

Physics 4AL

Final Project Lab Report - Elastic Pendulums

Lab 7

TA: Manoj Aravind Sankar

Table 5

Hannah Wang

Ethan Wong

Disha Zambani

I. Abstract

For our final project, we studied the effect on the motion of an elastic pendulum due to initial conditions imposed on the system. The system which comprises an elastic pendulum is made of a vertical hanging spring with a mass attached to the swinging end. When set in motion, the spring oscillates as a spring-mass system via compression and stretching and as a simple pendulum via its side-to-side swinging motion. The initial conditions varied were the initial angular displacement from the vertical axis and the spring constant of the spring. Using an accelerometer and the video modeling software, Tracker, we were able to trace the motion of the oscillating pendulum bob (Arduino). The accelerometer provided us with the acceleration of the Arduino in both the x and y directions, which was necessary for analysis due to the 2-dimensional nature of the system's path of motion. Tracker allowed us to observe the position of the pendulum, making it easier to analyze the radial nature of the elastic pendulum's motion. In addition, by combining these two data sets, we were able to increase the reliability of the data by comparing them and making sure they matched up. Using this analysis, we were able to determine the Lagrangian and equations of motion for the elastic pendulum system. We started by making a prediction of the path of motion based on what we knew about oscillation of a spring-mass system and what we knew about pendulum motion.

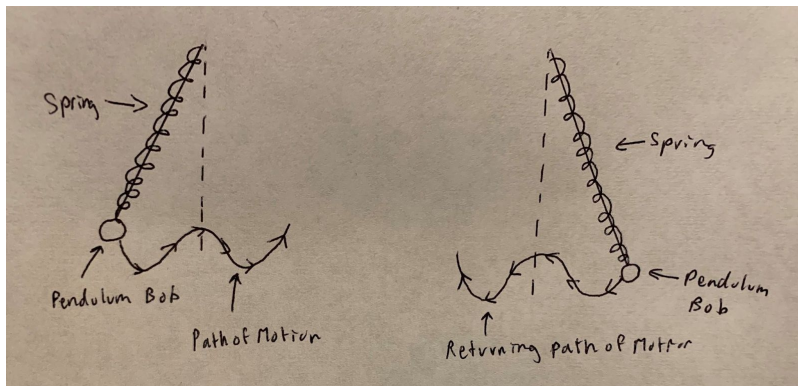


Figure 1: Prediction of Pendulum Bob's Path of Motion

With the Euler-Lagrange equations for this system and Newton's Second Law, we were numerically able to analyze the system and compare it to the ideal motions of the components of the elastic pendulum. We found that as the initial angle of release was increased the motion of the system had a larger range of motion as a pendulum but the range of motion as a spring mass system seemed to remain the same. We also found that as spring constant was decreased, the system modeled motion closer to that of an ideal elastic pendulum as described in the Introduction section of this report. As the spring constant became larger, however, the system acted like a simple pendulum with seemingly no spring-mass component. It can be said then that

as the spring constant is decreased, the spring-mass component becomes more prominent within the system and as the initial angle is increased, the simple pendulum component becomes more prominent within the system.

II. Introduction

The study of oscillation is a major topic in physics. For our final lab, we chose to combine the two types of oscillating systems we have studied in previous physics classes in our experiment by analyzing the motion of an elastic pendulum. The system is composed of a spring-mass element and a simple pendulum element. We varied various factors, such as spring constant, initial angle, and compression of the spring prior to oscillating the system to gain a better understanding for how oscillation and pendulum motion would work to produce the overall motion of the elastic pendulum system.

Equations for the motion of an elastic pendulum can be solved by finding the Lagrangian, L , using the equation:

$$L = T - V$$

where T is the kinetic energy of the system and V is the potential energy.

The potential energy of the system can be characterized by the sum of the individual potential energies of the spring-mass and pendulum components. They are as such:

$$\begin{aligned} V_{spring-mass} &= \frac{1}{2}kx^2 \\ V_{pendulum} &= -mg(l_0 + x)\cos\theta \\ V &= V_{spring-mass} + V_{pendulum} \end{aligned}$$

where k is the spring constant of the spring, x is the displacement of the spring stretched/compressed from its equilibrium position, m is the mass of the oscillating object attached to the spring, g is gravitational acceleration, l_0 is the length of the spring, and θ is the angle of oscillation of the pendulum.

The kinetic energy of the system can be characterized by a combination of movement perpendicular and along the spring. Using the definition of kinetic energy, we can find a relation that defines the total kinetic energy using the individual kinetic energies of the spring-mass and pendulum components as such:

$$T_{spring-mass} = \frac{1}{2}mv^2 = \frac{1}{2}m\left(\frac{dx}{dt}\right)^2$$

$$T_{\text{pendulum}} = \frac{1}{2}mv^2 = \frac{1}{2}m((l_0 + x)\frac{d\theta}{dt})^2$$

$$T_{\text{spring-mass}} = \frac{1}{2}mv^2 = T_{\text{spring-mass}} + T_{\text{pendulum}} = \frac{1}{2}m\left(\left(\frac{dx}{dt}\right)^2 + ((l_0 + x)\frac{d\theta}{dt})^2\right)$$

where v is the velocity of the oscillating mass. Note that for a simple pendulum, simple harmonic motion only occurs at small θ , which allows us to approximate $\sin \theta$ as θ when taking the derivative of velocity to find a relation between kinetic energy and angular displacement.

The Lagrangian, L , can now be defined as a function of displacement, angular displacement, velocity, and angular velocity:

$$L = \frac{1}{2}m\left(\left(\frac{dx}{dt}\right)^2 + ((l_0 + x)\frac{d\theta}{dt})^2\right) - \frac{1}{2}kx^2 + mg(l_0 + x)\cos\theta$$

To then find the equations of motion for analysis on the elastic pendulum system, we use the Euler-Lagrange equations:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial x'}\right) = \frac{\partial L}{\partial x}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \theta'}\right) = \frac{\partial L}{\partial \theta}$$

Solving for x and θ we obtain the following second order differential equations:

$$\frac{d^2x}{dt^2} = (l_0 + x)\frac{d\theta}{dt} + g\cos\theta - \frac{k}{m}x$$

$$\frac{d^2\theta}{dt^2} = -\left(\frac{g}{l_0+x}\right)\sin\theta - \left(\frac{2}{l_0+x}\right)\left(\frac{dx}{dt}\right)\frac{d\theta}{dt}$$

After obtaining these values through the trials run, we solve for the linear and angular accelerations numerically and draw conclusions based on the properties of the spring-mass and simple pendulum components of the elastic pendulum system mostly through qualitative analysis supported by reasoning through quantitative analysis with the path of motion of an ideal elastic pendulum as shown in Figure 2 as a qualitative benchmark.

