MM3110 - Expt 4 - Differential Equations

Vishal S
MM14B048

Kapil Chandra
MM14B022

1 Question 1

The differential equation we have is,

$$\frac{dy}{dx} = yx^3 - 1.5y, \tag{1}$$

It is given that $y(0)$ is 1 and $y(2)$ has to be found out. This can be calculated analytically as shown below,

$$\frac{dy}{dx} = yx^3 - 1.5y,$$
$$\frac{dy}{dx} = y(x^3 - 1.5),$$
$$\frac{dy}{y} = (x^3 - 1.5)dx,$$

Applying appropriate limits,

$$\int_1^{y(2)} \frac{dy}{y} = \int_0^2 (x^3 - 1.5)\, dx$$
$$ln(\frac{y(2)}{1}) = 1,$$
$$y(2) = e^1 = 2.7182s$$

Equation 1,is solved numerically using various techniques as shown in the code below,

```
clear all
clc
clf

func = @(x,y) y*x^3 - 1.5*y;
x0 = 0;
y0 = 1;
h = [0.01,0.05,0.1,0.2,0.3,0.4,0.5];
size_h = 7;
y2 = 2.718281;
analytic = zeros(size_h,1);
euler = zeros(size_h,1);
error_euler = zeros(size_h,1);
heun = zeros(size_h,1);
error_heun = zeros(size_h,1);
rk = zeros(size_h,1);
error_rk = zeros(size_h,1);
for i = 1:size_h
    analytic(i) = y2 ;
end
%Euler method
for i = 1:size_h
    y1 = y0;
    x1 = x0;
    while (x1<=2)
        y1 = y1 + h(i)*func(x1,y1);
        x1 = x1 + h(i);
    end
    euler(i) = y1;
    error_euler(i) = abs((y1 - y2)*100 /y2);
end

```

```

%Heun's method
for i = 1:size_h
    y1 = y0;
    x1 = x0;
    while (x1<=2)
        y10 = y1 + h(i)*func(x1,y1);
        y1 = y1 + 0.5*h(i)*(func(x1,y1) + func(x1,y10));
        x1 = x1 + h(i);
    end
    heun(i) = y1;
    error_heun(i) = abs((y1 - y2)*100 /y2);
end

%Runge-Kutta
for i = 1:size_h
    y1 = y0;
    x1 = x0;
    while (x1<=2)
        k1 = h(i)*func(x1,y1);
        k2 = h(i)*func(x1 + 0.5*h(i),y1 + 0.5*k1);
        k3 = h(i)*func(x1 + 0.5*h(i),y1 + 0.5*k2);
        k4 = h(i)*func(x1 + h(i),y1 + k3);
        y1 = y1 + 1/6*(k1 + 2*k2 + 2*k3 + k4);
        x1 = x1 + h(i);
    end
    rk(i) = y1;
    error_rk(i) = abs((y1 - y2)*100 /y2);
end
hold on;

loglog (h,error_euler,'-o','linewidth',4.0,'DisplayName','Euler');
loglog (h,error_heun,'-s','linewidth',4.0,'DisplayName','Heun');
loglog (h,error_rk,'-x','linewidth',4.0,'DisplayName','RungaKutta');
xlabel('Step size');
ylabel('Relative error (in %)');
ax = gca ;
set(ax, 'linewidth',2.0);
grid on;
legend('show');
```

The output of the above code for `h = 0.01` is as shown below

```
Using Euler Method
y(2) = 2.4804
Using Heun Method
y(2) = 2.6103
Using Rk method
y(2) = 2.7183
```

The relative error is plotted for these methods in the fig 1.

