CS201

Name: omkar damle

ID : 201401114

Assignment-5

1. **Oscillations (SHM)**

   Study the effects of damping in the case of a pendulum by starting with some initial angular displacement say (theta=.5 radians) and study how the motion decays with time. Repeat the simulations for different values of damping constant and investigate its effect on the oscillations. What is the effect of different initial conditions on freq. of damped oscillations?

   Investigate all the cases - underdamped, overdamped and crticially damped oscillations as discussed in the class. Report about the choice of initial conditions which lead to the above mentioned cases. Plot your result in a single graph for 3 cases. In which of the above cases the pendulum comes to the equilibrium position - the fastest i.e. minimum time to reach at rest.

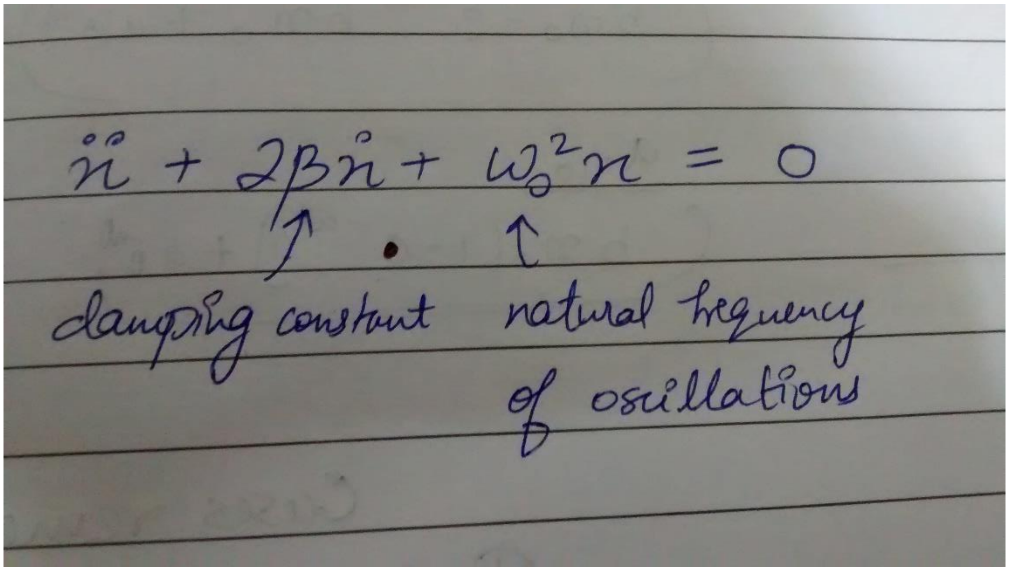
   Plot Phase space plot for all the three cases and explain the nature of the curve.

   For analytical solution and initial condition, you can refer to problems 3.2, 3.3 (Marion and Thornton)

   You can also take the case of Spring-Mass system instead of a pendulum for your investigations. For the case/code discussed in the class today you can start with the following parameters (initial v=0; initial displacement=100; w0=1, beta=2)

   Solution:

   The damped pendulum problem is a very standard problem in the theory of oscillations. Here I will be attempting to convert the pendulum problem to a more general damping problem which is applicable to other systems like the spring mass system. The equation common to all damping problems is :

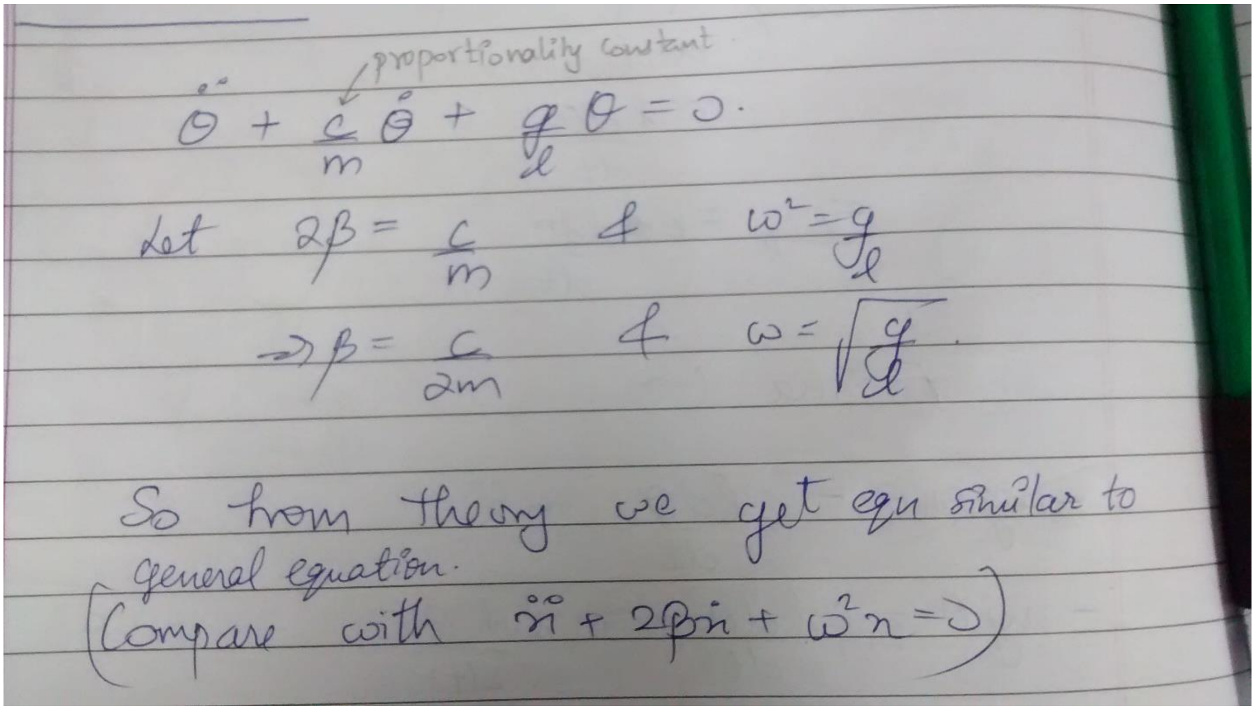


$$\ddot{x} + 2\beta\dot{x} + \omega_0^2 x = 0$$

damping constant

natural frequency of oscillations

For a pendulum we get the following equation which can be easily converted to the general equation –



proportionality constant

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

Let $2\beta = \frac{c}{m}$ & $\omega^2 = \frac{g}{l}$

$\Rightarrow \beta = \frac{c}{2m}$ & $\omega = \sqrt{\frac{g}{l}}$.

So from theory we get eqn similar to general equation.

(Compare with $\ddot{x} + 2\beta\dot{x} + \omega^2 x = 0$)

Now we can

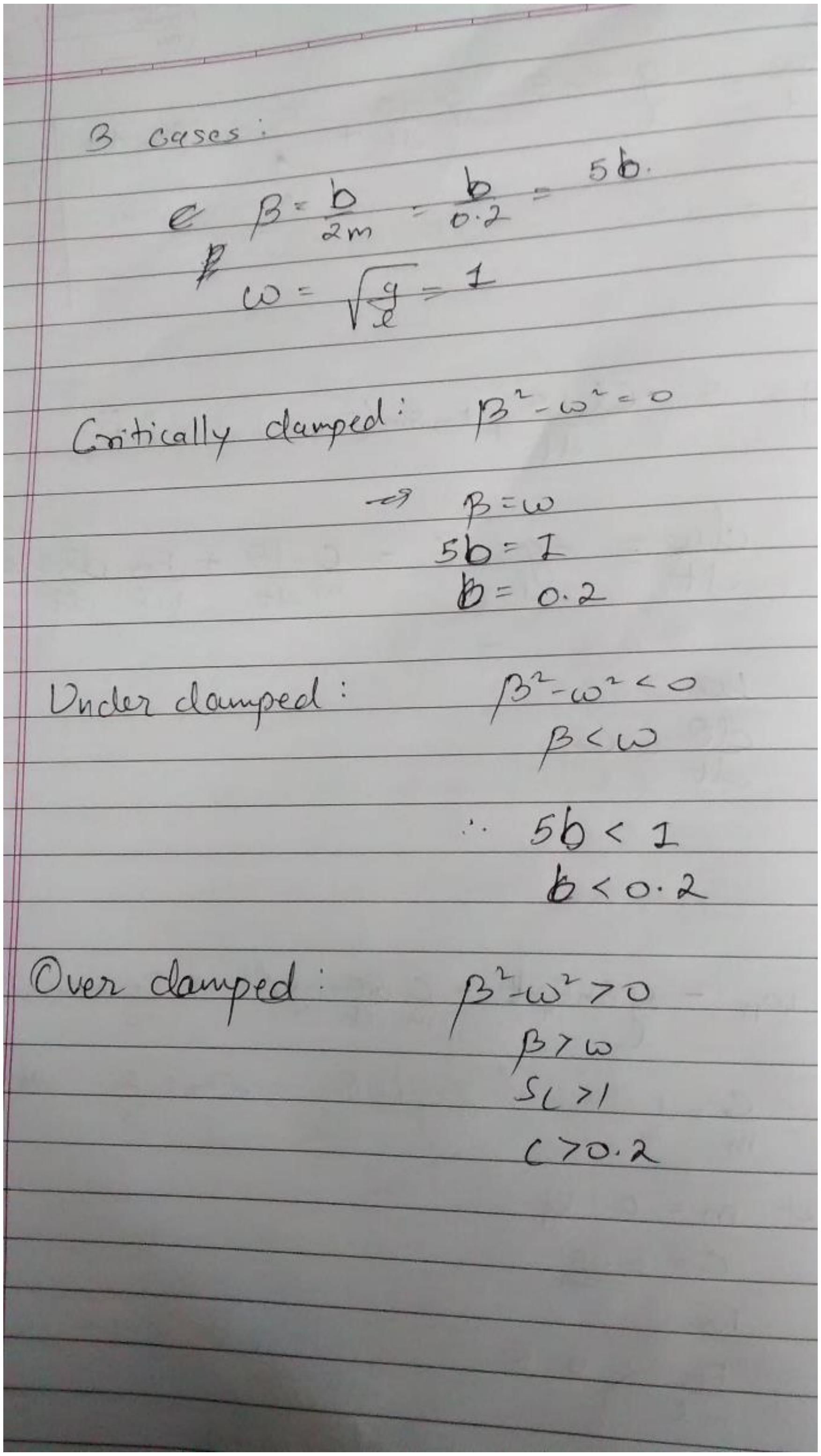
The values of the constants used are:

Mass of pendulum(m) = 0.1 Kg

Drag force constant (b or c) = 0.2 (for critical damping)
Length = 9.8 m
g = 9.8 m/s^2
initial angular displacement = 10 degrees.

