MA202- Group Project

Indian Institute of Technology Gandhinagar

Project 18: Scattering of a Quantum Mechanical Wave Packet by a Potential in One-Dimension

PROJECT REPORT

By

Group 5

Yash Ahire - 21110016

Anay Singh Sisodiya - 21110022

Darshi Doshi - 21110050

Diya Mahesh - 21110061

Sharika S - 21110194

Table of Contents

1. Acknowledgements…….3
2. Chapter 1.1: Introduction to the Problem…….4
    a. Chapter 1.2: Goal of Project…….4
3. Chapter 2.1: The Introduction to the Time-dependent Schrodinger Equation…….5
    a. Chapter 2.2: Detailed outline of the numerical discretization technique…….7
4. Chapter 3.1: Python Code
    a. Chapter 3.1: Code for Propagation of Wave packet**.....**12
    b. Chapter 3.2: Code for Spreading of a Wave Packet……15
    c. Chapter 3.3 : Code for Construction of a Propagating Wave Packet……18
    d. Chapter 3.4: Code for Reflection From Potential Wall ….21
5. References……..24

Acknowledgments

The project component of the course provided us with an opportunity to apply the various concepts taught in the lectures, in the real world, and therefore helped us to better understand those particular concepts. We are very grateful to **Professor Uddipta Ghosh** for providing us with constant support.

We also thanks to **Professor Anand Sengupta** for his guidance throughout the duration of our project. His insights into the topic helped us to navigate through the most complex parts of the project.

Chapter 1: Introduction to the Problem

When a quantum mechanical wavepacket encounters a potential, it can be scattered in various ways depending on the properties of the potential and the wavepacket. The behavior of the wavepacket can be analyzed using the time-dependent Schr ̈odinger equation. In one-dimensional (1D) scattering, the wavepacket can be represented as a superposition of plane waves with different energies and moments. As the wavepacket encounters the potential, it can be partially reflected and partially transmitted. The amount of reflection and transmission depends on the properties of the potential, such as its height, width, and shape. The reflection and transmission coefficients can be calculated using the wave function of the scattered wave packet.

The scattering of a quantum mechanical wavepacket by a potential can have various interesting physical effects, such as diffraction, interference, and tunneling. These effects can be analyzed using mathematical techniques such as the Born approximation and the WKB approximation, which provide approximations to the scattering amplitudes and the transmission probabilities. Scattering of wavepackets by potentials is an essential area of study in quantum mechanics and has numerous applications in areas such as solid-state physics, materials science, and nanotechnology.

Chapter 1.2: Goal of the Project

The project's purpose is to use numerical methods such as the Split-Operator method to the Time Dependent Schrödinger Equation (TDSE) to simulate the scattering of a quantum mechanical wavepacket by a 1D potential and visualize the results in Python or MATLAB. The code would show the wavepacket's behavior as it contacts the potential and is partially reflected and transmitted, as well as provide insight into the physical effects of scattering, such as diffraction, interference, and tunneling.

Chapter 2: Numerical Methods Used in the Project

Chapter 2.1: The Introduction to the Time-dependent Schrodinger Equation and its Derivation:

The time-dependent Schrodinger equation describes how a system changes from one state to another. It is a generalization of Schrodinger's wave equation to determine the time variation of the wave function as well as its spatial variation. The evolution of a closed quantum system is given by the time-dependent Schrödinger equation

$$i\hbar\frac{\partial|\psi\rangle}{\partial t} = \mathcal{H}|\psi\rangle$$

where H is the Hamiltonian of the system (the energy operator), and $\hbar$ is the reduced Planck's constant.

In quantum mechanics, the Hamiltonian of a system is an operator corresponding to the total energy of that system, including both kinetic energy and potential energy. Its spectrum, the system's energy spectrum or its set of energy eigenvalues, is the set of possible outcomes obtainable from a measurement of the system's total energy.

$$\begin{aligned}\hat{H} &= \hat{T} + \hat{V} \\ &= \frac{\hat{\mathbf{p}}\cdot\hat{\mathbf{p}}}{2m} + V(\mathbf{r},t) \\ &= -\frac{\hbar^2}{2m}\nabla^2 + V(\mathbf{r},t)\end{aligned}$$

Derivation of TDSE:

Considering a complex plane wave:

$$\Psi(x,t) = Ae^{i(kx-\omega t)}$$

Now the Hamiltonian system is

H = T + V

Where 'V' is the potential energy, and 'T' is the kinetic energy. As we already know that 'H' is the total energy, we can rewrite the equation as:

$$E = \frac{p^2}{2m} + V(x)$$

Now taking the derivatives,

$$\frac{\partial \Psi}{\partial t} = -i\omega A e^{i(kx-\omega t)} = -i\omega \Psi(x,t)$$
$$\frac{\partial^2 \Psi}{\partial x^2} = -k^2 A e^{i(kx-\omega t)} = -k^2 \Psi(x,t)$$

We know that,

$$p = \frac{2\pi\bar{h}}{\lambda} \text{ and } k = \frac{2\pi}{\lambda}$$

where 'λ' is the wavelength and 'k' is the wavenumber.