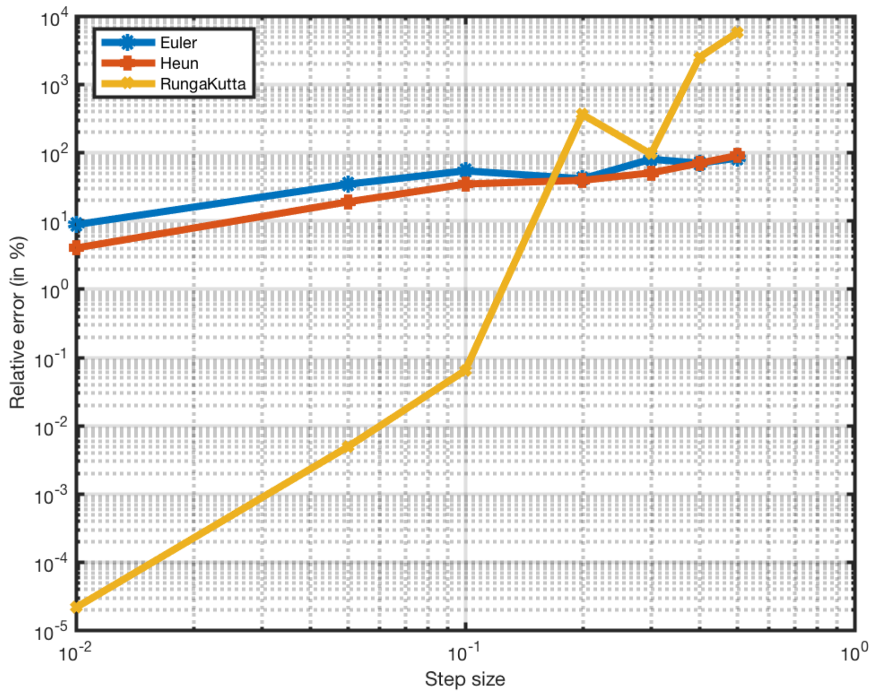

Figure 1: Relative error vs step size for various techniques

2 Question 2

In this question we have to solve a steady state heat balance equation using **shooting method**. The given differential equation to solve is

$$\frac{d^2T}{dx^2} = 0.15T \tag{2}$$

with the boundary conditions $T(0) = 240$ and $T(10) = 150$. In **shooting method** we convert the boundary value problem into two initial value problems and then solve them simultaneously.

$$\frac{dT}{dx} = Z \tag{3}$$

$$\frac{dZ}{dx} = 0.15T \tag{4}$$

Now the task is to solve the above equations separately and simultaneously. To solve the first equation we take $T(0) = 240$ as given in the question. To solve the second equation we need an initial value for Z but we don't have one, so we guess a value initially and compute $T(10)$. The goal is to find a good guess for $Z(0)$ such that we get get $T(10)$ close to 150. We have to try a few guesses before we get the solution hence the name **shooting method**.

```
clear all
clf
clc
h=1;

length = 10;
x = 0:h:length
size = length/h + 1;
T = zeros(size,1);
T(1) = 240;
Z0 = -88.6 %Our guess at the first derivative
for i = 2:size
    T(i) = T(i-1) + Z0*h;
```

```
    Z1 = Z0 + 0.15*T(i-1)*h;
    Z0 =Z1;
end
T(size)
plot (x,T,'LineWidth',4);
hold on;
title ("Temperature Profile")   ;
xlabel("Distance (in m)");
ylabel("Temperature");
box  on ;
grid on ;
ax = gca ;
set(ax, 'linewidth',2.0);
axis('square');
```

Below is a table which contains the initial guesses I made and the value of $T(10)$ obtained.

Guessed Z value	$T(10)$ value
-90	102.3801
-89	136.4602
-88	170.5402
-88.5	153.5002
-88.4	156.9082
-88.6	150.0922

So for $Z = -88.6$ we get a close value to $T(10)$. The temperature profile obtained is plotted in figure 2 and compared with analytical solution available.

