EN5004: Electronic Devices

Assignment : Modelling Time Independent Schrodinger Equation

Student Name : M.Shihar Halaldeen
Student ID : 208628A
University of Moratuwa
Department of Electronics and Telecommunications Engineering

May 30, 2021

Abstract

Obtaining solution for the Schrodinger's Equation is a widely researched topic in Quantum Mechanics. Schrodinger's equation is used to describe the behaviour of a quantum particle.The time in dependant Schrodinger's equation is considered in this report. The Schrodinger's equation presents the behaviour of the particle in a shape of a wave and also it presents the probability of finding a quantum particle in a given space. Shooting method and Eigen method is used to simulate the Schrodinger's equation for both symmetrical and asymmetrical potential wells. It was found out that the Eigen method is considered to be the most efficient method to solve the system of equations. The necessity of initial values and boundary values were a condition to simulate the particles behaviour in shooting method. The problems and errors observed during the simulation were discussed in addition to improvements which could make the simulation perform efficiently.

Contents

1 Introduction

Quantum Physics is the branch of physics which was used to describe the behaviour of quantum particles which the classical physics failed. The double slit experiments was one of the turning point which showed classical mechanics could not elaborately and efficiently explain the particles behaviour in that experiment. Thus the contributions of Quantum Physicist such as de Broglie , Plank and Schrodinger accelerated the field of Quantum Mechanics which was used to understand the behaviours of quantum particles.

This report focuses on simulating the dynamic behaviour of the mathematical time independent Schrodinger's Wave Equation for several different potential fields and study its dynamic characteristics under those fields. The Schrodinger's equation is a partial differential equation which is employed to describe the behaviour of quantum particles. This equation can be analogously related to the classical physics total energy which is a combination of both kinetic and potential energies.

2 Background

Schrodinger devised a formula which can be applied in order to understand these results obtained through electromagnetic waves and particles which was not possible to explain from classical physics. The equation devised by Schrodinger included the wave-particle duality principle of de Broglie and Quanta Principles of Planck. The Double slit experiment can be considered as a perfect scenario which shows that classical mechanics fails to predict the behaviour of electrons in the experiment.

The de Broglie Wave Particle duality presents that the particle's wavelength λ is inversely proportional to its linear momentum. [1]

$$\lambda = \frac{h}{p} \tag{1}$$

Where is h is plank's constant and p is the linear momentum of the particle. The wave function Ψ propagating in the z direction can be written as;

$$\Psi = exp(\frac{2\pi iz}{\lambda}) \tag{2}$$

This Ψ wave function should obey the Helmholtz Wave equation which can be written in the following form.

$$\frac{d^2\psi}{dz^2} = -k^2\psi \tag{3}$$

The above is presented for the 1D case the following is valid for the general case.

$$\Delta^2\psi = -k^2\psi \tag{4}$$

Using de Broglie hypothesis of $k = \frac{p}{\hbar}$

$$\frac{\hbar^2}{2m_e}\Delta^2\psi = \frac{p^2}{2m_e}\psi \tag{5}$$

From the analogy of classical mechanics it is known that total energy is the sum of kinetic energy and potential energy. From inspecting equation(5) the term $\frac{p^2}{2m_e}$ represents the Kinetic Energy therefore this can simplified as follows

$$\frac{p^2}{2m_e} = E - V(r) \tag{6}$$

Where E is the Total Energy and V is the Potential Energy.

$$-\frac{\hbar^2}{2m_e}\Delta^2\psi = (E - V(r))\psi \tag{7}$$

Therefore the Time-Independent Schrodinger equation is as follows;

$$-\frac{\hbar^2}{2m_e}\Delta^2\psi + V(r)\psi = E\psi \tag{8}$$

The Physical Importance of the wave function Ψ is such that it represents the probability of finding a quantum particle in a given dimension. Therefore from probability the normalized function should satisfy the following condition such that the total probability of finding a particle in a given space is 1.[2]

$$\int_{-\infty}^{+\infty} |\psi|^2 d\psi = 1 \tag{9}$$

3 Literature Review

Exploring techniques to solve the schrodinger equation is a highly sought field in Quantum Mechanics. Several methods and various tools were utilized in order to develop a visual understanding of the solution to Schrodinger's equation. The solutions of the Schrodinger's equation explained the behaviour of a quantum particle when a potential field is applied. It also helps to understand the energy levels of the particle for a particular potential.

The exact solution for Schrodinger equation is only limited to only a few potentials such as particle in a box. It should be understood that deriving exact solutions to certain potentials is time consuming and its not also possible. Therefore to derive solutions for the other potentials numerical methods are sought. There are several methods used to solve the Schrodinger's equation such as Runge-Kutta, Numerov algorithm or Variation method.[3]

Joel et al. emphasised the importance of using numerical methods to derive the solution for Schrodinger's equation.It was demonstrated for the 1D case that numerical methods can be used to solve for any potential and taking less than 40 lines BASIC code. The algorithms in numerical methods converge automatically and the accuracy in obtaining eigenvalues and eigenfucntions are high as well.[4]

Schrodinger equation can be solved as a boundary value problem if the boundary conditions are known. Taylor et al. presented that shooting method and finite difference approximation schemes can be considered as suitable numerical methods to solver schrodinger type of equations. In other words second order differential equations. it was further clarified the importance of numerical methods are deployed when it is hard and impossible to obtain analytical solution for the given equation. The numerical solution obtained will have a certain degree of approximation depending on the numerical method applied. This approximation error is the cost paid to obtain a solution through numerical methods.[5]

Numerov integration method is another technique to solve the second order Schrodinger's equation. Joel et al. presented that even when closed form solutions are known when the need for a tabulated wave function arises Numerov method can be used to generate it more quickly and efficiently. This scheme is considered to be efficient as it requires only four slow operations to perform. A brief summary of Numerov method can be presented as a trial Eigenvalue is adopted and by trial and error method and satisfying the boundary conditions the corresponding eigenvector spanning the classical and non-classical regions to both sides are found.[4]

The unavailability of resources to investigate the different impacts of of solving Schrodinger's equation in infinite , finite double finite harmonic , kronig-penny finite potential energy wells for all these led Matthew and Shiv to develop a python program which allows the user to investigate the Schrodinger's equation under these circumstances. During the development of this program a three point finite difference method is used in building the laplacian differential operator in the Schrodinger's equation. 1D representation of the Schrodinger's equation was used as the base to develop this as it raises the question on how would the Schrodinger's equation behave in the 2D space of these conditions. This program had given the ability to the user to choose the type of potential applied in order to study those effects. [6]

Groeneboom and Henk applied finite difference method with the Lanczos procedure to solve the schrodinger equation. The potentials which was investigated were the 1D Morse Oscillator and 2D Henon-Heles Potentials. A high 10^{th} Order finite difference scheme was used for this particular potential in contrast to low order difference scheme used by Matthew and Shiv. The Lanczos Procedure was used to find the Eigenvalues of the system. The main problem faced in this issue is the computation of eigenvectors. Inorder to overcome this the residue generation method developed by Nauts et al. was applied. The author argued the reason

for using a 10^{th} order finite difference due to been only few additional calculations needed to be done per grid point in this scheme thereby resulting in a higher resolution. It should also be noted the convergence of the scheme was not discussed citing it can be checked easily and with any degree of accuracy the solution of Schrodinger's equation can be obtained. [7]

4 Methods

4.1 1D Schrodinger Equation

The Schrodinger Equation reduced to 1D case is presented below.

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2} = (E - V(x))\psi \tag{10}$$

To derive the solutions for equation 10 Numerical methods such as Finite Difference Method equipped with shooting method , Finite Difference Method with Eigen values and Numerov methods were used.

4.1.1 Shooting Method

Shooting Method is a popular method used in deriving the numerical solution to the 1D case of Schrodinger equation. This method was combinedly used with Finite Difference Method. The Finite Difference Method used in deriving the solution is a Central Difference approximation scheme. Figure(1) presents the schemes available in Finite Difference Approximations.

Using Shooting both Eigen Energy states and the solution to the equation can be found at the same time with the cost of computation time.

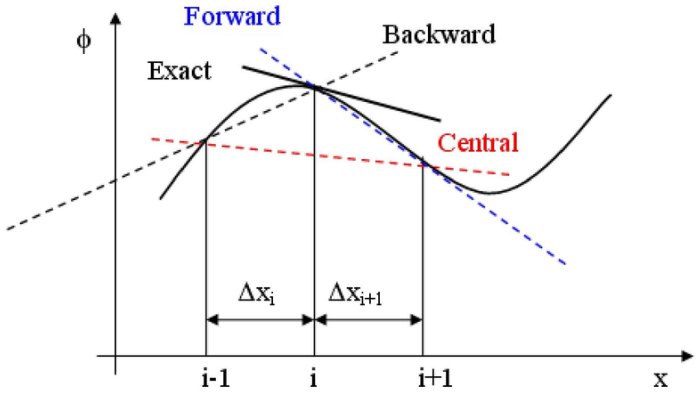

Figure 1: Finite Difference Approximation Schemes

$$\frac{d^2\psi}{dx^2} \approx \frac{\frac{d\psi}{dx}\Big|_{x+\frac{\Delta x}{2}} - \frac{d\psi}{dx}\Big|_{x-\frac{\Delta x}{2}}}{\Delta x} \tag{11}$$

$$\frac{d\psi}{dx}\Big|_{x+\frac{\Delta x}{2}} = \frac{\psi(x+\Delta x) - \psi(x)}{\Delta x} \tag{12}$$

$$\frac{d\psi}{dx}\Big|_{x-\frac{\Delta x}{2}} = \frac{\psi(x) - \psi(x-\Delta x)}{\Delta x} \tag{13}$$

$$\frac{d^2\psi}{dx^2} \approx \frac{\psi(x+\Delta x) - 2\psi(x) + \psi(x-\Delta x)}{\Delta x^2} \tag{14}$$

$$\frac{2m}{\hbar^2}(E - V(x))\psi(x) \approx \frac{\psi(x+\Delta x) - 2\psi(x) + \psi(x-\Delta x)}{\Delta x^2} \tag{15}$$

Rearranging equation(15) will result in the following final equation which will be coded in MATLAB.

$$\psi(x+\Delta x) = 2\psi(x) - \psi(x-\Delta x) - \frac{2m}{\hbar^2}(E - V(x))\psi(x)\Delta x^2 \tag{16}$$

To find the Energy States or the Eigen states for the particle the shooting method is utilised. The type of problem which can be solved through shooting method are only if the boundaries of the particle are known. Therefore one of the disadvantage of this method is the boundary conditions should be known which can be applied only for symmetric potential wells.

Initially the system simulation will begin with a particular Eigen(Energy) State and find if terminals of the well is approximately equals to known boundary value if not increase the Energy by ΔE and find the next wave function.

4.1.2 Eigen Matrix Method

The Eigen Matrix Method is the standard method for solving the Schrodinger's Equation. This method is considered to be quickest and the most efficient method compared to the shooting method. This method also can be used to simulate asymmetrical potential functions where as shooting method is only limited to symmetrical potential functions. Furthermore, the boundary conditions are not needed to know when solving this using Eigen Matrix Method.

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2} + V\psi(x) = E\psi(x) \tag{17}$$

Equation (17) can be expressed in terms of an Eigen Value Problem represented in equation(18).

$$H\psi(x) = E\psi(x) \tag{18}$$

Where Energy states of the system will be the eigen states of this equation and H is known as the Hamilton Matrix. $\frac{d^2\psi}{dx^2}$ is approximated as below using Central difference approximation as used in Finite Difference with shooting method.

$$\frac{d^2\psi}{dx^2} \approx \frac{\psi(x+\Delta x) - 2\psi(x) + \psi(x-\Delta x)}{\Delta x^2} \tag{19}$$

Rearranging after Applying above Approximation results in the final equation as follows.

$$-\frac{\hbar^2}{2m\Delta x^2}\psi(x+\Delta x) + (V(x) + 2\frac{\hbar^2}{2m\Delta x^2})\psi(x) - \frac{\hbar^2}{2m\Delta x^2}\psi(x-\Delta x) = E\psi(x) \tag{20}$$

This can be represented as the following system of Matrices.

$$\underbrace{\begin{bmatrix} V(x) + 2\frac{\hbar^2}{2m\Delta x^2} & -\frac{\hbar^2}{2m\Delta x^2} & \dots & 0 \\ -\frac{\hbar^2}{2m\Delta x^2} & V(x) + 2\frac{\hbar^2}{2m\Delta x^2} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & V(x) + 2\frac{\hbar^2}{2m\Delta x^2} \end{bmatrix}}_{\mathbf{Hamilton-Matrix-H}} \underbrace{\begin{bmatrix} \psi(1) \\ \psi(2) \\ \vdots \\ \psi(n) \end{bmatrix}}_{\psi} \approx E \underbrace{\begin{bmatrix} \psi(1) \\ \psi(2) \\ \vdots \\ \psi(n) \end{bmatrix}}_{\psi}$$

Solving the above matrix will result in the Eigen States Energy and the resulting Eigen Vectors which correspond to the wave function for the given potential.

