Jainil Desai

UFID – 96950890

Report – Lab 2

Structural Dynamics of Production Machinary

Index

Part I

A) Spring stiffness at the limit of stability,

klim = 2*m*g/l
= 2*0.5*9.81/0.3
= 32.7 N/m

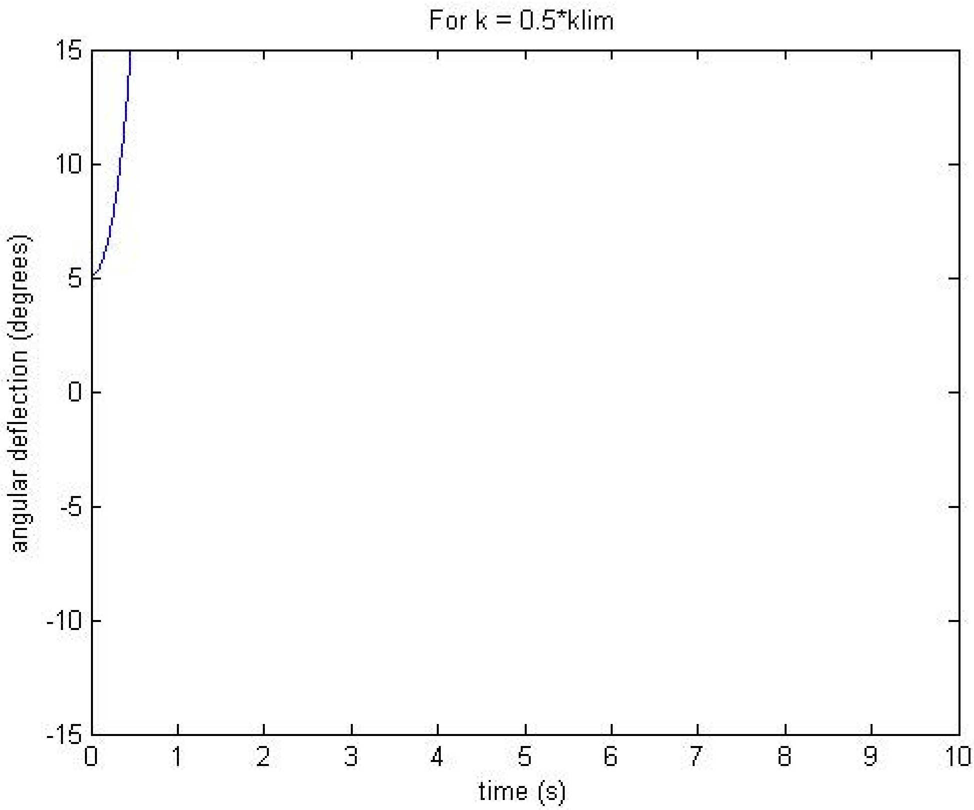
B)

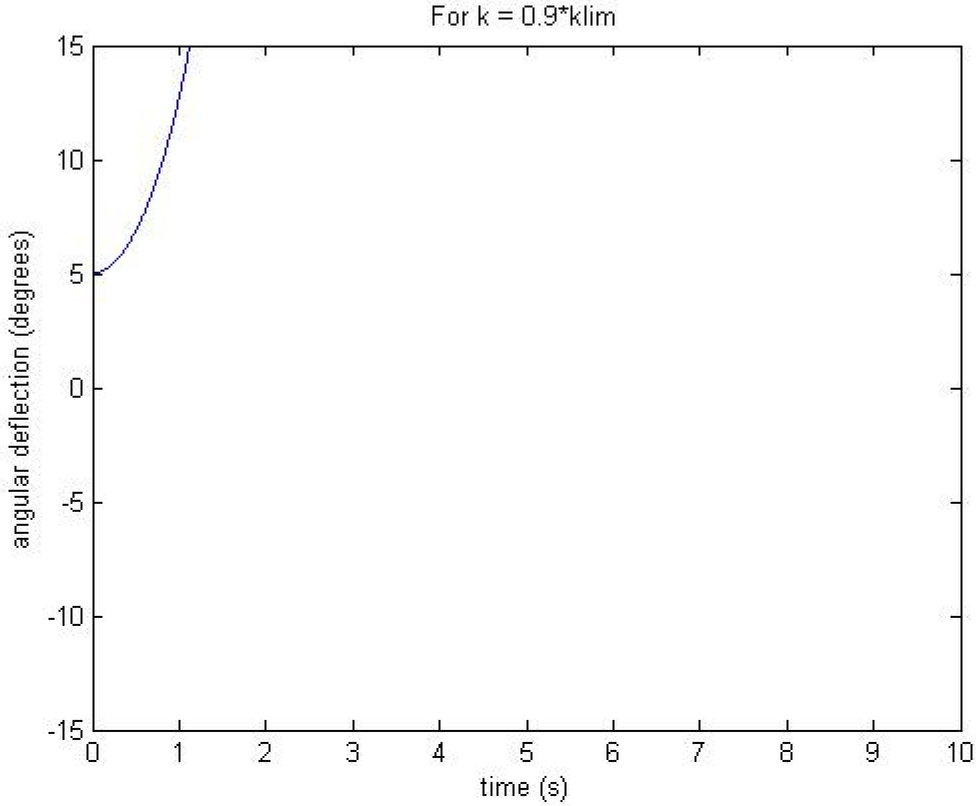
Now using this above values we plot the system behavior at different k values. Here we have plotted the angle theta vs the time for 0.1*klim, 0.5*klim, 0.9*klim, 0.99*klim, klim, 5*klim, 10*klim. The code for the time domain simulation for the system is as shown in the appendix 1.
In the code the value of k can be changed to get different system behavior. Simulation step size is dt = 0.005; seconds and total steps are, step = 1/0.005*10 = 2000. The angular acceleration in the terms of velocity and angular dis[placement used in the code is,