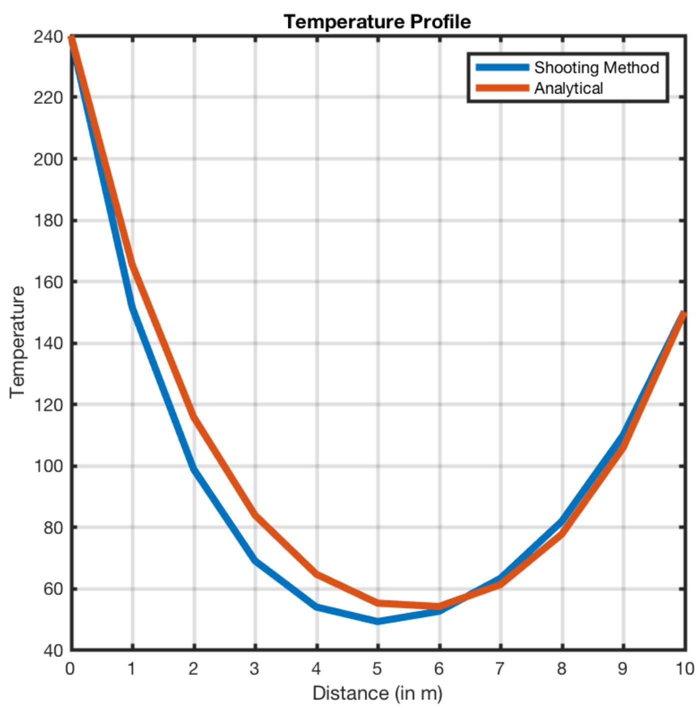

Figure 2: Temperature profile obtained using shooting method and compared with analytical solution

3 Question 3

In this question we have to find the steady state temperature profile of a square plate with a side length of $12cm$. We use Fick's law and discretize it to get the equation whose solution is the temperature profile. The derivation of the equation is as follows:

$$\nabla^2 T = 0 \tag{5}$$

$$\frac{d^2T}{dx^2} + \frac{d^2T}{dy^2} = 0 \tag{6}$$

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{(\Delta x)^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{(\Delta y)^2} = 0 \tag{7}$$

$$T_{i,j} = \frac{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}}{4} \tag{8}$$

We will be using equation 8 to update temperature at each point. Before we start calculating temperatures we have to define the boundary conditions. In the question we have three edges with 100 ^oC and one edge with 0 ^oC. Code for the implementation is given below.

```
%Initializing parameters and boundary conditions
size = 4;
T = zeros(size,size);
T(1,:) = 100;
T(size,:) = 100;
T(:,1) = 100;
T(:,size) = 0;
T_new = T;
error = 10;

%Calculation loop
while (error > 0.0001)
    for i = 2:size-1
        for j =2:size-1
            T_new(i,j) = (T(i+1,j) + T(i-1,j) +T(i,j+1) +T(i,j-1))/4.0;
        end
    end
    diff = abs(T-T_new);
    error = max(diff(:));
    T = T_new;
end
```

The plot obtained is given below.The right boundary is maintained at 0 ^oC and the remaining boundaries are maintained at 100 ^oC. The code is run at two different values of `size`. The temperature distribution obtained from the two runs are given below. Further temperature distribution obtained for the two runs is plotted along `x =6` in figure 5

