

SPLIT-STEP FOURIER METHOD FOR  
GENERALIZED NONLINEAR  
SCHRÖDINGER EQUATION

By

WEIMING ZHENG

Master of Science

Shanghai Institute of computer Technology

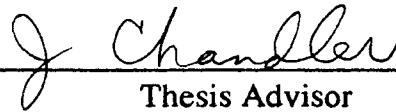
Shanghai, P.R.C

1986

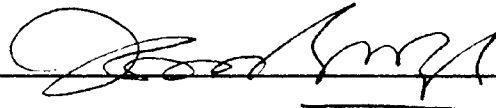
Submitted to the Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
MASTER OF SCIENCE  
December, 1994

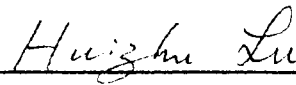
SPLIT-STEP FOURIER METHOD FOR  
GENERALIZED NONLINEAR  
SCHRÖDINGER EQUATION

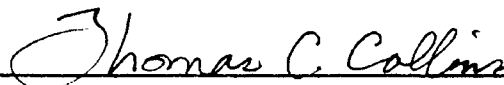
Thesis Approved:

  
\_\_\_\_\_

Thesis Advisor

  
\_\_\_\_\_

  
\_\_\_\_\_

  
\_\_\_\_\_

Dean of the Graduate College

## ACKNOWLEDGMENTS

I wish to express my gratitude to all the individuals who assisted me in this project and during my coursework at Oklahoma State University. In particular, I would like to thank my major advisor, Dr. J. P. Chandler, for his guidance and invaluable aid. Also, I wish to thank Dr. Lu and Dr. George for their very instructive courses which gave me the necessary knowledge to complete my research. Finally, I wish to express my appreciation to Dr. H. Burchard, Dr. J. S. Krasinski, and all other professors who taught me at Oklahoma State University. Without this background, this work would never have been completed.

## TABLE OF CONTENTS

Chapter	page
I. INTRODUCTION .....	1
II. NUMERICAL METHOD .....	4
III. FAST FOURIER TRANSFORM (FFT) .....	13
IV. ALGORITHM AND IMPLEMENTATION .....	17
V. NUMERICAL TEST .....	19
VI. SOLITON SOLUTIONS FOR GNLS EQUATION .....	25
VII. CONCLUSION .....	29
REFERENCES .....	31
APPENDIX A - PROGRAM TO SOLVE THE GNLS EQUATION .....	34
APPENDIX B - FIGURES OF OUTPUT OF SOLITON SOLUTIONS . . . . .	46

## LIST OF TABLES AND FIGURES

Table		page
I.	Output of associated liner equation . . . . .	20
II.	Output for plane wave solution with different $\Delta z$ . . . . .	21
III.	Output for plane wave solution with different N . . . . .	22
IV.	Output for plane wave solution with different $\Delta z$ . . . . . (enhanced algorithm )	23
V.	Output for plane wave solution with different N . . . . . (enhanced algorithm )	23
VI.	Output for plane wave solution with perturbation of periodicity . . . . .	24
VII.	Output for soliton solution . . . . .	26
VIII.	Output of invariant for soliton solution . . . . .	28
<b>Figures</b>		
I.	Hat function . . . . .	6
II.	Initial soliton plotted with length of the interval = $8\pi$ . . . . .	46
III.	Soliton with stable propagation . . . . .	47
IV.	Soliton with clearly visible deformation for big z . . . . .	48
V.	Initial soliton plotted with length of the interval = $20\pi$ . . . . .	49
VI.	Soliton with stable propagation ( $L = 20\pi$ ) . . . . .	50

## Chapter 1

### 1. Introduction

Nonlinear propagation techniques are now creating a revolution in telecommunications. Since Mollenauer et al. [22] demonstrated the propagation of solitons through a single-mode fiber, the potential application of optical solitons in the field of optical fiber communication has induced a large amount of theoretical and experimental work in this area.

A soliton is a kind of nonlinear wave. Usually, waves will disperse after a long distance propagation in the medium, but the nonlinear property of a medium can make waves become narrower and narrower. If these effects balance, these waves become solitons. So a soliton can move stably, and we can use this property in communication. Optical solitons are ideal carriers of information because they can improve the transmission rate of information.

As pulses get shorter and more intense, both dispersive and nonlinear effects become more important. The following generalized nonlinear Schrödinger equation including higher-order dispersion terms and higher-order nonlinear terms is suitable for description of pulses as short as 10 fs [12].

$$\frac{\partial A}{\partial z} + \frac{\alpha}{2}A + \frac{i}{2}\beta_2 \frac{\partial^2 A}{\partial T^2} - \frac{1}{6}\beta_3 \frac{\partial^3 A}{\partial T^3} = i\gamma[|A|^2 A + \frac{2i}{\omega_0} \frac{\partial |A|^2 A}{\partial T} - T_R A \frac{\partial |A|^2}{\partial T}] \quad (0.1)$$

where

$z$  is the  $z$  coordinate of an optical fiber,

$T = t - \beta_1 z$  and  $t$  is the time,

we use  $T$  instead of  $t$ , since in this way we can eliminate a term,

$A$  is the slowly varying amplitude of the wave,

$\gamma$  is the nonlinearity coefficient,

$\alpha$  is the absorption coefficient,

$$\beta_2 = \frac{d^2 \beta}{d\omega^2} \Big|_{\omega=\omega_0},$$

$$\beta_3 = \frac{d^3 \beta}{d\omega^3} \Big|_{\omega=\omega_0},$$

$\beta$  is the wavenumber,

$\omega_0$  is the carrier frequency,

$T_R$  is a parameter related to the slope of the Raman gain and it is estimated to be about 5 fs.

Since shorter pulses mean more information transferred on a single optical fiber, this will greatly reduce the cost of communication. On the other hand, "An understanding of the soliton behavior in the femtosecond regime is, however, far from complete." [12, pp. 43, pp. 142].

If we set the particular coefficient values in the above equation as  $\alpha = 0$ ,  $T_R = 0$ ,  $\beta_3 = 0$ ,  $\omega_0 = \infty$ , then the equation is the well-known nonlinear Schrödinger Equation (NLS)[12, 14].

The present thesis consists of the following chapters. In Chapter 1, the introduction, some background about the GNLS equation is presented. In Chapter 2, we introduce an overview of numerical methods, in particular, the spectral method

and split-step technique. In Chapter 3, we introduce the fast Fourier transform, which is important for the split-step Fourier method. In Chapter 4, we justify the split-step Fourier method for the GNLS and propose a numerical algorithm. In Chapter 5, we discuss the numerical results of our algorithm. In Chapter 6, we discuss soliton solutions, and finally in Chapter 7, we summarize our work and present the conclusions.



## Chapter 2

### Numerical Method

Roughly speaking, there are two numerical methods for solving initial value problems in partial differential equation ( the method of lines could belong to either of them ):

#### (1) Finite difference methods

Finite difference methods for solving initial boundary value problem determine approximation at finite number of points in the domain and involve four basic steps:

1. Subdivide the domain , for example by the uniform mesh,

$$x_{j+1} - x_j = \Delta x, j=1, 2, \dots N.$$

2. Replace derivatives by proper finite difference quotients to approximate the differential equation.
3. Impose the boundary and initial conditions on the system generated in step 2.
4. Solve the finite difference equations generated in steps 2 and 3 [34].

The basic idea is to replace a differential equation and auxiliary condition by a system of algebraic equations.

For example, if we apply finite difference methods to the GNLS Equation:

$$\frac{\partial u}{\partial t} + \frac{\alpha}{2}u + \frac{i}{2}\beta_2 \frac{\partial^2 u}{\partial x^2} - \frac{1}{6}\beta_3 \frac{\partial^3 u}{\partial x^3} = i\gamma[|u|^2 u + \frac{2i}{\omega_0} \frac{\partial |u|^2 u}{\partial x} - T_R u \frac{\partial |u|^2}{\partial x}] \quad (0.2)$$

$$u(x, 0) = f(x)$$

$$u(0, t) = u(2\pi, t)$$

then the above problem can be discretized as

$$\begin{aligned} & \frac{u_j^m - u_j^{m-1}}{\Delta t} + \frac{\alpha}{2} u_j^m + \frac{i}{2} \beta_2 \frac{u_{j+1}^m - 2u_j^m + u_{j-1}^m}{\Delta x^2} - \frac{1}{6} \beta_3 \frac{u_{j+2}^m - 3u_{j+1}^m + 3u_j^m - u_{j-1}^m}{\Delta x^3} \\ & = i\gamma [|u_j^m|^2 u_j^m + \frac{2i}{\omega_0} \frac{|u_{j+1}^m|^2 u_{j+1}^m - |u_{j-1}^m|^2 u_{j-1}^m}{2\Delta x} - T_R |u_j^m| \frac{|u_{j+1}^m|^2 - |u_{j-1}^m|^2}{2\Delta x}] \quad (0.3) \end{aligned}$$

$$u_j^0 = f(x_j)$$

$$u_0^m = u_N^m$$

(2) Function approximation methods.

Both spectral methods and finite element methods belong to this category, but finite element methods use local functions with fixed low degree to approximate unknown functions.

Suppose we are given the differential equation

$$\frac{\partial u}{\partial t} = Au \quad (0.4)$$

In the finite element method we determine an approximation of the form

$$u_N(x, t) = \sum_{n=1}^N a_n(t) \phi_n(x)$$

to the solution of the above equation. This method involves three steps:

1. Choose a finite dimension space  $S$ . For example, the space

spanned by  $\phi_i(x), i = 1, 2, \dots, N$ ,

where  $\phi_i(x)$  is hat function defined by

$$\phi_i(x) = \begin{cases} \frac{x-x_{i-1}}{\Delta x} & x_{i-1} \leq x \leq x_i \\ \frac{x_{i+1}-x}{\Delta x} & x_i \leq x \leq x_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

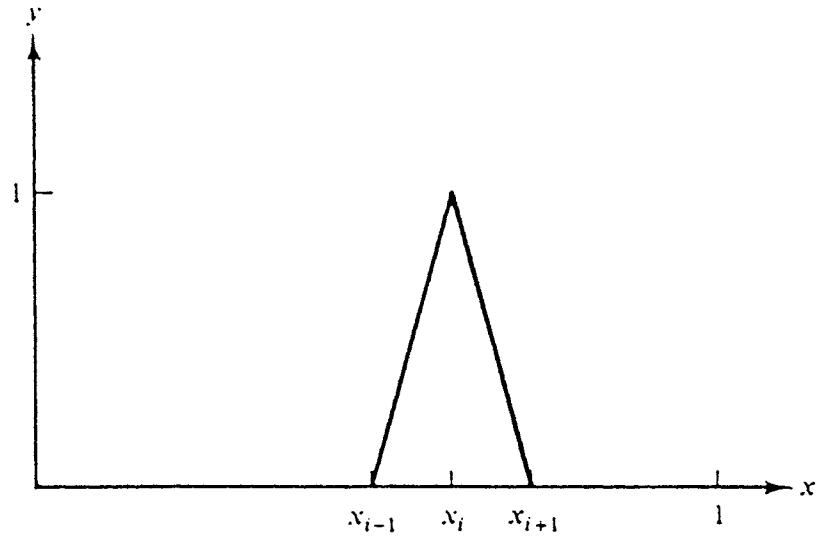


Figure 1. Hat functions are basis functions for finite element methods.

