# Numerical Simulations of Particle Tracks in Detector

King Zixuan Lin

Undergraduate Thesis
in
Mathematical and Scientific Computation

Supervisor: Martin Fraas

Department of Mathematics

University of California, Davis

Spring 2023

**Abstract:** In Quantum Mechanics, a direct measurement on a particles position will inherently collapses its wavefunction and other subsequent measurement on the particle will become uncertain. One way to avoid such uncertainties is to make indirect measurements. In this paper, we consider a model of a quantum particle propagating in a cloud chamber in which the ionization of cloud chamber particles will result in an approximate indirect measurement of the position of the particle. In particular, for every interval of time $\tilde{t}$ during the free evolution of the particle as described by the Schrödinger Equation and represented by the unitary operator $\exp\big(-it(\hat{P}^2 + V(x))\big)$, we will be indirectly measuring the particles position which is represented by the jump operator $\hat{V}_\xi = (1/(\sigma\sqrt{2\pi})^{1/2})\exp\big(-(X-\xi)^2/4\sigma^2\big)$. Specifically in this paper, we will be producing a visual representation of the position of the particle to simulate the particle's track. To better understand the particle's track, we first consider the case where the particle is not bound by any potential ($V(x) = 0$), and later the case where the particle is bound by some potential ($V(x) \neq 0$).

# Contents

# 1 Introduction

## 1.1 Motivation

Ever since the first appearance of quantum theory in the 1920s, there have been ongoing discussion on how quantum mechanics is related to classical mechanics. Quantum mechanics is a probability theory which leads to numerous features that seem to defy logic. However, as of today all experiments confirm predictions of quantum mechanics. Then does that makes classical mechanics wrong? Not necessarily. Quantum mechanics just provides a more general description of physical phenomenons. The relationship between classical and quantum mechanics have been studied thoroughly since the beginning of quantum theory. One possible connection between classical mechanics and quantum mechanics is seen through the problem of classical trajectories emerging from quantum systems. This problem was initially proposed by Einstein and was named as a Mott problem after a theoretical work by Mott in 1929 [3]. Mott was able to explain the $\alpha$-particle tracks seen in a Wilson Cloud Chamber using second-order stationary perturbation theory and considering the particle-environment interaction inside the cloud chamber [1, 2].

Originally, the cloud chamber was a device that was able to physically show the particles trajectory as it passes through the device. This is achieved through particle interaction with the environment inside the chamber. To be more specific, supersaturated water vapor are placed inside the sealed device and the molecules are in a state right before the liquid-gas phase transition. Once the particle enters the cloud chamber, it interacts with the molecules inside the device triggering the phase transition to occur. Now suppose the same set-up is kept where supersaturated water vapor is placed inside the chamber, but instead, we place a radioactive element in the center of the chamber. This radioactive element will go through decay emitting an $\alpha$-particle that will propagate inside the chamber. The same interaction happens and leaves a visible trajectory of the path it took. In this paper, we want to simulate this process and study its behavior. We consider a model outlined in "*The appearance of particle tracks in detectors*" by Ballesteros, Benoist, Fraas, and Fröhlich [1].

## 1.2 Background: Quantum Postulates

Quantum Mechanics is truly odd in the sense that it is counter-intuitive. In light of this, it nevertheless provides a complete description of physical systems in our world. One may think that Classical mechanics (Newtonian mechanics) already explains physical phenomenon we see everyday and this is true to some extent. Classical mechanics can be viewed as a limiting case of Quantum mechanics, and what Classical mechanics fails to explain that Quantum mechanics succeed in is the description of physical systems in the microscopic world, such as one particle interacting with another particle. Moreover, an aspect that differentiate Classical mechanics and Quantum mechanics is the concept of superposition, entanglement, and measurements. We will now go over three important Quantum Postulates [4] that put these ideas into a mathematical framework. We should note there are more than three Quantum postulates, and that there is no universal agreement as to what the specific postulates are. We only present the postulates that are relevant for this paper.

### 1.2.1 Postulate 1: State Space

A physical system is completely described by its wavefunction $\psi(x,t)$ or its ket $|\psi\rangle$ which resides in a vector space known to mathematicians and physicists as the Hilbert space. Hilbert space is a linear space with a scalar product that is complete with respect to the corresponding distance function. We will use the Hilbert space of square integrable functions denoted by $L^2(\mathbb{R})$. Additionally, wavefunctions satisfy the normalization relation

$$\langle\psi|\psi\rangle = 1 \tag{1.1}$$

This implies that the state vector in the vector space is a unit vector.

It is useful to consider an example. Let us consider a qubit which is a two-level system with possible states $|0\rangle$ and $|1\rangle$. Quantum mechanics tells us that before a measurement has occurred, the two-level system is in a state of superposition of all possible states

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1.2}$$

Indeed, if $|\psi\rangle$ obeys (1.1) then it follows that

$$\begin{aligned}
\langle\psi|\psi\rangle &= (\alpha|0\rangle + \beta|1\rangle)^\dagger \cdot (\alpha|0\rangle + \beta|1\rangle) && (1.3)\\
&= (\alpha^*\langle 0| + \beta^*\langle 1|) \cdot (\alpha|0\rangle + \beta|1\rangle) && (1.4)\\
&= \alpha^*\alpha\langle 0|0\rangle + \alpha^*\beta\langle 0|1\rangle + \beta^*\langle 1|0\rangle + \beta^*\beta\langle 1|1\rangle && (1.5)\\
&= |\alpha|^2 + |\beta|^2 && (1.6)\\
&= 1 && (1.7)
\end{aligned}$$

Hence, we see that for a state the coefficients modules squared sums to 1. Consequently, the coefficient modules squared determines the probability of measuring that particular state as we will see in the third postulate.

### 1.2.2 Postulate 2: Evolution

The second postulate explains how a system evolves through time. For a closed system, the state $|\tilde{\psi}\rangle$ at time $t_1$ is evolved according to the Schrödinger equation for some initial state $|\psi\rangle$ at time $t_0$. The mathematical formalism for this is

$$|\tilde{\psi}\rangle = U|\psi\rangle \tag{1.8}$$

where $U$ is a unitary transformation operator.

***Definition:*** *A unitary operator $U : \mathcal{H} \to \mathcal{H}$ on some Hilbert space $\mathcal{H}$ satisfies the following property*

$$U^\dagger U = UU^\dagger = I \tag{1.9}$$

*where $I$ is the identity operator.*

In Quantum Computing, unitary transformations appears everywhere and is often referred to as Quantum Gates. Examples of some common unitary transformations in quantum computing are the $X, Y, Z$, and Hadamard. The $X, Y$, and $Z$ are $\pi$ rotations around their respective

axis on the Bloch sphere. Meanwhile, the Hadamard is a special transformation that puts a state into superpositions. Let us consider the same two-level system whose possible state is $|0\rangle$ and $|1\rangle$. Suppose the initial state is $|\psi\rangle = |0\rangle$, then applying the Hadamard give us

$$|0\rangle \rightarrow \frac{|0\rangle + |1\rangle}{\sqrt{2}} \tag{1.10}$$

As we will see in section 2 the unitary transformation operator we are interested in is the free evolution operator

$$U = \exp\left(-\frac{i}{\hbar}\mathbf{H}t\right) \tag{1.11}$$

where $\mathbf{H}$ is the Hamiltonian of the system.

### *1.2.3 Postulate 3: Quantum Measurements*

The third postulate describes the unique concept of performing quantum measurements on a physical system. Let $\{M_n\}$ be a set of measurement operators and let $|\psi\rangle$ be the state before a measurement has occurred. Then the probability of measuring the state with outcome result $m$ is

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle \tag{1.12}$$

and the post-measurement state is

$$\left|\tilde{\psi}\right\rangle = \frac{M_m|\psi\rangle}{\sqrt{p(m)}} \tag{1.13}$$

Additionally, it must be true that the sum of all probabilities $p(m)$ is 1, thus

$$1 = \sum_m p(m) \tag{1.14}$$

$$= \sum_m \langle\psi|M_m^\dagger M_m|\psi\rangle \tag{1.15}$$

$$= \langle\psi|\sum_m M_m^\dagger M_m|\psi\rangle \tag{1.16}$$

Therefore, it follows from the above equation that

$$\sum_m M_m^\dagger M_m = I \tag{1.17}$$

This is known as the *completeness relation.*

## 1.3 Aims and Expectations

In this simulation we have four objectives. First is to observe the stochastic process from this model. Second is to confirm that regardless of the particle's initial wavefunction, over time it will converge to a distinct gaussian shaped curve that depends on $\sigma$ and time step $\tilde{t}$. Third is to observe the classical-like tracks that emerges out of this quantum system as discussed in section 1.1. Fourth is to observe how the wave function behaves if we add a potential, specifically, a double-peaked finite potential well. In theory, we should observe the wave function tunneling through the finite potential at some point in time.

# 2 Methods

In this section, we will go through the theory and numerics concerning our model. First we will outline the stochastic process of the model step by step. Then we will explore the theory of this stochastic process which is known as the *Markov Process* and we will end this section with a discussion of the numerical solution to the Schrödinger equation and the numerical implementation of the process in the MATLAB software.

