THE UNIVERSITY OF READING

On the numerical solution of selected integrable nonlinear wave equations

Malachy McConnell

Numerical Analysis Report 5/2002

DEPARTMENT OF MATHEMATICS

University of Reading

On the numerical solution of selected integrable nonlinear wave equations

by

Malachy McConnell

September 2002

Department of Mathematics
This dissertation is submitted in partial fulfilment of the requirements of the degree of Master of Science

Abstract

Numerical solutions of the two dimensional Davey-Stewartson equations are presented along with a review of the one dimensional analogue, the nonlinear Schrödinger equation. The numerical solutions are based on a two dimensional extension of the Runge-Kutta and split-step Fourier methods. Numerical simulations are performed on the focussing Davey-Stewartson II equation to study the formation and evolution of lumps and rational solitons which continue to propagate after collision, without distortion. Further simulations are performed on the evolution of distorted lumps and Gaussian-type initial conditions in an attempt to find the asymptotic form of the wave for arbitrary initial conditions.

Keywords

Integrable, Davey-Stewartson, DSI, DSII, nonlinear Schrödinger equation, NLS, Runge-Kutta, split-step Fourier, soliton, dromion, lump.

Declaration

This report is my own work.

Contents

List of Figures

Chapter 1

Introduction

In recent years the Davey-Stewartson equations have attracted a good deal of interest in various physical and mathematical problems. The Davey-Stewartson equations are non-linear partial differential equations, depending on one temporal and two spatial variables, and can be thought of as the two dimensional analogue of the nonlinear Schrodinger equation (NLS). The NLS equation is used extensively for modelling non-linear optical phenomena. The Davey-Stewartson equations are used in fluid dynamics and plasma physics and many other branches of physics[4]. The DSII equation is

$$iu_t + \frac{1}{2}(u_{xx} - u_{yy}) + \sigma|u|^2u - u\phi_x = 0, \tag{1.1}$$

$$\phi_{xx} + \phi_{yy} - 2\sigma(|u|^2)_x = 0. \tag{1.2}$$

and the DSI equation is

$$iu_t + \frac{1}{2}(u_{xx} + u_{yy}) + \sigma|u|^2u - u\phi_x = 0 \tag{1.3}$$

$$\phi_{xx} - \phi_{yy} - 2\sigma(|u|^2)_x = 0 \tag{1.4}$$

with $\sigma = -1$ for the focussing case.

These equations can also be written in terms of the complex parameter $z = x + iy$, for example the focussing DSII becomes[2]

$$iu_t + u_{zz} + u_{\bar{z}\bar{z}} + 4(g + \bar{g})u = 0 \tag{1.5}$$

$$2g_{\bar{z}} - (|u|^2)_z = 0 \tag{1.6}$$

where $\bar{z} = x - iy$, $u(z, \bar{z}, t)$ and $g(z, \bar{z}, t)$ are complex valued functions (g contains σ).

The function $u(x, y, t)$ is the amplitude of a surface wave packet (the main flow) and ϕ is the velocity potential of the mean flow interacting with the surface wave.

The equations fall in to the special class of integrable equations. Examples of other integrable equations are the KdV and the sine-Gordon equation in one spatial dimension[8], and the KPI and KPII in two spatial dimensions[1].

There are several possible definitions of an integrable PDE; one of its defining properties is the existence of infinitely many conservation laws. In one space dimension, an integrable PDE can be written as the compatibility condition of two linear ODEs depending on a complex parameter: this is the basis of the inverse scattering transform method, a complicated and much celebrated linearising transform discovered thirty years ago. The use of this transform leads to the complete analytic solution of the Cauchy problem.

In two space dimensions, some integrable equations admit an analogue of the inverse scattering transform; however, these inverse scattering transforms are only defined formally; the analyticity is not guaranteed. Hence only very few analytical results are available.

The Cauchy problem for integrable equations in one dimension can be completely solved using the inverse scattering technique[1]. Carrying out this analysis leads to the discovery that, asymptotically, all decaying, travelling-wave solutions have a specific localised structure. Some of these localised solutions were already known, at least their analytic expression was. What was not known was their asymptotic significance.

In one dimension these special localised structures are generally called *solitons*; and some forms are known as kinks, antikinks and breathers. Some such localised solutions have been found for the two dimensional DS equations and are called *dromions* or *lumps*. Dromion deriving from the Greek word "dromos"-tracks. The distinction between dromions and lumps is that lumps require no boundary conditions whereas dromions require solitons as boundary conditions. Example analytic expressions of these coherent structures are:

Soliton of the KdV equation

$$u(x,t) = 3a^2\mathrm{sech}^2(\frac{ax}{2-a^3t}) \tag{1.7}$$

Soliton of the NLS equation

$$u(x,t) = \frac{p_R e^{i(p_I x + (p_R^2 - p_I^2)t + \eta)}}{\cosh(p_R(x - 2p_I t) + \eta} \tag{1.8}$$

where p_R, p_I and η are real parameters.

Lump of the DSII equation

$$u(z,\bar{z},t) = \frac{\beta\exp(i(p^2+\bar{p}^2)t + pz - \bar{p}\bar{z})}{|z+\alpha+2ipt|^2 + |\beta|^2} \tag{1.9}$$

where α, β and p are complex parameters. $\alpha = 0 + 0i$, $\beta = 2 + 0i$ and $p = 1 + 0i$ are suitable values.

Dromion of the DSI equation

$$u(x,y,t) = \frac{4i\exp(-(x+y+4t) - i(x+y))}{1 + (1+\exp(-2x-4t))(1+exp(-2y-4t))} \tag{1.10}$$

or

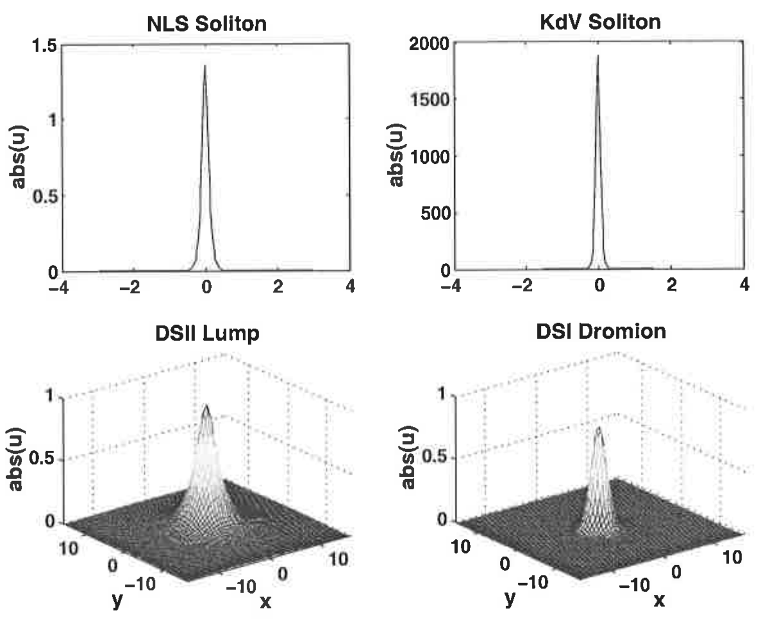

Figure 1.1: The Coherent structures of some integrable equations

$$u(x,y,t) = \frac{2\sqrt{2}\rho\lambda\exp(-\lambda(x+y)+2i\lambda^2t)}{|\rho^2| + (1+\exp(-2\lambda x))(1+exp(-2\lambda y))} \tag{1.11}$$

refer [15] and [1] for full descriptions of these equations. $\lambda = 1$ and $\rho = 1$ are suitable values. Figure(1.1) displays the forms of these structures.

The soliton solutions have a particular significance in 1-D as they emerge from any initial waveform carrying enough energy (amplitude). Hence they are the asymptotic structure of **any** decaying solution. For one dimensional integrable equations such as the NLS and KdV equations, there have been many numerical studies that any initial condition breaks up into some number of solitons after some sufficient time (Figure 1.2). This is also proved analytically using the inverse scattering transform.

In 2-D we do not know if all analogous phenomenon occur; hence this numerical study has been performed for the Davey-Stewartson equations. It is the goal of this research to study the solution for arbitrary initial conditions of the DSI and DSII equations, and particularly, to investigate the emergence of a lump structure in the case of the DSII

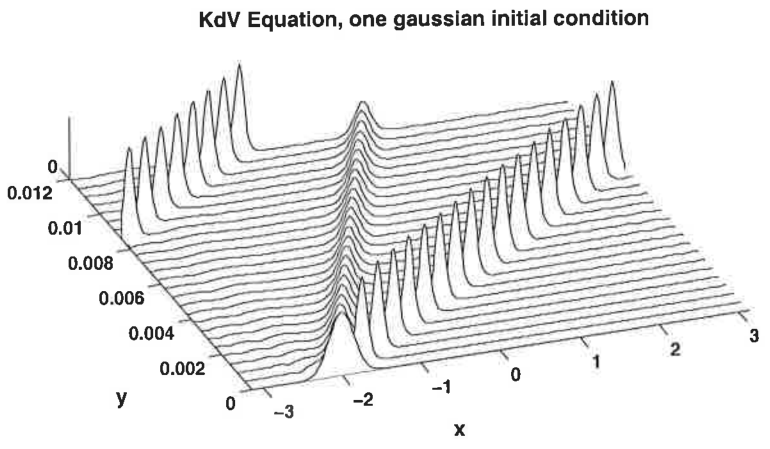

Figure 1.2: The splitting of a Gaussian initial condition in the focussing KdV equation.

equation.

One important difference between the KdV and the NLS equation is that the NLS has imaginary parts, and the NLS soliton has an $e^{i\theta}$ term (see [8]). This term determines the speed of the soliton. Compare the KdV soliton where the speed is proportional to the amplitude. If a real initial condition is given then the NLS equation will evolve without moving; while physically meaningless, this is useful for studying the evolution without concern over boundary conditions. In contrast all initial conditions in the KdV will form travelling waves. The DS equations are two dimensional analogues of the NLS and we expect the same amplitude/speed relation to be present. Real and imaginary initial conditions in the DSII equations were studied for this report.

We concentrate our attention on the DSII equation which is simpler; the DSI equation is more involved because it requires a boundary condition for the emergence of localised structures.

This report is organised as follows. In chapter 2, we give the expressions for two important one dimensional integrable equations, the NLS and KdV equations, and their 2+1 dimensional counterparts, DS and KP. We also review the numerical methods used in the literature and outline the numerical methods used in this research.

Numerical results for the NLS equation are given in chapter 3 and numerical results of the DSII equation are given in 4. Concluding discussions are given in chapter 5.

The appendices A through F contain algebraic derivations for various results, notes on practical application of fast Fourier transforms and matlab source code listings. Animations of the matlab output are available at `www.mathsconsultancy.co.uk`.

Chapter 2

Integrable equations of mathematical physics and their numerical solution

An example of an integrable equation in one dimension is the nonlinear Schrödinger equation (NLS)

$$iu_t + \sigma u_{xx} + \lambda |u|^2 u = 0 \tag{2.1}$$

where $\sigma = \lambda$ is the focussing case; in this case solitons are supported (figure 3.2). While for $\sigma = -\lambda$ the equation is defocussing (figure 3.3) and universally dispersive. The imaginary term in the NLS arises from physical nonlinearities so it models nonlinear media such as optical materials, plasma, and water waves in the presence of gravity and tension, e.g. capilliarity.

Another equation in this class is the Korteweg-de Vries equation (KdV),

$$u_t + uu_x + u_{xxx} = 0 \tag{2.2}$$

The KdV equation balances a nonlinear hyperbolic term uu_x and a linear dispersive term u_{xxx}. The KdV equation does not have an imaginary term and does not contain a focussing parameter.

The two dimensional (2+1 dimensional) extension of the KdV equation is the Kadomtsev-Petviashvili (KP) equation.

$$(u_t + 6uu_x + u_{xxx})_x + 3\sigma^2 u_{yy} = 0 \tag{2.3}$$

where $\sigma^2 = \pm 1$, see [1] relating to this equation.

A two dimensional extension of the NLS equation gives the DSI and DSII equations.

$$\begin{aligned} iu_t + \frac{1}{2}(u_{xx} + u_{yy}) + \sigma|u|^2 u - u\phi_x &= 0, & (2.4)\\ \phi_{xx} + \phi_{yy} - 2\sigma(|u|^2)_x &= 0. & (2.5)\end{aligned}$$

$$\begin{aligned} iu_t + \frac{1}{2}(u_{xx} - u_{yy}) + \sigma|u|^2 u - u\phi_x &= 0 & (2.6)\\ \phi_{xx} - \phi_{yy} - 2\sigma(|u|^2)_x &= 0 & (2.7)\end{aligned}$$

In one dimension, popular numerical methods for solving these equations are the split step Fourier method and the Runge-Kutta method via an integrating factor. In the split step Fourier method the equation is split in to a linear part and a nonlinear part which are then iterated in time sequentially. In the integrating factor method, an integrating factor is used to remove the linear part leaving a nonlinear equation in a form suitable for using a Runge-Kutta method.

For (2+1) dimensional equations, White and Weideman[15] use the split step Fourier method on the DSII and DSI equations. Yajima [9] studies the DSI equation using a pseudo spectral method for the numerical integration and either Burlish and Store method or fourth order Runge-Kutta method with adaptive step size control for the time integration. Besse and Bruneau use a modified Crank-Nicolson finite difference scheme also studying only the DSI equation[1].

We will concentrate our attention on the focussing DSII equation. This equation can be written in many equivalent ways; for example from White and Weideman[15]

$$\begin{aligned} iu_t + \frac{1}{2}(u_{yy} - u_{xx}) + \sigma|u|^2 u - u\phi_x &= 0 \qquad (2.8) \\ \phi_{xx} + \phi_{yy} - 2\sigma(|u|^2)_x &= 0 \qquad (2.9) \end{aligned}$$

The form modelled for this project is

$$\begin{aligned} iu_t + \frac{1}{2}(u_{xx} - u_{yy}) + \sigma|u|^2 u - u\phi_x &= 0 \qquad (2.10) \\ \phi_{xx} + \phi_{yy} - 2\sigma(|u|^2)_x &= 0 \qquad (2.11) \end{aligned}$$

Note the change of sign which essentially comes through as a change in the direction of the time, t. Fokas [6] uses the form

$$\begin{aligned} iu_t + u_{zz} + u_{\bar{z}z} + 4(g + \bar{g})u &= 0 \qquad (2.12) \\ 2g_{\bar{z}} - (|u|^2)_z &= 0 \qquad (2.13) \end{aligned}$$

This form is preferred for analytical investigations. See Appendix ref on deriving White from Fokas.

Besse and Bruneau [3] describe a full family of DS equations:

$$\begin{aligned} iu_t + \delta u_{xx} + u_{yy} &= \chi|u|^2u + bu\phi_x && (2.14) \\ \phi_{xx} + m\phi_{yy} &= \sigma(|u|^2)_x && (2.15) \end{aligned}$$

where the constants δ, χ, b, m and σ are real. This system describes the evolution of water surface waves in the presence of gravity and capilliarity. Following Ghidaglia-Saut, we classify these systems according to the sign of (δ, m) as elliptic-elliptic for $(\delta, m) : (+, +)$, elliptic-hyperbolic for $(+, -)$, hyperbolic-elliptic for $(-, +)$ and hyperbolic-hyperbolic for $(-, -)$.

The DSII equation described in this report is hyperbolic-elliptic.

Two numerical methods were used to model the NLS and DS equations: an integrating factor paired with a Runge-Kutta time-stepping scheme, and the split step Fourier method. The one dimensional case will be reviewed first, then this is extended into two dimensions. The numerical results are in chapters 3 and 4.

2.1 The NLS equation

2.1.1 The Split-Step Fourier Method in 1D

For the split step method the equation is split into a linear part 'L' and nonlinear part 'N'.

$$\mathrm{L} : iu_t + \sigma u_{xx} = 0 \qquad (2.16)$$

$$\mathrm{N} : iu_t + \lambda|u|^2u = 0 \qquad (2.17)$$

The linear part is solved in discrete Fourier space. In discretising and transforming to Fourier space...

$$\begin{aligned} u &\rightarrow \widehat{u}_k \\ u_t &\rightarrow \frac{d}{dt}\widehat{u}_k \qquad (2.18) \\ u_{xx} &\rightarrow -k^2\widehat{u}_k \qquad (2.19) \end{aligned}$$

$$k = \left(-\frac{M}{2}+1, \ldots -1, 0, 1, \ldots \frac{M}{2}\right)\left(\frac{2\pi}{2P}\right) \qquad (2.20)$$

see appendix A for detail of the discretisation, Fourier transform and explanation of the expression for k.

Equation (2.16) becomes

$$\frac{d}{dt}\widehat{u}_k = -ik^2\sigma\widehat{u}_k \qquad (2.21)$$

which is a separable ODE for $\widehat{u}$ with exact solution

$$\widehat{u}_k = Ae^{-ik^2\sigma t} \qquad (2.22)$$

and setting $A = \widehat{u}_0$ at $t = 0$ gives

$$\begin{aligned} \widehat{u}_k(t) &= \widehat{u}_0 e^{-ik^2\sigma t} \qquad (2.23) \\ \widehat{u}_k(t+\Delta t) &= \widehat{u}_k(t)e^{-ik^2\sigma\Delta t} \qquad (2.24) \end{aligned}$$

The nonlinear part is solved analytically too, by observing that the equation

$$iu_t + \lambda|a|^2 u = 0 \qquad (2.25)$$

has solution

$$u(x,t) = ae^{i\lambda|a|^2 t} \tag{2.26}$$

for any function $a(x) \in \mathbb{C}$, particularly $a(x) \equiv u(x)$ whence the solution is

$$u(x, t+\Delta t) = u(x,t)e^{i\lambda|u(x,t)|^2\Delta t} \tag{2.27}$$

where $|u|^2$ has been taken as time independent because $\int |u|^2 dxdy$ is conserved.