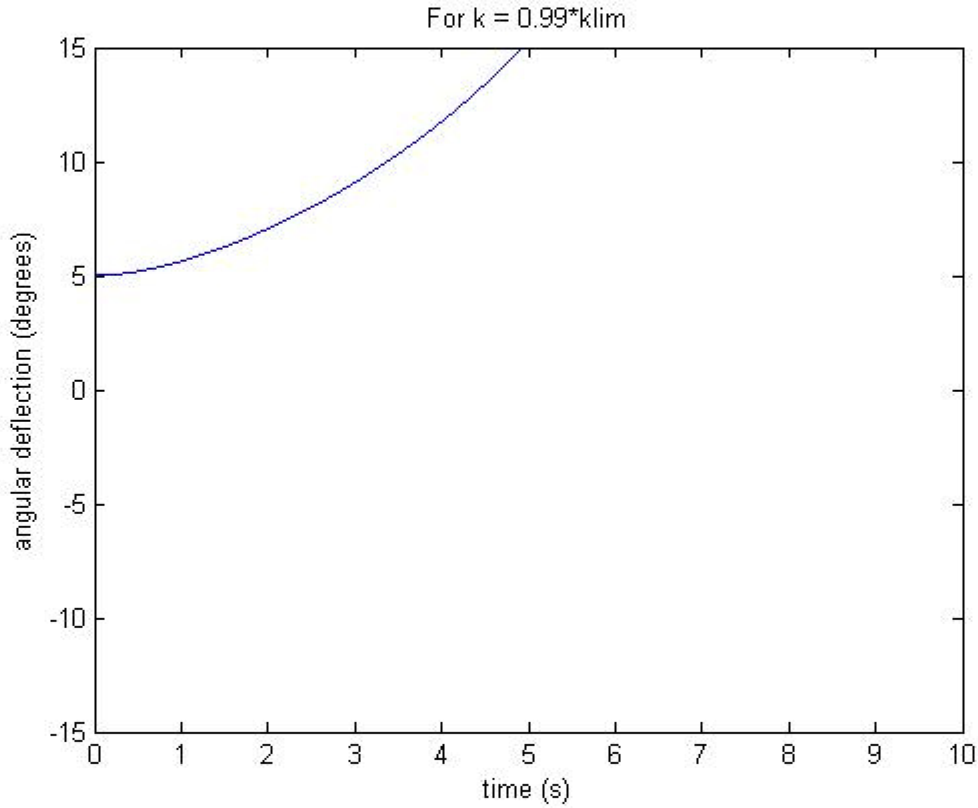
```
angacc = (-theta*(k*l^2/2-m*9.81*l)-(l^2*angvel/2))/(m*l^2);
```