We have k = p/h.

Therefore,

$$\frac{\partial^2 \Psi}{\partial x^2} = -\frac{p^2}{\bar{h}^2}\Psi(x,t)$$

Now multiplying Ψ (x, t) to the Hamiltonian we get,

$$E\Psi(x,t) = \frac{p^2}{2m}\Psi(x,t) + V(x)\Psi(x,t)$$

The above expression can be written as:

$$E\Psi(x,t) = \frac{-\bar{h}^2}{2m}\frac{\partial^2\Psi}{\partial x^2} + V(x)\Psi(x,t)$$

We already know that the energy wave of a matter wave is written as

$$E = \bar{h}\omega$$

So we can say that

$$E\Psi(x,t) = \frac{\bar{h}\omega}{-i\omega}\Psi(x,t)$$

Now combining the right parts, we can get the Schrodinger Wave Equation

$$i\bar{h}\frac{\partial\Psi}{\partial t} = \frac{-\bar{h}^2}{2m}\frac{\partial^2\Psi}{\partial x^2} + V(x)\Psi(x,t)$$

This is the derivation of Schrödinger Wave Equation (time-dependent)

Chapter 2.2: Detailed outline of the numerical discretization technique:

The Schrodinger equation is used in quantum mechanics to describe the behavior of wavefunctions in space and time. Numerical methods are frequently used to discretize the wavefunction and solve for its development in time in order to answer this equation for a given potential. One commonly used numerical discretization technique is the finite difference method, which involves approximating the derivatives in the Schrodinger equation using finite differences.

To apply this technique to scattering problems, one can use a numerical algorithm such as the time-dependent Schrödinger equation (TDSE) method, which involves iteratively solving the Schrödinger equation at each time step using finite differences. The TDSE

method can be used to simulate the propagation of a wavepacket through a potential barrier and calculate the reflection and transmission coefficients.

The Schrodinger equation for a one-dimensional quantum system is given as follows:

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2} + V(x)\psi(x) = E\psi(x)$$

For simplicity, we use a system of units such that m=1 and $\hbar = 1$. Hence our equation becomes

$$-\frac{1}{2}\left(\frac{\psi_{j+1} - 2\psi_j + \psi_{j-1}}{h^2}\right) + V_j\psi_j = E\psi_j$$

where $\hbar$ is the step size.

Let's suppose we want to solve this equation in the region x $\in$ [a,b], then we can create N+1 grid points such that x_0 =a and X_N =b. Since the particle is confined in the region x $\in$ [a,b], this leads to the following boundary conditions:

$$\psi_0 = 0$$

$$\psi_N = 0$$

Thus, we only need to compute ψ_j for the remaining N-1 grid points that are j = 1,2,3……, N-1. To illustrate, consider N=5, then we have

$$\psi_0 = 0$$

$$\psi_5 = 0$$

In summary, we've shown that the finite difference scheme is a very useful method for solving an eigenvalue equation such as the Schrodinger equation.

Chapter 2.3: An outline of the numerical solution methodology

A. Split Operator Method:

We consider the time-dependent Schrodinger equation,

$$i\hbar\frac{\partial}{\partial t}\psi(t) = \hat{H}\psi(t), \quad \text{..... 1}$$

with Hˆ the Hamiltonian for the motion of a particle interacting with an external time-dependent potential V (t), i.e.,

$$\hat{H} = \hat{K} + \hat{V} = \frac{\hat{P}^2}{2m} + V(t), \quad \text{..... 2}$$

where Kˆ and Vˆ are the kinetic and potential energy operators, respectively, Pˆ is the momentum operator, and m the mass of the particle.

The formal solution to eq. (1) is given by the time evolution operator Uˆ, itself a solution of the time-dependent Schr¨odinger equation

$$i\hbar\frac{\partial}{\partial t}\hat{U} = \hat{H}\hat{U},$$

such that, given an initial wave function at a time t_0, $\psi(t_0)$

Here we assume the wave function Ψ(x, t) to be Ψ(x, t) = C exp[-(x - x0)^2/σ^2]

We use an assumption of Gaussian distribution for the wave packet for approximating wave functions because it has a well defined position and momentum.
The time dependence of the wave packet can be further analyzed by calculating its time evolution using the time-dependent Schrödinger equation. This involves taking the partial derivative of the wave function with respect to time, and substituting it into the time-dependent Schrödinger equation:

iħ∂Ψ(x,t)/∂t = (-ħ^2/2m) ∂^2Ψ(x,t)/∂x^2 + V(x)Ψ(x,t)
where ħ is the reduced Planck constant, m is the mass of the particle, V(x) is the potential energy function, and ∂^2/∂x^2 is the second partial derivative with respect to x.

