MA5251 Project 1

Li Xuanguang, A0154735B

March 19, 2023

Contents

1 Conservation in the Equation

First, we show that there is conservation.

Given the equation

$$i\frac{\partial\psi(\mathbf{x},t)}{\partial t} = -\frac{1}{2}\triangle_{\mathbf{x}}\psi(\mathbf{x},t) + V(\mathbf{x})\psi(\mathbf{x},t) + \frac{1}{10}\left|\psi(\mathbf{x},t)\right|^2\psi(\mathbf{x},t)$$

We multiply this equation by the conjugate $\overline{\psi}$ and integrate with respect to $\mathbf{x}$

$$i\int\frac{\partial\psi}{\partial t}\overline{\psi}\ d\mathbf{x} = \frac{1}{2}\int\nabla\psi\cdot\overline{\nabla\psi}\ d\mathbf{x} + \int V\left|\psi\right|^2\ d\mathbf{x} + \frac{1}{10}\int\left|\psi\right|^2\left|\psi\right|^2\ d\mathbf{x}$$

Taking complex conjugate of entire equation, we then have

$$i\int\frac{\partial\overline{\psi}}{\partial t}\psi\ d\mathbf{x} = -\frac{1}{2}\int\overline{\nabla\psi}\cdot\nabla\psi\ d\mathbf{x} - \int V\left|\psi\right|^2\ d\mathbf{x} - \frac{1}{10}\int\left|\psi\right|^2\left|\psi\right|^2\ d\mathbf{x}$$

Adding both equations, we then have the following conservation result:

$$\frac{d}{dt}\int\left|\psi(\mathbf{x},t)\right|^2\ d\mathbf{x} = 0$$

2 Spatial Semi-Discretisation

We define the span as

$$X_N = \text{span}\left\{e^{ikx}e^{imy}e^{ipz}\,|\,|k|,|m|,|p| \leq \frac{N}{2}\right\}$$

Hence, we can spatially discretise the function:

$$\psi_N(\mathbf{x},t) = \sum_{\mathbf{k}=\left\{-\frac{N}{2},\ldots,\frac{N}{2}\right\}^3}\hat{\psi}_{\mathbf{k}}\exp(i\mathbf{k}\cdot\mathbf{x}) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}}\sum_{m=-\frac{N}{2}}^{\frac{N}{2}}\sum_{p=-\frac{N}{2}}^{\frac{N}{2}}\hat{\psi}_{k,m,p}(t)e^{ikx}e^{imy}e^{ipz}$$

Then we can use the Pseudo-spectral method

$$i\frac{\partial\psi_N}{\partial t} = -\frac{1}{2}\triangle_{\mathbf{x}}\psi_N + I_N(V\psi_N) + \frac{1}{10}\left|\psi_N\right|^2\psi_N$$

As a requirement of the Schrödinger equation, where

$$\frac{d}{dt}\int_{[0,2\pi]^3}\left|\psi_N(\mathbf{x},t)\right|^2\ d\mathbf{x} = 0$$

referring to the lecture notes, we can approximate $V(\mathbf{x})\psi_N(\mathbf{x},t)$ by

$$\tilde{F}(\mathbf{x},t) = \sum_{\mathbf{k}=\left\{-\frac{N}{2},\ldots,\frac{N}{2}\right\}^3}\tilde{F}_{\mathbf{k}}(t)\exp(-i\mathbf{k}\cdot\mathbf{x})$$

$$\text{where}\qquad \tilde{F}_{\mathbf{k}}(t) = \frac{1}{N^3}\sum_{\mathbf{j}=\{0,\ldots,N-1\}^3}V(\mathbf{x}_j)\psi_N(\mathbf{x}_j,t)\exp(-i\mathbf{k}\cdot\mathbf{x}_{\mathbf{j}})$$

Thus, we have the equation in form

$$\hat{\psi}'_{\mathbf{k}}(t) = -i\frac{1}{2}\left|\mathbf{k}\right|^2\hat{\psi}_{\mathbf{k}}(t) - i\frac{1}{N^3}\sum_{\mathbf{j}\in\{0,\ldots,N-1\}^3}\left[V(\mathbf{x}_{\mathbf{j}})\psi_N(\mathbf{x}_{\mathbf{j}},t) + \frac{1}{10}\left|\psi_N(\mathbf{x}_{\mathbf{j}},t)\right|^2\psi_N(\mathbf{x}_{\mathbf{j}},t)\right]\exp(-i\mathbf{k}\cdot\mathbf{x}_{\mathbf{j}})$$

