

# MA5251 Project 1

Li Xuanguang, A0154735B

March 19, 2023

# Contents

<b>1</b>	<b>Conservation in the Equation</b>	<b>3</b>
<b>2</b>	<b>Spatial Semi-Discretisation</b>	<b>3</b>
<b>3</b>	<b>Temporal-Discretisation</b>	<b>4</b>
<b>4</b>	<b>Computational Methodology</b>	<b>4</b>
<b>5</b>	<b>Numerical Results and Discussion</b>	<b>5</b>
<b>6</b>	<b>Disclosure</b>	<b>7</b>

# 1 Conservation in the Equation

First, we show that there is conservation.

Given the equation

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

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

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

Taking complex conjugate of entire equation, we then have

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

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

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

# 2 Spatial Semi-Discretisation

We define the span as

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

Hence, we can spatially discretise the function:

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

Then we can use the Pseudo-spectral method

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

As a requirement of the Schrödinger equation, where

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

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

$$\begin{aligned} \tilde{F}(\mathbf{x}, t) &= \sum_{\mathbf{k}=\{-\frac{N}{2}, \dots, \frac{N}{2}\}^3} \tilde{F}_{\mathbf{k}}(t) \exp(-i\mathbf{k} \cdot \mathbf{x}) \\ \text{where } \tilde{F}_{\mathbf{k}}(t) &= \frac{1}{N^3} \sum_{\mathbf{j}=\{0, \dots, N-1\}^3} V(\mathbf{x}_{\mathbf{j}}) \psi_N(\mathbf{x}_{\mathbf{j}}, t) \exp(-i\mathbf{k} \cdot \mathbf{x}_{\mathbf{j}}) \end{aligned}$$

Thus, we have the equation in form

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

Hence, we arrived at the following ODE:

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

### 3 Temporal-Discretisation

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

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

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

Run the code in Python directly with command `python LI_XUANGUANG_A0154735B_PRJ1.py`.

The computational steps are as follows:

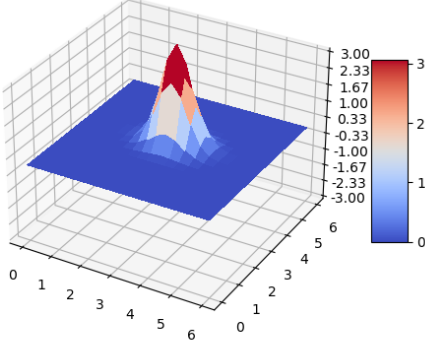
1. Initialise the  $V(x)$ ,  $\psi(x, 0)$  variables
2. For  $i$  in `range(0, timesteps)`, compute the inputs for the next time step by
  - i. Compute the Fourier transform of  $\psi(x, t)$  to get  $\psi(t)$
  - ii. `scrhod.nonlinear_compute` to compute nonlinear term
  - iii. `scrhod.linear_compute`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

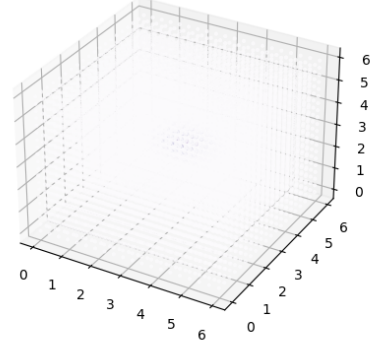
As the program is very computationally heavy and time consuming, a lower number of  $N$  and a higher timestep  $\Delta t$  is used. The parameters used are as follows:

1.  $N = 20$
2.  $\Delta t = 0.01$
3.  $t_{\text{upper}} = 40$  for  $t = 5k$ ,  $k = 0, 1, \dots, 8$

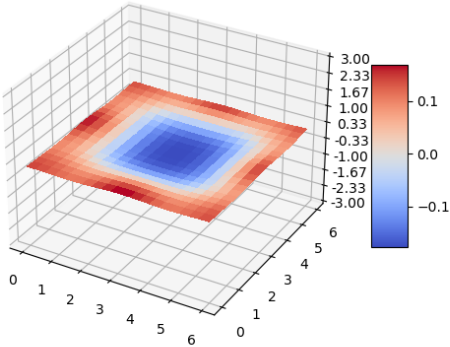
As seen from the figures, the peaks gradually decrease and the wave function spreads fast, before resulting in a very low absolute value. The plots are submitted as part of the submission package.