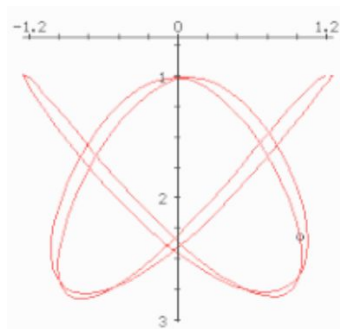


Figure 2: Ideal elastic pendulum path

III. Methods

Before proceeding with any experimentation, we first found the spring constants of the different springs we used to conduct our trials. As shown in Figure 4 and Figure 6 below, we hung a mass and noted the displacement of the spring from equilibrium. We replicated this process for a number of varying masses. Since the spring-mass system was at rest we were able to use Newton's Second Law as shown in the free body diagram in Figure 3 to calculate the spring constant as such:

$$\sum F = F_g + F_s = mg - kx = 0$$

And rearrange the relation to obtain:

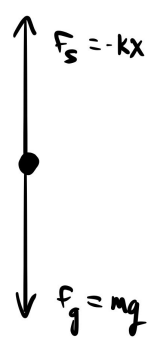
$$k = \frac{mg}{x}$$


Figure 3: Free body diagram of vertical mass-spring system at rest



Figure 4: A simple spring-mass system.

Table 2 in the Results portion of this report contains the experimental data collected in order to perform the calculations described above and Figure 10 shows the corresponding graph.

To collect data for the elastic pendulum system, we used two Arduinos. One Arduino acted as a receiver while the other acted as a transmitter. The transmitting Arduino powered an accelerometer and sent the collected data via radio to the receiver. The accelerometer would give us the acceleration of the system in the direction of oscillation. The transmitting Arduino also had a red sticker on it that would allow us to track its motion as a point mass using the Tracker app on our laptops. The transmitter, which would act as the pendulum bob in the system, was attached to the bottom of a hanging spring as shown in Figure 7 and released at 0° , 15° , 30° , and 60° angles from the equilibrium position. The transmitter was then released from rest at these varying angles. We then recorded the pendulum bob's path of motion in a csv file with the accelerometer data as well as in a video to create a visualized path of motion using Tracker.

When releasing the pendulum bob from rest in the different trials, we kept the spring as close to its natural length as possible. This would allow for a uniform displacement of 0 centimeters across all the trials for all the different springs. If we had tried stretching the spring, there would be an additional source of potential energy added into the system that we wanted to avoid. Also, we feared that stretching the spring would be somewhat of a safety hazard. The motion of the pendulum would be too violent and there was a high risk of the pendulum bob flying off the spring and getting damaged or hitting someone.

We also found it most logical to only alter one independent variable at a time. So, when we looked for the effect of varying spring constants on the motion of the system, we kept the angular displacement of the pendulum as consistent as possible at close to 60 degrees. Similarly, when we looked for the effect of varying angular displacement on the motion of the system, we conducted our trials with the same spring, keeping the spring constant constant. This approach made the process of analyzing data and obtaining corresponding results more sound and reliable. We took several iterations of data for the same trial and chose what we thought to be the best data sets to represent the results obtained through this experiment.

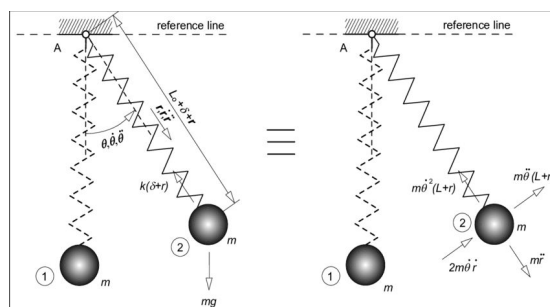


Figure 5: Free body diagram of the pendulum bob

To analyze the system we solved these equations numerically with relations and values obtained through experimentation. We were able to compare and contrast properties of the two oscillating systems as they worked in combination in the elastic pendulum.

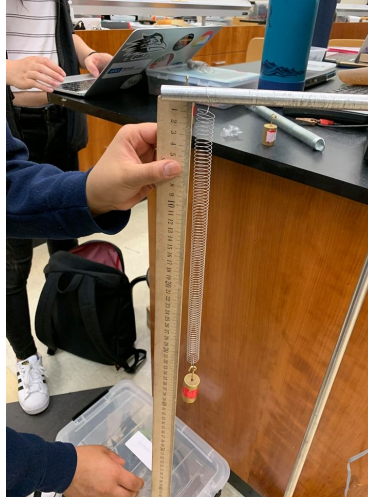


Figure 6: Measuring displacement of spring due to hanging mass to find spring constant

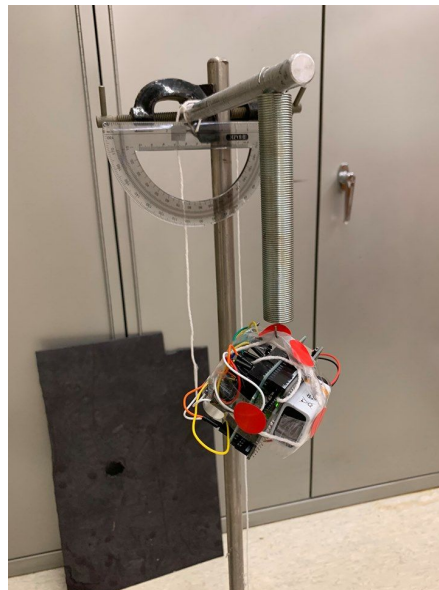


Figure 7: Elastic pendulum system at equilibrium position

IV. Analysis

To determine the spring constant for each spring, we plotted the displacement of the spring as a function of the mass hung from the spring. When a mass is hanging vertically from the spring, there are two forces acting in equal and opposite directions on the mass - the force of

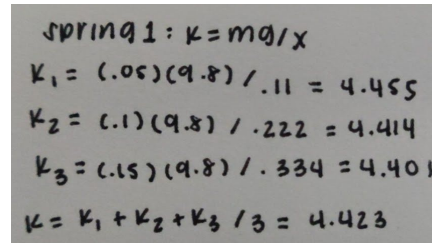
gravity acting downward and the restoring spring force acting upwards. Therefore, the net force equation is as follows:

$$\sum F = F_g + F_s = mg - kx = 0$$

where m is the mass attached to the spring, g is acceleration due to gravity (9.8 m/s^2), k is the spring constant, and x is the displacement of the spring after the mass is added. The spring constant k can be isolated and is the following:

$$k = \frac{mg}{x}.$$

For example, for Spring 1 in Table 2, we can calculate the spring constant using the relation above for each mass. Then we could take the average of the obtained set of spring constants for each spring to get a good approximation of the spring constant for Spring 1 as shown below in Figure 8.



