial \bar{\psi}}{\partial t} dx = 0. \tag{2.9}$$

Using integration by parts, we obtain

$$\begin{aligned} Re \int_{-\infty}^{\infty} \frac{\partial^2 \psi}{\partial x^2} \frac{\partial \bar{\psi}}{\partial t} dx &= Re \frac{\partial \psi}{\partial x} \frac{\partial \bar{\psi}}{\partial t}|_{-\infty}^{\infty} - Re \int_{-\infty}^{\infty} \frac{\partial \psi}{\partial x} \frac{\partial^2 \bar{\psi}}{\partial t \partial x} dx \\ &= - \int_{-\infty}^{\infty} \frac{1}{2} \frac{\partial}{\partial t} \left(\frac{\partial \psi}{\partial x} \frac{\partial \bar{\psi}}{\partial x} \right) dx = - \frac{1}{2} \frac{d}{dt} \int_{-\infty}^{\infty} \left| \frac{\partial \psi}{\partial x} \right|^2 dx. \end{aligned} \tag{2.10}$$

Furthermore, taking the real part we find

$$Re \int_{-\infty}^{\infty} |\psi|^{p-1}\psi \frac{\partial \bar{\psi}}{\partial t} dx = \frac{1}{2} \int_{-\infty}^{\infty} |\psi|^{p-1} \frac{\partial |\psi|^2}{\partial t} dx$$

$$= \frac{1}{p+1} \int_{-\infty}^{\infty} \frac{\partial}{\partial t} |\psi|^{p+2} dx = \frac{d}{dt} \frac{1}{p+1} \int_{-\infty}^{\infty} |\psi|^{p+2} dx \tag{2.11}$$

Substituting them into Eq. (2.8) and choosing the real part gives

$$\frac{d}{dt}\left[\int_{-\infty}^{\infty} \frac{1}{2} \left|\frac{\partial \psi}{\partial x}\right|^2 dx + \frac{\lambda}{p+1} \int_{-\infty}^{\infty} |\psi|^{p+2}\, dx \right] = 0, \tag{2.12}$$

which verifies the second conservation law, Eq. (2.4). It should be pointed out that if we continue to integrate over higher order derivatives such as $\partial_{tt}\psi$, $\partial_{ttt}\psi$, etc., many more conservation laws can be obtained.

2.2 Spectral Methods for Solving the Nonlinear Schrödinger Equation

Due to the nonlinear term of the NLSE, a numerical solution is often required. Some of the more popular numerical methods for solving the NLSE include the finite difference, finite element, and spectral methods such as the pseudospectral, split-step with Fourier transform, and integrating factor coupled with a Fourier transform. Spectral methods via Fourier transforms for space discretization coupled with Runge-Kutta methods for time stepping is now described in detail.

2.2.1 Pseudospectral Method

The development of the pseudospectral (Fourier) method [36, 37, 41, 68, 87] occurred as the spectral methods for solving wave equations first became popular. To describe this method requires that the wave function be transformed into Fourier space with respect to x. After doing so, the transformed variable will contain algebraic derivatives and operators. Upon this transformation, it is easy to confirm that the result

will maintain spectral accuracy (i.e., the error will decrease exponentially as $1/\Delta x$). To obtain this result, one must calculate the error as the infinity norm of the difference between the numerical solution and the exact solution.

To further specify how the pseudospectral method is derived for application with the NLSE, the first step is to discretize Eq. (2.1) in space like

$$\frac{\partial \psi^n}{\partial t} = i\left(-\frac{\partial^2 \psi^n}{\partial x^2} + \lambda|\psi^n|^{p-1}\psi^n\right), \tag{2.13}$$

where ψ^n is the solution at the grid point x_n. It is important to note that the pseudospectral method uses a discrete Fourier transform to evaluate the Laplacian of the wave function. Next, a time-stepping scheme (i.e., Runge-Kutta, Adams, Leapfrog) is used to temporally evolve the wave function. As previously mentioned, we need to solve $\frac{\partial^2 \psi^n}{\partial x^2}$ by the discrete Fourier transform (DFT)

$$\frac{\partial^2 \psi^n}{\partial x^2} = \mathfrak{F}^{-1}[(ik)^2 \mathfrak{F}(\psi^n)], \tag{2.14}$$

where $\mathfrak{F}$ is the DFT, $\mathfrak{F}^{-1}$ is the inverse DFT, and the Fourier wavenumber is k. The accuracy of this computation for a sufficiently smooth wave function is spectral, meaning that the error is smaller than any power of Δx [15, 87]. To describe why spectral accuracy exists here, one must realize that the discretized wave function has a finite spectral bandwidth that rests in between $-\pi/\Delta x$ and $\pi/\Delta x$. Conversely, the continuous wave function has an infinite spectral bandwidth. Therefore, the error that arises from discretization will be less than any power of Δx, making Eq. (2.14) spectrally accurate. After Eq. (2.14) is solved, Eq. (2.13) can be advanced using a time-stepping scheme. In our comparison of results, RK4 is used for time-stepping the pseudospectral method, and therefore admits a fourth-order temporal accuracy.

One of the detriments of the pseudospectral method is the numerical stability restriction on the time step Δt. In order to ensure stability for any numerical method, eigenvalues of the spatial discretization operator multiplied by Δt must be within the stability region of the time stepping scheme being used [87]. For a small Δx, the eigenvalues for Eq. (2.13) must be at most $\pm i\pi^2/\Delta x^2$, caused by both $i\frac{\partial^2}{\partial x^2}$ in Eq. (2.1) and the maximum wavenumbers ($\pm\pi/\Delta x$) permitted on the discretized space grid. The maximum eigenvalues determine the permissible time step values. Fortunately, the RK4 stability is well understood. On the imaginary axis, the boundary points required for stability are $\pm 2i\sqrt{2}$. As a result of this criterion, the pseudospectral method requires that

$$\frac{\Delta t}{\Delta x^2} \leq \frac{2\sqrt{2}}{\pi^2}[time/(space)^2]. \tag{2.15}$$

Moreover, stability requirements are usually determined by the highest Fourier mode on the spatial grid. Substituting the time evolution of the highest Fourier mode $c(t)e^{i\pi x/\Delta x}$ into Eq. (2.1) shows that $c'(t) = i(\pi^2/\Delta x^2)c(t)$ when Δx is sufficiently small. Thus, in order for RK4 to remain stable in Eq. (2.1), Eq. (2.15) must be satisfied. The pseudospectral method is beneficial because it allows a highly accurate and easy solution for use with all wave equations in any dimension. Unfortunately the stability criterion sometimes requires that a small time step be taken.

2.2.2 Split-Step Method

The split-step method has a variety of uses [78, 98]. Some of the more recent developments are found in [99], and physics implementations are contained in [1, 15]. Operating with the assumption that Fourier transforms are used for spatial derivatives, it is obvious that the spatial accuracy of the split-step method is spectral. Fortunately, the

split-step method conserves energy, as does the evolution equation it simulates. Energy conservation is one of the main advantages of the split-step method -- it maintains conserved quantities like energy because each split-step operator is power invariant.

We consider the split-step method for general nonlinear wave equations

$$\frac{\partial \vec{\psi}}{\partial t} = M(\vec{\psi}) + N(\vec{\psi}), \tag{2.16}$$

where either, or both, $M(\vec{\psi})$ and $N(\vec{\psi})$ are nonlinear in $\vec{\psi}$ and do not explicitly depend on time. Now, the split equations to be solved are

$$\frac{\partial \vec{\mathbf{v}}}{\partial t} = M(\vec{\mathbf{v}}), \tag{2.17a}$$

and

$$\frac{\partial \vec{\mathbf{v}}}{\partial t} = N(\vec{\mathbf{v}}). \tag{2.17b}$$

Considering the Strang splitting scheme $S_2(h)$, we start from the initial condition $\vec{\psi}(\mathbf{x}, 0)$ and integrate Eq. (2.17a) by half a step $h/2$ to result in $\vec{\mathbf{v}}_1(\mathbf{x})$. From $\vec{\mathbf{v}}_1(\mathbf{x})$, Eq. (2.17b) is integrated by a whole step h to result in $\vec{\mathbf{v}}_2(\mathbf{x})$. From $\vec{\mathbf{v}}_2(\mathbf{x})$, we again integrate by half a step $h/2$ to result in $S_2(h)\vec{\psi}(\mathbf{x}, 0)$ for the nonlinear solution $\vec{\psi}(\mathbf{x}, h)$. Applying splitting schemes such as $S_4(h)$, that have fourth order accuracy in time, to Eq. (2.16) is similar.

Applying the split-step scheme S_4 to Eq. (2.1) requires that $M(\psi) = -i\frac{\partial^2 \psi}{\partial x^2}$ and $N(\psi) = i\lambda|\psi|^{p-1}\psi$. Then, Eq. (2.17a) can be solved by a DFT, and Eq. (2.17b) will be given by the exact solution for $\psi(\mathrm{x}, t)$. The spatial accuracy of the S_4 split-step method is spectral due to the DFT evaluation of $\frac{\partial^2}{\partial x^2}$. The temporal accuracy is fourth order. For linear evolution equations, the temporal accuracy of the split-step method is known. However, this is not the case for nonlinear equations. The split-step method has a

numerical stability restriction on the time step Δt for nonlinear equations. According to [93], the numerical stability for S_4 requires that

$$\frac{\Delta t}{\Delta x^2} \lesssim \frac{1}{\pi}[time/(space)^2]. \tag{2.18}$$

Fortunately, the stability constraint of the split-step method is less restrictive than the pseudospectral constraint. Higher-order split-step methods are complicated and not necessarily the best choice. Larger time steps may often be used without numerical instability; however, this does not mean the split-step method is unconditionally stable. The numerical instability of the split-step method is weak compared to that of the pseudospectral method.

2.2.3 Integrating-Factor Method

Another spectral method that has an added benefit of liberal stability restriction is the integrating-factor method [15, 61, 87]. To describe this method, it is assumed that the linear differential of the wave equation in Fourier space admits basic analytical solutions. Next, these analytical solutions can be used as integrating factors to solve the transformed equation involving nonlinear terms. The linear terms are then treated analytically and, consequently, the remaining nonlinear equation is less stiff. This allows the use of large time steps without the onset of numerical instability. The Fourier transform of Eq. (2.1) is

$$\frac{\partial \hat{\psi}}{\partial t} - ik^2\hat{\psi} - i\lambda\mathfrak{F}(|\psi|^{p-1}\psi) = 0, \tag{2.19}$$

where $\hat{\psi} = \mathfrak{F}(\psi)$, $\mathfrak{F}$ is the Fourier transform, and the Fourier wavenumber is k. In general, for nonlinear wave equations in arbitrary dimensions,

$$\frac{\partial \vec{\psi}}{\partial t} = L\left(\frac{\partial}{\partial x}\right)\vec{\psi} + N\left(\vec{\psi}, \frac{\partial}{\partial x}\right). \tag{2.20}$$

In Eq. (2.20), $\vec{\psi}(x,t)$ is a vector function, and L is a linear differential operator with constant coefficients. In addition, N is a nonlinear function of $\vec{\psi}$ and the corresponding space derivatives. Following the integrating factor method

$$\frac{\partial \hat{\mathbf{v}}}{\partial t} = e^{-L(ik)t}\mathfrak{F}[N(\vec{\psi}, \frac{\partial}{\partial x})], \tag{2.21}$$

where $\hat{\mathbf{v}} = e^{-L(ik)t}\mathfrak{F}(\vec{\psi})$, $\mathfrak{F}$ is the Fourier transform, and the Fourier wavenumber is k. The RK4 method is used for time-stepping, although the time-stepping scheme used is a matter of preference. Once again, with the assumption that spatial differentiation is performed by Fourier transform, spatial accuracy is spectral. Like the split-step method, the integrating-factor method has a restriction on the time step sizes. The numerical instability of the integrating-factor method is identical to the split-step stability given by Eq. (2.18), and is much weaker than the pseudospectral method [97].

2.3 G-FDTD Method for Solving the Linear Schrödinger Equation

Recently the G-FDTD method for solving a time dependent linear Schrödinger equation was developed in [63, 65]. The 1D time-dependent linear Schrödinger equation, which is the basis of quantum mechanics [16, 43], can be expressed as follows [80, 90]:

$$i\hbar\frac{\partial \psi(x,t)}{\partial t} = -\frac{\hbar^2}{2m}\frac{\partial^2 \psi(x,t)}{\partial x^2} + V(x)\psi(x,t), \tag{2.22}$$

where m is the mass of the particle, $\hbar$ is Planck's constant, V is the potential, $\psi(x,t)$ is a complex number, and $i = \sqrt{-1}$. The product of $\psi(x,t)$ with its complex conjugate, $\psi(x,t)\cdot\bar{\psi}(x,t)$ indicates the probability of a particle being at spatial location x at time t. It can be seen that the classic explicit two-level in time finite difference scheme, *i.e.*,

$$\frac{\psi^{n+1}(j)-\psi^{n}(j)}{\Delta t}=i\frac{\hbar}{2m\Delta x^{2}}\delta_{x}^{2}\psi^{n}(j)-i\frac{V}{\hbar}\psi^{n}(j), \tag{2.23}$$

is unconditionally unstable, where $\psi^{n}(j)$ is the approximation of $\psi(j\Delta x, n\Delta t)$. Here, Δx and Δt are the spatial grid size and time step, respectively, $j \in Z$ that denotes the set of all positive and negative integers, and δ_{x}^{2} is a second-order central difference operator such that

$$\delta_{x}^{2}\psi^{n}(j)=\psi^{n}(j+1)-2\psi^{n}(j)+\psi^{n}(j-1). \tag{2.24}$$

There are many numerical schemes developed for solving linear Schrödinger equations [3, 6, 7, 10, 16, 21, 22, 26, 29, 30, 33, 34, 38, 43, 44, 49, 51, 53, 55, 57, 62, 70, 72, 74, 77, 79, 80, 81, 82, 83, 84, 85, 90]. In particular, Sullivan [80] and Visscher [90] applied the finite-difference time-domain (FDTD) method, which is often employed in simulations of electromagnetic fields, to solve the above Schrödinger equation. The application of the FDTD technique for the analysis of quantum devices is often called the FDTD-Q scheme, which can be described as follows [80].

2.3.1 Generalized FDTD Scheme

The variable $\psi(x,t)$ is first split into its real and imaginary components in order to avoid the use of complex numbers:

$$\psi(x,t)=\psi_{real}(x,t)+i\psi_{imag}(x,t). \tag{2.25}$$

Inserting Eq. (2.25) into Eq. (2.22) and then separating the real and imaginary parts result in the following coupled set of equations:

$$\frac{\partial\psi_{real}(x,t)}{\partial t}=-\frac{\hbar}{2m}\frac{\partial^{2}\psi_{imag}(x,t)}{\partial x^{2}}+\frac{V(x)}{\hbar}\psi_{imag}(x,t), \tag{2.26}$$

and

$$\frac{\partial \psi_{imag}(x,t)}{\partial t} = \frac{\hbar}{2m}\frac{\partial^2 \psi_{real}(x,t)}{\partial x^2} - \frac{V(x)}{\hbar}\psi_{real}(x,t). \tag{2.27}$$

Thus, the second-order central finite difference approximations in space and time result in the FDTD-Q schemes as follows:

$$\frac{\psi_{real}^{n}(j) - \psi_{real}^{n-1}(j)}{\Delta t} = -\frac{\hbar}{2m(\Delta x)^2}\delta_x^2 \psi_{imag}^{n-\frac{1}{2}}(j) + \frac{V(j)}{\hbar}\psi_{imag}^{n-\frac{1}{2}}(j), \tag{2.28}$$

and

$$\frac{\psi_{imag}^{n+1/2}(j) - \psi_{imag}^{n-1/2}(j)}{\Delta t} = \frac{\hbar}{2m(\Delta x)^2}\delta_x^2 \psi_{real}^{n}(j) - \frac{V(j)}{\hbar}\psi_{real}^{n}(j). \tag{2.29}$$

Here, we assume that V is dependent only on x for simplicity. The computation of the above FDTD-Q scheme is very simple and straight-forward because one may obtain $\psi_{real}^{n}(j)$ from Eq. (2.28) and $\psi_{imag}^{n+1/2}(j)$ from Equation (2.29). Previously, the stability of the FDTD-Q scheme was analyzed using the discrete energy method and a condition for determining the time step, Δt, was found so that the scheme is stable as follows [30]:

$$\frac{\hbar}{m}\cdot\frac{\Delta t}{\Delta x^2} + \frac{\Delta t}{2\hbar}\max|V| \le c < 1, \tag{2.30}$$

where c is a constant. It should be pointed out that Soriano *et al.* [77] and Visscher [90] also used the eigenvalue method to analyze the stability of the FDTD-Q scheme and obtained a very similar condition of $\frac{\hbar}{m}\cdot\frac{\Delta t}{\Delta x^2} + \frac{\Delta t}{2\hbar}\max|V| \le 1$. However, as pointed out in [30], even if the condition $\frac{\hbar}{m}\cdot\frac{\Delta t}{\Delta x^2} + \frac{\Delta t}{2\hbar}\max|V| = 1$ is chosen, the numerical solution is still divergent. Eq. (2.30) indicates that the condition must be less than, but not equal to, 1. Eq.'s (2.26) and (2.27) are then rewritten as

$$\frac{\partial \psi_{real}(x,t)}{\partial t} = -\left(\frac{\hbar}{2m}A + \frac{V}{\hbar}\right)\psi_{imag}(x,t), \tag{2.31}$$

$$\frac{\partial \psi_{imag}(x,t)}{\partial t} = \left(\frac{\hbar}{2m}A - \frac{V}{\hbar}\right)\psi_{real}(x,t), \tag{2.32}$$

where $A = \frac{\partial^2}{\partial x^2}$. The Taylor series method is employed to expand $\psi_{real}(x,t_n)$and $\psi_{real}(x,t_{n-1})$ at $t = t_{n-1/2} = (n-\frac{1}{2})\Delta t$ as follows:

$$\psi_{real}(x,t_n) = \psi_{real}(x,t_{n-1}) + 2\sum_{p=0}^{\infty}\left(\frac{\Delta t}{2}\right)^{2p+1}\frac{1}{(2p+1)!}\frac{\partial^{2p+1}\psi_{real}(x,t_{n-1/2})}{\partial t^{2p+1}}. \tag{2.33}$$

The derivatives in Eq. (2.33) are evaluated by using Eq.'s (2.31) and (2.32) repeatedly:

$$\frac{\partial \psi_{real}(x,t_{n-1/2})}{\partial t} = -\left(\frac{\hbar}{2m}A - \frac{V}{\hbar}\right)\psi_{imag}(x,t_{n-1/2}), \tag{2.34a}$$

$$\frac{\partial^3 \psi_{real}(x,t_{n-1/2})}{\partial t^3} = -\left(\frac{\hbar}{2m}A - \frac{V}{\hbar}\right)\frac{\partial^2 \psi_{imag}(x,t_{n-1/2})}{\partial t^2}$$

$$= -\left(\frac{\hbar}{2m}A - \frac{V}{\hbar}\right)\left(\frac{\hbar}{2m}A - \frac{V}{\hbar}\right)\frac{\partial \psi_{real}(x,t_{n-1/2})}{\partial t}$$

$$= -\left(\frac{\hbar}{2m}A - \frac{V}{\hbar}\right)^3\psi_{imag}(x,t_{n-1/2}), \tag{2.34b}$$

$$\frac{\partial^5 \psi_{real}(x,t_{n-1/2})}{\partial t^5} = \left(\frac{\hbar}{2m}A - \frac{V}{\hbar}\right)^3\frac{\partial^2 \psi_{imag}(x,t_{n-1/2})}{\partial t^2}$$

$$= \left(\frac{\hbar}{2m}A - \frac{V}{\hbar}\right)^3\left(\frac{\hbar}{2m}A - \frac{V}{\hbar}\right)\frac{\partial \psi_{real}(x,t_{n-1/2})}{\partial t}$$

$$= -\left(\frac{\hbar}{2m}A - \frac{V}{\hbar}\right)^5\psi_{imag}(x,t_{n-1/2}), \tag{2.34c}$$

and so on. Substituting Eq. (2.34) into Eq. (2.33) gives

$$\psi_{real}(x,t_n) = \psi_{real}(x,t_{n-1}) + 2\sum_{p=0}^{N}\left(\frac{\Delta t}{2}\right)^{2p+1}\frac{(-1)^{p+1}}{(2p+1)!}\left(\frac{\hbar}{2m}A - \frac{V}{\hbar}\right)^{2p+1}$$

$$\cdot \psi_{imag}\left(x, t_{n-\frac{1}{2}}\right) + O(\Delta t)^{2N+3}. \tag{2.35}$$

Similarly, the Taylor series method is used to expand $\psi_{imag}\left(x, t_{n+\frac{1}{2}}\right)$and $\psi_{imag}\left(x, t_{n-\frac{1}{2}}\right)$ at $t = t_n = n\Delta t$ as follows:

$$\psi_{imag}(x, t_{n+1/2}) = \psi_{imag}(x, t_{n-1/2}) + 2\sum_{p=0}^{\infty} (\frac{\Delta t}{2})^{2p+1} \frac{1}{(2p+1)!} \frac{\partial^{2p+1}\psi_{imag}(x, t_n)}{\partial t^{2p+1}}. \tag{2.36}$$

Again, using Eq.'s (2.31) and (2.32) repeatedly to evaluate those derivatives in Eq. (2.36), we obtain

$$\frac{\partial \psi_{imag}(x, t_n)}{\partial t} = \left(\frac{\hbar}{2m} A - \frac{V}{\hbar}\right) \psi_{real}(x, t_n), \tag{2.37a}$$

$$\frac{\partial^3 \psi_{imag}(x, t_n)}{\partial t^3} = (\frac{\hbar}{2m} A - \frac{V}{\hbar}) \frac{\partial^2 \psi_{real}(x, t_n)}{\partial t^2}$$

$$= -(\frac{\hbar}{2m} A - \frac{V}{\hbar})(\frac{\hbar}{2m} A - \frac{V}{\hbar}) \frac{\partial \psi_{imag}(x, t_n)}{\partial t}$$

$$= -(\frac{\hbar}{2m} A - \frac{V}{\hbar})^3 \psi_{real}(x, t_n), \tag{2.37b}$$

$$\frac{\partial^5 \psi_{imag}(x, t_n)}{\partial t^5} = -(\frac{\hbar}{2m} A - \frac{V}{\hbar})^3 \frac{\partial^2 \psi_{real}(x, t_n)}{\partial t^2}$$

$$= (\frac{\hbar}{2m} A - \frac{V}{\hbar})^3 (\frac{\hbar}{2m} A - \frac{V}{\hbar}) \frac{\partial \psi_{imag}(x, t_n)}{\partial t}$$

$$= (\frac{\hbar}{2m} A - \frac{V}{\hbar})^5 \psi_{real}(x, t_n), \tag{2.37c}$$

and so on. Substituting Eq. (2.37) into Eq. (2.36) gives

$$\psi_{imag}(x, t_{n+1/2}) = \psi_{imag}(x, t_{n-1/2})$$

$$+2\sum_{p=0}^{N}\left(\frac{\Delta t}{2}\right)^{2p+1}\frac{(-1)^p}{(2p+1)!}\left(\frac{\hbar}{2m}A-\frac{V}{\hbar}\right)^{2p+1}$$

$$\cdot\psi_{real}(x,t_n)+O(\Delta t)^{2N+3}. \tag{2.38}$$

Thus, if $A\psi_{imag}(j\Delta x,t_{n+1/2})$ and $A\psi_{real}(j\Delta x,t_n)$ are approximated using some accurate finite differences, a generalized FDTD scheme for solving the time-dependent linear Schrödinger equation is described as follows:

$$\psi_{real}^{n}(j)=\psi_{real}^{n-1}(j)$$

$$+2\sum_{p=0}^{N}\left(\frac{\Delta t}{2}\right)^{2p+1}\frac{(-1)^{p+1}}{(2p+1)!}\left(\frac{\hbar}{2m}A-\frac{V}{\hbar}\right)^{2p+1}\cdot\psi_{imag}^{n-\frac{1}{2}}(j), \tag{2.39a}$$

$$\psi_{imag}^{n+1/2}(j)=\psi_{imag}^{n-1/2}(j)$$

$$+2\sum_{p=0}^{N}\left(\frac{\Delta t}{2}\right)^{2p+1}\frac{(-1)^{p}}{(2p+1)!}\left(\frac{\hbar}{2m}A-\frac{V}{\hbar}\right)^{2p+1}\cdot\psi_{real}^{n}(j). \tag{2.39b}$$

It should be pointed out that in Eq. (2.39a), $\psi_{imag}(x,t_{n+1/2})$ may be approximated by a higher-order accurate Lagrange polynomial or some other higher-order accurate approximations. Once $\psi_{real}^{n}(j)$ is obtained from Eq. (2.39a), one may construct a similar higher-order accurate Lagrange polynomial or some other higher-order accurate approximations for $\psi_{real}(x,t_n)$ and then substitute it into Eq. (2.39b) to obtain $\psi_{imag}^{n+1/2}(j)$. Here, for simplicity, finite difference approximations are used for the Laplace operator A. Furthermore, it can be seen from the above derivations that Eq. (2.39) can be readily generalized to multi-dimensional cases. For the case where the potential V is dependent on both temporal and spatial variables, the derivations are similar to those in Eq. (2.37) except that the product rule of the derivative with respect to t should be used.

2.3.2 Stability

To prevent the numerical solution from diverging, the stability of the generalized FDTD method is analyzed in Eq. (2.39). Here, the Laplace operator A is only approximated by either a second-order central difference operator $\frac{1}{\Delta x^2}\delta_x^2$ or a fourth-order central difference operator $\frac{1}{\Delta x^2}D_x^2$, where

$$A\psi_{real}^n(j) \approx \frac{1}{\Delta x^2}\delta_x^2$$

$$= \frac{1}{\Delta x^2}[\psi_{real}^n(j+1) - 2\psi_{real}^n(j) + \psi_{real}^n(j-1)], \tag{2.40a}$$

with $O(\Delta x^2)$,

$$A\psi_{real}^n(j) \approx \frac{1}{\Delta x^2}\delta_x^2 A\psi_{real}^n(j)$$

$$\approx \frac{1}{\Delta x^2}D_x^2 = \frac{1}{12\Delta x^2}[-\psi_{real}^n(j+2) + 16\psi_{real}^n(j+1)$$

$$-30\psi_{real}^n(j) + 16\psi_{real}^n(j-1) - \psi_{real}^n(j-2)], \tag{2.40b}$$

with $O(\Delta x^4)$,

and similar finite difference approximations are used for $A\psi_{imag}^{n+1/2}(j)$. The potential V is held constant and Von Neumann analysis [62] is used to analyze the stability of the generalized FDTD schemes. To this end, $\psi_{real}^n(j) = \lambda_{real}^n e^{ij\beta\Delta x}$ and $\psi_{imag}^{n+1/2}(j) = \lambda_{imag}^n e^{ij\beta\Delta x}$, where λ_{real} and λ_{imag} are amplification factors for $\psi_{real}^n(j)$ and $\psi_{imag}^{n+1/2}(j)$, respectively, and substitute them into Eq. (2.40a). This gives

$$\frac{1}{\Delta x^2}\delta_x^2\psi_{real}^n(j) = \frac{1}{\Delta x^2}\left(-4\sin^2\frac{\beta\Delta x}{2}\right)\lambda_{real}^n e^{ij\beta\Delta x}, \tag{2.41a}$$

$$\frac{1}{\Delta x^2}\delta_x^2\psi_{imag}^{n+1/2}(j) = \frac{1}{\Delta x^2}\left(-4\sin^2\frac{\beta\Delta x}{2}\right)\lambda_{imag}^n e^{ij\beta\Delta x}. \tag{2.41b}$$

Replacing A with $\frac{1}{\Delta x^2}\delta_x^2$ in Eq. (2.39), substituting Eq. (2.40) into the resulting equations, and then deleting the common factor $e^{ij\beta\Delta x}$, we obtain

$$\lambda_{real}^{n} = \lambda_{real}^{n-1} + 2\sum_{p=0}^{N}\frac{(-1)^p}{(2p+1)!}\left(\frac{\hbar}{m}r\sin^2\frac{\beta\Delta x}{2} + \frac{V\Delta t}{2\hbar}\right)^{2p+1}\lambda_{imag}^{n-1}, \tag{2.42a}$$

$$\lambda_{imag}^{n} = \lambda_{imag}^{n-1} + 2\sum_{p=0}^{N}\frac{(-1)^{p+1}}{(2p+1)!}\left(\frac{\hbar}{m}r\sin^2\frac{\beta\Delta x}{2} + \frac{V\Delta t}{2\hbar}\right)^{2p+1}\lambda_{real}^{n}, \tag{2.42b}$$

where $r = \frac{\Delta t}{\Delta x^2}$. Since Eq. (2.42a) is true for any time level n, we rewrite Eq. (2.42a) as

$$\lambda_{real}^{n+1} = \lambda_{real}^{n} + 2\sum_{p=0}^{N}\frac{(-1)^p}{(2p+1)!}\left(\frac{\hbar}{m}r\sin^2\frac{\beta\Delta x}{2} + \frac{V\Delta t}{2\hbar}\right)^{2p+1}\lambda_{imag}^{n}, \tag{2.43}$$

substract it by Eq. (2.42a), and then use Eq. (2.42b) to eliminate λ_{imag}. As such, we obtain a quadratic equation for λ_{real} as follows:

$$\lambda_{real}^2 - (2-\alpha^2)\lambda_{real} + 1 = 0, \tag{2.44}$$

where $\alpha = 2\sum_{p=0}^{N}\frac{(-1)^p}{(2p+1)!}\left(\frac{\hbar}{m}r\sin^2\frac{\beta\Delta x}{2} + \frac{V\Delta t}{2\hbar}\right)^{2p+1}$. Using the fact that for a quadratic equation $x^2 + Bx + C = 0$ the solution x satisfies $|x| \le 1$ if and only if $|B| \le 1 + |C|$ and $|C| \le 1$ we obtain from Eq. (2.44) that $|\lambda_{real}| \le 1$ if and only if $|\alpha| \le 2$. By the Von Neumann analysis, we conclude that the generalized FDTD scheme is stable if $|\alpha| \le 2$ *i.e.*,

$$\left|\sum_{p=0}^{N}\frac{(-1)^p}{(2p+1)!}\left(\frac{\hbar}{m}r\sin^2\frac{\beta\Delta x}{2} + \frac{V\Delta t}{2\hbar}\right)^{2p+1}\right| \le 1. \tag{2.45}$$

It can be seen that

$$\lim_{N\to\infty}\sum_{p=0}^{N}\frac{(-1)^p}{(2p+1)!}\left(\frac{\hbar}{m}r\sin^2\frac{\beta\Delta x}{2} + \frac{V\Delta t}{2\hbar}\right)^{2p+1}$$

$$= \sum_{p=0}^{\infty} \frac{(-1)^p}{(2p+1)!} \left(\frac{\hbar}{m} r \sin^2 \frac{\beta \Delta x}{2} + \frac{V \Delta t}{2\hbar}\right)^{2p+1}$$

$$= \sin\left(\frac{\hbar}{m} r \sin^2 \frac{\beta \Delta x}{2} + \frac{V \Delta t}{2\hbar}\right), \tag{2.46}$$

implying that, when $N \to \infty$, Eq. (2.45) is automatically satisfied, and, hence, the scheme with $N \to \infty$, is unconditionally stable. However, we cannot choose $N = \infty$ and, therefore, the generalized FDTD scheme must obey the condition in Eq. (2.45). Noting that the condition in Eq. (2.45) gives only $|\lambda_{real}| \leq 1$ and does not indicate whether or not there is a double root with $|\lambda_{real}| = 1$ in Eq. (2.44) (for this case, the numerical solution may still diverge), we choose the maximum value of $\sin^2 \frac{\beta \Delta x}{2}$ and require

$$\left| \sum_{p=0}^{N} \frac{(-1)^p}{(2p+1)!} \left(\frac{\hbar}{m} r + \frac{V \Delta t}{2\hbar}\right)^{2p+1} \right| \leq c < 1, \tag{2.47}$$

where c is a constant. Using a similar argument, we may obtain the same inequality as that in Eq. (2.47) for λ_{imag}. Hence, we obtain the following theorem.

Theorem 1. The generalized FDTD scheme

$$\psi_{real}^{n}(j) = \psi_{real}^{n-1}(j) + 2 \sum_{p=0}^{N} \frac{(-1)^{p+1}}{(2p+1)!} \left(\frac{\hbar}{4m} r \delta_x^2 - \frac{V \Delta t}{2\hbar}\right)^{2p+1} \cdot \psi_{imag}^{n-\frac{1}{2}}(j), \tag{2.48a}$$

$$\psi_{imag}^{n+1/2}(j) = \psi_{imag}^{n-1/2}(j) + 2 \sum_{p=0}^{N} \frac{(-1)^{p}}{(2p+1)!} \left(\frac{\hbar}{4m} r \delta_x^2 - \frac{V \Delta t}{2\hbar}\right)^{2p+1} \cdot \psi_{real}^{n}(j). \tag{2.48b}$$

is stable if Eq. (2.47) is satisfied. It can be seen that when $N = 0$ the condition in Eq. (2.47) reduces to that found in Eq. (2.30). Also, the accuracy of the scheme is $O(\Delta x^2 +$

$\Delta x^2\Delta t^2 + \cdots + \Delta t^{2N+2}$). Similarly, for the fourth-order central difference $\frac{1}{\Delta x^2} D_x^2$ case, we let $\psi_{real}^n(j) = \lambda_{real}^n e^{ij\beta\Delta x}$, $\psi_{imag}^{n+1/2}(j) = \lambda_{imag}^n e^{ij\beta\Delta x}$ and substitute them into Eq. (2.40b). This gives

$$\frac{1}{\Delta x^2} D_x^2 \psi_{real}^n(j) = \frac{-4}{3\Delta x^2} \sin^2 \frac{\beta\Delta x}{2} \left(3 + \sin^2 \frac{\beta\Delta x}{2}\right) \lambda_{real}^n e^{ij\beta\Delta x}, \tag{2.49a}$$

$$\frac{1}{\Delta x^2} D_x^2 \psi_{imag}^{n+1/2}(j) = \frac{-4}{3\Delta x^2} \sin^2 \frac{\beta\Delta x}{2} \left(3 + \sin^2 \frac{\beta\Delta x}{2}\right) \lambda_{imag}^n e^{ij\beta\Delta x}. \tag{2.49b}$$

Replacing A with $\frac{1}{\Delta x^2} D_x^2$, substituting Eq. (2.49) into Eq. (2.39), and deleting the common factor $e^{ij\beta\Delta x}$, we obtain a quadratic equation for λ_{real} as follows:

$$\lambda_{real}^2 - (2 - \alpha^2)\lambda_{real} + 1 = 0, \tag{2.50}$$

where $\alpha = 2\sum_{p=0}^{N} \frac{(-1)^p}{(2p+1)!} (\frac{\hbar}{3m} r \sin^2 \frac{\beta\Delta x}{2} (3 + \sin^2 \frac{\beta\Delta x}{2}) + \frac{V\Delta t}{2\hbar})^{2p+1}$. Hence, we use a similar argument as before and obtain the following theorem:

Theorem 2. The generalized FDTD scheme

$$\psi_{real}^n(j) = \psi_{real}^{n-1}(j) + 2\sum_{p=0}^{N} \frac{(-1)^{p+1}}{(2p+1)!} (\frac{\hbar}{4m} r D_x^2 - \frac{V\Delta t}{2\hbar})^{2p+1} \cdot \psi_{imag}^{n-\frac{1}{2}}(j), \tag{2.51a}$$

$$\psi_{imag}^{n+1/2}(j) = \psi_{imag}^{n-1/2}(j) + 2\sum_{p=0}^{N} \frac{(-1)^{p}}{(2p+1)!} (\frac{\hbar}{4m} r D_x^2 - \frac{V\Delta t}{2\hbar})^{2p+1} \cdot \psi_{real}^{n}(j), \tag{2.51b}$$

is stable if the following condition is satisfied

$$\left| \sum_{p=0}^{N} \frac{(-1)^p}{(2p+1)!} (\frac{4\hbar}{3m} r + \frac{V\Delta t}{2\hbar})^{2p+1} \right| \le c < 1, \tag{2.52}$$

where c is a constant. The accuracy of the scheme is $O(\Delta x^2 + \Delta x^2 \Delta t^2 + \cdots + \Delta t^{2N+2})$. It can be seen from the above both schemes that for a larger N, the evaluation for powers of δ_x^2 or D_x^2 can be quite expansive. Therefore, it is our suggestion to choose a small N for purposes of computation simplicity.

2.3.3 Absorbing Boundary Condition

When a particle travels and hits the boundary, it will reflect back to the domain under consideration during numerical simulations. Reflection will distort the wave packet solution. It is ideal to create an absorbing boundary condition so that the particle will not reflect back during the numerical simulation. Here, a second-order absorbing boundary condition (ABC) is obtained from analyzing the group velocity of the wavepacket at the boundaries [34]. We assume group velocities of the traveling particle to be

$$v_1 = \frac{\hbar k_1}{m} = \frac{\hbar 2\pi}{\lambda_1 m}, \qquad v_2 = \frac{\hbar k_2}{m} = \frac{\hbar 2\pi}{\lambda_2 m}, \tag{2.53}$$

where $k_j = \frac{2\pi}{\lambda_j}$, $j=1,2$, is the wavenumber and λ_j is the wavelength of the particle. By incorporating the dispersion relation $\frac{\hbar^2 k_j^2}{2m} = 2\pi v_j \hbar - V$ derived from Eq. (2.1), one may see that the wavenumber $k_j = \frac{2\pi}{\lambda_j}$ corresponds to $-i\partial_x$. Thus, the differential form of the wavenumber can be obtained as

$$\left(i\partial_x + \frac{m v_j}{\hbar}\right)\psi(x,t) = 0. \tag{2.54}$$

Since a wave maintains various components with different group velocities, a higher-order boundary condition is used as follows:

$$\left(i\partial_x + \frac{mv_1}{\hbar}\right)\left(i\partial_x + \frac{mv_2}{\hbar}\right)\psi(x,t) = 0. \tag{2.55}$$

It should be pointed out that for a wave traveling towards the left, v_1 and v_2 are substituted by $-v_1$ and $-v_2$. It may be seen from Eq.'s (2.54) and (2.55) that if v_1 does not equal v_2 the two different wave components with group velocities v_1 and v_2 will be absorbed, and alternately, if v_1 is equal to v_2 the component of the wave with group velocity v_1 (or v_2) will be absorbed to the second order. With Eq.'s (2.26), (2.27) and (2.55), the wavefunctions at the left and right boundaries can be determined as

$$\pm\hbar\partial_x\psi_{real}(x,t) - \hbar c_1\partial_t\psi_{real}(x,t) + (c_1V - c_2)\psi_{imag}(x,t) = 0, \tag{2.56a}$$

$$\mp\hbar\partial_x\psi_{imag}(x,t) + \hbar c_1\partial_t\psi_{imag}(x,t) + (c_1V - c_2)\psi_{real}(x,t) = 0, \tag{2.56b}$$

where

$$c_1 = \frac{2}{(v_1 + v_2)}, \qquad c_2 = \frac{mv_1v_2}{(v_1 + v_2)}. \tag{2.57}$$

Here, the upper signs in Eq. (2.56) apply to the left boundary, whereas the lower signs apply to the right. We then use the second-order finite difference schemes to approximate ψ_{real} and ψ_{imag} at the left ($j = 1$) and right ($j = N - 1$) boundaries as follows, respectively,

$$\psi_{imag}\left(x_j, t_{n+\frac{1}{2}}\right) \approx \frac{\psi_{imag}^{n+\frac{1}{2}}(j) + \psi_{imag}^{n+\frac{1}{2}}(j+1)}{2}, \tag{2.58a}$$

$$\partial_x\psi_{imag}\left(x_j, t_{n+\frac{1}{2}}\right) \approx \frac{\psi_{imag}^{n+\frac{1}{2}}(j+1) - \psi_{imag}^{n+\frac{1}{2}}(j)}{\Delta x}, \tag{2.58b}$$

$$\partial_t\psi_{imag}\left(x_j, t_{n+\frac{1}{2}}\right) \approx \frac{\psi_{imag}^{n+\frac{1}{2}}(j) - \psi_{imag}^{n-\frac{1}{2}}(j)}{\Delta t}, \tag{2.58c}$$

and

$$\psi_{real}(x_j, t_n) \approx \frac{\psi^n_{real}(j) + \psi^n_{real}(j+1)}{2}, \tag{2.59a}$$

$$\partial_x \psi_{real}(x_j, t_n) \approx \frac{\psi^n_{real}(j+1) - \psi^n_{real}(j)}{\Delta x}, \tag{2.59b}$$

$$\partial_t \psi_{real}(x_j, t_n) \approx \frac{\psi^n_{real}(j) - \psi^{n-1}_{real}(j)}{\Delta t}. \tag{2.59c}$$

Upon substituting Eq.'s (2.58) and (2.59) into Eq. (2.56), we obtain discrete absorbing boundary conditions as follows:

$$-\psi^{n+\frac{1}{2}}_{imag}(j)\left(\frac{\pm 1}{\Delta x} + \frac{c_1}{\Delta t}\right) \pm \frac{\psi^n_{real}(j+1)}{\Delta x} + \frac{c_1}{\Delta t}\psi^{n-1}_{real}(j)$$

$$= -\frac{\psi^{n-\frac{1}{2}}_{imag}(j) + \psi^{n-\frac{1}{2}}_{imag}(j+1)}{2\hbar(c_1 V - c_2)^{-1}}, \tag{2.60a}$$

and

$$-\psi^{n+1}_{real}(j)\left(\frac{\mp 1}{\Delta x} - \frac{c_1}{\Delta t}\right) \pm \frac{\psi^{n+\frac{1}{2}}_{imag}(j+1)}{\Delta x} - \frac{c_1}{\Delta t}\psi^{n-\frac{1}{2}}_{imag}(j)$$

$$= -\frac{\psi^n_{real}(j) + \psi^n_{real}(j+1)}{2\hbar(c_1 V - c_2)^{-1}}. \tag{2.60b}$$

2.3.4 Numerical Examples

To test the stability of the generalized FDTD schemes in Eq. (2.48) and Eq. (2.51) with discrete absorbing boundary conditions, Eq. (2.60), we employed the present schemes and the original FDTD scheme to simulate a particle moving in free space and then hitting an energy potential as tested in [80]. We initiated an electron at a wavelength of λ in a Gaussian envelop of width σ with the following two equations:

$$\psi_{real}^{0}(j) = \cos\left(\frac{2\pi(j-j_0)}{\lambda}\right) e^{-\frac{1}{2}\left(\frac{j-j_0}{\sigma}\right)^2} \tag{2.61a}$$

and

$$\psi_{imag}^{0}(j) = \sin\left(\frac{2\pi(j-j_0)}{\lambda}\right) e^{-\frac{1}{2}\left(\frac{j-j_0}{\sigma}\right)^2}, \tag{2.61b}$$

where $j_0\Delta x$ is the center of the pulse, and j is the spatial index. We chose a mesh of 1600 spatial grid points and the following values for parameters [80]: $\sigma = \lambda = \Delta x = 0.1 \times 10^{-10}$ (m), and $j_0 = 400$. Furthermore, V was chosen to be 0 in the first 800 grid points and 100 eV in the next 800 grid points.

Two quantities of importance in quantum mechanics are the expected values of the kinetic energy (T) and the potential energy (U) of the particle. They are calculated from $\psi_{real}^{n}(j)$ and $\psi_{imag}^{n+1/2}(j)$ in the simulation as follows,

$$T = -\frac{\hbar^2}{2m}\sum_{k=1}^{N}\left\{\left[\psi_{real}^{n}(j) - i\psi_{imag}^{n+\frac{1}{2}}(j)\right]\right.$$
$$\left.\cdot\left[\partial_{xx}\psi_{real}^{n}(j) + i\partial_{xx}\psi_{imag}^{n+\frac{1}{2}}(j)\right]\right\}, \tag{2.62a}$$

and

$$U = \sum_{k=1}^{N} V(j)\{[\psi_{real}^{n}(j)]^2 + [\psi_{imag}^{n+\frac{1}{2}}(j)]^2\}, \tag{2.62b}$$

where $\partial_{xx}\psi_{real}^{n}(j)$ and $\partial_{xx}\psi_{imag}^{n+\frac{1}{2}}(j)$ are evaluated using the fourth-order finite difference approximations:

$$\partial_{xx}\psi_{real}^{n}(j) \approx \frac{1}{12\Delta x^2}[-\psi_{real}^{n}(j+2) + 16\psi_{real}^{n}(j+1)$$
$$-30\psi_{real}^{n}(j) + 16\psi_{real}^{n}(j-1) - \psi_{real}^{n}(j-2)], \tag{2.63a}$$

and

$$\partial_{xx}\psi_{imag}^{n+\frac{1}{2}}(j) \approx \frac{1}{12\Delta x^2}[-\psi_{imag}^{n+\frac{1}{2}}(j+2) + 16\psi_{imag}^{n+\frac{1}{2}}(j+1)$$

$$-30\psi_{imag}^{n+\frac{1}{2}}(j) + 16\psi_{imag}^{n+\frac{1}{2}}(j-1) - \psi_{imag}^{n+\frac{1}{2}}(j-2)]. \quad (2.63b)$$

Based on the above formula, the electron moves in free space and then hits an energy potential with a total energy of about 150 eV. Before the energy barrier is reached, the energy of the electron is purely kinetic due to the fact that there is no potential energy available. With an increase in time, the electron will propagate in the positive spatial direction. The waveform begins to spread, but the total kinetic energy remains constant. After the electron strikes the potential barrier, part of the energy will be converted to potential energy. The waveform indicates that there is some probability that the electron is reflected and some probability that it penetrates the potential barrier. However, the total energy should remain constant.

In our computations, we chose $N = 2$ in Eq. (2.48) and Equation (2.51), and let

$$\Delta t = \mu \frac{2m\Delta x^2}{\hbar}[time], \quad (2.64)$$

where μ is a parameter used in [56]. Using Eq. (2.64), we rewrite the conditions in Eq. (2.52) for $N = 2$ as

$$\left|\sum_{p=0}^{2} \frac{(-1)^p}{(2p+1)!}\left(\frac{8}{3}\mu + \frac{\max|V|\,\Delta t}{2\hbar}\right)^{2p+1}\right| \le c < 1, \quad (2.65)$$

It is noted that when μ = 0.5 the original FDTD-Q scheme produces a divergent solution, because $\frac{\hbar}{m}r + \frac{\Delta t}{2\hbar}\max|V| = 2\mu + \frac{\Delta t}{2\hbar}\max|V| > 1$ which violates the stability condition. Thus, we employed the generalized FDTD scheme, Eq. (2.51) with $N = 2$ for this case. It is noted that when μ = 0.5, and $\left|\sum_{p=0}^{2} \frac{(-1)^p}{(2p+1)!}\left(\frac{8}{3}\mu + \frac{\max|V|\Delta t}{2\hbar}\right)^{2p+1}\right| \le 0.9735 + 1.7 \times$

$10^{-26} < 1$, the stability condition for Eq. (2.52) is satisfied. Figures 2.1-2.2 show the simulation of an electron moving in free space and then hitting a potential of 100 eV, which was obtained using the generalized FDTD scheme, Eq. (2.51) with $N = 2$ and $\mu = 0.5$. Again, it can be seen from Figure 2.1 that when the absorbing boundary condition is not imposed, the wavepacket is distorted at $n = 5.0 \times 10^4$. On the other hand, when an absorbing boundary condition is imposed, the wave-packet disappears at $n = 5.0 \times 10^4$, as shown in Figure 2.2.

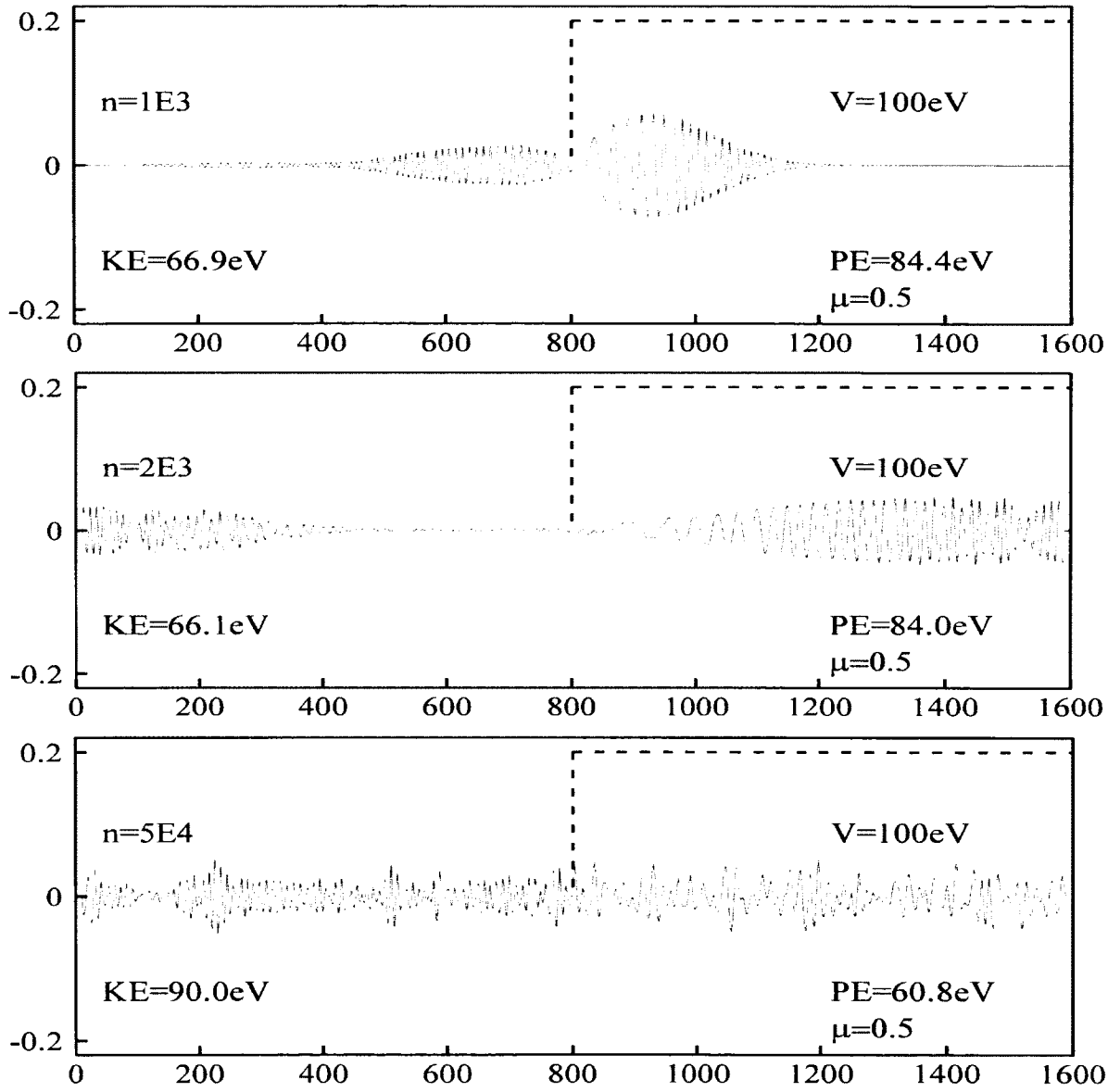


Fig. 2.1. Simulation of an electron moving in free space and then hitting a potential. Our fourth-order linear G-FDTD scheme was employed with $\mu = 0.5$ and no absorbing boundary condition [65].

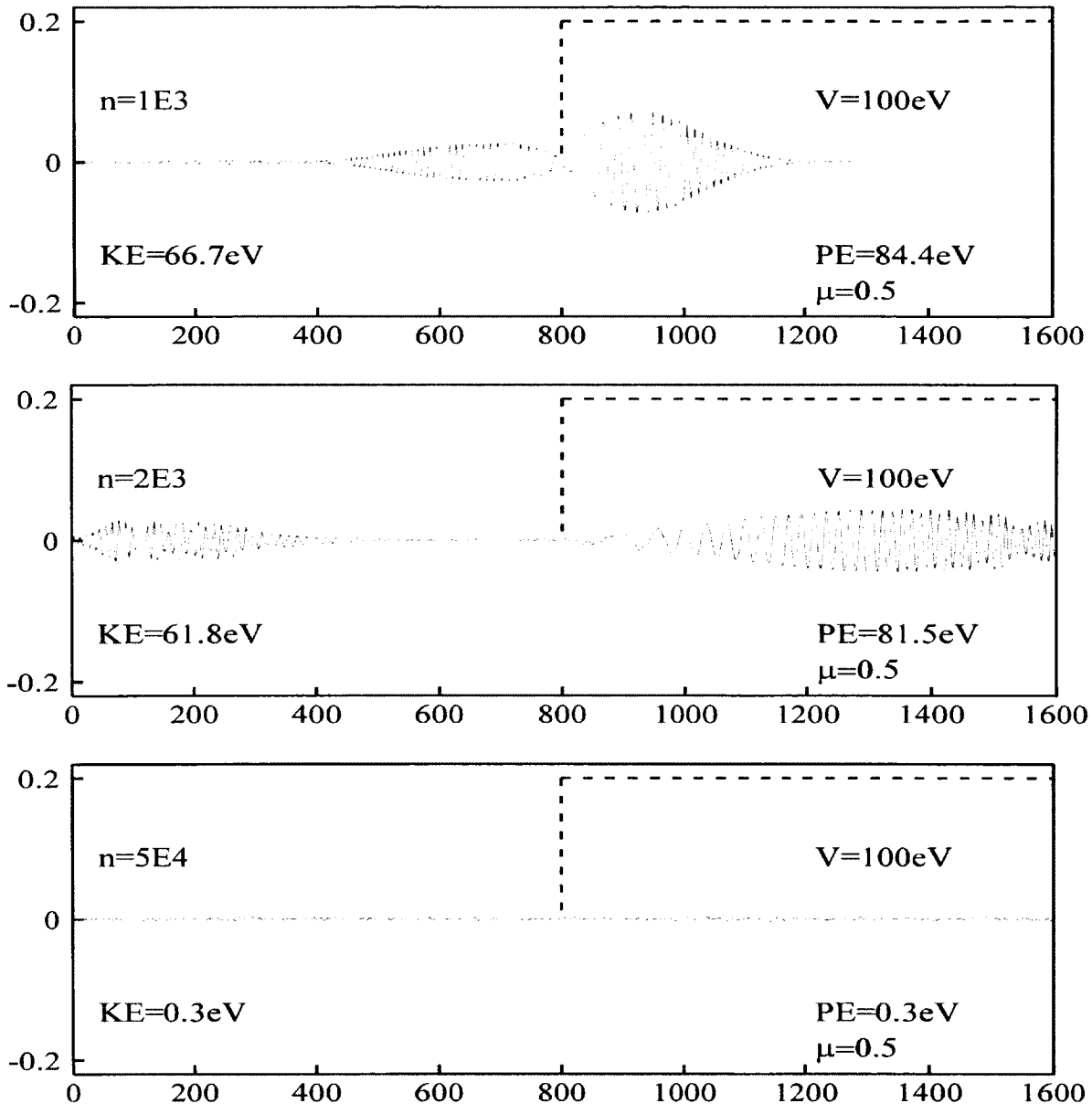


Fig. 2.2. Simulation of an electron moving in free space and then hitting a potential. Our fourth-order linear G-FDTD scheme was employed with $\mu = 0.5$ and absorbing boundary condition [65].

The above numerical example indicates that the generalized FDTD scheme breaks through the limitation ($\mu < 0.5$) of the original FDTD-Q scheme. It should be pointed out that one may obtain a larger value of μ if N is chosen to be larger in the generalized FDTD scheme.

2.4 Conclusion

Traditionally, finite-difference methods have a low spatial accuracy as evidenced in early evolution simulations of nonlinear wave equations [101]. In response to this low accuracy restraint, many spectral methods were developed in the 1970s. However, we have developed a linear G-FDTD method with absorbing boundary condition for solving the linear time-dependent Schrödinger equation. The linear G-FDTD overcomes the accuracy restraint, and we also have obtained a more relaxed condition for stability when central difference approximations are employed for the spatial derivatives. Numerical results coincide with those obtained based on the theoretical analysis. Spectral methods are more accurate than finite-differences in space. However, when coupled with the fourth-order methods for time-stepping, spectral methods deliver less accurate solutions than the explicit G-FDTD does. This is particularly noticeable when larger time steps are chosen in the simulation.

The G-FDTD method for solving the linear Schrödinger equation will now be extended for purposes of solving the nonlinear Schrödinger equation. This will provide an optimal finite-difference time-domain approach for solving this nonlinear wave equation, as it has been shown to optimize its linear counterpart. The spectral methods used in our numerical examples have spectral accuracy in space, which is higher than that offered by the G-FDTD, that is fourth-order accurate in space. Spectral accuracy means

that the spatial error decays exponentially as the grid spacing for sufficiently smooth functions. The higher-order accurate time-stepping benefit of the G-FDTD scheme allows for a more accurate solution in the time domain than do the spectral schemes that employ a RK4 method or split-step for time-stepping. The G-FDTD can be expressed as an explicit scheme that is simple and fast in computation, and will be shown to provide a higher-order accurate solution than do the spectral methods, particularly when the time step is large. The G-FDTD will aide in the understanding of nonlinear wave equations for the comprehension of nonlinear wave systems, especially when running long time simulations are required. The larger time steps that G-FDTD permits will make running long time simulations easier to perform in terms of computational costs.

CHAPTER THREE

EXPLICIT G-FDTD METHOD FOR SOLVING THE NONLINEAR SCHRÖDINGER EQUATION

3.1 Explicit G-FDTD

The objective of this dissertation is to generalize the previous research as found in [63, 65], to sufficiently handle a power-law nonlinear term [64]. To obtain the explicit G-FDTD scheme, we first assume that $\psi(x,t)$ be a sufficiently smooth function that vanishes for sufficiently large $|x|$. Using the idea of the G-FDTD method in [63, 64, 65], we split the variable $\psi(x,t)$ into real and imaginary components,

$$\psi(x,t) = \psi_{real}(x,t) + i\psi_{imag}(x,t). \tag{3.1}$$

Inserting Eq. (3.1) into Eq. (2.1) and then separating the real and imaginary parts results in the following coupled set of equations:

$$\begin{aligned}
&\frac{\partial \psi_{real}(x,t)}{\partial t} \\
&= \frac{\partial^2 \psi_{imag}(x,t)}{\partial x^2} - \lambda[\psi_{real}^2(x,t) + \psi_{imag}^2(x,t)]^{\frac{p-1}{2}}\psi_{imag}(x,t) \\
&= (A - \lambda|\psi|^{p-1})\psi_{imag}(x,t),
\end{aligned} \tag{3.2a}$$

$$\frac{\partial \psi_{imag}(x,t)}{\partial t}$$

$$= -\frac{\partial^2 \psi_{real}(x,t)}{\partial x^2} + \lambda[\psi_{real}^2(x,t) + \psi_{imag}^2(x,t)]^{\frac{p-1}{2}} \psi_{real}(x,t)$$

$$= -(A - \lambda|\psi|^{p-1})\psi_{real}(x,t), \tag{3.2b}$$

where $A = \frac{\partial^2}{\partial x^2}$ and $|\psi|^{p-1} = [\psi_{real}^2(x,t) + \psi_{imag}^2(x,t)]^{\frac{p-1}{2}}$.

We let $\psi_{real}^n(k)$ be the approximation of $\psi_{real}(k\Delta x, n\Delta t)$ and $\psi_{imag}^n(k)$for $\psi_{imag}(k\Delta x, n\Delta t)$. Using a Taylor series expansion at $t = (n - 1/2)\Delta t$, we obtain

$$\psi_{real}^n(k) - \psi_{real}^{n-1}(k)$$

$$= 2\sum_{m=0}^{M} (\frac{\Delta t}{2})^{2m+1} \frac{1}{(2m+1)!} \frac{\partial^{2m+1}\psi_{real}(x, t_{n-1/2})}{\partial t^{2m+1}} + O(\Delta t^{2M+3}), \tag{3.3a}$$

and

$$\psi_{imag}^n(k) - \psi_{imag}^{n-1}(k)$$

$$= 2\sum_{m=0}^{M} (\frac{\Delta t}{2})^{2m+1} \frac{1}{(2m+1)!} \frac{\partial^{2m+1}\psi_{imag}(x, t_{n-1/2})}{\partial t^{2m+1}} + O(\Delta t^{2M+3}). \tag{3.3b}$$

We then evaluate those derivatives in Eq. (3.3a) by using Eqs. (3.2a) and (3.2b) repeatedly, where t in $[\psi_{real}^2(x,t) + \psi_{imag}^2(x,t)]^{\frac{p-1}{2}}$ is fixed at $(n - 1/2)\Delta t$:

$$\frac{\partial \psi_{real}(x, t_{n-1/2})}{\partial t} = \left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right)\psi_{imag}(x, t_{n-1/2}), \tag{3.4a}$$

$$\frac{\partial^3 \psi_{real}(x, t_{n-1/2})}{\partial t^3} = \left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right)\frac{\partial^2 \psi_{imag}(x, t_{n-1/2})}{\partial t^2}$$

$$= -\left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right)\left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right)$$

$$\cdot \frac{\partial \psi_{real}(x, t_{n-1/2})}{\partial t}$$

$$= -\left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right)^3 \psi_{imag}(x, t_{n-1/2}), \tag{3.4b}$$

$$\frac{\partial^5 \psi_{real}(x, t_{n-1/2})}{\partial t^5} = -\left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right)^3 \frac{\partial^2 \psi_{imag}(x, t_{n-1/2})}{\partial t^2}$$

$$= \left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right)^3$$

$$\cdot \left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right) \frac{\partial \psi_{real}(x, t_{n-1/2})}{\partial t}$$

$$= \left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right)^5 \psi_{imag}(x, t_{n-1/2}), \tag{3.4c}$$

and so on, where $\left|\psi^{n-\frac{1}{2}}\right|^{p-1} = [\psi^2_{real}(k\Delta x, t_{n-1/2}) + \psi^2_{imag}(k\Delta x, t_{n-1/2})]^{\frac{p-1}{2}}$.