Then the solution is progressed in time by taking a small time step using the linear solution(2.24) followed by a small time step using the nonlinear solution(2.27). Schematically

$$u(t+\Delta t) = \mathrm{N}_{\Delta t}\{\mathrm{L}_{\Delta t}u(t)\} \tag{2.28}$$

or, to second order accuracy

$$u(t+\Delta t) = \mathrm{N}_{\frac{\Delta t}{2}}\{\mathrm{L}_{\Delta t}\{\mathrm{N}_{\frac{\Delta t}{2}}u(t)\}\} \tag{2.29}$$

The second order accurate scheme was used exclusively in this research.

2.1.2 The integrating factor method with Runge-Kutta time stepping in 1D

Before using the Runge-Kutta method for time stepping the nonlinear Schrödinger equation needs to be manipulated so that the linear part is absorbed by an integrating factor. First the equation is Fourier transformed and multiplied by $-i$

$$+\widehat{u}_t + i\sigma k^2\widehat{u} - i\lambda\widehat{|u|^2 u} = 0 \tag{2.30}$$

Multiplying by the integrating factor, found by inspection, $e^{+i\sigma k^2 t}$, we obtain

$$e^{+i\sigma k^2 t}\widehat{u}_t + i\sigma k^2 \widehat{u} e^{+i\sigma k^2 t} - i\lambda \widehat{|u|^2 u} e^{+i\sigma k^2 t} = 0 \tag{2.31}$$

Define $\widehat{U} = e^{+i\sigma k^2 t}\widehat{u}$ then $\widehat{U}_t = i\sigma k^2 e^{+i\sigma k^2 t}\widehat{u} + e^{+i\sigma k^2 t}\widehat{u}_t$ and $\widehat{u} = \widehat{U}e^{-i\sigma k^2 t}$, or

$$\widehat{U}_t = i\sigma k^2 \widehat{U} + e^{+i\sigma k^2 t}\widehat{u}_t \tag{2.32}$$

$$e^{+i\sigma k^2 t}\widehat{u}_t = \widehat{U}_t - ik^2 \widehat{U} \tag{2.33}$$

and substituting (2.33) into (2.35) gives

$$\widehat{U}_t - i\sigma k^2 \widehat{U} + i\sigma k^2 \widehat{U} - i\lambda \widehat{|u|^2 u} e^{+i\sigma k^2 t} = 0 \tag{2.34}$$

which rearranges to

$$\widehat{U}_t = i\lambda \widehat{|u|^2 u} e^{+i\sigma k^2 t} \tag{2.35}$$

Recalling that $\widehat{u} = \widehat{U}e^{-i\sigma k^2 t}$ and that $u = \mathcal{F}^{-1}(\widehat{u})$ where $\mathcal{F}$ represents the Fourier transform, the term $\widehat{|u|^2 u}$ can be written as

$$\widehat{|u|^2 u} = \mathcal{F}(\underbrace{|\mathcal{F}^{-1}(e^{-i\sigma k^2 t}\widehat{U})|^2}_{|u|^2}\underbrace{\mathcal{F}^{-1}(e^{-i\sigma k^2 t}\widehat{U})}_{u}) \tag{2.36}$$

and (2.35) becomes

$$\widehat{U}_t = i\lambda \mathcal{F}(|\mathcal{F}^{-1}(e^{-i\sigma k^2 t}\widehat{U})|^2 \mathcal{F}^{-1}(e^{-i\sigma k^2 t}\widehat{U})) \tag{2.37}$$

which is now in the form

$$\frac{d\widehat{U}}{dt} = f(\widehat{U}, t) \tag{2.38}$$

and can be solved for $\widehat{U}$ using a Runge-Kutta method. The Runge-Kutta method of fourth order is defined by the following algorithm:

```
for n = 0
    $a_1 = \Delta t f(\widehat{U}_n, t_n)$
    $a_2 = \Delta t f(\widehat{U}_n + \frac{1}{2}a_1, t_n + \frac{1}{2}\Delta t)$
    $a_3 = \Delta t f(\widehat{U}_n + \frac{1}{2}a_2, t_n + \frac{1}{2}\Delta t)$
    $a_4 = \Delta t f(\widehat{U}_n + a_3, t_n + \Delta t)$
    $t_{n+1} = t_n + \Delta t$
    $\widehat{U}_{n+1} = \widehat{U}_n + \frac{1}{6}(a_1 + 2a_2 + 2a_3 + a_4)$
next n
```

where $f(\widehat{U}_n, t_n) = i\lambda\mathcal{F}(|\mathcal{F}^{-1}(e^{-i\sigma k^2 t_n}\widehat{U}_n)|^2\mathcal{F}^{-1}(e^{-i\sigma k^2 t_n}\widehat{U}_n))$, and $\widehat{U}_0 = e^0\mathcal{F}(u_0) = \mathcal{F}(u_0)$.

The algorithm generates the solution of $\widehat{U}$ at time $t_n + \Delta t$. The desired answer, u at time $t_n + \Delta t$ is then calculated from

$$u(x,t) = \mathcal{F}^{-1}(\widehat{U}e^{-i\sigma k^2 t}) \qquad (2.39)$$

Accuracy and limits of stability were not calculated.

2.2 The KdV equation

The KdV equation can be analysed in a way similar to the one just outlined for the NLS. See Trefethen[12] for an integrating factor method and efficient RK4 code. Figure 1.2 shows the splitting behaviour for a Gaussian initial condition. The KdV causes the initial disturbance to split into any number of solitons of different amplitude. The soliton with greatest amplitude travels fastest and leads the wave train.

2.3 The DSII equation

2.3.1 Split step Fourier method for the DSII equation (2D)

We now turn to the DSII equation (2.10, 2.11), and split it into linear and nonlinear terms as described in White and Weideman [15], following the method of the previous section, but this time in two dimensions.

$$\mathrm{L}: iu_t - \frac{1}{2}(u_{yy} - u_{xx}) = 0 \tag{2.40}$$

$$iu_t + \sigma|u|^2u - u\phi_x = 0 \tag{2.41}$$

$$\mathrm{N}: \phi_{xx} + \phi_{yy} - 2\sigma(|u|^2)_x = 0 \tag{2.42}$$

Solving the linear term, we approximate the discrete u by the two dimensional discrete Fourier series

$$u_{jk} = \sum_m \sum_n \widehat{u}_{mn} e^{i(\mu_m x_j + \nu_n y_k)} \tag{2.43}$$

where $u_{jk} \approx u(x_j, y_k)$, see appendix A for definitions of x_j, y_k, μ_m, ν_n, j, k, m, n, and series limits.

The transformations

$$u \longrightarrow \widehat{u}_{mn}$$

$$u_t \longrightarrow \frac{d}{dt}\widehat{u}_{mn} \tag{2.44}$$

$$u_{xx} \longrightarrow -\mu_m^2\widehat{u}_{mn} \tag{2.45}$$

$$u_{yy} \longrightarrow -\nu_n^2\widehat{u}_{mn} \tag{2.46}$$

$$\mu = \left(-\frac{M}{2}+1,\dots -1,0,1,\dots \frac{M}{2}\right)\left(\frac{2\pi}{2P}\right) \tag{2.47}$$

$$\nu = \left(-\frac{N}{2}+1,\dots -1,0,1,\dots \frac{N}{2}\right)\left(\frac{2\pi}{2P}\right) \tag{2.48}$$

are used to write the Fourier space equivalent of the linear problem 2.40

$$i\frac{d\widehat{u}_{mn}}{dt} + \frac{1}{2}(\mu_m^2 - \nu_n^2)\widehat{u}_{mn} = 0 \tag{2.49}$$

which is a separable ODE with solution

$$\widehat{u}_{mn}(t) = \widehat{u}_{mn}(0)e^{(\frac{i}{2}(\mu_m^2-\nu_n^2)t)} \tag{2.50}$$

which gives the incremental solution

$$\widehat{u}_{mn}(t+\Delta t) = \widehat{u}_{mn}(t)e^{(\frac{i}{2}(\mu_m^2-\nu_n^2)\Delta t)} \tag{2.51}$$

White and Weideman[15] summarise this step...

The linear problem can therefore be solved as follows: Given the data $u_{jk}(t)$, compute the coefficients $\widehat{u}(t)$ from (2.43). This requires one two dimensional discrete Fourier Transform. Next advance the solution in Fourier space according to (2.51). The solution $u_{jk}(t+\Delta t)$ follows by taking the inverse Fourier transform [of $\widehat{u}_{mn}(t+\Delta t)$].

To solve the nonlinear part, equation 2.42 is the Laplace equation which is efficiently solved using spectral methods compared to five- or nine-point finite difference formula. In this particular case the Fourier representations

$$|u_{jk}|^2 = \sum_m \sum_n \widehat{|u_{mn}|^2} e^{i(\mu_m x_j + \nu_n y_k)} \tag{2.52}$$

$$\phi_{jk} = \sum_m \sum_n \widehat{\widehat{\phi}_{mn}} e^{i(\mu_m x_j + \nu_n y_k)} \tag{2.53}$$

are used to solve for $\phi(x, y)$ by writing the Fourier transformation of (2.42)

$$(\mu_m^2 + \nu_n^2)\widehat{\phi}_{mn} + 2\sigma i \mu_m \widehat{|u_{mn}|^2} = 0 \tag{2.54}$$

which rearranges to

$$\widehat{\phi}_{mn} = \frac{-2\sigma i \mu_m \widehat{|u_{mn}|^2}}{(\mu_m^2 + \nu_n^2)}, \quad m \neq n \neq 0 \tag{2.55}$$

Note that $\widehat{\phi}_{00}$ is undefined, but this does not hinder the solution because the required function for substituting into (2.41) is ϕ_x,

$$(\phi_x)_{jk} = \sum_m \sum_n i\mu_m \widehat{\phi}_{mn} e^{i(\mu_m x_j + \nu_n y_k)} \tag{2.56}$$

and clearly the $m = 0$, $n = 0$ value is zero.

To solve equation 2.41 the method of [15] is used.

> *...we follow the suggestion in [1] and use the strong coupling limit of the DS equations derived in [11]. Substituting $u = rie^{i\theta}$ into (2.41), with both r and θ assumed to be real functions of x, y, and t.*

which leads to the solution

$$u_{jk}(t+\Delta t) = u_{jk}(t)e^{(i(\sigma|u_{jk}(t)|^2-(\phi_x)_{jk}(t))\Delta t)} \tag{2.57}$$

Now equations (2.57), (2.56) and (2.51) can be used to solve for $u(x,y,t)$. White and Weideman[15] summarise the method...

The first order split step method for the DSII system proceeds as follows: Given the data u_{jk} at any time step t, first solve for $(\phi_x)_{jk}$ as indicated by 2.56. Then advance the solution according to the nonlinear part 2.57. This becomes the initial data for the linear problem which is solved by 2.51.

Great care must be taken when defining the variables in the numerical code particularly the definitions of j, k, μ_m, ν_n, M, N, and the grid size, P. See appendix A for a detailed description.

2.3.2 Integrating factor with Runge-Kutta method for the DSII equation

Here the method of section 2.1.2 is extended to solve

$$iu_t + \frac{1}{2}(u_{xx} - u_{yy}) + (\sigma|u|^2 - \phi_x)u = 0 \tag{2.58}$$

where ϕ_x is found using (2.56). As before, the first step is to multiply (2.58) by $-i$, discretise and Fourier transform...

$$\widehat{u}_t - \frac{1}{2}i(\mu_m^2 - \nu_n^2)\widehat{u} - \mathcal{F}\left[i\left(\sigma|u|^2 - \phi_x\right)u\right] = 0 \tag{2.59}$$

Now times this by the integrating factor of $e^{-\frac{i}{2}(k_x^2-k_y^2)t}$, which was found by inspection.

$$\widehat{u}_t e^{-\frac{i}{2}(k_x^2-k_y^2)t} - \frac{1}{2}i(\nu_n^2 - \mu_m^2)\widehat{u}e^{-\frac{i}{2}(k_x^2-k_y^2)t} - \mathcal{F}\left[i\left(\sigma|u|^2 - \phi_x\right)u\right]e^{-\frac{i}{2}(k_x^2-k_y^2)t} = 0 \tag{2.60}$$

Defining $\widehat{U} = e^{-\frac{i}{2}(k_x^2-k_y^2)t}\widehat{u}$ so that

$$\widehat{U}_t = -\frac{i}{2}(\mu_m^2 - \nu_n^2)e^{-\frac{i}{2}(k_x^2-k_y^2)t}\widehat{u} + \widehat{u}_t e^{-\frac{i}{2}(k_x^2-k_y^2)t} \tag{2.61}$$

which rearranges to

$$\widehat{u}_t = \widehat{U}_t e^{\frac{i}{2}(\mu_m^2-\nu_n^2)} + \frac{i}{2}(\mu_m^2 - \nu_n^2)\widehat{u} \tag{2.62}$$

substituting in (2.60) gives

$$\begin{aligned} e^{-\frac{i}{2}(k_x^2-k_y^2)t}\left[\widehat{U}_t e^{\frac{i}{2}(\mu_m^2-\nu_n^2)} + \frac{i}{2}(\mu_m^2 - \nu_n^2)\widehat{u}\right] &- \frac{1}{2}i(\nu_n^2 - \mu_m^2)\widehat{u}e^{-\frac{i}{2}(k_x^2-k_y^2)t} \\ -\mathcal{F}\left[i\left(\sigma|u|^2 - \phi_x\right)u\right]e^{-\frac{i}{2}(k_x^2-k_y^2)t} &= 0 \qquad (2.63) \\ \widehat{U}_t - \mathcal{F}\left[i\left(\sigma|u|^2 - \phi_x\right)u\right]e^{-\frac{i}{2}(k_x^2-k_y^2)t} &= 0 \qquad (2.64) \\ \widehat{U}_t &= \mathcal{F}\left[i\left(\sigma|u|^2 - \phi_x\right)u\right]e^{-\frac{i}{2}(k_x^2-k_y^2)t} \qquad (2.65) \end{aligned}$$

and $u = \mathcal{F}^{-1}\left[\widehat{U}e^{\frac{i}{2}(\mu_m^2-\nu_n^2)}\right]$ so finally we have

$$\widehat{U}_t = \mathcal{F}\left[\left[i\left(\sigma|\mathcal{F}^{-1}\left[\widehat{U}e^{\frac{i}{2}(\mu_m^2-\nu_n^2)}\right]|^2 - \phi_x\right)\mathcal{F}^{-1}\left[\widehat{U}e^{\frac{i}{2}(\mu_m^2-\nu_n^2)}\right]\right]e^{-\frac{i}{2}(k_x^2-k_y^2)t} \tag{2.66}$$

Comparing equation (2.37), this is in the suitable form for solving using the RK4 algorithm in section 2.1.2.

2.4 The DSI equation

2.4.1 Split step Fourier method for the DSI equation (2D)

The DSI equation

$$\begin{aligned} iu_t + \frac{1}{2}(u_{yy} + u_{xx}) + \sigma|u|^2u - u\phi_x &= 0 \qquad (2.67) \\ \phi_{xx} - \phi_{yy} - 2\sigma(|u|^2)_x &= 0 \qquad (2.68) \end{aligned}$$

is harder to solve because equation (2.68) is hyperbolic. Further, White and Weideman [15] point out

> *...the boundary conditions $\phi_x \to 0$, $x^2 + y^2 \to \infty$, do not give rise to any interesting solutions: all initial profiles simply disperse away (see [7]). To obtain coherent structures like dromions, non-trivial boundary conditions on ϕ need to be specified at infinity.*

To prepare the DSI for the split step method a change of variables to $\xi = x + y$, $\eta = x - y$ is made. This is the approach taken in [7]. The DSI system is transformed into

$$iu_t + (u_{xx} + u_{yy}) - uV = 0 \qquad (2.69)$$

where

$$V = \frac{1}{2}\sigma\left(\int_{-\infty}^{\xi} \left(|u|^2\right)_\eta d\xi + \int_{-\infty}^{\xi} \left(|u|^2\right)_\xi d\eta\right) + \phi_\xi(\xi, -\infty, t) + \phi_\eta(-\infty, \eta, t) \qquad (2.70)$$

Equation (2.69) can now be split into linear and nonlinear parts and solved using the split step method. The equation is split

$$\mathrm{L} : iu_t + (u_{xx} + u_{yy}) = 0 \tag{2.71}$$

$$\mathrm{N} : iu_t - uV = 0 \tag{2.72}$$

The linear part is solved as before, by taking the Fourier transform, advancing in time, and returning to physical space by an inverse transform. The equations are...

$$i\frac{d\widehat{u}_{mn}}{dt} + \frac{1}{2}(\mu_m^2 + \nu_n^2)\widehat{u}_{mn} = 0 \tag{2.73}$$

which is a separable ODE with solution

$$\widehat{u}_{mn}(t) = \widehat{u}_{mn}(0)e^{\frac{i}{2}(\mu_m^2+\nu_n^2)t} \tag{2.74}$$

which gives the incremental solution

$$\widehat{u}_{mn}(t+\Delta t) = \widehat{u}_{mn}(t)e^{\frac{i}{2}(\mu_m^2+\nu_n^2)\Delta t} \tag{2.75}$$

The nonlinear part (2.72) is solved as before by using the strong coupling limit [7, 15]. This leads to the solution

$$u(t+\Delta t) = u(t)e^{-V(t)\Delta t} \tag{2.76}$$

where the approximation $\int_t^{t+\Delta t} V\,dt = V(t)\Delta t + O(\Delta t^2)$ has been used.