Spring 1: $k = mg/x$
 $k_1 = (0.05)(9.8) / .11 = 4.455$
 $k_2 = (0.1)(9.8) / .222 = 4.414$
 $k_3 = (0.15)(9.8) / .334 = 4.401$
 $k = (k_1 + k_2 + k_3) / 3 = 4.423$

Figure 8: Calculation of spring constant for Spring 1

To analyze the effect of different initial angles on the movement of the spring, we measured the accelerometer data in the x (horizontal) and y (vertical) directions. We plotted this data as a function of time and found the best fit sinusoidal curves to fit these functions. Using the relationship

$$r = \sqrt{x^2 + y^2}$$

to determine the acceleration in the radial direction for each scenario.

To analyze video data of the trials being conducted, we assumed the oscillating Arduino acted as a point mass and used the software Tracker to mark the position of the point mass frame by frame. The software then took this data and returned plots of the radial and angular displacement as a function of time. The program is only able to return the magnitude of radial velocity, so in order to account for both magnitude and direction, we used the following definition of velocity:

$$v = \frac{dx}{dt}$$

to approximate the instantaneous velocity of the point mass. For each time t that a position datapoint was recorded, we found the change in position and divided this by the change in time

$$v_n = \frac{x_{n+1} - x_n}{t_{n+1} - t_n}$$

For example, for spring 1 released from 15 degrees, the following position data was collected:

Table 1: Position of Spring 1 at Different Times when released from 15°

Time (s)	Position (m)
0.0	0.188
0.0333	0.146
0.0667	0.112
0.100	0.0825

The instantaneous velocity at time 0 was determined with the following equation:

$$v_0 = \frac{x_1 - x_0}{t_1 - t_0} = \frac{0.146 - 0.189}{0.0333 - 0} = -1.28 \text{ m/s}$$

This was done for every time at which the position was measured. Calculations were done using formulas on a spreadsheet. A similar calculation was done to find the instantaneous acceleration, using the following definition of acceleration:

$$a = \frac{dv}{dt}$$

For each time t that a velocity was calculated using the calculations described previously, we found the change in velocity and divided this by the change in time according to the following calculation:

$$a_n = \frac{v_{n+1} - v_n}{t_{n+1} - t_n}$$

The radial displacement, velocity, and acceleration found using Tracker data were each plotted as functions of time and modeled using a best fit sine curve. Additionally, the angular displacement was plotted as a function of time. However, they could not be modeled with any familiar equation such as a sine curve, so we were not able to come up with an equation of best fit.

The data collected and equations of the curves were then used in the Euler-Lagrange equations described in the Introduction of this report to analyze the system. Since we were able to obtain the equation for radial acceleration from the data obtained, we only used the equation for angular acceleration in our calculations since we were unable to obtain any equations for angular acceleration through data collection. A sample calculation for Spring 1 released at an initial angle of 15 degrees is shown in Figure 9 below. We noticed that since we let the spring oscillate initially at its natural length with no imposed initial stretching or compressing, the value for x in the equation is 0. We also noticed that the value for angular velocity was the angular frequency obtained through experimentation. The value for velocity was obtained as a function of time through experimentation as well.

Handwritten calculation for angular acceleration of Spring 1 at 15 degrees:

$$\frac{d^2\theta}{dt^2} = -\frac{g}{(l_0 + x)} \sin\theta - \frac{2}{(l_0 + x)} \frac{dx}{dt} \frac{d\theta}{dt}$$

Given: $x = 0$, $\omega = d\theta/dt = 5.00 \text{ rad/s}$

$$\frac{d^2\theta}{dt^2} = -\frac{(-9.8)}{(0.95)} \sin 15^\circ - \frac{2}{(0.95)} (1.35 \sin(4.99t - 0.87)) (5.00)$$

$$= 20.7 - 142 \sin(4.99t - 0.87)$$

Figure 9: Calculation of angular acceleration for Spring 1 at 15 degrees

V. Results

Table 2: Displacement of Various Springs Due to a Mass

Mass (kg)	Spring 1 (m)	Spring 2 (m)	Spring 3 (m)
0.0	0.0	0.0	0.0
0.05	0.110	0.118	0.012
0.1	0.222	0.263	0.023
0.15	0.334	0.409	0.035
0.2	-	0.545	0.051

Table 2 contains the data collected when varying masses were placed on each of the three springs we used in the trials of our experiment in order to calculate the spring constants of each spring. Using the process to find the spring constant of a spring using this data described in the Analysis section of this report, Spring 1 was found to have a spring constant of 4.42, Spring 2 with a spring constant of 3.77, and Spring 3 with a spring constant of 40.9.

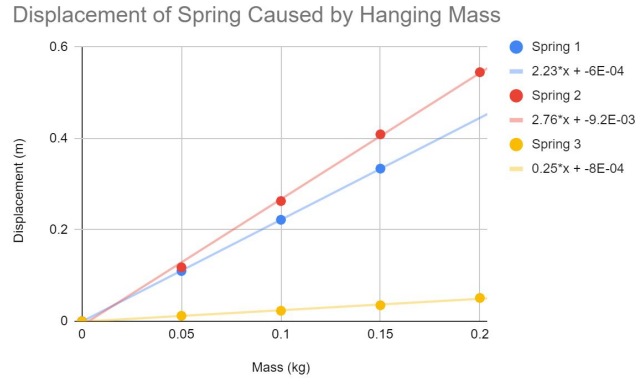


Figure 10: Various masses were hung from springs 1, 2, and 3 with unknown spring constants and the displacement of the springs were plotted as a function of the added masses. The best fit line for Spring 1 is modeled by the equation $\text{displacement} = 2.23(\text{mass}) + -6\text{E-}04$. The best fit line for Spring 2 is modeled by the equation $\text{displacement} = 2.76(\text{mass}) + -9.2\text{E-}03$. The best fit line for Spring 3 is modeled by the equation $\text{displacement} = 0.25(\text{mass}) + -8\text{E-}04$. The plot shows the displacement increasing linearly with an increase in the mass which produces a positive correlation. The best fit line appears to fit the data with no major outliers.

From Figure 10, we obtain an increasing linear relationship with a positive correlation between mass and displacement for all Springs 1, 2, and 3 where displacement is a function of mass from the data in Table. The best fit line fits the data very well.

