CS

LAB 5

DAMPED DRIVEN OSCILLATIONS

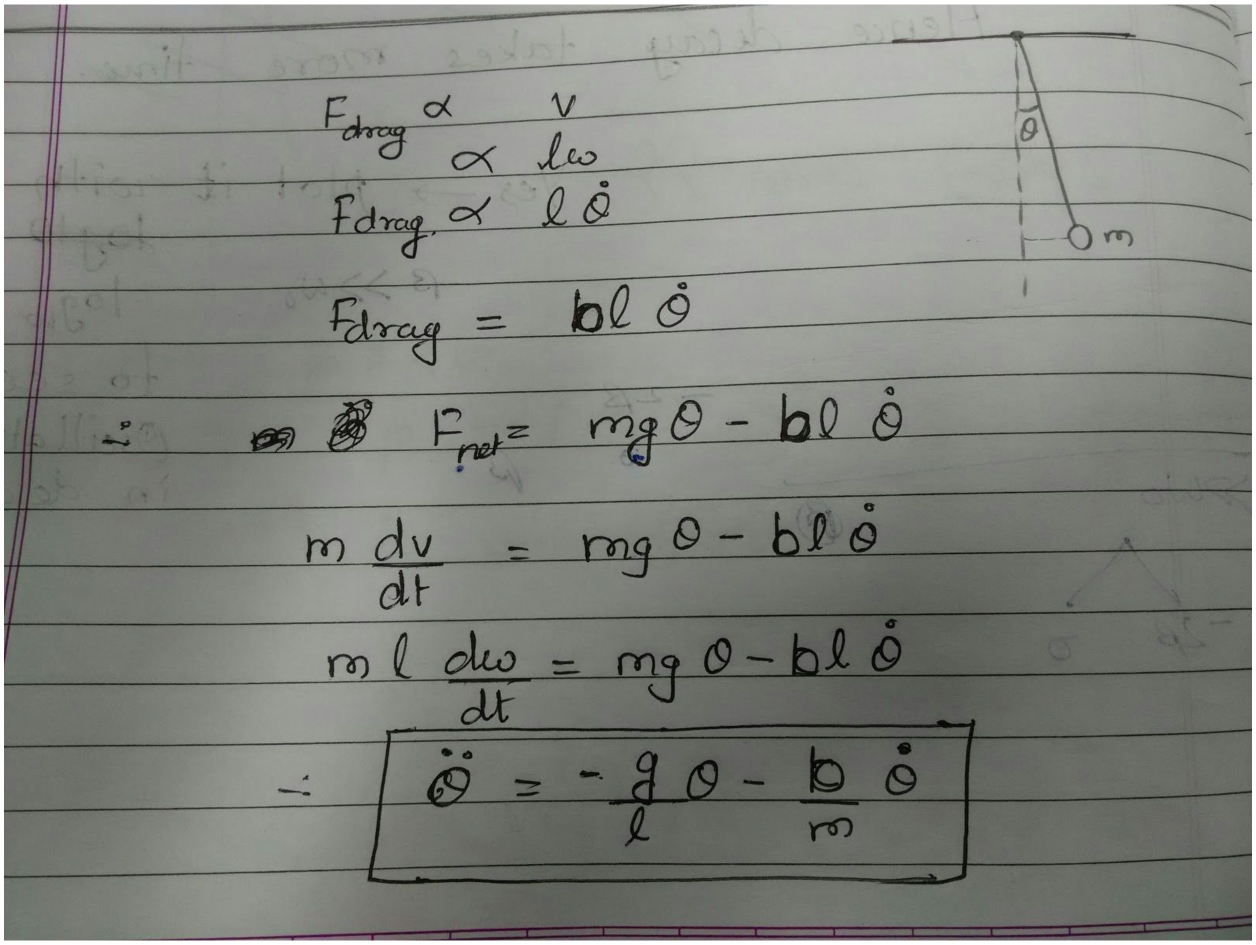
Rajdeep Pinge

201401103

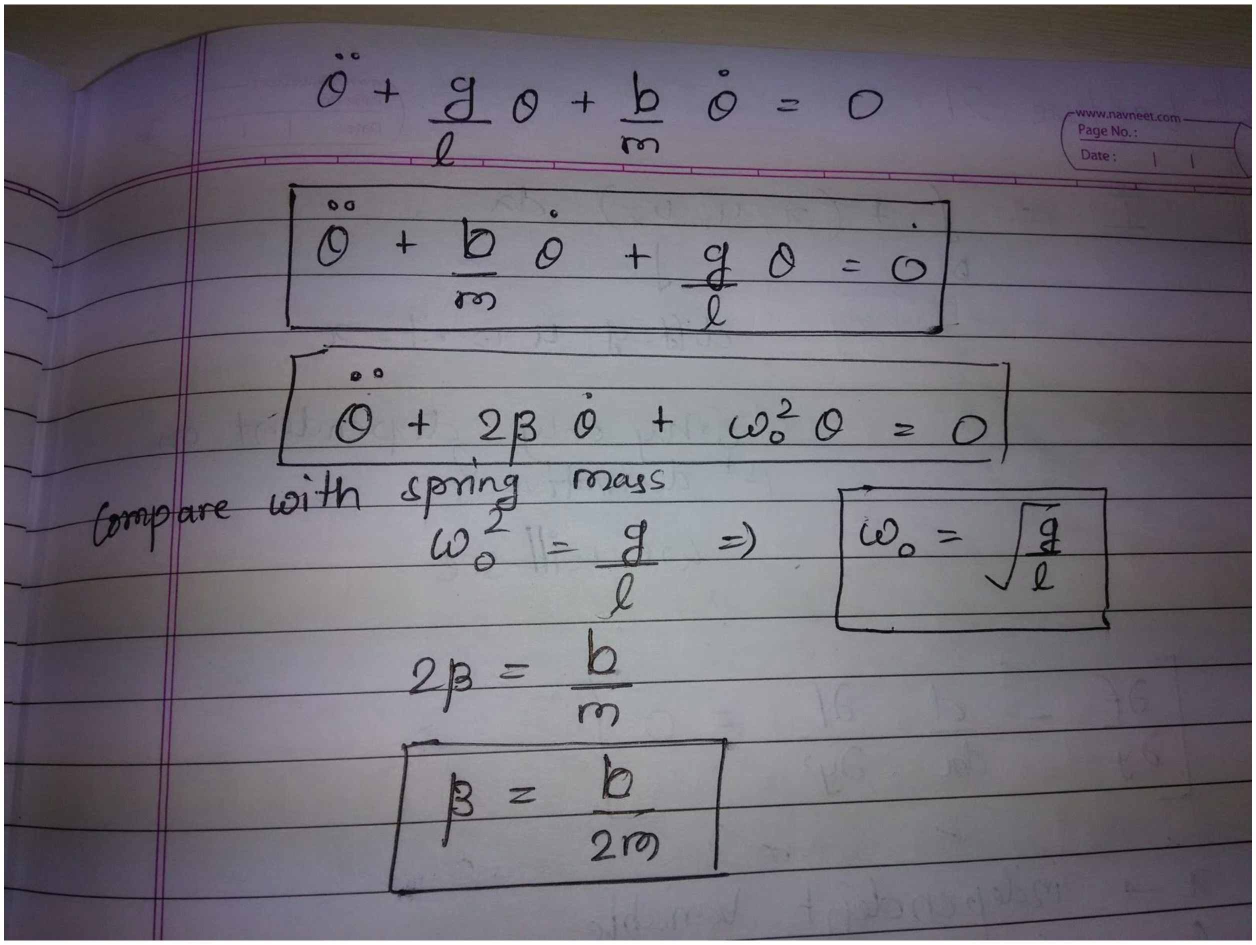
Aditya Joglekar

201401086

Q1) Damped Oscillations



$$F_{drag} \propto v$$
$$\propto l\omega$$
$$F_{drag} \propto l\dot{\theta}$$
$$F_{drag} = bl\dot{\theta}$$
$$\therefore F_{net} = mg\theta - bl\dot{\theta}$$
$$m\frac{dv}{dt} = mg\theta - bl\dot{\theta}$$
$$ml\frac{d\omega}{dt} = mg\theta - bl\dot{\theta}$$
$$\therefore \boxed{\ddot{\theta} = -\frac{g}{l}\theta - \frac{b}{m}\dot{\theta}}$$



www.navneet.com Page No.: Date: / /

$$\ddot{\theta} + \frac{g}{l}\theta + \frac{b}{m}\dot{\theta} = 0$$

$$\boxed{\ddot{\theta} + \frac{b}{m}\dot{\theta} + \frac{g}{l}\theta = 0}$$

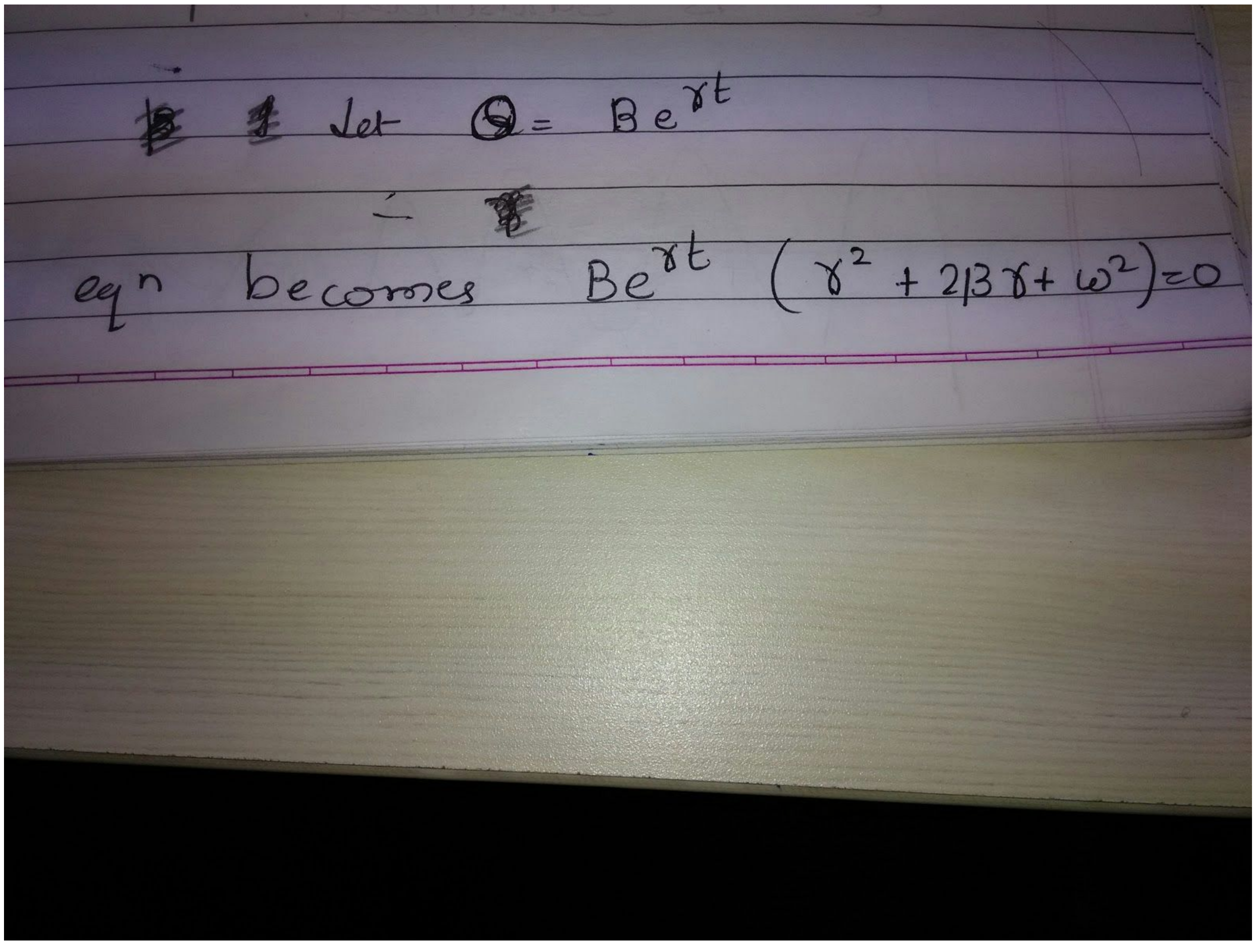
$$\boxed{\ddot{\theta} + 2\beta\dot{\theta} + \omega_0^2\theta = 0}$$

