Hooke's Law and Simple Harmonic Motion on a Vertical Mass-Spring System

Physics 4AL, Spring 2022, May 22
Lab Section 7, Group 5

Lana Lim, Faraz Murshed, Aaron Zhao

Abstract

The purpose of this lab was to study Hooke's Law measurement of spring constant and simple harmonic motion on a vertical mass-spring system. The spring constant was calculated by measuring a spring's equilibrium position when attached to different masses. Then, an Arduino setup acted as the mass in a vertical mass-spring system to track the position and acceleration of an oscillating spring. A vertical mass-spring system experiences simple harmonic motion about the equilibrium position. This contrasts with a horizontal mass-spring system where its motion is centered at the rest length of the spring. Using our setup, we were able to acquire ultrasound and accelerometer data so that we could analyze 3-4 complete oscillations using complex fitting. We constructed best fit sine functions that copied the curves for both datasets. We then differentiated our ultrasound best function and compared the derived curve with the best fit acceleration curve. Then, we were able to realize the accuracy of our experimental setup and complex fitting analysis.

Introduction

A spring-mass system is an excellent form of studying simple harmonic motion and oscillations. The spring acts as the restorative force that causes the system to oscillate and experience periodic motion. In a vertical mass-spring system, weight due to gravity changes the spring's equilibrium position from its normal rest length. The only two forces acting on the mass in a vertical system is its weight and the spring. Weight is held constant and the force by the spring changes depending on when it is being compressed or decompressed. Force is 0 N when the spring is at equilibrium position. When the mass is displaced or released, it oscillates about the spring's equilibrium position. The total force acting on the mass in a vertical system follows the equation

$$F_{net} = -k(y - y_1).$$

y1 is the spring's equilibrium position and *y* is its current position. If we set *y* = 0.00 m, net force becomes

$$F_{net} = -ky.$$

Note that this is the same equation for calculating the total force in a horizontal mass-spring system but oriented on the y-axis.

Methods

Equipment

- HC-SR04 Ultrasonic Sensor
- Arduino Uno
- Jumper wires
- Breadboard
- HC-06 Bluetooth module

- MPU-6050 Accelerometer
- Battery holder
- Tape
- String
- Spring
- Stand
- Weigher
- Masses
- Ruler
- Laptop
- Python

Setup

To find the spring constant of the spring, first fix the spring to the small rod on the stand. Then, hang masses from the spring and record the displacement using a ruler in Figure 1. You can combine masses by using the hooks at the ends to increase the weight.

After calculating the spring constant, set up the Arduino as the mass in a vertical mass-spring system. Verify that the voltage of the batteries powering the Arduino are at the most optimal setting. Voltages greater than 1.37 V are recommended for accurate data collection. Then, connect the ultrasound sensor, HC-06 Bluetooth module, and accelerometer to the Arduino. The hardware will be responsible for tracking the spring's oscillations. The circuit was assembled like in Figure 2. Next, guide a string into the holes on the Arduino board like in Figure 3. Tape the battery holder and Arduino setup together to make one compact unit and weight it. When you hang the Arduino setup by the end of the spring, ensure that the ultrasound sensor is facing downwards and is resting flat in the air. Also, ensure sure that the ultrasound will reflect off the ground and clear any other part of the stand or objects that are directly beneath it. The vertical spring-mass system should be assembled like the one in Figure 4.

Procedure

The spring's displacement when attached to different masses will be used to find the spring constant. Plot a graph with the spring's displacement on the y-axis, the force it experiences on the x-axis, and the best fit line using Python. Force is calculated by multiplying the weight of the mass by Earth's acceleration of gravity.

$$F = mass\ (kg) * 9.8\ (\frac{m}{s^2}).$$

The spring constant is the inverse slope of the best fit line. We also calculated the spring constant's uncertainty using the covariance matrix.

Our lab requires a HC-SR04 Ultrasonic sensor, HC-06 Bluetooth Module, and a MPU-6050 Accelerometer in the Arduino setup. This combination of hardware and software allows us to measure the spring's acceleration, position, and time when experiencing harmonic motion. The accelerometer measures acceleration based on the mass-spring system that is inside of it. Some force is required to stretch and compress the spring, which causes the spring to push or pull the mass by some displacement. The acceleration from the force required to displace the mass is what's measured. The MPU-6050 Accelerometer is a more portable, compact version of a standard accelerometer that can be connected it to our Arduino board using the pins on its surface. The ultrasound sensor records the time it takes for a chirp to be omitted from it, bounce off of a reflecting surface, and return to its detector. It also measures the distance between the sensor and reflecting surface. In this case, the reflecting surface will be the ground. The Bluetooth Module allows the Arduino code to wirelessly communicate with the accelerometer and ultrasound sensor. This reads in the important values of the spring-mass system in real time without the setup being connected to the laptop. When running the experiment, the Arduino code

receives 5 columns of data in the serial monitor. The first value is the elapsed time in milliseconds and the second one is the distance between the sensor and ground in centimeters. The third, fourth, and fifth values are the accelerations of the setup in the x, y, and z directions respectively.

