Analyzing Harmonic Oscillations in a Spring-Mass System

Clyde Villacrusis, Nathan Joshua, Wendy Mapaye

Physics 4AL, Fall Quarter 2023, November 19, 2023
Lab Section 4

Abstract

The goal of this lab was to measure and analyze the relationship between the distance and acceleration of a vertically hanging spring mass system. To do this we hung an Arduino with an ultrasonic sensor and accelerometer to an elevated spring that was attached to a bar. A string threaded through the Arduino, allowing it to be tied from the bottom of the spring. From a position of rest, the Arduino was pulled down slightly and then released, creating a series of oscillations. We took our ultrasonic sensor and accelerometer data and used Python to translate it into a best-fit sine function. We hypothesized we could find the acceleration and angular frequency of the oscillating arduino using the accelerometer and ultrasonic sensor and comparing it with the position function: $y(t) = A\sin(\omega t + \phi) + C$ and its corresponding acceleration function, $a(t) = -A^2\sin(\omega t + \phi)$.

After running the experiment, we were able to confirm our hypothesis because our distance vs. time and acceleration vs. time graphs both had a nearly identical sinusoidal shape that resembled simple harmonic oscillators, and we were able to create the corresponding sin-fit functions that overlapped this data. Additionally, we were able to extract a best-fit angular frequency of 9.26 ± 0.01 rad/s using *curve_fit*. So, in turn, this value is within the 2 standard deviation from the prediction of angular frequency we got, which was 11 ± 1 rad/s.

Introduction

The goal of this lab activity is to extract the best-fit values of amplitude and angular frequency of a vertically hanging spring mass system undergoing simple harmonic motion and to confirm that the theoretical acceleration of the system agrees with empirically obtained acceleration measurements using the MPU6050 accelerometer. The experiment involves an Arduino based system mass tied to a spring via string that is stretched in the vertical direction to produce simple harmonic oscillations. An image of the experimental setup can be seen in **Figure 1**.

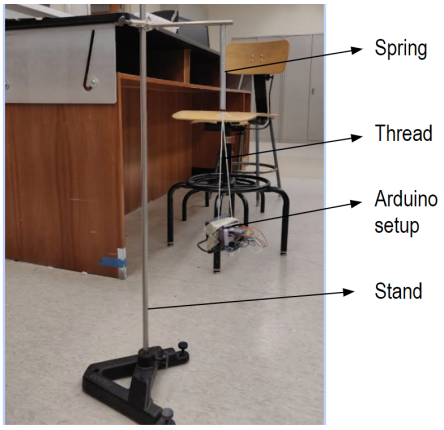

Figure 1: Representation of the spring-mass system

A vertically hanging spring mass is described by the following second order differential equation,

$$y'' + g + \frac{k}{m}y = 0$$

Equation 1.1: Differential equation describing a vertically hanging spring mass obtained using Newton's Second Law and Hooke's Law where k is the spring constant, m is the mass, y is the position and g is acceleration due to gravity

This system experiences simple harmonic motion at a shifted equilibrium due to the effects of acceleration due to gravity. A simple change of coordinates can be used to convert the differential equation to the standard form of simple harmonic oscillators,

$$y'' + \frac{k}{m}(\frac{mg}{k} + y) = 0 => y_s'' + \frac{k}{m}y_s = 0$$

Equation 1.2: New differential equation describing a vertically hanging spring mass at a shifted equilibrium

We will use the following relation to rewrite the differential equation in terms of the angular frequency of the spring mass,

$$\frac{k}{m} = \omega^2$$

Equation 1.3: Relationship between spring mass and its angular frequency ω

Solving the differential equation given by **Equation 1.2** will give us the equation that describes the position of the spring mass as it changes over time.

$$y(t) = Asin(\omega t + \phi) + C$$

Equation 2.1: Position of spring mass as a function of time where A is the Amplitude, ω is the angular frequency, ϕ is the phase shift and C is the offset of the simple harmonic motion

