

Numerical solution of Schrodinger equation for several potential fields

Course name: Numerical Technique Laboratory

Course No: EEE 212

Department of Electrical and Electronics Engineering, Bangladesh University of Engineering and Technology

Submitted by:

Mrinmoy Kundu ID:1706001

Aroni Ghosh ID:1706004

Date of Submission: 18/09/2019

Submitted to: Rajat Chakraborty

**1 Introduction**

**2 Algorithm and Procedure**

**Associated Functions**

**5**

**3**

**4** **GUI Analysis**

**Examples**

**Introduction:**

The project is based on finding a numerical solution of the wave function and the probability distribution of a quantum particle subjected to any of the potential field provided in GUI. The solution and graphs shown for one dimensional situation.

The main gui file project1.m file contains all functions and configuration windows. So, after running project1.m file, “User” will find homepage and there are 3 more subpages in homepage. Two for several well-known potential fields and Kronig-Penny model and the third one is set for user’s arbitrary input potential.

**Provided Potential Fields in Well Known Potential:**

## Algorithm and Procedure

1-dimentional Schrodinger equation is a second order partial differential equation of the form of

−ℏ2/mψ + V(x)ψ= *i* ℏ ψ

But taking reasonable simplification of the solution being ψ(x,t) = *ψ(x).Φ(t),* the partial differential equation can be turned into a differential equation of 1 spatial variable, namely

H*ψ*(x) = E*ψ*(x) … (1)

Where H is called Hamiltonian operator of the form

H = - (ℏ /2m) d2/dx2 + V(x)

Here V(x) is the potential energy function of the particular particle, and m is the mass of the particle.

The objective of our numerical analysis is to solve the vector *ψ*(x). If we think of *ψ* being a vector of n points denoting the value of *ψ* for an equally spaced vector of the spatial variable x, the equation (1) becomes an eigen value problem for unknown E and a matrix operator H, which can be expressed as a matrix with suitable prior analysis. Then the eigen vectors and eigen values of the matrix H will determine the *ψ* and E of a particular potential function.

Since the H operator is a second order differential operator, the numerical scheme of converting the operator into matrix determines the order of error. For order 2, the H matrix is expressed as

ψ ≈ ( ψ(x−d) − 2ψ(x)+ψ(x+d) )/ d2 ≡ (ψn−1−2ψn+ψn+1)/d2

−ℏ2/m (ψn−1−2ψn+ψn+1)/d2+ Vnψn = Eψn

From where H is

H =−ℏ2/md2 +

But for our project, we have used 8 order scheme, and the source of detailed matrix formulation is in the reference. This method of converting the differential equation into eigen value problem is called Numerov method, which was developed by Boris Vasil'evich Numerov, is a well-known numerical method for solving ordinary differential equations of second order that does not contain first order terms.

First of all, we are defining a parameter call x\_t which denotes the length being -x\_t to x\_t of special dimension where the solution of *ψ* will be calculated, practically, the value of *ψ* will be zeros outside this region, this length is the region of interest.

The unit for all of our physical parameters are measured in terms of nanometer (10-9 m) , electron volt (1.6x10-19J) and femto second (10-15s) to avoid round of error and unexpected blown up of numbers because of huge difference in significant figures of quantum mechanical parameters.

Number of grid points determination

The value of N is related to the maximum energy Emax of all stationary states we wish to find.

First, we fix the maximum energy Emax. The minimum local de Broglie wavelength is λ = h/sqrt(2mEm)

Sufficient accuracy is generally obtained by taking the grid spacing dx corresponding to about one point per radian, i.e., dx = λ/2π. The number of grid points needed can be estimated by finding the outer turning points x\_t such that V (x\_t) = Emax, and allowing for an extra 2λ in the classically forbidden region. Thus, N = 2(x\_t/dx + 4 λ) rounded to the nearest integer.

Here is the pseudo code for the overall algorithm to find desired number of eigen vectors *ψ* and eigen values E

% length is in unit of nm  
% energy is in unit of eV  
% time is in unit of fs  
% We can tweak the following parameter x\_t and m  
x\_t = 1; % Half Length of the field  
m = 5.69375; % electron mass in scaled unit  
  
% Grid parameter, to be high for accuracy  
% But too high entails round off error  
E\_max = 10000; % 100000 works well for most of cases  
  
% dx and N is determined from E\_max, x\_t  
global dx;  
global N; global h\_bar ;  
h\_bar = 0.659375; % in scaled unit  
dx = h\_bar/(sqrt(2\*m\*E\_max)); %  
N = round(2\*(x\_t/dx + 2\*pi)); % Number of grid point  
  
% Potentail declaration , V is the potentail energy vector with x being the spatial vector of the field  
[V, x] = function calling with suitable arguments;  
  
% 8th order approximation with numerov method  
[A, B] = Numerov8();  
  
% Hamiltonian matrix  
H = -h\_bar^2/(2\*m) \* (B \ A) + diag(V);  
desired\_mode\_num = 20; % How many eigen state we want  
  
% Finding the Energy and psi vector  
[Es, psis] = FindEigens(H, desired\_mode\_num);

function [A, B] = Numerov8()  
% This function outputs hamiltonian matrix  
% Of 8'th order approximation  
  
global dx;  
global N;  
  
a1(1:N-3)= 2;  
a2(1:N-2)= -27;  
a3(1:N-1) = 270;  
a4(1:N)= -490;  
a5(1:N-1)= 270;  
a6(1:N-2)= -27;  
a7(1:N-3) = 2;  
A = (diag(a1,3) + diag(a2,2)+diag(a3,1)...  
 + diag(a4,0) + diag(a5,-1)+diag(a6,-2)...  
 + diag(a7,-3))/(180\*dx^2);  
  
a1(1:N-3)= 1;  
a2(1:N-2)= -6;  
a3(1:N-1) = 150;  
a4(1:N)= 20140;  
a5(1:N-1)= 150;  
a6(1:N-2)= -6;  
a7(1:N-3) = 1;  
B = (diag(a1,3) + diag(a2,2)+diag(a3,1)...  
 + diag(a4,0) + diag(a5,-1)+diag(a6,-2)...  
 + diag(a7,-3))/20160;  
  
end

function [Es, psis] = FindEigens(H, desired\_mode\_num)  
  
[psis, Es] = eig(H);  
  
[Es,ind] = sort(diag(Es)); % converting E to vector and sort from low to high  
psis = psis(:,ind); % rearranging corresponding eigenvectors  
  
Es = Es(1:desired\_mode\_num);  
psis = psis(:,1:desired\_mode\_num);  
  
%Normalization  
for i = 1:desired\_mode\_num  
 psis(:,i) = psis(:,i) \* sqrt(1/trapzpsis(:,i).^2));  
end  
end

## Potential functions

One of the major parts for constructing the Hamiltonian matrix is to create the potential energy vector. In our project, we worked with several well-known potential function found in quantum mechanical system with suitable flexibility to configure them. All of the functions output is in the same form, being a vector of V and x.

function [V, x] = Square\_potential\_well(max\_strength, well\_width ,x\_t)  
  
global dx;  
global N;  
% Square potential well with symmetrical shape  
% well\_width as the per cent of the total width x\_t  
  
x = linspace(-x\_t,x\_t,N);  
V = zeros(1,N);  
  
V(1: round(N\*(0.5 - (well\_width/2)) )) = max\_strength;  
V(round(N\*(0.5 + (well\_width/2))) : end ) = max\_strength;  
  
end

function [V,x] = Double\_infinite\_potential\_well...  
 (max\_strength, well\_width, barrier\_width,x\_t) global N;  
  
x = linspace(-x\_t,x\_t,N);  
% Double Square potential well with symmetrical shape  
% well\_width, barrier\_width as in percent of x\_t  
V = zeros(1,N);  
V(1: round(N\*(0.5 - (barrier\_width/2) - well\_width)) ) = max\_strength;  
V(round(N\*(0.5 + (barrier\_width/2) + well\_width)) : end ) = max\_strength;  
  
V(round(N\*(0.5 - (barrier\_width/2))) : ...  
 round(N\*(0.5 + (barrier\_width/2)) ) ) = max\_strength; end

function [V, x] = linear\_potentail(N,max\_strength,per\_well\_width,x\_t)  
%%Linear potential well  
  
% max\_strength is the heigth of the potential  
% per\_well\_width is the percent of well width with respect to x\_t  
  
V = zeros(1,N);  
x = linspace(-x\_t,x\_t,N);  
  
temp = (1-per\_well\_width)/2;  
for i = 0:N  
 if(i < N\*temp)  
 V(i+1) = max\_strength/(temp\*N) \* i;  
 elseif (i > N\*(1-temp))  
 V(i) = -max\_strength/(temp\*N)\* i + ...  
 max\_strength/temp;  
 end  
end  
  
end

function [V,x] = Harmonic\_oscillator(m,x\_t,omega)  
% Harmonic oscillator  
% m is the mass of the particla % omega is the stiffness parameter for harmonic odcillator  
global N;  
x = linspace(-x\_t,x\_t,N);  
V = 1/2\*m\*omega^2\*x.^(2);  
end

function [V,x] = morse\_potential(Depth,a,r\_e)  
  
global N;  
x = linspace(0,r\_e\*5,N);  
V = Depth\*(1-exp(-a\*(x-r\_e))).^2 - Depth ;  
end

function [V,x] = poschl\_teller\_potential(lambda,x\_t)  
global N;  
x = linspace(-x\_t,x\_t,N);  
V = -lambda\*(lambda+1)\*sech(x\*8)/2;  
end

function [V,x] = lennard\_jones\_potential(Depth,r\_e,x\_t)  
  
global N;  
x = linspace(0.001,2\*x\_t,N);  
V = Depth\*((r\_e./x).^12 - 2\*(r\_e./x).^6);  
end

function [V,x] = coulomb\_potential(propotional\_constant,x\_t) global N;  
x = linspace(-x\_t,x\_t,N);  
V = (-(propotional\_constant\*1.44)./abs(x)); %1.44 factor corresponds to the case of hydrogen atom  
V(V==inf) = 100000;  
V(isnan(V)) = 0; end

function [V,x] = kronig\_penny\_potential(max\_strength,num\_of\_well,del\_per\_well,x\_t)  
  
% num\_of\_well is the desired number of well in the field  
global N;  
% del\_per\_well is the distance between consequent wells as a percent of the width of the well  
x = linspace(-x\_t,x\_t,N);  
  
L = round((N/num\_of\_well)/( 1 + del\_per\_well));  
del = round(L\*del\_per\_well);  
  
V = zeros(1,N);  
  
s = del;  
for i = 1:round(N/(L+del))  
 if(s+L > N)  
 V(s:end) = max\_strength;  
 else  
 V(s:s+L) = max\_strength;  
 s = s + L + del;  
 end  
end  
  
end

function [V,x] = user\_defined(fstr,x\_t)  
  
  
global N;  
x = linspace(-x\_t,x\_t,N);  
  
if ischar(fstr)  
 func =str2func(['@(x)' fstr]);  
end  
  
V = func(x);  
  
end

## Associated functions

After finding the eigen values (energy) and eigen vectors (psi), we need to present the result in a user friendly. For this purpose, we have created some associated functions to work with the resulted eigen values and vectors.

Plot\_PDE(psis,modes,x\_t,m,not\_coulomb)

Takes the arguments to plot the probability density of the desired modes of eigen vector

* eigen vectors,
* a vector with of which modes are to be displayed,
* field length x\_t,
* mass
* a binary signal whether the psis corresponds to the coulomb’s potential

function [] = plot\_PDF(psis,modes,x\_t,m,not\_coulomb)  
format long  
  
global h\_bar  
global N;  
global dx;  
  
x = linspace(-x\_t,x\_t,2\*N);  
  
lgd = [];  
  
for i = modes  
 mode = i;  
 %Interpolation for better result  
 detailed\_x\_grid = linspace(1,N,2\*N);  
 polished\_psi = interp1(1:N,psi,detailed\_x\_grid,'spline');  
  
  
 plot(x,polished\_psi.^2);  
 hold on;  
 if not\_coulomb  
 lgd = [lgd ;strcat('psi^2(',num2str(i),')')];  
 else  
 lgd = [lgd ;strcat('psi^2(',num2str(i/2),')')];  
 end  
  
end  
  
title('Probability Density Function');  
  
xlabel('x(nm)');  
ylabel('psi^2');  
xlim([-x\_t,x\_t]);  
  
legend(lgd);  
%ylim([min(V)-10, max(V)+10]);  
hold off  
end

Plot\_V(x,V,Es, not\_coulomb,lgd)

Takes arguments to plot V and the eigen values of the same plot

* the vector of spatial field
* the potential energy vector
* a vector of energy values to be displayed
* a binary signal whether the Es corresponds to the coulomb’s potential
* whether wants to display the legend

function [] = Plot\_V(x,V,Es, not\_coulomb,lgd)  
  
%If lgd is 1, provide legend, otherwise not  
  
global N;  
  
plot(x,V,'LineWidth',2,'DisplayName',...  
 'Potential');  
hold on;  
if not\_coulomb  
 for i = 1:length(Es)  
 plot(x,Es(i)\*ones(1,N),'DisplayName',...  
 strcat('E(',num2str(i),')=',num2str(Es(i)),'eV'));  
 hold on;  
 end  
else % For Coulumbo  
 for i = 2:2:length(Es)  
 plot(x,Es(i)\*ones(1,N),'DisplayName',...  
 strcat('E(',num2str(i/2),')=',num2str(Es(i)),'eV'));  
 hold on;  
 end  
end  
  
title('Potential energy');  
xlabel('x(nm)');  
ylabel('Potential Energy(eV)');  
  
xlim([min(x) max(x)]);  
ylim([min(Es)-10, max(Es)+10]);  
  
if lgd  
 legend();  
end  
hold off;  
end

E\_vs\_n(Es, Samples, type)

