Computational Physics Final Project

Solving Schrodinger equation

Mehdi Hatefipour

Fall 2018

Physics of the problem

In quantum mechanics, the Schrödinger equation is a mathematical equation that describes the changes over time of a physical system in which quantum effects, such as wave–particle duality, are significant. These systems are referred to as quantum (mechanical) systems. The equation is considered a central result in the study of quantum systems, and its derivation was a significant landmark in the development of the theory of quantum mechanics. It was named after Erwin Schrödinger, who derived the equation in 1925, and published it in 1926, forming the basis for his work that resulted in him being awarded the Nobel Prize in Physics in 1933.

Schrodinger equation could be written either in time-independent or in time-dependent format as well. In this project, both aspects have been considered. Indeed, half of this report is about time-independent SE and another half is about time-dependent SE.

How to solve time-independent Schrödinger equation (SE)?

Stationary states in quantum mechanics can be described by a simple form of the Schrödinger equation, the time-independent Schrödinger equation (TISE).

$$\left[-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}+V(x)\right]\Psi(x)=E\Psi(x)$$

where E is a constant equal to the total energy of the system. This is only used when the Hamiltonian itself is not dependent on time explicitly. However, even in this case the total wave function still has a time dependency.

In words, the equation states:

> When the Hamiltonian operator acts on a certain wave function Ψ, and the result is proportional to the same wave function Ψ, then Ψ is a stationary state, and the proportionality constant, E, is the energy of the state Ψ.

Now, it's time to see how it could be solved in computer language? TISE is a second order differential equation which can be solved by methods which exists like Euler,

Verlet, RK2, RK4 and so on. In this project, all the calculations for 2nd order diff. equation have been done by Verlet method.

Verlet Method

Verlet integration is a numerical method used to integrate Newton's equations of motion. It is frequently used to calculate trajectories of particles in molecular dynamics simulations and computer graphics. The algorithm was first used in 1791 by Delambre and has been rediscovered many times since then, most recently by Loup Verlet in the 1960s for use in molecular dynamics. It was also used by Cowell and Crommelin in 1909 to compute the orbit of Halley's Comet, and by Carl Størmer in 1907 to study the trajectories of electrical particles in a magnetic field (hence it is also called Störmer's method). The Verlet integrator provides good numerical stability, as well as other properties that are important in physical systems such as time reversibility and preservation of the symplectic form on phase space, at no significant additional computational cost over the simple Euler method.

For a second-order differential equation of the type $\ddot{\vec{x}}(t) = \vec{A}(\vec{x}(t))$ an approximate numerical solution can be obtained by the following method:

$$\vec{x}_{n+1} = 2\vec{x}_n - \vec{x}_{n-1} + \vec{A}(\vec{x}_n)\Delta t^2$$

Now, by rewriting the TISE with this method, the following equation would be obtained:

$$-\frac{\hbar^2}{2m}\frac{\psi_{n-1} - 2\psi_n + \psi_{n+1}}{d^2} + V_n\psi_n = E\psi_n$$

It’s obvious that for solving this equation numerically, space should be discretized. N stands for number of cells in the space with length of L. On the other hand, L has been taken 10 in this project, and it would be a fixed number for entire the project.

Now, the TISE could be written in matrix form. In that case, it would be like a eigenvalue problem which it comes in the following:

$$-\frac{\hbar^2}{2md^2}\begin{pmatrix} -2 & 1 & & & & & \\ 1 & -2 & 1 & & & & \\ & \ddots & & \ddots & & & \\ & & & & 1 & -2 & 1 \\ & & & & & 1 & -2 \end{pmatrix}\begin{pmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_{N-1} \\ \psi_N \end{pmatrix}$$

$$+\begin{pmatrix} V_1 & & & & \\ & V_2 & & & \\ & & \ddots & & \\ & & & V_{N-1} & \\ & & & & V_N \end{pmatrix}\begin{pmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_{N-1} \\ \psi_N \end{pmatrix} = E\begin{pmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_{N-1} \\ \psi_N \end{pmatrix}$$

Everything in this project is done by MATLAB. For instance, the above problem is an eigenvalue problem which its solution would determine Energy of the system and eigenstates of the problem. By sorting energies and its eigenstates respectively, ground state, 1st excited state, 2nd excited state and so on would be obtained. For solving this problem, eig() function in MATLAB was used.

