Soliton Solutions of the Complex Ginzburg Landau Equation

By Max Proft
u5190335

Contents

Sections I can already write **3**

Abstract **4**

Acknowledgements **5**

1 Introduction **6**
- 1.1 Things to include: 6
- 1.2 section 6
  - 1.2.1 subsection 6
  - 1.2.2 subsection 2 6
- 1.3 Second Section 6

2 Derivations of the CGLE **7**
- 2.1 For a Quantum System 7
- 2.2 For a Classical System 7

3 Types of solitons **8**
  - 3.0.1 Analytic Solutions 8
  - 3.0.2 Other Solutions 8

4 Mathematical Theory Behind Solving the CGLE **9**
- 4.1 A Comparison of Different Methods 9
- 4.2 Split Step Fourier and Runge-Kutta Method 9
  - 4.2.1 Linear evolution (Fourier) 9
  - 4.2.2 Nonlinear evolution (Runge-Kutta) 10
  - 4.2.3 The relationship between a Fourier Transform and a Discrete Fourier Transform 11
  - 4.2.4 Combining these operations 12
- 4.3 Testing the Code 12
  - 4.3.1 Comparison with Analytic Solutions 12
  - 4.3.2 Decreasing the step size and Determining Appropriate Values . 13

5 Machine Learning Theory **14**
5.1 Different Types of Machine Learning 14
5.2 Linear Regression 14
5.3 Training and Testing the Algorithm 16

6 The result from much work **17**
6.1 The sorts of things that I found 17

Conclusion **18**

Appendix A **19**

Appendix B **20**

Sections I can already write

3.0.1, 3.0.2 - Types of solitons, both analytic and others - I will need to do the pictures later though.

4.1 - A comparison of different methods

4.2.3 - Relationship between DFT and FT

5.1 - Different types of machine learning

Abstract

Abstract goes here.

Acknowledgements

Chuck in some very cliché acknowledgements here.

Chapter 1

Introduction

1.1 Things to include:

dissipative/nonconservative
Where do we see this happening?[2]
nonintegrable, limitations of calculating by hand, the reason for computers
why searching for solutions with machine learning is needed
[1, pp. 215]

1.2 section

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisissem.

1.2.1 subsection

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisissem.

1.2.2 subsection 2

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisissem.

1.3 Second Section

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisissem.

Chapter 2

Derivations of the CGLE

2.1 For a Quantum System

2.2 For a Classical System

Chapter 3

Types of solitons

3.0.1 Analytic Solutions

3.0.2 Other Solutions

Chaotic, but spatially localised.

Chapter 4

Mathematical Theory Behind Solving the CGLE

4.1 A Comparison of Different Methods

Decide whether to use non linear, non-linear, nonlinear Finite difference, what was used initially to get it all going, is bad because you do $\delta\psi/\delta x$, so over a small time period, the $\delta\psi$ term can lose several digits of precision, not good. There are a number of other ways for this to be solved though.

For the nonlinear schrodinger equation, (author) compared various methods for solving this. Other methods are EXAMPLE, and a split step fourier method. The first is better for large, slowly changing features, the second is better for sharp peaks, and the third is better for both of these. Given the similarity between the CGLE and NLSE, this means that this should also be a reasonably good method to do this part as well.

4.2 Split Step Fourier and Runge-Kutta Method

4.2.1 Linear evolution (Fourier)

Here we are solving the equation:

$$\frac{d\psi}{dt} = A\frac{d^2\psi}{dx^2} + B\psi$$

Defining the fourier transform by:

$$\hat{\psi}(f) = \int_{-\infty}^{\infty} \psi(x)e^{-i2\pi xf}dx$$

By taking the fourier transform of both sides, the equation becomes:

$$\frac{d\hat{\psi}}{dt} = C(-2\pi i f)^2 D\hat{\psi}(f) + \hat{\psi}(f) = (C - D4\pi^2 f^2)\hat{\psi}(f)$$
$$\implies \hat{\psi}_t(f) = e^{(C-D4\pi^2 f^2)t}\hat{\psi}_0(f)$$

And so upon inverse fourier transforming, we get the solution to this DE, where $\hat{\psi}_t(f)$ is the fourier transform at time t. The Fourier transform can be done numerically with a discrete fourier transform, as described in a later subsection.

4.2.2 Nonlinear evolution (Runge-Kutta)

Here we are solving the equation:

$$\frac{d\psi}{dt} = C|\psi|^2\psi + D|\psi|^4\psi$$

The runga kutta method allows you to numerically solve a 1st order differential equation. which are of the form $\frac{dy}{dt} = f(t, y)$. Similar to Euler's method (which is basically just doing a derivative from first principles), except that the runge kutta method has better convergence than this. (it goes as h^5 as opposed to Euler's Method which is h^2, the backward euler formula, h^2, or improved euler formula which goes as h^3. Here h is the step size.