Takes arguments to plot the relation between energy values and energy states

* a vector of energy values
* number of energy values to consider from the vector Es
* type of plotting

function [index] = E\_vs\_n(Es, Samples, type) ns = Samples;  
Es = Es(Samples)';  
  
% E is proportional to n^(index)  
[~,index,~] = regression(log(ns), log(Es) ); if type == '.'  
 plot(ns,Es,'.','MarkerSize',15);  
else  
 plot(ns,Es,'.-.','MarkerSize',15);  
end  
  
xlabel('State,n');  
ylabel('Energy,eV');  
title("Energy vs quantum number,n");  
  
lgd = strcat('E is proportional to n^i, i =',num2str(real(index)));  
legend(lgd); end

Time\_Dep\_Real\_Part\_Anim(psi,E,x\_t,cycle,speed)

Time\_Dep\_Im\_Part\_Anim(psi,E,x\_t,cycle,speed)

Takes arguments to plot an animation of the time dependent solution of the particular problem

* particular Eigen vector
* Energy value associated with that
* Field length
* Cycle to show
* speed

function [] = Time\_Dep\_Real\_Part\_Anim(psi,E,x\_t,cycle,speed)  
  
global h\_bar ;  
global N;  
  
x = linspace(-x\_t,x\_t,N);  
  
t\_range = abs(cycle\*pi\*h\_bar/E);  
  
for t = linspace(0,t\_range,t\_range\*100)  
 psi\_ = psi\*sin(-E\*t/h\_bar);  
 plot(x,psi\_);  
  
 lim = max( abs(min(psi)) , abs(max(psi)) );  
  
 title(strcat('Real part of psi, t =', num2str(t), 'fs'));  
 xlabel('x(nm)');  
 ylabel('psi');  
  
 ylim([-lim lim]);  
 pause(speed); %0.05  
end  
end

function [] = Time\_Dep\_Im\_Part\_Anim(psi,E,x\_t,cycle,speed)  
  
global h\_bar ;  
global N;  
  
x = linspace(-x\_t,x\_t,N);  
  
t\_range = abs(cycle\*pi\*h\_bar/E);  
  
for t = linspace(0,t\_range,t\_range\*100)  
 psi\_ = psi\*cos(-E\*t/h\_bar);  
 plot(x,psi\_);  
  
 lim = max( abs(min(psi)) , abs(max(psi)) );  
  
 title(strcat('Real part of psi, t =', num2str(t), 'fs'));  
 xlabel('x(nm)');  
 ylabel('psi');  
  
 ylim([-lim lim]);  
 pause(speed); %0.05  
end  
  
end

[D\_E] = density\_of\_states(Es)  
Takes arguments to draw a crude picture of the density of states for kronig-penny model and returns the density of states corresponding to the energy

* a vector containing all the eigen values of energy

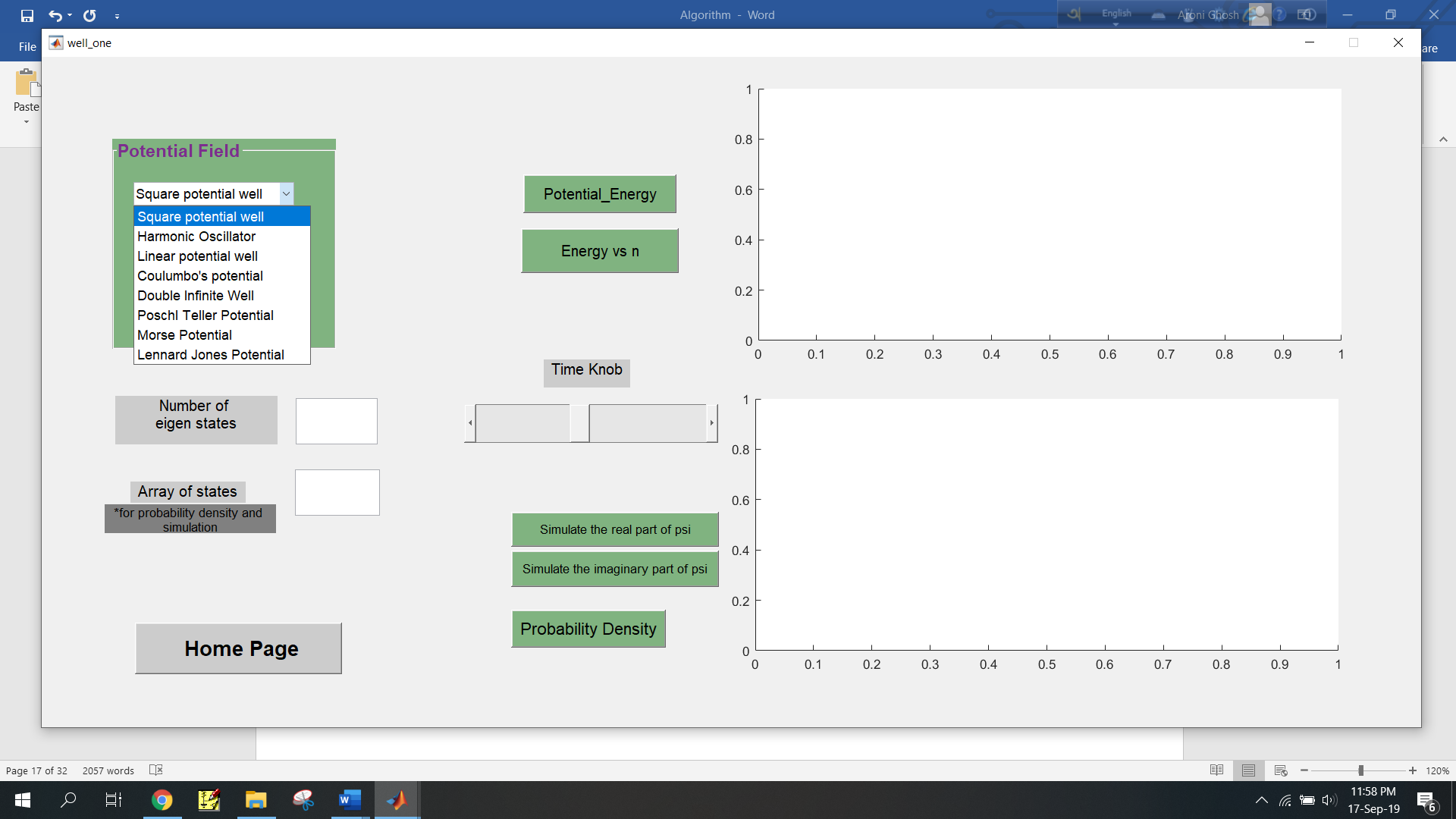
function [D\_E] = density\_of\_states(Es)  
  
D\_E = 1./gradient(Es);  
  
plot(D\_E,Es,'LineWidth',2);  
  
title('Energy vs Density of states]');  
xlabel('D(E)[1/(eV\*nm)]');  
ylabel('E[eV]');  
  
ylim([min(Es) max(Es)]);  
  
end

k\_vs\_E(Es,D\_E, num\_per\_well,del\_per\_well,x\_t)

Takes arguments to draw a crude picture of wave number vs energy graph for kronig-penny model

* a vector containing all the eigen values of energy
* vector containing density of states corresponding to the energy
* number of wells
* distance between consequent wells as a percent of the width of the well
* field length

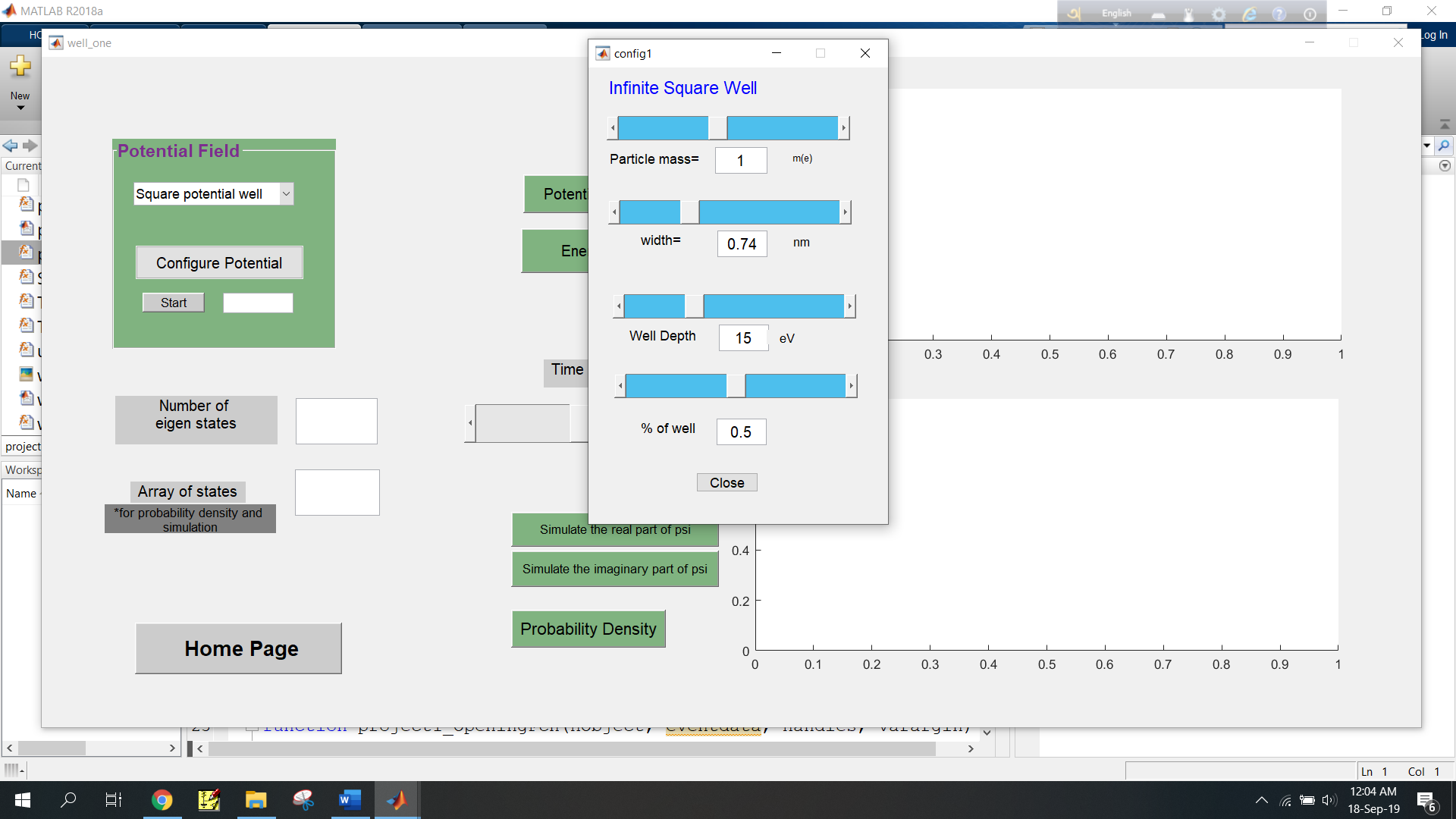
function [] = k\_vs\_E(Es,D\_E,num\_per\_well,del\_per\_well,x\_t)  
  
k = zeros(1,length(Es))';  
a = (2\*x\_t/num\_of\_well)/(1+del\_per\_well);  
for i=2:length(Es)  
 k(i) = (1/2\*pi)\*trapz(Es(1:i),D\_E(1:i));  
end  
  
plot([flip(-k),k]/(pi/a), [flip(Es), Es],'b','LineWidth',2);  
  
title('Wave number vs Energy');  
xlabel('k (pi/a) [1/nm]');  
ylabel('E[eV]');  
  