We may derive the time evolution of the wave function, and consequently the time evolution of the Gaussian wave packet, by solving the time-dependent Schrödinger equation. The spreading and oscillation behavior of the wave packet in time are dictated by the beginning conditions and the features of the potential energy function.

The assumption of a Gaussian expression for the time-dependent Schrödinger equation's tunneling and scattering operations is also based on its convenience as an approximation for the system's wave function.

The difficulty now is to compute the exponential of matrices K and V, which is only straightforward for a diagonal matrix. This is addressed in the original implementation of the split-operator approach by remembering that while the matrix for Vˆ is diagonal for a spatial representation of the wave function, the matrix for Kˆ is diagonal in momentum space. Using the Fourier transform

$$\exp\left[-\frac{\mathrm{i}\Delta t}{2\hbar}\hat{K}(\mathbf{x})\right]\psi(\mathbf{x}) = \mathcal{F}^{-1}\exp\left[-\frac{\mathrm{i}\Delta t}{2\hbar}\hat{K}(\mathbf{p})\right]\mathcal{F}\psi(\mathbf{x}),$$

where, considering that $\hat{K} = \hat{P}^2/2m$,

$$\hat{K}(\mathbf{p}) = \frac{\mathbf{p}^2}{2m},$$
$$\hat{K}(\mathbf{x}) = -\frac{\hbar^2}{2m}\nabla^2,$$

After the forward transform, the momentum grid, obtained from the wave vector k = p/h~, is discretized according to

$$p_{x,i} = 2\pi\hbar \frac{i}{n_x \Delta x}, \qquad i = -\frac{n_x}{2}, \ldots, \frac{n_x}{2},$$

$$p_{y,j} = 2\pi\hbar \frac{j}{n_y \Delta y}, \qquad j = -\frac{n_y}{2}, \ldots, \frac{n_y}{2},$$

$$p_{z,k} = 2\pi\hbar \frac{k}{n_z \Delta z}, \qquad k = -\frac{n_z}{2}, \ldots, \frac{n_z}{2}.$$

Chapter 3: Python Code

Chapter 3.1: Code for Propagation of Wave packet:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fftpack import fft, ifft

# Parameters
N = 1024 # number of grid points
L = 100 # grid length
x = np.linspace(-L/2, L/2, N) # position grid
dx = x[1] - x[0] # grid spacing
dt = 0.1 # time step
tmax = 10 # maximum time
timesteps = int(tmax/dt) # number of time steps

# Initial wavefunction
x0 = 0 # center of wavepacket
sigma = 1 # width of wavepacket
k0 = 5 # initial momentum
psi = np.exp(-(x-x0)*2/(2*sigma*2))*np.exp(1j*k0*x) # initial
wavefunction

# Potential energy
V = np.zeros(N) # no potential

# Kinetic energy operator in momentum space
dk = 2*np.pi/L
k = np.concatenate((np.arange(0,N/2+1)*dk,np.arange(-N/2+1,0)*dk))
T = np.exp(-1j*(k**2)*dt/2)