Spring 1: 0 Degree Angle of Release

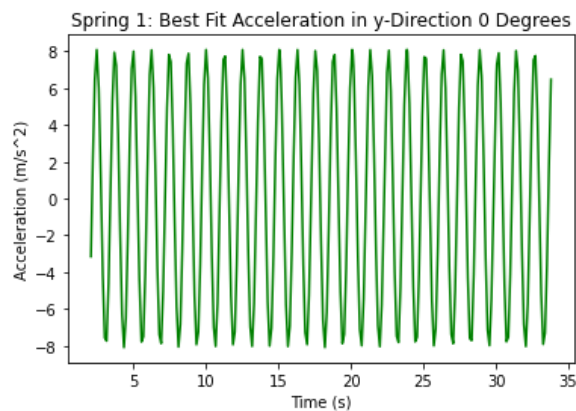


Figure 11: The graph shows the best fit acceleration obtained from the accelerometer for Spring 1 released at an initial angle of 0° .

The best fit curve for acceleration is shown in green modeled by the equation $acceleration = 8.10 \sin(4.99t + 1.95)$.

When the elastic pendulum system is released from zero degrees, it only oscillates vertically. As a result, it undergoes only simple harmonic motion due to the spring and does not experience any pendulum motion. The motion of a vertical oscillating spring has been studied previously in this class. For the purpose of this experiment, the data collected in Figure 11 served as a control for the trials we analyze below.

Spring 1: 15 Degree Angle of Release

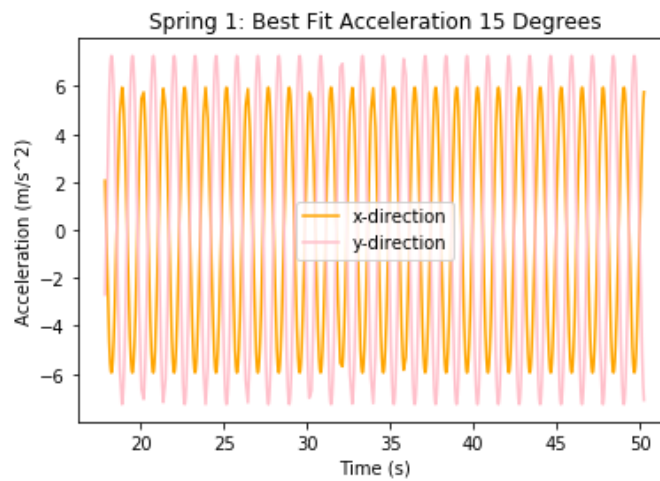


Figure 12: The graph shows the best fit acceleration in the x and y-directions obtained from the accelerometer for Spring 1 released at an initial angle of 15°. The best fit curve for acceleration in the x-direction is shown in orange modeled by the equation $acceleration = 5.95 \sin(5.00t + 1.42)$. The best fit curve for acceleration in the y-direction is shown in pink modeled by the equation $acceleration = -7.27 \sin(5.00t + 1.36)$.

The accelerometer data plotted in the graph in Figure 12 helped us gain a qualitative understanding of the motion of the pendulum. We observed that the acceleration in both the x- and y- direction created sine curves that resembled that of simple harmonic motion. The acceleration in both directions would max out when the pendulum bob reached the positive and negative amplitudes of its motion. The acceleration would also be 0 at the middle of its path of motion. This helped us come to the conclusion that the system still exhibited the properties of simple harmonic motion due to the motion of the spring and the pendulum motion combined.

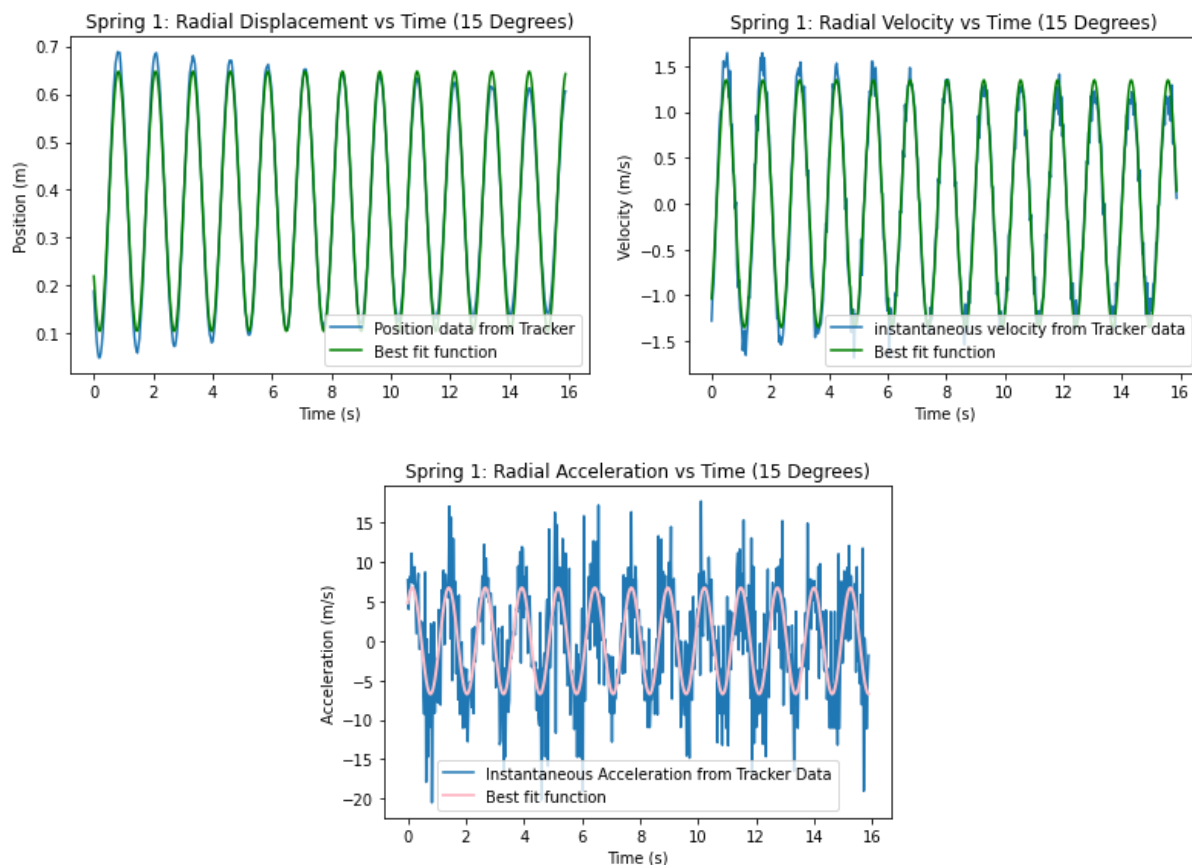


Figure 13: The position, velocity, and acceleration of the Arduino bob in the radial direction was plotted as a function of time from data collected using Tracker. The graphs above show data that was collected for spring 1 after being released from an initial angle of 15 degrees. The positive direction is pointed away from the pivot point of the elastic pendulum system.

