Physics 231 – Lab 6
Energy

(equipment: Launchers, meter sticks, 2meter sticks, and ruler)

Objectives

In this lab, you will do the following:

- Apply the Energy Principle to an experiment
- Look at different forms of energy in a simulation of orbital motion

Background

According the Energy Principle, the change in energy for a system is $\Delta E_{sys} = W_{surr} + Q$, where W_{surr} is the work done by the surroundings. The kinetic energy of an object moving much slower than the speed of light is $K \approx \frac{1}{2}mv^2$. The gravitational potential energy for a pair of objects is $U = -Gm_1m_2/r$, where r is the separation and $G = 6.7\times10^{-11}\mathrm{N\cdot m^2/kg^2}$. The potential energy for the Earth and an object near its surface is $U \approx mgy$, where the y axis must point upward (away from the Earth). The Potential energy of a stretched (or compressed) spring is $U = \frac{1}{2}k_s|s|^2$.

In addition to these new ideas / tools, you're going to draw on the old ones of projectile motion; specifically

$$\Delta y = v_{i.y}\Delta t - \tfrac{1}{2}g(\Delta t)^2$$
$$\Delta x = v_{i.x}\Delta t$$

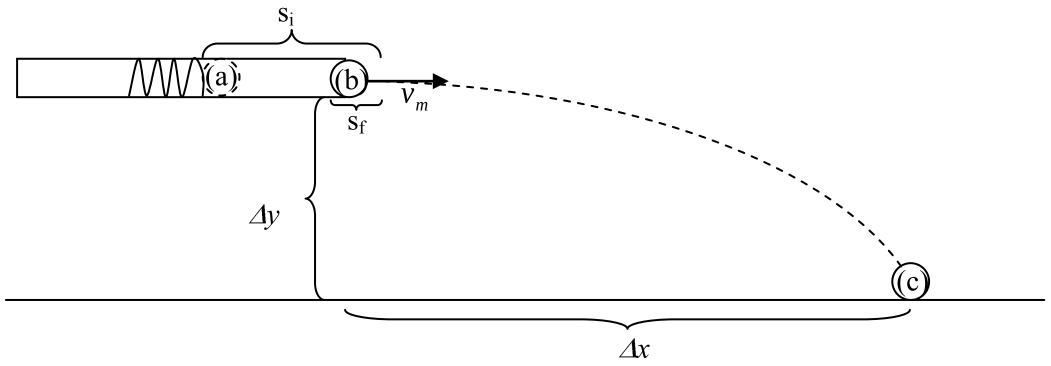
I. Ball shot from a Spring Gun.

You'll shoot a metal ball from a spring-loaded gun a few different times. The first time, you'll use the ball's motion to determine the spring constant. With that in hand, you'll be able to predict where the ball will land the next few times.

Determining k_s.

A. Theory

It's convenient to break this up into three instants: a) gun is cocked and ready; b)right when the ball is leaving the muzzle (and the spring is *almost* relaxed); c) and when the ball hits the ground.



- Consider the interval from (a) to (b); assuming you can measure the initial and final stretch of the spring, s_i & s_f, the ball's mass, m, and the muzzle speed, v_m , use energy considerations to develop an expression for the spring stiffness in terms of these.

- Unfortunately, we *don't* have a great way to measure the muzzle speed; however, it's pretty easy to measure how far the ball falls, Δy, and how far it flies, Δx, and, of course, g's well known. So, use a whiteboard to apply the projectile motion relations to the interval from (b) to (c) to write that expression for v_m.

> **WebAssign notes**: WebAssign may not handle triple-decker fractions well, so if you get your answer in the form $\frac{A}{\frac{B}{C}}$ flip it around to the form $\frac{A \cdot C}{B}$. The 'radical,' $\sqrt{\ }$, is produced by typing "sqrt(", Δ is produced by typing "Delta".

- Now substitute this second equation into the first so you have an expression for k_s in terms of all these things that we can measure. Check that the units are going to work out (if they don't, then you know something needs fixing.)
  > **WebAssign Note**: WebAssign is case-sensitive; s_f^2 *is produced by typing "s_f^2"..*

B. Experiment

In the spirit of 'it's all fun and games until someone loses an eye':
DON'T LOOK INTO A LOADED GUN!

From considering your theoretical treatment, you probably have a pretty good idea how to proceed.

Two measurements you can make right off the bat are the mass of the ball and the height of the muzzle off the ground.

Now, put the ball in the muzzle, but pause before you push it back with the plastic plunger – this is right where the ball will be (and how compressed the spring will be) when the ball launches off the spring, point (b). So let's measure this length. Just press the plunger gently against the ball and note the length of the plunger inserted. **Warning**: one end of the plunger is dimpled, to measure the depth, press with the *other* end.

It turns out that the compression of the spring is just about that minus 0.23 cm.

Okay, now press the ball back with the plunger until the spring *first* catches (there are three different catches, we'll start by using just the first.) Again, measure how deeply the plunger is inserted.

Again, subtract 0.23 cm from this to get how compressed the spring is going to be when it starts pushing on the ball, point (a).

You're just about ready to actually shoot the gun. Since, a priori, you have no idea where the ball will land, you're going to do a *trial* run first – you should watch carefully to see where the ball hits the floor. Next, you should put a sheet of carbon paper (black side down) just inside a pad of paper

and put the pad on the floor where you saw the ball hit. That way, when you shoot the ball again, the ball will leave a mark where it hits (a mark you can use in measuring Δx.)
Okay, you're ready – see the little metal tab with a pin hole on the top of the gun? That's the trigger, make sure 'all's clear' and you're ready to note where the ball falls, then fire away.