The solutions for eigen problems are such that for a particular eigen(energy state) will have a corresponding eigen vector which will satisfy the equation.

4.1.3 Validation and Verification

The Eigen Method with an Infinite Potential Well was used to verify the results obtained analytically. The shooting method was impractical to use for infinite potential well due to its long computation time taken due to discretising the Energy states and Particles of the wave function.

The Energy states for a particle in an infinite well can be found using following equation.

$$E_n = \frac{\hbar^2 \pi^2}{2mL^2} n^2 \tag{21}$$

The table(1) represents the values computed for first 3 states using both methods.

n	Numerical	Analytical	Error
1	0.584	0.5875	0.5%
2	2.338	2.3498	0.5%
3	5.260	5.2871	0.5%

Table 1: Numerical vs Analytical Eigen

Figures(2-21) shows the numerically calculated wave functions for the Infinite Potential Well using eigen matrix method.

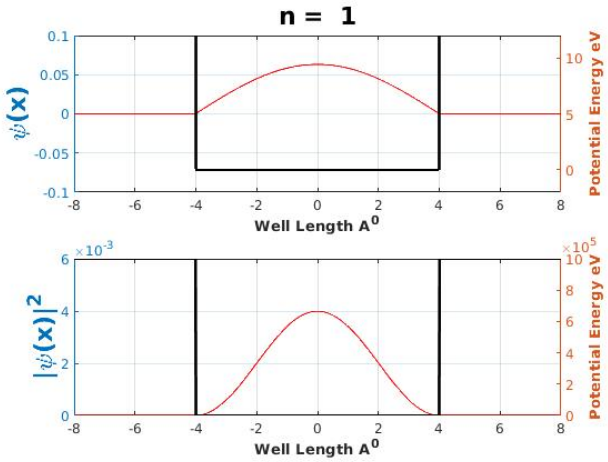

Figure 2: n = 1 Infinite Well Numerical Solution

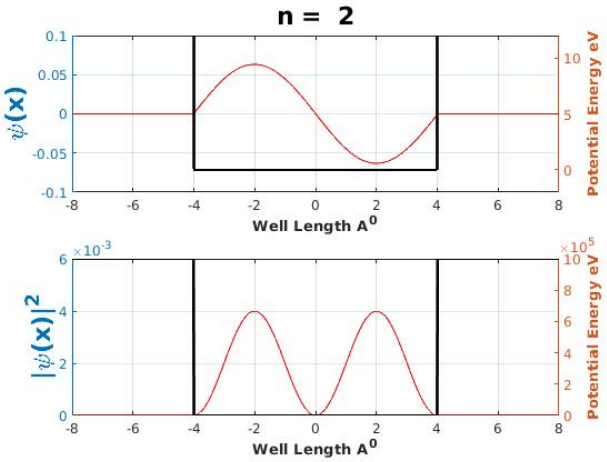

Figure 3: n = 2 Infinite well Numerical Solution

It can be shown that analytically when deriving the solution at either side of the well the wave function is deduced to be zero where as in this method numerically it is shown and proven that argument is valid. The boundary conditions at both terminals of the well is also found to be zero using numerically presenting that this method can be used to generate for other potential wells in order to study the behaviour of the particle under different potential wells. Thereby this shows this method is valid and can be used to find the solution for the Schrodinger's equation.

5 MATLAB Implementation

This section presents the procedure in implementing the shooting method and eigen method in MATLAB.

5.1 Shooting Method

The procedure in implementing the shooting method in matlab is presented below.

- Initially the variables required for simulation were assigned

```
m_e = 9.11e-31;          %mass of electron
e = 1.602e-19;           %charge of electron to convert for eV
h_bar = 1.055e-34;       %planks constant divided by 2
A_0 = 10^10;     % convert angstrom to nm ease of plotting the xaxis

%% Init variables for potential wells

V_upper = 10;
V_lower = 0 ;
Lw = 8e-10; %in nm
Wb = Lw/2 ; %well boundary
N = 1000;
x = linspace(-Lw,Lw,N);    %in nm
dx = x(2)-x(1);
epsilon = 1e-5;   %tolerance level for shooting method
```

- Generate Potential Function (the following is a function to generate the linear potential function)

```
for i = 1:length(x)
        if -Wb<=x(i)
            V(i) = f_linear(-x(i));
        else
              V(i) = f_linear(x(i));
        end
end
    for i = 1:length(x)
        if x(i)>=-Wb && x(i)<=Wb
            V(i) = V_lower;
        end
    end
    potentialFunct = V;
end
```

- Generation of Energy Array to simulate and find the Eigen Energy for each eigen state and wave function

```
E1 = 0;
E2 = 10;
nE = 10000001;
E = linspace(E1,E2,nE);      %build the energy vector to loop through for
    shooting method
```

- Generate the discretized wave function array for a particular energy

```
function psi = wave_function(E,V,dx,N)
    global m_e h_bar e ;
    C = (2*m_e*e/h_bar^2);
    psi = zeros(1,N);
    psi(2) = 1;
```

```
    for n = 2:N-1
        psi(n+1) = (2 - C*(E-V(n))*dx^2)*psi(n) - psi(n-1);
    end
    Area = trapz(psi.*psi);
    psi = psi /sqrt(Area);

end
```

- Simulate for each Energy value and Check if the boundary conditions are fulfilled

```
function EnergyEigen = EigenSim(E,V,dx,N)
global epsilon;
i = 1;
    for e = 1:length(E)
        psi = wave_function(E(e),V,dx,N);
        %find to see if terminal reaches boundary conditions
        if(abs(psi(end))<epsilon)
            EnergyEigen(i) = E(e);
            i = i+1;
        end
    end
%filter out unique energy wells
E_n_psi = unique(EnergyEigen);
for i = 2:length(E_n_psi)
    n_E(1) = E_n_psi(1);
    if E_n_psi(i)-E_n_psi(i-1)>0.1
        n_E(i) = E_n_psi(i);
    end
end
%the specific energy values
EnergyEigen = unique(n_E);
end
```

5.2 Eigen Method

The steps followed in implementing in MATLAB is presented below.

- Initially the variables required for simulation were assigned
- Generate the Potential Function (following is generating for square potential)

```
for i = 1:length(x)
        if -Wb<=x(i) && x(i)<=Wb
            V(i) = V_lower;
        else
            V(i) = V_upper;
        end
end
```

- Develop the Hamilton matrix combining Kinetic energy and the potential function matrix

```
T_hat = zeros(length(x),length(x));
for r = 1:length(x)
    T_hat(r,r) = -2;
    T_hat(r,r+1) = 1;
    T_hat(r+1,r) = 1;
end

```

```
T_hat(length(x)+1,:) = [];
T_hat(:,length(x)+1) = [];

Cdx = C/dx^2;  %constants

% Hamilton Matrix Join Kinetic Energy Matrix With Potential Energy Matrix
H_hat = Cdx*T_hat + V_hat;
```

- Solve the matrix using *eig* function in matlab by providing the hamilton matrix as argument. This will provide the wave function ψ and eigen states *Energy* of the particular in that potential function as outputs

```
%solve using eig
[psi_e,Energy] = eig(H_hat);
```

6 1D Simulation

The solutions obtained from solving both numerical methods are presented in this section. The well length was taken to be $8nm$ and the maximum potential applied is $10eV$. Simulations were carried out for these assumed parameters.

6.1 Shooting Method

This presents the simulation of 1D Schrodinger's equation using shooting method combined with a central difference approximation scheme.

6.1.1 Square Potential Well

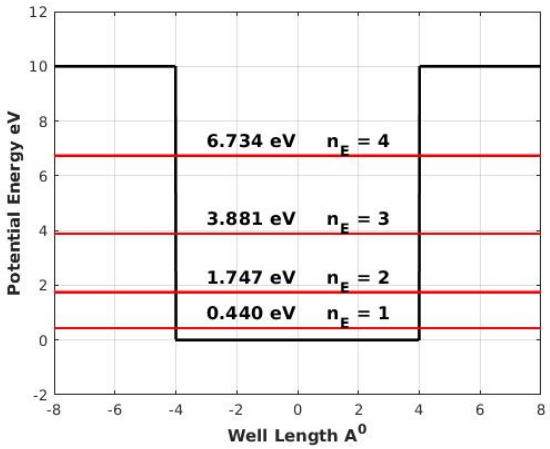

Figure 4: Allowed Energy States Square Well Shooting

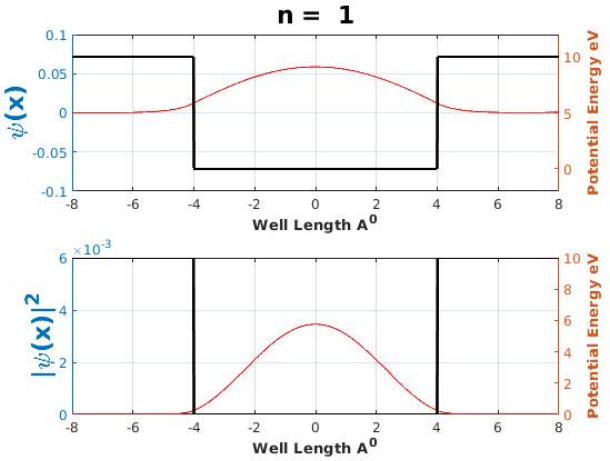

Figure 5: n = 1 Square well Shooting

The Square Potential Well was modelled in MATLAB as following function Where W_b is well boundary.

$$V(x) = \begin{cases} V_{max}, & \text{if } |x| \geq W_b \\ 0, & \text{otherwise} \end{cases}$$

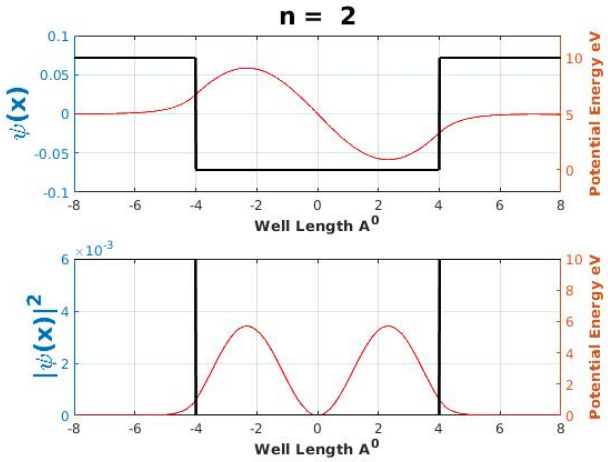

Figure 6: n = 3 Square well Shooting

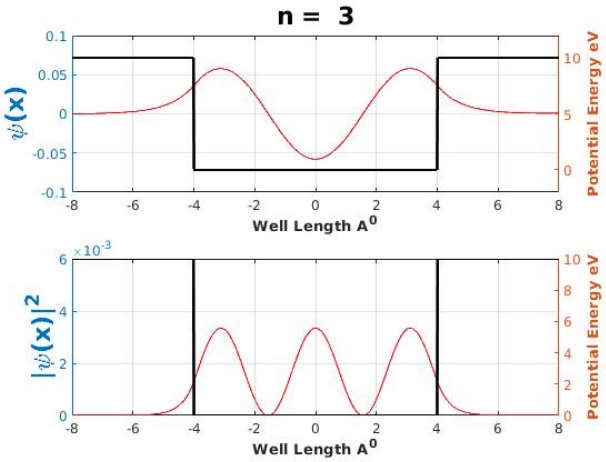

Figure 7: n = 3 Square well Shooting

6.1.2 Linear Potential Well

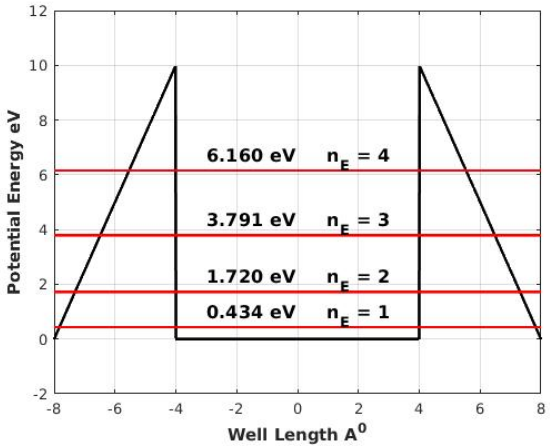

Figure 8: Allowed Energy States Linear Well Shooting

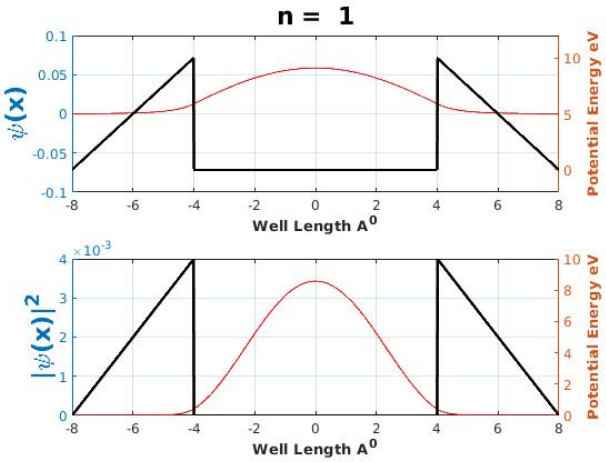

Figure 9: n = 1 Linear well Shooting

The Linear Potential Well was modelled in MATLAB as following function

$$V(x) = \begin{cases} \dfrac{2V_{max}}{L_w} + 2V_{max}, & \text{if } |x| \geq W_b \\ 0, & \text{otherwise} \end{cases}$$

Where W_b is well boundary and L_w is the well length.