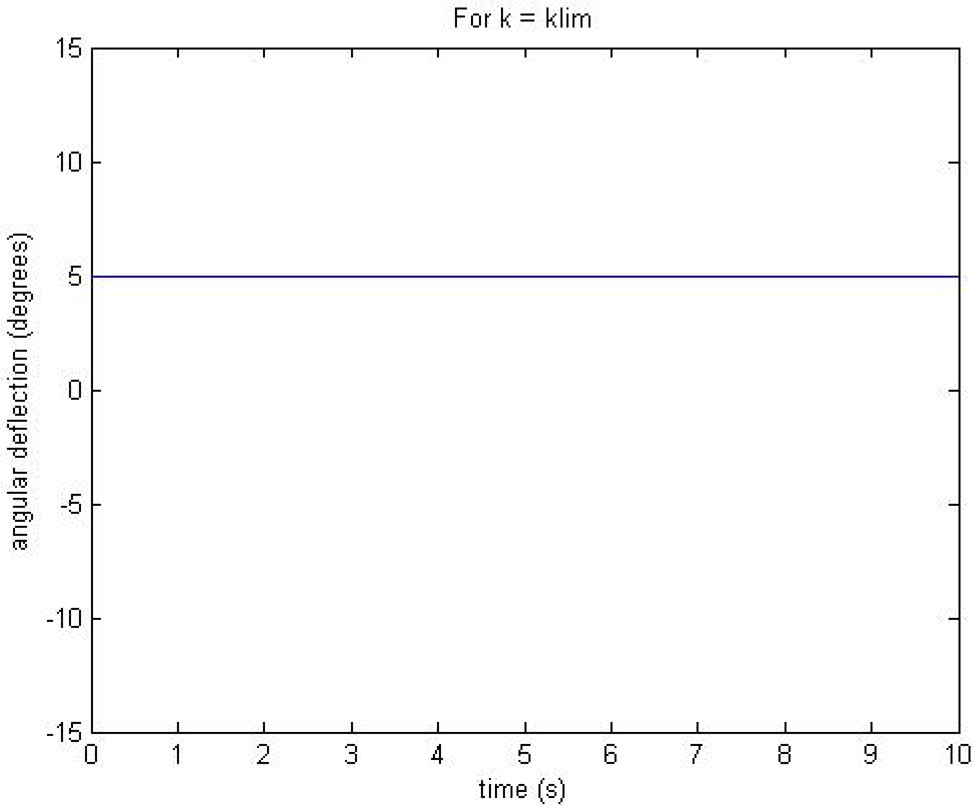
using the initial conditions we find the angular velocity, then we perform the euler integration to find the corresponding velocity and displacement and update the simulation.

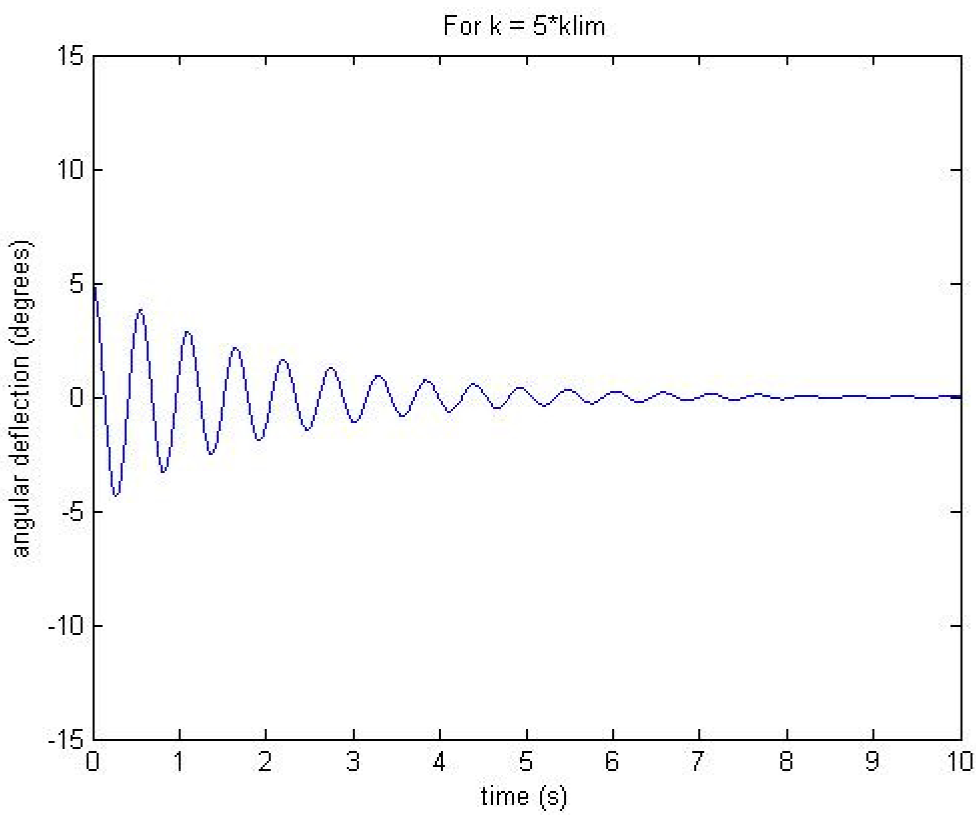
```
angvel = angvel + angacc*dt;
theta = theta + angvel*dt;
```