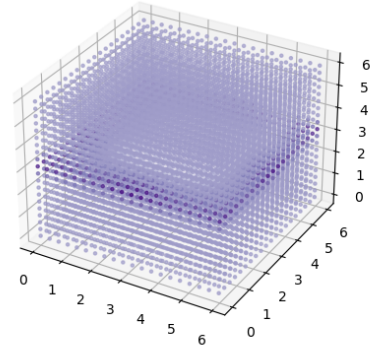
Real value surface graph,  $t = 0$



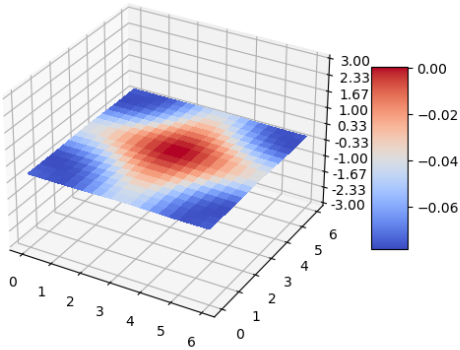
Real value 3D graph,  $t = 0$



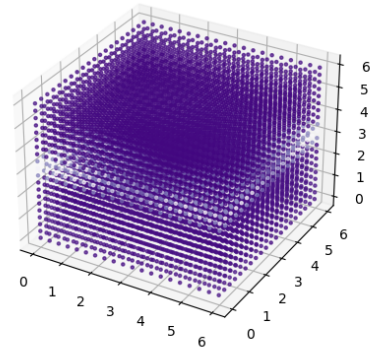
Real value surface graph,  $t = 5$



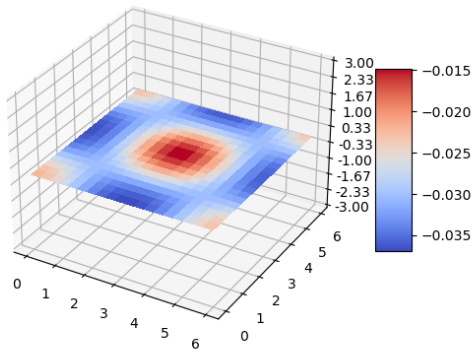
Real value 3D graph,  $t = 5$



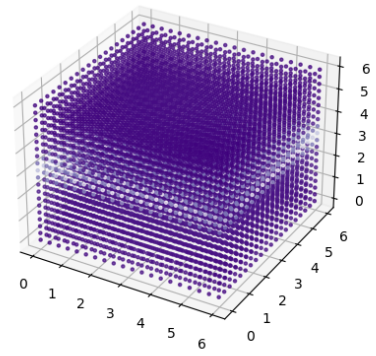
Real value surface graph,  $t = 10$



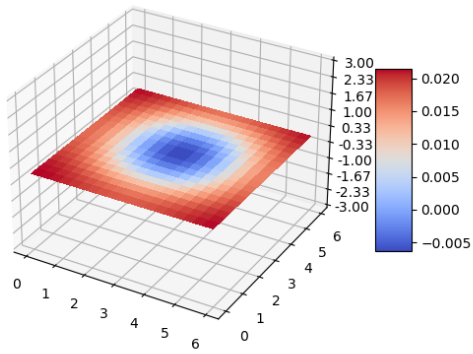
Real value 3D graph,  $t = 10$



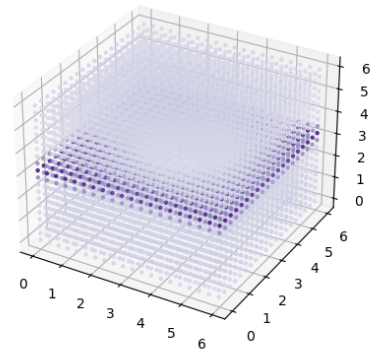
Real value surface graph,  $t = 15$



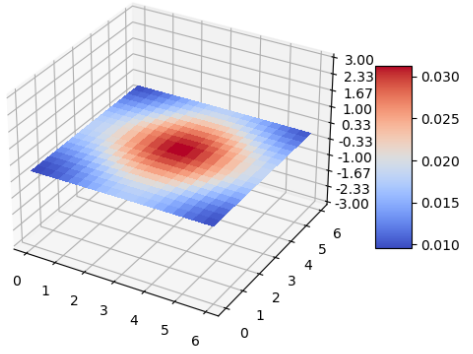
Real value 3D graph,  $t = 15$



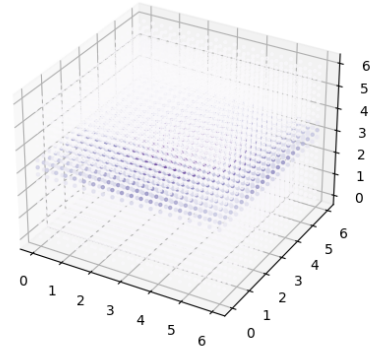