# Potential energy operator in position space
U = np.exp(-1j*V*dt)

# Time evolution
for t in range(timesteps):
    psi = ifft(T*fft(U*psi)) # split operator method
```

```
    plt.plot(x,np.abs(psi)**2)
    plt.xlabel('Position (x)')
    plt.ylabel('Probability Density (psi*psi)')
    plt.axis([-L/2,L/2,0,1])
    plt.draw()
    plt.pause(0.01)
    plt.clf()
```

Explanation of Code:

This code simulates the time evolution of a quantum mechanical wavepacket in a one-dimensional potential using the split operator method.

The parameters are set at the beginning of the code, including the number of grid points N, the grid length L, the position grid x, the grid spacing dx, the time step dt, the maximum time tmax, and the number of time steps timesteps.

An initial wavefunction is defined using the Gaussian wavepacket with the center at x0=0, width sigma=1, and initial momentum k0=0. The normalization constant C is calculated based on the wavefunction.

The potential energy is set to zero throughout the simulation, and the kinetic energy operator is defined in the momentum space. The operator T evolves the wavefunction for a small time step dt/2 in the momentum space. The potential energy operator U is defined in the position space, and it evolves the wavefunction for a time step dt in the position space.

The time evolution of the wavefunction is performed in a for loop over the number of time steps. At each time step, the wavefunction is evolved using the split operator method, which involves taking the Fourier transform of the wavefunction, applying the kinetic energy operator T in the momentum space, taking the inverse Fourier transform to go back to the position space, applying the potential energy operator U, and then repeating the process.

After each time step, the probability density of the wavefunction is plotted as a function of position x using the command plt.plot(x,np.abs(psi)**2), where np.abs(psi)**2 is the squared magnitude of the wavefunction, representing the probability density. The plt.draw() and plt.pause(0.01) commands update the plot and pause briefly to allow the plot to be seen before clearing it with plt.clf() for the next time step.

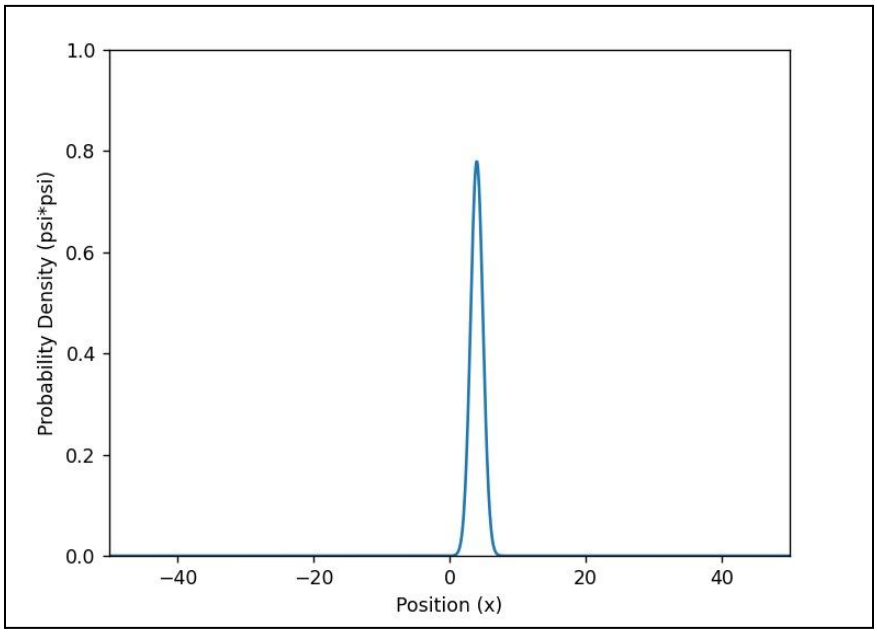

Figure 1: Simulated Python output of the initial position (at t = 0) of the propagating wave packet.

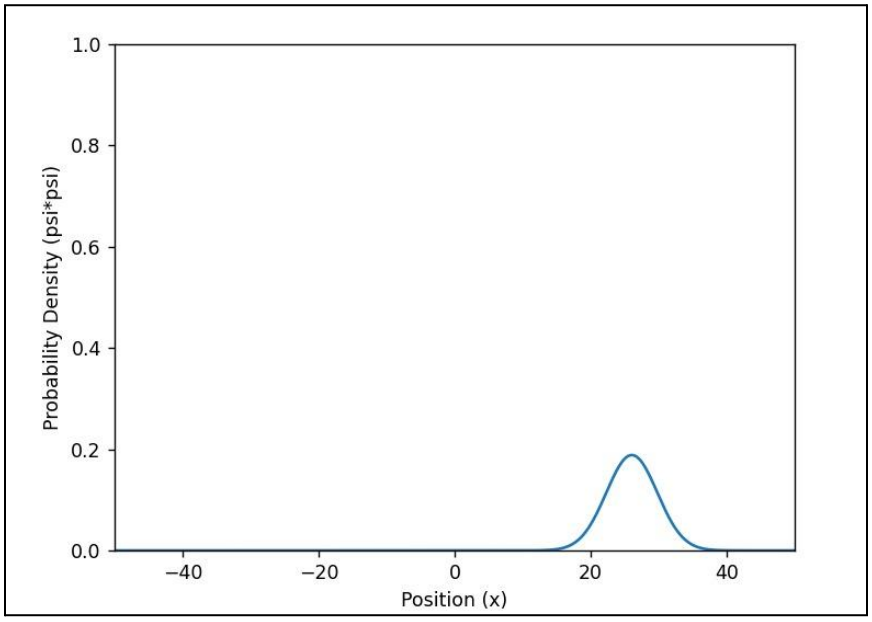

Figure 2: Simulated Python output of the position of the propagating wave packet at time t.

Chapter 3.2: Code for Spreading of a Wave Packet

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fftpack import fft, ifft

# Parameters
N = 1024 # number of grid points
L = 100 # grid length
x = np.linspace(-L/2, L/2, N) # position grid
dx = x[1] - x[0] # grid spacing
dt = 0.1 # time step
tmax = 10 # maximum time
timesteps = int(tmax/dt) # number of time steps

# Initial wavefunction
x0 = 0 # center of wavepacket
sigma = 1 # width of wavepacket
k0 = 0 # initial momentum (set to zero to prevent propagation)
C = 1/(np.pi*sigma**2)**0.25 # normalization constant
psi = C*np.exp(-(x-x0)*2/(2*sigma**2))*np.exp(1j*k0*x) # initial
wavefunction

# Potential energy
V = np.zeros(N) # no potential

# Kinetic energy operator in momentum space
dk = 2*np.pi/L
k =
np.concatenate((np.arange(0,N/2+1)*dk,np.arange(-N/2+1,0)*dk))
T = np.exp(-1j*(k**2)*dt/2)

# Potential energy operator in position space
U = np.exp(-1j*V*dt)
```

```
# Time evolution
for t in range(timesteps):
    psi = ifft(T*fft(U*psi)) # split operator method
    plt.plot(x,np.abs(psi)**2)
    plt.xlabel('Position (x)')
    plt.ylabel('Probability Density (psi*psi)')
    plt.axis([-L/2,L/2,0,1])
    plt.draw()
    plt.pause(0.01)
    plt.clf()
```

Explanation:

In the above code we have used numpy and matplotlib packages to execute our code.