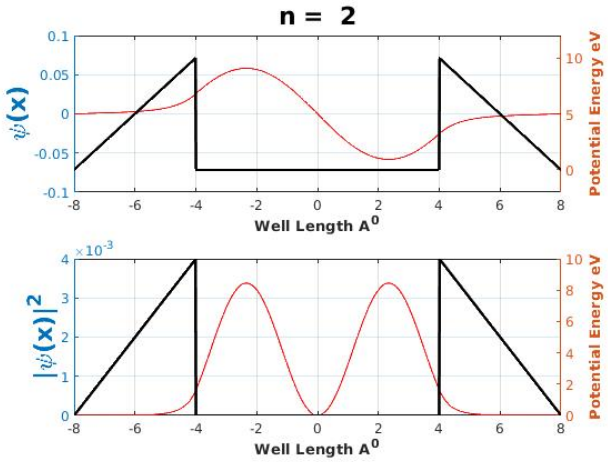

Figure 10: n = 2 Linear well Shooting

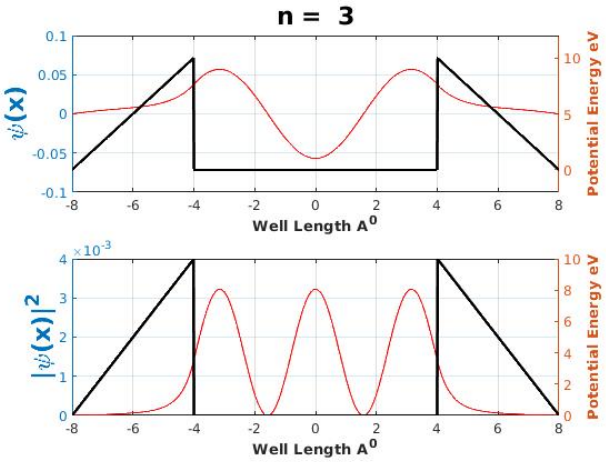

Figure 11: n = 3 Linear well Shooting

6.1.3 Quadratic Potential Well

Figures(13-15) represent the wave function in a Quadratic Potential Well.

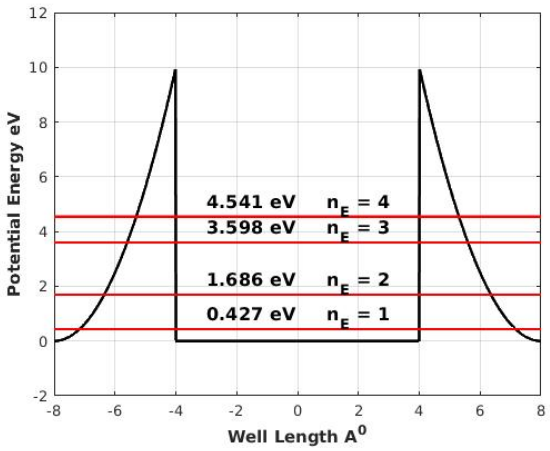

Figure 12: Energy States Quadratic Well Shooting

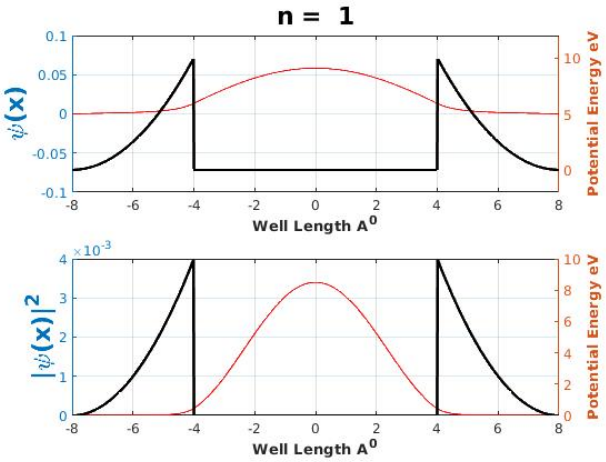

Figure 13: n = 1 Quadratic well Shooting

The Quadratic Potential Well was modelled in MATLAB as following function

$$V(x) = \begin{cases} \dfrac{2V_{max}(x+L_w)^2}{(Wb-Lw)^2}, & \text{if } |x| \geq W_b \\ 0, & \text{otherwise} \end{cases}$$

Where W_b is well boundary.

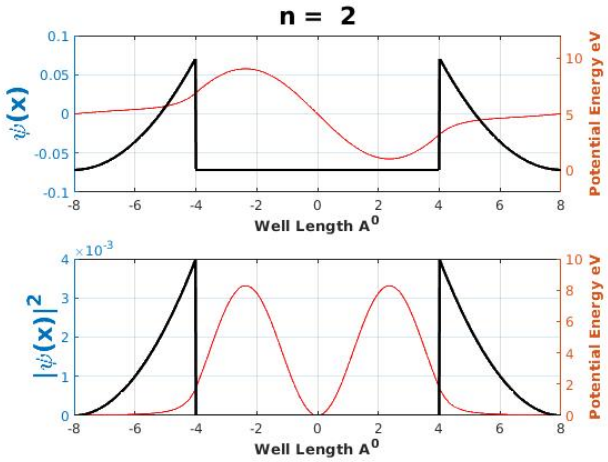

Figure 14: n = 2 Quadratic well Shooting

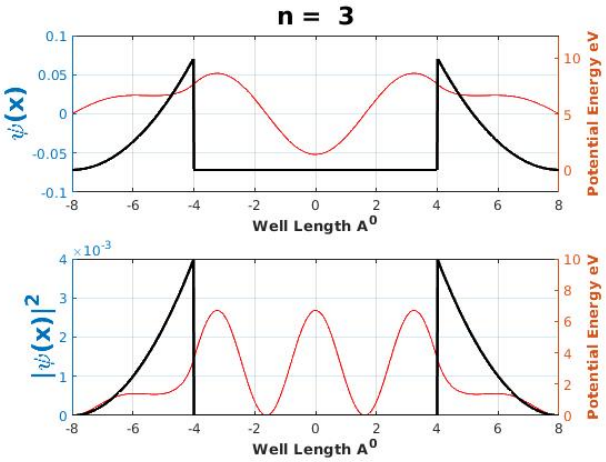

Figure 15: n = 3 Quadratic well Shooting

6.1.4 Harmonic Oscillator

Figures(17-19) represent the wave function in a Harmonic Oscillator Potential Well.

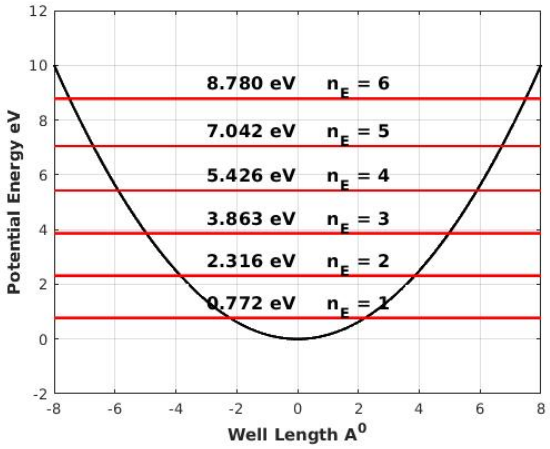

Figure 16: Energy States Harmonic Osci. Shooting

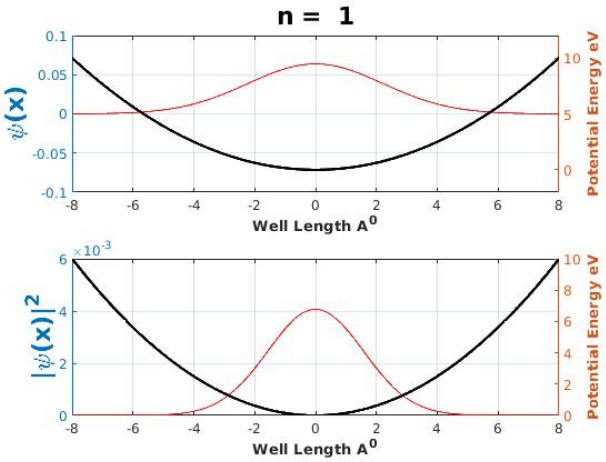

Figure 17: n = 1 Harmonic Oscillator Shooting

The Harmonic Oscillator Well was modelled in MATLAB as following function

$$V(x) = \begin{cases} \frac{V_{max}x^2}{L_w^2} \end{cases}$$

6.2 Eigen Matrix Method

This presents the simulation of 1D Schrodinger's equation using standard eigen matrix method solved using the *eig* function available in MATLAB.

6.2.1 Square Potential Well

Comparing figures(4-23) it can be seen that both solutions present the same identical results but the computation time taken by numerical methods greater than the time taken by Eigen Method when applied in