end

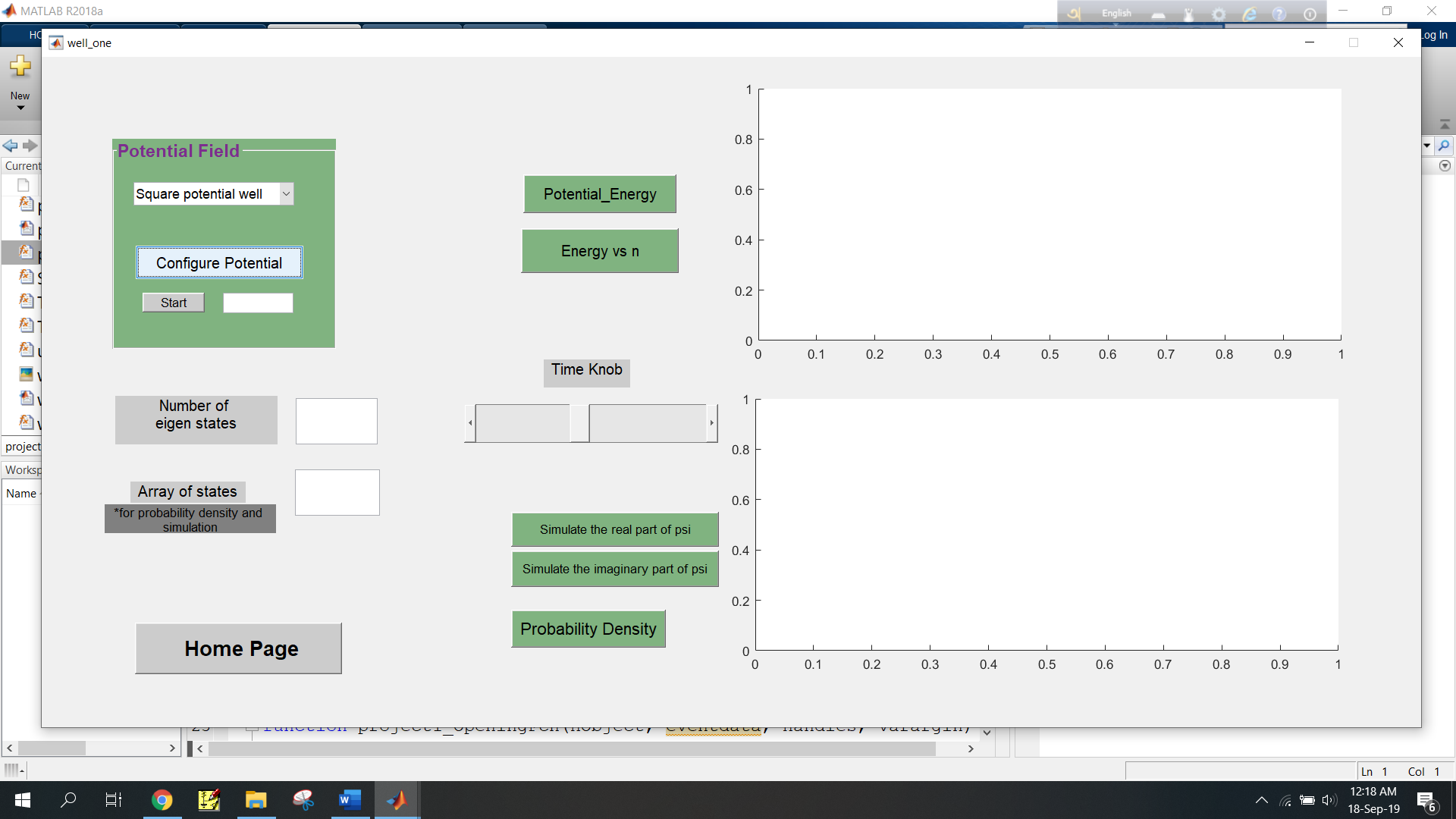
**GUI Analysis**

* Choose any subpage from the homepage

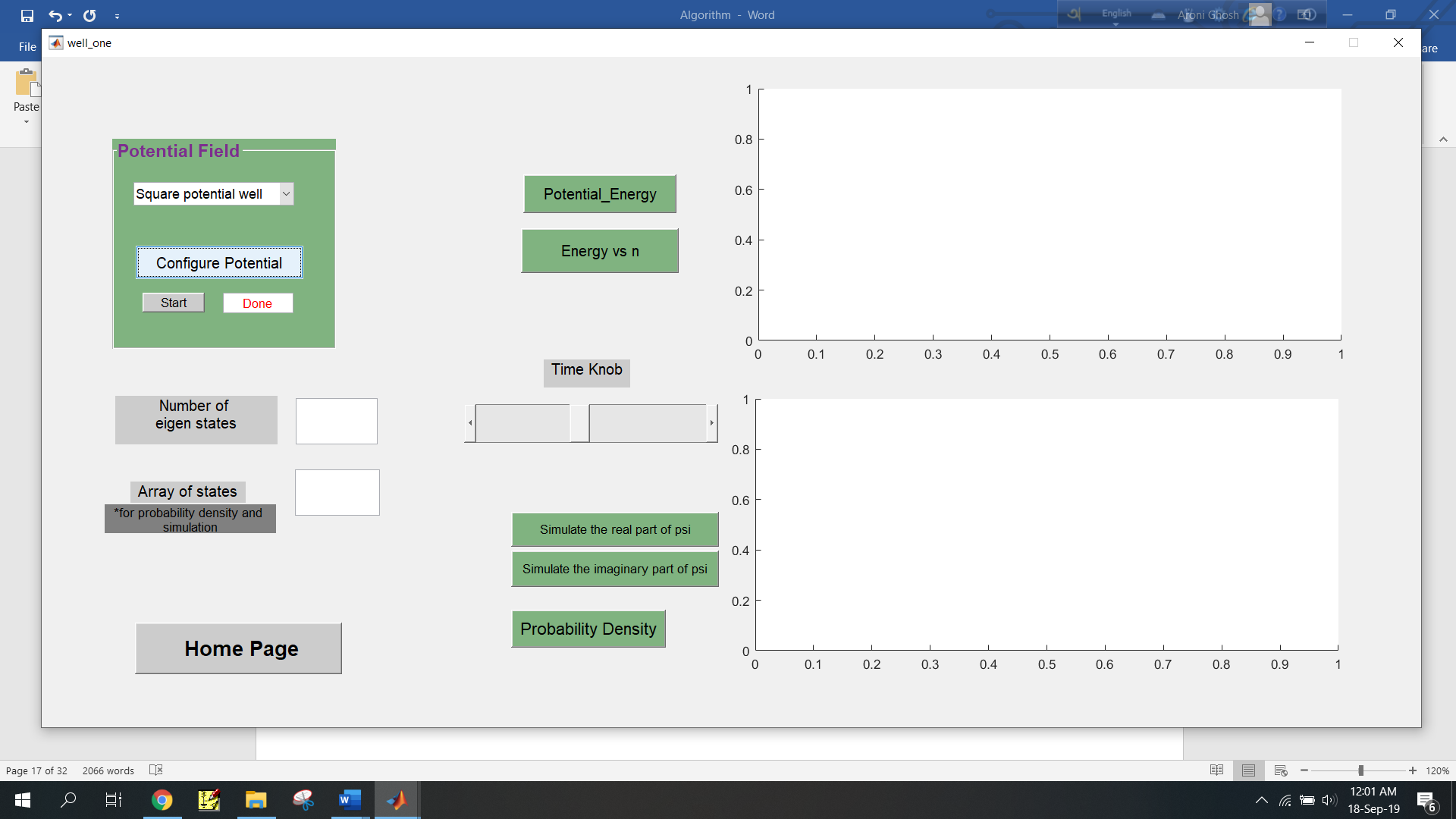
For well-known potential wells-

* Select a potential well from pop-up menu
* Click “Configure Potential” button. A configure window will appear.

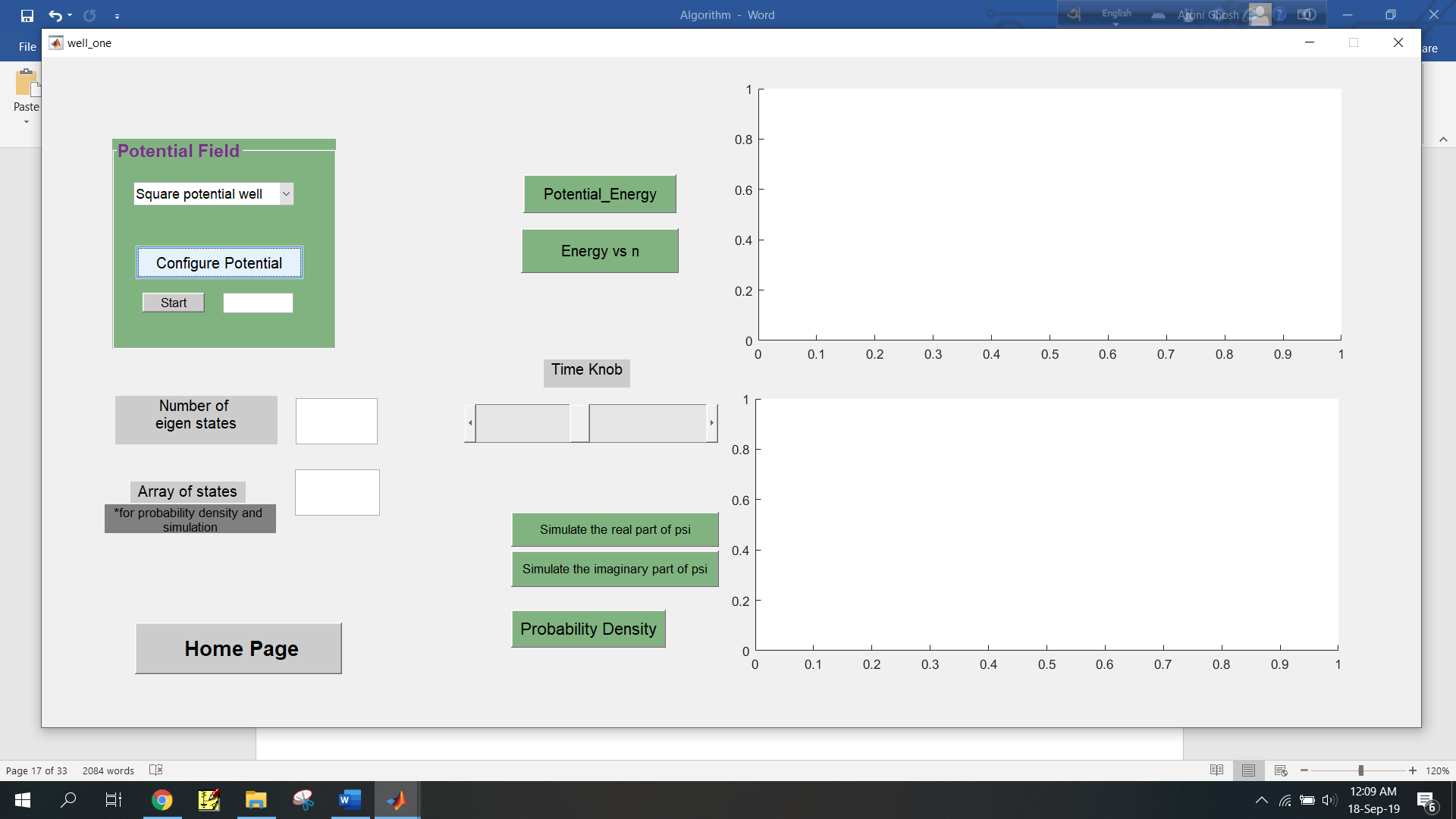




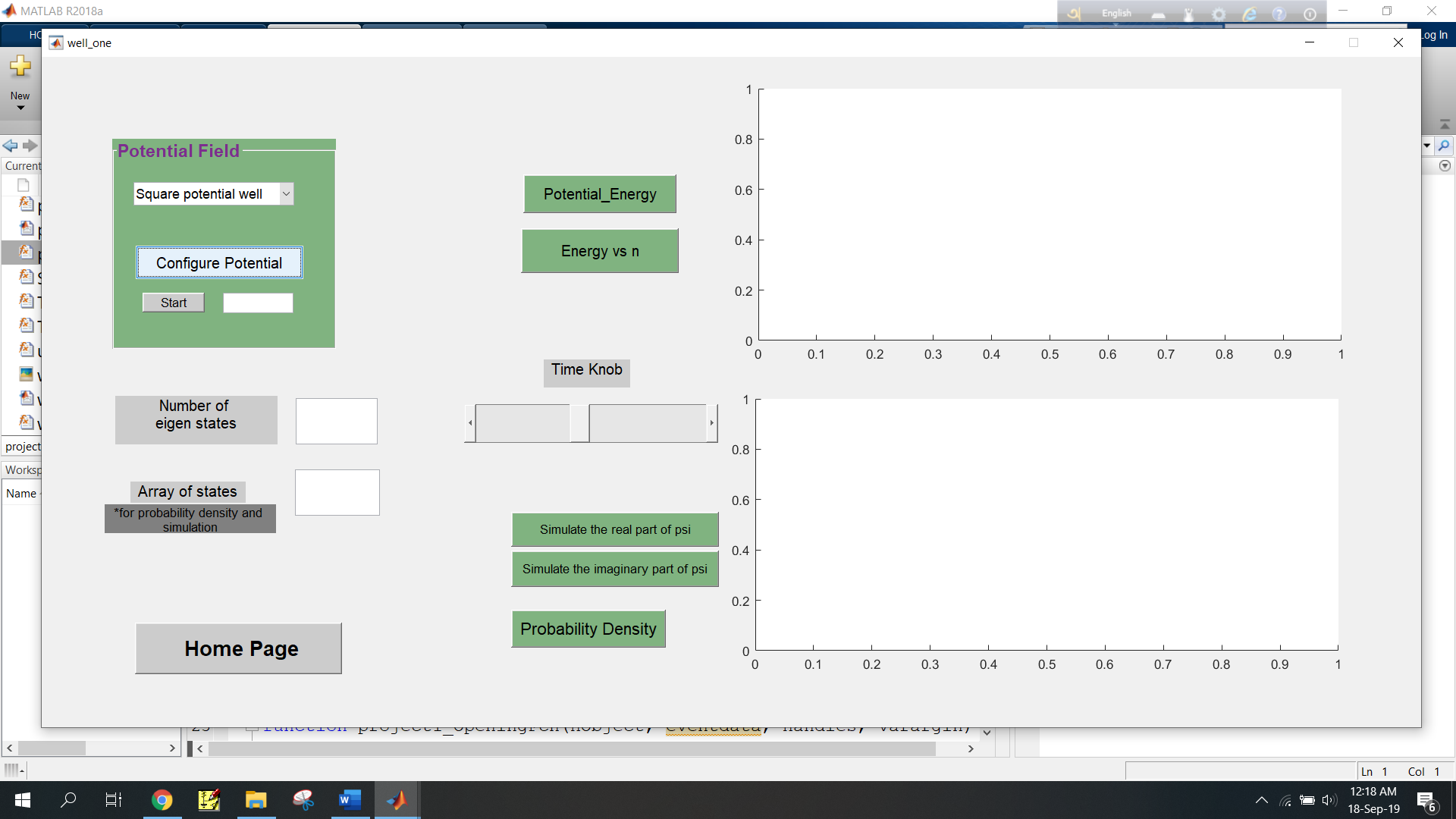
* Move sliders to give inputs and then click “Close”



