SUBJECT CODE - EP201

INTRODUCTION TO COMPUTING

PRACTICAL FILE

SUBMITTED BY: ADITYA SINGH

ROLL. NO: 2K19/EP/005

BRANCH/SEM: EP/3rd Sem

SUBMITTED TO: DR. AJEET KUMAR

LIST OF EXPERIMENTS

1. Basics of Matrix operation and Matrix manipulation
2. Write a Matlab program for very famous Blackbody radiation and verify Wein's displacement law.
3. Write a Matlab program to show the binding energy/mass number with mass number to find the most stable state.
4. Write a Matlab program to calculate the values of inbuilt defined trigonometric functions using series solution approach. Compare the results with inbuilt functions.
5. Write a Matlab program to study the behavior of Gaussian function using all appropriate inbuilt 2d and 3d plotting commands.
6. Write a Matlab program to find out the unknown coefficients by Polynomial fitting.
7. Write a Matlab program to solve the second order differential equation of the pendulum problem.
8. Write Matlab code to plot the intensity distribution of Single-slit, double slit and N-slit all together. Analyze the result. Show how young's double slit experiment is different from the double slit diffraction.
9. Write a Matlab program to find out the roots of a given equation using the bisection method. Compare the results using Matlab inbuilt functions.
10. Write Matlab code to show the propagation of a group wave as a function of time.

CONTENTS

Experiment 1: Basics of Matrix operation and Matrix manipulation.

Code/Commands:

```
A = [1 2 8; 5 6 1; 4 3 7];    B = [5 6 2; 8 2 3; 1 5 9];
a = A+B          e = A*B          i = A/B          m = inv(A)
b = B+A          f = B*A          j = B/A          n = A\B
c = A-B          g = A.*B         k = A'
d = B-A          h = B.*A         l = diag(A)

%RESHAPE
A = [1 2 8; 5 6 1]
o = reshape(A,3,2)          s = flip(A)
p = rot90(A)                t = ctranspose(A)
q = fliplr(A)                   u = tril(A)
r = flipud(A)               v = triu(A)

%CONCATENATION
C = [B B; B+4 B-1]
C(:,2) = [ ]
w = max(B)
whos B
```

OUTPUT:

```
Command Window

a =

     6     8    10
    13     8     4
     5     8    16


b =

     6     8    10
    13     8     4
     5     8    16


c =

    -4    -4     6
    -3     4    -2
     3    -2    -2


d =

     4     4    -6
     3    -4     2
    -3     2     2
```

```
e =

    29    50    80
    74    47    37
    51    65    80


f =

    43    52    60
    30    37    87
    62    59    76


g =

     5    12    16
    40    12     3
     4    15    63


h =

     5    12    16
    40    12     3
     4    15    63
```

```
i =

   -0.5232    0.3406    0.8916
    1.1176   -0.0588   -0.1176
   -0.2198    0.5573    0.6409


j =

    0.0947    0.9474    0.0421
   -2.3474   -0.4737    3.1789
    2.0737    0.7368   -1.1895


k =

     1     5     4
     2     6     3
     8     1     7


l =

     1
     6
     7
```

```
m =

   -0.4105   -0.1053    0.4842
    0.3263    0.2632   -0.4105
    0.0947   -0.0526    0.0421


n =

   -2.4105   -0.2526    3.2211
    3.3263    0.4316   -2.2526
    0.0947    0.6737    0.4105
```

Command Window

```
o =

     1     6
     5     8
     2     1


p =

     8     1
     2     6
     1     5


q =

     8     2     1
     1     6     5


r =

     5     6     1
     1     2     8
```

```
s =

     5     6     1
     1     2     8


t =

     1     5
     2     6
     8     1


u =

     1     0     0
     5     6     0


v =

     1     2     8
     0     6     1
```

Command Window

```
C =

     1     2     1     2
     3     4     3     4
     5     6     0     1
     7     8     2     3


C =

     1     1     2
     3     3     4
     5     0     1
     7     2     3


w =

     3     4

  Name      Size            Bytes  Class     Attributes

  B         2x2                32  double
```