Hence the natural angular frequency of oscillation(omega) is 1 which simplifies the calculations.

Three cases arise if we change the proportionality constant and keep all others fixed. (Physically we have the same pendulum with the same g and we are conducting the experiments in different environments where only the drag force is changing.)



3 cases:

$e \quad \beta = \frac{b}{2m} = \frac{b}{0.2} = 5b.$

$\cancel{\beta} \quad \omega = \sqrt{\frac{g}{l}} = 1$

Critically damped: $\beta^2 - \omega^2 = 0$

$\rightarrow \beta = \omega$

$5b = 1$

$b = 0.2$

Under damped: $\beta^2 - \omega^2 < 0$

$\beta < \omega$

$\therefore 5b < 1$

$b < 0.2$

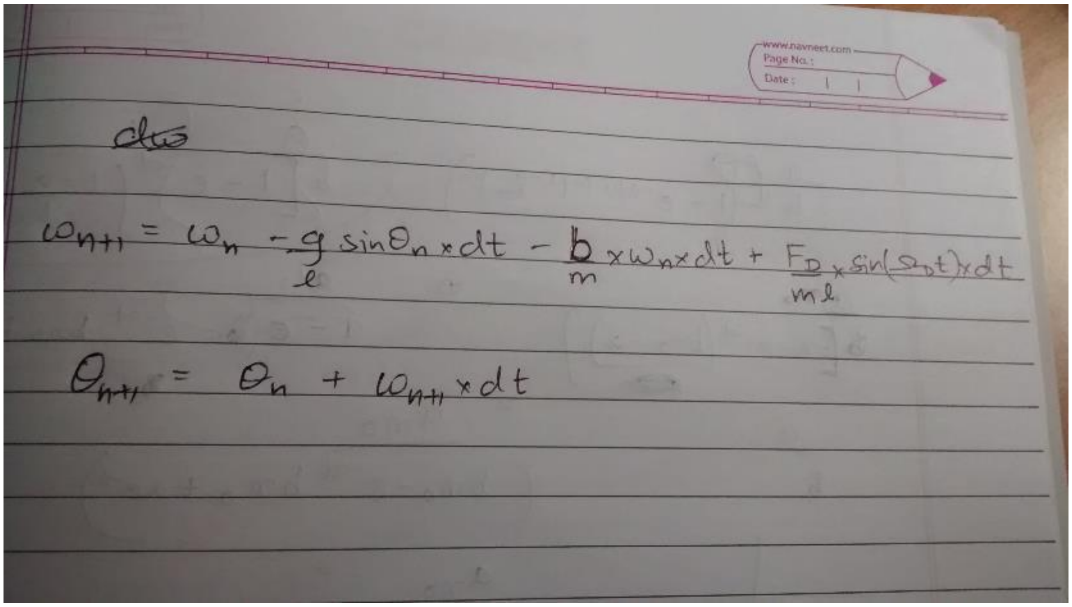
Over damped: $\beta^2 - \omega^2 > 0$

$\beta > \omega$

$5c > 1$

$c > 0.2$

The equations used are :


www.navneet.com
Page No.:
Date:

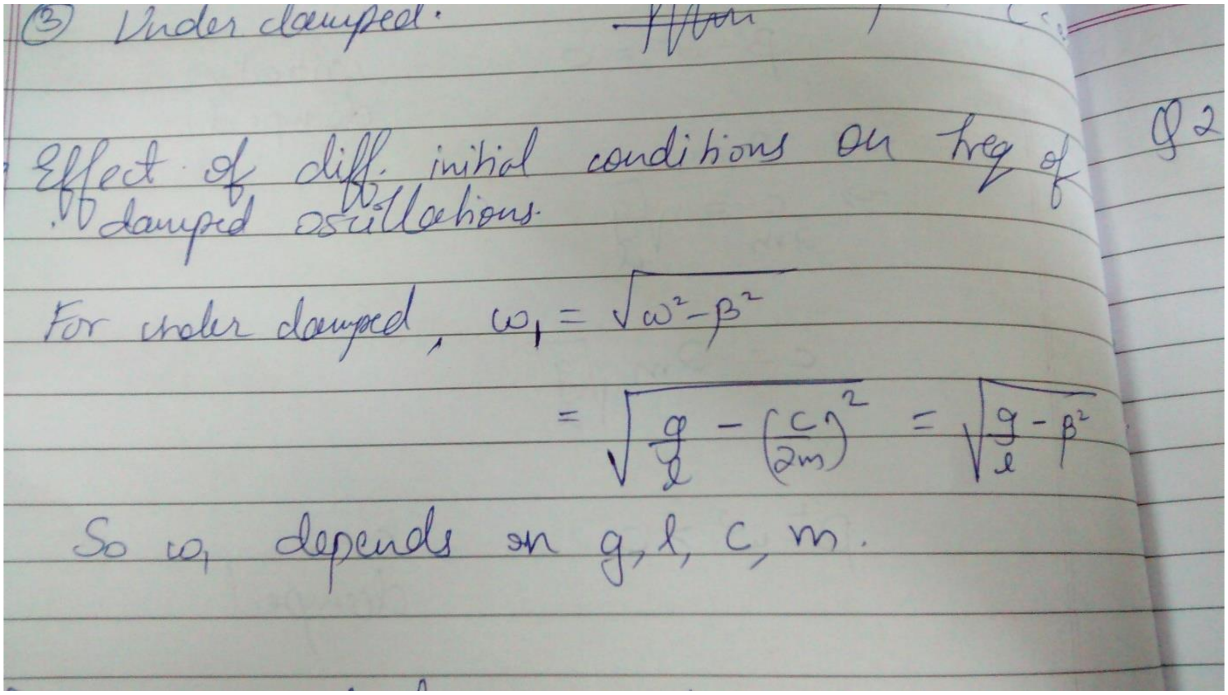
dω

$$\omega_{n+1} = \omega_n - \frac{g}{l}\sin\theta_n \times dt - \frac{b}{m} \times \omega_n \times dt + \frac{F_D}{ml} \times \sin(\Omega_D t) \times dt$$

$$\theta_{n+1} = \theta_n + \omega_{n+1} \times dt$$

1. **Under Damping**: The oscillations die down gradually. (b = 0.01 or beta = 0.05 <1)

   An interesting observation is that the frequency of damped oscillations is not the same as that of the natural frequency. It is in fact less than the natural frequency. The formula is:



③ Under damped.

Q2

Effect of diff. initial conditions on freq of damped oscillations.

For under damped, $\omega_1 = \sqrt{\omega^2 - \beta^2}$

$$= \sqrt{\frac{g}{l} - \left(\frac{c}{2m}\right)^2} = \sqrt{\frac{g}{l} - \beta^2}$$

So ω_1 depends on g, l, c, m.

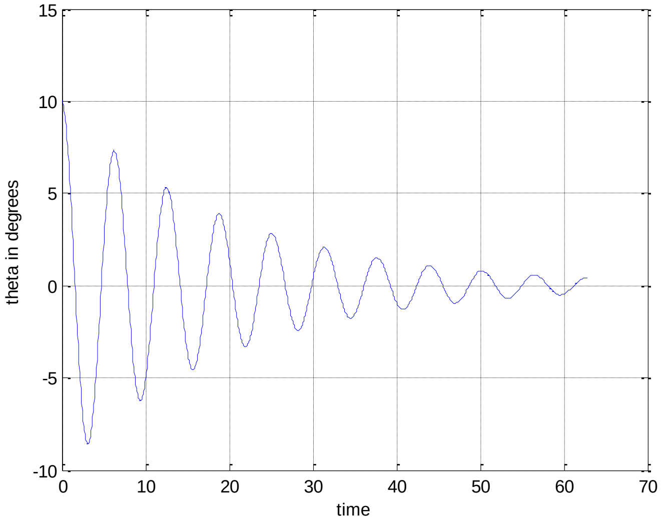
The frequency decreases from 1 to 0.99875 and the time period increases from 6.283(natural time period) to 6.291 for underdamped oscillations.

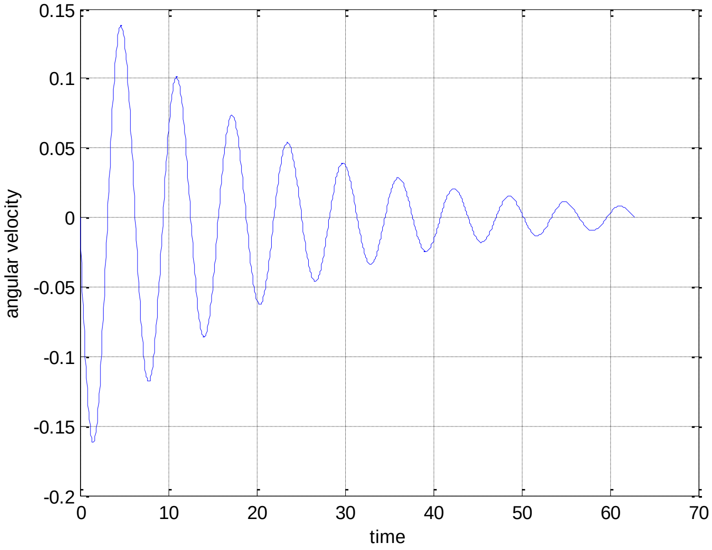
Computationally, the under damped oscillations time period comes out to be 6.2895 which is very close to 6.291
Computationally the under damped angular frequency comes out to be 0.9990 which is analytically 0.99875