Now, for a given potential the program has been written can determine all the states and energies. On the other hand, it is necessary to check the program with a test function. Since quantum harmonic oscillator has easy energy values, it was used for this check.

In the following table, the results for different value for N has been came:

n	Energy (exact)	Energy (N=50)	Energy (N=100)	Energy (N=250)
0	0.5	0.5	0.5	0.5
1	1.5	1.49	1.5	1.5
2	2.5	2.48	2.5	2.5
3	3.5	3.47	3.49	3.5
4	4.5	4.45	4.49	4.5
5	5.5	5.42	5.48	5.5
6	6.5	6.39	6.47	6.5
7	7.5	7.36	7.47	7.5

According to the above table, N=250 is a good candidate for this program. All the results which will come in the following has been taken by N=250. One of the goals which had been mentioned in the proposal was finding energies and eigenstates of ground state, 1st excited state, …, 4th excited state. All these results for different potential wells and barriers will come in the following. The interesting point is that this

program can solve every potential numerically with a reasonable precision which maybe is impossible to solve them analytically. For instance, in this project potentials like exponential potential or triangular potential have been solved. In the following pictures, first few eigenstates and eigenenergy of different potential shapes have been came.

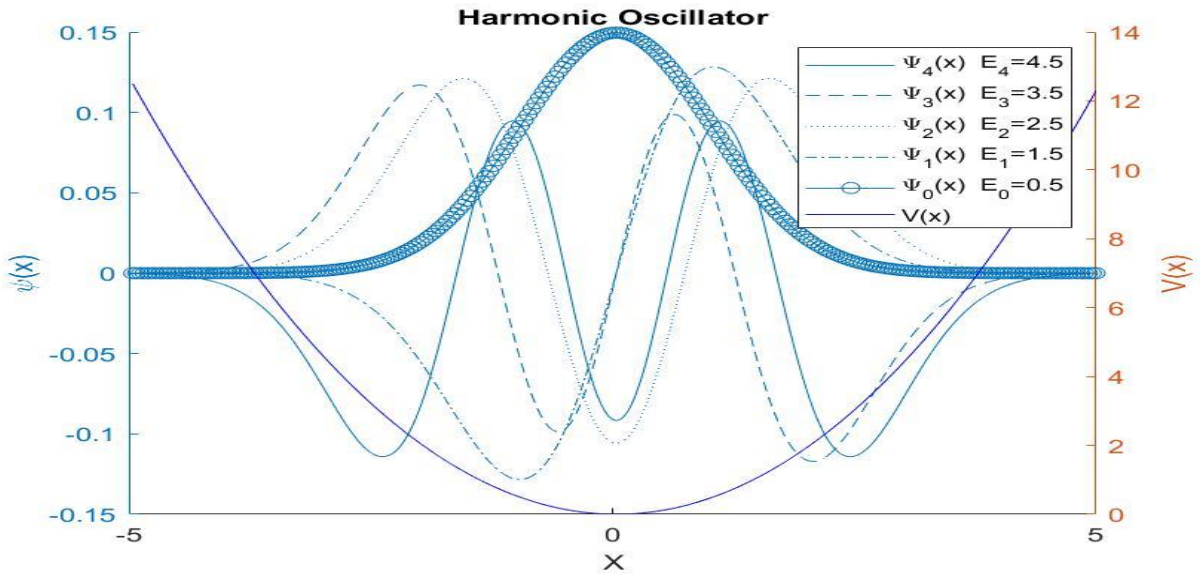


Figure 1 Harmonic Oscillator

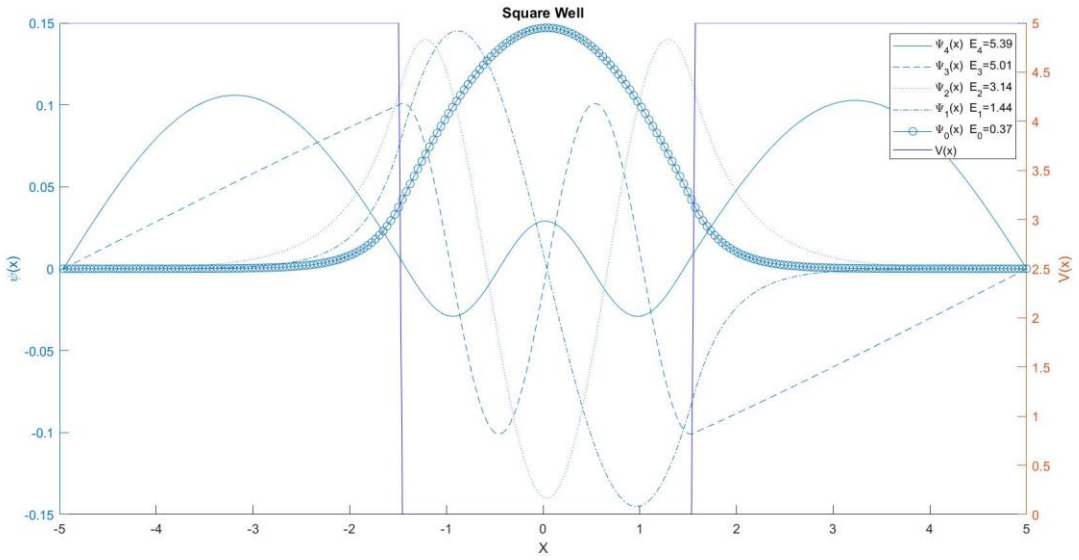


Figure 2Simple Square Well

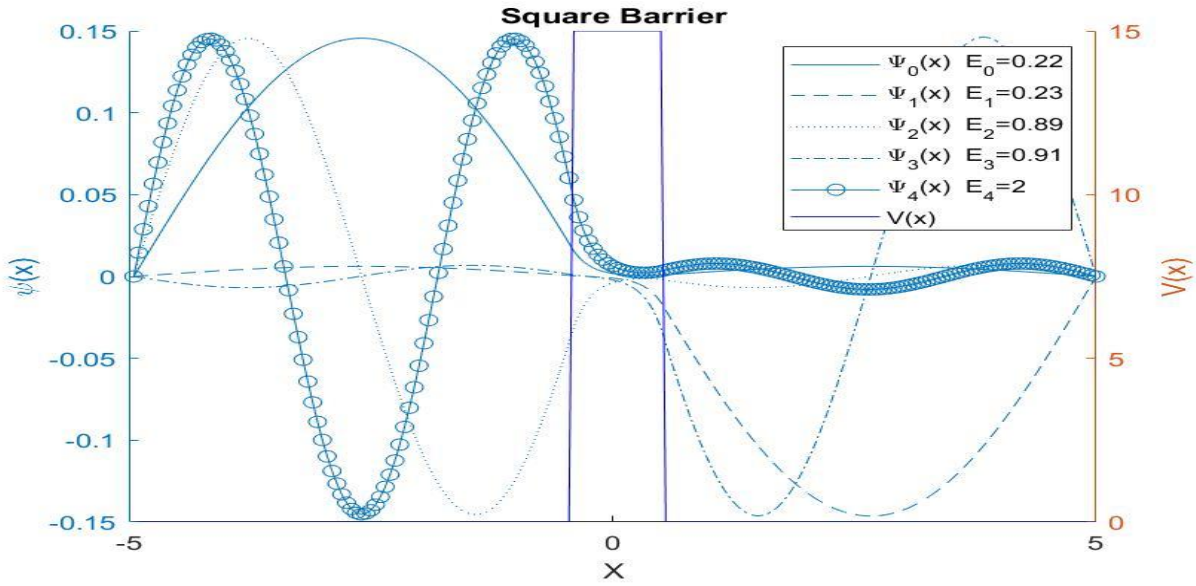


Figure 3 Simple Square Barrier

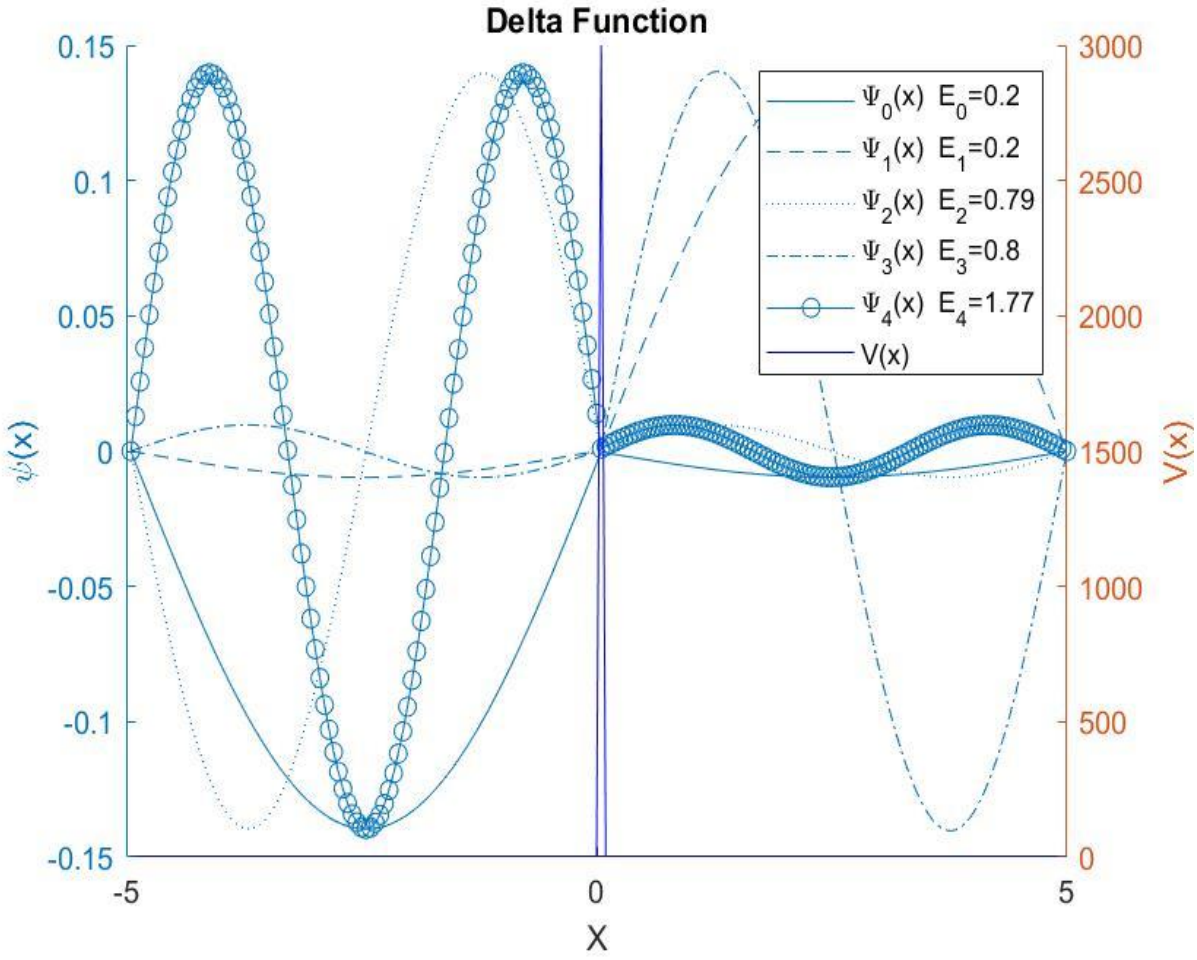


Figure 4 Delta Function

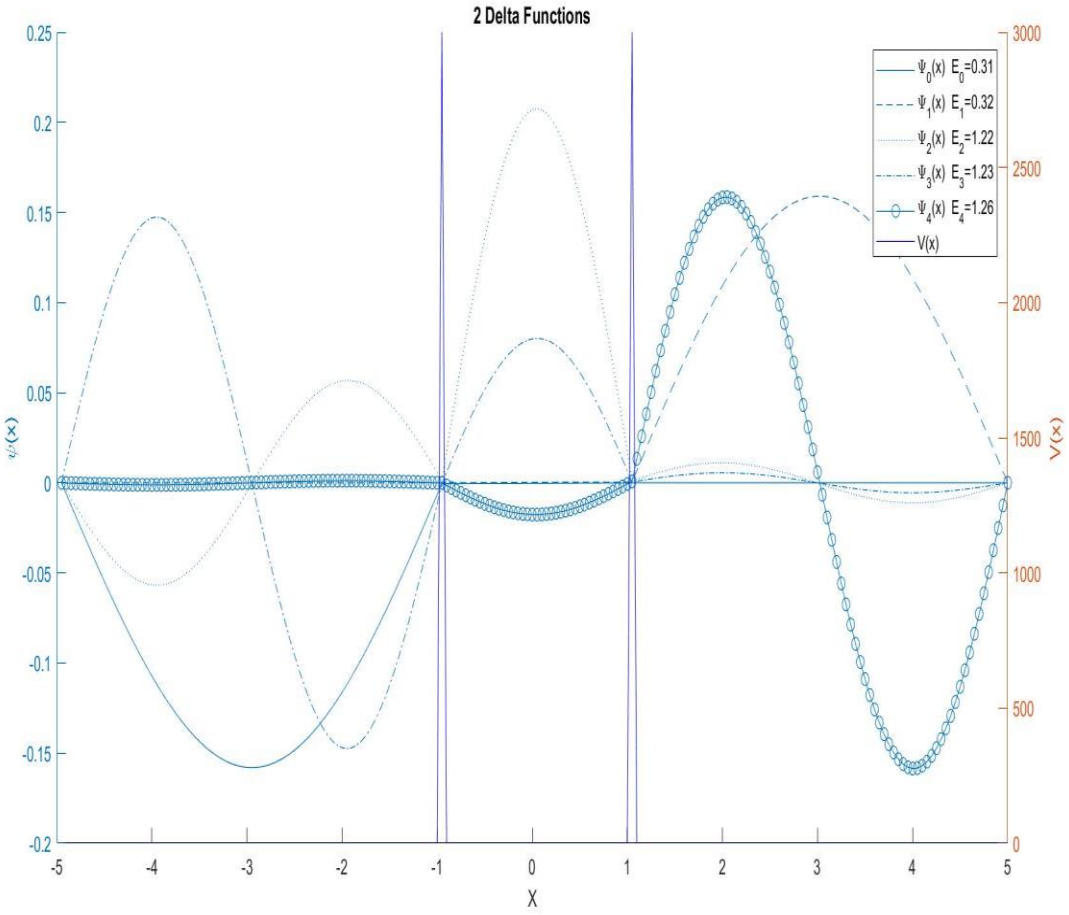


Figure 5 Two Delta Functions

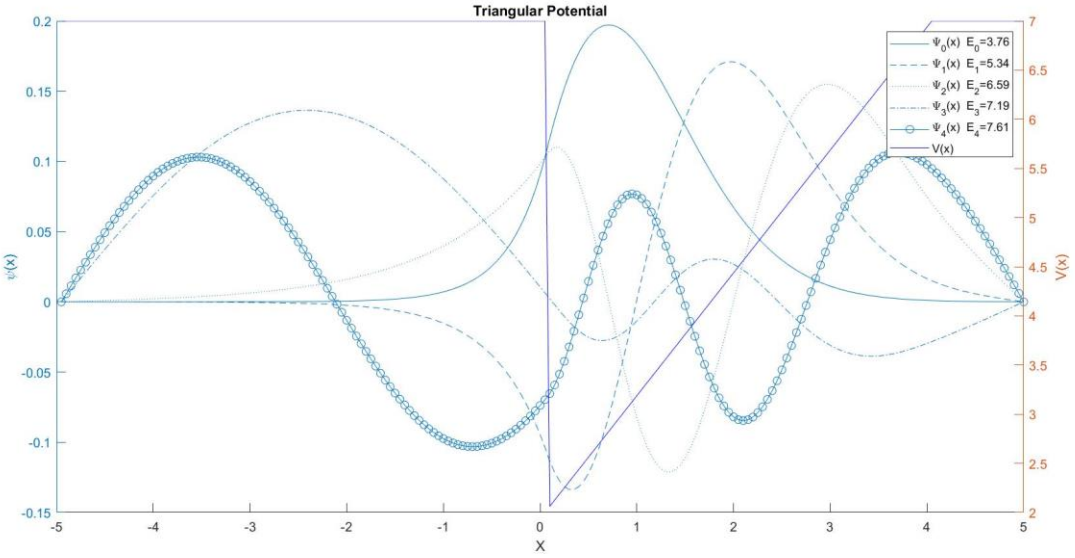


Figure 6 Triangular Potential

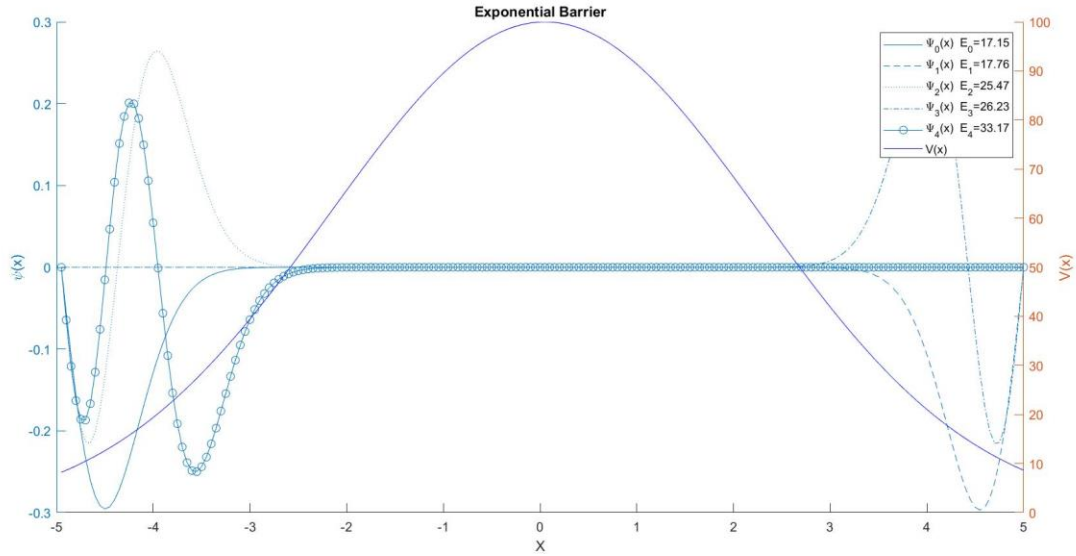