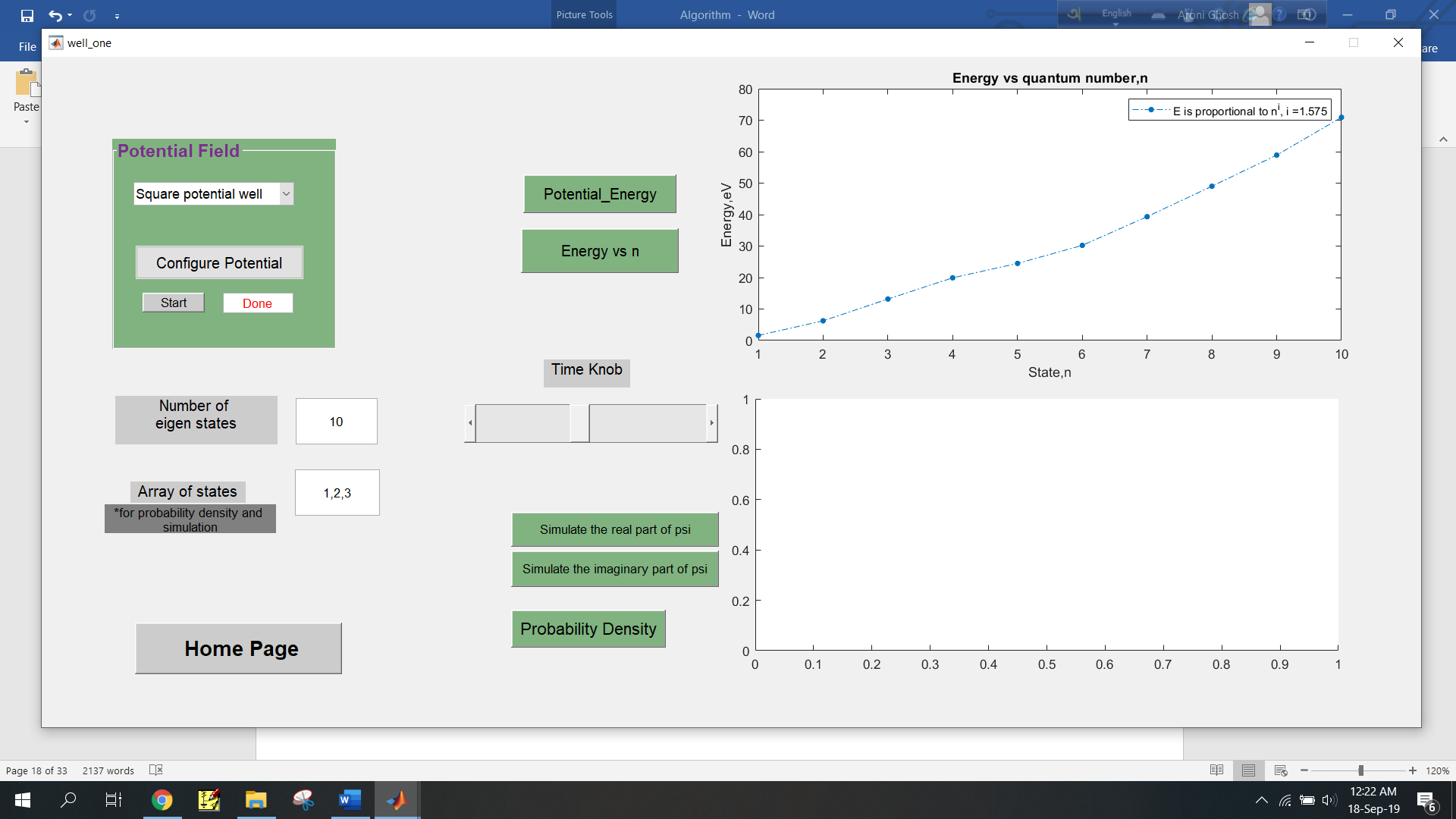
* Click “Start”. After the calculation is over “Done”



* Input the number of eigenstates you want to calculate and array of states you want to see for probability density



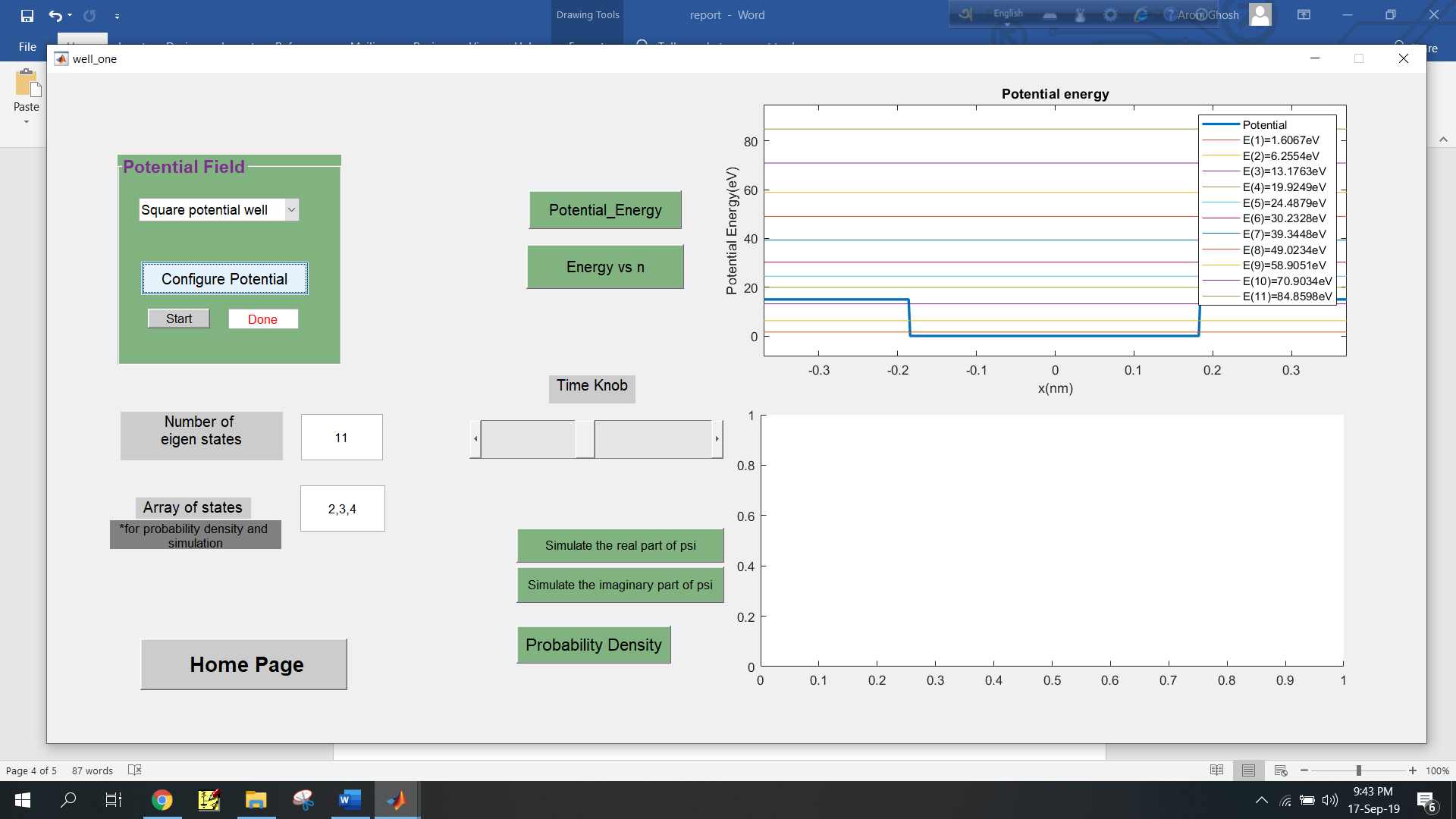
* Click “Potential Energy” button to see potential energy vs width graph and click “Energy vs n” button to see energy vs states graph.



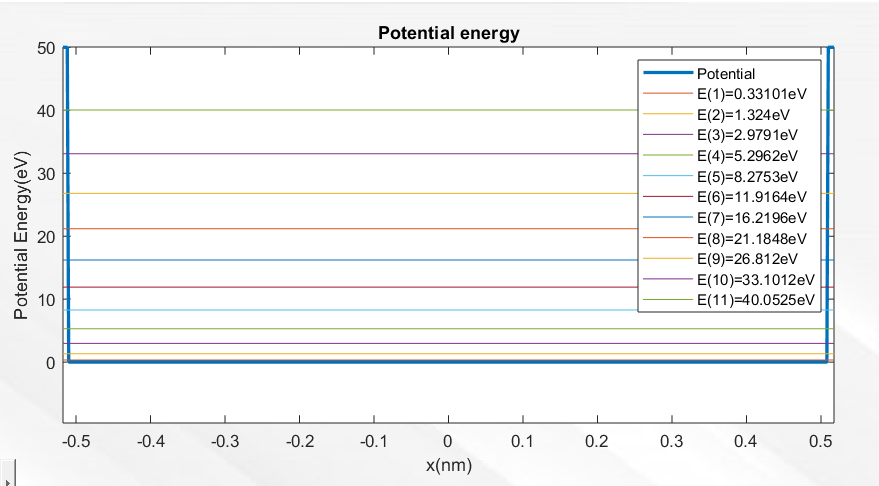
* Click “Probability density” button to get graphs of various states in same plot. The upper two buttons are mainly for real time simulation of real and imaginary part of wave function. Time knob determines speed of the simulation.