## 2.1 Process of the Model

In this model, we consider a single particle on a 1D with a Hilbert space $L^2(\mathbb{R})$. We denote $\mathbf{X}$ and $\mathbf{P}$ to be the position and momentum operator, respectively. For a single particle, we initialize the particle's state wavefunction $\psi(x)$ at time $t = 0$ to be some square integrable function.

***Definition:*** *Let $\psi : \mathbb{R} \to \mathbb{C}$ be some function. Then $\psi$ is square integrable if and only if*

$$\int_{\mathbb{R}} |\psi(x)|^2 \, dx < \infty \tag{2.1}$$

*We denote the space of all such functions by $L^2(\mathbb{R})$.*

As a remark, we can think of it as wavefunctions living in the Hilbert space. An example of a square integrable wavefunction is $\psi(x) = x^2 e^{-x^2}$. It can easily be check by taking a direct integration and utilizing integration by parts method. This function is rather special and will appear frequently throughout the paper as our initial wavefunction. Recall the first postulate of quantum mechanics where a physical system is completely described by the state vector and the probability of the system being in a particular state is given by the coefficient modulus squared. In our case, the wavefunction represent a continuous distribution of all possible states and the probability of the system being in a state corresponding to position $\tilde{x}$ is simply $|\psi(\tilde{x})|^2$. Therefore, the sum of all probability must be 1 giving us the condition

$$\int |\psi(x)|^2 \, dx = 1 \tag{2.2}$$

This is also known as normalizing the wavefunction and we define the normalized wavefunction to be

$$\Psi(x) = \frac{1}{\int |\psi(x)|^2 dx} \psi(x) \tag{2.3}$$

We make the assumption that for every time step $\tilde{t}$, the particle will evolve according to the Schrödinger equation

$$i\hbar \frac{\partial \Psi}{\partial t} = -\frac{\hbar^2}{2m} \frac{\partial^2 \Psi}{\partial x^2} + V\Psi \tag{2.4}$$

subject to the initial condition $\Psi(x, 0) = \Psi(x)$. The numerical method to solve the Schrödinger equation will be explored in section 2.3. Moreover, using the definition of the momentum

operator and squaring it $\mathbf{P}^2 = -\hbar^2 \frac{\partial^2}{\partial x^2}$, we can rewrite the Schrödinger equation as

$$i\hbar \frac{\partial \Psi}{\partial t} = -\frac{\hbar^2}{2m} \frac{\partial^2 \Psi}{\partial x^2} + V\Psi \qquad \Rightarrow \qquad i\hbar \frac{\partial \Psi}{\partial t} = \frac{\mathbf{P}^2}{2m}\Psi + V\Psi \tag{2.5}$$

$$\Rightarrow \qquad \frac{\partial \Psi}{\partial t} = -\frac{i}{\hbar}\left[\frac{\mathbf{P}^2}{2m} + V\right]\Psi \tag{2.6}$$

$$\Rightarrow \qquad \frac{\partial \Psi}{\partial t} = -\frac{i}{\hbar}\mathbf{H}\Psi \tag{2.7}$$

$$\Rightarrow \qquad \Psi(t) = \exp\left(-\frac{i}{\hbar}\mathbf{H}t\right) \tag{2.8}$$

where $\mathbf{H}$ is the Hamiltonian of the particle. For this model, this is simply the sum of the particle's kinetic energy and potential energy. As a result, we introduce an equivalent formalism to the free evolution during time step $\tilde{t}$. Let $\mathbf{U}$ be a unitary evolution operator such that it satisfy the Schrödinger equation. Then $\mathbf{U} = e^{-i\hbar\mathbf{H}\tilde{t}}$ and the evolution through time step $\tilde{t}$ is

$$\Psi(x,1) = \mathbf{U}\Psi(x,0) = \exp\left(-\frac{i}{\hbar}\mathbf{H}\tilde{t}\right)\Psi(x,0) \tag{2.9}$$

subject to the initial condition $\Psi(x,0) = \Psi(x)$.

One of the goals in this numerical simulation is to see classical trajectories appear from a quantum system. That is we expect to observe a classical trajectory from multiple successive position measurements on the particle. However, these measurements must not be direct as it will collapse the wavefunction, therefore, we instead apply an indirect measurement on the particle. This is achieved by randomly sampling a probability distribution. Let $\xi$ be the position measured. We introduce the jump operator

$$V_\xi(x) = \frac{1}{(\sigma\sqrt{2\pi})^{1/2}} \exp\left(-\frac{(X-\xi)^2}{4\sigma^2}\right) \tag{2.10}$$

Although in a different notation, this jump operator is precisely our measurement operator $M_m$ as described in the third postulate. Acting the jump operator to our evolved state, we define our probability distribution to be

$$p(\xi) = \int_{\mathbb{R}} |V_\xi(x)\Psi(x,1)|^2 \, dx \tag{2.11}$$

and the position obtained is the $\xi$ randomly sampled from $p(\xi)$. This process is repeated $n$ times with the post measurement wavefunction defined as

$$\psi(x) = V_\xi(x)\Psi(x,1) \tag{2.12}$$

After $n$ successive measurements, we have the position track $(\xi_1, \xi_2, \ldots, \xi_n)$ and the final wavefunction up to normalization $A$ at time $\tau_n = n\tilde{t}$ is

$$\Psi(x,\tau_n) = AV_{\xi_n} \exp\left(-i\hbar\mathbf{H}\tilde{t}\right) \ldots V_{\xi_2} \exp\left(-i\hbar\mathbf{H}\tilde{t}\right) V_{\xi_1} \exp\left(-i\hbar\mathbf{H}\tilde{t}\right)\Psi(x) \tag{2.13}$$

It is worthy to note that for our simulations, constants $\hbar$ and $m$ are taken to be 1 for simplicity.

## 2.2 Markov Process

The step in which we define on equation (2.12) is a recursive step that appears in a process known as Markov Process. We now state the general definition of a Markov Process, then we will see how Markov process is applied to our model. In section 2.2, we separated each measurements by some time step $\tilde{t}$. Hence, we are only concerned with the discrete-time Markov process.

***Definition:*** *Let $\boldsymbol{S}$ be some state space and some sequence of random variables $\{X_0, X_1, X_2, \dots\}$. A process $\{X_0, X_1, X_2, \dots\}$ is a Markov process if it satisfies the Markov condition*

$$\mathbb{P}(X_{n+1} = s_{n+1} | X_0 = s_0, X_1 = s_1, \dots, X_n = s_n) = \mathbb{P}(X_{n+1} = s_{n+1} | X_n = s_n) \tag{2.14}$$

*for $s_0, s_1, \dots, s_n, s_{n+1} \in \boldsymbol{S}$.*

Additionally, we will also define the transition matrix. For a discrete number of random variables, a transition matrix is useful to visualize the probabilities of transitioning from one state to another.

***Definition:*** *A transition matrix $\boldsymbol{P}$ is a $|\boldsymbol{S}| \times |\boldsymbol{S}|$ matrix with entries $p_{ij}$*

$$\boldsymbol{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & \cdots \\ p_{21} & p_{22} & p_{23} & \cdots \\ p_{31} & p_{32} & p_{33} & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix} \tag{2.15}$$

*where $p_{ij}$ are the transition probabilities defined as*

$$p_{ij} = \mathbb{P}(X_{n+1} = j | X_n = i) \tag{2.16}$$

*Moreover, a transition matrix has the following properties*

1. *Every entries $p_{ij}$ in matrix $\boldsymbol{P}$ are non-negative, $p_{ij} \geq 0$ for all $i, j$,*
2. *The sum of each row equals to 1, $\sum_j p_{ij} = 1$ for all $i$*

To put this into practice, let us consider a simple example of a random walk. Let our state space be $\mathbf{S} = \{0, 1, 2, 3, \dots\}$. Suppose we have the following transition probabilities

$$p_{ij} = \begin{cases} p & \text{if } j = i + 1, \ i = 1, 2, 3, \dots \\ q = 1 - p & \text{if } j = i - 1, \ i = 1, 2, 3, \dots \\ 1 & \text{if } i = 0, \ j = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.17}$$

Then we have the following transition matrix

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \dots \\ 1-p & 0 & p & 0 & 0 & \dots \\ 0 & 1-p & 0 & p & 0 & \dots \\ 0 & 0 & 1-p & 0 & p & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \tag{2.18}$$

In this random walk example, this is indeed a Markov process since the next state is purely determined by the current state. If the current state is $i = 3 \in \mathbf{S}$, then the next state is determined by probabilities $p$ and $1 - p$ for outcomes states $i = 4$ and $i = 2$, respectively. Additionally, if the state is $i = 0$, then regardless of previous states, it will stay at $i = 0$.

