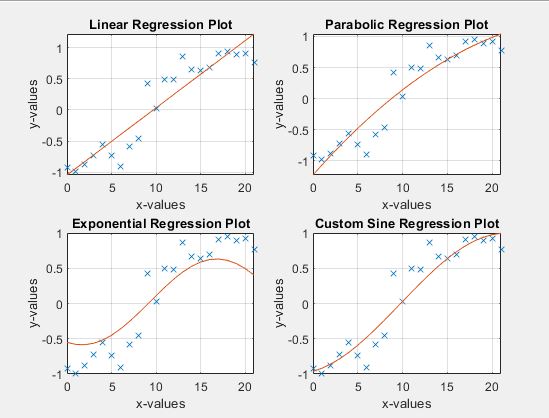
Johnny Donza MEC 320

Programming Assignment 3 Matlab

Submission Number: **0d5b0c2c-f461-41d6-a590-a940de4f7b61**

**Task 1:**

1. What were your equations for the Linear, Parabolic, and Custom Sine fits?
   1. Linear:
   2. Parabolic:
   3. Custom Sine fit:
2. Is it possible to do the exponential fit?
   1. No, it is not possible to perform an exponential fit to this data because taking the natural log of negative data points and data points that are equal to zero return a domain error due to the presence of imaginary or complex numbers.
3. What was the coefficient of determination and correlation coefficient for each of the fits?
   1. Linear:
   2. Parabolic:
   3. Custom Sine Fit:
   4. Exponential:
4. Plot the original data as data points and plot the best fit equations on the same graph using Matlab.

****

1. The fit that does the best job at fitting the data is the parabolic function because its correlation coefficient is the closest to one.
2. Matlab Code:

close all

clear

clc

%%%%%%%%%%Linear Least-Squares Regression

%Step 1: Extract data from sheet one in excel file into matlab matrix

A = xlsread('Data.xlsx',1);

%Step 2: Define unknowns in a1 and a0 equations

[u,m] = size(A);

sum\_x = 0;

sum\_y = 0;

sum\_xy = 0;

sum\_x2 = 0;

for i = 1:u

sum\_x = sum\_x + A(i,1);

sum\_y = sum\_y + A(i,2);

sum\_xy = sum\_xy + (A(i,1)\*A(i,2));

sum\_x2 = sum\_x2 + (A(i,1)\*A(i,1));

end

%Step 3: Solve for a1 and a0

a1 = ((u\*sum\_xy) - (sum\_x\*sum\_y)) / ((u\*sum\_x2) - (sum\_x)^2);

y\_average = sum\_y/u;

x\_average = sum\_x/u;

a0 = y\_average - (a1\*x\_average);

%Step 4: Display the linear regressed equation

fprintf('The linear regressed equation to fit the data is y = %f + %f\*x\n',a0,a1)

%Step 5: Find correlation coefficient

St = 0;

Sr = 0;

for i = 1:u

St = St + (A(i,2)-y\_average)^2;

Sr = Sr + (A(i,2)-a0-(a1\*A(i,1)))^2;

end

r2 = (St-Sr)/(St); %Coefficient of determination

r = sqrt(r2); %Correlation coefficient

disp('Correlation Coefficient for Linear Regressed Model = ');

disp(r);

disp('Coefficient of determination for the Linear Regressed Model =');

disp(r2);

%Step 6: Determine the new y-values for the linear regressed model

for i = 1:u

C(i,1) = a0 + a1\*A(i,1);

end

%%%%%%%%%%Parabolic Least-Squares Regression

%Step 1: Define unknowns for normal equations

sum\_x = 0;

sum\_y = 0;

sum\_xy = 0;

sum\_x2 = 0;

sum\_x3 = 0;

sum\_x4 = 0;

sum\_x2y = 0;

for i = 1:u

sum\_x = sum\_x + A(i,1);

sum\_y = sum\_y + A(i,2);

sum\_xy = sum\_xy + (A(i,1)\*A(i,2));

sum\_x2 = sum\_x2 + (A(i,1)\*A(i,1));

sum\_x3 = sum\_x3 + (A(i,1))^3;

sum\_x4 = sum\_x4 + (A(i,1))^4;

sum\_x2y = sum\_x2y + ((A(i,1))^2)\*A(i,2);

end

%Step 2: Use symbolic toolbox to set up matrix. x = a0, y = a1, z = a2.

syms x y z

eqn1 = (u\*x) + (sum\_x\*y) + (sum\_x2\*z) == sum\_y;

eqn2 = (sum\_x\*x) + (sum\_x2\*y) + (sum\_x3\*z) == sum\_xy;

eqn3 = (sum\_x2\*x) + (sum\_x3\*y) + (sum\_x4\*z) == sum\_x2y;

[a,b] = equationsToMatrix([eqn1, eqn2, eqn3],[x, y, z]);

%Step 3: Use Gauss Elimination with partial pivoting to solve for a0, a1, and a2

n = input('Number of unknowns for Gauss Elimination =');%In this particular case, n=3.