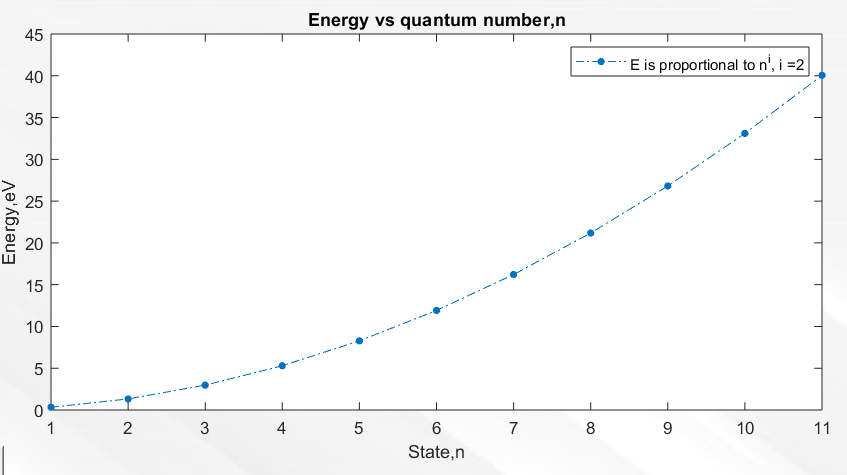
**Examples**

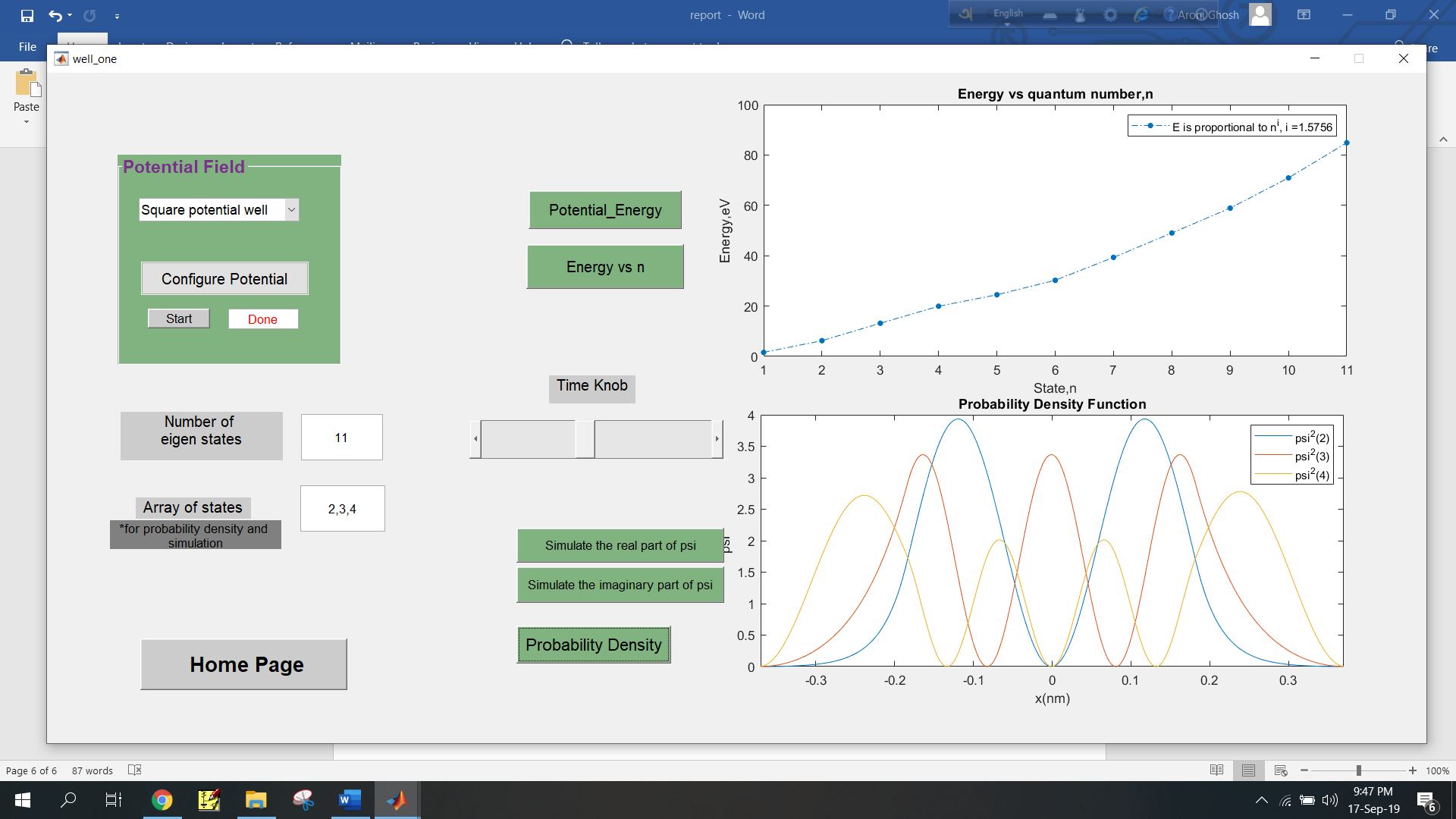
**Square Potential Well:**

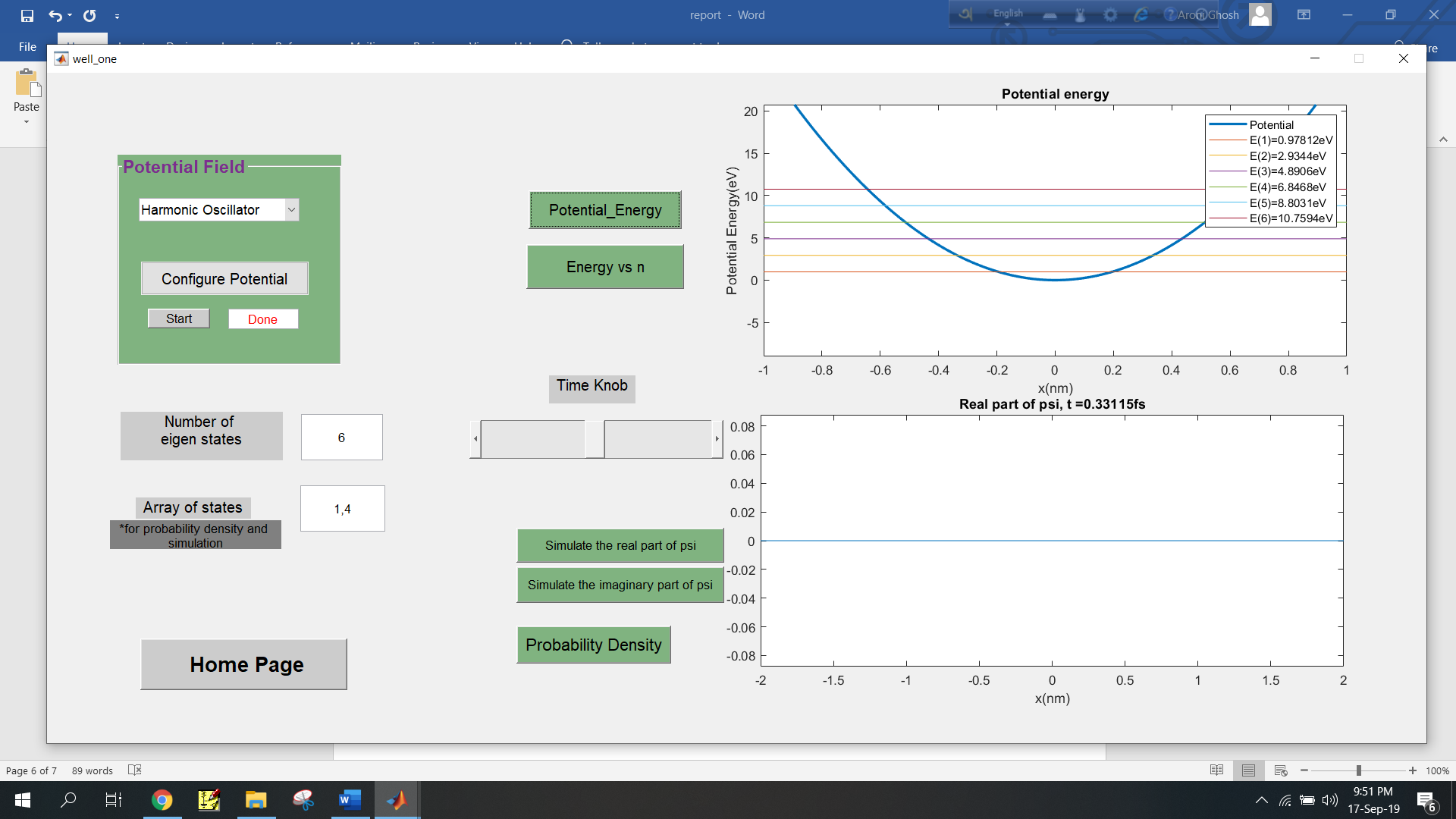


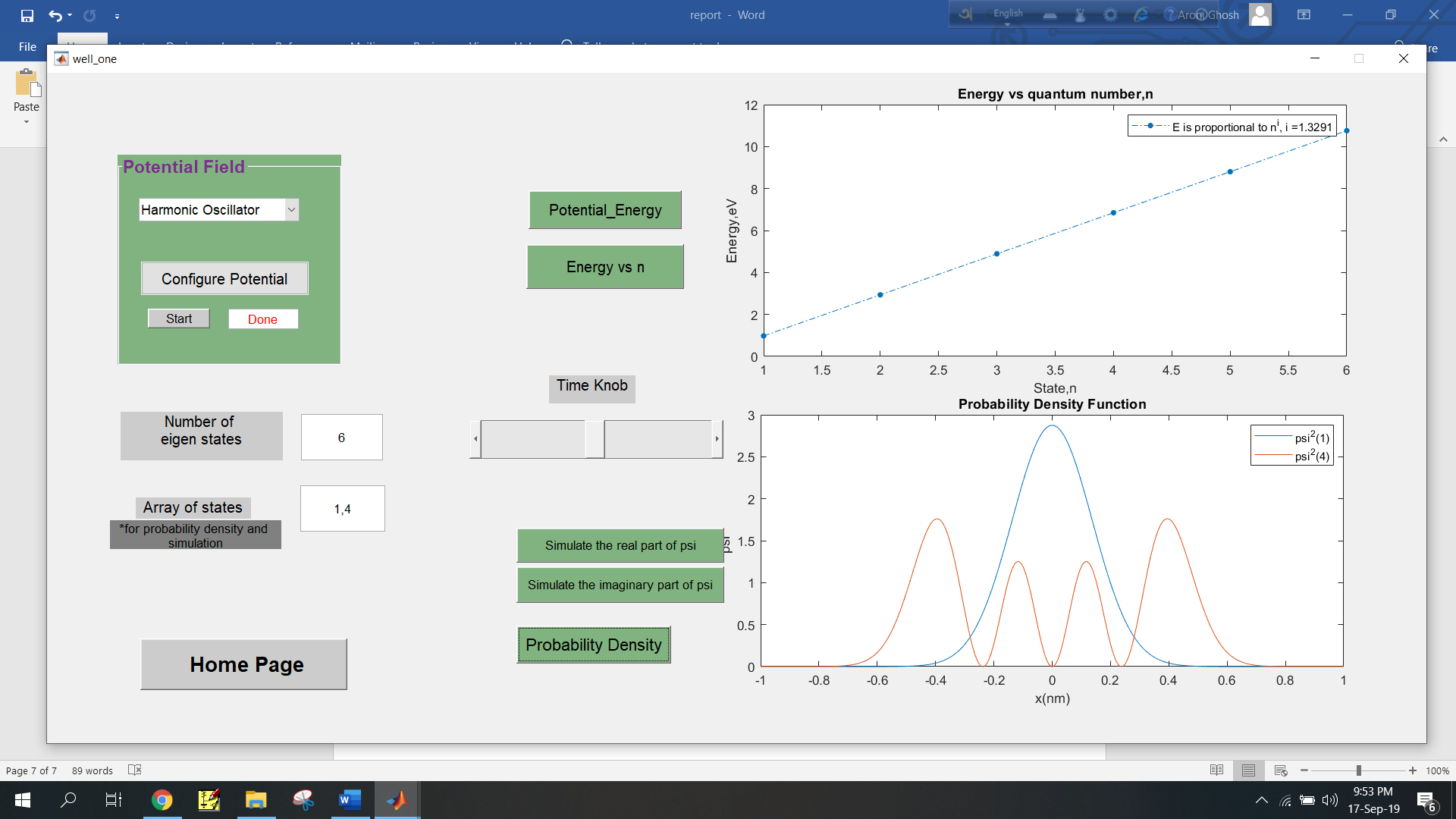