Before the vertical mass-spring system experiences oscillations, re-calibrate the accelerometer in the axis of oscillations. We will be using the x-axis for this experiment since the ultrasound sensor is facing downwards.

To calibrate the Arduino data from the accelerometer, we define a calibrate function that takes in two points from two different axes and returns the slope and intercept for the calibration plot. We use this slope and intercept with the acceleration data from the accelerometer to get the total acceleration. Adding 9.81 to this number to account for the acceleration due to gravity gives the relative acceleration of the Arduino board due to the spring oscillation. We plot this data relative to time.

Pull the mass slightly down and let for the system to oscillate about 15 cycles. Data will be transmitted from the accelerometer and ultrasound sensor to the Arduino's serial monitor via Bluetooth. From the first two columns of data, plot the Distance vs. Time graph in Python. Then, plot the Relative Acceleration vs. Time using the calibration function created earlier.

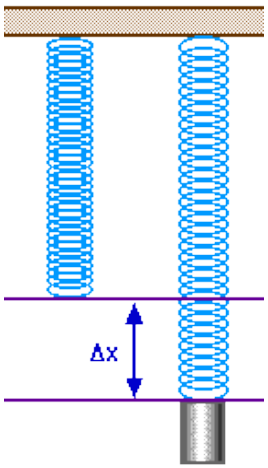

Figure 1. A spring is fixed to the small rod on the stand. We can attach a mass at the end of the spring and measure the displacement from its original equilibrium position. This was done with differently weighted masses.

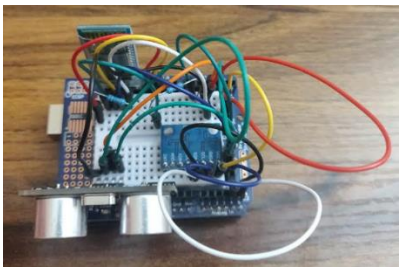

Figure 2. Depicts the circuit diagram and setup of connecting the ultrasound sensor and accelerometer to the Arduino board. This will be connected and taped to a battery holder to make one compact unit hanging from the spring.

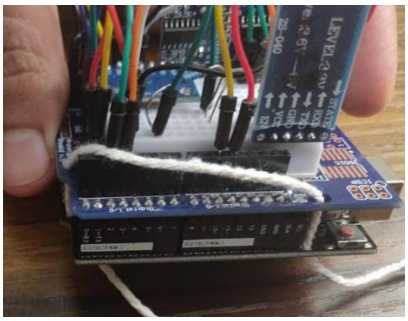

Figure 3. Pass the thread through the four holes of the Arduino setup in this fashion.

Figure 4. The vertical mass-spring system. The thread holds the Arduino setup in the air and is attached to the end of the spring that is held up by the stand. The ultrasound sensor is facing downwards to capture the spring's position and time relative to the ground.

Results

The data recorded from the ultrasound sensor displayed in Figure 5 shows fairly consistent oscillation, with an approximate amplitude of 0.025m centered at 0.15m. In addition, we can approximate a period and thus approximate an omega value. We use these approximations as guess parameters in calculating the best-fit sine function for distance.

Similar to in Figure 5, we see consistent oscillation in the data recorded from the accelerometer as well. We can approximate an amplitude of 2m/s^2 and an offset of –0.5 m/s^2, and approximate omega in the same way as well. We use these approximations once again as guess parameters when calculating the best-fit sine function for acceleration.

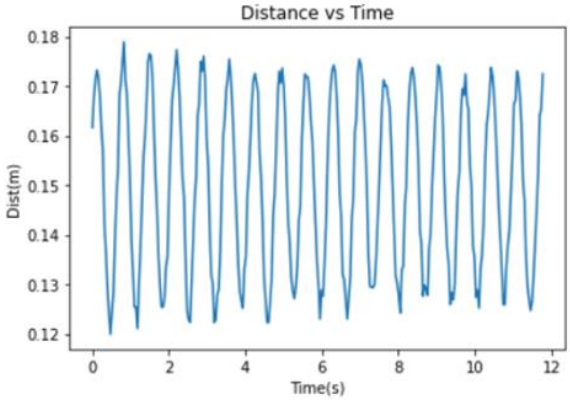

Figure 5. Ultrasound data for a vertical mass-spring system.

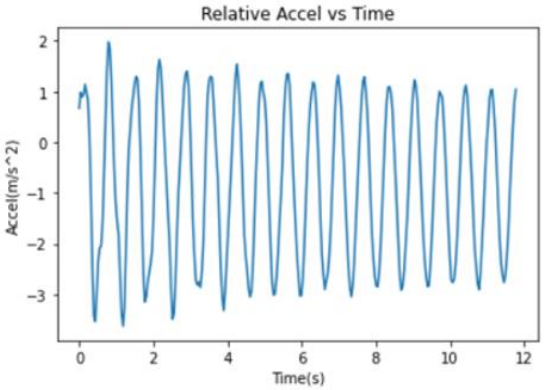

Figure 6. Accelerometer data for a vertical mass-spring system.