The first order split step method is progressed by firstly using (2.76) which provides the initial data for the linear problem which, in turn, is solved using(2.75). The second order method proceeds as before; a half nonlinear step using (2.76) followed by a full linear step with (2.75) and finally a half nonlinear time step using (2.76)again.

The hard part here is calculating the function V. White and Weidman [15] write...

It remains to discuss the computation of V from (2.70) when u and boundary conditions $\phi_\xi(\xi,-\infty,t)$, $\phi_\eta(-\infty,\eta,t)$, are supplied. Continuing to assume boundary conditions $u \to 0$ as $\xi^2+\eta^2 \to \infty$, we consider the computational domain $[-P,P]\times[-P,P]$ as an approximation to $\mathbb{R}\times\mathbb{R}$. Thus the boundary conditions are assumed to be supplied at ξ, $\eta=-P$ rather than $-\infty$.

Before we look at the detail of the calculation, it is worth pointing out here that we were unsuccessful in reproducing the examples of White and Weideman [15] most likely because of incorrectly computing V. Peter White points out[14] that in his simulations he modified the boundary function from that at $-\infty$ to that at $-P$ using algebraic software to find the function ϕ at $-P$ exactly for the analytic dromion, rather than assuming $\phi_\xi(\xi,-\infty,t)\approx\phi_\xi(\xi,-P,t)$ as we have done here. The application of the modified boundary is left for future work.

Notwithstanding the boundary function, it is worth examining V and explaining its computation. To compute V from (2.70) the function is discretised and Fourier transformed. Taking the term

$$\int_{-\infty}^{\xi}\left(|u|^2\right)_\eta d\xi \tag{2.77}$$

of (2.70) we proceed as follows...

$$|u|^2 = \sum_m \sum_n b_{mn} e^{i(\mu_m \xi_j + \nu_n \eta_k)} \quad \text{i.e. } b_{mn} = \mathcal{F}(|u|^2) \;\&\; |u|^2 = \mathcal{F}^{-1}(b_{mn})$$

$$\int_{-P}^{\xi} \left(|u|^2\right)_\eta d\xi = \int_{-P}^{\xi} \left(\sum_m \sum_n b_{mn} e^{i(\mu_m \xi_j + \nu_n \eta_k)} \right)_\eta d\xi \tag{2.78}$$

$$= \int_{-P}^{\xi} \sum_m \sum_n b_{mn} i\nu_n e^{i(\mu_m \xi_j + \nu_n \eta_k)} \; d\xi \tag{2.79}$$

$$= \left[\sum_{m\neq 0} \sum_n b_{mn} \frac{i\nu_n}{i\mu_m} e^{i(\mu_m \xi_j + \nu_n \eta_k)} \right]_{-P}^{\xi_j} \tag{2.80}$$

$$= \sum_{m\neq 0} \sum_n b_{mn} \frac{\nu_n}{\mu_m} e^{i(\mu_m \xi_j + \nu_n \eta_k)} - \sum_{m\neq 0} \sum_n b_{mn} \frac{\nu_n}{\mu_m} e^{i(-\mu_m P + \nu_n \eta_k)} \tag{2.81}$$

$$\tag{2.82}$$

and for $m = 0$, put $m = 0$, $\mu_m = 0$ in (2.79) giving

$$= \int_{-P}^{\xi} \sum_n b_{0n} i\nu_n e^{i\nu_n \eta_k} d\xi \tag{2.83}$$

$$= \sum_n b_{0n} i\nu_n e^{i\nu_n \eta_k} \int_{-P}^{\xi} d\xi \tag{2.84}$$

$$= \sum_n b_{0n} i\nu_n e^{i\nu_n \eta_k} \; [\xi]_{-P}^{\xi_j} \tag{2.85}$$

$$= (\xi_j + P) \sum_n b_{0n} i\nu_n e^{i\nu_n \eta_k} \tag{2.86}$$

finally then

$$\int_{-P}^{\xi} \left(|u|^2\right)_\eta d\xi = \overbrace{\sum_{m\neq 0} \sum_n b_{mn} \frac{\nu_n}{\mu_m} e^{i(\mu_m \xi_j + \nu_n \eta_k)}}^{(1)} - \overbrace{\sum_{m\neq 0} \sum_n b_{mn} \frac{\nu_n}{\mu_m} e^{i(-\mu_m P + \nu_n \eta_k)}}^{(2)}$$

$$\overbrace{+ (\xi_j + P) \sum_n b_{0n} i\nu_n e^{i\nu_n \eta_k}}^{(3)} \tag{2.87}$$

The translation of equation (2.87) into computer code is worth a few remarks. Term (1) is a two-dimensional inverse fast Fourier transform, with the $m = 0$ result ignored and is coded

```
    bmn = fft2(abs(u).^2);
    dummy1 = kyyuneven./kxxuneven.*bmn;
    dummy1(:,1) = 0;      %pad the m=0 column with zeroes
    dummy1(:,M/2+1) = 0; %pad the other m=0 column with zeroes
    Vterm1 = real(ifft2(dummy1));
```

Term (2) is the $j = 1$, $\xi_j = -P$ column of term (1) and is coded

```
%subtract the j = 1 ie x(j) = -P column from every column in Vterm1
    dummy1 = Vterm1(:,1);
    for s = 1:M
       Vterm1(:,s) = Vterm1(:,s)-dummy1;
    end
```

Term (3) is a one-dimensional inverse Fourier transform and can be coded

```
    dummy1 = (1/M)*real(ifft(i*2*pi*(kyuneven)'.*bmn(:,1)));
    dummy2 = dummy1;
    for s = 2:M
        dummy2 = [dummy2 dummy1];
    end
    for s = 1:M
        dummy2(s,:) = dummy2(s,:).*(x+P);
    end
    Vterm1 = Vterm1+dummy2
```

Note in these code snippets $x \equiv \xi$, $y \equiv \eta, kxx \equiv \mu$ and $kyy \equiv \nu$.

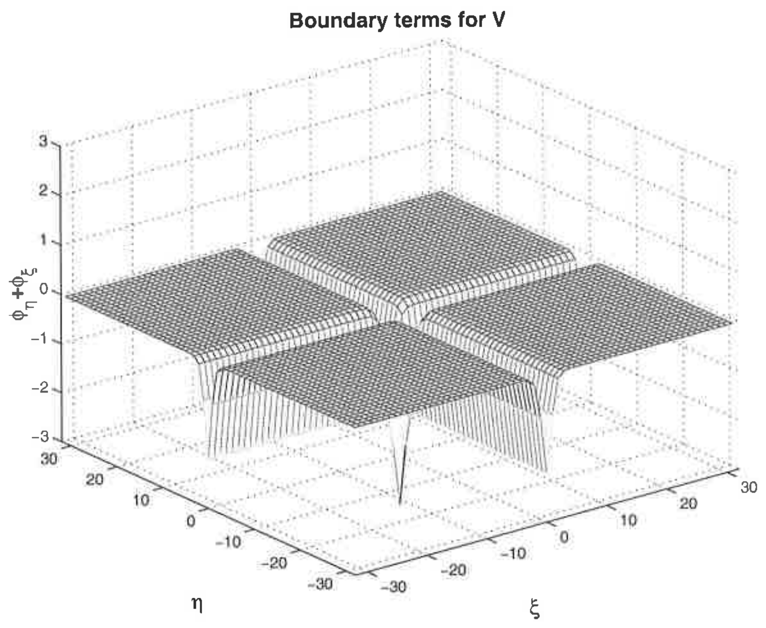

Figure 2.1: Visualisation of the boundary terms of the DSI equation

The process is repeated for the second part of equation (2.70) with η and ξ interchanged. The remaining part of V is constructed from the boundary values ϕ_ξ and ϕ_η. We advise that 'boundary value' is an unfortunate term because in the transformation to ξ, η these terms now run throughout the ξ, η plane, and are no longer associated with the boundary. It is better to think of these as the mean flow (the tracks for the dromion). For the DSI dromion to be supported ϕ_ξ and ϕ_η are given by

$$\phi_\xi(\xi,t) = -2\text{sech}^2(\xi+2t) \tag{2.88}$$

$$\phi_\eta(\eta,t) = -2\text{sech}^2(\eta+2t) \tag{2.89}$$

which in the computation of V, look like figure(2.1). The full function V is visualised in figure(2.2) along with individual plots of the constituent parts of equation (2.70) for V.

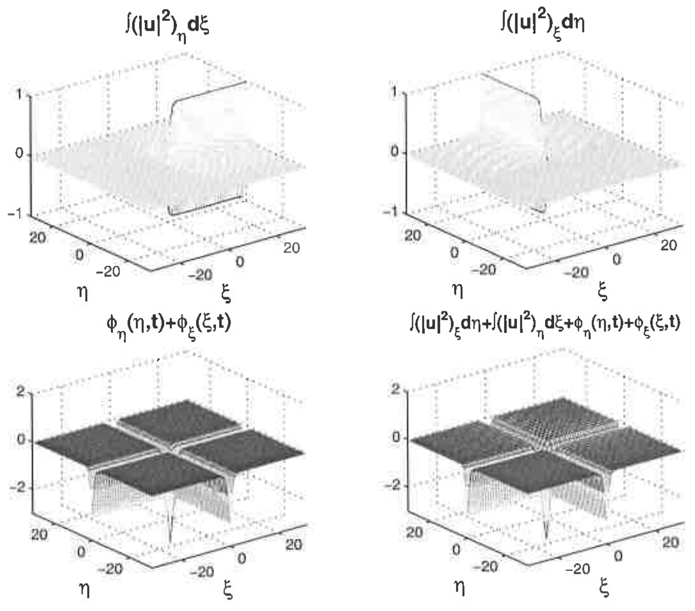

Figure 2.2: Visualisation of the function V, bottom right, and its constituent parts.

2.5 Boundary considerations

The Fourier method can be used because the numerical domain can be chosen to be large enough that the initial function is close to zero around the boundaries; hence for numerical purposes, the problem can be considered to be periodic. However, the DSII is a wave equation and in the study of long time evolution the boundaries are not always zero as energy (amplitude) travels toward the boundary.

This places a limit on the numerical method; namely any result after the function approaches the boundary is no longer valid. When the solution reaches the boundary it aliases round the back of the model and comes into the opposite boundary. In some cases this continues to be valid but is not a recommended modelling strategy.

To stop energy wrapping around the boundaries a filter can be used to ensure the boundary is all zero. Unfortunately, in a spectral method the presence of a filter causes reflection of energy back into the numerical domain all be it with significantly reduced amplitude. Nishinari & Yajima [9] use a linear edge filter in their model to good effect.

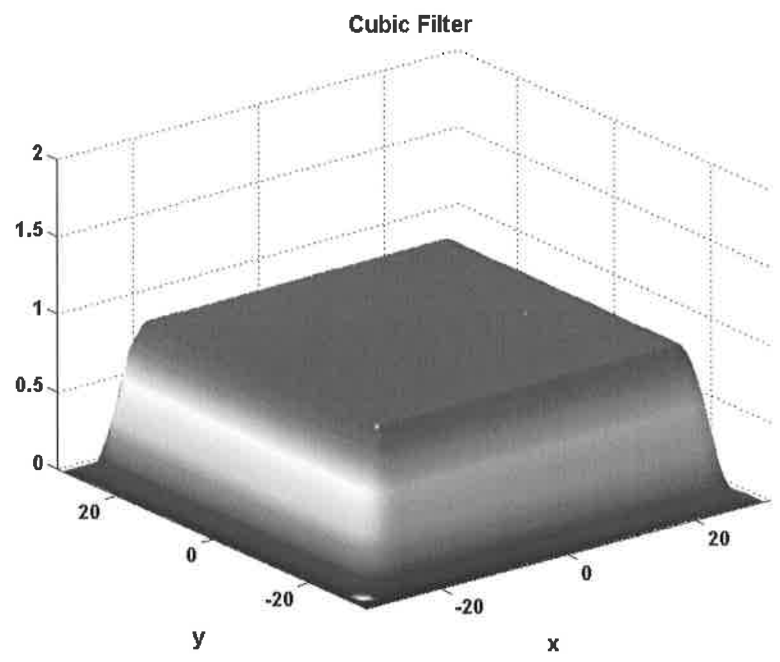

Figure 2.3: The serpentine filter function. This was not always used because it causes reflections, but is very useful in cases where small amplitude fast moving structure shoot off toward the boundaries.

The serpentine curve was used to design a filter. The serpentine curve was studied by Newton in 1701[10]. This curve was chosen because it is a smooth S shape (cubic). For Fourier methods smooth functions are preferred because they have no sharp changes, that is, no high frequency components. A cross section of the filter function is given in figure B.1 and the full three-dimensional effect on the computational domain shown in figure 2.3. Although the filter is useful in some cases, the best results are found using a large domain which forces a high number of grid points M and N; this in turn leads to long computational time.

The filter was used to good effect when studying Gaussian initial condit where small 'wavepackets' move away quickly and reach the model boundary within a few time steps.

Chapter 3

Numerical results of modelling the NLS equation

In this chapter the numerical results of the NLS equation are presented and discussed.

Both the split step and RK4 methods were implemented in matlab. Figure 3.1 shows the single soliton of equation (1.8) advancing in time without a loss of amplitude and no dispersion. Figure 3.2 shows two solitons travelling with different speed interacting and passing without exchanging energy; this is a celebrated result, and in fact this interacting property is sometimes taken as the definition of *integrable equation.* Figure 3.4 shows an initial gaussian function shedding energy and transforming into a single soliton in agreement with theory. These are all important characteristics of the one dimensional integrable equations and soliton solutions. Fokas writes[6]

> *. . . [Pasta and Ulam] discovered the defining property of Solitons: After interaction these waves regained exactly the shapes they had before. . . . it can be shown that $q(x,t)$ asymptotes to $q_N(x,t)$, where $q_N(x,t)$ is the exact N-soliton solution. This underlines the physical and mathematical significance of solitons: they are the coherent structures emerging from any initial data as $t \to \infty$.*

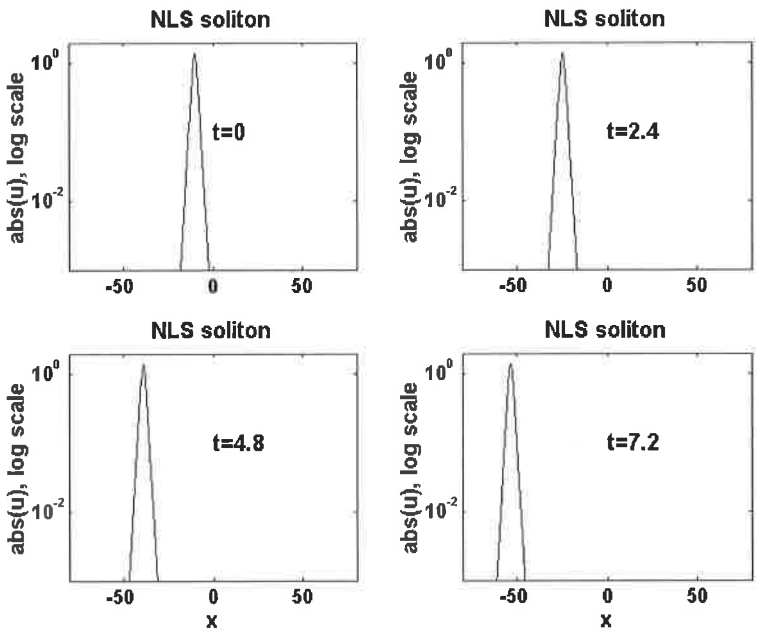

Figure 3.1: A soliton modelled using the split step Fourier method.

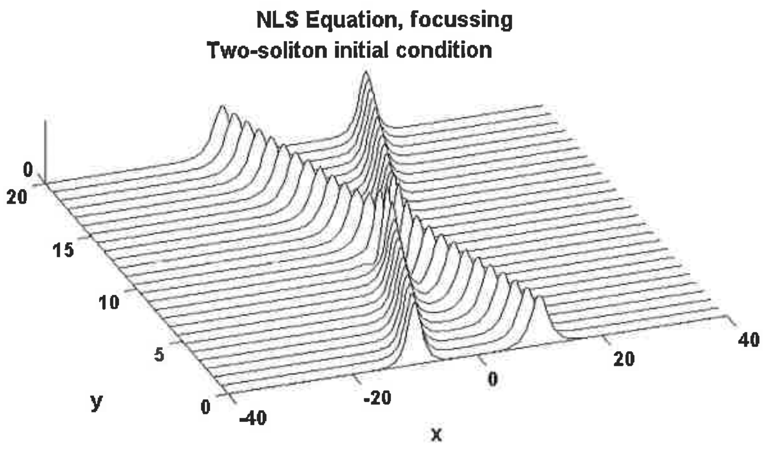

Figure 3.2: Two initial solitons propagate and cross over without change in shape.