**Output Graphs:**

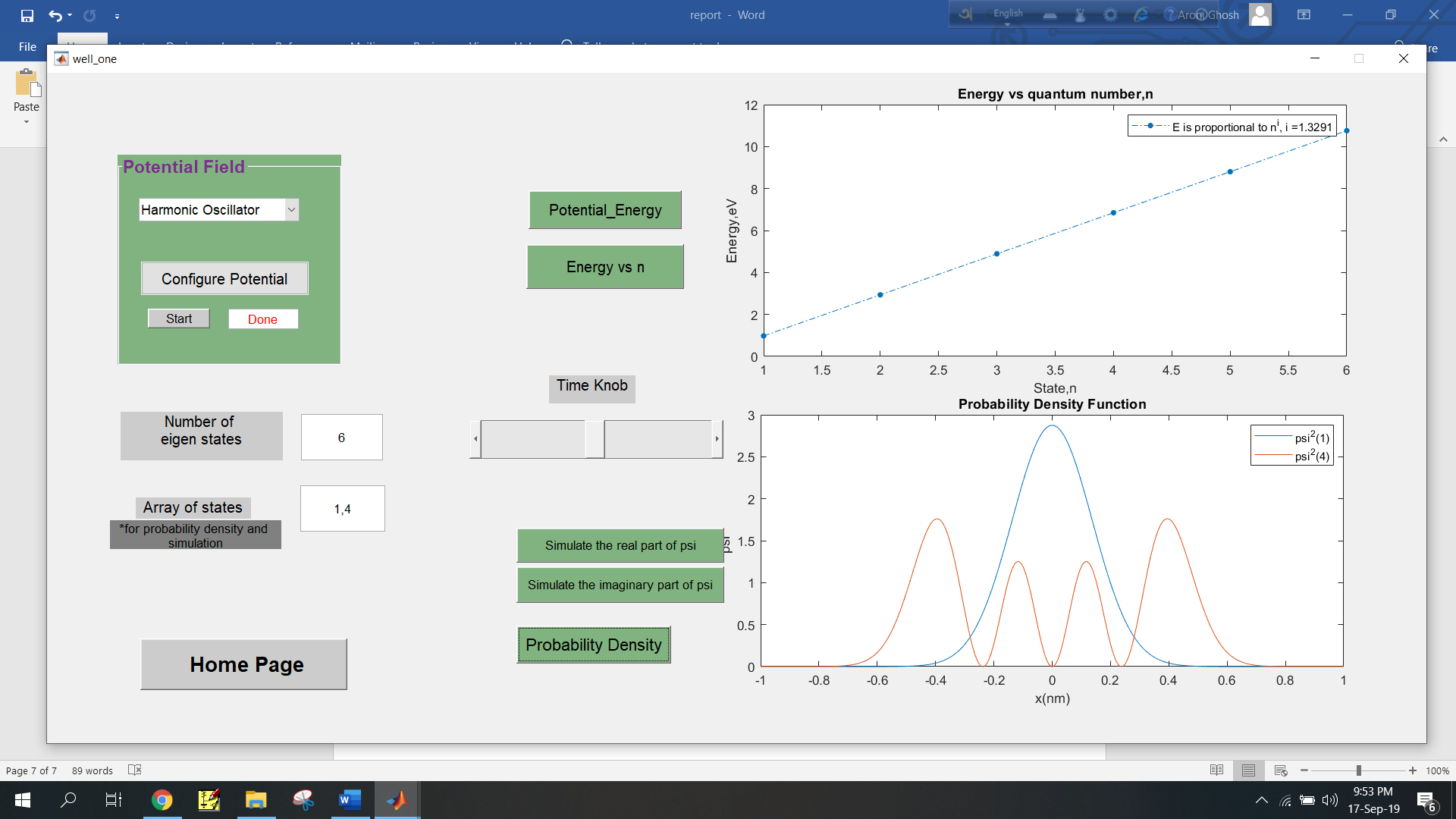
****

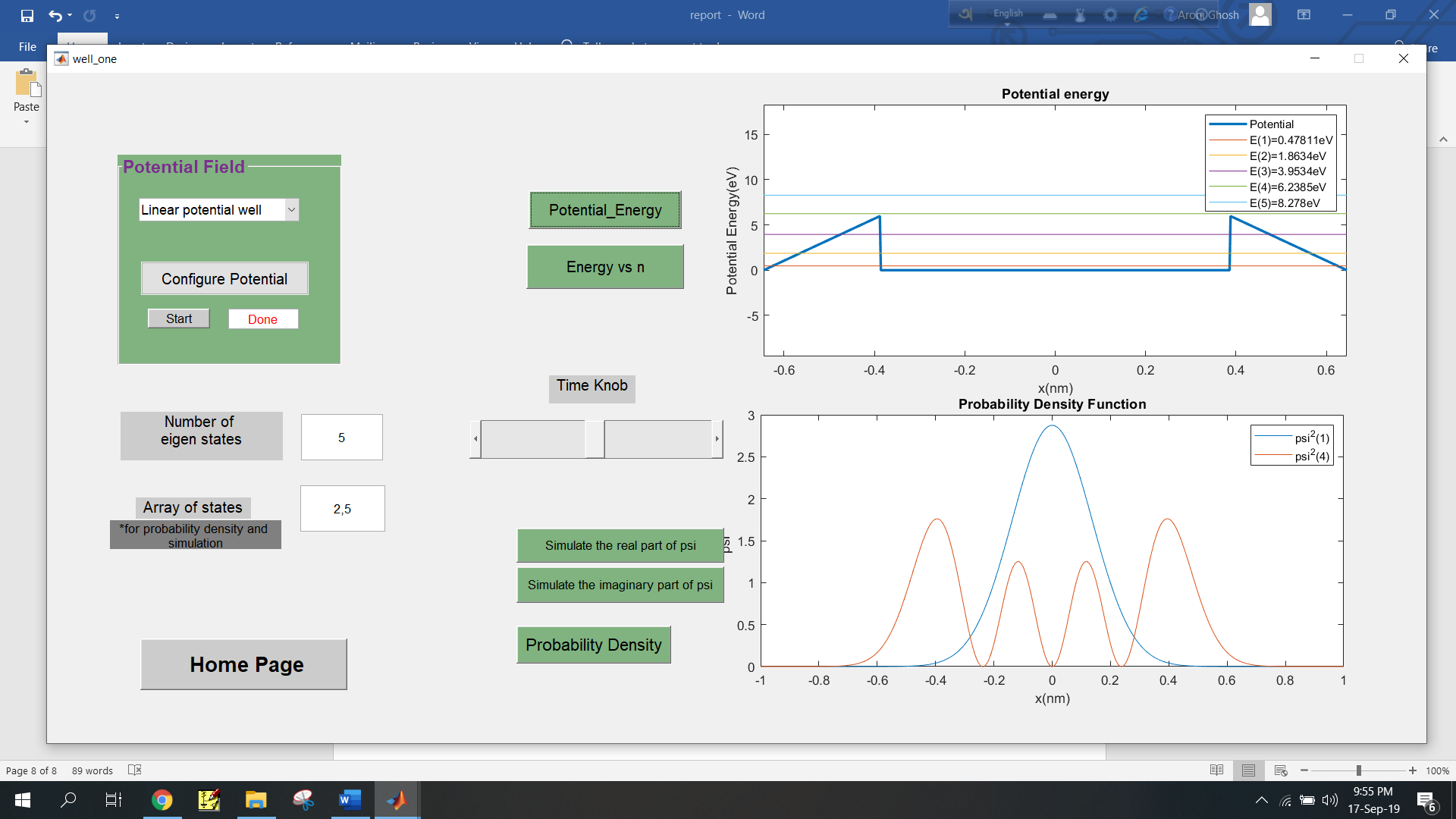
****

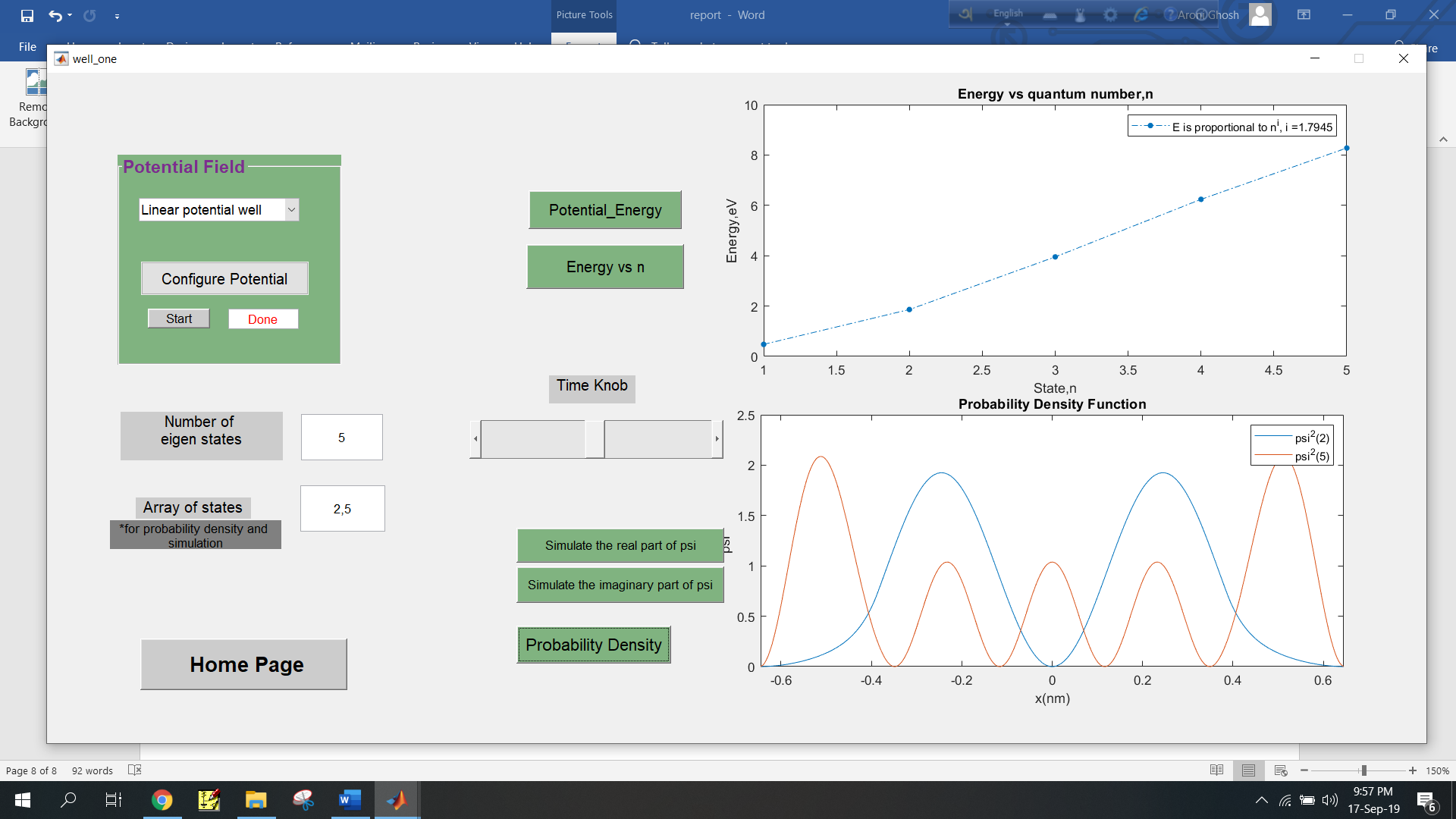
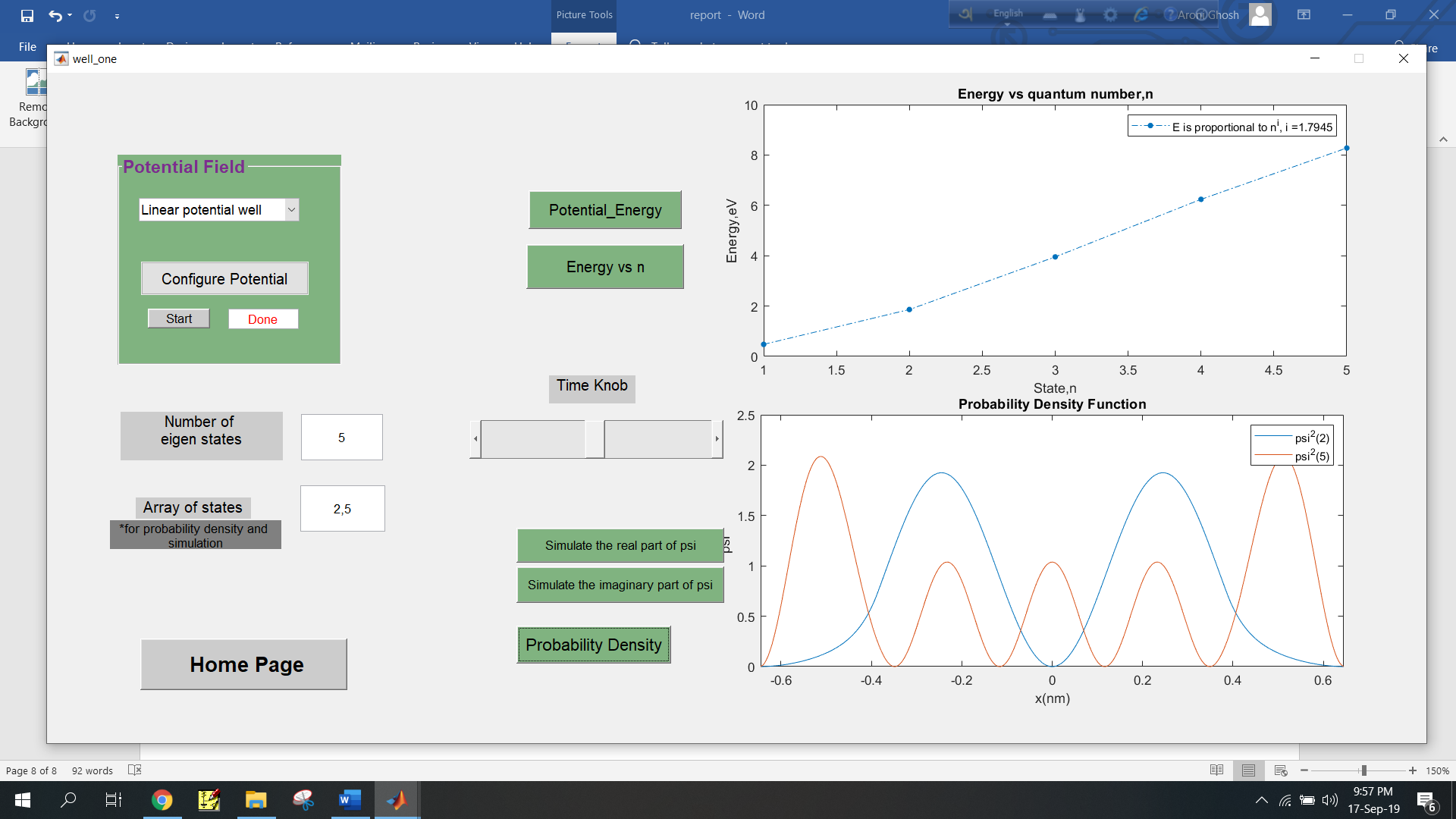