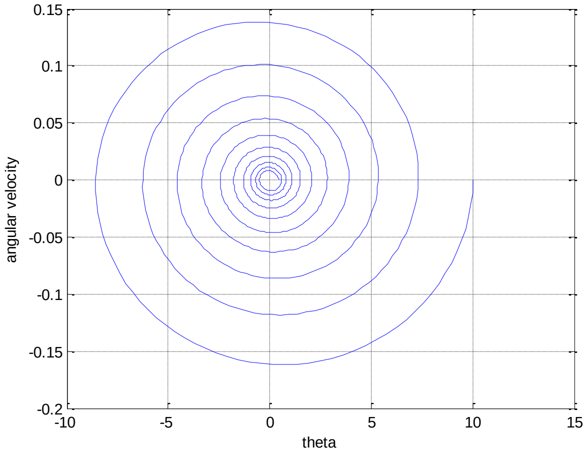
Code fraction used to calculate frequency of underdamped oscillations:

```
%we are calculating frequency of underdamped oscillations
%how?? - we calculate 2 consecutive zero crossings. The difference between
%the 2 times gives us half the time period. From that we calculate the
%angular frequency.
count = 0;
index = 1;
zero_crossing_time_1 = 0;
zero_crossing_time_2 = 0;

while count < 2

    if(count == 0 && (theta1(index)*theta1(index + 1)) <= 0  )
        zero_crossing_time_1 = t(index) ;
         count = count + 1;
         index = index + 1;
         continue;
    end

    if(count == 1 && (theta1(index)*theta1(index + 1)) <= 0  )
        zero_crossing_time_2 = t(index) ;
        count = count + 1;
        break;
    end

    index = index + 1 ;
end

%now lets calculate frequency!

time_period_underdamped_oscillations =
2*(zero_crossing_time_2 - zero_crossing_time_1)

angular_frequency_underdamped_oscillations =
2*pi*(1/time_period_underdamped_oscillations)
```

Graphs:


15
10
5
0
-5
-10
theta in degrees
0
10
20
30
40
50
60
70
time

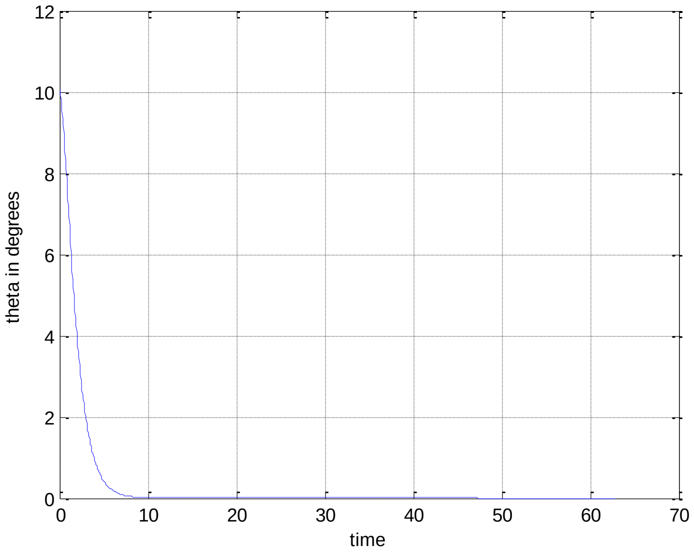

0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
angular velocity
0
10
20
30
40
50
60
70
time

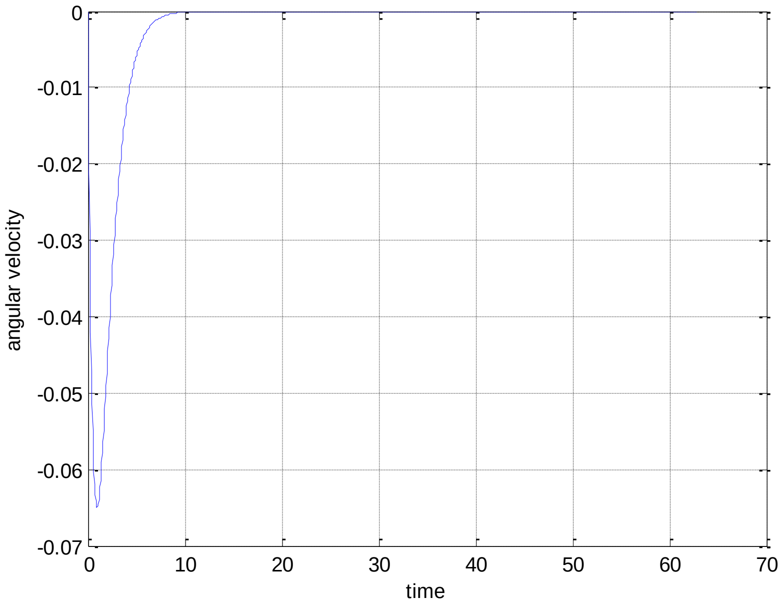


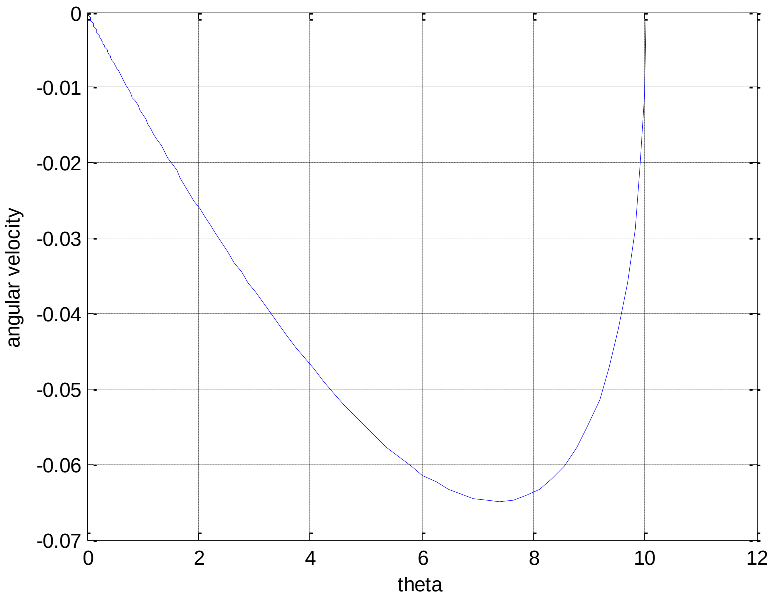
Explanation: By observing the phase space plot, we verify our initial observations about the decaying amplitude of angular displacement and velocity. As time increases, both the amplitudes die down and hence we get a spiral shape in the clockwise direction(which is characteristic of SHM).

2. Critical Damping :

No oscillations are seen. Instead the angle theta goes to 0 almost immediately.(b = 0.2 or beta = 1)

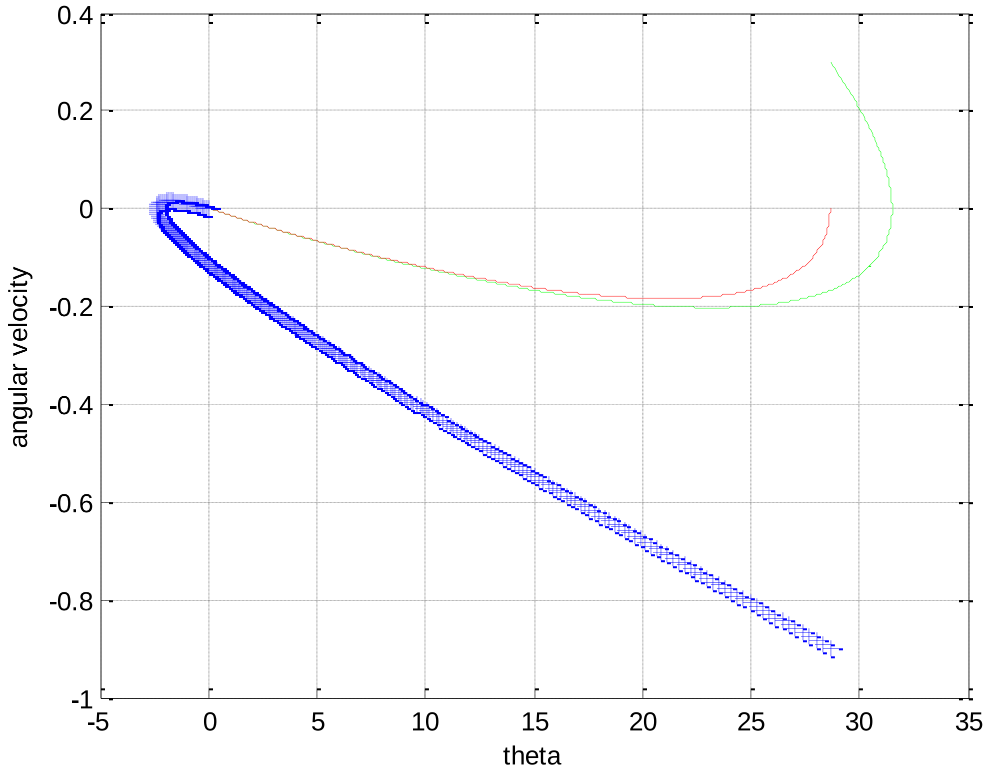






Explanation : Note that the phase space plot starts at theta = 10 and omega = 0 point. Then theta decreases rapidly and angular velocity first increases and then decreases rapidly. Finally the pendulum comes to rest.

If we plot the phase plots for 3 cases of critically damping in which the initial velocities differ we get the following observations:



All plots are for critically damped case. Only the **initial angular velocities are changed**. The initial position is constant for all 3 which is – 10 degrees.