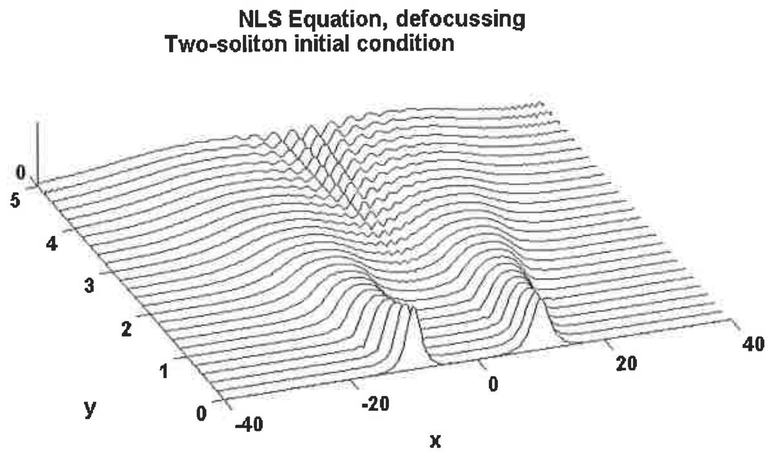

Figure 3.3: When $\sigma = 1$ the NLS equation is defocussing and solitons are not supported.

All these results were known from modelling and analytical studies. The reproduction here serves to show that the numerical schemes are working correctly.

An important characteristic of the NLS equation is that the gaussian initial condition sheds and moves. Previously this characteristic had not been modelled for the DSII equation and this research aims to model two gaussian initial conditions in a repeat of figures 3.5 and 3.6 for the two dimensional DSII equation.

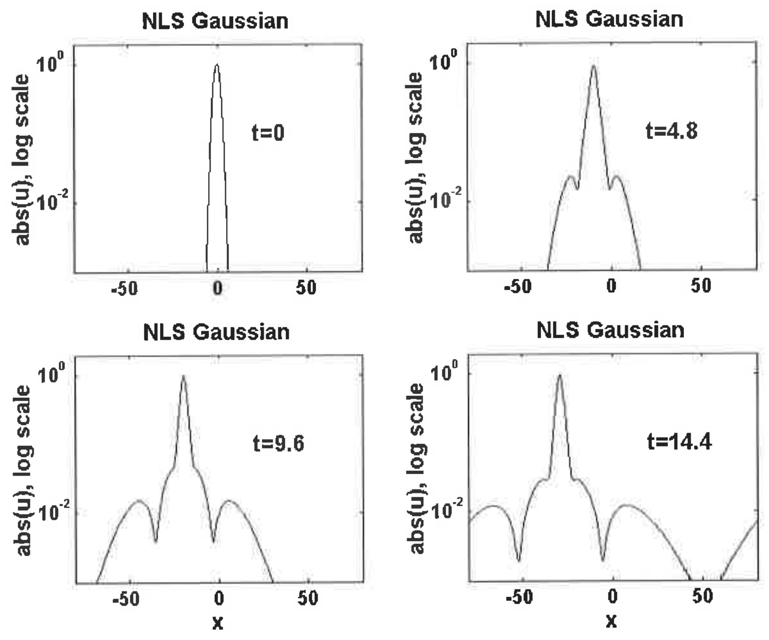

Figure 3.4: A Gaussian-type $(e^{-0.2x^2+ix})$ modelled using the split step Fourier method. Observe the shedding of faster moving waves as the initial Gaussian profile narrows to a soliton profile. Note the log scale.

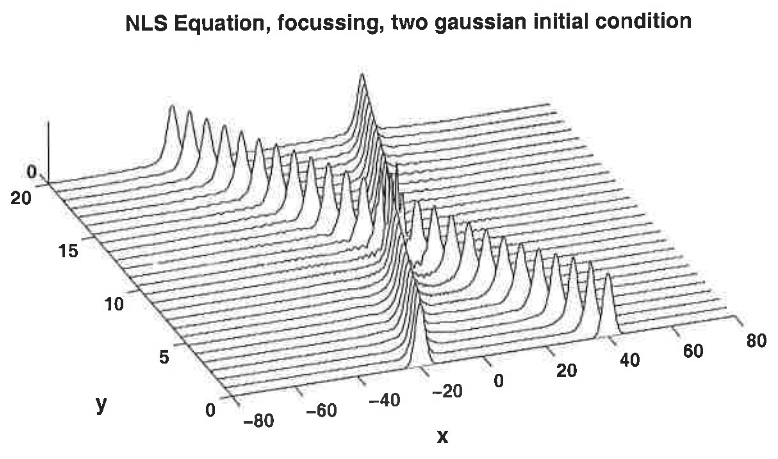

Figure 3.5: The two initial Gaussian inputs decay into solitons then continue to propagate without change of shape even after passing over each other.

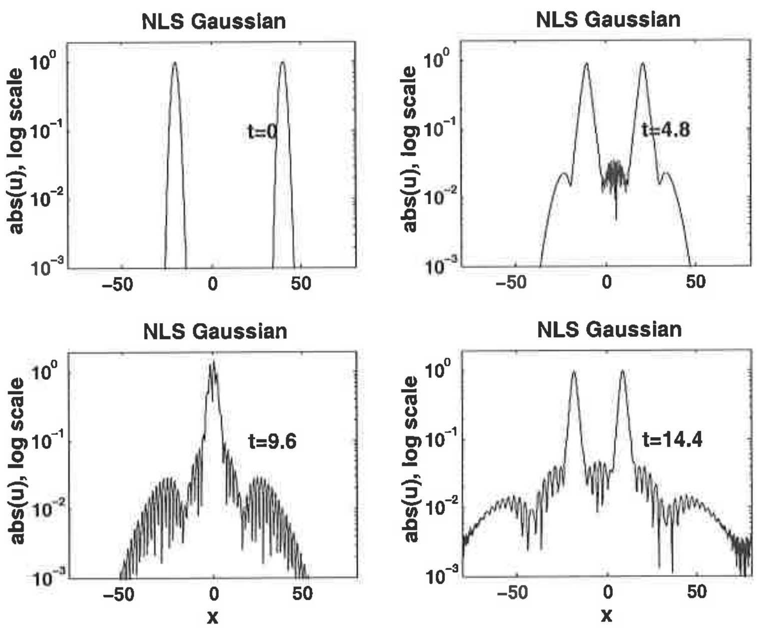

Figure 3.6: As figure 3.5, but plotted on log scale. Note the shedding and regaining of shape after crossing.

Chapter 4

Numerical results of modelling the DSII equation

In this chapter the numerical results of the DSII equation are presented and discussed.

4.1 Verifying the model

4.1.1 Reproducing known results

The first test was to reproduce the results of [15] where by the rational soliton, equation (4.1), was modelled for t in the interval (-3.5, 3.5), $x \times y = [-16,\ 16] \times [-16,\ 16]$ and $M = N = 64$, using a time step of $\Delta t = 0.01$ and $\sigma = -1$, the focussing case.

$$u(x, y, t) = \frac{2e^{2i(y-t)}}{1 + (x+1)^2 + (y-2t)^2} \tag{4.1}$$

Visually comparing figure 4.1 of this report with figure 1 of [15] is very encouraging, those results looking identical. The integrating factor and Runge-Kutta method code, referred to as RK4 from here on, was also used to reproduce this result. However, the spatial resolution of $h = \frac{32}{64} = 0.5$ is insufficient to model the rational soliton (see figure 4.2 - by $t = 7$ the tip is rounded and has fallen in amplitude). Figure 4.3 has $M = N = 256$

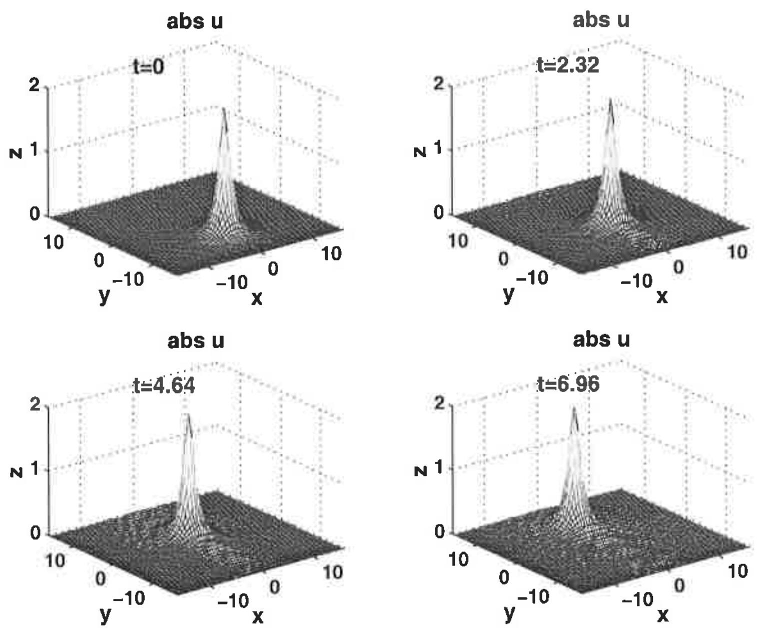

Figure 4.1: A reproduction of the result in [15]. A rational soliton propagates without distortion.

giving $h = 0.125$ and the rational soliton propagates as expected.

The same initial function was used with $\sigma = 1$, the defocussing case. Figure 4.4 shows the initial rational soliton diffusing away as expected. The same occurs when the RK4 model is used but the figures are not reproduced here because they visually identical to figure 4.4.

For an additional test the one lump solution, equation (1.9), was modelled. Figure 4.5 shows that the initial lump propagates without dispersion or loss of amplitude as expected (σ being equal to -1, focussing, for the remainder of these tests). We note that the one-lump has an amplitude equal to one for lossless propagation but the rational soliton requires an amplitude of two.

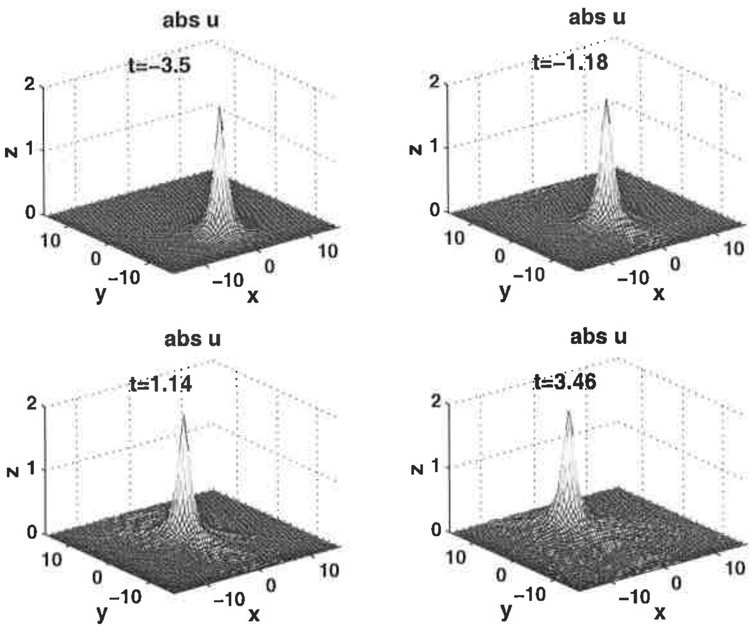

Figure 4.2: A reproduction of the result in [15], by the RK4 method. Note the resolution of $h = 0.5$ is insufficient and the initial wave form has started to disperse. See figure 4.3 with increased resolution.

4.1.2 Conservation laws

In their paper[15] Peter White and J.A.C. Weideman give two conservation laws for the focussing DSII equation

$$I_1 = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} |u|^2 dx\, dy \tag{4.2}$$

$$I_2 = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (|u_y|^2 - |u_x|^2 - \sigma|u|^4 + \frac{1}{2}(\phi_x^2 + \phi_y^2)) dx\, dy \tag{4.3}$$

and comment...

It is easy to show that our scheme preserves the discrete analogue of the integral I_1, and this was verified for the simulation. On the other hand, our method does not conserve the integral I_2. In practice, however, its discrete analogue

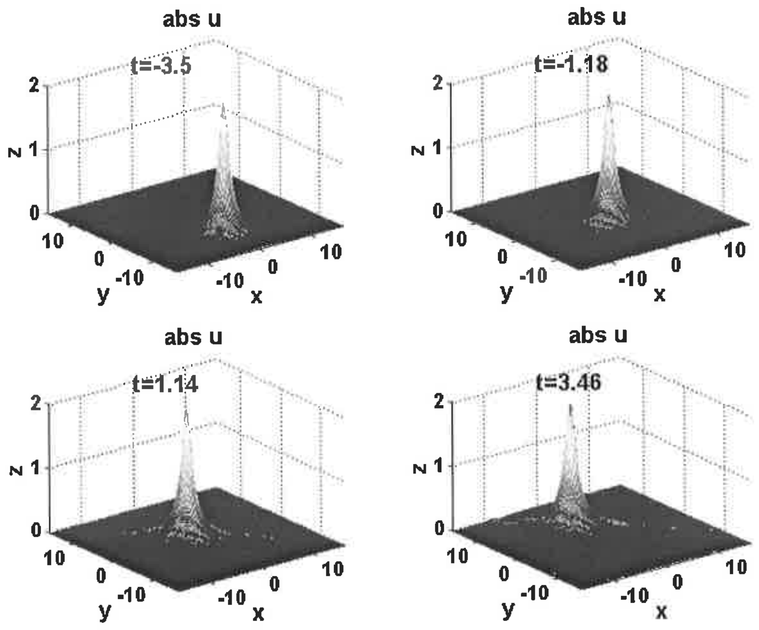

Figure 4.3: A reproduction of the result in [15], by the RK4 method. It was necessary to increase the resolution to $M = N = 256$ for the model to correctly simulate the rational soliton initial condition.

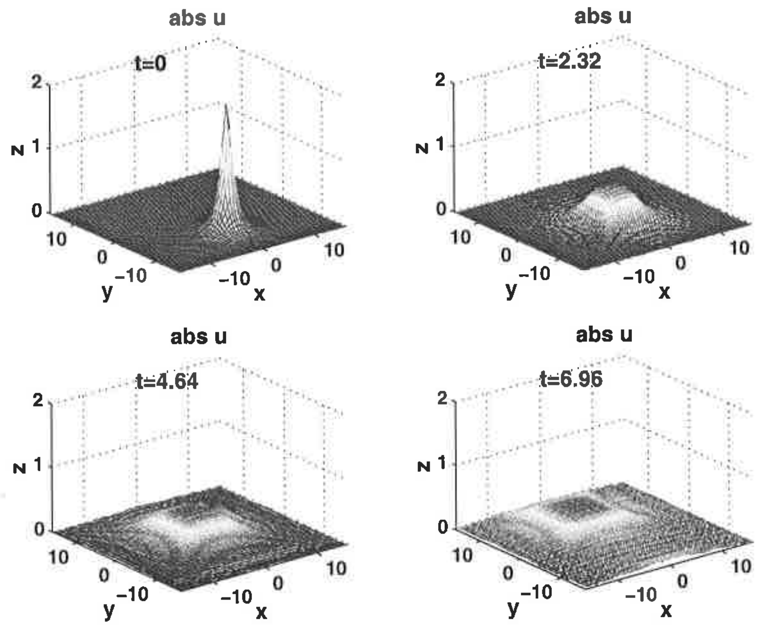

Figure 4.4: Any initial waveform diffuses away when $\sigma = 1$.

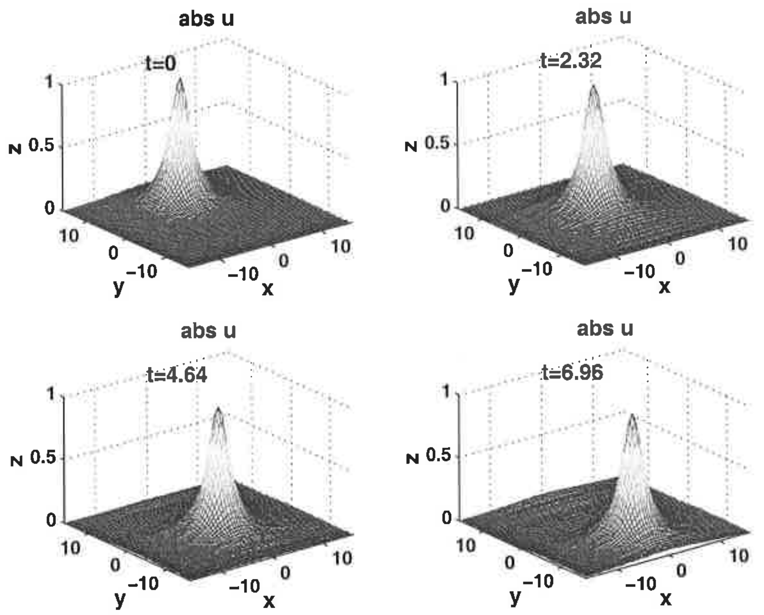

Figure 4.5: A propagating 1-lump solution, equation (1.9) with $\alpha = 0$, $\beta = 2$, $p = 1$.

was found to be constant to at least six significant digits over the period of the simulation.

The integral I_1 was evaluated for the split step method and varies by the machine precision (figure 4.6). Evaluating I_1 in the RK4 model, figure 4.7, shows a slight reduction in I_1 due to the loss at the boundaries; the RK4 method does not have cyclic boundaries.

The integral I_2 was not evaluated successfully (or I_2 is not conserved in the scheme!). This is left for future work. We are reasonably confident that the model is correct because it seems to model the correct evolution of the exact lump solutions.