**Harmonic Oscillator:**

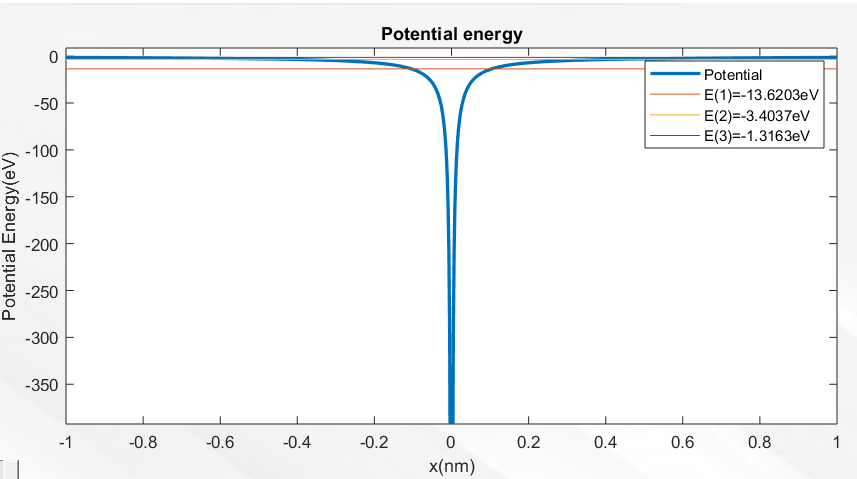


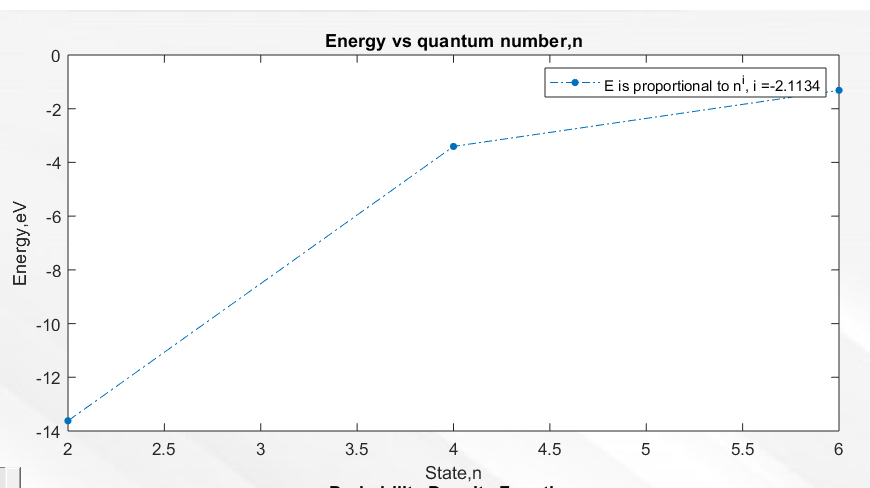


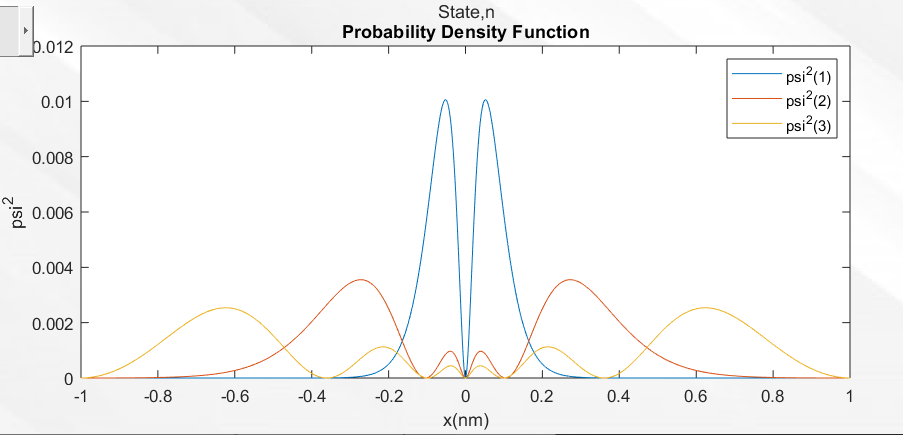
**Linear Potential Well:**



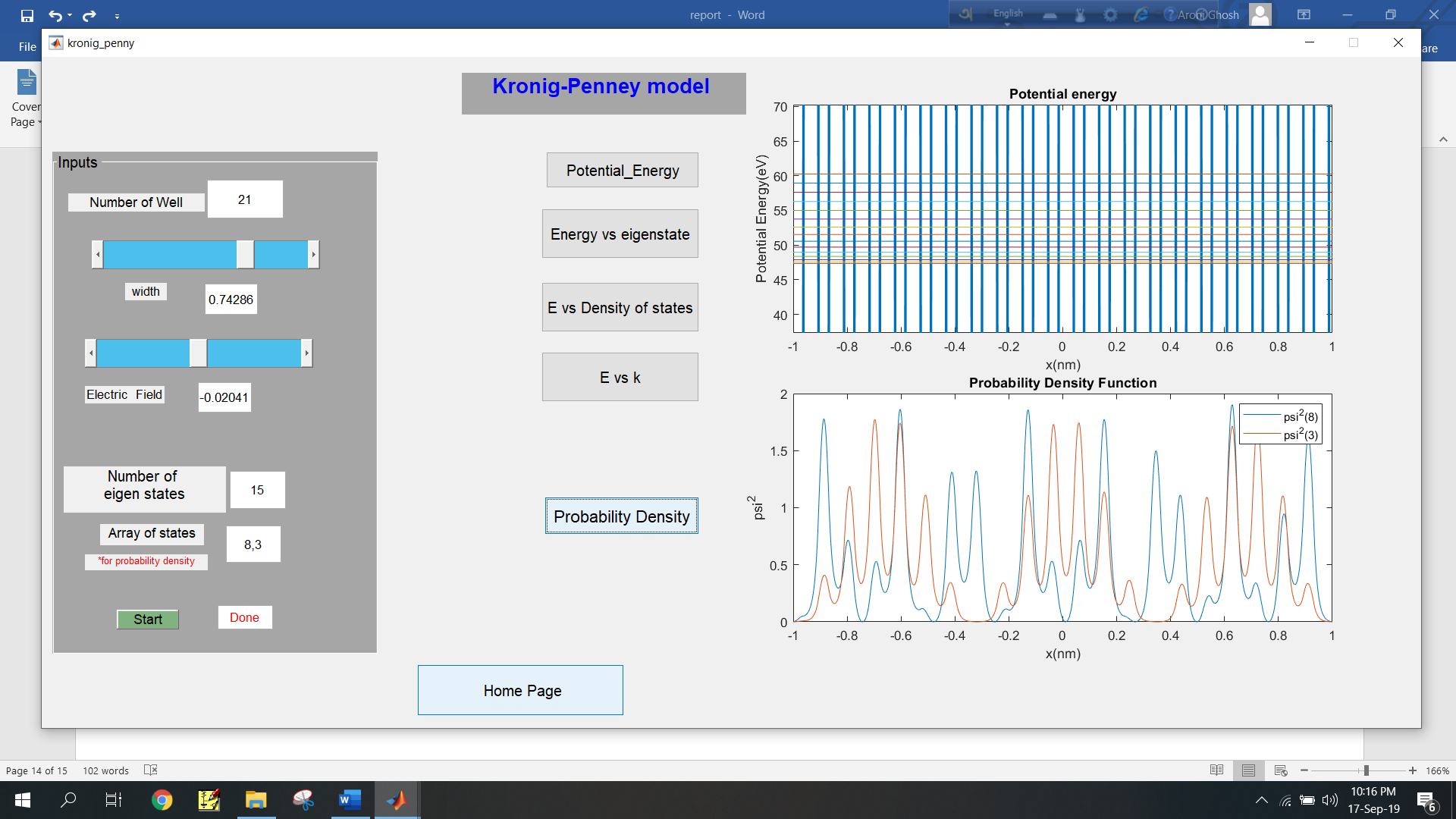
**Coulomb’s Potential:**

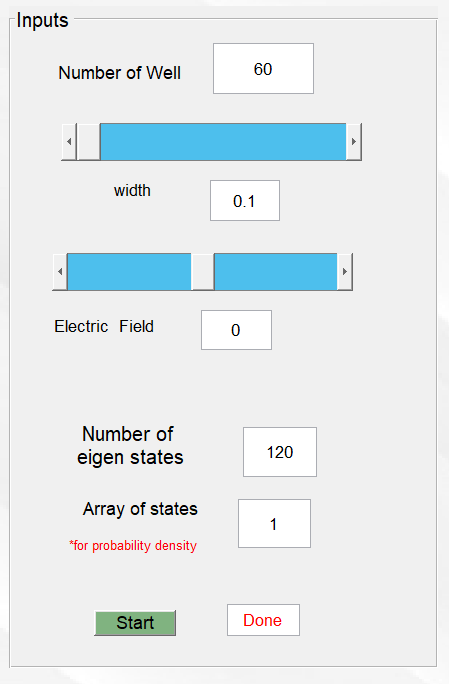
****



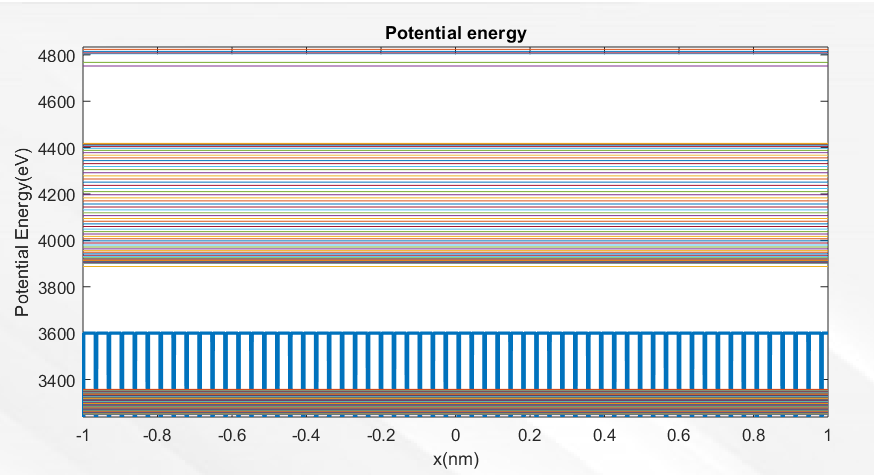


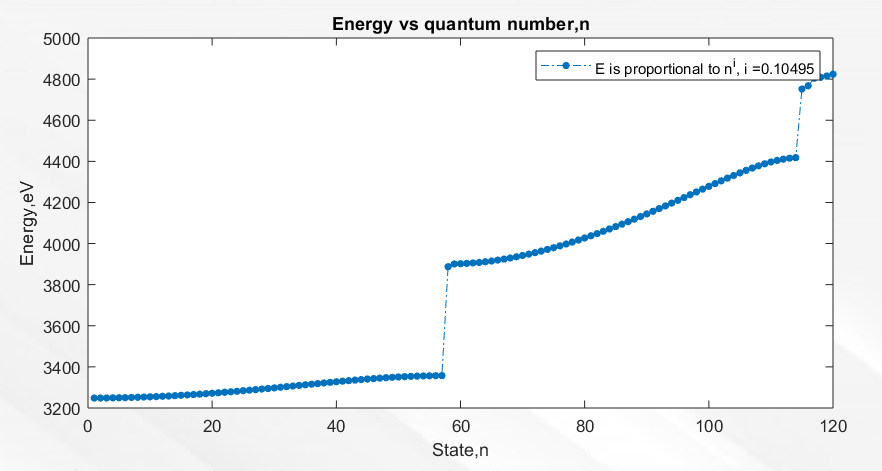
**Kronig Penny Model**

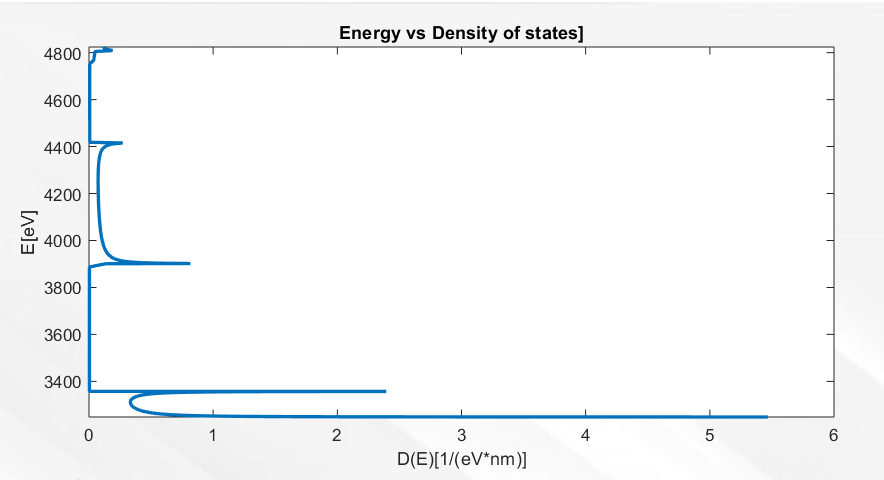


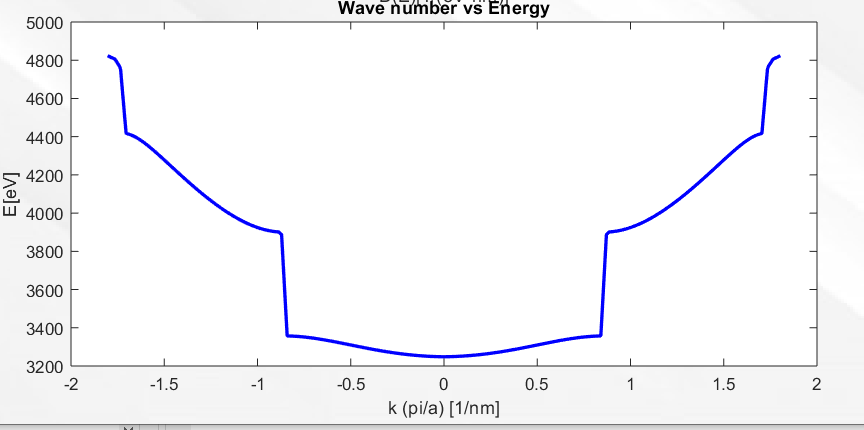
