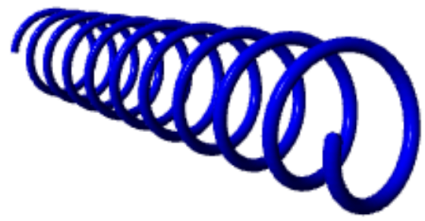
4. REAL SPRINGS AND FAKE PENDULUMS

In our previous calculations for simple harmonic motion, we used a fake spring. This spring exerted a force that was proportional to the position of the mass. I mean, you can pick your origin so that this is true, but it's not the same as a real spring. Real springs have some unstretched length.

We can model a real spring but we need to first introduce a new 3D object to represent these things. Here is the helix object in VPython.



Here is the code to produce this thingy.

```
3
4 spring = helix(pos = vector(0,0,0), axis=vector(1,0,0), radius=0.1, color=color.blue,
5 thickness=0.02,coils=10)
```

The helix has the following important properties:

- pos - this is the vector location of one end of the helix.
- axis - this is a vector from the pos to the other end of the helix.
- radius - if the helix was inside a cylinder, the radius would be the radius of that cylinder.

• thickness - the thickness of the “wire” in the helix.

• coils - the number of turns (loops in the helix)

Remember that this helix is just a drawing. It’s not real physics. We need to do the physics ourselves.

So, suppose you have a real spring like the one above. It has some unstretched length (L_0) and a spring constant (k). However, there will only be a force exerted by the spring if it’s either stretched so that it’s longer than L_0 or compressed to be shorter. Let’s define the length of the spring as the vector $\overrightarrow{L}$. This means that the length vector is from one end of the spring to the other end. With that, we can write the scalar version of Hooke’s law as:

$$F_s = ks$$

$$s = (|\overrightarrow{L}| - L_0)$$

Notice that my scalar version of the spring force does not have a negative sign. That’s because F_s is the magnitude of the force (direction doesn’t make sense there). We can calculate the value of the stretch of the spring as the difference in the length and the unstretched length. Since L is a vector, we have to first take the magnitude and subtract the unstretched scalar length.

This is fine, but we want a vector expression for the spring force. Here’s how to do that.

$$\overrightarrow{F}_s = -k(|\overrightarrow{L}| - L_0)\hat{L}$$

Oh, there’s something new we need to go over.

UNIT VECTORS

Let’s take the vector:

$$\overrightarrow{A} = <1,2,3>$$

It’s just a math vector so that we don’t need to worry about the units. Well, this is a 3D vector so that it has values in the x,y, and z direction. We already know that we can find the scalar magnitude of this vector.

$$|\overrightarrow{A}| = \sqrt{A_x^2 + A_y^2 + A_z^2} = \sqrt{1^2 + 2^2 + 3^2} = \sqrt{14}$$

We can think of vectors as having a magnitude and a direction, and that's the magnitude. But what about just the direction? We call this the unit vector ($\hat{A}$). If you want to say that symbol with your moth (or maybe telepathically with your mind) it's pronounce "A-hat" because it has that nice little hat on top of it. This means that a vector can be described using other its magnitude and direction as:

$$\overrightarrow{A} = |\overrightarrow{A}|\hat{A}$$

The unit vector in the direction of A would then be:

$$\hat{A} = \frac{\overrightarrow{A}}{|\overrightarrow{A}|}$$

Let's do this for the numerical values of the vector A.

$$\hat{A} = \frac{<1,2,3>}{\sqrt{14}} = <\frac{1}{\sqrt{14}}, \frac{2}{\sqrt{14}}, \frac{3}{\sqrt{14}}>$$

Suppose A really did have units. Well, then the vector A and the magnitude of A would both have the same units. When you divide A by the magnitude of A, you get a unitless quantity. That means that the unit vector $\hat{A}$ doesn't have units. Crazy, right?