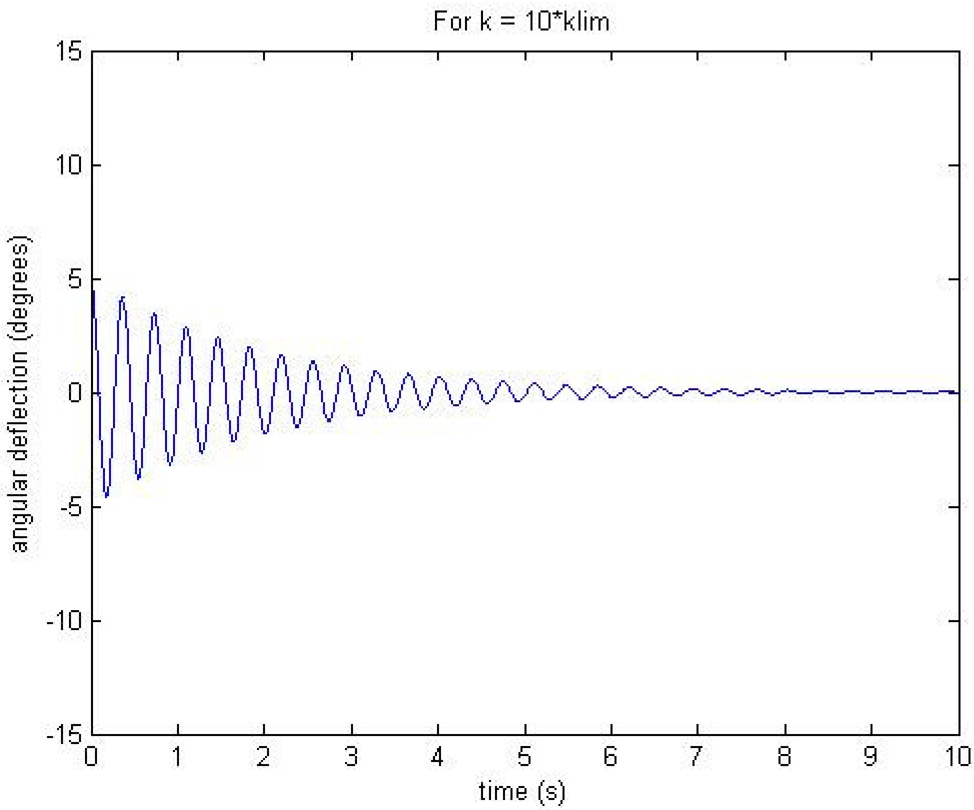

For k = 0.5*klim
angular deflection (degrees)
time (s)
15
10
5
0
-5
-10
-15
0
1
2
3
4
5
6
7
8
9
10


For k = 0.9*klim
angular deflection (degrees)
time (s)
15
10
5
0
-5
-10
-15
0
1
2
3
4
5
6
7
8
9
10


For k = 0.99*klim
angular deflection (degrees)
time (s)
15
10
5
0
-5
-10
-15
0
1
2
3
4
5
6
7
8
9
10


For k = klim
15
10
5
0
-5
-10
-15
0 1 2 3 4 5 6 7 8 9 10
angular deflection (degrees)
time (s)


For k = 5*klim
15
10
5
0
-5
-10
-15
0 1 2 3 4 5 6 7 8 9 10
angular deflection (degrees)
time (s)



C) As we see that wen k < klim, the pendulum falls off as the spring less stiff than required to support the mass. As the stiffness approaches klim, the time to drop off increases as the spring could resist the mass more. At k = klim, the stiffness of the spring is just enough to support the mass and it remains stable at the same position. When k > klim, the spring stiffness is more than required to support the mass. So the spring oscillates initially and gets to a stable condition. The time to reach stability decreases as k increases.