4.1.3 Step size and grid resolution

The step size dt and grid spacing h_x and h_y were also varied. It was noted that misleading results occur when the grid spacing, h, is too large. Great care must be taken that these false solutions are not interpreted as valid waveforms. For the split step method, it was found that the grid spacing needed to be $h \leq 0.5$. The RK4 works for $h \leq 0.5$ but it is

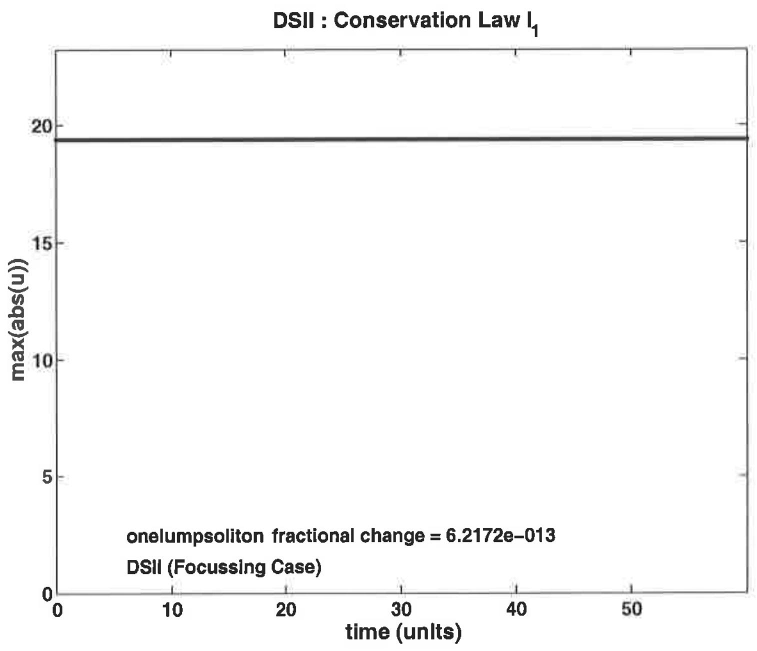

Figure 4.6: The split step fourier method preserves conservation law I_1 to machine precision.

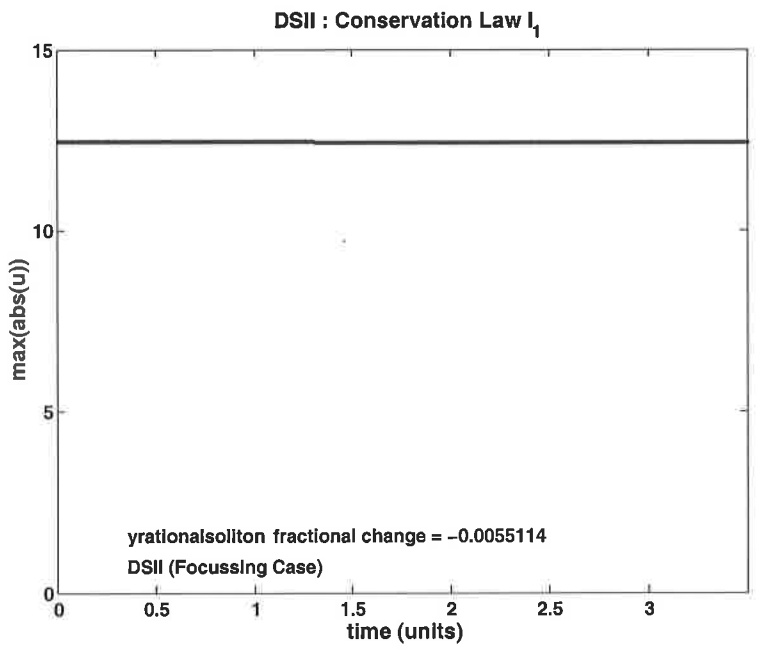

Figure 4.7: The RK4 method shows a 0.5% loss in I_1 here, due to loss out of the boundary.

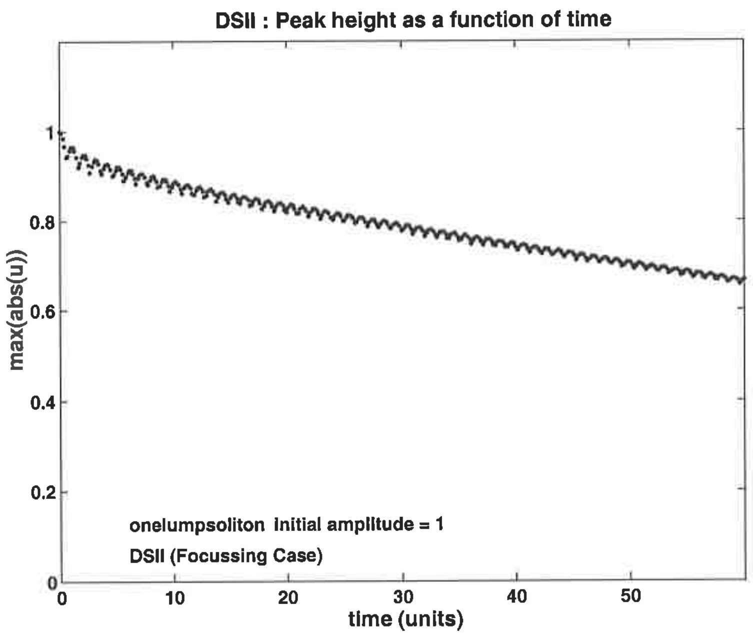

Figure 4.8: Incorrect tip height for a lump modelled with grid spacing $h = 1.00$. See figure 4.9 for the correct result.

better with $h \leq 0.25$. See figures 4.8 and 4.9 which show the tip height as a function of time; each of these models is on the same one lump initial condition, so the tip height was expected to be constant over time.

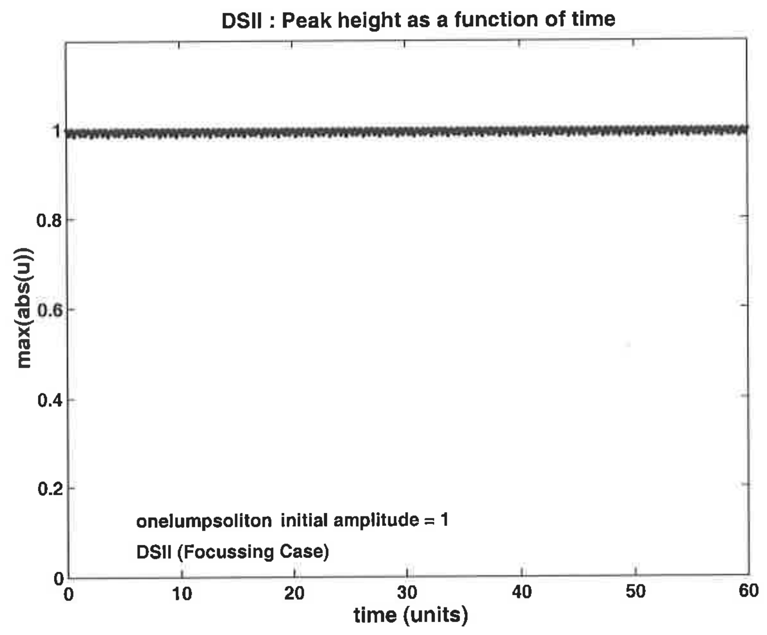

Figure 4.9: Tip height for a lump modelled with grid spacing $h = 0.50$.

4.2 Numerical Experiments

Having verified that lump and rational soliton initial values behave as expected, experiments were performed with different initial functions to investigate if the initial function would split into some form of slowly decaying lump or other coherent structure. Considering that the DS equations are the two-dimensional manifestation of the NLS equation, we might expect the initial real Gaussian to focus to a single 'soliton' as per the NLS result(3.4) and as per the DSI result reported by Yajima and Nishinari[9].

Yajima and Nishinari studied Gaussian type initial conditions, $u(x, y, 0) = a\exp[-\mu(x^2 + y^2) + i\theta]$, for the DSI equation and concluded that a standing form of dromion solution is formed by the shedding of energy along the main axes. In their paper they write

As for soliton equations in one dimension, such as the KdV equation and the nonlinear Schrodinger equation, the initial value problem is well studied by the inverse scattering transform or numerical analysis. These analyses show that *solitons emerge from an initial wave packet emitting radiations. In these one dimensional equations, solitons have their origins from the zeroes of scattering data, while dromions do not. Since dromions come from the focusing effect of boundaries, the effect of boundary conditions can be considered to play an important role in their formation.*

To begin with we investigate lump and rational soliton initial conditions and combinations of these functions.

4.2.1 Lumps and rational solitons

The situation of propagating lumps is very sensitive to every parameter in the DS equation and the initial condition. To demonstrate this the lump and rational soliton initial conditions were tested with 5% variation in initial amplitude, A. Figures 4.10 and 4.11 plot the maximum of function $u(x, y, t)$ with time for the three cases $A = 1$, $A = 1.05$, and $A = 0.95$ for each type of initial condition.

The rate of growth of the rational soliton is greater than that of the lump because the rational soliton starts with a larger magnitude and travels faster. Whether this is blow up or not is a moot point.

The increase in initial amplitude causes growth of the peak as the nonlinear term dominates. The shorter amplitude diffuses away because the nonlinear term is not strong enough to over come the diffusion process. For the case of $A = 1$ the diffusion and nonlinear focussing are balanced perfectly and the resulting waveform propagates without change.

Fokas comments[5] that this is not so curious because we have only changed the amplitude and not the speed. But he is not sure what should happen: Is this a blow up condition or is the lump growing up into some other lump? This is left for future work.

To observe larger amplitude solitons, the speed also needs to be increased appropriately. Unfortunately it is beyond the scope of this project to understand the theory behind

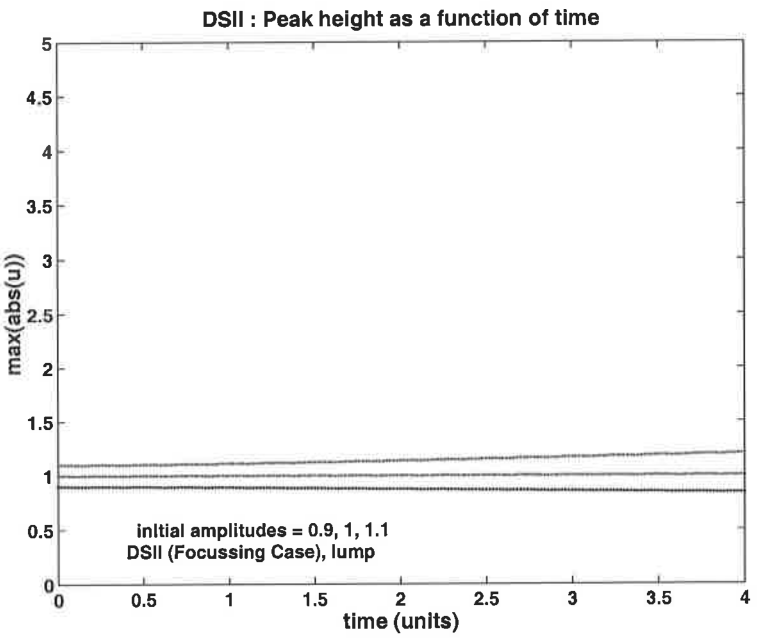

Figure 4.10: Tip height for a lump modelled with grid spacing $h = 0.25$.

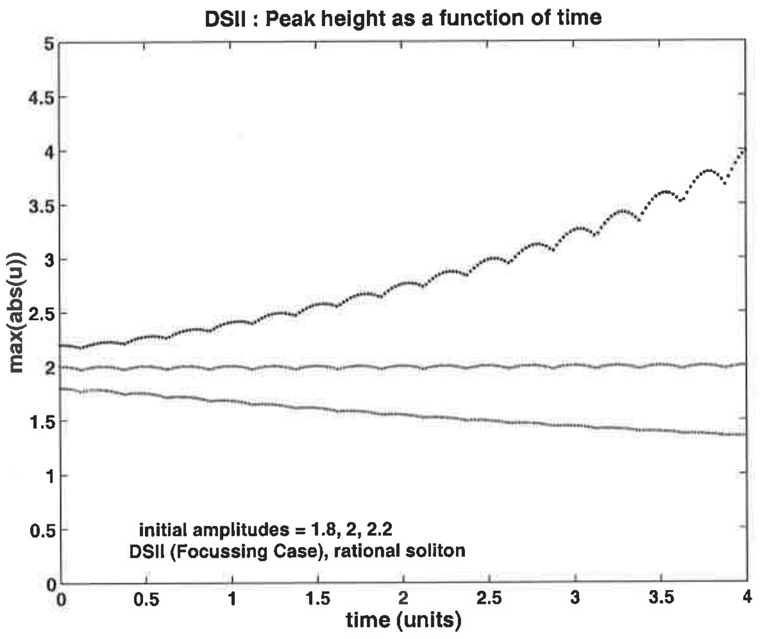

Figure 4.11: Tip height for a rational soliton modelled with grid spacing $h = 0.25$.

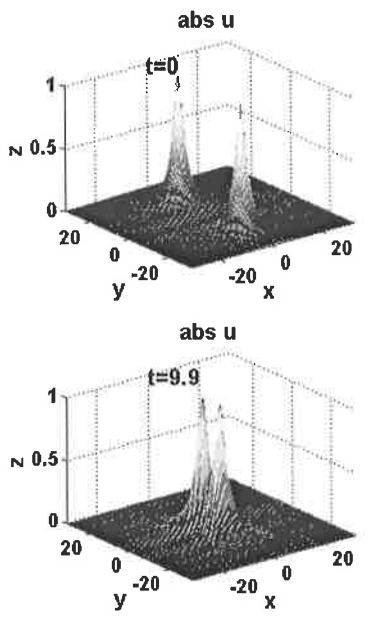

Figure 4.12: Two lumps approach, combine and pass, without interference.

the analytic relationship between speed and amplitude. Numerical testing of the evolution of large amplitude initial conditions is left to future work because there are many variables and the large amplitude tests require high grid resolutions of $h \leq 0.0625$ to resolve the high frequencies of the sharp tips.

Here we will concentrate on the combination of lumps and rational solitons in collisions to investigate if they propagate, mix, and pass without change. Figures 4.12 to 4.15 show examples of these coherent initial conditions passing without change. A detailed study of phase changes has not been made.

The fact that these initial conditions all pass over each other without change is encouraging because if we look at the superposition at the point of intersection it represents a large amplitude initial condition decaying into many lumps. In the next figure 4.16 the initial condition is the linear superposition of four lumps which evolves into four separate lumps. This raises the question 'Can we find less contrived initial conditions that decay into one or more lumps'.

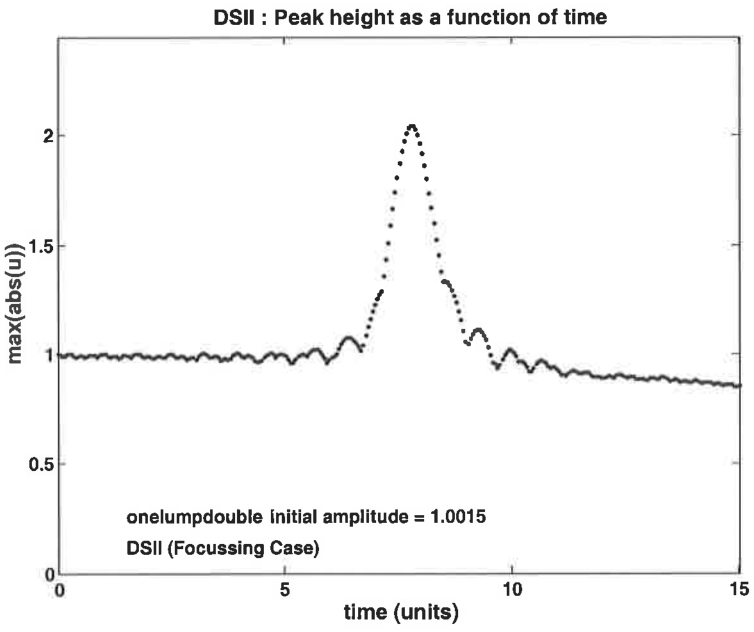

Figure 4.13: Tip height plot of the simulation in figure 4.12. We propose the slight reduction in amplitude after crossing is due to insufficient resolution.

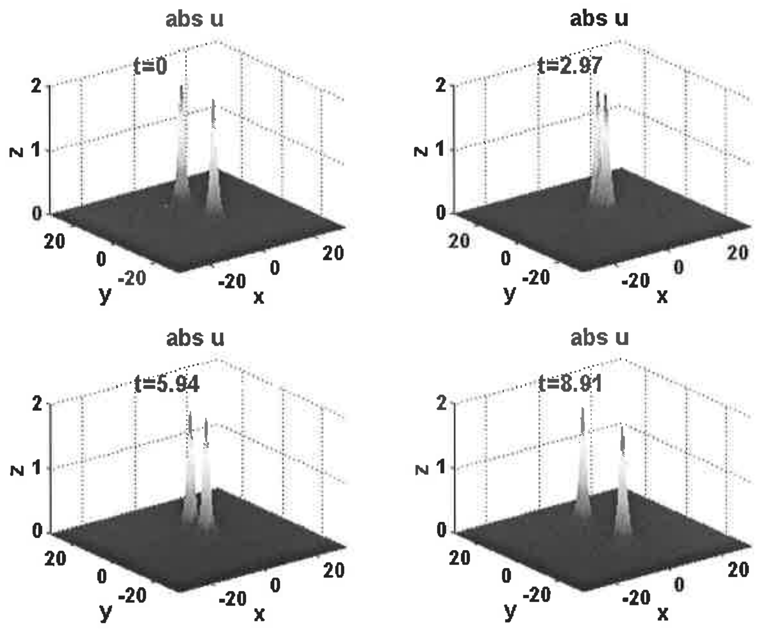

Figure 4.14: Two rational solitons approach, combine and pass, without interference.