2. Approximate  $u(x,t)$  by

$$u_N(x,t) = \sum_{n=1}^N a_n(t) \phi_n(x)$$

Substitute  $u(x,t)$  into the equation

$$\frac{\partial u}{\partial t} = Au.$$

To minimize the the residual  $R = \frac{\partial u}{\partial t} - Au$

we choose  $\phi_i(x), i = 1, 2, \dots, N$  as weight functions.

Then we get  $(\phi_k, \frac{\partial u_N}{\partial t} - Au_N) = 0$

$k=1, 2, 3, \dots, N,$

or the Galerkin Equation:

$$\frac{\partial(\phi_n, u_N)}{\partial t} = (\phi_n, Au_N(x, t))$$

where  $(u, v)$  denotes the scalar product of the vectors  $u$  and  $v$ .

3. Solve the above system of ordinary differential equations subject to the initial condition.

The system of ordinary differential equations could be solved by single step (Runge-Kutta) or multistep methods.

We have the following mathematical framework for spectral methods [9]. Let  $H$  be a Hilbert space,  $u(x, t) \in H$ . Let  $B_N$  be the approximation space. Then we can write

$$u_N(x, t) = \sum_{n=1}^N a_n(t)\phi_n(x)$$

For the differential equation

$$\frac{\partial u}{\partial t} = Au \tag{0.5}$$

the semi-discrete spectral approximations have the form

$$\frac{\partial u_N}{\partial t} = P_N A(P_N u_N)$$

where  $P_n$  is a projection operator. We have various choices for  $B_N$  and  $P_N$ .

The different spectral methods and pseudospectral methods differ mainly in their way of minimizing the following residual function:

$$R = \frac{\partial u_N}{\partial t} - Au_N \tag{0.6}$$

Some important spectral methods are:

### 1. Galerkin approximation [9]

This method is very similar to finite element method, but here we use different base function and weight function. If we expand the solution in the form:

$$u_N(x, t) = \sum_{n=1}^N a_n \phi_n \quad (0.7)$$

Natural idea is to choose  $a_i$  to minimize

$$\left( \frac{\partial u_N}{\partial t} - Au_N, \frac{\partial u_N}{\partial t} - Au_N \right)$$

To do so, we only need [9]

$$P_N R = 0$$

That is,

$$\left( \phi_k, \frac{\partial u_N}{\partial t} - Au_N \right)$$

$k=1, 2, 3, \dots, N$ .

Equivalently, this method determines the expansion coefficient  $a_n$  by the Galerkin equation

$$\frac{\partial(\phi_n, u_N)}{\partial t} = (\phi_n, Au_N(x, t)) \quad (0.8)$$

$$n = 1, 2, 3, \dots, N$$

For the Galerkin method, we assume that all the expansion functions satisfy homogeneous boundary condition individually, but this is not required for the Tau approximation [9].

## 2. Tau method

The approximate solution is assumed to have the form

$$u_N(x, t) = \sum_{n=1}^{N+k} a_n \phi_n \quad (0.9)$$

$$n = 1, \dots, N$$

which satisfy the equation

$$\frac{da_n}{dt} = (\phi_n, Au_N(x, t)) \quad (0.10)$$

$$n = 1, 2, 3, \dots, N, \text{ and}$$

$$\sum_{n=1}^{N+k} a_n B \phi_n = 0 \quad (0.11)$$

Here we expand more terms than the expansion in the Galerkin method to give more freedom to satisfy boundary constraints.

## 3. Pseudospectral approximation [3,9]

Let  $D$  be the domain of a unknown function:  $\{x_i | i = 1, 2, \dots, N\} \subset D$ . These points are called collocation points. In this method we determine the expansion coefficients by the equation

$$u(x_i, t) = \sum_{n=1}^N a_n \phi_n(x_i) \quad (0.12)$$

$$i = 1, \dots, N.$$

If  $\phi_n(x) = e^{i2\pi nx}$ ,  $i = \sqrt{-1}$  and we let

$$\sum_{n=1}^N a_n \phi_n(x_i) = u(x_i)$$

then this method is called the Fourier pseudospectral method. We can prove that  $(a_1, a_2, \dots, a_n)$  is the DFT (discrete Fourier transform) of  $(u_1, u_2, \dots, u_n)$ . If  $A$  is linear, we can write equation (2) in the form:

$$\frac{\partial u_N}{\partial t} \approx F_N^{-1} A F[u_i]$$

where  $F$  is the DFT of  $(u_1, u_2, \dots, u_n)$

#### 4. Split-step Fourier method

Given the nonlinear equation

$$\frac{\partial u}{\partial t} = Au \tag{0.13}$$

where  $A$  is a nonlinear differential operator, the basic idea of the split-step method is to split the operator  $A$  into different pieces. For each of the pieces, if we have different schemes for updating the variable from timestep  $n$  to timestep  $n+1$ , then we let these pieces of the operator act separately.

For example, we may rewrite the above equation in the form:

$$\frac{\partial u}{\partial t} = (\hat{D} + \hat{N})u \tag{0.14}$$

where  $\hat{D}$  is a linear differential operator and  $\hat{N}$  is a nonlinear operator. The split-step Fourier method use the following scheme:

In the first step, let  $\hat{D} = 0$ , and we have

$$u_1(x, t + h) = \exp(h\hat{N})u(x, t) \tag{0.15}$$

In the second step, let  $\hat{N} = 0$ , and we have

$$u(x, t + h) = \exp(h\hat{D})u_1(x, t + h) \tag{0.16}$$

The linear step  $\hat{D}$  is implemented by the FFT (fast Fourier transform), that is,

$$\exp(h\hat{D})y(x, t) = \{F^{-1}\exp[h\hat{D}(i\omega)]F\}y(x, t)$$

where  $F$  denote the DFT and  $\hat{D}(i\omega)$  is obtained by replacing  $\frac{\partial}{\partial x}$  by  $i\omega$ .

For the NLS equation T. R. Taha and M. J. Ablowitz [2] compared seven numerical methods and concluded that the split method is the best method, followed by pseudospectral method. The reason why the split step spectral method is so fast may be explained by the following considerations.

(1) Generally, the Fourier expansions have exponential convergence, if the solution function is infinitely differentiable. But this is not always true; D. Gottlieb and S. A. Orszag [9] gave some examples showing spectral approximation with algebraic order for some mixed initial-boundary value problem. On the other hand, if a function has a discontinuity at  $x_0$ , then its Fourier expansion will not converge uniformly in the neighborhood of  $x_0$ . This nonuniform behavior of convergence is called the Gibbs phenomenon [9]. In other words, if  $f(x)$  has some discontinuity at  $x_0$ , its Fourier expansion may have bad accuracy near  $x_0$ .

(2) If the Fourier methods or Chebyshev methods [3,9] are used, then the fast Fourier transform can be used which also makes the spectral methods more efficient than finite difference methods.

Since the spectral method may get same accuracy for a relatively small number of grid points, it can also save memory space.

There are some drawbacks of spectral methods. Usually, spectral methods are



more difficult to program than finite difference methods. If the domains of the problems are irregular, spectral methods will lose accuracy and efficiency heavily [3]. For Fourier spectral methods, if the solution of the problem is not a periodic function, then the error will have a lower bound, that is, the solution has limited accuracy, even as  $\Delta z \rightarrow 0$ , and  $N \rightarrow \infty$ .

## Chapter 3

### Fast Fourier Transform (FFT)

To make this thesis self-contained, we introduce the fast Fourier transform (FFT) briefly.

Let  $x = (x_1, x_2, \dots, x_n)$  be a vector. Then the discrete Fourier transform (DFT) of  $x$  is defined by  $X = (X_1, X_2, \dots, X_n)$ ,

where

$$X_k = \sum_{j=0}^{n-1} x_j \omega^{jk}, \quad k = 0, 1, 2, \dots, n-1$$

$$\omega = \exp\{-i2\pi/n\}, \quad i = \sqrt{-1}$$

The DFT plays a key role in physics, because it can be used to describe the relation between the time domain and frequency domain. The DFT has also many applications in mathematics problems such as interpolation problems, and solving partial differential equations. The discrete Fourier transform of an  $n$ -vector can be computed in a straightforward way using  $n^2$  multiplication and fewer than  $n^2$  additions. An efficient method for computing the DFT (using  $O(n \log n)$  arithmetic operations) is called the fast Fourier transform or FFT [19,20].

Since Cooley and Tukey introduced the fast Fourier transform in 1965 [20], the use of the FFT method has increased in various areas. Some authors even claim that the FFT has changed the face of science and engineering so much so that life as we know it would be very different without the FFT [8].

Although the FFT algorithm has been known only since the mid-1960s, the

efficient method for computing the DFT had been independently discovered by as many as a dozen individuals, starting with Gauss in 1805 [21,24]. Danielson and Lanczos showed (1940s) that a DFT of length  $N$  can be rewritten as the sum of two DFTs each of length  $N/2$  [34]. It is wonderful that the D-L lemma can be used recursively. If  $N$  is a power of 2 then we can continuously use the D-L lemma until a transform of length 1 is reached that is the identity transform. (If possible we prefer  $n = 2^m$ .)

After Cooley and Tukey proposed the FFT, many developments have been made in the area. Various radices such as radix-2, radix-4, radix-8, mixed radix and split-radix transforms [23,33] have been considered. To improve the efficiency of the algorithm, many authors proposed various algorithms from different standpoints. Winograd and some others developed an algorithm that used only  $O(n)$  multiplications [27]. Some other researchers developed algorithms to reduce the numbers of both additions and multiplications.