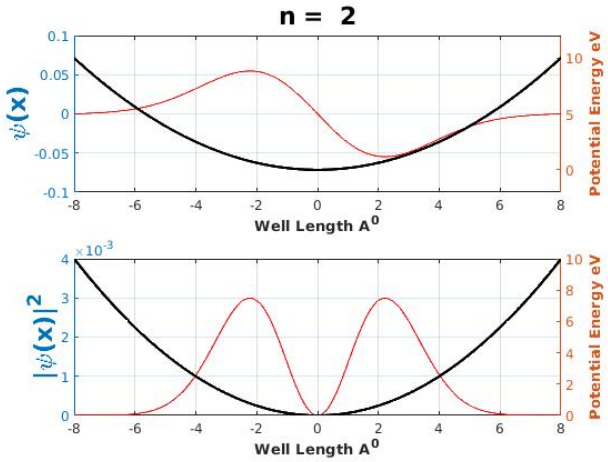

Figure 18: n = 2 Harmonic Oscillator Shooting

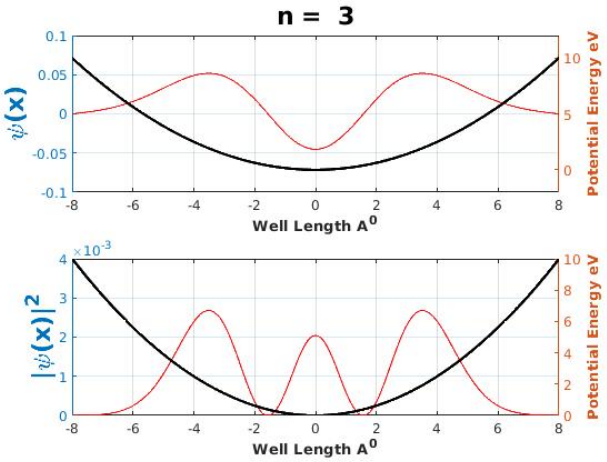

Figure 19: n = 3 Harmonic Oscillator Shooting

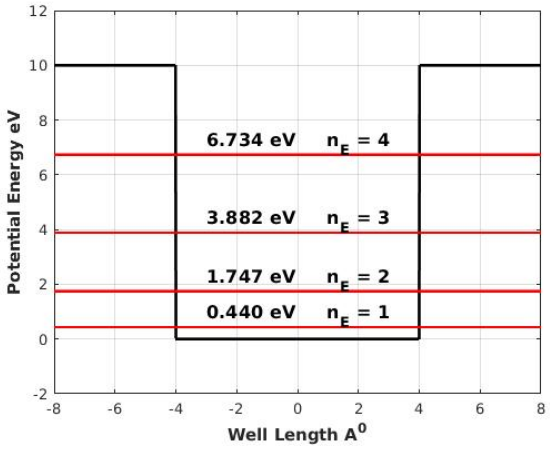

Figure 20: Eigen Energy States Square Well

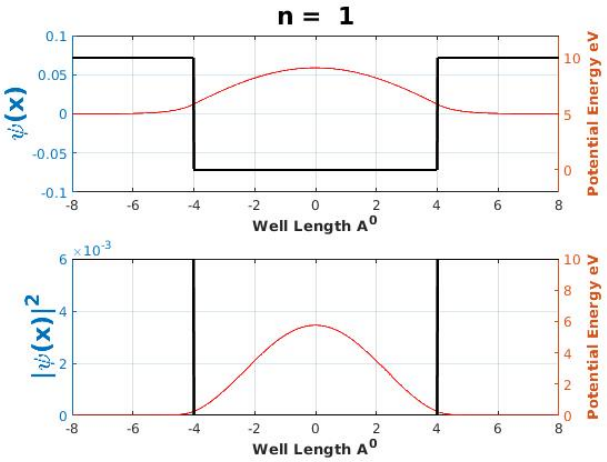

Figure 21: n = 1 Square well Eigen

MATLAB. The following summarizes the time taken by both methods to solve the square potential well in matlab.

Method	**Time Taken**
Eigen	0.64188 s
Shooting	228.788 s

It can be seen that the amount of time take in Shooting method is far greater than the time taken in Eigen Method.

For the symmetrical potentials simulated using Shooting Method is also solved using the Eigen Method is presented in the appendix for validation purposes. Eigen Method was mainly used to solve asymmetrical potential wells where the boundary conditions are not known. This is because solving the system of equations with out knowing the boundaries is the main speciality of this method.

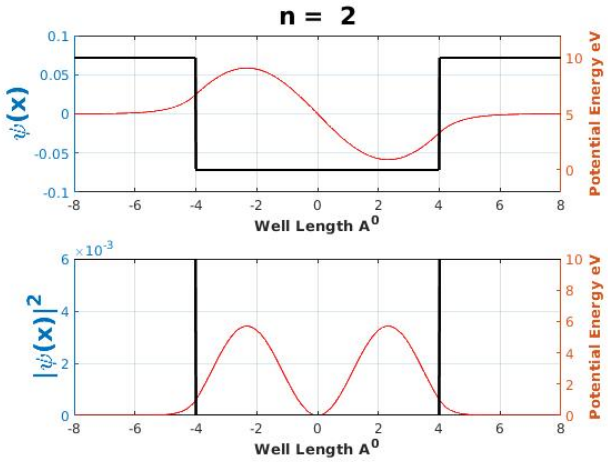

Figure 22: n = 2 Square well Eigen

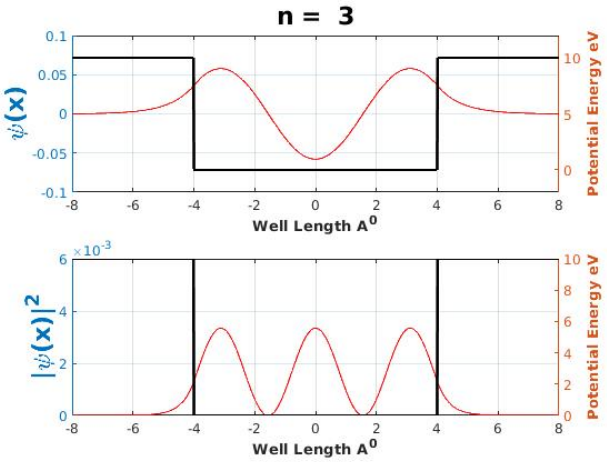

Figure 23: n = 3 Square well Eigen

6.2.2 Free Particle No Potential Field

Figures(24-35) represents the Eigen Simulation for a free Particle.

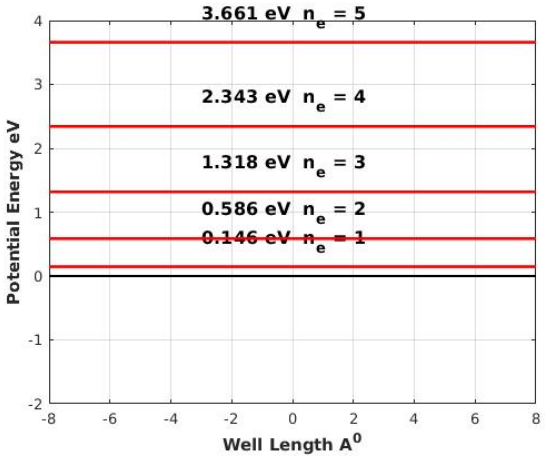

Figure 24: Energy States Free Eigen

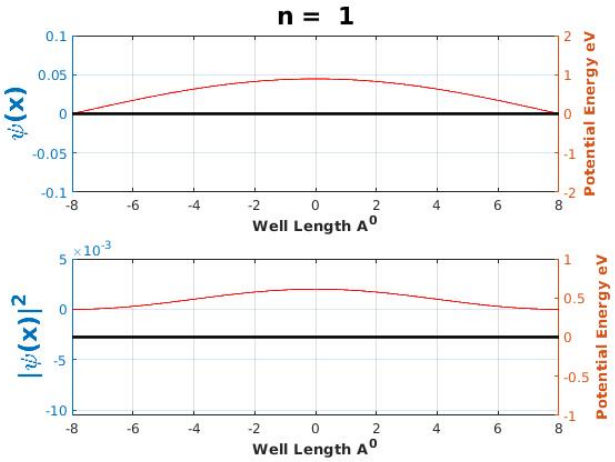

Figure 25: n = 1 Free Particle Eigen

The potential for the free particle is modelled as no potential field applied . Therefore the total energy of the particle will be equivalent to the kinetic energy.

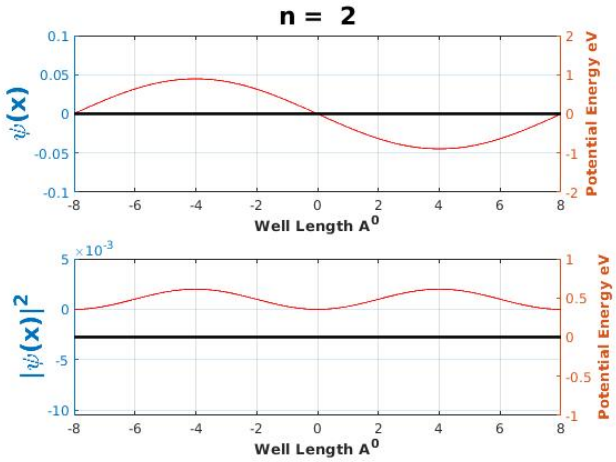

Figure 26: n = 2 Free Particle Eigen

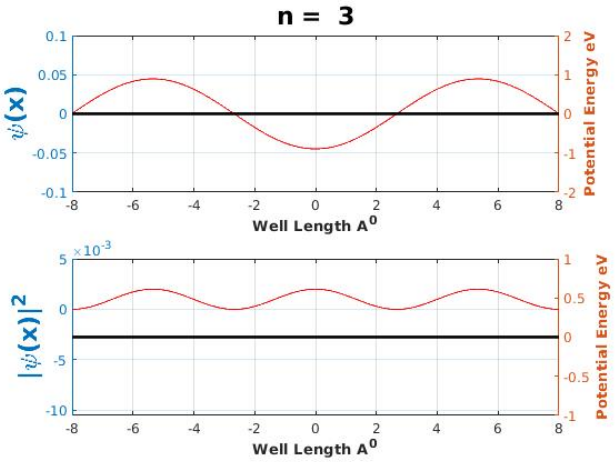

Figure 27: n = 3 Free Particle Eigen

6.2.3 Stepped Potential

Figures(28-31) represents the Eigen Simulation for Stepped Potential.

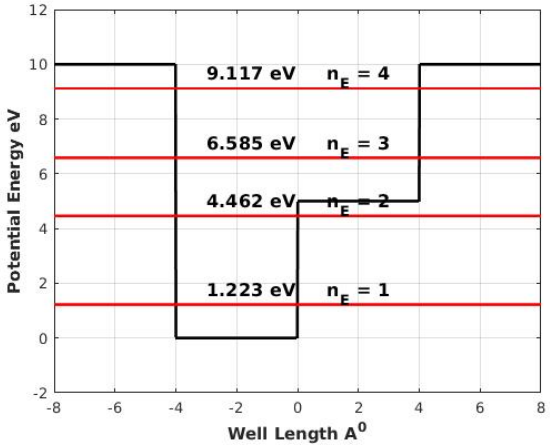

Figure 28: Energy States Stepped Potential Eigen

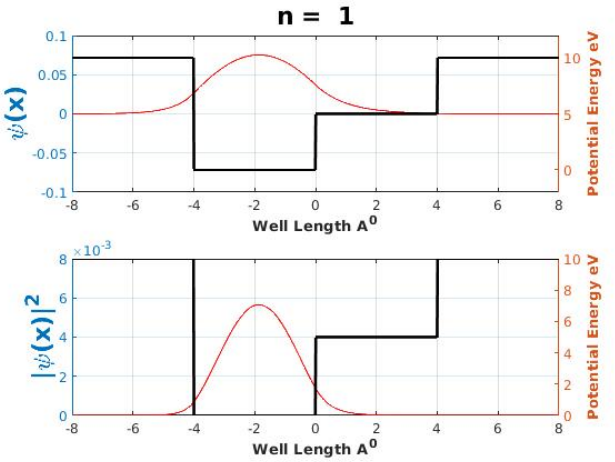

Figure 29: n = 1 Stepped Potential Eigen

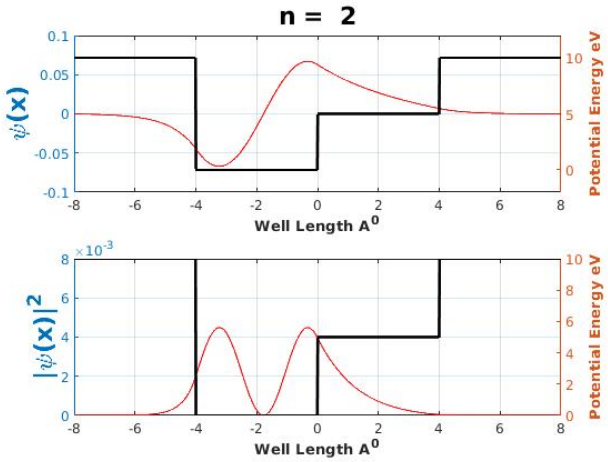

Figure 30: n = 2 Stepped Potential Eigen

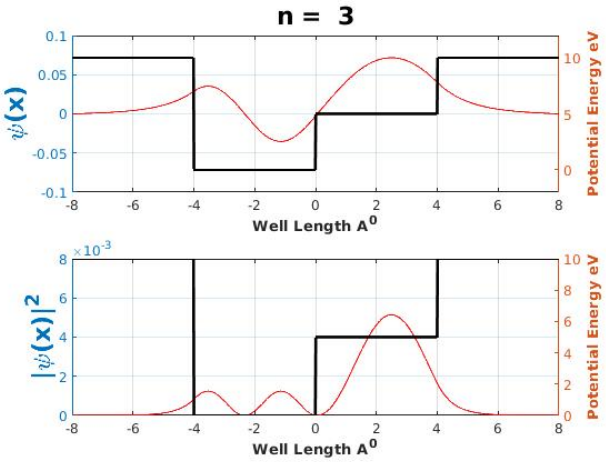

Figure 31: n = 3 Stepped Potential Eigen

6.2.4 Symmetrical Double Potential Well

Figures(32-35) represents the Eigen Simulation for Stepped Potential.