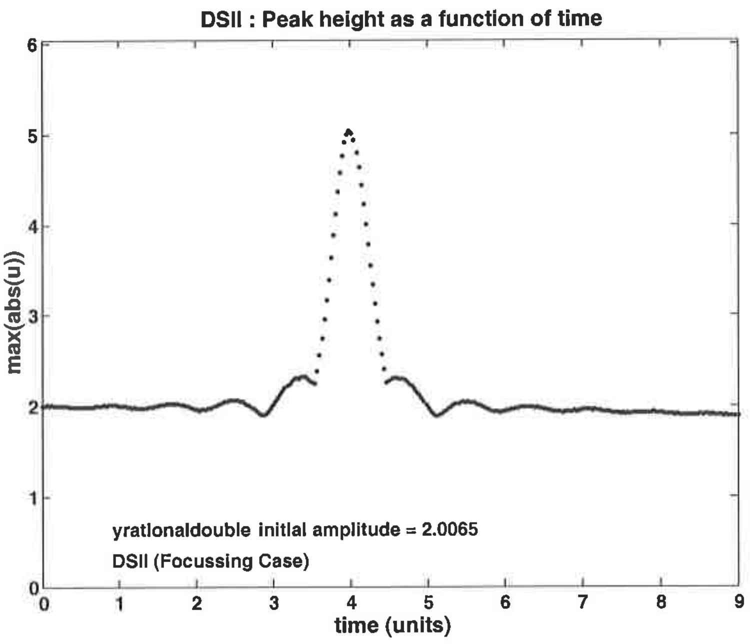

Figure 4.15: Tip height plot of the simulation in figure 4.14. We propose the slight reduction in amplitude after crossing is due to insufficient resolution.

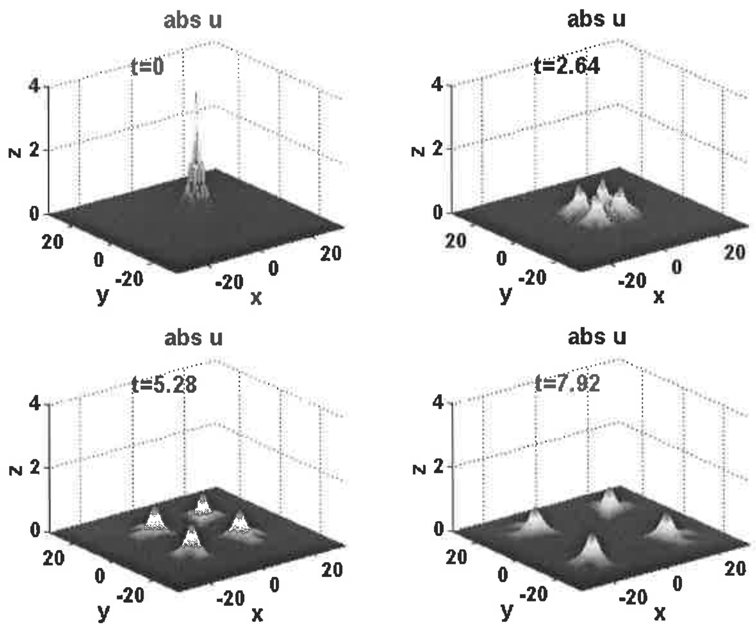

Figure 4.16: The initial condition is the linear superposition of four lumps. The initial amplitude is 4, rises to 7, then the individual lumps separate and move off.

4.2.2 Gaussian type initial condition

Modelling the Gaussian type initial conditions is difficult because the initial forms move very slowly or not at all and require a high grid resolution ($h \leq 0.25$ to resolve any shedding). Three Gaussian type initial conditions were tested. They are the real Gaussian

$$u(x, y, 0) = Ae^{-\mu(x^2+y^2)} \tag{4.4}$$

where μ is a real parameter typically $\mu \approx 1$. The complex Gaussian

$$u(x, y, 0) = Ae^{-\mu(x^2+y^2)+i\theta} \tag{4.5}$$

θ a real parameter, and

$$u(x, y, 0) = Ae^{-\mu(x^2+y^2)+i\theta y} \tag{4.6}$$

which we call the moving Gaussian because this is the only form of Gaussian initial condition found, that travels as the model runs. The speed is slow, and dependent on θ - the higher θ the faster the movement, but the model requires smaller h in these cases because $i\theta y$ represents large frequencies for large θ ($\theta \geq 3$). We note $u(x, y, 0) = Ae^{-\mu(x^2+y^2)+i\theta x}$ also moves.

The real and complex Gaussian initial conditions do not travel. The real Gaussian is not a physically meaningful solution, although mathematically valid, however it is the simplest to study.

Figure 4.17 shows a low amplitude real Gaussian initial condition diffusing away. When the amplitude is increased to $A = 4$ (figures 4.18 and 4.19) and $A = 6$ (figures 4.20 and 4.21), the initial function sheds energy like the NLS except that the amplitude increases and the dominant peak narrows. Compare figures 4.18 and 3.4.

What is not known is what these are evolving into. They appear to be stationary lumps which maintain shape and amplitude -this may be a consequence of the grid resolution.

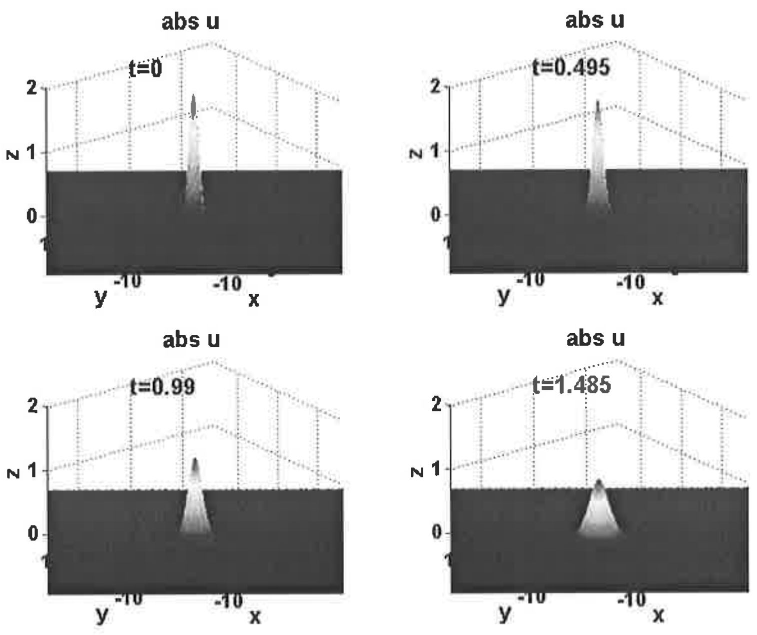

Figure 4.17: The real Gaussian initial condition, equation (4.4), diffuses when the initial amplitude is small, $A = 2$ in this case.

It is observed over many simulations that the tip height depends on the resolution, h.

The complex Gaussian, equation(4.5) exhibits the same behaviour as the real Gaussian and the results are not given here.

The moving Gaussian initial condition acts in a similar way to the real Gaussian. For amplitudes $A \leq 2$ moving Gaussians disperse away; figure 4.22 shows this for $\theta = 2,\ A = 2,\ h = 0.25$. For larger amplitudes the moving Gaussian has a more complex splitting; figure 4.23. We suspect this is due to insufficient modelling resolution.

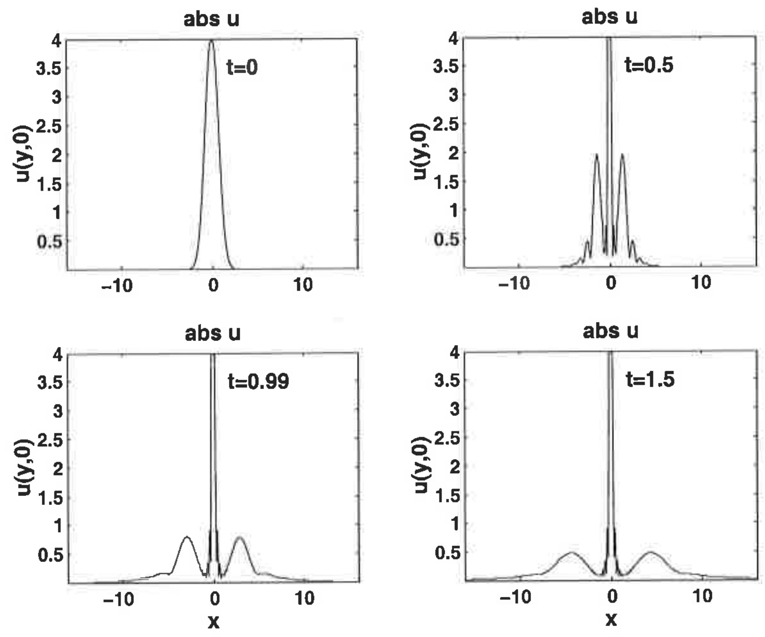

Figure 4.18: The real Gaussian initial condition, equation (4.4), sheds, narrows and grows for $A > 2$, $A = 4$ in this case. This 2D plot is a cross section through the y-axis.

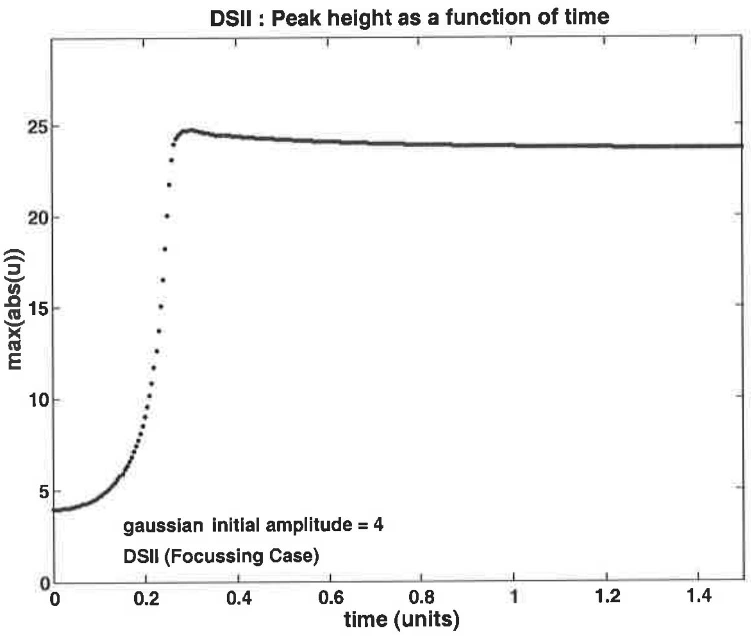

Figure 4.19: The tip height plot of the simulation in figure 4.18, above.

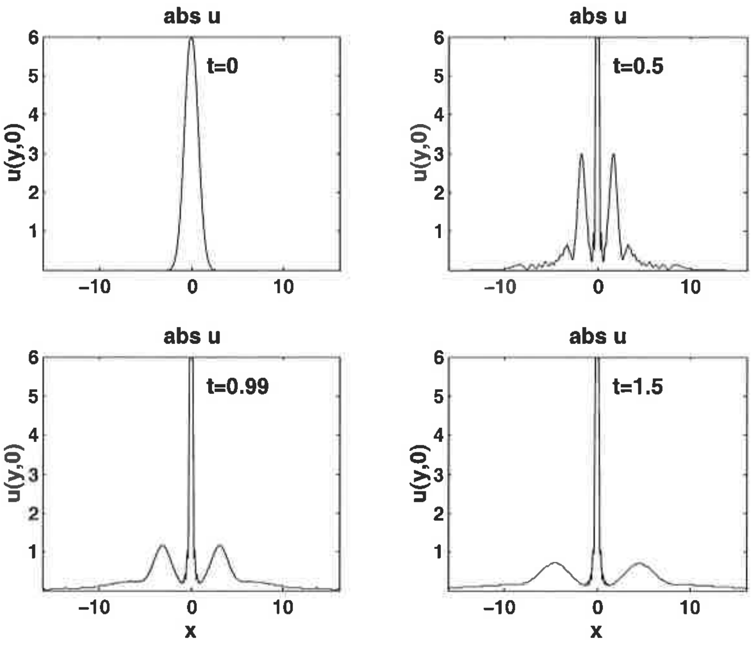

Figure 4.20: The real Gaussian initial condition, equation (4.4), sheds, narrows and grows for $A > 2$, $A = 6$ in this case. This 2D plot is a cross section through the y-axis.

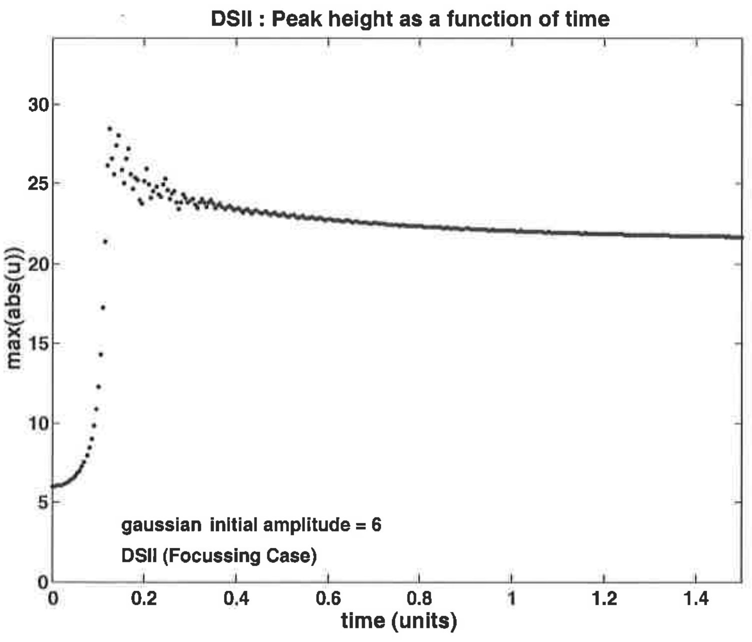

Figure 4.21: The tip height plot of the simulation in figure 4.18, above. Compare figure 4.19.

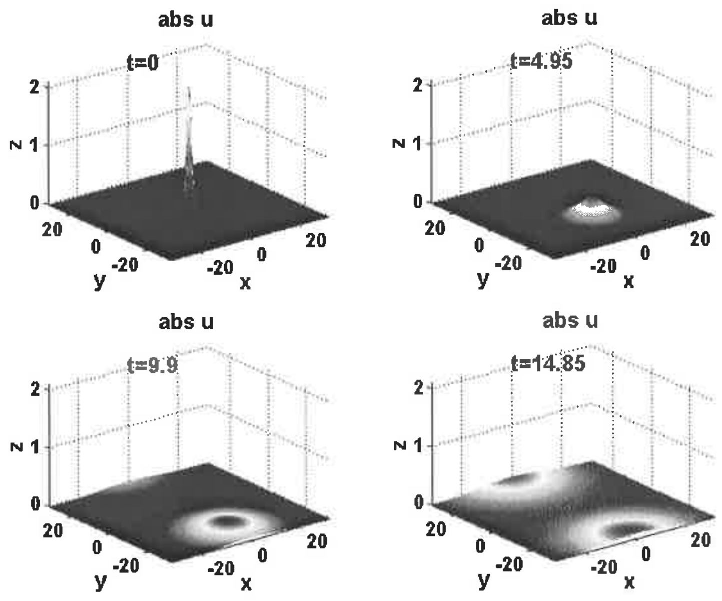

Figure 4.22: The moving Gaussian initial condition, equation (4.6), travels and diffuses for initial amplitudes $A \leq 2$.

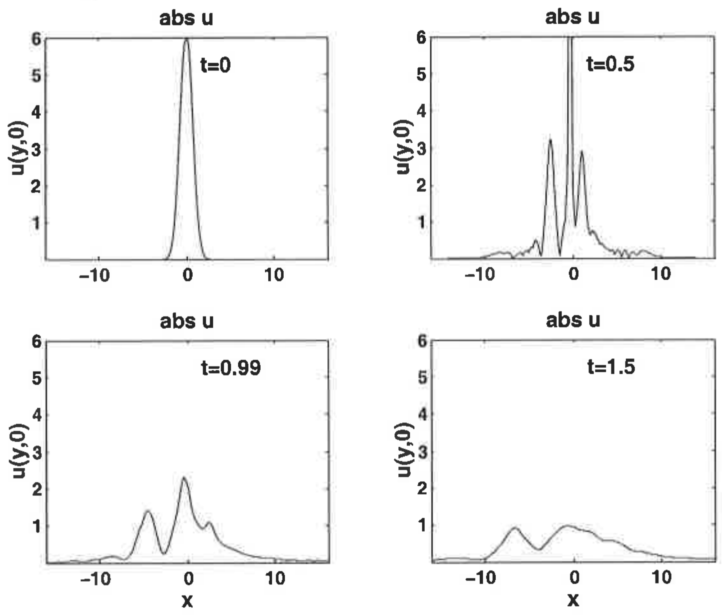

Figure 4.23: The moving Gaussian initial condition, equation (4.6), breaks into many peaks, then diffuses away when the initial amplitude is large, $A = 6$ in this case. Compare the real (static) Gaussian of figure 4.19 which grows to a single (apparently) stable peak.