The radial position, velocity, and acceleration can all be modeled by a sinusoidal curve of best fit as shown in Figure 13. The raw data for velocity and acceleration were derived from the displacement data discussed earlier. The equations for these models are as follows:

$$r = 0.271\sin(5.00t - 2.526)$$

$$v = 1.35\sin(4.99t - 0.87)$$

$$a = 6.75\sin(5.00t + 0.79)$$

The amplitude of oscillation was found to be approximately 27.1 centimeters and the angular frequencies of the system is approximately 5.00 radians/second. The elastic pendulum system's motion in this direction can be attributed to the spring component of the system, which exerts a restoring force on the oscillating Arduino.

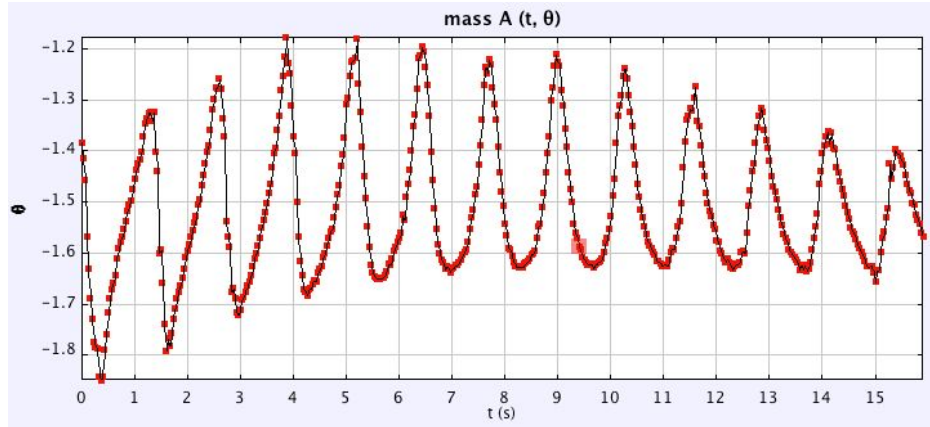


Figure 14: Plot of rotational angle as a function of time for Spring 1 released from 15 degrees. Units for the y axis are in radians.

Using Tracker, we were able to find a function for radial acceleration as a function of time as described above. However, we were not able to find a function for angular displacement and further derive the function to obtain a function for angular acceleration as the angular displacement graph in Figure 14 was not differentiable. Using the Euler-Lagrange equations described in the Introduction section of this report and other pieces of collected data, we were able to find an equation for angular acceleration as a function of time. An example calculation of this process can be found in the Analysis section of this report. For Spring 1 released at an initial angle of 15 degrees, the equation modeling angular acceleration is as follows:

$$\frac{d^2\theta}{dt^2} = 26.7 - 142\sin(4.99t - 0.87)$$



Figure 15: Path of motion of the pendulum bob when released at a 15° angle as mapped using Tracker. The red line traces the path of the pendulum over the period of time for which data was recorded.

For an initial angle of 15° , Spring 1 shows little horizontal motion for the pendulum bob most likely because the initial release angle was so small. As seen in Figure 15, this led to an ellipsoidal path of motion where the Arduino started out further away from the equilibrium position (in front of the metal rod) and came closer to the equilibrium position over time as damping interfered with the system. If the trial were to be run for longer possibly with some driving force we would see something similar to the ideal elastic pendulum motion but much narrower as motion is being limited by the small 15 degree angle. However, due to the nature of resources available and the fact that Tracker cannot analyze long videos, we were not able to conduct such lengthy trials. There are some intersections of the lines that represent the path of motion, but there is mostly overlap due to the bob following an ellipsoid path of motion that just shrunk over time.

Spring 1: 30 Degree Angle of Release

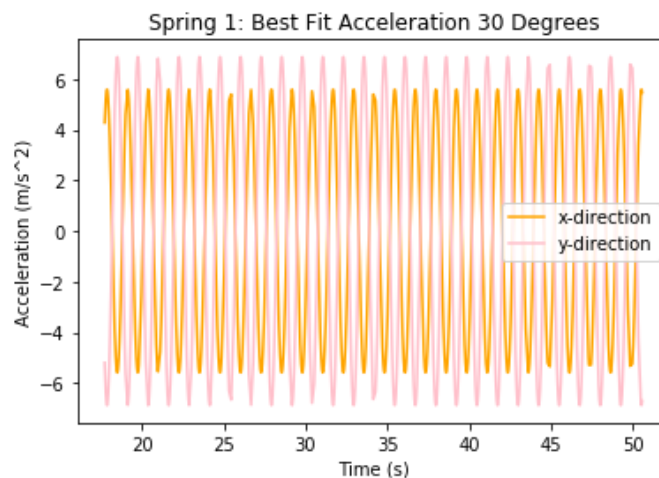


Figure 16: The graph shows the best fit acceleration in the x and y-directions obtained from the accelerometer for Spring 1 released at an initial angle of 30° . The best fit curve for acceleration in the x-direction is shown in orange modeled by the equation $\text{acceleration} = 5.58 \cdot \sin(5.00 \cdot t + 6.45)$. The best fit curve for acceleration in the y-direction is shown in pink modeled by the equation $\text{acceleration} = 6.87 \cdot \sin(5.00 \cdot t + 3.28)$.

In Figure 16, we again see that the acceleration in both the x- and y- direction created sine curves that resembled that of simple harmonic motion. The acceleration in both directions would max out when the pendulum bob reached the positive and negative amplitudes of its motion. The acceleration would also be 0 at the middle of its path of motion. This system exhibits properties of simple harmonic motion as the system did at an initial angle of 15 degrees.

