33-151 Matter & Interactions I Fall 2019

Name: ____

Section: A (8:30) B (9:30) C (10:30) ______________

Instructor Signature: ________________________________

Recitation Assignment for Tuesday of Week 4. This is due on Wednesday of week 5.

Problems

3.P69 In this problem we want to numerically solve the mass on a spring problem. The goal of this problem will be to numerically solve the mass on a spring problem, and then examine how how the period of oscillation depends on the mass, spring constant, and the amplitude of the motion. (a) Write a program that will predict the motion of a mass, m, attached to the end of a spring with constant k. You may limit the motion to be along a single dimension. (b) Use your program to examine the motion and measure the period of the motion. Start you program with a $10\,g$ mass, a spring constant of $10\,N/m$ and an initial amplitude of $10\,cm$. Vary the time step you use and determine a reasonable value. (c) Now vary the amplitude by a factor of four, both decreasing and increasing it. Measure the resulting periods of motion and compare with you value from part (b). (d) Now vary the mass by a factor of four and compare the periods. Then vary the spring constant by a factor of four. Are your results consistent with what you expect?

Do parts **a** through **d**. You will get a signature you have a program that accurately plots the mass on a spring, and plots the position as a function of time. Please consult the handout on graphic windows in VPython for information on opening an plotting in these windows. The following are clarifications to problem 3.P70.

When you vary the amplitude, how does the period change?

When you vary the mass, how does the period change?

When you vary the spring constant, how does the period change?

(i) Initial conditions: Make the initial position of the block be such that the stretch of the spring is equal to the amplitude of the oscillation in your experiment, and release the block with zero initial momentum. Display the motion in two ways: (i) Plot a graph of the position of the block as a function of time. Label the scales on both axes so your numerical results are clear. (ii) Make an animation of the motion of the block as a function of time.

(ii) Vary the time step a report the maximum step size that gives reasonable results (in the sense that a smaller step has little effect on your answer).

(iii) Read the *period* of oscillation off the graph. Compare this to your measured value.

(iv) Repeat your calculation with double the initial stretch. How does the period change? Is this what you observed in your expriment?

3.P70. (a) Modify your program from problem 3.P69 to use an x^3 force law,

$$\vec{F} \;=\; -\kappa s^3 \hat{s}\,.$$

(b) Use your program to examine the motion and measure the period of the motion. Start you program with a $10\,g$ mass, a spring constant of $10\,N/m$ and an initial amplitude of $10\,cm$. Vary the time step you use and determine a reasonable value. (c) Now vary the amplitude by a factor of four, both decreasing and increasing it. Measure the resulting periods of motion and compare with you value from part (b). (d) Now vary the mass by a factor of four and compare the periods. Then vary the spring constant by a factor of four.

When you vary the amplitude, how does the period change?

When you vary the mass, how does the period change?

When you vary the spring constant, how does the period change?

3.P71. Consider a mass m connected between two springs as shown in Figure 1 where the mass and springs are constrained to be along the horizontal axis. (a) Write a program that will numerically solve this problem. (b) Using initial values of $m = 10\,g$ and $k = 10\,N/m$, what is the period of motion of the system? (c) Does the period depend on the amplitude of the motion?

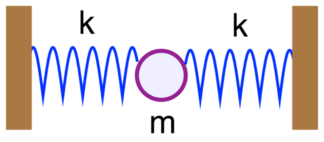

Figure 1: The figure from problem 3.P71.

For the initial values of 10 g and 10 N/m, what is the period of the system?

How does the period depend on the amplitude that you use?

New Program Constructs

The new elements of vpython in this program is the plotting of a graph in addition to our physics animation. To do this, we are going to create a *canvas* where we will do the animation. You can see this in the program shell below, but the relevant command follows where we can set both a title and a caption, control the background color, and set where the center of the scene is.

```
scene2 = canvas(title='Mass on a Spring',caption='Animated Display',
      center=vector(equil_length,0,0), background=color.white)
```

You will see that later in the program, we also create a *graph*. We also show here how we can create a text string, s, and use it for the *title* of the graph. The *max* and *min* variables set the physical limits in MKS units on the graph. The x and y locate the origin in pixel coordinates on the screen, and the *width* and *height* controls the size of the window in pixels.

```
s='<b>Mass and Spring: Graph</b>'
#
graph(title=s, xtitle='Time', ytitle='Displacement',
           xmax=10.0, ymax=0.25, ymin=-0.25, x=0, y=500, width=500, height=300)
```

Once we have the graph window, we need to add a graphical curve, or *gcurve* to it.

```
drawit = gcurve(color=color.cyan, label='Position')
```

Then, during the program loop, we add to the curve on our graph using the command

```
drawit.plot(pos=(t,block.x))
```

that adds a point at the position given by the current time on the x axis and the position of the block on the y axis.

The Program Shell

```
from math import *
from vpython import *
#
# Initialize so spring properties. All units are assumed to me
# in MKS units.
#
spring_constant = 0.0
equil_length    = 0.0
initial_stretch = 0.0
bob_mass        = 0.0
#
# Set up the animation canvas where we display the mass on the spring.
#
scene2 = canvas(title='Mass on a Spring',caption='Animated Display',
     center=vector(equil_length,0,0), background=color.white)
#
# Set up the wall, spring and mass
#
wall   = box(pos=vector(0,0,0),size=vector(0.01,.2,.2),color=color.cyan)
spring = helix(pos=wall.pos,axis=bob.pos-wall.pos,radius=0.01,thickness=.01,coils=10,col
bob  = box(pos=vector(0,0,0),size=vector(.05,.05,.05),color=color.red)
#
# equilibrium position of the end of the spring.
#
spring.equil = vector(equil_length,0,0)
bob.pos = wall.pos + spring.equil + vector(stretch,0,0)
#
# Input Parameters needed in the program. Be sure to
# choose sensible values.
#
bob.mass  = bob_mass
bob.mom   = vector(0,0,0)
spring.ks = spring_constant
dt = 0.01
t  = 0.0
#
#----------------------------------------------------------
#
scene.autoscale = 0           # Turn off auto scaling
#
# Setup a graph window to plot things in
# 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.
#
s='<b>Mass and Spring: Graph</b>'
#
graph(title=s, xtitle='Time', ytitle='Displacement',xmax=10.0, ymax=0.25, ymin=-0.25,
      x=0, y=500, width=500, height=300)
#
# Set up the curve to plot information on the graph.
#
drawit = gcurve(color=color.cyan, label='Position')
#
#
while(t<10):
    rate(100)
    t = t + dt
#
#
    spring.force = 0
    bob.mom = 0
    bob.pos = 0
#
# Plot the x-coordinate of the block as a function of time.
#
    spring.axis = bob.pos-wall.pos
    xblock = bob.pos.x - spring.equil.x + wall.pos.x
    drawit.plot(pos=(t,xblock))
#
```