Indian Institute of Technology, Dhanbad

SIGNALS AND SYSTEMS

MINI PROJECT

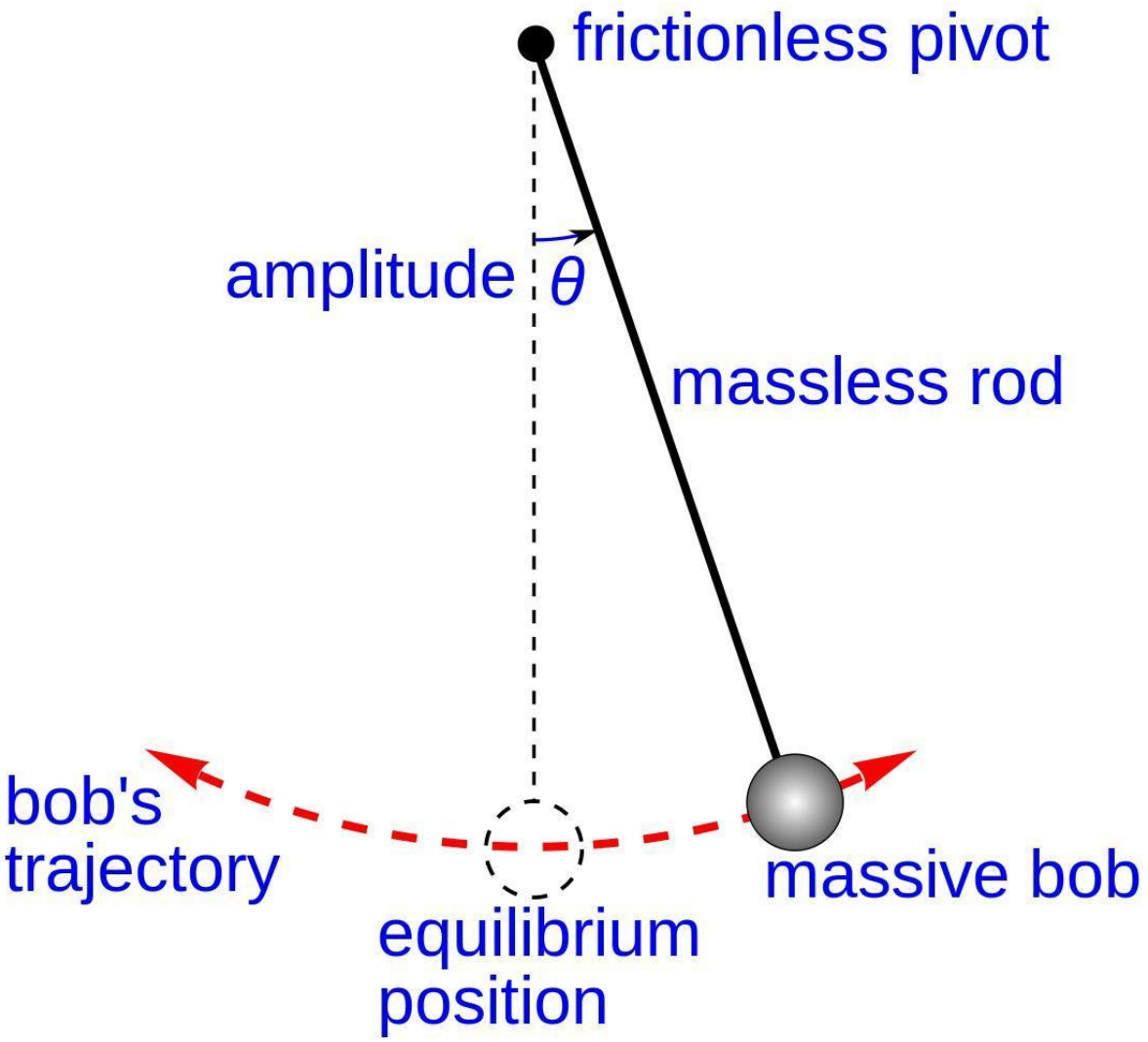
A Project Report on "Oscillations of The Simple Pendulum"

Designed by:-
Abhishek Goyal (20JE0031)

Idea ->

A non linear simple pendulum is simulated on the Matlab. Phase plane plot and time vs displacement plots will be depicted.