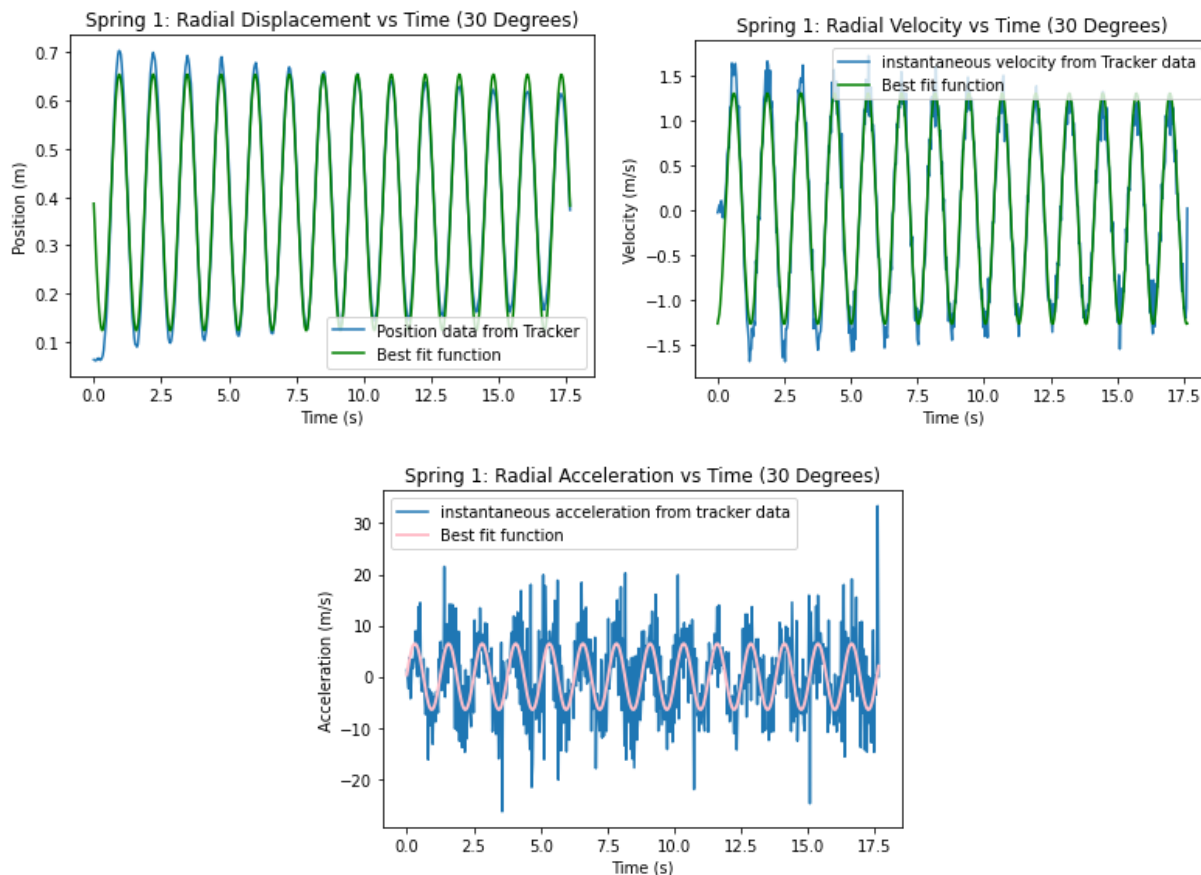


Figure 17: The plots above show the position, velocity, and acceleration of the Arduino bob in the radial direction as a function of time for spring 1 after being released from an initial angle of 30 degrees. The positive direction is pointed away from the pivot point of the elastic pendulum system.

The radial position, velocity, and acceleration can all be modeled by a sinusoidal curve of best fit as shown in Figure 17. The raw data for velocity and acceleration were derived from the displacement data discussed earlier. The equations for these models are as follows:

$$r = 0.27\sin(4.99t - 3.13)$$

$$v = 1.29\sin(4.99t - 1.48)$$

$$a = 6.46\sin(5.00t + 0.05)$$

The amplitude of oscillation was found to be approximately 27 centimeters and the angular frequency of the system is approximately 4.99 radians/second, which is very similar to the amplitude and angular frequency when it was released from 15 degrees.

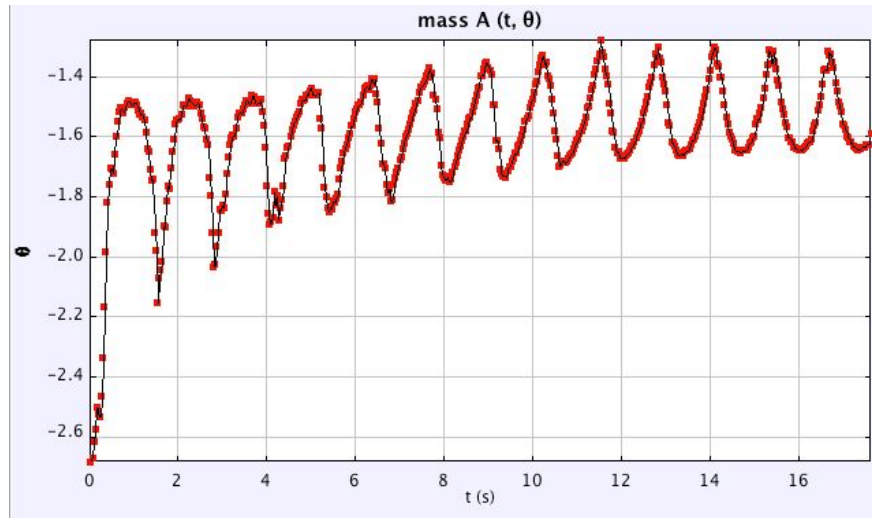


Figure 18: Plot of rotational angle as a function of time for spring 1 released from 30 degrees. Units for the y axis are in radians measured from the positive horizontal axis.

Using the method described in the Analysis section of this report, the equation modeling angular acceleration for Spring 1 released at an initial angle of 30 degrees is:

$$\frac{d^2\theta}{dt^2} = 51.6 - 136\sin(4.99t - 1.48)$$

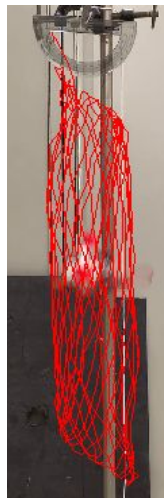


Figure 19: Tracker plot of the motion of the pendulum bob when released at a 30° angle