The initial parameters of the psi function and also initial conditions for the simulation have first been defined. The function of psi used is as follows:

$$\psi(x,\ t\ =\ 0)\ =\ C\ e^{-\frac{\left(x-x_0\right)^2}{2\,\sigma^2}}\ .\ e^{ikx}$$

The psi function has both real and imaginary parts.

We then use plotting functions to define the axes and create a plot that gets updated with each timestep.

The update function is called for each frame of the animation to update the plot accordingly.

Finally the FuncAnimation function creates and displays the final animation.

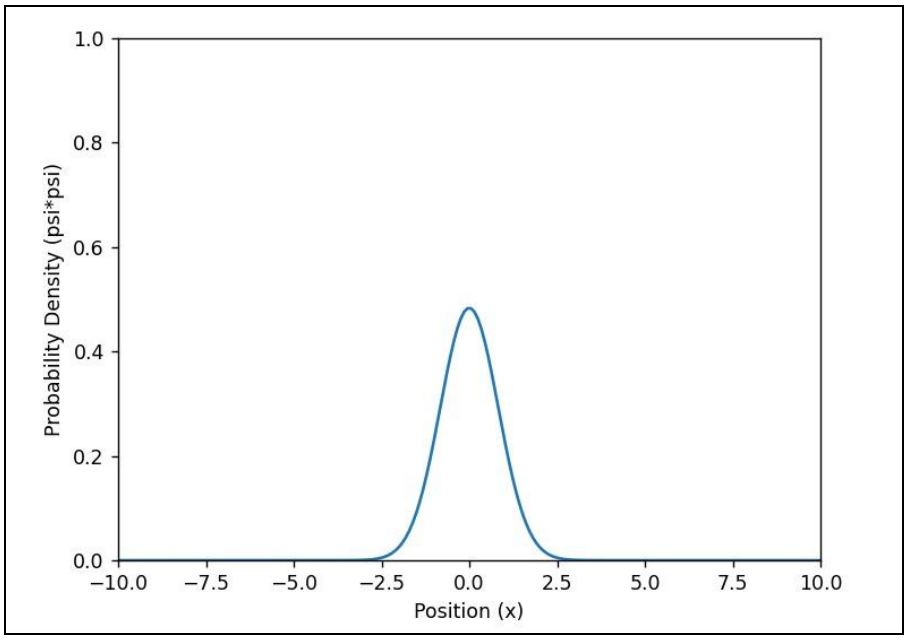

Figure 3: Simulated Python output of the stationary wave packet at time t = 0

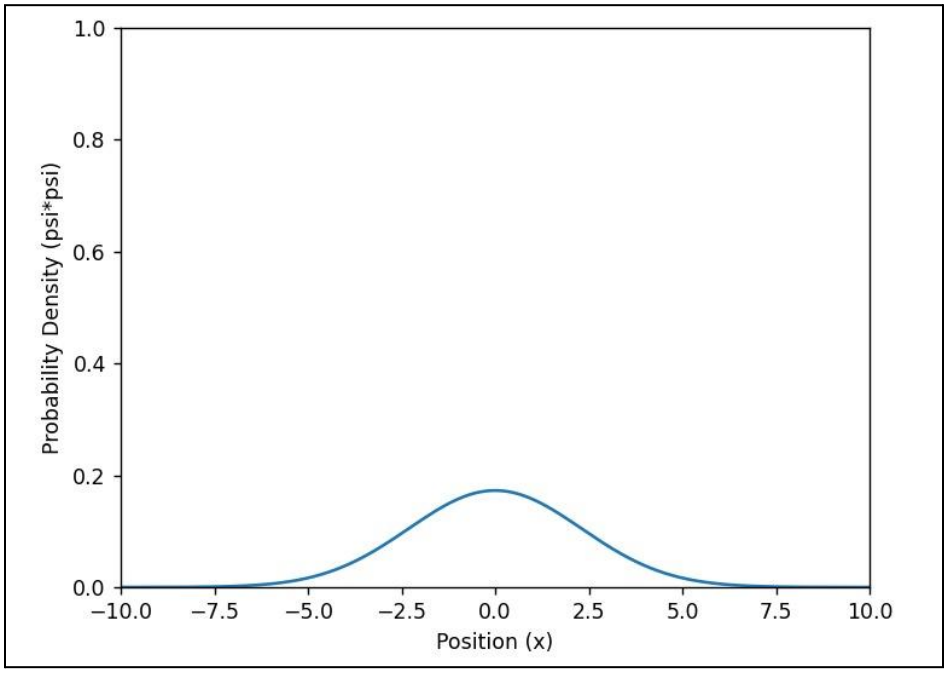

Figure 4: Simulated Python output of the spreading of the wave packet at time t

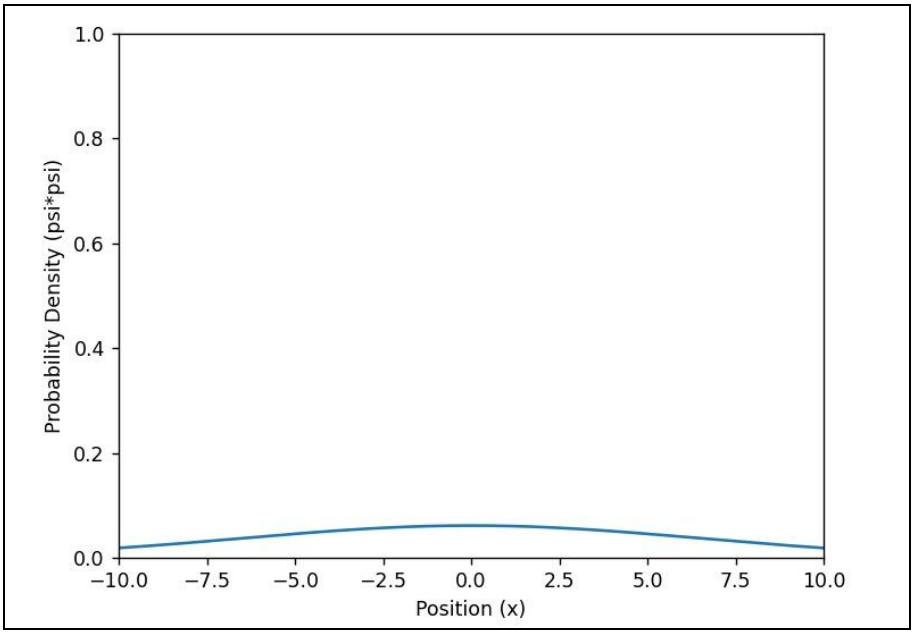

Figure 5: Simulated Python output of the spreading of the wave packet at the end.

Chapter 3.3 : Code for Construction of a Propagating Wave Packet

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fftpack import fft, ifft

# Parameters
N = 1024 # number of grid points
L = 100 # grid length