The Pendulum equation will be non linear. It will be solved using Matlab. Mass, length, damping and duration of pendulum can be changed.



The Simple Pendulum

A simple pendulum consists of a mass *m* hanging from a string of length *L* and fixed at a pivot point P. When displaced to an initial angle and released, the pendulum will swing back and forth with periodic motion. By applying Newton's second law for rotational systems, the equation of motion for the pendulum may be obtained

$$\tau = I\alpha \Rightarrow -mgL\sin\theta = mL^2(d^2\theta/dt^2)$$

and rearranged as

$$(d^2\theta/dt^2) + g\sin\theta/L = 0$$

If the amplitude of angular displacement is small enough that the small angle approximation($\sin\theta \sim \theta$) holds true, then the equation of motion reduces to the equation of simple harmonic motion.

$$(d^2\theta/dt^2) + g\theta/L = 0$$

The simple harmonic solution is

$$\theta(t) = (\theta o)\cos(\omega t + \Phi)$$

with $\omega = \sqrt{(g/L)}$ being the natural frequency of the motion.

The Real (Non linear) Pendulum

When the angular displacement amplitude of the pendulum is large enough that the small angle approximation no longer holds, then the equation of motion must remain in its non linear form

$$(d^2\theta/dt^2) + g\sin\theta/L = 0$$

Small Initial Amplitude

The small angle approximation is valid for initial angular displacements of about 20° or less. For small initial angular displacements the error in the small angle approximation becomes evident only after several oscillations.

Large Initial Amplitude

When the initial angular displacement is significantly large that the small angle approximation is no longer valid, the error between the simple harmonic solution and the actual solution becomes apparent almost immediately, and grows as time progresses.

Three cases of Damping:-

Under Damped :- "The condition in which damping of an oscillator causes it to return to equilibrium with the amplitude gradually decreasing to zero; system returns to equilibrium faster but overshoots and crosses the equilibrium position one or more times."

Critically Damped :- "The condition in which the damping of an oscillator causes it to return as quickly as possible to its equilibrium position without oscillating back and forth about this position."

Over Damped:- "The condition in which damping of an oscillator causes it to return to equilibrium without oscillating; oscillator moves more slowly toward equilibrium than in the critically damped system."

In a real environment we usually have some friction force which is directly proportional to the angular velocity 'ω' of the pendulum,

say $F = -mk\omega$

Therefore in case of damping the non linear equation becomes $(d^2\theta/dt^2) - k(d\theta/dt) + g\sin\theta/L = 0$