Substituting Eq. (3.4) and other similar equations into Eq. (3.3a) gives

$$\psi^n_{real}(k) - \psi^{n-1}_{real}(k)$$

$$= 2\sum_{m=0}^{M} \left(\frac{\Delta t}{2}\right)^{2m+1} \frac{(-1)^m}{(2m+1)!}\left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right)^{2m+1} \psi_{imag}(x, t_{n-1/2})$$

$$+O(\Delta t^{2M+3}). \tag{3.5}$$

Similarly, using Eqs. (3.2a) and (3.2b) repeatedly to evaluate those derivatives in Eq. (3.3b), we obtain

$$\frac{\partial \psi_{imag}(x, t_{n-1/2})}{\partial t} = -\left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right) \psi_{real}(x, t_{n-1/2}), \tag{3.6a}$$

$$\frac{\partial^3 \psi_{imag}(x, t_{n-1/2})}{\partial t^3} = -\left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right) \frac{\partial^2 \psi_{real}(x, t_{n-1/2})}{\partial t^2}$$

$$= -\left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right)\left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right)$$

$$\cdot \frac{\partial \psi_{imag}(x, t_{n-1/2})}{\partial t}$$

$$= \left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right)^3 \psi_{real}(x, t_{n-1/2}), \tag{3.6b}$$

$$\frac{\partial^5 \psi_{imag}(x, t_{n-1/2})}{\partial t^5} = \left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right)^3 \frac{\partial^2 \psi_{real}(x, t_{n-1/2})}{\partial t^2}$$

$$= \left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right)^3 \left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right)$$

$$\cdot \frac{\partial \psi_{imag}(x, t_{n-1/2})}{\partial t}$$

$$= -\left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right)^5 \psi_{real}(x, t_{n-1/2}), \tag{3.6c}$$

and so on. Substituting Eq. (3.6) and other similar equations into Eq. (3.3b) gives

$$\psi_{imag}^{n}(k) - \psi_{imag}^{n-1}(k)$$

$$= 2\sum_{m=0}^{M} \left(\frac{\Delta t}{2}\right)^{2m+1} \frac{(-1)^{m+1}}{(2m+1)!}\left(A - \lambda\left|\psi^{n-1/2}\right|^{p-1}\right)^{2m+1} \psi_{real}(x, t_{n-1/2})$$

$$+O(\Delta t^{2M+3}). \tag{3.7}$$

Noting that the term $\left|\psi^{n-\frac{1}{2}}\right|^{p-1}$ in Eqs. (3.5) and (3.7) needs to be evaluated, we use a similar argument and obtain

$$\psi_{real}^{n+1/2}(k) - \psi_{real}^{n-1/2}(k)$$

$$= 2\sum_{m=0}^{M} \left(\frac{\Delta t}{2}\right)^{2m+1} \frac{(-1)^{m}}{(2m+1)!} (A - \lambda|\psi^{n}|^{p-1})^{2m+1} \psi_{imag}(x, t_n)$$

$$+O(\Delta t^{2M+3}). \tag{3.8a}$$

$$\psi_{imag}^{n+1/2}(k) - \psi_{imag}^{n-1/2}(k)$$

$$= 2\sum_{m=0}^{M} \left(\frac{\Delta t}{2}\right)^{2m+1} \frac{(-1)^{m+1}}{(2m+1)!} (A - \lambda|\psi^{n}|^{p-1})^{2m+1} \psi_{real}(x, t_n)$$

$$+O(\Delta t^{2M+3}), \tag{3.8b}$$

where $|\psi^n|^{p-1} = [\psi^2_{real}(k\Delta x, t_n) + \psi^2_{imag}(k\Delta x, t_n)]^{\frac{p-1}{2}}$. Next, we couple Eqs. (3.5), (3.7) and (3.8) together, drop out the truncation error $O(\Delta t^{2M+3})$, and replace $\frac{\partial^2}{\partial x^2}$ by a fourth-order accurate central difference operator, $\frac{1}{\Delta x^2} D_x^2 u(k) \approx \frac{1}{12\Delta x^2}[-u(k+2) + 16u(k+1) - 30u(k) + 16u(k-1) - u(k-2)]$. This results in our G-FDTD scheme for solving the NLSE as follows:

$$\psi^n_{real}(k) - \psi^{n-1}_{real}(k) = 2\sum_{m=0}^{M} \frac{(-1)^m}{(2m+1)!}\left(\frac{\sigma}{2} D_x^2 - \frac{\lambda\Delta t}{2}\left|\psi^{n-1/2}\right|^{p-1}\right)^{2m+1} \psi_{imag}(x, t_{n-1/2}), \tag{3.9a}$$

$$\psi^n_{imag}(k) - \psi^{n-1}_{imag}(k) = 2\sum_{m=0}^{M} \frac{(-1)^{m+1}}{(2m+1)!}\left(\frac{\sigma}{2} D_x^2 - \frac{\lambda\Delta t}{2}\left|\psi^{n-1/2}\right|^{p-1}\right)^{2m+1} \psi_{real}(x, t_{n-1/2}); \tag{3.9b}$$

$$\psi^{n+1/2}_{real}(k) - \psi^{n-1/2}_{real}(k) = 2\sum_{m=0}^{M} \frac{(-1)^m}{(2m+1)!}\left(\frac{\sigma}{2} D_x^2 - \frac{\lambda\Delta t}{2}|\psi^n|^{p-1}\right)^{2m+1} \psi_{imag}(x, t_n), \tag{3.9c}$$

$$\psi^{n+1/2}_{imag}(k) - \psi^{n-1/2}_{imag}(k) = 2\sum_{m=0}^{M} \frac{(-1)^{m+1}}{(2m+1)!}\left(\frac{\sigma}{2} D_x^2 - \frac{\lambda\Delta t}{2}|\psi^n|^{p-1}\right)^{2m+1} \psi_{real}(x, t_n), \tag{3.9d}$$

where, $\sigma = \Delta t/\Delta x^2$. Thus, once $\psi^{n-1/2}_{imag}(k)$ and $\psi^{n-1/2}_{real}(k)$ are given, one may explicitly calculate $\psi^n_{imag}(k)$ and $\psi^n_{real}(k)$ using Eqs. (3.9a) and (3.9b), and further obtain $\psi^{n+1/2}_{imag}(k)$ and $\psi^{n+1/2}_{real}(k)$ using Eqs. (3.9c) and (3.9d). It is an explicit iteration, and therefore the computation is simple and fast. It can be seen that the truncation error between the above scheme and Eqs. (3.5), (3.7) and (3.8) is $O(\Delta x^4 + \cdots + \Delta t^{2M+3})$.

However, the truncation error between the above scheme and the original NLSE, Eq. (2.1), may be different because the time in the term $|\psi(x,t)|^{p-1}$ is fixed in the small time interval either (t_{n-1}, t_n) or $(t_{n-1/2}, t_{n+1/2})$ in the above derivation. Furthermore, it should be pointed out that the partial derivative $\frac{\partial^2}{\partial x^2}$ can alternatively be approximated using a spectral, or other higher-order method. In this study, we confine our attention to the finite difference method with a fourth-order central difference approximation. For two-dimensional (2D) cases, one may replace the Laplace operator A by a two-dimensional finite difference operator and hence the 2D G-FDTD scheme for solving the 2D NLSE as follows:

$$\psi_{real}^{n}(j,k) - \psi_{real}^{n-1}(j,k) = 2\sum_{m=0}^{M}\frac{(-1)^m}{(2m+1)!}\left(\frac{\sigma_x}{2}D_x^2 + \frac{\sigma_y}{2}D_y^2 - \frac{\lambda\Delta t}{2}\left|\psi^{n-1/2}\right|^{p-1}\right)^{2m+1}\psi_{imag}^{n-1/2}(j,k), \tag{3.10a}$$

$$\psi_{imag}^{n}(j,k) - \psi_{imag}^{n-1}(j,k) = 2\sum_{m=0}^{M}\frac{(-1)^{m+1}}{(2m+1)!}\left(\frac{\sigma_x}{2}D_x^2 + \frac{\sigma_y}{2}D_y^2 - \frac{\lambda\Delta t}{2}\left|\psi^{n-1/2}\right|^{p-1}\right)^{2m+1}\psi_{real}^{n-1/2}(j,k); \tag{3.10b}$$

$$\psi_{real}^{n+1/2}(j,k) - \psi_{real}^{n-1/2}(j,k) = 2\sum_{m=0}^{M}\frac{(-1)^m}{(2m+1)!}\left(\frac{\sigma_x}{2}D_x^2 + \frac{\sigma_y}{2}D_y^2 - \frac{\lambda\Delta t}{2}|\psi^{n}|^{p-1}\right)^{2m+1}\psi_{imag}^{n}(j,k), \tag{3.10c}$$

$$\psi_{imag}^{n+1/2}(j,k) - \psi_{imag}^{n-1/2}(j,k) = 2\sum_{m=0}^{M}\frac{(-1)^{m+1}}{(2m+1)!}\left(\frac{\sigma_x}{2}D_x^2 + \frac{\sigma_y}{2}D_y^2 - \frac{\lambda\Delta t}{2}|\psi^{n}|^{p-1}\right)^{2m+1}\psi_{real}^{n}(j,k), \tag{3.10d}$$

where, $\sigma_x = \Delta t/\Delta x^2$, $\sigma_y = \Delta t/\Delta y^2$, $\frac{1}{\Delta x^2} D_x^2 u(j,k) \approx \frac{1}{12\Delta x^2}[-u(j+2,k) + 16u(j+1,k) - 30u(j,k) + 16u(j-1,k) - u(j-2,k)]$, and $\frac{1}{\Delta y^2} D_y^2 u(j,k) \approx \frac{1}{12\Delta x^2}[-u(j,k+2) + 16u(j,k+1) - 30u(j,k) + 16u(j,k-1) - u(j,k-2)]$.

The G-FDTD scheme for 3D NLSE can be obtained using a similar argument.

3.2 Discrete Conservation Law for the Explicit G-FDTD

To show that the Explicit G-FDTD scheme satisfies the first conservation law, Eq. (2.3), we consider *M=1* in the scheme without loss of generality and introduce some finite difference operators, δ_x^2, δ_{2x}^2, ∇_x, $\overline{\nabla}_x$, ∇_{2x}, and $\overline{\nabla}_{2x}$ as

$$\delta_x^2 u(k) = u(k+1) - 2u(k) + u(k-1),$$

$$\delta_{2x}^2 u(k) = u(k+2) - 2u(k) + u(k-2),$$

$$\nabla_x u(k) = u(k+1) - u(k), \qquad \overline{\nabla}_x u(k) = u(k) - u(k-1),$$

$$\nabla_{2x} u(k) = u(k+2) - u(k), \qquad \overline{\nabla}_{2x} u(k) = u(k) - u(k-2).$$

It can be seen that these finite difference operators satisfy the relations: $\delta_x^2 = \frac{1}{12}[-\delta_{2x}^2 + 16\delta_x^2]$, $\delta_x^2 = \overline{\nabla}_x \cdot \nabla_x = \nabla_x - \overline{\nabla}_x$, $\delta_{2x}^2 = \overline{\nabla}_{2x} \cdot \nabla_{2x} = \nabla_{2x} - \overline{\nabla}_{2x}$, $\overline{\nabla}_{2x} \cdot \nabla_x = \nabla_x \cdot \overline{\nabla}_{2x}$, etc. Furthermore, one may observe that for any mesh functions, $u(k)$ and $v(k)$, which are assumed to be zero for sufficiently large $|k|$,

$$\sum_{k\in Z} \overline{\nabla}_x u(k) \cdot v(k) = -\sum_{k\in Z} u(k) \cdot \nabla_x v(k),$$

$$\sum_{k\in Z} \nabla_x u(k) \cdot v(k) = -\sum_{k\in Z} u(k) \cdot \overline{\nabla}_x v(k),$$

$$\sum_{k\in Z} \nabla_x u(k) \cdot v(k) = -\sum_{k\in Z} u(k) \cdot \overline{\nabla}_x v(k), \tag{3.11a}$$

$$\sum_{k\in Z} \bar{\nabla}_{2x} u(k) \cdot v(k) = -\sum_{k\in Z} u(k) \cdot \nabla_{2x} v(k),$$

$$\sum_{k\in Z} \nabla_{2x} u(k) \cdot v(k) = -\sum_{k\in Z} u(k) \cdot \bar{\nabla}_{2x} v(k), \tag{3.11b}$$

$$\sum_{k\in Z} \delta_x^2 u(k) \cdot v(k) = -\sum_{k\in Z} \nabla_x u(k) \cdot \nabla_x v(k)$$

$$= \sum_{k\in Z} u(k) \cdot \delta_x^2 v(k), \tag{3.11c}$$

$$\sum_{k\in Z} \delta_{2x}^2 u(k) \cdot v(k) = -\sum_{k\in Z} \nabla_{2x} u(k) \cdot \nabla_{2x} v(k)$$

$$= \sum_{k\in Z} u(k) \cdot \delta_{2x}^2 v(k), \tag{3.11d}$$

where Z is the set of all positive and negative integers. The first step is to replace n in Eq. (3.9a) and (3.9b) by $n+1$, and then take the resulting Eq. (3.9a) $\times\ \psi_{real}^{n+1/2}(k)$ + Eq. (3.9b) $\times\ \psi_{imag}^{n+1/2}(k)$, Eq. (3.9c) $\times\ \psi_{real}^{n}(k)$ + Eq. (3.9d) $\times\ \psi_{imag}^{n}(k)$, then sum k over all integers Z. This gives

$$\sum_{k\in Z} \{\psi_{real}^{n+1}(k) \cdot \psi_{real}^{n+\frac{1}{2}}(k) - \psi_{real}^{n}(k) \cdot \psi_{real}^{n+\frac{1}{2}}(k) + \psi_{imag}^{n+1}(k) \cdot \psi_{imag}^{n+\frac{1}{2}}(k)$$

$$-\psi_{imag}^{n}(k) \cdot \psi_{imag}^{n+\frac{1}{2}}(k) + \psi_{real}^{n}(k) \cdot \psi_{real}^{n+\frac{1}{2}}(k) - \psi_{real}^{n}(k) \cdot \psi_{real}^{n-\frac{1}{2}}(k)$$

$$+\psi_{imag}^{n}(k) \cdot \psi_{imag}^{n+\frac{1}{2}}(k) - \psi_{imag}^{n}(k) \cdot \psi_{imag}^{n-\frac{1}{2}}(k)\}$$

$$= \sum_{k\in Z} \{[\sigma D_x^2 - \Delta t\lambda \left|\psi^{n+\frac{1}{2}}\right|^{p-1}] \psi_{imag}^{n+1/2}(k) \cdot \psi_{real}^{n+1/2}(k)$$

$$-[\sigma D_x^2 - \Delta t\lambda \left|\psi^{n+\frac{1}{2}}\right|^{p-1}] \psi_{real}^{n+1/2}(k) \cdot \psi_{imag}^{n+1/2}(k)\}$$

$$+\sum_{k\in Z}\{\frac{1}{24}[\sigma D_x^2-\Delta t\lambda\left|\psi^{n+\frac{1}{2}}\right|^{p-1}]^3\psi_{imag}^{n+1/2}(k)\cdot\psi_{real}^{n+1/2}(k)$$

$$-\frac{1}{24}[\sigma D_x^2-\Delta t\lambda\left|\psi^{n+\frac{1}{2}}\right|^{p-1}]^3\psi_{real}^{n+1/2}(k)\cdot\psi_{imag}^{n+1/2}(k)\}$$

$$+\sum_{k\in Z}\{[\sigma D_x^2-\Delta t\lambda|\psi^n|^{p-1}]\psi_{imag}^{n}(k)\cdot\psi_{real}^{n}(k)$$

$$-[\sigma D_x^2-\Delta t\lambda|\psi^n|^{p-1}]\psi_{real}^{n}(k)\cdot\psi_{imag}^{n}(k)\}$$

$$+\sum_{k\in Z}\{\frac{1}{24}[\sigma D_x^2-\Delta t\lambda|\psi^n|^{p-1}]^3\psi_{imag}^{n}(k)\cdot\psi_{real}^{n}(k)$$

$$-\frac{1}{24}[\sigma D_x^2-\Delta t\lambda|\psi^n|^{p-1}]^3\psi_{real}^{n}(k)\cdot\psi_{imag}^{n}(k)\}. \tag{3.12}$$

It can be seen that the left-hand-side (LHS) of Eq. (3.12) can be simplified as

$$LHS=\sum_{k\in Z}\{\psi_{real}^{n+1}(k)\cdot\psi_{real}^{n+\frac{1}{2}}(k)+\psi_{imag}^{n+1}(k)\cdot\psi_{imag}^{n+\frac{1}{2}}(k)-\psi_{real}^{n}(k)$$

$$\cdot\psi_{real}^{n-\frac{1}{2}}(k)-\psi_{imag}^{n}(k)\cdot\psi_{imag}^{n-\frac{1}{2}}(k)\}. \tag{3.13}$$

We will next demonstrate that the right-hand-side (RHS) of Eq. (3.12) must be zero. To this end, we first use the aforementioned properties and obtain

$$\sum_{k\in Z}\sigma\delta_x^2\psi_{imag}^{n+1/2}(k)\cdot\psi_{real}^{n+\frac{1}{2}}(k)$$

$$=\sum_{k\in Z}\frac{\sigma}{12}[-\delta_{2x}^2+16\delta_x^2]\psi_{imag}^{n+1/2}(k)\cdot\psi_{real}^{n+\frac{1}{2}}(k)$$

$$=\frac{\sigma}{12}\sum_{k\in Z}\nabla_{2x}\psi_{imag}^{n+1/2}(k)\cdot\nabla_{2x}\psi_{real}^{n+\frac{1}{2}}(k)$$

$$-\frac{16}{12}\sigma\sum_{k\in Z}\nabla_x\psi_{imag}^{n+1/2}(k)\cdot\nabla_x\psi_{real}^{n+\frac{1}{2}}(k)$$

$$= \sum_{k\in Z} \frac{\sigma}{12}[-\delta_{2x}^2 + 16\delta_x^2]\psi_{real}^{n+1/2}(k) \cdot \psi_{imag}^{n+\frac{1}{2}}(k)$$

$$= \sum_{k\in Z} \sigma\delta_x^2\psi_{real}^{n+1/2}(k) \cdot \psi_{imag}^{n+\frac{1}{2}}(k). \tag{3.14}$$

Hence, the first summation on the right-hand-side of Eq. (3.12) is zero and disappears. Similarly, the third summation on the right-hand-side of Eq. (3.12) is also zero and disappears. For the second summation on the right-hand-side of Eq. (3.12), we note that the operators

$$[\sigma D_x^2 - \Delta t\lambda|\psi^{n+1/2}|^{p-1}]^3$$

$$= \sigma^3[D_x^2]^3 - \Delta t\lambda\sigma^2[D_x^2]^2|\psi^{n+1/2}|^{p-1} - \Delta t\lambda\sigma^2|\psi^{n+1/2}|^{p-1}[D_x^2]^2$$

$$-\Delta t\lambda\sigma^2 D_x^2\left|\psi^{n+\frac{1}{2}}\right|^{p-1} D_x^2 + \Delta t^2\lambda^2\sigma D_x^2[\left|\psi^{n+\frac{1}{2}}\right|^{p-1}]^2$$

$$+\Delta t^2\lambda^2\sigma\left|\psi^{n+\frac{1}{2}}\right|^{p-1} D_x^2\left|\psi^{n+\frac{1}{2}}\right|^{p-1} + \Delta t^2\lambda^2\sigma[\left|\psi^{n+\frac{1}{2}}\right|^{p-1}]^2 D_x^2$$

$$-\Delta t^3\lambda^3[\left|\psi^{n+\frac{1}{2}}\right|^{p-1}]^3, \tag{3.15a}$$

$$[D_x^2]^3 = \frac{1}{12^2}\{[\delta_{2x}^2]^2 - 32\delta_{2x}^2\delta_x^2 + 16^2[\delta_x^2]^2\}, \tag{3.15b}$$

$$[D_x^2]^3 = \frac{1}{12^3}\{-[\delta_{2x}^2]^3 + 3\cdot 16[\delta_{2x}^2]^2\delta_x^2 - 3\cdot 16^2\delta_{2x}^2[\delta_x^2]^2$$

$$+16^3[\delta_x^2]^3\}. \tag{3.15c}$$

Using the above properties, we obtain that for any mesh functions, $u(k)$ and $v(k)$ which are assumed to be zero for sufficiently large $|k|$,

$$\sum_{k\in Z}([D_x^2]^3u(k)) \cdot v(k) = \frac{1}{12^3}\{\sum_{k\in Z} \nabla_{2x}D_{2x}^2u(k) \cdot \nabla_{2x}D_{2x}^2v(k)$$

$$-48\sum_{k\in Z}\nabla_x D_{2x}^2 u(k)\cdot\nabla_x D_{2x}^2 v(k)$$

$$+3\cdot 16^2\sum_{k\in Z}\nabla_{2x} D_x^2 u(k)\cdot\nabla_{2x} D_x^2 v(k)$$

$$-16^3\sum_{k\in Z}\nabla_x D_x^2 u(k)\cdot\nabla_x D_x^2 v(k)\},\tag{3.16a}$$

$$\sum_{k\in Z}([D_x^2]^2\left|\psi^{n-\frac{1}{2}}\right|^{p-1}u(k))\cdot v(k)$$

$$=\frac{1}{12^2}\{\sum_{k\in Z}D_{2x}^2(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}u(k))\cdot D_{2x}^2 v(k)$$

$$-32\sum_{k\in Z}\nabla_{2x}\nabla_x(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}u(k))\cdot\nabla_{2x}\nabla_x v(k)$$

$$+16^2\sum_{k\in Z}D_x^2(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}u(k))\cdot D_x^2 v(k)\},\tag{3.16b}$$

$$\sum_{k\in Z}\left|\psi^{n-\frac{1}{2}}\right|^{p-1}[D_x^2]^2 u(k)\cdot v(k)$$

$$=\frac{1}{12^2}\{\sum_{k\in Z}D_{2x}^2 u(k)\cdot D_{2x}^2(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}v(k))$$

$$-32\sum_{k\in Z}\nabla_{2x}\nabla_x u(k)\cdot\nabla_{2x}\nabla_x(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}v(k))$$

$$+16^2\sum_{k\in Z}D_x^2 u(k)\cdot D_x^2(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}v(k))\}\tag{3.16c}$$

$$\sum_{k\in Z}D_x^2(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}D_x^2 u(k))\cdot v(k)$$

$$=\sum_{k\in Z}\left|\psi^{n-\frac{1}{2}}\right|^{p-1}D_x^2 u(k)\cdot D_x^2 v(k),\tag{3.16d}$$

$$\sum_{k\in Z} D_x^2([\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^2 u(k)) \cdot v(k)$$

$$= \frac{1}{12}\{\sum_{k\in Z} \nabla_{2x}([\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^2 u(k)) \cdot \nabla_{2x} v(k)$$

$$-16\sum_{k\in Z} \nabla_x([\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^2 u(k)) \cdot \nabla_x v(k)\}, \tag{3.16e}$$

$$\sum_{k\in Z} \left|\psi^{n-\frac{1}{2}}\right|^{p-1} D_x^2(\left|\psi^{n-\frac{1}{2}}\right|^{p-1} u(k)) \cdot v(k)$$

$$= \frac{1}{12}\{\sum_{k\in Z} \nabla_{2x}(\left|\psi^{n-\frac{1}{2}}\right|^{p-1} u(k)) \cdot \nabla_{2x}(\left|\psi^{n-\frac{1}{2}}\right|^{p-1} v(k))$$

$$-16\sum_{k\in Z} \nabla_x(\left|\psi^{n-\frac{1}{2}}\right|^{p-1} u(k)) \cdot \nabla_x(\left|\psi^{n-\frac{1}{2}}\right|^{p-1} v(k))\}, \tag{3.16f}$$

$$\sum_{k\in Z} [\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^2 D_x^2 u(k) \cdot v(k)$$

$$= \frac{1}{12}\{\sum_{k\in Z} \nabla_{2x} u(k) \cdot \nabla_{2x}([\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^2 v(k))$$

$$-16\sum_{k\in Z} \nabla_x u(k) \cdot \nabla_x([\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^2 v(k))\}, \tag{3.16g}$$

$$\sum_{k\in Z} [\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^3 u(k) \cdot v(k) = \sum_{k\in Z} u(k) \cdot [\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^3 v(k). \tag{3.16h}$$

Letting $u(k) = \psi_{imag}^{n+1/2}(k)$, $v(k) = \psi_{real}^{n+1/2}(k)$, and then $u(k) = \psi_{real}^{n+1/2}(k)$, $v(k) = \psi_{imag}^{n+1/2}(k)$ in Eq. (3.16), in addition to using Eq. (3.15a), we obtain that the second summation on the RHS of Eq. (3.12) is zero as a consequence of symmetry. Using a similar argument, we also obtain that the fourth summation on the RHS of Eq. (3.12) is zero and hence

$$\sum_{k\in Z}\{\psi_{real}^{n+1}(k)\cdot\psi_{real}^{n+\frac{1}{2}}(k)+\psi_{imag}^{n+1}(k)\cdot\psi_{imag}^{n+\frac{1}{2}}(k)\}$$

$$=\sum_{k\in Z}\{\psi_{real}^{n}(k)\cdot\psi_{real}^{n-\frac{1}{2}}(k)+\psi_{imag}^{n}(k)\cdot\psi_{imag}^{n-\frac{1}{2}}(k)\}, \tag{3.17}$$

for any time level n. Thus, we consider Eq. (3.17) to be a discrete analogous form of the conservation law, Eq. (2.3).

3.3 Numerical Examples for the Explicit G-FDTD

The first example is to consider a bright soliton propagation, where $\lambda = -2$ and $p = 3$ in

$$i\frac{\partial\psi(x,t)}{\partial t}-\frac{\partial^2\psi(x,t)}{\partial x^2}+\lambda|\psi(x,t)|^{p-1}\psi(x,t)=0. \tag{3.18}$$

The initial condition was chosen such that the analytical solution is

$$\psi(x,t)=\operatorname{sech}(x+10-4t)\cdot e^{-i(2x+20-3t)}. \tag{3.19}$$

In our computation, the interval was taken to be $-20 \le x \le 20$. The solution was defined to be zero outside the interval since the value of $\psi(x,t)$ is initially negligible outside the computation interval. We chose *M=1* in our scheme, Eq. (3.9), and the number of grid points to be 300, 400, and 500, respectively, with $\Delta t = 0.001$. Using a various number of grid points allows the accuracy of the numerical solution to be studied when compared to those obtained using alternative methods. Numerical results were compared with the popular pseudospectral, split-step in time with the Fourier transform (FT) in space, and integrating-factor in time with FT in space methods, all of which were implemented using MATLAB software. The maximum errors are listed in Table I, from which one may see that the G-FDTD scheme provides a more accurate solution than the others do, as compared with the analytical solution. The evaluation of maximum error at

each time level n *vs.* t within $0 \leq t \leq 1$ can be seen in Figure 3.1. From Table 3.1 and Figure 3.1, one may see that our present scheme provides a more accurate solution.

Figure 3.2 illustrates the bright soliton propagation in free space at $t=1$ and 2, obtained using $N=400$. It can be seen from Figure 3.2 that there is no significant difference between the numerical solution and the analytical solution. In particular, Figures 3.3 and 3.4 illustrate bright soliton propagation in free space near the right boundary. This conveniently demonstrates the soliton propagates out of the boundary with analytical solution continuation.

Table 3.1 Maximum error obtained using fourth-order explicit G-FDTD for bright soliton propagation when $0 \leq t \leq 1, \Delta t = 0.001$.

Grid Points	PseudoSpectral	Split-Step	Integrating-Factor	G-FDTD
300	5.56284×10^{-2}	5.56284×10^{-2}	5.56284×10^{-2}	2.53137×10^{-4}
400	4.81756×10^{-2}	4.81757×10^{-2}	4.81757×10^{-2}	8.11204×10^{-5}
500	4.30896×10^{-2}	4.30896×10^{-2}	4.30896×10^{-2}	3.39131×10^{-5}

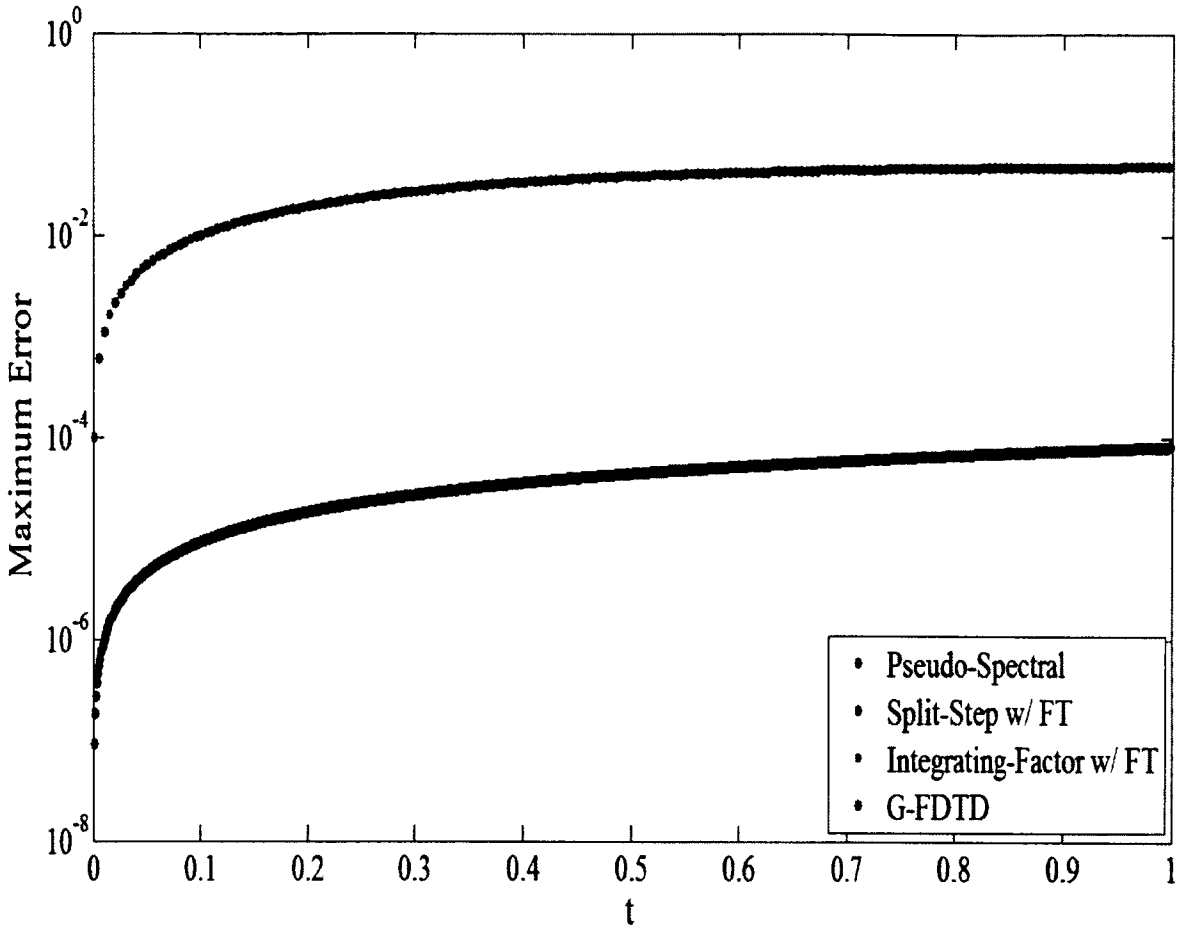


Fig. 3.1 Comparison of maximum errors at each time level n vs. t within $0 \leq t \leq 1$ between the explicit G-FDTD scheme and other numerical methods where $\Delta t = 0.001$ and $\Delta x = 0.1$.

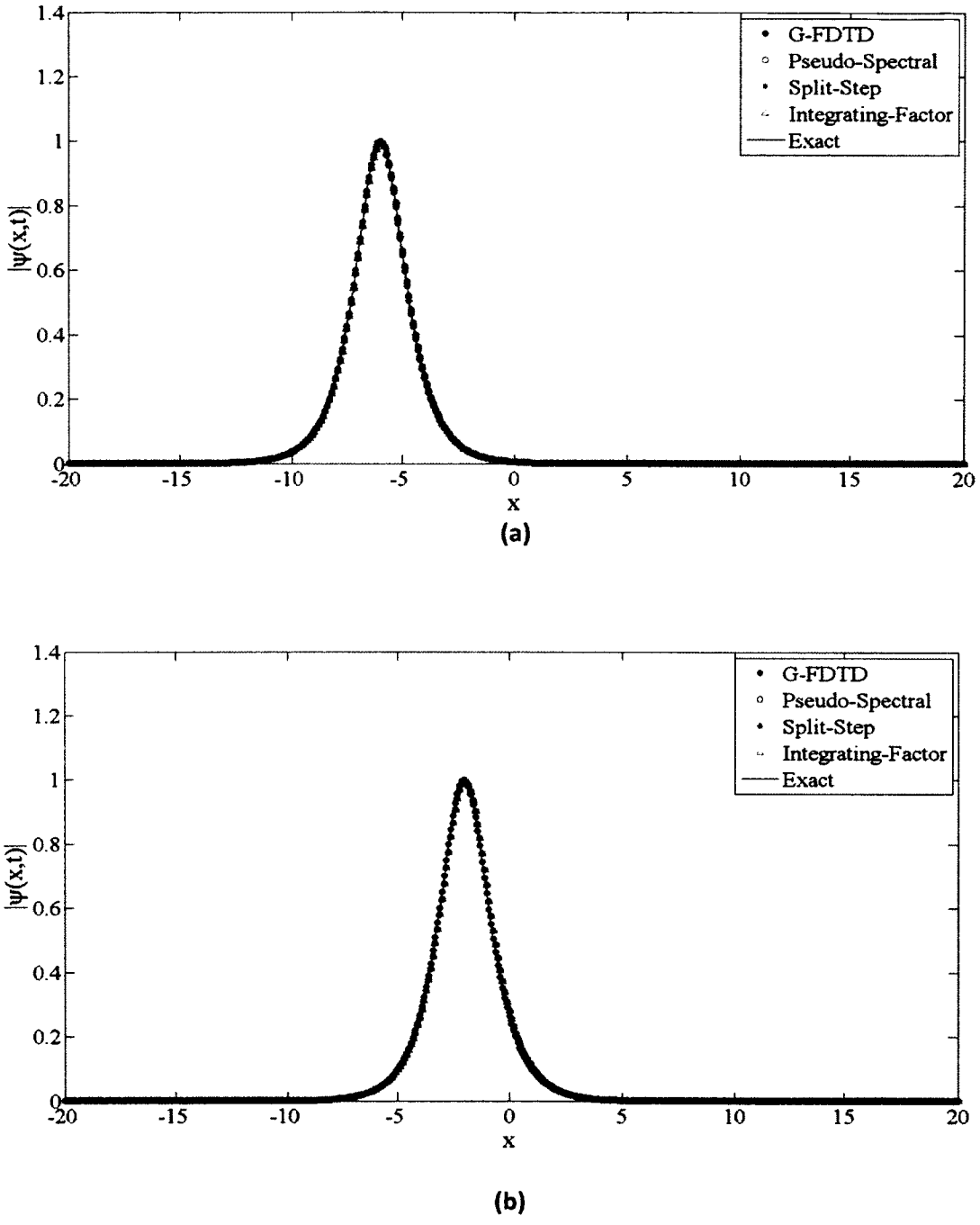


Fig. 3.2 Simulation of a bright soliton propagating in free space, where the explicit G-FDTD scheme and other numerical methods were employed with $\Delta t = 0.001$, $\Delta x = 0.1$ at **(a)** $t = 1$ and **(b)** $t = 2$.

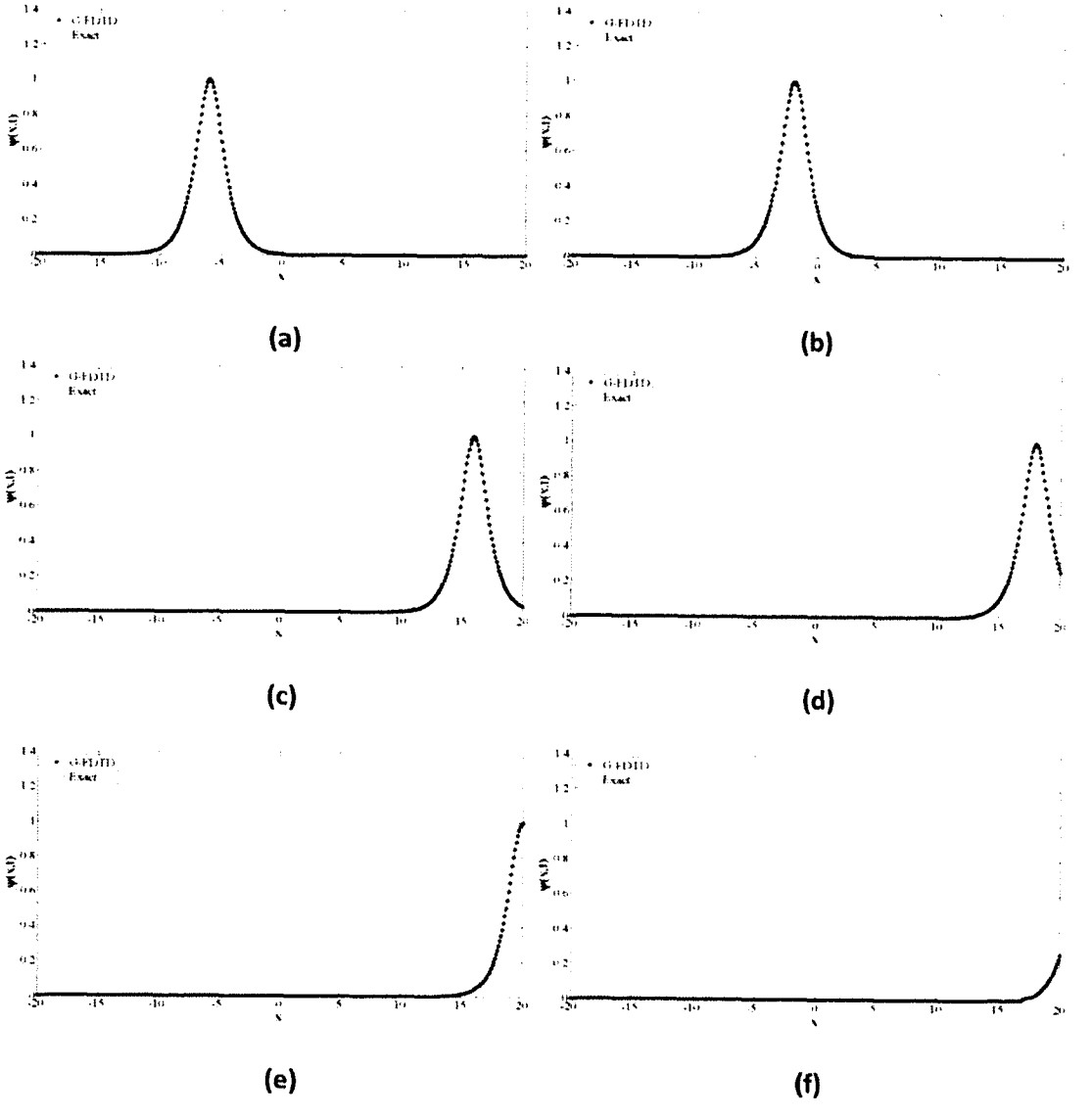


Fig. 3.3 Simulation of a bright soliton propagating in free space to the right, where the explicit G-FDTD scheme was employed with $\Delta t = 0.001$, $\Delta x = 0.1$ at **(a)** $t = 1$, **(b)** $t = 2$, **(c)** $t = 6.5$, **(d)** $t = 7$, **(e)** $t = 7.5$, and **(f)** $t = 8$.

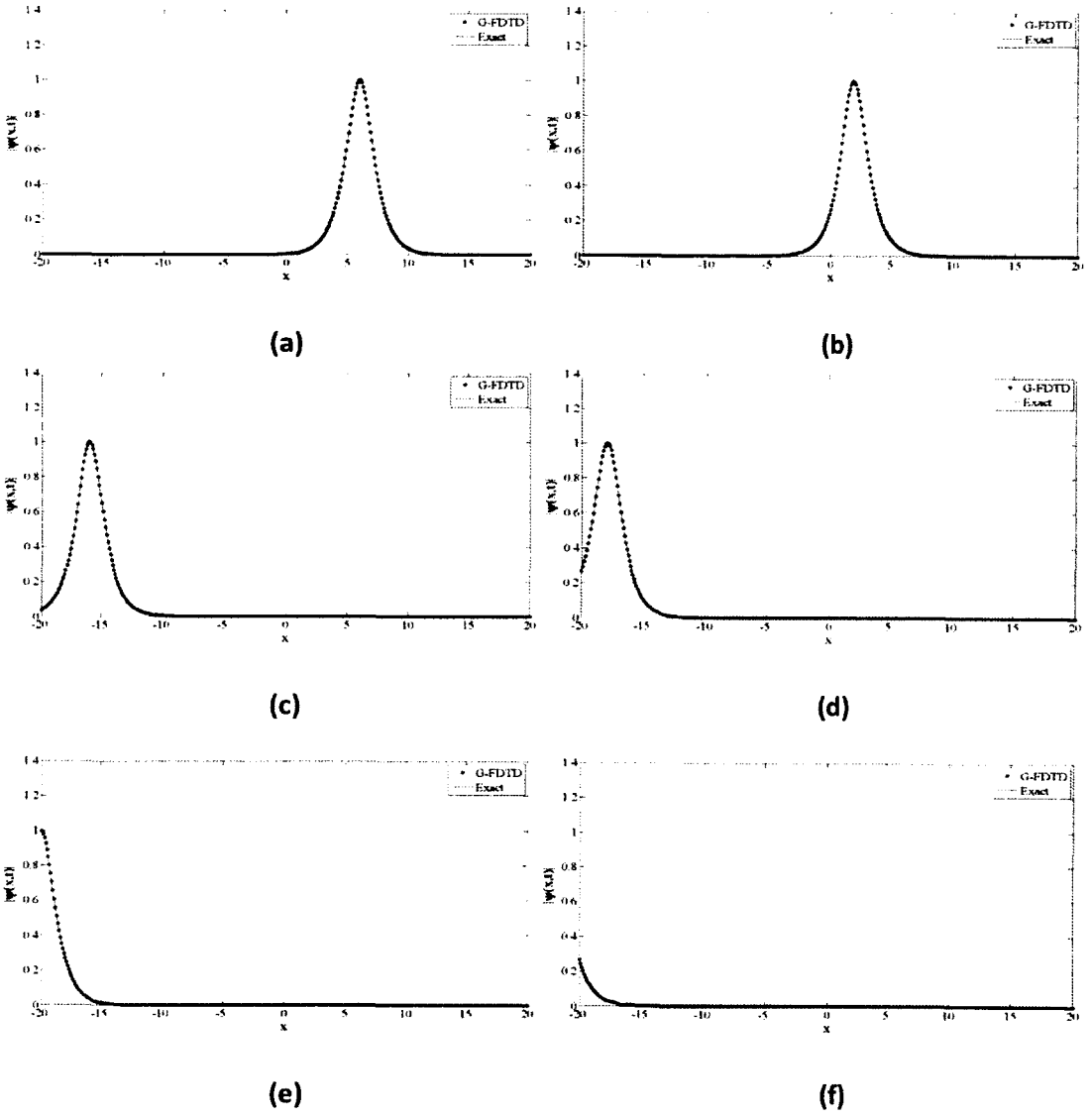


Fig. 3.4 Simulation of a bright soliton propagating in free space to the left, where the explicit G-FDTD scheme was employed with $\Delta t = 0.001$, $\Delta x = 0.1$ at **(a)** $t = 1$, **(b)** $t = 2$, **(c)** $t = 6.5$, **(d)** $t = 7$, **(e)** $t = 7.5$, and **(f)** $t = 8$.

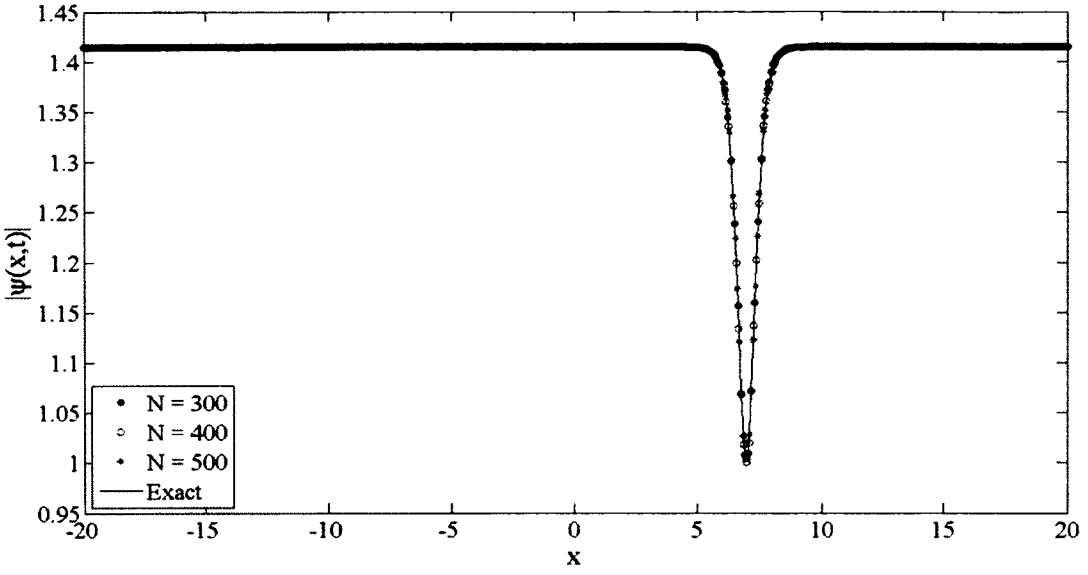
The second example is to consider a dark soliton propagation, where $\lambda = 2$ and $p = 3$ in

$$i\frac{\partial\psi(x,t)}{\partial t} - \frac{\partial^2\psi(x,t)}{\partial x^2} + \lambda|\psi(x,t)|^{p-1}\psi(x,t) = 0 \tag{3.20}$$

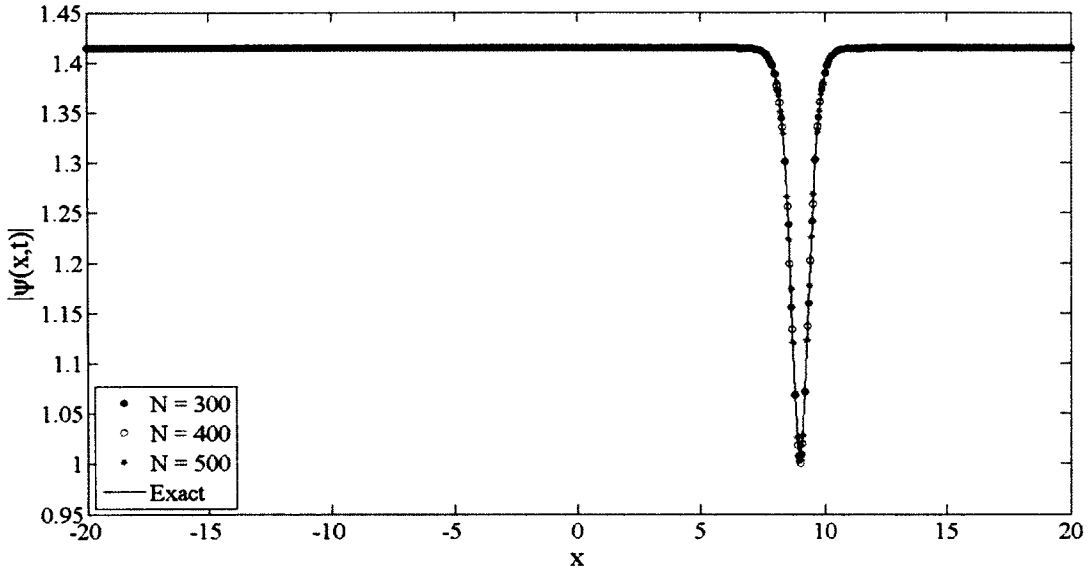
The initial condition was chosen such that the analytical solution is

$$\psi(x,t) = [tan\, h(2x - 10 - 4t) + i] \cdot e^{-i(8t)}. \tag{3.21}$$

The interval was taken to be $-20 \leq x \leq 20$ and the solution was defined to be zero outside the interval since the value of $\psi(x,t)$ is initially negligible there. We chose *M=1* in our scheme, Eq. (3.9), and the number of grid points to be 300, 400, and 500, respectively, with $\Delta t = 0.001$. Figure 3.5 illustrates the dark soliton propagation in free space at *t=1* and *2*, obtained using the three different meshes.



(a)



(b)

Fig. 3.5 Simulation of a dark soliton propagating in free space, where the explicit G-FDTD scheme was employed with $\Delta t = 0.001$ and three meshes at **(a)** $t = 1$, **(b)** $t = 2$.

The third example is to consider a bright soliton propagation in two spatial dimensions, where $\lambda = -4$ and $p = 3$ in

$$i\frac{\partial\psi(x,y,t)}{\partial t} - \frac{\partial^2\psi(x,y,t)}{\partial x^2} - \frac{\partial^2\psi(x,y,t)}{\partial y^2} + \lambda|\psi(x,y,t)|^{p-1}\psi(x,y,t) = 0. \tag{3.22}$$

For this case, we chose the initial condition such that the analytical solution is

$$\psi(x,y,t) = \operatorname{sech}(x + y + 10 - 8t) \cdot e^{-i(2x+2y+20-6t)}. \tag{3.23}$$

The interval was taken to be $-20 \leq x, y \leq 20$ and the solution was defined to be zero outside the interval since the value of $\psi(x,y,t)$ is initially negligible there. We chose *M=1* in our scheme, Eq. (3.10), and the number of grid points in both *x* and *y* to be 800 with $\Delta t = 0.001$. Figure 3.6 illustrates the two-dimensional bright soliton propagation in free space at various times. The maximum error between the analytical solution and the numerical solution within $0 \leq t \leq 5$ is 4.50512×10^{-4}.

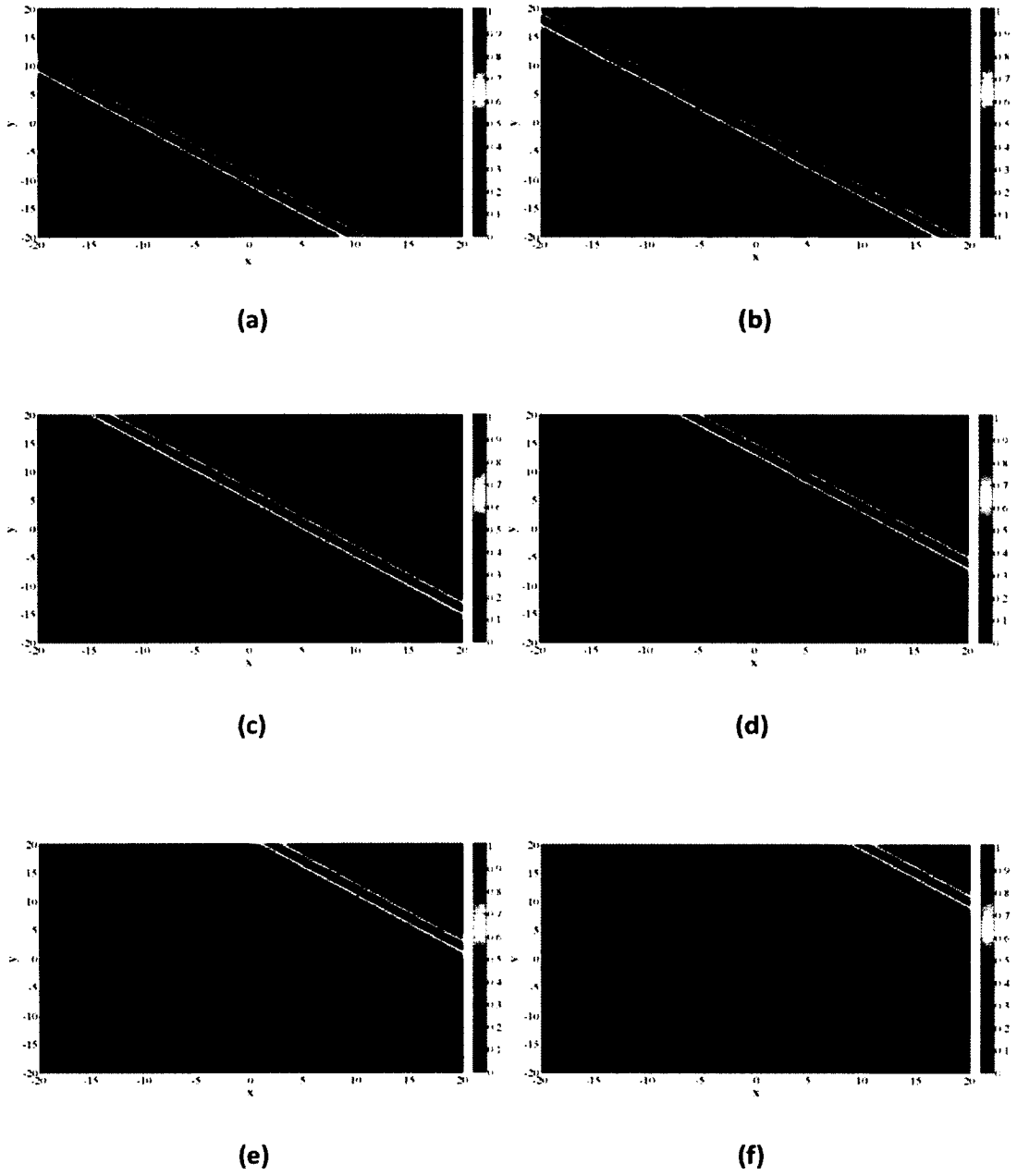


Fig. 3.6 Simulation of a 2D bright soliton propagating in free space, where the explicit G-FDTD scheme was employed with $\Delta t = 0.001$, $\Delta x = \Delta y = 0.05$ at **(a)** $t = 0$, **(b)** $t = 1$, **(c)** $t = 2$, **(d)** $t = 3$, **(e)** $t = 4$, and **(f)** $t = 5$.

CHAPTER FOUR

IMPLICIT G-FDTD METHOD FOR SOLVING THE NONLINEAR SCHRÖDINGER EQUATION

4.1 Implicit G-FDTD

Originally, we started to develop the implicit G-FDTD method instead of the explicit G-FDTD. The idea was that an implicit G-FDTD method would be unconditionally stable, and this would allow a large Δt to be chosen. However, the implicit G-FDTD scheme requires solving a linear system of equations. To solve this system of linear equations, iterations are required over a set of grid points. As this set of grid points increases, it becomes more computationally expensive to solve the NLSE numerically. Moreover, this difficulty is particularly evident in multiple space dimensions. For these reasons, we believe the explicit G-FDTD is a more computationally effective scheme. However, for purposes of completeness, it remains essential to also derive the new G-FDTD method implicitly.

To obtain the implicit G-FDTD scheme, we first assume that $\psi(x,t)$ be a sufficiently smooth function which vanishes for sufficiently large $|x|$. Using the idea of the generalized FDTD method in [63, 64, 65], we split the variable $\psi(x,t)$ into real and imaginary components,

$$\psi(x,t) = \psi_{real}(x,t) + i\psi_{imag}(x,t). \tag{4.1}$$

Inserting Eq. (4.1) into Eq. (2.1) and then separating the real and imaginary parts resulted in the following coupled set of equations:

$$\frac{\partial \psi_{real}(x,t)}{\partial t}$$

$$= \frac{\partial^2 \psi_{imag}(x,t)}{\partial x^2} + \lambda[\psi_{real}^2(x,t) + \psi_{imag}^2(x,t)]^{\frac{p-1}{2}}\psi_{imag}(x,t), \tag{4.2a}$$

$$\frac{\partial \psi_{imag}(x,t)}{\partial t}$$

$$= -\frac{\partial^2 \psi_{real}(x,t)}{\partial x^2} - \lambda[\psi_{real}^2(x,t) + \psi_{imag}^2(x,t)]^{\frac{p-1}{2}}\psi_{real}(x,t). \tag{4.2b}$$

We let $\psi_{real}^n(k)$ be the approximation of $\psi_{real}(k\Delta x, n\Delta t)$ and so on for $\psi_{imag}(k\Delta x, n\Delta t)$. Using Taylor series expansion at $t = (n-1/2)\Delta t$, and using the above equations repeatedly, where t in $[\psi_{real}^2(x,t) + \psi_{imag}^2(x,t)]^{\frac{p-1}{2}}$ is fixed at $(n-1/2)\Delta t$, we obtain

$$\begin{aligned}\psi_{real}^n(k) - \psi_{real}^{n-1}(k) &= \Delta t \frac{\partial \psi_{real}^{n-\frac{1}{2}}(k)}{\partial t} + \frac{\Delta t^3}{24}\frac{\partial^3 \psi_{real}^{n-\frac{1}{2}}(k)}{\partial t^3} + O(\Delta t^5) \\ &= \Delta t[\frac{\partial^2}{\partial x^2} + \left|\psi^{n-\frac{1}{2}}\right|^{p-1}]\psi_{imag}^{n-1/2}(k) \\ &\quad -\frac{\Delta t^3}{24}[\frac{\partial^2}{\partial x^2} + \left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^3\psi_{imag}^{n-1/2}(k) + O(\Delta t^5)\end{aligned} \tag{4.3a}$$

and

$$\begin{aligned}\psi_{imag}^n(k) - \psi_{imag}^{n-1}(k) &= \Delta t \frac{\partial \psi_{imag}^{n-\frac{1}{2}}(k)}{\partial t} + \frac{\Delta t^3}{24}\frac{\partial^3 \psi_{imag}^{n-\frac{1}{2}}(k)}{\partial t^3} + O(\Delta t^5) \\ &= -\Delta t[\frac{\partial^2}{\partial x^2} + \left|\psi^{n-\frac{1}{2}}\right|^{p-1}]\psi_{real}^{n-1/2}(k)\end{aligned}$$

$$+\frac{\Delta t^3}{24}\left[\frac{\partial^2}{\partial x^2}+\left|\psi^{n-\frac{1}{2}}\right|^{p-1}\right]^3\psi_{real}^{n-\frac{1}{2}}(k)+O(\Delta t^5), \quad (4.3b)$$

where $\left|\psi^{n-\frac{1}{2}}\right|^{p-1}=[\psi_{real}^2(k\Delta x,t_{n-1/2})+\psi_{imag}^2(k\Delta x,t_{n-1/2})]^{\frac{p-1}{2}}$. To evaluate $\psi_{imag}^{n-1/2}(k)$ in Eq. (4.3a), we further use the Taylor series expansion as

$$\psi_{imag}^{n}(k)-\psi_{imag}^{n-1/2}(k)$$

$$=\frac{\Delta t}{2}\frac{\partial\psi_{imag}^{n-\frac{1}{2}}(k)}{\partial t}+\frac{\Delta t^2}{8}\frac{\partial^2\psi_{imag}^{n-\frac{1}{2}}(k)}{\partial t^2}+\frac{\Delta t^3}{48}\frac{\partial^3\psi_{imag}^{n-\frac{1}{2}}(k)}{\partial t^3}+O(\Delta t^4), \quad (4.4a)$$

$$\psi_{imag}^{n-1}(k)-\psi_{imag}^{n-1/2}(k)$$

$$=-\frac{\Delta t}{2}\frac{\partial\psi_{imag}^{n-\frac{1}{2}}(k)}{\partial t}+\frac{\Delta t^2}{8}\frac{\partial^2\psi_{imag}^{n-\frac{1}{2}}(k)}{\partial t^2}-\frac{\Delta t^3}{48}\frac{\partial^3\psi_{imag}^{n-\frac{1}{2}}(k)}{\partial t^3}+O(\Delta t^4). \quad (4.4b)$$

Taking an average of Eqs. (4.4a) and (4.4b), we obtain

$$\psi_{imag}^{n-1/2}(k)=\frac{\psi_{imag}^{n}(k)+\psi_{imag}^{n-1}(k)}{2}-\frac{\Delta t^2}{8}\frac{\partial^2\psi_{imag}^{n-\frac{1}{2}}(k)}{\partial t^2}+O(\Delta t^4)$$

$$=\frac{\psi_{imag}^{n}(k)+\psi_{imag}^{n-1}(k)}{2}+\frac{\Delta t^2}{8}\left[\frac{\partial^2}{\partial x^2}+\left|\psi^{n-\frac{1}{2}}\right|^{p-1}\right]^2$$

$$\cdot\psi_{imag}^{n-\frac{1}{2}}(k)+O(\Delta t^4). \quad (4.5a)$$

Similarly, we have

$$\psi_{real}^{n-1/2}(k)=\frac{\psi_{real}^{n}(k)+\psi_{real}^{n-1}(k)}{2}-\frac{\Delta t^2}{8}\frac{\partial^2\psi_{real}^{n-\frac{1}{2}}(k)}{\partial t^2}+O(\Delta t^4)$$

$$=\frac{\psi_{real}^{n}(k)+\psi_{real}^{n-1}(k)}{2}-\frac{\Delta t^2}{8}\left[\frac{\partial^2}{\partial x^2}+\left|\psi^{n-\frac{1}{2}}\right|^{p-1}\right]^2$$

$$\cdot\psi_{real}^{n-\frac{1}{2}}(k)+O(\Delta t^4). \quad (4.5b)$$

Substituting Eqs. (4.5a) and (4.5b) into Eqs. (4.3a) and (4.3b), respectively, and keeping the terms up to $O(\Delta t^3)$, we obtain

$$\psi_{real}^{n}(k) - \psi_{real}^{n-1}(k) = \Delta t[\frac{\partial^2}{\partial x^2} + \left|\psi^{n-\frac{1}{2}}\right|^{p-1}]\frac{\psi_{imag}^{n}(k) + \psi_{imag}^{n-1}(k)}{2}$$

$$+\frac{\Delta t^3}{12}[\frac{\partial^2}{\partial x^2} + \left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^3$$

$$\cdot\frac{\psi_{imag}^{n}(k) + \psi_{imag}^{n-1}(k)}{2} + O(\Delta t^5), \tag{4.6a}$$

$$\psi_{imag}^{n}(k) - \psi_{imag}^{n-1}(k) = -\Delta t[\frac{\partial^2}{\partial x^2} + \left|\psi^{n-\frac{1}{2}}\right|^{p-1}]\frac{\psi_{real}^{n}(k) + \psi_{real}^{n-1}(k)}{2}$$

$$-\frac{\Delta t^3}{12}[\frac{\partial^2}{\partial x^2} + \left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^3$$

$$\cdot\frac{\psi_{real}^{n}(k) + \psi_{real}^{n-1}(k)}{2} + O(\Delta t^5). \tag{4.6b}$$

Noting that the term $\left|\psi^{n-\frac{1}{2}}\right|^{p-1}$ in Eq. (4.6) needs to be evaluated, we use a similar argument for Eq. (4.6) and obtain

$$\psi_{real}^{n+1/2}(k) - \psi_{real}^{n-1/2}(k)$$

$$= \Delta t[\frac{\partial^2}{\partial x^2} + |\psi^n|^{p-1}]\frac{\psi_{imag}^{n+1/2}(k) + \psi_{imag}^{n-1/2}(k)}{2}$$

$$+\frac{\Delta t^3}{12}[\frac{\partial^2}{\partial x^2} + |\psi^n|^{p-1}]^3\frac{\psi_{imag}^{n+\frac{1}{2}}(k) + \psi_{imag}^{n-\frac{1}{2}}(k)}{2} + O(\Delta t^5), \tag{4.7a}$$

$$\psi_{imag}^{n+1/2}(k) - \psi_{imag}^{n-1/2}(k)$$

$$= -\Delta t[\frac{\partial^2}{\partial x^2} + |\psi^n|^{p-1}] \frac{\psi_{real}^{n+1/2}(k) + \psi_{real}^{n-1/2}(k)}{2}$$

$$-\frac{\Delta t^3}{12}[\frac{\partial^2}{\partial x^2} + |\psi^n|^{p-1}]^3 \frac{\psi_{real}^{n+\frac{1}{2}}(k) + \psi_{real}^{n-\frac{1}{2}}(k)}{2} + O(\Delta t^5), \tag{4.7b}$$

where $|\psi^n|^{p-1} = [\psi_{real}^2(k\Delta x, t_n) + \psi_{imag}^2(k\Delta x, t_n)]^{\frac{p-1}{2}}$. Next, we couple Eqs. (4.6) and (4.7) together, dropping out the truncation error $O(\Delta t^5)$, and replacing $\frac{\partial^2}{\partial x^2}$ with a second-order accurate central difference operator, $\delta_x^2 u(k) \approx \frac{1}{\Delta x^2}[u(k+1) - 2u(k) + u(k-1)]$, or a fourth-order accurate central difference operator, $\frac{1}{\Delta x^2} D_x^2 u(k) \approx \frac{1}{12\Delta x^2}[-u(k+2) + 16u(k+1) - 30u(k) + 16u(k-1) - u(k-2)]$. This results in our G-FDTD scheme for solving the NLSE as follows:

$$\psi_{real}^n(k) - \psi_{real}^{n-1}(k) = \{[\sigma D_x^2 + \Delta t|\psi^{n-1/2}|^{p-1}]$$

$$+\frac{1}{12}[\sigma D_x^2 + \Delta t\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^3\}$$

$$\cdot\frac{\psi_{imag}^n(k) + \psi_{imag}^{n-1}(k)}{2}, \tag{4.8a}$$

$$\psi_{imag}^n(k) - \psi_{imag}^{n-1}(k) = -\{[\sigma D_x^2 + \Delta t|\psi^{n-1/2}|^{p-1}]$$

$$+\frac{1}{12}[\sigma D_x^2 + \Delta t\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^3\}$$

$$\cdot\frac{\psi_{real}^n(k) + \psi_{real}^{n-1}(k)}{2}; \tag{4.8b}$$

$$\psi_{real}^{n+1/2}(k) - \psi_{real}^{n-1/2}(k) = \{[\sigma D_x^2 + \Delta t|\psi^n|^{p-1}]$$

$$+\frac{1}{12}[\sigma D_x^2 + \Delta t|\psi^n|^{p-1}]^3\}$$

$$\cdot \frac{\psi_{imag}^{n+\frac{1}{2}}(k) + \psi_{imag}^{n-\frac{1}{2}}(k)}{2}, \tag{4.9a}$$

$$\psi_{imag}^{n+1/2}(k) - \psi_{imag}^{n-1/2}(k) = -\{[\sigma D_x^2 + \Delta t|\psi^n|^{p-1}]$$

$$+\frac{1}{12}[\sigma D_x^2 + \Delta t|\psi^n|^{p-1}]^3\}$$

$$\cdot \frac{\psi_{real}^{n+\frac{1}{2}}(k) + \psi_{real}^{n-\frac{1}{2}}(k)}{2}. \tag{4.9b}$$

Here, $\sigma = \Delta t/\Delta x^2$. The truncation error of the above scheme is $O(\Delta x^2 + \Delta t^5)$ if δ_x^2 is used and $O(\Delta x^4 + \Delta t^5)$ if D_x^2 is used, as compared with Eqs. (4.6) and (4.7). It should be noted that the partial derivative $\frac{\partial^2}{\partial x^2}$ can alternatively be approximated using a spectral or other higher-order method. In this study, we confine our attention to the finite difference method with central difference approximations. It can be seen that in order to obtain $\psi_{real}^n(k)$ and $\psi_{imag}^n(k)$, one must solve Eqs. (4.8a) and (4.8b) together. This is a linear system, which must be solved iteratively. Due to this requirement, the implicit G-FDTD scheme will be computationally slower than the explicit G-FDTD, most notably in multiple space dimensions.