Figure 7 Exponential Barrier

Time evolution of some wavefunctions

The most general form is the time-dependent Schrödinger equation (TDSE), which gives a description of a system evolving with time. It could be written in terms of time-evolution operator:

$$\psi(t) = U(t)\psi(0),\ \mathrm{U(t)} = \exp(-\frac{\mathrm{iHt}}{\hbar})$$

There are two methods for solving this equation numerically in computer:

One is diagonalizing Hamiltonian which is like this: first, Hamiltonian should be written in diagonalize form $H = RH_DR^{\dagger}$, then time-evolution operator could be calculated in this form

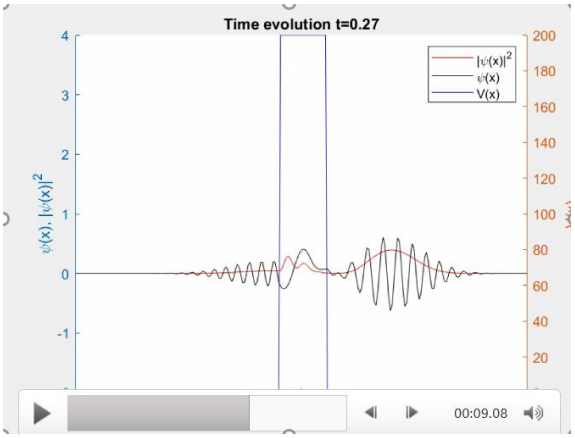
$$\mathrm{U(t)} = R\exp\left(-\frac{\mathrm{i}H_D\mathrm{t}}{\hbar}\right)R^{\dagger}$$

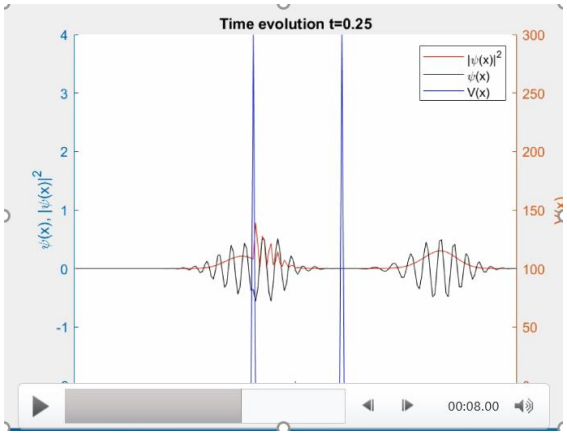
Second method is Taylor expansion: $\mathrm{U(t)} = \exp\left(-\frac{\mathrm{iHt}}{\hbar}\right) = \sum_{n=0}^{\infty}\frac{1}{n!}\left(-\frac{iHt}{\hbar}\right)^n$, but for doing this method, there is a sum over infinite terms and it should be truncated somewhere. It seems that first method is more accurate than second method. In this project, first method has been used and second method could be one of the future works.

