

PHY221 Lab-04-4: Modeling Simple Harmonic Motion

The objective of this activity is to write a VPython program that simulates an oscillating mass-spring system. Assume that the spring is compressible and that the only force on the object is the force by the spring.

Background

The force by a spring on an object attached to the spring is given by Hooke's Law:

$$|\vec{F}_{spring}| = ks \quad (1)$$

where s is the distance the spring is stretched or compressed from its unstretched, uncompressed length. Let's orient our spring so that the end of the spring when it is unstretched is at the origin of the coordinate system, as shown below.

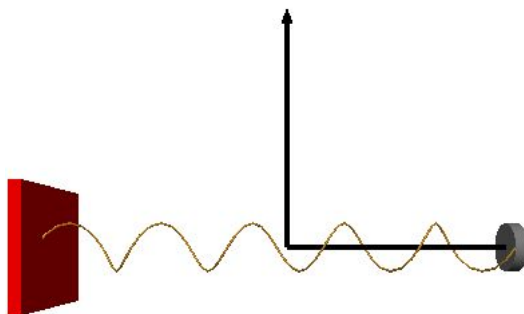


Figure 1: Orientation of the spring and coordinate system.

In this orientation, the net force on the object is

$$\vec{F}_{net} = \langle -kx, 0, 0 \rangle \quad (2)$$

Note that when the x -position of the object (and end of the spring) is positive, the force by the spring on the object points to the left. When the x -position of the object (and end of the spring) is negative, then the spring is compressed and the force by the spring on the object points to the right. By orienting our spring and coordinate system in this way, it makes it easy to write the force by the spring on the object as a vector in terms of x , the position of the object, so that it always gives us the correct direction of the force.

Initial Conditions

We need to define initial conditions and constants for our model.

| | |
|--|---|
| unstretched length of the spring | 0.2 m |
| initial distance the spring is stretched | 0.1 m |
| position of the bracket | $\langle -0.2, 0, 0 \rangle$ m |
| dimensions (size) of the bracket | x: 0.01 m, y: 0.05 m, z: 0.05 m |
| ball's radius | 0.01 m |
| ball's initial position | the spring is initially stretched 0.1 m so the ball is at $\langle 0.1, 0, 0 \rangle$ m |
| ball's initial velocity | $\langle 0, 0, 0 \rangle$ |
| ball's mass | 0.2 kg (or whatever mass you used in your experiment) |
| spring's position (left end) | bracket.pos |
| spring's radius | 0.005 m |
| spring's orientation (axis) | ball.pos-bracket.pos |
| spring stiffness | 10 N/m (or whatever stiffness you measured in your experiment) |

Writing your program

1. You are going to write this program from scratch. However, it is useful to have an old program open, such as your projectile motion simulation, to remind you of the syntax of certain commands and how to perform numerical integration.
2. Start with a blank file in VPython. Save it and name it something like *oscillator.py*. Remember to include the *.py* filename extension.
3. You will need to begin your program with `import` statements. Be sure to import the division library, the visual library, and the visual.graph library.
4. Create a bracket using the `box()` statement. Name it `bracket` and give it the position and size indicated above.
5. Create a ball using the `sphere()` statement. Name it `ball` and give it the initial position and radius indicated above. Note: we are placing it at a certain position so that the spring is initially stretched 0.1 m from equilibrium (which is at the origin).
6. Run your program. You should see a 3-D scene with the bracket at the left and the ball to the right (of the origin which is at the center of the window).
7. Now, create a spring using the code below.

```
spring=helix(pos=bracket.pos axis=ball.pos-bracket.pos, radius=0.01, color=color.orange)
```

Note that the axis is a vector written as “final - initial” where the final position is the right-end of the spring and the initial position is the left end of the spring.

8. Run your program. You should see a spring that goes from the bracket on the left to the ball on the right.
9. Define the mass, initial velocity, initial momentum, clock reading, time step, and spring stiffness. A time step of about 0.01 s is reasonable if the period is on the order of a second. If the period is much smaller than a second, then you will need to decrease the size of the time step to achieve accuracy.
10. Run your program and be sure there are no errors. Note that it will not do anything yet because we have not performed the numerical integration required to calculate the position of the object at various times.

All of the code above simply sets the scene, defines constants, and defines initial conditions. It contains NO PHYSICS. At this point, we have to apply Newton’s second law and use numerical integration to calculate the momentum, velocity, and position of the object for each small time step.

11. Write a `while` loop. For now, make it an infinite loop, meaning that it will run indefinitely. If you want to make it stop after a certain time, then have it run for at least 10 second or so.
12. Calculate the net force on the object using Hooke’s law.

$$\vec{F}_{net} = \langle -kx, 0, 0 \rangle \quad (3)$$

Note that x in this equation refers to the x-position of the ball. So write the above statement using VPython syntax, but refer to x as `ball.pos.x`.

13. Use numerical integration to calculate the momentum and position of the object, just as we’ve done for each of our simulations that we’ve written so far. Also, remember to update the clock reading for each time step.

14. The ball moves during each time step. Thus, at the end of the loop, you have to redraw the spring after each time step. Note that the spring always points from the bracket to the ball. Thus,

```
spring.axis=ball.pos-spring.pos
```

should be the last line in the while loop.

15. Run your simulation. Is its motion similar to what you noticed in the lab for an oscillating mass-spring system?
16. You will want to view a graph of x vs. t for the ball. Thus, add a single graph of x vs. t to the simulation. Before the while loop, you have to create the graph. Inside the while loop, you have to add a data point for $(t, \text{ball.pos.x})$ to the graph for each time step. Be sure to correctly label the axes and title the graph as something like "x vs. t for an oscillating mass-spring system."
17. Run your program. Is the graph of x vs. t similar to what you expected based on experiments that you've done?

Application

1. Print data for t and ball.pos.x for at least one cycle. Examine the data and use it to determine the period, frequency, and angular frequency of the motion.
2. Calculate the period theoretically using $f = 1/T$ and $\omega = 2\pi f$ and $\omega = \sqrt{k/m}$. Compare the measured period to the theoretical period based on the mass and spring stiffness. What is the % difference?
3. Compare the simulated result for angular frequency to what you measured in lab. How do they compare? Calculate the % difference.
4. Change the initial position of the ball to 0.15 m (thus, increasing its amplitude). Measure the period. Is it the same or different from what you measured before? Does amplitude affect period (and likewise frequency)?
5. In real-life, air resistance (drag) slows the oscillator and causes its amplitude to decrease with each cycle until eventually the amplitude is zero and the system does not oscillate at all. For low-speed oscillators, the drag force is proportional to the speed of the object (instead of speed-squared as it is for higher speed objects like sky divers, for example).

$$\vec{F}_{drag} = -D\vec{v} \quad (4)$$

Define a drag constant $D = 0.1 \text{ N/(m/s)}$. Add the drag force to the expression for the net force on the ball. Note that \vec{v} in the above expression is the velocity of the ball, `ball.v`.

Run your program. Sketch what you observe below.

This is called a *damped oscillator*. By taking a math course in differential equations or a physics course in classical mechanics, you can learn how to solve the exact expression for $x(t)$ in this case.