Compare with spring mass

$$\omega_0^2 = \frac{g}{l} \Rightarrow \boxed{\omega_0 = \sqrt{\frac{g}{l}}}$$

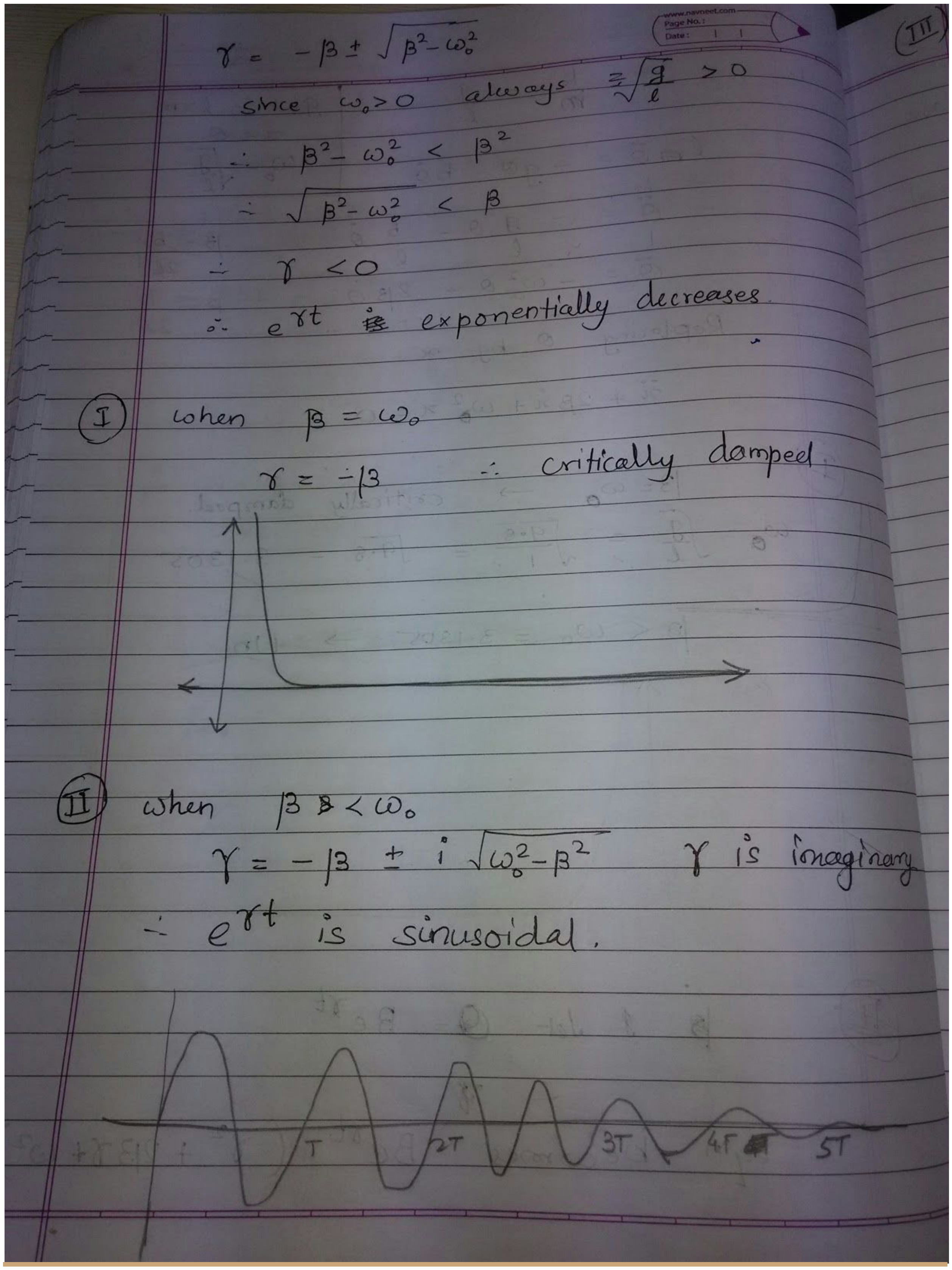
$$2\beta = \frac{b}{m}$$

$$\boxed{\beta = \frac{b}{2m}}$$



let $\theta = Be^{\gamma t}$

eq^n becomes $Be^{\gamma t}\left(\gamma^2 + 2\beta\gamma + \omega^2\right) = 0$



$\gamma = -\beta \pm \sqrt{\beta^2 - \omega_0^2}$

since $\omega_0 > 0$ always $= \sqrt{\frac{g}{l}} > 0$

$\therefore \beta^2 - \omega_0^2 < \beta^2$

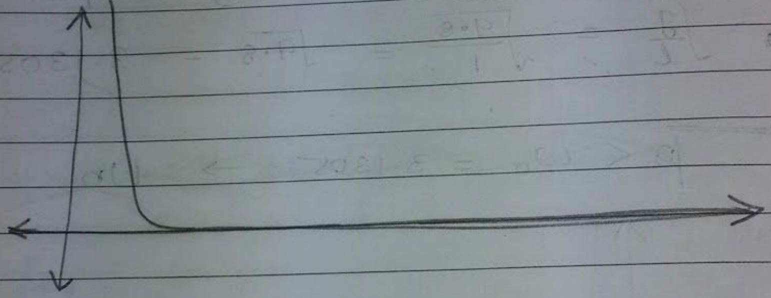
$\therefore \sqrt{\beta^2 - \omega_0^2} < \beta$

$\therefore \gamma < 0$

$\therefore e^{\gamma t}$ ~~is~~ exponentially decreases

(I) when $\beta = \omega_0$

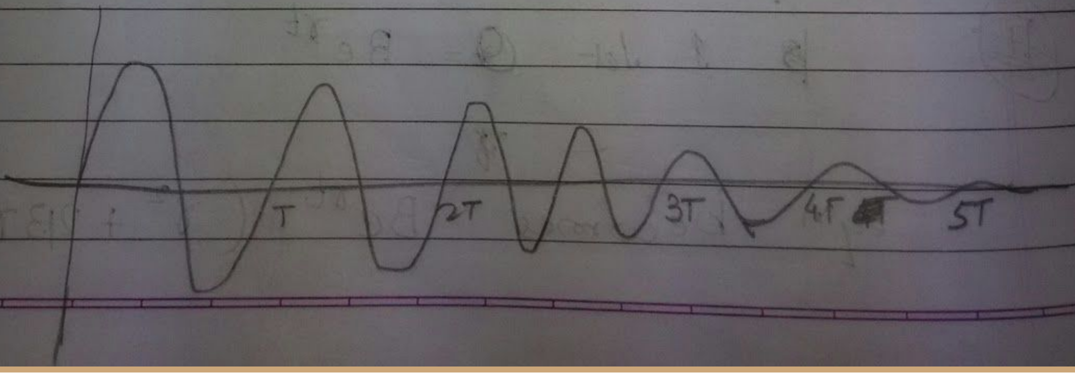
$\gamma = -\beta$ $\therefore$ critically damped

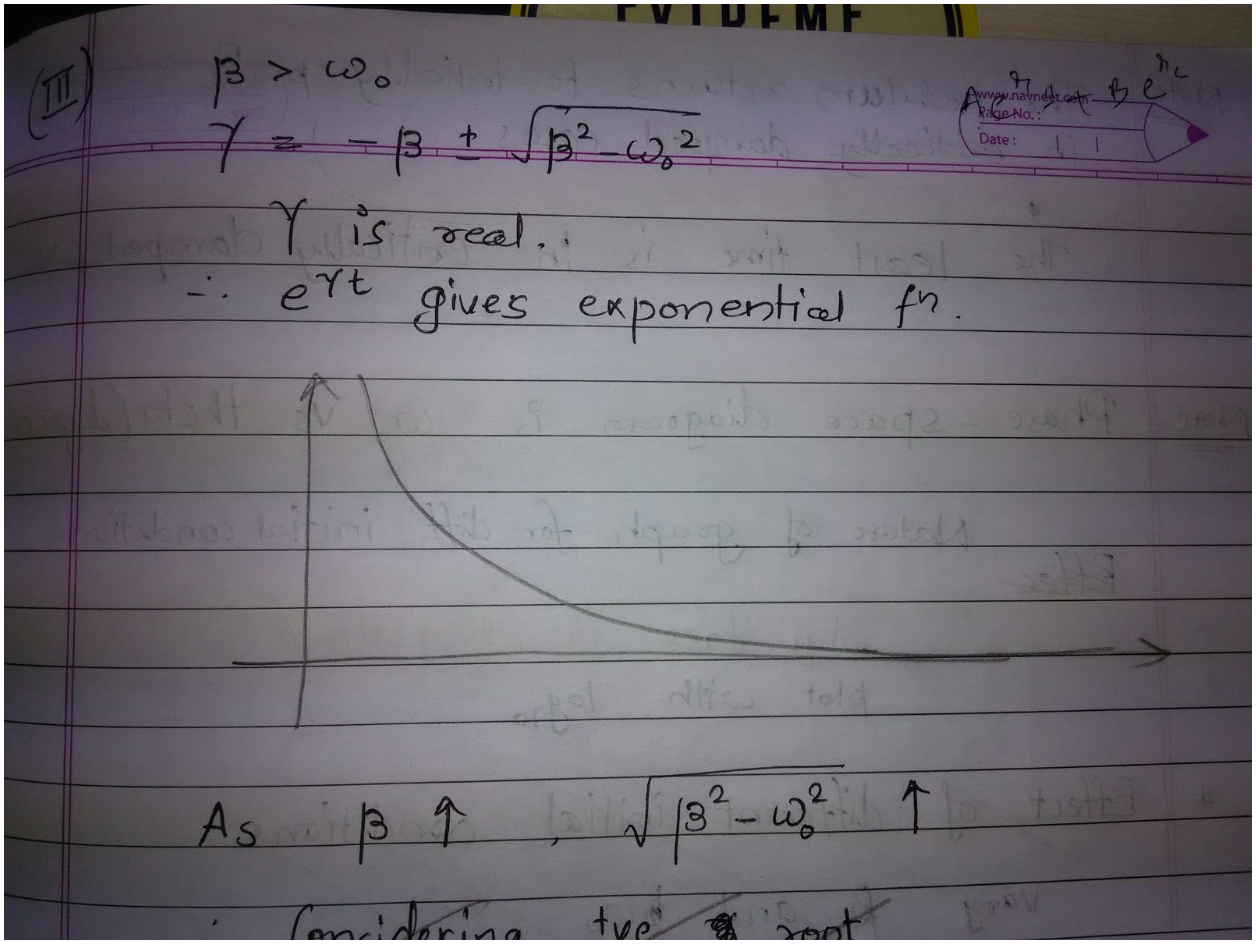


(II) when β ~~β~~ $< \omega_0$

$\gamma = -\beta \pm i\sqrt{\omega_0^2 - \beta^2}$ γ is imaginary

$\therefore e^{\gamma t}$ is sinusoidal.





(III) $\beta > \omega_0$

$\gamma = -\beta \pm \sqrt{\beta^2 - \omega_0^2}$

$Ae^{r_1 t} + Be^{r_2 t}$

Page No.:
Date: / /

γ is real.