4.2 Discrete Conservation Law for the Second-Order Implicit G-FDTD

The first step is to take Eq. (4.8a) $\times \frac{\psi_{real}^n(k)+\psi_{real}^{n-1}(k)}{2}$ + Eq. (4.8b) $\times \frac{\psi_{imag}^n(k)+\psi_{imag}^{n-1}(k)}{2}$, then sum k over all integers Z. This gives

$$\frac{1}{2}\sum_{k\in Z}\{[\psi_{real}^n(k)]^2 - [\psi_{real}^{n-1}(k)]^2 + [\psi_{imag}^n(k)]^2 - [\psi_{imag}^{n-1}(k)]^2\}$$

$$= \sum_{k\in Z}\{[\sigma\delta_x^2 + \Delta t\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]\frac{\psi_{imag}^n(k) + \psi_{imag}^{n-1}(k)}{2}$$

$$\cdot \frac{\psi_{real}^{n}(k)+\psi_{real}^{n-1}(k)}{2}$$

$$-[\sigma\delta_x^2+\Delta t\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]\frac{\psi_{real}^{n}(k)+\psi_{real}^{n-1}(k)}{2}\cdot\frac{\psi_{imag}^{n}(k)+\psi_{imag}^{n-1}(k)}{2}\}$$

$$+\sum_{k\in Z}\{\frac{1}{12}[\sigma\delta_x^2+\Delta t\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^3\frac{\psi_{imag}^{n}(k)+\psi_{imag}^{n-1}(k)}{2}$$

$$\cdot\frac{\psi_{real}^{n}(k)+\psi_{real}^{n-1}(k)}{2}$$

$$-\frac{1}{12}[\sigma\delta_x^2+\Delta t\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^3\frac{\psi_{real}^{n}(k)+\psi_{real}^{n-1}(k)}{2}$$

$$\cdot\frac{\psi_{imag}^{n}(k)+\psi_{imag}^{n-1}(k)}{2}\}. \tag{4.10}$$

Using the property

$$\sum_{k\in Z}\delta_x^2\,u(k)\cdot v(k)=-\sum_{k\in Z}\nabla_x u(k)\cdot\nabla_x v(k), \tag{4.11}$$

where $\nabla_x u(k)=u(k+2)-2u(k+1)+u(k)$. We obtain that

$$\sum_{k\in Z}\delta_x^2[\psi_{imag}^{n}(k)+\psi_{imag}^{n-1}(k)]\cdot[\psi_{real}^{n}(k)+\psi_{real}^{n-1}(k)]$$

$$=-\sum_{k\in Z}\nabla_x[\psi_{imag}^{n}(k)+\psi_{imag}^{n-1}(k)]\cdot\nabla_x[\psi_{real}^{n}(k)+\psi_{real}^{n-1}(k)]$$

$$=\sum_{k\in Z}\delta_x^2[\psi_{real}^{n}(k)+\psi_{real}^{n-1}(k)]\cdot[\psi_{imag}^{n}(k)+\psi_{imag}^{n-1}(k)], \tag{4.12a}$$

$$\sum_{k\in Z}(\delta_x^2)^2[\psi_{imag}^{n}(k)+\psi_{imag}^{n-1}(k)]\cdot[\psi_{real}^{n}(k)+\psi_{real}^{n-1}(k)]$$

$$=\sum_{k\in Z}\delta_x^2[\psi_{imag}^{n}(k)+\psi_{imag}^{n-1}(k)]\cdot\delta_x^2[\psi_{real}^{n}(k)+\psi_{real}^{n-1}(k)], \tag{4.12b}$$

$$\sum_{k\in Z}(\delta_x^2)^3[\psi_{imag}^{n}(k)+\psi_{imag}^{n-1}(k)]\cdot[\psi_{real}^{n}(k)+\psi_{real}^{n-1}(k)]$$

$$= -\sum_{k\in Z} (\nabla_x)^3 \left[\psi^n_{imag}(k) + \psi^{n-1}_{imag}(k)\right] \cdot (\nabla_x)^3 \left[\psi^n_{real}(k) + \psi^{n-1}_{real}(k)\right]$$

$$= \sum_{k\in Z} \left[\psi^n_{imag}(k) + \psi^{n-1}_{imag}(k)\right] \cdot (\delta^2_x)^3 \left[\psi^n_{real}(k) + \psi^{n-1}_{real}(k)\right]. \quad (4.12c)$$

Hence, the RHS of Eq. (4.10) is zero, and

$$\sum_{k\in Z} \left\{\left[\psi^n_{real}(k)\right]^2 + \left[\psi^n_{imag}(k)\right]^2\right\}$$

$$= \sum_{k\in Z} \left\{\left[\psi^{n-1}_{real}(k)\right]^2 + \left[\psi^{n-1}_{imag}(k)\right]^2\right\} = const. \quad (4.13)$$

It should be noted that we treat $\left|\psi^{n-\frac{1}{2}}\right|^{p-1}$ as a constant with respect to k for the sake of simplicity. If $\left|\psi^{n-\frac{1}{2}}\right|^{p-1}$ is considered to be dependent on k, one may obtain Eq. (4.13) using a similar argument. Eq. (4.13) indicates that the scheme satisfies the first conservation law. Similarly, we can reason that Eq. (4.9) also satisfies the first conservation law.

The second step is to take Eq. (4.8a) $\times 2\left[\psi^n_{imag}(k) + \psi^{n-1}_{imag}(k)\right]$ − Eq. (4.8b) $\times 2\left[\psi^n_{real}(k) + \psi^{n-1}_{real}(k)\right]$ + Eq. (4.8a) $\times 2\left[\psi^{n+1/2}_{imag}(k) + \psi^{n-1/2}_{imag}(k)\right]$ − Eq. (4.8b) $\times 2\left[\psi^{n+1/2}_{real}(k) + \psi^{n-1/2}_{real}(k)\right]$, then sum k over all integers Z. This gives

$$\sum_{k\in Z} \{-\sigma\left[\nabla_x \psi^n_{imag}(k)\right]^2 + \sigma\left[\nabla_x \psi^{n-1}_{imag}(k)\right]^2 - \sigma\left[\nabla_x \psi^n_{real}(k)\right]^2$$

$$+\sigma\left[\nabla_x \psi^{n-1}_{real}(k)\right]^2 + \Delta t \left|\psi^{n-\frac{1}{2}}\right|^{p-1} \left[\nabla_x \psi^n_{real}(k)\right]^2$$

$$-\Delta t \left|\psi^{n-\frac{1}{2}}\right|^{p-1} \left[\nabla_x \psi^{n-1}_{real}(k)\right]^2$$

$$+\Delta t \left|\psi^{n-\frac{1}{2}}\right|^{p-1} \left[\nabla_x \psi^n_{imag}(k)\right]^2 - \Delta t \left|\psi^{n-\frac{1}{2}}\right|^{p-1} \left[\nabla_x \psi^{n-1}_{imag}(k)\right]^2\}$$

$$+\frac{1}{12}\sum_{k\in Z}\{-\sigma^3\left[(\nabla_x)^3\psi_{imag}^{n+1/2}(k)\right]^2+\sigma^3\left[(\nabla_x)^3\psi_{imag}^{n-1/2}(k)\right]^2$$

$$+3\sigma^2|\psi^n|^{p-1}\left[(\nabla_x)^2\psi_{imag}^{n+1/2}(k)\right]^2-3\sigma^2|\psi^n|^{p-1}\left[(\nabla_x)^2\psi_{imag}^{n-1/2}(k)\right]^2$$

$$-3\sigma|\psi^n|^{2(p-1)}[\nabla_x\psi_{imag}^{n+\frac{1}{2}}(k)]^2+3\sigma|\psi^n|^{2(p-1)}\left[\nabla_x\psi_{imag}^{n-1/2}(k)\right]^2$$

$$+|\psi^n|^{3(p-1)}[\psi_{imag}^{n+\frac{1}{2}}(k)]^2-|\psi^n|^{3(p-1)}[\psi_{imag}^{n-\frac{1}{2}}(k)]^2\}$$

$$+\frac{1}{12}\sum_{k\in Z}\{-\sigma^3\left[(\nabla_x)^3\psi_{real}^{n+1/2}(k)\right]^2+\sigma^3\left[(\nabla_x)^3\psi_{real}^{n-1/2}(k)\right]^2$$

$$+3\sigma^2|\psi^n|^{p-1}\left[(\nabla_x)^2\psi_{real}^{n+1/2}(k)\right]^2-3\sigma^2|\psi^n|^{p-1}\left[(\nabla_x)^2\psi_{real}^{n-1/2}(k)\right]^2$$

$$-3\sigma|\psi^n|^{2(p-1)}\left[\nabla_x\psi_{real}^{n+1/2}(k)\right]^2+3\sigma|\psi^n|^{2(p-1)}\left[\nabla_x\psi_{real}^{n-1/2}(k)\right]^2$$

$$+|\psi^n|^{3(p-1)}\left[\psi_{real}^{n+1/2}(k)\right]^2-|\psi^n|^{3(p-1)}\left[\psi_{real}^{n-1/2}(k)\right]^2\}=0. \tag{4.14}$$

That is,

$$\sum_{k\in Z}\{\sigma\left[\nabla_x\psi_{imag}^{n}(k)\right]^2+\sigma[\nabla_x\psi_{real}^{n}(k)]^2+\sigma\left[\nabla_x\psi_{imag}^{n+1/2}(k)\right]^2$$

$$+\sigma\left[\nabla_x\psi_{real}^{n+1/2}(k)\right]^2$$

$$+\Delta t\left|\psi^{n-\frac{1}{2}}\right|^{p-1}[\nabla_x\psi_{real}^{n}(k)]^2+\Delta t\left|\psi^{n-\frac{1}{2}}\right|^{p-1}\left[\nabla_x\psi_{imag}^{n}(k)\right]^2$$

$$+\frac{\sigma^3}{12}\left[(\nabla_x)^3\psi_{imag}^{n}(k)\right]^2+\frac{\sigma^3}{12}[(\nabla_x)^3\psi_{real}^{n}(k)]^2$$

$$+\frac{\sigma^3}{12}\left[(\nabla_x)^3\psi_{imag}^{n+1/2}(k)\right]^2$$

$$+\frac{\sigma^3}{12}\left[(\nabla_x)^3\psi_{real}^{n+1/2}(k)\right]^2-\frac{3\sigma^2}{12}\left|\psi^{n-\frac{1}{2}}\right|^{p-1}([(\nabla_x)^2\psi_{imag}^{n}(k)]^2$$

$$+[(\nabla_x)^2\psi_{real}^{n}(k)]^2)$$

$$-\frac{3\sigma^2}{12}|\psi^n|^{p-1}\left(\left[(\nabla_x)^2\psi_{imag}^{n+1/2}(k)\right]^2+\left[(\nabla_x)^2\psi_{real}^{n+1/2}(k)\right]^2\right)$$

$$+3\sigma\left|\psi^{n-1/2}\right|^{2(p-1)}\left([\nabla_x\psi_{real}^{n}(k)]^2+\left[\nabla_x\psi_{imag}^{n}(k)\right]^2\right)$$

$$+3\sigma|\psi^n|^{3(p-1)}\left(\left[\nabla_x\psi_{real}^{n+1/2}(k)\right]^2+\left[\nabla_x\psi_{imag}^{n+1/2}(k)\right]^2\right)$$

$$-\left|\psi^{n-1/2}\right|^{3(p-1)}\left([\psi_{real}^{n}(k)]^2+\left[\psi_{imag}^{n}(k)\right]^2\right)$$

$$-\left|\psi^{n-\frac{1}{2}}\right|^{3(p-1)}\left(\left[\psi_{real}^{n+\frac{1}{2}}(k)\right]^2+\left[\psi_{imag}^{n+\frac{1}{2}}(k)\right]^2\right)\}$$

$$=\sum_{k\in Z}\{\sigma\left[\nabla_x\psi_{imag}^{n-1}(k)\right]^2+\sigma[\nabla_x\psi_{real}^{n-1}(k)]^2+\sigma\left[\nabla_x\psi_{imag}^{n-1/2}(k)\right]^2$$

$$+\sigma\left[\nabla_x\psi_{real}^{n-1/2}(k)\right]^2$$

$$+\Delta t\left|\psi^{n-\frac{1}{2}}\right|^{p-1}[\nabla_x\psi_{real}^{n-1}(k)]^2+\Delta t\left|\psi^{n-\frac{1}{2}}\right|^{p-1}\left[\nabla_x\psi_{imag}^{n-1}(k)\right]^2$$

$$+\frac{\sigma^3}{12}\left[(\nabla_x)^3\psi_{imag}^{n-1}(k)\right]^2+\frac{\sigma^3}{12}[(\nabla_x)^3\psi_{real}^{n-1}(k)]^2$$

$$+\frac{\sigma^3}{12}\left[(\nabla_x)^3\psi_{imag}^{n-1/2}(k)\right]^2$$

$$+\frac{\sigma^3}{12}\left[(\nabla_x)^3\psi_{real}^{n-1/2}(k)\right]^2-\frac{3\sigma^2}{12}\left|\psi^{n-\frac{1}{2}}\right|^{p-1}\left(\left[(\nabla_x)^2\psi_{imag}^{n-1}(k)\right]^2\right.$$

$$\left.+[(\nabla_x)^2\psi_{real}^{n-1}(k)]^2\right)$$

$$-\frac{3\sigma^2}{12}|\psi^n|^{p-1}\left(\left[(\nabla_x)^2\psi_{imag}^{n-1/2}(k)\right]^2+\left[(\nabla_x)^2\psi_{real}^{n-1/2}(k)\right]^2\right)$$

$$+3\sigma\left|\psi^{n-1/2}\right|^{2(p-1)}\left([\nabla_x\psi_{real}^{n-1}(k)]^2+\left[\nabla_x\psi_{imag}^{n-1}(k)\right]^2\right)$$

$$+3\sigma|\psi^n|^{3(p-1)}\left(\left[\nabla_x\psi_{real}^{n-1/2}(k)\right]^2+\left[\nabla_x\psi_{imag}^{n-1/2}(k)\right]^2\right)$$

$$-\left|\psi^{n-1/2}\right|^{3(p-1)}\left([\psi_{real}^{n-1}(k)]^2+\left[\psi_{imag}^{n-1}(k)\right]^2\right)$$

$$-\left|\psi^{n-\frac{1}{2}}\right|^{3(p-1)}([\psi_{real}^{n-\frac{1}{2}}(k)]^2+[\psi_{imag}^{n-\frac{1}{2}}(k)]^2)\}. \tag{4.15}$$

This is a discrete analogous form of the second conservation law for any n.

4.3 Numerical Examples for the Second-Order Implicit G-FDTD

To test the accuracy of our proposed numerical scheme, we chose $\lambda = -2$ and $p = 3$ in

$$i\frac{\partial\psi(x,t)}{\partial t}-\frac{\partial^2\psi(x,t)}{\partial x^2}+\lambda|\psi(x,t)|^{p-1}\psi(x,t)=0 \tag{4.16}$$

An interval of $-20 \le x \le 20$ is chosen for second-order accuracy to avoid numerical oscillations at the boundaries. The first example is to consider a single soliton propagation where the exact solution is given by

$$\psi(x,t)=\operatorname{sech}(x+10-4t)\cdot e^{-i(2x+20-3t)}. \tag{4.17}$$

Figures 4.1-4.4 illustrate a soliton propagation in free space obtained using $N = 400$, 800, and 1600, respectively. In our computation, we chose a second-order central difference operator $\delta_x^2 u(k) \approx \frac{1}{\Delta x^2}[u(k+1)-2u(k)+u(k-1)]$. The operator chosen will determine the order of accuracy in space. The number of grid points are taken to be 400, 800, and 1600, depending on the prefered order of accuracy. The time step is chosen to be $\Delta t = 0.0001$. Maximum errors between the numerical solution and the exact solution when $0 \le t \le 1$ are listed in Table 4.1. The maximum errors clearly show second-order accuracy in the spatial direction and fifth-order accuracy in time.

Table 4.1 Maximum error obtained using a second-order implicit G-FDTD for a single soliton propagation when $0 \leq t \leq 1$, $\Delta t = 0.0001$.

Grid Points	Second-Order (δ_x^2) Maximum Error	Rate of Convergence
400	5.05119×10^{-2}	
800	1.25967×10^{-2}	2.00357
1600	3.18888×10^{-3}	1.98193

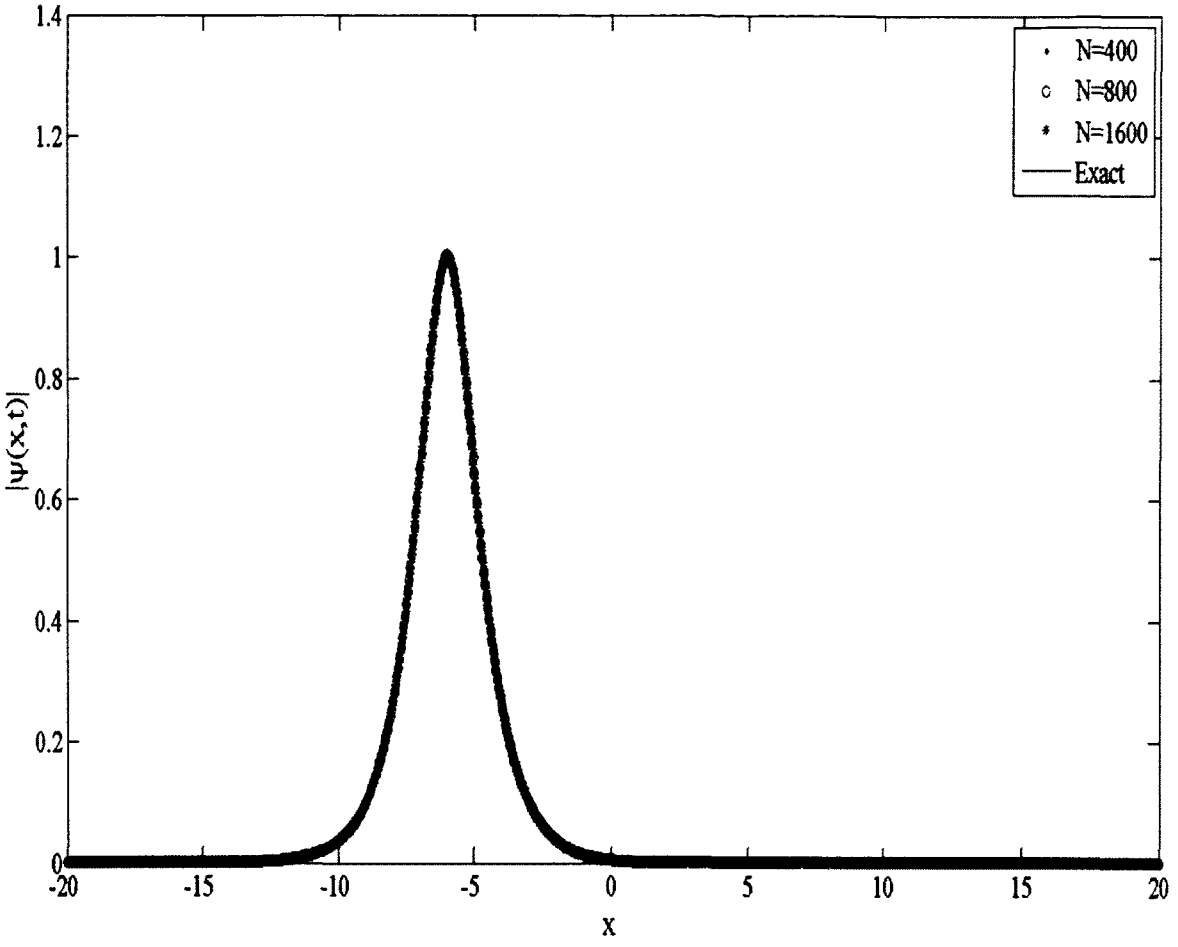


Fig. 4.1 Simulation of a soliton propagating in free space at $t = 1$. A second-order central difference operator δ_x^2 is employed.

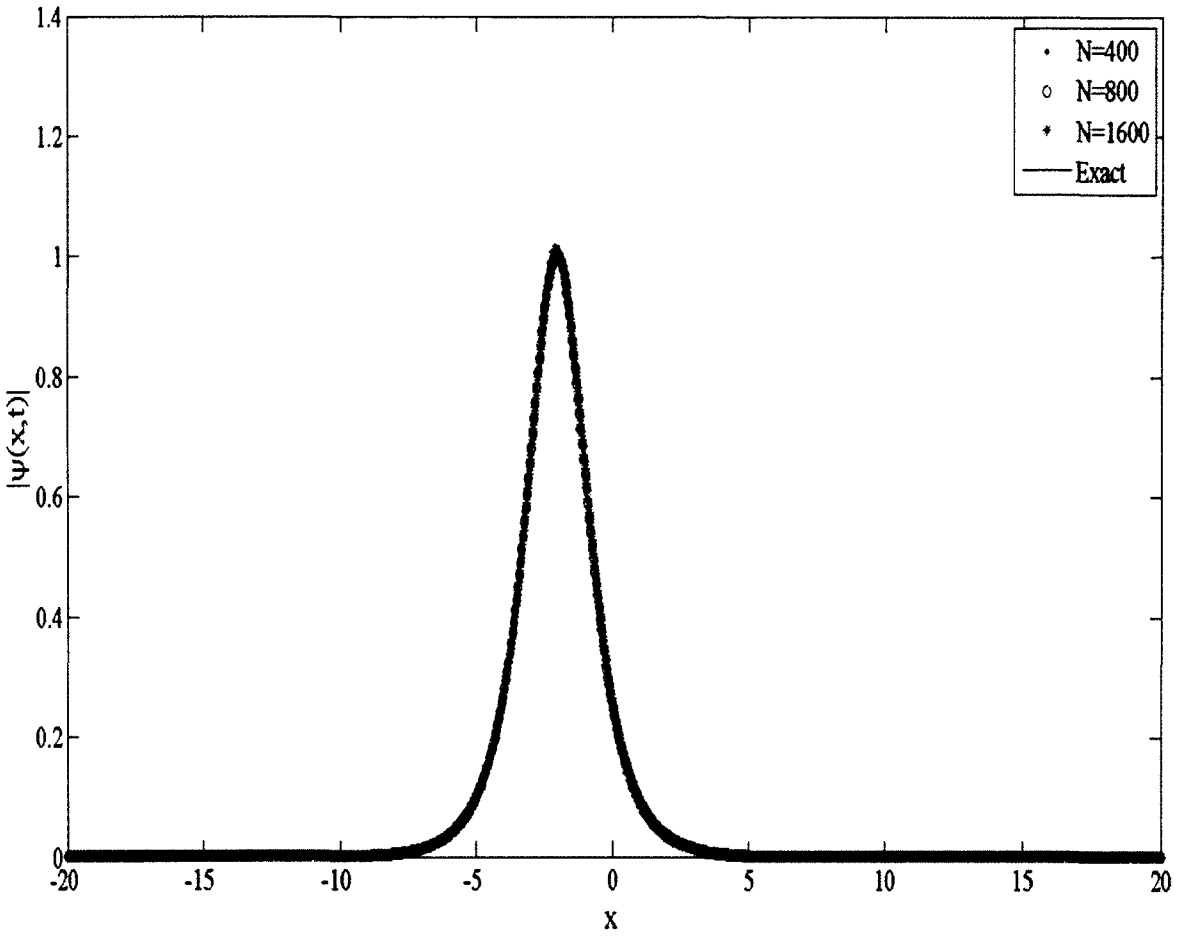


Fig. 4.2 Simulation of a soliton propagating in free space at $t = 2$. A second-order central difference operator δ_x^2 is employed.

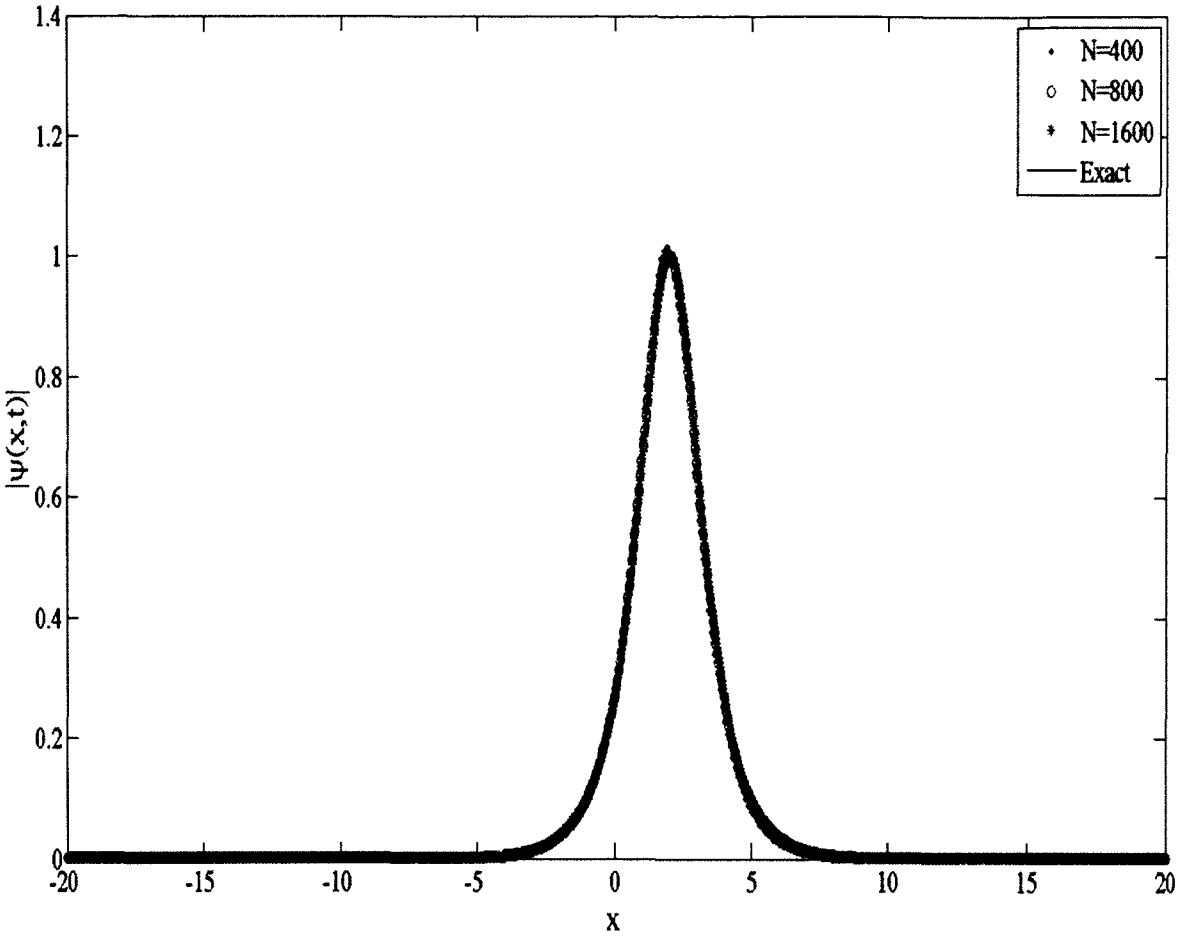


Fig. 4.3 Simulation of a soliton propagating in free space at $t = 3$. A second-order central difference operator δ_x^2 is employed.

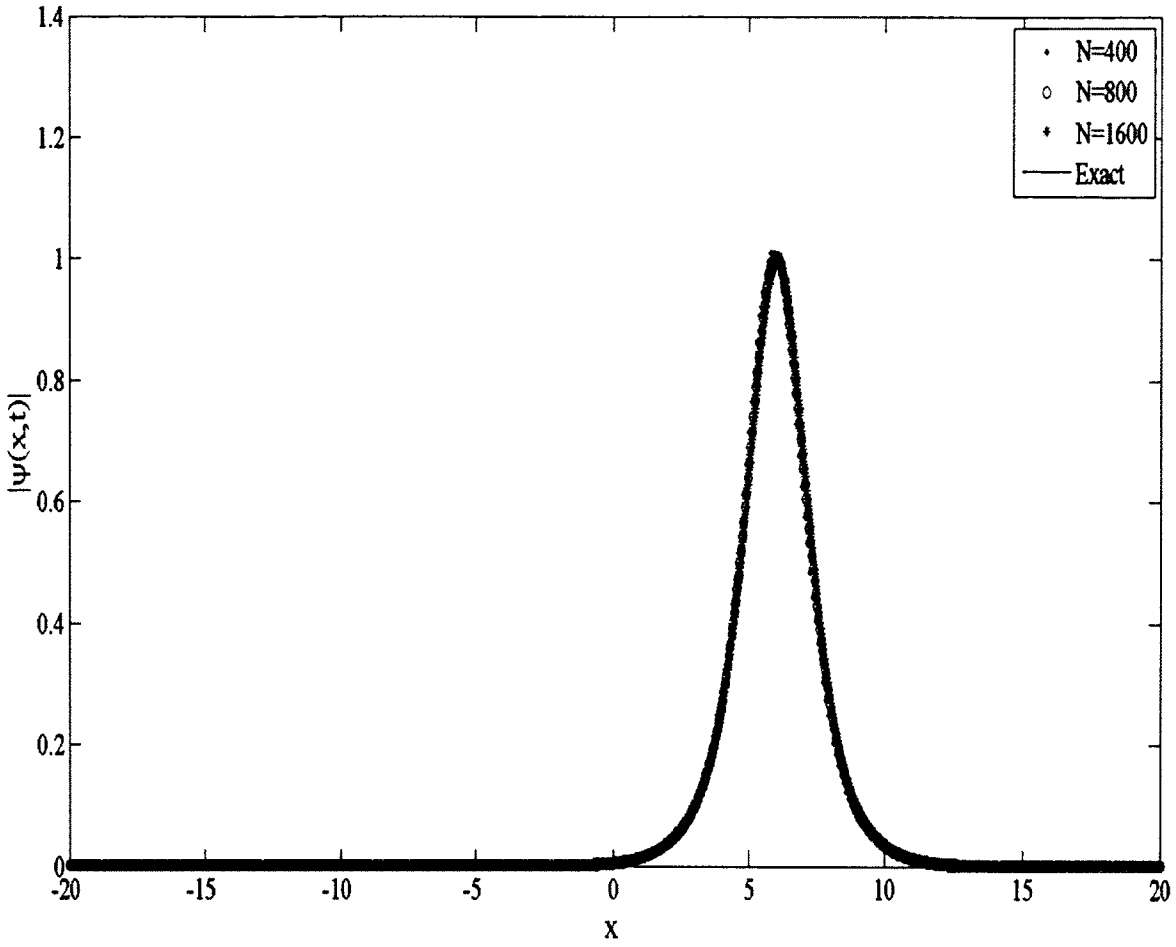


Fig. 4.4 Simulation of a soliton propagating in free space at $t = 4$. A second-order central difference operator δ_x^2 is employed.

The second example is to consider two-soliton collision where the exact solution for

$$i\frac{\partial\psi(x,t)}{\partial t}-\frac{\partial^2\psi(x,t)}{\partial x^2}+\lambda|\psi(x,t)|^{p-1}\psi(x,t)=0 \tag{4.18}$$

is

$$\psi(x,t)=\operatorname{sech}(x+10-4t)\cdot e^{-i(2x+20-3t)} + \operatorname{sech}(x-10+4t)\cdot e^{i(2x-20+3t)}. \tag{4.19}$$

Figures 4.5-4.8 illustrate a two-soliton collision in free space obtained using $N = 400$, 800, and 1600, respectively. In our computation, we chose a second-order central difference operator $\delta_x^2 u(k) \approx \frac{1}{\Delta x^2}[u(k+1)-2u(k)+u(k-1)]$. The operator chosen will determine the order of accuracy in space. The number of grid points are taken to be 400, 800, and 1600, depending on the preferred order of accuracy. The time step is chosen to be $\Delta t = 0.0001$.

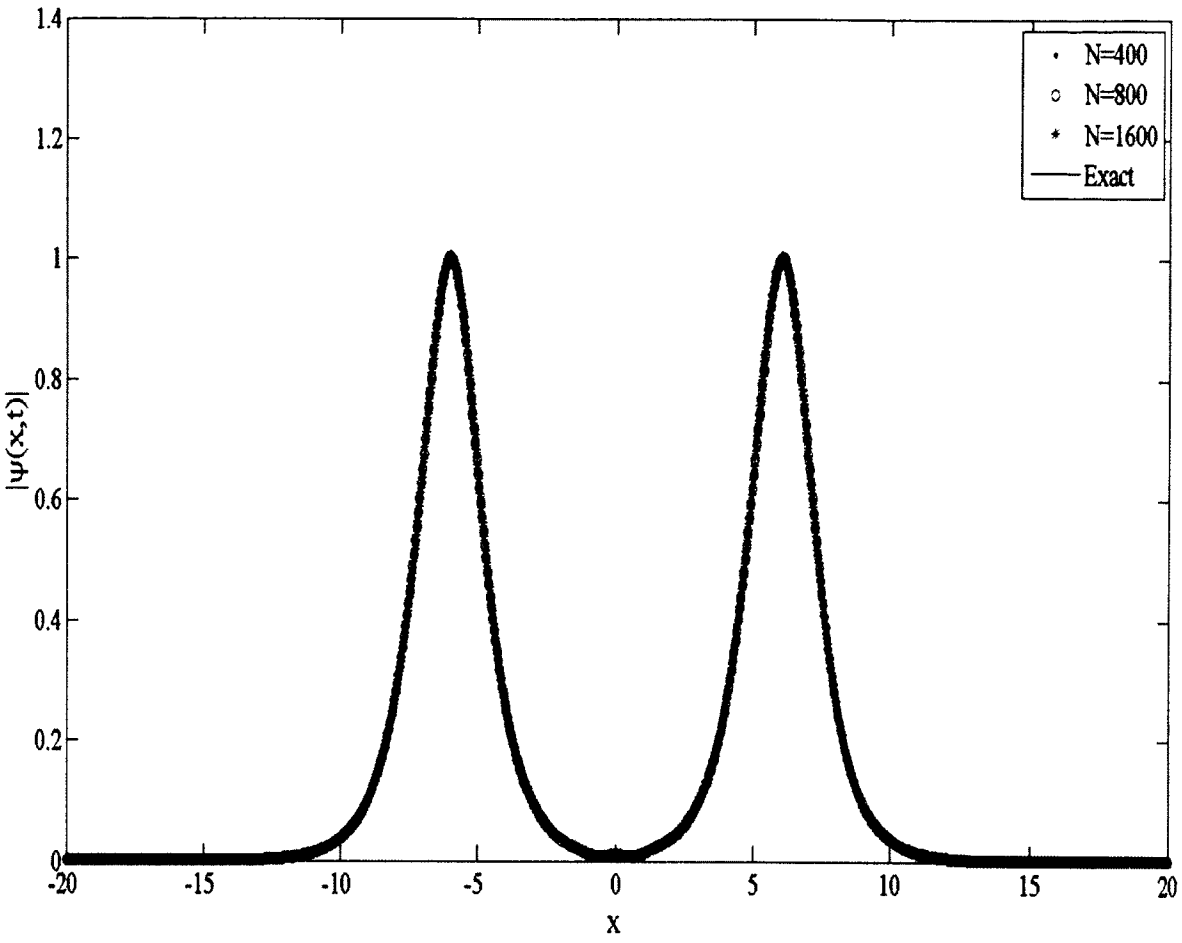


Fig. 4.5 Simulation of a soliton collision in free space at $t = 1$. A second-order central difference operator δ_x^2 is employed.

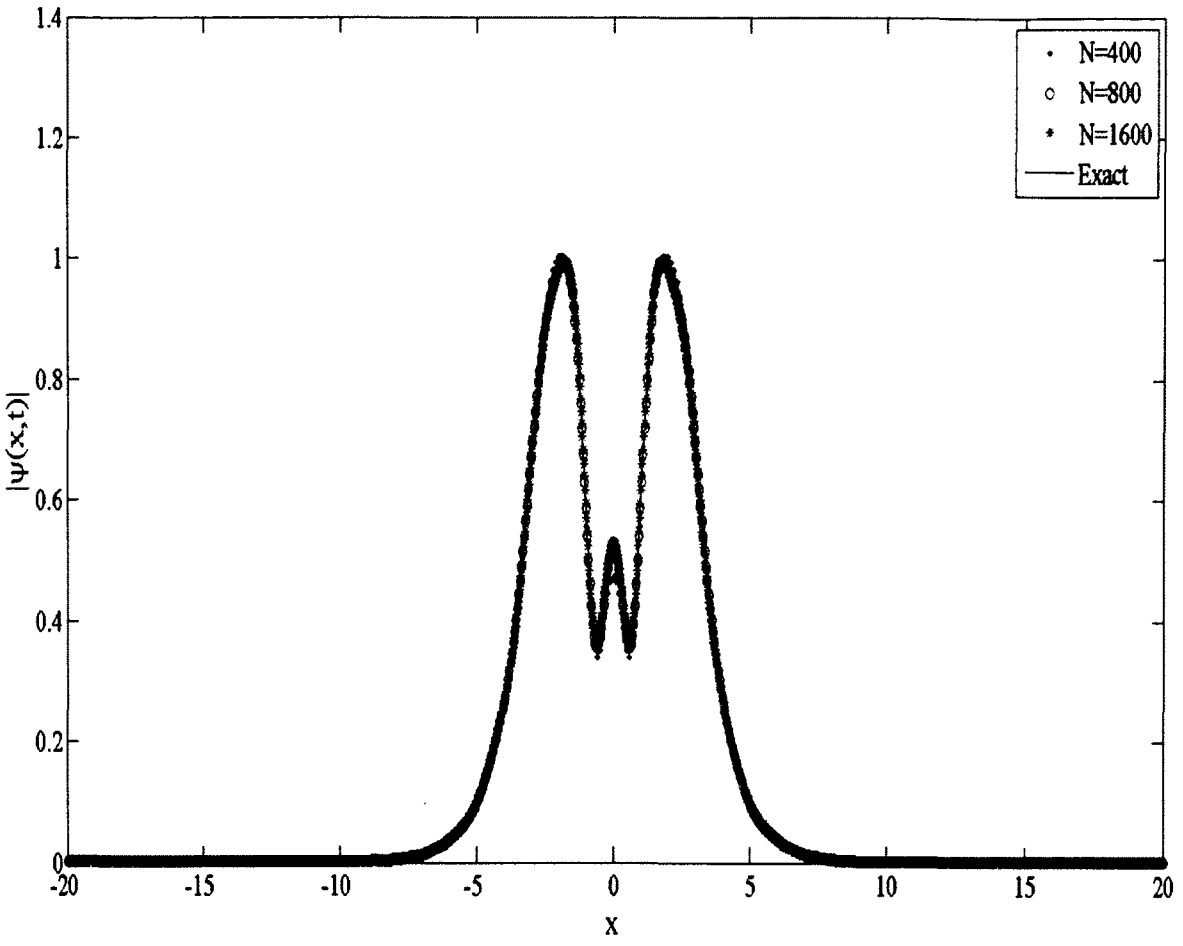


Fig. 4.6 Simulation of a soliton collision in free space at $t = 2$. A second-order central difference operator δ_x^2 is employed.

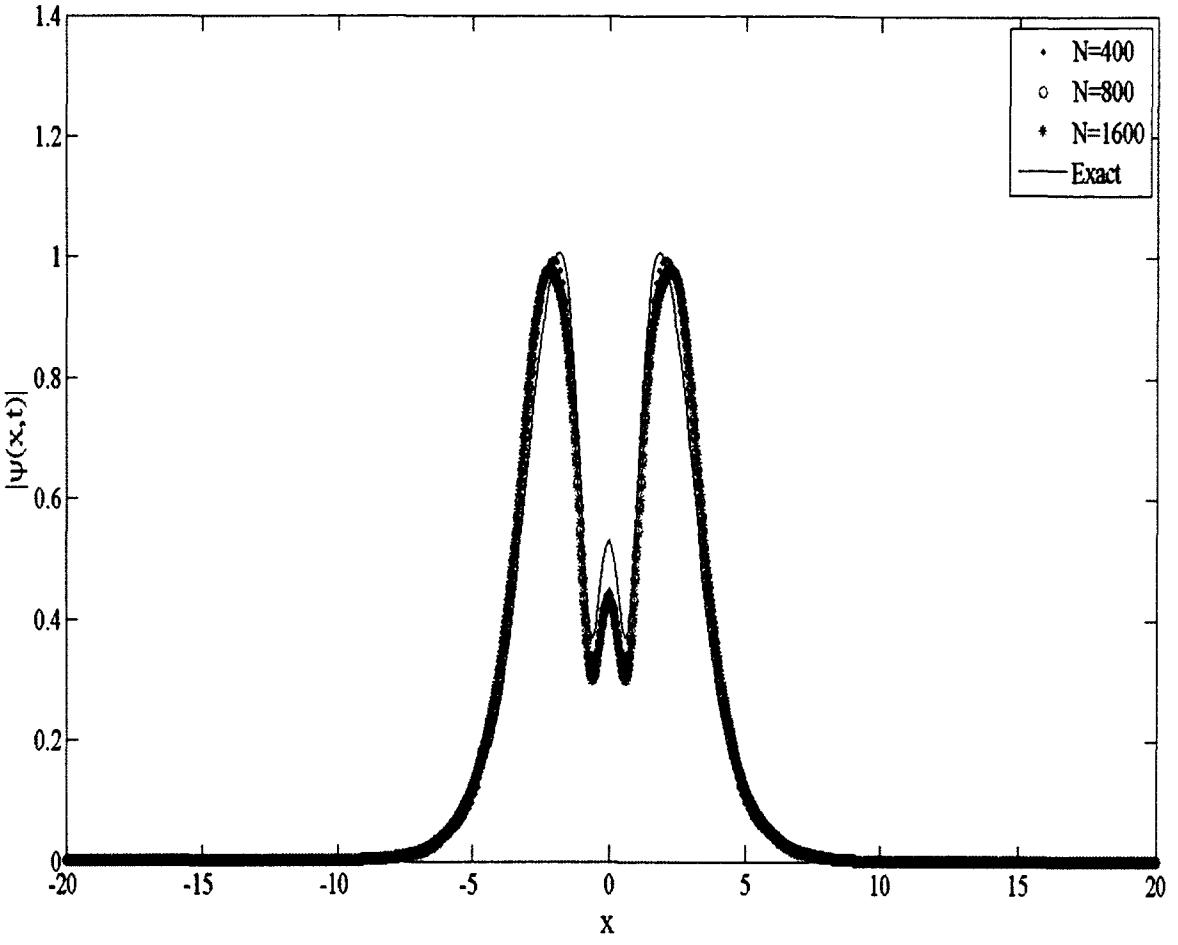


Fig. 4.7 Simulation of a soliton collision in free space at $t = 3$. A second-order central difference operator δ_x^2 is employed.

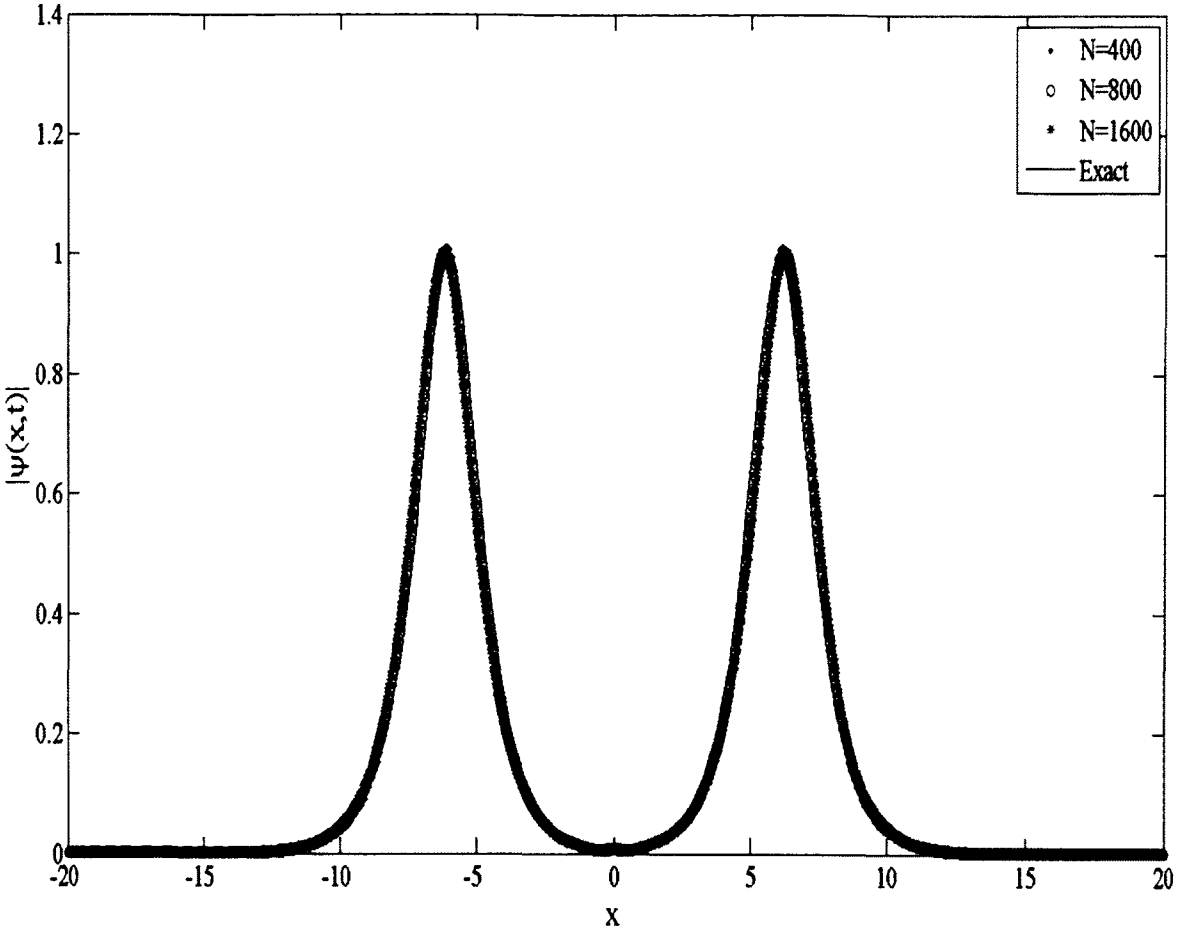


Fig. 4.8 Simulation of a soliton collision in free space at $t = 4$. A second-order central difference operator δ_x^2 is employed.

4.4 Discrete Conservation Law for the Fourth-Order Implicit G-FDTD

To show that the implicit G-FDTD scheme satisfies the first conservation law, Eq. (2.3), we introduce some finite difference operators, D_x^2, D_{2x}^2, ∇_x, $\overline{\nabla}_x$, ∇_{2x}, and $\overline{\nabla}_{2x}$ as

$$D_x^2 u(k) = u(k+1) - 2u(k) + u(k-1),$$

$$D_{2x}^2 u(k) = u(k+2) - 2u(k) + u(k-2),$$

$$\nabla_x u(k) = u(k+1) - u(k), \qquad \overline{\nabla}_x u(k) = u(k) - u(k-1),$$

$$\nabla_{2x} u(k) = u(k+2) - u(k), \qquad \overline{\nabla}_{2x} u(k) = u(k) - u(k-2).$$

It can be seen that these finite difference operators satisfy the relations: $D_x^2 = \frac{1}{12}[-D_{2x}^2 + 16D_x^2]$, $D_x^2 = \overline{\nabla}_x \cdot \nabla_x = \nabla_x - \overline{\nabla}_x$, $D_{2x}^2 = \overline{\nabla}_{2x} \cdot \nabla_{2x} = \nabla_{2x} - \overline{\nabla}_{2x}$, $\overline{\nabla}_{2x} \cdot \nabla_x = \nabla_x \cdot \overline{\nabla}_{2x}$, etc. Furthermore, one may observe that for any mesh functions, $u(k)$ and $v(k)$, which are assumed to be zero for sufficiently large $|k|$,

$$\sum_{k\in Z} \overline{\nabla}_x u(k) \cdot v(k) = -\sum_{k\in Z} u(k) \cdot \nabla_x v(k),$$

$$\sum_{k\in Z} \nabla_x u(k) \cdot v(k) = -\sum_{k\in Z} u(k) \cdot \overline{\nabla}_x v(k), \tag{4.20a}$$

$$\sum_{k\in Z} \overline{\nabla}_{2x} u(k) \cdot v(k) = -\sum_{k\in Z} u(k) \cdot \nabla_{2x} v(k),$$

$$\sum_{k\in Z} \nabla_{2x} u(k) \cdot v(k) = -\sum_{k\in Z} u(k) \cdot \overline{\nabla}_{2x} v(k), \tag{4.20b}$$

$$\sum_{k\in Z} D_x^2 u(k) \cdot v(k) = -\sum_{k\in Z} \nabla_x u(k) \cdot \nabla_x v(k) = \sum_{k\in Z} u(k) \cdot D_x^2 v(k), \tag{4.20c}$$

$$\sum_{k\in Z} D_{2x}^2 u(k) \cdot v(k) = -\sum_{k\in Z} \nabla_{2x} u(k) \cdot \nabla_{2x} v(k) = \sum_{k\in Z} u(k) \cdot D_{2x}^2 v(k), \tag{4.20d}$$

where Z is the set of all positive and negative integers. The first step is to take Eq. (4.8a) $\times \frac{\psi_{real}^{n}(k)+\psi_{real}^{n-1}(k)}{2}$ + Eq. (4.8b) $\times \frac{\psi_{imag}^{n}(k)+\psi_{imag}^{n-1}(k)}{2}$, then sum k over all integers Z. This gives

$$\frac{1}{2}\sum_{k\in Z}\{[\psi_{real}^{n}(k)]^2-[\psi_{real}^{n-1}(k)]^2+[\psi_{imag}^{n}(k)]^2-[\psi_{imag}^{n-1}(k)]^2\}$$

$$=\sum_{k\in Z}\{[\sigma D_x^2+\Delta t\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]\frac{\psi_{imag}^{n}(k)+\psi_{imag}^{n-1}(k)}{2}$$

$$\cdot\frac{\psi_{real}^{n}(k)+\psi_{real}^{n-1}(k)}{2}$$

$$-[\sigma D_x^2+\Delta t\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]\frac{\psi_{real}^{n}(k)+\psi_{real}^{n-1}(k)}{2}$$

$$\cdot\frac{\psi_{imag}^{n}(k)+\psi_{imag}^{n-1}(k)}{2}\}$$

$$+\sum_{k\in Z}\{\frac{1}{12}[\sigma D_x^2+\Delta t\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^3\frac{\psi_{imag}^{n}(k)+\psi_{imag}^{n-1}(k)}{2}$$

$$\cdot\frac{\psi_{real}^{n}(k)+\psi_{real}^{n-1}(k)}{2}$$

$$-\frac{1}{12}[\sigma D_x^2+\Delta t\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^3\frac{\psi_{real}^{n}(k)+\psi_{real}^{n-1}(k)}{2}$$

$$\cdot\frac{\psi_{imag}^{n}(k)+\psi_{imag}^{n-1}(k)}{2}\}. \tag{4.21}$$

Using the aforementioned properties, we obtain that

$$\sum_{k\in Z}\sigma D_x^2[\psi_{imag}^{n}(k)+\psi_{imag}^{n-1}(k)]\cdot[\psi_{real}^{n}(k)+\psi_{real}^{n-1}(k)]$$

$$=\sum_{k\in Z}\frac{\sigma}{12}[-D_{2x}^2+16D_x^2][\psi_{imag}^{n}(k)+\psi_{imag}^{n-1}(k)]\cdot[\psi_{real}^{n}(k)$$

$$+\psi_{real}^{n-1}(k)]$$

$$= \frac{\sigma}{12}\sum_{k\in Z} \nabla_{2x}[\psi_{imag}^{n}(k) + \psi_{imag}^{n-1}(k)] \cdot \nabla_{2x}[\psi_{real}^{n}(k) + \psi_{real}^{n-1}(k)]$$

$$-\frac{16}{12}\sigma \sum_{k\in Z} \nabla_{x}[\psi_{imag}^{n}(k) + \psi_{imag}^{n-1}(k)] \cdot \nabla_{x}[\psi_{real}^{n}(k) + \psi_{real}^{n-1}(k)]$$

$$= \sum_{k\in Z} \frac{\sigma}{12}[-D_{2x}^2 + 16D_x^2][\psi_{real}^{n}(k) + \psi_{real}^{n-1}(k)] \cdot [\psi_{imag}^{n}(k)$$

$$+\psi_{imag}^{n-1}(k)]$$

$$= \sum_{k\in Z} \sigma D_x^2[\psi_{real}^{n}(k) + \psi_{real}^{n-1}(k)] \cdot [\psi_{imag}^{n}(k) + \psi_{imag}^{n-1}(k)]. \tag{4.22}$$

Hence, the first summation on the right-hand-side of Eq. (4.21) is zero and disappears.