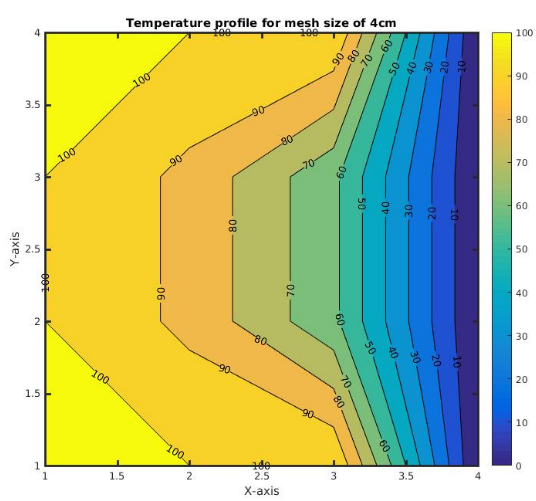

Figure 3: Temperature profile for mesh size = 4cm

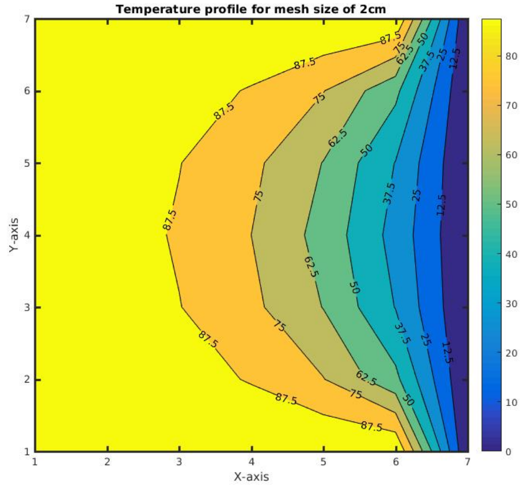

Figure 4: Temperature profile for mesh size = 2cm

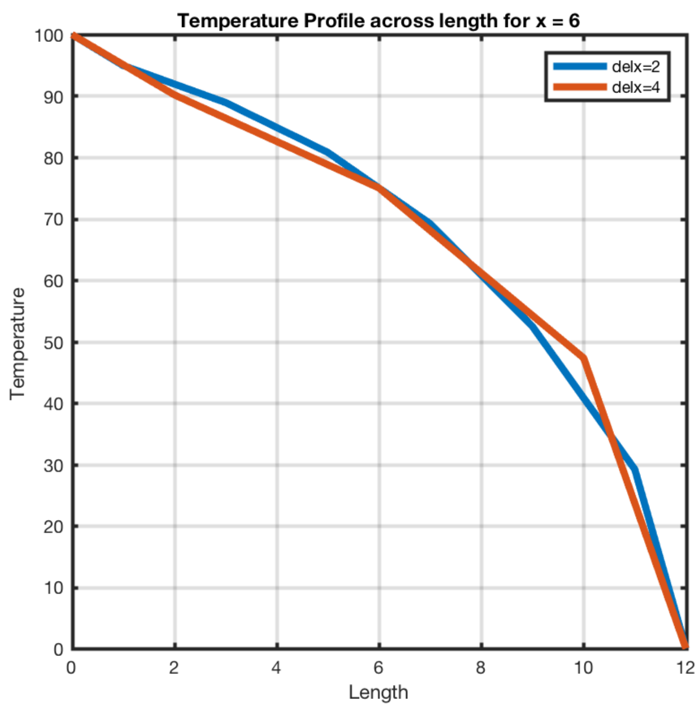

Figure 5: Temperature profile along x = 6

The temperature profile for `x =6` for different `size` values seem to be consistent with each other. Their differences are within the limits of numerical error.