Green – omega_initial = 0.3

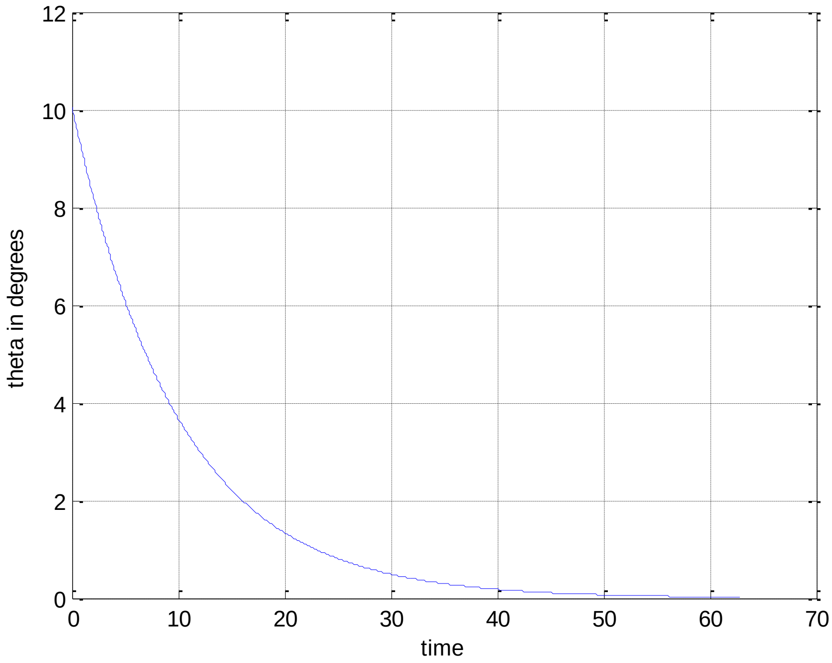
Red -- omega_initial = 0

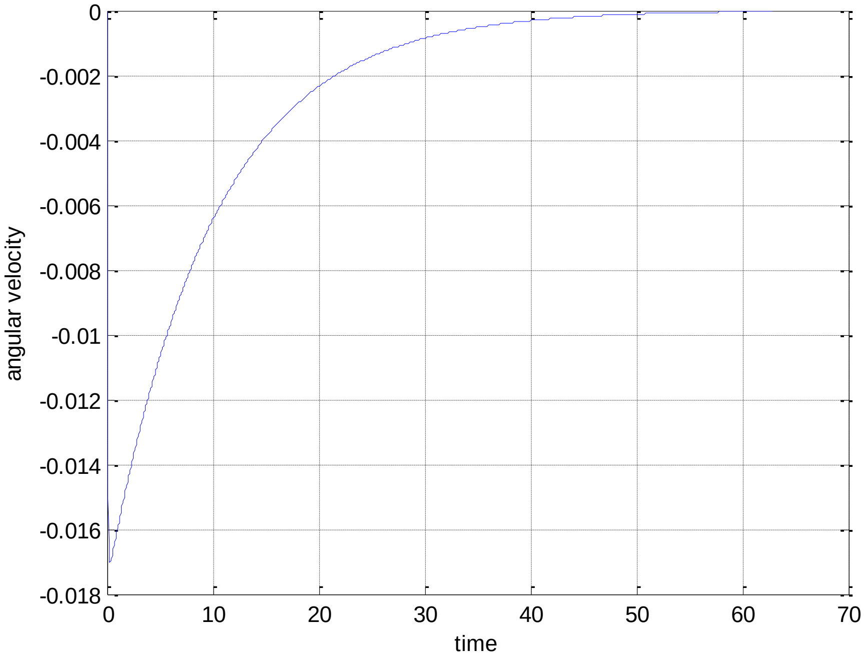
Blue -- omega_initial = -0.9

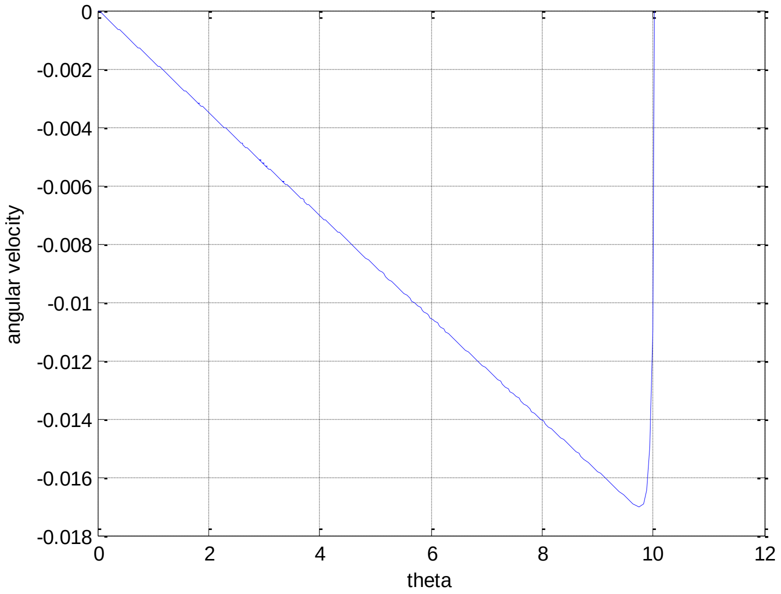
Notice how the pendulum goes to the left of the equilibrium position before coming to rest in the 3rd case.

3. Over Damping :

The angle goes down to 0 exponentially. (b = 1 or beta = 5 >1)

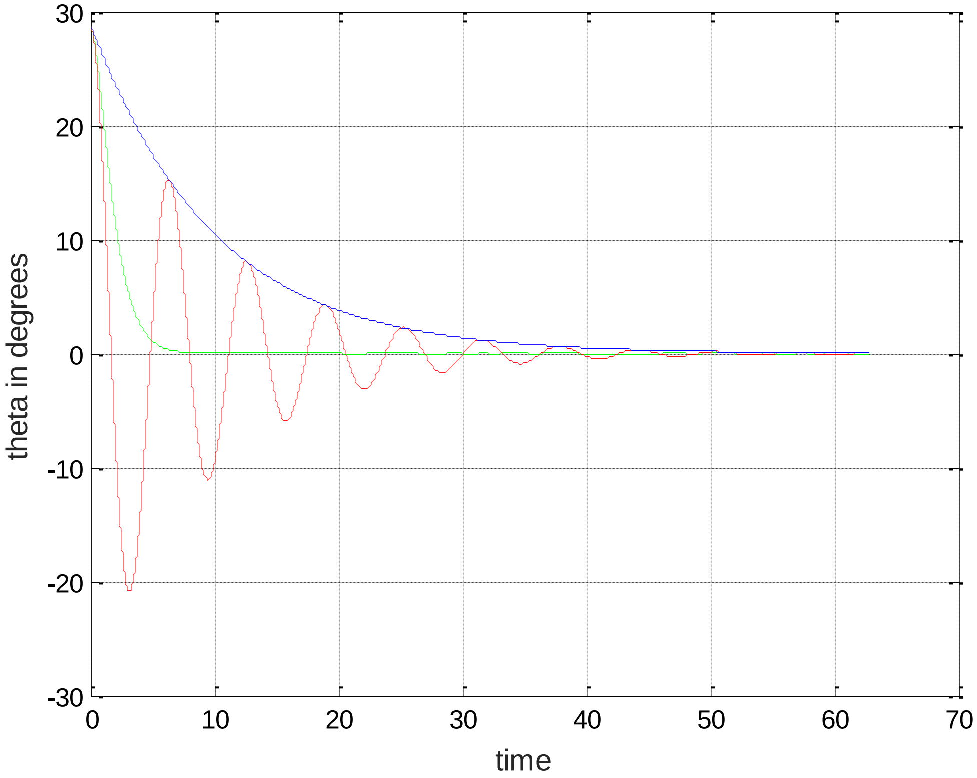






Explanation : Note that the phase space plot starts at theta = 10 and omega = 0 point. Then theta decreases and angular velocity first increases and then decreases. Finally the pendulum comes to rest. Though this takes more time as compared to the critical damping case.

If we plot all the 3 cases in a single graph,



Red – underdamped oscillations

Green – critically damped motion

Blue – Overdamped motion.

The critically damped motion comes to the equilibrium position in the least time i.e. fastest. This is because its period of decay is very small as compared to its period of oscillations.

Main Code

```
% this script is used to computationally observe the damped oscillator and
% its 3 different cases of underdamped, critically damped and over damped.


clear all;
close all;

global C mass l g beta;

g=9.8;
l=9.8; % so that g/l is 1
```

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

mass = 0.1;

%C = 0.05 % C is the drag constant Fdamp = C*length*omega

dt=timescale/1000;
simulationt=10*timescale;
start_time = 0;

%{
npoints=round(simulationt/dt);
theta=zeros(npoints,1);
omega=zeros(npoints,1);
time=zeros(npoints,1);
%}

theta_degrees_initial = 0.5*180/pi; %given 0.5 radians
omega_initial = 0.3;
%theta_degrees = 10 ;
%theta(1)= theta_degrees * pi / 180;
%using ode solver
%give initial conditions
u0 = zeros(2,1);
u0(1) = theta_degrees_initial*pi/180;
u0(2) = omega_initial;

C = 0.2;
beta = C/(2*mass);
%for positive angular velocity
[t,u] = ode45(@dampingFunction , [start_time : dt : simulationt] , u0);

theta1=u(:,1)*180/pi;
omega1 = u(:,2);

%C = 0.02;
%beta = C/(2*mass);
%for zero angular velocity
u0(2) = 0;
[t,u] = ode45(@dampingFunction , [start_time : dt : simulationt] , u0);

theta2=u(:,1)*180/pi;
omega2 = u(:,2);

%C = 1;
%beta = C/(2*mass);
```

```
%for negative angular velocity
u0(2) = -0.9;
[t,u] = ode45(@dampingFunction , [start_time : dt : simulationt] , u0);

theta3=u(:,1)*180/pi;
omega3 = u(:,2);

plot(t,theta1,'g',t,theta2,'r',t,theta3,'b')
xlabel('time')
ylabel('theta in degrees')
grid on;
figure
plot(t,omega1,'g',t,omega2,'r',t,omega3,'b')
xlabel('time')
ylabel('angular velocity')
grid on;
figure
plot(theta1,omega1,'g',theta2,omega2,'r',theta3 ,omega3 , 'b+')
xlabel('theta')
ylabel('angular velocity')
grid on;
```

Damping Function code -

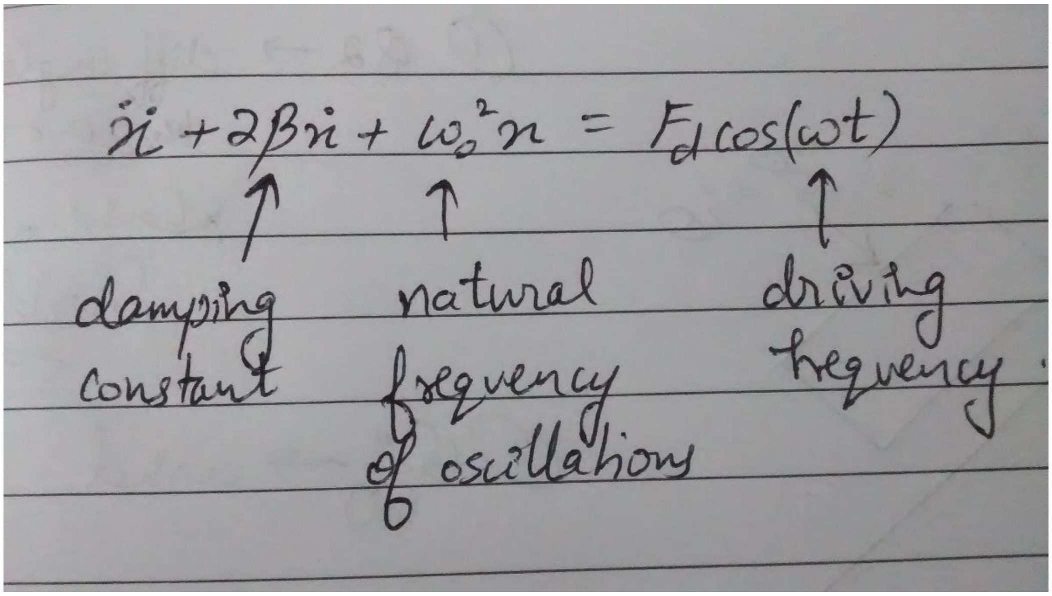
```
function F = dampingFunction( t , u )
%DAMPINGFUNCTION Summary of this function goes here