Taking the first derivative of y will give us the velocity of the spring mass as it changes over time,

$$v(t) = A\omega cos(\omega t + \phi)$$

Equation 2.2: Velocity of spring mass as a function of time

The second derivative of y will further give us the theoretical acceleration of the spring mass that we want to verify using the accelerometer,

$$a(t) = -A\omega^2 sin(\omega t + \phi)$$

Equation 2.3: Acceleration of spring mass as a function of time

The mass-object in our experiment is an Arduino based setup that collects data using an ultrasonic sensor and accelerometer. Arduinos are micro-controllers that collect and analyze data while controlling various systems such as LEDs, ultrasonic sensors and bluetooth modules. Ultrasonic sensors are used to measure the distance and time from objects using soundwaves. They sense proximity and detect objects under high reliability. In particular, the HC-SR04 Ultrasonic Sensor used in the experiment emits a chirp using its transceiver which is reflected off of an object and returns to its detector. The distance is then calculated using the following equation,

$$d = \frac{1}{2} v_s \Delta t$$

Equation 3: Distance between object and HC-SR04 Ultrasonic Sensor where v_s is the speed of sound and Δt is the time elapsed since the soundwave left the transceiver until it returns to the detector

The accelerometer is a module used to measure its own acceleration in the x, y and z axes. The MPU6050 accelerometer is a capacitive accelerometer that uses a mechanism involving an internal spring-mass system and capacitor plate to measure acceleration. The acceleration values for each axis will not have the appropriate units at first, which is why the accelerometer needs to be calibrated prior to use.

The data that is obtained using the ultrasonic sensor and accelerometer will require analysis before **Equation 2.3** can be applied to compare the theoretical acceleration with the accelerometer readings. This analysis can be done using Python. We hypothesize that our Python based analysis of the data collected by the accelerometer will yield values for acceleration of the Arduino setup that agree with the theoretical prediction given by **Equation 2.3**. Moreover we hypothesize that the angular frequency that we obtain from the analysis of the data collected by the ultrasonic sensor will agree with our hand-measured value for angular frequency.

Methods

Equipment

- Arduino
- Laptop
- String
- Spring
- Spring Stand
- HC-SR04 Ultrasonic Sensor
- HC-06 Bluetooth module
- MPU6050 Accelerometer
- Tape
- Four 1.5V batteries
- Voltmeter
- Weighing Scale

Begin by setting up the Arduino board so that it has the HC-SR04 Ultrasonic Sensor, HC-06 bluetooth module and MPU6050 accelerometer mounted on it as shown in **Figure 3**. The circuit diagram that describes the wirings for the setup can be seen in **Figure 2**. Following completion of the wiring process, the Arduino must be connected to a computer via USB so that the required code can be uploaded to the device from the Arduino IDE. The required code is written in C++ and uses the SoftwareSerial library for serial communication between the computer and the Arduino's digital pins. It also uses Wire, I2Cdev and MPU6050 libraries to read accelerometer values.

In the code, one pin is set as a receiver and another is set as a transmitter to enable bluetooth communication with the computer. The remaining code ensures that the time and distance data collected by the Ultrasonic sensor as well as the acceleration in x, y and z axes as determined by the accelerometer are displayed in the Serial Monitor of the Arduino IDE. Once the code is uploaded, remove the USB cable and connect the setup to a battery holder. The battery holder must contain four batteries with at least 1.4 V each so that the setup can continue to function without losing power during data collection. This can be checked beforehand using a voltmeter.

Before using the accelerometer, it must be calibrated to reflect the appropriate units in each axis. This can be done by orienting the Arduino such that the accelerometer experiences acceleration due to gravity along each axis and measuring the respective accelerometer values using the Arduino code. Using the collected accelerometer data, we can extract the fit-coefficients with numpy's polyfit() function which can then be used to convert the accelerometer values to the appropriate units. **Figures 4-6** show the x-axis calibration, y-axis calibration, and z-axis calibration respectively. After calibrating the arduino, loop a string through the Arduino's holes and tape the battery holder to its bottom. Measure the weight of this completed Arduino setup using a weighing scale.

Now place the spring stand on a level surface and hook the spring on it. Before starting the experiment, we must find out the value of the spring constant for this spring. This can be done by attaching different masses to the spring and measuring the displacement caused in its length. Plot the displacement vs. mass graph using the matplotlib library. Use numpy's polyfit() function to create a best-fit line and solve for k using **Equation 4.1**. The associated error can also be calculated using error propagation as described by **Equation 4.2**.

$$k = \frac{g}{m_{slope}}$$

Equation 4.1: The spring constant found using the slope of the best-fit line m_{slope}

$$\delta k = |k| \frac{\delta m_{slope}}{|m_{slope}|}$$

Equation 4.2: The error of the spring constant K in a spring mass system, where delta s is the slope

Once the spring constant is known, use the string to tie the Arduino setup to the bottom of the spring as shown in **Figure 1**, while making sure that the ultrasonic sensor faces downwards, perpendicular to the surface. Ensure that the hanging Arduino setup is about 15-20 cm away from the surface so that it does not generate noisy data. Gently pull down on the Arduino to displace it from equilibrium, and release it to produce back and forth oscillation. The ultrasonic sensor and accelerometer will generate values based on their workings described in the introduction. Record the time, position, and x acceleration data that is generated on the serial monitor for about 15 oscillations and convert it to .txt files for analysis using Python. It's important to note that we have assumed that there is no effect of air resistance in this experiment.