Well, how do you do this stuff in VPython? Check out this code.

```
4 A = vector(1,2,3)
5 Amag = sqrt(A.x**2+A.y**2+A.z**2)
6 Ahat = A/Amag
7 print(Amag, mag(A))
8 print(Ahat, norm(A))
```

Here I am finding the magnitude of the vector A the manual way (with the square root). But wait! VPython has a built-in function to find the magnitude of a vector. It's just mag() and then put your vector inside there. We can also manually find the unit vector A-hat, but there's the function norm() that returns a unit vector in the direction of the vector A. Here's the output.

```
3.74166 3.74166
< 0.267261, 0.534522, 0.801784 > < 0.267261, 0.534522, 0.801784 >
```

See. The same. Let's go back to the springs.

VERTICAL MASS SPRING

How about a nice little spring calculation? Suppose a 100 gram mass is hanging from a spring that has a length of 5 cm and a spring constant k = 10 Newtons per meter. Not only can we model the motion if it starts 1 centimeter below the equilibrium point—but we can also build a very nice looking 3D model. Here's the code.

```
3
4  k = 10 #N/m
5  m = 0.1 #kg
6  g = vector(0,-9.8,0) #N/kg
7  L0 = 0.05 #m
8  s0 = m*mag(g)/k
9  top = box(pos=vector(0,L0/2,0), size=vector(0.05,0.005,0.02))
10 mass = sphere(pos=top.pos-vector(0,L0+s0+0.01,0), radius=0.01, color=color.red)
11 spring = helix(pos=top.pos, axis=mass.pos-top.pos, radius=0.005, color=color.blue,
12 thickness=0.001, coils=10)
13
14 t = 0
15 dt = 0.01
16 mass.m = m
17 mass.p = mass.m*vector(0,0,0)
18
19 while t<5:
20     rate(100)
21     L = vector(mass.pos-top.pos)
22     Fs = -k*(mag(L)-L0)*norm(L)
23     Fnet = mass.m*g + Fs
24     mass.p = mass.p + Fnet*dt
25     mass.pos = mass.pos + mass.p*dt/mass.m
26     spring.axis=mass.pos - top.pos
27     t = t + dt
```

I mean, I know you are pretty much a python expert by now—but let me go over some of the key ideas in this code.

- Line 8: s0 is the amount of stretch for the mass to be in equilibrium (so, I'm calling it the equilibrium stretch). You can find this by setting the spring force equal to the weight. No biggie—but it's the STRETCH not the position.

- Line 9: the top is just a box so that I can connect my spring to something. Of course you don't need this because it's not even real. If you want your spring connected to the emptiness of space, who am I to argue?

- Line 10: the mass (maybe I should have called it a ball). Notice that for the initial position, I'm defining it as a value BELOW the top plus the equilibrium stretch plus the initial position (so it's not in equilibrium).

- Line 11: This is the spring. I just talked about the helix object so we should be good—right?

- Line 21: Inside the loop, I need to calculate the L vector for each time interval.

- Line 22: Calculate the spring force. I'm just going to let you know that half of the time I get this thing backwards so that the minus sign is wrong. If your spring explodes instead of oscillates—that's what happened.

- Line 26: Don't forget that since the mass moves you have to also update the axis of the spring.

Imagine the following is an animation.

It's cool. It works.

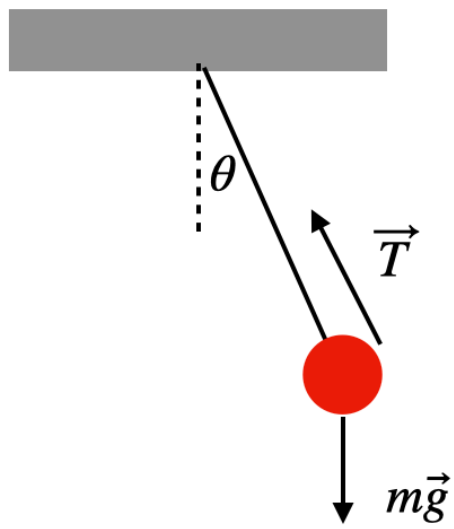
2D MASS-SPRING

Now for something really awesome. What if I give the mass an initial x-position so that it's not right below the top point? Let's just do it. I'm going to move it over in the positive x-direction by 3 cm. Here's the result (I included a trail since this isn't an animation).