Part II

A) clim = gamma
=10^7 N/m

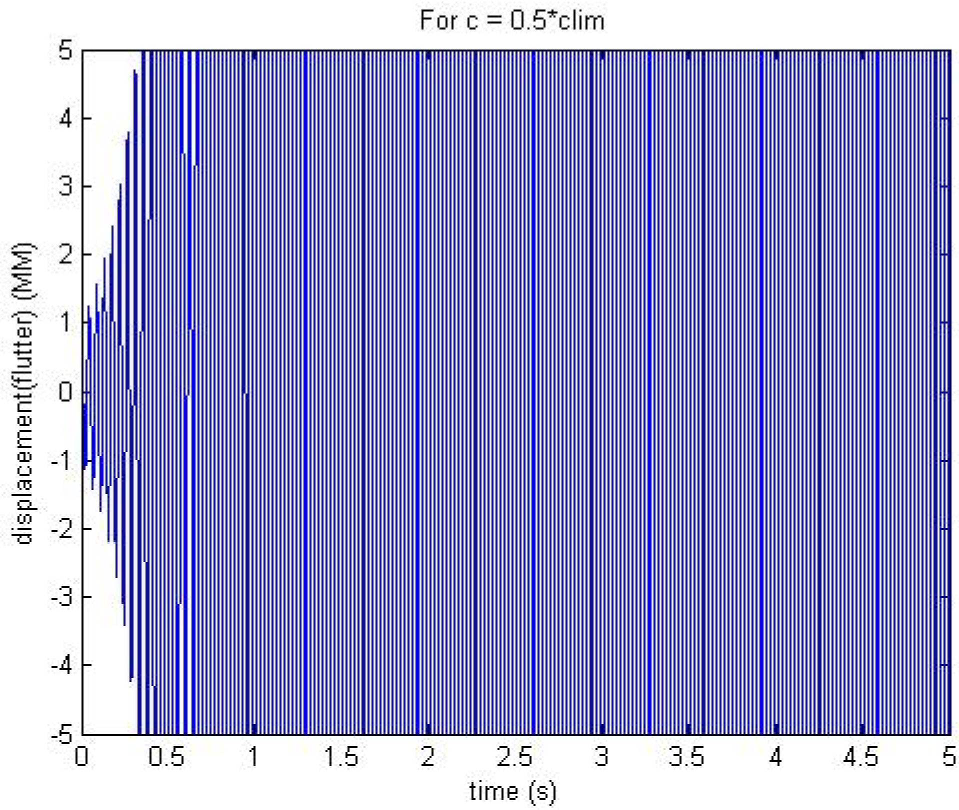
B) using the same steps described above we do the time domain simulation for the system using the code mentioned in appendix 2. Here we have used the time steps of 0.002 seconds and 2500 steps,

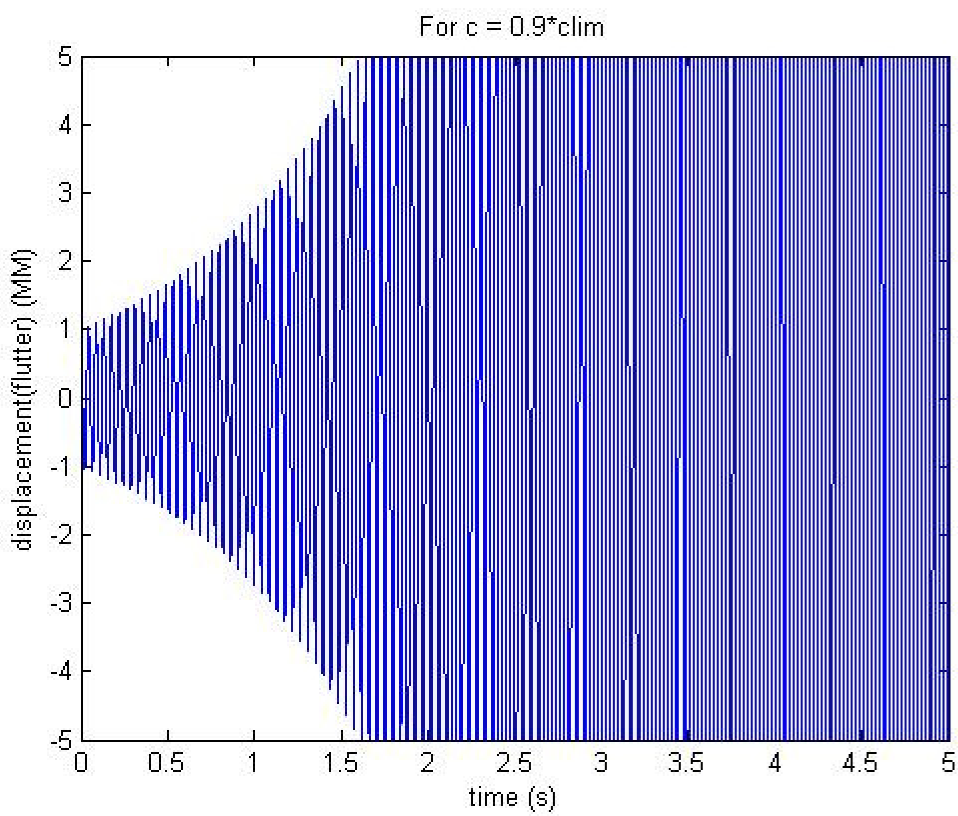
using the given initial conditions, we fing the acceleration,