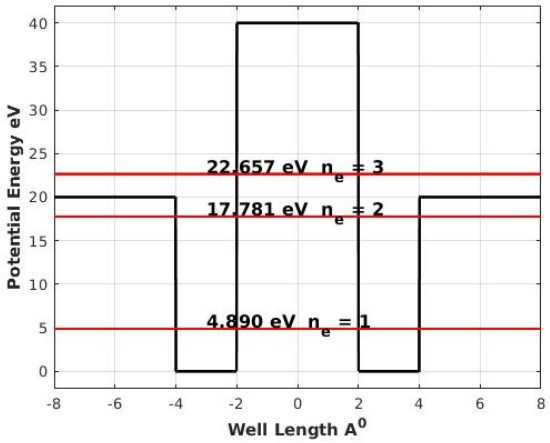

Figure 32: Energy States Double Well Eigen

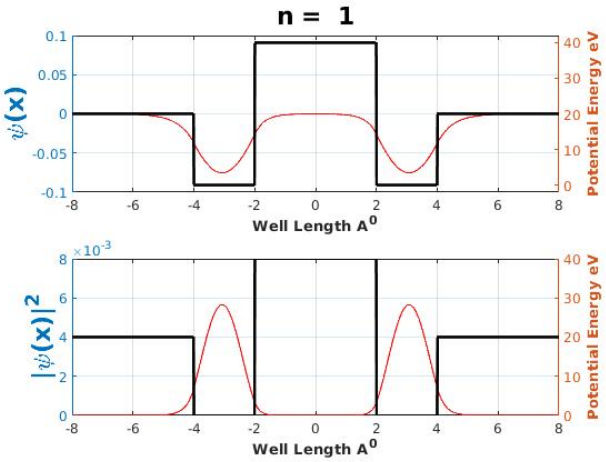

Figure 33: n = 1 Double well Eigen

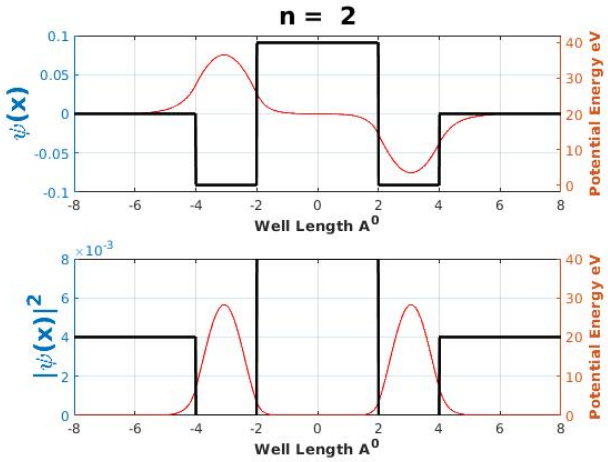

Figure 34: n = 2 Double well Eigen

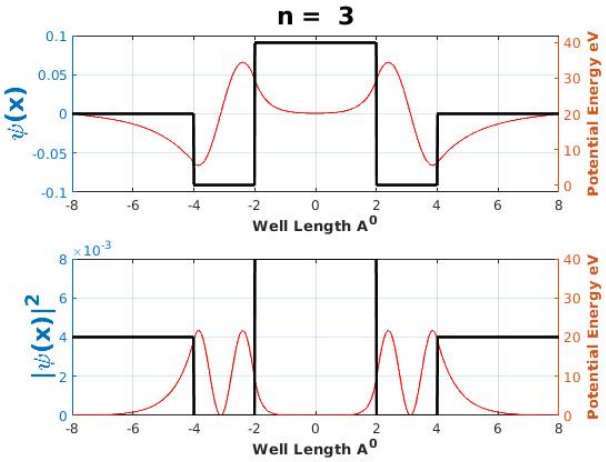

Figure 35: n = 3 Double well Eigen

6.2.5 Triangle Potential

Figures(36-39) represents the Eigen Simulation for Triangle Potential.

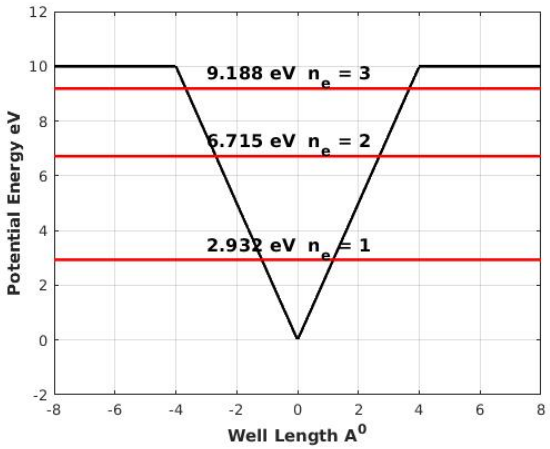

Figure 36: Energy States Triangle Potential Eigen

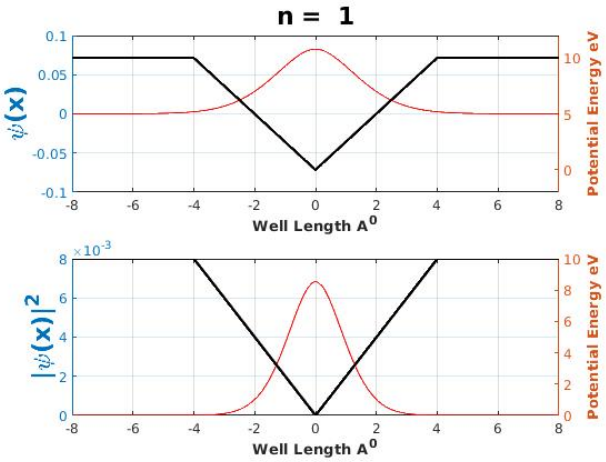

Figure 37: n = 1 Triangle Potential Eigen

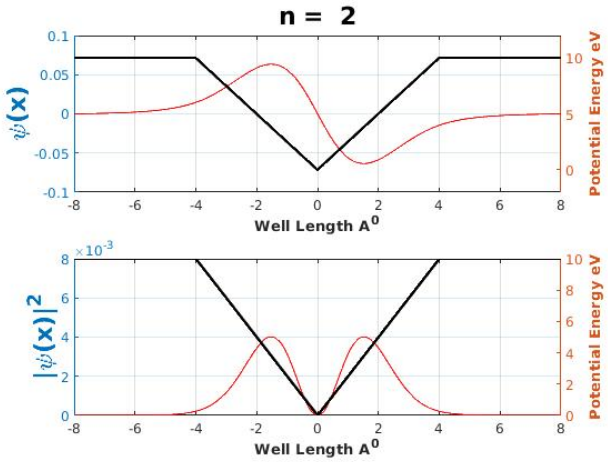

Figure 38: n = 2 Triangle Potential Eigen

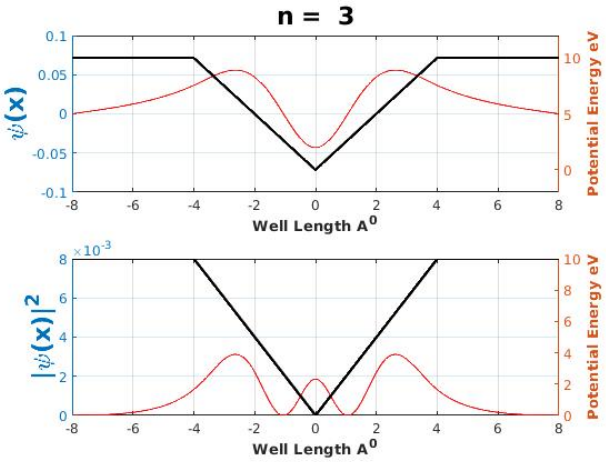

Figure 39: n = 3 Triangle Potential Eigen

6.2.6 Asymmetrical Double Potential Well

Figures(32-35) represents the Eigen Simulation for Assymetrical Double Well Potential.

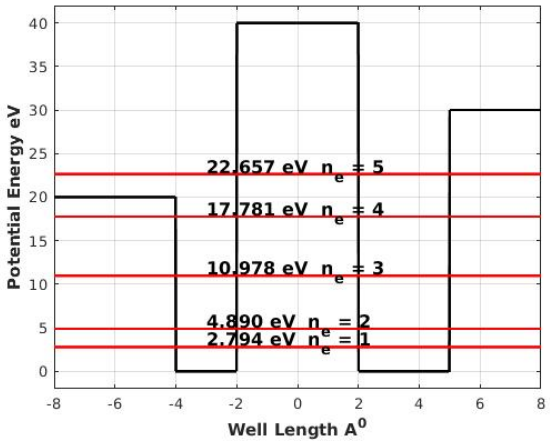

Figure 40: Energy States Asymmetric Double Well Eigen

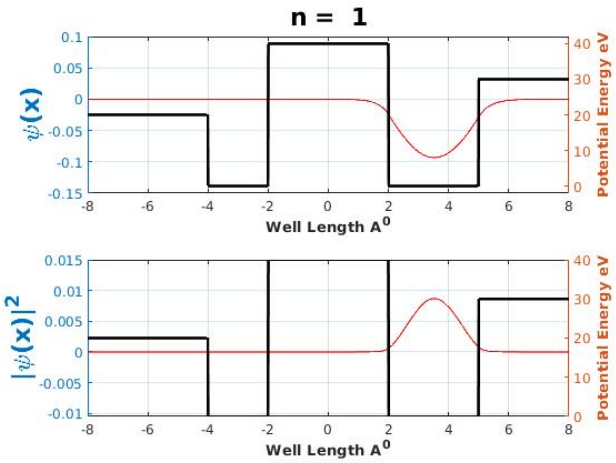

Figure 41: n = 1 Asymmetric Double Well Eigen

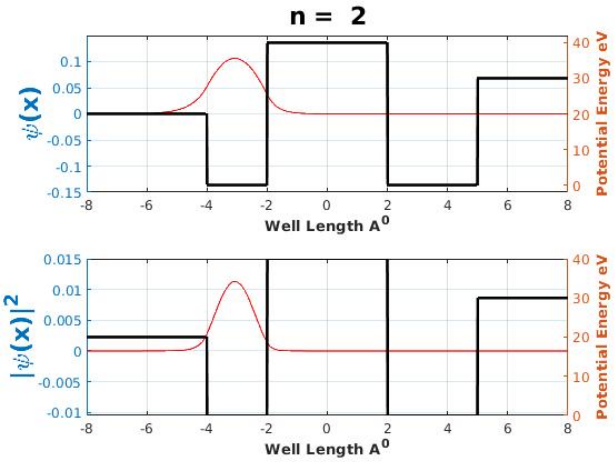

Figure 42: n = 2 Asymmetric Double Well Eigen

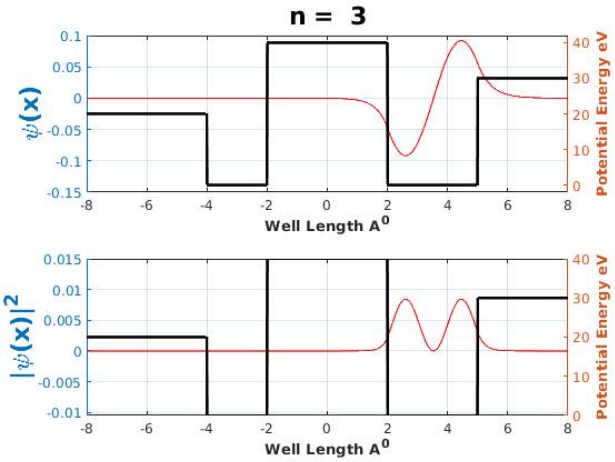

Figure 43: n = 3 Asymmetric Double Well Eigen

6.2.7 Series Potential Well

Figures(44-47) represents the Eigen Simulation for a Series of Square Potential Wells. This is also known as Kronig Penny Model.