%{
u(1) -> theta
u(2) -> omega

F(1) -> omega
F(2) -> derivative of omega
%}

global C mass l g beta flag1 flag2 ;

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

F(1) = u(2);
F(2) = -(2*beta)*u(2) - (g/l)*u(1);


End
```

2. Driven (damped) Oscillator: Investigate the above problem (particularly the under damped case) with an external force with a given freq as discussed in the class. Compare the initial/transient behavior with the steady state behavior.
You can also take the case of Spring-Mass system instead of a pendulum for your investigations. [For the case/code discussed in the class today you may start with the following parameters (initial v=0; initial displacement=100; w0=1, beta=.5, F=1000; wd=10)

Generate a graph for frequency (driving frew. omega) vs. amplitude of the oscillating system for different values of β (beta) (scan over freq. with someΔω, starting from ω=0 to ω=5ω0). Note down where is the maximum amplitude for a given β. (i.e. computationally reproducing the figure 3.16 from Marion and Thronton using "for" loop, we are interested in stady state solution). Also investigate the phase angle between the driving force and the resultant motion computationally, supported by figure.

Also refer problem 3.24 (Marion and Thronton) for further investigations.

Solution: Here again I will be attempting to convert the pendulum problem to a more general damp-driven problem which is applicable to other systems like the spring mass system. The equation common to all damped driven problems is :



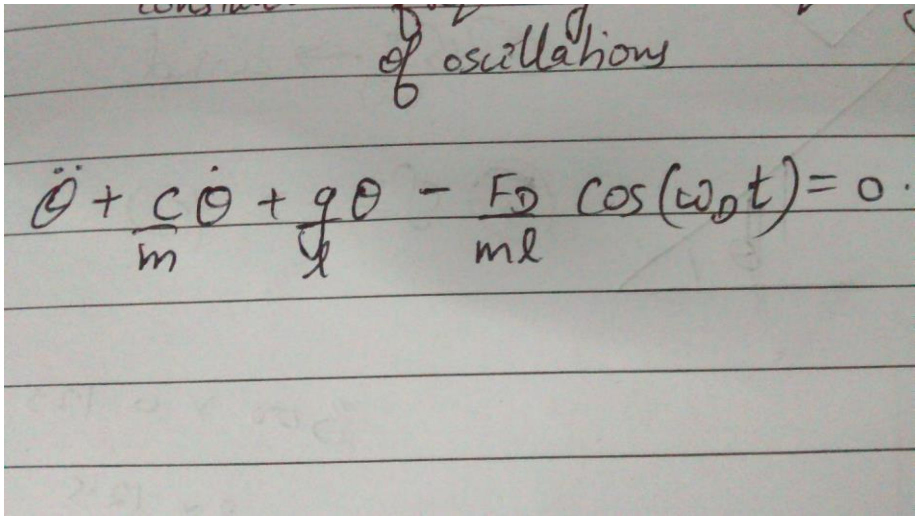
$$\ddot{x} + 2\underset{\uparrow}{\beta}\dot{x} + \underset{\uparrow}{\omega_0^2} x = F_d\cos(\underset{\uparrow}{\omega} t)$$

damping constant

natural frequency of oscillations

driving frequency.

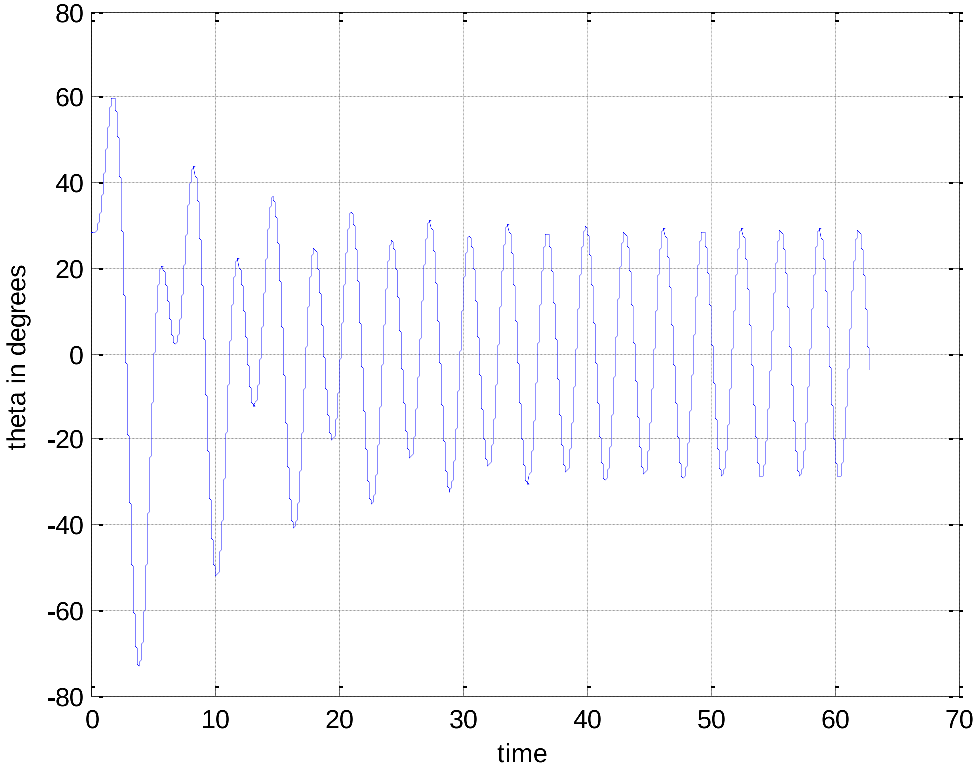
For the case of pendulum the equations are :

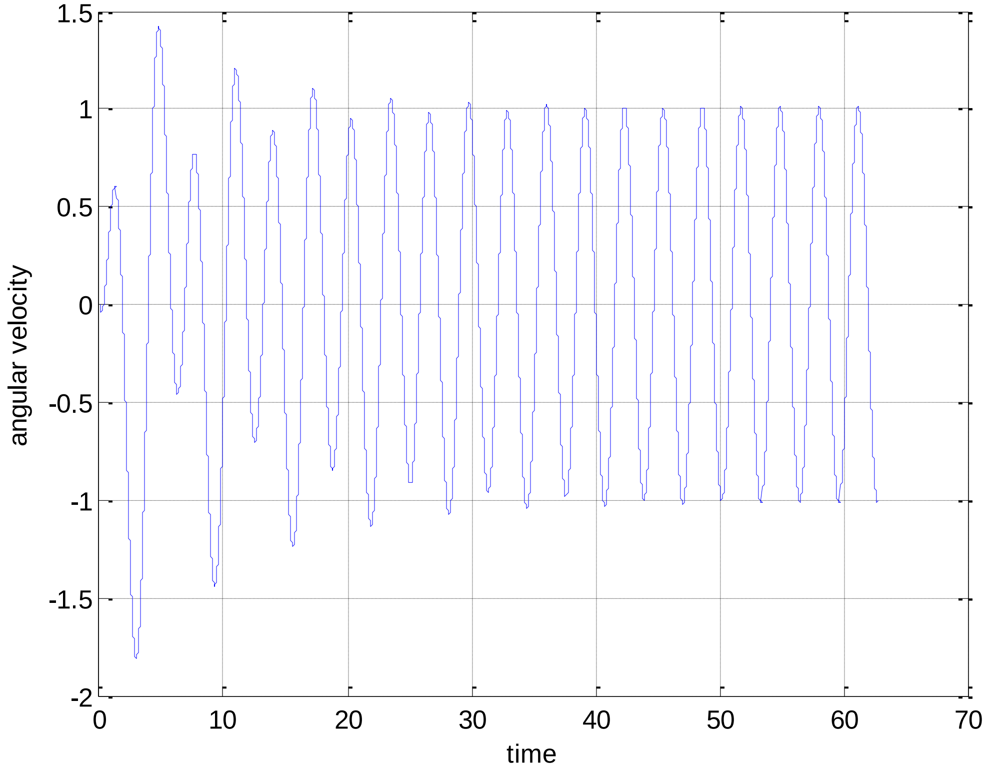


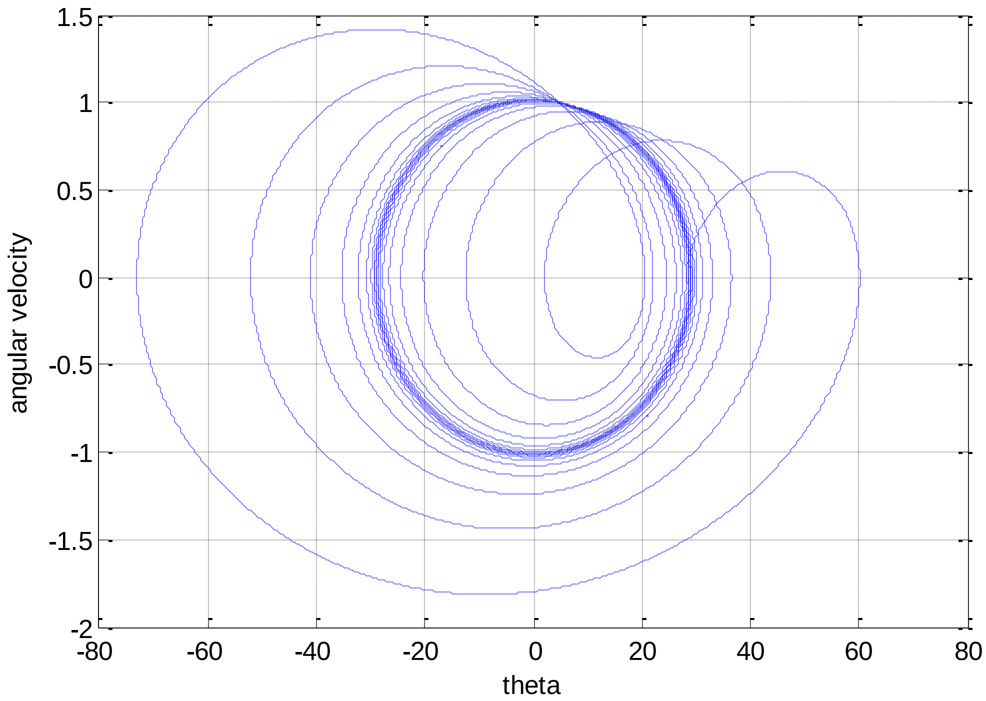
of oscillations

$$\ddot{\theta} + \frac{c}{m}\dot{\theta} + \frac{g}{l}\theta - \frac{F_D}{ml}\cos(\omega_D t) = 0.$$

In general the **oscillations tend to oscillate at the driving force frequency after steady state has been reached.** This is shown by the following graphs:
The parameters used are:
driving_angular_frequency = 2
driving_force_amplitude = 1.5N
natural_frequency = 1.


80
60
40
20
0
-20
-40
-60
-80
theta in degrees
0
10
20
30
40
50
60
70
time


1.5
1
0.5
0
-0.5
-1
-1.5
-2
angular velocity
0
10
20
30
40
50
60
70
time

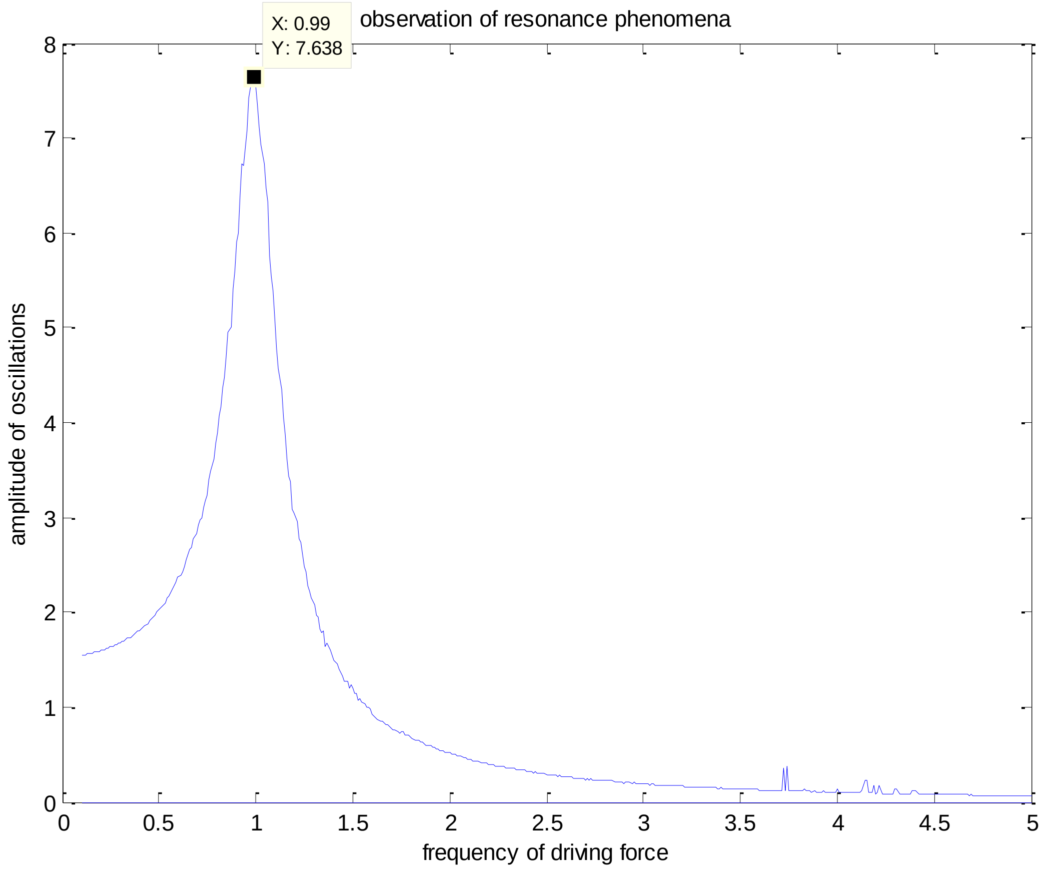

1.5
1
0.5
0
-0.5
-1
-1.5
-2
angular velocity
-80
-60
-40
-20
0
20
40
60
80
theta

OBSERVATION OF RESONANCE PHENOMENA :

We can generate a graph of amplitude vs frequency of driving signal. This graph shows us the phenomena of resonance. At a driving freq **just LESS THAN** the natural frequency, the amplitude of the oscillations is maximum. This was expected if we see the analytical solution. Analytically we maximum amplitude at

Driving frequency = 0.995

Computaionally we got it at 0.99 which is a fairly good approximation.



Note : In the above figure, theta is in radians.

Optimisation of code:

We don't need to run the solver for the same simulation time for each of the angular frequencies. In fact as the frequency increases, the oscillations stabilise relatively quickly and hence computational time reduces.This property is utilised here and two end cases have been used:They are

driver_angular_freq = 0.1 , approx. stabilising time = 300s

driver_angular_freq = 5 , approx. stabilising time = 60s

CODE:

The main code and the code of function is given below :

Main code:

```
%this script is used to calculate the variation of amplitude of the damped
%oscillations with driving frequency

clear all;
close all;

global C mass l g beta Fd driver_angular_freq max_ampl driver_time_period
simulationt;


g=9.8;
l=9.8; % so that g/l is 1

%natural angular_frequency is 1. We will run the code for driving_freq = 0
%to 5.

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

mass = 0.1;

%we will be considering only the underdamped case.
C = 0.02/3 % C is the drag constant Fdamp = C*length*omega
beta = C/(2*mass);
Fd = 1.5 % this is the driving force

driver_angular_freq = 1; % its the angular frequency
dt=timescale/100;
simulationt=50*timescale;
start_time = 0;


theta_degrees_initial = 0.5*180/pi; %given 0.5 radians
omega_initial = 0;
%using ode solver
%give initial conditions
u0 = zeros(2,1);
u0(1) = theta_degrees_initial*pi/180;
u0(2) = omega_initial;

start_freq = 0.1;
end_freq = 5;
incremental_freq = 0.02;

frequency_array = start_freq : incremental_freq : end_freq ;
```

```
amplitude_array1 = zeros(length(frequency_array));
amplitude_array2 = zeros(length(frequency_array));
amplitude_array3 = zeros(length(frequency_array));

index = 1;

% we are now plotting 3 graphs based on different values of beta and
% comparing their quality factors.

C = 0.02/3 ;
beta = C/(2*mass);
for driver_angular_freq = start_freq : incremental_freq : end_freq

    max_ampl = 0;
    driver_time_period = 2*pi / driver_angular_freq;

    %explain this step
    simulationt = -48.979*driver_angular_freq + 304.897;                % for optimising the
simulation time
    %for under damping
    [t,u] = ode45(@damp_driven_function , [start_time : dt : simulationt] , u0);
    amplitude_array1(index) = max_ampl;
    index = index + 1;
    %theta2=u(:,1)*180/pi;
    %omega2 = u(:,2);
end

C = 0.02;
beta = C/(2*mass);
index = 1;
for driver_angular_freq = start_freq : incremental_freq : end_freq

    max_ampl = 0;
    driver_time_period = 2*pi / driver_angular_freq;

    %explain this step
    simulationt = -48.979*driver_angular_freq + 304.897;                % for optimising the
simulation time
    %for under damping
    [t,u] = ode45(@damp_driven_function , [start_time : dt : simulationt] , u0);
    amplitude_array2(index) = max_ampl;
    index = index + 1;
    %theta2=u(:,1)*180/pi;
    %omega2 = u(:,2);
end
```

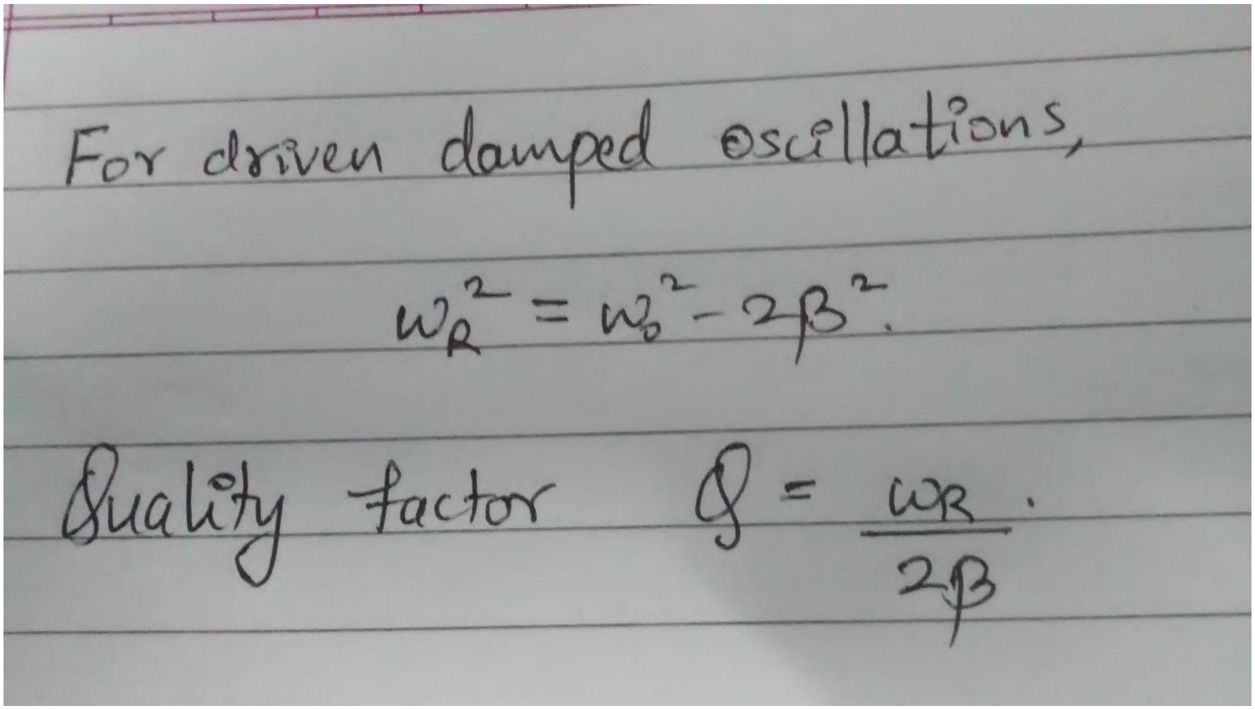
```
C=0.02*3;
beta = C/(2*mass);
index = 1;
for driver_angular_freq = start_freq : incremental_freq : end_freq

    max_ampl = 0;
    driver_time_period = 2*pi / driver_angular_freq;