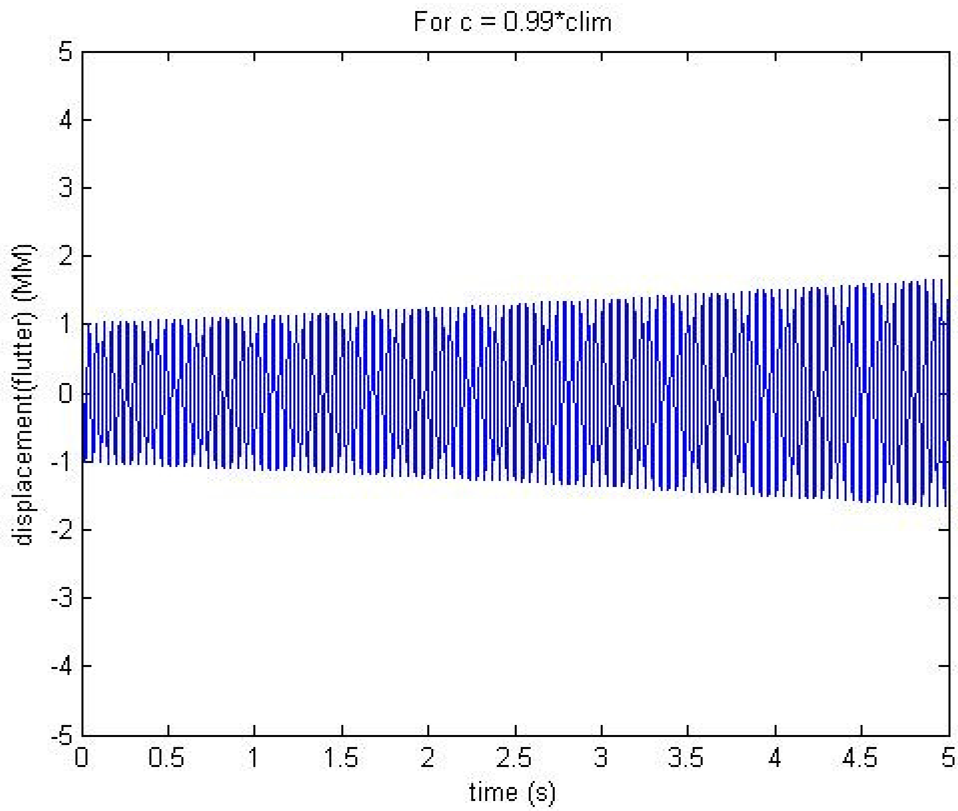
```
accx = (-k*x-(c-gamma)*velx)/m;
```

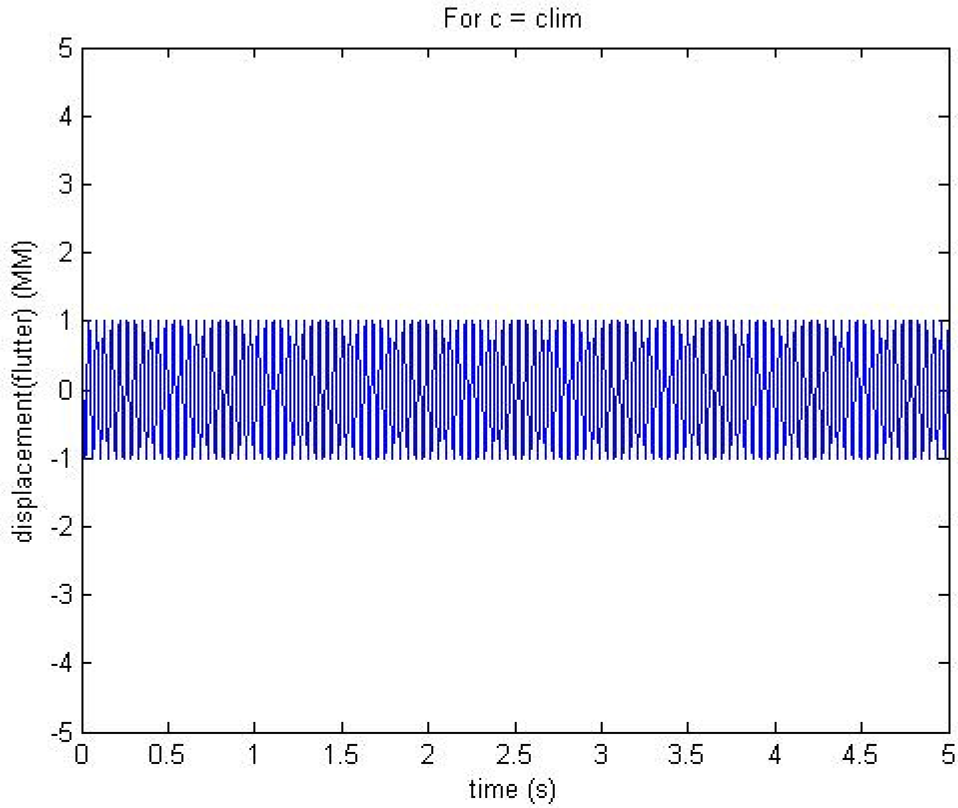