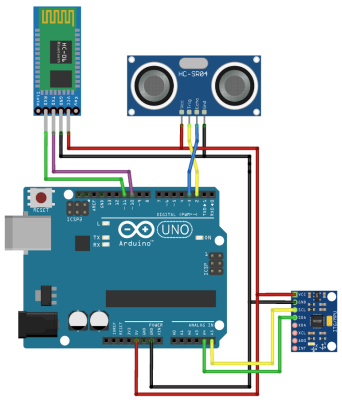

Figure 2: Arduino circuit board with ultrasonic sensor, bluetooth module, and accelerometer

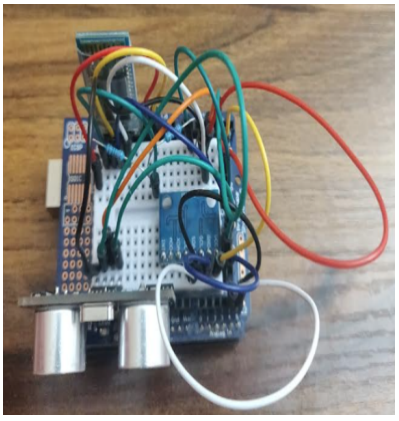

Figure 3: Our wired Arduino

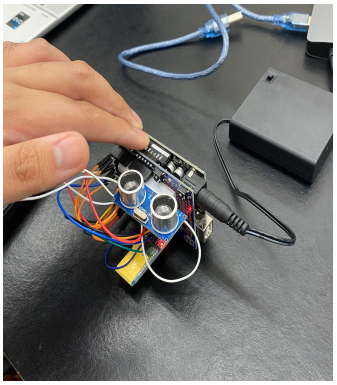

Figure 4: X-axis calibration orientation

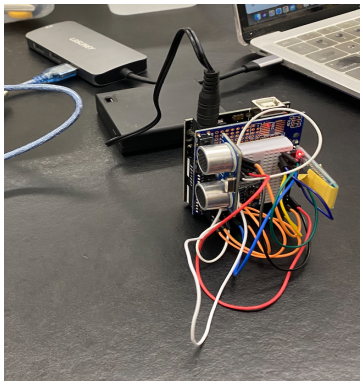

Figure 5: Y-axis calibration orientation

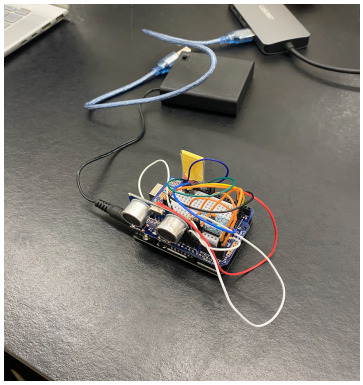

Figure 6: Z-axis calibration orientation

Upload the .txt files to a Python IDE. We can now graph the position vs. time data using the matplotlib library to see what it looks like. Clip the data so that you have 3-4 oscillations. Subtract the start time from your time data to zero out the phase. Now, make appropriate guesses for the values of amplitude, angular frequency and offset from the resulting graph. Use these along with the theoretical function for position of the Arduino mass given by the right hand side of **Equation 2.1** to produce the optimal parameters and associated covariance matrix needed for curve-fitting the position vs. time graph using the scipy library's curve_fit() function.

Recall that we found the spring constant for the spring using **Equations 4.1-4.2**. We can use this result along with the mass of the Arduino setup to calculate the angular frequency of the oscillations using **Equation 1.3**. The associated error in the angular frequency can be calculated using error propagation as described by **Equation 5**.

$$\delta\omega = \frac{1}{2}|\omega|\frac{\delta c}{|c|}$$

Equation 5: Uncertainty in the angular frequency, where ω is angular frequency, and $c = \frac{k}{m}$

We can compare this result with the best-fit angular frequency from the optimal parameters that we obtained. The associated error for the best-fit angular frequency can be found using the covariance matrix.