Real value surface graph,  $t = 20$



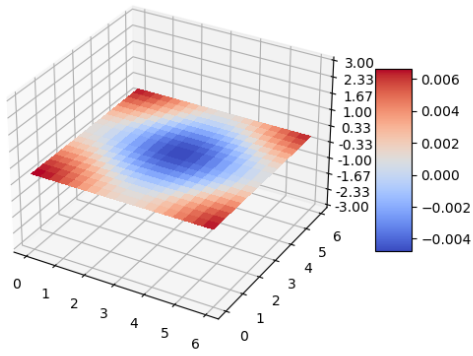
Real value 3D graph,  $t = 20$



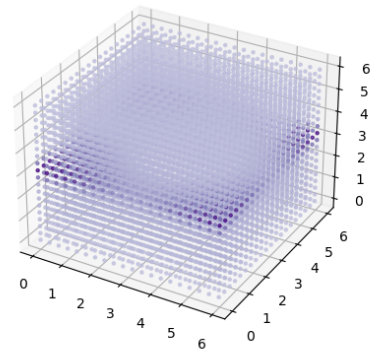
Real value surface graph,  $t = 25$



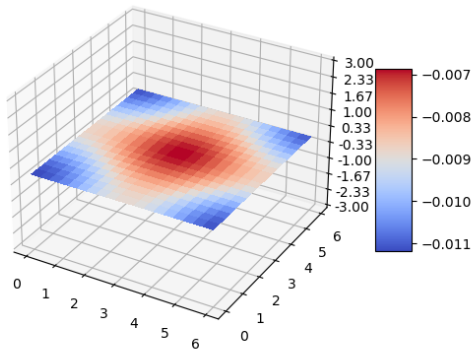
Real value 3D graph,  $t = 25$



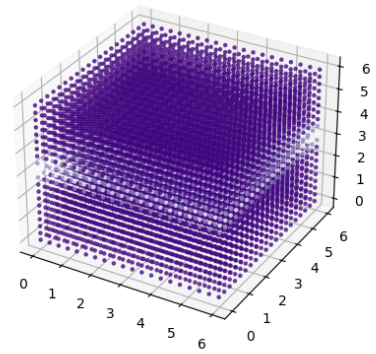
Real value surface graph,  $t = 30$



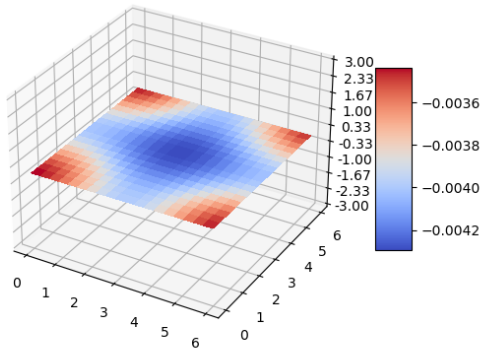
Real value 3D graph,  $t = 30$



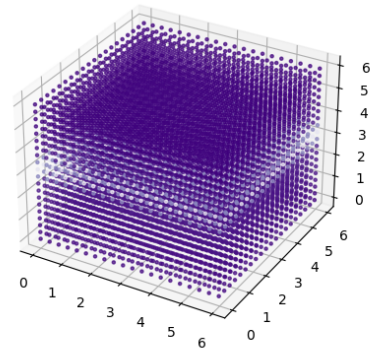
Real value surface graph,  $t = 35$



Real value 3D graph,  $t = 35$



Real value surface graph,  $t = 40$



Real value 3D graph,  $t = 40$

## 6 Disclosure

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