33-151 Matter & Interactions I Fall 2019

Tuesday Week 7 Recitation

Name (Printed) _____ Section _____

Instructor Signature: _____
Due: Wednesday of Week 8 in lecture.

Damping and Driving with Masses and Springs

Problems from The Modern Mechanical Universe

4.P69. Write a program to model a mass oscillating on a spring. Have your program display the motion of the mass on the spring, and display a plot of the kinetic energy, potential energy, and total energy of the system as a function of time. Modify your program to include friction. Study the behavior of your resulting system.

4.P70. Modify your program from 4.P69 to include a driving force. Study the response of your system to different driving frequencies from much smaller than the natural frequency to much larger than the natural frequency.

Programing Assignment

In this assignment, we will modify our earlier mass-spring simulation to include damping (dissipative forces) and driving forces. This is based on problems 4.P71 and 4.P72. For convenience, a shell is provided as part of this assignment. To make sure that everyone is starting at the same point, use the following values in your program:

- spring constant: $k_s = 0.7$ N/m.
- bob mass: $m = 0.020$, kg.
- initial stretch, $s = 20.0$ cm.
- initial speed: 0 m/s.

Once your program runs, then modify the graphics window to display energy as a function of time, rather than position as a function of time for time going through several periods.

```
graph(title=s, xtitle='Time', ytitle='Energy',xmax=10.0, ymax=0.25, ymin=-0.25,
      x=0, y=500, width=500, height=300)
#
drawit = gcurve(color=color.cyan, label='Energy')
```

Next, set up curves to display kinetic energy, spring-potential energy and total energy as a function of time. Finally, modify your while loop so that the program will run for 5 seconds of elapsed time, and then stops.

Viscous Damping

Frictional forces will damp out the motion of the mass spring system. Consider a damping force of the form $\vec{F}_{damp} = -\beta\vec{p}/m$, a force proportional and opposite to the velocity of the mass. Add such a term to your program using an initial value of $\beta = 0.02$. Sketch the shape of the resulting "total energy curve".

Set the damping factor to be $\beta = 2 \cdot \sqrt{k_s \cdot m}$ and observe the resulting behavior. Note that to see the kinetic energy, it might be necessary to change the energy scale on your plot.

Describe what you see.

Now increase β to four times $2 \cdot \sqrt{k_s \cdot m}$ and observe what happens.

What do you see?

Driving Forces

Set the damping term to zero, ($\beta = 0$) and add a driving force to the system. The driving force should be of the form:

$$\vec{F}_{drive} = -F_0 \cos(\omega_d t)$$

set the constant $F_0 = 0.15 \times k_s \times s_{initial}$ where $s_{initial}$ is the initial stretch of the mass spring system. Set the driving frequency to be a large (non-integer) multiple of the natural frequency of the mass-spring system, $\omega_d = 25.33 \times \sqrt{k_s/m}$.

Observe what happens to the system and comment on what you see.

Now change the driving frequency to be a fraction of the natural frequency, $\omega_d = 0.16 \times \sqrt{k_s/m}$ and observe what happens.

Describe what you observe.

Finally set the driving frequency to be exactly equal to the natural frequency and observe what happens.

Change the driving force to be sine instead of cosine and observe what happens (when ω_d is the natural frequency).

Driving Forces with a Damped System

Now combine both a cosine driving force at the natural frequency and a damping factor of $\beta = 0.03$.

Observe what happens to the system.

Continue to play with your program to observe what happens to the system.

Program Shell

```
from math import *
from vpython import *
#
# Set program constants
#
BobMass = 0
SpringConstant = 0
DampingBeta = 0
```

```
InitialStretch = 0
EquilLength = 0
ExpectedPeriod = 2*pi*sqrt(BobMass/SpringConstant)
#
scene2 = canvas(title='Damped Oscillator',caption='Animated Display',
     center=vector(0.50,0,0), background=color.white)
#
bob    = sphere(pos=vector(0,0,0),radius=0.05,color=color.red)
wall   = box(pos=vector(0,0,0),size=vector(0.05,.1,.1),color=color.blue)
spring = helix(pos=wall.pos,axis=bob.pos-wall.pos,radius=0.01,thickness=.004,coils=10,co
#
# equilibrium position of the end of the spring.
#
length  = vector(EquilLength,0,0)
stretch = vector(InitialStretch,0,0)
#
# Set the initial position of the bob
#
bob.pos = wall.pos + length + stretch
bob.mom  = vector(0,0,0)
#
# Input Parameters needed in the program. Be sure to
# choose sensible values.
#
bob.mass = BobMass
spring.ks  = SpringConstant
beta = DampingBeta
#
print('Damped Mass on a Spring')
#
# Time step and total elapsed time
#
dt = 0.005
t  = 0.0
#
# Used to look for zero crossings to measure the period.
#
told = 0.0
xold = bob.pos.x
#
# Setup a graph window to plot things in
#
```

```
s='<b>Mass and Spring: Graph</b>'
#
# Move the mouse over the graph to explore its interactivity.
# Drag a rectangle in the graph to zoom. Examine the icons at the upper right.
# Click the "Reset axes" icon to restore. Drag along the bottom or left to pan.
#
graph(title=s, xtitle='Time', ytitle='Displacement',xmax=10.0, ymax=0.25, ymin=-0.25,
      x=0, y=500, width=500, height=300)
#
drawit = gcurve(color=color.cyan, label='Position')
#
#
while(t<10):
    rate(100)
    t += dt
#
#    spring.stretch = block.pos-wall.pos
#
    spring.force = 0
    bob.mom = 0
    bob.pos = 0
    spring.axis = bob.pos-wall.pos
#
# Check for a zero crossing
#
    xnew = bob.pos.x - wall.pos.x - length.x
    if xnew*xold <= 0:
        period = 2*(t - told)
        if told != 0:
            scene2.caption=('Expected period is',ExpectedPeriod,' actual period is',peri
        told = t
    xold = xnew
#
# Plot the x-coordinate of the block as a function of time.
#
    xbob = bob.pos.x - length.x + wall.pos.x
    drawit.plot(pos=(t,xbob))
#
print("Expected Period (sec)",ExpectedPeriod)
print("Actual Period   (sec)",period)
print('All Done')
```