Hence, we arrived at the following ODE:

$$\begin{aligned}\hat{\psi}'_{k,m,p}(t) &= -i\frac{1}{2}(k^2+m^2+p^2)\hat{\psi}_{k,m,p}(t) + \hat{f}_{k,m,p}(\hat{\psi}),\\ \hat{\psi} &= \left(\hat{\psi}_{-\frac{N}{2},-\frac{N}{2},-\frac{N}{2}},\ldots,\hat{\psi}_{\frac{N}{2},\frac{N}{2},\frac{N}{2}}\right)\\ \text{where}\quad \hat{f}_{k,m,p}(\hat{\psi}) &= -i\frac{1}{N^3}\sum_{\mathbf{j}\in\{0,\ldots,N-1\}^3}\left[V(\mathbf{x}_{\mathbf{j}})\psi_N(\mathbf{x}_{\mathbf{j}},t) + \frac{1}{10}\left|\psi_N(\mathbf{x}_{\mathbf{j}},t)\right|^2\psi_N(\mathbf{x}_{\mathbf{j}},t)\right]\exp(-i\mathbf{k}\cdot\mathbf{x}_{\mathbf{j}})\end{aligned}$$

3 Temporal-Discretisation

Due to computational resource constraints, a first-order exponential Runge-Kutta Method is used.

$$\hat{\psi}^{n+1}_{k,m,p} = \exp\left(-i\frac{1}{3}(k^2+m^2+p^2)\triangle t_n\right)\left(\hat{\psi}^{n}_{k,m,p} + \triangle t_n \hat{f}_{k,m,p}(\hat{\psi}^n)\right)$$

4 Computational Methodology

Language: Python (V3.10)

Python is used for clean code and version control, due to the ability to use Object Oriented Programming to control classes and functions. In addition, Python has several highly efficient packages, which was used for FFT computation. The package used is `scipy.fft.fftn` and `scipy.fft.ifftn`. The package `numpy` is used for matrix manipulation, and `matplotlib` for plotting.

The class structure and function are as follows

schrod

gamma : int[ ] = [2,4]
varepsilon : float = 1/4
dt : float = 1/timestep
timesteps : int = time * timestep
coord = N
t_upper = time
cent_1, cent_1_sq, cent_2 : complex[ ] centered around 0
xx, yy, zz : meshgrid(N,N,N) for coordinate
x1, y1, z1 : meshgrid(N,N,N) for value store
psi_x : initial spatial psi x value
psi_t_0 : initial temporal psi x value
v_x : time independent potential value
mu : value of mu used in exponential

meshgrid_calc_3d() : xx, yy, zz of meshgrid(N,N,N)
meshgrid_calc_2d() : x1, y1, z1 of meshgrid(N,N,N)
centre_array() : cent_1, cent_1_sq, cent_2 of complex[ ]
init_V_x() : v_x, the time independent potential value
init_psi_x() : psi_x, the initial spatial psi x value
init_psi_t0() : psi_t_0, the initial temporal psi x value
mu_val() : mu, the value of mu used in exponential
nonlinear_compute(u_k_t0) : spatial u_k_n, computes the nonlinear spatial psi term
linear_compute(u_k_t0, u__n) : spatial u_k_n, computes the F(u) term
exp_rk_1storder(u_k_t0, u__n) : temporal u_k_t+1, computes next time step
flatten_z(f) : reduces meshgrid values by 1 dimension, used for surface plot
create_surface_plot(data, func, time) : plots the function by abs, imag, angle, or real values
run() : runs the solver given initial conditions

Run the code in Python directly with command `python XUANGUANG_A0154735B_PRJ1.py`.

The computational steps are as follows:

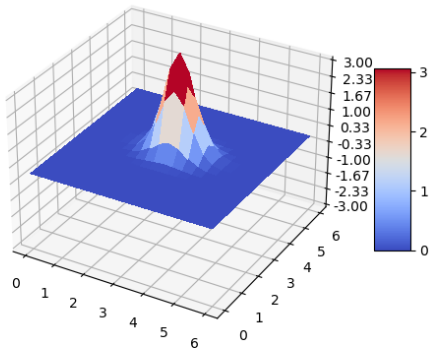
1. Initialise the $V(x)$, $\psi(x,0)$ variables
2. For i in `range(0, timesteps)`, compute the inputs for the next time step by
   i. Compute the Fourier transform of $\psi(x,t)$ to get $\psi(t)$
   ii. `scrhod.nonlinear_compute` to compute nonlinear term
   iii. `scrhod.linear_compute` to compute overall $F(u)$ term
   iv. `scrhod.exp_rk_1storder` to compute the next time step $\psi(x,t+1)$

5 Numerical Results and Discussion

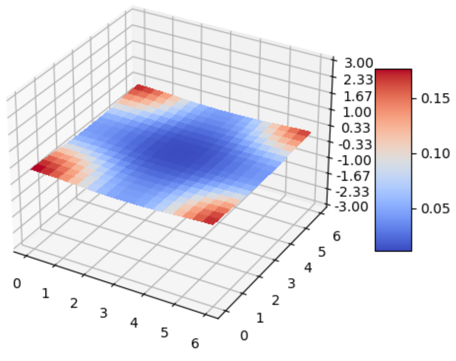
As the program is very computationally heavy and time consuming, a lower number of N and a higher timestep $\triangle t$ is used. The parameters used are as follows:

1. `N = 20`
2. `△t = 0.01`
3. `t_upper = 40` for $t = 5k$, $k = 0, 1, \ldots, 8$

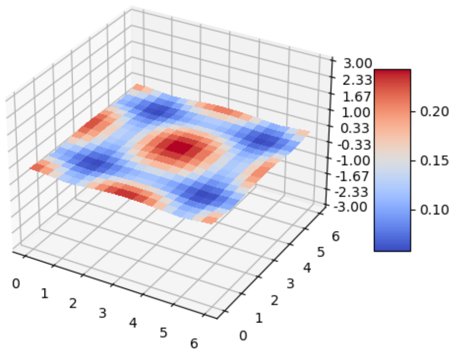
As seen from the figures, the peaks gradually decrease and the wave function spreads fast, before resulting in a very low absolute value.



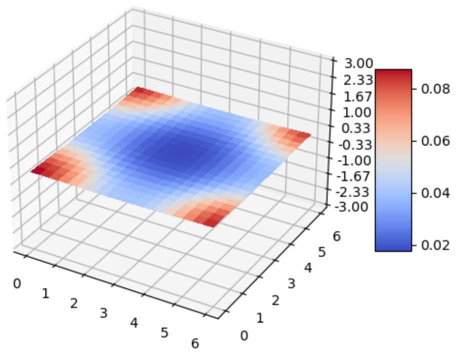
Absolute value graph, $t = 0$



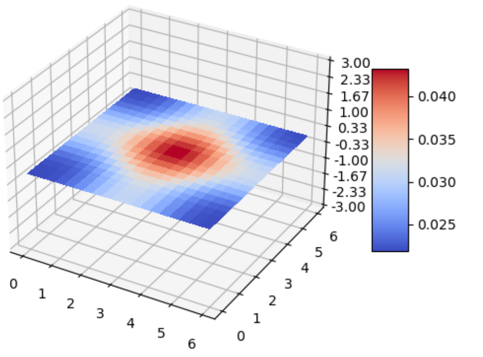
Absolute value graph, $t = 10$



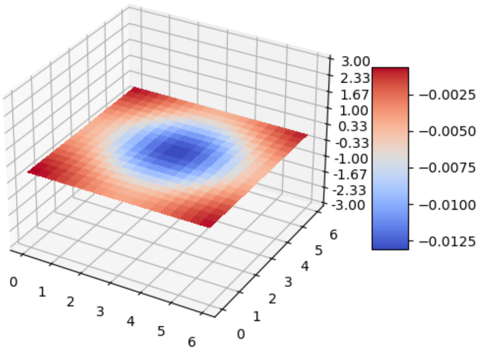
Absolute value graph, $t = 5$



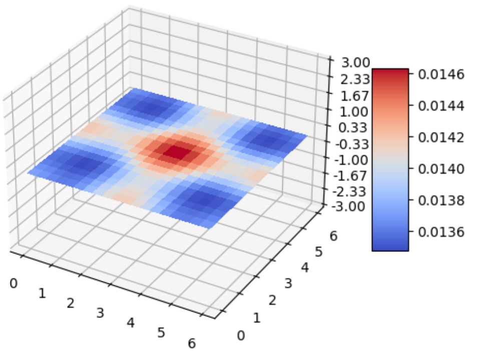
Absolute value graph, $t = 15$



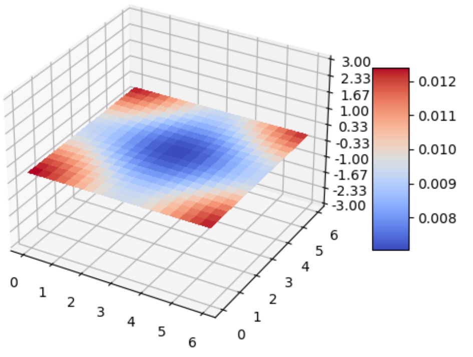
Absolute value graph, $t = 20$



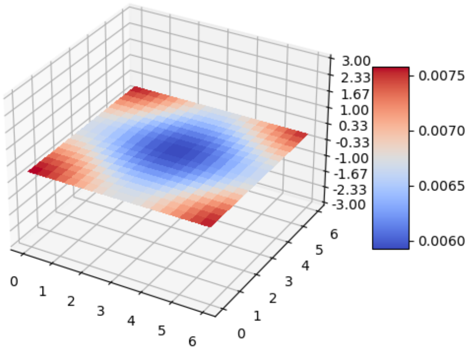
Absolute value graph, $t = 25$



Absolute value graph, $t = 30$



Absolute value graph, $t = 35$



Absolute value graph, $t = 40$

6 Disclosure

This project has been done in discussion with another classmate, Song Zhigao.