4 Question 4

The equation to be solved is

$$\frac{\partial T}{\partial t} = \alpha \nabla^2 T, \tag{9}$$

The initial condition and the boundary condition are known to us. Thus a MATLAB code can to written to solve it.

```
clear all
clc
clf

h= 0.01;
x= 0:h:1;
n = 1/h + 1;
r = 1;
k = r*h^2;
t_now = zeros(n,1);
t0 = zeros(n,1);
ae = zeros(n,1);
aw = zeros(n,1);
ap = zeros(n,1);
su = zeros(n,1);
for i = 1:n
    t0(i) = sin(pi*x(i));
end



t_before =t0;
plot (x,t_before,'LineWidth',4,'DisplayName','Inital Profile');
hold on ;
title ("Temperature Profile across length")   ;
ylabel("Temperature ");
xlabel("Length");
box  on ;
grid on ;
for time = 0:k:0.2
    ap(1) = 1.0;
    su(1) = 0.0;
    ae(1) = 0.0;
    aw(1) = 0.0;
    for i = 2:n-1
        ap(i) = 2+2*r;
        ae(i) = -r;
        aw(i) = -r;
        su(i) = (2-2*r)*t_before(i) + r*(t_before(i-1) + t_before(i+1));
    end
    ap(n) = 1.0;
    su(n) = 0.0;
    ae(n) = 0.0;
    aw(n) = 0.0;
    for i = 1:n
        if(ap(i) == 0)
            disp ("Error ap becomes zero");
        end
    end
    ae(1) = ae(1)/ap(1);
    su(1) = su(1)/ap(1);
    ap(1) = 1.0;
    for i = 2:n
        ap(i) = ap(i) - ae(i-1)*aw(i)/ap(i-1) ;
        su(i) = su(i) - su(i-1)*aw(i)/ap(i-1) ;
        aw(i) = 0;
        ae(i) = ae(i) / ap(i);
        aw(i) = aw(i) / ap(i);
        su(i) = su(i) / ap(i);
        ap(i) = 1.0;
    end
    t_now(n) = su(n)/ap(n);
    for i = 2 : n
        j = n-i + 1;
        t_now(j) = (su(j) - ae(j)*t_now(j+1))/(ap(i)) ;
    end
    t_before = t_now;

```

```
end
% For plotting properly
plot (x,t_now,'LineWidth',4,'DisplayName','Temperature Profile at t=0.2');
hold on ;
title ("Temperature Profile across length")    ;
ylabel("Temperature ");
xlabel("Length");
box  on ;
grid on ;
legend('show')
ax = gca ;
set(ax, 'linewidth',2.0);
axis('square');
```

The temperature porfile obtained is plotted in figure 6.

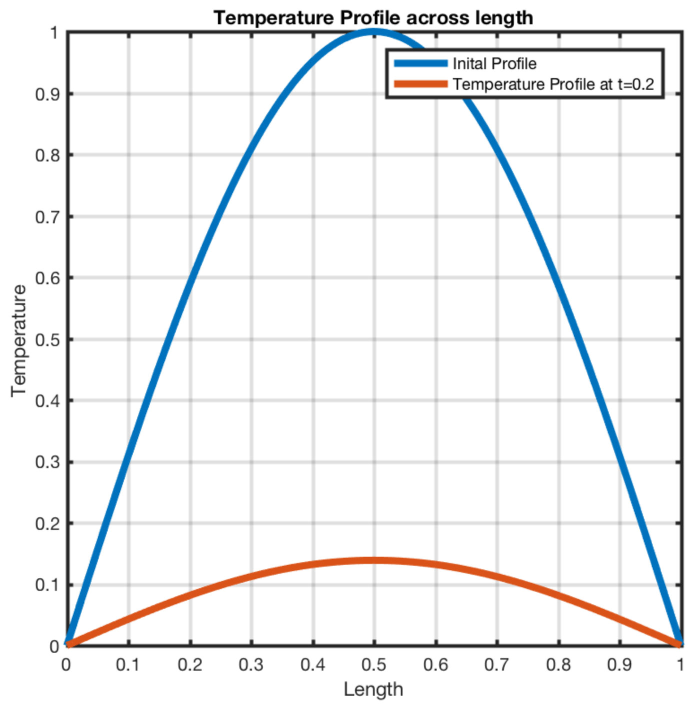

Figure 6: Temperature profile

5 Question 5

1D time-independent Schrödinger equation takes the form,

$$-\frac{\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2}+V(x)\psi=E\psi, \tag{10}$$

Where, the symbols take the usual meanings. Since, we are solving for a particle in a box, V(x) is taken to be 0.