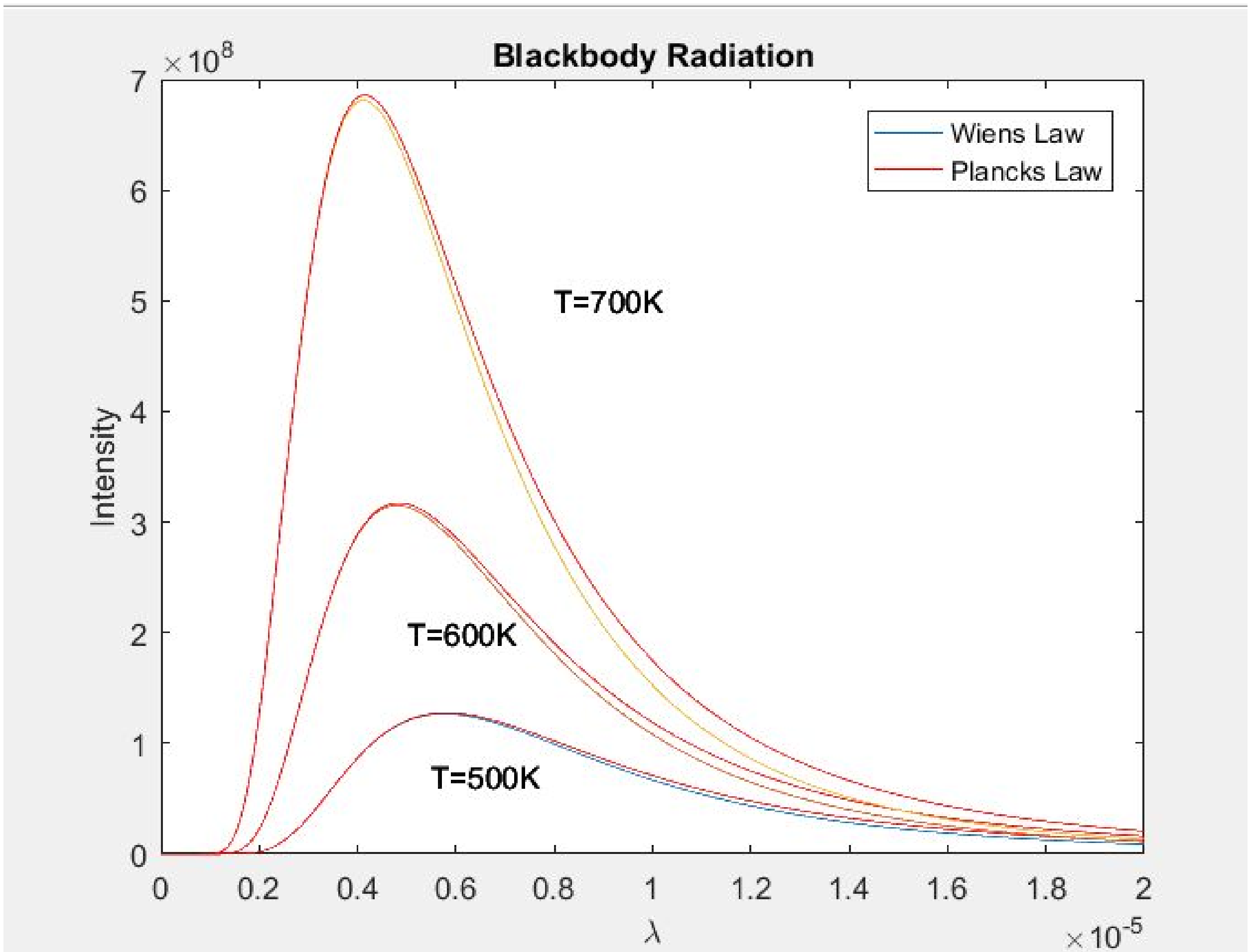
Experiment 2: Write Matlab program for very famous Blackbody radiation and verify Wein’s displacement law.

Code/Commands:

```
c=3*10^8;                  % speed of light in vacuum

h=6.625*10.^-34;           % Planck constant

k=1.38*10.^-23;            % Boltzmann constant

T=[ 500 600 700 ];         % Temperatures in Kelvin

Lam=(0.0:0.01:20).*1e-6;


for i=1:3

    % Wien's Displacement Law

    I1(:,i)= ((2*h*c*c)./(Lam.^5)).*(exp(-(h*c)./(Lam*k*T(i))));


    % Planck's Law

    I2(:,i)=(2*h*c*c)./((Lam.^5).*(exp((h.*c)./(k.*T(i).*Lam))-1));


    plot(Lam,I1(:,i))

    hold on

    plot(Lam,I2(:,i),'r')


    text(.55e-5,.7e8,'T=500K')

    text(.5e-5,2e8,'T=600K')

    text(.8e-5,5e8,'T=700K')


    legend('Wien's Law', 'Planck's Law')
```

```
    xlabel('\lambda')
    ylabel('Intensity')

    title('Blackbody Radiation')

end
```

OUTPUT:



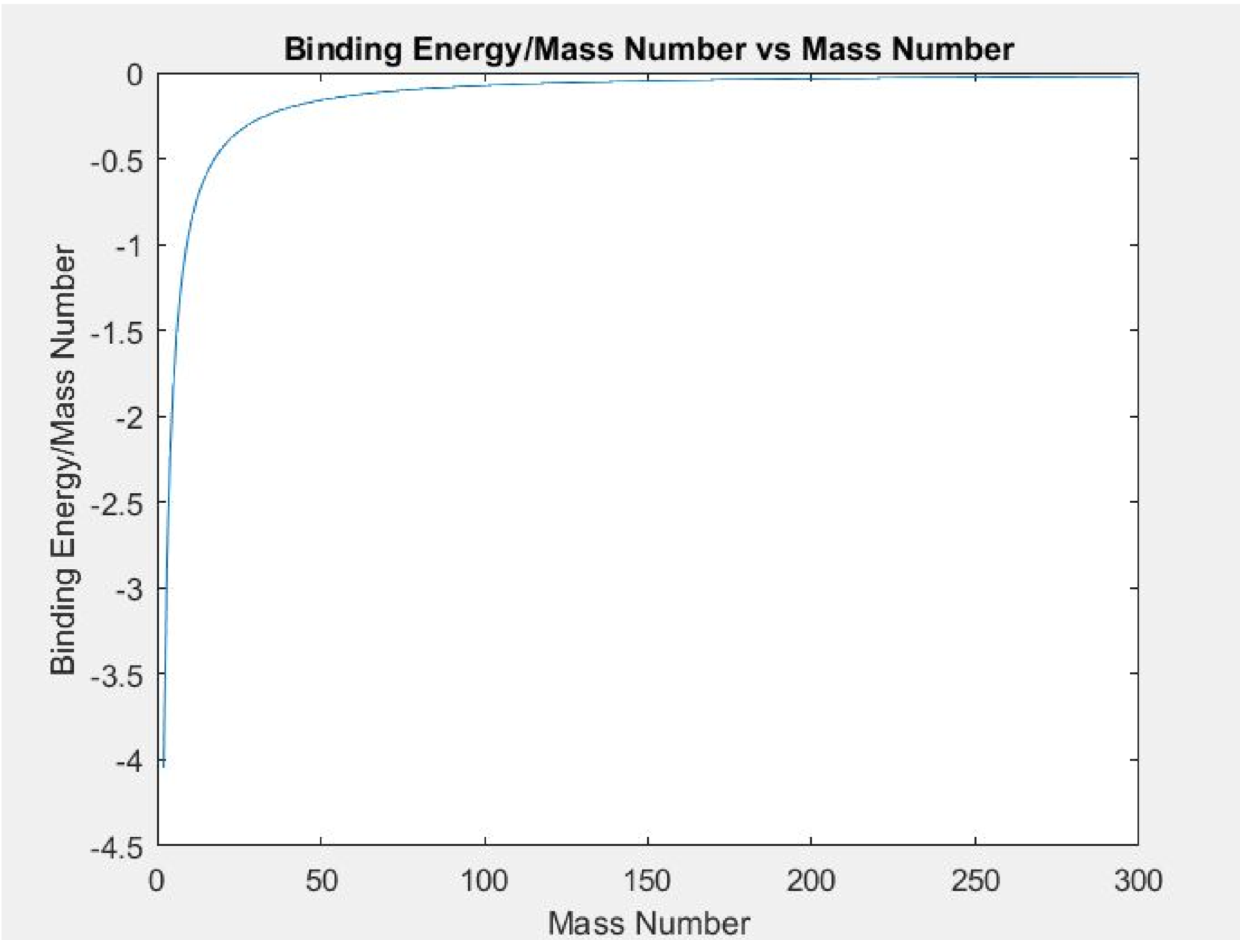
Experiment 3: Write Matlab program to show the binding energy/mass number with mass number to find the most stable state.

Code/Commands:

```
av = 14.1;      % Volume coefficient
as = 13.0;      % Surface energy coefficient
ac = 0.595;  % Coulomb energy coefficient
aa  = 19.0;     % Asymmetric energy coefficient
ap = 33.5;  A = 2:1:300;

Z = (1/2)*A./(1+A*(2/3)*(ac/4*aa));
N = A.*(-Z);

b = mod(Z,2);    % Check Even
c = mod(A,2);

if b==c
   if c==0
  BE=av-(as/A.^(1/3))-(ac*Z.^2./A.^(4/3))-(aa*((A-(2.*Z)).^2)./A.^2)+(ap./A.^(7/4));
   elseif b==1
BE=av-(as./A.^(1/3))-(ac*Z.^2./A.^(4/3))-(aa*((A-(2.*Z)).^2)./A.^2)-(ap./A.^(7/4));
   end
else
   BE=av-(as./A.^(1/3))-(ac*Z.^2./A.^(4/3))-(aa*((A-(2.*Z)).^2)./A.^2);
end

plot(A, BE./A)
```

```
xlabel('Mass Number');
ylabel('Binding Energy/Mass Number')
title('Binding Energy/Mass Number vs Mass Number')
```

OUTPUT:



Experiment 4: Write Matlab program to calculate the values of inbuilt defined trigonometric functions using series solution approach. Compare the results with inbuilt functions.

Code/Commands:

```
n = 100;

x = pi/3;

a = zeros(1,n);

b = zeros(1,n);

for i = 0:n

  a(i+1) = (-1)^i*x^(2*i+1)/factorial(2*i+1);

  b(i+1) = (-1)^i*x^(2*i)/factorial(2*i);

end

sinx_series = sum(a)

sinx_inbuilt = sin(x)

cosx_series = sum(b)

cosx_inbuilt = cos(x)
```

OUTPUT:

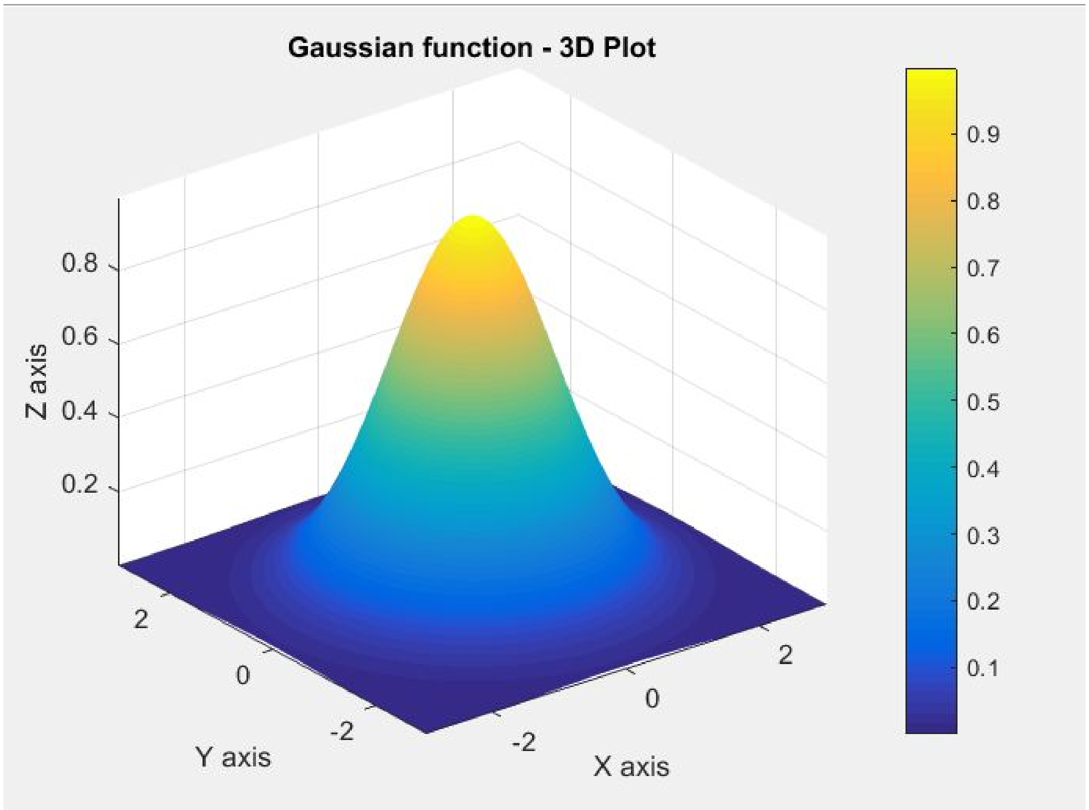
```
Command Window
sinx_series =

    0.8660


sinx_inbuilt =

    0.8660

cosx_series =

    0.5000


cosx_inbuilt =

    0.5000
```

Experiment 5: Write Matlab program to study the behavior of Gaussian function using all appropriate inbuilt 2d and 3d plotting commands.

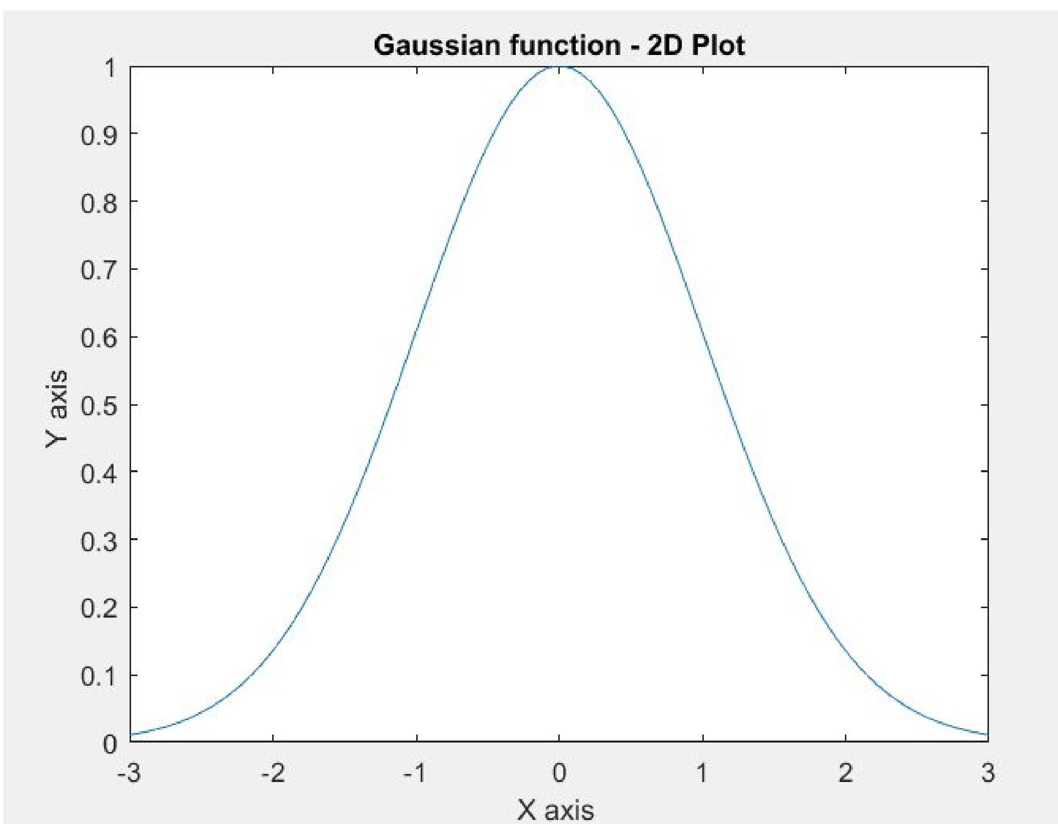
Code/Commands:

```
x=linspace(-3, 3,100);
y=x;
[X,Y]=meshgrid(x,y);
z=exp(-(X.^2/2)-(Y.^2/2));
y1=exp(-(x.^2/2));

figure;
surf(X,Y,z);
xlabel('X axis '), ylabel('Y axis '), zlabel('Z axis');
title('Gaussian function - 3D Plot');

colorbar
shading interp
axis tight

figure;
plot(x,y1);
xlabel('X axis '),ylabel('Y axis ');
title('Gaussian function - 2D Plot');
```