Therefore in case of critical damping the Damping Coefficient is calculated by:-

$$C = 2m\omega$$

Results :-

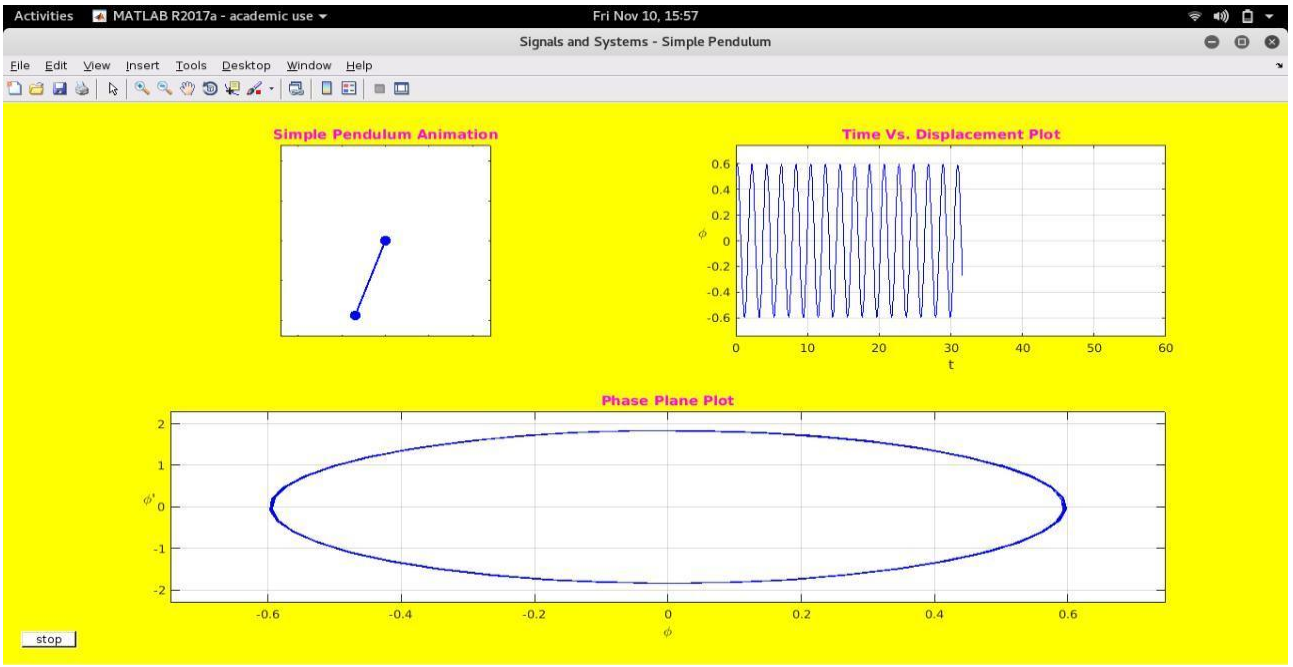
From this project we analyzed various types of equilibriums like unstable, marginally stable and asymptotically stable.

When $C < 0$; the system becomes an unstable system. When $C = 0$; the system becomes a marginally stable system.

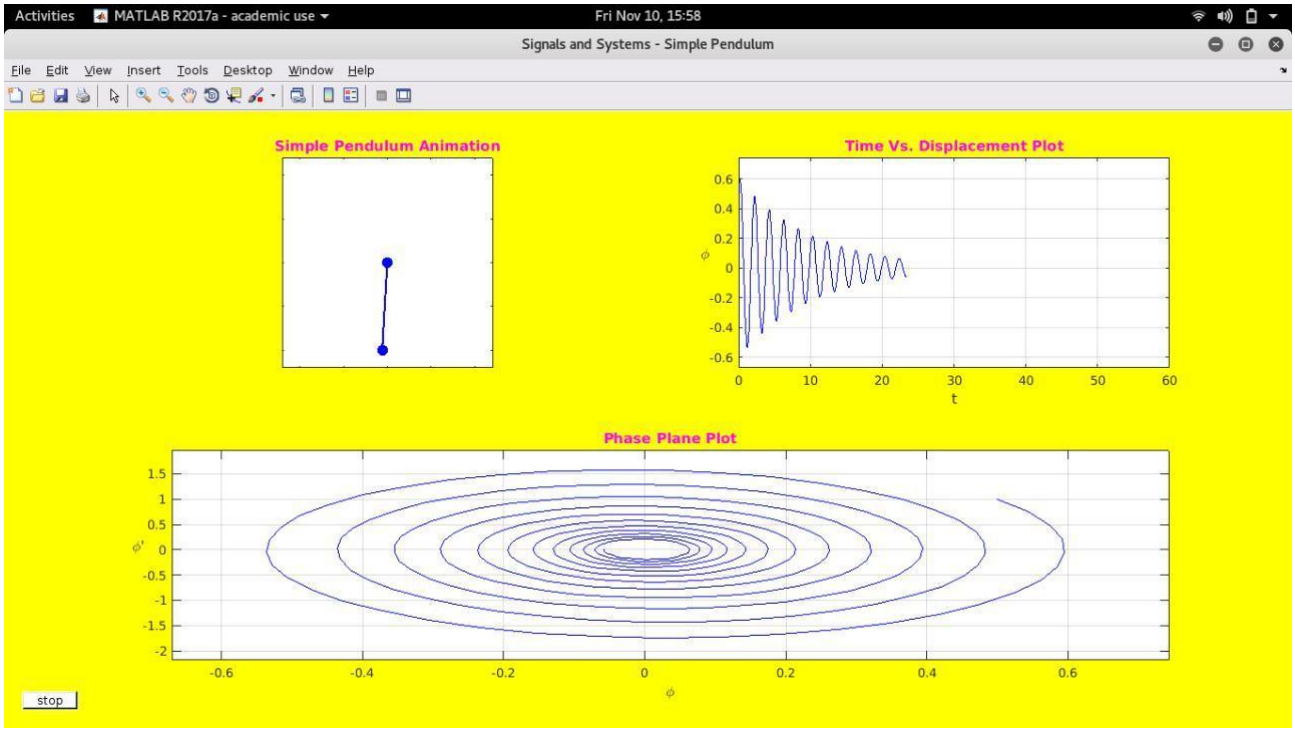
When $C > 0$; the system becomes an asymptotically stable system.