The usual way of solving differential equations by discretising it and applying the appropriate boundary conditions won't help here as $\psi(x)=0$ is a trivial solution to this equation and hence $\psi(x)$ takes this solution which is not of interest. To overcome this, we can treat the Hamiltonian operator as a matrix operation.

Thus equation 10 can be re-written as ,

$$\begin{bmatrix} \frac{\hbar^2}{m(\Delta x)^2} & -\frac{\hbar^2}{2m(\Delta x)^2} & 0 & 0 & 0 \\ -\frac{\hbar^2}{2m(\Delta x)^2} & \ddots & \ddots & & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & & \ddots & \ddots & -\frac{\hbar^2}{2m(\Delta x)^2} \\ 0 & 0 & 0 & -\frac{\hbar^2}{2m(\Delta x)^2} & \frac{\hbar^2}{m(\Delta x)^2} \end{bmatrix} \begin{pmatrix} \psi(x(1)) \\ \psi(x(2)) \\ \vdots \\ \vdots \\ \vdots \\ \psi(x(N)) \end{pmatrix} = E \begin{pmatrix} \psi(x(1)) \\ \psi(x(2)) \\ \vdots \\ \vdots \\ \vdots \\ \psi(x(N)) \end{pmatrix} \tag{11}$$

Now, instead of solving for $\psi(x)$, we can just find the eigen values of the Hamiltonian matrix, they would be the energy states of the paritcle and their correspondinf eigen vectors would be the waveform of $\psi(x)$ The code written to find the energy states of the particle.

```
clear all
clc
clf

a = 0.5e-09 ; % in nm
h = 6.626e-34;
m = 9.1e-31;
hbar = h/(2*3.1415);
N = 1000; % Excluding the boundary points
del = a/(N+1);
H = zeros(N,N);
%Populating the Hamiltonian Matrix
value = hbar*hbar/((del*del)*m);
for i = 2:N-1
    H(i,i) = value;
    H(i,i+1) = -0.5*value;
    H(i,i-1) = -0.5*value;
end
H(1,1) = value;
H(1,2) = -0.5*value;
H(N,N) = value;
H(N,N-1) = -0.5*value;
%Finding the eigen values and eigen vectors
[e,d,w] = eig(H);
e = eig(H);
%Plotting
no_of_levels = 10;
n = 1:no_of_levels;
y = 1:no_of_levels;
energy = zeros(10,1);
factor = (h*h)/(8*m*a*a);
for i = 1:no_of_levels
    y(i) = n(i)*n(i) * factor/(1.6e-19);
    energy(i) = e(i)/(1.6e-19);
end
plot (n,energy,'ro','LineWidth',4);
hold on;
plot (n,y,'LineWidth',4);
title ("Energy for the first 10 levels")    ;
ylabel("Energy (in eV)");
xlabel("Levels");
legend('Numerical','Analytical')
box on ;
grid on ;
ax = gca ;
set(ax, 'linewidth',2.0);
axis('square');

hold off;

clf;

```

```
%Plotting Eigen Vectors
x= zeros(N+2,1);
wav = zeros(N+2,1);
for i = 2:N+2
    x(i) = x(i-1) + del;
end
Wave = zeros(N+2,no_of_levels);
for j = 1:no_of_levels
    for i = 2:N+1
        Wave(i,j) = w(i-1,j);
    end
end
for j = 1:5
    for i = 1:N+2
        wav(i) = Wave(i,j) +  (j-1)*0.5;
    end
     str = ['n=',num2str(j)];
    plot (x,wav,'LineWidth',4,'DisplayName',str);
    hold on;
    title ("Form of Psi for the first 5 levels")    ;
    xlabel("Distance (in m");
    box  on ;
    grid on ;
    ax = gca ;
    set(ax, 'linewidth',2.0);
    axis('square');
end
legend('show');
legend('location','eastoutside');
```

The first 10 energy states are plotted and compared with the analytically derived solution in fig 7