The total acceleration experienced by the Arduino setup can be calculated using the accelerometer data and the calibration fit-coefficients as described by the following equation,

$$a_T = ma_A + C$$

Equation 6.1 : Total acceleration a_T given by slope m, accelerometer output a_A and intercept C where slope and intercept were obtained through calibration of accelerometer

In this experiment, the effect of gravity causes the equilibrium position of the Arduino mass to be shifted. After applying a change of coordinates, we find that the Arduino mass behaves as if there is no effect of gravity on it as shown in **Equation 1.2**. But this will still be measured by the accelerometer. To adjust for this we must calculate a relative acceleration where we subtract the value of acceleration due to gravity as described by the following equation,

$$a = a_T + g$$

Equation 6.2 : Relative Acceleration formula to nullify the effect of gravity

We can use **Equation 6.2** to plot an acceleration vs. time graph for the accelerometer output. This graph can be curve-fitted using scipy's curve_fit() function. We can also use numerical differentiation to obtain the acceleration of the Arduino setup. This can be done by using numpy's gradient() function to obtain the velocity using position and time data from the Ultrasonic sensor followed by using the gradient() function again to obtain the acceleration using the new velocity and time data. Lastly, we can use theoretical differentiation to obtain the acceleration of the Arduino setup as shown in **Equation 2.3** with the optimal parameters we obtained earlier from curve fitting.

Results

We found that the mass of the arduino setup was 205 ± 0.5 g using the weighing scale. **Figure 7** shows the displacement vs. mass plot for increasing masses attached to the spring. We created the line of best fit for this data and used **Equations 4.1-4.2** to solve for k, which turned out to be 25 ± 5 N/m. We then predicted our angular frequency using **Equation 1.3** and **Equation 5** and obtained the value 11 ± 1 rad/s.

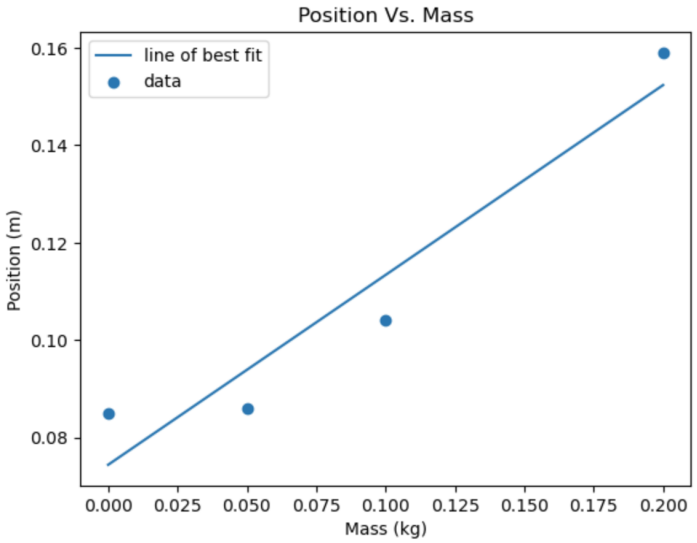

Figure 7: Displacement vs. Mass of spring with different attached masses

Figure 8, shows our ultrasound Position vs. Time plot along with its best fit curve. The blue line represents our raw data as the Arduino oscillated, and the green data is our best fit curve.

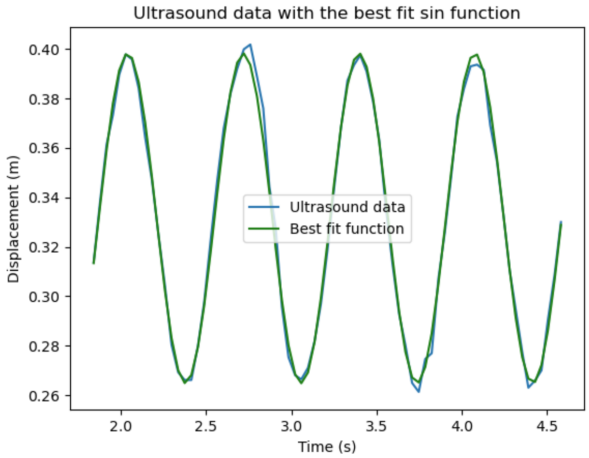

Figure 8: Position v. time data of ultrasonic sensor data with best fit sine function

Our best-fit coefficients for the ultrasound position data that we obtained using scipy's curve_fit() function are shown in **Table 1.**

Table 1 - Ultrasonic Position vs. Time best-fit coefficients

A(m)	ω(rad/s)	ϕ(rad)	C
0.0666 ± 0.0006	9.26 ± 0.01	-4.74 ± 0.04	0.3315 ± 0.0004

Putting the coefficients in **Equation 2.1,** our ultrasonic fit equation was:

$$y(t) = 0.0666sin(9.26t - 4.74) + 0.3315$$

Equation 7.1 : Ultrasonic Fit Equation

To find the best fit equation for the accelerometer data and time, we used the same process as the ultrasonic sensor data. **Figure 9,** shows our accelerometer's Relative Acceleration vs. Time plot along with its best fit curve. The blue line represents our raw data as the Arduino oscillated, and the green data is our best fit curve.