Mathematically, there are at least two interpretations of FFT. First, we can let each FFT algorithm correspond to a factorization of the DFT matrix. In fact, the complex Fourier transform of a vector  $x$  can be expressed as a matrix [8] multiplication

$$X = Tx$$

where  $T$  is an  $n \times n$  matrix of complex exponentials with

$$t_{jk} = \exp(i2\pi jk/n)$$

In computing the fast Fourier transform, we factor  $T$  as

$$T = PF_m \dots F_2 F_1$$

where  $F_i$  is the matrix corresponding to the  $i$ -th transform step and  $P$  is the permutation matrix corresponding to bit-reversal, which is a symmetric permutation matrix. The permutation  $P$  is required because the transformed result is initially in bit-reversed order, i.e. the Fourier coefficient  $X_j$  with  $j = j_m 2^{m-1} + \dots + j_2 2 + j_1$  is found in location  $j'$ , where  $j'$  is the bit-reversal of  $j$  and  $j'$  is defined by  $j' = j_1 2^{m-1} + \dots + j_{m-1} 2 + j_m$ .

In other words, if we represent  $j$  in binary form by  $[j_1 j_2 \dots j_{m-1}]$  where  $j_i$  is 0 or 1, then the bit-reversal of  $j$  is  $[j_{m-1} \dots j_1]$ . i.e. the same bits in reverse order. The matrices  $F_i$  can be further factored to yield

$$F_i = R_i T_i$$

where  $R_i$  is a diagonal matrix of rotation factors called twiddle factors and  $T_i$  can be partitioned into  $n/2$  identical square submatrices, each a matrix of the complex Fourier transform of dimension  $n/2$  [8].

Another point of view of the FFT is that we can interpret the component of  $x = (a_0, a_1 \dots a_{n-1})$  as the coefficients of a polynomial [34]

$$P(x) = a_0 x^0 + a_1 x^1 + \dots a_{n-1} x^{n-1}$$

Let  $\omega$  be the  $n$ -th roots of 1; then computing the DFT of  $x$  is equivalent to evaluating the polynomial at  $\omega^0 \dots \omega^{n-1}$  i.e. at each of the  $n$ -th roots of unity. Based on

the same strategy of Divide and Conquer, we may use recursion to evaluate the polynomial  $p$  at  $n$  points. That is

$$P(x) = P_{\text{even}}(x^2) + xP_{\text{odd}}(x^2)$$

$$P(x) = P_{\text{even}}(x^2) - xP_{\text{odd}}(x^2)$$

It suffices to evaluate  $p_{\text{even}}$  and  $p_{\text{odd}}$  at  $1, \omega^2, \dots$ , then do  $n/2$  multiplications for  $xP_{\text{odd}}(x^2)$  and  $n$  additions and subtractions. The polynomials  $p_{\text{even}}$  and  $p_{\text{odd}}$  can be evaluated recursively by the same scheme. That is, they are polynomials of degree  $n/2-1$  and will be evaluated at the  $n/2$ th roots of unity:  $1, \omega, \dots, \omega^{n/2-1}$ . Clearly, when the polynomial to be evaluated is a constant, there is no work to be done, and hence we have finished the recursion.

$$\begin{aligned} \frac{\partial A}{\partial z} = & -\frac{\alpha}{2}A - \frac{i}{2}\beta_2 \frac{\partial^2 A}{\partial T^2} + \frac{1}{6}\beta_3 \frac{\partial^3 A}{\partial T^3} \\ & + i\gamma[|A|^2 A + (\frac{4i}{\omega_0} - T_R)|A|^2 \frac{\partial A}{\partial T} + (\frac{2i}{\omega_0} - T_R)A^2 \frac{\partial \bar{A}}{\partial T}] \end{aligned} \quad (0.17)$$

Let

$$\begin{aligned} N &= i\gamma|A|^2 \\ M &= i\gamma[(\frac{4i}{\omega_0} - T_R)\bar{A} \frac{\partial A}{\partial T} + (\frac{2i}{\omega_0} - T_R)A \frac{\partial \bar{A}}{\partial T}] \\ L &= -\frac{\alpha}{2} - \frac{i}{2}\beta_2 \frac{\partial^2}{\partial T^2} + \frac{1}{6}\beta_3 \frac{\partial^3}{\partial T^3} \end{aligned}$$

Then the algorithm is:

$$A_j^{(0)} = [\exp(N(z_m)\Delta z)] \cdot A_j^m \quad (0.18)$$

$$A_j^{(1)} = [\exp(M(z_m)\Delta z)] \cdot A_j^{(0)} \quad (0.19)$$

$$A_j^{m+1} = F_j^{-1}[\exp(\hat{L}\Delta z)] \cdot F[A_i^{(1)}] \quad (0.20)$$

where  $[A_i^{(1)}]$  is a vector, and  $F[A_i^{(1)}]$  is its DFT ( discrete Fourier transform),  $\hat{L}$  is obtained by replacing the differential operator  $\frac{\partial}{\partial T}$  in the operator by  $i\omega$ , where  $\omega$  is the frequency in the Fourier domain.  $[\exp(\hat{L}\Delta z)] \cdot F[A_i^{(1)}]$  is the inner product of the vector  $[\exp(\hat{L}\Delta z)]$  and the vector  $F[A_i^{(1)}]$ . F is implemented by the fast Fourier transform (FFT).

We implement the algorithm in a double precision FORTRAN code. To improve the speed of the computation, we use real arithmetic to implement complex operations. The FFT subroutine is obtained from the book "Numerical Recipes" [21]. The user may rewrite the subroutine for the initial condition to fit a special problem.

## Chapter 5

### Numerical tests

If we set  $\alpha = 0$ , then the GNLS equation has the progressive plane wave solution

$$A(z, T) = a * \exp(i(kz - sT))$$

If we substitute the above equation into the GNLS equation, we will get the following dispersion relation:

$$k = \beta_2 s^2 / 2 + \beta_3 s^3 / 6 + \gamma a_2 (1 + s) / \omega_0$$

The solution is quite simple , but it can be used to test the program.

To test the program, we set  $\beta_2 = 2$ ,  $\beta_3 = 0.1$ ,  $\gamma = 0$ ,  $\alpha = 0$  then the equation becomes:

$$\frac{\partial A}{\partial z} + i \frac{\partial^2 A}{\partial T^2} - \frac{1}{60} \frac{\partial^3 A}{\partial T^3} = 0. \quad (0.21)$$

This is a linear equation with dispersion relation:

$$k = s^2 + 0.1s^3/6$$

Table 1. If we set  $s = 6$  then the theoretical solution is  $A(z, T) = \exp(i(39.6z - 6T))$

We run the program with  $N = 128$ .

$\Delta z$	z-out	$L_\infty$ error
1.0000000000000000D-03	1.0000000000000000D-01	3.3938130084010D-14
1.0000000000000000D-03	1.0000000000000000D-02	2.9809488211185D-14
1.0000000000000000	1.0000000000000000	3.3084646133830D-14
0.5000000000000000	1.0000000000000000	3.3306690738755D-14
1.0000000000000000D-01	1.0000000000000000	3.5638159090468D-14
1.0000000000000000D-02	1.0000000000000000	3.8413716652030D-14
5.0000000000000000D-03	1.0000000000000000	4.8849813083507D-14
1.0000000000000000D-03	1.0000000000000000	1.0080825063596D-13

where z-out is the value of z at which the solution is desired.

$L_\infty$  is infinity norm of errors [39].

From Table 1, we can see that it does not make a big difference when the stepsize of z decreases. Actually, the errors in Table 1 are only roundoff errors due to finite precision; There is no truncation error. The reason could be explained by that

1. The pseudospectral method is exact at the collocation points.
2. "...time discretization errors in both difference and spectral methods are typically much smaller than are spatial discretization errors." [9]

The program works well for a linear problem.

Now, we consider the following mixed initial-boundary value problem with  $\beta_2$



$=2$ ,  $\beta_3 = 0.1$ ,  $\gamma = -2$ ,  $\alpha = 0$ ,  $T_R = 0.1$ ,  $\omega_0 = 100$ ,  $N = 128$ . That is, we include the nonlinear step for the GNLS equation.

$$\frac{\partial A}{\partial z} + i \frac{\partial^2 A}{\partial T^2} - \frac{1}{60} \frac{\partial^3 A}{\partial T^3} = -i2[|A|^2 A + \frac{i}{50} \frac{\partial |A|^2 A}{\partial T} - 0.1 \frac{\partial |A|^2}{\partial T}] \quad (0.22)$$

Table 2.

$\Delta z$	z-out	$L_\infty$ error	$L_\infty$ relative error
1.0000D-04	1.0000D-03	3.4523536330405D-06	9.2420709913220D-05
1.0000D-03	1.0000D-01	3.4547117312748D-04	5.1077108002296D-04
1.0000D-02	1.0000D-01	3.4547117313591D-04	5.1077108004455D-04
1.0000D-01	1.0000	3.4515901820111D-03	9.8970811276756D-03
5.0000D-02	1.0000	3.4515901820225D-03	9.8970811277047D-03
1.0000D-02	1.0000	3.4515901820309D-03	9.8970811277655D-03
1.0000D-03	1.0000	3.4515901821377D-03	9.8970811280756D-03

Table 3

N	$\Delta z$	z-out	$L_\infty$ error	$L_\infty$ relative error
256	1.0D-03	1.D-01	8.6649296361174D-05	1.2816912888162D-04
512	1.0D-03	1.D-01	2.1679952097020D-05	3.2072137772496D-05
1024	1.0D-03	1.D-01	5.4210902762615D-06	8.0199016464118D-06
1024	1.0D-02	1.0	2.2384277262317D-02	10.2103249330695
1024	1.0D-03	1.0	4.7118171342954D-02	7.0282392416296

where  $L_\infty$  relative error is the infinity norm of relative error [3,21].

Table 2, and Table 3 show that the errors are much bigger than the errors of the linear problem and the errors do not converge rapidly to zero as  $N$  increases, or as  $\Delta z$  decreases. Since the above equation is obtained by adding nonlinear steps to the linear equation, the bigger error must be caused by the nonlinear part. If we want to improve the accuracy efficiently, we only need to improve the accuracy of the nonlinear steps. If we examine the nonlinear part carefully, we can see that the derivative in the nonlinear operator is evaluated by a 3-point formula, and the error for this formula is second order, which is the bottle-neck in the accuracy of the overall algorithm.

Now we use a 5-point formula to evaluate the derivative in nonlinear steps, in which the error is fifth order.