Now, we will see how our model is a Markov process. Let our state space be the Hilbert space $L^2(\mathbb{R})$. Then if the process, as described in section 2.1, is indeed a Markov process then it must satisfy the Markov condition

$$\mathbb{P}(\Psi_{n+1}|\Psi_n, \Psi_{n-1}, \ldots, \Psi_1, \Psi_0) = \mathbb{P}(\Psi_{n+1}|\Psi_n) \tag{2.19}$$

Indeed, according to equation (2.12), the next state is given by $\Psi_{n+1} = \frac{\tilde{\Psi}_{n+1}}{\int |\tilde{\Psi}_{n+1}|^2 dx}$ where $\tilde{\Psi}_{n+1} = V_\xi \exp(-i\mathbf{H}t)\Psi_n$ and $\Psi_n$ is the current state of the system. This is the recursive relation that we are looking for in a Markov process; where the outcome state is only dependent on the current state of the system and independent of any previous states. Additionally, the probability that the system will transition into some state $\Psi_{n+1}$ for some $\tilde{\xi} \in \mathbb{R}$ is given by equation (2.11)

$$\mathbb{P}(\Psi_{n+1}|\Psi_n) = p(\tilde{\xi}) = \int_{\mathbb{R}} |V_{\tilde{\xi}}\Psi_n|^2 \, dx \tag{2.20}$$

and only depends on state $\Psi_n$. Therefore, the model is a Markov process implying that the model is stochastic. It is also important to note that the model is somewhat different from the regular Markov process as seen in the example above. This difference is a consequence from performing measurements at each recursive step, where the next state $\Psi_{n+1}$ is only determined when making a state transition and reversibility does not happen. In other words, this process only evolves forward and does not jump backwards in states.

## 2.3 Numerical Solution and MATLAB Implementation

In this section, we will go through the numerical method of solving the Schrödinger equation as described in equation (2.9) and see how this is implemented in the MATLAB software. The method we have chosen to solve the Schrödinger equation is through the Fourier transform method. Without a present potential, the Fourier transform is sufficient in giving a solution to the Schrödinger equation. However, when a potential is present, we use the Trotter Kato product formula in addition to the Fourier transform method to give an approximate solution to the Schrödinger equation. Lastly, we will briefly explore the implementation of performing an measurement in MATLAB.

### *2.3.1 Solving the Schrödinger Equation and the Implementation*

We begin with the analytical solution to the Schrödinger equation for no potential, then we follow up the solution by adding a potential and using the Trotter Kato product formula. The MATLAB implementation, Algorithm 1, of the solution uses the combination of Fourier transform and Trotter-Kato product formula. If there is a potential present, then this implementation will give us an approximate solution. On the other hand, if there is no potential then $V(x)$ in Algorithm 1 will be set to 0 and the algorithm will just become Fourier transform method without the Trotter Kato aspect.

***Definition:*** *Let $x$ be our position space and $p$ be our momentum space. Then*

$$\mathcal{F}\{\Psi(x,t)\} = \hat{\Psi}(p,t) = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \Psi(x,t) \exp(-ipx)\ dx \tag{2.21}$$

*is the Fourier Transform of $\Psi(x,t)$, and*

$$\mathcal{F}^{-1}\{\hat{\Psi}(p,t)\} = \Psi(x,t) = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \hat{\Psi}(p,t) \exp(ipx)\ dp \tag{2.22}$$

*is the Inverse Fourier Transform of $\hat{\Psi}(p,t)$*

Consider the simpler form of the Schrödinger equation where constants such as $m, \hbar$ have been set to 1 and potential $V(x) = 0$,

$$\frac{\partial \Psi}{\partial t} = i \frac{\partial^2 \Psi}{\partial x^2} \tag{2.23}$$

on $x \in (-\infty, \infty)$ and $t \in [0, \infty)$ subject to the initial condition $\Psi(x,0) = \Psi(x)$. Let $\hat{\Psi}(p,t)$ be the Fourier transform of our solution $\Psi(x,t)$. Then taking advantage of Fourier's identity of a derivative

$$\mathcal{F}\left\{ \frac{\partial^2}{\partial x^2} \Psi(x,t) \right\} = -p^2 \hat{\Psi}(p,t) \tag{2.24}$$

Then substituting into equation (2.23), we have the equation in the momentum space

$$\frac{\partial \hat{\Psi}(p,t)}{\partial t} = -ip^2 \hat{\Psi}(p,t) \tag{2.25}$$

subject to the Fourier transformed initial condition $\hat{\Psi}(p,0) = \hat{\Psi}(p)$. Notice that the PDE becomes an ODE and this ODE is in the form of a trivial ODE with the solution

$$\hat{\Psi}(p,t) = A(p) \exp\left(-ip^2 t\right) \overset{I.C.}{\Longrightarrow} \hat{\Psi}(p,t) = \hat{\Psi}(p,0) \exp\left(-ip^2 t\right) \tag{2.26}$$

We now state the *Convolution Theorem* as this will prove useful in the derivation of the solution.

***Definition:*** *Let $f(x)$ and $g(x)$ be some arbitrary function. The convolution is a binary operation on $f, g$ such that*

$$(f * g)(x) = \int_{\mathbb{R}} f(x-\gamma) g(\gamma)\ d\gamma \tag{2.27}$$

***Theorem:*** *Let $f(x)$ and $g(x)$ be two arbitrary function with the Fourier transform $\hat{f}(p)$ and $\hat{g}(p)$. Then*

$$\mathcal{F}^{-1}\left\{\hat{f}(p)\hat{g}(p)\right\} = \mathcal{F}^{-1}\{\hat{f}(p)\} * \mathcal{F}^{-1}\{\hat{g}(p)\} = (f * g)(x) \tag{2.28}$$

*where $*$ denotes the convolution binary operation.*

***Proof.*** *We want to show* $\mathcal{F}^{-1}\left\{\hat{f}(p)\hat{g}(p)\right\} = (f * g)(x)$. *Let* $f(x)$ *and* $g(x)$ *be two functions with F.T.* $\hat{f}(p)$ *and* $\hat{g}(p)$. *We consider the Fourier transform of their convolution,* $\mathcal{F}\{(f * g)(x)\}$. *Using the definition of convolution and Fourier transform, we have*

$$\begin{aligned}
\mathcal{F}\{(f * g)(x)\} &= \mathcal{F}\left\{\int_{\mathbb{R}} f(x-\gamma)g(\gamma)\ d\gamma\right\} && (2.29)\\
&= \int_{\mathbb{R}}\int_{\mathbb{R}} f(x-\gamma)g(\gamma)\exp(-ipx)\ d\gamma dx && (2.30)\\
&= \int_{\mathbb{R}}\int_{\mathbb{R}} f(y)g(\gamma)\exp(-ipy)\exp(-ip\gamma)\ d\gamma dy && (2.31)\\
&= \int_{\mathbb{R}} f(y)\exp(-ipy)\ dy \cdot \int_{\mathbb{R}} g(\gamma)\exp(-ip\gamma)\ d\gamma && (2.32)\\
&= \mathcal{F}\{f(y)\}\cdot\mathcal{F}\{g(\gamma)\} && (2.33)\\
&= \hat{f}(p)\hat{g}(p) && (2.34)
\end{aligned}$$

*Thus we have the relation,* $\mathcal{F}\{(f * g)(x)\} = \hat{f}(p)\hat{g}(p)$. *Taking the Inverse Fourier transform, we have*

$$\mathcal{F}^{-1}\left\{\mathcal{F}\{(f * g)(x)\}\right\} = \mathcal{F}^{-1}\{\hat{f}(p)\hat{g}(p)\} \Rightarrow (f * g)(x) = \mathcal{F}^{-1}\{\hat{f}(p)\hat{g}(p)\} \quad (2.35)$$

*Hence, we have proved our desired result.* ■

Consequently, an alternative form to the theorem is $\mathcal{F}\{(f*g)(x)\} = \hat{f}(p)\hat{g}(p)$. Therefore, using this relation and taking the Inverse Fourier transform of equation (2.26), we have

$$\begin{aligned}
\mathcal{F}^{-1}\left\{\hat{\Psi}(p,t)\right\} &= \mathcal{F}^{-1}\left\{\hat{\Psi}(p,0)\cdot\exp\left(-ip^2t\right)\right\} && (2.36)\\
\Psi(x,t) &= \mathcal{F}^{-1}\left\{(\hat{\Psi}(p,0)\cdot 1)\cdot\exp\left(-ip^2t\right)\right\} && (2.37)\\
\Psi(x,t) &= \mathcal{F}^{-1}\left\{\mathcal{F}\{(\Psi * 1)(x,0)\}\cdot\exp\left(-ip^2t\right)\right\} && (2.38)\\
\Psi(x,t) &= \mathcal{F}^{-1}\left\{\mathcal{F}\{\Psi(x,0)\}\cdot\exp\left(-ip^2t\right)\right\} && (2.39)
\end{aligned}$$

Hence, we arrive at our solution to the Schrödinger equation with the abstract form

$$\Psi(x,t) = \mathcal{F}^{-1}\left\{\mathcal{F}\{\Psi(x,0)\}\cdot\exp\left(-ip^2t\right)\right\} \quad (2.40)$$

Expressing Fourier transform as an integral, the analytical solution becomes (with a change of variable to denote a different integration for $\mathcal{F}\{\Psi(x,0)\}$)

$$\begin{aligned}
\Psi(x,t) &= \mathcal{F}^{-1}\left\{\mathcal{F}\{\Psi(x,0)\}\cdot\exp\left(-ip^2t\right)\right\} && (2.41)\\
&= \mathcal{F}^{-1}\left\{\frac{1}{\sqrt{2\pi}}\int_{\mathbb{R}}\Psi(y,0)\exp(-ipy)\exp\left(-ip^2t\right)\ dy\right\} && (2.42)\\
&= \frac{1}{2\pi}\int_{\mathbb{R}}\int_{\mathbb{R}}\Psi(x,0)\exp(ipx)\exp(-ipy)\exp\left(-ip^2t\right)\ dp\ dy && (2.43)
\end{aligned}$$

Although we arrived at equation (2.43) with the Fourier transform method, it is also possible to derive the same solution by considering separable solution to the Schrödinger equation (time-independent and time-dependent solutions), and noting that the general solution is a linear

combination of continuous momentum variable $p$ instead of some discrete states as seen in some common quantum mechanics examples. Even though we derived an analytical solution, we can obtain a numerical solution by simply considering a discrete Fourier transform rather than a continuous one as we will see later.