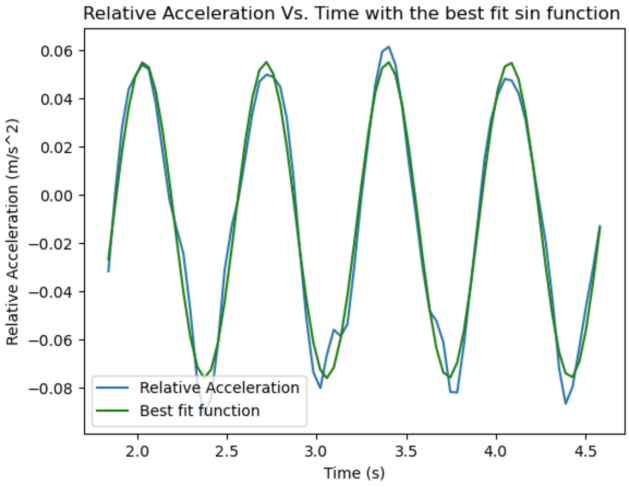

Figure 9: Accelerometer Output and its best-fit function

Our best fit coefficients for our accelerometer acceleration data are shown in **Table 2.**

Table 2 - Accelerometer Acceleration vs. Time best fit coefficients

A(m)	ω(rad/s)	ϕ(rad)	C
0.066 ± 0.001	9.24 ± 0.02	1.60 ± 0.08	-0.0104 ± 0.0009

Putting the coefficients in **Equation 2.1,** our accelerometer fit equation was:

$$y(t) = 0.066sin(9.24t + 1.60) - 0.0104$$

Equation 7.2 : Accelerometer Fit Equation

Figure 12 shows the plot for acceleration obtained by theoretical differentiation based on plugging the best-fit coefficients from **Table 1** in **Equation 2.3**. **Figure 13** shows the plot for acceleration obtained by numerical differentiation using numpy's gradient() function.