Table 4. Progressive plane wave solution with  $\beta_2 = 2$ ,  $\beta_3 = 0.1$ ,  $\gamma = -2$ ,  $\alpha = 0$   
 $T_R = 0$ ,  $\omega_0 = 100$

$\Delta z$	z-out	$L_\infty$ error
1.00000000000000D-01	1.00000000000000	5.9519327331538D-05
1.00000000000000D-02	1.00000000000000	5.9519327315544D-05
5.00000000000000D-03	1.00000000000000	5.9519327334993D-05
1.00000000000000D-03	1.00000000000000	5.9519327400503D-05
1.00000000000000D-03	1.00000000000000D-01	5.9577604658101D-06
1.00000000000000D-04	1.00000000000000D-02	5.9567595906210D-07
1.00000000000000D-07	1.00000000000000D-05	5.9577526527639D-10

Table 5. Progressive plane wave solution with  $\beta_2 = 2, \beta_3 = 0.12, \gamma = -2, \alpha = 0$ 

$$T_R = 0, \omega_0 = 100$$

N	$\Delta z$	z-out	$L_\infty$ error
128	1.250000000000D-03	1.0000000000000000	5.9574326256490D-05
256	0.01	1.0	3.7523477387173D-06
512	0.01	1.0	2.3497629336039D-07
1024	0.01	1.0	1.4694140970184D-08

We have improved the accuracy dramatically. The results in Table 4 and Table 5 show that we have improved accuracy about 100 times over the one with 3 point formula. In another words, the errors have been reduced down to 1% of error of algorithm with 3-point formula.

Since we can get accuracy of more than 5 digits within 20 seconds when we solve the progressive plane wave solution, we did not count the CPU time. From table 5, we can find easily that the error will decrease as the number of grid points N increases, so we may improve accuracy further if we use a bigger N. But the split step Fourier method is very sensitive for the periodicity of boundary condition. If we disturb the periodicity by subtracting a small number from  $2\pi$ , we have the following results:

Table 6. Progressive plane wave solution with  $\beta_2 = 2, \beta_3 = 0.1, \gamma = -2, \alpha = 0$  $T_R = 0.1, \omega_0 = 100, N = 256, L = \text{length of the interval}$ 

L	$\Delta z$	z-out	$L_\infty$ error
$2\pi$	1.0000000D-02	1.00000	3.7525799286659D-06
$2\pi - 0.0001$	1.000000D-02	1.0000	0.35021404768869
$2\pi - 0.000001$	1.0000D-02	1.0000	3.2529175756306D-03
$2\pi$	1.0D-03	1.0000000000000D-01	3.7525553560087D-07
$2\pi - 0.000001$	1.000D-03	1.00000D-01	7.3680337973683D-06

So if the periodicity is not satisfied and  $z$  is big enough, then accuracy could be bad. Since the basis functions in the Fourier method are periodic and approximate solutions are also periodic, the approximate solution can not converge to a solution without periodicity. We got very good accuracy for the progressive plane wave solution since it is periodic.

## Chapter 6

### Soliton Solutions for GNLS the Equation

Generally, the GNLS equation has no soliton solutions, but for some special coefficients, the GNLS equation has a soliton solution, or exact solution. For example, the NLS equation is the special case of the GNLS equation and it has envelope soliton solutions. There are various methods to get the soliton solutions for a soliton equation such as the inverse scattering method, Hirota method, prolongation method, Bäcklund transform method [40], trace method [14,16,17] and so on. I got N-soliton solutions of NLS equation by the trace methods and proved that the N-soliton solutions obtained by inverse scattering, Hirota method, and trace method are equivalent [14].

If we set the coefficient of the higher-order nonlinear term and higher-order disperse terms very small, then we can consider the GNLS as a perturbed NLS equation.

We set the initial condition (mathematically, for  $z = 0$ ) of the GNLS equation as follows:

$$A(T, 0) = \operatorname{sech}(T)$$

Then we have output as follows:

Table 7. Soliton solution with  $\beta_2 = 0.02$ ,  $\beta_3 = 0.0012$ ,  $\gamma = -0.02$ ,  $\alpha = 0$ ,  
 $T_R = 0$ ,  $\omega_0 = 100$

N	$L_\infty$ error	$\Delta z$	z-out
64	2.5443370713321D-06	1.0000000000D-02	1.00000000000000
128	2.5244348991198D-06	1.0D-02	1.00000000000000
256	2.5147018525744D-06	1.0D-02	1.00000000000000
512	2.5098352640913D-06	1.0D-02	1.00000000000000
1024	2.5074020049061D-06	1.0D-02	1.00000000000000
2048	2.5061853653815D-06	1.0D-02	1.00000000000000
512	3.9253598949636D-04	1.00000000000000D-02	10.00000000000000
512	3.9254346963622D-04	1.00000000000000D-01	10.00000000000000
512	3.9261830140740D-04	1.00000000000000	10.00000000000000
512	3.9270151896993D-04	2.00000000000000	10.00000000000000

From the Table 7, we find that  $L_\infty$  decreases very slowly as  $\Delta z$  decreases or as N increases. This can be explained by that the major part of the errors is caused by nonperiodicity of soliton solution, which does not depend on  $\Delta z$  or N and it is related z-out.

We plot numerical output in Figure 2, 3, 4, 5, 6.

Figure 2 is an initial soliton with  $A(T) = \text{sech}(T)$  and the number of grid points  $N = 256$ , length =  $8\pi$ .

Figure 3 is the output of the GNLS equation for above initial soliton with  $\beta_2 = 0.02$ ,  $\beta_3 = 0.0012$ ,  $\gamma = -0.02$ ,  $\alpha = 0$ ,  $T_R = 0$ ,  $\omega_0 = 100$ ,  $N = 256$ , length =  $8\pi$ , z

= 100. The figure shows that the soliton propagates stably.

Figure 4 is the output of the GNLS equation for the same initial soliton as above with  $\beta_2 = 0.02$ ,  $\beta_3 = 0.0012$ ,  $\gamma = -0.02$ ,  $\alpha = 0$ ,  $T_R = 0$ ,  $\omega_0 = 100$ ,  $N = 256$ ,  $\text{length} = 8\pi$ ,  $z = 300$ .

The figure shows that the soliton has clearly visible deformation, if  $z$  is big.

Figure 5 is initial soliton with  $A(T) = \text{sech}(T)$  and the number of grid points  $N = 256$ ,  $\text{length} = 20\pi$

Figure 6 is output of the GNLS equation for initial soliton in Figure 5 with  $\beta_2 = 0.02$ ,  $\beta_3 = 0.0012$ ,  $\gamma = -0.02$ ,  $\alpha = 0$ ,  $T_R = 0$ ,  $\omega_0 = 100$ ,  $N = 256$ ,  $\text{length} = 20\pi$ ,  $z = 60$ . The figure shows that the soliton propagates stably.

From these figures, we find that if  $z$  is not very big, then the errors are small.

Generally, a soliton equation has an infinite number of conserved quantities. The conserved quantities are not only useful to keep track of the numerical calculation, but also very useful as mathematical tool to prove the existence and uniqueness of the solution.

We can prove that

$$E = \int_{-\infty}^{\infty} |A|^2 dT = \text{constant}$$

is an invariant.

Table 8. shows an invariant of the soliton solution with  $\beta_2 = 0.02$ ,  $\beta_3 = 0.0012$ ,  
 $\gamma = -0.02$ ,  $\alpha = 0$ ,  $T_R = 0$ ,  $\omega_0 = 100$ ,  $N = 128$

$\Delta z$	z-out	invariant E
1.0000000000D-01	50.0000000000	40.591701777239
1.0000000000D-01	10.0000000000	40.591663813317
1.0000000000D-01	5.0000000000	40.591659328647
1.0000000000D-02	1.0000000D-01	40.591654865618
1.0000000000D-01	1.0000000000	40.591655750701

From the output of our program, we can observe the conservation of the invariant for the soliton solution.



## Chapter 7

### Conclusions

We have presented a split step Fourier method for solving the GNLS equation. According to numerical test, we have the following conclusions:

1. When we use the split step Fourier method to solve the GNLS equation, if solution is a periodic function, then errors mainly are caused by nonlinear steps. To improve the accuracy, we should concentrate on nonlinear steps. By using a more precise formula to approximate derivatives, we have improved the accuracy dramatically.

2. Errors increase as  $z$  increases and decrease as  $N$  ( the number of grid points for  $T$ ) increases.

3. The split step method is very sensitive to the periodic boundary condition. If the periodicity is not satisfied and  $z$  is big enough, then we will lose accuracy heavily.

4. If the coefficients of the GNLS equation are not big or  $z$  is not big, then we can get good accuracy for the soliton solution, and the soliton solution can keep the invariant constant quite well. If  $z$  is big, we have only limited accuracy for soliton solution due to nonperiodicity of the solutions, even as  $\Delta z \rightarrow 0$ , and  $N \rightarrow \infty$ . This is a major disadvantage of Fourier spectral methods.

Since the GNLS equation was deduced in 1987 [41], it is quite new and it

has direct physical application. There is a lots of work we can do, such as exact solution under some restrictions, the existence and uniqueness of solutions for the GNLS equation, stability and error estimation of spectral methods, analytic methods under some restrictions, other numerical methods, and so on. But these are beyond the scope of this thesis.

## REFERENCES

1. Bertrand Mercier, An Introduction to the Numerical Analysis of Spectral Methods, Springer-Verlag (1989)
2. Thiab R. Taha and Mark J. Ablowitz, Analytical and Numerical Aspects of Certain Nonlinear Schrödinger Equations. *Journal of Computational Physics*, 55 (1984) pp. 231-253
3. John Philip Boyd, Chebyshev and Fourier Spectral Methods, Springer-Verlag (1989)
4. D. Pathria and J. L. Morris, Pseudo-spectral Solution of Nonlinear Schrödinger Equations, *Journal of Computational Physics*, 87 (1990) pp. 108-125.
5. Wilhelm Heinrichs, Splitting Techniques for the Pseudospectral Approximation of the Unsteady Stokes Equations, *SIAM J. Numer. Anal.* 30 (1993) pp. 19-39.
6. Christophe Devulder and Martine Marion, A Class of Numerical Algorithms For Large Time Integration: the Nonlinear Galerkin Method. *SIAM J. Numer. Anal.* 29 (1992) pp. 462-483
7. Govind P. Agrawal and M. J. Potasek, Nonlinear Pulse Distortion in Single-mode Optical Fiber at the Zero-dispersion Wavelength. *Physical Review A*. 33, (March, 1986) pp. 1765-1766.
8. Charles Van Loan, Computational Frameworks for the Fast Fourier Transform, SIAM (1992)
9. David Gottlieb and Steven A. Orszag, Numerical Analysis of Spectral Methods: Theory and Applications, SIAM (1977)
10. P. Dutt, Stability and Convergence of Spectral Method for Hyperbolic Initial-Boundary Value Problems, *Mathematics of Computation* 53 (1989) pp. 547-561
11. J. A. C. Weideman and A. Clout, Spectral Methods and Mappings for Evolution Equations on Infinite Line, *Spectral and High Order Method for P.D.E.*, Icosom'89 Conference, North-Holland (1990)
12. Govind P. Agrawal, Nonlinear Fiber Optics, Academic Press (1989)