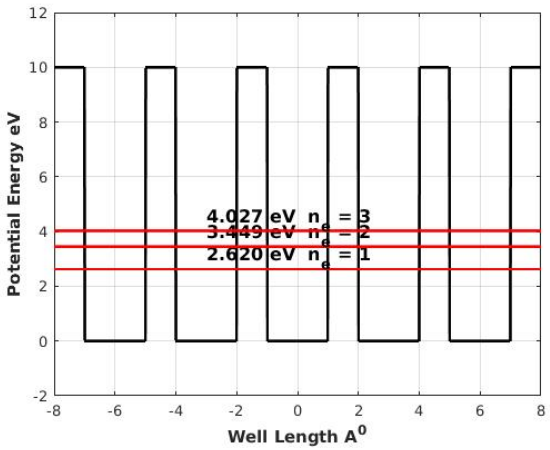

Figure 44: Energy States Kronig Eigen

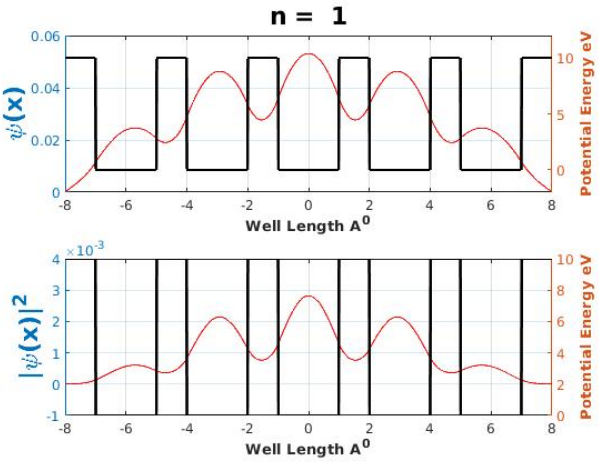

Figure 45: n = 1 Kronig Eigen

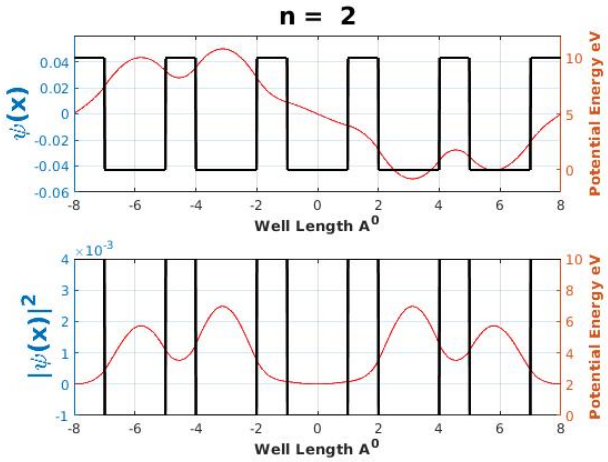

Figure 46: n = 2 Kronig Eigen

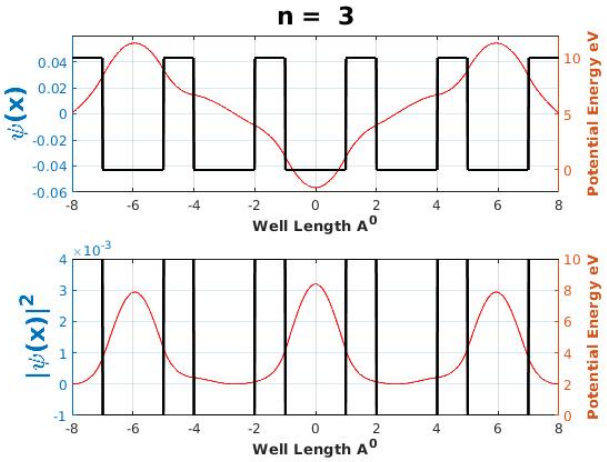

Figure 47: n = 3 Kronig Eigen

7 Discussion

This report simulated and presented the solution for Schrodinger's Equation under several different potential well functions where analytical solutions are hard to obtain. The methods used in simulating the wave function are shooting method and eigen matrix method.

7.1 Comparison of Result and Validation

When observing the probability of finding the wave function it can visually understood that at the edges of the well the finding the particle is close to zero and the highest probability of finding the particle is highest at the middle of the well. When this is compared with the theory of Schrodinger's equation matches approximately with the results obtained from numerical simulations.

Both shooting method and eigen method provided the same simulation results for the same system when symmetrical potential fields were applied. Shooting method was not used to plot the asymmetrical potential fields due to its complexity in computational time.

7.2 Errors and Improvements

Errors and problems were observed when the equation and numerical methods were coded in to MATLAB. To overcome these errors and produce approximately accurate and precise results certain simplification and programming techniques were used. The Energy array constructed and the given potential functions are in eV where as the equation required in SI units. Thus extra precautions were taken as the values used should be in standard SI units in order to obtain appropriate solutions.

Understanding the amount of time taken to obtain the solution is necessary as it becomes vital in certain situations. The time taken for simulating shooting method is higher than the time required for calculation in Eigen method. The shooting method simultaneously found both wave function and the eigen energy states for Schrodinger's equation. This was both costly in terms of computation and time. This was because the energy vector was constructed highly discretizing the Energy values between given two extremes of Energy States. The smaller the difference between each energy state increased the computation time to find the eigen states for a given potential field. If a method can prescribed in order to obtain efficient and appropriate Energy values thereby the time complexity of this method can be reduced thereby increasing its efficiency. It should be noted when simulating for each energy value a corresponding new function should be generated discrediting the previous one. This also consumes time which affects the effeciency of the code directly. This approach is not considered to be suitable for problems where the boundaries or the initial values are

unknown. Furthermore the MATLAB code had to separated into segments in order to reduce the number of lines of the code if not the length of the code will increase. Global variables were used in order to reduce the amount of arguments passed in to the functions.An efficient optimized algorithm is needed verify the terminal boundary validation in shooting would be beneficial in reducing the computation time in shooting method.

The complexity of Eigen method is less when it is compared with the shooting method. This method doesn't need to construct the energy array. Thus simplifying the amount of code lines required to perform the calculations. Schrodinger's equation is basically an Eigen Problem. This method is the most standard and effective method used in solving the equation. The boundaries or the initial states are not required in solving this method where this was a necessary condition in shooting method approach.

The mathematical Schrodinger's equation doesn't limit the energy of the particle it can attain. This is the same issue in finding the energy states of the particle from Shooting method and also eigen method as its impossible to detect the maximum energy state of the particle. In shooting method a particular maximum energy value was used to simulate and a level was placed when plotting the results.

8 Conclusion

This report presented the simulation of Schrodinger's Equation in 1D using shooting and eigen method. Several potential wells were modelled and implemented in MATLAB and plotted to visualise the behaviour of the particle in a potential field. The probability of finding the particle is also plotted in order to understand the physical importance of the wave function. Both symmetrical and asymmetrical potential functions were plotted only using Eigen method and shooting method was used for symmetrical functions only. It was found that shooting method computationally expensive. Therefore the most convenient method was considered to be the Eigen Method where a number of energy states can be computed with required less time.

References

[1] D. M. Gunawardena, "Electronic devices lecture notes," 2021.

[2] D. Neamen, *Semiconductor Physics and Devices: Basic Principles*. McGraw-Hill international edition, McGraw-Hill, 2012.

[3] G. S. Beddard, "Solution of the schrödinger equation for one-dimensional anharmonic potentials: An undergraduate computational experiment," *Journal of Chemical Education*, vol. 88, no. 7, pp. 929–931, 2011.

[4] J. Tellinghuisen, "Accurate numerical solutions of the one-dimensional schrödinger equation," *Journal of Chemical Education*, vol. 66, no. 1, p. 51, 1989.

[5] L. Taylor, "A comparison between the shooting and finite-difference method in solving a nonlinear boundary value problem found in the context of light propagation," 2017.

[6] M. N. Srnec, S. Upadhyay, and J. D. Madura, "A python program for solving schrödinger's equation in undergraduate physical chemistry," *Journal of Chemical Education*, vol. 94, no. 6, pp. 813–815, 2017.

[7] G. C. Groenenboom and H. M. Buck, "Solving the discretized time-independent schrödinger equation with the lanczos procedure," *The Journal of Chemical Physics*, vol. 92, no. 7, pp. 4374–4379, 1990.

A Eigen Method

A.1 Linear Potential Well

Figures(48-51) represents the Eigen Simulation for Linear Potential Well.

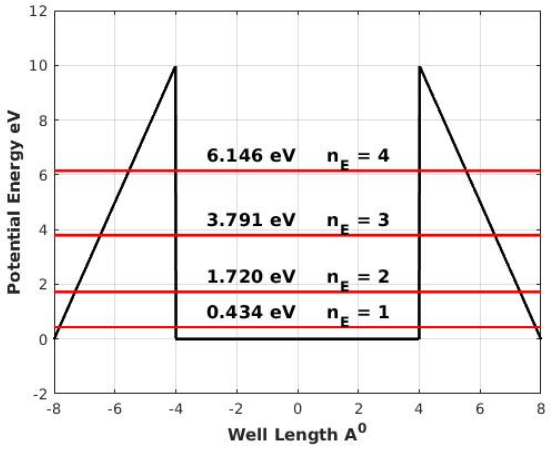

Figure 48: Allowed Energy States Linear Well Eigen

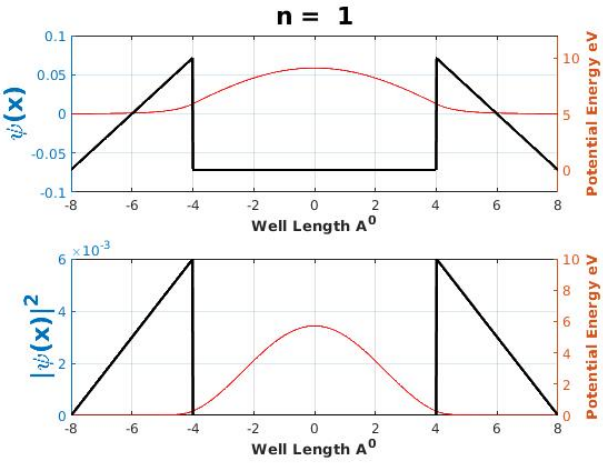

Figure 49: n = 1 Linear Well Eigen

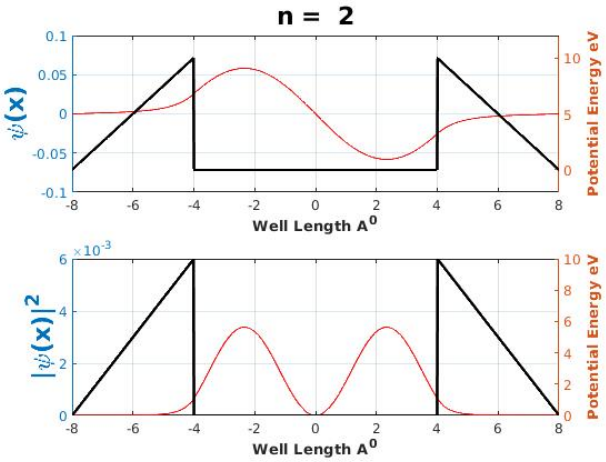

Figure 50: n = 2 Linear Well Eigen

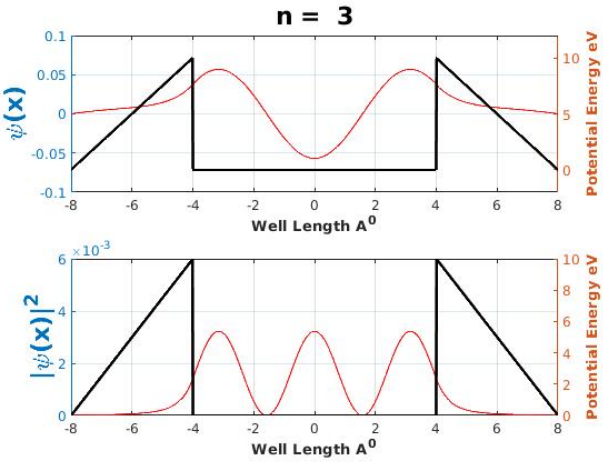

Figure 51: n = 3 Linear Well Eigen

A.2 Quadratic Potential Well

Figures(52-55) represents the Eigen Simulation for Quadratic Potential Well.