x = np.linspace(-L/2, L/2, N) # position grid
dx = x[1] - x[0] # grid spacing
dt = 0.1 # time step
tmax = 10 # maximum time
timesteps = int(tmax/dt) # number of time steps

# Initial wavefunction
x0 = 0.4 # center of wavepacket
```

```
sigma = 1 # width of wavepacket
k0 = 500 # initial momentum
C = 1/(np.pi*sigma**2)**0.25 # normalization constant
psi = C*np.exp(-(x-x0)*2/(2*sigma*2))*np.exp(1j*k0*x) # initial
wavefunction

# Potential energy
V = np.zeros(N) # no potential

# Kinetic energy operator in momentum space
dk = 2*np.pi/L
k =
np.concatenate((np.arange(0,N/2+1)*dk,np.arange(-N/2+1,0)*dk))
T = np.exp(-1j*(k**2)*dt/2)

# Potential energy operator in position space
U = np.exp(-1j*V*dt)

# Time evolution
for t in range(timesteps):
    psi = ifft(T*fft(U*psi)) # split operator method

    plt.subplot(3,1,1)
    plt.plot(x,np.real(psi))
    plt.xlabel('Position (x)')
    plt.ylabel('Real Part')

    plt.subplot(3,1,2)
    plt.plot(x,np.imag(psi))
    plt.xlabel('Position (x)')
    plt.ylabel('Imaginary Part')

    plt.subplot(3,1,3)
    plt.plot(x,np.abs(psi)**2)
    plt.xlabel('Position (x)')
```

```
plt.ylabel('Probability Density (psi*psi)')