For an initial angle of 30°, Spring 1 shows noticeably more horizontal motion for the pendulum bob probably because the initial release angle is double that of the previous trial. As seen in Figure 19, this led to an ellipsoidal path of motion similar to that seen in the previous

trial. There are some intersections of the lines that represent the path of motion, but there is mostly overlap due to the bob following the ellipsoid path of motion that just shrunk over time. In this trial however, since the release angle was greater in this trial, the bob follows a more unique path of motion — the red lines intersect a lot more and there is less overlap than in the 15° trial as the motion occurs over a larger horizontal range. This can be attributed to the fact that in a simple pendulum the total energy introduced into the system is the initial potential energy imposed by the initial position of the pendulum bob before oscillation. The initial potential energy can be described as mgh and we see that h increases as the angular displacement of the pendulum bob increases. Due to the Law of Conservation of Energy, for an angle of 30 degrees, the amount of energy in the system is definitely larger than that of the previous trial. This energy is converted to kinetic energy and back to potential energy over the course of the pendulum bob's motion, providing for the wider range in pendulum motion. As compared to the trial with initial angular displacement of 15 degrees, we see that the radial displacement, velocity, and acceleration are very similar. This corresponds with the fact that the two systems have similar spring-mass components. However, the angular acceleration of the system at 30 degrees compared to the system 15 degrees is smaller in magnitude. This corresponds with the fact that there is a much larger range of motion as a pendulum due to the initial angular displacement. The angular acceleration becomes smaller due to the fact there there is more motion the pendulum covers in the same amount of time.

Spring 1: 60 Degree Angle of Release

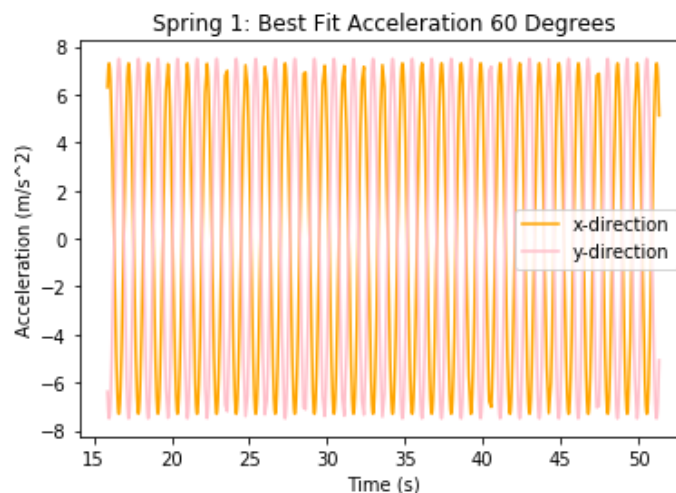


Figure 20: The graph shows the best fit acceleration in the x and y-directions obtained from the accelerometer for Spring 1 released at an initial angle of 60° . The best fit curve for acceleration in the x-direction is shown in orange modeled by the equation $\text{acceleration} = 7.30 \cdot \sin(5.00 \cdot t + 3.39)$. The best fit curve for

acceleration in the y -direction is shown in pink modeled by the equation $acceleration = -7.49 \sin(5.00t + 3.33)$.

Once again the accelerometer data in Figure 20 shows sine curves representative of simple harmonic motion. This system exhibits the same type of motion as the system in Figure 16 where the acceleration is at its maximum when the pendulum bob reaches its amplitude and it becomes zero at the middle of its path of motion.

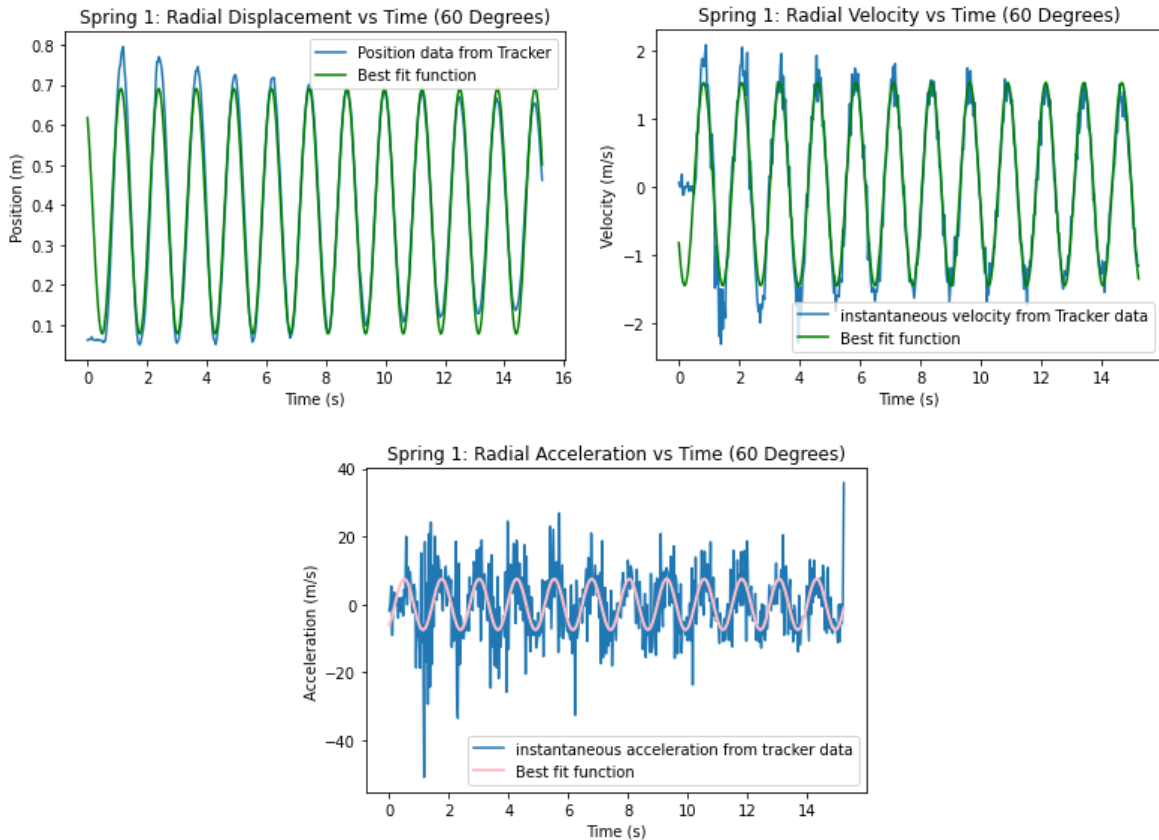


Figure 21: The position, velocity, and acceleration of the Arduino bob in the radial direction was plotted as a function of time from data collected using Tracker. The graphs above show data that was collected for spring 1 after being released from an initial angle of 60 degrees. The positive direction is pointed away from the pivot point of the elastic pendulum system.