$\therefore$ $e^{\gamma t}$ gives exponential fn.

As $\beta \uparrow$, $\sqrt{\beta^2 - \omega_0^2} \uparrow$

Considering +ve root

MATLAB Code:

```
clear;

close all;

% declare the pendulam variables to be global and set it


g = 9.8;

l = 10;


global omega0;
```

```
omega0 = sqrt(g/l);

b = 0.1;          % critical case 2*l*sqrt(g/l);
m = 1;

global beta;

timescale = 2*pi*sqrt(l/g);
dt = timescale/100;

% set the initial and final times
tstart = 0;
tfinal = 10*timescale;

% set the initial conditions in the y0 column vector
u_init = zeros(2,1);
u_init(1) = 0.5; % initial position theta;
u_init(2) = 0; % initial velocity



%without damping
beta = 0;                 % critical case 2*l*sqrt(g/l);

% call ode solver
```

```
options=odeset('RelTol',1e-8);

[t,u]=ode45(@q4_pendulumodefunction, [tstart:dt:tfinal], u_init, options);


% store the solution that comes back into x and v arrays

x = (180/pi)*u(:,1); % radian-->degree

v = u(:,2);




%underdamped

beta = .02;                   % critical case 2*l*sqrt(g/l);


% set the solve options

options=odeset('RelTol',1e-8);

[t,u]=ode45(@q4_pendulumodefunction, [tstart:dt:tfinal], u_init, options);


% store the solution that comes back into x and v arrays

x2 = (180/pi)*u(:,1); % radian-->degree

v2 = u(:,2);




%critically damped

beta = sqrt(g/l);           % critical case sqrt(g/l);
```

```
% set the solve options

options=odeset('RelTol',1e-8);

[t,u]=ode45(@q4_pendulumodefunction, [tstart:dt:tfinal], u_init, options);


% store the solution that comes back into x and v arrays

x3 = (180/pi)*u(:,1); % radian-->degree

v3 = u(:,2);




%overdamped

beta = 15;                    % critical case 2*l*sqrt(g/l);


% set the solve options

options=odeset('RelTol',1e-8);

[t,u]=ode45(@q4_pendulumodefunction, [tstart:dt:tfinal], u_init, options);


% store the solution that comes back into x and v arrays

x4 = (180/pi)*u(:,1); % radian-->degree

v4 = u(:,2);




% plot the position vs. time

plot(t,x, t,x2,'r-', t,x3,'g', t,x4, 'b--')
```

```
title('Motion of pendulum')

xlabel('time')

ylabel('position')


% make a "phase-space" plot of v vs. x

figure

plot(x,v, x2,v2,'r-', x3,v3,'g',  x4,v4, 'b--')

title('Phase-space diagram')

xlabel('position')

ylabel('velocity')
```

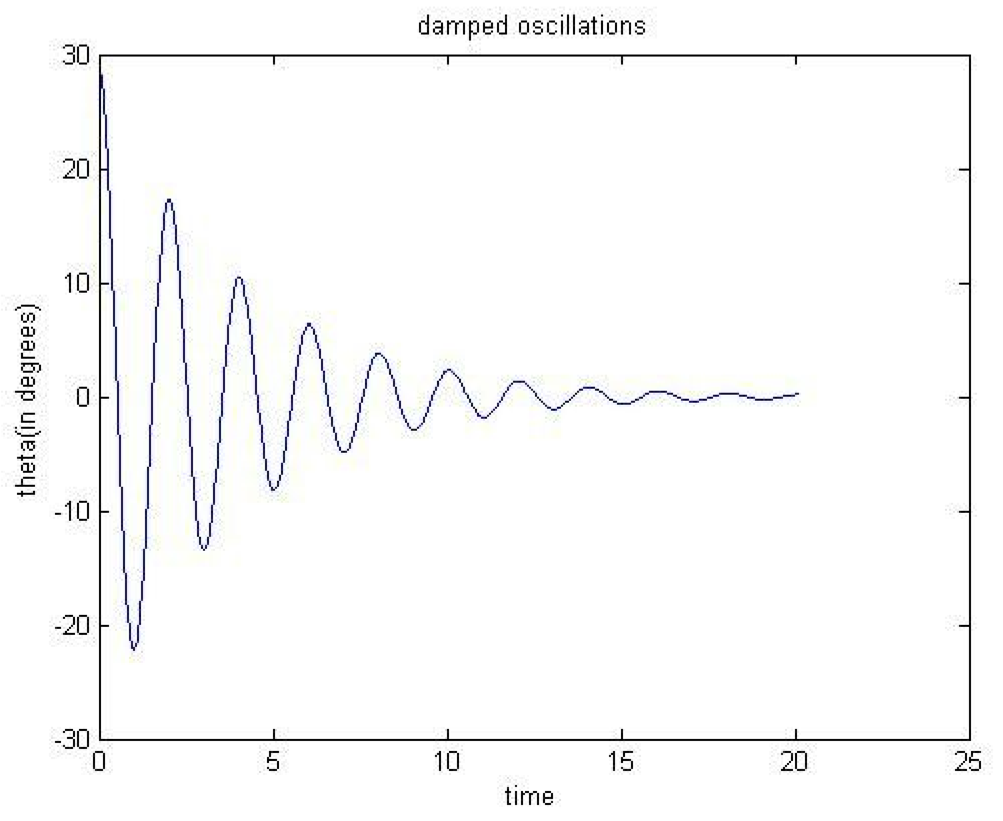
ODE solver Function:

```
function F=q4_pendulumodefunction(t,u)

% function output =name(input)

% right-hand side function for Matlab's ODE solver,


% In our case we will use:

% u(1) -> theta

% u(2) -> omega


% declare the globals so its value

% set in the main script can be used here

global omega0;

global beta;
```

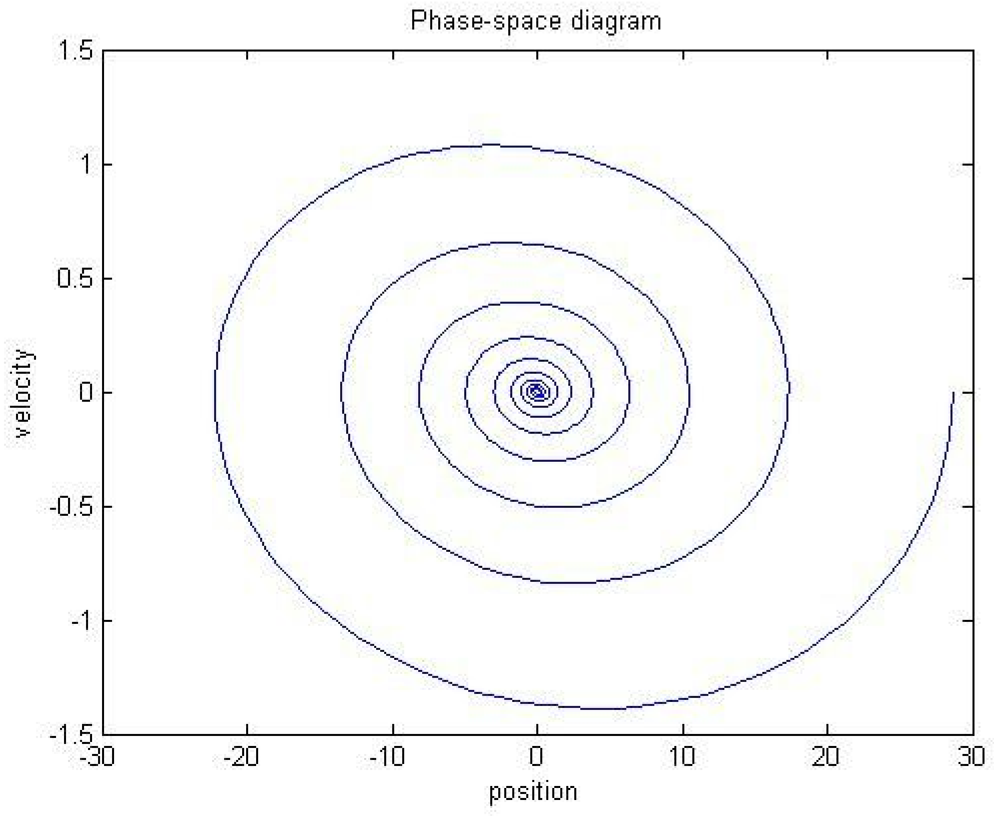
```
% make the column vector F filled with zeros

F = zeros(length(u),1);


% Now build the elements of F

%

% so the equation dx/dt=v means that F(1)=u(2)

F(1) = u(2);

% Again, in our original ODEs we have:


% so the equation dv/dt=-....

F(2) = -omega0*omega0*u(1) - 2*beta*u(2);
```

Underdamped Case:

beta = 0.3

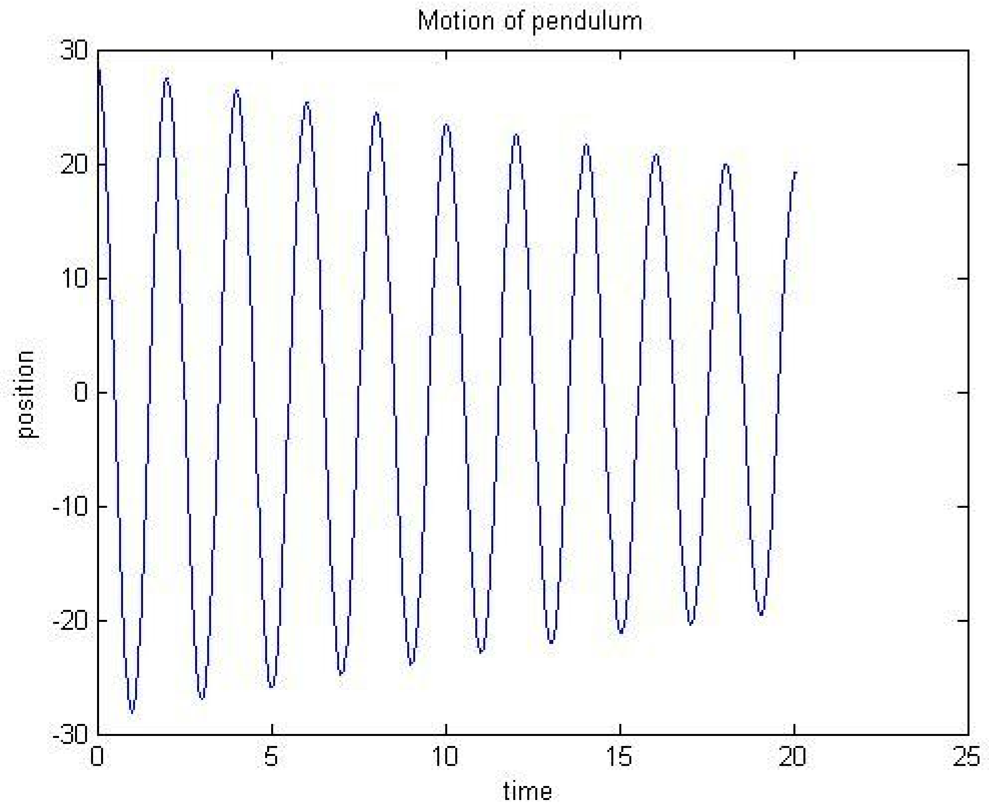


Here we consider a pendulum which is initially at a small angle from the equilibrium position. We release and it starts performing damped oscillations eventually coming to rest. We have assumed that the angle of oscillations is small and hence it can be studied as approximately a one dimensional SHM. Also, we have assumed that the damping force is proportional to the angular velocity of the pendulum a realistic assumption in many cases. We get the expected plots, in the first diagram we see sinusoidal behaviour which gradually dies down. This is however only one of 3 possible ways in which the motion can happen. This is the underdamped case. When the square of the damping coefficient B is less than the square of the natural frequency.