13. E. O. Brigham, *The Fast Fourier Transformation and Its Application*, Prentice-Hall, Englewood Cliffs, N.J. (1988)
14. Weiming Zheng, *Trace Method and Some Nonlinear Evolution Equations*, *Appl. Math. and Compu. Math.* Vol. 5, No. 2 Oct. (1991) 1-11
15. R. Tolimieri, Man and C. Lu, *Algorithm for Discrete Fourier Transform and Convolution*, Springer-Verlag, New York, (1989)
16. Weiming Zheng, *The Sine-Gordon Equation and The Trace Method*, *J. Phys. A: Math Gen.* 19 (1986) L485
17. Weiming Zheng, *N-soliton Solution of the G. Z. Tu Equation*, *Science Bulletin* Vol. 32, No. 3 Feb. (1987) 210
18. Cheng Ansheng, Weiming Zheng, Yu Guai Ping, *Some New Feature of Optical Soliton Communication*, *Optical Fiber and Electric Cable*, April (1990)
19. E. Linzer and E. Feig, *Implementation of efficient FFT algorithm on fused multiply/add architectures*, Technical Report RC17330 IBM Research, Yorktown Heights, NY, 1990; *IEEE Trans. Signal Process.*
20. J. W. Cooley and J. W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, *Math.Comput.*, Vol. 19, pp. 297-301, Apr. (1965)
21. W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes*, Cambridge (1992)
22. L. F. Mollenauer, R. H. Stolen, and J. P. Gordon, *Expeimental observation of picosecond pulse narrowing and soliton effect in single-mode optical fibers*, *Opt. Lett.*, Vol. 8 pp. 289, (1983)
23. Henrick V. Sorensen, Michael T. Heideman and Sidney Burrus, *On Computing the Split-Radix FFT*, *IEEE Trans. Acoust. Speech Signal Process* (1986) pp.152-156
24. M. T. Heideman, D. H. Johnson, and C. S. Burrus, *Gauss and the history of the FFT*, *IEEE Acoust., Speech, Signal Processing, Mag.*, Vol. 1, Oct. (1984) pp. 14-21
25. L. R. Rabiner and C. M. Rader, Eds., *Digital Signal Processing, Selected Reprints*. New York: IEEE Press (1972)
26. R. Yavne, *An economical method for calculating the discrete Fourier transform*, in *Proc. Fall Joint Comput. Conf.*

- (1968) pp.115-125.
27. S. Winograd, On computing the discrete Fourier transform, *Math. Comput.*, Vol. 32, (1978) pp. 175-199
  28. S. Winograd, On the multiplicative complexity of the discrete Fourier transform, *Adv. Math.*, Vol.32, May (1979) pp. 83-117
  29. S. Winograd, *Arithmetic Complexity of Computation*, SIAM CBMS-NSF Series, No. 33 Philadelphia PA: SIAM, (1980)
  30. T. Heideman and C. S. Burrus, Multiply/add tradeoffs in length-2n FFT algorithms, in *Proc. IEEE Int. Conf. ASSP*, Tampa, FL, Apr. (1985)
  31. T. Heideman and C. S. Burrus, On the number of multiplications necessary to compute a length-2n DFT, *IEEE Trans. Acoust., Speech, Signal Processing*, Vol 34 No. 1 February (1986) pp. 91-95.
  32. P. Duhamel and H. Hollmann, Existence of a  $2^n$  FFT algorithm with a number of multiplications lower than  $2^{n+1}$ , *Electron. Lett.*, Vol. 20, Aug. 16 (1984) pp. 690-692
  33. P. Duhamel and H. Hollmann, Split radix FFT algorithm, *Electron. Lett.*, Vol. 20, Jan. 5, (1984) pp. 14-16
  34. Sara Baase, Computer Algorithms: Introduction to Design and Analysis, Addison-Wesley Publishing Company (1978)
  35. J. B. Martens, Recursive cyclotomic factorization-A new algorithm for calculating the discrete Fourier transform, *IEEE Trans. Acoust., Speech, Signal processing*, Vol. ASSP-32, Aug. (1984) pp. 750-761
  36. M. Vetterli and H. J. Nussbaumer, Simple FFT and DCT algorithms with reduced number of operations, *Signal Processing*, Vol. 6, Aug. (1984) pp. 267-278
  37. C. S. Burrus and T. W. Parks, *DFT/FFT and Convolution Algorithms*. New York: Wiley (1984)
  38. H. W. Johnson and C. S. Burrus, Twiddle factors in the radix-2 FFT, in *Proc. 1982 Asilomar conf. Circuits, Syst., Comput.*, Nov. 1982, pp.413-416.
  39. C. A. Hall and T. A. Porsching, Numerical Analysis of Partial Differential equations, Prentice Hall, Englewood Cliffs, N.J. (1990)
  40. M. J. Ablowitz and H. Segur, Solitons and the Inverse Scattering Transform,

SIAM (1981)

41. Y. Kodama and A. Hasegawa, Nonlinear Pulse Propagation in a Monomode Dielectric Guide, *IEEE Journal of Quantum Electronics*, Vol. Qe-23, No. 5, May (1987)

## APPENDIX A

FORTRAN PROGRAM TO SOLVE  
THE GNLS EQUATION

```

C*****C
C Ref: Thesis W. Zheng C
C This program solves the generalized C
C nonlinear Schödinger equation C
C by the split step Fourier method. C
C  $D(A)/DZ + \alpha*A/2+i*\beta_2*DDA/DTT/2-$  C
C  $\beta_3*DDDA/DTTT = i*\gamma*(|A||A +$  C
C  $2i* d(|A||A)/DT /\omega_0-Tr*A*D|A|^2/dt)$  C
C under periodic boundary condition: C
C  $A(z,-\pi) = A(z,\pi)$  C
C and initial condition C
C  $A(0,T) = g(T)$  C
C Author: Weiming Zheng C
C Adviser: Dr. J. P. Chandler C
C Computer Science Department C
C Oklahoma State University C
C Aug. 2, 1994 C
C*****C
C
C Usage: To fit a special application, the user may rewrite
C the subroutine for the initial condition, and call that
C subroutine.
C To apply the program to an interval other than
C  $(-\pi,\pi)$ , the user should set the length which is the
C interval for T.
C
C Parameters and Arguments:
C NN -- The number of grid points for the variable T.
C dz -- The step size for the variable z.
C L -- The number of steps for the variable z
C ds -- The step size for the variable T.
C X -- The array of values for the variable A at all grid points.
C It sets the initial value of A (at  $z = 0$ ). And finally, X is
C the solution at  $z = zout$ .
C Y -- The exact solution components at  $z = zout$ .
C zout -- This is a point at which a solution is desired.
C DX --- The numerical derivative of A with respect to T.
C gamma, beta2, beta3, alpha, omega0, Tr -- Coefficients of the

```

```

C   GNLS p.d.e.
C   Er -- The error in the solution = max|Y(i)-X(i)|
C   REr -- The relative error = max| (Y(i)-X(i))/X(i)|
C   E --- invariant of the GNLS equation
C -----
C
C   This program solves the generalized NLS.
PROGRAM gnlsps3
implicit none
INTEGER NN,NN2,j,L,model
real*8 PI,h,lenth
PARAMETER ( PI = 3.14159265358979d0)
PARAMETER ( NN=512,NN2=2*NN)
PARAMETER ( outpt = 10)
C   lenth is the length of the interval of T
PARAMETER ( lenth = PI*2d0)
C   PARAMETER ( lenth = 20d0)
CHARACTER flg
DOUBLE PRECISION X(NN2),DX(NN2),Y(NN2)
real*8 c,ds,dz,b,Tr,w
real*8 end,gama, omega0, beta2,beta3,alpha
b= 2*PI/lenth
open( UNIT= 10, FILE = 'Error', status='UNKOWN')
c=-PI
C   tm = time
1 write(*,*)'# Enter the zout '
read(*,*) end
write(*,*)'# Enter the dz '
read(*,*) dz
L = end /dz
ds = 2d0*PI/NN
C***** Sets the coefficient of the pde *****
gama = -2D0
gama = -2D-2
beta2 = 2D0
beta2 = 2D-2
alpha = 0D0
beta3 = 1.2D-1
beta3 = 1.2D-3
TR = 0d0
omega0 = 1D2
C***** Initialization of X *****
call inislt(X,NN,ds,gama,beta2,beta3,omega0,b)
w = 6d0
call iniplw(X,NN,b,ds,w)

```



```

C***** Compute the solution *****
  beta2 = beta2*b*b
  beta3 = beta3*b*b*b
  omega0 = omega0/b
  TR = TR*b
  do 50 j=1,L,1
    call liner(X,NN,ds,dz, alpha, beta2, beta3)
    call nolin(X,DX,NN,ds,dz,gama,omega0,TR)
C   call liner(X,NN,ds,dz/2., alpha, beta2, beta3)
  50 CONTINUE
C   *****Compute the exact solution *****
C   model = 1 for soliton solution, model = 2 for plane wave solution
  model = 2
  call extslt(Y,NN,ds,gama,beta2,beta3,omega0,b,model,end,model,w)
C   ***** Output*****
  call output(X,Y,NN,model)
  WRITE(outpt,*) 'dz=', dz
  WRITE(outpt,*) 'zout =',end
  WRITE(*,*) 'Would you like another test? y/n'
  read(*,*) flg
  if (flg .EQ. 'y') then
    goto 1
  end if
  close(outpt)
  END

C
  subroutine output(X,Y,NN,model)
C-----
C   SUBROUTINE OUTPUT (X,Y,NN,model)
C   This subroutine outputs the results for GNLS PDE.
C   NN -- The number of grid points for the variable T.
C   X -- The array of approximate solution values at all grid points.
C   Y -- The array of exact solution values at all grid points.
C   model -- model=1 for soliton solution,
C           model=2 for plane wave solution
C-----
  implicit NONE
  integer i,NN,model
C   lowbd is a small number to check errors and to avoid underflow.
  PARAMETER ( lowbd = 1d-30)
C   output is parameter of unit when open a file
  PARAMETER ( outpt = 10)
  DOUBLE PRECISION X, Y,Er, REr,sum, tmp,E
  DOUBLE PRECISION tmp1,tmp2
  dimension X(*),Y(*)

