VPython and Glowscript

Welcome to the computational side of physics!

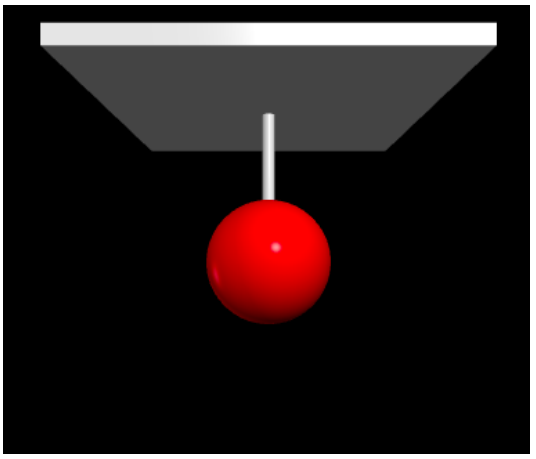
Lesson 5: Comparing simulations with data

Expected time: 60-90 minutes
Objectives: In this tutorial you will

- Create and analyze graphs of energy vs time
- Compare simulation results to data collected

This vpython assignment is paired with a lab, with the main objective of comparing the lab data to the simulation. There will only be one assignment to submit for both.

This vpython assignment is based on one created by Rhett Allain.



Create the Scene

Start by creating the holder, spring, and ball as shown at right. The code is given for you below.

```
# initial constants
L=.05 # relaxed length of the spring
k=15 # spring constant
pos0=vec(0,0,0) # the initial position of the ball, used later so useful as a definition early
g=vec(0,- 9.8,0)
t=0
dt=0.005

# Create the holder for the spring
holder=box(pos=(pos0+vec(0,L,0)), size=vec(.1,.005,.1))
# Holder is a distance L above the ball initial position, because the relaxed length of the spring is L.
# Thus spring is starting fully relaxed

# Create the spring and the ball at the end of it
ball=sphere(pos=pos0, radius=.02, color=color.red)
ball.v=vec(0,0,0)
ball.m=0.1
spring=cylinder(pos=holder.pos, axis=ball.pos-holder.pos, radius=.002)
```

Note the clever spring definition. Since the position is the same as the holder.pos, it stays rooted in the same spot. However, since the axis is the ball.pos-holder.pos, the length of the spring will change as the ball changes position (though that means we will have to update the axis in our loop).

You will likely change the length L and the spring constant k when you perform the actual experiment. Measurements will be easiest, however, if we consider the ball initial position to be (0,0,0); we'll explore this when we get to the lab piece.

Run your program to make sure the objects generate correctly.

Define your graph

After you have defined the mass on a spring, add a graph with four curves; Spring potential energy, gravitational potential energy, kinetic energy, and total energy. Note that all four curves should be on the same graph. Color code the graphs, and add a title that explains the colors (Ug is green, etc….)

If you need a refresher on creating graphs, see lesson 2.

Note that you haven't actually defined any energies yet, just stated that c1 is green and that the green graph is Ug. We will assign Ug to c1 in the loop later.

Define the loop

Next you will need the while loop. This will be largely the same as your previous while loops, with the following notable differences;

- You will have a spring force -kx which depends on a changing value of x
- You will need to update the spring axis, as mentioned above (which is only to make it look right, it doesn't affect the physics at all)
- You will need to define and update U_g, U_s, KE, and E_{tot}.
- You will graph the four above energies on curves c1-c4

Before we define the graphs let's get the program working correctly, starting with the spring force.

Update the forces, velocities, and positions.

The reason spring force has to be defined with a negative is that x is the direction of spring stretch and then the spring force must act in the opposite direction of it. How will we define x? Let's try an example. If the resting position of the relaxed spring was y=4 m, and the ball was currently at y=1 m, the x for the spring would be -3 m. That shows that it stretched 3 meters downward. Thus the force value would be up, since the negative from -kx would cancel with the negative from -3 to give a positive force.

Therefore, since we defined the ball's initial position to be when the spring was relaxed (and it's currently set at (0,0,0)), you can then make the x for spring force be the difference between the ball's position and its initial position (ball.pos-pos0).

Next define F_g, F_{net}, and update the ball's velocity and position in the same way as the last lesson.

Lastly update the cylinder axis as the difference between the ball's position and the holder position, just like the original definition. You likely want to update the time at this point as well.

Run your program. Does it look like it operates like a ball on a spring? Now seems like a good time to hit save as well.

Define the energies

You will need four definitions of energies, U_g, U_s, KE, and E_{tot}.

Gravitational potential energy only depends on the height of the ball, not the overall position; you can use ball.pos.y in your definition. Make sure not to use g in your definition; it is likely you defined g as a vector above, and you just want the value of 9,8 so that U_g is a scalar.

It is interesting to note that the initial gravitational potential energy of our system is zero, and that the rest of them thereafter will be negative. We are doing this because in the lab it will be easiest to measure x and h using the same origin; that will become more clear when we get to that point.

Kinetic energy depends on the *speed* of the ball, not the velocity. Example: As you may recall, the speed of a projectile at the end of its flight is found by finding the x and y velocities, then using the pythagorean theorem to find the *magnitude* of the velocity. Thankfully vpython makes this much easier; we can find the magnitude of a vector by simply using the mag(ball.v) command. Don't forget to square it by using a double asterisk (**2).

Spring energy depends on the *magnitude* of the stretch amount x; you can use the mag command again with your same definition for x as the spring force definition.

Don't forget to define the total energy as the sum of the other three. Speaking of total energy; think for a second about what the total energy of the system should be. What are all three values at the start, when you would let go of the ball? Should that value stay constant?

Define and update your curves

Finally, define curves c1 through c4 with their appropriate energies. Don't forget that you already assigned them via color when you created the graph with the four curves on it toward the beginning; make sure you are consistent with those definitions.

Take a close look at your graph and make sure it makes sense. Pick a couple of points in time and analyze whether each energy makes sense at that time. Also, is the total energy constant? It likely won't be perfectly; it is a simulation after all. It should be close though.

Don't forget to hit save.

Final thoughts

In the lab we will compare the qualitative shape, the quantitative values of the energies, and the quantitative period of oscillation of your simulation with real data. Lab Handout here.