We will now consider the Schrödinger equation with a potential. The potential of interest to us is the double-peaked finite potential well. This particular potential will allow us to see how the wavefunction will behave as it goes near the potential and potentially see the unique effect of quantum tunneling that is impossible classically. For our analytical solution, we will simply consider a general potential $V(x)$. First we will state a necessary theorem from "*Methods of Modern Mathematical Physics*" by Reed and Simon [5] for the approximate solution to the Schrödinger equation with a potential.

***Theorem (Lie-Trotter-Kato product formula):*** *Let $A$ and $B$ be self-adjoint operators on the Hilbert space $\mathcal{H}$ and suppose $C = A + B$ is self-adjoint on $D = D(A) \cap D(B)$. Then*

$$s - \lim_{n\to\infty} \left(\exp\left(\frac{it}{\Delta}A\right)\exp\left(\frac{it}{\Delta}B\right)\right)^{\Delta} = \exp(itC) \tag{2.44}$$

We should note that $D$ is the domain restriction depended on the intersection of the domain $A$ and domain $B$. In our model, we have $A \coloneqq p^2$ and $B \coloneqq V$ and since $V$ is defined in the whole Hilbert space, the restriction is immediately satisfy. Recall equation (2.9), for arbitrary time $t$, the evolution is defined as $\Psi(x,t) = \exp(-i\mathbf{H}t)\Psi(x,0)$ where the hamiltonian is defined as $\mathbf{H} = p^2 + V$ where $V$ is the potential and constants have been set to 1. Applying the Lie-Trotter-Kato product formula, we can approximate equation (2.9) as

$$\begin{aligned}
\Psi(x,t) &= \exp(-i\mathbf{H}t)\Psi(x,0) && (2.45)\\
&= \exp\big(-it(p^2+V)\big)\Psi(x,0) && (2.46)\\
&\approx \underbrace{\exp\big(-itp^2/\Delta\big)\exp(-itV/\Delta)\ldots\exp\big(-itp^2/\Delta\big)\exp(-itV/\Delta)}_{\Delta \text{ times of } \exp\left(-itp^2/\Delta\right)\exp(-itV/\Delta)}\Psi(x,0) && (2.47)
\end{aligned}$$

The solution to equation (2.47) is rather straight forward. We simple consider applying equation (2.40) $n$ times. In other words, we solve for $\exp\big(-itp^2/n\big)\Psi(x,0)$ with equation (2.40) and multiple it with $\exp(-itV/n)$ and define that to be $\Psi'$. Using $\Psi'$ as our $\Psi(x,0)$ we iterate this process $n$ times with equation (2.40) at which we will arrive at an approximate solution to $\Psi(x,t)$.

The MATLAB implementation is shown in Algorithm 1. This implementation considers a general potential $V(x)$, and uses both the Fourier transform method and Trotter-Kato product formula. For a numerical approximation to equation (2.40), we use the built in MATLAB function Fast Fourier transform (FFT) and Inverse Fast Fourier transform (IFFT) which is equivalent to a discrete Fourier transforms. Moreover, with our chose of discrete Fourier transforms, we need to specify an interval for which FFT and IFFT will be evaluated. The interval range we have chosen is $(\tilde{x} - 40, \tilde{x} + 40)$ for some center $\tilde{x}$ with spacing $\delta x = 0.02$. Lastly, our chose of $\Delta$ is 20 as this is a sufficient amount of iterations.

**Algorithm 1** Solution to Schrödinger Equation

**procedure** SCHRÖDINGER SOLN($\Psi_n$ , $V(x)$)
  $\mathbf{v} \leftarrow \Psi_n$
  **while** $j \leq n$ **do**
    $\Delta \leftarrow 20$
    **while** $k \leq \Delta$ **do**
      **v'** ← `compute fast fourier transform of s`
      **v'** ← `evolve s' with time evolution operator`
      **v** ← `compute inverse fast fourier transform of s'`
      **v** ← `add potential` $V(x)$ `for a time discretize by` $\Delta$
      $k = k + 1$
    **end while**
    $j = j + 1$
  **end while**
  `return` $\Psi_n$
**end procedure**

### *2.3.2 Measurement Implementation*

The implementation of (2.10) and (2.11) is shown in Algorithm 2. This procedure isn't as technical as Algorithm 1 as it just simply involves MATLAB evaluations of functions. However, we must consider every step numerically. Since (2.10) is a function of both $\xi$ and $x$, we represent the jump operator in MATLAB as a 2-D array. The probability distribution is obtained by taking a numerical integration with spacing $\delta x = 0.02$. Additionally, the random sampling of $\xi$ is acquired through sampling the 1D array of positions with weighted probabilities as described by (2.11).

**Algorithm 2** Position Measurement

**procedure** MEASUREMENT($\Psi_n$, $\sigma$)
  $\mathbf{v} \leftarrow \Psi_n$
  $\mathbf{s} \leftarrow \sigma$
  **while** $j \leq n$ **do**
    $\mathbf{V} \leftarrow V_\xi(x)$
    **v'** ← $\mathbf{V}\mathbf{v}$
    $p(\xi)$ ← `probability distribution defined as` $\int |\mathrm{v'}|^2 dx$
    $\vec{\xi}$ ← `randomly sample` $p(\xi)$
    $j = j + 1$
  **end while**
  `return` $\vec{\xi}$
**end procedure**

# 3 Results

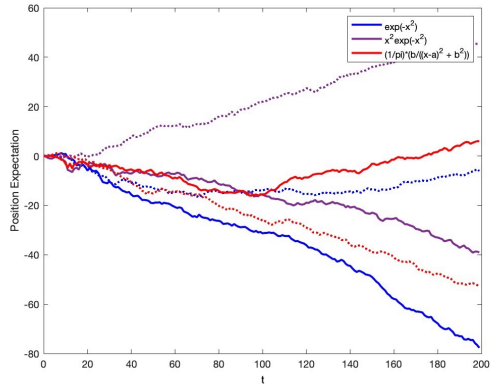
In this section, we display some simulations of various cases and discuss the four objectives mention in section 1.3. Before we begin, we state some useful equations used in the generation of the plots below. First, we define the position expectation to be

$$\langle x \rangle = \langle \Psi | x | \Psi \rangle = \int_{\mathbb{R}} x |\Psi|^2 \, dx \tag{3.1}$$

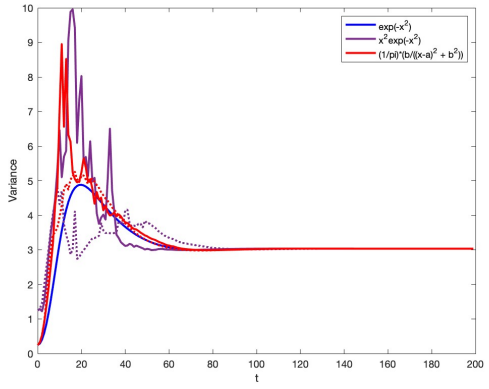
and this can be numerically approximated with a summation since we discretized $x$ and considered a finite interval. Second, we also define the variance to be

$$Var(x) = \int_{\mathbb{R}} (x - \mu)^2 f(x) \, dx \tag{3.2}$$

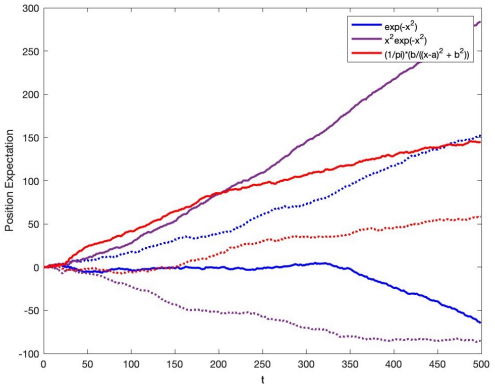
In our model, $\mu$ is simply the position expectation $\langle x \rangle$ and $f(x)$ is our wavefunction modulus squared $|\Psi|^2$.