Various cases of Damping:-

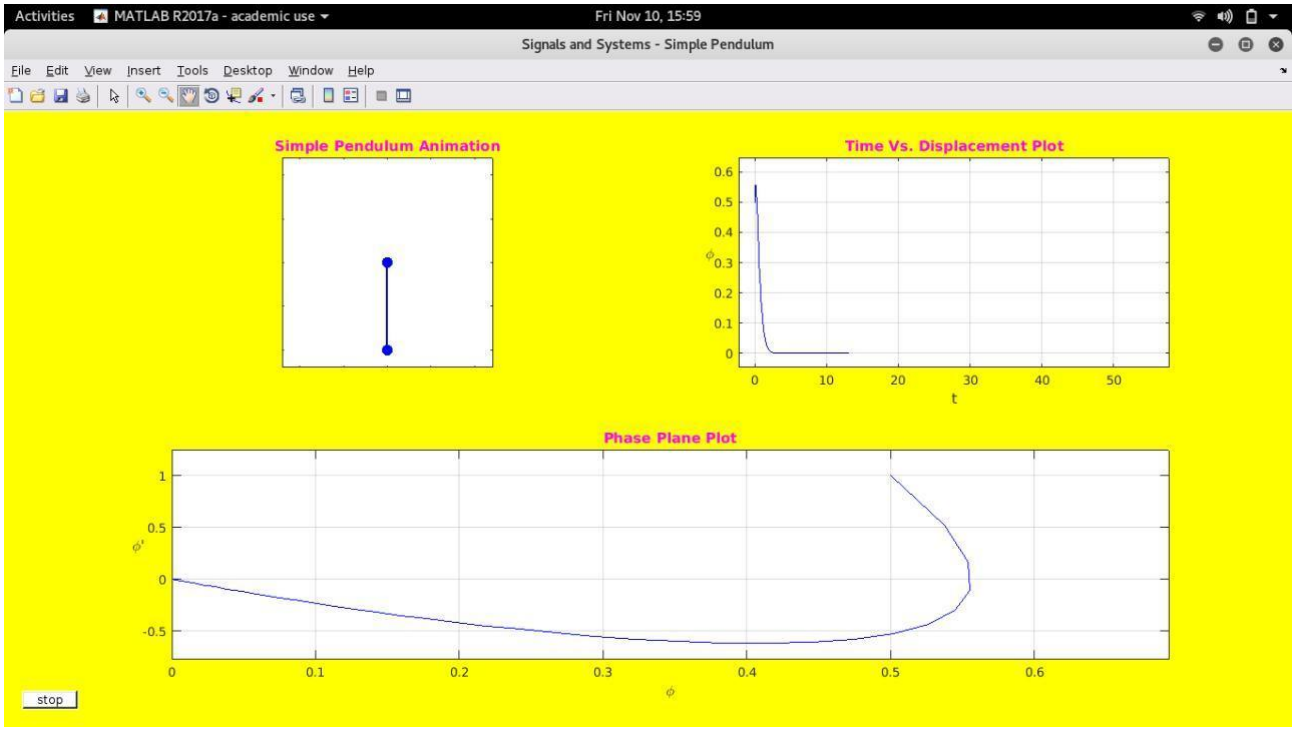
1. Damping Coefficient = 0 (Under Damped)