In the phase space diagram, clearly, we can see that the system is losing energy because of damping and that is the reason why we get a spiralling phase space plot. Note that after reaching the 0 position point in each cycle, the pendulum goes a lesser distance in the opposite direction. That is why it eventually comes to rest at the equilibrium position. Note here the direction of motion/plot is clockwise.

Here beta = 0.02

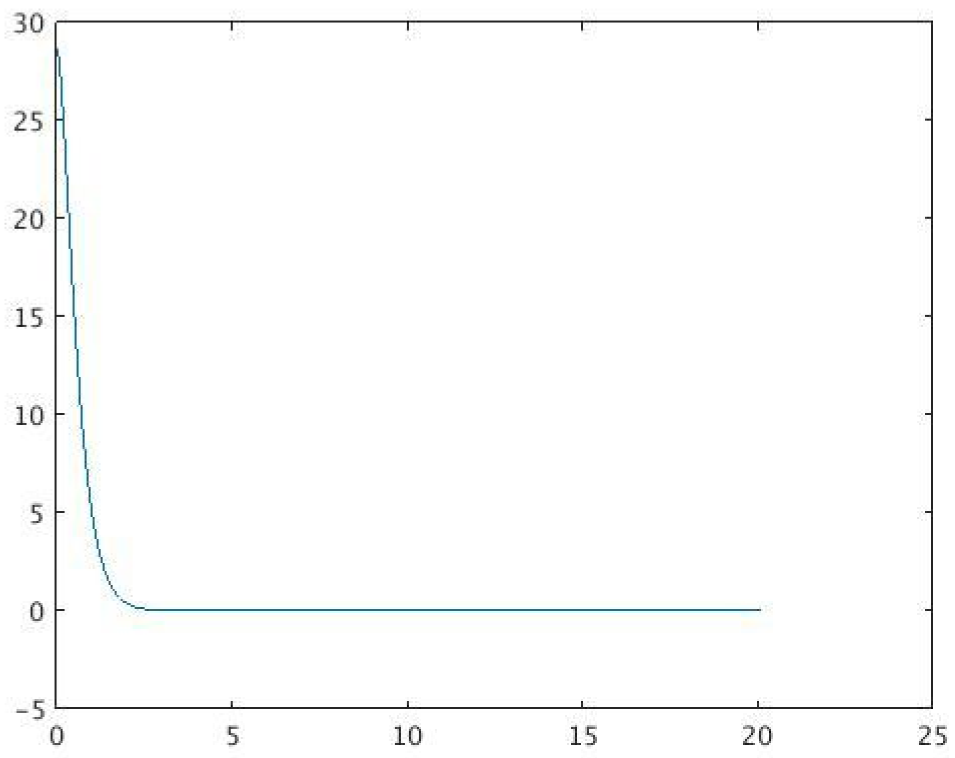


Therefore, as damping constant decreases, the pendulum takes more time to achieve equilibrium.

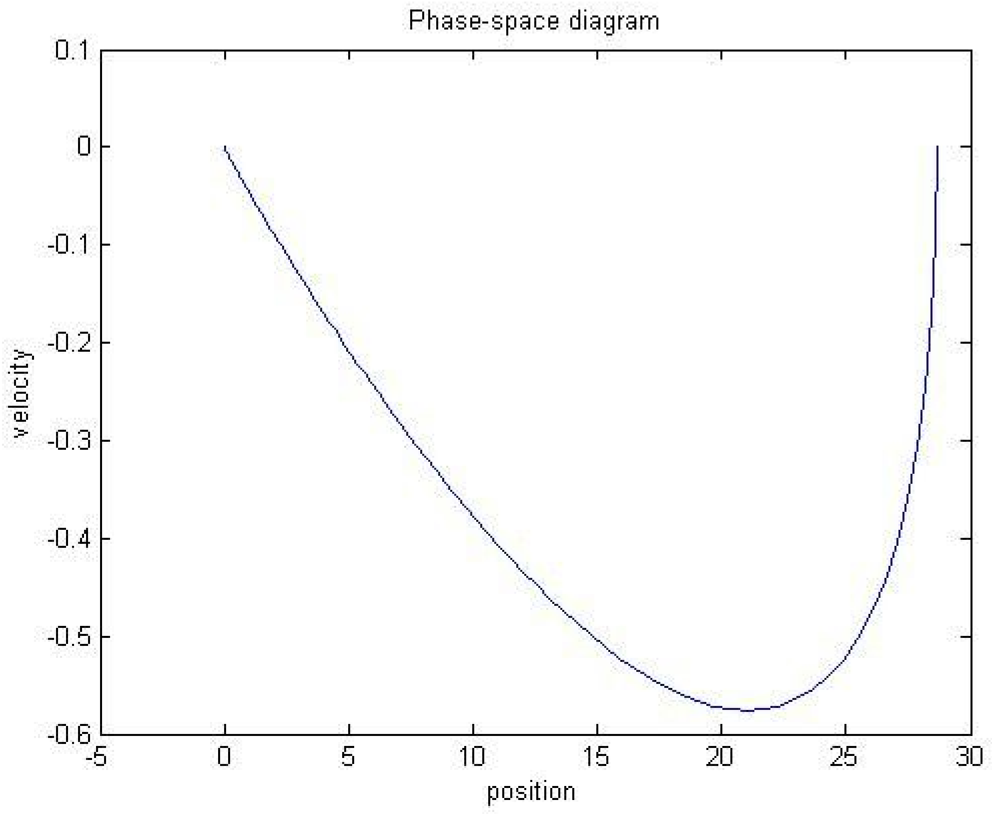
As shown in the calculations, change in beta affects the frequency of pendulum. Here, as beta is very small, the change in frequency is not visible. The change in frequency will only be visible if contribution of beta is significant to the motion. Hence we will see the effect of change in beta in the case of overdamped oscillations.

Critical Damping Case:

Now, we consider the case when square of the damping coefficient B is exactly equal to the square of the natural frequency. This is the case of critical damping.



In case of the critically damped system, the term -beta (+/-) sqrt (beta^2 - natural_freq^2) just becomes real. It is a boundary case between getting complex roots and real roots. Hence the system instead of oscillating quickly comes to rest.



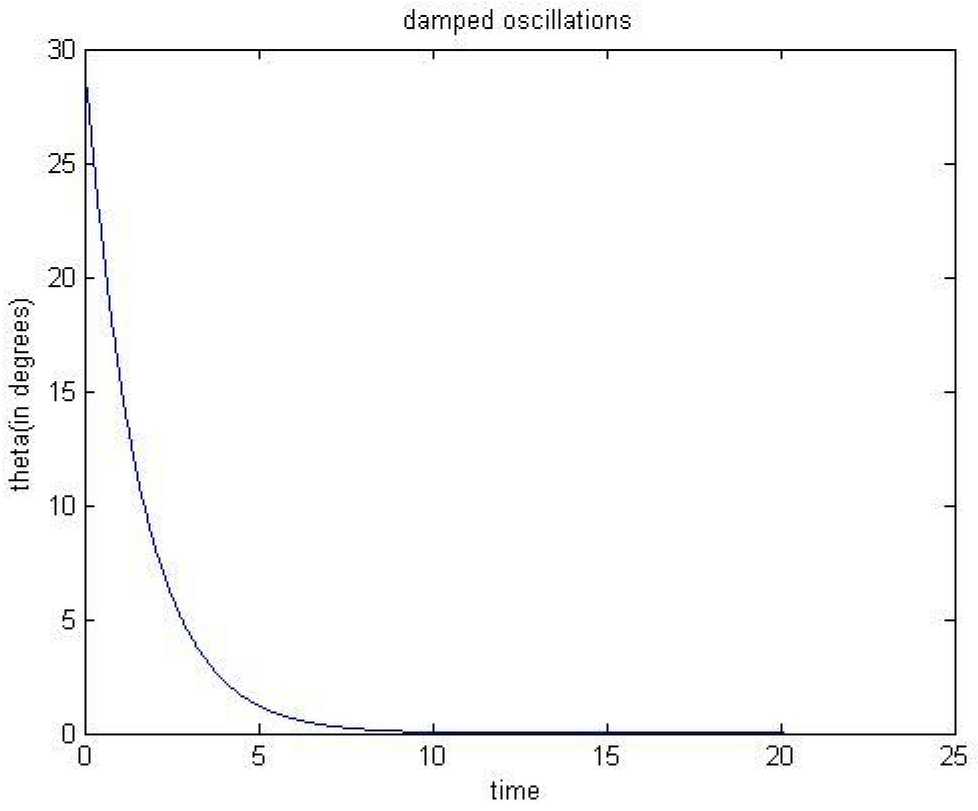
Looking at the phase space diagram we make the following observations:

Initially the pendulum seems to be accelerating. Then after a certain time it starts decelerating and the system starts losing energy. This is because, drag force is directly proportional to velocity. So, initially the gravitational force dominates and accelerates the pendulum. After some time, when velocity becomes large enough, the drag force starts dominating and hence we see that the velocity starts decreasing and eventually the pendulum comes to rest at the equilibrium position. Since, we are in the critical case, the velocity at the equilibrium point just becomes zero.

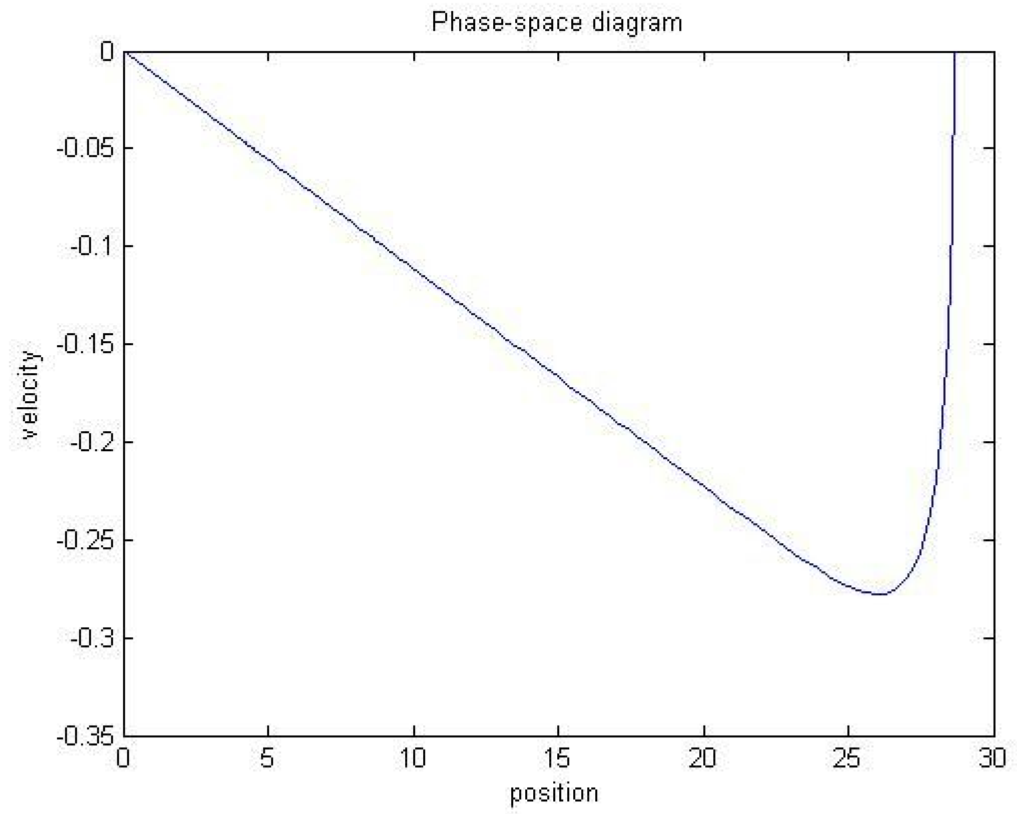
Overdamping Case:

Now, we consider the case when square of the damping coefficient B is greater than the square of the natural frequency. This is the case of overdamping.

beta = 6

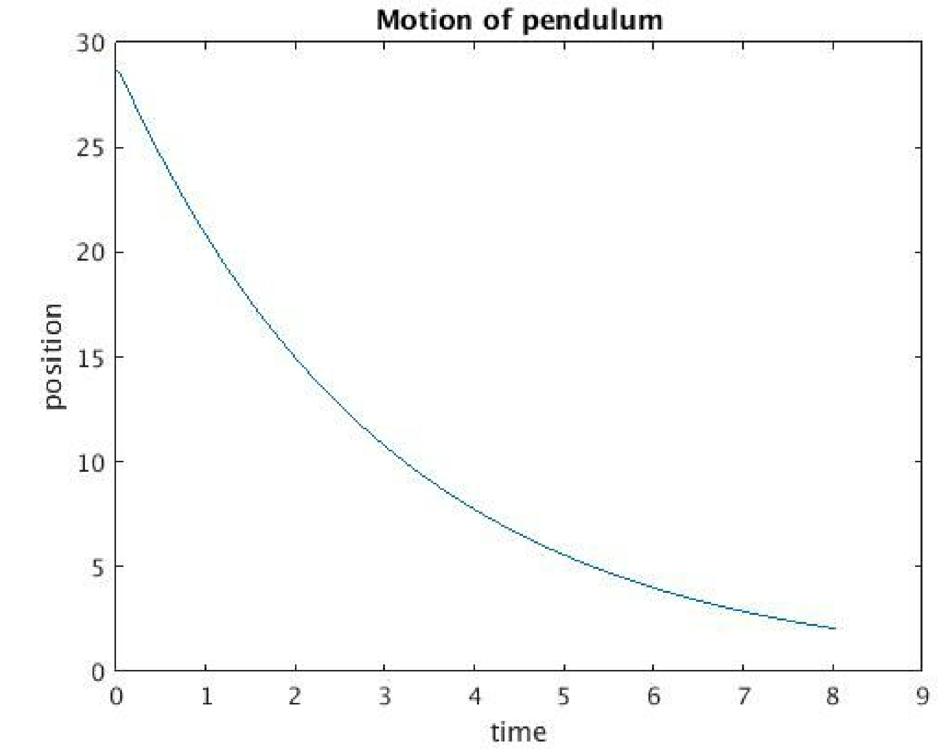


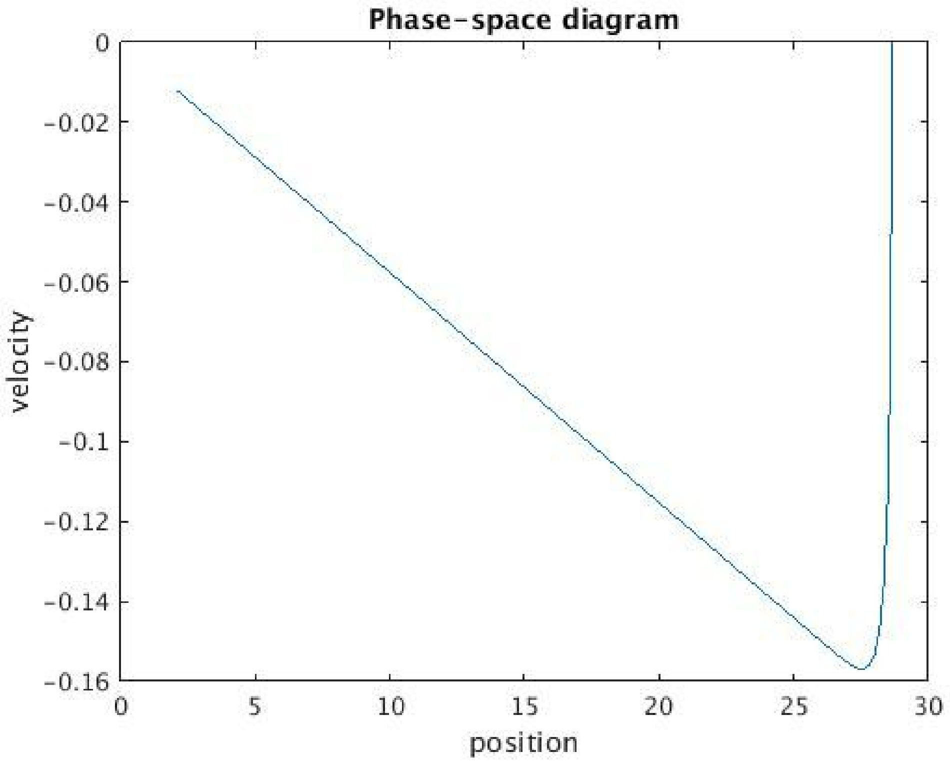
In the third case, we find a similar decaying motion but the pendulum comes to rest a little later. That is why it is called ‘over’ damped. Because if our objective is to make the particle come to rest by taking beta greater than critical case(just enough) we are overdoing the job.



Even the phase space behaviour is more or less the same. The point of inflection is achieved earlier because beta is higher and hence the more powerful drag force is able to overcome gravity earlier.

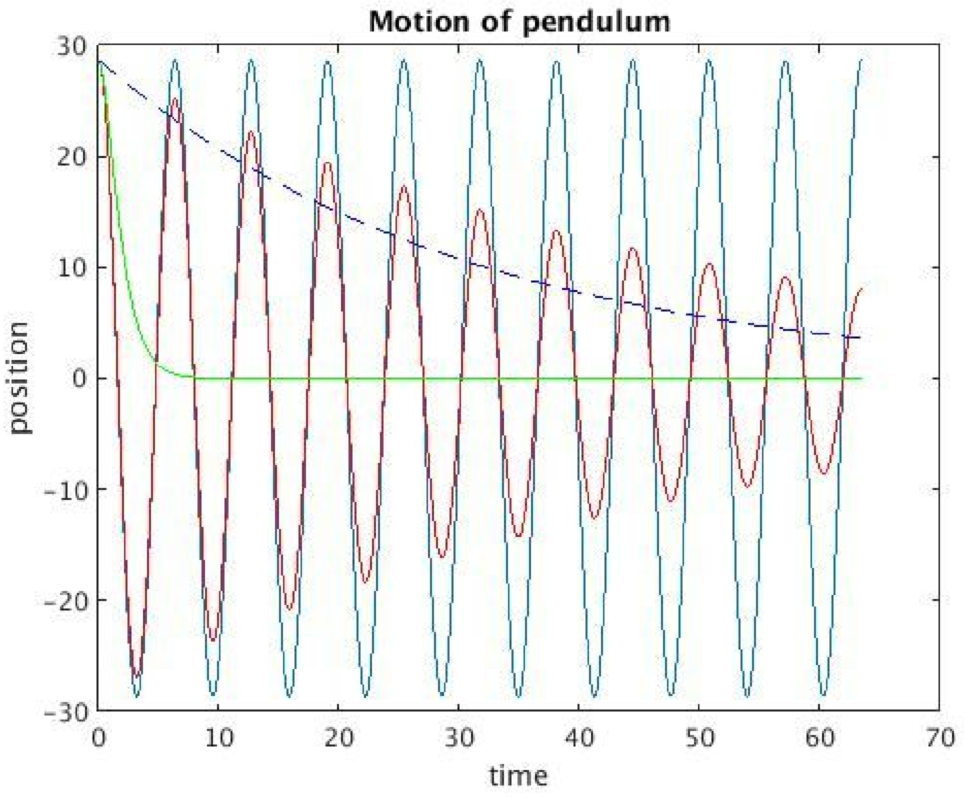
Effect of change in beta: As beta increases the total time taken by the pendulum to reach the equilibrium point increases.

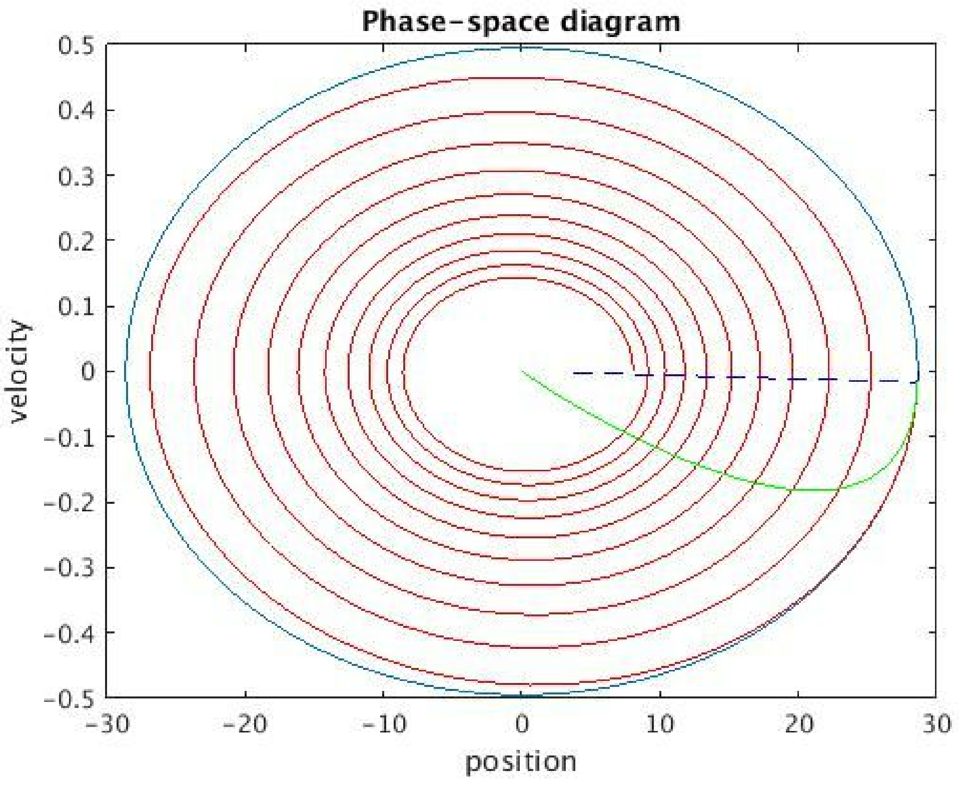

Motion of pendulum
position
time
0
5
10
15
20
25
30
0
1
2
3
4
5
6
7
8
9


Phase-space diagram
velocity
position
0
-0.02
-0.04
-0.06
-0.08
-0.1
-0.12
-0.14
-0.16
0
5
10
15
20
25
30

Here we have taken beta = 15. We can clearly see that total time to achieve equilibrium has been increased and the damping force has also been increased due to which the point of inflection has shifted rightwards.

Comparison of all the cases mentioned above along with the ideal pendulum motion case:





The first diagram indicates the motion of pendulum for different values of drag force.

The blue graph is the ideal case, The red graph is the underdamped case, the green curve is the critically damped case and the blue dashed line is the overdamped case.

The phase space diagram precisely shows the comparison of the motion of pendulum with respect to its velocity in all the cases.

Q. In which of the above cases the pendulum comes to the equilibrium position - the fastest i.e. minimum time to reach at rest?

In Critically damped case it takes least time. Critically damped is a special case where beta = w0^2. The sinusoidal part just goes away but at the same time damping is not so strong to dictate the motion. Hence particle quickly comes to rest at stable equilibrium. In both the other cases it takes more time to come to rest as observed.