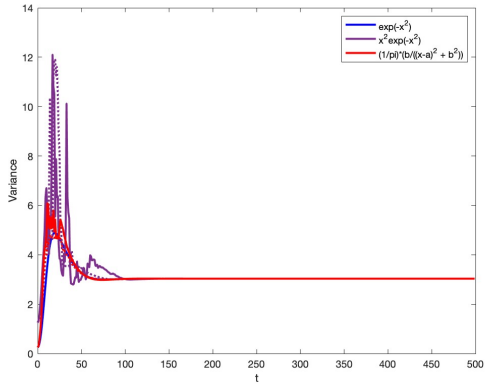

**(a)** Expectation Plot (Simulation 1)




**(b)** Variance Plot (Simulation 1)



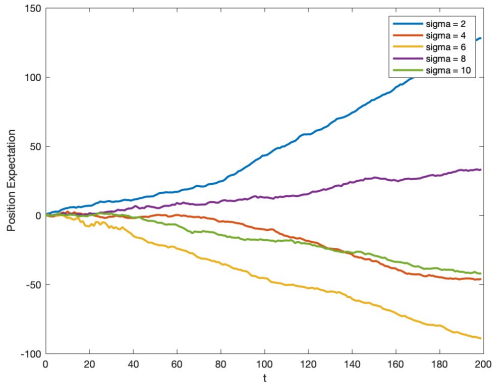

**(c)** Expectation Plot (Simulation 2)



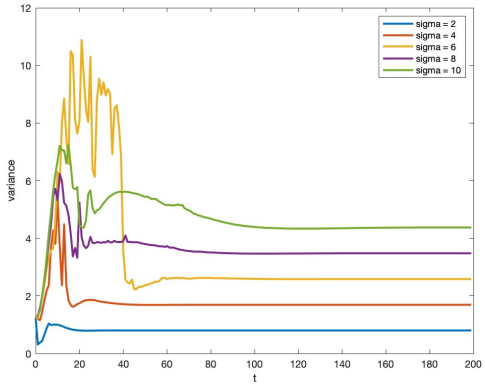

**(d)** Variance Plot (Simulation 2)

**Figure 1:** Left and right shows the position expectation and variance plot, respectively, for three different initial state (Blue: $\psi = \exp(-x^2)$, Purple: $\psi = x^2 \exp(-x^2)$, Red: $\frac{1}{\pi} \frac{b}{(x-a)^2+b^2}$ with a=0, b=0.5). The solid line denotes one simulation and the dotted line denotes another simulation of the same initial state. The simulation is generated with parameters $\sigma = 7$, $\tilde{t} = 0.1$, and $V(x) = 0$. The top two plot ran with 200 time step iteration while the bottom two plot ran with 500 iterations. Note: The top two plot are from the same simulations, while the bottom two are from another simulation.

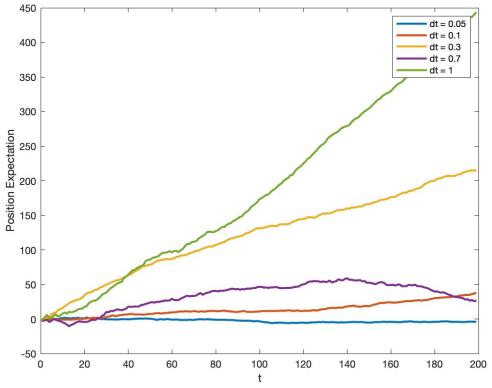
The first objective is to observe the stochastic process as described in section 2.2. We chose initial states such that it is centered at zero, and from postulate 3 the probability of the state being on the left or right is 50%. Therefore, as the wavefunction evolves through time, the particles trajectory is random and not predetermined. As seen in figure 1 and 2, the position expectation plots depicts the randomness of this process. Although on the larger scale, the behavior trend might look similar and predetermined, each curve represents a different simulation that branches off in a direction not known to us. Moreover, these expectation plots are the classical trajectories that we seek in our third objective. It's classical nature refers to the distinct direction and speed the particle propagated in. More importantly, the particle "picks up" this velocity at random throughout its evolution and once the particle has some velocity it moves in that direction in a straight line. This is beautifully captured in figure 1. Figure 1a shows the trajectories for 200 time step iterations simulation runs and has rough curves with a somewhat noticeable velocity. However, if we ran another simulation for a longer period of time (500 iterations), these trajectories becomes more distinct in their speed and direction.



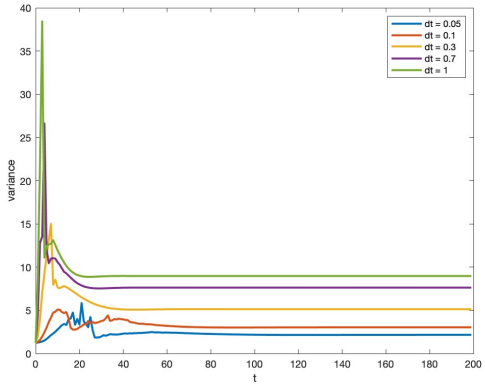

**(a)** Position Expectation (Simulation 3)




**(b)** Variance (Simulation 3)




**(c)** Position Expectation (Simulation 4)



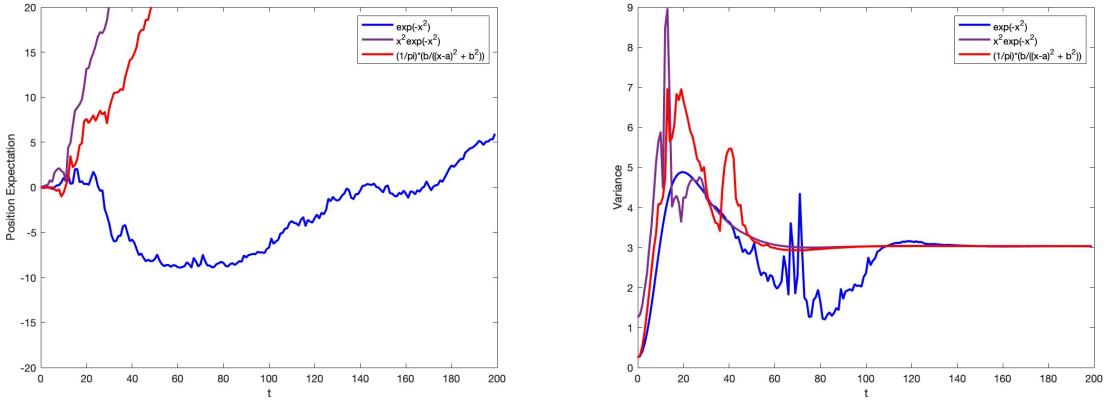

**(d)** Variance (Simulation 4)

**Figure 2:** Left and right shows the position expectation and variance plot, respectively, for the initial state $\psi = x^2 \exp(-x^2)$ with varying $\sigma$ (top) and time step $\tilde{t}$ (bottom). The simulation is generated with parameters $\sigma = 7$ (bottom only), $\tilde{t} = 0.1$ (top only), and $V(x) = 0$. Both simulation is ran with 200 time step iterations.

Our second objective was to observe the wavefunction converging to a distinct gaussian

curve that depends on $\sigma$ and $\tilde{t}$. The variance plot shows precisely the convergence of the wavefunction over time. In figure 1b, we see very noisy curves in the beginning that eventually disappear around time step iteration 80. Also, this convergence is not due to some off chance that occur from a random process. We ran another simulation for a longer period of time and the result holds as seen in figure 1d. Additionally, we ran simulation for initial state $\psi = x^2 \exp\left(-x^2\right)$ with varying time step $\tilde{t}$ and $\sigma$. The outcome we see in figure 2b and 2d displays a different convergence as we vary those parameters. Moreover, the time it takes to converge also depends on the parameters as seen in the two figures.

Lastly we simulate with a potential present, specifically, a double finite potential well. Figure 4 shows the wavefunctions at different time. The tunneling effect can be seen in the top right plot in figure 4 where the red curve begins to tunnel while the purple curve has tunneled through. This can also be seen in the trajectory plot where the red and purple curve leaves the confined region while the whole blue curve is still present. Additionally, the converge of the variance does not occur until the particle fully tunnels through the potential barrier as seen in figure 3b.




**(a)** Position Expectation (Simulation 5)

**(b)** Variance (Simulation 5)

**Figure 3:** Left and right shows the position expectation and variance plot, respectively, for three different initial state (Blue: $\psi = \exp\left(-x^2\right)$, Purple: $\psi = x^2 \exp\left(-x^2\right)$, Red: $\frac{1}{\pi}\frac{b}{(x-a)^2+b^2}$ with a=0, b=0.5). The simulation is generated with parameters $\sigma = 7$, $\tilde{t} = 0.1$, and $V(x) = 2$. The simulation is ran for 200 time step iterations.

It is also worthy to note the difference between a quantum system with measurement and without measurement. Figure 4 and 5 describes the symmetry broken as a result of performing a measurement on a quantum system. We know that the wavefunction is centerd at zero. With a measurement, the wavefunction symmetry about the center is broken as the whole curve moves left or right. Meanwhile, without the measurement the wavefunction's symmetry is preserved with more oscillation as it evolves in time. In a sense, it becomes more like a wave and exhibits properties of a wave.