I mean, that's sort of cool. It's a 2D spring and we didn't even change anything in the code except for the initial conditions. You can make it move in 3D if you also give the mass an initial velocity in the z-direction. I tilted the view so you can see it.

PENDULUMS

Most introductory physics textbooks include a section on the pendulum. Yes, I think they are cool—but they aren't really that simple. Here's a force diagram for a mass on the end of a vertical string.



Why is this a difficult problem? Although there are only two forces (the tension and the gravitational force), one of these forces is fairly problematic—yes, the tension. We can easily calculate the gravitational force but that's not true for the tension. As the ball moves in a circular path, the tension changes in both magnitude and direction. There's no equation for this force so you can't really calculate it (without some tricks). It's what we call a force of constraint in that the tension applies whatever force is necessary to keep that ball a distance R (the length of the string) from the pivot point.

I'm not going to go into a full solution (at least not now) so I will just say that if the angle (θ) is "small" then the angle will change just like a simple harmonic oscillator. We have the following solution.

$$\theta(t) = \theta_0 \cos(\omega t)$$

Where

$$\omega = \sqrt{\frac{g}{R}}$$

Let's model this. Here's the code.

```
 4 R = 0.1
 5 theta0 = 10*pi/180
 6 g = 9.8
 7 top = sphere(pos=vector(0,R/2,0),radius=0.002)
 8 ball = sphere(pos=top.pos+vector(R*sin(theta0),-R*cos(theta0),0),
 9 radius=0.01, color=color.red,make_trail=True)
10 string=cylinder(pos=top.pos, axis=ball.pos-top.pos, radius=0.001)
11
12 t = 0
13 dt = 0.01
14 w = sqrt(g/R)
15 while t<4:
16     rate(100)
17     theta = theta0*cos(w*t)
18     ball.pos = top.pos + vector(R*sin(theta),-R*cos(theta),0)
19     string.axis=ball.pos - top.pos
20     t = t + dt
21
22
```

There's not too much to add here, but why stop now?

- Notice that this is NOT a numerical calculation. Instead, it's just an animation. I have the solution for the angular position of the ball as a function of time and I'm using that to move the ball. I'm not making approximations during each time step like we've done before.

- For the ball, I need to put it in a location using the angle θ. If we start from the top point (which I made a sphere called "top") then it will go over in the x-direction an amount $R\sin\theta$ and down $-R\cos\theta$.

- Line 17: This is where the calculation happens. After that, I just move the ball and string.

Here's what it looks like.

Seems to work. I mean, it does work.

SPRING PENDULUM

Now for something fun. Is it possible to get a pendulum-like motion by replacing the string with a spring? I mean, it's just changing the t to a p—right? It should work. There's another big difference between a spring and string. We actually know how to calculate the force from a spring—it's not a constraint force. We don't even need to build this code—we already did that. The only thing we need to play with is the strength of the spring (the spring constant). We want this to be a large enough value so that the mass doesn't really get too far away from the path of an actual pendulum.

I haven't assigned homework yet, so here you go:

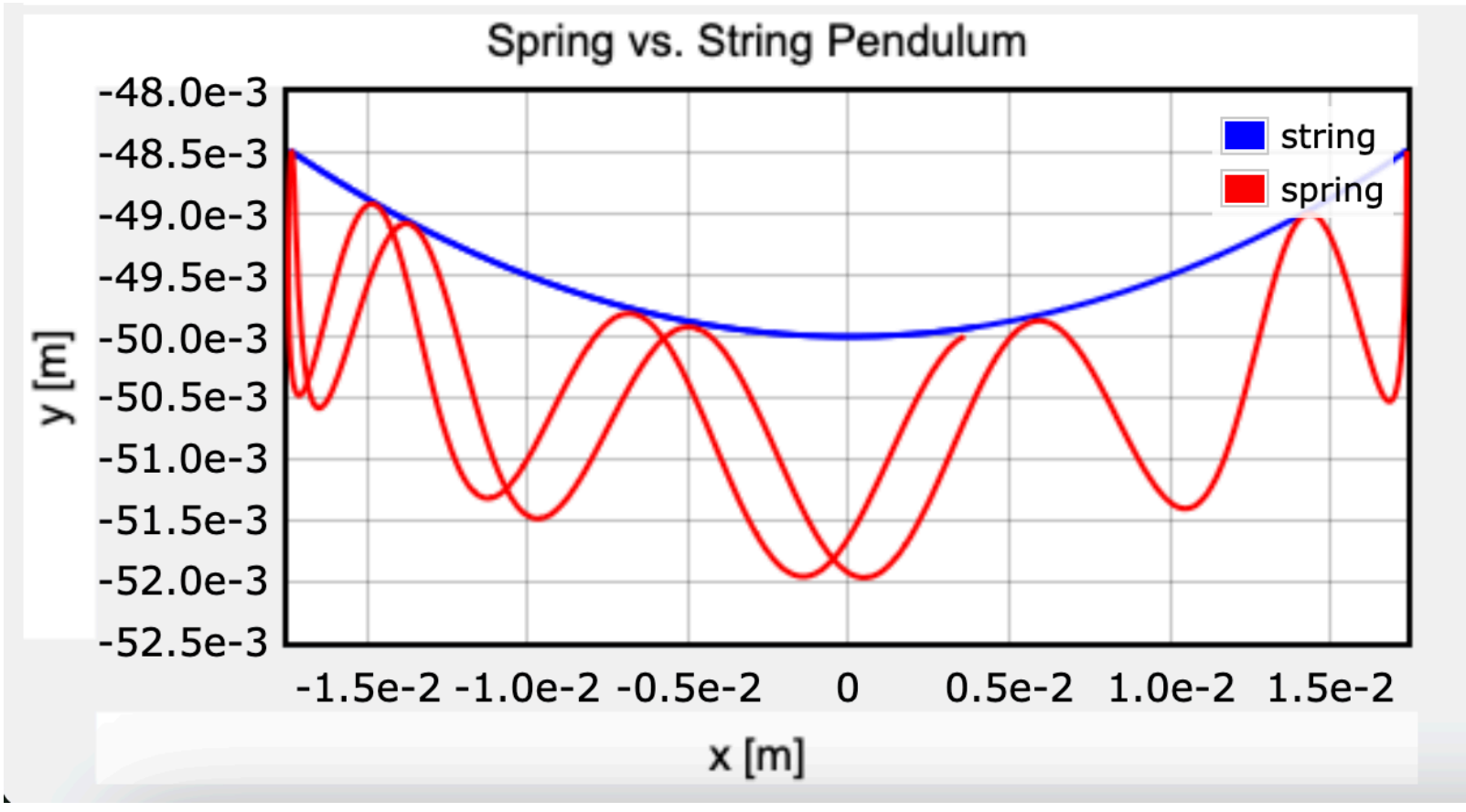
Homework

Use the pendulum model above. Add a second ball (I called it "ball2") that starts at the same position as the pendulum ball. However, this one has a spring connected to it instead of a string. The unstretched length of the string is R (same as the length of the string) and give it a spring constant of something like k = 1000 Newtons per meter. Decrease the time step to 0.001 seconds.

Now create a plot of the trajectory (x vs y) for both balls.

Key

I'm not going to show you the code, but here's the output.



Notice that the two solutions don't match up—but they also kind of DO match. In this case, I'm using a string length of 0.5 meters. The pendulum motion does exactly what we expect (because we used an analytical solution for this). The springdulum (I just made up that word) deviates from this trajectory. If you look at the values of the position, the difference between the two paths is actually quite tiny. So, in a way, this spring solution works.

If you think about it, a string really is a spring. When you pull on these things, they stretch —at least a tiny little bit. How else would a string know the EXACT force to exert on a swinging mass? There isn't a computer in a string, it's just a string. In fact all constraint forces can be modeled with springs. Why would we do that? Because remember, with a spring force we can actually calculate the force and then use that to update momentum. You can't do that with a force of constraint.

OK, there's a BUNCH more cool stuff you can do with springs but I think we will just look at that later.