OUTPUT:



Gaussian function - 3D Plot
Z axis
0.8
0.6
0.4
0.2
2
0
-2
Y axis
2
0
-2
X axis
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1



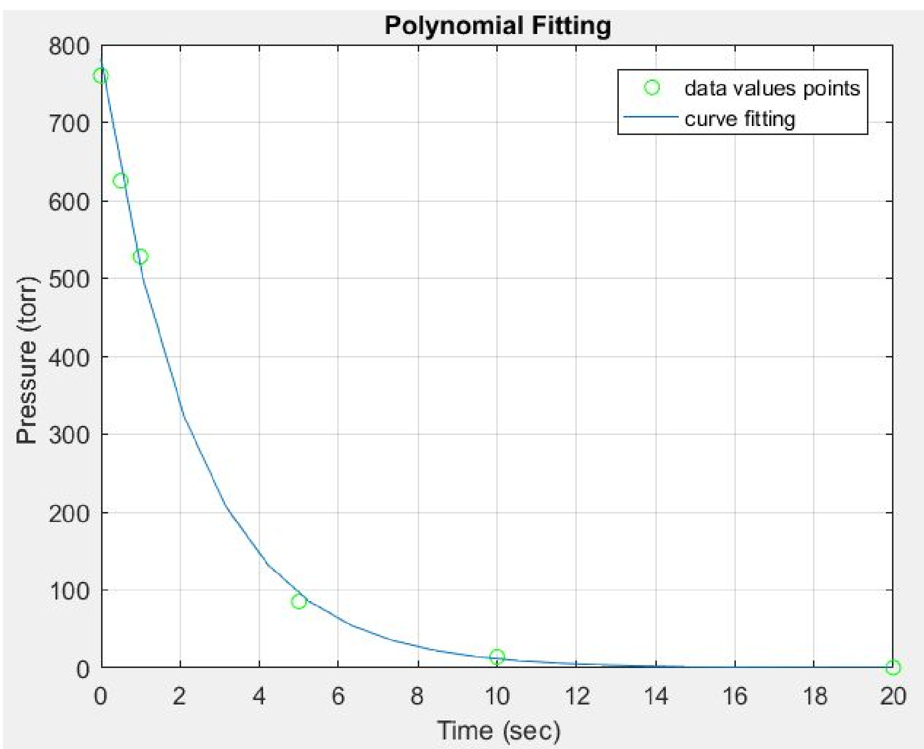
Gaussian function - 2D Plot
Y axis
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-3
-2
-1
0
1
2
3
X axis

Experiment 6: Write Matlab program to find out the unknown coefficients by Polynomial fitting.

Code/Commands:

```
t = [0 0.5 1.0 5.0 10.0 20.0]; p= [760 625 528 85 14 0.16];

tbar = t ; pbar = log(p);

a = polyfit(tbar,pbar,1);

tau = -1./(a(1)); p0 = exp(a(2));

disp(['Coefficients :  tau = ' num2str(tau) ' & p0 = ' num2str(p0)]);

tnew = linspace(0,20,20); pnew = p0 .*exp(-tnew./tau);

plot(t,p,'go',tnew,pnew),grid;

xlabel('Time (sec)'), ylabel('Pressure (torr) ');

title('Polynomial Fitting');

legend('data values points','curve fitting');
```

OUTPUT:



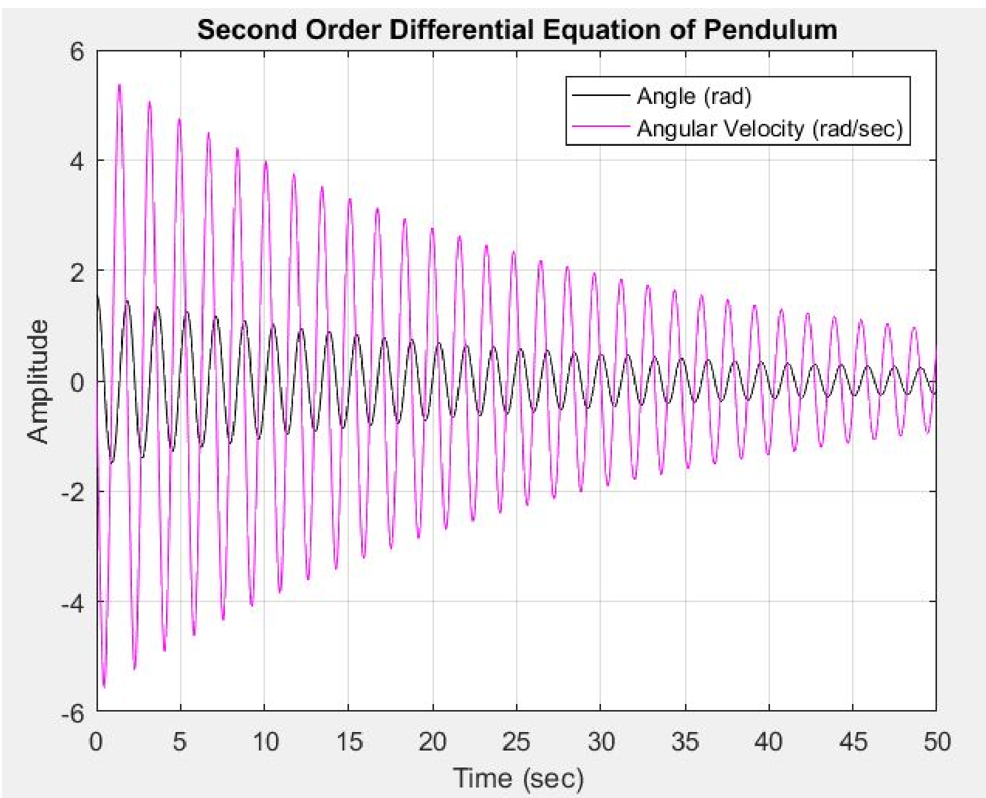

```
Coefficients :  tau = 2.3739 & p0 = 781.1432
```

Experiment 7: Write a Matlab program to solve the second order differential equation of the pendulum problem.

Code/Commands:

```
m=0.8; l=0.613; B=0.095; g=9.8;
pendulum = @(t,x) [x(2);-B/m*l.*x(2)-g/l.*sin(x(1))];
tspan = [0,50]; x0 = [pi/2,0];
[t,x] = ode45(pendulum,tspan,x0);
plot(t,x(:,1),'k');
grid on; hold on;
plot(t,x(:,2),'m');
xlabel('Time (sec)'); ylabel('Amplitude');
legend('Angle (rad)','Angular Velocity (rad/sec)');
title('Second Order Differential Equation of Pendulum');
```

OUTPUT:



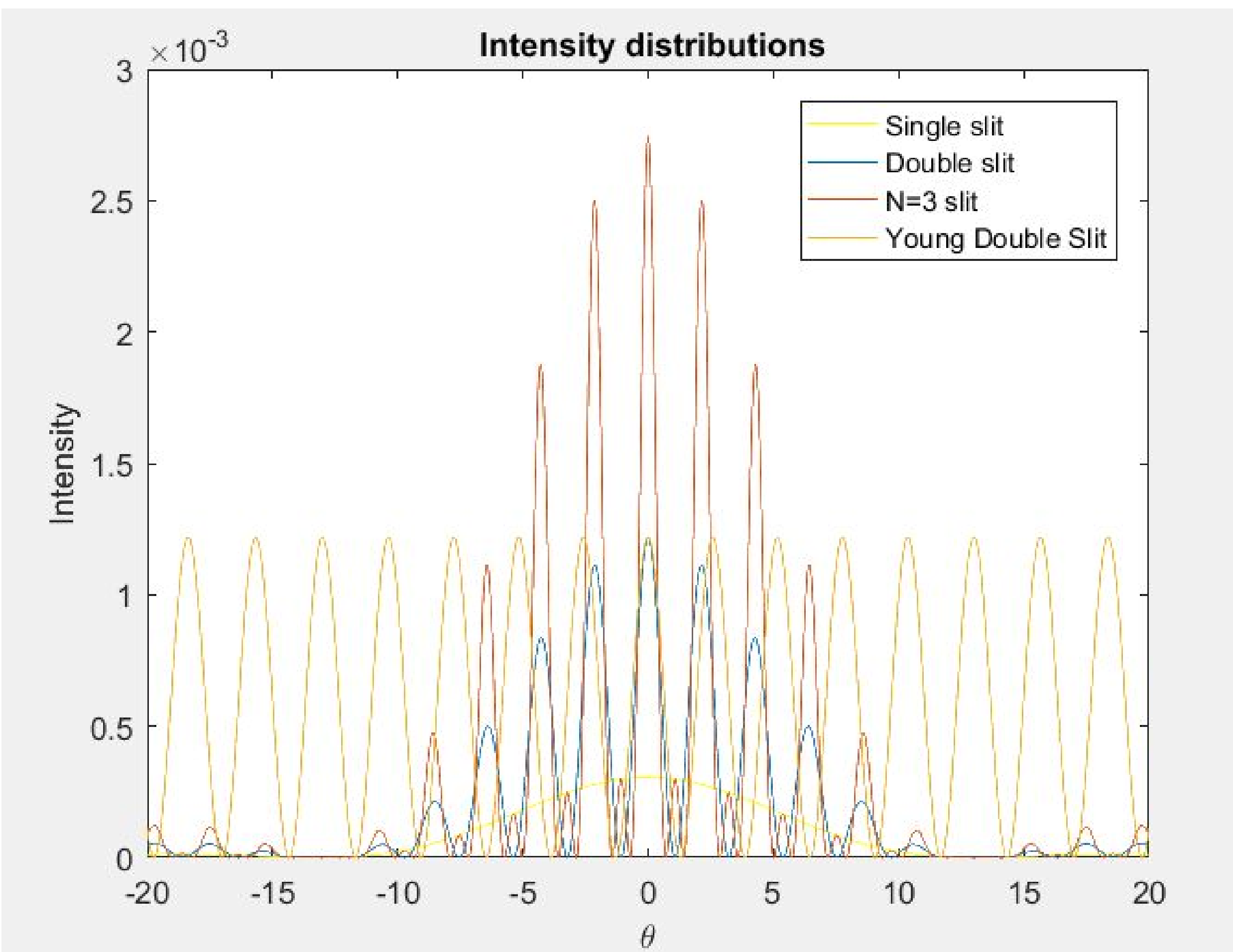
Experiment 8: Write Matlab code to plot the intensity distribution of Single-slit, double slit and N-slit all together. Analyze the result. Show how young's double slit experiment is different from the double slit diffraction.