For the second summation on the right-hand-side of Eq. (4.21), we note that the operators

$$[\sigma D_x^2 + \Delta t \left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^3$$

$$= \sigma^3[D_x^2]^3 + \Delta t\sigma^2[D_x^2]^2 \left|\psi^{n-\frac{1}{2}}\right|^{p-1} + \Delta t\sigma^2 \left|\psi^{n-\frac{1}{2}}\right|^{p-1}[D_x^2]^2$$

$$+\Delta t\sigma^2[D_x^2]^2 \left|\psi^{n-\frac{1}{2}}\right|^{p-1} D_x^2 + \Delta t^2\sigma D_x^2[\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^2$$

$$+\Delta t^2\sigma \left|\psi^{n-\frac{1}{2}}\right|^{p-1} D_x^2 \left|\psi^{n-\frac{1}{2}}\right|^{p-1} + \Delta t^2\sigma[\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^2 D_x^2$$

$$+\Delta t^3[\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^3, \tag{4.23a}$$

$$[D_x^2]^2 = \frac{1}{12^2}\{[D_{2x}^2]^2 - 32D_{2x}^2D_x^2 + 16^2[D_x^2]^2\}, \tag{4.23b}$$

$$[D_x^2]^3 = \frac{1}{12^3}\{-[D_{2x}^2]^3 + 3\cdot 16[D_{2x}^2]^2\delta_x^2 - 3\cdot 16^2 D_{2x}^2[D_x^2]^2$$

$$+16^3[D_x^2]^3\}. \tag{4.23c}$$

Using the above properties, we obtain that for any mesh functions, $u(k)$ and $v(k)$, which are assumed to be zero for sufficiently large $|k|$,

$$\sum_{k\in Z}([D_x^2]^3 u(k))\cdot v(k) = \frac{1}{12^3}\{\sum_{k\in Z}\nabla_{2x}D_{2x}^2 u(k)\cdot\nabla_{2x}D_{2x}^2 v(k)$$

$$-48\sum_{k\in Z}\nabla_x D_{2x}^2 u(k)\cdot\nabla_x D_{2x}^2 v(k)$$

$$+3\cdot 16^2\sum_{k\in Z}\nabla_{2x}D_x^2 u(k)\cdot\nabla_{2x}D_x^2 v(k)$$

$$-16^3\sum_{k\in Z}\nabla_x D_x^2 u(k)\cdot\nabla_x D_x^2 v(k)\}, \tag{4.24a}$$

$$\sum_{k\in Z}([D_x^2]^2\left|\psi^{n-\frac{1}{2}}\right|^{p-1}u(k))\cdot v(k)$$

$$=\frac{1}{12^2}\{\sum_{k\in Z}D_{2x}^2(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}u(k))\cdot D_{2x}^2 v(k)$$

$$-32\sum_{k\in Z}\nabla_{2x}\nabla_x(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}u(k))\cdot\nabla_{2x}\nabla_x v(k)$$

$$+16^2\sum_{k\in Z}D_x^2(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}u(k))\cdot D_x^2 v(k)\}, \tag{4.24b}$$

$$\sum_{k\in Z}\left|\psi^{n-\frac{1}{2}}\right|^{p-1}[D_x^2]^2 u(k)\cdot v(k)$$

$$=\frac{1}{12^2}\{\sum_{k\in Z}D_{2x}^2 u(k)\cdot D_{2x}^2(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}v(k))$$

$$-32\sum_{k\in Z}\nabla_{2x}\nabla_x u(k)\cdot\nabla_{2x}\nabla_x\left(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}v(k)\right)$$

$$+16^2\sum_{k\in Z}D_x^2u(k)\cdot D_x^2\left(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}v(k)\right)\},\tag{4.24c}$$

$$\sum_{k\in Z}D_x^2\left(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}D_x^2u(k)\right)\cdot v(k)$$

$$=\sum_{k\in Z}\left|\psi^{n-\frac{1}{2}}\right|^{p-1}D_x^2u(k)\cdot D_x^2v(k),\tag{4.24d}$$

$$\sum_{k\in Z}D_x^2\left(\left[\left|\psi^{n-\frac{1}{2}}\right|^{p-1}\right]^2u(k)\right)\cdot v(k)$$

$$=\frac{1}{12}\{\sum_{k\in Z}\nabla_{2x}\left(\left[\left|\psi^{n-\frac{1}{2}}\right|^{p-1}\right]^2u(k)\right)\cdot\nabla_{2x}v(k)$$

$$-16\sum_{k\in Z}\nabla_x\left(\left[\left|\psi^{n-\frac{1}{2}}\right|^{p-1}\right]^2u(k)\right)\cdot\nabla_x v(k)\},\tag{4.24e}$$

$$\sum_{k\in Z}\left|\psi^{n-\frac{1}{2}}\right|^{p-1}D_x^2\left(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}u(k)\right)\cdot v(k)$$

$$=\frac{1}{12}\{\sum_{k\in Z}\nabla_{2x}\left(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}u(k)\right)$$

$$\cdot\nabla_{2x}\left(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}v(k)\right)$$

$$-16\sum_{k\in Z}\nabla_x\left(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}u(k)\right)$$

$$\cdot\nabla_x\left(\left|\psi^{n-\frac{1}{2}}\right|^{p-1}v(k)\right)\},\tag{4.24f}$$

$$\sum_{k\in Z}\left[\left|\psi^{n-\frac{1}{2}}\right|^{p-1}\right]^2D_x^2u(k)\cdot v(k)$$

$$= \frac{1}{12}\{\sum_{k\in Z} \nabla_{2x}u(k)\cdot\nabla_{2x}([\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^2 v(k))$$

$$-16\sum_{k\in Z} \nabla_{x}u(k)\cdot\nabla_{x}([\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^2 v(k))\}, \tag{4.24g}$$

$$\sum_{k\in Z}[\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^3 u(k)\cdot v(k) = \sum_{k\in Z} u(k)\cdot[\left|\psi^{n-\frac{1}{2}}\right|^{p-1}]^3 v(k). \tag{4.24h}$$

Letting $u(k) = \frac{\psi^n_{imag}(k)+\psi^{n-1}_{imag}(k)}{2}$, $v(k) = \frac{\psi^n_{real}(k)+\psi^{n-1}_{real}(k)}{2}$, and then $u(k) = \frac{\psi^n_{real}(k)+\psi^{n-1}_{real}(k)}{2}$, $v(k) = \frac{\psi^n_{imag}(k)+\psi^{n-1}_{imag}(k)}{2}$ in Eq. (4.24), in addition to using Eq. (4.23a), we obtain that the second summation on the RHS of Eq. (4.9) is zero, and hence

$$\sum_{k\in Z}\{[\psi^n_{real}(k)]^2 + [\psi^n_{imag}(k)]^2\}$$

$$= \sum_{k\in Z}\{[\psi^{n-1}_{real}(k)]^2 + [\psi^{n-1}_{imag}(k)]^2\} = constant. \tag{4.25}$$

Similarly, we can prove that Eq. (4.9) also satisfies a similar result as in Eq. (4.25), implying that the present scheme also satisfies the first conservation law.

4.5 Numerical Examples for the Fourth-Order Implicit G-FDTD

The first example is to consider a single soliton propagation, where $\lambda = -2$ and $p = 3$ in

$$i\frac{\partial\psi(x,t)}{\partial t} - \frac{\partial^2\psi(x,t)}{\partial x^2} + \lambda|\psi(x,t)|^{p-1}\psi(x,t) = 0. \tag{4.26}$$

The analytical solution is therefore

$$\psi(x,t) = \text{sech}(x+10-4t)\cdot e^{-i(2x+20-3t)}. \tag{4.27}$$

In our computation, the interval is taken to be $-20 \le x \le 20$. The solution is defined to be zero outside the interval since the value of $\psi(x,t)$ is initially negligible outside the

computation interval. We chose the number of grid points to be 200, 400, and 800, respectively, and $\Delta t = 0.0001$ in order to study the convergence with respect to space x. The maximum error and the rate of convergence are listed in Table 4.2, from which one may see that the rate of convergence is approximately *4.0*. The rate of convergence implies that the numerical accuracy of the G-FDTD scheme is similar to that of the theoretical analysis. Figure 4.10 illustrates the single soliton propagation in free space at $t=1$ and *2*, obtained using N=200, 400, and 800, which clearly shows no significant difference between the numerical solution and the analytical solution. The evaluation of maximum error at each time level *n vs. t* within $0 \leq t \leq 1$ can be seen in Figure 4.9. From Table 4.3 and Figure 4.9, one may see that our present scheme provides a more accurate solution.

Table 4.2 Maximum error obtained using a fourth-order implicit G-FDTD for a single soliton propagation when $0 \leq t \leq 1, \Delta t = 0.0001$.

Grid Points	Fourth-Order (D_x^2) Maximum Error	Rate of Convergence
200	1.26551×10^{-3}	
400	8.00973×10^{-5}	3.982
800	5.00361×10^{-6}	4.001

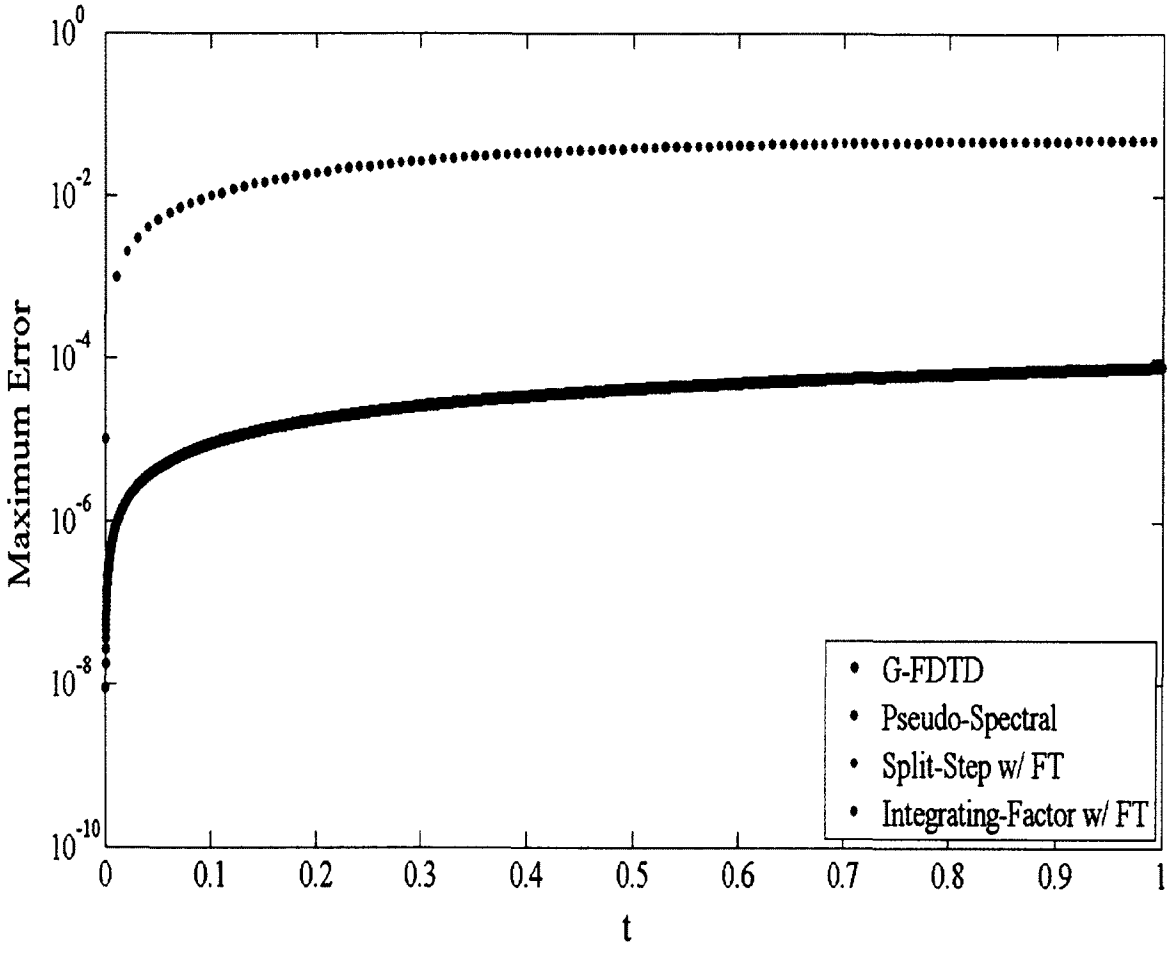
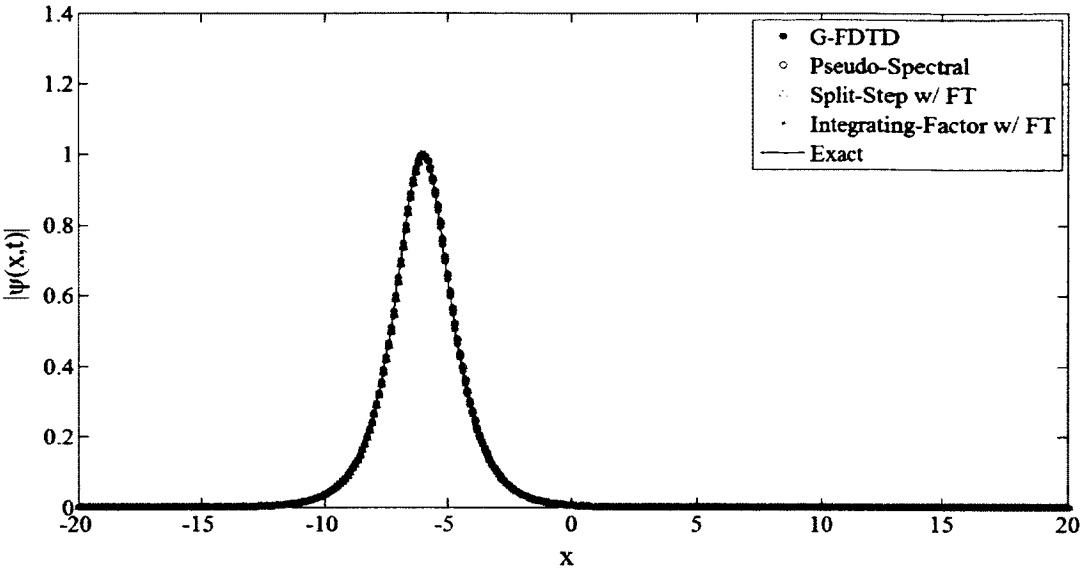


Fig. 4.9 Comparison of of maximum errors at each time level n vs. t within $0 \leq t \leq 1$ between the implicit G-FDTD scheme and other numerical methods where $\Delta t = 0.0001$ and $\Delta x = 0.1$.

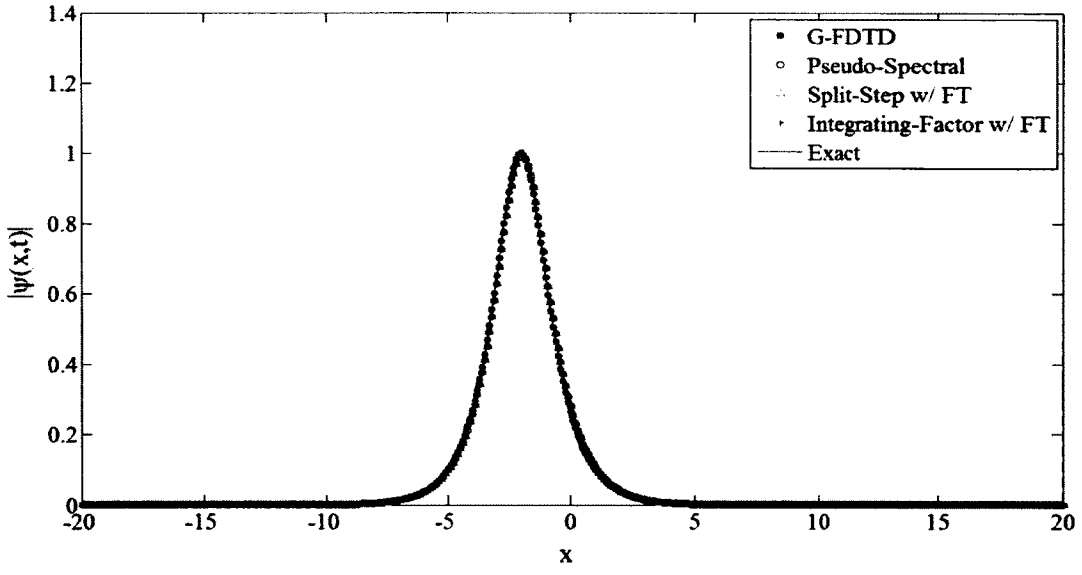
Table 4.3 Maximum error obtained using a fourth-order implicit G-FDTD for a bright soliton propagation when $0 \leq t \leq 1$, $\Delta t = 0.0001$.

Grid Points	Pseudospectral	Split-Step	Integrating-Factor	G-FDTD
200	6.82510×10^{-2}	6.82510×10^{-2}	6.82510×10^{-2}	1.27000×10^{-3}
400	4.82150×10^{-2}	4.82150×10^{-2}	4.82150×10^{-2}	8.00974×10^{-5}
800	3.40950×10^{-2}	3.40950×10^{-2}	3.40950×10^{-2}	5.00362×10^{-6}

Figure 4.10 illustrates the bright soliton propagation in free space at $t=1$ and 2, obtained using $N=400$. It can be seen from Figure 4.10 that there is no significant difference between the numerical solution and the analytical solution. In particular, Figures 4.11 and 4.12 illustrate the bright soliton propagation in free space near the right and left boundaries, respectively. Soliton behavior near the boundary conveniently demonstrates that the soliton propagates out of the boundary with analytical solution continuation.



(a)



(b)

Fig. 4.10 Simulation of a bright soliton propagating in free space, where the implicit G-FDTD scheme and other numerical methods were employed with $\Delta t = 0.0001$, $\Delta x = 0.1$ at **(a)** $t = 1$ and **(b)** $t = 2$.

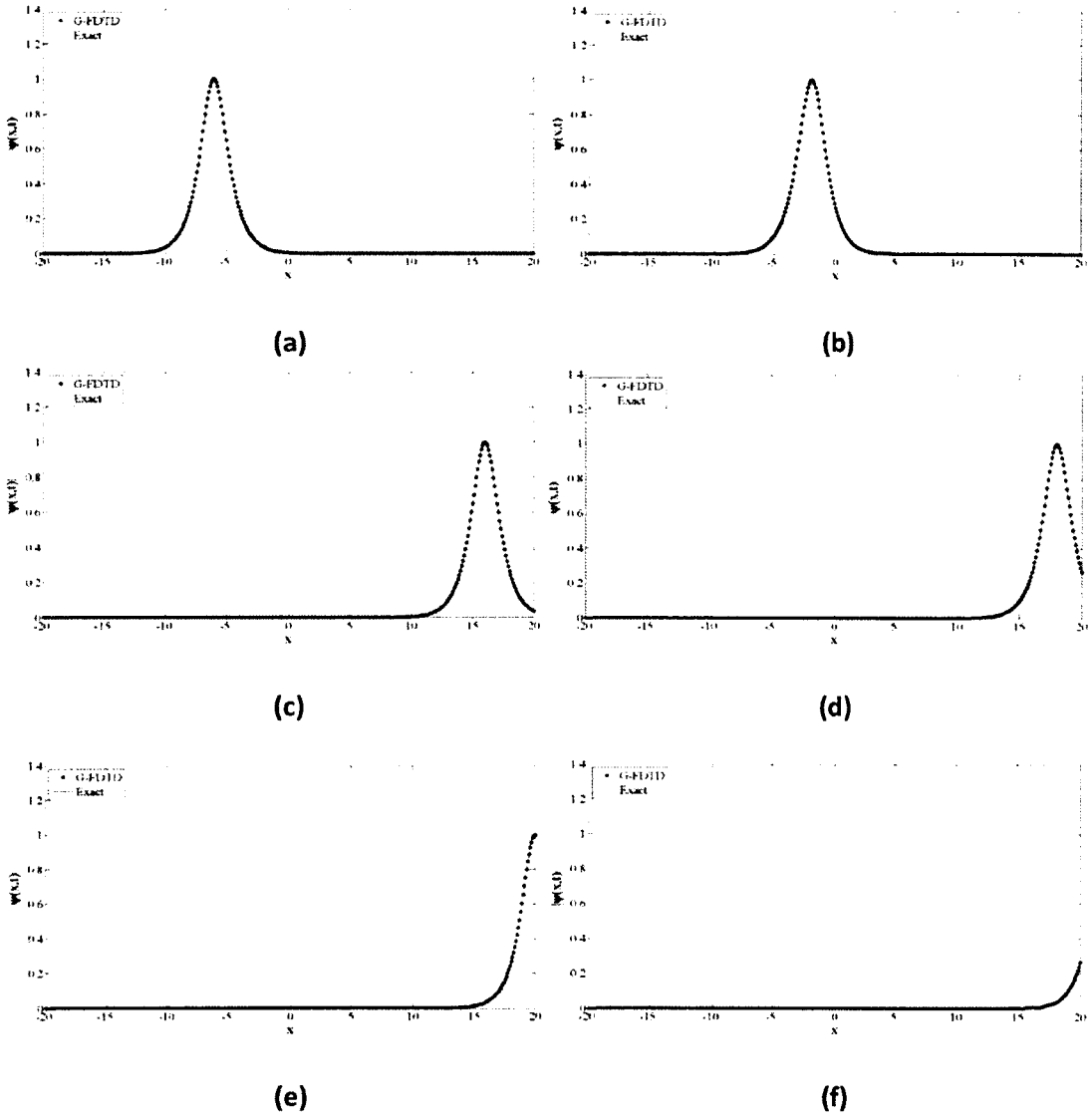


Fig. 4.11 Simulation of a bright soliton propagating in free space to the right, where the implicit G-FDTD scheme was employed with $\Delta t = 0.0001$, $\Delta x = 0.1$ at **(a)** $t = 1$, **(b)** $t = 2$, **(c)** $t = 6.5$, **(d)** $t = 7$, **(e)** $t = 7.5$, and **(f)** $t = 8$.

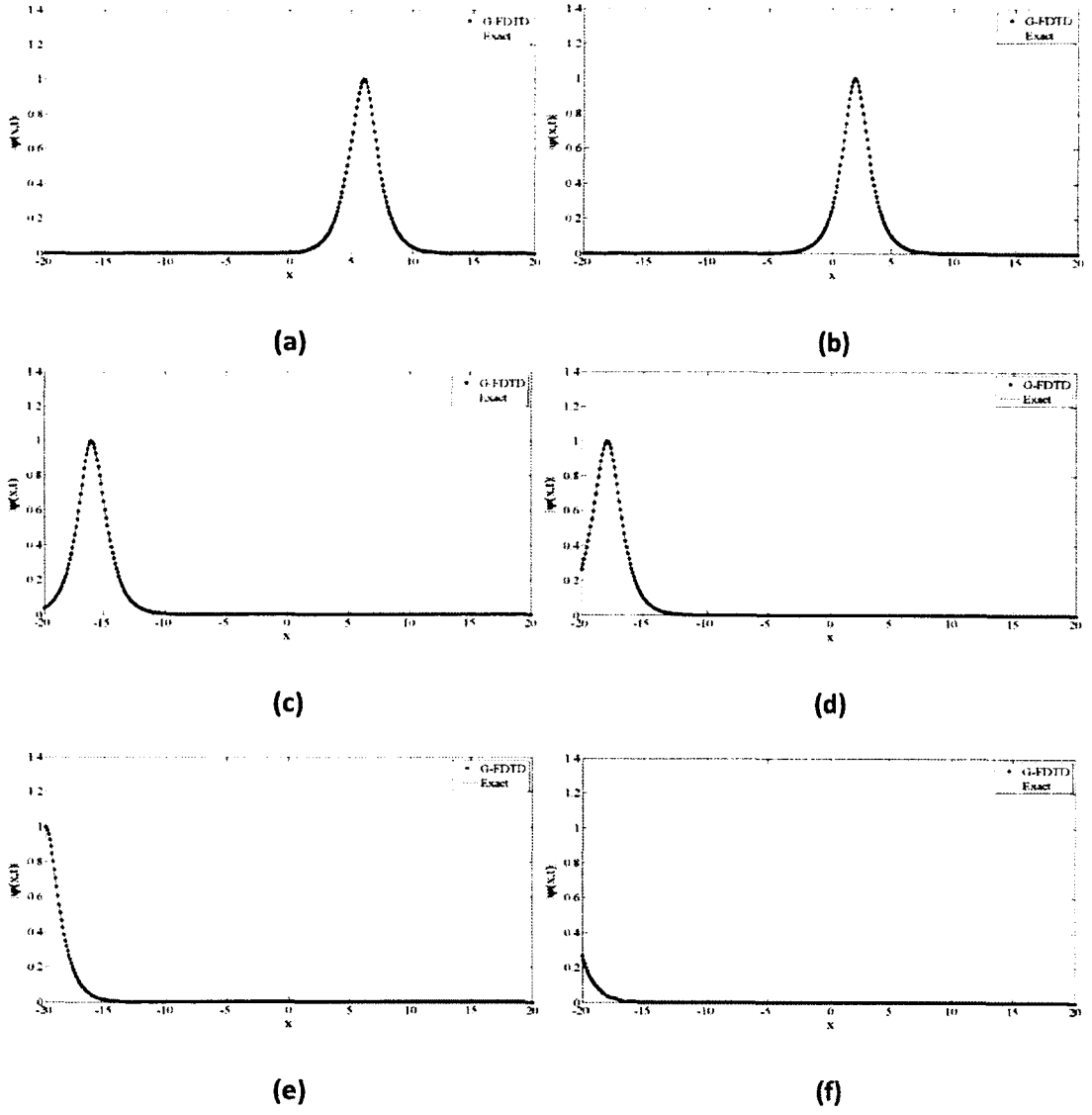


FIG. 4.12 Simulation of a bright soliton propagating in free space to the left, where the implicit G-FDTD scheme was employed with $\Delta t = 0.0001$, $\Delta x = 0.1$ at **(a)** $t = 1$, **(b)** $t = 2$, **(c)** $t = 6.5$, **(d)** $t = 7$, **(e)** $t = 7.5$, and **(f)** $t = 8$.

The second example is to consider a dark soliton propagation, where $\lambda = 2$ and $p = 3$ in

$$i\frac{\partial\psi(x,t)}{\partial t} - \frac{\partial^2\psi(x,t)}{\partial x^2} + \lambda|\psi(x,t)|^{p-1}\psi(x,t) = 0. \tag{4.28}$$

The initial condition was chosen such that the analytical solution is

$$\psi(x,t) = [tan\, h(2x - 10 - 4t) + i] \cdot e^{-i(8t)}. \tag{4.29}$$

Again, in our computation, the interval was taken to be $-20 \leq x \leq 20$. The solution was defined to be zero outside the interval since the value of $\psi(x,t)$ is initially negligible outside the computation interval. We chose the number of grid points to be 1000, 1500, and 2000, respectively, with $\Delta t = 0.0001$. Figure 4.13 illustrates the dark soliton propagation in free space at *t=1* and *2,* obtained using the three different meshes. Again, it can be seen from Figure 4.13 that there is no significant difference between the numerical solution and the analytical solution.

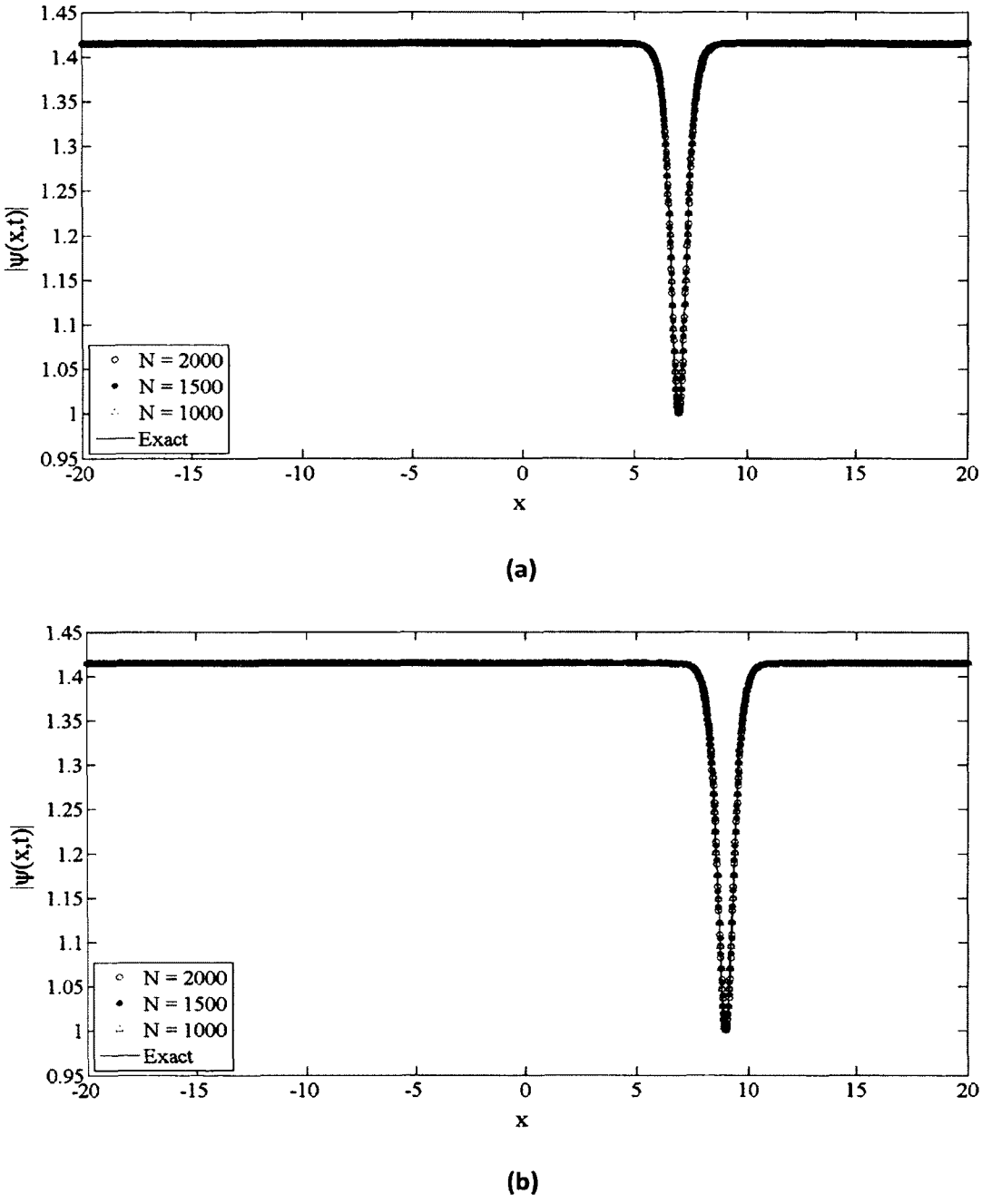


Fig. 4.13 Simulation of a dark soliton propagating in free space, where the implicit G-FDTD scheme was employed with $\Delta t = 0.0001$ and three meshes at **(a)** $t = 1$, **(b)** $t = 2$.

The third example is to consider a bright soliton propagation in two spatial dimensions, where $\lambda = -4$ and $p = 3$ in

$$i\frac{\partial\psi(x,y,t)}{\partial t} - \frac{\partial^2\psi(x,y,t)}{\partial x^2} - \frac{\partial^2\psi(x,y,t)}{\partial y^2} + \lambda|\psi(x,y,t)|^{p-1}\psi(x,y,t) = 0. \tag{4.30}$$

For this case, we chose the initial condition such that the analytical solution is

$$\psi(x,y,t) = \operatorname{sech}(x+y+10-8t)\cdot e^{-i(2x+2y+20-6t)}. \tag{4.31}$$

In our computation, the interval was taken to be $-20 \le x, y \le 20$. The solution was defined to be zero outside the interval since the value of $\psi(x,y,t)$ is initially negligible outside the computation interval. We chose the number of grid points in both x and y to be 400 with $\Delta t = 0.001$. Figure 4.14 illustrates the two-dimensional bright soliton propagation in free space at various times. The maximum error between the analytical solution and the numerical solution within $0 \le t \le 5$ is 4.46446×10^{-3}.

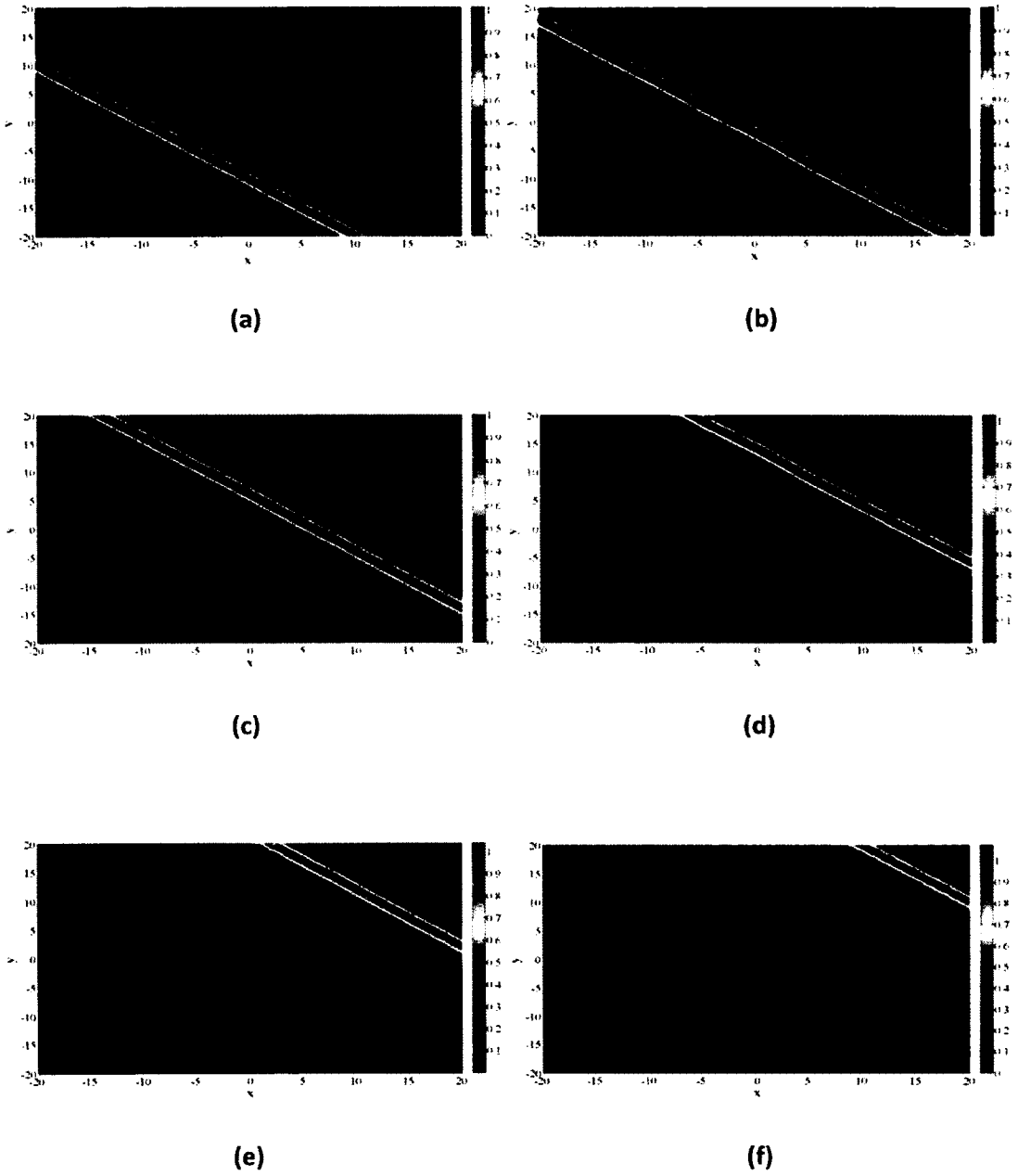


Fig. 4.14 Simulation of a 2D bright soliton propagating in free space, where the implicit G-FDTD scheme was employed with $\Delta t = 0.001$, $\Delta x = \Delta y = 0.1$ at **(a)** $t = 0$, **(b)** $t = 1$, **(c)** $t = 2$, **(d)** $t = 3$, **(e)** $t = 4$, and **(f)** $t = 5$.

We have developed a novel G-FDTD method for solving the nonlinear Schrödinger equation. The scheme has been shown to satisfy the discrete analogous form of the conservation laws, and was tested by examples of soliton propagation and collision. Compared with other existing methods, numerical results show that the present scheme provides a more accurate solution. For the sake of visual comprehension, Figure 4.15 shows a two-soliton collision in free space simulated using the G-FDTD scheme with $\Delta t = 0.01$ and $N = 232$, and a comparison is made with the analytical solution.

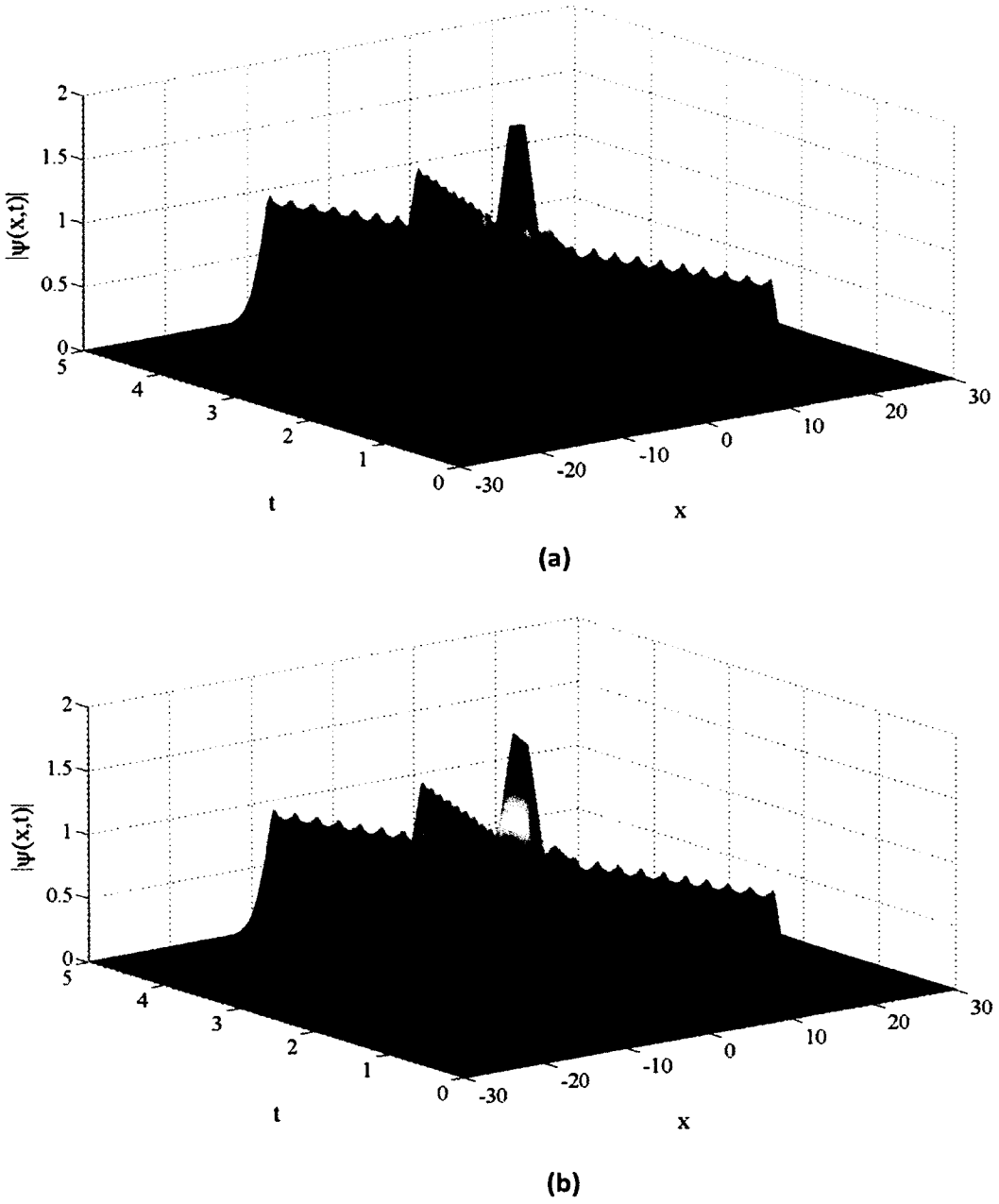


Fig. 4.15 Simulation of a two soliton collision in free space with $0 \leq t \leq 5$, where (a) the implicit G-FDTD scheme was employed with $N = 232$ and $\Delta t = 0.01$, and compared with (b) the analytical solution.

CHAPTER FIVE

CONCLUSION AND FUTURE WORK

In this dissertation, a G-FDTD scheme to simulate soliton propagation and collision has been described. Moreover, the new G-FDTD was derived both implicitly and explicitly, and then compared and contrasted with various orders of accuracy for demonstration. The new scheme was illustrated by several different numerical examples. For the purpose of comparison, both the proposed G-FDTD method and the conventional spectral methods were employed to run the simulations. Numerical results show that the proposed G-FDTD scheme is more accurate than the popular spectral methods, particularly when the time step is large. Efficiency comes from the fact that G-FDTD employs a higher-order accuracy time-stepping than does the RK4 or split-step method that is commonly used with the spectral methods. The new G-FDTD scheme provides stable solutions with large time steps, where the conventional spectral scheme is less accurate. Therefore, G-FDTD can greatly reduce the running time required by the simulation. As a result, the selection of the time step depends on what kinds of problems are simulated. The rates of convergence correspond appropriately to the order of accuracy in space, as expected. Future work in this research may focus on the G-FDTD method applied to other equations.

The future of G-FDTD numerical techniques developed will allow for a powerful way of simulating exciton-polariton condensates in the superfluid regime. This particular system has great promise in terms of new technologies involving the superfluid state, due to its possibility of room-temperature operation. Such devices will offer a set of physical rules to devise new types of circuitry based on superfluid currents, instead of classical currents. Future use of G-FDTD should be able to answer questions of exactly how well such systems are able to display the superfluid properties under a variety of different situations, and also give answers to which regimes' superfluidity can exist in such systems. To truly understand dynamics in the system, such numerical methods as G-FDTD are typically the only way to fully simulate the system. We envision that the codes developed in this study, once made freely available, will be the standard method for simulation of such systems and will greatly benefit the understanding of exciton-polariton condensates in the superfluid regime.

The G-FDTD method can be described as a novel and higher-order accurate numerical scheme for solving linear as well as nonlinear, Schrödinger equations. The FDTD methods were originally used to solve the electrodynamics governed by Maxwell's equations, but recently have been suited to handle Schrödinger equations in a more convenient and generalized manner [63, 64, 65]. The G-FDTD provides an easily accessible avenue for achieving high accuracy in the time domain where typical Runge-Kutta time stepping methods are complicated or inconvenient. Moreover, the G-FDTD also offers a high accuracy in the spatial domain that is determined by the central difference operator used to approximate the Laplacian in the NLSE. To develop the G-FDTD scheme, the wave function of the NLSE is first separated into real and imaginary

components. Substituting these split equations back into the original NLSE admits coupled equations in both real and imaginary terms. Operating under the assumption that both components of the wave function vanish as space increases to infinity, these components are then approximated using higher-order Taylor series expansions in time, and then the derivatives in time are substituted into the derivatives in space via the coupled equations. Finally, the derivatives in space are approximated using higher-order accurate discrete central difference approximations, which replace the Laplacian operator. Doing this reveals the G-FDTD scheme.

The new G-FDTD scheme is shown to satisfy the discrete analogous form of conservation law, and is derived both explicitly and implicitly, respectively. In addition, the new scheme is tested by examples of bright and dark soliton propagation and collision, in single- and multiple- space dimensions. Compared with other popular existing methods (e.g., pseudo-spectral, split-step, integrating-factor), numerical results demonstrate that the G-FDTD scheme provides a more accurate solution, particularly when the desired time step is large. The increase in permitted time-stepping allows for fast and easy simulations of nonlinear wave propagation. Furthermore, it is found that this is accomplished without the necessity of absorbing boundary conditions when using the G-FDTD applied to analytical solution continuation.

It may be posited that the study of nonlinear waves requires numerical computations to assist in the understanding of such processes. A fine example of this is demonstrated in the discovery of fractal scattering during soliton interaction. This type of scattering was found via numerical computation [101]. Numerical results such as these stimulate analytical investigation in order to further understand what is occurring

phenomenologically. Moreover, nonlinear wave evolutions given certain various initial conditions can assist in understanding how a specific nonlinear wave system truly behaves. In actuality, both the inverse scattering transform [40] was developed, and solitons were discovered as a consequence of numerical simulations [101] aimed at solving the Korteweg–de Vries (KdV) equation. During the early stages of simulation development for nonlinear wave equations, finite difference methods were primarily used [101]. Unfortunately, these finite-difference methods have low spatial accuracy, and as a result, the spectral methods were developed. Spectral methods typically have a higher spatial accuracy than finite difference methods, however, to preserve accuracy in the time-domain, a higher order time-stepping scheme is required. The convenience of the G-FDTD is seen because it offers both a simple and higher-order accurate solution in both domains of space and time without the need of a complicated higher-order time-stepping scheme. We have found that when using the G-FDTD method, it is unnecessary to implement absorbing boundary conditions (ABC's) when analytical solution continuation is applied. Not having to use ABC's is particularly advantageous from a numerical perspective, as it allows one to avoid the complications and tedious nature of applying boundary conditions in the computer code used to solve these problems. Not having to use ABC's is an added benefit of the G-FDTD method found in our numerical examples. Fortunately, using a G-FDTD approach will allow a decrease in the number of grid points used without sacrificing accuracy of the solution, or broken analytical requirements. After extensive testing of the explicit G-FDTD, it is found to be both computationally fast and precise. This result is beneficial. The implicit G-FDTD is

demonstrated to offer a precise, simple, and convenient numerical solution, albeit computationally slower than the explicit G-FDTD calculation.