Q2. :

MATLAB Code:

```
clear;

close all;

% declare the pendulam variables to be global and set it


g = 9.8;

l = 1;


global omega_not;

omega_not = sqrt(g/l);


b = 1;          % critical case 2*l*sqrt(g/l);

m = 1;

global beta;

beta = b/(2*m);


global f0;

f0 = 100;


global omega_dri;
```

```
omega_dri_step = 0.5;

omega_dri=0:omega_dri_step:(5*omega_not);



timescale = 2*pi*sqrt(l/g);

dt = timescale/100;


% set the initial and final times

tstart = 0;

tfinal = 4*timescale;


% set the initial conditions in the y0 column vector

u_init = zeros(2, 1);

u_init(1) = 0.5; % initial position theta;

u_init(2) = 0; % initial velocity



%computationally finding the maximum amplitude

amplitude_arr = zeros(ceil(5*omega_not/omega_dri_step),1);

pointer=1;


for omega_dri = 0:omega_dri_step:(5*omega_not)

% call ode solver

options=odeset('RelTol',1e-8);

[t,u]=ode45(@q2_ode_driven_damped, tstart:dt:tfinal, u_init, options);
```

```
%extract quan opf x(angular disp) ie u(1)
x = (180/pi)*u(:,1);


k=length(x);
temp_amp=abs(x(k)); %last val


while  k>380
%for each B loop freq find ampli. for each case find amp by exam form back
        if temp_amp < abs(x(k))
        temp_amp=abs(x(k));
        end
        k=k-1;


end


amplitude_arr(pointer)=temp_amp;
pointer=pointer+1;


end


omega_dri=0:omega_dri_step:(5*omega_not);
figure
plot(omega_dri,amplitude_arr)
title('Phase-space diagram')
```

```
xlabel('position')

ylabel('velocity')
```

ODE solver Function:

```
function F=q2_ode_driven_damped(t,u)

% function output =name(input)

% right-hand side function for Matlab's ODE solver,


% In our case we will use:

% u(1) -> theta

% u(2) -> omega


% declare the globals so its value

% set in the main script can be used here

global omega_not;

global beta;

global f0;

global omega_dri;


% make the column vector F filled with zeros

F = zeros(length(u),1);


% Now build the elements of F

%

% so the equation dx/dt=v means that F(1)=u(2)
```

F(1) = u(2);

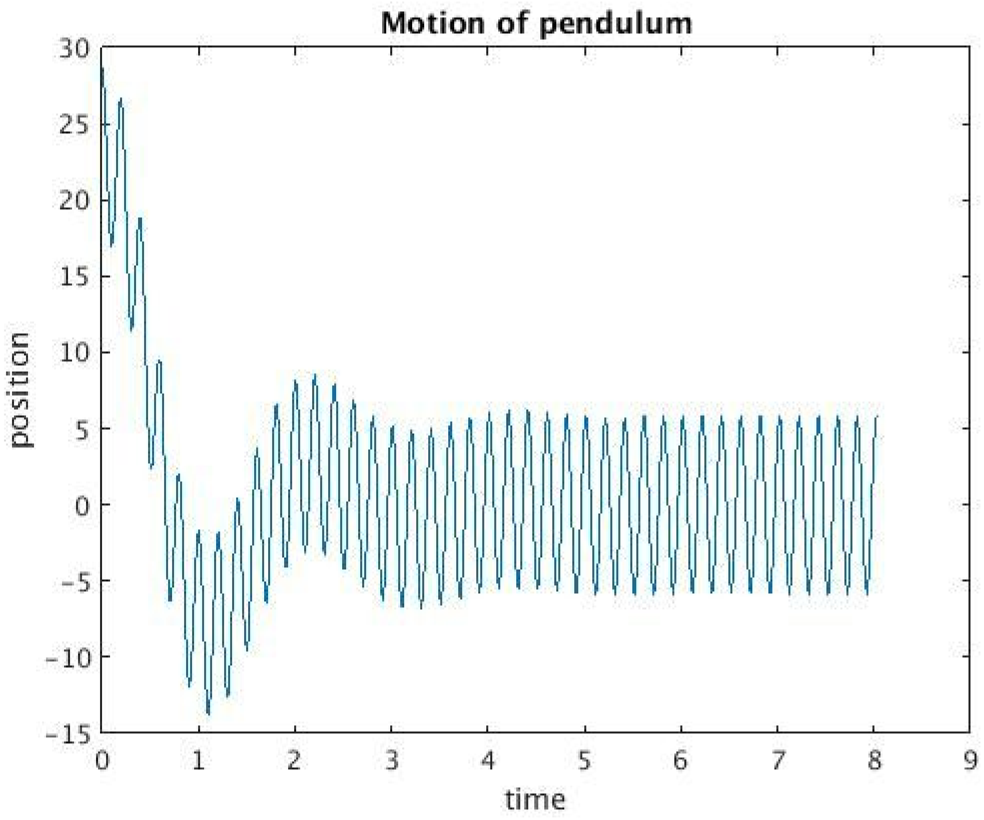
% Again, in our original ODEs we have:

% so the equation dv/dt=-....

F(2) = -omega_not*omega_not*u(1) - 2*beta*u(2) - f0*cos(omega_dri*t);

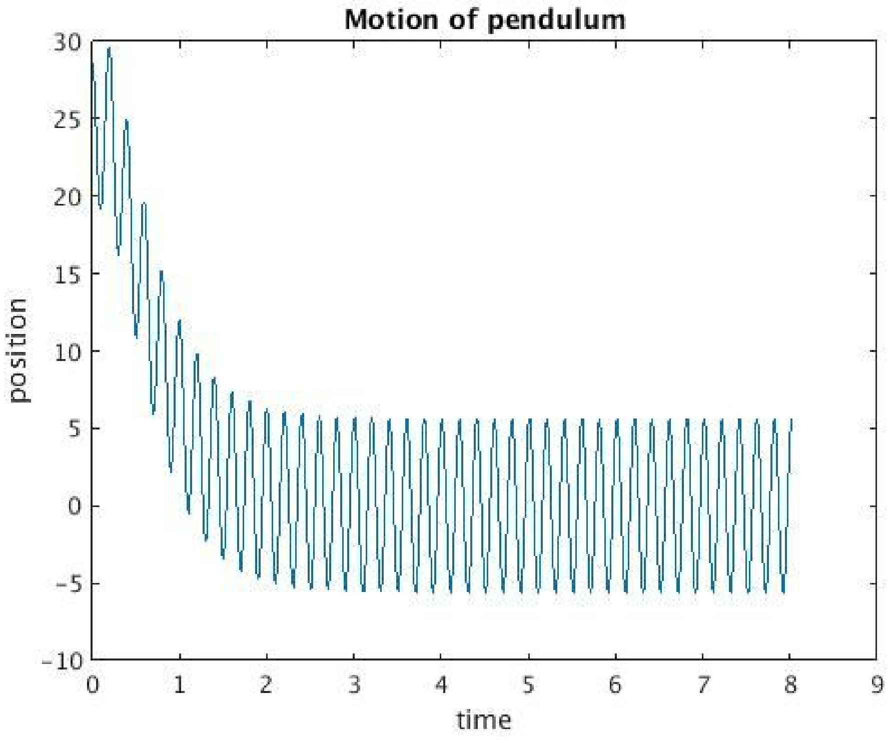
Comparing transient state with the steady state.

Under-damped motion:

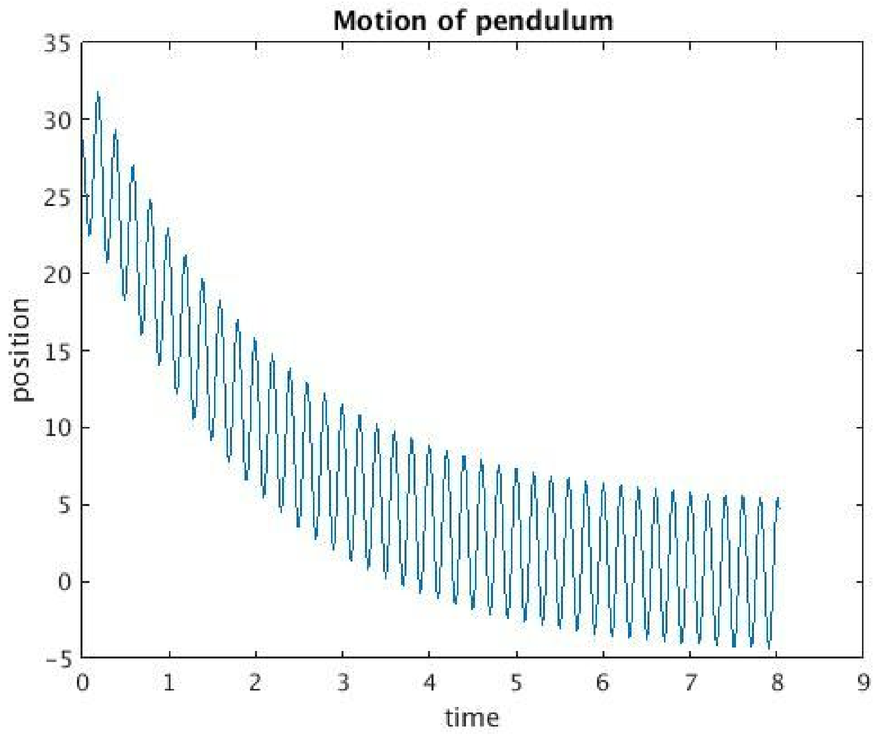


Initially the pendulum moves with its own frequency but as the energy dies down, it starts following the frequency of driving force.

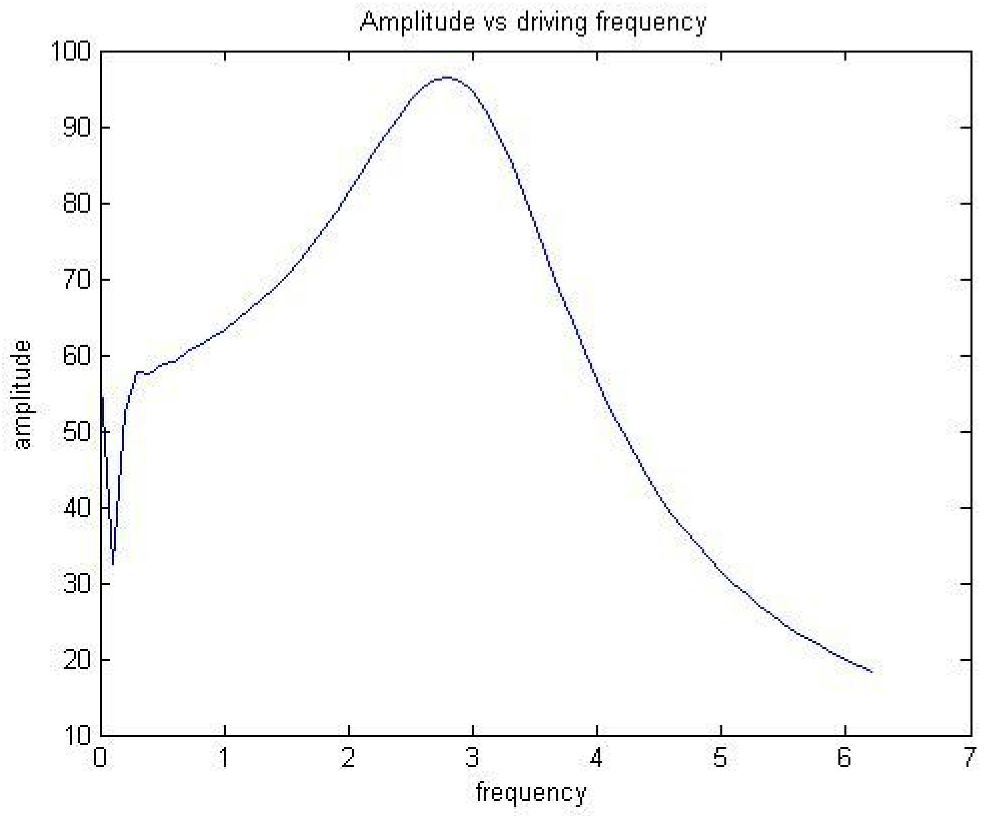
Critically Damped:


Motion of pendulum
position
time
30
25
20
15
10
5
0
-5
-10
0
1
2
3
4
5
6
7
8
9

Over damped:


Motion of pendulum
position
time
35
30
25
20
15
10
5
0
-5
0
1
2
3
4
5
6
7
8
9

In critically damped and overdamped case, the pendulum does not have its own frequency and hence it continues to oscillate with the frequency of the driving force when it reaches the steady state. Also, since in this case the pendulum does not have oscillations of its own, we expect the transition to the steady state to happen quicker than in the underdamped case.