Code/Commands:

```
e=0.14*(10^(-3)); Io=1; d=5*e;

lambda=5500*(10^(-10));

theta=-20:0.001:20;


% Single slit

a=(pi.*e.*sind(theta))./lambda;

Is=(Io.*(sind(a)).^2)./((a).^2);


% Double slit

beta=(pi*(e+d).*sind(theta))./lambda;

Id=(4*Io*(sind(a)).^2.*(cosd(beta)).^2)./((a).^2);


plot(theta,Is,'y'); hold on; plot(theta,Id)

xlabel('\theta'); ylabel('Intensity')


% N slit

N = 3; % for n=3

In=Io*(((sind(a)).*(sind(N*beta)))./(a.*sind(beta))).^2;

[In_max,I]=max(In); In_norm=In./In_max;

plot(theta,In);
```

```
% Young's Double Slit Exp
ey=e/1000; ay=(pi.*ey.*sind(theta))./lambda;
beta_y=(pi*(ey+d).*sind(theta))./lambda;
Id_y=(4*Io*(sind(ay)).^2.*(cosd(beta_y)).^2)./((ay).^2);
plot(theta,Id_y); hold off
legend('Single slit','Double slit','N=3 slit','Young Double Slit')
title(' Intensity distributions')
```

OUTPUT:



Experiment 9: Write a Matlab program to find out the roots of a given equation using bisection method. Compare the results using Matlab inbuilt functions.

Code/Commands:

```
f=@(x) x.^2-6;
x_lower=0; x_upper=5;
x_mid=(x_upper+x_lower)/2;
while abs(f(x_mid))>0.01;
   if (f(x_mid)*f(x_upper))<0;
      x_lower=x_mid;
   else
      x_upper=x_mid;
   end
   x_mid=(x_upper+x_lower)/2;
end
p=[1 0 -6]; r=roots(p);
fprintf(['Root obtained by Bisection Method: ',num2str(x_mid),'\n']);
fprintf('Roots obtained by Inbuilt Function are: \n');
disp(r);
```

OUTPUT:

Command Window

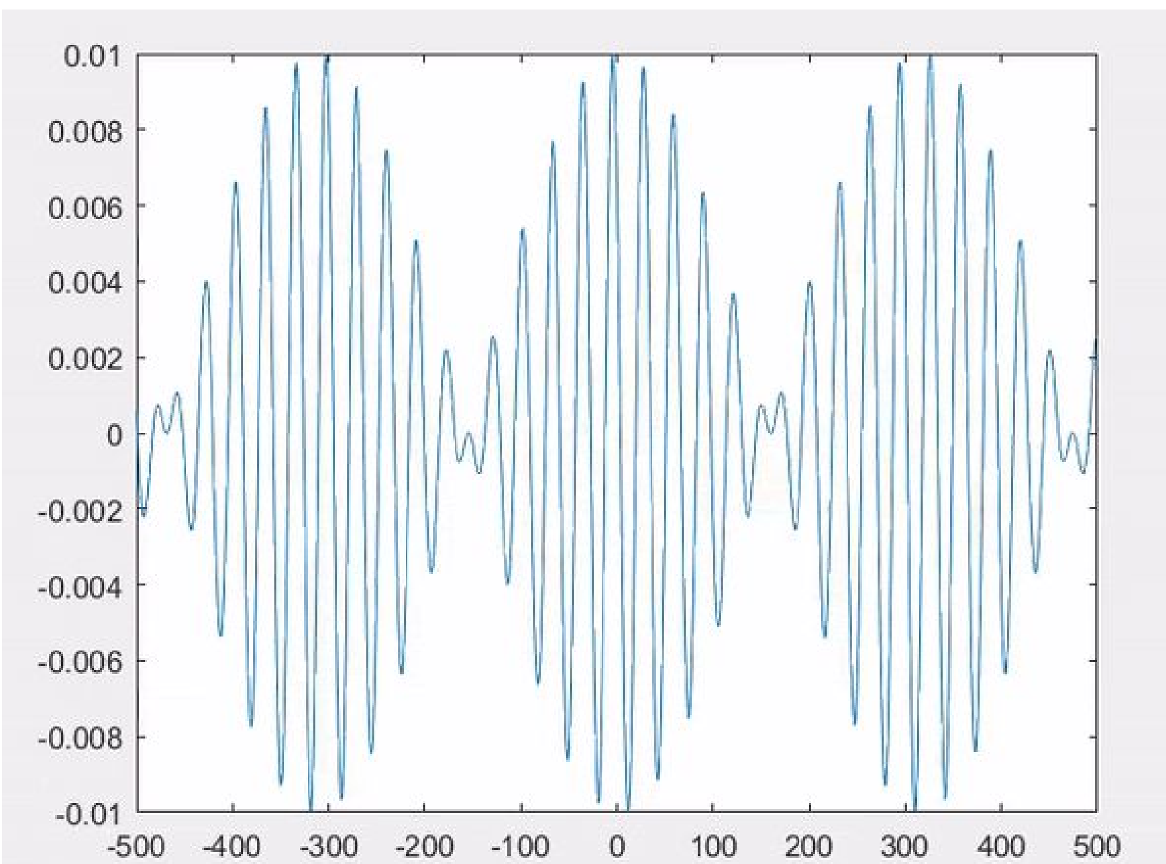
```
>> sample
Root obtained by Bisection Method: 2.4512
Roots obtained by Inbuilt Function are:
    2.4495
   -2.4495
```