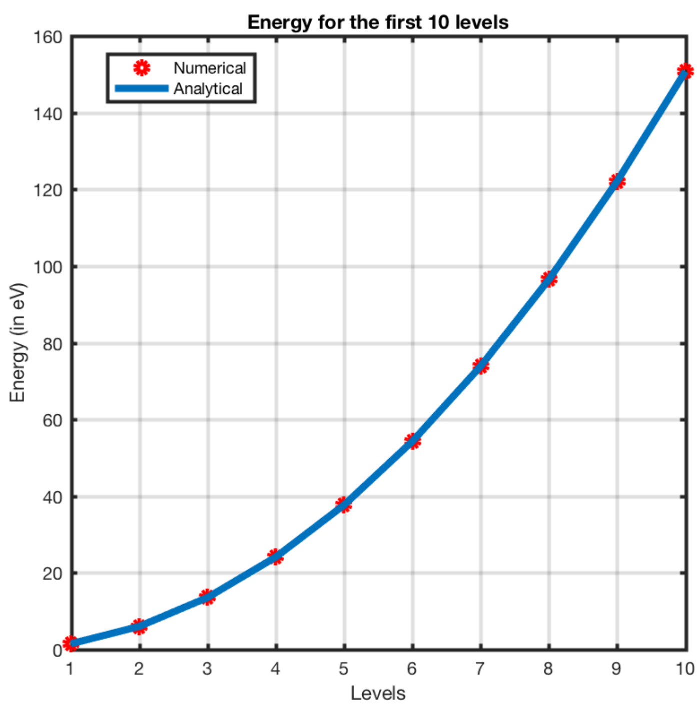

Figure 7: Comparison between energy states calculated numerically and analytically

The eigen vectors are also plotted in fig 8 for the first 5 energy states and they are just *sine* functions with no node for first energy state, one node for second energy state and so on.

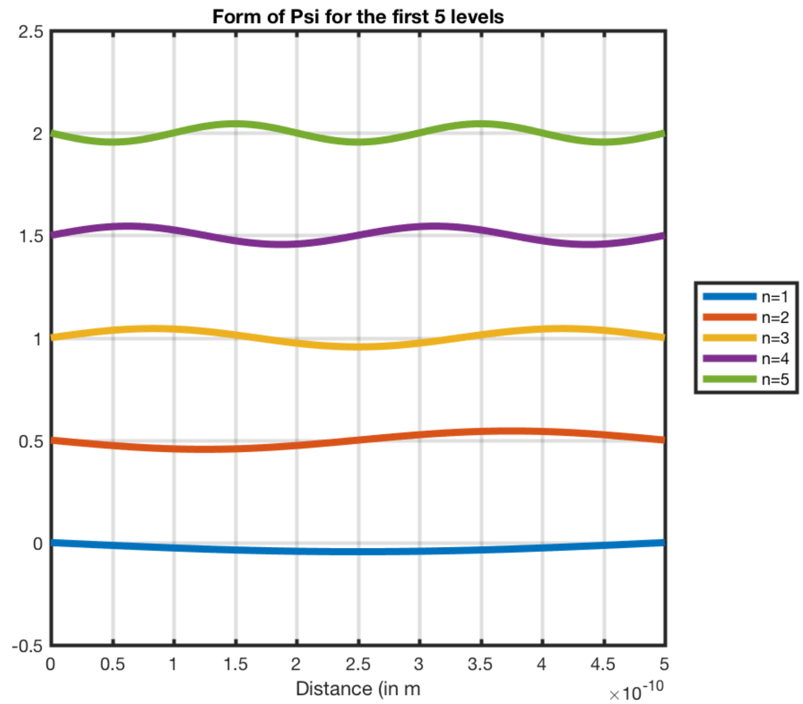

Figure 8: Eigen-vectors for first 5 energy states. The eigen-vectors are shifted above by $0.5 \times n$

The ground state of the system is calculated to be `1.5078 eV`.