APPENDIX A

SOURCE CODE OF EXAMPLE

```
c
c     This is the second order implicit scheme for the
NLSE
c
      implicit double precision (a-h,o-z)
      dimension rexac(0:5000),rbexac(0:5000),
     &          sexac(0:5000),sbexac(0:5000),x(0:5000),
     &          rn1(0:5000),rbn(0:5000),rn(0:5000),
     &          sn(0:5000),sn1(0:5000),sbn(0:5000),
     &          sbn1(0:5000),rbn1(0:5000),
     &
rn1old(0:5000),rbn1old(0:5000),u(0:802,0:41000),
     &          uexact(0:802,0:41000)

c input data
c       rn: realvalue at time n
c       rn1: realvalue at time n+1
c       sn: imaginary_value at time n
c       sn1: imaginary_value at time n+1
c       rbn: real_value at time n-1/2
c       rbn1: real_value at time n+1/2
c       sbn: imaginary_value at time n-1/2
c       sbn1: imaginary_value at time n+1/2

c Here we define the "grid size"
      np = 3
      mp=(np-1)/2
      dt = 0.001D0
      dx = 60.0D0/500.0D0
      nx=500
      dtx = dt/(dx*dx)
      dtx2 = dtx*dtx
      dtx3 = dtx2*dtx
      dtx4 = dtx3*dtx
      dtx5 = dtx4*dtx
      dtx6 = dtx5*dtx
      dtx7 = dtx6*dtx
      dtx8 = dtx7*dtx
      dtx9 = dtx8*dtx
      dtx10 = dtx9*dtx
      dt2  = dt*dt
      dt3  = dt2*dt
      dt4  = dt3*dt
      dt5  = dt4*dt
      dt6  = dt5*dt
      dt7  = dt7*dt
      dt8  = dt7*dt
      dt9  = dt8*dt
      dt10 = dt9*dt

      nt=2000
      x(0)=-30.0D0

      do i=1,nx
      x(i)=x(0)+ i*dx
      enddo

c Initial conditions
      do i=0,nx
        rn(i)=2.0D0*cos(-20.0D0-2.0D0*x(i))
     &        /(exp(x(i)+10.0D0)+exp(-10.0D0-x(i)))
     &        +2.0D0*cos(2.0D0*x(i)-20.0D0)
     &        /(exp(x(i)-10.0D0)+exp(-x(i)+10.0D0))

        sn(i)=2.0D0*sin(-20.0D0-2.0D0*x(i))
     &        /(exp(x(i)+10.0D0)+exp(-10.0D0-x(i)))
     &        +2.0D0*sin(2.0D0*x(i)-20.0D0)
     &        /(exp(x(i)-10.0D0)+exp(-x(i)+10.0D0))

        rbn(i)=2.0D0*cos(-20.0D0-
2.0D0*x(i)+1.5D0*dt)
     &        /(exp(x(i)-
2.0D0*dt+10.0D0)+exp(2.0D0*dt-x(i)-10.0D0))
     &        +2.0D0*cos(2.0D0*x(i)+1.5D0*dt-
20.0D0)
     &        /(exp(x(i)+2.0D0*dt-10.0D0)+exp(-
2.0D0*dt-x(i)+10.0D0))

        sbn(i)=2.0D0*sin(-20.0D0-
2.0D0*x(i)+1.5D0*dt)
     &        /(exp(x(i)-
2.0D0*dt+10.0D0)+exp(2.0D0*dt-x(i)-10.0D0))
     &        +2.0D0*sin(2.0D0*x(i)+1.5D0*dt-
20.0D0)
     &        /(exp(x(i)+2.0D0*dt-10.0D0)+exp(-
2.0D0*dt-x(i)+10.0D0))
      enddo

c Starting time step
      it=1

 1    do i=10,nx-10
       rn1old(i)=rn(i)
      enddo

c Boundary conditions for rn1(i)
     do i=0,9
      rn1(i)=2.0D0*cos(-20.0D0-
2.0D0*x(i)+3.0D0*it*dt)
     &        /(exp(x(i)-4.0D0*it*dt+10.0D0)
     &        +exp(4.0D0*it*dt-x(i)-10.0D0))
     &        +2.0D0*cos(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
     &        /(exp(x(i)+4.0D0*it*dt-10.0D0)
     &        +exp(-4.0D0*it*dt-x(i)+10.0D0))
      rn1old(i)=rn1(i)
     enddo

     do i=nx-9,nx
     rn1(i)=2.0D0*cos(-20.0D0-
2.0D0*x(i)+3.0D0*it*dt)
     &        /(exp(x(i)-4.0D0*it*dt+10.0D0)
     &        +exp(4.0D0*it*dt-x(i)-10.0D0))
     &        +2.0D0*cos(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
     &        /(exp(x(i)+4.0D0*it*dt-10.0D0)
     &        +exp(-4.0D0*it*dt-x(i)+10.0D0))
      rn1old(i)=rn1(i)
     enddo

c Set up the system for rn1(i)
 2    do i=10,nx-10
```

```
      c1=2.0*(rbn(i)*rbn(i)+sbn(i)*sbn(i))**mp
      c12=c1*c1
      c13=c12*c1
      c14=c13*c1
      c15=c14*c1
      c16=c15*c1
      c17=c16*c1
      c18=c17*c1
      c19=c18*c1
      c110=c19*c1

      dx2old=rn1old(i-1)+rn1old(i+1)
      dx4old=rn1old(i-2)-4.0*rn1old(i-1)-
4.0*rn1old(i+1)+rn1old(i+2)
      dx6old=rn1old(i-3)-6.0*rn1old(i-
2)+15.0*rn1old(i-1)
     &       +15.0*rn1old(i+1)-
6.0*rn1old(i+2)+rn1old(i+3)
      dx8old=rn1old(i-4)-8.0*rn1old(i-
3)+28.0*rn1old(i-2)
     &       -56.0*rn1old(i-1)-
56.0*rn1old(i+1)+28.0*rn1old(i+2)
     &       -8.0*rn1old(i+3)+rn1old(i+4)
      dx10old=rn1old(i-5)-10.0*rn1old(i-
4)+45.0*rn1old(i-3)
     &       -120.0*rn1old(i-2)+210.0*rn1old(i-
1)+210.0*rn1old(i+1)
     &       -120.0*rn1old(i+2)+45.0*rn1old(i+3)-
10.0*rn1old(i+4)
     &       +rn1old(i+5)
      dx12old=rn1old(i-6)-12.0*rn1old(i-
5)+66.0*rn1old(i-4)
     &       -220.0*rn1old(i-3)+495.0*rn1old(i-2)-
792.0*rn1old(i-1)
     &       -792.0*rn1old(i+1)+495.0*rn1old(i+2)-
220.0*rn1old(i+3)
     &       +66.0*rn1old(i+4)-
12.0*rn1old(i+5)+rn1old(i+6)
      dx14old=rn1old(i-7)-14.0*rn1old(i-
6)+91.0*rn1old(i-5)
     &       -364.0*rn1old(i-4)
     &       +1001.0*rn1old(i-3)
     &       -2002.0*rn1old(i-2)+3003.0*rn1old(i-1)
     &       +rn1old(i+7)-
14.0*rn1old(i+6)+91.0*rn1old(i+5)
     &       -364.0*rn1old(i+4)
     &       +1001.0*rn1old(i+3)
     &       -2002.0*rn1old(i+2)+3003.0*rn1old(i+1)
      dx16old=rn1old(i-8)-16.0*rn1old(i-
7)+120.0*rn1old(i-6)
     &       -560.0*rn1old(i-5)
     &       +1820.0*rn1old(i-4)
     &       -4368.0*rn1old(i-3)
     &       +8008.0*rn1old(i-2)-11440.0*rn1old(i-1)
     &       +rn1old(i+8)-
16.0*rn1old(i+7)+120.0*rn1old(i+6)
     &       -560.0*rn1old(i+5)
     &       +1820.0*rn1old(i+4)
     &       -4368.0*rn1old(i+3)
     &       +8008.0*rn1old(i+2)-11440.0*rn1old(i+1)
      dx18old=rn1old(i-9)-18.0*rn1old(i-
8)+153.0*rn1old(i-7)
     &       -816.0*rn1old(i-6)
     &       +3060.0*rn1old(i-5)
     &       -8568.0*rn1old(i-4)
     &       +18564.0*rn1old(i-3)
     &       -31824.0*rn1old(i-2)+43758.0*rn1old(i-1)
     &       +rn1old(i+9)-
18.0*rn1old(i+8)+153.0*rn1old(i+7)
     &       -816.0*rn1old(i+6)
     &       +3060.0*rn1old(i+5)
     &       -8568.0*rn1old(i+4)
     &       +18564.0*rn1old(i+3)
     &       -
31824.0*rn1old(i+2)+43758.0*rn1old(i+1)
      dx20old=rn1old(i-10)-20.0*rn1old(i-
9)+190.0*rn1old(i-8)
     &       -1140.0*rn1old(i-7)
     &       +4845.0*rn1old(i-6)
     &       -15504.0*rn1old(i-5)
     &       +38760.0*rn1old(i-4)
     &       -77520.0*rn1old(i-3)
     &       +125970.0*rn1old(i-2)-167960.0*rn1old(i-
1)
     &       +rn1old(i+10)-
20.0*rn1old(i+9)+190.0*rn1old(i+8)
     &       -1140.0*rn1old(i+7)
     &       +4845.0*rn1old(i+6)
     &       -15504.0*rn1old(i+5)
     &       +38760.0*rn1old(i+4)
     &       -77520.0*rn1old(i+3)
     &       +125970.0*rn1old(i+2)-
167960.0*rn1old(i+1)

      dx2=rn(i-1)-2.0*rn(i)+rn(i+1)
      dx4=rn(i-2)-4.0*rn(i-1)+6.0*rn(i)-
4.0*rn(i+1)+rn(i+2)
      dx6=rn(i-3)-6.0*rn(i-2)+15.0*rn(i-1)-
20.0*rn(i)+15.0*rn(i+1)
     &       -6.0*rn(i+2)+rn(i+3)
      dx8=rn(i-4)-8.0*rn(i-3)+28.0*rn(i-2)-56.0*rn(i-
1)+70.0*rn(i)
     &       -56.0*rn(i+1)+28.0*rn(i+2)-
8.0*rn(i+3)+rn(i+4)
      dx10=rn(i-5)-10.0*rn(i-4)+45.0*rn(i-3)
     &       -120.0*rn(i-2)+210.0*rn(i-1)-
252.0*rn(i)+210.0*rn(i+1)
     &       -120.0*rn(i+2)+45.0*rn(i+3)-
10.0*rn(i+4)+rn(i+5)
      dx12=rn(i-6)-12.0*rn(i-5)+66.0*rn(i-4)-
220.0*rn(i-3)
     &       +495.0*rn(i-2)-792.0*rn(i-1)+924.0*rn(i)
     &       -792.0*rn(i+1)+495.0*rn(i+2)-
220.0*rn(i+3)
     &       +66.0*rn(i+4)-12.0*rn(i+5)+rn(i+6)
      dx14=rn(i-7)-14.0*rn(i-6)+91.0*rn(i-5)-
364.0*rn(i-4)
     &       +1001.0*rn(i-3)
     &       -2002.0*rn(i-2)+3003.0*rn(i-1)-
3432.0*rn(i)
     &       +rn(i+7)-14.0*rn(i+6)+91.0*rn(i+5)-
364.0*rn(i+4)
     &       +1001.0*rn(i+3)
     &       -2002.0*rn(i+2)+3003.0*rn(i+1)
```

```
      dx16=rn(i-8)-16.0*rn(i-7)+120.0*rn(i-6)-
560.0*rn(i-5)
     &        +1820.0*rn(i-4)
     &        -4368.0*rn(i-3)
     &        +8008.0*rn(i-2)-11440.0*rn(i-
1)+12870.0*rn(i)
     &        +rn(i+8)-16.0*rn(i+7)+120.0*rn(i+6)-
560.0*rn(i+5)
     &        +1820.0*rn(i+4)
     &        -4368.0*rn(i+3)
     &        +8008.0*rn(i+2)-11440.0*rn(i+1)
      dx18=rn(i-9)-18.0*rn(i-8)+153.0*rn(i-7)-
816.0*rn(i-6)
     &        +3060.0*rn(i-5)
     &        -8568.0*rn(i-4)
     &        +18564.0*rn(i-3)
     &        -31824.0*rn(i-2)+43758.0*rn(i-1)-
48620.0*rn(i)
     &        +rn(i+9)-18.0*rn(i+8)+153.0*rn(i+7)-
816.0*rn(i+6)
     &        +3060.0*rn(i+5)
     &        -8568.0*rn(i+4)
     &        +18564.0*rn(i+3)
     &        -31824.0*rn(i+2)+43758.0*rn(i+1)
      dx20=rn(i-10)-20.0*rn(i-9)+190.0*rn(i-8)-
1140.0*rn(i-7)
     &        +4845.0*rn(i-6)
     &        -15504.0*rn(i-5)
     &        +38760.0*rn(i-4)
     &        -77520.0*rn(i-3)
     &        +125970.0*rn(i-2)-167960.0*rn(i-
1)+184756.0*rn(i)
     &        +rn(i+10)-20.0*rn(i+9)+190.0*rn(i+8)-
1140.0*rn(i+7)
     &        +4845.0*rn(i+6)
     &        -15504.0*rn(i+5)
     &        +38760.0*rn(i+4)
     &        -77520.0*rn(i+3)
     &        +125970.0*rn(i+2)-167960.0*rn(i+1)

      dx2s=sn(i-1)-2.0*sn(i)+sn(i+1)
      dx4s=sn(i-2)-4.0*sn(i-1)+6.0*sn(i)-
4.0*sn(i+1)+sn(i+2)
      dx6s=sn(i-3)-6.0*sn(i-2)+15.0*sn(i-1)-20.0*sn(i)
     &        +15.0*sn(i+1)-6.0*sn(i+2)+sn(i+3)
      dx8s=sn(i-4)-8.0*sn(i-3)+28.0*sn(i-2)-56.0*sn(i-
1)+70.0*sn(i)
     &        -56.0*sn(i+1)+28.0*sn(i+2)-
8.0*sn(i+3)+sn(i+4)
      dx10s=sn(i-5)-10.0*sn(i-4)+45.0*sn(i-3)
     &        -120.0*sn(i-2)+210.0*sn(i-1)-
252.0*sn(i)+210.0*sn(i+1)
     &        -120.0*sn(i+2)+45.0*sn(i+3)-
10.0*sn(i+4)+sn(i+5)

      A2r = -0.25*(dtx2*dx4old +
2.0*dtx*dx2old*dt*c1)
     &     -0.25*(dtx2*dx4 + 2.0*dtx*dx2*dt*c1 +
dt2*c12*rn(i))

      A4r = -(dtx4*dx8old + 4.0*dtx3*dx6old*dt*c1
     &     + 6.0*dtx2*dx4old*dt2*c12 +
4.0*dtx*dx2old*dt3*c13)/24.0
     &     -(dtx4*dx8 + 4.0*dtx3*dx6*dt*c1
     &     + 6.0*dtx2*dx4*dt2*c12 +
4.0*dtx*dx2*dt3*c13
     &     + dt4*c14*rn(i))/24.0

      A6r = -17.0*(dtx6*dx12old +
6.0*dtx5*dx10old*dt*c1
     &     + 15.0*dtx4*dx8old*dt2*c12
     &     + 20.0*dtx3*dx6old*dt3*c13
     &     + 15.0*dtx2*dx4old*dt4*c14
     &     + 6.0*dtx*dx2old*dt5*c15)/5760.0

     &     -17.0*(dtx6*dx12 + 6.0*dtx5*dx10*dt*c1
     &     + 15.0*dtx4*dx8*dt2*c12
     &     + 20.0*dtx3*dx6*dt3*c13
     &     + 15.0*dtx2*dx4*dt4*c14
     &     + 6.0*dtx*dx2*dt5*c15
     &     + dt6*c16*rn(i))/5760.0

      A8r = -(dtx8*dx16old + 8.0*dtx7*dx14old*dt*c1
     &     + 28.0*dtx6*dx12old*dt2*c12
     &     + 56.0*dtx5*dx10old*dt3*c13
     &     + 70.0*dtx4*dx8old*dt4*c14
     &     + 56.0*dtx3*dx6old*dt5*c15
     &     + 28.0*dtx2*dx4old*dt6*c16
     &     + 8.0*dtx*dx2old*dt7*c17)/2880.0

     &     -(dtx8*dx16 + 8.0*dtx7*dx14*dt*c1
     &     + 28.0*dtx6*dx12*dt2*c12
     &     + 56.0*dtx5*dx10*dt3*c13
     &     + 70.0*dtx4*dx8*dt4*c14
     &     + 56.0*dtx3*dx6*dt5*c15
     &     + 28.0*dtx2*dx4*dt6*c16
     &     + 8.0*dtx*dx2*dt7*c17
     &     + dt8*c18*rn(i))/2880.0

      A10r = -(dtx10*dx20old +
10.0*dtx9*dx18old*dt*c1
     &     + 45.0*dtx8*dx16old*dt2*c12
     &     + 120.0*dtx7*dx14old*dt3*c13
     &     + 210.0*dtx6*dx12old*dt4*c14
     &     + 252.0*dtx5*dx10old*dt5*c15
     &     + 210.0*dtx4*dx8old*dt6*c16
     &     + 120.0*dtx3*dx6old*dt7*c17
     &     + 45.0*dtx2*dx4old*dt8*c18
     &     + 10.0*dtx*dx2old*dt9*c19)/57600.0

     &     -(dtx10*dx20 + 10.0*dtx9*dx18*dt*c1
     &     + 45.0*dtx8*dx16*dt2*c12
     &     + 120.0*dtx7*dx14*dt3*c13
     &     + 210.0*dtx6*dx12*dt4*c14
     &     + 252.0*dtx5*dx10*dt5*c15
     &     + 210.0*dtx4*dx8*dt6*c16
     &     + 120.0*dtx3*dx6*dt7*c17
     &     + 45.0*dtx2*dx4*dt8*c18
     &     + 10.0*dtx*dx2*dt9*c19
     &     + dt10*c110*rn(i))/57600.0

      A1s = dtx*dx2s + dt*c1*sn(i)
```

```
      A3s = (dtx3*dx6s + 3.0*dtx2*dx4s*dt*c1 +
3.0*dtx*dx2s*dt2*c12
     &      + dt3*c13*sn(i))/12.0

      A5s = (dtx5*dx10s + 5.0*dtx4*dx8s*dt*c1
     &      + 10.0*dtx3*dx6s*dt2*c12
     &      + 10.0*dtx2*dx4s*dt3*c13
     &      + 5.0*dtx*dx2s*dt4*c14
     &      + dt5*c15*sn(i))/120.0


      ff = 1.0 + 0.25*(6.0*dtx2 - 4.0*dtx*dt*c1 +
dt2*c12)

     &      + (dtx4*70.0 - 80.0*dtx3*dt*c1 +
36.0*dtx2*dt2*c12
     &       - 8.0*dtx*dt3*c13 + dt4*c14)/24.0

     &      + 17.0*(dtx6*924.0 - 6.0*dtx5*252.0*dt*c1
     &      + 15.0*dtx4*70.0*dt2*c12 -
20.0*dtx3*20.0*dt3*c13
     &      + 15.0*dtx2*6.0*dt4*c14
     &      - 12.0*dtx*dt5*c15
     &      + dt6*c16)/5760.0

     &      +(12870.0*dtx8 -8.0*dtx7*c1*dt*3432.0
     &      + 28.0*dtx6*dt2*924.0*c12 -
56.0*dtx5*c13*252.0*dt3
     &      + 70.0*dtx4*70.0*dt4*c14
     &      - 56.0*dtx3*20.0*dt5*c15
     &      + 28.0*dtx2*6.0*dt6*c16
     &      - 16.0*dtx*dt7*c17
     &      + dt8*c18)/2880.0

     &      +(184756.0*dtx10 -486200.0*dtx9*c1*dt
     &      + 45.0*12870.0*dtx8*c12*dt2
     &      - 120.0*dtx7*c13*dt3*3432.0
     &      + 210.0*dtx6*dt4*924.0*c14
     &      - 252.0*dtx5*c15*252.0*dt5
     &      + 210.0*dtx4*70.0*dt6*c16
     &      - 120.0*dtx3*20.0*dt7*c17
     &      + 45.0*dtx2*6.0*dt8*c18
     &      - 20.0*dtx*dt9*c19
     &      + dt10*c110)/57600.0

       rn1(i) =(A2r + A4r + A6r + A8r + A10r + A1s +
A3s + A5s
     &          + rn(i))/ff
      enddo

      tol=1.0D-12
      err_max=0.0
      do i=10,nx-10
      err = abs(rn1(i)-rn1old(i))
      if (err.gt.err_max) then
      err_max=err
      endif
      enddo

      if (err_max.le.tol) goto 3
      do i=10,nx-10
       rn1old(i)=rn1(i)
      enddo
      goto 2

c solve sn1(i)
 3    do i=0,9
      sn1(i)= 2.0D0*sin(-20.0D0-
2.0D0*x(i)+3.0D0*it*dt)
     &        /(exp(x(i)-4.0D0*it*dt+10.0D0)
     &        +exp(4.0D0*it*dt-x(i)-10.0D0))
     &        +2.0D0*sin(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
     &        /(exp(x(i)+4.0D0*it*dt-10.0D0)
     &        +exp(-4.0D0*it*dt-x(i)+10.0D0))
      enddo

      do i=10,nx-10

       c1=2.0*(rbn(i)*rbn(i)+sbn(i)*sbn(i))**mp
       c12=c1*c1
       c13=c12*c1
       c14=c13*c1
       c15=c14*c1
       c16=c15*c1

      dx21=rn1(i-1)-2.0*rn1(i)+rn1(i+1)
      dx41=rn1(i-2)-4.0*rn1(i-1)+6.0*rn1(i)-
4.0*rn1(i+1)+rn1(i+2)
      dx61=rn1(i-3)-6.0*rn1(i-2)+15.0*rn1(i-1)-
20.0*rn1(i)
     &       +15.0*rn1(i+1)-6.0*rn1(i+2)+rn1(i+3)
      dx81=rn1(i-4)-8.0*rn1(i-3)+28.0*rn1(i-2)-
56.0*rn1(i-1)
     &  +70.0*rn1(i)
     &       -56.0*rn1(i+1)+28.0*rn1(i+2)-
8.0*rn1(i+3)+rn1(i+4)
      dx101=rn1(i-5)-10.0*rn1(i-4)+45.0*rn1(i-3)
     &       -120.0*rn1(i-2)+210.0*rn1(i-1)-
252.0*rn1(i)
     &       +210.0*rn1(i+1)
     &       -120.0*rn1(i+2)+45.0*rn1(i+3)-
10.0*rn1(i+4)+rn1(i+5)

      dx2=rn(i-1)-2.0*rn(i)+rn(i+1)
      dx4=rn(i-2)-4.0*rn(i-1)+6.0*rn(i)-
4.0*rn(i+1)+rn(i+2)
      dx6=rn(i-3)-6.0*rn(i-2)+15.0*rn(i-1)-
20.0*rn(i)+15.0*rn(i+1)
     &       -6.0*rn(i+2)+rn(i+3)
      dx8=rn(i-4)-8.0*rn(i-3)+28.0*rn(i-2)-56.0*rn(i-
1)+70.0*rn(i)
     &       -56.0*rn(i+1)+28.0*rn(i+2)-
8.0*rn(i+3)+rn(i+4)
      dx10=rn(i-5)-10.0*rn(i-4)+45.0*rn(i-3)
     &       -120.0*rn(i-2)+210.0*rn(i-1)-
252.0*rn(i)+210.0*rn(i+1)
     &       -120.0*rn(i+2)+45.0*rn(i+3)-
10.0*rn(i+4)+rn(i+5)

      sn1(i)=sn(i) - 0.5*(dtx*dx21 + dt*c1*rn1(i))
     &       - 0.5*(dtx*dx2+ dt*c1*rn(i))

     &       - (dtx3*dx61 + 3.0*dtx2*dx41*dt*c1
```

```
   &          + 3.0*dtx*dx21*dt2*c12 +
dt3*c13*rn1(i))/24.0

   &        - (dtx3*dx6 + 3.0*dtx2*dx4*dt*c1
   &          + 3.0*dtx*dx2*dt2*c12  +
dt3*c13*rn(i))/24.0

   &        - (dtx5*(dx101+dx10) +
5.0*dtx4*(dx81+dx8)*dt*c1
   &          + 10.0*dtx3*(dx61+dx6)*dt2*c12
   &          + 10.0*dtx2*(dx41+dx4)*dt3*c13
   &          + 5.0*dtx*(dx21+dx2)*dt4*c14
   &          + dt5*c15*(rn(i)+rn1(i)))/240.0

     enddo

     do i=nx-9,nx
     sn1(i)= 2.0D0*sin(-20.0D0-
2.0D0*x(i)+3.0D0*it*dt)
   &          /(exp(x(i)-4.0D0*it*dt+10.0D0)
   &          +exp(4.0D0*it*dt-x(i)-10.0D0))
   &          +2.0D0*sin(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
   &          /(exp(x(i)+4.0D0*it*dt-10.0D0)
   &          +exp(-4.0D0*it*dt-x(i)+10.0D0))
     enddo

      do i=10,nx-10
       rbn1old(i)=rbn(i)
      enddo

c boundary condition for rbn1(i)
      do i=0,9
       rbn1(i)= 2.0D0*cos(-20.0D0-
2.0D0*x(i)+3.0D0*(it+0.5D0)*dt)
   &          /(exp(x(i)-4.0D0*(it+0.5D0)*dt+10.0D0)
   &          +exp(4.0D0*(it+0.5D0)*dt-x(i)-10.0D0))
   &
+2.0D0*cos(2.0D0*x(i)+3.0D0*(it+0.5D0)*dt-
20.0D0)
   &          /(exp(x(i)+4.0D0*(it+0.5D0)*dt-10.0D0)
   &          +exp(-4.0D0*(it+0.5D0)*dt-x(i)+10.0D0))
       rbn1old(i)=rbn1(i)
      enddo

      do i=nx-9,nx
       rbn1(i)=2.0D0*cos(-20.0D0-
2.0D0*x(i)+3.0D0*(it+0.5D0)*dt)
   &          /(exp(x(i)-4.0D0*(it+0.5D0)*dt+10.0D0)
   &          +exp(4.0D0*(it+0.5D0)*dt-x(i)-10.0D0))
   &
+2.0D0*cos(2.0D0*x(i)+3.0D0*(it+0.5D0)*dt-
20.0D0)
   &          /(exp(x(i)+4.0D0*(it+0.5D0)*dt-10.0D0)
   &          +exp(-4.0D0*(it+0.5D0)*dt-x(i)+10.0D0))
       rbn1old(i)=rbn1(i)
      enddo


c setp up the system for rbn1(i)
 5     do i=10,nx-10
       cb1=2.0*(rn1(i)*rn1(i)+sn1(i)*sn1(i))**mp
       cb12=cb1*cb1
       cb13=cb12*cb1
       cb14=cb13*cb1
       cb15=cb14*cb1
       cb16=cb15*cb1
       cb17=cb16*cb1
       cb18=cb17*cb1
       cb19=cb18*cb1
       cb110=cb19*cb1

       dbx2old=rbn1old(i-1)+rbn1old(i+1)
       dbx4old=rbn1old(i-2)-4.0*rbn1old(i-1)-
4.0*rbn1old(i+1)
   &          +rbn1old(i+2)
       dbx6old=rbn1old(i-3)-6.0*rbn1old(i-
2)+15.0*rbn1old(i-1)
   &          +15.0*rbn1old(i+1)-
6.0*rbn1old(i+2)+rbn1old(i+3)
       dbx8old=rbn1old(i-4)-8.0*rbn1old(i-
3)+28.0*rbn1old(i-2)
   &          -56.0*rbn1old(i-1)-
56.0*rbn1old(i+1)+28.0*rbn1old(i+2)
   &          -8.0*rbn1old(i+3)+rbn1old(i+4)
       dbx10old=rbn1old(i-5)-10.0*rbn1old(i-
4)+45.0*rbn1old(i-3)
   &          -120.0*rbn1old(i-2)+210.0*rbn1old(i-1)
   &          +210.0*rbn1old(i+1)
   &          -120.0*rbn1old(i+2)+45.0*rbn1old(i+3)-
10.0*rbn1old(i+4)
   &          +rbn1old(i+5)
       dbx12old=rbn1old(i-6)-12.0*rbn1old(i-
5)+66.0*rbn1old(i-4)
   &          -220.0*rbn1old(i-3)+495.0*rbn1old(i-2)
   &          -792.0*rbn1old(i-1)
   &          -792.0*rbn1old(i+1)+495.0*rbn1old(i+2)
   &          -220.0*rbn1old(i+3)
   &          +66.0*rbn1old(i+4)-
12.0*rbn1old(i+5)+rbn1old(i+6)
       dbx14old=rbn1old(i-7)-14.0*rbn1old(i-
6)+91.0*rbn1old(i-5)
   &          -364.0*rbn1old(i-4)
   &          +1001.0*rbn1old(i-3)
   &          -2002.0*rbn1old(i-2)+3003.0*rbn1old(i-1)
   &          +rbn1old(i+7)-
14.0*rbn1old(i+6)+91.0*rbn1old(i+5)
   &          -364.0*rbn1old(i+4)
   &          +1001.0*rbn1old(i+3)
   &          -
2002.0*rbn1old(i+2)+3003.0*rbn1old(i+1)
       dbx16old=rbn1old(i-8)-16.0*rbn1old(i-
7)+120.0*rbn1old(i-6)
   &          -560.0*rbn1old(i-5)
   &          +1820.0*rbn1old(i-4)
   &          -4368.0*rbn1old(i-3)
   &          +8008.0*rbn1old(i-2)-11440.0*rbn1old(i-
1)
   &          +rbn1old(i+8)-
16.0*rbn1old(i+7)+120.0*rbn1old(i+6)
   &          -560.0*rbn1old(i+5)
   &          +1820.0*rbn1old(i+4)
   &          -4368.0*rbn1old(i+3)
```

```
     &       +8008.0*rbn1old(i+2)-
11440.0*rbn1old(i+1)
        dbx18old=rbn1old(i-9)-18.0*rbn1old(i-
8)+153.0*rbn1old(i-7)
     &       -816.0*rbn1old(i-6)
     &       +3060.0*rbn1old(i-5)
     &       -8568.0*rbn1old(i-4)
     &       +18564.0*rbn1old(i-3)
     &       -31824.0*rbn1old(i-2)+43758.0*rbn1old(i-
1)
     &       +rbn1old(i+9)-
18.0*rbn1old(i+8)+153.0*rbn1old(i+7)
     &       -816.0*rbn1old(i+6)
     &       +3060.0*rbn1old(i+5)
     &       -8568.0*rbn1old(i+4)
     &       +18564.0*rbn1old(i+3)
     &       -
31824.0*rbn1old(i+2)+43758.0*rbn1old(i+1)
        dbx20old=rbn1old(i-10)-20.0*rbn1old(i-
9)+190.0*rbn1old(i-8)
     &       -1140.0*rbn1old(i-7)
     &       +4845.0*rbn1old(i-6)
     &       -15504.0*rbn1old(i-5)
     &       +38760.0*rbn1old(i-4)
     &       -77520.0*rbn1old(i-3)
     &       +125970.0*rbn1old(i-2)-
167960.0*rbn1old(i-1)
     &       +rbn1old(i+10)-
20.0*rbn1old(i+9)+190.0*rbn1old(i+8)
     &       -1140.0*rbn1old(i+7)
     &       +4845.0*rbn1old(i+6)
     &       -15504.0*rbn1old(i+5)
     &       +38760.0*rbn1old(i+4)
     &       -77520.0*rbn1old(i+3)
     &       +125970.0*rbn1old(i+2)-
167960.0*rbn1old(i+1)

        dbx2=rbn(i-1)-2.0*rbn(i)+rbn(i+1)
        dbx4=rbn(i-2)-4.0*rbn(i-1)+6.0*rbn(i)-
4.0*rbn(i+1)+rbn(i+2)
        dbx6=rbn(i-3)-6.0*rbn(i-2)+15.0*rbn(i-1)-
20.0*rbn(i)
     &       +15.0*rbn(i+1)-6.0*rbn(i+2)+rbn(i+3)
        dbx8=rbn(i-4)-8.0*rbn(i-3)+28.0*rbn(i-2)
     &       -56.0*rbn(i-1)+70.0*rbn(i)-
56.0*rbn(i+1)+28.0*rbn(i+2)
     &       -8.0*rbn(i+3)+rbn(i+4)
        dbx10=rbn(i-5)-10.0*rbn(i-4)+45.0*rbn(i-3)
     &       -120.0*rbn(i-2)+210.0*rbn(i-1)-
252.0*rbn(i)
     &       +210.0*rbn(i+1)
     &       -120.0*rbn(i+2)+45.0*rbn(i+3)-
10.0*rbn(i+4)
     &       +rbn(i+5)
        dbx12=rbn(i-6)-12.0*rbn(i-5)+66.0*rbn(i-4)
     &       -220.0*rbn(i-3)+495.0*rbn(i-2)-
792.0*rbn(i-1)
     &       +924.0*rbn(i)
     &       -792.0*rbn(i+1)+495.0*rbn(i+2)-
220.0*rbn(i+3)
     &       +66.0*rbn(i+4)-12.0*rbn(i+5)+rbn(i+6)
        dbx14=rbn(i-7)-14.0*rbn(i-6)+91.0*rbn(i-5)-
364.0*rbn(i-4)
     &       +1001.0*rbn(i-3)
     &       -2002.0*rbn(i-2)+3003.0*rbn(i-1)-
3432.0*rbn(i)
     &       +rbn(i+7)-14.0*rbn(i+6)+91.0*rbn(i+5)-
364.0*rbn(i+4)
     &       +1001.0*rbn(i+3)
     &       -2002.0*rbn(i+2)+3003.0*rbn(i+1)
        dbx16=rbn(i-8)-16.0*rbn(i-7)+120.0*rbn(i-6)-
560.0*rbn(i-5)
     &       +1820.0*rbn(i-4)
     &       -4368.0*rbn(i-3)
     &       +8008.0*rbn(i-2)-11440.0*rbn(i-
1)+12870.0*rbn(i)
     &       +rbn(i+8)-16.0*rbn(i+7)+120.0*rbn(i+6)-
560.0*rbn(i+5)
     &       +1820.0*rbn(i+4)
     &       -4368.0*rbn(i+3)
     &       +8008.0*rbn(i+2)-11440.0*rbn(i+1)
        dbx18=rbn(i-9)-18.0*rbn(i-8)+153.0*rbn(i-7)-
816.0*rbn(i-6)
     &       +3060.0*rbn(i-5)
     &       -8568.0*rbn(i-4)
     &       +18564.0*rbn(i-3)
     &       -31824.0*rbn(i-2)+43758.0*rbn(i-1)-
48620.0*rbn(i)
     &       +rbn(i+9)-18.0*rbn(i+8)+153.0*rbn(i+7)-
816.0*rbn(i+6)
     &       +3060.0*rbn(i+5)
     &       -8568.0*rbn(i+4)
     &       +18564.0*rbn(i+3)
     &       -31824.0*rbn(i+2)+43758.0*rbn(i+1)
        dbx20=rbn(i-10)-20.0*rbn(i-9)+190.0*rbn(i-8)-
1140.0*rbn(i-7)
     &       +4845.0*rbn(i-6)
     &       -15504.0*rbn(i-5)
     &       +38760.0*rbn(i-4)
     &       -77520.0*rbn(i-3)
     &       +125970.0*rbn(i-2)-167960.0*rbn(i-
1)+184756.0*rbn(i)
     &       +rbn(i+10)-20.0*rbn(i+9)+190.0*rbn(i+8)-
1140.0*rbn(i+7)
     &       +4845.0*rbn(i+6)
     &       -15504.0*rbn(i+5)
     &       +38760.0*rbn(i+4)
     &       -77520.0*rbn(i+3)
     &       +125970.0*rbn(i+2)-167960.0*rbn(i+1)



        dbx2s=sbn(i-1)-2.0*sbn(i)+sbn(i+1)
        dbx4s=sbn(i-2)-4.0*sbn(i-1)+6.0*sbn(i)-
4.0*sbn(i+1)+sbn(i+2)
        dbx6s=sbn(i-3)-6.0*sbn(i-2)+15.0*sbn(i-1)-
20.0*sbn(i)
     &       +15.0*sbn(i+1)-6.0*sbn(i+2)+sbn(i+3)
        dbx8s=sbn(i-4)-8.0*sbn(i-3)+28.0*sbn(i-2)-
56.0*sbn(i-1)
     &       +70.0*sbn(i)
     &       -56.0*sbn(i+1)+28.0*sbn(i+2)-
8.0*sbn(i+3)+sbn(i+4)
        dbx10s=sbn(i-5)-10.0*sbn(i-4)+45.0*sbn(i-3)
```

```
     &        -120.0*sbn(i-2)+210.0*sbn(i-1)-
252.0*sbn(i)
     &        +210.0*sbn(i+1)
     &        -120.0*sbn(i+2)+45.0*sbn(i+3)-
10.0*sbn(i+4)+sbn(i+5)



      bA2r = -0.25*(dtx2*dbx4old
+2.0*dtx*dbx2old*dt*cb1)
     &        -0.25*(dtx2*dbx4+2.0*dtx*dbx2*dt*cb1 +
dt2*cb12*rbn(i))

      bA4r = -(dtx4*dbx8old +
4.0*dtx3*dbx6old*dt*cb1
     &      + 6.0*dtx2*dbx4old*dt2*cb12
     &      + 4.0*dtx*dbx2old*dt3*cb13)/24.0

     &      -(dtx4*dbx8 + 4.0*dtx3*dbx6*dt*cb1
     &      + 6.0*dtx2*dbx4*dt2*cb12
     &      + 4.0*dtx*dbx2*dt3*cb13
     &      + dt4*cb14*rbn(i))/24.0

      bA6r = -17.0*(dtx6*dbx12old +
6.0*dtx5*dbx10old*dt*cb1
     &      + 15.0*dtx4*dbx8old*dt2*cb12
     &      + 20.0*dtx3*dbx6old*dt3*cb13
     &      + 15.0*dtx2*dbx4old*dt4*cb14
     &      + 6.0*dtx*dbx2old*dt5*cb15)/5760.0

     &      -17.0*(dtx6*dbx12 +
6.0*dtx5*dbx10*dt*cb1
     &      + 15.0*dtx4*dbx8*dt2*cb12
     &      + 20.0*dtx3*dbx6*dt3*cb13
     &      + 15.0*dtx2*dbx4*dt4*cb14
     &      + 6.0*dtx*dbx2*dt5*cb15
     &      + dt6*cb16*rbn(i))/5760.0

      bA8r = -(dtx8*dbx16old +
8.0*dtx7*dbx14old*dt*cb1
     &      + 28.0*dtx6*dbx12old*dt2*cb12
     &      + 56.0*dtx5*dbx10old*dt3*cb13
     &      + 70.0*dtx4*dbx8old*dt4*cb14
     &      + 56.0*dtx3*dbx6old*dt5*cb15
     &      + 28.0*dtx2*dbx4old*dt6*cb16
     &      + 8.0*dtx*dbx2old*dt7*cb17)/2880.0

     &      -(dtx8*dbx16 + 8.0*dtx7*dbx14*dt*cb1
     &      + 28.0*dtx6*dbx12*dt2*cb12
     &      + 56.0*dtx5*dbx10*dt3*cb13
     &      + 70.0*dtx4*dbx8*dt4*cb14
     &      + 56.0*dtx3*dbx6*dt5*cb15
     &      + 28.0*dtx2*dbx4*dt6*cb16
     &      + 8.0*dtx*dbx2*dt7*cb17
     &      + dt8*cb18*rbn(i))/2880.0

      bA10r = -(dtx10*dbx20old +
10.0*dtx9*dbx18old*dt*cb1
     &      + 45.0*dtx8*dbx16old*dt2*cb12
     &      + 120.0*dtx7*dbx14old*dt3*cb13
     &      + 210.0*dtx6*dbx12old*dt4*cb14
     &      + 252.0*dtx5*dbx10old*dt5*cb15
     &      + 210.0*dtx4*dbx8old*dt6*cb16
     &      + 120.0*dtx3*dbx6old*dt7*cb17
     &      + 45.0*dtx2*dbx4old*dt8*cb18
     &      + 10.0*dtx*dbx2old*dt9*cb19)/57600.0

     &      -(dtx10*dbx20 + 10.0*dtx9*dbx18*dt*cb1
     &      + 45.0*dtx8*dbx16*dt2*cb12
     &      + 120.0*dtx7*dbx14*dt3*cb13
     &      + 210.0*dtx6*dbx12*dt4*cb14
     &      + 252.0*dtx5*dbx10*dt5*cb15
     &      + 210.0*dtx4*dbx8*dt6*cb16
     &      + 120.0*dtx3*dbx6*dt7*cb17
     &      + 45.0*dtx2*dbx4*dt8*cb18
     &      + 10.0*dtx*dbx2*dt9*cb19
     &      + dt10*cb110*rbn(i))/57600.0

      bA1s = dtx*dbx2s + dt*cb1*sbn(i)

      bA3s = (dtx3*dbx6s + 3.0*dtx2*dbx4s*dt*cb1
     &      + 3.0*dtx*dbx2s*dt2*cb12
     &      + dt3*cb13*sbn(i))/12.0

      bA5s = (dtx5*dbx10s + 5.0*dtx4*dbx8s*dt*cb1
     &      + 10.0*dtx3*dbx6s*dt2*cb12
     &      + 10.0*dtx2*dbx4s*dt3*cb13
     &      + 5.0*dtx*dbx2s*dt4*cb14
     &      + dt5*cb15*sbn(i))/120.0

      fb = 1.0 + 0.25*(6.0*dtx2 - 4.0*dtx*dt*cb1 +
dt*dt*cb12)

     &      +(dtx4*70.0 - 80.0*dtx3*dt*cb1 +
36.0*dtx2*dt2*cb12
     &      - 8.0*dtx*dt3*cb13 + dt4*cb14)/24.0

     &      + 17.0*(dtx6*924.0 -
6.0*dtx5*252.0*dt*cb1
     &      + 15.0*dtx4*70.0*dt2*cb12 -
20.0*dtx3*20.0*dt3*cb13
     &      + 15.0*dtx2*6.0*dt4*cb14
     &      - 12.0*dtx*dt5*cb15
     &      + dt6*cb16)/5760.0

     &      + (12870.0*dtx8 -8.0*dtx7*cb1*dt*3432.0
     &      + 28.0*dtx6*dt2*924.0*cb12
     &      - 56.0*dtx5*cb13*252.0*dt3
     &      + 70.0*dtx4*70.0*dt4*cb14
     &      - 56.0*dtx3*20.0*dt5*cb15
     &      + 28.0*dtx2*6.0*dt6*cb16
     &      - 16.0*dtx*dt7*cb17
     &      + dt8*cb18)/2880.0

     &      + (184756.0*dtx10 -486200.0*dtx9*cb1*dt
     &      + 45.0*12870.0*dtx8*cb12*dt2
     &      - 120.0*dtx7*cb13*dt3*3432.0
     &      + 210.0*dtx6*dt4*924.0*cb14
     &      - 252.0*dtx5*cb15*252.0*dt5
     &      + 210.0*dtx4*70.0*dt6*cb16
     &      - 120.0*dtx3*20.0*dt7*cb17
     &      + 45.0*dtx2*6.0*dt8*cb18
     &      - 20.0*dtx*dt9*cb19
     &      + dt10*cb110)/57600.0
```

```
      rbn1(i) =(bA2r + bA4r + bA6r + bA8r + bA10r +
bA1s + bA3s
     &      + bA5s + rbn(i))/fb

      enddo

      tol=1.0D-12
      err_max=0.0
      do i=10,nx-10
      err=abs(rbn1(i)-rbn1old(i))
      if (err.gt.err_max) then
      err_max = err
      endif
      enddo

      if (err_max.le.tol) goto 6
      do i=10,nx-10
      rbn1old(i)=rbn1(i)
      enddo
      goto 5

c solve sbn1(i)
 6    do i=0,9
      sbn1(i)= 2.0D0*sin(-20.0D0-
2.0D0*x(i)+3.0D0*(it+0.5D0)*dt)
     &         /(exp(x(i)-4.0D0*(it+0.5D0)*dt+10.0D0)
     &         +exp(4.0D0*(it+0.5D0)*dt-x(i)-10.0D0))
     &
+2.0D0*sin(2.0D0*x(i)+3.0D0*(it+0.5D0)*dt-
20.0D0)
     &         /(exp(x(i)+4.0D0*(it+0.5D0)*dt-10.0D0)
     &         +exp(-4.0D0*(it+0.5D0)*dt-x(i)+10.0D0))
      enddo

      do i=10,nx-10

      cb1=2.0*(rn1(i)*rn1(i)+sn1(i)*sn1(i))**mp
      cb12=cb1*cb1
      cb13=cb12*cb1
      cb14=cb13*cb1
      cb15=cb14*cb1
      cb16=cb15*cb1

      dbx21=rbn1(i-1)-2.0*rbn1(i)+rbn1(i+1)
      dbx41=rbn1(i-2)-4.0*rbn1(i-1)+6.0*rbn1(i)-
4.0*rbn1(i+1)
     &      +rbn1(i+2)
      dbx61=rbn1(i-3)-6.0*rbn1(i-2)+15.0*rbn1(i-1)-
20.0*rbn1(i)
     &      +15.0*rbn1(i+1)-6.0*rbn1(i+2)+rbn1(i+3)
      dbx81=rbn1(i-4)-8.0*rbn1(i-3)+28.0*rbn1(i-2)-
56.0*rbn1(i-1)
     &      +70.0*rbn1(i)
     &      -56.0*rbn1(i+1)+28.0*rbn1(i+2)-
8.0*rbn1(i+3)+rbn1(i+4)
      dbx101=rbn1(i-5)-10.0*rbn1(i-4)+45.0*rbn1(i-3)
     &      -120.0*rbn1(i-2)+210.0*rbn1(i-1)-
252.0*rbn1(i)
     &      +210.0*rbn1(i+1)
     &      -120.0*rbn1(i+2)+45.0*rbn1(i+3)-
10.0*rbn1(i+4)+rbn1(i+5)

      dbx2=rbn(i-1)-2.0*rbn(i)+rbn(i+1)
      dbx4=rbn(i-2)-4.0*rbn(i-1)+6.0*rbn(i)-
4.0*rbn(i+1)+rbn(i+2)
      dbx6=rbn(i-3)-6.0*rbn(i-2)+15.0*rbn(i-1)-
20.0*rbn(i)
     &      +15.0*rbn(i+1)
     &      -6.0*rbn(i+2)+rbn(i+3)
      dbx8=rbn(i-4)-8.0*rbn(i-3)+28.0*rbn(i-2)-
56.0*rbn(i-1)
     &      +70.0*rbn(i)
     &      -56.0*rbn(i+1)+28.0*rbn(i+2)-
8.0*rbn(i+3)+rbn(i+4)
      dbx10=rbn(i-5)-10.0*rbn(i-4)+45.0*rbn(i-3)
     &      -120.0*rbn(i-2)+210.0*rbn(i-1)-
252.0*rbn(i)
     &      +210.0*rbn(i+1)
     &      -120.0*rbn(i+2)+45.0*rbn(i+3)-
10.0*rbn(i+4)+rbn(i+5)

      sbn1(i)=sbn(i) - 0.5*(dtx*dbx21 +
dt*cb1*rbn1(i))
     &      - 0.5*(dtx*dbx2+ dt*cb1*rbn(i))

     &      - (dtx3*dbx61
     &        + 3.0*dtx2*dbx41*dt*cb1
     &        + 3.0*dtx*dbx21*dt2*cb12
     &        + dt3*cb13*rbn1(i))/24.0

     &      - (dtx3*dbx6
     &        + 3.0*dtx2*dbx4*dt*cb1
     &        + 3.0*dtx*dbx2*dt2*cb12
     &        + dt3*cb13*rbn(i))/24.0

     &      - (dtx5*(dbx101+dbx10) +
5.0*dtx4*(dbx81+dbx8)*dt*cb1
     &        + 10.0*dtx3*(dbx61+dbx6)*dt2*cb12
     &        + 10.0*dtx2*(dbx41+dbx4)*dt3*cb13
     &        + 5.0*dtx*(dbx21+dbx2)*dt4*cb14
     &        + dt5*cb15*(rbn(i)+rbn1(i)))/240.0
      enddo

      do i=nx-9,nx
       sbn1(i)= 2.0D0*sin(-20.0D0-
2.0D0*x(i)+3.0D0*(it+0.5D0)*dt)
     &         /(exp(x(i)-4.0D0*(it+0.5D0)*dt+10.0D0)
     &         +exp(4.0D0*(it+0.5D0)*dt-x(i)-10.0D0))
     &
+2.0D0*sin(2.0D0*x(i)+3.0D0*(it+0.5D0)*dt-
20.0D0)
     &         /(exp(x(i)+4.0D0*(it+0.5D0)*dt-10.0D0)
     &         +exp(-4.0D0*(it+0.5D0)*dt-x(i)+10.0D0))
      enddo

c calculate the exact solution
      do i=10,nx-10
      rexac(i)= 2.0D0*cos(-20.0D0-
2.0D0*x(i)+3.0D0*it*dt)
     &         /(exp(x(i)-4.0D0*it*dt+10.0D0)
     &         +exp(4.0D0*it*dt-x(i)-10.0D0))
     &         +2.0D0*cos(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
```

```
     &        /(exp(x(i)+4.0D0*it*dt-10.0D0)
     &        +exp(-4.0D0*it*dt-x(i)+10.0D0))

      sexac(i)=2.0D0*sin(-20.0D0-
2.0D0*x(i)+3.0D0*it*dt)
     &        /(exp(x(i)-4.0D0*it*dt+10.0D0)
     &        +exp(4.0D0*it*dt-x(i)-10.0D0))
     &        +2.0D0*sin(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
     &        /(exp(x(i)+4.0D0*it*dt-10.0D0)
     &        +exp(-4.0D0*it*dt-x(i)+10.0D0))

      rbexac(i)=2.0D0*cos(-20.0D0-
2.0D0*x(i)+3.0D0*(it+0.5D0)*dt)
     &        /(exp(x(i)-4.0D0*(it+0.5D0)*dt+10.0D0)
     &        +exp(4.0D0*(it+0.5D0)*dt-x(i)-10.0D0))
     &
+2.0D0*cos(2.0D0*x(i)+3.0D0*(it+0.5D0)*dt-
20.0D0)
     &        /(exp(x(i)+4.0D0*(it+0.5D0)*dt-10.0D0)
     &        +exp(-4.0D0*(it+0.5D0)*dt-x(i)+10.0D0))

      sbexac(i)=2.0D0*sin(-20.0D0-
2.0D0*x(i)+3.0D0*(it+0.5D0)*dt)
     &        /(exp(x(i)-4.0D0*(it+0.5D0)*dt+10.0D0)
     &        +exp(4.0D0*(it+0.5D0)*dt-x(i)-10.0D0))
     &
+2.0D0*sin(2.0D0*x(i)+3.0D0*(it+0.5D0)*dt-
20.0D0)
     &        /(exp(x(i)+4.0D0*(it+0.5D0)*dt-10.0D0)
     &        +exp(-4.0D0*(it+0.5D0)*dt-x(i)+10.0D0))
      enddo

c calculate the error
      temp=0.0
      do i=10,nx-10
C       temp=temp+(rn1(i)-rexac(i))*(rn1(i)-rexac(i))
C     &     +(sn1(i)-sexac(i))*(sn1(i)-sexac(i))
C     &     +(rbn1(i)-rbexac(i))*(rbn1(i)-rbexac(i))
C     &     +(sbn1(i)-sbexac(i))*(sbn1(i)-sbexac(i))
      temp=temp+(sqrt(rn1(i)*rn1(i)+sn1(i)*sn1(i))
     &        -sqrt(rexac(i)*rexac(i)+sexac(i)*sexac(i)))
      enddo
C       errnew=sqrt(temp*dx/2.0D0)
      errnew=abs(temp*dx)
      print *, it, errnew

c continue time step
      it=it+1
      if(it.eq.nt) goto 7
      do i=0,nx
      rn(i)=rn1(i)
      sn(i)=sn1(i)
      rbn(i)=rbn1(i)
      sbn(i)=sbn1(i)
      u(i,it)= sqrt(rn1(i)*rn1(i)+sn1(i)*sn1(i))
      uexact(i,it)=
sqrt(rexac(i)*rexac(i)+sexac(i)*sexac(i))
      enddo
      goto 1

 7    open(unit=03, file='error_demo.dat')
      do i=0,nx
      write(03,*)i*dx,sqrt(rn1(i)*rn1(i)+sn1(i)*sn1(i)),
     &     sqrt(rexac(i)*rexac(i)+sexac(i)*sexac(i))
      enddo
      close (03)

      open(unit=03, file='error_demo_two.dat')
      do i=0,nx
      write(03,*)i*dx,sn1(i),sexac(i)
      enddo
      close (03)
      end
```

```
c
c This is the fourth-order 1D implicit scheme for the
NLSE/bright soliton
c
      implicit double precision (a-h,o-z)
      dimension rexac(0:10000),rbexac(0:10000),
     &
sexac(0:10000),sbexac(0:10000),x(0:10000),
     &        rn1(0:10000),rbn(0:10000),rn(0:10000),
     &        sn(0:10000),sn1(0:10000),sbn(0:10000),
     &
sbn1(0:10000),rbn1(0:10000),sn1old(0:10000),
     &        sbn1old(0:10000)

c input data
c     rn: realvalue at time n
c     rn1: realvalue at time n+1
c     sn: imaginary_value at time n
c     sn1: imaginary_value at time n+1
c     rbn: real_value at time n-1/2
c     rbn1: real_value at time n+1/2
c     sbn: imaginary_value at time n-1/2
c     sbn1: imaginary_value at time n+1/2

c Here we define the "grid size"
      np = 3
      mp=(np-1)/2
      dt = 0.0001D0
      dx = 0.1D0
      nx=400
      dtx = dt/(dx*dx)
      dtx2 = dtx*dtx
      dtx3 = dtx2*dtx
      dt2  = dt*dt
      dt3  = dt2*dt

      nt=10000
      x(0)=-20.0D0

      do i=1,nx
      x(i)=x(0)+ i*dx
      enddo

c Initial conditions
      do i=0,nx
        rn(i)=2.0D0*cos(-20.0D0-2.0D0*x(i))
     &        /(exp(x(i)+10.0D0)+exp(-10.0D0-x(i)))
C     &        +2.0D0*cos(2.0D0*x(i)-20.0D0)
```

```
C     &         /(exp(x(i)-10.0D0)+exp(-x(i)+10.0D0))

          sn(i)=2.0D0*sin(-20.0D0-2.0D0*x(i))
     &         /(exp(x(i)+10.0D0)+exp(-10.0D0-x(i)))
C     &          +2.0D0*sin(2.0D0*x(i)-20.0D0)
C     &         /(exp(x(i)-10.0D0)+exp(-x(i)+10.0D0))

          rbn(i)=2.0D0*cos(-20.0D0-
2.0D0*x(i)+1.5D0*dt)
     &         /(exp(x(i)-
2.0D0*dt+10.0D0)+exp(2.0D0*dt-x(i)-10.0D0))
C     &          +2.0D0*cos(2.0D0*x(i)+1.5D0*dt-
20.0D0)
C     &         /(exp(x(i)+2.0D0*dt-10.0D0)+exp(-
2.0D0*dt-x(i)+10.0D0))

          sbn(i)=2.0D0*sin(-20.0D0-
2.0D0*x(i)+1.5D0*dt)
     &         /(exp(x(i)-
2.0D0*dt+10.0D0)+exp(2.0D0*dt-x(i)-10.0D0))
C     &          +2.0D0*sin(2.0D0*x(i)+1.5D0*dt-
20.0D0)
C     &         /(exp(x(i)+2.0D0*dt-10.0D0)+exp(-
2.0D0*dt-x(i)+10.0D0))
      enddo


c Starting time step
      it=1

 1    do i=6,nx-6
       sn1old(i)=sn(i)
      enddo

c Boundary conditions for rn1(i)

     do i=0,5
     sn1(i)= 2.0D0*sin(-20.0D0-
2.0D0*x(i)+3.0D0*it*dt)
     &         /(exp(x(i)-4.0D0*it*dt+10.0D0)
     &         +exp(4.0D0*it*dt-x(i)-10.0D0))
C     &          +2.0D0*sin(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C     &         /(exp(x(i)+4.0D0*it*dt-10.0D0)
C     &          +exp(-4.0D0*it*dt-x(i)+10.0D0))
     sn1old(i)=sn1(i)
     enddo

     do i=nx-5,nx
     sn1(i)= 2.0D0*sin(-20.0D0-
2.0D0*x(i)+3.0D0*it*dt)
     &         /(exp(x(i)-4.0D0*it*dt+10.0D0)
     &         +exp(4.0D0*it*dt-x(i)-10.0D0))
C     &          +2.0D0*sin(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C     &         /(exp(x(i)+4.0D0*it*dt-10.0D0)
C     &          +exp(-4.0D0*it*dt-x(i)+10.0D0))
     sn1old(i)=sn1(i)
     enddo

     do i=0,5
     rn1(i)=2.0D0*cos(-20.0D0-
2.0D0*x(i)+3.0D0*it*dt)
     &         /(exp(x(i)-4.0D0*it*dt+10.0D0)
     &         +exp(4.0D0*it*dt-x(i)-10.0D0))
C     &          +2.0D0*cos(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C     &         /(exp(x(i)+4.0D0*it*dt-10.0D0)
C     &          +exp(-4.0D0*it*dt-x(i)+10.0D0))
     enddo

     do i=nx-5,nx
     rn1(i)=2.0D0*cos(-20.0D0-
2.0D0*x(i)+3.0D0*it*dt)
     &         /(exp(x(i)-4.0D0*it*dt+10.0D0)
     &         +exp(4.0D0*it*dt-x(i)-10.0D0))
C     &          +2.0D0*cos(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C     &         /(exp(x(i)+4.0D0*it*dt-10.0D0)
C     &          +exp(-4.0D0*it*dt-x(i)+10.0D0))
     enddo

 2     do i=6,nx-6
      c1=2.0*(rbn(i)*rbn(i)+sbn(i)*sbn(i))**mp
      c12=c1*c1
      c13=c12*c1

      dx2old=(-sn1old(i+2)+16.0*sn1old(i+1)-
30.0*sn1old(i)
     &        +16.0*sn1old(i-1)
     &        -sn1old(i-2))/12.0

      dx4old= sn1old(i+4)/144.0 - 2.0*sn1old(i+3)/9.0
     &        +79.0*sn1old(i+2)/36.0 -
62.0*sn1old(i+1)/9.0
     &        +707.0*sn1old(i)/72.0  - 62.0*sn1old(i-
1)/9.0
     &        +79.0*sn1old(i-2)/36.0 - 2.0*sn1old(i-
3)/9.0
     &        +sn1old(i-4)/144.0

      dx6old= -sn1old(i+6)/1728.0 + sn1old(i+5)/36.0
     &        -143.0*sn1old(i+4)/288.0 +
439.0*sn1old(i+3)/108.0
     &        -3031.0*sn1old(i+2)/192.0 +
203.0*sn1old(i+1)/6.0
     &        -6233.0*sn1old(i)/144.0 + 203.0*sn1old(i-
1)/6.0
     &        -3031.0*sn1old(i-2)/192.0 +
439.0*sn1old(i-3)/108.0
     &        -143.0*sn1old(i-4)/288.0 + sn1old(i-
5)/36.0
     &        -sn1old(i-6)/1728.0

      dx2=  (-sn(i+2)+16.0*sn(i+1)-
30.0*sn(i)+16.0*sn(i-1)
     &       -sn(i-2))/12.0

      dx4=sn(i+4)/144.0 - 2.0*sn(i+3)/9.0
     &        +79.0*sn(i+2)/36.0 - 62.0*sn(i+1)/9.0
     &        +707.0*sn(i)/72.0  - 62.0*sn(i-1)/9.0
     &        +79.0*sn(i-2)/36.0 - 2.0*sn(i-3)/9.0
     &        +sn(i-4)/144.0
```

```
      dx6= -sn(i+6)/1728.0 + sn(i+5)/36.0
     &       -143.0*sn(i+4)/288.0 +
439.0*sn(i+3)/108.0
     &       -3031.0*sn(i+2)/192.0 + 203.0*sn(i+1)/6.0
     &       -6233.0*sn(i)/144.0 + 203.0*sn(i-1)/6.0
     &       -3031.0*sn(i-2)/192.0 + 439.0*sn(i-
3)/108.0
     &       -143.0*sn(i-4)/288.0 + sn(i-5)/36.0
     &       -sn(i-6)/1728.0

      rn1(i) = rn(i) + dtx*dx2old/2.0 + dtx*dx2/2.0
     &       + dt*c1*sn1old(i)/2.0 + dt*c1*sn(i)/2.0
     &       + dtx3*dx6old/24.0 +
3.0*dtx2*dt*c1*dx4old/24.0
     &       + 3.0*dtx*dt2*c12*dx2old/24.0 +
dt3*c13*sn1old(i)/24.0
     &       + dtx3*dx6/24.0 +
3.0*dtx2*dt*c1*dx4/24.0
     &       + 3.0*dtx*dt2*c12*dx2/24.0 +
dt3*c13*sn(i)/24.0

     enddo

     do i=6,nx-6

      c1=2.0*(rbn(i)*rbn(i)+sbn(i)*sbn(i))**mp
      c12=c1*c1
      c13=c12*c1

      dx2old=(-rn1(i+2)+16.0*rn1(i+1)-30.0*rn1(i)
     &       +16.0*rn1(i-1)
     &       -rn1(i-2))/12.0

      dx4old=rn1(i+4)/144.0 - 2.0*rn1(i+3)/9.0
     &       +79.0*rn1(i+2)/36.0 - 62.0*rn1(i+1)/9.0
     &       +707.0*rn1(i)/72.0  - 62.0*rn1(i-1)/9.0
     &       +79.0*rn1(i-2)/36.0 - 2.0*rn1(i-3)/9.0
     &       +rn1(i-4)/144.0

      dx6old= -rn1(i+6)/1728.0 + rn1(i+5)/36.0
     &       -143.0*rn1(i+4)/288.0 +
439.0*rn1(i+3)/108.0
     &       -3031.0*rn1(i+2)/192.0 +
203.0*rn1(i+1)/6.0
     &       -6233.0*rn1(i)/144.0 + 203.0*rn1(i-1)/6.0
     &       -3031.0*rn1(i-2)/192.0 + 439.0*rn1(i-
3)/108.0
     &       -143.0*rn1(i-4)/288.0 + rn1(i-5)/36.0
     &       -rn1(i-6)/1728.0

      dx2=(-rn(i+2)+16.0*rn(i+1)-
30.0*rn(i)+16.0*rn(i-1)
     &       -rn(i-2))/12.0

      dx4=rn(i+4)/144.0 - 2.0*rn(i+3)/9.0
     &       +79.0*rn(i+2)/36.0 - 62.0*rn(i+1)/9.0
     &       +707.0*rn(i)/72.0  - 62.0*rn(i-1)/9.0
     &       +79.0*rn(i-2)/36.0 - 2.0*rn(i-3)/9.0
     &       +rn(i-4)/144.0

      dx6=  -rn(i+6)/1728.0 + rn(i+5)/36.0
     &       -143.0*rn(i+4)/288.0 +
439.0*rn(i+3)/108.0
     &       -3031.0*rn(i+2)/192.0 + 203.0*rn(i+1)/6.0
     &       -6233.0*rn(i)/144.0 + 203.0*rn(i-1)/6.0
     &       -3031.0*rn(i-2)/192.0 + 439.0*rn(i-
3)/108.0
     &       -143.0*rn(i-4)/288.0 + rn(i-5)/36.0
     &       -rn(i-6)/1728.0

      sn1(i)=sn(i) - dtx*dx2old/2.0 - dtx*dx2/2.0
     &       - dt*c1*rn1(i)/2.0 - dt*c1*rn(i)/2.0
     &       - dtx3*dx6old/24.0 -
3.0*dtx2*dt*c1*dx4old/24.0
     &       - 3.0*dtx*dt2*c12*dx2old/24.0 -
dt3*c13*rn1(i)/24.0
     &       - dtx3*dx6/24.0 - 3.0*dtx2*dt*c1*dx4/24.0
     &       - 3.0*dtx*dt2*c12*dx2/24.0 -
dt3*c13*rn(i)/24.0
     enddo

     tol=1.0D-15
     err_max=0.0
     do i=6,nx-6
     err = abs(sn1(i)-sn1old(i))
     if (err.gt.err_max) then
     err_max=err
     endif
     enddo

     if (err_max.le.tol) goto 3
     do i=6,nx-6
      sn1old(i)=sn1(i)
     enddo
     goto 2

 3    do i=6,nx-6
       sbn1old(i)=sbn(i)
     enddo

     do i=0,5
     sbn1(i)= 2.0D0*sin(-20.0D0-
2.0D0*x(i)+3.0D0*(it+0.5D0)*dt)
     &       /(exp(x(i)-4.0D0*(it+0.5D0)*dt+10.0D0)
     &       +exp(4.0D0*(it+0.5D0)*dt-x(i)-10.0D0))
C    &
+2.0D0*sin(2.0D0*x(i)+3.0D0*(it+0.5D0)*dt-
20.0D0)
C    &       /(exp(x(i)+4.0D0*(it+0.5D0)*dt-10.0D0)
C    &       +exp(-4.0D0*(it+0.5D0)*dt-
x(i)+10.0D0))
     sbn1old(i)=sbn1(i)
     enddo

     do i=nx-5,nx
      sbn1(i)= 2.0D0*sin(-20.0D0-
2.0D0*x(i)+3.0D0*(it+0.5D0)*dt)
     &       /(exp(x(i)-4.0D0*(it+0.5D0)*dt+10.0D0)
     &       +exp(4.0D0*(it+0.5D0)*dt-x(i)-10.0D0))
C    &
+2.0D0*sin(2.0D0*x(i)+3.0D0*(it+0.5D0)*dt-
20.0D0)
C    &       /(exp(x(i)+4.0D0*(it+0.5D0)*dt-10.0D0)
```

```
C     &        +exp(-4.0D0*(it+0.5D0)*dt-
x(i)+10.0D0))
      sbn1old(i)=sbn1(i)
      enddo

      do i=0,5
       rbn1(i)= 2.0D0*cos(-20.0D0-
2.0D0*x(i)+3.0D0*(it+0.5D0)*dt)
     &        /(exp(x(i)-4.0D0*(it+0.5D0)*dt+10.0D0)
     &        +exp(4.0D0*(it+0.5D0)*dt-x(i)-10.0D0))
C     &
+2.0D0*cos(2.0D0*x(i)+3.0D0*(it+0.5D0)*dt-
20.0D0)
C     &        /(exp(x(i)+4.0D0*(it+0.5D0)*dt-10.0D0)
C     &        +exp(-4.0D0*(it+0.5D0)*dt-
x(i)+10.0D0))
      enddo

      do i=nx-5,nx
      rbn1(i)=2.0D0*cos(-20.0D0-
2.0D0*x(i)+3.0D0*(it+0.5D0)*dt)
     &        /(exp(x(i)-4.0D0*(it+0.5D0)*dt+10.0D0)
     &        +exp(4.0D0*(it+0.5D0)*dt-x(i)-10.0D0))
C     &
+2.0D0*cos(2.0D0*x(i)+3.0D0*(it+0.5D0)*dt-
20.0D0)
C     &        /(exp(x(i)+4.0D0*(it+0.5D0)*dt-10.0D0)
C     &        +exp(-4.0D0*(it+0.5D0)*dt-
x(i)+10.0D0))
      enddo

 5    do i=6,nx-6

      cb1=2.0*(rn1(i)*rn1(i)+sn1(i)*sn1(i))**mp
      cb12=cb1*cb1
      cb13=cb12*cb1

      dbx2old=(-sbn1old(i+2)+16.0*sbn1old(i+1)-
30.0*sbn1old(i)
     &       +16.0*sbn1old(i-1)
     &       -sbn1old(i-2))/12.0

      dbx4old= sbn1old(i+4)/144.0 -
2.0*sbn1old(i+3)/9.0
     &       +79.0*sbn1old(i+2)/36.0 -
62.0*sbn1old(i+1)/9.0
     &       +707.0*sbn1old(i)/72.0  - 62.0*sbn1old(i-
1)/9.0
     &       +79.0*sbn1old(i-2)/36.0 - 2.0*sbn1old(i-
3)/9.0
     &       +sbn1old(i-4)/144.0

      dbx6old= -sbn1old(i+6)/1728.0 +
sbn1old(i+5)/36.0
     &       -143.0*sbn1old(i+4)/288.0 +
439.0*sbn1old(i+3)/108.0
     &       -3031.0*sbn1old(i+2)/192.0 +
203.0*sbn1(i+1)/6.0
     &       -6233.0*sbn1old(i)/144.0 +
203.0*sbn1old(i-1)/6.0
     &       -3031.0*sbn1old(i-2)/192.0 +
439.0*sbn1old(i-3)/108.0
     &       -143.0*sbn1old(i-4)/288.0 + sbn1old(i-
5)/36.0
     &       -sbn1old(i-6)/1728.0

      dbx2=  (-sbn(i+2)+16.0*sbn(i+1)-
30.0*sbn(i)+16.0*sbn(i-1)
     &       -sbn(i-2))/12.0

      dbx4=sbn(i+4)/144.0 - 2.0*sbn(i+3)/9.0
     &       +79.0*sbn(i+2)/36.0 - 62.0*sbn(i+1)/9.0
     &       +707.0*sbn(i)/72.0  - 62.0*sbn(i-1)/9.0
     &       +79.0*sbn(i-2)/36.0 - 2.0*sbn(i-3)/9.0
     &       +sbn(i-4)/144.0

      dbx6= -sbn(i+6)/1728.0 + sbn(i+5)/36.0
     &       -143.0*sbn(i+4)/288.0 +
439.0*sbn(i+3)/108.0
     &       -3031.0*sbn(i+2)/192.0 +
203.0*sbn(i+1)/6.0
     &       -6233.0*sbn(i)/144.0 + 203.0*sbn(i-1)/6.0
     &       -3031.0*sbn(i-2)/192.0 + 439.0*sbn(i-
3)/108.0
     &       -143.0*sbn(i-4)/288.0 + sbn(i-5)/36.0
     &       -sbn(i-6)/1728.0

      rbn1(i) = rbn(i) + dtx*dbx2old/2.0 +
dtx*dbx2/2.0
     &       + dt*cb1*sbn1old(i)/2.0 +
dt*cb1*sbn(i)/2.0
     &       + dtx3*dbx6old/24.0 +
3.0*dtx2*dt*cb1*dbx4old/24.0
     &       + 3.0*dtx*dt2*cb12*dbx2old/24.0
     &       + dt3*cb13*sbn1old(i)/24.0
     &       + dtx3*dbx6/24.0 +
3.0*dtx2*dt*cb1*dbx4/24.0
     &       + 3.0*dtx*dt2*cb12*dbx2/24.0 +
dt3*cb13*sbn(i)/24.0

      enddo

     do i=6,nx-6

      cb1=2.0*(rn1(i)*rn1(i)+sn1(i)*sn1(i))**mp
      cb12=cb1*cb1
      cb13=cb12*cb1

      dbx2old=(-rbn1(i+2)+16.0*rbn1(i+1)-
30.0*rbn1(i)
     &       +16.0*rbn1(i-1)
     &       -rbn1(i-2))/12.0

      dbx4old=rbn1(i+4)/144.0 - 2.0*rbn1(i+3)/9.0
     &       +79.0*rbn1(i+2)/36.0 - 62.0*rbn1(i+1)/9.0
     &       +707.0*rbn1(i)/72.0  - 62.0*rbn1(i-1)/9.0
     &       +79.0*rbn1(i-2)/36.0 - 2.0*rbn1(i-3)/9.0
     &       +rbn1(i-4)/144.0

      dbx6old= -rbn1(i+6)/1728.0 + rbn1(i+5)/36.0
     &       -143.0*rbn1(i+4)/288.0 +
439.0*rbn1(i+3)/108.0
     &       -3031.0*rbn1(i+2)/192.0 +
203.0*rbn1(i+1)/6.0
```

```
     &        -6233.0*rbn1(i)/144.0 + 203.0*rbn1(i-
1)/6.0
     &        -3031.0*rbn1(i-2)/192.0 + 439.0*rbn1(i-
3)/108.0
     &        -143.0*rbn1(i-4)/288.0 + rbn1(i-5)/36.0
     &        -rbn1(i-6)/1728.0

       dbx2=(-rbn(i+2)+16.0*rbn(i+1)-
30.0*rbn(i)+16.0*rbn(i-1)
     &       -rbn(i-2))/12.0

       dbx4=rbn(i+4)/144.0 - 2.0*rbn(i+3)/9.0
     &        +79.0*rbn(i+2)/36.0 - 62.0*rbn(i+1)/9.0
     &        +707.0*rbn(i)/72.0  - 62.0*rbn(i-1)/9.0
     &        +79.0*rbn(i-2)/36.0 - 2.0*rbn(i-3)/9.0
     &        +rbn(i-4)/144.0

       dbx6=  -rbn(i+6)/1728.0 + rbn(i+5)/36.0
     &        -143.0*rbn(i+4)/288.0 +
439.0*rbn(i+3)/108.0
     &        -3031.0*rbn(i+2)/192.0 +
203.0*rbn(i+1)/6.0
     &        -6233.0*rbn(i)/144.0 + 203.0*rbn(i-1)/6.0
     &        -3031.0*rbn(i-2)/192.0 + 439.0*rbn(i-
3)/108.0
     &        -143.0*rbn(i-4)/288.0 + rbn(i-5)/36.0
     &        -rbn(i-6)/1728.0

       sbn1(i)=sbn(i) - dtx*dbx2old/2.0 - dtx*dbx2/2.0
     &        - dt*cb1*rbn1(i)/2.0 - dt*cb1*rbn(i)/2.0
     &        - dtx3*dbx6old/24.0 -
3.0*dtx2*dt*cb1*dbx4old/24.0
     &        - 3.0*dtx*dt2*cb12*dbx2old/24.0 -
dt3*cb13*rbn1(i)/24.0
     &        - dtx3*dbx6/24.0 -
3.0*dtx2*dt*cb1*dbx4/24.0
     &        - 3.0*dtx*dt2*cb12*dbx2/24.0 -
dt3*cb13*rbn(i)/24.0
       enddo

       tol=1.0D-15
       err_max=0.0
       do i=6,nx-6
       err=abs(sbn1(i)-sbn1old(i))
       if (err.gt.err_max) then
       err_max = err
       endif
       enddo

       if (err_max.le.tol) goto 6
       do i=6,nx-6
       sbn1old(i)=sbn1(i)
       enddo
       goto 5

c calculate the exact solution
 6     do i=0,nx
       rexac(i)= 2.0D0*cos(-20.0D0-
2.0D0*x(i)+3.0D0*it*dt)
     &        /(exp(x(i)-4.0D0*it*dt+10.0D0)
     &        +exp(4.0D0*it*dt-x(i)-10.0D0))
C    &        +2.0D0*cos(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &        /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &        +exp(-4.0D0*it*dt-x(i)+10.0D0))

       sexac(i)=2.0D0*sin(-20.0D0-
2.0D0*x(i)+3.0D0*it*dt)
     &        /(exp(x(i)-4.0D0*it*dt+10.0D0)
     &        +exp(4.0D0*it*dt-x(i)-10.0D0))
C    &        +2.0D0*sin(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &        /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &        +exp(-4.0D0*it*dt-x(i)+10.0D0))

       rbexac(i)=2.0D0*cos(-20.0D0-
2.0D0*x(i)+3.0D0*(it+0.5D0)*dt)
     &        /(exp(x(i)-4.0D0*(it+0.5D0)*dt+10.0D0)
     &        +exp(4.0D0*(it+0.5D0)*dt-x(i)-10.0D0))
C    &
+2.0D0*cos(2.0D0*x(i)+3.0D0*(it+0.5D0)*dt-
20.0D0)
C    &        /(exp(x(i)+4.0D0*(it+0.5D0)*dt-10.0D0)
C    &        +exp(-4.0D0*(it+0.5D0)*dt-
x(i)+10.0D0))

       sbexac(i)=2.0D0*sin(-20.0D0-
2.0D0*x(i)+3.0D0*(it+0.5D0)*dt)
     &        /(exp(x(i)-4.0D0*(it+0.5D0)*dt+10.0D0)
     &        +exp(4.0D0*(it+0.5D0)*dt-x(i)-10.0D0))
C    &
+2.0D0*sin(2.0D0*x(i)+3.0D0*(it+0.5D0)*dt-
20.0D0)
C    &        /(exp(x(i)+4.0D0*(it+0.5D0)*dt-10.0D0)
C    &        +exp(-4.0D0*(it+0.5D0)*dt-
x(i)+10.0D0))
       enddo

c calculate the error
C      temp=0.0
C      do i=6,nx-6
C       temp1=(sqrt(rn1(i)*rn1(i)+sn1(i)*sn1(i))
C    &        -
sqrt(rexac(i)*rexac(i)+sexac(i)*sexac(i)))
C      temp=temp+temp1*temp1
C      enddo
C      errnew=sqrt(temp/(nx-10))
C      print *, it, errnew

       temp=0.0
       open(unit=23, file='implicit_error.dat')
       do i=6,nx-6
       temp1=(sqrt(rn1(i)*rn1(i)+sn1(i)*sn1(i))
     &        -sqrt(rexac(i)*rexac(i)+sexac(i)*sexac(i)))
       temp=temp+temp1*temp1
       enddo
       errnew=sqrt(temp/(nx-10))
       print *, it, errnew
       write(23,*)it*dt,errnew

c continue time step
       it=it+1
       if(it.eq.nt) goto 7
```

```
      do i=0,nx
      rn(i)=rn1(i)
      sn(i)=sn1(i)
      rbn(i)=rbn1(i)
      sbn(i)=sbn1(i)
      enddo
      goto 1

 7    open(unit=03, file='implicit_400_1.dat')
      do i=0,nx
      write(03,*)i*dx,sqrt(rn1(i)*rn1(i)+sn1(i)*sn1(i)),
     &     sqrt(rexac(i)*rexac(i)+sexac(i)*sexac(i))
      enddo
      close (03)

C     open(unit=03, file='trial_s_400_8.dat')
C     do i=0,nx
C     write(03,*)i*dx,sn1(i),sexac(i)
C     enddo
C     close (03)
      End
```

```
c
c This is the 2D implicit scheme for the NLSE/bright
soliton
c
      implicit double precision (a-h,o-z)
      dimension
rexac(0:1000,0:1000),rbexac(0:1000,0:1000),
     &
sexac(0:1000,0:1000),sbexac(0:1000,0:1000),
     &        x(0:1000),y(0:1000),
     &        rn1(0:1000,0:1000),rbn(0:1000,0:1000),
     &        rn(0:1000,0:1000),
     &        sn(0:1000,0:1000),sn1(0:1000,0:1000),
     &        sbn(0:1000,0:1000),
     &
sbn1(0:1000,0:1000),rbn1(0:1000,0:1000),
     &
sn1old(0:1000,0:1000),sbn1old(0:1000,0:1000)

c input data
c     rn: realvalue at time n
c     rn1: realvalue at time n+1
c     sn: imaginary_value at time n
c     sn1: imaginary_value at time n+1
c     rbn: real_value at time n-1/2
c     rbn1: real_value at time n+1/2
c     sbn: imaginary_value at time n-1/2
c     sbn1: imaginary_value at time n+1/2

c Here we define the "grid size"
      np = 3
      mp=(np-1)/2
      dt = 0.001D0
      dx = 0.05D0
      nx=800
      dy = 0.05D0
      ny=800
      dtx = dt/(dx*dx)
      dtx2 = dtx*dtx
      dtx3 = dtx2*dtx
      dt2  = dt*dt
      dt3  = dt2*dt

      nt=1000
      x(0)=-20.0D0
      y(0)=-20.0D0

      do i=1,nx
      x(i)=x(0)+ i*dx
      enddo

      do j=1,ny
      y(j)=y(0)+ j*dy
      enddo

c Initial conditions
      do i=0,nx
      do j=0,ny
        rn(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j))
     &        /(exp(x(i)+y(j)+10.0D0)+exp(-10.0D0-
x(i)-y(j)))
C     &        +2.0D0*cos(2.0D0*x(i)-20.0D0)
C     &        /(exp(x(i)-10.0D0)+exp(-x(i)+10.0D0))

        sn(i,j)=2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j))
     &        /(exp(x(i)+y(j)+10.0D0)+exp(-10.0D0-
x(i)-y(j)))
C     &        +2.0D0*sin(2.0D0*x(i)-20.0D0)
C     &        /(exp(x(i)-10.0D0)+exp(-x(i)+10.0D0))

        rbn(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+3.0D0*dt)
     &        /(exp(x(i)+y(j)-4.0D0*dt+10.0D0)
     &        +exp(4.0D0*dt-x(i)-y(j)-10.0D0))
C     &        +2.0D0*cos(2.0D0*x(i)+1.5D0*dt-
20.0D0)
C     &        /(exp(x(i)+2.0D0*dt-10.0D0)+exp(-
2.0D0*dt-x(i)+10.0D0))

        sbn(i,j)=2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+3.0D0*dt)
     &        /(exp(x(i)+y(j)-4.0D0*dt+10.0D0)
     &        +exp(4.0D0*dt-x(i)-y(j)-10.0D0))
C     &        +2.0D0*sin(2.0D0*x(i)+1.5D0*dt-
20.0D0)
C     &        /(exp(x(i)+2.0D0*dt-10.0D0)+exp(-
2.0D0*dt-x(i)+10.0D0))
      enddo
      enddo


c Starting time step
      it=1

 1    do i=6,nx-6
      do j=6,ny-6
```

```
      sn1old(i,j)=sn(i,j)
     enddo
     enddo

c Boundary conditions for rn1(i)

     do i=0,nx
     do j=0,5
     rn1(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
    &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
    &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))

      sn1(i,j)= 2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
    &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
    &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))
     sn1old(i,j)=sn1(i,j)
     enddo
     enddo

     do i=0,nx
     do j=ny-5,ny
     rn1(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
    &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
    &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))

      sn1(i,j)= 2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
    &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
    &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))
     sn1old(i,j)=sn1(i,j)
     enddo
     enddo

     do i=0,5
     do j=6,ny-6
      rn1(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
    &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
    &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))

      sn1(i,j)= 2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
    &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
    &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))
     sn1old(i,j)=sn1(i,j)
     enddo
     enddo

     do i=nx-5,nx
     do j=6,ny-6
      rn1(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
    &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
    &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))

      sn1(i,j)= 2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
    &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
    &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))
     sn1old(i,j)=sn1(i,j)
     enddo
     enddo

2     do i=6,nx-6
      do j=6,ny-6
      c1=4.0*(rbn(i,j)*rbn(i,j)+sbn(i,j)*sbn(i,j))**mp
      c12=c1*c1
      c13=c12*c1

     dx4dy2old= -sn1old(i+4,j+2)/1728.0 +
sn1old(i+3,j+2)/54.0
    &        -79.0*sn1old(i+2,j+2)/432.0
    &        +31.0*sn1old(i+1,j+2)/54.0
    &        -707.0*sn1old(i,j+2)/864.0
    &        +31.0*sn1old(i-1,j+2)/54.0
    &        -79.0*sn1old(i-2,j+2)/432.0
    &        +sn1old(i-3,j+2)/54.0
    &        -sn1old(i-4,j+2)/1728.0

    &        +16.0*sn1old(i+4,j+1)/1728.0
    &        -16.0*sn1old(i+3,j+1)/54.0
    &        +16.0*79.0*sn1old(i+2,j+1)/432.0
    &        -16.0*31.0*sn1old(i+1,j+1)/54.0
    &        +16.0*707.0*sn1old(i,j+1)/864.0
    &        -16.0*31.0*sn1old(i-1,j+1)/54.0
    &        +16.0*79.0*sn1old(i-2,j+1)/432.0
    &        -16.0*sn1old(i-3,j+1)/54.0
    &        +16.0*sn1old(i-4,j+1)/1728.0

    &        -30.0*sn1old(i+4,j)/1728.0
    &        +30.0*sn1old(i+3,j)/54.0
    &        -30.0*79.0*sn1old(i+2,j)/432.0
    &        +30.0*31.0*sn1old(i+1,j)/54.0
    &        -30.0*707.0*sn1old(i,j)/864.0
    &        +30.0*31.0*sn1old(i-1,j)/54.0
    &        -30.0*79.0*sn1old(i-2,j)/432.0
    &        +30.0*sn1old(i-3,j)/54.0
    &        -30.0*sn1old(i-4,j)/1728.0

    &        +16.0*sn1old(i+4,j-1)/1728.0
    &        -16.0*sn1old(i+3,j-1)/54.0
    &        +16.0*79.0*sn1old(i+2,j-1)/432.0
    &        -16.0*31.0*sn1old(i+1,j-1)/54.0
    &        +16.0*707.0*sn1old(i,j-1)/864.0
    &        -16.0*31.0*sn1old(i-1,j-1)/54.0
    &        +16.0*79.0*sn1old(i-2,j-1)/432.0
    &        -16.0*sn1old(i-3,j-1)/54.0
    &        +16.0*sn1old(i-4,j-1)/1728.0

    &        -sn1old(i+4,j-2)/1728.0 + sn1old(i+3,j-
2)/54.0