```

```

backspace (10)
C   *** Output soliton solution ****
   if ( model .eq. 1) then
   WRITE(*,*) ' approximate          exact'
   WRITE(*,*) ' solution            solution'
   do 51 i=1,2*NN-1,2
   tmp1 = sqrt((X(i))**2 + (X(i+1))**2 )
   tmp2 = sqrt((Y(i))**2 + (Y(i+1))**2 )
   WRITE(*,*) tmp1, tmp2
C   ***** For plot only *****
C   WRITE(outpt,*) '# soliton solution Zout= ', zout
C   WRITE(outpt,*) i/2, tmp1
51  CONTINUE
   Er= 0d0
   REr = 0d0
   do 55 i=1,2*NN
   if(abs(tmp1-tmp2) .GT. Er) then
   Er = abs(tmp1-tmp2)
   end if
   if( abs(tmp1) .GT. lowbd) then
   if(abs((tmp1-tmp2)/tmp1) .GT. REr) then
   REr = abs(tmp1-tmp2)/abs(tmp1)
   end if
   end if
55  CONTINUE
   end if
C   *** Output plane wave solution ****
   if ( model .eq. 2) then
   WRITE(*,*) ' approximate          exact'
   WRITE(*,*) ' solution            solution'
   do 61 i=1,2*NN-1,2
   WRITE(*,*)(i+1)/2, X(i), X(i+1), Y(i), Y(i+1)
61  CONTINUE
   Er= 0d0
   REr = 0d0
   do 65 i=1,2*NN
   if(abs(X(i)-Y(i)) .GT. Er) then
   Er = abs(X(i)-Y(i))
   end if
   if( abs(X(i)) .GT. 1d-30) then
   if(abs(X(i)-Y(i)/X(i)) .GT. REr) then
   REr = abs((X(i)-Y(i))/X(i))
   end if
   end if
65  CONTINUE

```

```

do 12 i=1,NN,1
  s = (-PI + i*ds)/b
  X(2*i-1) = DCOS(d1*s)/DCOSH(c1*s)
  X(2*i) = DSIN(d1*s)/DCOSH(c1*s)
C   X(2*i-1) = DCOS(d1*(a+ds*I))/DCOSH(c1*(a+I*ds))
C   X(2*i) = DSIN(d1*(a+ds*I))/DCOSH(c1*(a+I*ds))
12 CONTINUE
  return
end

C
C
  subroutine iniplw(X,NN,b,ds,w)
C -----
C   subroutine iniplw(X,NN,b,ds,w)
C   This subroutine sets the initial condition for PDE
C   with progressive plane waves.
C   NN -- The number of grid points for the variable T.
C   ds -- The step size for the variable T.
C   X -- The array of values for the variable A at all grid points.
C   It sets the initial value of A (at z = 0).
C   b = 2*pi /lenth
C   w -- A parameter in dispersion relation.
C -----
  implicit NONE
  integer i,NN
  DOUBLE PRECISION PI
  PARAMETER ( PI = 3.14159265358979d0)
C   DOUBLE PRECISION X(NN*2),a,ds
  DOUBLE PRECISION X,b,ds,w,s
  dimension X(*)
  do 11 i=1,NN,1
    s = (-PI + i*ds)/b
    X(2*i-1) = DCOS( -w*s)
    X(2*i) = DSIN( -w*s)
11 CONTINUE
  RETURN
  END

C
C
  subroutine extslt(Y,NN,ds,gama,beta2,beta3,omega0,b,model,zend)
C -----
C   SUBROUTINE EXTSLT(X,NN,ds,gama,beta2,beta3,omega0,b,model,zend)
C   This subroutine computes the exact solution for GNLS PDE.
C   NN -- The number of grid points for variable T.
C   ds -- The step size for the variable T.

```

```

C   X -- The array of value for variable A at all grid points.
C   gama, beta2, beta3, alpha, omega0, Tr -- Coefficients of the
C   GNLS p.d.e.
C   end -- this is the value of z at which a solution desired.
C-----
C   This subroutine sets the exact solution for GNLS PDE.
implicit NONE
DOUBLE PRECISION PI,gama,beta2,beta3,omega0
integer i,NN, model
DOUBLE PRECISION Y,ds,c1,d1,b,s,c2,d2, zend,f,w
dimension Y(*)
PARAMETER ( PI = 3.14159265358979d0)
c1 = SQRT(-6d0*gama/beta3/omega0)
c   c1=1d0
d1 = omega0*beta3*c1*c1+2d0*gama
if( d1 .NE. 0d0) then
  d1 = omega0*(gama+beta2*c1*c1)/d1
end if
d2 = -beta2*(c1*c1-d1*d1)/2d0-beta3*(d1*d1*d1-3*c1*c1*d1)/6d0
c2 = beta2*c1*d1 + beta3*(c1*c1*c1-3d0*c1*d1*d1)/6d0
do 55 i=1,NN,1
  s = (-PI + i*ds)/b
  y(2*i-1) = DCOS(d1*s+d2*zend)/DCOSH(c1*s+c2*zend)
  y(2*i) = DSIN(d1*s+d2*zend)/DCOSH(c1*s+c2*zend)
C
C   Compute the dispersion relation *****
w = 6d0
f=beta2*w*w/2d0+beta3*w*w*w/6d0+gama*(1d0+2d0*w/omega0)
if ( model .eq. 2 ) then
  Y(2*i-1) = COS(f*zend -w*s)
  Y(2*i) = SIN( f*zend -w*s)
  end if
55 CONTINUE
  return

  end

C
C
  subroutine liner(X,NN,ds,dz, alpha, beta2, beta3)
C-----
C   subroutine liner(X,NN,ds,dz, alpha, beta2, beta3)
C   This subroutine computes the linear step for the PDE
C   NN -- The number of grid points for the variable T.
C   ds -- The step size for the variable T.
C   X -- The array of values for the variable A at all grid points.

```

```

C   beta2, beta3, alpha -- Coefficients of the GNLS p.d.e.
C   dz -- stepsize for z
C-----
C
C   implicit NONE
C   integer NN,i, isign
C   DOUBLE PRECISION X
C   DOUBLE PRECISION ds,dz
C   DOUBLE PRECISION pi, tmpi,temp, cr, ci
C   DOUBLE PRECISION alpha, beta2, beta3
C   DOUBLE PRECISION b1, b3,be2
C   dimension X(*)
C   PARAMETER ( PI = 3.14159265358979d0)
C***** L operation *****
C   b1 = -alpha/2.0
C   be2 = beta2/2.0
C   b3 = -beta3/6.0
C   isign=1
C   call dfour1(X,NN,isign)
C   do 20 i=1,2*NN-1,2
C     temp = -PI*(i-1)/ds/NN
C     cr = b1*dz
C     ci = (b3*temp*temp*temp + be2*temp*temp)*dz
C     tmpi = X(i)*COS(ci)-X(i+1)*SIN(ci)
C     X(i+1) = X(i+1)*COS(ci)+X(i)*SIN(ci)
C     X(i) = tmpi
C     cr = (exp(cr))/NN
C     X(i+1) = X(i+1)*cr
C     X(i) = X(i)*cr
C 20 CONTINUE
C   isign=-1
C   call dfour1(X,NN,isign)
C   return
C   end
C
C
C   subroutine nolin(X,DX,NN,ds,dz,gama,omega0,TR)
C-----
C   subroutine nolin(X,DX,NN,ds,dz,gama,omega0,TR)
C   This subroutine computes a nonlinear step for GNLS PDE.
C   NN -- The number of grid points for the variable T.
C   ds -- The step size for the variable T.
C   dz -- The step size for the variable z.
C   X -- The array of values for the variable A at all grid points.
C   gama, omega0, TR -- Coefficients of the GNLS p.d.e.

```

C DX -- The array of derivative of X.

C-----

```

implicit NONE
integer NN,i
DOUBLE PRECISION X,DX
DOUBLE PRECISION ds,dz
DOUBLE PRECISION tmpr, tmpi,temp, cr, ci
DOUBLE PRECISION gama, omega0,TR
dimension X(*),DX(*)
do 1 i=1,2*NN-1,2
  ci = gama*( X(i)* X(i) + X(i+1)* X(i+1))*dz
  temp = X(i)*COS(ci)-X(i+1)*SIN(ci)
  X(i+1) = X(i+1)*COS(ci)+X(i)*SIN(ci)
  X(i) = temp
1 CONTINUE
C DX operation ****
C call deriv(X,DX,NN,NN2,ds)
  call deriv5(X,DX,NN,ds)
C call dxfft(DX,X,NN)
C***** M operation (exp(i*ci). X )*****
  cr = -2D0*gama/omega0
  ci= -gama*TR
  do 17 i=1,2*NN-1,2
    tmpr = 3D0*cr*( X(i)*DX(i)+X(i+1 )*DX(i+1))
    tmpi = 2D0*ci*( X(i)*DX(i)+X(i+1 )*DX(i+1))
    * + cr*( X(i)*DX(i+1)-X(i+1)*DX(i))
    tmpr = tmpr * dz
    tmpi= tmpi * dz
    temp = X(i)*DCOS(tmpi)-X(i+1)*DSIN(tmpi)
    X(i+1) = X(i+1)*DCOS(tmpi)+X(i)*DSIN(tmpi)
    X(i) = temp
    X(i) = X(i)*dexp(tmpr)
    X(i+1) = X(i+1)*dexp(tmpr)
17 CONTINUE
  return
  end

```

C  
 subroutine deriv5(X,DX,NN,ds)

C-----

```

C subroutine deriv5(X,DX,NN,ds)
C This subroutine computes the numerical derivative
C for periodic function.
C NN -- The number of grid points for the variable T.
C ds -- The step size for the variable T.

```

```

C   X -- The array of values for the variable A at all grid points.
c   DX -- The array of derivative of X.
C-----
C   implicit NONE
      integer NN,NN2,i
C   lowbd is a small number to check errors and to avoid underflow.
      PARAMETER ( lowbd = 1d-30)
      DOUBLE PRECISION X,DX
      DOUBLE PRECISION ds
      DOUBLE PRECISION f1,f2,f4,f5
      DIMENSION X(*),DX(*)
      NN2 = 2*NN
      do 1 i = 1,NN2
      if (i.LE.4) then
      f1 = X(i+NN2-4)
      else
      f1 = X(i-4)
      end if
      if (i.LE.2) then
      f2 = X(i+NN2-2)
      else
      f2 = X(i-2)
      end if
      if (i.gt.NN2-2) then
      f4 = X(i+2-NN2)
      else
      f4 = X(i+2)
      end if
      if (i.GT.NN2-4) then
      f5 = X(i+4-NN2)
      else
      f5 = X(i+4)
      end if
C
      if (abs( f4-f2) .LT. lowbd) then
      f4= 0d0
      else
      f4 =2.*(f4 -f2)/3.
      end if
      if (abs( f1-f5) .LT. lowbd) then
      f1= 0d0
      else
      f1 =(f1 -f5)/12.
      end if
      DX(i) = (f1+f4)/ds

```

```
1 CONTINUE
  return
  end
C
C SUBROUTINE dfour1(data,nn,sign)
C Refer "Numerical Recipes" Software (1986-92)
```



Figure 2. Initial 1-soliton with  $A(T) = \text{sech}(T)$  and the number of grid points  $N = 256$ ,  
length  $= 8\pi$