Chapter 5

Conclusion and further work

5.1 Conclusion

The NLS and DSII integrable equations were successfully modelled using both the split step Fourier method and fourth order Runge-Kutta method. Results from the NLS confirmed that soliton waves do not change after collision, and that Gaussian initial wave forms shed and evolve into solitons. From modelling the DSII equation we have learned that lumps and rational solitons can also interact and cross over without change in shape. When the amplitude (but not the speed) of the lump is increased the wave form grows, and may or may not blow up; we were unable to confirm the asymptotic state in this case. For Gaussian initial conditions we observed that a complex parameter which scales with y (or x) position evolves into a moving wave which either diffuses or splits into some number of wavelets depending on amplitude. Gaussian initial conditions with constant or no complex part do not travel, but evolve *in situ*. If the initial amplitude is small, then the initial disturbance diffuses away, but if the initial amplitude is large then the disturbance sheds some energy and narrows into a single large stable structure. The form of this structure is not known, and the stability may be a consequence of the limited model resolution. The DSI equation was not modelled successfully.

5.2 Further work on the DSII equation

The relationship between speed and amplitude of lumps should be investigated to try to understand if the initial conditions are growing into stable structures or if they are blowing up. We would like to study the break up of moving Gaussian type initial conditions with high resolution models to see if lumps or other coherent structures evolve.

5.3 Further work on the DSI equation

The numerical model for the DSI is advanced but still not working. We suspect the interpretation of the function V, equation (2.70). Peter White[14] suggests using the analytic solution to calculate ϕ_ξ and ϕ_η at the boundary rather than at $-\infty$. The code does require some inspection because we feel there is some incorrect scaling. Figure 5.1 shows the model running with the boundary functions reduced by $1/(2\sqrt{2})$ which slows down the blow up.

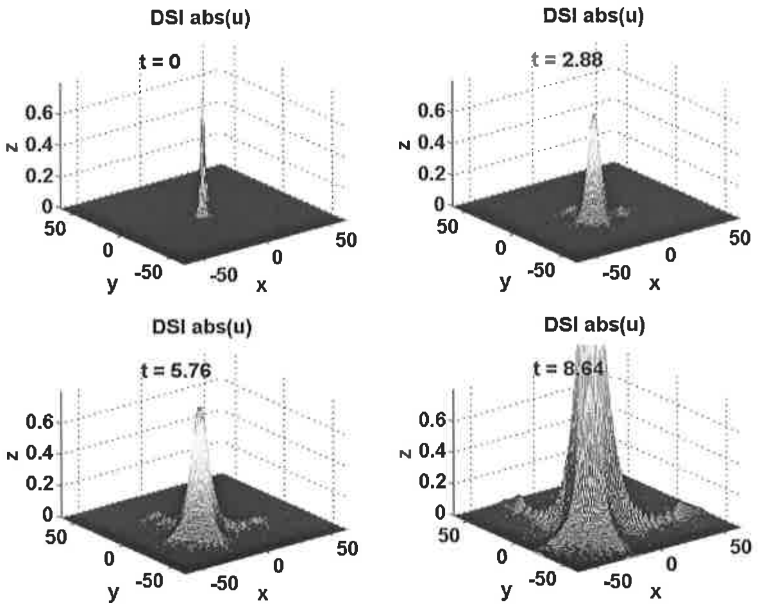

Figure 5.1: The result of incorrectly modelling the DSI equation with a dromion initial condition, equation (1.10) and corresponding boundary conditions, equations (2.88, 2.89). The initial condition at $t = 0$ should propagate without change. Correction of the code is left for future work.

Acknowledgments

I would like to acknowledge Beatrice Pelloni for inspiration and guidance; Thanasis Fokas for his time and enlightening discussions on the results of these models; Peter White for conversation by email; the patience of Mike Baines and Peter Sweby; the support and friendship of all those in the Reading University Maths Department, especially Sue Davis for her charm and those lifts in in the morning; and Hazel Butters for tolerating my student lifestyle this year.

Appendix A

Practical considerations for numerical Fourier transforms

A.1 Introduction

In this appendix we aim to give advice on the use of fast Fourier transforms (FFT) by drawing on our own experience. It is a confusing subject, but two excellent books for learning about Fourier transforms are Numerical Recipes[13] by W.H. Press *et al* and Spectral Methods in Matlab[12] by L.N. Trefethen.

A.2 The Fourier transform

The continuous form of the Fourier transform and its inverse, for some function g(x) are

$$G(\mu) = \int_{-\infty}^{\infty} g(x) e^{2\pi i \mu x}\, dx \tag{A.1}$$

$$g(x) = \int_{-\infty}^{\infty} G(\mu) e^{-2\pi i \mu x}\, d\mu \tag{A.2}$$

where μ is the spatial frequency and has units m^{-1}. These transforms are continuous and contain infinite frequencies. However, in the practical world of computer modelling both the initial function $g(x)$ and its Fourier transform are discrete (sampled). The discrete Fourier transform and its (discrete) inverse can be written in many forms, here we write

$$G(\mu_m) = h \sum_{j=0}^{M-1} g(x_j) e^{ix_j \mu_m} \tag{A.3}$$

$$g(x_j) = \frac{1}{M} \sum_{m=-\frac{M}{2}}^{m=+\frac{M}{2}} G(\mu_m) e^{-ix_j \mu_m} \tag{A.4}$$

where

$$x_j = jh \quad \text{(h is the sampling interval)} \tag{A.5}$$

$$h = \frac{\text{domain size}^1}{M} \tag{A.6}$$

$$j = 0,\ 1 \ldots M-1 \quad \text{(M is the number of samples)} \tag{A.7}$$

$$\mu_m = \frac{m2\pi}{Mh} \tag{A.8}$$

$$m = -\frac{M}{2}, -\frac{M}{2}+1, \ldots \frac{M}{2} \tag{A.9}$$

Using these definitions, we note that the exponential term $e^{ix_j \mu_m}$ can be written

$$e^{ijh\frac{m2\pi}{Mh}} = e^{2\pi ij\frac{m}{M}} \tag{A.10}$$

which is done in Numerical Recipes[13], but not here because it confuses the derivative terms. Trefethen[12] chooses to rescale everything to a domain size of 2π so that $h = \frac{2\pi}{M}$, always, then

$$\mu_m = \frac{m2\pi}{Mh} = \frac{m2\pi M}{M2\pi} = m \tag{A.11}$$

[1] If we define the domain $[-P, P]$ then the domain size $= 2P$ and $j = -\frac{M}{2} \to \frac{M}{2} - 1$

and the exponential term $e^{ix_j\mu_m}$ becomes simply

$$e^{ix_j m} \tag{A.12}$$

and Trefethen[12] writes

$$G(\mu_m) = h\sum_{j=0}^{M-1} g(x_j)e^{-imx_j} \tag{A.13}$$

as the discrete Fourier transform (to compare (A.13) to that in [12], translate the symbols: our $m \equiv$ Trefethen's k).

We do not use the efficiencies suggested by Trefethen either. We stick to

$$G(\mu_m) = h\sum_{j=0}^{M-1} g(x_j)e^{ix_j\mu_m} \tag{A.14}$$

$$g(x_j) = \frac{1}{M}\sum_{m=-\frac{M}{2}}^{m=+\frac{M}{2}} G(\mu_m)e^{-ix_j\mu_m} \tag{A.15}$$

and remember that the μ_m has the 2π in it.

Then

$$\text{x space} \qquad \mu \text{ space} \tag{A.16}$$

$$g(x_j)_x \equiv i\mu_m G(\mu_m) \tag{A.17}$$

$$g(x_j)_{xx} \equiv -\mu_m^2 G(\mu_m) \tag{A.18}$$

Turning to computer code, the fast Fourier transform (FFT) and inverse fast Fourier transform (iFFT) are efficient algorithms[13] for these transforms. The functions `fft` and `ifft` in Matlab are used with the following syntax

$$G(\mu_m) = \texttt{fft}\,(g(x_j)) \qquad \text{compare } G(\mu_m) = h\sum_{j=0}^{M-1} g(x_j)e^{ix_j\mu_m} \tag{A.19}$$

$$g(x_j) = \texttt{ifft}\,(G(\mu_m)) \qquad \text{compare } g(x_j) = \frac{1}{M}\sum_{m=0}^{M-1} G(\mu_m)e^{-ix_j\mu_m} \tag{A.20}$$

where $g(x_j)$ is a one dimensional array built from equation (A.5), for example

```
x = (domainsize/M)*(-M/2:M/2-1);
```

$G(\mu_m)$ however, is returned from the iFFT routine in the order that corresponds to wavenumbers

$$\mu_0,\ \mu_1 \cdots \mu_{\frac{M}{2}},\ \mu_{-\frac{M}{2}+1} \cdots \mu_{-1} \tag{A.21}$$

Essentially shifting the negative frequencies to the top end of the array...

$$\underbrace{\mu_{-\frac{M}{2}+1} \cdots \mu_{-1},}_{Shift\ \longrightarrow}\ \mu_0,\ \mu_1 \cdots \mu_{\frac{M}{2}} \tag{A.22}$$

Therefore, to use the Fourier method in finding the derivative $g(x_j)_x = \texttt{ifft}(i\mu_m G(\mu_m))$, μ_m needs to be in the order as above(A.21).

Matlab provides a function `fftshift` which performs shifting either way, or one can simply define the array μ_m in this shifted order in the original definition (recommended)

```
mum = [0:M/2 -M/2+1:-1]*2*pi/domainsize;
```

Then, for example, `plot(fftshift(mum), fftshift(G))` will plot $G(\mu_m)$ correctly.

There is an additional subtlety here. For odd derivatives there is a loss of symmetry and we have to set $G(\mu_{\frac{M}{2}}) = 0$. In other words we can define two arrays of the spatial frequencies μ_m, `mumodd = [0:M/2-1 0 -M/2+1:-1]*2*pi/domainsize;` for odd derivatives and `mumeven = [0:M/2 -M/2+1:-1]*2*pi/domainsize;` for calculating even derivatives. See chapter 3 of [12].

A.3 Choosing h

The sampling interval, h is $h = \frac{\text{domain size}}{M}$ and defining $\frac{1}{h}$ as *the* spatial frequency, the spatial frequencies, μ_m, run from

$$\begin{aligned} \mu_m &= -\frac{M}{2}\frac{2\pi}{Mh} \text{ to } +\frac{M}{2}\frac{2\pi}{Mh} && \text{(A.23)} \\ &= -\frac{2\pi}{2h} \text{ to } \frac{2\pi}{2h} && \text{(A.24)} \end{aligned}$$

This means that $\frac{1}{2h}$ is the maximum frequency in the transformed function $G(\mu_m)$ and is equal to half of *the* spatial frequency, $\frac{1}{h}$. This is the Nyquist Sampling Theorem.

If $g(x_j)$ contains spatial frequencies higher than $\frac{1}{2h}$ then there will be errors in the split step Fourier scheme. See [13] for an excellent description of *aliasing*.

This leads to errors if the wave form in our model evolves higher frequencies, for example when rising into a sharp peak c.f. figure 4.20.

A.4 Extension to two dimensions

All this extends to two dimensions, for a domain $x \times y = [-P, P] \times [-Q, Q]$ with $M \times N$ samples,

$$\begin{aligned} G(\mu_m, \nu_n) &= h_x h_y \sum_n \sum_m g(x_j, y_k) e^{i(\mu_m x_j + \nu_n y_k)} && \text{(A.25)} \\ g(x_j, y_k) &= \frac{1}{MN} \sum_k \sum_j G(\mu_m, \nu_n) e^{-i(\mu_m x_j + \nu_n y_k)} && \text{(A.26)} \end{aligned}$$

where

$$\begin{aligned}
x_j &= jh_x && \text{(A.27)}\\
j &= -\frac{M}{2}\dots\frac{M}{2}-1 && \text{(A.28)}\\
h_x &= \frac{2P}{M} && \text{(A.29)}\\
\mu_m &= \frac{2\pi m}{h_x M} = \frac{2\pi m}{2P} && \text{(A.30)}\\
m &= 0\dots\frac{M}{2}, -\frac{M}{2}+1\dots-1 && \text{(A.31)}\\
y_k &= kh_y && \text{(A.32)}\\
k &= -\frac{N}{2}\dots\frac{N}{2}-1 && \text{(A.33)}\\
h_y &= \frac{2Q}{N} && \text{(A.34)}\\
\nu_n &= \frac{2\pi n}{h_y N} = \frac{2\pi n}{2Q} && \text{(A.35)}\\
n &= 0\dots\frac{N}{2}, -\frac{N}{2}+1\dots-1 && \text{(A.36)}
\end{aligned}$$

The interested reader will find the Matlab functions `fft2`, `ifft2` and `fftshift` very useful for performing two dimensional Fourier transforms in Matlab.

Appendix B

The serpentine function as a filter

The Serpentine function is any cubic of the form

$$x^2y + aby - a^2x = 0 \qquad ab > 0 \tag{B.1}$$

to design a filter for use in a Fourier method it is desirable that there are no fast changes, or equivalently no high frequencies. The desired form of the filter will have zero gradient at the start, a maximum gradient of one and zero gradient at the end. To control the gradient, the derivative is required, rearranging (B.1),

$$y = \frac{a^2x}{x^2 + ab} \tag{B.2}$$

so

$$\begin{aligned}\frac{dy}{dx} &= \frac{a^2}{(x^2+ab)} - \frac{2x^2a^2}{(x^2+ab)^2} \\ &= \frac{a^3b - a^2x^2}{(x^2+ab)^2} = 0 \text{ at the turning points}\end{aligned}$$

that is,

$$\begin{aligned}
a^3b - a^2x^2 &= 0 \\
a^2x^2 &= a^3b \\
x^2 &= ab \\
x &= \pm\sqrt{ab} \text{ at the turning points}
\end{aligned}$$

and we find $y = \pm\frac{1}{2}a\sqrt{\frac{a}{b}}$.

For a 45° slope at $(0,0)$, putting $x = 0$ and $y = 0$ into $\frac{dy}{dx}$

$$\begin{aligned}
\frac{dy}{dx}|_{0,0} &= \frac{a}{b} \\
&= 1 \text{ for } 45^\circ
\end{aligned}$$

so any $a = b$ will do. When $a = b$, the turning points are $(a, \frac{a}{2})$ and $(-a, -\frac{a}{2})$.

Figure (B.1) displays a plot of the function (B.1) with a = 1. This shape was used to create the filter of figure (2.3).

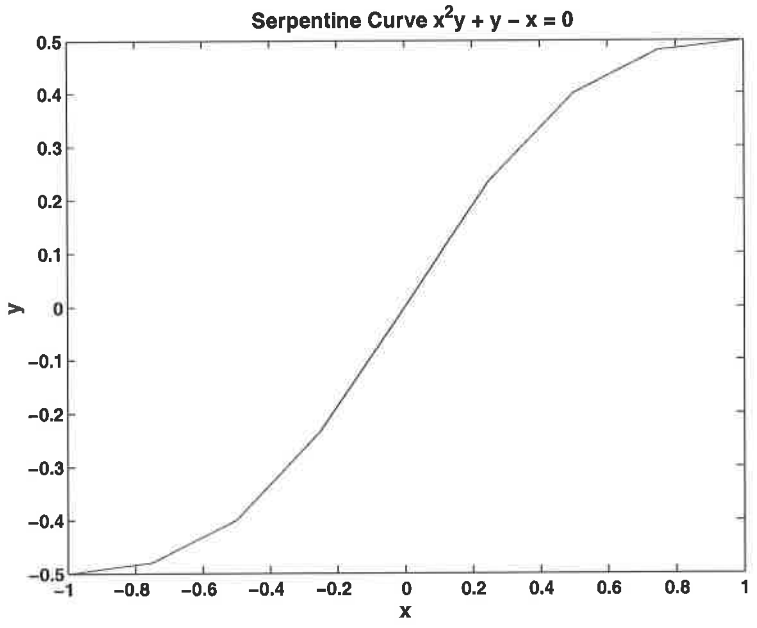

Figure B.1: The curve used as a filter to stop waves wrapping around the boundary.