    &        -79.0*sn1old(i+2,j-2)/432.0
    &        +31.0*sn1old(i+1,j-2)/54.0
    &        -707.0*sn1old(i,j-2)/864.0
    &        +31.0*sn1old(i-1,j-2)/54.0
    &        -79.0*sn1old(i-2,j-2)/432.0
    &        +sn1old(i-3,j-2)/54.0
    &        -sn1old(i-4,j-2)/1728.0

     dy4dx2old= -sn1old(i+2,j+4)/1728.0 +
sn1old(i+2,j+3)/54.0
```

```
     &          -79.0*sn1old(i+2,j+2)/432.0
     &          +31.0*sn1old(i+2,j+1)/54.0
     &          -707.0*sn1old(i+2,j)/864.0
     &          +31.0*sn1old(i+2,j-1)/54.0
     &          -79.0*sn1old(i+2,j-2)/432.0
     &          +sn1old(i+2,j-3)/54.0
     &          -sn1old(i+2,j-4)/1728.0

     &          +16.0*sn1old(i+1,j+4)/1728.0
     &          -16.0*sn1old(i+1,j+3)/54.0
     &          +16.0*79.0*sn1old(i+1,j+2)/432.0
     &          -16.0*31.0*sn1old(i+1,j+1)/54.0
     &          +16.0*707.0*sn1old(i+1,j)/864.0
     &          -16.0*31.0*sn1old(i+1,j-1)/54.0
     &          +16.0*79.0*sn1old(i+1,j-2)/432.0
     &          -16.0*sn1old(i+1,j-3)/54.0
     &          +16.0*sn1old(i+1,j-4)/1728.0

     &          -30.0*sn1old(i,j+4)/1728.0
     &          +30.0*sn1old(i,j+3)/54.0
     &          -30.0*79.0*sn1old(i,j+2)/432.0
     &          +30.0*31.0*sn1old(i,j+1)/54.0
     &          -30.0*707.0*sn1old(i,j)/864.0
     &          +30.0*31.0*sn1old(i,j-1)/54.0
     &          -30.0*79.0*sn1old(i,j-2)/432.0
     &          +30.0*sn1old(i,j-3)/54.0
     &          -30.0*sn1old(i,j-4)/1728.0

     &          +16.0*sn1old(i-1,j+4)/1728.0
     &          -16.0*sn1old(i-1,j+3)/54.0
     &          +16.0*79.0*sn1old(i-1,j+2)/432.0
     &          -16.0*31.0*sn1old(i-1,j+1)/54.0
     &          +16.0*707.0*sn1old(i-1,j)/864.0
     &          -16.0*31.0*sn1old(i-1,j-1)/54.0
     &          +16.0*79.0*sn1old(i-1,j-2)/432.0
     &          -16.0*sn1old(i-1,j-3)/54.0
     &          +16.0*sn1old(i-1,j-4)/1728.0

     &          -sn1old(i-2,j+4)/1728.0 + sn1old(i-
2,j+3)/54.0
     &          -79.0*sn1old(i-2,j+2)/432.0
     &          +31.0*sn1old(i-2,j+1)/54.0
     &          -707.0*sn1old(i-2,j)/864.0
     &          +31.0*sn1old(i-2,j-1)/54.0
     &          -79.0*sn1old(i-2,j-2)/432.0
     &          +sn1old(i-2,j-3)/54.0
     &          -sn1old(i-2,j-4)/1728.0

       dy2dx2old= (sn1old(i+2,j+2) -
16.0*sn1old(i+1,j+2)
     &          +30.0*sn1old(i,j+2)
     &          -16.0*sn1old(i-1,j+2)
     &          +sn1old(i-2,j+2)

     &          -16.0*sn1old(i+2,j+1) +
16.0*16.0*sn1old(i+1,j+1)
     &          -16.0*30.0*sn1old(i,j+1)
     &          +16.0*16.0*sn1old(i-1,j+1)
     &          -16.0*sn1old(i-2,j+1)

     &          +30.0*sn1old(i+2,j) -
30.0*16.0*sn1old(i+1,j)
     &          +30.0*30.0*sn1old(i,j)
     &          -30.0*16.0*sn1old(i-1,j)
     &          +30.0*sn1old(i-2,j)

     &          -16.0*sn1old(i+2,j-1) +
16.0*16.0*sn1old(i+1,j-1)
     &          -16.0*30.0*sn1old(i,j-1)
     &          +16.0*16.0*sn1old(i-1,j-1)
     &          -16.0*sn1old(i-2,j-1)

     &          +sn1old(i+2,j-2) - 16.0*sn1old(i+1,j-2)
     &          +30.0*sn1old(i,j-2)
     &          -16.0*sn1old(i-1,j-2)
     &          +sn1old(i-2,j-2))/144.0

       dy2dx2= (sn(i+2,j+2) - 16.0*sn(i+1,j+2)
     &          +30.0*sn(i,j+2)
     &          -16.0*sn(i-1,j+2)
     &          +sn(i-2,j+2)

     &          -16.0*sn(i+2,j+1) +
16.0*16.0*sn(i+1,j+1)
     &          -16.0*30.0*sn(i,j+1)
     &          +16.0*16.0*sn(i-1,j+1)
     &          -16.0*sn(i-2,j+1)

     &          +30.0*sn(i+2,j) - 30.0*16.0*sn(i+1,j)
     &          +30.0*30.0*sn(i,j)
     &          -30.0*16.0*sn(i-1,j)
     &          +30.0*sn(i-2,j)

     &          -16.0*sn(i+2,j-1) + 16.0*16.0*sn(i+1,j-1)
     &          -16.0*30.0*sn(i,j-1)
     &          +16.0*16.0*sn(i-1,j-1)
     &          -16.0*sn(i-2,j-1)

     &          +sn(i+2,j-2) - 16.0*sn(i+1,j-2)
     &          +30.0*sn(i,j-2)
     &          -16.0*sn(i-1,j-2)
     &          +sn(i-2,j-2))/144.0

       dx4dy2=    -sn(i+4,j+2)/1728.0 + sn(i+3,j+2)/54.0
     &          -79.0*sn(i+2,j+2)/432.0
     &          +31.0*sn(i+1,j+2)/54.0
     &          -707.0*sn(i,j+2)/864.0
     &          +31.0*sn(i-1,j+2)/54.0
     &          -79.0*sn(i-2,j+2)/432.0
     &          +sn(i-3,j+2)/54.0
     &          -sn(i-4,j+2)/1728.0

     &          +16.0*sn(i+4,j+1)/1728.0
     &          -16.0*sn(i+3,j+1)/54.0
     &          +16.0*79.0*sn(i+2,j+1)/432.0
     &          -16.0*31.0*sn(i+1,j+1)/54.0
     &          +16.0*707.0*sn(i,j+1)/864.0
     &          -16.0*31.0*sn(i-1,j+1)/54.0
     &          +16.0*79.0*sn(i-2,j+1)/432.0
     &          -16.0*sn(i-3,j+1)/54.0
     &          +16.0*sn(i-4,j+1)/1728.0

     &          -30.0*sn(i+4,j)/1728.0
     &          +30.0*sn(i+3,j)/54.0
```

```
&       -30.0*79.0*sn(i+2,j)/432.0
&       +30.0*31.0*sn(i+1,j)/54.0
&       -30.0*707.0*sn(i,j)/864.0
&       +30.0*31.0*sn(i-1,j)/54.0
&       -30.0*79.0*sn(i-2,j)/432.0
&       +30.0*sn(i-3,j)/54.0
&       -30.0*sn(i-4,j)/1728.0

&       +16.0*sn(i+4,j-1)/1728.0
&       -16.0*sn(i+3,j-1)/54.0
&       +16.0*79.0*sn(i+2,j-1)/432.0
&       -16.0*31.0*sn(i+1,j-1)/54.0
&       +16.0*707.0*sn(i,j-1)/864.0
&       -16.0*31.0*sn(i-1,j-1)/54.0
&       +16.0*79.0*sn(i-2,j-1)/432.0
&       -16.0*sn(i-3,j-1)/54.0
&       +16.0*sn(i-4,j-1)/1728.0

&       -sn(i+4,j-2)/1728.0 + sn(i+3,j-2)/54.0
&       -79.0*sn(i+2,j-2)/432.0
&       +31.0*sn(i+1,j-2)/54.0
&       -707.0*sn(i,j-2)/864.0
&       +31.0*sn(i-1,j-2)/54.0
&       -79.0*sn(i-2,j-2)/432.0
&       +sn(i-3,j-2)/54.0
&       -sn(i-4,j-2)/1728.0

  dy4dx2=   -sn(i+2,j+4)/1728.0 + sn(i+2,j+3)/54.0
&       -79.0*sn(i+2,j+2)/432.0
&       +31.0*sn(i+2,j+1)/54.0
&       -707.0*sn(i+2,j)/864.0
&       +31.0*sn(i+2,j-1)/54.0
&       -79.0*sn(i+2,j-2)/432.0
&       +sn(i+2,j-3)/54.0
&       -sn(i+2,j-4)/1728.0

&       +16.0*sn(i+1,j+4)/1728.0
&       -16.0*sn(i+1,j+3)/54.0
&       +16.0*79.0*sn(i+1,j+2)/432.0
&       -16.0*31.0*sn(i+1,j+1)/54.0
&       +16.0*707.0*sn(i+1,j)/864.0
&       -16.0*31.0*sn(i+1,j-1)/54.0
&       +16.0*79.0*sn(i+1,j-2)/432.0
&       -16.0*sn(i+1,j-3)/54.0
&       +16.0*sn(i+1,j-4)/1728.0

&       -30.0*sn(i,j+4)/1728.0
&       +30.0*sn(i,j+3)/54.0
&       -30.0*79.0*sn(i,j+2)/432.0
&       +30.0*31.0*sn(i,j+1)/54.0
&       -30.0*707.0*sn(i,j)/864.0
&       +30.0*31.0*sn(i,j-1)/54.0
&       -30.0*79.0*sn(i,j-2)/432.0
&       +30.0*sn(i,j-3)/54.0
&       -30.0*sn(i,j-4)/1728.0

&       +16.0*sn(i-1,j+4)/1728.0
&       -16.0*sn(i-1,j+3)/54.0
&       +16.0*79.0*sn(i-1,j+2)/432.0
&       -16.0*31.0*sn(i-1,j+1)/54.0
&       +16.0*707.0*sn(i-1,j)/864.0
&       -16.0*31.0*sn(i-1,j-1)/54.0
&       +16.0*79.0*sn(i-1,j-2)/432.0
&       -16.0*sn(i-1,j-3)/54.0
&       +16.0*sn(i-1,j-4)/1728.0

&       -sn(i-2,j+4)/1728.0 + sn(i-2,j+3)/54.0
&       -79.0*sn(i-2,j+2)/432.0
&       +31.0*sn(i-2,j+1)/54.0
&       -707.0*sn(i-2,j)/864.0
&       +31.0*sn(i-2,j-1)/54.0
&       -79.0*sn(i-2,j-2)/432.0
&       +sn(i-2,j-3)/54.0
&       -sn(i-2,j-4)/1728.0

    dx2old=(-sn1old(i+2,j)+16.0*sn1old(i+1,j)-
30.0*sn1old(i,j)
   &       +16.0*sn1old(i-1,j)
   &       -sn1old(i-2,j))/12.0

    dy2old=(-sn1old(i,j+2)+16.0*sn1old(i,j+1)-
30.0*sn1old(i,j)
   &       +16.0*sn1old(i,j-1)
   &       -sn1old(i,j-2))/12.0

    dx4old= sn1old(i+4,j)/144.0 -
2.0*sn1old(i+3,j)/9.0
   &       +79.0*sn1old(i+2,j)/36.0 -
62.0*sn1old(i+1,j)/9.0
   &       +707.0*sn1old(i,j)/72.0  - 62.0*sn1old(i-
1,j)/9.0
   &       +79.0*sn1old(i-2,j)/36.0 - 2.0*sn1old(i-
3,j)/9.0
   &       +sn1old(i-4,j)/144.0

    dy4old=sn1old(i,j+4)/144.0 -
2.0*sn1old(i,j+3)/9.0
   &       +79.0*sn1old(i,j+2)/36.0 -
62.0*sn1old(i,j+1)/9.0
   &       +707.0*sn1old(i,j)/72.0  - 62.0*sn1old(i,j-
1)/9.0
   &       +79.0*sn1old(i,j-2)/36.0 - 2.0*sn1old(i,j-
3)/9.0
   &       +sn1old(i,j-4)/144.0

    dx6old= -sn1old(i+6,j)/1728.0 +
sn1old(i+5,j)/36.0
   &       -143.0*sn1old(i+4,j)/288.0 +
439.0*sn1old(i+3,j)/108.0
   &       -3031.0*sn1old(i+2,j)/192.0 +
203.0*sn1old(i+1,j)/6.0
   &       -6233.0*sn1old(i,j)/144.0 +
203.0*sn1old(i-1,j)/6.0
   &       -3031.0*sn1old(i-2,j)/192.0 +
439.0*sn1old(i-3,j)/108.0
   &       -143.0*sn1old(i-4,j)/288.0 + sn1old(i-
5,j)/36.0
   &       -sn1old(i-6,j)/1728.0

    dy6old=-sn1old(i,j+6)/1728.0 +
sn1old(i,j+5)/36.0
   &       -143.0*sn1old(i,j+4)/288.0 +
439.0*sn1old(i,j+3)/108.0
```

```
   &        -3031.0*sn1old(i,j+2)/192.0 +
203.0*sn1old(i,j+1)/6.0
   &        -6233.0*sn1old(i,j)/144.0 +
203.0*sn1old(i,j-1)/6.0
   &        -3031.0*sn1old(i,j-2)/192.0 +
439.0*sn1old(i,j-3)/108.0
   &        -143.0*sn1old(i,j-4)/288.0 + sn1old(i,j-
5)/36.0
   &        -sn1old(i,j-6)/1728.0

     dx2= (-sn(i+2,j)+16.0*sn(i+1,j)-
30.0*sn(i,j)+16.0*sn(i-1,j)
   &        -sn(i-2,j))/12.0

     dy2= (-sn(i,j+2)+16.0*sn(i,j+1)-
30.0*sn(i,j)+16.0*sn(i,j-1)
   &        -sn(i,j-2))/12.0

     dx4=sn(i+4,j)/144.0 - 2.0*sn(i+3,j)/9.0
   &        +79.0*sn(i+2,j)/36.0 - 62.0*sn(i+1,j)/9.0
   &        +707.0*sn(i,j)/72.0  - 62.0*sn(i-1,j)/9.0
   &        +79.0*sn(i-2,j)/36.0 - 2.0*sn(i-3,j)/9.0
   &        +sn(i-4,j)/144.0

     dy4=sn(i,j+4)/144.0 - 2.0*sn(i,j+3)/9.0
   &        +79.0*sn(i,j+2)/36.0 - 62.0*sn(i,j+1)/9.0
   &        +707.0*sn(i,j)/72.0  - 62.0*sn(i,j-1)/9.0
   &        +79.0*sn(i,j-2)/36.0 - 2.0*sn(i,j-3)/9.0
   &        +sn(i,j-4)/144.0

     dx6= -sn(i+6,j)/1728.0 + sn(i+5,j)/36.0
   &        -143.0*sn(i+4,j)/288.0 +
439.0*sn(i+3,j)/108.0
   &        -3031.0*sn(i+2,j)/192.0 +
203.0*sn(i+1,j)/6.0
   &        -6233.0*sn(i,j)/144.0 + 203.0*sn(i-1,j)/6.0
   &        -3031.0*sn(i-2,j)/192.0 + 439.0*sn(i-
3,j)/108.0
   &        -143.0*sn(i-4,j)/288.0 + sn(i-5,j)/36.0
   &        -sn(i-6,j)/1728.0

     dy6= -sn(i,j+6)/1728.0 + sn(i,j+5)/36.0
   &        -143.0*sn(i,j+4)/288.0 +
439.0*sn(i,j+3)/108.0
   &        -3031.0*sn(i,j+2)/192.0 +
203.0*sn(i,j+1)/6.0
   &        -6233.0*sn(i,j)/144.0 + 203.0*sn(i,j-1)/6.0
   &        -3031.0*sn(i,j-2)/192.0 + 439.0*sn(i,j-
3)/108.0
   &        -143.0*sn(i,j-4)/288.0 + sn(i,j-5)/36.0
   &        -sn(i,j-6)/1728.0

     rn1(i,j) = rn(i,j) + dtx*dx2old/2.0 + dtx*dx2/2.0
   &          + dtx*dy2old/2.0 + dtx*dy2/2.0
   &          + dt*c1*sn1old(i,j)/2.0 + dt*c1*sn(i,j)/2.0
   &          + dtx3*dx6old/24.0 + dtx3*dx6/24.0
   &          + dtx3*dx4dy2old/8.0 + dtx3*dx4dy2/8.0
   &          + dtx3*dy4dx2old/8.0 + dtx3*dy4dx2/8.0
   &          + dtx2*dt*c1*dy2dx2old/4.0 +
dtx2*dt*c1*dy2dx2/4.0
   &          + dtx2*dt*c1*dx4old/8.0 +
dtx2*dt*c1*dx4/8.0
   &          + dtx3*dy6old/24.0 + dtx3*dy6/24.0
   &          + dtx2*dt*c1*dy4old/8.0 +
dtx2*dt*c1*dy4/8.0
   &          + dtx*dt2*c12*dx2old/8.0 +
dtx*dt2*c12*dx2/8.0
   &          + dtx*dt2*c12*dy2old/8.0 +
dtx*dt2*c12*dy2/8.0
   &          + dt3*c13*sn1old(i,j)/24.0 +
dt3*c13*sn(i,j)/24.0
     enddo
     enddo

     do i=6,nx-6
     do j=6,ny-6
      c1=4.0*(rbn(i,j)*rbn(i,j)+sbn(i,j)*sbn(i,j))**mp
      c12=c1*c1
      c13=c12*c1

       dx4dy2old1= -rn1(i+4,j+2)/1728.0 +
rn1(i+3,j+2)/54.0
   &          -79.0*rn1(i+2,j+2)/432.0
   &          +31.0*rn1(i+1,j+2)/54.0
   &          -707.0*rn1(i,j+2)/864.0
   &          +31.0*rn1(i-1,j+2)/54.0
   &          -79.0*rn1(i-2,j+2)/432.0
   &          +rn1(i-3,j+2)/54.0
   &          -rn1(i-4,j+2)/1728.0

   &          +16.0*rn1(i+4,j+1)/1728.0
   &          -16.0*rn1(i+3,j+1)/54.0
   &          +16.0*79.0*rn1(i+2,j+1)/432.0
   &          -16.0*31.0*rn1(i+1,j+1)/54.0
   &          +16.0*707.0*rn1(i,j+1)/864.0
   &          -16.0*31.0*rn1(i-1,j+1)/54.0
   &          +16.0*79.0*rn1(i-2,j+1)/432.0
   &          -16.0*rn1(i-3,j+1)/54.0
   &          +16.0*rn1(i-4,j+1)/1728.0

   &          -30.0*rn1(i+4,j)/1728.0
   &          +30.0*rn1(i+3,j)/54.0
   &          -30.0*79.0*rn1(i+2,j)/432.0
   &          +30.0*31.0*rn1(i+1,j)/54.0
   &          -30.0*707.0*rn1(i,j)/864.0
   &          +30.0*31.0*rn1(i-1,j)/54.0
   &          -30.0*79.0*rn1(i-2,j)/432.0
   &          +30.0*rn1(i-3,j)/54.0
   &          -30.0*rn1(i-4,j)/1728.0

   &          +16.0*rn1(i+4,j-1)/1728.0
   &          -16.0*rn1(i+3,j-1)/54.0
   &          +16.0*79.0*rn1(i+2,j-1)/432.0
   &          -16.0*31.0*rn1(i+1,j-1)/54.0
   &          +16.0*707.0*rn1(i,j-1)/864.0
   &          -16.0*31.0*rn1(i-1,j-1)/54.0
   &          +16.0*79.0*rn1(i-2,j-1)/432.0
   &          -16.0*rn1(i-3,j-1)/54.0
   &          +16.0*rn1(i-4,j-1)/1728.0

   &          -rn1(i+4,j-2)/1728.0 + rn1(i+3,j-2)/54.0
   &          -79.0*rn1(i+2,j-2)/432.0
   &          +31.0*rn1(i+1,j-2)/54.0
   &          -707.0*rn1(i,j-2)/864.0
```

```
     &        +31.0*rn1(i-1,j-2)/54.0
     &        -79.0*rn1(i-2,j-2)/432.0
     &        +rn1(i-3,j-2)/54.0
     &        -rn1(i-4,j-2)/1728.0

       dy4dx2old1= -rn1(i+2,j+4)/1728.0 +
rn1(i+2,j+3)/54.0
     &        -79.0*rn1(i+2,j+2)/432.0
     &        +31.0*rn1(i+2,j+1)/54.0
     &        -707.0*rn1(i+2,j)/864.0
     &        +31.0*rn1(i+2,j-1)/54.0
     &        -79.0*rn1(i+2,j-2)/432.0
     &        +rn1(i+2,j-3)/54.0
     &        -rn1(i+2,j-4)/1728.0

     &        +16.0*rn1(i+1,j+4)/1728.0
     &        -16.0*rn1(i+1,j+3)/54.0
     &        +16.0*79.0*rn1(i+1,j+2)/432.0
     &        -16.0*31.0*rn1(i+1,j+1)/54.0
     &        +16.0*707.0*rn1(i+1,j)/864.0
     &        -16.0*31.0*rn1(i+1,j-1)/54.0
     &        +16.0*79.0*rn1(i+1,j-2)/432.0
     &        -16.0*rn1(i+1,j-3)/54.0
     &        +16.0*rn1(i+1,j-4)/1728.0

     &        -30.0*rn1(i,j+4)/1728.0
     &        +30.0*rn1(i,j+3)/54.0
     &        -30.0*79.0*rn1(i,j+2)/432.0
     &        +30.0*31.0*rn1(i,j+1)/54.0
     &        -30.0*707.0*rn1(i,j)/864.0
     &        +30.0*31.0*rn1(i,j-1)/54.0
     &        -30.0*79.0*rn1(i,j-2)/432.0
     &        +30.0*rn1(i,j-3)/54.0
     &        -30.0*rn1(i,j-4)/1728.0

     &        +16.0*rn1(i-1,j+4)/1728.0
     &        -16.0*rn1(i-1,j+3)/54.0
     &        +16.0*79.0*rn1(i-1,j+2)/432.0
     &        -16.0*31.0*rn1(i-1,j+1)/54.0
     &        +16.0*707.0*rn1(i-1,j)/864.0
     &        -16.0*31.0*rn1(i-1,j-1)/54.0
     &        +16.0*79.0*rn1(i-1,j-2)/432.0
     &        -16.0*rn1(i-1,j-3)/54.0
     &        +16.0*rn1(i-1,j-4)/1728.0

     &        -rn1(i-2,j+4)/1728.0 + rn1(i-2,j+3)/54.0
     &        -79.0*rn1(i-2,j+2)/432.0
     &        +31.0*rn1(i-2,j+1)/54.0
     &        -707.0*rn1(i-2,j)/864.0
     &        +31.0*rn1(i-2,j-1)/54.0
     &        -79.0*rn1(i-2,j-2)/432.0
     &        +rn1(i-2,j-3)/54.0
     &        -rn1(i-2,j-4)/1728.0

       dy2dx2old1= (rn1(i+2,j+2) - 16.0*rn1(i+1,j+2)
     &        +30.0*rn1(i,j+2)
     &        -16.0*rn1(i-1,j+2)
     &        +rn1(i-2,j+2)

     &        -16.0*rn1(i+2,j+1) +
16.0*16.0*rn1(i+1,j+1)
     &        -16.0*30.0*rn1(i,j+1)
     &        +16.0*16.0*rn1(i-1,j+1)
     &        -16.0*rn1(i-2,j+1)

     &        +30.0*rn1(i+2,j) - 30.0*16.0*rn1(i+1,j)
     &        +30.0*30.0*rn1(i,j)
     &        -30.0*16.0*rn1(i-1,j)
     &        +30.0*rn1(i-2,j)

     &        -16.0*rn1(i+2,j-1) + 16.0*16.0*rn1(i+1,j-
1)
     &        -16.0*30.0*rn1(i,j-1)
     &        +16.0*16.0*rn1(i-1,j-1)
     &        -16.0*rn1(i-2,j-1)

     &        +rn1(i+2,j-2) - 16.0*rn1(i+1,j-2)
     &        +30.0*rn1(i,j-2)
     &        -16.0*rn1(i-1,j-2)
     &        +rn1(i-2,j-2))/144.0

       dy2dx21= (rn(i+2,j+2) - 16.0*rn(i+1,j+2)
     &        +30.0*rn(i,j+2)
     &        -16.0*rn(i-1,j+2)
     &        +rn(i-2,j+2)

     &        -16.0*rn(i+2,j+1) +
16.0*16.0*rn(i+1,j+1)
     &        -16.0*30.0*rn(i,j+1)
     &        +16.0*16.0*rn(i-1,j+1)
     &        -16.0*rn(i-2,j+1)

     &        +30.0*rn(i+2,j) - 30.0*16.0*rn(i+1,j)
     &        +30.0*30.0*rn(i,j)
     &        -30.0*16.0*rn(i-1,j)
     &        +30.0*rn(i-2,j)

     &        -16.0*rn(i+2,j-1) + 16.0*16.0*rn(i+1,j-1)
     &        -16.0*30.0*rn(i,j-1)
     &        +16.0*16.0*rn(i-1,j-1)
     &        -16.0*rn(i-2,j-1)

     &        +rn(i+2,j-2) - 16.0*rn(i+1,j-2)
     &        +30.0*rn(i,j-2)
     &        -16.0*rn(i-1,j-2)
     &        +rn(i-2,j-2))/144.0

       dx4dy21=    -rn(i+4,j+2)/1728.0 +
rn(i+3,j+2)/54.0
     &        -79.0*rn(i+2,j+2)/432.0
     &        +31.0*rn(i+1,j+2)/54.0
     &        -707.0*rn(i,j+2)/864.0
     &        +31.0*rn(i-1,j+2)/54.0
     &        -79.0*rn(i-2,j+2)/432.0
     &        +rn(i-3,j+2)/54.0
     &        -rn(i-4,j+2)/1728.0

     &        +16.0*rn(i+4,j+1)/1728.0
     &        -16.0*rn(i+3,j+1)/54.0
     &        +16.0*79.0*rn(i+2,j+1)/432.0
     &        -16.0*31.0*rn(i+1,j+1)/54.0
     &        +16.0*707.0*rn(i,j+1)/864.0
     &        -16.0*31.0*rn(i-1,j+1)/54.0
     &        +16.0*79.0*rn(i-2,j+1)/432.0
```

```
     &          -16.0*rn(i-3,j+1)/54.0
     &          +16.0*rn(i-4,j+1)/1728.0

     &          -30.0*rn(i+4,j)/1728.0
     &          +30.0*rn(i+3,j)/54.0
     &          -30.0*79.0*rn(i+2,j)/432.0
     &          +30.0*31.0*rn(i+1,j)/54.0
     &          -30.0*707.0*rn(i,j)/864.0
     &          +30.0*31.0*rn(i-1,j)/54.0
     &          -30.0*79.0*rn(i-2,j)/432.0
     &          +30.0*rn(i-3,j)/54.0
     &          -30.0*rn(i-4,j)/1728.0

     &          +16.0*rn(i+4,j-1)/1728.0
     &          -16.0*rn(i+3,j-1)/54.0
     &          +16.0*79.0*rn(i+2,j-1)/432.0
     &          -16.0*31.0*rn(i+1,j-1)/54.0
     &          +16.0*707.0*rn(i,j-1)/864.0
     &          -16.0*31.0*rn(i-1,j-1)/54.0
     &          +16.0*79.0*rn(i-2,j-1)/432.0
     &          -16.0*rn(i-3,j-1)/54.0
     &          +16.0*rn(i-4,j-1)/1728.0

     &          -rn(i+4,j-2)/1728.0 + rn(i+3,j-2)/54.0
     &          -79.0*rn(i+2,j-2)/432.0
     &          +31.0*rn(i+1,j-2)/54.0
     &          -707.0*rn(i,j-2)/864.0
     &          +31.0*rn(i-1,j-2)/54.0
     &          -79.0*rn(i-2,j-2)/432.0
     &          +rn(i-3,j-2)/54.0
     &          -rn(i-4,j-2)/1728.0

       dy4dx21=   -rn(i+2,j+4)/1728.0 +
rn(i+2,j+3)/54.0
     &          -79.0*rn(i+2,j+2)/432.0
     &          +31.0*rn(i+2,j+1)/54.0
     &          -707.0*rn(i+2,j)/864.0
     &          +31.0*rn(i+2,j-1)/54.0
     &          -79.0*rn(i+2,j-2)/432.0
     &          +rn(i+2,j-3)/54.0
     &          -rn(i+2,j-4)/1728.0

     &          +16.0*rn(i+1,j+4)/1728.0
     &          -16.0*rn(i+1,j+3)/54.0
     &          +16.0*79.0*rn(i+1,j+2)/432.0
     &          -16.0*31.0*rn(i+1,j+1)/54.0
     &          +16.0*707.0*rn(i+1,j)/864.0
     &          -16.0*31.0*rn(i+1,j-1)/54.0
     &          +16.0*79.0*rn(i+1,j-2)/432.0
     &          -16.0*rn(i+1,j-3)/54.0
     &          +16.0*rn(i+1,j-4)/1728.0

     &          -30.0*rn(i,j+4)/1728.0
     &          +30.0*rn(i,j+3)/54.0
     &          -30.0*79.0*rn(i,j+2)/432.0
     &          +30.0*31.0*rn(i,j+1)/54.0
     &          -30.0*707.0*rn(i,j)/864.0
     &          +30.0*31.0*rn(i,j-1)/54.0
     &          -30.0*79.0*rn(i,j-2)/432.0
     &          +30.0*rn(i,j-3)/54.0
     &          -30.0*rn(i,j-4)/1728.0

     &          +16.0*rn(i-1,j+4)/1728.0
     &          -16.0*rn(i-1,j+3)/54.0
     &          +16.0*79.0*rn(i-1,j+2)/432.0
     &          -16.0*31.0*rn(i-1,j+1)/54.0
     &          +16.0*707.0*rn(i-1,j)/864.0
     &          -16.0*31.0*rn(i-1,j-1)/54.0
     &          +16.0*79.0*rn(i-1,j-2)/432.0
     &          -16.0*rn(i-1,j-3)/54.0
     &          +16.0*rn(i-1,j-4)/1728.0

     &          -rn(i-2,j+4)/1728.0 + rn(i-2,j+3)/54.0
     &          -79.0*rn(i-2,j+2)/432.0
     &          +31.0*rn(i-2,j+1)/54.0
     &          -707.0*rn(i-2,j)/864.0
     &          +31.0*rn(i-2,j-1)/54.0
     &          -79.0*rn(i-2,j-2)/432.0
     &          +rn(i-2,j-3)/54.0
     &          -rn(i-2,j-4)/1728.0

       dx2old1=(-rn1(i+2,j)+16.0*rn1(i+1,j)-
30.0*rn1(i,j)
     &          +16.0*rn1(i-1,j)
     &          -rn1(i-2,j))/12.0

       dy2old1=(-rn1(i,j+2)+16.0*rn1(i,j+1)-
30.0*rn1(i,j)
     &          +16.0*rn1(i,j-1)
     &          -rn1(i,j-2))/12.0

       dx4old1=rn1(i+4,j)/144.0 - 2.0*rn1(i+3,j)/9.0
     &          +79.0*rn1(i+2,j)/36.0 - 62.0*rn1(i+1,j)/9.0
     &          +707.0*rn1(i,j)/72.0  - 62.0*rn1(i-1,j)/9.0
     &          +79.0*rn1(i-2,j)/36.0 - 2.0*rn1(i-3,j)/9.0
     &          +rn1(i-4,j)/144.0

       dy4old1=rn1(i,j+4)/144.0 - 2.0*rn1(i,j+3)/9.0
     &          +79.0*rn1(i,j+2)/36.0 - 62.0*rn1(i,j+1)/9.0
     &          +707.0*rn1(i,j)/72.0  - 62.0*rn1(i,j-1)/9.0
     &          +79.0*rn1(i,j-2)/36.0 - 2.0*rn1(i,j-3)/9.0
     &          +rn1(i,j-4)/144.0

       dx6old1=-rn1(i+6,j)/1728.0 + rn1(i+5,j)/36.0
     &          -143.0*rn1(i+4,j)/288.0 +
439.0*rn1(i+3,j)/108.0
     &          -3031.0*rn1(i+2,j)/192.0 +
203.0*rn1(i+1,j)/6.0
     &          -6233.0*rn1(i,j)/144.0 + 203.0*rn1(i-
1,j)/6.0
     &          -3031.0*rn1(i-2,j)/192.0 + 439.0*rn1(i-
3,j)/108.0
     &          -143.0*rn1(i-4,j)/288.0 + rn1(i-5,j)/36.0
     &          -rn1(i-6,j)/1728.0

       dy6old1=-rn1(i,j+6)/1728.0 + rn1(i,j+5)/36.0
     &          -143.0*rn1(i,j+4)/288.0 +
439.0*rn1(i,j+3)/108.0
     &          -3031.0*rn1(i,j+2)/192.0 +
203.0*rn1(i,j+1)/6.0
     &          -6233.0*rn1(i,j)/144.0 + 203.0*rn1(i,j-
1)/6.0
     &          -3031.0*rn1(i,j-2)/192.0 + 439.0*rn1(i,j-
3)/108.0
```

```
     &        -143.0*rn1(i,j-4)/288.0 + rn1(i,j-5)/36.0
     &        -rn1(i,j-6)/1728.0

       dx21= (-rn(i+2,j)+16.0*rn(i+1,j)-
30.0*rn(i,j)+16.0*rn(i-1,j)
     &       -rn(i-2,j))/12.0

       dy21= (-rn(i,j+2)+16.0*rn(i,j+1)-
30.0*rn(i,j)+16.0*rn(i,j-1)
     &       -rn(i,j-2))/12.0

       dx41=rn(i+4,j)/144.0 - 2.0*rn(i+3,j)/9.0
     &        +79.0*rn(i+2,j)/36.0 - 62.0*rn(i+1,j)/9.0
     &        +707.0*rn(i,j)/72.0  - 62.0*rn(i-1,j)/9.0
     &        +79.0*rn(i-2,j)/36.0 - 2.0*rn(i-3,j)/9.0
     &        +rn(i-4,j)/144.0

       dy41=rn(i,j+4)/144.0 - 2.0*rn(i,j+3)/9.0
     &        +79.0*rn(i,j+2)/36.0 - 62.0*rn(i,j+1)/9.0
     &        +707.0*rn(i,j)/72.0  - 62.0*rn(i,j-1)/9.0
     &        +79.0*rn(i,j-2)/36.0 - 2.0*rn(i,j-3)/9.0
     &        +rn(i,j-4)/144.0

       dx61= -rn(i+6,j)/1728.0 + rn(i+5,j)/36.0
     &        -143.0*rn(i+4,j)/288.0 +
439.0*rn(i+3,j)/108.0
     &        -3031.0*rn(i+2,j)/192.0 +
203.0*rn(i+1,j)/6.0
     &        -6233.0*rn(i,j)/144.0 + 203.0*rn(i-1,j)/6.0
     &        -3031.0*rn(i-2,j)/192.0 + 439.0*rn(i-
3,j)/108.0
     &        -143.0*rn(i-4,j)/288.0 + rn(i-5,j)/36.0
     &        -rn(i-6,j)/1728.0

       dy61= -rn(i,j+6)/1728.0 + rn(i,j+5)/36.0
     &        -143.0*rn(i,j+4)/288.0 +
439.0*rn(i,j+3)/108.0
     &        -3031.0*rn(i,j+2)/192.0 +
203.0*rn(i,j+1)/6.0
     &        -6233.0*rn(i,j)/144.0 + 203.0*rn(i,j-1)/6.0
     &        -3031.0*rn(i,j-2)/192.0 + 439.0*rn(i,j-
3)/108.0
     &        -143.0*rn(i,j-4)/288.0 + rn(i,j-5)/36.0
     &        -rn(i,j-6)/1728.0

        sn1(i,j) = sn(i,j) - dtx*dx2old1/2.0 -
dtx*dx21/2.0
     &         - dtx*dy2old1/2.0 - dtx*dy21/2.0
     &         - dt*c1*rn1(i,j)/2.0 - dt*c1*rn(i,j)/2.0
     &         - dtx3*dx6old1/24.0 - dtx3*dx61/24.0
     &         - dtx3*dx4dy2old1/8.0 -
dtx3*dx4dy21/8.0
     &         - dtx3*dy4dx2old1/8.0 -
dtx3*dy4dx21/8.0
     &         - dtx2*dt*c1*dy2dx2old1/4.0 -
dtx2*dt*c1*dy2dx21/4.0
     &         - dtx2*dt*c1*dx4old1/8.0 -
dtx2*dt*c1*dx41/8.0
     &         - dtx3*dy6old1/24.0 - dtx3*dy61/24.0
     &         - dtx2*dt*c1*dy4old1/8.0 -
dtx2*dt*c1*dy41/8.0
     &         - dtx*dt2*c12*dx2old1/8.0 -
dtx*dt2*c12*dx21/8.0
     &         - dtx*dt2*c12*dy2old1/8.0 -
dtx*dt2*c12*dy21/8.0
     &         - dt3*c13*rn1(i,j)/24.0 -
dt3*c13*rn(i,j)/24.0
      enddo
      enddo

      tol=1.0D-15
      err_max=0.0
      do i=6,nx-6
      do j=6,ny-6
      err = abs(sn1(i,j)-sn1old(i,j))
      if (err.gt.err_max) then
      err_max=err
      endif
      enddo
      enddo

      if (err_max.le.tol) goto 3
      do i=6,nx-6
      do j=6,ny-6
       sn1old(i,j)=sn1(i,j)
      enddo
      enddo
      goto 2

 3    do i=6,nx-6
      do j=6,ny-6
        sbn1old(i,j)=sbn(i,j)
      enddo
      enddo

      do i=0,nx
      do j=0,5
       rbn1(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
     &        +6.0D0*(it+0.5D0)*dt)
     &        /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
     &        +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))

       sbn1(i,j)= 2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
     &        +6.0D0*(it+0.5D0)*dt)
     &        /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
     &        +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))
       sbn1old(i,j)=sbn1(i,j)
      enddo
      enddo

      do i=0,nx
      do j=ny-5,ny
       rbn1(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
     &        +6.0D0*(it+0.5D0)*dt)
     &        /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
```

```
   &        +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))

     sbn1(i,j)= 2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
   &        +6.0D0*(it+0.5D0)*dt)
   &        /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
   &        +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))
     sbn1old(i,j)=sbn1(i,j)
    enddo
    enddo

    do i=0,5
    do j=6,ny-6
     rbn1(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
   &        +6.0D0*(it+0.5D0)*dt)
   &        /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
   &        +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))

     sbn1(i,j)= 2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
   &        +6.0D0*(it+0.5D0)*dt)
   &        /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
   &        +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))
     sbn1old(i,j)=sbn1(i,j)
    enddo
    enddo

    do i=nx-5,nx
    do j=6,ny-6
     rbn1(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
   &        +6.0D0*(it+0.5D0)*dt)
   &        /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
   &        +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))

     sbn1(i,j)= 2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
   &        +6.0D0*(it+0.5D0)*dt)
   &        /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
   &        +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))
     sbn1old(i,j)=sbn1(i,j)
    enddo
    enddo

5   do i=6,nx-6
    do j=6,ny-6
     cb1=4.0*(rn1(i,j)*rn1(i,j)+sn1(i,j)*sn1(i,j))**mp
     cb12=cb1*cb1
     cb13=cb12*cb1

      dbx4dy2old= -sbn1old(i+4,j+2)/1728.0 +
sbn1old(i+3,j+2)/54.0
   &        -79.0*sbn1old(i+2,j+2)/432.0
   &        +31.0*sbn1old(i+1,j+2)/54.0
   &        -707.0*sbn1old(i,j+2)/864.0
   &        +31.0*sbn1old(i-1,j+2)/54.0
   &        -79.0*sbn1old(i-2,j+2)/432.0
   &        +sbn1old(i-3,j+2)/54.0
   &        -sbn1old(i-4,j+2)/1728.0

   &        +16.0*sbn1old(i+4,j+1)/1728.0
   &        -16.0*sbn1old(i+3,j+1)/54.0
   &        +16.0*79.0*sbn1old(i+2,j+1)/432.0
   &        -16.0*31.0*sbn1old(i+1,j+1)/54.0
   &        +16.0*707.0*sbn1old(i,j+1)/864.0
   &        -16.0*31.0*sbn1old(i-1,j+1)/54.0
   &        +16.0*79.0*sbn1old(i-2,j+1)/432.0
   &        -16.0*sbn1old(i-3,j+1)/54.0
   &        +16.0*sbn1old(i-4,j+1)/1728.0

   &        -30.0*sbn1old(i+4,j)/1728.0
   &        +30.0*sbn1old(i+3,j)/54.0
   &        -30.0*79.0*sbn1old(i+2,j)/432.0
   &        +30.0*31.0*sbn1old(i+1,j)/54.0
   &        -30.0*707.0*sbn1old(i,j)/864.0
   &        +30.0*31.0*sbn1old(i-1,j)/54.0
   &        -30.0*79.0*sbn1old(i-2,j)/432.0
   &        +30.0*sbn1old(i-3,j)/54.0
   &        -30.0*sbn1old(i-4,j)/1728.0

   &        +16.0*sbn1old(i+4,j-1)/1728.0
   &        -16.0*sbn1old(i+3,j-1)/54.0
   &        +16.0*79.0*sbn1old(i+2,j-1)/432.0
   &        -16.0*31.0*sbn1old(i+1,j-1)/54.0
   &        +16.0*707.0*sbn1old(i,j-1)/864.0
   &        -16.0*31.0*sbn1old(i-1,j-1)/54.0
   &        +16.0*79.0*sbn1old(i-2,j-1)/432.0
   &        -16.0*sbn1old(i-3,j-1)/54.0
   &        +16.0*sbn1old(i-4,j-1)/1728.0

   &        -sbn1old(i+4,j-2)/1728.0 + sbn1old(i+3,j-
2)/54.0
   &        -79.0*sbn1old(i+2,j-2)/432.0
   &        +31.0*sbn1old(i+1,j-2)/54.0
   &        -707.0*sbn1old(i,j-2)/864.0
   &        +31.0*sbn1old(i-1,j-2)/54.0
   &        -79.0*sbn1old(i-2,j-2)/432.0
   &        +sbn1old(i-3,j-2)/54.0
   &        -sbn1old(i-4,j-2)/1728.0

      dby4dx2old= -sbn1old(i+2,j+4)/1728.0 +
sbn1old(i+2,j+3)/54.0
   &        -79.0*sbn1old(i+2,j+2)/432.0
   &        +31.0*sbn1old(i+2,j+1)/54.0
   &        -707.0*sbn1old(i+2,j)/864.0
   &        +31.0*sbn1old(i+2,j-1)/54.0
   &        -79.0*sbn1old(i+2,j-2)/432.0
   &        +sbn1old(i+2,j-3)/54.0
   &        -sbn1old(i+2,j-4)/1728.0

   &        +16.0*sbn1old(i+1,j+4)/1728.0
   &        -16.0*sbn1old(i+1,j+3)/54.0
```

```
     &          +16.0*79.0*sbn1old(i+1,j+2)/432.0
     &          -16.0*31.0*sbn1old(i+1,j+1)/54.0
     &          +16.0*707.0*sbn1old(i+1,j)/864.0
     &          -16.0*31.0*sbn1old(i+1,j-1)/54.0
     &          +16.0*79.0*sbn1old(i+1,j-2)/432.0
     &          -16.0*sbn1old(i+1,j-3)/54.0
     &          +16.0*sbn1old(i+1,j-4)/1728.0

     &          -30.0*sbn1old(i,j+4)/1728.0
     &          +30.0*sbn1old(i,j+3)/54.0
     &          -30.0*79.0*sbn1old(i,j+2)/432.0
     &          +30.0*31.0*sbn1old(i,j+1)/54.0
     &          -30.0*707.0*sbn1old(i,j)/864.0
     &          +30.0*31.0*sbn1old(i,j-1)/54.0
     &          -30.0*79.0*sbn1old(i,j-2)/432.0
     &          +30.0*sbn1old(i,j-3)/54.0
     &          -30.0*sbn1old(i,j-4)/1728.0

     &          +16.0*sbn1old(i-1,j+4)/1728.0
     &          -16.0*sbn1old(i-1,j+3)/54.0
     &          +16.0*79.0*sbn1old(i-1,j+2)/432.0
     &          -16.0*31.0*sbn1old(i-1,j+1)/54.0
     &          +16.0*707.0*sbn1old(i-1,j)/864.0
     &          -16.0*31.0*sbn1old(i-1,j-1)/54.0
     &          +16.0*79.0*sbn1old(i-1,j-2)/432.0
     &          -16.0*sbn1old(i-1,j-3)/54.0
     &          +16.0*sbn1old(i-1,j-4)/1728.0

     &          -sbn1old(i-2,j+4)/1728.0 + sbn1old(i-
2,j+3)/54.0
     &          -79.0*sbn1old(i-2,j+2)/432.0
     &          +31.0*sbn1old(i-2,j+1)/54.0
     &          -707.0*sbn1old(i-2,j)/864.0
     &          +31.0*sbn1old(i-2,j-1)/54.0
     &          -79.0*sbn1old(i-2,j-2)/432.0
     &          +sbn1old(i-2,j-3)/54.0
     &          -sbn1old(i-2,j-4)/1728.0

       dby2dx2old= (sbn1old(i+2,j+2) -
16.0*sbn1old(i+1,j+2)
     &          +30.0*sbn1old(i,j+2)
     &          -16.0*sbn1old(i-1,j+2)
     &          +sbn1old(i-2,j+2)

     &          -16.0*sbn1old(i+2,j+1) +
16.0*16.0*sbn1old(i+1,j+1)
     &          -16.0*30.0*sbn1old(i,j+1)
     &          +16.0*16.0*sbn1old(i-1,j+1)
     &          -16.0*sbn1old(i-2,j+1)

     &          +30.0*sbn1old(i+2,j) -
30.0*16.0*sbn1old(i+1,j)
     &          +30.0*30.0*sbn1old(i,j)
     &          -30.0*16.0*sbn1old(i-1,j)
     &          +30.0*sbn1old(i-2,j)

     &          -16.0*sbn1old(i+2,j-1) +
16.0*16.0*sbn1old(i+1,j-1)
     &          -16.0*30.0*sbn1old(i,j-1)
     &          +16.0*16.0*sbn1old(i-1,j-1)
     &          -16.0*sbn1old(i-2,j-1)
     &          +sbn1old(i+2,j-2) - 16.0*sbn1old(i+1,j-2)
     &          +30.0*sbn1old(i,j-2)
     &          -16.0*sbn1old(i-1,j-2)
     &          +sbn1old(i-2,j-2))/144.0

       dby2dx2= (sbn(i+2,j+2) - 16.0*sbn(i+1,j+2)
     &          +30.0*sbn(i,j+2)
     &          -16.0*sbn(i-1,j+2)
     &          +sbn(i-2,j+2)

     &          -16.0*sbn(i+2,j+1) +
16.0*16.0*sbn(i+1,j+1)
     &          -16.0*30.0*sbn(i,j+1)
     &          +16.0*16.0*sbn(i-1,j+1)
     &          -16.0*sbn(i-2,j+1)

     &          +30.0*sbn(i+2,j) - 30.0*16.0*sbn(i+1,j)
     &          +30.0*30.0*sbn(i,j)
     &          -30.0*16.0*sbn(i-1,j)
     &          +30.0*sbn(i-2,j)

     &          -16.0*sbn(i+2,j-1) +
16.0*16.0*sbn(i+1,j-1)
     &          -16.0*30.0*sbn(i,j-1)
     &          +16.0*16.0*sbn(i-1,j-1)
     &          -16.0*sbn(i-2,j-1)

     &          +sbn(i+2,j-2) - 16.0*sbn(i+1,j-2)
     &          +30.0*sbn(i,j-2)
     &          -16.0*sbn(i-1,j-2)
     &          +sbn(i-2,j-2))/144.0

       dbx4dy2=    -sbn(i+4,j+2)/1728.0 +
sbn(i+3,j+2)/54.0
     &          -79.0*sbn(i+2,j+2)/432.0
     &          +31.0*sbn(i+1,j+2)/54.0
     &          -707.0*sbn(i,j+2)/864.0
     &          +31.0*sbn(i-1,j+2)/54.0
     &          -79.0*sbn(i-2,j+2)/432.0
     &          +sbn(i-3,j+2)/54.0
     &          -sbn(i-4,j+2)/1728.0

     &          +16.0*sbn(i+4,j+1)/1728.0
     &          -16.0*sbn(i+3,j+1)/54.0
     &          +16.0*79.0*sbn(i+2,j+1)/432.0
     &          -16.0*31.0*sbn(i+1,j+1)/54.0
     &          +16.0*707.0*sbn(i,j+1)/864.0
     &          -16.0*31.0*sbn(i-1,j+1)/54.0
     &          +16.0*79.0*sbn(i-2,j+1)/432.0
     &          -16.0*sbn(i-3,j+1)/54.0
     &          +16.0*sbn(i-4,j+1)/1728.0

     &          -30.0*sbn(i+4,j)/1728.0
     &          +30.0*sbn(i+3,j)/54.0
     &          -30.0*79.0*sbn(i+2,j)/432.0
     &          +30.0*31.0*sbn(i+1,j)/54.0
     &          -30.0*707.0*sbn(i,j)/864.0
     &          +30.0*31.0*sbn(i-1,j)/54.0
     &          -30.0*79.0*sbn(i-2,j)/432.0
     &          +30.0*sbn(i-3,j)/54.0
     &          -30.0*sbn(i-4,j)/1728.0
```

```
     &          +16.0*sbn(i+4,j-1)/1728.0
     &          -16.0*sbn(i+3,j-1)/54.0
     &          +16.0*79.0*sbn(i+2,j-1)/432.0
     &          -16.0*31.0*sbn(i+1,j-1)/54.0
     &          +16.0*707.0*sbn(i,j-1)/864.0
     &          -16.0*31.0*sbn(i-1,j-1)/54.0
     &          +16.0*79.0*sbn(i-2,j-1)/432.0
     &          -16.0*sbn(i-3,j-1)/54.0
     &          +16.0*sbn(i-4,j-1)/1728.0

     &          -sbn(i+4,j-2)/1728.0 + sbn(i+3,j-2)/54.0
     &          -79.0*sbn(i+2,j-2)/432.0
     &          +31.0*sbn(i+1,j-2)/54.0
     &          -707.0*sbn(i,j-2)/864.0
     &          +31.0*sbn(i-1,j-2)/54.0
     &          -79.0*sbn(i-2,j-2)/432.0
     &          +sbn(i-3,j-2)/54.0
     &          -sbn(i-4,j-2)/1728.0

       dby4dx2=    -sbn(i+2,j+4)/1728.0 +
sbn(i+2,j+3)/54.0
     &          -79.0*sbn(i+2,j+2)/432.0
     &          +31.0*sbn(i+2,j+1)/54.0
     &          -707.0*sbn(i+2,j)/864.0
     &          +31.0*sbn(i+2,j-1)/54.0
     &          -79.0*sbn(i+2,j-2)/432.0
     &          +sbn(i+2,j-3)/54.0
     &          -sbn(i+2,j-4)/1728.0

     &          +16.0*sbn(i+1,j+4)/1728.0
     &          -16.0*sbn(i+1,j+3)/54.0
     &          +16.0*79.0*sbn(i+1,j+2)/432.0
     &          -16.0*31.0*sbn(i+1,j+1)/54.0
     &          +16.0*707.0*sbn(i+1,j)/864.0
     &          -16.0*31.0*sbn(i+1,j-1)/54.0
     &          +16.0*79.0*sbn(i+1,j-2)/432.0
     &          -16.0*sbn(i+1,j-3)/54.0
     &          +16.0*sbn(i+1,j-4)/1728.0

     &          -30.0*sbn(i,j+4)/1728.0
     &          +30.0*sbn(i,j+3)/54.0
     &          -30.0*79.0*sbn(i,j+2)/432.0
     &          +30.0*31.0*sbn(i,j+1)/54.0
     &          -30.0*707.0*sbn(i,j)/864.0
     &          +30.0*31.0*sbn(i,j-1)/54.0
     &          -30.0*79.0*sbn(i,j-2)/432.0
     &          +30.0*sbn(i,j-3)/54.0
     &          -30.0*sbn(i,j-4)/1728.0

     &          +16.0*sbn(i-1,j+4)/1728.0
     &          -16.0*sbn(i-1,j+3)/54.0
     &          +16.0*79.0*sbn(i-1,j+2)/432.0
     &          -16.0*31.0*sbn(i-1,j+1)/54.0
     &          +16.0*707.0*sbn(i-1,j)/864.0
     &          -16.0*31.0*sbn(i-1,j-1)/54.0
     &          +16.0*79.0*sbn(i-1,j-2)/432.0
     &          -16.0*sbn(i-1,j-3)/54.0
     &          +16.0*sbn(i-1,j-4)/1728.0

     &          -sbn(i-2,j+4)/1728.0 + sbn(i-2,j+3)/54.0
     &          -79.0*sbn(i-2,j+2)/432.0
     &          +31.0*sbn(i-2,j+1)/54.0
     &          -707.0*sbn(i-2,j)/864.0
     &          +31.0*sbn(i-2,j-1)/54.0
     &          -79.0*sbn(i-2,j-2)/432.0
     &          +sbn(i-2,j-3)/54.0
     &          -sbn(i-2,j-4)/1728.0

       dbx2old=(-
sbn1old(i+2,j)+16.0*sbn1old(i+1,j)+16.0*sbn1old(i-
1,j)
     &       -30.0*sbn1old(i,j)-sbn1old(i-2,j))/12.0

       dby2old=(-
sbn1old(i,j+2)+16.0*sbn1old(i,j+1)+16.0*sbn1old(i,j-
1)
     &       -30.0*sbn1old(i,j)-sbn1old(i,j-2))/12.0

       dbx4old=sbn1old(i+4,j)/144.0 -
2.0*sbn1old(i+3,j)/9.0
     &       +79.0*sbn1old(i+2,j)/36.0 -
62.0*sbn1old(i+1,j)/9.0
     &       +707.0*sbn1old(i,j)/72.0  - 62.0*sbn1old(i-
1,j)/9.0
     &       +79.0*sbn1old(i-2,j)/36.0 - 2.0*sbn1old(i-
3,j)/9.0
     &       +sbn1old(i-4,j)/144.0

       dby4old=sbn1old(i,j+4)/144.0 -
2.0*sbn1old(i,j+3)/9.0
     &       +79.0*sbn1old(i,j+2)/36.0 -
62.0*sbn1old(i,j+1)/9.0
     &       +707.0*sbn1old(i,j)/72.0  -
62.0*sbn1old(i,j-1)/9.0
     &       +79.0*sbn1old(i,j-2)/36.0 -
2.0*sbn1old(i,j-3)/9.0
     &       +sbn1old(i,j-4)/144.0

       dbx6old=-sbn1old(i+6,j)/1728.0 +
sbn1old(i+5,j)/36.0
     &       -143.0*sbn1old(i+4,j)/288.0 +
439.0*sbn1old(i+3,j)/108.0
     &       -3031.0*sbn1old(i+2,j)/192.0 +
203.0*sbn1old(i+1,j)/6.0
     &       -6233.0*sbn1old(i,j)/144.0 +
203.0*sbn1old(i-1,j)/6.0
     &       -3031.0*sbn1old(i-2,j)/192.0 +
439.0*sbn1old(i-3,j)/108.0
     &       -143.0*sbn1old(i-4,j)/288.0 + sbn1old(i-
5,j)/36.0
     &       -sbn1old(i-6,j)/1728.0

       dby6old=-sbn1old(i,j+6)/1728.0 +
sbn1old(i,j+5)/36.0
     &       -143.0*sbn1old(i,j+4)/288.0 +
439.0*sbn1old(i,j+3)/108.0
     &       -3031.0*sbn1old(i,j+2)/192.0 +
203.0*sbn1old(i,j+1)/6.0
     &       -6233.0*sbn1old(i,j)/144.0 +
203.0*sbn1old(i,j-1)/6.0
     &       -3031.0*sbn1old(i,j-2)/192.0 +
439.0*sbn1old(i,j-3)/108.0
     &       -143.0*sbn1old(i,j-4)/288.0 + sbn1old(i,j-
5)/36.0
```

```
     &        -sbn1old(i,j-6)/1728.0

       dbx2=  (-sbn(i+2,j)+16.0*sbn(i+1,j)-30.0*sbn(i,j)
     &       +16.0*sbn(i-1,j) - sbn(i-2,j))/12.0

       dby2=  (-sbn(i,j+2)+16.0*sbn(i,j+1)-30.0*sbn(i,j)
     &       +16.0*sbn(i,j-1)-sbn(i,j-2))/12.0

       dbx4=sbn(i+4,j)/144.0 - 2.0*sbn(i+3,j)/9.0
     &       +79.0*sbn(i+2,j)/36.0 - 62.0*sbn(i+1,j)/9.0
     &       +707.0*sbn(i,j)/72.0  - 62.0*sbn(i-1,j)/9.0
     &       +79.0*sbn(i-2,j)/36.0 - 2.0*sbn(i-3,j)/9.0
     &       +sbn(i-4,j)/144.0

       dby4=sbn(i,j+4)/144.0 - 2.0*sbn(i,j+3)/9.0
     &       +79.0*sbn(i,j+2)/36.0 - 62.0*sbn(i,j+1)/9.0
     &       +707.0*sbn(i,j)/72.0  - 62.0*sbn(i,j-1)/9.0
     &       +79.0*sbn(i,j-2)/36.0 - 2.0*sbn(i,j-3)/9.0
     &       +sbn(i,j-4)/144.0

       dbx6= -sbn(i+6,j)/1728.0 + sbn(i+5,j)/36.0
     &       -143.0*sbn(i+4,j)/288.0 +
439.0*sbn(i+3,j)/108.0
     &       -3031.0*sbn(i+2,j)/192.0 +
203.0*sbn(i+1,j)/6.0
     &       -6233.0*sbn(i,j)/144.0 + 203.0*sbn(i-
1,j)/6.0
     &       -3031.0*sbn(i-2,j)/192.0 + 439.0*sbn(i-
3,j)/108.0
     &       -143.0*sbn(i-4,j)/288.0 + sbn(i-5,j)/36.0
     &       -sbn(i-6,j)/1728.0

       dby6= -sbn(i,j+6)/1728.0 + sbn(i,j+5)/36.0
     &       -143.0*sbn(i,j+4)/288.0 +
439.0*sbn(i,j+3)/108.0
     &       -3031.0*sbn(i,j+2)/192.0 +
203.0*sbn(i,j+1)/6.0
     &       -6233.0*sbn(i,j)/144.0 + 203.0*sbn(i,j-
1)/6.0
     &       -3031.0*sbn(i,j-2)/192.0 + 439.0*sbn(i,j-
3)/108.0
     &       -143.0*sbn(i,j-4)/288.0 + sbn(i,j-5)/36.0
     &       -sbn(i,j-6)/1728.0

        rbn1(i,j) = rbn(i,j) + dtx*dbx2old/2.0 +
dtx*dbx2/2.0
     &        + dtx*dby2old/2.0 + dtx*dby2/2.0
     &        + dt*cb1*sbn1old(i,j)/2.0 +
dt*cb1*sbn(i,j)/2.0
     &        + dtx3*dbx6old/24.0 + dtx3*dbx6/24.0
     &        + dtx3*dbx4dy2old/8.0 +
dtx3*dbx4dy2/8.0
     &        + dtx3*dby4dx2old/8.0 +
dtx3*dby4dx2/8.0
     &        + dtx2*dt*cb1*dby2dx2old/4.0 +
dtx2*dt*cb1*dby2dx2/4.0
     &        + dtx2*dt*cb1*dbx4old/8.0 +
dtx2*dt*cb1*dbx4/8.0
     &        + dtx3*dby6old/24.0 + dtx3*dby6/24.0
     &        + dtx2*dt*cb1*dby4old/8.0 +
dtx2*dt*cb1*dby4/8.0
     &        + dtx*dt2*cb12*dbx2old/8.0 +
dtx*dt2*cb12*dbx2/8.0
     &        + dtx*dt2*cb12*dby2old/8.0 +
dtx*dt2*cb12*dby2/8.0
     &        + dt3*cb13*sbn1old(i,j)/24.0 +
dt3*cb13*sbn(i,j)/24.0
      enddo
      enddo

     do i=6,nx-6
     do j=6,ny-6
      cb1=4.0*(rn1(i,j)*rn1(i,j)+sn1(i,j)*sn1(i,j))**mp
      cb12=cb1*cb1
      cb13=cb12*cb1

     dbx4dy2old1= -rbn1(i+4,j+2)/1728.0 +
rbn1(i+3,j+2)/54.0
     &        -79.0*rbn1(i+2,j+2)/432.0
     &        +31.0*rbn1(i+1,j+2)/54.0
     &        -707.0*rbn1(i,j+2)/864.0
     &        +31.0*rbn1(i-1,j+2)/54.0
     &        -79.0*rbn1(i-2,j+2)/432.0
     &        +rbn1(i-3,j+2)/54.0
     &        -rbn1(i-4,j+2)/1728.0

     &        +16.0*rbn1(i+4,j+1)/1728.0
     &        -16.0*rbn1(i+3,j+1)/54.0
     &        +16.0*79.0*rbn1(i+2,j+1)/432.0
     &        -16.0*31.0*rbn1(i+1,j+1)/54.0
     &        +16.0*707.0*rbn1(i,j+1)/864.0
     &        -16.0*31.0*rbn1(i-1,j+1)/54.0
     &        +16.0*79.0*rbn1(i-2,j+1)/432.0
     &        -16.0*rbn1(i-3,j+1)/54.0
     &        +16.0*rbn1(i-4,j+1)/1728.0

     &        -30.0*rbn1(i+4,j)/1728.0
     &        +30.0*rbn1(i+3,j)/54.0
     &        -30.0*79.0*rbn1(i+2,j)/432.0
     &        +30.0*31.0*rbn1(i+1,j)/54.0
     &        -30.0*707.0*rbn1(i,j)/864.0
     &        +30.0*31.0*rbn1(i-1,j)/54.0
     &        -30.0*79.0*rbn1(i-2,j)/432.0
     &        +30.0*rbn1(i-3,j)/54.0
     &        -30.0*rbn1(i-4,j)/1728.0

     &        +16.0*rbn1(i+4,j-1)/1728.0
     &        -16.0*rbn1(i+3,j-1)/54.0
     &        +16.0*79.0*rbn1(i+2,j-1)/432.0
     &        -16.0*31.0*rbn1(i+1,j-1)/54.0
     &        +16.0*707.0*rbn1(i,j-1)/864.0
     &        -16.0*31.0*rbn1(i-1,j-1)/54.0
     &        +16.0*79.0*rbn1(i-2,j-1)/432.0
     &        -16.0*rbn1(i-3,j-1)/54.0
     &        +16.0*rbn1(i-4,j-1)/1728.0

     &        -rbn1(i+4,j-2)/1728.0 + rbn1(i+3,j-
2)/54.0
     &        -79.0*rbn1(i+2,j-2)/432.0
     &        +31.0*rbn1(i+1,j-2)/54.0
     &        -707.0*rbn1(i,j-2)/864.0
     &        +31.0*rbn1(i-1,j-2)/54.0
     &        -79.0*rbn1(i-2,j-2)/432.0
```

```
     &          +rbn1(i-3,j-2)/54.0
     &          -rbn1(i-4,j-2)/1728.0

       dby4dx2old1= -rbn1(i+2,j+4)/1728.0 +
rbn1(i+2,j+3)/54.0
     &          -79.0*rbn1(i+2,j+2)/432.0
     &          +31.0*rbn1(i+2,j+1)/54.0
     &          -707.0*rbn1(i+2,j)/864.0
     &          +31.0*rbn1(i+2,j-1)/54.0
     &          -79.0*rbn1(i+2,j-2)/432.0
     &          +rbn1(i+2,j-3)/54.0
     &          -rbn1(i+2,j-4)/1728.0

     &          +16.0*rbn1(i+1,j+4)/1728.0
     &          -16.0*rbn1(i+1,j+3)/54.0
     &          +16.0*79.0*rbn1(i+1,j+2)/432.0
     &          -16.0*31.0*rbn1(i+1,j+1)/54.0
     &          +16.0*707.0*rbn1(i+1,j)/864.0
     &          -16.0*31.0*rbn1(i+1,j-1)/54.0
     &          +16.0*79.0*rbn1(i+1,j-2)/432.0
     &          -16.0*rbn1(i+1,j-3)/54.0
     &          +16.0*rbn1(i+1,j-4)/1728.0

     &          -30.0*rbn1(i,j+4)/1728.0
     &          +30.0*rbn1(i,j+3)/54.0
     &          -30.0*79.0*rbn1(i,j+2)/432.0
     &          +30.0*31.0*rbn1(i,j+1)/54.0
     &          -30.0*707.0*rbn1(i,j)/864.0
     &          +30.0*31.0*rbn1(i,j-1)/54.0
     &          -30.0*79.0*rbn1(i,j-2)/432.0
     &          +30.0*rbn1(i,j-3)/54.0
     &          -30.0*rbn1(i,j-4)/1728.0

     &          +16.0*rbn1(i-1,j+4)/1728.0
     &          -16.0*rbn1(i-1,j+3)/54.0
     &          +16.0*79.0*rbn1(i-1,j+2)/432.0
     &          -16.0*31.0*rbn1(i-1,j+1)/54.0
     &          +16.0*707.0*rbn1(i-1,j)/864.0
     &          -16.0*31.0*rbn1(i-1,j-1)/54.0
     &          +16.0*79.0*rbn1(i-1,j-2)/432.0
     &          -16.0*rbn1(i-1,j-3)/54.0
     &          +16.0*rbn1(i-1,j-4)/1728.0

     &          -rbn1(i-2,j+4)/1728.0 + rbn1(i-
2,j+3)/54.0
     &          -79.0*rbn1(i-2,j+2)/432.0
     &          +31.0*rbn1(i-2,j+1)/54.0
     &          -707.0*rbn1(i-2,j)/864.0
     &          +31.0*rbn1(i-2,j-1)/54.0
     &          -79.0*rbn1(i-2,j-2)/432.0
     &          +rbn1(i-2,j-3)/54.0
     &          -rbn1(i-2,j-4)/1728.0

       dby2dx2old1= (rbn1(i+2,j+2) -
16.0*rbn1(i+1,j+2)
     &          +30.0*rbn1(i,j+2)
     &          -16.0*rbn1(i-1,j+2)
     &          +rbn1(i-2,j+2)

     &          -16.0*rbn1(i+2,j+1) +
16.0*16.0*rbn1(i+1,j+1)
     &          -16.0*30.0*rbn1(i,j+1)
     &          +16.0*16.0*rbn1(i-1,j+1)
     &          -16.0*rbn1(i-2,j+1)

     &          +30.0*rbn1(i+2,j) -
30.0*16.0*rbn1(i+1,j)
     &          +30.0*30.0*rbn1(i,j)
     &          -30.0*16.0*rbn1(i-1,j)
     &          +30.0*rbn1(i-2,j)

     &          -16.0*rbn1(i+2,j-1) +
16.0*16.0*rbn1(i+1,j-1)
     &          -16.0*30.0*rbn1(i,j-1)
     &          +16.0*16.0*rbn1(i-1,j-1)
     &          -16.0*rbn1(i-2,j-1)

     &          +rbn1(i+2,j-2) - 16.0*rbn1(i+1,j-2)
     &          +30.0*rbn1(i,j-2)
     &          -16.0*rbn1(i-1,j-2)
     &          +rbn1(i-2,j-2))/144.0

       dby2dx21= (rbn(i+2,j+2) - 16.0*rbn(i+1,j+2)
     &          +30.0*rbn(i,j+2)
     &          -16.0*rbn(i-1,j+2)
     &          +rbn(i-2,j+2)

     &          -16.0*rbn(i+2,j+1) +
16.0*16.0*rbn(i+1,j+1)
     &          -16.0*30.0*rbn(i,j+1)
     &          +16.0*16.0*rbn(i-1,j+1)
     &          -16.0*rbn(i-2,j+1)

     &          +30.0*rbn(i+2,j) - 30.0*16.0*rbn(i+1,j)
     &          +30.0*30.0*rbn(i,j)
     &          -30.0*16.0*rbn(i-1,j)
     &          +30.0*rbn(i-2,j)

     &          -16.0*rbn(i+2,j-1) + 16.0*16.0*rbn(i+1,j-
1)
     &          -16.0*30.0*rbn(i,j-1)
     &          +16.0*16.0*rbn(i-1,j-1)
     &          -16.0*rbn(i-2,j-1)

     &          +rbn(i+2,j-2) - 16.0*rbn(i+1,j-2)
     &          +30.0*rbn(i,j-2)
     &          -16.0*rbn(i-1,j-2)
     &          +rbn(i-2,j-2))/144.0

       dbx4dy21=   -rbn(i+4,j+2)/1728.0 +
rbn(i+3,j+2)/54.0
     &          -79.0*rbn(i+2,j+2)/432.0
     &          +31.0*rbn(i+1,j+2)/54.0
     &          -707.0*rbn(i,j+2)/864.0
     &          +31.0*rbn(i-1,j+2)/54.0
     &          -79.0*rbn(i-2,j+2)/432.0
     &          +rbn(i-3,j+2)/54.0
     &          -rbn(i-4,j+2)/1728.0

     &          +16.0*rbn(i+4,j+1)/1728.0
     &          -16.0*rbn(i+3,j+1)/54.0
     &          +16.0*79.0*rbn(i+2,j+1)/432.0
     &          -16.0*31.0*rbn(i+1,j+1)/54.0
     &          +16.0*707.0*rbn(i,j+1)/864.0
```

```
     &        -16.0*31.0*rbn(i-1,j+1)/54.0
     &        +16.0*79.0*rbn(i-2,j+1)/432.0
     &        -16.0*rbn(i-3,j+1)/54.0
     &        +16.0*rbn(i-4,j+1)/1728.0

     &        -30.0*rbn(i+4,j)/1728.0
     &        +30.0*rbn(i+3,j)/54.0
     &        -30.0*79.0*rbn(i+2,j)/432.0
     &        +30.0*31.0*rbn(i+1,j)/54.0
     &        -30.0*707.0*rbn(i,j)/864.0
     &        +30.0*31.0*rbn(i-1,j)/54.0
     &        -30.0*79.0*rbn(i-2,j)/432.0
     &        +30.0*rbn(i-3,j)/54.0
     &        -30.0*rbn(i-4,j)/1728.0

     &        +16.0*rbn(i+4,j-1)/1728.0
     &        -16.0*rbn(i+3,j-1)/54.0
     &        +16.0*79.0*rbn(i+2,j-1)/432.0
     &        -16.0*31.0*rbn(i+1,j-1)/54.0
     &        +16.0*707.0*rbn(i,j-1)/864.0
     &        -16.0*31.0*rbn(i-1,j-1)/54.0
     &        +16.0*79.0*rbn(i-2,j-1)/432.0
     &        -16.0*rbn(i-3,j-1)/54.0
     &        +16.0*rbn(i-4,j-1)/1728.0

     &        -rbn(i+4,j-2)/1728.0 + rbn(i+3,j-2)/54.0
     &        -79.0*rbn(i+2,j-2)/432.0
     &        +31.0*rbn(i+1,j-2)/54.0
     &        -707.0*rbn(i,j-2)/864.0
     &        +31.0*rbn(i-1,j-2)/54.0
     &        -79.0*rbn(i-2,j-2)/432.0
     &        +rbn(i-3,j-2)/54.0
     &        -rbn(i-4,j-2)/1728.0

      dby4dx21=    -rbn(i+2,j+4)/1728.0 +
rbn(i+2,j+3)/54.0
     &        -79.0*rbn(i+2,j+2)/432.0
     &        +31.0*rbn(i+2,j+1)/54.0
     &        -707.0*rbn(i+2,j)/864.0
     &        +31.0*rbn(i+2,j-1)/54.0
     &        -79.0*rbn(i+2,j-2)/432.0
     &        +rbn(i+2,j-3)/54.0
     &        -rbn(i+2,j-4)/1728.0

     &        +16.0*rbn(i+1,j+4)/1728.0
     &        -16.0*rbn(i+1,j+3)/54.0
     &        +16.0*79.0*rbn(i+1,j+2)/432.0
     &        -16.0*31.0*rbn(i+1,j+1)/54.0
     &        +16.0*707.0*rbn(i+1,j)/864.0
     &        -16.0*31.0*rbn(i+1,j-1)/54.0
     &        +16.0*79.0*rbn(i+1,j-2)/432.0
     &        -16.0*rbn(i+1,j-3)/54.0
     &        +16.0*rbn(i+1,j-4)/1728.0

     &        -30.0*rbn(i,j+4)/1728.0
     &        +30.0*rbn(i,j+3)/54.0
     &        -30.0*79.0*rbn(i,j+2)/432.0
     &        +30.0*31.0*rbn(i,j+1)/54.0
     &        -30.0*707.0*rbn(i,j)/864.0
     &        +30.0*31.0*rbn(i,j-1)/54.0
     &        -30.0*79.0*rbn(i,j-2)/432.0
     &        +30.0*rbn(i,j-3)/54.0
     &        -30.0*rbn(i,j-4)/1728.0

     &        +16.0*rbn(i-1,j+4)/1728.0
     &        -16.0*rbn(i-1,j+3)/54.0
     &        +16.0*79.0*rbn(i-1,j+2)/432.0
     &        -16.0*31.0*rbn(i-1,j+1)/54.0
     &        +16.0*707.0*rbn(i-1,j)/864.0
     &        -16.0*31.0*rbn(i-1,j-1)/54.0
     &        +16.0*79.0*rbn(i-1,j-2)/432.0
     &        -16.0*rbn(i-1,j-3)/54.0
     &        +16.0*rbn(i-1,j-4)/1728.0

     &        -rbn(i-2,j+4)/1728.0 + rbn(i-2,j+3)/54.0
     &        -79.0*rbn(i-2,j+2)/432.0
     &        +31.0*rbn(i-2,j+1)/54.0
     &        -707.0*rbn(i-2,j)/864.0
     &        +31.0*rbn(i-2,j-1)/54.0
     &        -79.0*rbn(i-2,j-2)/432.0
     &        +rbn(i-2,j-3)/54.0
     &        -rbn(i-2,j-4)/1728.0

      dbx2old1=(-
rbn1(i+2,j)+16.0*rbn1(i+1,j)+16.0*rbn1(i-1,j)
     &        -30.0*rbn1(i,j)-rbn1(i-2,j))/12.0

      dby2old1=(-
rbn1(i,j+2)+16.0*rbn1(i,j+1)+16.0*rbn1(i,j-1)
     &        -30.0*rbn1(i,j)-rbn1(i,j-2))/12.0

      dbx4old1=rbn1(i+4,j)/144.0 - 2.0*rbn1(i+3,j)/9.0
     &        +79.0*rbn1(i+2,j)/36.0 -
62.0*rbn1(i+1,j)/9.0
     &        +707.0*rbn1(i,j)/72.0  - 62.0*rbn1(i-
1,j)/9.0
     &        +79.0*rbn1(i-2,j)/36.0 - 2.0*rbn1(i-3,j)/9.0
     &        +rbn1(i-4,j)/144.0

      dby4old1=rbn1(i,j+4)/144.0 - 2.0*rbn1(i,j+3)/9.0
     &        +79.0*rbn1(i,j+2)/36.0 -
62.0*rbn1(i,j+1)/9.0
     &        +707.0*rbn1(i,j)/72.0  - 62.0*rbn1(i,j-
1)/9.0
     &        +79.0*rbn1(i,j-2)/36.0 - 2.0*rbn1(i,j-3)/9.0
     &        +rbn1(i,j-4)/144.0

      dbx6old1=-rbn1(i+6,j)/1728.0 + rbn1(i+5,j)/36.0
     &        -143.0*rbn1(i+4,j)/288.0 +
439.0*rbn1(i+3,j)/108.0
     &        -3031.0*rbn1(i+2,j)/192.0 +
203.0*rbn1(i+1,j)/6.0
     &        -6233.0*rbn1(i,j)/144.0 + 203.0*rbn1(i-
1,j)/6.0
     &        -3031.0*rbn1(i-2,j)/192.0 + 439.0*rbn1(i-
3,j)/108.0
     &        -143.0*rbn1(i-4,j)/288.0 + rbn1(i-5,j)/36.0
     &        -rbn1(i-6,j)/1728.0

      dby6old1=-rbn1(i,j+6)/1728.0 + rbn1(i,j+5)/36.0
     &        -143.0*rbn1(i,j+4)/288.0 +
439.0*rbn1(i,j+3)/108.0
     &        -3031.0*rbn1(i,j+2)/192.0 +
203.0*rbn1(i,j+1)/6.0
```

```
     &        -6233.0*rbn1(i,j)/144.0 + 203.0*rbn1(i,j-
1)/6.0
     &        -3031.0*rbn1(i,j-2)/192.0 + 439.0*rbn1(i,j-
3)/108.0
     &        -143.0*rbn1(i,j-4)/288.0 + rbn1(i,j-5)/36.0
     &        -rbn1(i,j-6)/1728.0

       dbx21= (-rbn(i+2,j)+16.0*rbn(i+1,j)-
30.0*rbn(i,j)
     &        +16.0*rbn(i-1,j) - rbn(i-2,j))/12.0

       dby21= (-rbn(i,j+2)+16.0*rbn(i,j+1)-
30.0*rbn(i,j)
     &        +16.0*rbn(i,j-1) - rbn(i,j-2))/12.0

       dbx41=rbn(i+4,j)/144.0 - 2.0*rbn(i+3,j)/9.0
     &        +79.0*rbn(i+2,j)/36.0 - 62.0*rbn(i+1,j)/9.0
     &        +707.0*rbn(i,j)/72.0  - 62.0*rbn(i-1,j)/9.0
     &        +79.0*rbn(i-2,j)/36.0 - 2.0*rbn(i-3,j)/9.0
     &        +rbn(i-4,j)/144.0

       dby41=rbn(i,j+4)/144.0 - 2.0*rbn(i,j+3)/9.0
     &        +79.0*rbn(i,j+2)/36.0 - 62.0*rbn(i,j+1)/9.0
     &        +707.0*rbn(i,j)/72.0  - 62.0*rbn(i,j-1)/9.0
     &        +79.0*rbn(i,j-2)/36.0 - 2.0*rbn(i,j-3)/9.0
     &        +rbn(i,j-4)/144.0

       dbx61= -rbn(i+6,j)/1728.0 + rbn(i+5,j)/36.0
     &        -143.0*rbn(i+4,j)/288.0 +
439.0*rbn(i+3,j)/108.0
     &        -3031.0*rbn(i+2,j)/192.0 +
203.0*rbn(i+1,j)/6.0
     &        -6233.0*rbn(i,j)/144.0 + 203.0*rbn(i-
1,j)/6.0
     &        -3031.0*rbn(i-2,j)/192.0 + 439.0*rbn(i-
3,j)/108.0
     &        -143.0*rbn(i-4,j)/288.0 + rbn(i-5,j)/36.0
     &        -rbn(i-6,j)/1728.0

       dby61= -rbn(i,j+6)/1728.0 + rbn(i,j+5)/36.0
     &        -143.0*rbn(i,j+4)/288.0 +
439.0*rbn(i,j+3)/108.0
     &        -3031.0*rbn(i,j+2)/192.0 +
203.0*rbn(i,j+1)/6.0
     &        -6233.0*rbn(i,j)/144.0 + 203.0*rbn(i,j-
1)/6.0
     &        -3031.0*rbn(i,j-2)/192.0 + 439.0*rbn(i,j-
3)/108.0
     &        -143.0*rbn(i,j-4)/288.0 + rbn(i,j-5)/36.0
     &        -rbn(i,j-6)/1728.0

        sbn1(i,j) = sbn(i,j) - dtx*dbx2old1/2.0 -
dtx*dbx21/2.0
     &         - dtx*dby2old1/2.0 - dtx*dby21/2.0
     &         - dt*cb1*rbn1(i,j)/2.0 -
dt*cb1*rbn(i,j)/2.0
     &         - dtx3*dbx6old1/24.0 - dtx3*dbx61/24.0
     &         - dtx3*dbx4dy2old1/8.0 -
dtx3*dbx4dy21/8.0
     &         - dtx3*dby4dx2old1/8.0 -
dtx3*dby4dx21/8.0
     &         - dtx2*dt*cb1*dby2dx2old1/4.0
     &         - dtx2*dt*cb1*dby2dx21/4.0
     &         - dtx2*dt*cb1*dbx4old1/8.0 -
dtx2*dt*cb1*dbx41/8.0
     &         - dtx3*dby6old1/24.0 - dtx3*dby61/24.0
     &         - dtx2*dt*cb1*dby4old1/8.0 -
dtx2*dt*cb1*dby41/8.0
     &         - dtx*dt2*cb12*dbx2old1/8.0 -
dtx*dt2*cb12*dbx21/8.0
     &         - dtx*dt2*cb12*dby2old1/8.0 -
dtx*dt2*cb12*dby21/8.0
     &         - dt3*cb13*rbn1(i,j)/24.0 -
dt3*cb13*rbn(i,j)/24.0
      enddo
      enddo

      tol=1.0D-15
      err_max=0.0
      do i=6,nx-6
      do j=6,ny-6
      err=abs(sbn1(i,j)-sbn1old(i,j))
      if (err.gt.err_max) then
      err_max = err
      endif
      enddo
      enddo

      if (err_max.le.tol) goto 6
      do i=6,nx-6
      do j=6,ny-6
      sbn1old(i,j)=sbn1(i,j)
      enddo
      enddo
      goto 5

c calculate the exact solution
 6    do i=0,nx
      do j=0,ny
      rexac(i,j)= 2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
     &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
     &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))

      sexac(i,j)=2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
     &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
     &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))

      rbexac(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
     &        +6.0D0*(it+0.5D0)*dt)
     &        /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
     &        +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))

      sbexac(i,j)=2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
     &        +6.0D0*(it+0.5D0)*dt)
     &        /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
     &        +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))
```

```
      enddo
      enddo

c calculate the error
      temp=0.0
      do i=6,nx-6
      do j=6,ny-6
       temp1=(sqrt(rn1(i,j)*rn1(i,j)+sn1(i,j)*sn1(i,j))
     &       -
sqrt(rexac(i,j)*rexac(i,j)+sexac(i,j)*sexac(i,j)))
      temp=temp+temp1*temp1
      enddo
      enddo
      errnew=sqrt(temp/((nx-10)*(ny-10)))
      print *, it, errnew

c continue time step
      it=it+1
      if(it.eq.nt) goto 7
      do i=0,nx
      do j=0,ny
      rn(i,j)=rn1(i,j)
      sn(i,j)=sn1(i,j)
      rbn(i,j)=rbn1(i,j)
      sbn(i,j)=sbn1(i,j)
      enddo
      enddo
