Visualizing Quantum Wave Packets

Nate Chrisman
Brian Pickens
Andrew Shurman

May 2, 2017

Abstract

The goal of this project was to visualize a quantum wavepacket in 2-D and 3-D within a box and isotropic harmonic oscillator potential. The method of approach included the use of a stepping algorithm to determine the real and imaginary parts of the wave equation in alternating half time-steps (Maestri et al.) and then using two modules, YT and imageio, to create images of the time-evolution of the wavepacket. To test the accuracy of the solutions, constant calculations of total probability and the observation of phenomena such as quantum revival were used. Images of a wavepacket in both potentials was observed, however, these proved to be incorrect as neither probability was conserved nor was quantum revival observed. The cause of the error has yet to be determined even through extensive reworking of the algorithm implementation This has lead us to believe that the problem maybe with the algorithm itself and another will need to be found. Future work would include exploring other algorithms that meet our requirements.

Code is available at `https://github.com/ASU-CompMethodsPhysics-PHY494/final-2017-bogus_project` under MIT License

Background

An electron which is placed in a region with a known potential has behavior that can be determined by solving the time-dependent Schrödinger Equation in 2D, which is presented as

$$i\frac{\partial \psi(x,y,t)}{\partial t} = -\left(\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2}\right) + V(x,y)\psi$$

Given that the electron's initial momentum and position via calculation of a Gaussian, the behavior of the electron can be modeled over time and a calculation of probability density can be made.

Methods

Using the real/imaginary position integration scheme by Maestri *et al.*, the function is separated into real and imaginary parts. A leapfrog integration is performed.

$$R_{i,j}^{n+1} = R_{i,j}^{n-1} + 2\left[(4\alpha + \frac{1}{2}\Delta t V_{i,j})I_{i,j}^{n} - \alpha(I_{i+1,j}^{n} + I_{i-1,j}^{n} + I_{i,j+1}^{n} + I_{i,j-1}^{n})\right]$$

$$I_{i,j}^{n+1} = I_{i,j}^{n-1} + 2\left[(4\alpha + \frac{1}{2}\Delta t V_{i,j})R_{i,j}^{n} - \alpha(R_{i+1,j}^{n} + R_{i-1,j}^{n} + R_{i,j+1}^{n} + R_{i,j-1}^{n})\right]$$

The probability density, $\rho(t)$, is given for the real and imaginary parts at integer and half-integer t values, respectively.

$$\rho(t) = \begin{cases} R^2(t) + I\left(t + \frac{\Delta t}{2}\right) I\left(t - \frac{\Delta t}{2}\right), & \text{for integer t} \\ R^2(t) + I\left(t + \frac{\Delta t}{2}\right) I\left(t - \frac{\Delta t}{2}\right), & \text{for half-integer t} \end{cases}$$

As for visualization, Python modules `yt` and `imageio` were used to capture the probability density at each time step. Each frame was then compiled into a `gif` file for visualization over time.