%Step 4: Create augmented matrix

Am = [a b];

%Step 5: Determine the size of the matrix

[nA,mA] = size(a);%nA = rows; mA = columns

[nb,mb] = size(b);%nb = rows; mb = columns

%Step 6: Foward Elimination with partial pivoting

for j = 1:nb %j is for columns

%Partial Pivoting

m = n+1;

p = 1;

k = p;

pivot = Am(k,k);

for ii = k+1:1:n

pivot2 = Am(ii,k);

if pivot2 > pivot

pivot = pivot2;

p = ii;

end

end

if p ~= k

for jj = k:1:n

pivot2 = Am(p,jj);

Am(p,jj) = Am(k,jj);

Am(k,jj) = pivot2;

end

pivot2 = Am(p,m);

Am(p,m) = Am(k,m);

Am(k,m) = pivot2;

end

for i = j+1:nA %i is for rows

Am(i,:) =Am(i,:)-(Am(j,:)\* Am(i,j)/Am(j,j));

end

end

%Step 7: Back Substitution

x = zeros(n,1); %Sets up a zero matrix to fill in solution values

m = n+1; %Written to insure we call last column in augmented matrix

x(n) = Am(n,m)/Am(n,n); %Defines the value for your last variable in matrix

for i = n-1:-1:1

solution = Am(i,m); %Calls upon solution value as given in b-vector

for j = i+1:n

solution = solution - Am(i,j)\*x(j,:);

end

x(i) = solution/Am(i,i);

end

disp('[x] =');

disp(x);

%Step 8: Check to see if code works

b = a\*x;

disp('b =');

disp(vpa(b));

%Step 9: Display Parabolic Equation

fprintf('The polynomial regressed equation to fit the data is y = %f + %f\*x + %f\*x^2\n',x(1),x(2),x(3))

%Step 10: Find correlation coefficient

St1 = 0;

Sr1 = 0;

for i = 1:n

St1 = St1 + (A(i,2)-y\_average)^2;

Sr1 = Sr1 + (A(i,2) - x(1) - (x(2)\*A(i,1)) - (x(3)\*(A(i,2))^2))^2;

end

r2\_1 = (St1-Sr1)/(St1); %Coefficient of determination

r\_1 = sqrt(r2\_1); %Correlation coefficient

disp('Correlation Coefficient for the Polynomial Regressed Model = ');

disp(r\_1);

disp('Coefficient of determination for the Polynomial Regressed Model =');

disp(r2\_1);

%Step 11: Determine the new y-values for the parabola

for i = 1:u

D(i,1) = x(1,1) + x(2,1)\*A(i,1) + (x(3,1)\*(A(i,1))^2);

end

%%%%%%%%%% Custom Sine Regression Model

%Step 1: Linearize the sine function and data into the form of

%sin^-1(y) = A + Bx where A = a0 and B = a1

for i = 1:u

ylin(i,:) = asin(A(i,2));

xlin(i,:) = A(i,1);

end

F = [xlin ylin]; %Puts the data into one matrix for simplicity

%Step 2: Define unknowns in a1 and a0 equations

[u,m] = size(F);

sum\_x = 0;

sum\_y = 0;

sum\_xy = 0;

sum\_x2 = 0;

for i = 1:u

sum\_x = sum\_x + F(i,1);

sum\_y = sum\_y + F(i,2);

sum\_xy = sum\_xy + (F(i,1)\*F(i,2));

sum\_x2 = sum\_x2 + (F(i,1)\*F(i,1));

end

%Step 3: Solve for a1 and a0

a1 = ((u\*sum\_xy) - (sum\_x\*sum\_y)) / ((u\*sum\_x2) - (sum\_x)^2);

y\_average = sum\_y/u;

x\_average = sum\_x/u;

a0 = y\_average - (a1\*x\_average);

%Step 4: Display the custom sine equation

fprintf('The custom sine function to fit the data is y = sin(%f + %f\*x)\n',a0,a1)

%Step 5: Find correlation coefficient

St = 0;