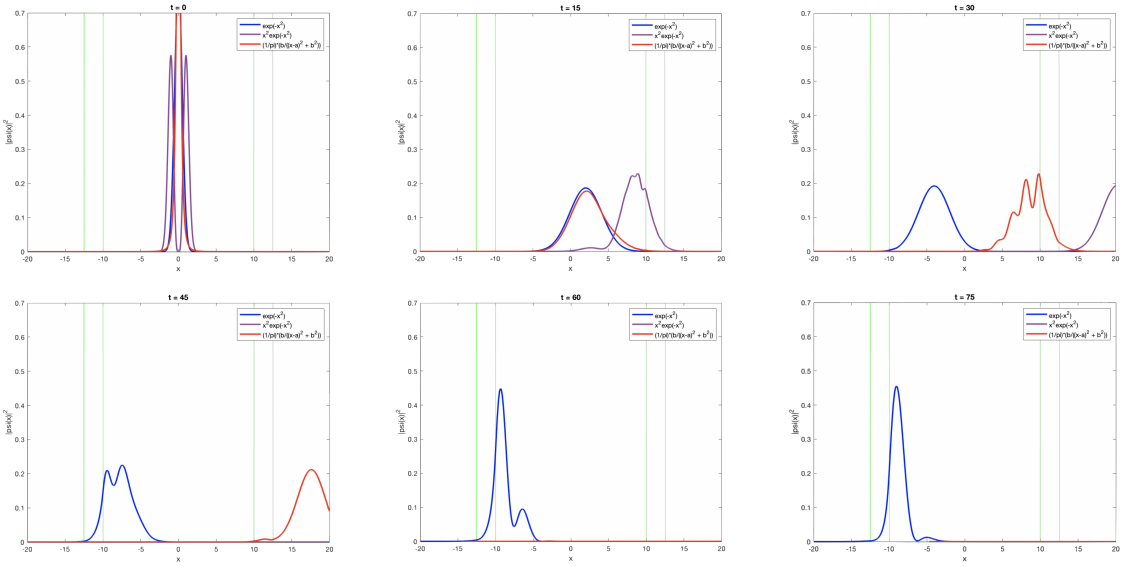



**Figure 4:** Time step iteration at 0, 15, 30, 45, 60, 75 for simulation 5.

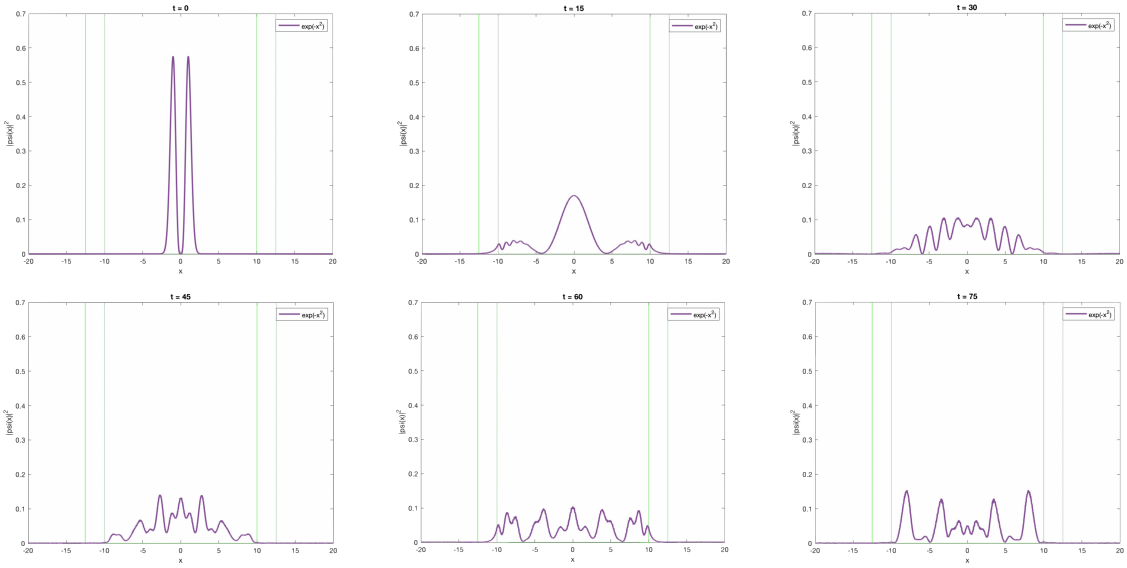



**Figure 5:** Simulation with no measurement of initial state $\psi = x^2 \exp(-x^2)$

# 4 Conclusion

As demonstrated in this paper, we have simulated particle tracks in a detector and observed classical trajectories emerging from a quantum system. Specifically, we considered a stochastic model that represents the state of a particle by some arbitrary square integrable function. An assumption is made on the evolution such that during time step $\tilde{t}$, the wavefunction evolves according to the free evolution operator 2.8 as described by the Schrödinger equation. An

additional assumption was made for simplicity where each measurement separated by the time step $\tilde{t}$. Moreover, we have seen that with figure 4 and 5, symmetry seen in classical systems is broken as a result of performing a measurement. Measurement in this model is represented by a gaussian jump operator 2.10.

We have explored various cases such as how the wavefunction behaves with and without potential, and its behavior over time with varying parameters $\sigma$ and time step $\tilde{t}$. Most importantly, we have simulated the propagation of a particle and tracked its position expectation which showed a line with a distinct direction and speed which is what we expect from a classical trajectory.

This paper simulates the specific case of a particle propagating in a 1D space. However, a particle in a cloud chamber can also propagate in all direction in a 3D space. In other words, we have placed a restriction on the particle's propagation direction in this simulation. This restriction can be lifted by considering a 3D Schrödinger equation and solving it in all three coordinate space. In doing so, it will model a particle's track in a more realistic scenario and will allow us to better understand its behavior. Additionally, we have considered a measurement operator that was gaussian. However, not all detectors can be represented in a gaussian operator, thus we can extend this work to simulate trajectories for a non-gaussian measurement operator.

## Acknowledgement

I would like to take this chance to thank Professor Martin Fraas for his patience and guidance throughout this project. The discussions we had were very informative and I have learned a lot from it. I would like to also thank the University Honors Program for giving me the push I needed to be involved in research.

## References

[1] M. Ballesteros, T. Benoist, M. Fraas, and J. Fröhlich. The appearance of particle tracks in detectors. *Communications in Mathematical Physics*, 385:429–463, 2021.

[2] R. Figari and A. Teta. Emergence of classical trajectories in quantum systems: the cloud chamber problem in the analysis of mott (1929). *Archive for history of exact sciences*, 67 (2):215–234, 2013.

[3] N. F. Mott. The wave mechanics of alpha-ray tracks. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 126(800): 79–84, 1929.

[4] M. A. Nielsen and I. Chuang. Quantum computation and quantum information, 2002.

[5] M. Reed and B. Simon. *Methods of Modern Mathematical Physics: Functional Analysis; Rev. ed.* Academic press, 1980.

# Appendix

## A. Solving Schrödinger Equation Code

```
function [psi] = Solve_Schrodinger_Eq(n,dx,dt,X,psi,potential)
    N = length(X(1,:));
    for j = 1:n
        t = dt;
        delta = 20;
        for m = 1:delta
            phi = fft(psi(j,:))*dx;
            dp = 2*pi/(N*dx);
            p = (-N/2:N/2-1)*dp;
            phi_shift = fftshift(phi);
            phi_shift_even = phi_shift(2:2:end)*-1;
            phi_shift(2:2:end) = phi_shift_even;
            time_evo = exp(-1i.*p.^2.*t/delta);
            phi = phi_shift.*time_evo;
            psi_time_evo = ifftshift(ifft(phi)/dx); %state psi after time
                evolution
            psi_time_evo_even = psi_time_evo(2:2:end)*-1;
            psi_time_evo(2:2:end) = psi_time_evo_even;
            psi_time_evo = psi_time_evo.*exp(-1i.*potential(j,:).*t/delta);
            psi(j,:) = psi_time_evo;
        end
    end
end
```

## B. Measurement Code

```
function [psi,xi] = Position_Meas(n,dx,X,C,xi,psi,sigma,k)
    for j = 1:n
        s = sigma;
        psi_time_evo = psi(j,:);
        jump_op = @(x,c) exp(-((x-c).^2)/(4*s^2))/(s^(1/2)*(2*pi)^(1/4));
        jump_op_eval = bsxfun(jump_op,X(j,:)',C(j,:));
        new_psi_time_evo = jump_op_eval.*psi_time_evo;
        prob_dist = trapz(abs(new_psi_time_evo).^2,2)*dx;

        pos_measure = datasample(C(j,:),1,"weights",prob_dist);

        xi(j,k) = pos_measure;
        psi(j,:) = jump_op(X(j,:),xi(j,k)).*psi_time_evo;
        clear pos_measure;
    end
end
```

## C. Interval Shift Code

```
function [psi,X,C,state_mean] = Interval_Shift(dx,n,X_initial,X,C,psi)
    state_mean = zeros(n,1);
    for j = 1:n
```

```
        center_indx = (X_initial(end)+dx-X_initial(1))/(2*dx)+1;
        state_mean(j,1) = sum(X(j,:).*abs(psi(j,:).^2))*dx;
        [~,mean_indx] = min(abs(X(j,:)-state_mean(j,1)));
        mean_X = X(j,mean_indx);
        X_diff = mean_indx-center_indx;
        X(j,:) = mean_X-40:dx:(mean_X+40)-dx;
        C(j,:) = mean_X-40:dx:(mean_X+40)-dx;
        if X_diff > 0 % moved to the right
            psi(j,:) = [psi(j,X_diff+1:end) psi(j,X_diff:-1:1)];
        elseif X_diff < 0 % moved to the left
            X_diff = abs(X_diff);
            psi(j,:) = [psi(j,end:-1:end-(X_diff-1)) psi(j,1:end-X_diff)];
        end
    end
end
```