    %explain this step
    simulationt = -48.979*driver_angular_freq + 304.897;                % for optimising the simulation time
    %for under damping
    [t,u] = ode45(@damp_driven_function , [start_time : dt : simulationt] , u0);
    amplitude_array3(index) = max_ampl;
    index = index + 1;
    %theta2=u(:,1)*180/pi;
  % omega2 = u(:,2);
end

plot(frequency_array , amplitude_array1,'r' , frequency_array,amplitude_array2 ,'g',frequency_array, amplitude_array3,'b')
xlabel('frequency of driving force')
ylabel('amplitude of oscillations')
title('observation of resonance phenomena')
```

Code of rhs function used :

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


%{
u(1) -> theta
u(2) -> omega

F(1) -> omega
F(2) -> derivative of omega
%}

global C mass l g beta Fd driver_angular_freq max_ampl simulationt;
global driver_time_period ;

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

```
F(1) = u(2);
F(2) = -(2*beta)*u(2) - (g/l)*u(1) + (Fd/(mass*l))*cos(driver_angular_freq*t); %cosine


if(t>(simulationt - driver_time_period)) %because the oscillations stabilise during this
time
   if(u(1) > max_ampl)
      max_ampl = u(1);
   end
end
%}
end
```

Changes in value of beta and quality factor:

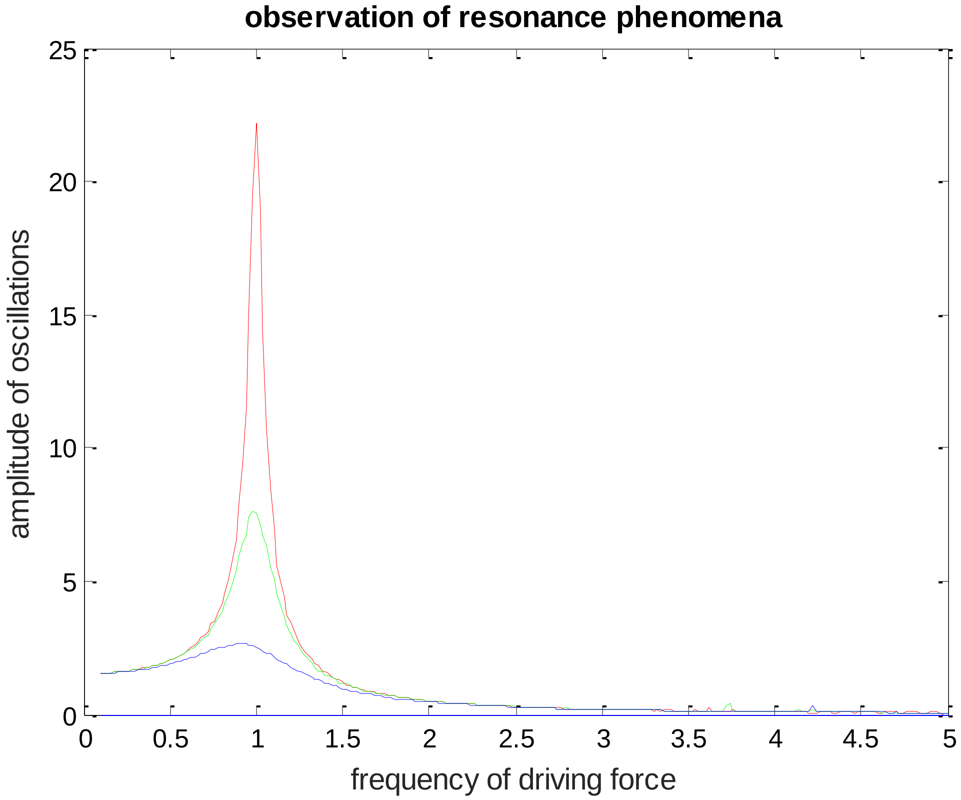


For driven damped oscillations,

$$\omega_R^2 = \omega_0^2 - 2\beta^2.$$

Quality factor $Q = \frac{\omega_R}{2\beta}$.

sr no.	damping constant (beta)	resonant angular frequency(wr)	quality factor	colour
1.	0.1	0.9898	4.949	green
2.	0.3	0.905	1.5	blue
3.	0.033	0.9989	15.134	red



Observation: As the quality factor increases because of decrease in beta, the full width half magnitude (FWHM) decreases.

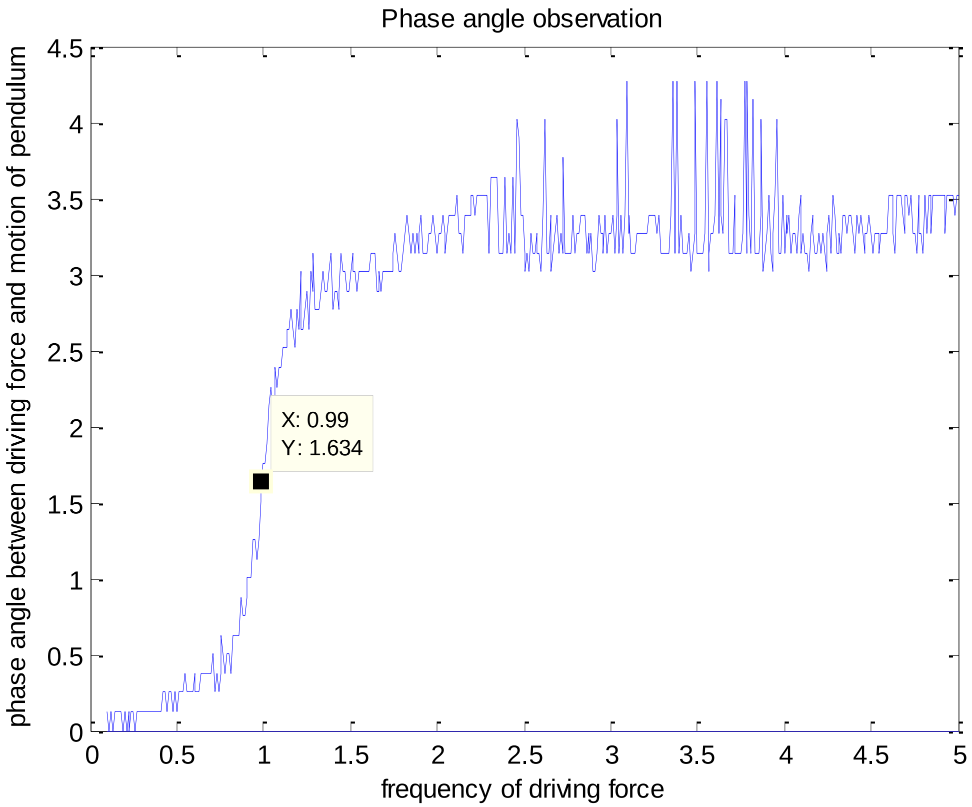
Variation of PHASE ANGLE between the driving force and the displacement of pendulum for different driving frequencies :

The phase angle between the driving force and the displacement of pendulum varies with the driving frequency. If the driving frequency is closer to natural frequency, the phase angle is close to pi/2. If driving frequency is greater than natural frequency, the phase angle increases and vice-versa.

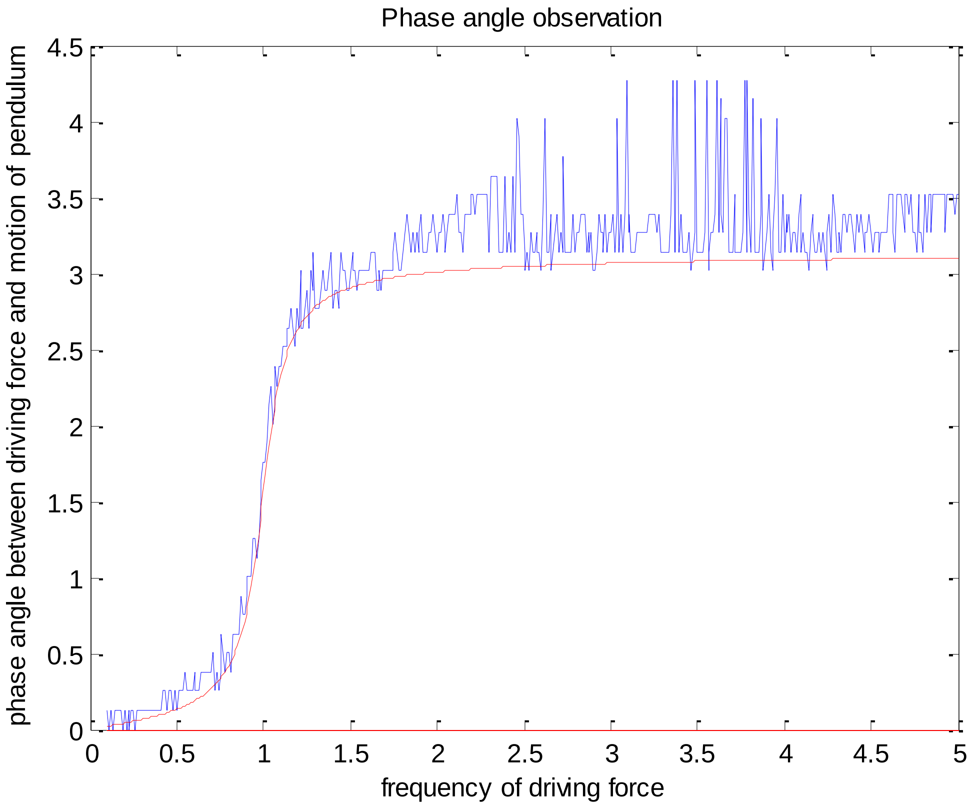
How to calculate phase angle computationally :
If we have 2 plots with same frequency but different phase then we need to find the phase angle between the two. Computationally there are 3 steps.

1. **Find any positive slope zero crossing for one of the plots.**
2. **Find the positive slope zero crossing for the other plot which is closest to the first one on the time scale.**
3. **Now we just need to use the formula:**
   Phase angle = (difference between zero crossings / Time period)* 2*pi

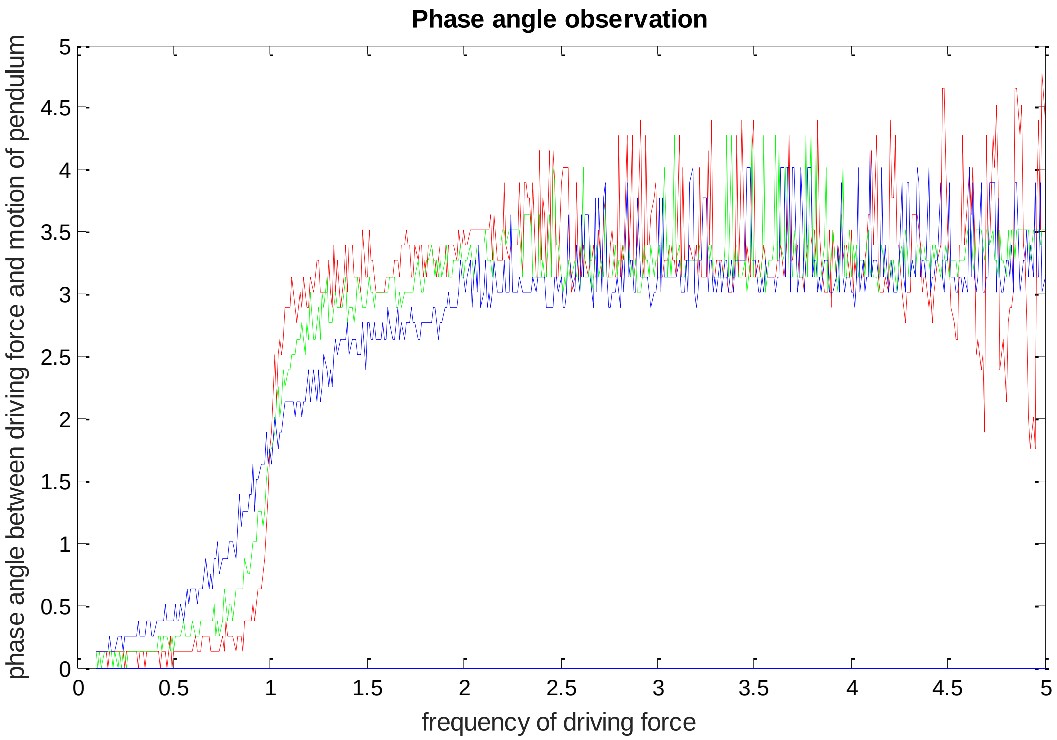
The graph is as follows



Comparison with analytical solution (in red):

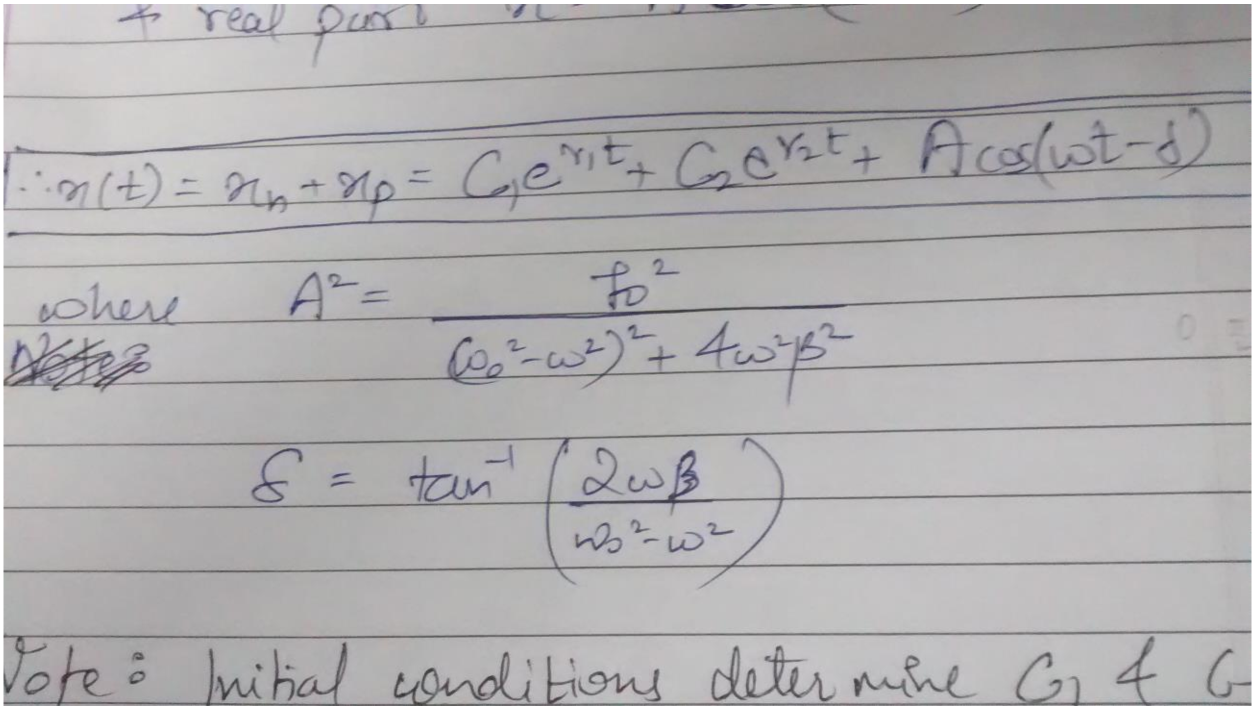


For different values of beta and quality factor we get different plots:



Observation: As the quality factor increases, the slope of the graph at resonant frequency increases. The values for red, green and blue plots are as mentioned in the table given above.

We can verify our two graphs analytically also. The equations for the amplitude and phase angle are enough to verify our observations.



+ real part

$$\therefore x(t) = x_h + x_p = C_1 e^{r_1 t} + C_2 e^{r_2 t} + A\cos(\omega t - \delta)$$

where ~~Notes~~ $A^2 = \dfrac{f_0^2}{(\omega_0^2-\omega^2)^2 + 4\omega^2\beta^2}$

$$\delta = \tan^{-1}\left(\frac{2\omega\beta}{\omega_0^2-\omega^2}\right)$$

Note: Initial conditions determine C_1 & C

Code: There are 2 files – one for the function and the other for the main code

1. **Q2_phase_angle.m**