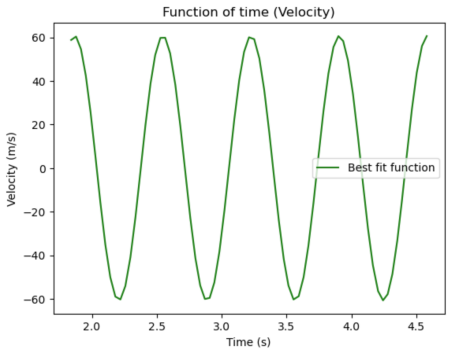

Figure 10: Theoretical differentiation of velocity of sin-fit function

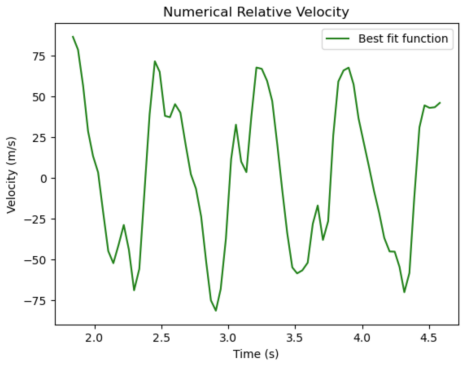

Figure 11: Numerical differentiation of velocity of sin-fit function

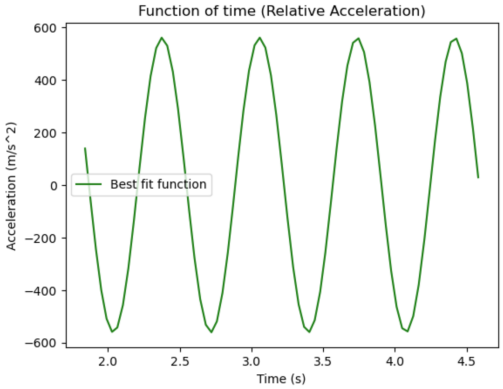

Figure 12: Theoretical differentiation of acceleration of sin-fit function

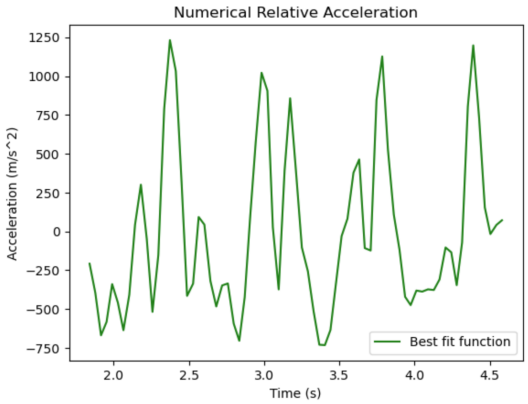

Figure 13: Numerical differentiation of acceleration of sin-fit function

By comparing **Figure 9, Figure 12** and **Figure 13**, we find that the acceleration function obtained by the accelerometer and theoretical differentiation share a nearly identical sinusoidal shape which is what was expected. Moreover from **Tables 1-2** we see that the amplitude and angular frequency of the accelerometer best-fit are within 2 standard deviations of the ultrasound sensor's amplitude and angular frequency best-fit, respectively. This means that the two acceleration functions that we have obtained are in agreement. This proves our hypothesis that the accelerometer will yield values for acceleration of the Arduino setup agree with the theoretical prediction given by **Equation 2.3.** The acceleration function that we obtained using numerical differentiation as seen in **Figure 13** also has a nearly sinusoidal shape, which makes sense. However, unlike the other two methods, it is not as accurate.

The best-fit angular frequency can be seen in **Table 1**. It is approximately 9.26 ± 0.01 rad/s. Earlier we discussed that our predicted angular frequency was 11 ± 1 rad/s. They are within two standard deviations of each other. Thus, the best-fit angular frequency is in agreement with our predicted angular frequency.

Conclusion

In our lab activity, we set out in finding the acceleration and the angular frequency of an oscillating glider in a spring-mass system using an Arduino with an ultrasonic sensor, bluetooth module, and accelerometer sensor. We hypothesized that the acceleration values and angular frequency that we got from the ultrasonic sensor and the Python analysis should correspond with **Equation 2.3** and **1.3.** Based on our Python analysis of the accelerometer output and its theoretical differentiations, we find that the graphs share an almost identical sinusoidal shape in **Figure 9, Figure 12,** and **Figure 13**. In addition, the coefficients between **Table 1-2** are nearly identical to each other, however, the two most important coefficients are the amplitude and the angular frequency, which are within 2 standard deviations of each other. So, this proves one of our hypotheses that our accelerometer will yield the values for acceleration that agree with the theoretical prediction as shown by **Equation 2.3**. Moreover the angular frequency, shown in **Table 1**, was approximately 9.26 ± 0.01 rad/s. This value is within the two standard deviations from what we predicted earlier, which was 11 ± 1 rad/s. Therefore, we've shown that our best-fit angular frequency is in agreement with our predicted angular frequency, proving our 2nd hypothesis.

One possible systematic error was the timing of the countdown between releasing the arduino from rest and starting the serial monitor. These delays might have skewed the results in both **Table 1** and **Table 2**, showing small discrepancies in the coefficients. Another systematic error in our experiment could be that the batteries lost power since the experiment has been going on for a long time. This would skew the results in the Python analysis as the data collected from the Arduino would have noisy levels that would not be matched with our original position sin-fit function best fit and its corresponding graph. Additionally, there were times when our Arduino would sway back and forth, which could have created noisy data. Fortunately, it did not appear this error greatly affected our data.

To improve our oscillating experiment so that we can yield better results, we can run multiple experiments so that we don't have to rely on one oscillating data of the Arduino. As shown in the results, there were small discrepancies in the coefficients, possibly due to the only run that we did. As such, multiple runs will average our inaccurate data onto the overall data. Another change we could improve our experiment would be to ensure that our batteries have enough voltage, preferably more than 3-4V, so that there would not be any noisy data for the duration of the experiment. Most importantly, making sure that all the steps in the **Methods** section are followed diligently can reduce the chances of errors.

Ultimately, this experiment demonstrated that we can use **Equations 2.1-2.3** to find the acceleration and its relative acceleration and the best angular frequency are more or less accurate predictions that agree with our experimental results. With enough runs and clean data, the calculated accelerations and its corresponding graphs should agree with each other and with the predictions.