Appendix C

Matlab listings

C.1 NLS by split step

```
%Split step fourier method for the nonlinear schrodinger equation
%M.McC June 2002

% Set up grid and initialise variables
  N = 512;
  dt = 0.005;
  hdt = 0.5*dt;

  tmax = dt*8000;
  nplt = floor((tmax/25)/dt);
  nmax = round(tmax/dt);
  gridscale = 80;
  sigma = -1;    %dispersive term
  lambda = -1;  %nonlinear term

  x = (2*gridscale/N)*(-N/2:N/2-1)';
  k = (pi/gridscale)*[0:N/2-1 0 -N/2+1:-1]';
  ik2sigma = i*k.^2*sigma;

 %Choose the initial conditions
  u = 1.0*exp(-0.2*((x-40).^2)).*exp(i*x*2)+1.0*exp(-0.2*((x+20).^2)).*exp(-i*x/1.0);
  zmax = max(abs(u));

% Solve PDE and plot results:

  udata = abs(u);
  tdata = 0;

  for n = 1:nmax

    t = n*dt;

   %Solve nonlinear part (half time step)
    u = u.*exp(i*lambda*abs(u).^2*hdt);

   %Solve linear part in fourier space (full time step)
    v = fft(u);
    v = v.*exp(-ik2sigma*dt);     %analytical solution in fourier space

   %Return to real space and solve final half step
    u = ifft(v);
    u = u.*exp(i*lambda*abs(u).^2*hdt);

    if mod(n,nplt) == 0
      udata = [udata abs(u)]; tdata = [tdata t];
    end

  end

 %Graphical Output
  figure(2);
  waterfall(x,tdata,udata'), colormap(1e-6*[1 1 1]); view(-20,25)
  xlabel x, ylabel t, axis([-gridscale gridscale 0 tmax 0 zmax]), grid off
  set(gca,'ztick',[0 2000]), pbaspect([1 1 .13])
```

C.2 NLS by 4th order Runge-Kutta method and integrating factor

```
%RK4 method for the nonlinear schrodinger equation
%M.McC June 2002

% Set up grid and initialise variables
  N = 256;
  dt = 0.005;
  hdt = 0.5*dt;

  tmax = dt*8000;
  nplt = floor((tmax/25)/dt);
  nmax = round(tmax/dt);
  gridscale = 80;
  sigma = -1;   %dispersive term
  lambda = -1;  %nonlinear term

  x = (2*gridscale/N)*(-N/2:N/2-1)';
  k = (pi/gridscale)*[0:N/2-1 0 -N/2+1:-1]';
  ik2sigma = i*k.^2*sigma;

 %Choose the initial conditions
  u = sqrt(2.0)*sech(x+10.0).*exp(-i*x/2.0)+0.8*sqrt(2.0)*sech(0.8*(x-10)).*exp(i*x/2.0);
  zmax = max(abs(u));

% Solve PDE and plot results:
  udata = abs(u);
  tdata = 0;
  v = fft(u);
  ik2sigma = i*k.^2*sigma;
  U = v; %Etm = 1 when t = 0

  for n = 1:nmax

    t   = n*dt;
    g   = i*lambda*dt;
    Ep  = exp(dt*ik2sigma/2);
    Ep2 = Ep.^2;
    Em  = exp(-dt*ik2sigma/2);
    Em2 = Em.^2;
    Etp = exp(t*ik2sigma);
    Etm = exp(-t*ik2sigma);

    a = g.*Etp.*     fft(  abs( ifft( Etm.*      U     ) ).^2  .* (ifft( Etm.*      U      )) );
    b = g.*Etp.*Ep .* fft(  abs( ifft( Etm.*Em.* (U+a/2)) ).^2  .* (ifft( Etm.*Em.* (U+a/2) )) );
    c = g.*Etp.*Ep .* fft(  abs( ifft( Etm.*Em.* (U+b/2)) ).^2  .* (ifft( Etm.*Em.* (U+b/2) )) );
    d = g.*Etp.*Ep2.* fft(  abs( ifft( Etm.*Em2.*(U+c)   ) ).^2  .* (ifft( Etm.*Em2.*(U+c)    )) );
    U = U + (a + 2*(b+c) + d)/6;

    if mod(n,nplt) == 0
      udata = [udata abs(ifft(U.*Etm))]; tdata = [tdata t];
    end
  end

 %Graphical Output
  figure(2);
  waterfall(x,tdata,udata'), colormap(1e-6*[1 1 1]); view(-20,25)
  xlabel x, ylabel t, axis([-gridscale gridscale 0 tmax 0 zmax]), grid off
  set(gca,'ztick',[0 2000]), pbaspect([1 1 .13])
```

C.3 KdV by 4th order Runge-Kutta method and integrating factor

```
% p27.m - Solve KdV eq. u_t + uu_x + u_xxx = 0 on [-pi,pi] by
%         FFT with integrating factor v = exp(-ik^3t)*u-hat.

% Set up grid and two-soliton initial data:
  N = 256; dt = .4/N^2; x = (2*pi/N)*(-N/2:N/2-1)';
  A = 25; B = 16;
  u = x.*0;
  u = 1250*exp(-20*((x+2).^2))
  v = fft(u); k = [0:N/2-1 0 -N/2+1:-1]'; ik3 = 1i*k.^3;
  U = v; %Etm = 1 when t = 0

% Solve PDE and plot results:
  tmax  = 0.012; nplt = floor((tmax/30)/dt); nmax = round(tmax/dt);
  udata = u; tdata = 0; h = waitbar(0,'please wait...');
  for n = 1:nmax
    t   = n*dt;
    g   = -.5i*dt*k;
    Ep  = exp(dt*ik3/2);
    Ep2 = Ep.^2;
    Em  = exp(-dt*ik3/2);
    Em2 = Em.^2;
    Etp = exp(t*ik3);
    Etm = exp(-t*ik3);

    a = g.*     Etm.*fft(( ifft(Etp.*      U      ) ).^2);
    b = g.*Em  .*Etm.*fft(( ifft(Etp.*Ep.* (U+a/2)) ).^2);      % 4th-order
    c = g.*Em  .*Etm.*fft(( ifft(Etp.*Ep.* (U+b/2)) ).^2);      % Runge-Kutta
    d = g.*Em2.*Etm.*fft(( ifft(Etp.*Ep2.*(U+c  )) ).^2);
    U = U + (a + 2*(b+c) + d)/6;

    if mod(n,nplt) == 0
      u = abs(ifft(U.*Etp)); waitbar(n/nmax)
      udata = [udata u]; tdata = [tdata t];
    end
  end
  waterfall(x,tdata,udata'), colormap(1e-6*[1 1 1]); view(-20,25)
  xlabel x, ylabel t, axis([-pi pi 0 tmax 0 2000]), grid off
  set(gca,'ztick',[0 2000]), close(h), pbaspect([1 1 .13])
```

C.4 DSII by split step

```
%Split step fourier method for the DSII equation
%DSII equation as per Peter White's paper
%M.McC 12th June 2002

% Set up grid and initial data:

 %initialise nonlinear coefficient
  sigma = -1.0; %-1  NOTE : sigma = -1 focusses, 1 defocusses

 %initialise time variables
  dt     = 0.05;
  hdt    = 0.5*dt;
  steps  = 200;
  frames = 20;

 %initialise space variables
  M         = 512/2/2/2;
  N         = 512/2/2/2;
  gridscale = 32*2;
  x         = (gridscale/M)*(-M/2:M/2-1);
  y         = (gridscale/M)*(-N/2:N/2-1);
  [xx, yy]  = meshgrid(x,y);

 %initialise wavenumbers
  kx        = [0:M/2 -M/2+1:-1]; % kxuneven = [0:M/2-1 0 -M/2+1:-1]';
  ky        = [0:N/2 -N/2+1:-1]; % kyuneven = [0:N/2-1 0 -N/2+1:-1]';
  [kxx,kyy] = meshgrid(kx,ky);
  kxx       = kxx*2*pi/gridscale; %scaling
  kyy       = kyy*2*pi/gridscale; %scaling

 %initial conditions
  xscale  = 1;
  yscale  = 1;
  xoffset = 0;
  yoffset = 0;
  A       = 2;
  t       = 0; tvector = [t];
  tscale  = 1;
  toffset = 0;
  alpha   = 0;
  beta    = 2;
  p       = 1;
  n       = 0;

 %select initial function
  u = A*onelumpsoliton(xx,yy,xscale,yscale,xoffset,yoffset,beta,p,alpha,t,tscale,toffset);

 %initialise loop variables
  tmax           = dt*steps+t;
  nplt           = floor(((tmax-t)/frames)/dt);
  nmax           = round((tmax-t)/dt);
  bmnfactor      = 2*sigma*(kxx.^2)./(kxx.^2+kyy.^2);
  bmnfactor(1,1) = 0; %divide by zero term = zero
  linear_exp     = exp(i/2*(kyy.^2-kxx.^2)*dt);

 %loop
  for n = 1:nmax

   %half step
    bmn    = fft2(abs(u).^2);
    phixjk = ifft2(bmnfactor.*bmn);
    u      = u.*exp(i*hdt*(sigma*abs(u).^2-phixjk));

   %full step
    amn = fft2(u);
    aprimemn = linear_exp.*amn;
    u = ifft2(aprimemn);

   %half step
    bmn = fft2(abs(u).^2);
    phixjk = ifft2(bmnfactor.*bmn);
    u = u.*exp(i*hdt*(sigma*abs(u).^2-phixjk));

    if mod(n,nplt) == 0 %then record the solution and the statistics
      mesh(xx,yy,abs(u))
    end %if

  end
```

C.5 DSII by 4th order Runge-Kutta method and integrating factor

Only the central loop is given here

```
  bmnfactor      = 2*sigma*(kxx.^2)./(kxx.^2+kyy.^2);
  bmnfactor(1,1) = 0; %divide by zero term = zero
  linear_exp     = exp(-i/2*(kxx.^2-kyy.^2)*dt);
  ikl     = -0.5*i*(kxxeven.^2-kyyeven.^2);
  Exmt    = exp(-ikl*t);
  V       = fft2(u).*Exmt;

 %loop
  for n = 1:nmax

   %precondition
    if (usefilter == 1)
        u = u.*filter;
    end

    %calculate phixjk from V
    u = ifft2(V.*Expt);
    bmn = fft2(abs(u).^2);
    phixjk = ifft2(bmnfactor.*bmn);

   %variables used in the RK4 method
    Expt    = exp(+ikl*t);
    Expthdt = exp(+ikl*(t+hdt));
    Exptdt  = exp(+ikl*(t+dt));
    Exmt    = exp(-ikl*t);
    Exmthdt = exp(-ikl*(t+hdt));
    Exmtdt  = exp(-ikl*(t+dt));

    % RK4 method
    a = dt*Exmt    .*fft2( i*(sigma*abs(u).^2 - phixjk).*u );

    u = ifft2((V+a/2).*Expthdt );
    b = dt*Exmthdt.*fft2( i*(sigma*abs(u).^2 - phixjk).*u );

    u = ifft2((V+b/2).*Expthdt );
    c = dt*Exmthdt.*fft2( i*(sigma*abs(u).^2 - phixjk).*u );
    u = ifft2((V+c).*Exptdt );
    d = dt*Exmtdt .*fft2( i*(sigma*abs(u).^2 - phixjk).*u );

    V = V + (a+2*(b+c)+d)/6;
    t = t+dt %move on to next time step

    if mod(n,nplt) == 0 %then record the solution
      u = ifft2(V.*Expt);
     %draw frames
      figure(fig1);
      mesh(xx,yy,abs(u))
    end
  end %main for loop
```

C.6 DSI by split step Fourier method

Only the central loop is given here. Beware this code is not running correctly see section 2.4.1.

```
 %select initial function
  u = A*onelumpsoliton(xx,yy,xscale,yscale,xoffset,yoffset,beta,p,alpha,t,tscale,toffset);

 %initialise loop variables
  tmax          = dt*steps+t;
  nplt          = floor(((tmax-t)/frames)/dt);
  nmax          = round((tmax-t)/dt);
  bmnfactor     = 2*sigma*(kyy.^2)./(kxx.^2+kyy.^2);
  bmnfactor(1,1) = 0; %divide by zero = zero
  linear_exp    = exp(-i/2*(kxx.^2+kyy.^2)*dt); %note '+' for DSI

  for n = 1:nmax

%%% half step (non-linear part) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    bmn = fft2(abs(u).^2);
    dummy1 = kyyuneven./kxxuneven.*bmn;
    dummy1(:,1) = 0; %pad the m=0 column with zeroes
    dummy1(:,M/2+1) = 0; %pad the m=0 column with zeroes
    Vterm1 = real(ifft2(dummy1));
    %subtract the j = 1 ie x(j) = -P column from every column in Vterm1
    dummy1 = Vterm1(:,1);
    for s = 1:M
       Vterm1(:,s) = Vterm1(:,s)-dummy1;
    end
    % m = 0 part....
    dummy1 = real(ifft(i*2*pi*(kyuneven)'.*bmn(:,1)));
    dummy2 = dummy1;
    for s = 2:M
        dummy2 = [dummy2 dummy1];
    end
    for s = 1:M
        dummy2(s,:) = dummy2(s,:).*(x+P); %multiply each row element by the x vector elements
    end
    Vterm1 = Vterm1+dummy2/M;
    %%%%%%%% end of Vterm1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%% now do Vterm2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    dummy1 =kxxuneven./kyyuneven.*bmn;
    dummy1(1,:) = 0; %pad the n=0 row with zeroes
    dummy1(N/2+1,:) = 0; %pad the n=0 row with zeroes
    Vterm2 = real(ifft2(dummy1));
    %subtract the k = 1 ie y(k) = -P row from every row in Vterm2
    dummy1 = Vterm2(1,:);
    for s = 1:N
      Vterm2(s,:) = Vterm2(s,:)-dummy1;
    end
    % n = 0 part....
    dummy1 = real(ifft(i*2*pi*kxuneven.*bmn(1,:)));
    dummy2 = dummy1;
    for s = 2:N
        dummy2 = [dummy2; dummy1];
    end
    for s = 1:N
        dummy2(:,s) = dummy2(:,s).*(y'+P); %multiply each column element by the y vector elements
    end
    Vterm2 = Vterm2+dummy2/(N);
    V = 0.5*sigma*(real(Vterm1) + real(Vterm2));
    [xtrack ytrack] = meshgrid(phixbo(x,P,t),phiybo(P,y,t));
    V = V+(xtrack+ytrack)/2/sqrt(2);
    %%%%%%%% now V is calculated, step forward the half step in time %%%%%%%%
    u = u.*exp(-V*hdt);
    t = t + hdt;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% full step (linear part) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    amn = fft2(u);
    aprimemn = linear_exp.*amn;
    u = ifft2(aprimemn);
    t = t + dt;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% half step (non-linear part) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    bmn = fft2(abs(u).^2);
    dummy1 =kyyuneven./kxxuneven.*bmn;
    dummy1(:,1) = 0; %pad the m=0 column with zeroes
    dummy1(:,M/2+1) = 0; %pad the m=0 column with zeroes
    Vterm1 = real(ifft2(dummy1));
    %subtract the j = 1 ie x(j) = -P column from every column in Vterm1
    dummy1 = Vterm1(:,1);
    for s = 1:M
      Vterm1(:,s) = Vterm1(:,s)-dummy1;
    end
    % m = 0 part....
    dummy1 = real(ifft(i*2*pi*kyuneven'.*bmn(:,1)));
    dummy2 = dummy1;
    for s = 2:M
        dummy2 = [dummy2 dummy1];
    end
    for s = 1:M
        dummy2(s,:) = dummy2(s,:).*(x+P); %multiply each row element by the x vector elements
    end
    Vterm1 = Vterm1+dummy2/M;
    %%%%%%%% end of Vterm1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%% now do Vterm2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    dummy1 =kxxuneven./kyyuneven.*bmn;
    dummy1(1,:) = 0; %pad the n=0 row with zeroes
    dummy1(N/2+1,:) = 0; %pad the n=0 row with zeroes
    Vterm2 = real(ifft2(dummy1));
    %subtract the k = 1 ie y(k) = -P row from every row in Vterm2
    dummy1 = Vterm2(1,:);
    for s = 1:N
      Vterm2(s,:) = Vterm2(s,:)-dummy1;
    end
```

```
  % n = 0 part....
  dummy1 = real(ifft(i*2*pi*kxuneven.*bmn(1,:)));
  dummy2 = dummy1;
  for s = 2:N
      dummy2 = [dummy2; dummy1];
  end
  for s = 1:N
      dummy2(:,s) = dummy2(:,s).*(y'+P); %multiply each column element by the y vector elements
  end
  Vterm2 = Vterm2+dummy2/N;
  V = 0.5*sigma*(real(Vterm1) + real(Vterm2));
  [xtrack ytrack] = meshgrid(phixbo(x,P,t),phiybo(P,y,t));
  V = V+(xtrack+ytrack);
  %%%%%%%% now V is calculated, step forward the half step in time
  u = u.*exp(-V*hdt);
  t = t + hdt;

  if mod(n,nplt) == 0 %then record the solution and the statistics
   %draw frames
    figure(fig1);
    mesh(xx,yy,abs(u))
  end %if

end %for
```

Bibliography

[1] M.J. Ablowitz and P.A. Clarkson. *Solitons, Nonlinear Evolution Equations and Inverse Scattering.* Cambridge University Press, 1992.

[2] D.E. Pelinovsky A.S. Fokas and C.Sulem. Interaction of lumps with a line soliton for the DSII equation. *Physica D*, 152-153:189 – 198, 2001.

[3] C. Besse and C.H. Bruneau. Numerical study of elliptic-hyperbolic Davey-Stewartson system: Dromions simulation and blow-up. *Mathematical Models and Methods in Applied Sciences*, 8:1363 – 1386, 1998.

[4] D. Crighton. Applications of KdV. *in KdV '95, M. Hazewinkel, H. Capel and E. de Jager, eds., Kluwer*, 1991:2977 – 2984, 1995.

[5] A.S. Fokas. Private discussion.

[6] A.S. Fokas. On the integrability of linear and non-linear partial differential equations. *Journal of Mathematical Physics*, 41:4188 –, 2000.

[7] A.S. Fokas and P.M. Santini. Dromions and a boundary value problem for the Davey-Stewartson I equation. *Physics D*, 37:99 – 130, 1990.

[8] Alan C. Newell. *Solitons in Mathematics and Physics.* Siam, 1985.

[9] Katsuhiro Nishinari and Tetsu Yajima. Time evolution of gaussian type initial conditions associated with the Davey-Stewartson equations. *J. Phys. A: Math. Gen.*, 29:4237 – 4245, 1996.

[10] Web page. Famous curves index.
http://www-gap.dcs.st-and.ac.uk/~history/Curves/Curves.html.

[11] C.L. Schultz and M.J. Ablowitz. Strong coupling limit of certain nonlinear evolution equations II. Numerical nonlinear schrodinger equation. *J. Comp. Phys.*, 55:203 – 230, 1984.

[12] Lloyd N Trefethen. *Spectral Methods in Matlab.* Siam, 2000.

[13] W.H. Press B.P. Flannery S.A. Teukolsky W.T. Vetterling. *Numerical Recipes.* Cambridge University Press, 1986.

[14] P. White. Private discussion by email.

[15] P. White and J.A.C. Weideman. Numerical simulation of solitons and dromions in the Davey-Stewartson system. *Journal of Mathematics and Computers in Simulation*, 37:469 – 479, 1994.