      goto 1

 7    open(unit=03, file='implicit_2D.dat')
      do i=0,nx
      do j=0,ny

write(03,*)i,j,sqrt(rn1(i,j)*rn1(i,j)+sn1(i,j)*sn1(i,j))
      enddo
      enddo
      close (03)

C     open(unit=03, file='trial_s_400_8.dat')
C     do i=0,nx
C     write(03,*)i*dx,sn1(i),sexac(i)
C     enddo
C     close (03)
      End
```

```
c
c This is the 1D explicit GFDTD for the NLSE/dark
soliton
c
      implicit double precision (a-h,o-z)
      dimension rexac(0:10000),rbexac(0:10000),
     &
sexac(0:10000),sbexac(0:10000),x(0:10000),
     &          rn1(0:10000),rbn(0:10000),rn(0:10000),
     &          sn(0:10000),sn1(0:10000),sbn(0:10000),
     &          sbn1(0:10000),rbn1(0:10000)

c input data
c      rn: realvalue at time n
c      rn1: realvalue at time n+1
c      sn: imaginary_value at time n
c      sn1: imaginary_value at time n+1
c      rbn: real_value at time n-1/2
c      rbn1: real_value at time n+1/2
c      sbn: imaginary_value at time n-1/2
c      sbn1: imaginary_value at time n+1/2

c Here we define the "grid size"
      np = 3
      mp=(np-1)/2
      dt = 0.001D0
      dx = 0.05D0
      nx=800
      dtx = -dt/(2.0*dx*dx)
      dtx2 = dtx*dtx
      dtx3 = dtx2*dtx

      nt=4000
      x(0)=-20.0D0

      do i=1,nx
      x(i)=x(0)+ i*dx
      enddo

c Initial conditions
c Dark Soliton
      do i=0,nx
        rn(i)=(exp(2.0D0*x(i)-10.0D0)-exp(10.0D0-
2.0D0*x(i)))
     &       /(exp(2.0D0*x(i)-10.0D0)+exp(-
2.0D0*x(i)+10.0D0))

        sn(i)=1.0D0

        rbn(i)=cos(4.0D0*dt)*(exp(2.0D0*x(i)-
2.0D0*dt-10.0D0)
     &       -exp(10.0D0-2.0D0*x(i)+2.0D0*dt))
     &       /(exp(2.0D0*x(i)-2.0D0*dt-10.0D0)
     &       +exp(-2.0D0*x(i)+2.0D0*dt+10.0D0))
     &       +sin(4.0D0*dt)

        sbn(i)=-sin(4.0D0*dt)*(exp(2.0D0*x(i)-
2.0D0*dt-10.0D0)
     &       -exp(10.0D0-2.0D0*x(i)+2.0D0*dt))
     &       /(exp(2.0D0*x(i)-2.0D0*dt-10.0D0)
     &       +exp(-2.0D0*x(i)+2.0D0*dt+10.0D0))
     &       +cos(4.0D0*dt)
      enddo
c*****************************
c Starting time step
      it=1

c Boundary conditions for rn1(i)
c Dark Soliton
 1    do i=0,5
      rn1(i)=cos(8.0D0*it*dt)*(exp(2.0D0*x(i)-
4.0D0*it*dt-10.0D0)
     &       -exp(10.0D0-2.0D0*x(i)+4.0D0*it*dt))
     &       /(exp(2.0D0*x(i)-4.0D0*it*dt-10.0D0)
     &       +exp(-2.0D0*x(i)+4.0D0*it*dt+10.0D0))
     &       +sin(8.0D0*it*dt)
      enddo
```

```
      do i=nx-5,nx
       rn1(i)=cos(8.0D0*it*dt)*(exp(2.0D0*x(i)-
4.0D0*it*dt-10.0D0)
     &        -exp(10.0D0-2.0D0*x(i)+4.0D0*it*dt))
     &        /(exp(2.0D0*x(i)-4.0D0*it*dt-10.0D0)
     &        +exp(-2.0D0*x(i)+4.0D0*it*dt+10.0D0))
     &        +sin(8.0D0*it*dt)
      enddo
c*****************************

c Set up the system for rn1(i)
      do k=6,nx-6
       v1=4.0*(rbn(k)*rbn(k)+sbn(k)*sbn(k))**mp
       v12=v1*v1
       v13=v12*v1

            c0=30.0/12.0
            c1=16.0/12.0
            c2=1.0/12.0
            d1=-c2*sbn(k+2)+c1*sbn(k+1)-
c0*sbn(k)+c1*sbn(k-1)-c2*sbn(k-2)

            c0=1414.0/144.0
            c1=992.0/144.0
            c2=316.0/144.0
            c3=32.0/144.0
            c4=1.0/144.0
            d2=c4*sbn(k+4)-c3*sbn(k+3)+c2*sbn(k+2)-
c1*sbn(k+1)+c0*sbn(k)
     &    -c1*sbn(k-1)+c2*sbn(k-2)-c3*sbn(k-
3)+c4*sbn(k-4)

            c0=74796.0/1728.0
            c1=58464.0/1728.0
            c2=27279.0/1728.0
            c3=7024.0/1728.0
            c4=858.0/1728.0
            c5=48.0/1728.0
            c6=1.0/1728.0
            d3=-c6*sbn(k+6)+c5*sbn(k+5)-
c4*sbn(k+4)+c3*sbn(k+3)-c2*sbn(k+2)
     &     +c1*sbn(k+1)-c0*sbn(k)+c1*sbn(k-1)-
c2*sbn(k-2)+c3*sbn(k-3)
     &     -c4*sbn(k-4)+c5*sbn(k-5)-c6*sbn(k-6)

       rn1(k)=rn(k)+ dt*v1*sbn(k)+ d1*dtx
     &        -1.0/24.0*dtx3*d3
     &        -1.0/8.0*dtx2*d2*v1*dt
     &        -1.0/8.0*dtx*d1*v12*dt*dt
     &        -1.0/24.0*dt*dt*dt*v13*sbn(k)
            enddo

c Set up the system for sn1(i)
c Dark Soliton
      do i=0,5
      sn1(i)= -sin(8.0D0*it*dt)*(exp(2.0D0*x(i)-
4.0D0*it*dt-10.0D0)
     &        -exp(10.0D0-2.0D0*x(i)+4.0D0*it*dt))
     &        /(exp(2.0D0*x(i)-4.0D0*it*dt-10.0D0)
     &        +exp(-2.0D0*x(i)+4.0D0*it*dt+10.0D0))
     &        +cos(8.0D0*it*dt)
      enddo
      do i=nx-5,nx
      sn1(i)= -sin(8.0D0*it*dt)*(exp(2.0D0*x(i)-
4.0D0*it*dt-10.0D0)
     &        -exp(10.0D0-2.0D0*x(i)+4.0D0*it*dt))
     &        /(exp(2.0D0*x(i)-4.0D0*it*dt-10.0D0)
     &        +exp(-2.0D0*x(i)+4.0D0*it*dt+10.0D0))
     &        +cos(8.0D0*it*dt)
      enddo
c*************************
       do k=6,nx-6
       v1=4.0*(rbn(k)*rbn(k)+sbn(k)*sbn(k))**mp
       v12=v1*v1
       v13=v12*v1

            c0=30.0/12.0
            c1=16.0/12.0
            c2=1.0/12.0
            d1=-c2*rbn(k+2)+c1*rbn(k+1)-
c0*rbn(k)+c1*rbn(k-1)-c2*rbn(k-2)

            c0=1414.0/144.0
            c1=992.0/144.0
            c2=316.0/144.0
            c3=32.0/144.0
            c4=1.0/144.0
            d2=c4*rbn(k+4)-c3*rbn(k+3)+c2*rbn(k+2)-
c1*rbn(k+1)+c0*rbn(k)
     &    -c1*rbn(k-1)+c2*rbn(k-2)-c3*rbn(k-
3)+c4*rbn(k-4)

            c0=74796.0/1728.0
            c1=58464.0/1728.0
            c2=27279.0/1728.0
            c3=7024.0/1728.0
            c4=858.0/1728.0
            c5=48.0/1728.0
            c6=1.0/1728.0
            d3=-c6*rbn(k+6)+c5*rbn(k+5)-
c4*rbn(k+4)+c3*rbn(k+3)-c2*rbn(k+2)
     &     +c1*rbn(k+1)-c0*rbn(k)+c1*rbn(k-1)-
c2*rbn(k-2)+c3*rbn(k-3)
     &     -c4*rbn(k-4)+c5*rbn(k-5)-c6*rbn(k-6)

       sn1(k)=sn(k)- dt*v1*rbn(k)- d1*dtx
     &        +1.0/24.0*dtx3*d3
     &        +1.0/8.0*dtx2*d2*v1*dt
     &        +1.0/8.0*dtx*d1*v12*dt*dt
     &        +1.0/24.0*dt*dt*dt*v13*rbn(k)
            enddo

c setp up the system for rbn1(i)
c Dark soliton
      do i=0,5
       rbn1(i)=
cos(8.0D0*(it+0.5D0)*dt)*(exp(2.0D0*x(i)
     &        -4.0D0*(it+0.5D0)*dt-10.0D0)
     &        -exp(10.0D0-
2.0D0*x(i)+4.0D0*(it+0.5D0)*dt))
     &        /(exp(2.0D0*x(i)-4.0D0*(it+0.5D0)*dt-
10.0D0)
     &        +exp(-
2.0D0*x(i)+4.0D0*(it+0.5D0)*dt+10.0D0))
```

```
     &        +sin(8.0D0*(it+0.5D0)*dt)
      enddo

      do i=nx-5,nx

rbn1(i)=cos(8.0D0*(it+0.5D0)*dt)*(exp(2.0D0*x(i)
     &        -4.0D0*(it+0.5D0)*dt-10.0D0)
     &        -exp(10.0D0-
2.0D0*x(i)+4.0D0*(it+0.5D0)*dt))
     &        /(exp(2.0D0*x(i)-4.0D0*(it+0.5D0)*dt-
10.0D0)
     &        +exp(-
2.0D0*x(i)+4.0D0*(it+0.5D0)*dt+10.0D0))
     &        +sin(8.0D0*(it+0.5D0)*dt)
      enddo
c*****************************
      do k=6,nx-6
      v1=4.0*(rn1(k)*rn1(k)+sn1(k)*sn1(k))**mp
      v12=v1*v1
      v13=v12*v1

            c0=30.0/12.0
            c1=16.0/12.0
            c2=1.0/12.0
            d1=-c2*sn1(k+2)+c1*sn1(k+1)-
c0*sn1(k)+c1*sn1(k-1)-c2*sn1(k-2)

            c0=1414.0/144.0
            c1=992.0/144.0
            c2=316.0/144.0
            c3=32.0/144.0
            c4=1.0/144.0
            d2=c4*sn1(k+4)-c3*sn1(k+3)+c2*sn1(k+2)-
c1*sn1(k+1)+c0*sn1(k)
     &    -c1*sn1(k-1)+c2*sn1(k-2)-c3*sn1(k-
3)+c4*sn1(k-4)

            c0=74796.0/1728.0
            c1=58464.0/1728.0
            c2=27279.0/1728.0
            c3=7024.0/1728.0
            c4=858.0/1728.0
            c5=48.0/1728.0
            c6=1.0/1728.0
            d3=-c6*sn1(k+6)+c5*sn1(k+5)-
c4*sn1(k+4)+c3*sn1(k+3)-c2*sn1(k+2)
     &    +c1*sn1(k+1)-c0*sn1(k)+c1*sn1(k-1)-
c2*sn1(k-2)+c3*sn1(k-3)
     &    -c4*sn1(k-4)+c5*sn1(k-5)-c6*sn1(k-6)

       rbn1(k)=rbn(k)+ dt*v1*sn1(k)+ d1*dtx
     &       -1.0/24.0*dtx3*d3
     &       -1.0/8.0*dtx2*d2*v1*dt
     &       -1.0/8.0*dtx*d1*v12*dt*dt
     &       -1.0/24.0*dt*dt*dt*v13*sn1(k)
            enddo

c solve sbn1(i)
c  Dark Soliton
      do i=0,5
      sbn1(i)= -
sin(8.0D0*(it+0.5D0)*dt)*(exp(2.0D0*x(i)
     &        -4.0D0*(it+0.5D0)*dt-10.0D0)
     &        -exp(10.0D0-
2.0D0*x(i)+4.0D0*(it+0.5D0)*dt))
     &        /(exp(2.0D0*x(i)-4.0D0*(it+0.5D0)*dt-
10.0D0)
     &        +exp(-
2.0D0*x(i)+4.0D0*(it+0.5D0)*dt+10.0D0))
     &        +cos(8.0D0*(it+0.5D0)*dt)
      enddo
      do i=nx-5,nx
       sbn1(i)= -
sin(8.0D0*(it+0.5D0)*dt)*(exp(2.0D0*x(i)
     &        -4.0D0*(it+0.5D0)*dt-10.0D0)
     &        -exp(10.0D0-
2.0D0*x(i)+4.0D0*(it+0.5D0)*dt))
     &        /(exp(2.0D0*x(i)-4.0D0*(it+0.5D0)*dt-
10.0D0)
     &        +exp(-
2.0D0*x(i)+4.0D0*(it+0.5D0)*dt+10.0D0))
     &        +cos(8.0D0*(it+0.5D0)*dt)
      enddo
c*************************
      do k=6,nx-6
      v1=4.0*(rn1(k)*rn1(k)+sn1(k)*sn1(k))**mp
      v12=v1*v1
      v13=v12*v1

            c0=30.0/12.0
            c1=16.0/12.0
            c2=1.0/12.0
            d1=-c2*rn1(k+2)+c1*rn1(k+1)-
c0*rn1(k)+c1*rn1(k-1)-c2*rn1(k-2)

            c0=1414.0/144.0
            c1=992.0/144.0
            c2=316.0/144.0
            c3=32.0/144.0
            c4=1.0/144.0
            d2=c4*rn1(k+4)-c3*rn1(k+3)+c2*rn1(k+2)-
c1*rn1(k+1)+c0*rn1(k)
     &    -c1*rn1(k-1)+c2*rn1(k-2)-c3*rn1(k-
3)+c4*rn1(k-4)

            c0=74796.0/1728.0
            c1=58464.0/1728.0
            c2=27279.0/1728.0
            c3=7024.0/1728.0
            c4=858.0/1728.0
            c5=48.0/1728.0
            c6=1.0/1728.0
            d3=-c6*rn1(k+6)+c5*rn1(k+5)-
c4*rn1(k+4)+c3*rn1(k+3)-c2*rn1(k+2)
     &    +c1*rn1(k+1)-c0*rn1(k)+c1*rn1(k-1)-
c2*rn1(k-2)+c3*rn1(k-3)
     &    -c4*rn1(k-4)+c5*rn1(k-5)-c6*rn1(k-6)

       sbn1(k)=sbn(k)- dt*v1*rn1(k)- d1*dtx
     &       +1.0/24.0*dtx3*d3
     &       +1.0/8.0*dtx2*d2*v1*dt
     &       +1.0/8.0*dtx*d1*v12*dt*dt
     &       +1.0/24.0*dt*dt*dt*v13*rn1(k)
            enddo
```

```
c calculate the exact solution
c Dark soliton
      do i=0,nx
      rexac(i)= cos(8.0D0*it*dt)*(exp(2.0D0*x(i)-
4.0D0*it*dt-10.0D0)
     &        -exp(10.0D0-2.0D0*x(i)+4.0D0*it*dt))
     &        /(exp(2.0D0*x(i)-4.0D0*it*dt-10.0D0)
     &        +exp(-2.0D0*x(i)+4.0D0*it*dt+10.0D0))
     &        +sin(8.0D0*it*dt)

      sexac(i)= -sin(8.0D0*it*dt)*(exp(2.0D0*x(i)-
4.0D0*it*dt-10.0D0)
     &        -exp(10.0D0-2.0D0*x(i)+4.0D0*it*dt))
     &        /(exp(2.0D0*x(i)-4.0D0*it*dt-10.0D0)
     &        +exp(-2.0D0*x(i)+4.0D0*it*dt+10.0D0))
     &        +cos(8.0D0*it*dt)

      rbexac(i)=
cos(8.0D0*(it+0.5D0)*dt)*(exp(2.0D0*x(i)
     &        -4.0D0*(it+0.5D0)*dt-10.0D0)
     &        -exp(10.0D0-
2.0D0*x(i)+4.0D0*(it+0.5D0)*dt))
     &        /(exp(2.0D0*x(i)-4.0D0*(it+0.5D0)*dt-
10.0D0)
     &        +exp(-
2.0D0*x(i)+4.0D0*(it+0.5D0)*dt+10.0D0))
     &        +sin(8.0D0*(it+0.5D0)*dt)

      sbexac(i)=-
sin(8.0D0*(it+0.5D0)*dt)*(exp(2.0D0*x(i)
     &       -4.0D0*(it+0.5D0)*dt-10.0D0)
     &        -exp(10.0D0-
2.0D0*x(i)+4.0D0*(it+0.5D0)*dt))
     &        /(exp(2.0D0*x(i)-4.0D0*(it+0.5D0)*dt-
10.0D0)
     &        +exp(-
2.0D0*x(i)+4.0D0*(it+0.5D0)*dt+10.0D0))
     &        +cos(8.0D0*(it+0.5D0)*dt)
      enddo
c*****************************
c calculate the error
      temp=0.0
      do i=0,nx
      temp=temp+(sqrt(rn1(i)*rn1(i)+sn1(i)*sn1(i))
     &        -sqrt(rexac(i)*rexac(i)+sexac(i)*sexac(i)))
      enddo
       errnew=abs(temp*dx)
      print *, it, errnew

c continue time step
      it=it+1
      if(it.eq.nt) goto 7
      do i=0,nx
      rn(i)=rn1(i)
      sn(i)=sn1(i)
      rbn(i)=rbn1(i)
      sbn(i)=sbn1(i)
      enddo
      goto 1

 7    open(unit=03, file='trial_r_E_400_10.dat')
      do i=0,nx
      write(03,*)i*dx,sqrt(rn1(i)*rn1(i)+sn1(i)*sn1(i)),
     &     sqrt(rexac(i)*rexac(i)+sexac(i)*sexac(i))
      enddo
      close (03)

C     open(unit=03, file='trial_s_E.dat')
C     do i=0,nx
C     write(03,*)i*dx,sn1(i),sexac(i)
C     enddo
C     close (03)
      end
```

```
c
c This is the 2D explicit GFDTD for the NLSE
c
      implicit double precision (a-h,o-z)
      dimension
rexac(0:1000,0:1000),rbexac(0:1000,0:1000),
     &
sexac(0:1000,0:1000),sbexac(0:1000,0:1000),
     &        x(0:1000),y(0:1000),
     &        rn1(0:1000,0:1000),rbn(0:1000,0:1000),
     &        rn(0:1000,0:1000),
     &        sn(0:1000,0:1000),sn1(0:1000,0:1000),
     &        sbn(0:1000,0:1000),
     &
sbn1(0:1000,0:1000),rbn1(0:1000,0:1000),
     &
sn1old(0:1000,0:1000),sbn1old(0:1000,0:1000)

c input data
c     rn: realvalue at time n
c     rn1: realvalue at time n+1
c     sn: imaginary_value at time n
c     sn1: imaginary_value at time n+1
c     rbn: real_value at time n-1/2
c     rbn1: real_value at time n+1/2
c     sbn: imaginary_value at time n-1/2
c     sbn1: imaginary_value at time n+1/2

c Here we define the "grid size"
      np = 3
      mp=(np-1)/2
      dt = 0.001D0
      dx = 0.05D0
      nx=800
      dy = 0.05D0
      ny=800
      dtx = dt/(dx*dx)
      dtx2 = dtx*dtx
      dtx3 = dtx2*dtx
      dt2  = dt*dt
      dt3  = dt2*dt

      nt=5000
      x(0)=-20.0D0
      y(0)=-20.0D0
```

```
      do i=1,nx
       x(i)=x(0)+ i*dx
      enddo

      do j=1,ny
       y(j)=y(0)+ j*dy
      enddo

c Initial conditions
c Bright Soliton
      do i=0,nx
      do j=0,ny
        rn(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j))
     &        /(exp(x(i)+y(j)+10.0D0)+exp(-10.0D0-
x(i)-y(j)))
C    &         +2.0D0*cos(2.0D0*x(i)-20.0D0)
C    &         /(exp(x(i)-10.0D0)+exp(-x(i)+10.0D0))

        sn(i,j)=2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j))
     &        /(exp(x(i)+y(j)+10.0D0)+exp(-10.0D0-
x(i)-y(j)))
C    &         +2.0D0*sin(2.0D0*x(i)-20.0D0)
C    &         /(exp(x(i)-10.0D0)+exp(-x(i)+10.0D0))

        rbn(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+3.0D0*dt)
     &        /(exp(x(i)+y(j)-4.0D0*dt+10.0D0)
     &        +exp(4.0D0*dt-x(i)-y(j)-10.0D0))
C    &         +2.0D0*cos(2.0D0*x(i)+1.5D0*dt-
20.0D0)
C    &         /(exp(x(i)+2.0D0*dt-10.0D0)+exp(-
2.0D0*dt-x(i)+10.0D0))

        sbn(i,j)=2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+3.0D0*dt)
     &        /(exp(x(i)+y(j)-4.0D0*dt+10.0D0)
     &        +exp(4.0D0*dt-x(i)-y(j)-10.0D0))
C    &         +2.0D0*sin(2.0D0*x(i)+1.5D0*dt-
20.0D0)
C    &         /(exp(x(i)+2.0D0*dt-10.0D0)+exp(-
2.0D0*dt-x(i)+10.0D0))
      enddo
      enddo
c*****************************
c Starting time step
      it=1

c Boundary conditions for rn1(i)
c Bright Soliton

  1   do i=0,nx
      do j=0,6
       rn1(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
     &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
     &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))
C    &         +2.0D0*cos(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &         /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &         +exp(-4.0D0*it*dt-x(i)+10.0D0))
      enddo
      enddo

      do i=0,nx
      do j=ny-6,ny
       rn1(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
     &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
     &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))
C    &         +2.0D0*cos(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &         /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &         +exp(-4.0D0*it*dt-x(i)+10.0D0))
      enddo
      enddo

      do i=0,6
      do j=7,ny-7
       rn1(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
     &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
     &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))
C    &         +2.0D0*cos(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &         /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &         +exp(-4.0D0*it*dt-x(i)+10.0D0))
      enddo
      enddo

      do i=nx-6,nx
      do j=7,ny-7
       rn1(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
     &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
     &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))
C    &         +2.0D0*cos(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &         /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &         +exp(-4.0D0*it*dt-x(i)+10.0D0))
      enddo
      enddo

c*****************************

c Set up the system for rn1(i)
      do i=7,nx-7
      do j=7,ny-7
       c1=4.0*(rbn(i,j)*rbn(i,j)+sbn(i,j)*sbn(i,j))**mp
       c12=c1*c1
       c13=c12*c1

       dbx2=  (-sbn(i+2,j)+16.0*sbn(i+1,j)-30.0*sbn(i,j)
     &       +16.0*sbn(i-1,j) - sbn(i-2,j))/12.0

       dby2=  (-sbn(i,j+2)+16.0*sbn(i,j+1)-30.0*sbn(i,j)
     &       +16.0*sbn(i,j-1)-sbn(i,j-2))/12.0

       dbx4=sbn(i+4,j)/144.0 - 2.0*sbn(i+3,j)/9.0
     &       +79.0*sbn(i+2,j)/36.0 - 62.0*sbn(i+1,j)/9.0
     &       +707.0*sbn(i,j)/72.0  - 62.0*sbn(i-1,j)/9.0
     &       +79.0*sbn(i-2,j)/36.0 - 2.0*sbn(i-3,j)/9.0
     &       +sbn(i-4,j)/144.0
```

```
      dby4=sbn(i,j+4)/144.0 - 2.0*sbn(i,j+3)/9.0
     &        +79.0*sbn(i,j+2)/36.0 - 62.0*sbn(i,j+1)/9.0
     &        +707.0*sbn(i,j)/72.0  - 62.0*sbn(i,j-1)/9.0
     &        +79.0*sbn(i,j-2)/36.0 - 2.0*sbn(i,j-3)/9.0
     &        +sbn(i,j-4)/144.0

      dbx6= -sbn(i+6,j)/1728.0 + sbn(i+5,j)/36.0
     &        -143.0*sbn(i+4,j)/288.0 +
439.0*sbn(i+3,j)/108.0
     &        -3031.0*sbn(i+2,j)/192.0 +
203.0*sbn(i+1,j)/6.0
     &        -6233.0*sbn(i,j)/144.0 + 203.0*sbn(i-
1,j)/6.0
     &        -3031.0*sbn(i-2,j)/192.0 + 439.0*sbn(i-
3,j)/108.0
     &        -143.0*sbn(i-4,j)/288.0 + sbn(i-5,j)/36.0
     &        -sbn(i-6,j)/1728.0

      dby6= -sbn(i,j+6)/1728.0 + sbn(i,j+5)/36.0
     &        -143.0*sbn(i,j+4)/288.0 +
439.0*sbn(i,j+3)/108.0
     &        -3031.0*sbn(i,j+2)/192.0 +
203.0*sbn(i,j+1)/6.0
     &        -6233.0*sbn(i,j)/144.0 + 203.0*sbn(i,j-
1)/6.0
     &        -3031.0*sbn(i,j-2)/192.0 + 439.0*sbn(i,j-
3)/108.0
     &        -143.0*sbn(i,j-4)/288.0 + sbn(i,j-5)/36.0
     &        -sbn(i,j-6)/1728.0

      dby2dx2= (sbn(i+2,j+2) - 16.0*sbn(i+1,j+2)
     &          +30.0*sbn(i,j+2)
     &          -16.0*sbn(i-1,j+2)
     &          +sbn(i-2,j+2)

     &          -16.0*sbn(i+2,j+1) +
16.0*16.0*sbn(i+1,j+1)
     &          -16.0*30.0*sbn(i,j+1)
     &          +16.0*16.0*sbn(i-1,j+1)
     &          -16.0*sbn(i-2,j+1)

     &          +30.0*sbn(i+2,j) - 30.0*16.0*sbn(i+1,j)
     &          +30.0*30.0*sbn(i,j)
     &          -30.0*16.0*sbn(i-1,j)
     &          +30.0*sbn(i-2,j)

     &          -16.0*sbn(i+2,j-1) +
16.0*16.0*sbn(i+1,j-1)
     &          -16.0*30.0*sbn(i,j-1)
     &          +16.0*16.0*sbn(i-1,j-1)
     &          -16.0*sbn(i-2,j-1)

     &          +sbn(i+2,j-2) - 16.0*sbn(i+1,j-2)
     &          +30.0*sbn(i,j-2)
     &          -16.0*sbn(i-1,j-2)
     &          +sbn(i-2,j-2))/144.0

      dbx4dy2=    -sbn(i+4,j+2)/1728.0 +
sbn(i+3,j+2)/54.0
     &          -79.0*sbn(i+2,j+2)/432.0
     &          +31.0*sbn(i+1,j+2)/54.0
     &          -707.0*sbn(i,j+2)/864.0
     &          +31.0*sbn(i-1,j+2)/54.0
     &          -79.0*sbn(i-2,j+2)/432.0
     &          +sbn(i-3,j+2)/54.0
     &          -sbn(i-4,j+2)/1728.0

     &          +16.0*sbn(i+4,j+1)/1728.0
     &          -16.0*sbn(i+3,j+1)/54.0
     &          +16.0*79.0*sbn(i+2,j+1)/432.0
     &          -16.0*31.0*sbn(i+1,j+1)/54.0
     &          +16.0*707.0*sbn(i,j+1)/864.0
     &          -16.0*31.0*sbn(i-1,j+1)/54.0
     &          +16.0*79.0*sbn(i-2,j+1)/432.0
     &          -16.0*sbn(i-3,j+1)/54.0
     &          +16.0*sbn(i-4,j+1)/1728.0

     &          -30.0*sbn(i+4,j)/1728.0
     &          +30.0*sbn(i+3,j)/54.0
     &          -30.0*79.0*sbn(i+2,j)/432.0
     &          +30.0*31.0*sbn(i+1,j)/54.0
     &          -30.0*707.0*sbn(i,j)/864.0
     &          +30.0*31.0*sbn(i-1,j)/54.0
     &          -30.0*79.0*sbn(i-2,j)/432.0
     &          +30.0*sbn(i-3,j)/54.0
     &          -30.0*sbn(i-4,j)/1728.0

     &          +16.0*sbn(i+4,j-1)/1728.0
     &          -16.0*sbn(i+3,j-1)/54.0
     &          +16.0*79.0*sbn(i+2,j-1)/432.0
     &          -16.0*31.0*sbn(i+1,j-1)/54.0
     &          +16.0*707.0*sbn(i,j-1)/864.0
     &          -16.0*31.0*sbn(i-1,j-1)/54.0
     &          +16.0*79.0*sbn(i-2,j-1)/432.0
     &          -16.0*sbn(i-3,j-1)/54.0
     &          +16.0*sbn(i-4,j-1)/1728.0

     &          -sbn(i+4,j-2)/1728.0 + sbn(i+3,j-2)/54.0
     &          -79.0*sbn(i+2,j-2)/432.0
     &          +31.0*sbn(i+1,j-2)/54.0
     &          -707.0*sbn(i,j-2)/864.0
     &          +31.0*sbn(i-1,j-2)/54.0
     &          -79.0*sbn(i-2,j-2)/432.0
     &          +sbn(i-3,j-2)/54.0
     &          -sbn(i-4,j-2)/1728.0

      dby4dx2=    -sbn(i+2,j+4)/1728.0 +
sbn(i+2,j+3)/54.0
     &          -79.0*sbn(i+2,j+2)/432.0
     &          +31.0*sbn(i+2,j+1)/54.0
     &          -707.0*sbn(i+2,j)/864.0
     &          +31.0*sbn(i+2,j-1)/54.0
     &          -79.0*sbn(i+2,j-2)/432.0
     &          +sbn(i+2,j-3)/54.0
     &          -sbn(i+2,j-4)/1728.0

     &          +16.0*sbn(i+1,j+4)/1728.0
     &          -16.0*sbn(i+1,j+3)/54.0
     &          +16.0*79.0*sbn(i+1,j+2)/432.0
     &          -16.0*31.0*sbn(i+1,j+1)/54.0
     &          +16.0*707.0*sbn(i+1,j)/864.0
     &          -16.0*31.0*sbn(i+1,j-1)/54.0
     &          +16.0*79.0*sbn(i+1,j-2)/432.0
```

```
     &          -16.0*sbn(i+1,j-3)/54.0
     &          +16.0*sbn(i+1,j-4)/1728.0

     &          -30.0*sbn(i,j+4)/1728.0
     &          +30.0*sbn(i,j+3)/54.0
     &          -30.0*79.0*sbn(i,j+2)/432.0
     &          +30.0*31.0*sbn(i,j+1)/54.0
     &          -30.0*707.0*sbn(i,j)/864.0
     &          +30.0*31.0*sbn(i,j-1)/54.0
     &          -30.0*79.0*sbn(i,j-2)/432.0
     &          +30.0*sbn(i,j-3)/54.0
     &          -30.0*sbn(i,j-4)/1728.0

     &          +16.0*sbn(i-1,j+4)/1728.0
     &          -16.0*sbn(i-1,j+3)/54.0
     &          +16.0*79.0*sbn(i-1,j+2)/432.0
     &          -16.0*31.0*sbn(i-1,j+1)/54.0
     &          +16.0*707.0*sbn(i-1,j)/864.0
     &          -16.0*31.0*sbn(i-1,j-1)/54.0
     &          +16.0*79.0*sbn(i-1,j-2)/432.0
     &          -16.0*sbn(i-1,j-3)/54.0
     &          +16.0*sbn(i-1,j-4)/1728.0

     &          -sbn(i-2,j+4)/1728.0 + sbn(i-2,j+3)/54.0
     &          -79.0*sbn(i-2,j+2)/432.0
     &          +31.0*sbn(i-2,j+1)/54.0
     &          -707.0*sbn(i-2,j)/864.0
     &          +31.0*sbn(i-2,j-1)/54.0
     &          -79.0*sbn(i-2,j-2)/432.0
     &          +sbn(i-2,j-3)/54.0
     &          -sbn(i-2,j-4)/1728.0

       rn1(i,j) = rn(i,j) + dtx*dbx2 + dtx*dby2
     &          + dt*c1*sbn(i,j)
     &          - dtx3*dbx6/24.0
     &          - 3.0*dtx3*dbx4dy2/24.0
     &          - 3.0*dtx3*dby4dx2/24.0
     &          - 6.0*dtx2*dt*c1*dby2dx2/24.0
     &          - 3.0*dtx2*dt*c1*dbx4/24.0
     &          - dtx3*dby6/24.0
     &          - 3.0*dtx2*dt*c1*dby4/24.0
     &          - 3.0*dtx*dt2*c12*dbx2/24.0
     &          - 3.0*dtx*dt2*c12*dby2/24.0
     &          - dt3*c13*sbn(i,j)/24.0

          enddo
          enddo

c Set up the system for sn1(i)
c Bright Soliton

      do i=0,nx
      do j=0,6
       sn1(i,j)= 2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
     &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
     &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))
C    &         +2.0D0*sin(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &         /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &         +exp(-4.0D0*it*dt-x(i)+10.0D0))
      enddo
      enddo

      do i=0,nx
      do j=ny-6,ny
       sn1(i,j)= 2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
     &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
     &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))
C    &         +2.0D0*sin(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &         /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &         +exp(-4.0D0*it*dt-x(i)+10.0D0))
      enddo
      enddo

      do i=0,6
      do j=7,ny-7
       sn1(i,j)= 2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
     &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
     &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))
C    &         +2.0D0*sin(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &         /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &         +exp(-4.0D0*it*dt-x(i)+10.0D0))
      enddo
      enddo

      do i=nx-6,nx
      do j=7,ny-7
       sn1(i,j)= 2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
     &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
     &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))
C    &         +2.0D0*sin(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &         /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &         +exp(-4.0D0*it*dt-x(i)+10.0D0))
      enddo
      enddo

c**************************
      do i=7,nx-7
      do j=7,ny-7
      c1=4.0*(rbn(i,j)*rbn(i,j)+sbn(i,j)*sbn(i,j))**mp
      c12=c1*c1
      c13=c12*c1

      dbx21=  (-rbn(i+2,j)+16.0*rbn(i+1,j)-
30.0*rbn(i,j)
     &        +16.0*rbn(i-1,j) - rbn(i-2,j))/12.0

      dby21=  (-rbn(i,j+2)+16.0*rbn(i,j+1)-
30.0*rbn(i,j)
     &        +16.0*rbn(i,j-1) - rbn(i,j-2))/12.0

      dbx41=rbn(i+4,j)/144.0 - 2.0*rbn(i+3,j)/9.0
     &        +79.0*rbn(i+2,j)/36.0 - 62.0*rbn(i+1,j)/9.0
     &        +707.0*rbn(i,j)/72.0  - 62.0*rbn(i-1,j)/9.0
     &        +79.0*rbn(i-2,j)/36.0 - 2.0*rbn(i-3,j)/9.0
     &        +rbn(i-4,j)/144.0
```

```
      dby41=rbn(i,j+4)/144.0 - 2.0*rbn(i,j+3)/9.0
     &       +79.0*rbn(i,j+2)/36.0 - 62.0*rbn(i,j+1)/9.0
     &       +707.0*rbn(i,j)/72.0  - 62.0*rbn(i,j-1)/9.0
     &       +79.0*rbn(i,j-2)/36.0 - 2.0*rbn(i,j-3)/9.0
     &       +rbn(i,j-4)/144.0

      dbx61= -rbn(i+6,j)/1728.0 + rbn(i+5,j)/36.0
     &       -143.0*rbn(i+4,j)/288.0 +
439.0*rbn(i+3,j)/108.0
     &       -3031.0*rbn(i+2,j)/192.0 +
203.0*rbn(i+1,j)/6.0
     &       -6233.0*rbn(i,j)/144.0 + 203.0*rbn(i-
1,j)/6.0
     &       -3031.0*rbn(i-2,j)/192.0 + 439.0*rbn(i-
3,j)/108.0
     &       -143.0*rbn(i-4,j)/288.0 + rbn(i-5,j)/36.0
     &       -rbn(i-6,j)/1728.0

      dby61= -rbn(i,j+6)/1728.0 + rbn(i,j+5)/36.0
     &       -143.0*rbn(i,j+4)/288.0 +
439.0*rbn(i,j+3)/108.0
     &       -3031.0*rbn(i,j+2)/192.0 +
203.0*rbn(i,j+1)/6.0
     &       -6233.0*rbn(i,j)/144.0 + 203.0*rbn(i,j-
1)/6.0
     &       -3031.0*rbn(i,j-2)/192.0 + 439.0*rbn(i,j-
3)/108.0
     &       -143.0*rbn(i,j-4)/288.0 + rbn(i,j-5)/36.0
     &       -rbn(i,j-6)/1728.0

      dby2dx21= (rbn(i+2,j+2) - 16.0*rbn(i+1,j+2)
     &         +30.0*rbn(i,j+2)
     &         -16.0*rbn(i-1,j+2)
     &         +rbn(i-2,j+2)

     &         -16.0*rbn(i+2,j+1) +
16.0*16.0*rbn(i+1,j+1)
     &         -16.0*30.0*rbn(i,j+1)
     &         +16.0*16.0*rbn(i-1,j+1)
     &         -16.0*rbn(i-2,j+1)

     &         +30.0*rbn(i+2,j) - 30.0*16.0*rbn(i+1,j)
     &         +30.0*30.0*rbn(i,j)
     &         -30.0*16.0*rbn(i-1,j)
     &         +30.0*rbn(i-2,j)

     &         -16.0*rbn(i+2,j-1) + 16.0*16.0*rbn(i+1,j-
1)
     &         -16.0*30.0*rbn(i,j-1)
     &         +16.0*16.0*rbn(i-1,j-1)
     &         -16.0*rbn(i-2,j-1)

     &         +rbn(i+2,j-2) - 16.0*rbn(i+1,j-2)
     &         +30.0*rbn(i,j-2)
     &         -16.0*rbn(i-1,j-2)
     &         +rbn(i-2,j-2))/144.0

      dbx4dy21=   -rbn(i+4,j+2)/1728.0 +
rbn(i+3,j+2)/54.0
     &         -79.0*rbn(i+2,j+2)/432.0
     &         +31.0*rbn(i+1,j+2)/54.0
     &         -707.0*rbn(i,j+2)/864.0
     &         +31.0*rbn(i-1,j+2)/54.0
     &         -79.0*rbn(i-2,j+2)/432.0
     &         +rbn(i-3,j+2)/54.0
     &         -rbn(i-4,j+2)/1728.0

     &         +16.0*rbn(i+4,j+1)/1728.0
     &         -16.0*rbn(i+3,j+1)/54.0
     &         +16.0*79.0*rbn(i+2,j+1)/432.0
     &         -16.0*31.0*rbn(i+1,j+1)/54.0
     &         +16.0*707.0*rbn(i,j+1)/864.0
     &         -16.0*31.0*rbn(i-1,j+1)/54.0
     &         +16.0*79.0*rbn(i-2,j+1)/432.0
     &         -16.0*rbn(i-3,j+1)/54.0
     &         +16.0*rbn(i-4,j+1)/1728.0

     &         -30.0*rbn(i+4,j)/1728.0
     &         +30.0*rbn(i+3,j)/54.0
     &         -30.0*79.0*rbn(i+2,j)/432.0
     &         +30.0*31.0*rbn(i+1,j)/54.0
     &         -30.0*707.0*rbn(i,j)/864.0
     &         +30.0*31.0*rbn(i-1,j)/54.0
     &         -30.0*79.0*rbn(i-2,j)/432.0
     &         +30.0*rbn(i-3,j)/54.0
     &         -30.0*rbn(i-4,j)/1728.0

     &         +16.0*rbn(i+4,j-1)/1728.0
     &         -16.0*rbn(i+3,j-1)/54.0
     &         +16.0*79.0*rbn(i+2,j-1)/432.0
     &         -16.0*31.0*rbn(i+1,j-1)/54.0
     &         +16.0*707.0*rbn(i,j-1)/864.0
     &         -16.0*31.0*rbn(i-1,j-1)/54.0
     &         +16.0*79.0*rbn(i-2,j-1)/432.0
     &         -16.0*rbn(i-3,j-1)/54.0
     &         +16.0*rbn(i-4,j-1)/1728.0

     &         -rbn(i+4,j-2)/1728.0 + rbn(i+3,j-2)/54.0
     &         -79.0*rbn(i+2,j-2)/432.0
     &         +31.0*rbn(i+1,j-2)/54.0
     &         -707.0*rbn(i,j-2)/864.0
     &         +31.0*rbn(i-1,j-2)/54.0
     &         -79.0*rbn(i-2,j-2)/432.0
     &         +rbn(i-3,j-2)/54.0
     &         -rbn(i-4,j-2)/1728.0

      dby4dx21=   -rbn(i+2,j+4)/1728.0 +
rbn(i+2,j+3)/54.0
     &         -79.0*rbn(i+2,j+2)/432.0
     &         +31.0*rbn(i+2,j+1)/54.0
     &         -707.0*rbn(i+2,j)/864.0
     &         +31.0*rbn(i+2,j-1)/54.0
     &         -79.0*rbn(i+2,j-2)/432.0
     &         +rbn(i+2,j-3)/54.0
     &         -rbn(i+2,j-4)/1728.0

     &         +16.0*rbn(i+1,j+4)/1728.0
     &         -16.0*rbn(i+1,j+3)/54.0
     &         +16.0*79.0*rbn(i+1,j+2)/432.0
     &         -16.0*31.0*rbn(i+1,j+1)/54.0
     &         +16.0*707.0*rbn(i+1,j)/864.0
     &         -16.0*31.0*rbn(i+1,j-1)/54.0
     &         +16.0*79.0*rbn(i+1,j-2)/432.0
     &         -16.0*rbn(i+1,j-3)/54.0
```

```
     &          +16.0*rbn(i+1,j-4)/1728.0

     &          -30.0*rbn(i,j+4)/1728.0
     &          +30.0*rbn(i,j+3)/54.0
     &          -30.0*79.0*rbn(i,j+2)/432.0
     &          +30.0*31.0*rbn(i,j+1)/54.0
     &          -30.0*707.0*rbn(i,j)/864.0
     &          +30.0*31.0*rbn(i,j-1)/54.0
     &          -30.0*79.0*rbn(i,j-2)/432.0
     &          +30.0*rbn(i,j-3)/54.0
     &          -30.0*rbn(i,j-4)/1728.0

     &          +16.0*rbn(i-1,j+4)/1728.0
     &          -16.0*rbn(i-1,j+3)/54.0
     &          +16.0*79.0*rbn(i-1,j+2)/432.0
     &          -16.0*31.0*rbn(i-1,j+1)/54.0
     &          +16.0*707.0*rbn(i-1,j)/864.0
     &          -16.0*31.0*rbn(i-1,j-1)/54.0
     &          +16.0*79.0*rbn(i-1,j-2)/432.0
     &          -16.0*rbn(i-1,j-3)/54.0
     &          +16.0*rbn(i-1,j-4)/1728.0

     &          -rbn(i-2,j+4)/1728.0 + rbn(i-2,j+3)/54.0
     &          -79.0*rbn(i-2,j+2)/432.0
     &          +31.0*rbn(i-2,j+1)/54.0
     &          -707.0*rbn(i-2,j)/864.0
     &          +31.0*rbn(i-2,j-1)/54.0
     &          -79.0*rbn(i-2,j-2)/432.0
     &          +rbn(i-2,j-3)/54.0
     &          -rbn(i-2,j-4)/1728.0

       sn1(i,j) = sn(i,j) - dtx*dbx21 - dtx*dby21
     &          - dt*c1*rbn(i,j)
     &          + dtx3*dbx61/24.0
     &          + 3.0*dtx3*dbx4dy21/24.0
     &          + 3.0*dtx3*dby4dx21/24.0
     &          + 6.0*dtx2*dt*c1*dby2dx21/24.0
     &          + 3.0*dtx2*dt*c1*dbx41/24.0
     &          + dtx3*dby61/24.0
     &          + 3.0*dtx2*dt*c1*dby41/24.0
     &          + 3.0*dtx*dt2*c12*dbx21/24.0
     &          + 3.0*dtx*dt2*c12*dby21/24.0
     &          + dt3*c13*rbn(i,j)/24.0

          enddo
          enddo

c setp up the system for rbn1(i)
c Bright Soliton

      do i=0,nx
      do j=0,6
       rbn1(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
     &        +6.0D0*(it+0.5D0)*dt)
     &        /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
     &        +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))
C    &         +2.0D0*cos(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &         /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &         +exp(-4.0D0*it*dt-x(i)+10.0D0))
      enddo
      enddo

      do i=0,nx
      do j=ny-6,ny
       rbn1(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
     &        +6.0D0*(it+0.5D0)*dt)
     &        /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
     &        +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))
C    &         +2.0D0*cos(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &         /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &         +exp(-4.0D0*it*dt-x(i)+10.0D0))
      enddo
      enddo

       do i=0,6
      do j=7,ny-7
       rbn1(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
     &        +6.0D0*(it+0.5D0)*dt)
     &        /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
     &        +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))
C    &         +2.0D0*cos(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &         /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &         +exp(-4.0D0*it*dt-x(i)+10.0D0))
      enddo
      enddo

      do i=nx-6,nx
      do j=7,ny-7
       rbn1(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
     &        +6.0D0*(it+0.5D0)*dt)
     &        /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
     &        +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))
C    &         +2.0D0*cos(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &         /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &         +exp(-4.0D0*it*dt-x(i)+10.0D0))
      enddo
      enddo

c**************************
      do i=7,nx-7
      do j=7,ny-7
      cb1=4.0*(rn1(i,j)*rn1(i,j)+sn1(i,j)*sn1(i,j))**mp
      cb12=cb1*cb1
      cb13=cb12*cb1

      dx21=(-sn1(i+2,j)+16.0*sn1(i+1,j)-30.0*sn1(i,j)
     &        +16.0*sn1(i-1,j)
     &        -sn1(i-2,j))/12.0
```

```
      dy21=(-sn1(i,j+2)+16.0*sn1(i,j+1)-30.0*sn1(i,j)
     &       +16.0*sn1(i,j-1)
     &       -sn1(i,j-2))/12.0

      dx41=sn1(i+4,j)/144.0 - 2.0*sn1(i+3,j)/9.0
     &       +79.0*sn1(i+2,j)/36.0 - 62.0*sn1(i+1,j)/9.0
     &       +707.0*sn1(i,j)/72.0  - 62.0*sn1(i-1,j)/9.0
     &       +79.0*sn1(i-2,j)/36.0 - 2.0*sn1(i-3,j)/9.0
     &       +sn1(i-4,j)/144.0

      dy41=sn1(i,j+4)/144.0 - 2.0*sn1(i,j+3)/9.0
     &       +79.0*sn1(i,j+2)/36.0 - 62.0*sn1(i,j+1)/9.0
     &       +707.0*sn1(i,j)/72.0  - 62.0*sn1(i,j-1)/9.0
     &       +79.0*sn1(i,j-2)/36.0 - 2.0*sn1(i,j-3)/9.0
     &       +sn1(i,j-4)/144.0

      dx61=-sn1(i+6,j)/1728.0 + sn1(i+5,j)/36.0
     &       -143.0*sn1(i+4,j)/288.0 +
439.0*sn1(i+3,j)/108.0
     &       -3031.0*sn1(i+2,j)/192.0 +
203.0*sn1(i+1,j)/6.0
     &       -6233.0*sn1(i,j)/144.0 + 203.0*sn1(i-
1,j)/6.0
     &       -3031.0*sn1(i-2,j)/192.0 + 439.0*sn1(i-
3,j)/108.0
     &       -143.0*sn1(i-4,j)/288.0 + sn1(i-5,j)/36.0
     &       -sn1(i-6,j)/1728.0

      dy61=-sn1(i,j+6)/1728.0 + sn1(i,j+5)/36.0
     &       -143.0*sn1(i,j+4)/288.0 +
439.0*sn1(i,j+3)/108.0
     &       -3031.0*sn1(i,j+2)/192.0 +
203.0*sn1(i,j+1)/6.0
     &       -6233.0*sn1(i,j)/144.0 + 203.0*sn1(i,j-
1)/6.0
     &       -3031.0*sn1(i,j-2)/192.0 + 439.0*sn1(i,j-
3)/108.0
     &       -143.0*sn1(i,j-4)/288.0 + sn1(i,j-5)/36.0
     &       -sn1(i,j-6)/1728.0

      dx4dy21= -sn1(i+4,j+2)/1728.0 +
sn1(i+3,j+2)/54.0
     &         -79.0*sn1(i+2,j+2)/432.0
     &         +31.0*sn1(i+1,j+2)/54.0
     &         -707.0*sn1(i,j+2)/864.0
     &         +31.0*sn1(i-1,j+2)/54.0
     &         -79.0*sn1(i-2,j+2)/432.0
     &         +sn1(i-3,j+2)/54.0
     &         -sn1(i-4,j+2)/1728.0

     &         +16.0*sn1(i+4,j+1)/1728.0
     &         -16.0*sn1(i+3,j+1)/54.0
     &         +16.0*79.0*sn1(i+2,j+1)/432.0
     &         -16.0*31.0*sn1(i+1,j+1)/54.0
     &         +16.0*707.0*sn1(i,j+1)/864.0
     &         -16.0*31.0*sn1(i-1,j+1)/54.0
     &         +16.0*79.0*sn1(i-2,j+1)/432.0
     &         -16.0*sn1(i-3,j+1)/54.0
     &         +16.0*sn1(i-4,j+1)/1728.0

     &         -30.0*sn1(i+4,j)/1728.0
     &         +30.0*sn1(i+3,j)/54.0
     &         -30.0*79.0*sn1(i+2,j)/432.0
     &         +30.0*31.0*sn1(i+1,j)/54.0
     &         -30.0*707.0*sn1(i,j)/864.0
     &         +30.0*31.0*sn1(i-1,j)/54.0
     &         -30.0*79.0*sn1(i-2,j)/432.0
     &         +30.0*sn1(i-3,j)/54.0
     &         -30.0*sn1(i-4,j)/1728.0

     &         +16.0*sn1(i+4,j-1)/1728.0
     &         -16.0*sn1(i+3,j-1)/54.0
     &         +16.0*79.0*sn1(i+2,j-1)/432.0
     &         -16.0*31.0*sn1(i+1,j-1)/54.0
     &         +16.0*707.0*sn1(i,j-1)/864.0
     &         -16.0*31.0*sn1(i-1,j-1)/54.0
     &         +16.0*79.0*sn1(i-2,j-1)/432.0
     &         -16.0*sn1(i-3,j-1)/54.0
     &         +16.0*sn1(i-4,j-1)/1728.0

     &         -sn1(i+4,j-2)/1728.0 + sn1(i+3,j-2)/54.0
     &         -79.0*sn1(i+2,j-2)/432.0
     &         +31.0*sn1(i+1,j-2)/54.0
     &         -707.0*sn1(i,j-2)/864.0
     &         +31.0*sn1(i-1,j-2)/54.0
     &         -79.0*sn1(i-2,j-2)/432.0
     &         +sn1(i-3,j-2)/54.0
     &         -sn1(i-4,j-2)/1728.0

      dy4dx21= -sn1(i+2,j+4)/1728.0 +
sn1(i+2,j+3)/54.0
     &         -79.0*sn1(i+2,j+2)/432.0
     &         +31.0*sn1(i+2,j+1)/54.0
     &         -707.0*sn1(i+2,j)/864.0
     &         +31.0*sn1(i+2,j-1)/54.0
     &         -79.0*sn1(i+2,j-2)/432.0
     &         +sn1(i+2,j-3)/54.0
     &         -sn1(i+2,j-4)/1728.0

     &         +16.0*sn1(i+1,j+4)/1728.0
     &         -16.0*sn1(i+1,j+3)/54.0
     &         +16.0*79.0*sn1(i+1,j+2)/432.0
     &         -16.0*31.0*sn1(i+1,j+1)/54.0
     &         +16.0*707.0*sn1(i+1,j)/864.0
     &         -16.0*31.0*sn1(i+1,j-1)/54.0
     &         +16.0*79.0*sn1(i+1,j-2)/432.0
     &         -16.0*sn1(i+1,j-3)/54.0
     &         +16.0*sn1(i+1,j-4)/1728.0

     &         -30.0*sn1(i,j+4)/1728.0
     &         +30.0*sn1(i,j+3)/54.0
     &         -30.0*79.0*sn1(i,j+2)/432.0
     &         +30.0*31.0*sn1(i,j+1)/54.0
     &         -30.0*707.0*sn1(i,j)/864.0
     &         +30.0*31.0*sn1(i,j-1)/54.0
     &         -30.0*79.0*sn1(i,j-2)/432.0
     &         +30.0*sn1(i,j-3)/54.0
     &         -30.0*sn1(i,j-4)/1728.0

     &         +16.0*sn1(i-1,j+4)/1728.0
     &         -16.0*sn1(i-1,j+3)/54.0
     &         +16.0*79.0*sn1(i-1,j+2)/432.0
     &         -16.0*31.0*sn1(i-1,j+1)/54.0
```

```
     &          +16.0*707.0*sn1(i-1,j)/864.0
     &          -16.0*31.0*sn1(i-1,j-1)/54.0
     &          +16.0*79.0*sn1(i-1,j-2)/432.0
     &          -16.0*sn1(i-1,j-3)/54.0
     &          +16.0*sn1(i-1,j-4)/1728.0

     &          -sn1(i-2,j+4)/1728.0 + sn1(i-2,j+3)/54.0
     &          -79.0*sn1(i-2,j+2)/432.0
     &          +31.0*sn1(i-2,j+1)/54.0
     &          -707.0*sn1(i-2,j)/864.0
     &          +31.0*sn1(i-2,j-1)/54.0
     &          -79.0*sn1(i-2,j-2)/432.0
     &          +sn1(i-2,j-3)/54.0
     &          -sn1(i-2,j-4)/1728.0

        dy2dx21= (sn1(i+2,j+2) - 16.0*sn1(i+1,j+2)
     &          +30.0*sn1(i,j+2)
     &          -16.0*sn1(i-1,j+2)
     &          +sn1(i-2,j+2)

     &          -16.0*sn1(i+2,j+1) +
16.0*16.0*sn1(i+1,j+1)
     &          -16.0*30.0*sn1(i,j+1)
     &          +16.0*16.0*sn1(i-1,j+1)
     &          -16.0*sn1(i-2,j+1)

     &          +30.0*sn1(i+2,j) - 30.0*16.0*sn1(i+1,j)
     &          +30.0*30.0*sn1(i,j)
     &          -30.0*16.0*sn1(i-1,j)
     &          +30.0*sn1(i-2,j)

     &          -16.0*sn1(i+2,j-1) +
16.0*16.0*sn1(i+1,j-1)
     &          -16.0*30.0*sn1(i,j-1)
     &          +16.0*16.0*sn1(i-1,j-1)
     &          -16.0*sn1(i-2,j-1)

     &          +sn1(i+2,j-2) - 16.0*sn1(i+1,j-2)
     &          +30.0*sn1(i,j-2)
     &          -16.0*sn1(i-1,j-2)
     &          +sn1(i-2,j-2))/144.0

        rbn1(i,j) = rbn(i,j) + dtx*dx21 + dtx*dy21
     &          + dt*cb1*sn1(i,j)
     &          - dtx3*dx61/24.0
     &          - 3.0*dtx3*dx4dy21/24.0
     &          - 3.0*dtx3*dy4dx21/24.0
     &          - 6.0*dtx2*dt*cb1*dy2dx21/24.0
     &          - 3.0*dtx2*dt*cb1*dx41/24.0
     &          - dtx3*dy61/24.0
     &          - 3.0*dtx2*dt*cb1*dy41/24.0
     &          - 3.0*dtx*dt2*cb12*dx21/24.0
     &          - 3.0*dtx*dt2*cb12*dy21/24.0
     &          - dt3*cb13*sn1(i,j)/24.0

           enddo
           enddo

c solve sbn1(i)
c Bright soliton

      do i=0,nx
      do j=0,6
       sbn1(i,j)= 2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
     &          +6.0D0*(it+0.5D0)*dt)
     &          /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
     &          +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))
C    &          +2.0D0*sin(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &          /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &          +exp(-4.0D0*it*dt-x(i)+10.0D0))
      enddo
      enddo

      do i=0,nx
      do j=ny-6,ny
       sbn1(i,j)= 2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
     &          +6.0D0*(it+0.5D0)*dt)
     &          /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
     &          +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))
C    &          +2.0D0*sin(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &          /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &          +exp(-4.0D0*it*dt-x(i)+10.0D0))
      enddo
      enddo

      do i=0,6
      do j=7,ny-7
       sbn1(i,j)= 2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
     &          +6.0D0*(it+0.5D0)*dt)
     &          /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
     &          +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))
C    &          +2.0D0*sin(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &          /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &          +exp(-4.0D0*it*dt-x(i)+10.0D0))
      enddo
      enddo

      do i=nx-6,nx
      do j=7,ny-7
       sbn1(i,j)= 2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
     &          +6.0D0*(it+0.5D0)*dt)
     &          /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
     &          +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))
C    &          +2.0D0*sin(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &          /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &          +exp(-4.0D0*it*dt-x(i)+10.0D0))
      enddo
      enddo
```

```
c*************************
      do i=7,nx-7
      do j=7,ny-7
      cb1=4.0*(rn1(i,j)*rn1(i,j)+sn1(i,j)*sn1(i,j))**mp
      cb12=cb1*cb1
      cb13=cb12*cb1

      dx21=(-rn1(i+2,j)+16.0*rn1(i+1,j)-30.0*rn1(i,j)
     &       +16.0*rn1(i-1,j)
     &       -rn1(i-2,j))/12.0

      dy21=(-rn1(i,j+2)+16.0*rn1(i,j+1)-30.0*rn1(i,j)
     &       +16.0*rn1(i,j-1)
     &       -rn1(i,j-2))/12.0

      dx41=rn1(i+4,j)/144.0 - 2.0*rn1(i+3,j)/9.0
     &       +79.0*rn1(i+2,j)/36.0 - 62.0*rn1(i+1,j)/9.0
     &       +707.0*rn1(i,j)/72.0  - 62.0*rn1(i-1,j)/9.0
     &       +79.0*rn1(i-2,j)/36.0 - 2.0*rn1(i-3,j)/9.0
     &       +rn1(i-4,j)/144.0

      dy41=rn1(i,j+4)/144.0 - 2.0*rn1(i,j+3)/9.0
     &       +79.0*rn1(i,j+2)/36.0 - 62.0*rn1(i,j+1)/9.0
     &       +707.0*rn1(i,j)/72.0  - 62.0*rn1(i,j-1)/9.0
     &       +79.0*rn1(i,j-2)/36.0 - 2.0*rn1(i,j-3)/9.0
     &       +rn1(i,j-4)/144.0

      dx61=-rn1(i+6,j)/1728.0 + rn1(i+5,j)/36.0
     &       -143.0*rn1(i+4,j)/288.0 +
439.0*rn1(i+3,j)/108.0
     &       -3031.0*rn1(i+2,j)/192.0 +
203.0*rn1(i+1,j)/6.0
     &       -6233.0*rn1(i,j)/144.0 + 203.0*rn1(i-
1,j)/6.0
     &       -3031.0*rn1(i-2,j)/192.0 + 439.0*rn1(i-
3,j)/108.0
     &       -143.0*rn1(i-4,j)/288.0 + rn1(i-5,j)/36.0
     &       -rn1(i-6,j)/1728.0

      dy61=-rn1(i,j+6)/1728.0 + rn1(i,j+5)/36.0
     &       -143.0*rn1(i,j+4)/288.0 +
439.0*rn1(i,j+3)/108.0
     &       -3031.0*rn1(i,j+2)/192.0 +
203.0*rn1(i,j+1)/6.0
     &       -6233.0*rn1(i,j)/144.0 + 203.0*rn1(i,j-
1)/6.0
     &       -3031.0*rn1(i,j-2)/192.0 + 439.0*rn1(i,j-
3)/108.0
     &       -143.0*rn1(i,j-4)/288.0 + rn1(i,j-5)/36.0
     &       -rn1(i,j-6)/1728.0

      dx4dy21= -rn1(i+4,j+2)/1728.0 +
rn1(i+3,j+2)/54.0
     &         -79.0*rn1(i+2,j+2)/432.0
     &         +31.0*rn1(i+1,j+2)/54.0
     &         -707.0*rn1(i,j+2)/864.0
     &         +31.0*rn1(i-1,j+2)/54.0
     &         -79.0*rn1(i-2,j+2)/432.0
     &         +rn1(i-3,j+2)/54.0
     &         -rn1(i-4,j+2)/1728.0

     &         +16.0*rn1(i+4,j+1)/1728.0
     &         -16.0*rn1(i+3,j+1)/54.0
     &         +16.0*79.0*rn1(i+2,j+1)/432.0
     &         -16.0*31.0*rn1(i+1,j+1)/54.0
     &         +16.0*707.0*rn1(i,j+1)/864.0
     &         -16.0*31.0*rn1(i-1,j+1)/54.0
     &         +16.0*79.0*rn1(i-2,j+1)/432.0
     &         -16.0*rn1(i-3,j+1)/54.0
     &         +16.0*rn1(i-4,j+1)/1728.0

     &         -30.0*rn1(i+4,j)/1728.0
     &         +30.0*rn1(i+3,j)/54.0
     &         -30.0*79.0*rn1(i+2,j)/432.0
     &         +30.0*31.0*rn1(i+1,j)/54.0
     &         -30.0*707.0*rn1(i,j)/864.0
     &         +30.0*31.0*rn1(i-1,j)/54.0
     &         -30.0*79.0*rn1(i-2,j)/432.0
     &         +30.0*rn1(i-3,j)/54.0
     &         -30.0*rn1(i-4,j)/1728.0

     &         +16.0*rn1(i+4,j-1)/1728.0
     &         -16.0*rn1(i+3,j-1)/54.0
     &         +16.0*79.0*rn1(i+2,j-1)/432.0
     &         -16.0*31.0*rn1(i+1,j-1)/54.0
     &         +16.0*707.0*rn1(i,j-1)/864.0
     &         -16.0*31.0*rn1(i-1,j-1)/54.0
     &         +16.0*79.0*rn1(i-2,j-1)/432.0
     &         -16.0*rn1(i-3,j-1)/54.0
     &         +16.0*rn1(i-4,j-1)/1728.0

     &         -rn1(i+4,j-2)/1728.0 + rn1(i+3,j-2)/54.0
     &         -79.0*rn1(i+2,j-2)/432.0
     &         +31.0*rn1(i+1,j-2)/54.0
     &         -707.0*rn1(i,j-2)/864.0
     &         +31.0*rn1(i-1,j-2)/54.0
     &         -79.0*rn1(i-2,j-2)/432.0
     &         +rn1(i-3,j-2)/54.0
     &         -rn1(i-4,j-2)/1728.0

      dy4dx21= -rn1(i+2,j+4)/1728.0 +
rn1(i+2,j+3)/54.0
     &         -79.0*rn1(i+2,j+2)/432.0
     &         +31.0*rn1(i+2,j+1)/54.0
     &         -707.0*rn1(i+2,j)/864.0
     &         +31.0*rn1(i+2,j-1)/54.0
     &         -79.0*rn1(i+2,j-2)/432.0
     &         +rn1(i+2,j-3)/54.0
     &         -rn1(i+2,j-4)/1728.0

     &         +16.0*rn1(i+1,j+4)/1728.0
     &         -16.0*rn1(i+1,j+3)/54.0
     &         +16.0*79.0*rn1(i+1,j+2)/432.0
     &         -16.0*31.0*rn1(i+1,j+1)/54.0
     &         +16.0*707.0*rn1(i+1,j)/864.0
     &         -16.0*31.0*rn1(i+1,j-1)/54.0
     &         +16.0*79.0*rn1(i+1,j-2)/432.0
     &         -16.0*rn1(i+1,j-3)/54.0
     &         +16.0*rn1(i+1,j-4)/1728.0

     &         -30.0*rn1(i,j+4)/1728.0
     &         +30.0*rn1(i,j+3)/54.0
     &         -30.0*79.0*rn1(i,j+2)/432.0
```

```
     &        +30.0*31.0*rn1(i,j+1)/54.0
     &        -30.0*707.0*rn1(i,j)/864.0
     &        +30.0*31.0*rn1(i,j-1)/54.0
     &        -30.0*79.0*rn1(i,j-2)/432.0
     &        +30.0*rn1(i,j-3)/54.0
     &        -30.0*rn1(i,j-4)/1728.0

     &        +16.0*rn1(i-1,j+4)/1728.0
     &        -16.0*rn1(i-1,j+3)/54.0
     &        +16.0*79.0*rn1(i-1,j+2)/432.0
     &        -16.0*31.0*rn1(i-1,j+1)/54.0
     &        +16.0*707.0*rn1(i-1,j)/864.0
     &        -16.0*31.0*rn1(i-1,j-1)/54.0
     &        +16.0*79.0*rn1(i-1,j-2)/432.0
     &        -16.0*rn1(i-1,j-3)/54.0
     &        +16.0*rn1(i-1,j-4)/1728.0

     &        -rn1(i-2,j+4)/1728.0 + rn1(i-2,j+3)/54.0
     &        -79.0*rn1(i-2,j+2)/432.0
     &        +31.0*rn1(i-2,j+1)/54.0
     &        -707.0*rn1(i-2,j)/864.0
     &        +31.0*rn1(i-2,j-1)/54.0
     &        -79.0*rn1(i-2,j-2)/432.0
     &        +rn1(i-2,j-3)/54.0
     &        -rn1(i-2,j-4)/1728.0

       dy2dx21= (rn1(i+2,j+2) - 16.0*rn1(i+1,j+2)
     &        +30.0*rn1(i,j+2)
     &        -16.0*rn1(i-1,j+2)
     &        +rn1(i-2,j+2)

     &        -16.0*rn1(i+2,j+1) +
16.0*16.0*rn1(i+1,j+1)
     &        -16.0*30.0*rn1(i,j+1)
     &        +16.0*16.0*rn1(i-1,j+1)
     &        -16.0*rn1(i-2,j+1)

     &        +30.0*rn1(i+2,j) - 30.0*16.0*rn1(i+1,j)
     &        +30.0*30.0*rn1(i,j)
     &        -30.0*16.0*rn1(i-1,j)
     &        +30.0*rn1(i-2,j)

     &        -16.0*rn1(i+2,j-1) + 16.0*16.0*rn1(i+1,j-
1)
     &        -16.0*30.0*rn1(i,j-1)
     &        +16.0*16.0*rn1(i-1,j-1)
     &        -16.0*rn1(i-2,j-1)

     &        +rn1(i+2,j-2) - 16.0*rn1(i+1,j-2)
     &        +30.0*rn1(i,j-2)
     &        -16.0*rn1(i-1,j-2)
     &        +rn1(i-2,j-2))/144.0

       sbn1(i,j) = sbn(i,j) - dtx*dx21 - dtx*dy21
     &        - dt*cb1*rn1(i,j)
     &        + dtx3*dx61/24.0
     &        + 3.0*dtx3*dx4dy21/24.0
     &        + 3.0*dtx3*dy4dx21/24.0
     &        + 6.0*dtx2*dt*cb1*dy2dx21/24.0
     &        + 3.0*dtx2*dt*cb1*dx41/24.0
     &        + dtx3*dy61/24.0
     &        + 3.0*dtx2*dt*cb1*dy41/24.0
     &        + 3.0*dtx*dt2*cb12*dx21/24.0
     &        + 3.0*dtx*dt2*cb12*dy21/24.0
     &        + dt3*cb13*rn1(i,j)/24.0

          enddo
          enddo

c calculate the exact solution
c Bright Soliton

      do i=0,nx
      do j=0,ny
      rexac(i,j)= 2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
     &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
     &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))
C    &         +2.0D0*cos(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &         /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &         +exp(-4.0D0*it*dt-x(i)+10.0D0))

      sexac(i,j)=2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)+6.0D0*it*dt)
     &        /(exp(x(i)+y(j)-8.0D0*it*dt+10.0D0)
     &        +exp(8.0D0*it*dt-x(i)-y(j)-10.0D0))
C    &         +2.0D0*sin(2.0D0*x(i)+3.0D0*it*dt-
20.0D0)
C    &         /(exp(x(i)+4.0D0*it*dt-10.0D0)
C    &         +exp(-4.0D0*it*dt-x(i)+10.0D0))

      rbexac(i,j)=2.0D0*cos(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
     &        +6.0D0*(it+0.5D0)*dt)
     &        /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
     &        +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))
C    &
+2.0D0*cos(2.0D0*x(i)+3.0D0*(it+0.5D0)*dt-
20.0D0)
C    &         /(exp(x(i)+4.0D0*(it+0.5D0)*dt-10.0D0)
C    &         +exp(-4.0D0*(it+0.5D0)*dt-
x(i)+10.0D0))

      sbexac(i,j)=2.0D0*sin(-20.0D0-2.0D0*x(i)-
2.0D0*y(j)
     &        +6.0D0*(it+0.5D0)*dt)
     &        /(exp(x(i)+y(j)-
8.0D0*(it+0.5D0)*dt+10.0D0)
     &        +exp(8.0D0*(it+0.5D0)*dt-x(i)-y(j)-
10.0D0))
C    &
+2.0D0*sin(2.0D0*x(i)+3.0D0*(it+0.5D0)*dt-
20.0D0)
C    &         /(exp(x(i)+4.0D0*(it+0.5D0)*dt-10.0D0)
C    &         +exp(-4.0D0*(it+0.5D0)*dt-
x(i)+10.0D0))
      enddo
      enddo
c*****************************
c calculate the error
      temp=0.0
```

```
      open(unit=23, file='2D_error_5.dat')
      do i=7,nx-7
      do j=7,ny-7
      temp1=(sqrt(rn1(i,j)*rn1(i,j)+sn1(i,j)*sn1(i,j))
     &          -
sqrt(rexac(i,j)*rexac(i,j)+sexac(i,j)*sexac(i,j)))
      temp=temp+temp1*temp1
      enddo
      enddo
      errnew=sqrt(temp/((nx-12)*(ny-12)))
      print *, it, errnew
      write(23,*)it*dt,errnew

c continue time step
      it=it+1
      if(it.eq.nt) goto 7
      do i=0,nx
      do j=0,ny
      rn(i,j)=rn1(i,j)
      sn(i,j)=sn1(i,j)
      rbn(i,j)=rbn1(i,j)
      sbn(i,j)=sbn1(i,j)
      enddo
      enddo
      goto 1

 7    close (23)

C      open(unit=04, file='trial_s_E_2D.dat')
C      do i=0,nx
C      do j=0,ny
C
write(04,10)i,j,sqrt(rexac(i,j)*rexac(i,j)+sexac(i,j)*sex
ac(i,j))
C      i*dx,sqrt(rn1(i)*rn1(i)+sn1(i)*sn1(i)),
C     &      sqrt(rexac(i)*rexac(i)+sexac(i)*sexac(i))
C      enddo
C      enddo
C      close (04)

C 10   FORMAT(I5,1x,I5,1x,F25.15)
      end
```

```
% The pseudo-spectral method for solving the NLS equation, written in MATLAB [45]
% iu_t-u_{xx}-2|u|^2u=0.

L=40; N=400; dt=0.0001; tmax=1; nmax=round(tmax/dt);
 dx=L/N; x=[-L/2:dx:L/2]'; k=[0:N/2 -N/2:-1]'*2*pi/L; k2=k.^2;
  u=sech(x+10.0-4*dt).*exp(-i*(2.0*x+20.0-3.0*dt));
  %  +sech(x-10.0+4*dt).*exp(i*(2.0*x-20.0+3.0*dt));
 udata=u; tdata=0; temp=0;
 fid = fopen('ps.dat', 'wt');
 for nn=1:nmax                          % integration begins
  du1=i*(ifft(k2.*fft(u))-2*u.*u.*conj(u));  v=u+0.5*du1*dt;
  du2=i*(ifft(k2.*fft(v))-2*v.*v.*conj(v));  v=u+0.5*du2*dt;
  du3=i*(ifft(k2.*fft(v))-2*v.*v.*conj(v));  v=u+   du3*dt;
  du4=i*(ifft(k2.*fft(v))-2*v.*v.*conj(v));
  u=u+(du1+2*du2+2*du3+du4)*dt/6;

  temp = (abs(u)-abs(sech(x+10.0-4*dt))).^2;
  q = [nn*dt; sqrt(max(temp)/N)];
  fprintf(fid, '%d %12.15d\n', q);

  if mod(nn,round(nmax/25)) == 0
    udata=[udata u]; tdata=[tdata nn*dt];
  end
 end                                    % integration ends

 fclose(fid);
 load ps.dat
 psx=ps(:,1);
 psy=ps(:,2);
 plot(psx,psy);

%  surf(x, tdata, abs(udata'));         % solution plotting
```

```
% The fourth-order split-step method for solving the NLS equation, written in MATLAB [45]
% iu_t-u_{xx}-2|u|^2u=0.

L=40; N=400; dt=0.0001; tmax=1; nmax=round(tmax/dt);
 dx=L/N; x=[-L/2:dx:L/2]'; k=[0:N/2 -N/2:-1]'*2*pi/L;
 u=sech(x+10.0-4*dt).*exp(-i*(2.0*x+20.0-3.0*dt));
  %  +sech(x-10.0+4*dt).*exp(i*(2.0*x-20.0+3.0*dt));
 udata=u; tdata=0; temp=0;
 c=1/(2-2^(1/3));                       % scheme coefficients
 a1=c/2; a2=(1-c)/2; a3=a2; a4=c/2;
 b1=c; b2=1-2*c; b3=c;
 E1=exp(a1*dt*i*k.^2);
 E2=exp(a2*dt*i*k.^2);
 E3=exp(a3*dt*i*k.^2);
 E4=exp(a4*dt*i*k.^2);
 fid = fopen('ss.dat', 'wt');
 for nn=1:nmax                          % integration begins
  v=ifft(fft(u).*E1);
   v=v.*exp(-b1*dt*i*2*v.*conj(v));
   v=ifft(fft(v).*E2);
   v=v.*exp(-b2*dt*i*2*v.*conj(v));
```

```
  v=ifft(fft(v).*E3);
  v=v.*exp(-b3*dt*i*2*v.*conj(v));
  u=ifft(fft(v).*E4);

  temp = (abs(u)-abs(sech(x+10.0-4*dt))).^2;
  q = [nn*dt; sqrt(max(temp)/N)];
  fprintf(fid, '%d %12.15d\n', q);

  if mod(nn,round(nmax/25)) == 0
    udata=[udata u]; tdata=[tdata nn*dt];
  end

 end                          % integration ends

 fclose(fid);
 load ss.dat
 ssx=ss(:,1);
 ssy=ss(:,2);
 plot(ssx,ssy);

 % surf(x, tdata, abs(udata'));   % solution plotting
```

```
% The 4th-order integrating-factor method for solving the NLS, written in MATLAB [45]
% iu_t-u_{xx}-2|u|^2u=0.

L=40; N=400; dt=0.0001; tmax=1; nmax=round(tmax/dt);
 dx=L/N; x=[-L/2:dx:L/2]';
 k=[0:N/2 -N/2:-1]'*2*pi/L; E=exp(-i*k.^2*dt/2); E2 = E.*E;
 u=sech(x+10.0-4*dt).*exp(-i*(2.0*x+20.0-3.0*dt));
  %   +sech(x-10.0+4*dt).*exp(i*(2.0*x-20.0+3.0*dt));
 udata=u; tdata=0; temp=0;
 fid = fopen('int.dat', 'wt');
 for nn = 1:nmax                    % integration begins
  v=fft(u);              dv1=2*i*    fft(-u.*u.*conj(u));
  w=ifft((v+dv1*dt/2)./E );  dv2=2*i*E.* fft(-w.*w.*conj(w));
  w=ifft((v+dv2*dt/2)./E );  dv3=2*i*E.* fft(-w.*w.*conj(w));
  w=ifft((v+dv3*dt  )./E2);  dv4=2*i*E2.*fft(-w.*w.*conj(w));
  v=v+(dv1+2*dv2+2*dv3+dv4)*dt/6;  u=ifft(v./E2);

  temp = (abs(u)-abs(sech(x+10.0-4*dt))).^2;
  q = [nn*dt; sqrt(max(temp)/N)];
  fprintf(fid, '%d %12.15d\n', q);

  if mod(nn,round(nmax/25)) == 0
    udata=[udata u]; tdata=[tdata nn*dt];
  end
 end                          % integration ends

 fclose(fid);
 load int.dat
 intx=int(:,1);
 inty=int(:,2);
 plot(intx,inty);

% surf(x, tdata, abs(udata'));    % solution plotting
```