Use the carbon paper to help mark where the ball landed, and then measure Δx. Note: the carbon paper's mark may be quite faint, but it should be there. To be on the safe side, shoot a couple more times (every once in a while the gun might undershoot) and then take a good representative value.

You've got everything you need. Calculate the spring stiffness (don't forget to convert all values to base units – kg, m, s.)

Determining Δx.

A. Theory

Now that you know the spring constant, k_s, take the equation you developed on page 2 and rearrange it to solve for Δx.

With this, given different initial spring compressions, s_i, you'll be able to predict how far the ball will fly, and then actually let 'er rip and see how good your predictions are. Load and cock the gun, but this time push the ball back until it catches a *second* time, further down the barrel.

A. Experiment

Again, measure the depth of the plunger into the barrel and subtract 0.23 cm to it to get the initial stretch of the spring.

For this initial stretch, what distance do you predict?

Move the pad with the carbon paper to cover this new location on the floor so the ball should land on it. Then fire away. What distance did you measure? (Again, you may want to shoot a couple of times incase one shoot glitches.)

How do your measured and predicted values compare? (% difference; if it's not less than 10%, go back and check your work.)

II. Energies for Orbits

Back in Lab 3, you'd simulated the Earth orbiting the Sun. Now you'll track the forms of energy as they change throughout the orbit.

A. Planetary Motion

First, you will look at the energies for system consisting of a star and a planet orbiting around it. It is a very good approximation that the star does not move.

- Find the program from Lab 3 called "EarthSun.py" which predicts the motion of a planet around the Sun. Make a *copy* of that program and name it "PlanetEnergy.py", add your new group members' names in a comment.
- Make sure the program is set up to model the *elliptical* orbit of a planet around a star:
  - Star of mass 2×10^{30} kg located at < 0, 0, 0> m.
  - Planet of mass 6×10^{24} kg starting at < 1.5×10^{11}, 0, 0> m with an initial velocity of < 0, 3.8×10^{4}, 0> m/s.
  - Set the loop up to run until the time is 3×10^{8} s so that there are about two complete orbits.
- Comment out or delete any "`print`" statements inside the loop to make it run faster.
- Similarly, you don't need the force and momentum arrows, so you can (but don't have to) comment out or delete the lines related to those.
- Test that the program works as you expect.

Now you're ready to add the code for graphing kinetic and potential energies.

- To enable the graph-making capabilities, add the following line just *after* the statement "`from visual import *`"

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

- Add the following lines just *before* the "`while`" loop. They define and name the plots that will be made as the program runs.

```
Kgraph = gcurve(color=color.blue)    #create a graph's curve for kinetic
Ugraph = gcurve(color=color.green)   #create a graph's curve for potential
KUgraph = gcurve(color=color.red)    #create a graph's curve for K+U
```

The energies depend on the *magnitudes* of momentum and position vectors, so you'll add some code to calculate those. The function "`mag`" calculates the magnitude of a vector, so it'll be handy.

- Use it to find the magnitude of the planet's momentum and the separation of the star and planet. The following lines should go inside of the loop so that you can plot the various energies as functions of the time. (You have to fill in the gray parts.)

```
pmag = mag(          )
```

- Use these new variables to calculate the kinetic energy of the planet and the potential energy of the system. This should be done right after the previous lines. Note that

$$K \approx \tfrac{1}{2} m|v|^2 \approx \frac{|p|^2}{2m}.$$

```
K =
U =
```

Fill in the blanks in terms of the variable names in your program.

- Add points to the each of the energy vs. time graphs using the following lines.

```
Kgraph.plot(pos=(t,K))         #add a point to the kinetic energy graph
Ugraph.plot(pos=(t,U))         #add a point to the potential energy graph
KUgraph.plot(pos=(t,K+U))      #add a point to the K+U graph
```

Note: we *could* plot the *total* energy, K+U + rest energy, but since the rest energy doesn't change, we won't learn much from that (except that the rest energy is *way* bigger than K+U).

Pause and consider: What sign should the graph of *K+U* have? Qualitatively, what's the significance of its having that sign? How should its magnitude depend on *t*? Why? When the kinetic energy of the system increases, what should happen to the potential energy of the system?

- Run the program.

If your graphs of K, U, and K+U *aren't* as you've reasoned they should be, check your program.

Too Big Time-steps: In writing computer simulations, there's a 'just right' size for time steps – just small enough that the inherent approximations (position and thus force are essentially constant while updating momentum and momentum is essentially constant while updating position) are valid but not so small that your program takes longer to run than necessary. In Lab 3, you checked out how using too large a time step made your planet move unrealistically. Now see what happens to the corresponding energy plots – especially the *K+U* plot. Try "`deltat = 1e6`" (which is in seconds). In general, looking for that misbehavior is a good way to check whether a time step is the right size. After you've explored this, make deltat smaller again.

- Okay, change your `deltat` back to something reasonable.
- Change the plot commands to make graphs of the energies versus the separation (`rmag`) instead of the time.

Bound and Free: Run the simulation a few times with slightly greater initial speed each time – you should see the orbits vary from being nearly circular to pronouncedly elliptical, and finally a segment of a parabola (when the planet flies away). Notice what changes and what stays the same with the energy graphs.

- Change the initial speed back to 3.8×10^4 m/s.

Save and Upload PlanetEnergy.py.

5 Bonus Lab Points: Outside of class, modify your 'cart on spring' python program to simulate the gun launch. I'd suggest setting a condition so that when the final stretch length is achieved the gravitational force is 'turned on' and the spring force is 'turned off.' Email it to me when you've got it.

3D, Just for Fun: If you happen to have red-green 3D classes, you can see any VPython display in 3D! Just add this statement near the beginning of your program:

```
scene.stereo = "redcyan"
```