Euler integration steps for updating the system are,

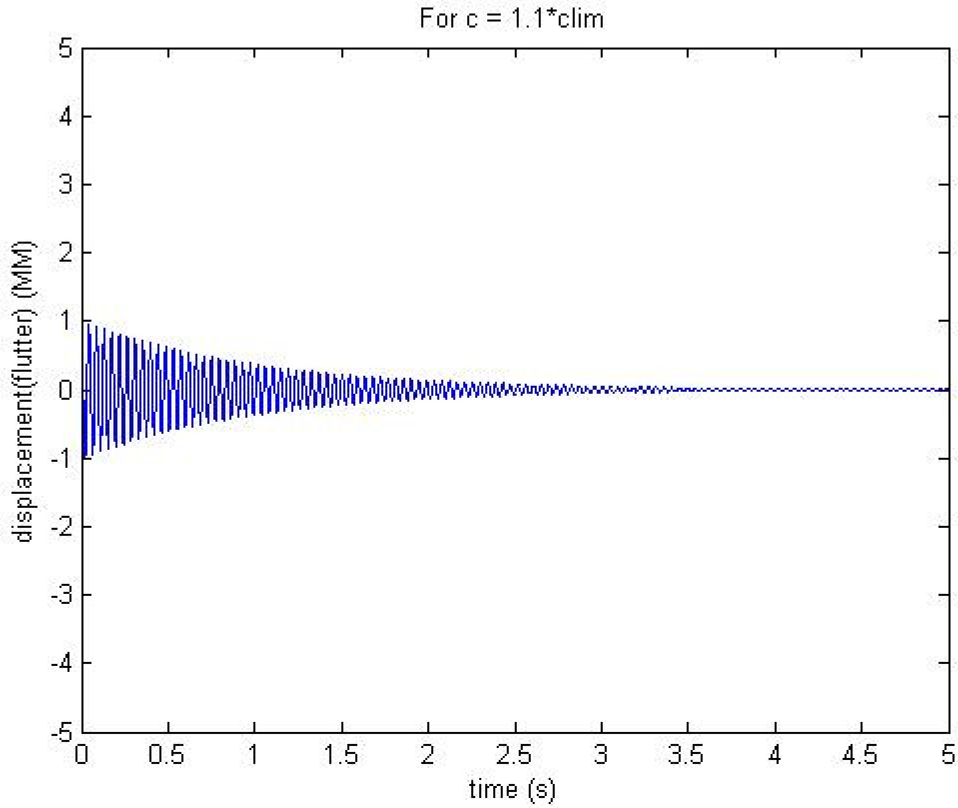
```
velx = velx + accx*dt;
   x = x + velx*dt;
```