Sr = 0;

for i = 1:u

St = St + (A(i,2)-y\_average)^2;

Sr = Sr + (A(i,2)-a0-(a1\*A(i,1)))^2;

end

r2 = (St-Sr)/(St); %Coefficient of determination

r = sqrt(r2); %Correlation coefficient

disp('Correlation Coefficient for Custom Sine fit = ');

disp(r);

disp('Coefficient of determination for the Custom Sine fit =');

disp(r2);

%Step 6: Determine the new y-values for the linear regressed model

for i = 1:u

G(i,1) = sin(a0 + (a1 \* A(i,1)));

end

%%%%%%%%%%Exponential Least Sqaures Regression

%Step 1: Linearize the exponential function and data into the form of

%log(y) = log(alpha) + beta\*x; where log(alpha)=a0, beta=a1,

%log(y)=ylin

for i = 1:u

ylin(i,:) = log(A(i,2));

xlin(i,:) = A(i,1);

end

B = [xlin ylin]; %Puts the data into one matrix for simplicity

%Step 2: Define unknowns in a1 and a0 equations

[u,m] = size(B);

sum\_x = 0;

sum\_y = 0;

sum\_xy = 0;

sum\_x2 = 0;

for i = 1:u

sum\_x = sum\_x + B(i,1);

sum\_y = sum\_y + B(i,2);

sum\_xy = sum\_xy + (B(i,1)\*B(i,2));

sum\_x2 = sum\_x2 + (B(i,1)\*B(i,1));

end

%Step 3: Solve for a1 and a0

a1 = ((u\*sum\_xy) - (sum\_x\*sum\_y)) / ((u\*sum\_x2) - (sum\_x)^2);

y\_average = sum\_y/u;

x\_average = sum\_x/u;

a0 = y\_average - (a1\*x\_average);

%Step 4: Solve for alpha and beta

beta = a1;

alpha = exp(a0);

%Step 5: Display the Exponential regressed equation

fprintf('The exponential regressed equation to fit the data is y = %f\*e^%f\*x\n',alpha,beta)

%Step 6: Find correlation coefficient

St = 0;

Sr = 0;

for i = 1:u

St = St + (A(i,2)-y\_average)^2;

Sr = Sr + (A(i,2)-a0-(a1\*A(i,1)))^2;

end

r2 = (St-Sr)/(St); %Coefficient of determination

r = sqrt(r2); %Correlation coefficient

disp('Correlation Coefficient for the Exponential Model = ');

disp(r);

disp('Coefficient of determination for the Exponential Model =')

disp(r2);

%Step 7:Determine the new y-values for the linear regressed model

for i = 1:u

E(i,1) = alpha\*exp(beta\*A(i,1));

end

%%%%%%%%%% Plot all the data

figure

subplot(2,2,1)

plot(A(:,1),A(:,2),'x',A(:,1),C(:,1));

grid on

xlabel('x-values')

ylabel('y-values')

title('Linear Regression Plot')

subplot(2,2,2)

plot(A(:,1),A(:,2),'x',A(:,1),D(:,1));

grid on

xlabel('x-values')

ylabel('y-values')

title('Parabolic Regression Plot');

subplot(2,2,3)

plot(A(:,1),A(:,2),'x',A(:,1),E(:,1));

grid on

xlabel('x-values')

ylabel('y-values')

title('Exponential Regression Plot');

subplot(2,2,4)

plot(A(:,1),A(:,2),'x',A(:,1),G(:,1));

grid on

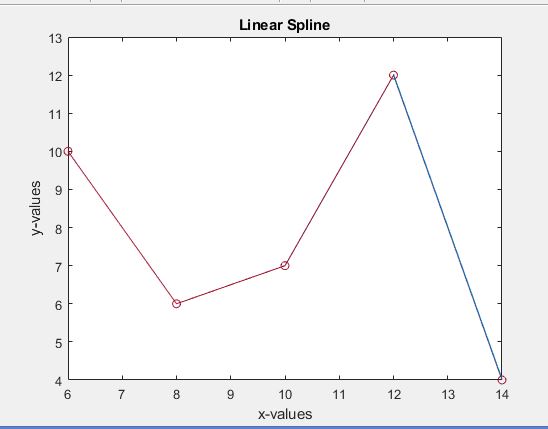
xlabel('x-values')

ylabel('y-values')

title('Custom Sine Regression Plot');

**Task 2:**

1. What are the equations for the linear and quadratic splines?
   1. Linear: y = 10.000000 + -2.000000\*x ; y = 6.000000 + 0.500000\*x ; y = 7.000000 + 2.500000\*x ; y = 12.000000 + -4.000000\*x
   2. Quadratic: ????
2. Plot the original data points and the spline.