REFERENCES

[1] G.P. Agrawal, Applications of Nonlinear Fiber Optics (Academic Press, Boston, 2001).

[2] G.D. Akrivis, V.A. Dougalis, O. Karakashian, "Solving the Systems of Equations Arising in the Discretization of Some Nonlinear PDEs by Implicit Runge–Kutta Methods," *Model. Math. Anal. Numer.* **31** (1997) pp. 251–287.

[3] G.D. Akrivis, "Finite Difference Discretization of the Cubic Schrödinger Equation," *IMA J. Numer. Anal.* **13** (1993) pp. 115–124.

[4] X. Antoine, A. Arnold, C. Besse, M. Ehrhardt, A. Schädle, "A Review of Transparent and Artificial Boundary Condition Techniques for Linear and Nonlinear Schrödinger Equations," *Commun. Comput. Phys.* **4** (2008) pp. 729–796.

[5] X. Antoine, C. Besse, P. Klein, "Absorbing Boundary Conditions for General Nonlinear Schrödinger Equations," *SIAM J. Sci. Comput.* **33** (2011) pp. 1008–1033.

[6] A. Arnold, M. Ehrhardt, I. Sofronov, "Discrete Transparent Boundary Conditions for the Schrödinger Equation: Fast Calculation, Approximation and Stability," *Commun. Math. Sci.* **1** (2003) pp. 501-556.

[7] A. Askar, A.S. Cakmak, "Explicit Integration Method for the Time-Dependent Schrödinger Equation," *J. Chem. Phys.* **68** (1978) pp. 2794-2798.

[8] W. Bao, Q. Tang, Z. Xu, "Numerical Methods and Comparison for Computing Dark and Bright Solitons in the Nonlinear Schrödinger Equation," *J. Comp. Phys.* **235** (2013) pp. 423 – 445.

[9] W. Bao, D. Jaksch, "An Explicit Unconditionally Stable Numerical Method for Solving Damped Nonlinear Schrödinger Equations with a Focusing Nonlinearity," *SIAM J. Numer. Anal.* **41** (2004) pp. 1406-1426.

[10] W. Bao, S. Jin, P. Markowich, "On Time-Splitting Spectral Approximations for the Schrödinger Equation in the Semiclassical Regime," *J. Comput. Phys.* **26** (2005) pp. 487-524.

[11] W. Bao, "Numerical Methods for the Nonlinear Schrödinger Equation with Nonzero Far-Field Conditions," *Methods Appl. Anal.* **11** (2004) pp. 367–387.

[12] W. Bao, D. Jaksch, P.A. Markowich, "Numerical Solution of the Gross–Pitaevskii Equation for Bose–Einstein Condensation," *J. Comput. Phys.* **187** (2003) pp. 318–342.

[13] W. Bao, S. Jin, P.A. Markowich, "On Time-Splitting Spectral Approximations for the Schrödinger Equation in the Semiclassical Regime," *J. Comput. Phys.* **175** (2002) pp. 487–524.

[14] C. Besse, B. Bidegaray, S. Descombes, "Order Estimates in Time of Splitting Methods for the Nonlinear Schrödinger Equation," *SIAM J. Numer. Anal.* **40** (2002) pp. 26–40.

[15] J.P. Boyd, Chebyshev and Fourier Spectral Methods, 2nd edition (Dover, Mineola, NY, 2001).

[16] S. Brandt and H. D. Dahmen, "The Picture Book of Quantum Mechanics," Springer Verlag, Berlin, 1995.

[17] A.G. Bratsos, "A Linearized Finite-Difference Scheme for the Numerical Solution of the Nonlinear Schrödinger Equation," *Korean J. Comput. Appl. Math.* **8** (2001) pp. 459-467.

[18] A.G. Bratsos, "On the Numerical Solution of the Nonlinear Cubic Schrödinger Equation," *Int. J. Pure Appl. Math. Sci.* **2** (2005) pp. 217-226.

[19] A.G. Bratsos, M. Ehrhardt, I. Th. Famelis, "A Discrete Adomain Decomposition Method for Discrete Nonlinear Schrödinger Equations," *Appl. Math. Comput.* **197** (2008) pp. 190-205.

[20] P.J. Bryant, "Nonlinear Wave Groups in Deep Water," *Stud. Appl. Math.* **61** (1979) pp. 1-30.

[21] N. Carjan, M. Rizea, S. Strottman, "Efficient Numerical Solution of the Time-Dependent Schrödinger Equation for Deep Tunneling," *Rom. Rep. Phys.* **55** (2003) pp. 555-579.

[22] T. F. Chan, D. Lee, L. Shen, "Stable Explicit Schemes for Equations of the Schrödinger Type," *SIAM J. Numer. Anal.* **23** (1986) pp. 274-281.

[23] Q. Chang, L. Xu, "A Numerical Method for a System of Generalized Nonlinear Schrödinger Equations," *J. Comput. Math.* **4(3)** (1986) 191-199.

[24] Q.S. Chang, E. Jia, W. Sun, "Difference Schemes for Solving the Generalized Nonlinear Schrödinger Equation," *J. Comput. Phys.* **148** (1999) pp. 397–415.

[25] H. Chen, B. D. Shizgal, "The Quadrature Discretization Method in the Solution of the Schrödinger Equation," *J. Chem.* **24** (1998) pp. 321-343.

[26] Z. Chen, J. Zhang, Z. Yu, "Solution of the Time-Dependent Schrödinger Equation with Absorbing Boundary Conditions," *J. Semicond.* **30** (2009) pp. 012001.

[27] M.A. Christou, "Numerical Investigation of the Nonlinear Schrödinger Equation with Saturation," *AIP Conf. Proc.* **1067** (2008) pp. 105-113.

[28] W. Dai, R. Nassar, "A Finite Difference Scheme for the Generalized Nonlinear Schrödinger Equation with Variable Coefficients," *J. Comput. Math.* **18** (2000) pp. 123-132.

[29] W. Dai, "An Unconditionally Stable Three-Level Explicit Difference Scheme for the Schrödinger Equation with A Variable Coefficient," *SIAM J. Numer. Anal.* **29** (1992) pp. 174-181.

[30] W. Dai, G. Li, R. Nassar, S. Su, "On the Stability of the FDTD Method for Solving A Time-Dependent Schrödinger Equation," *Numer. Methods. PDE.* **21** (2005) pp. 1140-1154.

[31] L.M. Degtyarev, V.V. Krylov, "A Method for the Numerical Solution of Problems of the Dynamics Wave Fields with Singularities," *USSR Comput. Math. Math. Phys.* **17(6)** (1977) pp. 172.

[32] M. Delfour, M. Fortin, G. Payre, "Finite-Difference Solutions of a Nonlinear Schrödinger Equation," *J. Comput. Phys.* **44** (1981) pp. 277–288.

[33] S. Descomb, M. Thalhammer, "An Exact Local Error Representation of Exponential Operator Splitting Methods for Evolutionary Problems and Applications to Linear Schrödinger Equations in the Semi-Classical Regime," *BIT Numer Math*, **50** (2010) pp. 729-749.

[34] T. Fevens, H. Jiang, "Absorbing Boundary Conditions for the Schrödinger equation," *SIAM J. Sci. Comput.* **21** (1999) pp. 255-282.

[35] J. Fleck, J.K. Morris, M.J. Feit, "Time-Dependent Propagation of High Energy

Laser Beams Through the Atmosphere," *Appl. Phys. A. Mater. Sci. Process.* **10** (1976) pp. 129-160.

[36] B. Fornberg, "A Practical Guide to Pseudospectral Methods," (Cambridge University Press, Cambridge, UK, 1998).

[37] B. Fornberg, G.B. Whitham, "A Numerical and Theoretical Study of Certain Nonlinear Wave Phenomena," *Philos. Trans. Roy. Soc. London Ser. A* **289** (1978) pp. 373-404.

[38] F. Fujiwara, "FDTD-Q Analysis of a Mott Insulator Quantum Phase," Ph.D. diss., University of Tokyo, Japan 2007.

[39] L. Gagon, P. Winterniz, "Lie Symmetries of a Generalized Nonlinear Schrödinger Equation. I. The Symmetry Group and its Subgroups," *J. Phys. A* **21** (1988) pp. 1493-1511.

[40] L.R.T. Gardner, G.A. Gardner, S.I. Zaki, Z. El Sahrawi, "B-Spline Finite Element Studies of the Nonlinear Schrödinger Equation," *Comput. Meths. Appl. Mech. Eng.* **108** (1993) pp. 303-318.

[41] D. Gottlieb, S.A. Orszag, "Numerical Analysis of Spectral Methods: Theory and Applications," (SIAM, Philadelphia, 1977).

[42] D.F. Griffiths, A.R. Mitchell, J.L. Morris, "A Numerical Study of the Nonlinear Schrödinger Equation," *Comput. Meth. Appl. Mech. Eng.* **45** (1984) pp. 177–215.

[43] D.J. Griffiths, "Introduction to Quantum Mechanics. Englewood Cliffs," Prentice-Hall, NJ, 1995.

[44] H. Han, J. Jin, X. Wu, "A Finite-Difference Method for the One Dimensional Time Dependent Schrödinger Equation on Unbounded Domain," *Comput. Math. Appl.* **50** (2005) pp. 1345-1362.

[45] H. Han, Z. Huang, "Exact Artificial Boundary Conditions for Schrödinger Equation in $\mathbb{R}^2$," *Commun. Math. Sci.* **2** (2004) pp. 79–94.

[46] R.H. Hardin, F.D. Tappert, "Applications of the Split-Step Fourier Method to the Numerical Solution of Nonlinear and Variable Coefficient Wave Equations," *SIAM Rev.* **15(423)** (1973) pp. 0-021.

[47] A. Hasegawa, F. Tappert, "Transmission of Stationary Nonlinear Optical Pulses in Dispersive Dielectric Fibers. I. Anomalous Dispersion," *Appl. Phys. Lett.* **23(3)** (1973) pp. 142-144.

[48] B.M. Herbst, J. LI. Morris, A.R. Mitchell, "Numerical Experience with the Nonlinear Schrödinger Equation," *J. Comput. Phys.* **60** (1985) pp. 282-305.

[49] M. Hochbruuck, C. Lubich, "On Magnus Integrators for Time-Dependent Schrödinger Equations," *SIAM J. Numer. Anal.* **41** (2003) pp. 945-963.

[50] F. Ivanauskas, M. Radziunas, "On Convergence and Stability of the Explicit Difference Method for Solution of Nonlinear Schrödinger Equations," *SIAM J. Numer. Anal.* **36(5)** (1999) pp. 1466-1481.

[51] B. Jazbi, M. Moini, "On the Numerical Solution of One Dimensional Schrödinger Equation with Boundary Conditions Involving Fractional Differential Operators," *IUST Int. J. Engin. Sci.* **19** (2008) pp. 21-26.

[52] O. Karakashian, G.D. Akrivis, V.A. Dougalis, "On Optimal Order Error Estimates for the Nonlinear Schrödinger Equation," *SIAM J. Numer. Anal.* **30** (1993) pp. 377–400.

[53] Z. Kalogiratou, Th. Monovasilis, T.E. Simos, "Asymptotically Symplectic Integrators of 3^{rd} and 4^{th} Order for the Numerical Solution of the Schrödinger Equation," *Comput. Fluid Solid Mech.* (2003) pp. 2012-2015.

[54] V.I. Karpman, E. M. Krushkal, "Modulated Waves in Nonlinear Dispersive Media," *Sov. Phys. JETP* **28** (1969) pp. 277.

[55] K.S. Kunz, R.J. Luebbers, "The Finite Difference Time Domain Method for Electromagnetics," CRC Press, Boca Raton, FL, 1993.

[56] A. Kurtinaitis, F. Ivanauskas, "Finite Difference Solution Methods for a System of the Nonlinear Schrödinger Equations," *Nonlin. Anal. Model. Cont.* **9(3)** (2004) pp. 247–258.

[57] J. P. Kuska, "Absorbing Boundary Conditions for the Schrödinger Equation On Finite Intervals," *Phys. Rev. B* **46** (1992) pp. 5000-5003.

[58] M. Lax, J.H. Batteh, G.P. Agrawal, "Channeling of Intense Electromagnetic Beams," *J. Appl. Phys.* **52(1)** (1981) pp. 109-125.

[59] K. Leung, B.D. Shizgal, H. Chen, "The Quadrature Discretization Method (QDM) in Comparison with Other Numerical Methods of Solution of the Fokker-Plank Equation for Electron Thermalization," *J. Math. Chem.* **24(4)** (1998) pp. 291-319.

[60] J. Lo, B.D. Shizgal, "Spectral Convergence of the Quadrature Discretization Method in the Solution of the Schrödinger and Fokker-Planck Equations:

Comparison with Sinc Methods," *J. Chem. Phys.* **125** (2006) pp. 194108.

[61] P.A. Milewski, E.G. Tabak, "A Pseudospectral Procedure for the Solution of Nonlinear Wave Equations with Examples from Free-Surface Flows," *SIAM J. Sci. Comput.* **21(3)** (2000) pp. 1102-1114.

[62] K.W. Morton, D.F. Mayers, "Numerical Solution of Partial Differential Equations," Cambridge University Press, London, 1994.

[63] F.I. Moxley III, T. Byrnes, F. Fujiwara, W. Dai, "A Generalized Finite-Difference Time-Domain Quantum Method for the N-Body Interacting Hamiltonian," *Comput. Phys. Commun.* **183** (2012) pp. 2434-2440.

[64] F.I. Moxley III, D.T. Chuss, W. Dai, "A Generalized Finite-Difference Time-Domain Scheme for Solving Nonlinear Schrödinger Equations," *Comput. Phys. Commun.* **184** (2013) pp. 1834-1841.

[65] F.I. Moxley III, F. Zhu, W. Dai, "A Generalized FDTD Method with Absorbing Boundary Condition for Solving a Time-Dependent Linear Schrödinger Equation," *Amer. J. Comput. Math.* **2** (2012) pp. 163-172.

[66] P. Muruganandam, S.K. Adhikari, "Fortran Programs for the Time-Dependent Gross–Pitaevskii Equation in a Fully Anisotropic Trap," *Comput. Phys. Commun.* **180** (2009) pp. 1888–1912.

[67] P.L. Nash, L.Y. Chen, "Efficient Finite Difference Solutions to the Time-Dependent Schrödinger Equation," *J. Comput. Phys.* **130** (1997) pp. 266-268.

[68] S.A. Orszag, "Accurate Solution of the Orr-Sommerfeld Stability Equation," *J. Fluid Mech.* **50(4)** (1971) pp. 689-703.

[69] J.M. Sanz-Serna, J.G. Verwer, "Conservative and Nonconservative Scheme for the Solution of the Nonlinear Schrödinger Equation," *IMA J. Numer. Anal.* **6(1)** (1986) pp. 25-42.

[70] J.M. Sanz-Serna, "Methods for the Numerical Solution of the Nonlinear Schrödinger Equation," *Math. Comp.* **43** (1984) pp. 21-27.

[71] J. Satsuma, N. Yajima, "B. Initial Value Problems of One-Dimensional Self-Modulation of Nonlinear Waves in Dispersive Media," *Prog. Theor. Phys. Supp.* **55** (1974) pp. 284-306.

[72] A.B. Shamardan, "The Numerical Treatment of the Nonlinear Schrödinger Equation," *Comput. Math. Appl.* **19** (1990) pp. 67–73.

[73] J. Shen, T. Tang, "Spectral and High-Order Methods with Applications," (Science Press, Beijing, 2006).

[74] T. Shibata, "Absorbing Boundary Conditions for Finite-Difference Time-Domain Calculation of the One-Dimensional Schrödinger Equation," *Phys. Rev. B* **42** (1991) pp. 6760-6763.

[75] B.D. Shizgal, H. Chen, "The Quadrature Discretization Method (QDM) in the Solution of the Schrödinger Equation with Nonclassical Basis Functions," *J. Chem. Phys.* **104(11)** (1996) pp. 4137-4150.

[76] W.J. Sonnier, C.I. Christov, "Strong Coupling of Schrödinger Equations: Conservative Scheme Approach," *Math. Comput. Simul.* **69(5)** (2005) pp. 514-525.

[77] A. Soriano, E.A. Navarro, J.A. Porti, V. Such, "Analysis of the Finite-Difference Time-Domain Technique to Solve the Schrödinger Equation for Quantum Devices," *J. Appl. Phys.* **95** (2004) pp. 8011-8018.

[78] G. Strang, "On the Construction and Comparison of Difference Schemes," *SIAM J. Numer. Anal.* **5(3)** (1968) pp. 506-517.

[79] M. Stricklans, D. Yager-Elorriaga, "A Parallel Algorithm for Solving the 3D Schrödinger Equation," *J. Comp. Phys.* **229(17)** (2010) pp. 6015-6026.

[80] D.M. Sullivan, "Electromagnetic Simulation Using The FDTD Method," (IEEE Press. New York, 2000).

[81] D.M. Sullivan, D.S. Citrin, "Time-domain Simulation of Two Electrons in A Quantum Dot," *J. Appl. Phys.* **89** (2001) pp. 3841-3846.

[82] D.M. Sullivan, D.S. Citrin, "Time-domain Simulation of A Universal Quantum Gate," *J. Appl. Phys.* **96** (2002) pp. 3219-3226.

[83] J. Szeftel, "Design of Absorbing Boundary Condition for Schrödinger Equations in $\mathbb{R}^d$," *SIAM J. Numer. Anal.* **42** (2004) pp. 1527-1551.

[84] A. Taflove, "Computational Electrodynamics: The Finite-Difference Time-Domain," (Artech House, Boston, 1995).

[85] T.R. Taha, M.J. Ablowitz, "Analytical and Numerical Aspects of Certain Nonlinear Evolution Equations: II. Numerical, Nonlinear Schrödinger Equation," *J. Comput. Phys.* **55** (1984) pp. 203–230.

[86] T.R. Taha, "A Numerical Scheme for the Nonlinear Schrödinger Equation," *Comput. Math. Appl.* **22** (1991) pp. 77-84.

[87] L.N. Trefethen, "Spectral Methods in MATLAB," (SIAM, Philadelphia, 2000).

[88] E.H. Twizell, A.G. Bratsos, J.C. Newby, "A Finite-Difference Method for Solving the Cubic Schrödinger Equation," *Math. Comput. Simulation* **43** (1997) pp. 67-75.

[89] T. Utsumi, T. Aoki, J. Koga, M. Yamagiwa, "Solutions of the 1D Coupled Nonlinear Schrödinger Equations by the CIP-BS Method," *Commun. Comput. Phys.* **1(2)** (2006) pp. 261-275.

[90] P.B. Visscher, "A Fast Explicit Algorithm for the Time-Dependent Schrödinger Equation," *Comput. Phys.* **5** (1991) pp. 596-598.

[91] D. Vudragovic, I. Vidanovic, A. Balaz, P. Muruganandam, S.K. Adhikari, "C Programs for Solving the Time-Dependent Gross–Pitaevskii Equation in a Fully Anisotropic Trap," *Comput. Phys. Commun.* **183** (2012) pp. 2021–2025.

[92] A.M. Wazwaz, "A Study on Linear and Nonlinear Schrödinger Equations by the Variational Iteration Method," *Chaos, Solitons & Fractals* **37(4)** (2008) pp. 1136-1142.

[93] J.A.C. Weideman, B.M. Herbst, "Split-Step Methods for the Solution of the Nonlinear Schrödinger Equation," *SIAM J. Numer. Anal.* **23** (1986) pp. 485–507.

[94] L. Wu, "Dufort-Frankel-Type Methods for Linear and Nonlinear Schrödinger Equations," *SIAM J Numer Anal.* **33** (1996) pp. 1526-1533.

[95] N. Yajima, M. Oikawa, J. Satsuma, C. Namba, "Modulated Langmiur Waves and Nonlinear Landau Damping," *Rep. Res. Inst. Appl. Mech. XXII* **70** (1975).

[96] N. Yajima, A. Outi, "A New Example of Stable Solitary Waves," *Prog. Theor. Phys.* **45(6)** (1971) pp. 1997-1998.

[97] J. Yang, "Nonlinear Waves in Integrable and Nonintegrable Systems," (SIAM, Philadelphia, 2010).

[98] N.N. Yanenko, "The Method of Fractional Steps," (Springer-Verlag, Berlin, 1971).

[99] H. Yoshida, "Construction of Higher Order Symplectic Integrators," *Phys. Lett. A* **150(5)** (1990) pp. 262-268.

[100] H.C. Yuen, W.E. Ferguson, "Relationship Between Benjamin-Feir Instability and Recurrence in the Nonlinear Schrödinger Equation," *Phys. Fluids* **21** (1978) pp. 1275.

[101] N.J. Zabusky, M.D. Kruskal, "Interaction of Solitons in a Collisionless Plasma and the Recurrence of Initial States," *Phys. Rev. Lett.* **15(6)** (1965) pp. 240-243.

[102] V.E. Zakharov, A.B. Shabat, "Exact Theory of Two-Dimensional Self-Focusing and One-Dimensional Self-Modulation of Waves in Nonlinear Media (Differential Equation Solution for Plane Self Focusing and One Dimensional Self Modulation of Waves Interacting in Nonlinear Media)," *Sov. Phys. JETP* **34** (1972) pp. 62-69.

[103] L. Zhang, "A High Accurate and Conservative Finite-Difference Scheme for Nonlinear Schrödinger Equation," *Acta Math. Appl. Sin.* **28** (2005) pp. 178-186.

[104] C. Zheng, "Exact Nonreflecting Boundary Conditions for One-Dimensional Cubic Nonlinear Schrödinger Equations," *J. Comput. Phys.* **215** (2006) pp. 552–562.

[105] G. Zouraris, "On the Convergence of a Linear Two-Step Finite Element Method for the Nonlinear Schrödinger Equation," ESAIM: *Math. Model. Numer. Anal.* **35(03)** (2001) pp. 389–405.