****

**Output Graphs:**

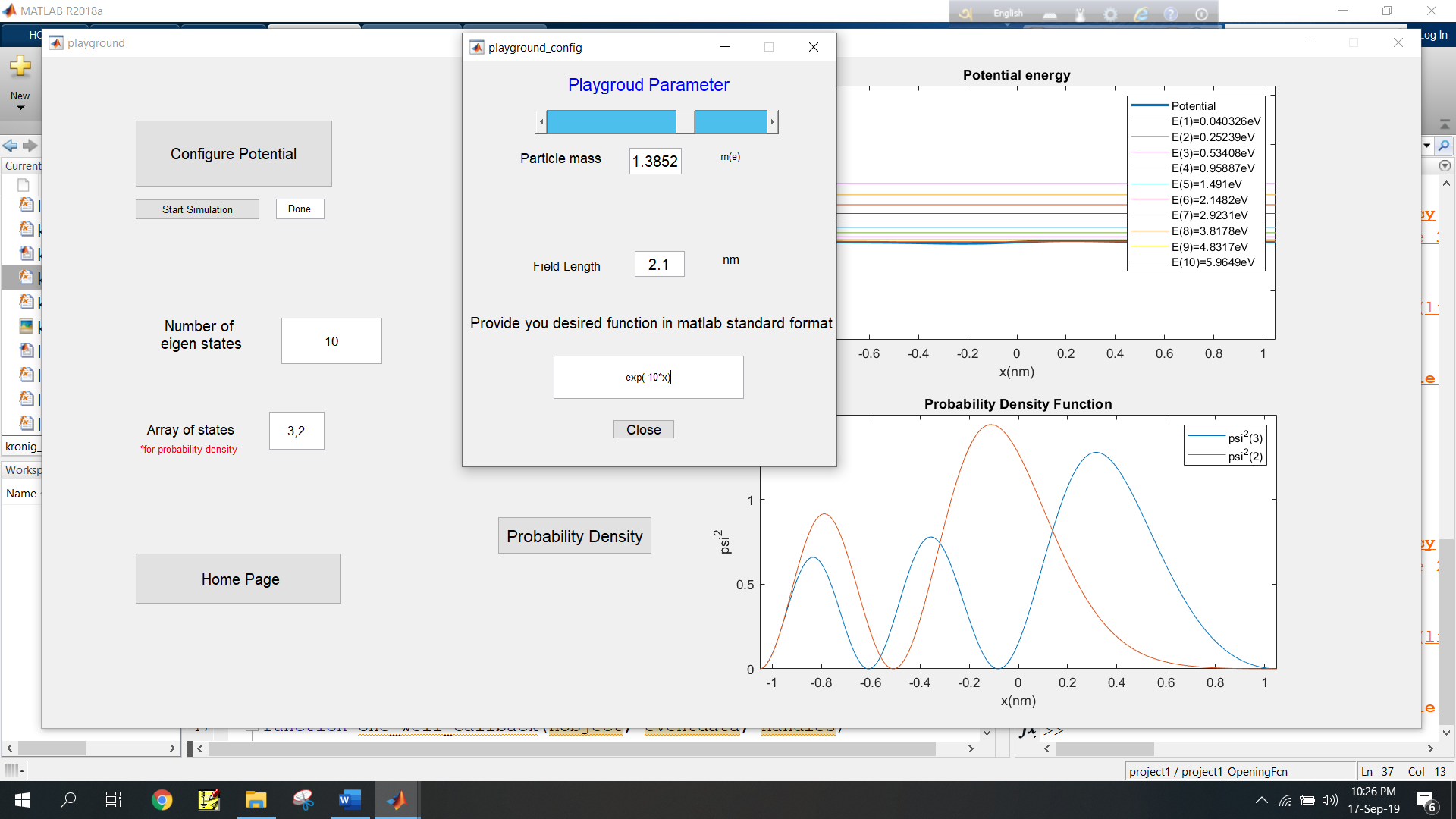
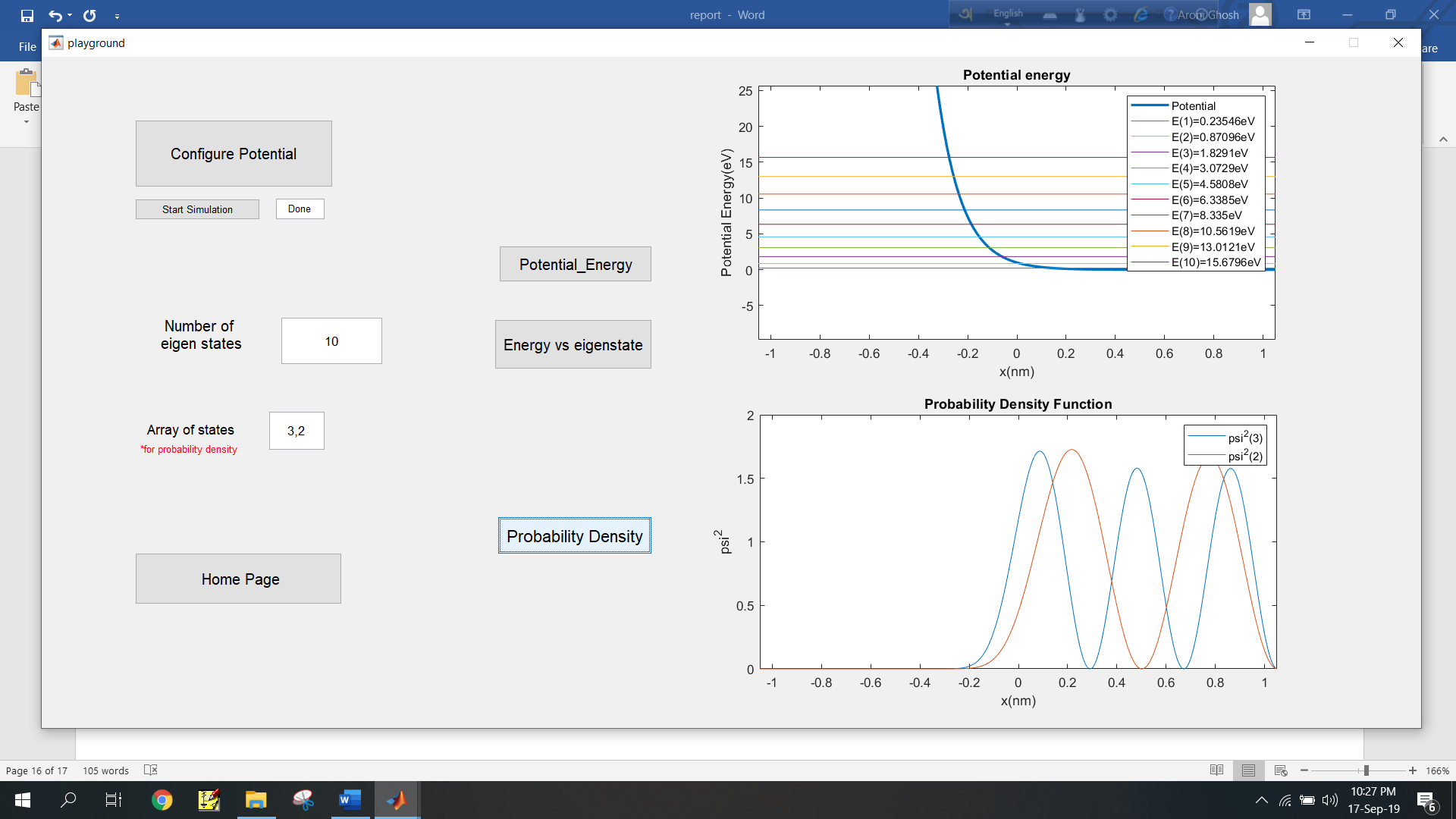
****

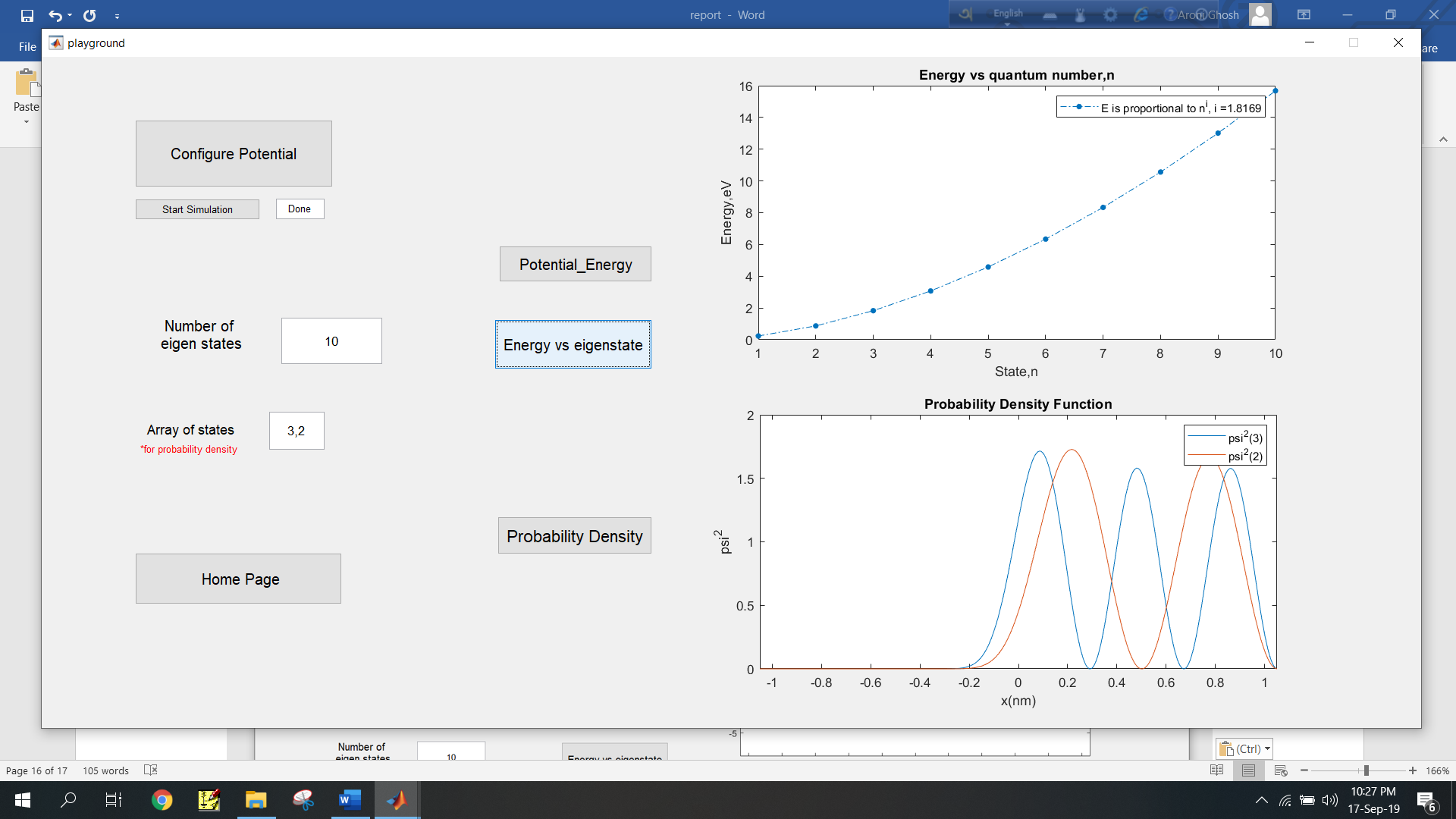


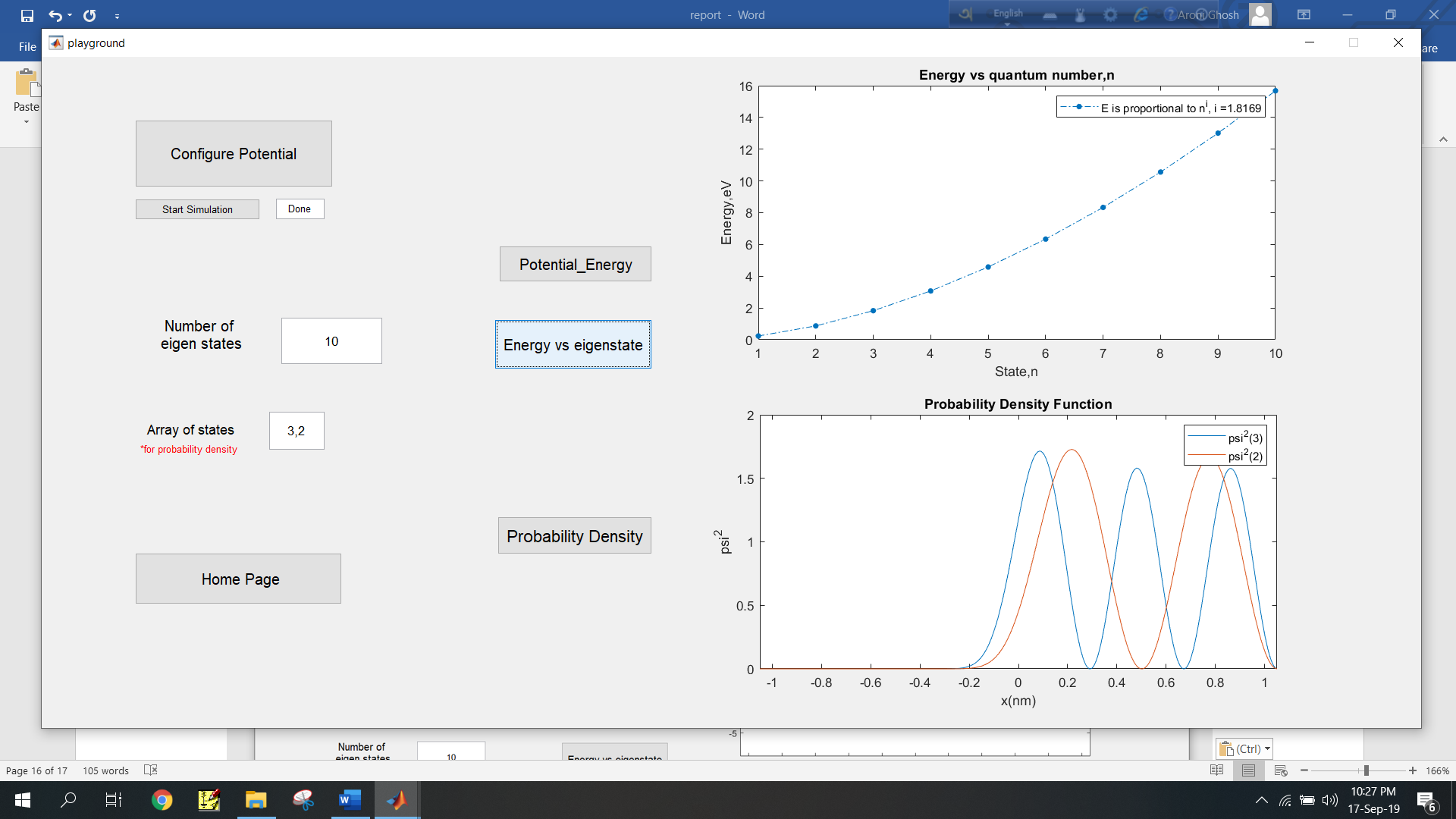




**Playground**







### Reference

[1] Tang Dongjiao, Dr. Yeo Ye, Mr. Andreas Dewanto, “*Generalized Matrix Numerov Solutions to the Schr¨odinger Equation”* , National University of Singapore, Faculty of Science ,Department of Physics, April 21, 2014