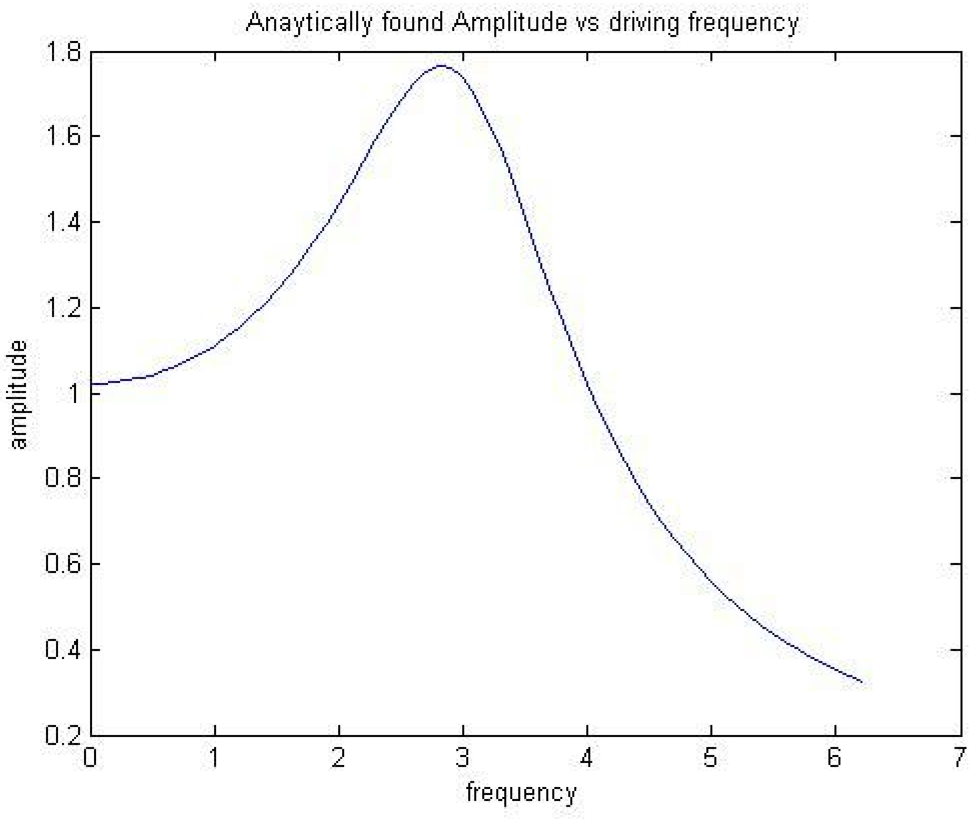


The question asks us to plot for particular beta, the variation of the amplitude of the driven system vs frequency of the driving force. We expect to see the phenomenon of resonance which means that when frequency of the driving force will come close to the natural frequency, the amplitude will become very high. The above is the plot, which we have plotted for beta=1, varying frequency from 0 to 2*natural_frequency.

Theoratically, the relation between amplitude of a driven, damped system and the driving frequency is given as follows.

```
amplitude _system=  f0./( sqrt( (omega_not*omega_not
-(omega_dri.*omega_dri)).*(omega_not*omega_not -(omega_dri.*omega_dri))
+4*beta*beta.*omega_dri.*omega_dri ) );
```

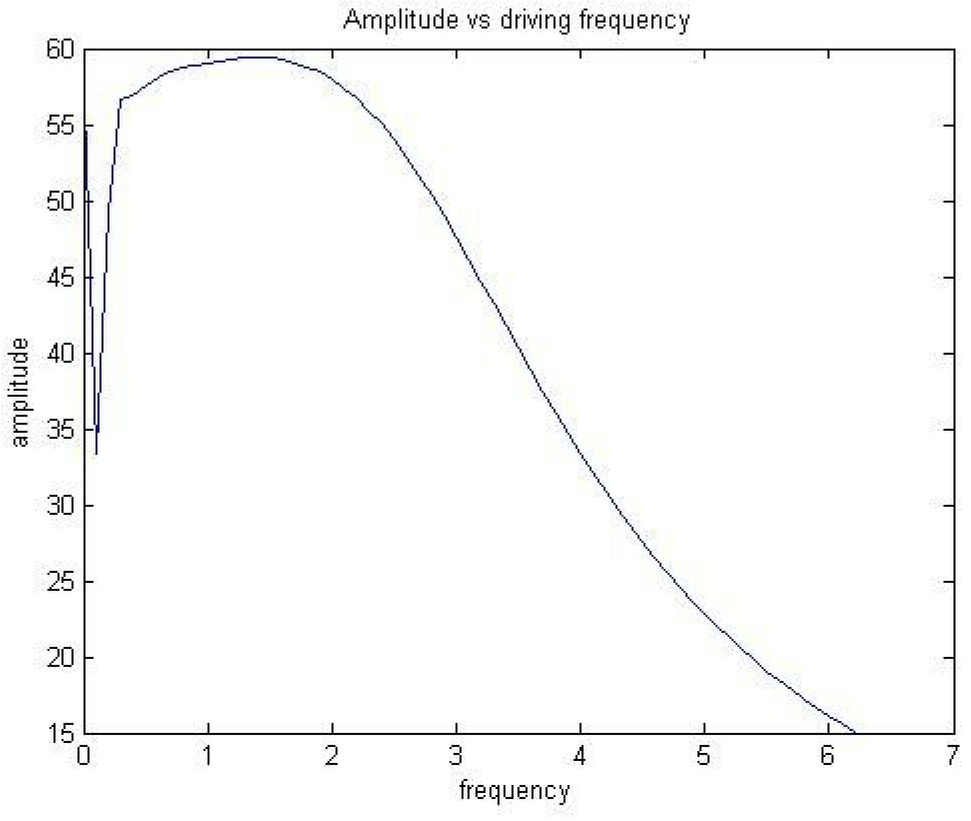
The analytic formula yields the following plot,



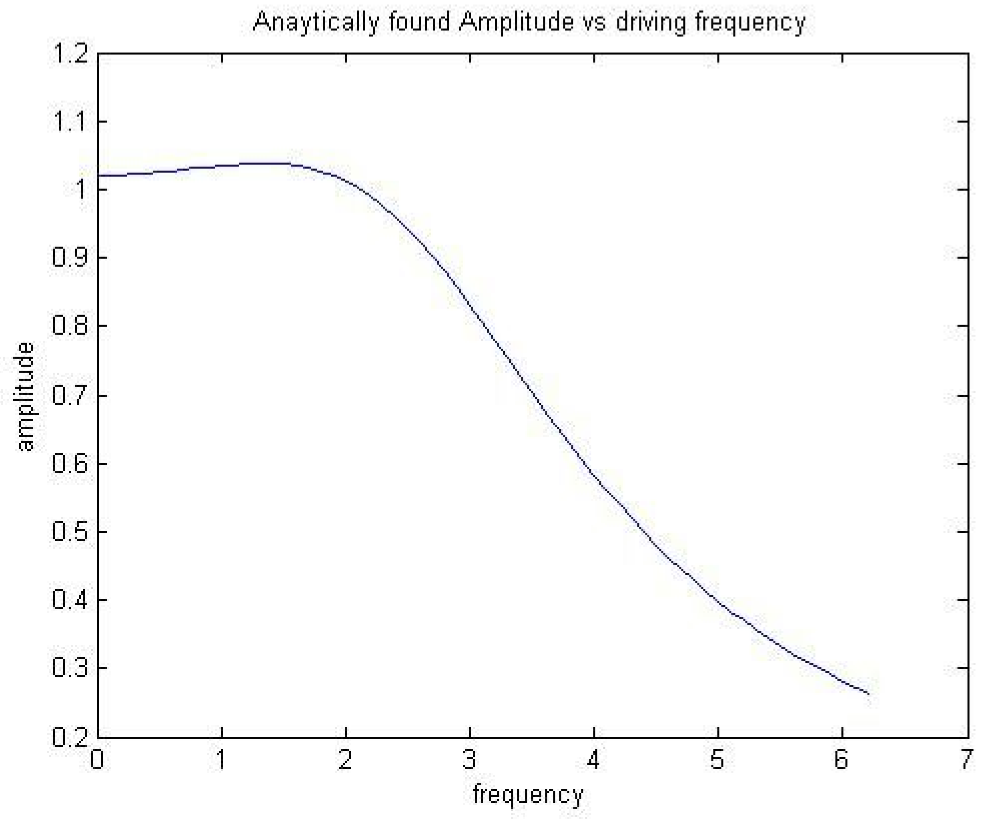
Notice that except for the kink in the initial part, the computational solution matches perfectly with the analytic solution. We observe that resonance is around 3 Hz. As frequency approaches natural frequency the amplitude of the system shoots up. It decreases on either sides.

As asked, we know repeat the same exercise for two other values of beta.

