Name: _____

Section: A (8:30) B (9:30) C (10:30) D (11:30) _____

Instructor Signature:_____

Recitation Assignment for Tuesday of Week 4. This is due on Friday, together with the rest of the problems assigned.

Problems

3.P70 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.

(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.P71. (a) Modify your program from problem P70 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.

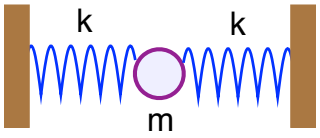

Figure 1: The figure from problem P72.

Optional problem:

3.P72. 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?

New Program Constructs

The new elements of VPython in this program is the plotting of a graph in addition to our physics animation. The graph capabilities are added by importing them into Python with the statement.

```
from visual.graph import *
```

We then need to set up a graph window on the screen. This is done using the following command. The window has a title printed on it, and the x-axis is labeled "time" and the y-axis is labeled "displacement". We then set the maximum value of x to be 10, and y to range from -0.12 to 0.12. Finally, we specify the window size as 500 by 300. Once this is done, we have a blank graph on the screen.

```
gdisplay(title='Mass and Spring: Graph', xtitle='Time', ytitle='Displacement',
         xmax=10.0, ymax=.12, ymin=-.12, x=0, y=500, width=500, height=300)
```

To draw a curve on the graph, we define a magenta-colored curve which we call "drawit".

```
drawit = gcurve(color=color.magenta)
```

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

```
#
# Mass and Spring simulation (Problem 4.HW.73)
# Everything is in MKS units
#
from visual import *
from visual.graph import *
#
# Define objects needed for display. We will use a helix object for the spring.
#
block  = box(size=(.02,.02,.02),color=color.red)
wall   = box(pos=(0,0,0),size=(0.005,.1,.1),color=color.blue)
spring = helix(pos=wall.pos,axis=-wall.pos+block.pos,radius=0.005,thickness=.002,
         coils=10,color=color.green)
#
# equilibrium position of the end of the spring.
#
spring.equil = vector(.12,0,0)
#
#------------------------------------------------------------
#
# Input Parameters needed in the program. Be sure to
# choose sensible values.
#
block.mass = 0.0              #mass of block in kilograms
block.pos  = vector(0,0,0)    #initial position of block meters
block.mom  = vector(0,0,0)    #initial mometum of block in kg m/s
spring.ks  = 0.0              #Newtons per meter
dt = 0.0                      #time step in seconds
t  = 0.0                      #total elapsed time (leave as zero here)
#
#------------------------------------------------------------
#
scene.autoscale = 0           # Turn off auto scaling
#
```

```
# Setup a graph window to plot things in
#
gdisplay(title='Mass and Spring: Graph', xtitle='Time', ytitle='Displacement',
         xmax=10.0, ymax=.12, ymin=-.12, x=0, y=500, width=500, height=300)
#
#
#
drawit = gcurve(color=color.magenta)
#
while(1==1):
    rate(100)
    t = t + dt
#
#      Compute the relevant physics part of the program to get the block
#      to oscilate at the end of the spring.
#
    spring.stretch = ??
    spring.force = 0.0
    block.mom = 0.0
    block.pos = 0.0
#
#      Specify that the spring conncets the wall to the block.
#
    spring.axis=-wall.pos+block.pos
#
#      Plot the x-coordinate of the block as a function of time.
#
    drawit.plot(pos=(t,block.x))
#
```