Figure 7 displays the best fit sin function fitted to ultrasound data on top of the actual ultrasound data. The approximation is quite accurate, neatly overlapping the data while averaging out amplitude peaks. Through using the least squares function and best parameters, we are able to create a standardized representation of the data with a well-defined amplitude, omega, offset, and phi.

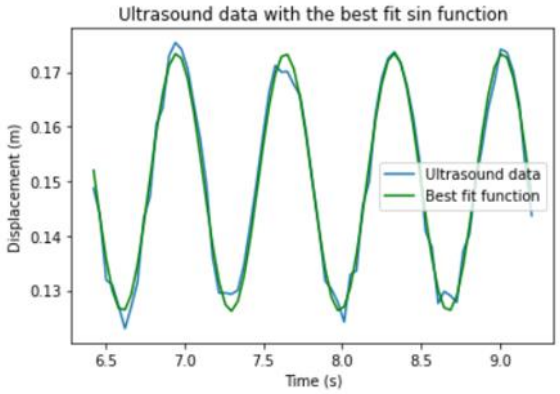

Figure 7. Best-fit sin function fitted to ultrasound data.

Through the same process as in Figure 7, Figure 8 shows the best fit sin function for acceleration on top of the raw accelerometer data. The approximations lines up well and is also accurate. The best-fit amplitude, omega, offset, and phi values for both ultrasound and acceleration data are listed in Table 1 for reference.

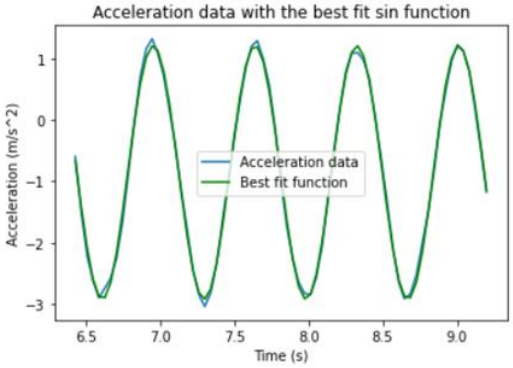

Figure 8. Best-fit sin function fitted to accelerometer data.

	Amplitude	Omega	Offset	Phi
Ultrasound Data (m)	0.023	9.137	0.149	0.534
Acc Data (m/s^2)	1.917	9.154	-0.403	0.391

Table 1. Amplitude, Omega, Offset, and Phi for best-fit sin functions.

Figure 9 plots the best fit acceleration sin function from Figure 8 with the plot of acceleration derived from the ultrasound data. The data was derived from the best parameters used in the ultrasound graph, and calculated by taking its derivative twice. The plots do not line up exactly. This may be due to differences in the data recorded by the ultrasound sensor and accelerometer on the Arduino board.

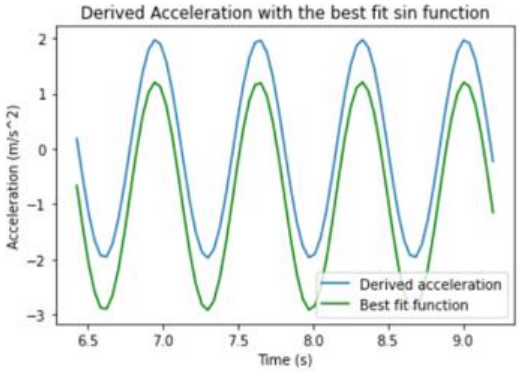

Figure 9. Best-fit sin function fitted to accelerometer data and derived acceleration from differentiation of best-fit ultrasound data.

Conclusion

The original goal of this experiment was to build upon our existing knowledge of the vertical mass-spring system and to learn about complex fitting in physics applications. Along with understanding the experimental and conceptual setups, we were able to perform further analysis upon the resulting oscillations using Python and sinusoidal functions.

If we inspect Figures 5 and 6, we can see that both the ultrasound and accelerometer datasets seem to demonstrate the expected oscillatory behavior.

Figure 7 shows that we were able to construct a best fit sine function within Python using residuals that exactly mimics the behavior of the ultrasound data. Figure 8 exhibits a similar result, but for the acceleration data. Therefore, we were able to be construct adequate best fit functions.

Figure 9 demonstrates the potency of our data and complex fitting. As shown, symbolic differentiation resulted in a curve that does not exactly cover the best fit function, as the others did. However, it does behave in a completely aligned manner to the best fit sine function; there are just some small gaps at the local minima and maxima. Therefore, we can conclude that our experimental setup of the vertical-mass spring system and our complex fitting analysis provided meaningful and understandable results.

If we were to reproduce this experiment, there could be some minor improvements in the physical aspects. In the actual performance, we pull the mass down slightly and let go of the setup. When we did this, there is the chance that we unintentionally applied a small force that could create some noise in the data or variance in the acceleration behavior. A mechanical mechanism to perform this would be more controlled. Additionally, the Arduino was connected to the spring system using a thread. There is a chance that our threading was not optimal, or that the thread itself was not the ideal type for oscillations. Lastly, issues with the Arduino itself (wiring, Bluetooth, ultrasound sensitivity, etc.) or the calibration process could create minimal deviations. However, this part is not exclusive to this experiment. Therefore, making some of the aforementioned improvements could lead to results that are optimal.