plt.axis([-L/2,L/2,-1,1])
plt.draw()
plt.pause(0.01)
plt.clf()
```

Explanation:

Here we plot the propagating wave packet and also separately plot its real and imaginary parts.

A function cmplx_propwave has been defined that represents the wavefunction.

Using plotting functions, subplots are created to simultaneously display all graphs, by separating the real and imaginary part of the wavefunction.

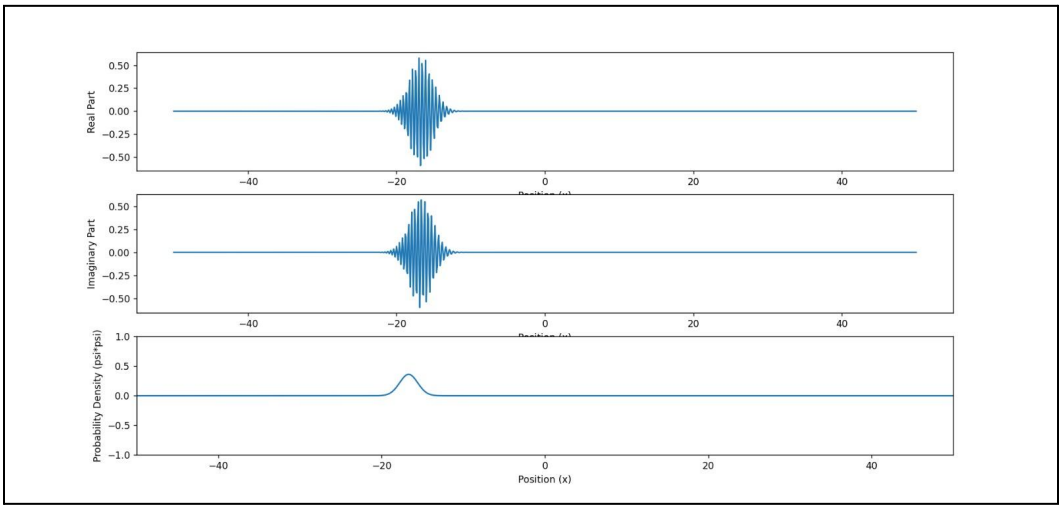

Figure 6: Simulated Python output of the real and the imaginary part of the propagating wave packet at the initial condition.

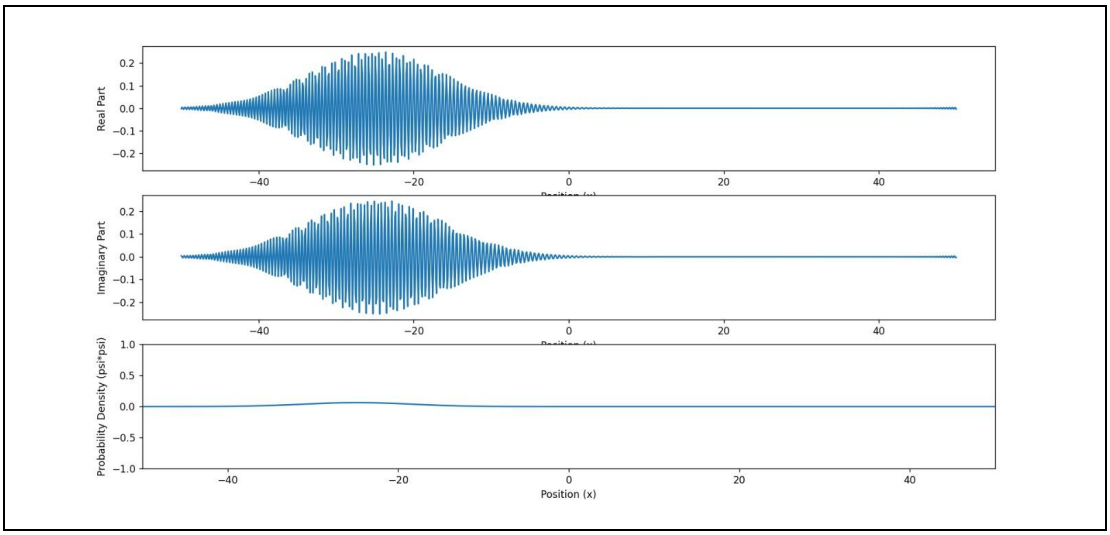

Figure 7: Simulated Python output of the real and the imaginary part of the propagating wave packet at time t.

Chapter 3.4: Code for Reflection From Potential Wall

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fftpack import fft, ifft

# Parameters
N = 1024 # number of grid points
L = 100 # grid length
x = np.linspace(-L/2, L/2, N) # position grid
dx = x[1] - x[0] # grid spacing
dt = 0.1 # time step
tmax = 10 # maximum time
timesteps = int(tmax/dt) # number of time steps

# Initial wavefunction
x0 = -20 # center of wavepacket
sigma = 1 # width of wavepacket
k0 = 5 # initial momentum
```

```
C = 1/(np.pi*sigma**2)**0.25 # normalization constant
psi = C*np.exp(-(x-x0)*2/(2*sigma*2))*np.exp(1j*k0*x) # initial
wavefunction

# Potential energy
V0 = 1e6 # height of potential barrier
V = np.zeros(N) # potential energy
V[x > 0.6] = V0 # potential barrier at x > 0.6

# Kinetic energy operator in momentum space
dk = 2*np.pi/L
k =
np.concatenate((np.arange(0,N/2+1)*dk,np.arange(-N/2+1,0)*dk))
T = np.exp(-1j*(k**2)*dt/2)

# Potential energy operator in position space
U = np.exp(-1j*V*dt)

# Time evolution
for t in range(timesteps):
    psi = ifft(T*fft(U*psi))
    plt.plot(x,np.abs(psi)**2)
    plt.xlabel('Position (x)')
    plt.ylabel('Probability Density (psi*psi)')
    plt.axis([-L/2,L/2,0,1])
    plt.draw()
    plt.pause(0.01)
    plt.clf()
```

Explanation:

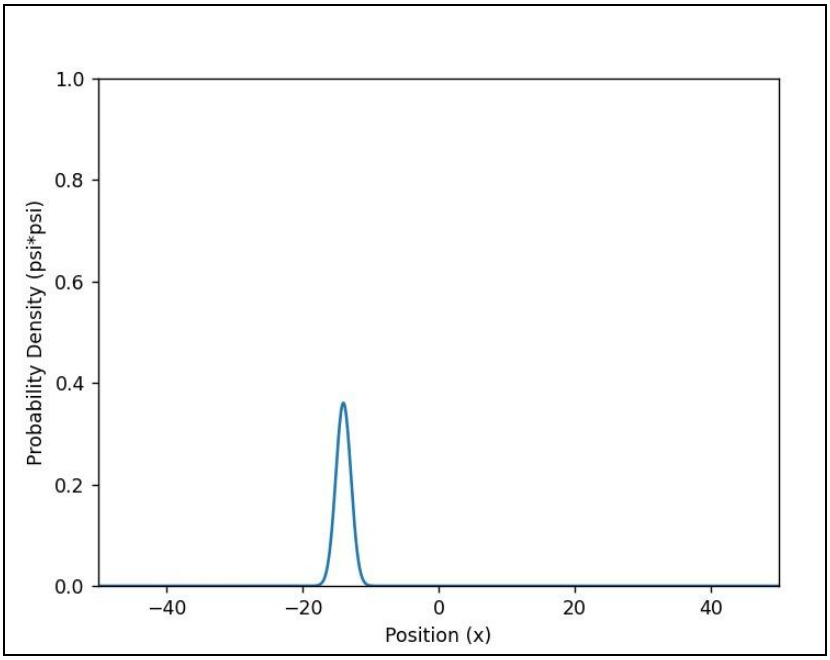

Figure 8: Simulated Python output of the Reflection from potential wall (V=50)

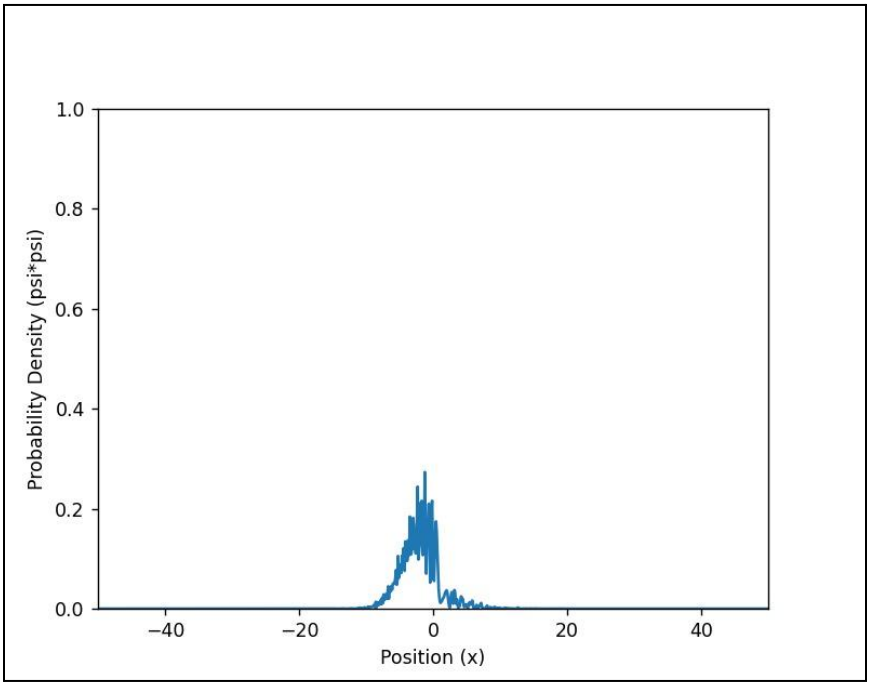

Figure 9: Simulated Python output of the Reflection from potential wall (V=50)

We tried generating the graphs for the reflection of the propagation wave packet from an infinite wall but we were unable to get the desired result for the complete reflection.

References:

1. Cappellaro, P. (2017). Introduction to Applied Nuclear Physics. LibreTexts, 6.01: Time-dependent Schrödinger equation. Retrieved from https://phys.libretexts.org/Bookshelves/Nuclear_and_Particle_Physics/Book%3A_Introduction_to_Applied_Nuclear_Physics_(Cappellaro)/06%3A_Time_Evolution_in_Quantum_Mechanics/6.01%3A_Time-dependent_Schrodinger_equation
2. Aqib, M. (2019). Finite Difference Solution of the Schrödinger Equation. Modern Physics. Retrieved from https://medium.com/modern-physics/finite-difference-solution-of-the-schrodinger-equation-c49039d161a8.
3. Le, H. A., & Vu, V. T. (2013). Split-Step Fourier Methods for the Solution of the Nonlinear Schrödinger Equation. arXiv preprint arXiv:1306.3247.