1. Use your splines to interpolate the data set x = 11.2. What y-value did you get for each spline?
   1. Y = 10 + -2(11.2) = -12.4
   2. Y = 6 + 0.5(11.2) = 11.6
   3. Y = 7 + 2.5(11.2) = 35
   4. Y = 12 – 4(11.2) = -32.8
2. Use your code to perform inverse interpolation to find the x-value that corresponds to a y-value of 5.
   1. ?????
3. Matlab Code:

close all

clear

clc

%%%%%%%%%%Linear Spline

%Step 1: Extract data from sheet one in excel file into matlab matrix

A = xlsread('Data.xlsx',2);

%Step 2: Determine the size of matrix A

[n,u] = size(A);

%Step 3: Place values from A matric in matrix f and x

for i = 1:n

f(i,:) = A(i,2);

x(i,:) = A(i,1);

end

%Step 4: Determine the function values at each x-value utilizing the slopes determined in

%step 3.

for i = 1:n-1

r = linspace(x(i),x(i+1),1000);

m(i,:) = (f(i+1) - f(i)) / (x(i+1) - x(i));

fx = @(x) f(i,:) + m(i,:)\*(x-x(i));

fprintf('The linear spline equation to fit the data is y = %f + %f\*x\n',f(i,:),m(i,:))

plot(x,f,'-o',r,fx(r))

title('Linear Spline')

xlabel('x-values')

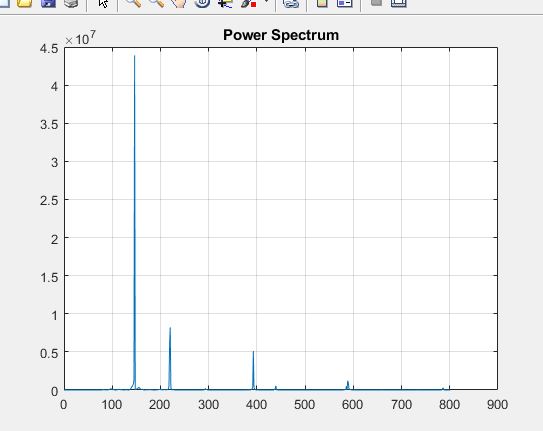
ylabel('y-values')

hold on

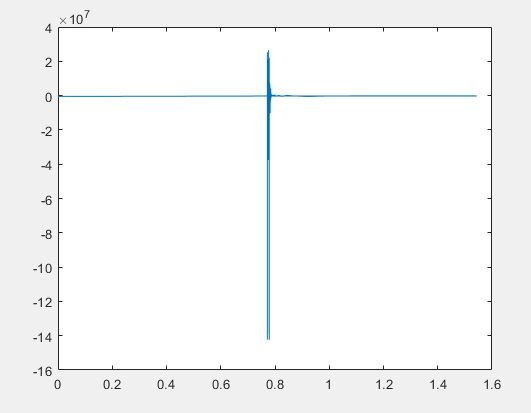
end

**Task 3:**

1. Make two plots of the power spectrum.
   1. Second Plot is made in Matlab.



1. What three notes was I playing?
   1. ???
2. I see 4 peaks in the power spectrum versus frequency data and the reason for this is because the last peak is due to the vibrations in the air of the instrument.
3. Take the inverse Fourier transform and plot.



1. ???
2. Matlab Code:

close all

clc

clear

%%%%%%%%%% Inverse Fourier Transform

%Step 1: Extract data from sheet one in excel file into matlab matrix

[num,A] = xlsread('Data.xlsx',3);

Fk = str2double(A);

[n,m] = size(Fk);

%Step 2: Calculate the power spectrum

for i = 1:n

pk(i,:) = 2\*abs(Fk(i))^2;

end

% The sampling frequency of the audio file is 44,100 Hz. The data is

% equi-spaced in the frequency domain starting at zero and ending at the

% sampling frequency. Therefore, the frequency is incremented at

% 44,100/34,032 = 1.2958 Hz per pk.

