0.1 The Nonlinear Schrödinger Equation

The basic equation that describes the propagation of pulses of light in a nonlinear dispersive medium, such as a fiber optic, is the nonlinear Schrödinger equation (NLSE). It is written

$$-i\frac{\partial u}{\partial z} = \frac{1}{2}\frac{\partial^2 u}{\partial t^2} + |u|^2 u, \tag{1}$$

where u represents the amplitude envelope of the pulse and z and t refer to position and time. A useful check to see if an algorithm works for the NLSE is to use an initial condition that is a solution itself to the NLSE. Therefore, the wave function should stay stationary as one propagates through time. The analytic solution for the above equation is

$$u(z,t) = a\text{sech}(at)e^{i(a^2 z/2+\theta_0)}. \tag{2}$$

This solution is often called a bright soliton. When using this solution as an initial condition, the phase factor was ignored as the amplitude of the wave function was the characteristic being observed and a was set to 2.

0.2 Split-Step Fourier Method

The split-step Fourier method is based on the idea of the time evolution propagator $\hat{U}$ in quantum mechanics. If a wave-function ψ is defined at a specific time t then it can be calculated a short time-step dt later by

$$\psi(z, t+dt) = \hat{U}(t+dt|t)\psi(z,t). \tag{3}$$

$\hat{U}$ is related to the Hamiltonian in quantum mechanics by

$$\hat{U} \approx \exp\left(-idt\left(-\frac{1}{2}\frac{\partial^2}{\partial z^2} + V\left(z, t+\frac{dt}{2}\right)\right)\right), \tag{4}$$

where V is the potential. Since the two parts of the exponent do not commute, the exponential cannot be split by normal factorisation. However, a variant of the Baker–Campbell–Hausdorff formula (BCH) [**?**] can be made use of:

$$\exp\left(\hat{A}+\hat{B}\right) = \exp\left(\hat{A}/2\right)\exp\left(\hat{B}\right)\exp\left(\hat{A}/2\right) + O\left(\left(\hat{A},\hat{B}\right)^3\right) \tag{5}$$

to write the time evolution propagator as

$$\hat{U}\left(t+dt|t\right) \approx \hat{U}_K \cdot \hat{U}_V\left(t+\frac{dt}{2}\right)\cdot \hat{U}_K, \tag{6}$$

where

$$\hat{U}_K = \exp\left(\frac{idt}{4}\frac{d^2}{dz^2}\right),\ \hat{U}_V(t) = \exp\left(-idtV(z,t)\right) \tag{7}$$

are the kinetic and potential propagators respectively. Using the symmetric variant of the BCH reduces the error in the split-step method to third order.

To utilise the method numerically, the position variable must be discretised. Similarly the wave function ψ can be written as a vector with each component $\psi_n(t)$ equal to $\psi(x_n, t)$. The potential propagator is then easy to use on ψ as it can be represented by the matrix

$$\text{diag}\left[\exp\left(-idtV\left(z_n, t+\frac{dt}{2}\right)\right)\right]. \tag{8}$$

On the other hand, the kinetic propagator contains spatial derivatives and is therefore not diagonal in a position basis. However, it is diagonal in k-space, where k represents wave number. If one writes $\psi(x)$ in the Fourier representation:

$$\psi(x) = \int_{-\infty}^{\infty}\frac{dk}{2\pi}e^{ikz}\psi_k, \tag{9}$$

where ψ_k in the wave function in k-space, then

$$\hat{U}_K \psi(x) = \int_{-\infty}^{\infty} \frac{dk}{2\pi} e^{-\frac{1}{4}i(dt)^2 k^2} e^{ikz} \psi_k. \tag{10}$$

Therefore, to use the kinetic propagator on a wave function in position space, one needs to Fourier transform it to k-space, multiply the result by the k^2 exponential and then inverse Fourier transform back to position space.

0.3 Considerations When Applying Spectral Methods

During the numerical analysis, many Fourier transforms had to be computed. The fast Fourier transform from SciPy was used. It was important to consider the domain when applying this. Spectral methods are far more sensitive to the presence of boundaries than finite difference methods, which can cause stability issues that are not well understood and are difficult to analyse. Also, they can often impose strict limitations on the time-step. To address this issue, a numerical domain was chosen that was large enough such that the initial function was close to zero near the boundaries. This requires an initial condition function that tends to zero as position tends to infinity. Therefore, only a small part of the position domain actually calculated is plotted as most of it is near-zero amplitudes. Additionally, it is important to consider the size of the time-step. If chosen too large then the algorithm will propagate the wave function poorly but if chosen too small then the algorithm will take a lot longer to run to a specific time. This was also true when discretising space. The absolute value of the wave function squared was plotted against position and time. This was because the wave-function is a complex function and so to obtain the amplitude of the physical wave the absolute value has to be taken.

0.4 Error Analysis

To construct an error analysis scheme, the fundamental soliton solution was used. This was because the wave-function should stay stationary and so any deviation from that could be measured and then an error calculated. Hence, the algorithm was given the fundamental soliton as an initial condition and it then propagated the wave for 1000 time-steps. To quantify this deviation, the squared difference of the initial condition and the propagated wave was summed over every discretised position point and then the square root was calculated; this was taken to be the error. This is illustrated in (11), where u_{obs} and u_{int} are the observed and initial amplitudes for each position point. Originally, the deviation of the wave from the initial condition was calculated after the 1000 iterations, however, it was found that the propagated wave function oscillated around the initial condition. Therefore, to avoid the error being skewed by what part of the oscillation the wave was in after 1000 iterations, the deviation was calculated periodically through the propagation and then averaged. Comparatively, this led to a fit of the error with less noise.

$$\text{error} = \left(\sum (u_{obs} - u_{ini})^2 \right)^{0.5} \tag{11}$$

To calculate the dependency of the error with the time-step, the error was calculated for equally spaced time-steps between $0.0025t_c$ and $0.0625t_c$. These small time-steps were chosen to ensure the algorithms were functioning correctly. Next, the natural logarithm of the error was plotted against the natural logarithm of the time-step as a power relationship was suspected, which would then give a straight line. This line of best fit was then calculated using the polyfit function from the numpy library on Python. Alongside this plot, a plot of the residuals was also constructed by subtracting the line of best fit from the original plot. This was to illustrate if there were any other underlying relationships other than a straight line and to spot anomalies.