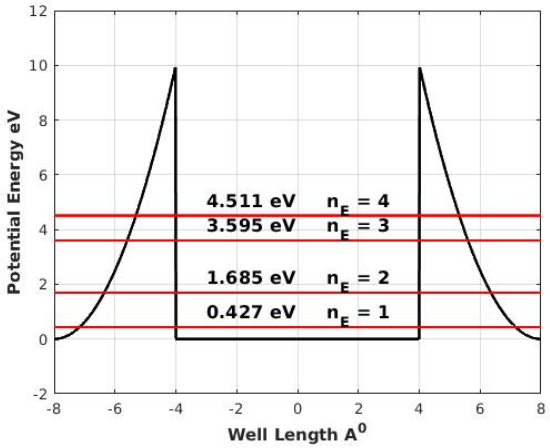

Figure 52: Energy States Quadratic Well Eigen

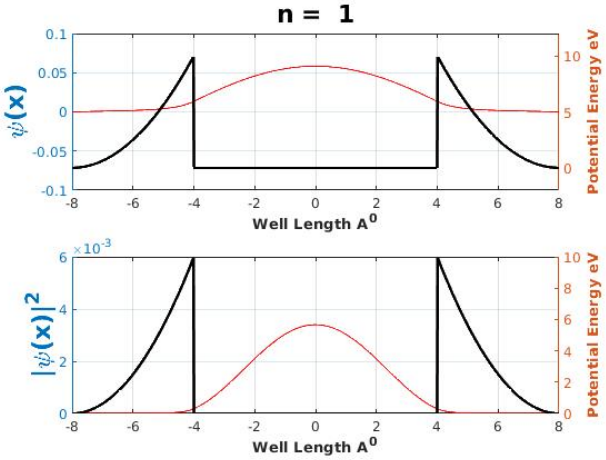

Figure 53: n = 1 Quadratic Well Eigen

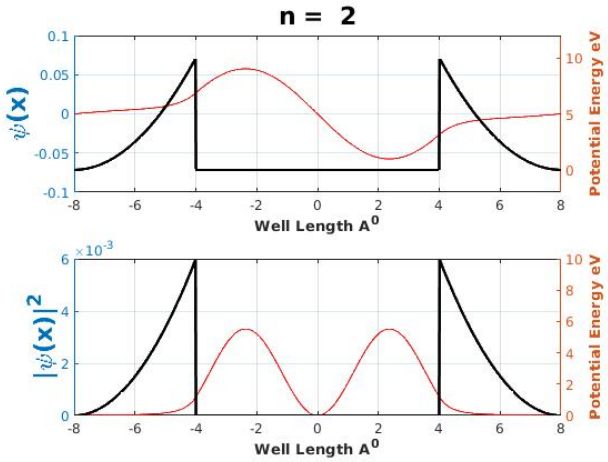

Figure 54: n = 2 Quadratic Well Eigen

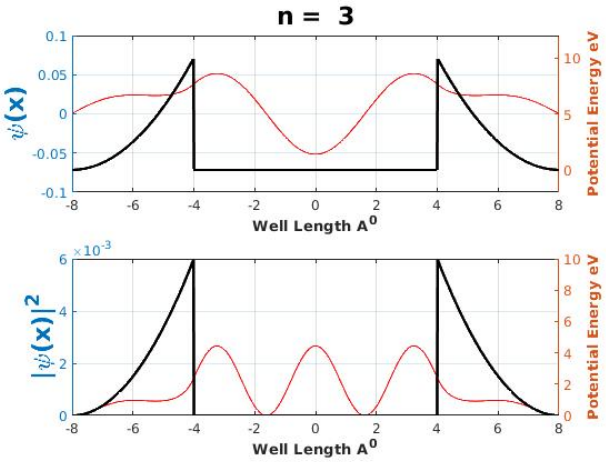

Figure 55: n = 3 Quadratic Well Eigen

A.3 Harmonic Oscillator Potential Well

Figures(56-59) represents the Eigen Simulation for Harmonic Oscillator Potential Well.

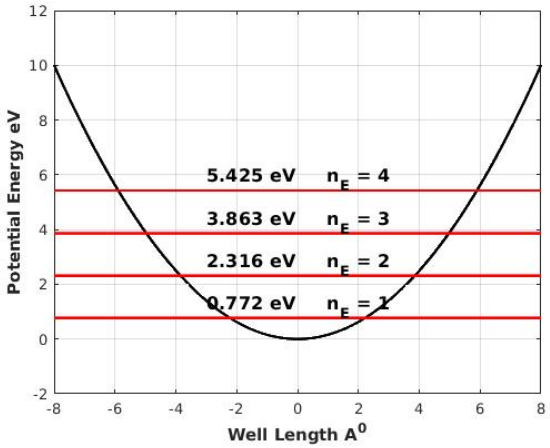

Figure 56: Energy States Harmonic Oscillator Eigen

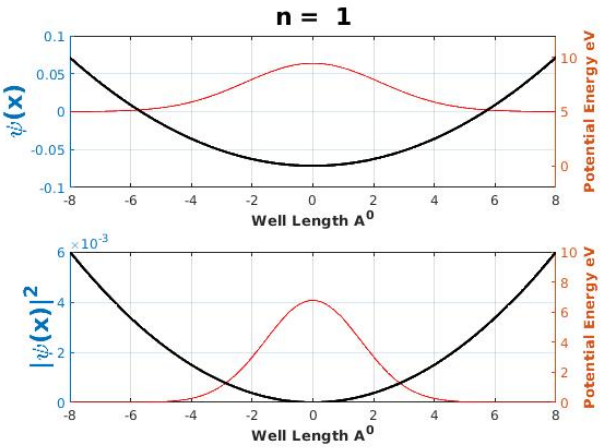

Figure 57: n = 1 Harmonic Oscillator Well Eigen

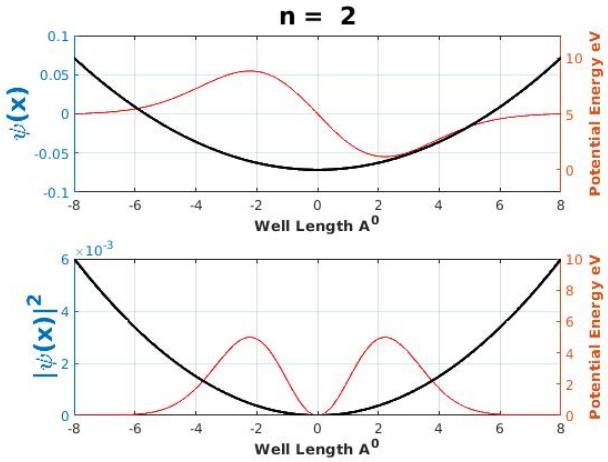

Figure 58: n = 2 Harmonic Oscillator Well Eigen

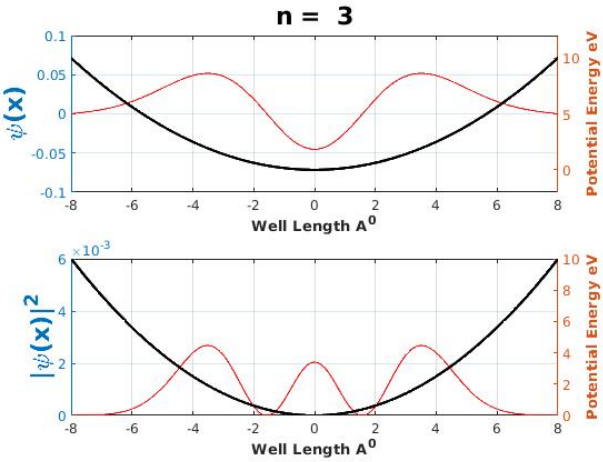

Figure 59: n = 3 Harmonic Oscillator Well Eigen

B MATLAB Codes

B.1 Eigen Code

The main script file.

```
%%% Author - M Shihar
%%% ID     - 208628A
%%% Electronics Devices
%%% Eigen Method



clc;
close all;
clear ;
tic;

disp('Running Eigen Method Simulation of Schrodinger 1D Equation');
disp('Initialising Variables');

%use global variables so amount arguments passed to function simplifies
global m_e h_bar e V_upper V_lower Wb Lw A_0;
m_e = 9.11e-31;
e = 1.602e-19;
h_bar = 1.055e-34;
A_0 = 10^10;

C =-h_bar^2/2/m_e;

%% Init variables for potential wells
V_upper = 10;
V_lower = 0 ;

Lw = 8e-10; %in nm
Wb = Lw/2 ; %well boundary
N = 1000;
x = linspace(-Lw,Lw,N);    %in nm
dx = x(2)-x(1); %difference between two discretized points

%% Building Potential Energy Matrix
disp('Generating Potential Function');

%uncomment to run a particular function
sel = 1;      %Linear
%sel = 2;     %Quadratic
%sel = 3;     %Harmonic
%sel = 4;     %Square
%sel = 5;     %stepped Well
%sel = 6;     %Double Well
%sel = 7;     %triangle well
%sel = 8;     %series potential well
%sel = 9;     %Assymetrical Double Well
%sel = 10;    %free particle

% debug = true;   %to view potential function
```

```
51 debug = false ;
52
53 [V_hat,V] =potential_Generator(sel,x,debug);
54
55
56 %% Building Hamilton Matrix
57
58 disp('Developing Hamilton Matrix');
59
60 %Build Kinetic Energy Matrix
61 T_hat = zeros(length(x),length(x));
62 for r = 1:length(x)
63     T_hat(r,r) = -2;
64     T_hat(r,r+1) = 1;
65     T_hat(r+1,r) = 1;
66 end
67
68 T_hat(length(x)+1,:) = [];
69 T_hat(:,length(x)+1) = [];
70
71 Cdx = C/dx^2;
72
73 %% Eigen Calculations
74
75 % Hamilton Matrix Join Kinetic Energy Matrix With Potential Energy Matrix
76 H_hat = Cdx*T_hat + V_hat;
77
78 %solve using eig
79 [psi_e,Energy] = eig(H_hat);
80
81 %%
82
83 disp('Calculating of Energy States');
84 E = diag(Energy)/e;
85 r =5;
86 E = E(1:r);
87
88 %filter out unique ones
89 c = 1;
90 E_old = 0;
91 % for i=1:length(E)
92 %     if abs(E(i) - E_old) > 0.5;%  && E(i) < V_upper;
93 %         E_new(c) = E(i) ;
94 %         c= c+1;
95 %         E_old = E(i);
96 %     end
97 % end
98
99 %EnergyPlotter(x,V/e,E,8,8,2,12);
100 psi = psi_e(:,1:r);
101
102 disp('Plotting Wave Functions');
103 PlotGraphsE(-psi,E,V/e,dx,N,x)
104
105 toc;
```

The Script to plot wave functions and energy states.

```
function PlotGraphsE(psi,Energy,V,dx,N,x)

%global V_upper V_lower Lw wb
i =1; %for the ease of plotting knowing how many figures plotted
%funtion to plot wave function and probability of finding the particle
xmin = 8;
xmax = xmin;
ymin_p = 0;
ymax_p = 3e-3;
for e = 1:length(Energy)


    psi_e = psi(:,i);
%     if e==1
%         psi_e = -psi_e;
%     end
    ymax = max(psi_e);
    figure(i);
    subplot(211);
    yyaxis right
    plot(x*10^10,V,'k','LineWidth',2)
    ylabel('Potential Energy eV','fontweight','bold')
    axis([-xmin xmax -2 max(V)+2])
    hold on
    grid on
    yyaxis left
    plot(x*10^10,psi_e,'r','linewidth',1);
    axis([-xmin xmax -0.1 1.5*max(max(psi))])
    hold on
    xlabel('Well Length A^0','fontweight','bold')
    ylabel('\psi(x)','fontweight','bold','fontsize',16)
    title(sprintf('n =  %.0f', i),'fontsize',16)
    subplot(212);
    yyaxis right
    plot(x*10^10,V,'k','LineWidth',2)
    ylabel('Potential Energy eV','fontweight','bold')
    hold on
    grid on
    yyaxis left
    plot(x*10^10,psi_e.^2,'r','linewidth',1);
    hold on
    xlabel('Well Length A^0','fontweight','bold')
    ylabel('|\psi(x)|^2','fontweight','bold','fontsize',16)
    axis([-xmin xmax -0 1.5*max(psi_e.^2)])
    i=i+1;
end
figure(i);
ymin = 2;
ymax = 12;
plot(x*10^10,V,'k','linewidth',2)
hold on
grid on

%function to plot energy levels with potential function
for e = 1:length(Energy)
```

```
56      plot(x*10^10,Energy(e)*ones(size(x)),'r','linewidth',2);
57      text(-3,Energy(e)+0.4,sprintf('%.3f eV  n_e = %.0f',Energy(e),e),'
            fontweight','bold','fontsize',12)
58      hold on
59  end
60  axis([-xmin xmax -ymin max(V)+4])
61  xlabel('Well Length A^0','fontweight','bold')
62  ylabel('Potential Energy eV','fontweight','bold')
63
64  end
```