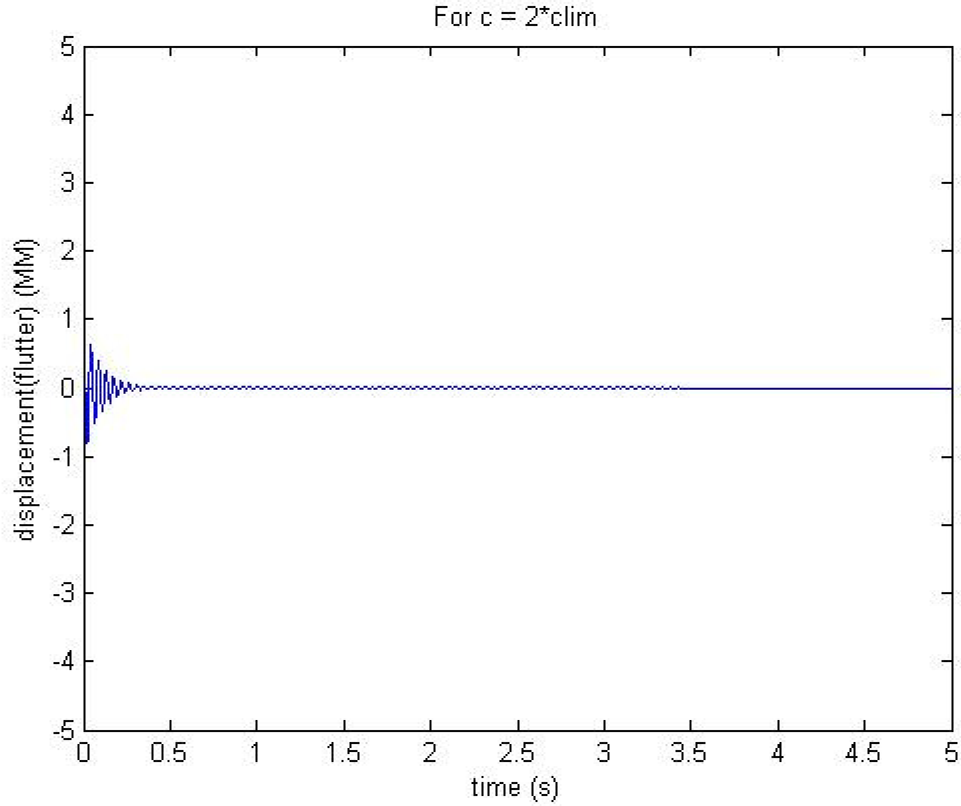

For c = 0.5*clim
displacement(flutter) (MM)
time (s)
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


For c = 0.9*clim
displacement(flutter) (MM)
time (s)
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


For c = 0.99*clim
displacement(flutter) (MM)
time (s)
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


For c = clim
displacement(flutter) (MM)
time (s)
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


For c = 1.1*clim
displacement(flutter) (MM)
time (s)
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5



C) Here we see that as c increases the system stability increases. Specifically when c < clim, the system is extremely unstable and the flutter increases gradually with time. At c= clim, this is the critical damping limit and at this point the the flutter is constant. As c increases and it gets more than clim, the system gets stable. Initially there is some flutter and it gets stable gradually with time.

Appendix 1

```
clc
clear all
close all
l = 0.3; %m
c = 1; %N/m
m = 0.5; %kg
klim = 32.7; %N/m
k = 0.5*klim;
dt = 0.005;
theta = 5; angvel = 0;
%toal steps
step = 1/0.005*10
for n = 1:(step)
    theta1(n) = 0;
end
% performing euler integration
for n = 1:(step)
    angacc = (-theta*(k*l^2/2-m*9.81*l)-(l^2*angvel/2))/(m*l^2);
    angvel = angvel + angacc*dt;
    theta = theta + angvel*dt;
    theta1(n) = theta;
    t(n) = n*dt;
end

%Plotting the results
figure(1)
plot(t, theta1)
ylim([-15 15])
xlim([0 10])
xlabel('time (s)')
ylabel('angular deflection (degrees)')
title('For k = 0.5*klim')
```

Appendix 2

```
clc
clear all
close all
k = 1e7;
gamma = 1e4;
m = 500;
clim = 1e4;
c = 0.5*clim;
dt = 0.002;
x = 1; velx = 0;
step = 1/0.002*5
for n = 1:(step)
    x1(n) = 0;
end

for n = 1:(step)
    accx = (-k*x-(c-gamma)*velx)/m;
    velx = velx + accx*dt;
    x = x + velx*dt;
    x1(n) = x;
    t(n) = n*dt;
end

figure(1)
plot(t, x1)
xlim([0 5])
ylim([-5 5])
xlabel('time (s)')
ylabel('displacement(flutter) (MM)')
title('For c = 0.5*clim')
```