increment = 44100/n;

for i = 1:n

if i == 1

xi(i,:) = 0;

else

xi(i,:) = xi(i-1,:) + increment;

if xi(i,:) > 800

break

end

end

end

figure(1)

plot (xi,pk(1:i))

xlabel('0 Hz to 800 Hz');

ylabel('Decibel')

title('Power Spectrum')

grid on

i = i+1;

for j = i:n

xi(j,:) = xi(j-1,:) + increment;

if xi(j,:) > 3500

break

end

end

figure(2)

plot (xi(i:j),pk(i:j))

xlabel('0 Hz to 3500 Hz');

ylabel('Decibels');

title('Power Spectrum')

grid on

% Take the inverse fourier transform

T = 1/44100;

w0 = (2\*pi)/T;

for i = 1:n-1

if i == 1

xj(i,:) = 0;

else

xj(i,:) = xj(i-1,:) + increment;

end

end

for g = 1:n-1

if g == 1

ti(g,:) = 0;

elseif g == 2

ti(g,:) = 1/increment;

else

ti(g,:) = 1/(xj(g-1,:)) + (1/increment);

end

end

for k = 1:n-1

fn(k,1) = 0;

%imaginary(k,1) = 0;

for s = 0:n-1

angle = k\*w0\*s;

fn(k,1) = fn(k,1) + (Fk(k,1)\*cos(angle)) - (1i\*Fk(k,1)\*sin(angle)) ;

%imaginary(k,1) = imaginary(k,1) - (1i\*Fk(k,1)\*sin(angle));

end

end

figure (3)

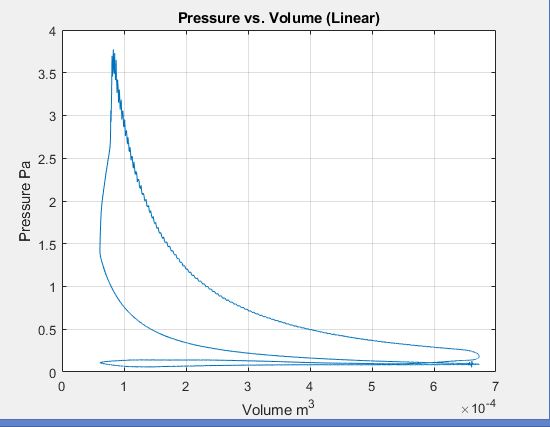
plot(ti,fn)

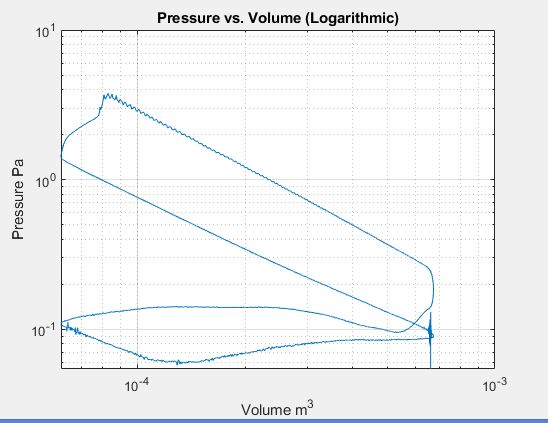
title('Amplitude Data vs. Time')

grid on

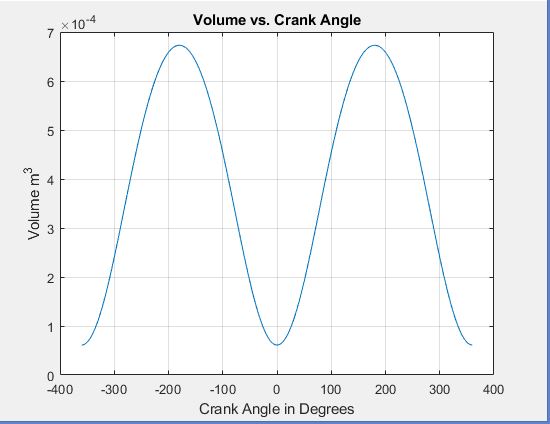
**Task 4:**

1. The work in joules over the whole cycle is equal to 3.4684x10^-4 joules.
2. Plot pressure as a function of volume.





1. Plot the volume vs. crank angle.



1. Shown in Matlab Code
2. ???
3. ???
4. Matlab Code

close all

clear

clc

%%%%%%%%%% Trapezoidal Rule

%Step 1: Extract the pressure and volume data from excel

P\_kpa = xlsread('Data.xlsx',4,'E3:E3602'); %Extracts pressure data in kpa

V\_m3 = xlsread('Data.xlsx',4,'D3:D3602'); %Extracts volume data in m^3

%Step 2: Convert pressure data into pascals

[n,m] = size(P\_kpa);