```
% this code is used to plot the variation of the phase angle between
% driving force and the displacement motion as a function of the driving
% frequency.

clear all;
close all;

global C mass l g beta Fd driver_angular_freq driver_time_period simulationt
zero_crossing_time min_square_theta;



g=9.8;
l=9.8; % so that g/l is 1
%natural angular_frequency is 1. We will run the code for driving_freq = 0
%to 5.

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

mass = 0.1;

%we will be considering only the underdamped case.
C = 0.02 % C is the drag constant Fdamp = C*length*omega
beta = C/(2*mass);
Fd = 1.5 % this is the driving force

driver_angular_freq = 1; % its the angular frequency
```

```
dt=timescale/100;
simulationt=50*timescale;
start_time = 0;

theta_degrees_initial = 0.5*180/pi; %given 0.5 radians
omega_initial = 0;

%using ode solver
%give initial conditions
u0 = zeros(2,1);
u0(1) = theta_degrees_initial*pi/180;
u0(2) = omega_initial;

start_freq = 0.1;
end_freq = 5;
incremental_freq = 0.01;

frequency_array = start_freq : incremental_freq : end_freq ;
phase_array = zeros(length(frequency_array));
index = 1;


for driver_angular_freq = start_freq : incremental_freq : end_freq

    min_square_theta = 500*500; %used in odesolver. Set to some high value
    driver_time_period = 2*pi / driver_angular_freq;

    %We don't need to run the solver for the same simulation time for each of
the angular frequencies.
    % In fact as the frequency increases, the oscillations stabilise
    % relavively quickly and hence computational time reduces.
    % This property is utilised here and 2 end cases have been ised:
    % They are driver_angular_freq = 0.1 , stabilising time = 300s
    % driver_angular_freq = 5 , stabilising time = 60s

    simulationt = -48.979*driver_angular_freq + 304.897;                % for
optimising the simulation time

    [t,u] = ode45(@damp_driven_function_for_phase_angle , [start_time : dt :
simulationt] , u0);

    %we have the +ve slope zero crossing for the displacement motion graph.
    %search for the first zero crossing for the force after
    %zero_crossing_time

    time = zero_crossing_time;
    force_zero_crossing = 0;
    incremental_step = (driver_time_period/50);


    for t1 = time : incremental_step : (time + driver_time_period)
       if(cos(driver_angular_freq*t1) < 0 && (cos(driver_angular_freq*(t1 +
incremental_step)) > 0) )
```

```
            force_zero_crossing = t1;
            break;
        end
    end

    phase_array(index) = ((( - force_zero_crossing +
zero_crossing_time)*2*pi)/driver_time_period) + 2*pi;

    if(phase_array(index) > 6.28)
        phase_array(index) = phase_array(index) - 2*pi;
    end

    index = index + 1;
end

plot(frequency_array , phase_array);
xlabel('frequency of driving force')
ylabel('phase angle between driving force and motion of pendulum')
title('Phase angle observation')
```

2. damp driven function for phase angle.m

```
function F = damp_driven_function_for_phase_angle( t , u )
%This function is used in ode solver which itself is used to plot the
variation of the phase angle between
% driving force and the displacement motion as a function of the driving
% frequency.

%{
u(1) -> theta
u(2) -> omega

F(1) -> omega
F(2) -> derivative of omega
%}

global C mass l g beta Fd driver_angular_freq max_ampl simulationt;
global driver_time_period zero_crossing_time min_square_theta;

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

F(1) = u(2);
F(2) = -(2*beta)*u(2) - (g/l)*u(1) +
(Fd/(mass*l))*cos(driver_angular_freq*t); %cosine !!!!!!!!!!!!!!!!!!!!


if(t>(simulationt - driver_time_period)) %because the oscillations stabilise
during this time
    if(u(1)*u(1) < min_square_theta && F(1) > 0)    %we calculate the
amplitude val closest to 0 (either +ve or -ve)
        %+ve slope zero crossing
        min_square_theta = u(1)*u(1);
        zero_crossing_time = t;
    end
```

```
end
%}
end
```