## D. Generation Code

```
%% ------------ Three State Simulation with Movie/Position Expectation/Variance ------------ %%
clc; clear;
format short
workspace;

n = 6;
t = 500;
freq = 50;
dx = 1/freq;
X_initial = -40:dx:40-dx;
C_initial = -40:dx:40-dx;
N = length(X_initial);
sigma = 7;
dt = 0.1;
X = zeros(n,N);
C = zeros(n,N);
for j = 1:n
    X(j,:) = X_initial;
    C(j,:) = C_initial;
end

b = 0.5;
a = 0;
initial_psi1 = exp(-X_initial.^2);
initial_psi2 = X_initial.^2.*exp(-X_initial.^2);
initial_psi3 = (1/pi)*(b./((X_initial-a).^2 + b.^2));

potential = zeros(n,N);

state_psi = [initial_psi1;initial_psi1;initial_psi2;initial_psi2;initial_psi3;initial_psi3];
xi = zeros(n,t);
variance = zeros(n,t);
position_expectation = zeros(n,t);

allTheFrames = cell(1,t);
```

```
allTheColorMaps = cell(1,t);
myMovie = struct('cdata', allTheFrames, 'colormap', allTheColorMaps);

for k = 1:t
    for j = 1:n
        A = 1/sqrt(sum(abs(state_psi(j,:)).^2)*dx);
        state_psi(j,:) = A*state_psi(j,:);
    end

    [state_psi,X,C,state_mean] = Interval_Shift(dx,n,X_initial,X,C,state_psi);
    for j = 1:n
        position_expectation(j,k) = state_mean(j,1);
        temp_state = (X(j,:)-state_mean(j,1)).^2.*(abs(state_psi(j,:)).^2);
        variance(j,k) = sum(temp_state)*dx;
    end

    potential_val = potential;

    cla reset;
    ax = gca;
    hold on;
    plot(X(1,:),abs(state_psi(1,:)).^2,'LineWidth',2,'Color','b');
    hold on;
    plot(X(2,:),abs(state_psi(2,:)).^2,":",'LineWidth',2,'Color','b');
    hold on;
    plot(X(3,:),abs(state_psi(3,:)).^2,'LineWidth',2,'Color','#7E2F8E');
    hold on;
    plot(X(4,:),abs(state_psi(4,:)).^2,":",'LineWidth',2,'Color','#7E2F8E');
    hold on;
    plot(X(5,:),abs(state_psi(5,:)).^2,'LineWidth',2,'Color','r');
    hold on;
    plot(X(6,:),abs(state_psi(6,:)).^2,":",'LineWidth',2,'Color','r');
    hold on;
    temp_t = k-1;
    title("t = " + temp_t)
    legend("exp(-x^2)","","x^2exp(-x^2)","","(1/pi)*(b/((x-a)^2 + b^2))","")
    ax.YLim = ([0,0.7]);
    ax.XLim = ([-20,20]);
    xlabel('x')
    ylabel('|psi(x)|^2')
    drawnow;
    myMovie(k) = getframe(gcf);

    state_psi = Solve_Schrodinger_Eq(n,dx,dt,X,state_psi,potential_val);
    [state_psi,xi] = Position_Meas(n,dx,X,C,xi,state_psi,sigma,k);
end

Replay_Save_Movie(myMovie)

figure;
plot(0:t-1,position_expectation(1,:),'LineWidth',2,'Color','b');
hold on;
plot(0:t-1,position_expectation(2,:),":",'LineWidth',2,'Color','b');
hold on;
plot(0:t-1,position_expectation(3,:),'LineWidth',2,'Color','#7E2F8E');
hold on;
```

```
plot(0:t-1,position_expectation(4,:),":",'LineWidth',2,'Color','#7E2F8E');
hold on;
plot(0:t-1,position_expectation(5,:),'LineWidth',2,'Color','r');
hold on;
plot(0:t-1,position_expectation(6,:),":",'LineWidth',2,'Color','r');
ylabel('Position Expectation')
%ylim([-20,20])
legend("exp(-x^2)","","x^2exp(-x^2)","","(1/pi)*(b/((x-a)^2 + b^2))","")

figure;
plot(0:t-1,variance(1,:),'LineWidth',2,'Color','b');
hold on;
plot(0:t-1,variance(2,:),":",'LineWidth',2,'Color','b');
hold on;
plot(0:t-1,variance(3,:),'LineWidth',2,'Color','#7E2F8E');
hold on;
plot(0:t-1,variance(4,:),":",'LineWidth',2,'Color','#7E2F8E');
hold on;
plot(0:t-1,variance(5,:),'LineWidth',2,'Color','r');
hold on;
plot(0:t-1,variance(6,:),":",'LineWidth',2,'Color','r');
xlabel('t')
ylabel('Variance')
%ylim([-20,20])
legend("exp(-x^2)","","x^2exp(-x^2)","","(1/pi)*(b/((x-a)^2 + b^2))","")


%% ------------ Single State Simulation with Movie/Position Expectation/Variance
    Vary Sigma ------------ %%
clc; clear;
format short
workspace;

n = 5;
t = 500;
freq = 50;
dx = 1/freq;
X_initial = -40:dx:40-dx;
C_initial = -40:dx:40-dx;
N = length(X_initial);
sigma = [2,4,6,8,10];
dt = 0.1;
X = zeros(n,N);
C = zeros(n,N);
for j = 1:n
    X(j,:) = X_initial;
    C(j,:) = C_initial;
end

b = 0.5;
a = 0;
initial_psi = X.^2.*exp(-X.^2);


potential = zeros(n,N);

```

```
147
148 state_psi = initial_psi;
149 xi = zeros(n,t);
150 variance = zeros(n,t);
151 position_expectation = zeros(n,t);
152
153 allTheFrames = cell(1,t);
154 allTheColorMaps = cell(1,t);
155 myMovie = struct('cdata', allTheFrames, 'colormap', allTheColorMaps);
156
157 for k = 1:t
158     for j = 1:n
159         A = 1/sqrt(sum(abs(state_psi(j,:)).^2)*dx);
160         state_psi(j,:) = A*state_psi(j,:);
161     end
162
163     [state_psi,X,C,state_mean] = Interval_Shift(dx,n,X_initial,X,C,state_psi);
164     for j = 1:n
165         position_expectation(j,k) = state_mean(j,1);
166         temp_state = (X(j,:)-state_mean(j,1)).^2.*(abs(state_psi(j,:)).^2);
167         variance(j,k) = sum(temp_state)*dx;
168     end
169
170     potential_val = potential;
171
172
173     cla reset;
174     ax = gca;
175     hold on;
176     for j = 1:n
177         hold on;
178         plot(X(j,:),abs(state_psi(j,:)).^2,'LineWidth',2);
179         temp_t = k-1;
180         title("t = " + temp_t)
181         hold on;
182     end
183     legend("sigma = 2", "sigma = 4","sigma = 6","sigma = 8","sigma = 10")
184     ax.YLim = ([0,0.7]);
185     ax.XLim = ([-20,20]);
186     xlabel('x')
187     ylabel('|psi(x)|^2')
188     drawnow;
189     myMovie(k) = getframe(gcf);
190
191     state_psi = Solve_Schrodinger_Eq(n,dx,dt,X,state_psi,potential_val);
192     [state_psi,xi] = Position_Meas(n,dx,X,C,xi,state_psi,sigma,k);
193 end
194
195 Replay_Save_Movie(myMovie)
196
197 figure;
198 for j = 1:n
199     plot(0:t-1,position_expectation(j,:),'LineWidth',2)
200     xlabel('t')
201     ylabel('Position Expectation')
202     hold on;
```

```
end
legend("sigma = 2", "sigma = 4","sigma = 6","sigma = 8","sigma = 10")

figure;
for j = 1:n
    plot(0:t-1,variance(j,:),'LineWidth',2)
    xlabel('t')
    ylabel('variance')
    hold on;
end
legend("sigma = 2", "sigma = 4","sigma = 6","sigma = 8","sigma = 10")


%% ------------ Single State Simulation with Movie/Position Expectation/Variance Vary dt ------------ %%
clc; clear;
format short
workspace;

n = 5;
t = 500;
freq = 50;
dx = 1/freq;
X_initial = -40:dx:40-dx;
C_initial = -40:dx:40-dx;
N = length(X_initial);
sigma = 7;
dt = [0.05,0.1,0.3,0.7,1];
X = zeros(n,N);
C = zeros(n,N);
for j = 1:n
    X(j,:) = X_initial;
    C(j,:) = C_initial;
end

initial_psi = X.^2.*exp(-X.^2);


potential = zeros(n,N);


state_psi = initial_psi;
xi = zeros(n,t);
variance = zeros(n,t);
position_expectation = zeros(n,t);

allTheFrames = cell(1,t);
allTheColorMaps = cell(1,t);
myMovie = struct('cdata', allTheFrames, 'colormap', allTheColorMaps);

for k = 1:t
    for j = 1:n
        A = 1/sqrt(sum(abs(state_psi(j,:)).^2)*dx);
        state_psi(j,:) = A*state_psi(j,:);
    end