The steps for doing this are as follows: Step 1: define $f(t, y) = f(y) = C|y|^2y+D|y|^4y$ Step 2: Have an initial value for t and y, and step size and number of steps Step 3: define the following:

$$K_1 = f(t, y) = f(y)$$
$$K_2 = f(t + 0.5h, y + 0.5 * h * K_1) = f(y + h * K_1/2)$$
$$K_3 = f(t + 0.5h, y + 0.5 * h * K_2) = f(y + h * K_2/2)$$
$$K_4 = f(t + h, y + h * K_3) = f(y + h * K_3)$$

Step 4: we then progress forwards t and y in the following way:

$$t \to t + h$$

$$y \to y + (K_1 + 2K_2 + 2K_3 + K_4) * h/6$$

4.2.3 The relationship between a Fourier Transform and a Discrete Fourier Transform

I am pretty sure this entire section needs to be rewritten.
We can approximate any function with a Fourier series. (by applying periodic boundary conditions)

$$f(t) = \sum_{-\infty}^{\infty} c_n e^{int}$$

Multiplying both sides by e^{-imt} and integrating from 0 to 2π, and dividing by 2π:

$$\frac{1}{2\pi}\int_0^{2\pi} f(t)e^{-imt}dt = \frac{1}{2\pi}\sum_{n=-\infty}^{\infty} c_n \int_0^{2\pi} d^{i(n-m)t}dt = c_m$$

$$\frac{1}{2\pi}\int_0^{2\pi} f(t)e^{-imt}dt = c_m$$

We use the substitution:

$$t = \frac{2\pi}{L}x \implies x = \frac{L}{2\pi}t$$
$$dt = \frac{2\pi}{L}dx$$

And then this becomes:

$$c_m = \frac{1}{L}\int_0^{L} f(x)e^{-im2\pi x/L}dx$$

For a fourier series $\{c_n\}$, the fourier transform is given given by:

$$\hat{f}(freq) = \sum_{-\infty}^{\infty} c_n \hat{\mathscr{F}}(e^{in2\pi x/L})(freq)$$

But we have

$$\hat{\mathscr{F}}(e^{in2\pi x/L}) = \int_{-\infty}^{\infty} e^{in2\pi x/L}e^{-i2\pi freqx}dx$$

$$= \int_{-\infty}^{\infty} e^{ix(n2\pi/L-2\pi freq)}dx = 2\pi\delta(n2\pi/L - 2\pi freq) = \delta(n/L - freq)$$

And so we get:

$$\hat{f}(freq) = \sum_{-\infty}^{\infty} c_n\delta(freq. - n/L)$$

$$= \sum_{-\infty}^{\infty} c_n/L\delta(freq.L - n)$$

If we inverse fourier transform we get the following. We can also use $x/L = n/N$

$$Data = spacial, Fata = frequency$$

I don't think the following is useful:

$$f(x) = \sum_{n=-\infty}^{\infty} f_n e^{2\pi inx/L}$$

If we multiply both sides by $e^{-2\pi imx/L}$ and integrate we get:

$$\int_{-\infty}^{\infty} f(x)e^{2\pi inx/L}dx = \sum_{n=-\infty}^{\infty} f_n \int_{-\infty}^{\infty} e^{i2\pi x/L(n-m)}dx = \sum_{n=-\infty}^{\infty} f_n \int_{-\infty}^{\infty} e^{i2\pi x(n-m)/L}dx$$

$$= \sum_{n=-\infty}^{\infty} f_n\delta((n-m)/L) = f_n\delta((n-m)/L)$$

4.2.4 Combining these operations

With the split step fourier method, we assume that the linear and nonlinear terms do not affect each other over short timescales. This means that we can propegate forwards according to the linear evolution equation, and then propegate forwards according to the nonlinear evolution equation, and we should get a good approximation for the solution to the original differential equation. This can be represented as the following, where $\hat{N}$ refers to the nonlinear evolution operator, and $\hat{L}$ refers to the linear evolution operator.

$$\psi(t) = e^{\hat{N}(t)}e^{\hat{L}(t)}\psi(0)$$

It turns out, however, that a more stable way of doing this evolution is with the following (notice that the two outside terms are both evolution operators for half of the time period):

$$\psi(t) = e^{\hat{L}(t/2)}e^{\hat{N}(t)}e^{\hat{L}(t/2)}\psi(0)$$

I need to check whether it should be NLN or LNL for the order of the time evolution operators.

4.3 Testing the Code

4.3.1 Comparison with Analytic Solutions

Find exact solutions (to both CGLE and NLSE), and show that it matches well. After x amount of time, how accurate is the answer

4.3.2 Decreasing the step size and Determining Appropriate Values

Find interesting solitons and show that we can reproduce these solutions.