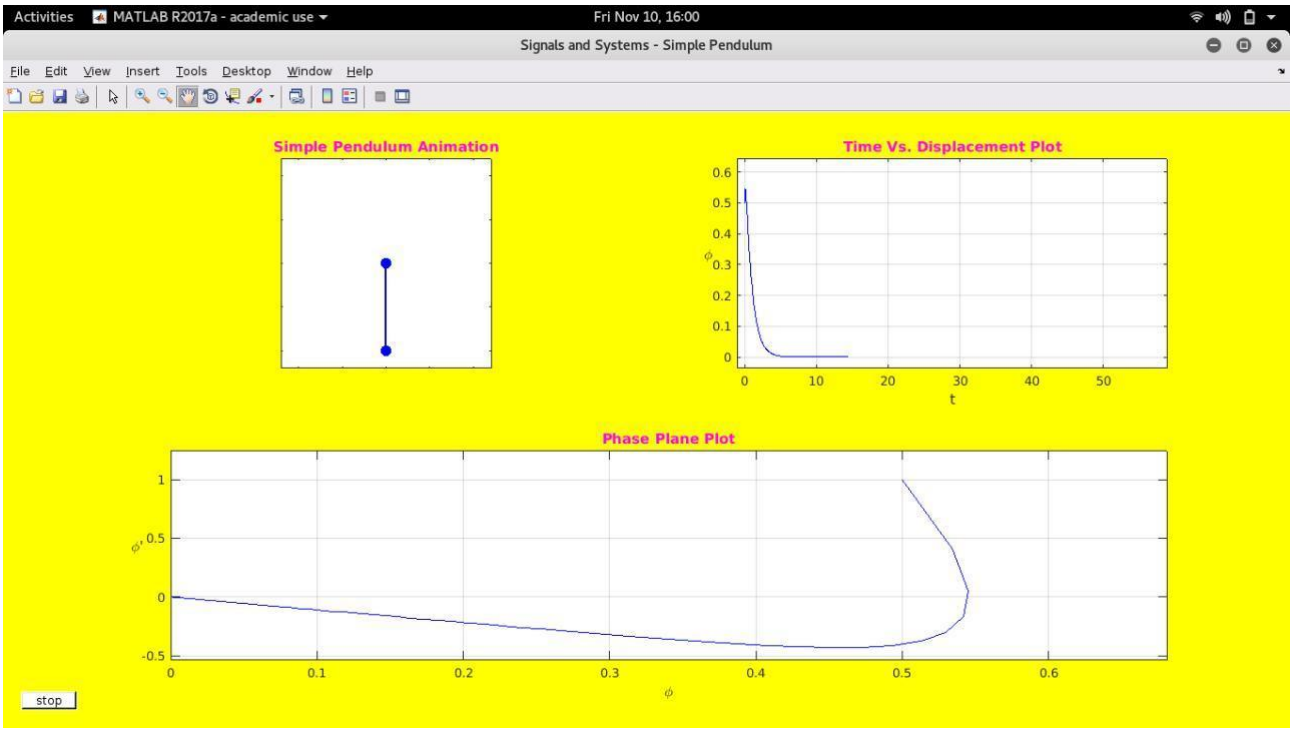
2. Damping Coefficient = 2 (Under Damped)