P\_pa = zeros(n,m);

for i = 1:n

P\_pa(i,m) = P\_kpa(i,m)/1000;

end

%Step 3: Calculate the work in Joules over the whole cycle

Work\_j = zeros(n,m);

for i = 1:n-1

Work\_j(i,m) = (V\_m3(i+1,m) - V\_m3(i,m)) \* ((P\_pa(i,m) + P\_pa(i+1,m))/2); %Calculates the work in joules

end

S = sum(Work\_j);

%Step 4: Plot the pressure as a function of volume in matlab

figure (1)

plot(V\_m3,P\_pa); %Plot made with linear x and y axes

title('Pressure vs. Volume (Linear)');

xlabel('Volume m^3')

ylabel('Pressure Pa')

grid on

figure (2)

loglog(V\_m3,P\_pa); %Plot made with logarithmic axes

title('Pressure vs. Volume (Logarithmic)');

xlabel('Volume m^3')

ylabel('Pressure Pa')

grid on

%Step 5: Plot the volume versus the crank angle

crank\_angle = xlsread('Data.xlsx',4,'C3:C3602'); %Grabs crank angle data from excel

figure (3)

plot(crank\_angle,V\_m3);

title ('Volume vs. Crank Angle');

xlabel('Crank Angle in Degrees');

ylabel('Volume m^3');

grid on

%Step 6: Use a central finite difference to find the derivative of the

%volume

V\_prime = zeros(n,m);

h = .2; %Defines step size of crank angle data

for i = 2:n-1

V\_prime(i,m) = (V\_m3(i+1,m) - V\_m3(i-1,m)) / (2\*h);

end

figure (4)

plot (crank\_angle,V\_prime);

title('Derivative of Volume vs. Crank Angle');

xlabel('Crank Angle')

ylabel('Derivative of Volume')

grid on

%Step 7: Implement an incremental search to estimate the range in which the

%zeros are located

vi = 1;

vii = 1;

j = 0;

for x = 2:n-2

vi= V\_prime(x,m);

vii = V\_prime(x+2,m);

if vi\*vii < 0

j = j+1;

lower(j,m) = crank\_angle(x);

lower(j,m+1) = x; %Puts cell number of value in next column

upper(j,m) = crank\_angle(x+2);

upper(j,m+1) = x+2; %Puts cell number of value in next column

end

end

%Step 8: Determine an approximate sine function to model the graph

max\_v = max(V\_prime);

min\_v = min(V\_prime);

max\_crank = max(crank\_angle);

min\_crank = min(crank\_angle);

for x = 1:n

y(x,m) = max\_v\*sind(crank\_angle(x,m));

end

figure (5)

plot(crank\_angle,y);

title('Approximate Sine Function to Represent Derivative of Volume')

xlabel('Crank Angle')

ylabel('Derivative of Volume')

grid on

fprintf('The sine function to approximately fit the data is y = %f\*sinx\n',max\_v)

%Step 9: Utilizing the approximate sine function and the bisection method,

%determine the location of the root.

ea = 100;

es = .0005; %units in percent

i = 0;

for k = 1:j

while ea > es

i = i+1;

xu = upper(k,m);

xl = lower(k,m);

Xr = (xu+xl)/2;

func\_Xr = max\_v\*(sind(Xr));

func\_xl = max\_v\*(sind(xl));

func\_xu = max\_v\*(sind(xu));

if func\_xl\*func\_Xr < 0 %If true, the root lies in the lower sub-interval

xu = Xr;

elseif func\_xl\*func\_Xr > 0 %If true, the root lies in the upper sub-interval

xl = Xr;

else %Only true if func\_xl\*func\_Xr = 0

xroot(k,m) = Xr;

break

end

ea = abs((xu-xl)/(xu+xl)) \*100;

end

end

disp('Crank Angle in which root occurs =');

disp(xroot);

%Step 10: Calculate the work for each individual stroke

for i = 1:lower(i,m+1)

k = 1;

Work = zeros(lower(i,m+1),m);

while V\_prime(k,m) >= 0

Work(k,m) = (V\_m3(k+1,m) - V\_m3(k,m)) \* ((P\_pa(k,m) + P\_pa(k+1,m))/2); %Calculates the work in joules

k = k+1;

end

while V\_prime(k,m) <=0

Work(k,m) = (V\_m3(k+1,m) - V\_m3(k,m)) \* ((P\_pa(k,m) + P\_pa(k+1,m))/2); %Calculates the work in joules

k = k+1;

end

end