For unusual, and non-analytic solutions, especially spiny solitons, as you decrease the step size, after a length of time t, how close is the answer to the limit? I.e. make the time step size 0.1, and propagate forwards in time until t=10. repeat with step sizes 0.01, 0.001 and plot the final answer. It should approach a constant value. Take the step size to be the smallest value such that the solution is within 1% of the answer with the previous step size.

Chapter 5

Machine Learning Theory

5.1 Different Types of Machine Learning

I want to enter in an image, and get it to tell me if it is oscillating/etc.
I want another machine learning algorithm to predict the required precision, and wait time before recording the output.
Monte Carlo and gradient descent to find new solns.

5.2 Linear Regression

Former Title: Different Options for the Machine Learning Algorithm
Another Title Removed: A detailed look at neural networks

Step 1: Define a cost function:
Suppose we have input parameters $\vec{x}_i$ with the true value given by y_i. For linear regression, we choose $\vec{\theta}$ such that our prediction for the value of y_i is $\vec{\theta} \cdot \vec{x}_i$. We want to vary the vector $\vec{\theta}$ such that the predictions match as best as possible. This is typically done by providing a cost function that is lowest when the predictions match the true value best. A typical cost function is (least squares):

$$J_\theta = \sum_i \left(\vec{\theta} \cdot \vec{x}_i - y_i \right)^2$$

Suppose we have two parameters, we might get an image as shown below. The minimum of this graph give θ which causes the best predictions. Typically the algorithm to find the minimum is done by continually taking the steepest path. However the algorithm may end up finding a local minima, as in the subsequent figure. Octave has lots of fancy functions that can do this really efficiently, and can try to avoid local minima.

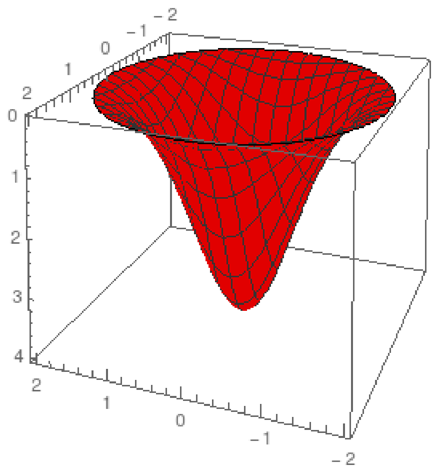

Figure 5.1: Gradient descent example

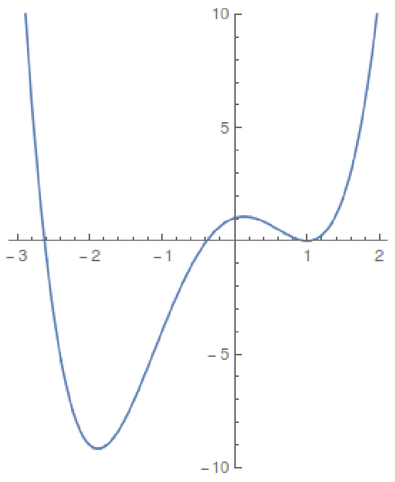

Figure 5.2: Gradient descent example

We can increase the number of input features. One feature we want to add is an intercept term, which will equal 1 for every training example. Other features we might want to add, for example, would be the polynomial features $\{x_1^2, x_1 x_2, x_1 x_3, ...\}$. However if we add in these extra features, we are faced with another issue, and that is overfitting. If we have a high degree polynomial, it becomes much easier to overfit the data. In order to prevent this, we can adjust the cost function to:

$$J_\theta = \frac{1}{2m} \sum_i \left(\vec{\theta} \cdot \vec{x}_i - y_i \right)^2 + \frac{\lambda}{2m} |\vec{\theta}|^2$$

By doing this, if several of the elements of $\vec{\theta}$ are large, then the cost will increase, and so only the important features will remain large. However for this to work we must normalise the data, and in my program this is done by transforming each feature $x_i^{(j)}$ in the following manner, where the i indicates that it is the i^{th} training example, and the j index refers to the j^{th} component of the vector. Below $\mu^{(j)}$ refers to the mean of the j^{th} component of all the training examples, and $\sigma^{(j)}$ the standard deviation

$$x_i^{(j)} \to \frac{x_i^{(j)} - \mu^{(j)}}{\sigma^{(j)}}$$

This ensures that all of the features are on the same scale, and so the elements of θ are penalised equally by the cost function.

5.3 Training and Testing the Algorithm

Tried initially with only linear terms, then expanded to polynomial terms with regularisation and it was able to fit data with polynomial features $(0.5x_1 + x_2^3)$. This game me the confidence that the algorithm was working correctly, and so I could go ahead and train the algorithm on real data with confidence.

My algorithm went up to 4th order polynomial terms, and with 6 different parameters being varied, this results in a total of 210 features.

Chapter 6

The result from much work

6.1 The sorts of things that I found

Conclusion

Appendix A

Appendix B

Bibliography

[1] Name *Title*, Publishing info, etc

[2] Another Guy *title2* pub,etc.