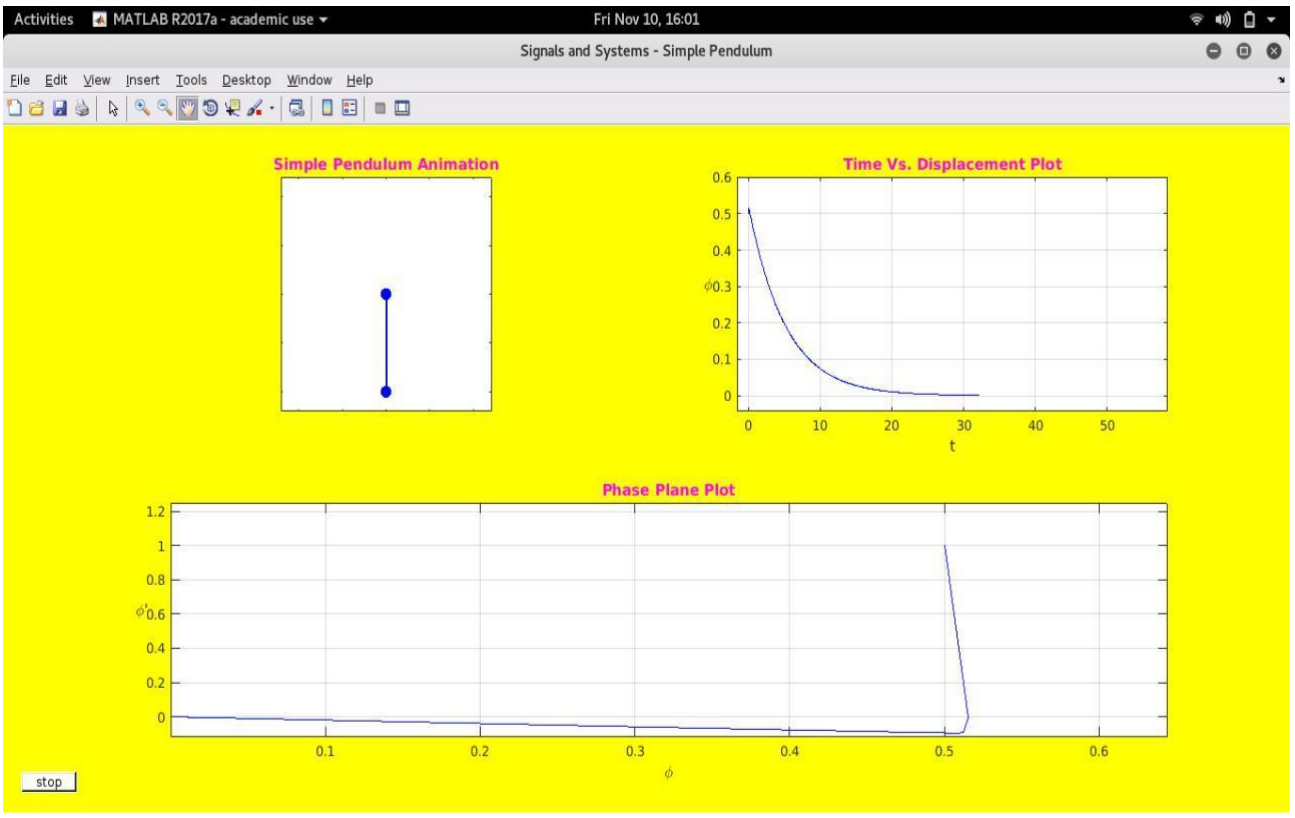
3. Damping Coefficient = 62.641839 (Critically Damped)



4. Damping Coefficient = 100 (Over Damped)



5. Damping Coefficient = 500 (Over Damped)



Conclusion ->

From this project we learn about how to analyse physical experiments using Matlab and to predict the result. Also learned various applications of Matlab.