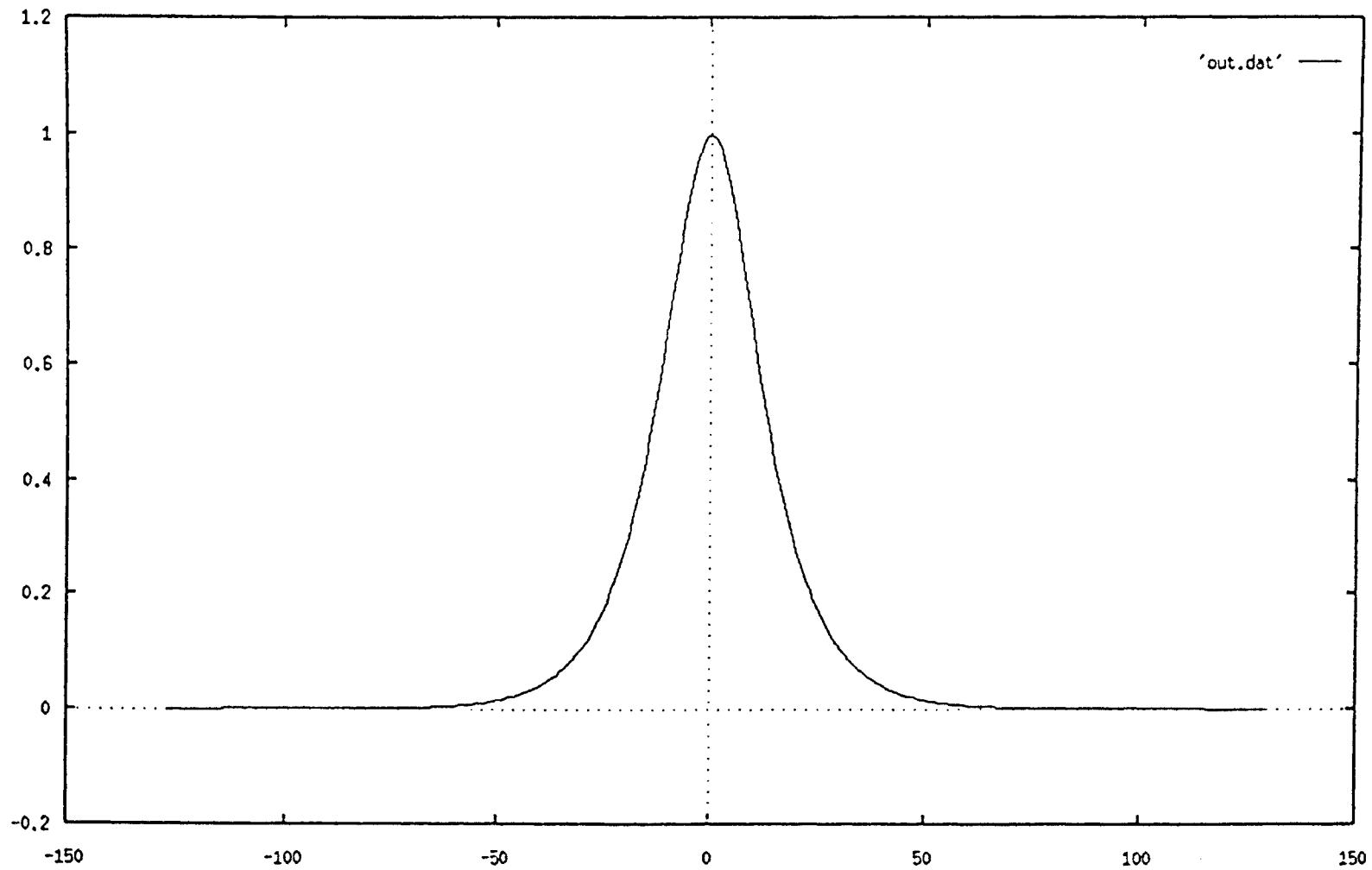


Figure 3. Output of the GNLS equation with  $\beta_2 = 0.02$ ,  $\beta_3 = 0.0012$ ,  $\gamma = -0.02$ ,  $\alpha = 0$ ,  
 $T_R = 0$ ,  $\omega_0 = 100$ ,  $N = 256$ , length =  $8\pi$ ,  $z = 100$ .  
The figure shows that the soliton propagates stably.

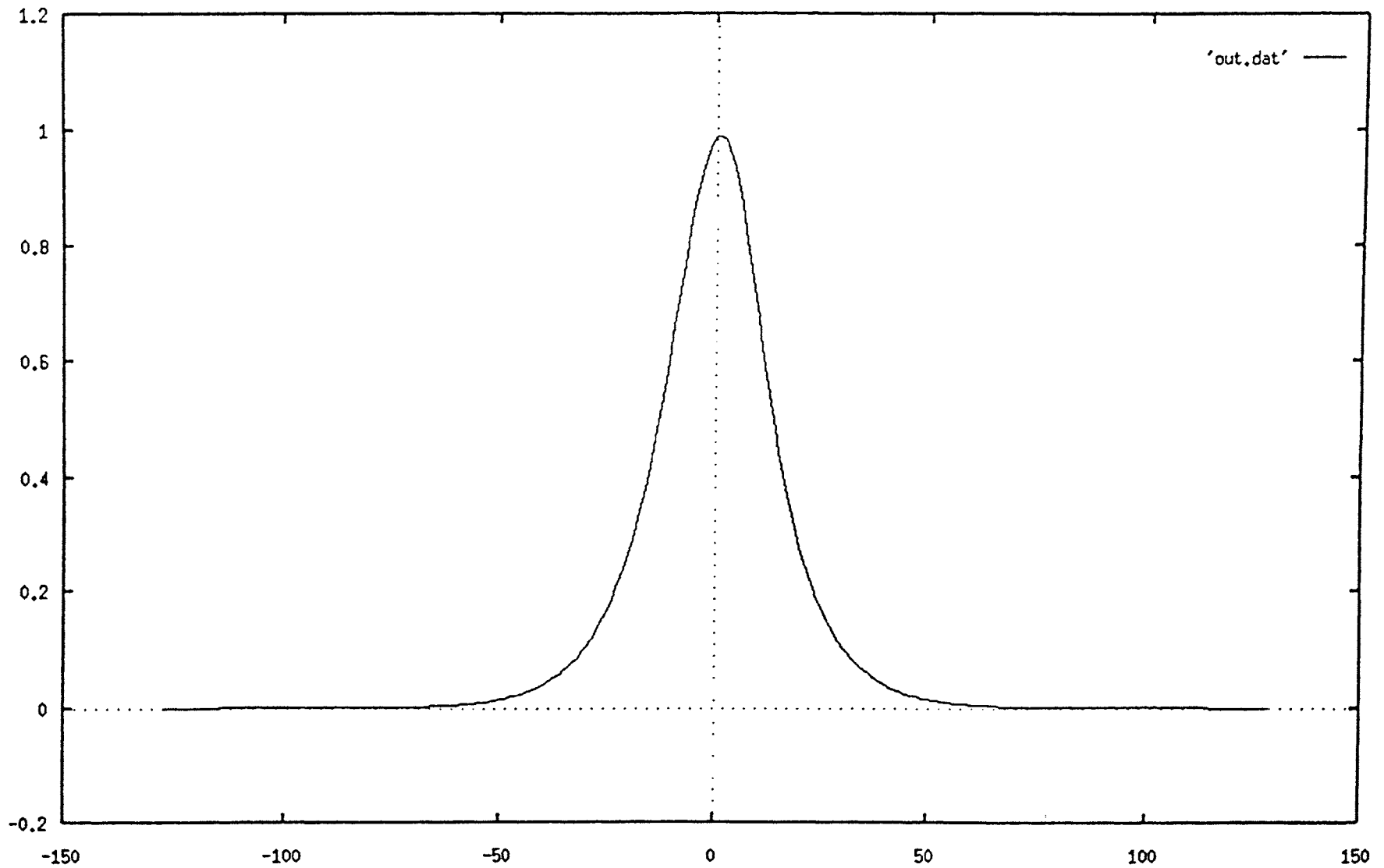


Figure 4. Output of the GNLS equation with  $\beta_2 = 0.02$ ,  $\beta_3 = 0.0012$ ,  $\gamma = -0.02$ ,  $\alpha = 0$ ,  $T_R = 0$ ,  $\omega_0 = 100$ ,  $N = 256$ ,  $\text{length} = 8\pi$ ,  $z = 300$ .

The figure shows that the soliton has clearly visible deformation, if  $z$  is big.