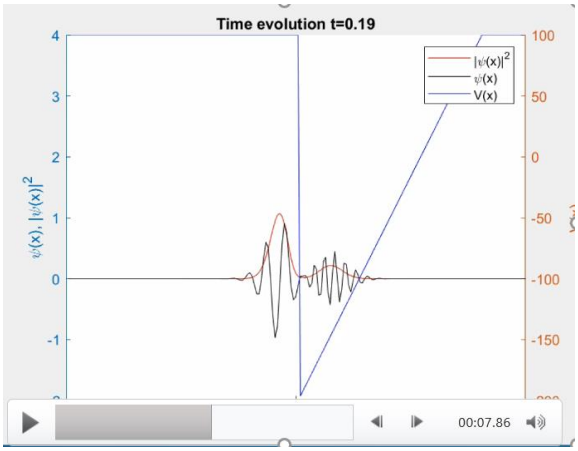
Now, there is a way for checking the solution for this part and that's again harmonic oscillator and its ground state. According to the theory of the problem, the expectation for time-evolution of ground state is just adding a phase. (All the animations for this part have been uploaded in git hub and some screen-shot of them have been put here).

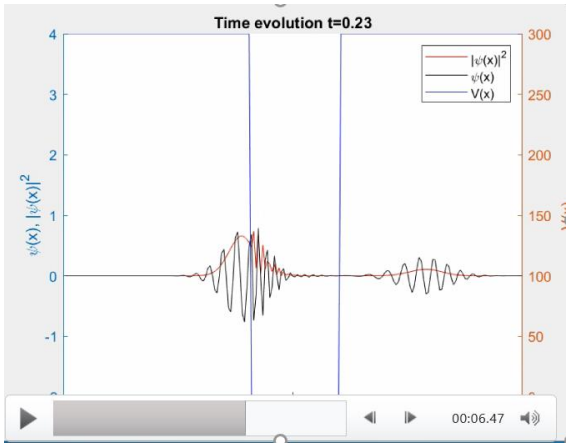
Figure 8 Screen shot of ground state time-evolution

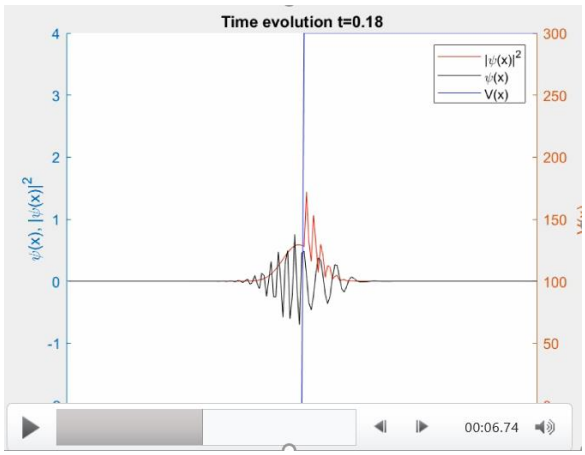
Now, it's time to see time-evolution of some function like exponential function and wave packet after putting them inside a potential well. According to quantum mechanics principles, there are tunneling, reflection, transition, etc. in this world in a certain condition. In the following, some screen-shots of the animations have been came.


Time evolution t=0.27
$|\psi(x)|^2$
$\psi(x)$
V(x)
$\psi(x), |\psi(x)|^2$
00:09.08


Time evolution t=0.25
$|\psi(x)|^2$
$\psi(x)$
V(x)
$\psi(x), |\psi(x)|^2$
00:08.00


Time evolution t=0.19
$|\psi(x)|^2$
$\psi(x)$
V(x)
$\psi(x), |\psi(x)|^2$
00:07.86


Time evolution t=0.23
$|\psi(x)|^2$
$\psi(x)$
V(x)
$\psi(x), |\psi(x)|^2$
00:06.47


Time evolution t=0.18
$|\psi(x)|^2$
$\psi(x)$
V(x)
$\psi(x), |\psi(x)|^2$
00:06.74

Future Works

In this project, the only method which was used for solving 2^{nd} order diff. equation was Verlet. This program could be written by RK4 to see the results, errors and run-time, and It would be a good comparison between these two methods for this problem.

Writing a GUI as user-friendly program has been initiated but because of the time challenge it was not finished. (GUI code has already been uploaded in git hub).