The general script to produce a wide range of potential functions.

```
1   function [V_hat,V] = potential_Generator(sel,x,debug)
2
3   global Lw Wb e A_0 V_upper;
4   V = zeros(1,length(x)); %create and initially assign the matrix
5
6   %selection of potential functions to perform simulations
7   if  sel==1
8       f_linear = @(x)2*V_upper*x*A_0/Lw/A_0 +(2*V_upper);
9       V_linear = generatePotentialLinear(f_linear,V,x)*e; %linear well
10      V = V_linear;
11      disp('Running Sim for Linear Potential Well');
12  elseif sel == 2
13      f_quad   = @(x) 2*V_upper/((Wb-Lw)*A_0)^2*(x*A_0 + Lw*A_0).^2;
14      V_quad = generatePotentialLinear(f_quad,V,x)*e; %quadWell
15      V = V_quad;
16      disp('Running Sim for Quadratic Potential Well');
17  elseif sel ==3
18      f_harmo  = @(x)V_upper/(Lw*A_0)^2*(x*A_0).^2;                %function to
            generate harmonic oscialltor
19      V_harmon= f_harmo(x)*e; %Harmonic oscillator
20      V = V_harmon;
21      disp('Running Sim for Harmonic Oscillator Potential Well');
22  elseif sel==4
23      V = generatePotential(V,x)*e;   %finite square well
24      disp('Running Sim for Square Potential Well');
25  elseif sel==5
26      disp('Running Sim for Stepped Potential Well');
27      V =V_upper*e*ones(1,length(x));
28      V(-Wb <= x &x<=Wb ) = 0;
29      V( x >= 0 & x<= Wb ) = V_upper*e/2;
30  elseif sel==6
31      disp('Running Sim for Double Well Potential');
32       V = zeros(1,length(x));
33       V(-Wb/2 <= x & x <= Wb/2 ) = 4*V_upper*e;
34       for i = 1:length(x)
35           if x(i)<=-Wb|| x(i)>=Wb
36               V(i) = 2*V_upper*e;
37           end
38       end
39  elseif sel==7
40      disp('Running Sim for Triangle Well Potential');
41      for i = 1:length(x)
42          if x(i) <=-Wb || x(i)>=Wb
```

```
43              V(i) = V_upper*e;
44          elseif  -Wb<=x(i)  && x(i)<=0
45              V(i) =-e*x(i)*A_0*10/4;
46          else
47              V(i) =e*x(i)*A_0*10/4;
48          end
49      end
50   elseif sel==9
51      disp('Running Sim for Asymmetrical Double Well Potential');
52       V = zeros(1,length(x));
53       V(-Wb/2 <= x & x <= Wb/2 ) = 4*V_upper*e;
54       for i = 1:length(x)
55           if x(i)<=-Wb
56               V(i) = 2*V_upper*e;
57           elseif x(i) >=Wb+(1/A_0)
58               V(i) = V_upper*e*3;
59           end
60       end
61       V = zeros(1,length(x));
62       V(-Wb/2 <= x & x <= Wb/2 ) = 4*V_upper*e;
63       for i = 1:length(x)
64           if x(i)<=-Wb
65               V(i) = 2*V_upper*e;
66           elseif x(i) >=Wb+(1/A_0)
67               V(i) = V_upper*e*3;
68           end
69       end
70  elseif sel==8
71      disp('Running Sim for Series Well Potential');
72       V = zeros(1,length(x));
73       z = [1,2,3,4];
74
75       for i = 1:length(x)
76           if x(i)*A_0 >=-2 && x(i)*A_0 <=-1
77               V(i) = V_upper*e;
78           elseif x(i)*A_0 >=-5 && x(i)*A_0 <=-4
79               V(i) = V_upper*e;
80             elseif x(i)*A_0 >=-8 && x(i)*A_0 <=-7
81               V(i) = V_upper*e;
82             elseif x(i)*A_0 >=1 && x(i)*A_0 <=2
83               V(i) = V_upper*e;
84               elseif x(i)*A_0 >=4 && x(i)*A_0 <=5
85               V(i) = V_upper*e;
86               elseif x(i)*A_0 >=7 && x(i)*A_0 <=8
87               V(i) = V_upper*e;
88           end
89
90       end
91  elseif sel==10
92      disp("Running for Free Particle");
93      V = zeros(1,length(x));
94
95
96  %       for r = 1:length(z)-2
97  %           for i = 1:length(x)
98  %               if abs(x(i))*A_0 <=z(2*r-1) && abs(x(i))*A_0>=z(r)
```

```
%                   V(i) = 0;
%               else
%                   V(i) = V_upper*e;
%               end
%           end
%       end
%       for i = 1:length(x)
%       if x(i)*A_0 >=-1 && x(i)*A_0<=1
%           V(i) = 0;
%       elseif x(i)*A_0 <=-7 || x(i)*A_0>=7
%           V(i) = V_upper*e;
%       end
%       end
%

%       V(-Wb/2 <= x & x <= Wb/2 ) = 4*V_upper*e;
%       for i = 1:length(x)
%           if x(i)<=-Wb
%               V(i) = 2*V_upper*e;
%           elseif x(i) >=Wb+(1/A_0)
%               V(i) = V_upper*e*3;
%           end
%       end

end

%for debug to check if potential well is correct
if debug
    plot(x*10^10,V/e);
end

V_hat = zeros(length(x),length(x));

%fill the matrix correct indices with the potential values
for r = 1:length(x)
   V_hat(r,r) = V(r);
end
end
```

B.2 Shooting Method Codes

```
%%% Author - M Shihar
%%% ID     - 208628A
%%% Electronics Devices
%%% Shooting Method

clc;
close all;
clear ;
tic;

%%
disp('Running Shooting Method Simulation of Schrodinger 1D Equation');
disp('Initialising Variables');
global m_e h_bar e V_upper V_lower A_0 Wb Lw epsilon;
```

```
m_e = 9.11e-31;        %mass of electron
e = 1.602e-19;         %charge of electron to convert for eV
h_bar = 1.055e-34;      %planks constant divided by 2
A_0 = 10^10;    % convert angstrom to nm ease of plotting the xaxis

%% Init variables for potential wells

V_upper = 10;
V_lower = 0 ;
Lw = 8e-10; %in nm
Wb = Lw/2 ; %well boundary
N = 1000;
x = linspace(-Lw,Lw,N);    %in nm
dx = x(2)-x(1);
epsilon = 1e-5;   %tolerance level for shooting method

%% potential functions to run the 1D sim
disp('Generating Potential Function');
%sel = 1 ;         %Linear
% sel = 2    ;      %Quadratic
% sel = 3   ;       %Harmonic
sel = 0    ;      %Square

%debug = true;   %to view potential function
debug = false ;

V  = potentialGenShooting(sel,x,debug);

%% setting up Energy values for shooting method
disp('Generating Energy Vector for Shooting Method');
E1 = 0;
E2 = 10;
nE = 10000001;
E = linspace(E1,E2,nE);      %build the energy vector to loop through for
    shooting method

%% find eigen energies using shooting method
disp('Shooting Simulation for Calculating Energy States')
%function to find eigen energy
Energy = EigenSim(E,V,dx,N) ;%find eigen energies

%% plot again the functions properly with proper E values
disp('Plotting Wave Functions')
PlotGraphs(Energy,V,dx,N,x);
toc;
disp('End of Simulation')
```

The script to generate Square Potential Well.

```
function potentialFunct = generatePotential(V,x,Wb)
global V_upper V_lower
    for i = 1:length(x)
        if -Wb<=x(i) && x(i)<=Wb
            V(i) = V_lower;
        else
            V(i) = V_upper;
```

```
        end
    end
    potentialFunct = V;
end
```

The Script to plot wave functions and energy states.

```
function PlotGraphs(Energy,V,dx,N,x)
i =1;
%funtion to plot wave function and probablity of finding it

%limits to plot function
xmin = 8;
xmax = xmin;
ymin_p = 0;
ymax_p = 3e-3;
%call from the 2nd energy as the first one is 0 so to bypass it
for e = 2:length(Energy)
    psi_e = wave_function(Energy(e),V,dx,N); %generate wave fucntion for eigen
        energy states using found
    figure(i);
    subplot(211);
    yyaxis right
    plot(x*10^10,V,'k','LineWidth',2)
    ylabel('Potential Energy eV','fontweight','bold')
    axis([-xmin xmax -2 12])
    hold on
    grid on
    yyaxis left
    plot(x*10^10,psi_e,'r','linewidth',1);
    axis([-xmin xmax -0.1 1.5*max(max(psi_e))])
    hold on
    xlabel('Well Length A^0','fontweight','bold')
    ylabel('\psi(x)','fontweight','bold','fontsize',16)
    title(sprintf('n =  %.0f', i),'fontsize',16)
    subplot(212);
    yyaxis right
    plot(x*10^10,V,'k','LineWidth',2)
    ylabel('Potential Energy eV','fontweight','bold')
    hold on
    grid on
    yyaxis left
    plot(x*10^10,psi_e.^2,'r','linewidth',1);
    hold on
    xlabel('Well Length A^0','fontweight','bold')
    ylabel('|\psi(x)|^2','fontweight','bold','fontsize',16)
    axis([-xmin xmax -0 1.5*max(psi_e.^2)])
    i=i+1;
end
figure(i);
ymin = 2;
ymax = 12;
plot(x*10^10,V,'k','linewidth',2)
hold on
grid on
%function to plot energy levels with potential function
```

```
for e = 2:length(Energy)
    plot(x*10^10,Energy(e)*ones(size(x)),'r','linewidth',2);
    text(-3,Energy(e)+0.4,sprintf('%.3f eV      n_E = %.0f',Energy(e),e-1),'
        fontweight','bold','fontsize',12)
    hold on
end
axis([-xmin xmax -ymin max(V)+4])
xlabel('Well Length A^0','fontweight','bold')
ylabel('Potential Energy eV','fontweight','bold')

end
```

The general script to produce a wide range of potential functions.

```
function potential = potentialGenShooting(sel,x,debug)

%generating symmetrical potential functions for shooting method
global Wb Lw V_upper A_0;
V = zeros(1,length(x)); %create and initially assign the matrix
if sel ==1
    disp('Running Simulation for Linear Well Using Shooting Method');
    f_linear = @(x)2*V_upper*x*A_0/Lw/A_0 +(2*V_upper);                %function
        to generate Linear Well
    V_linear      = generatePotentialLinear(f_linear,V,x,Wb);              %
        linear well
    V = V_linear;
elseif sel==2
    disp('Running Simulation for Quadratic Well Using Shooting Method');
    f_quad    = @(x) V_upper/((Wb-Lw)*A_0)^2*(x*A_0 + Lw*A_0).^2;
    V_quad        = generatePotentialLinear(f_quad,V,x,Wb);              %
        quadWell
    V = V_quad;
elseif sel==3
    disp('Running Simulation for Harmonic Oscillator Well Using Shooting
        Method');
    f_harmo   = @(x)V_upper/(Lw*A_0)^2*(x*A_0).^2;  %function to generate
        harmonic oscialltor
    V_harmon      = f_harmo(x);
    V = V_harmon;
elseif sel==0
    disp('Running Simulation for finite Square Well Using Shooting Method');
    V = generatePotential(V,x,Wb);                                %finite square
        well

end

%required to check the plot of the potential function to see if it was
%developed appropriately
if debug
plot(x*10^10,V);
end

potential = V ;
end
```

The script to perform Eigen Energy Simulation.

```
function EnergyEigen = EigenSim(E,V,dx,N)
global epsilon;
i = 1;
    for e = 1:length(E)
        psi = wave_function(E(e),V,dx,N);
        %find to see if terminal reaches boundary conditions
        if(abs(psi(end))<epsilon)
            EnergyEigen(i) = E(e);
            i = i+1;
        end
    end
%filter out unique energy wells
E_n_psi = unique(EnergyEigen);
for i = 2:length(E_n_psi)
    n_E(1) = E_n_psi(1);
    if E_n_psi(i)-E_n_psi(i-1)>0.1
        n_E(i) = E_n_psi(i);
    end
end
%the specific energy values
EnergyEigen = unique(n_E);
end
```

The script to generate closed form solution for analytically derived infinite potential well.

```
% E = habr^2*pi^2/2/m n^2;

global m_e h_bar e ;
m_e = 9.11e-31;
e = 1.602e-19;
h_bar = 1.055e-34;
Lw = 8e-10
C =-h_bar^2/2/m_e;


for i = 1:3
    En(i) = -C*3.14^2*i^2/Lw^2/e;
end

x = linspace(0,Lw,N);    %in nm

A = sqrt(2/Lw);
for i = 1:length(x)
    psi(i) = A * sin(1*3.14*x(i)/Lw);
end
plot(x,psi)
```