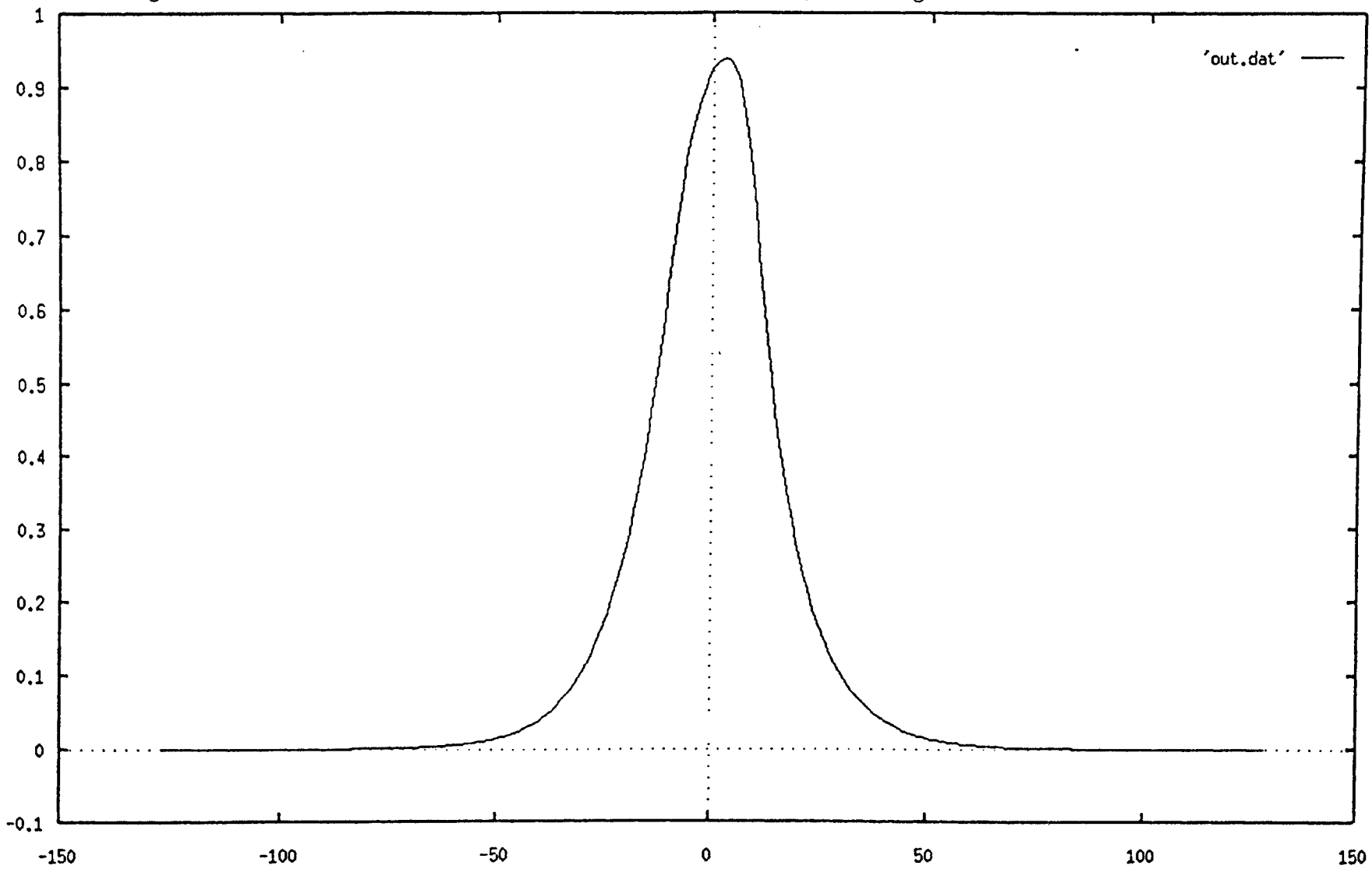


Figure 5. Initial 1-soliton with  $A(T) = \text{sech}(T)$  and the number of grid points  $N = 256$ ,  
length  $= 20\pi$

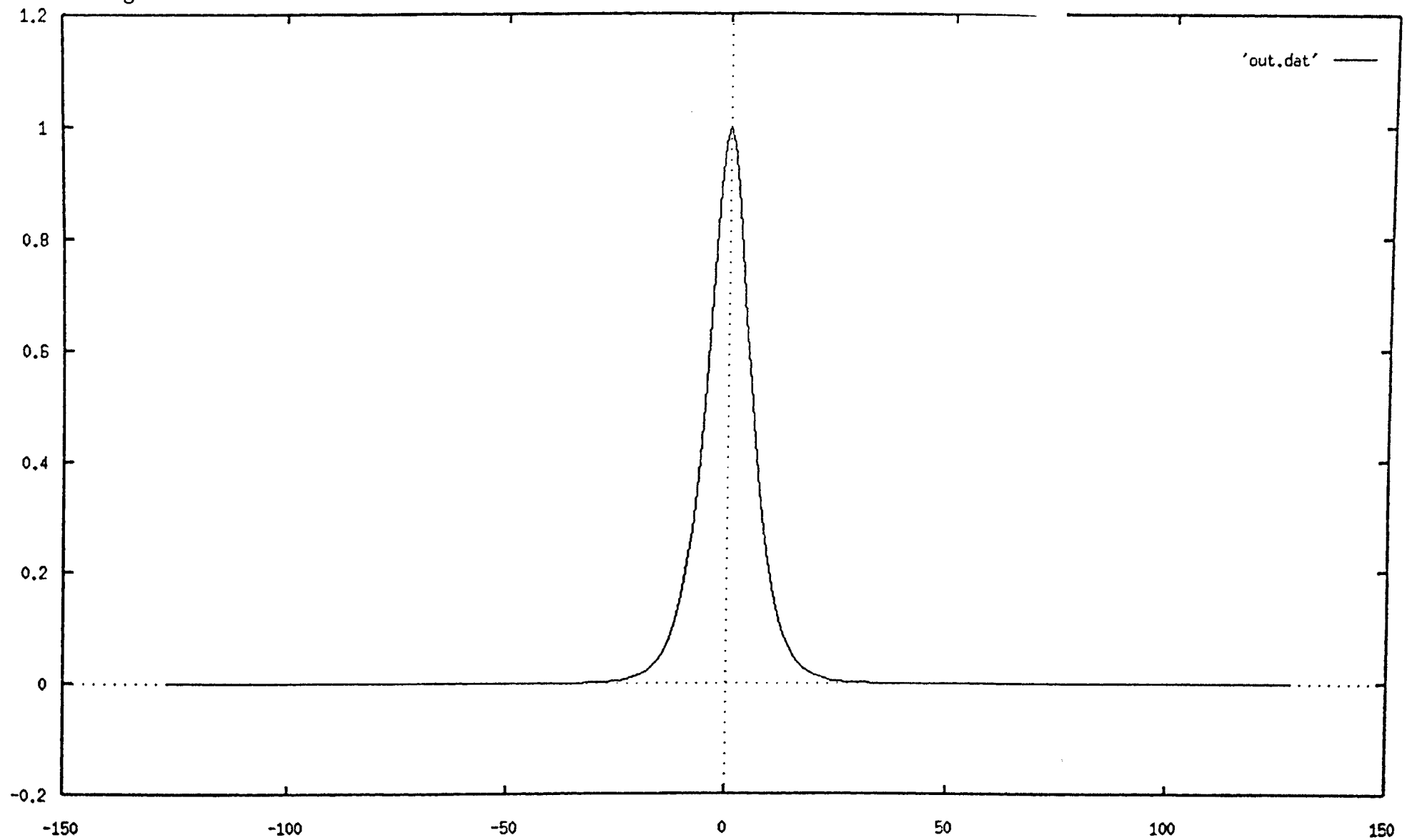
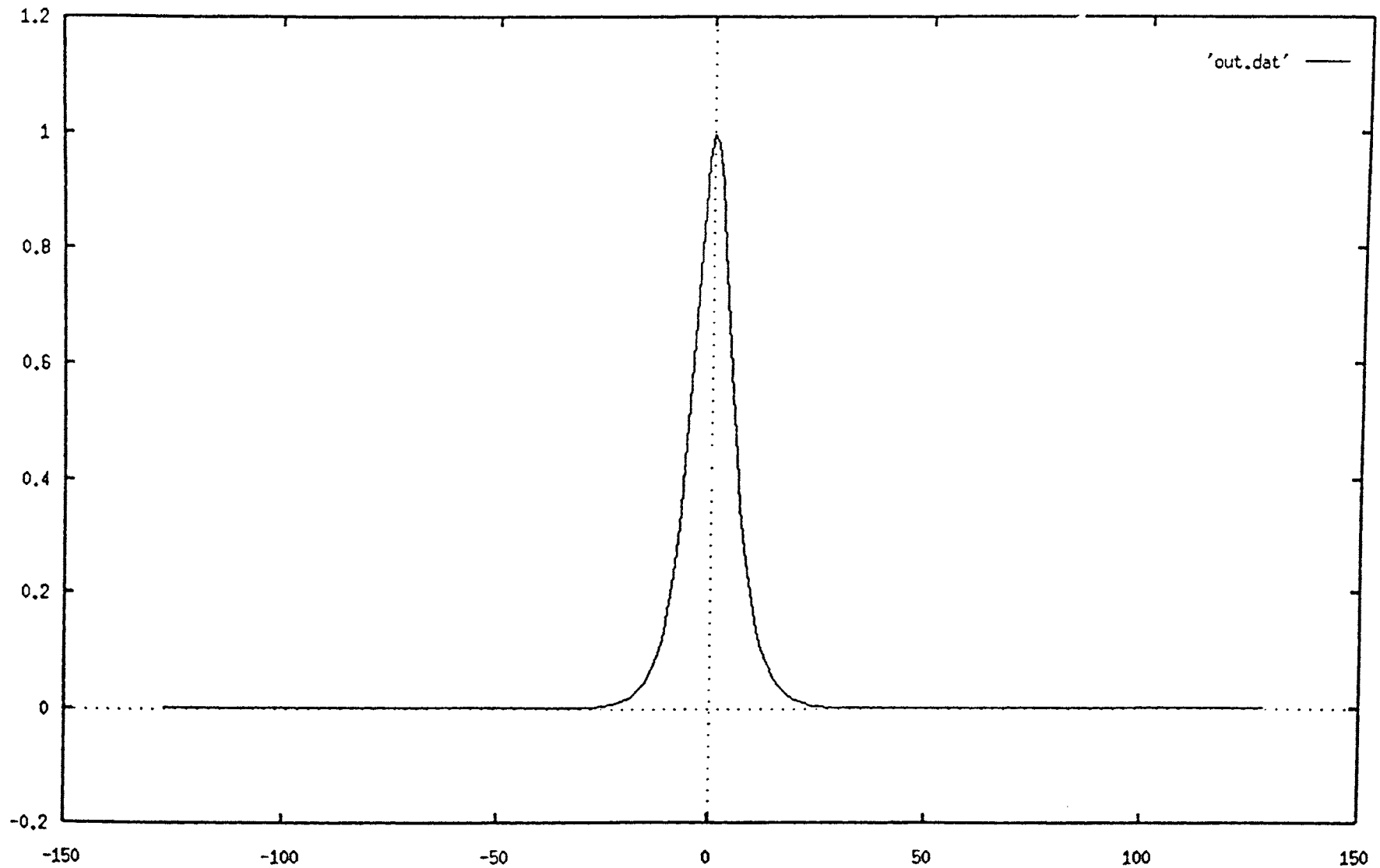


Figure 6. Output of the GNLS equation with  $\beta_2 = 0.02$ ,  $\beta_3 = 0.0012$ ,  $\gamma = -0.02$ ,  $\alpha = 0$ ,  $T_R = 0$ ,  $\omega_0 = 100$ ,  $N = 256$ , length =  $20\pi$ ,  $z = 60$ .  
The figure shows that the soliton propagates stably.



VITA<sup>v</sup>

Weiming Zheng

Candidate for the Degree of

Master of Science

Thesis: SPLIT-STEP FOURIER METHOD FOR GENERALIZED  
NONLINEAR SCHRÖDINGER EQUATION

Major field: Computer Science

Biographical:

Education: Received Master of Science in applied math from Shanghai Institute of Computer Technology, Shanghai, P. R. C in 1986. In 1992, passed the Comprehensive Exam for Ph.D at Oklahoma State University. Completed requirements for Master of Science in Computer Science in December, 1994.

Professional Experience: Research Assistant of Shanghai Institute of Computer Technology, Shanghai, P. R. C from August, 1986 till December, 1989. Graduate Teaching Assistant of Oklahoma State University, August, 1990 to present.

Professional Memberships: Association for Computing Machinery, Society for Industrial and Applied Mathematics, American Mathematical Society, Mathematical Association of America