Interests for Future Work:-

1. Analysis of the oscillations of compound pendulum.

2. Analysis of Harmonic Oscillator.

3. Stability analysis in Transients etc.

References:-

1. Getting Started with MATLAB: A quick introduction for Scientists & Engineers - By Pratap, Rudra

2.https://www.maths.tcd.ie/~robinson/labs/pendulum.pdf

Matlab Code

```
clear ;
clc ;
%Properties of Pendulum
g = 9.81 ;                  % Acceleration due to gravity
mass = 10 ;                 % Mass of the pendulum
len = 1 ;                   % Length of the Pendulum
damCoff = 1000 ;            % Damping coefficient
% Initial Boundary Conditions for the motion of the pendulum
Phi = 0.5;                  % Angular Position
PhiDot = 1.0;               % Angular Velocity
totDuration = 60;           % Duration of the Simulation
fps = 20;                   % Number of frames per second
interval = [0, totDuration];% Time span represented as vector
initial=[Phi; PhiDot; g; mass; len ; damCoff]; % Initial conditions for the problem
% Simulation of Simple Pendulum by calling graph() function
graph(initial, totDuration, fps);


%defining graph() function
function graph(initial, totDuration, fps)
totFrames=totDuration*fps; % Total number of frames
% Solving the Differential Equation of the pendulum using ode45
solutions=ode45(@Equation,[0 totDuration], initial);
time = linspace(0,totDuration,totFrames);
%obtaining the solution to the problem from the 'solution' structure returned by ode45
y = deval(solutions,time);
%Assigning Angular Position and velocity to new variables
phi = y(1,:)';
phiDot = y(2,:)';
len = initial(5);
% To set the Range os Phase plane, time vs. depl plots
minPhi = 1.25*min(phi) ; maxPhi = 1.25*max(phi);
minPhiDot = 1.25*min(phiDot) ; maxPhiDot = 1.25*max(phiDot);
% Drawing the graph
newFig = figure ;
set(newFig,'name','Signals and Systems - Simple Pendulum','numbertitle','off','color', 'y','menubar','figure') ;
%toggle button to stop the simulation
stop = uicontrol('style','toggle','string','stop','background','w');
% Plot for swinging pendulum
subplot(2,2,1);
SubP1 = plot(0,0,'MarkerSize',30,'Marker','.','LineWidth',1.5,'Color','b');
title('Simple Pendulum Animation','Color','m');
range = 1.2*len;
axis([-range range -range range]);
axis square;
set(gca,'XTickLabelMode', 'manual', 'XTickLabel', [],'YTickLabelMode', .....
    'manual', 'YTickLabel', []);
% Plot for the phase plane description of pendulum motion
subplot(2,2,[3 4]) ;
subP2 = plot(initial(1),initial(2),'LineWidth',1,'Color','b') ;
axis([minPhi maxPhi minPhiDot maxPhiDot]) ;
xlabel('\phi') ;ylabel('\phi'') ;
set(get(gca,'YLabel'),'Rotation',0.0)
grid on ;
title('Phase Plane Plot','Color','m')
% Plot for time Vs. displacement graph
subplot(2,2,2) ;
subP3 = plot(time(1),initial(1),'LineWidth',1,'Color','b') ;
axis([0 totDuration minPhi maxPhi]) ;
xlabel('t') ;ylabel('\phi') ;
set(get(gca,'YLabel'),'Rotation',0.0)
grid on ;
title('Time Vs. Displacement Plot','Color','m');
% Animation starts
for i=1:length(phi)-1
    % Animation Plot
    if (ishandle(SubP1)==1)
        PendX=[0,len*sin(phi(i))];
        PendY=[0,-len*cos(phi(i))];
        set(SubP1,'XData',PendX,'YData',PendY);
        if get(stop,'value')==0
            drawnow;
        elseif get(stop,'value')==1
            break;
        end
        % Phase Plane Plot
        if (ishandle(subP2)==1)
            PhaPlot(i,(1:2)) = [phi(i) phiDot(i)];
            set(subP2,'XData',PhaPlot(:,1),'YData',PhaPlot(:,2));
            drawnow;
        end
            % Time Vs. displacement Plot
        if (ishandle(subP3)==1)
            TimePlot(i,(1:2)) = [time(i) phi(i)] ;
            set(subP3,'Xdata',TimePlot(:,1),'YData',TimePlot(:,2)) ;
            drawnow ;
        end
   end
end
% Close the Figure window
set(stop,'style','pushbutton','string','close','callback','close(gcf)');
end


%function defining differential equation for the angular velocity of the pendulum
%function is passed as an argument to ode45
function xVel = Equation(~,x)
w0 = x(3)/x(5);
eta = x(6)/(x(4)*x(5));
xVel=zeros(length(x),1);
xVel(1)=x(2);       %angular velocity
xVel(2)=-(w0*sin(x(1))+eta*x(2));   %angular acceleration
%xVel=[x(2);  -(w0*sin(x(1))+eta*x(2));];
end
```