Experiment 10: Write a Matlab code to show the propagation of group wave as a function of time.

Code/Commands:

```
x=-500:1:500;

a=0.01;

for t=0:1:50

    y=0.01.*cos((0.01.*x)-(0.002.*t)).*cos((0.2.*x)-(2.*t));

    plot(x,y)

    pause(0.10)

end
```

OUTPUT:



ANNEXURE

```
% CIRCLE OF RADIUS R
r=10; th = linspace(0,2*pi,100);
x =0; y=0;
cx = r.*cos(th) + x;
cy = r.*sin(th) + y;
plot(cx,cy); xlabel('x'); ylabel('y'); title('circle of r=10');

% RELATIONAL OPERATORS  < <= > >= ==
A = [1 2 3; 4 5 6; 7 8 9]; B = [4 5 6; 1 2 3; 7 8 9];
A == B          A<=B            A>B

% LOGICAL OPERATORS & | ~
C = [0 0 1 1]; D = [0 1 0 1];
C&D             C|D             ~C

% SORT FUNCTIONS
% sort(matrix, dimension)  dimension = 1(for column-wise), 2(for row-wise)
E = rand(3,5)*10       F = sort(E,1)
G = sort(E,2,'descend')  % for descending order

% sum of series 5x^2-2x not to exceed 1500
sum=0;i=1; count=0;
while sum<=1500
   sum = sum + 5*i^2-2*i;
   i=i+1;count=count+1;
end
count
```

hold on % retains previous graph and shows in next plot

subplot % multiple plots in same window

subplot(m,n,p) % divides the figure in mxn grid and create axes in the position specified by p

format short/ long/ e/ long g/ short g/ bank/ hex % default is - format short

ceil(x) % nearest greater

floor(x) % nearest smaller

fix(x) % rounds towards zero

[M,I] = max(A) % max element with index in every column.

```
% fibonacci series

n = input('enter num: '); fibo = [1,1];

for i=3:n

   fibo(i)=fibo(i-1)+fibo(i-2);

end
```

POLYNOMIALS - defined in MATLAB as a row vector, made up of coefficients. Dimension of row vector=n+1; n=degree of the polynomial.

p = [3 4 1]; % 3x^2 + 4x + 1

polyval(p,[5 7 4]) % It returns the value of a polynomial of degree n evaluated at x

CURVE FITTING - process of adjusting a mathematical function so that it lays as closely as possible to a set of data points. Curve fitting is based on " LEAST SQUARES TECHNIQUE". This technique minimizes the squared errors b/w the curve and set of measured data.

```
p = polyfit(x,y,n);

% p = vector of coefficients of polynomial that fits the data

% x = vector of horizontal coordinates of data points (independent)

% y = vector of vertical coordinates of data points (dependent)

% n = degree of polynomial
```

% ROOTS OF POLYNOMIAL

```
z = [4 -3 2]; % 4x^2-3x+2
r=roots(z)
```

ANONYMOUS FUNCTIONS - not stored in a program file, but associated with a variable whose data type is function_handle.

Syntax: sqr=@(x) x.^2 [f(x)=x^2].

@=operator that creates the handle, (x) is the function argument.

```
k=@(x) x.^2; k(4) % output = 16
```

INLINE FUNCTION - it creates a function of any number of variables by giving a string containing a function followed by a series of strings denoting the order of the input variable.

```
% x=inline('expression')
l=inline('x.^2','x');
l(5)
l([4 6])
```

ROOTS OF NONLINEAR FUNCTION

```
% Syntax: x=fzero(func,x0)
f=@(x) sin(x) - 0.5;
n=fzero(f,[0 pi/2])
```

END