```

```
    [state_psi,X,C,state_mean] = Interval_Shift(dx,n,X_initial,X,C,state_psi);
    for j = 1:n
        position_expectation(j,k) = state_mean(j,1);
        temp_state = (X(j,:)-state_mean(j,1)).^2.*(abs(state_psi(j,:)).^2);
        variance(j,k) = sum(temp_state)*dx;
    end

    potential_val = potential;


    cla reset;
    ax = gca;
    hold on;
    for j = 1:n
        hold on;
        plot(X(j,:),abs(state_psi(j,:)).^2,'LineWidth',2);
        temp_t = k-1;
        title("t = " + temp_t)
        hold on;
    end
    legend("dt = 0.05", "dt = 0.1","dt = 0.3","dt = 0.7","dt = 1")
    ax.YLim = ([0,0.7]);
    ax.XLim = ([-20,20]);
    xlabel('x')
    ylabel('|psi(x)|^2')
    drawnow;
    myMovie(k) = getframe(gcf);

    state_psi = Solve_Schrodinger_Eq(n,dx,dt,X,state_psi,potential_val);
    [state_psi,xi] = Position_Meas(n,dx,X,C,xi,state_psi,sigma,k);
end

Replay_Save_Movie(myMovie)

figure;
for j = 1:n
    plot(0:t-1,position_expectation(j,:),'LineWidth',2)
    xlabel('t')
    ylabel('Position Expectation')
    hold on;
end
legend("dt = 0.05", "dt = 0.1","dt = 0.3","dt = 0.7","dt = 1")

figure;
for j = 1:n
    plot(0:t-1,variance(j,:),'LineWidth',2)
    xlabel('t')
    ylabel('variance')
    hold on;
end
legend("dt = 0.05", "dt = 0.1","dt = 0.3","dt = 0.7","dt = 1")


%% ------------ Three State Simulation with Movie/Position Expectation/Variance (
    Potential)------------ %%
```

```
clc; clear;
format short
workspace;

n = 3;
t = 200;
freq = 50;
dx = 1/freq;
X_initial = -40:dx:40-dx;
C_initial = -40:dx:40-dx;
N = length(X_initial);
sigma = 7;
dt = 0.1;
X = zeros(n,N);
C = zeros(n,N);
for j = 1:n
    X(j,:) = X_initial;
    C(j,:) = C_initial;
end

b = 0.5;
a = 0;
initial_psi1 = exp(-X_initial.^2);
initial_psi2 = X_initial.^2.*exp(-X_initial.^2);
initial_psi3 = (1/pi)*(b./((X_initial-a).^2 + b.^2));


V_0 = 2;
potential = @(y) V_0;
potential_val = potential(X_initial);
X_indx = 10<=abs(X_initial) & abs(X_initial)<=12.5;
initial_potential = potential_val.*X_indx;

state_psi = [initial_psi1;initial_psi2;initial_psi3];
xi = zeros(n,t);
variance = zeros(n,t);
position_expectation = zeros(n,t);

allTheFrames = cell(1,t);
allTheColorMaps = cell(1,t);
myMovie = struct('cdata', allTheFrames, 'colormap', allTheColorMaps);

for k = 1:t
    for j = 1:n
        A = 1/sqrt(sum(abs(state_psi(j,:)).^2)*dx);
        state_psi(j,:) = A*state_psi(j,:);
    end

    [state_psi,X,C,state_mean] = Interval_Shift(dx,n,X_initial,X,C,state_psi);
    for j = 1:n
        position_expectation(j,k) = state_mean(j,1);
        temp_state = (X(j,:)-state_mean(j,1)).^2.*(abs(state_psi(j,:)).^2);
        variance(j,k) = sum(temp_state)*dx;
    end

    potential_val = potential(X);
```

```
    X_indx = 10<=abs(X) & abs(X)<=12.5;
    potential_val = potential_val.*X_indx;

    cla reset;
    ax = gca;
    plot(X_initial,initial_potential,'Color','g')
    hold on;
    plot(X(1,:),abs(state_psi(1,:)).^2,'LineWidth',2,'Color','b');
    hold on;
    plot(X(2,:),abs(state_psi(2,:)).^2,'LineWidth',2,'Color','#7E2F8E');
    hold on;
    plot(X(3,:),abs(state_psi(3,:)).^2,'LineWidth',2,'Color','r');
    hold on;
    temp_t = k-1;
    title("t = " + temp_t)
    legend("","exp(-x^2)","x^2exp(-x^2)","(1/pi)*(b/((x-a)^2 + b^2))")
    ax.YLim = ([0,0.7]);
    ax.XLim = ([-20,20]);
    xlabel('x')
    ylabel('|psi(x)|^2')
    drawnow;
    myMovie(k) = getframe(gcf);

    state_psi = Solve_Schrodinger_Eq(n,dx,dt,X,state_psi,potential_val);
    [state_psi,xi] = Position_Meas(n,dx,X,C,xi,state_psi,sigma,k);
end

Replay_Save_Movie(myMovie)

figure;
plot(0:t-1,position_expectation(1,:),'LineWidth',2,'Color','b');
hold on;
plot(0:t-1,position_expectation(2,:),'LineWidth',2,'Color','#7E2F8E');
hold on;
plot(0:t-1,position_expectation(3,:),'LineWidth',2,'Color','r');
xlabel('t')
ylabel('Position Expectation')
ylim([-20,20])
legend("exp(-x^2)","x^2exp(-x^2)","(1/pi)*(b/((x-a)^2 + b^2))")

figure;
plot(0:t-1,position_expectation(1,:),'LineWidth',2,'Color','b');
hold on;
plot(0:t-1,position_expectation(2,:),'LineWidth',2,'Color','#7E2F8E');
hold on;
plot(0:t-1,position_expectation(3,:),'LineWidth',2,'Color','r');
xlabel('t')
ylabel('Position Expectation')
legend("exp(-x^2)","x^2exp(-x^2)","(1/pi)*(b/((x-a)^2 + b^2))")

figure;
plot(0:t-1,variance(1,:),'LineWidth',2,'Color','b');
hold on;
plot(0:t-1,variance(2,:),'LineWidth',2,'Color','#7E2F8E');
hold on;
plot(0:t-1,variance(3,:),'LineWidth',2,'Color','r');
```

```
xlabel('t')
ylabel('Variance')
legend("exp(-x^2)","x^2exp(-x^2)","(1/pi)*(b/((x-a)^2 + b^2))")



%% ------------ Single State Simulation with Movie/Position Expectation/Variance (Potential)------------ %%
clc; clear;
format short
workspace;

n = 1;
t = 200;
freq = 50;
dx = 1/freq;
X_initial = -40:dx:40-dx;
C_initial = -40:dx:40-dx;
N = length(X_initial);
dt = 0.1;
X = zeros(n,N);
C = zeros(n,N);
for j = 1:n
    X(j,:) = X_initial;
    C(j,:) = C_initial;
end

initial_psi2 = X_initial.^2.*exp(-X_initial.^2);

V_0 = 10;
potential = @(y) V_0;
potential_val = potential(X_initial);
X_indx = 10<=abs(X_initial) & abs(X_initial)<=12.5;
initial_potential = potential_val.*X_indx;

state_psi = initial_psi2;
xi = zeros(n,t);
variance = zeros(n,t);
position_expectation = zeros(n,t);

allTheFrames = cell(1,t);
allTheColorMaps = cell(1,t);
myMovie = struct('cdata', allTheFrames, 'colormap', allTheColorMaps);

for k = 1:t
    for j = 1:n
        A = 1/sqrt(sum(abs(state_psi(j,:)).^2)*dx);
        state_psi(j,:) = A*state_psi(j,:);
    end

    [state_psi,X,C,state_mean] = Interval_Shift(dx,n,X_initial,X,C,state_psi);
    for j = 1:n
        position_expectation(j,k) = state_mean(j,1);
        temp_state = (X(j,:)-state_mean(j,1)).^2.*(abs(state_psi(j,:)).^2);
        variance(j,k) = sum(temp_state)*dx;
    end
```

```
480
481     potential_val = potential(X);
482     X_indx = 10<=abs(X) & abs(X)<=12.5;
483     potential_val = potential_val.*X_indx;
484
485     cla reset;
486     ax = gca;
487     plot(X_initial,initial_potential,'Color','g')
488     hold on;
489     plot(X(1,:),abs(state_psi(1,:)).^2,'LineWidth',2,'Color','#7E2F8E');
490     temp_t = k-1;
491     title("t = " + temp_t)
492     legend("","exp(-x^2)")
493     ax.YLim = ([0,0.7]);
494     ax.XLim = ([-20,20]);
495     xlabel('x')
496     ylabel('|psi(x)|^2')
497     drawnow;
498     myMovie(k) = getframe(gcf);
499
500     state_psi = Solve_Schrodinger_Eq(n,dx,dt,X,state_psi,potential_val);
501 end
502
503 Replay_Save_Movie(myMovie)
```