Consider the case for beta=2

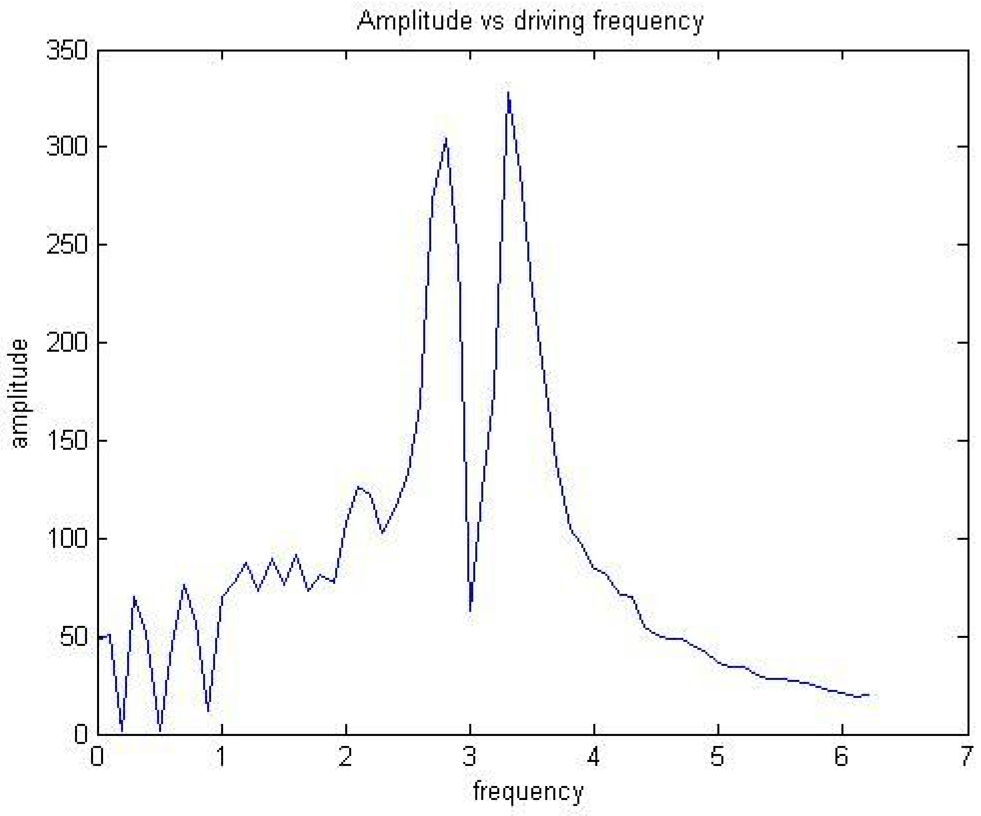


The corresponding analytical solution is



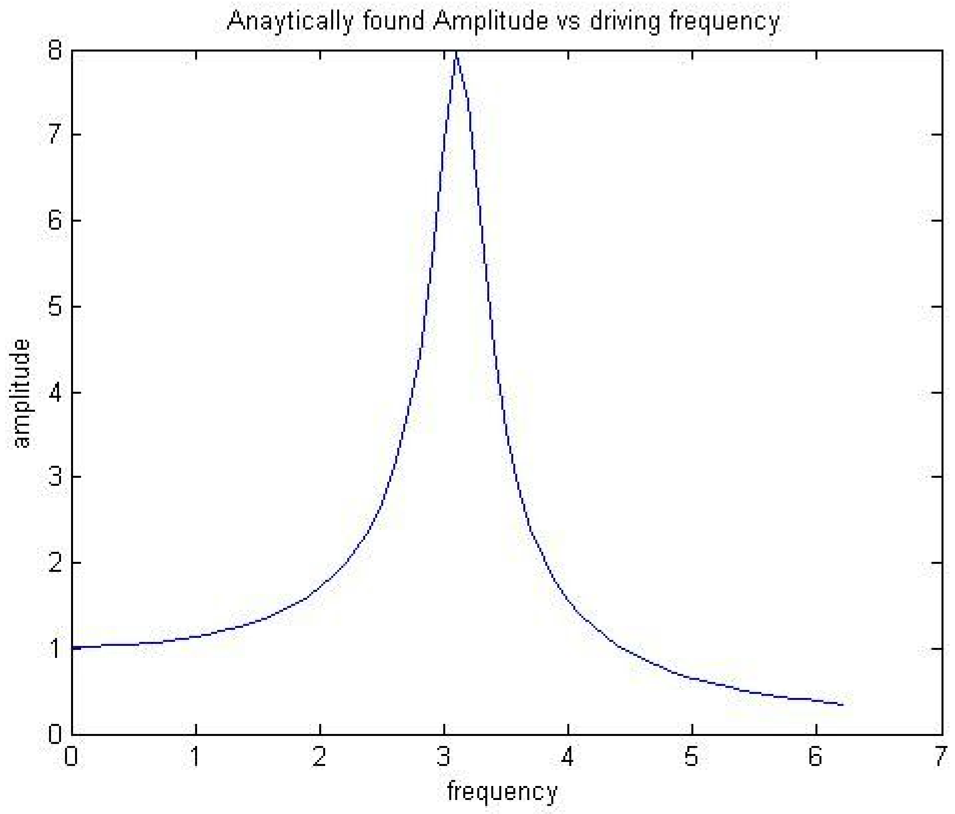
Even in this case the computational solution is giving remarkably good results. Notice that as the damping factor is greater, the curve is more flattened and hence more away from the ideal behaviour of shooting up at resonance.

Case when beta is 0.1 (low).



Notice that the plot has a lot of kinks but the general nature of the plot is quite similar to the analytical solution below.

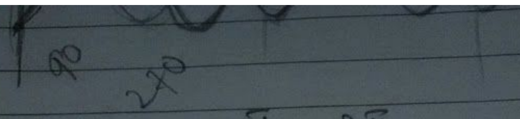
Analytic solution is:



The curve for low beta means that at resonance it is quite close to the ideal case of infinite amplitude at resonance. Hence we get a sharp spike around the resonance frequency.

Phase angle between the driving force and the resultant motion

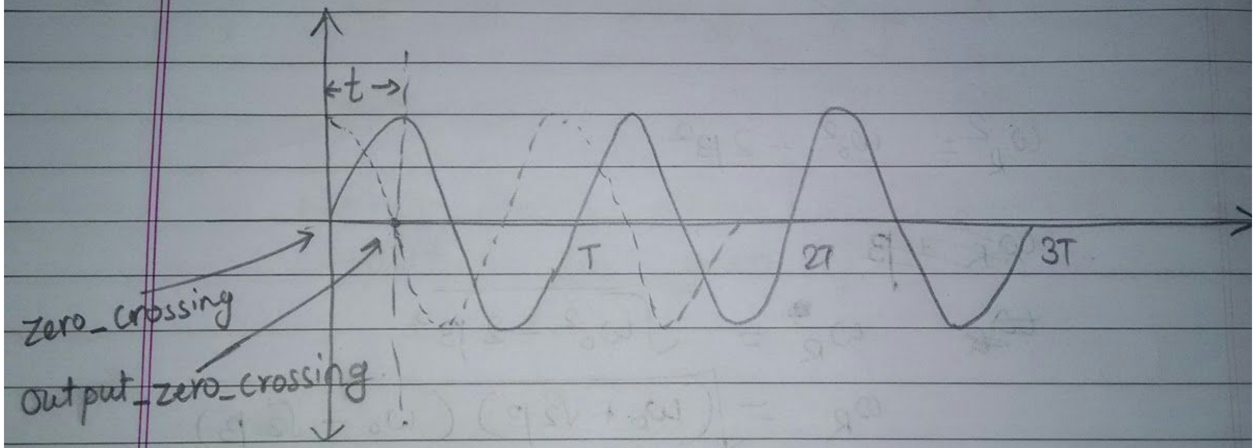
The principle used here is depicted by the following figure:



$$T = \frac{2\pi}{\omega}$$

$$\omega = 2\pi f$$

$$T = \frac{1}{\cancel{2\pi} f} = \frac{1}{\omega} 2\pi = \frac{2\pi}{\omega}$$



in time T = drive_f_time
it covers $\theta = 2\pi$
∴ in time t, it covers θ

MATLAB Code:

```
clear;

close all;

% declare the pendulam variables to be global and set it


g = 9.8;

l = 9.8;


global omega_not;

omega_not = sqrt(g/l);


b = 0.02;                    % critical case 2*l*sqrt(g/l);

m = 0.1;

global beta;

beta = b/(2*m);


global f0;

f0 = 1.5;


global omega_dri;


omega_dri_step = 0.1;

%omega_dri=0:omega_dri_step:(5*omega_not);
```

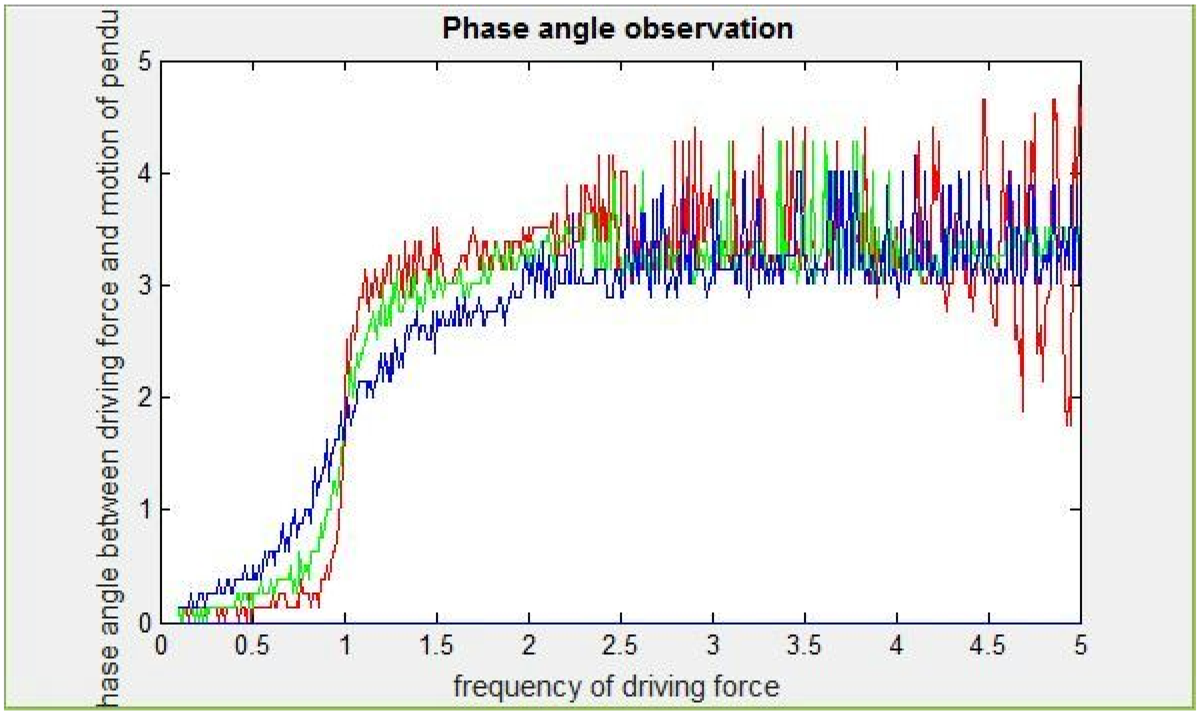
```
timescale = 2*pi*sqrt(l/g);
dt = timescale/100;

% set the initial and final times
%tstart = 0;



% set the initial conditions in the y0 column vector
u_init = zeros(2, 1);
u_init(1) = 0.5; % initial position theta radians;
u_init(2) = 0; % initial velocity

omega_arr = 0.1:omega_dri_step:(5*omega_not);
%computationally finding the phase difference
phase_arr = zeros(length(omega_arr));
pointer = 1;
%time_diff_zero_crossing = 0;

global zero_time; global drive_f_time_period; global tfinal;



for omega_dri = 0.1:omega_dri_step:(5*omega_not)
% call ode solver
drive_f_time_period = 2*pi/omega_dri;
```

```
tstart = 0;
tfinal = 50*drive_f_time_period;
dt = (tfinal-tstart)/500;

options=odeset('RelTol',1e-8);
[t,u]=ode45(@q2_ode_driven_damped, tstart:dt:tfinal, u_init, options);

time_zero_crossing = zero_time;
output_zero_crossing = 0;

drive_f_dt = drive_f_time_period/50;

for time = time_zero_crossing : drive_f_dt: (time_zero_crossing + drive_f_time_period)
        if( cos(omega_dri*time) > 0 && ( cos(omega_dri*(time+drive_f_dt) ) < 0) )
        output_zero_crossing = time;
        break;
        end
end

phase_arr(pointer) = ( (output_zero_crossing-time_zero_crossing) / drive_f_time_period) * 2*pi;

if(phase_arr(pointer) >= 2*pi)
        phase_arr(pointer) = phase_arr(pointer) - 2*pi;
end
```

```
pointer = pointer+1;

end

% plot the position vs. time
plot(omega_arr, phase_arr);
title('Phase angle observation')
xlabel('frequency of drivin force')
ylabel('phase angle between driving force and motion of pendulum')
```

ODE Solver function:

```
function F=q2_ode_driven_damped(t,u)
% function output =name(input)
% right-hand side function for Matlab's ODE solver,

% In our case we will use:
% u(1) -> theta
% u(2) -> omega

% declare the globals so its value
% set in the main script can be used here
global omega_not;
global beta;
global f0;
global omega_dri;

global zero_time; global drive_f_time_period; global tfinal;


% make the column vector F filled with zeros

F = zeros(length(u),1);


% Now build the elements of F

%

% so the equation dx/dt=v means that F(1)=u(2)

F(1) = u(2);

% Again, in our original ODEs we have:


% so the equation dv/dt=-....

F(2) = -omega_not*omega_not*u(1) - 2*beta*u(2) - f0*cos(omega_dri*t);



if( t > (tfinal - 5*drive_f_time_period) ) %because the oscillations stabilise during this time


if t-drive_f_time_period < 0.01

        zero_time = t;

end

end
```



The beta values are for the above plots are

Red : 0.03

Green : 0.1

Blue: 0.5

Note the behaviour of the graph closely resembles the graph gives in the book.