The radial position, velocity, and acceleration can all be modeled by a sinusoidal curve of best fit as shown in Figure 21. The raw data for velocity and acceleration were derived from the displacement data discussed earlier. The equations for these models are as follows:

$$r = 0.31 \sin(4.97t - 0.86)$$

$$v = 1.49 \sin(4.99t - 2.53)$$

$$a = 7.48\sin(5.00t - 0.95)$$

The amplitude of oscillation was found to be approximately 31 centimeters and the angular frequency of the system is approximately 4.98 radians/second, which is again very similar to the amplitude and angular frequency when it was released from smaller angles.

This data, along with the data collected for spring 1 when released from 15 degrees and 30 degrees, reveals that there is no significant effect of the initial angle at which an elastic pendulum is released from on the radial motion of the system. This is to be expected, as the movement in the radial direction is caused by the force of the spring, which acts only in the radial direction. When the system is released from a different angle, the pendulum component of motion is affected, not the spring component.

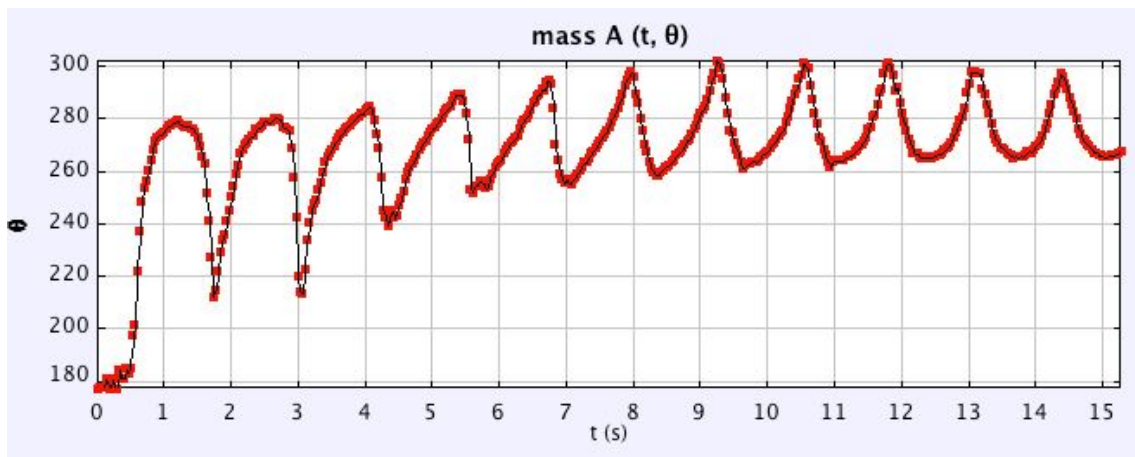


Figure 22: Plot of rotational angle as a function of time for spring 1 released from 60 degrees. Units for the y axis are in radians measured from the positive horizontal axis.

Using the method described in the Analysis section of this report, the equation modeling angular acceleration for Spring 1 released at an initial angle of 60 degrees is:

$$\frac{d^2\theta}{dt^2} = 89.3 - 157\sin(4.99t - 2.53)$$

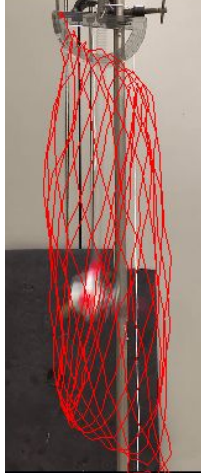


Figure 23: Tracker plot of the motion of the pendulum bob when released at a 60° angle

For an initial angle of 60° , Spring 1 shows the most horizontal range of motion for the pendulum bob of the three trials conducted with varying angles probably because the initial release angle is double that of the previous trial. As seen in Figure 23, this led to an ellipsoidal path of motion similar to those seen in the previous trials. This trial exhibits the least amount of overlap as the horizontal range of motion is the largest in this scenario. However, the amount of intersections is roughly the same as the 30° trial, just spread out over a larger range. Again, we attribute this to the fact that a larger angular displacement increases the total amount of energy in the system as with the previous trial. Compared to both the 15 degree and 30 degree systems, we see that the radial displacement, velocity, and acceleration are slightly larger but still quite similar. This corresponds with the fact that the two systems have similar spring-mass components. The reason that for this trial the values were slightly higher is probably because we unintentionally gave the spring a small initial displacement or the fact that the arduino is not very secure and possibly imposed its own torque as a smaller pendulum component into the system. Again, we see that the angular acceleration of the system at 60 degrees compared to the system 30 degrees is smaller in magnitude. This corresponds with the fact that there is a much larger range of motion as a pendulum due to the initial angular displacement. The angular acceleration becomes smaller due to the fact there there is more motion the pendulum covers in the same amount of time.

Spring 2: 60 Degree Angle of Release

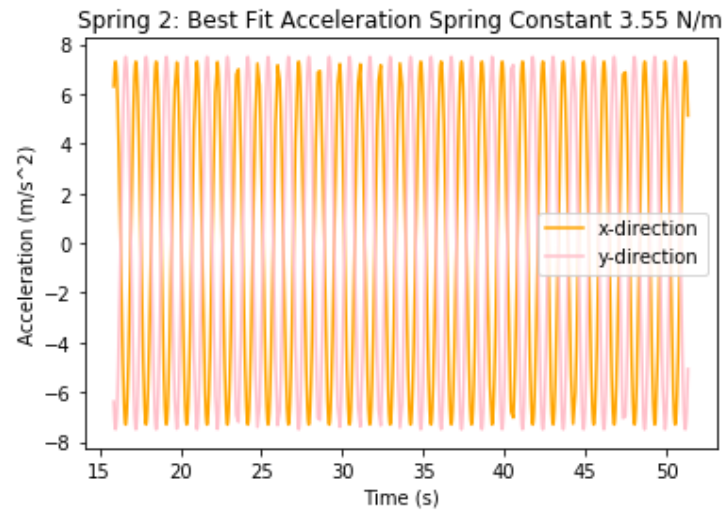


Figure 24: The graph shows the best fit acceleration in the x and y-directions obtained from the accelerometer for Spring 1 released at an initial angle of 60°. The best fit curve for acceleration in the x-direction is shown in orange modeled by the equation $\text{acceleration} = 0.29\sin(4.99t + 3.39)$. The best fit curve for acceleration in the y-direction is shown in pink modeled by the equation $\text{acceleration} = 0.30\sin(5.00t + 4.91)$.

The accelerometer data in Figure 24 further shows that elastic pendulum systems still show the properties of simple harmonic motion while they move. In this trial Spring 2 was released at an initial angle of 60 degrees but the shape of the graph is almost identical to that of Spring 1 at 60 degrees in Figure 20. This shows that it is normal for elastic pendulums to have some simple harmonic motion as they oscillate and go through their path of motion.

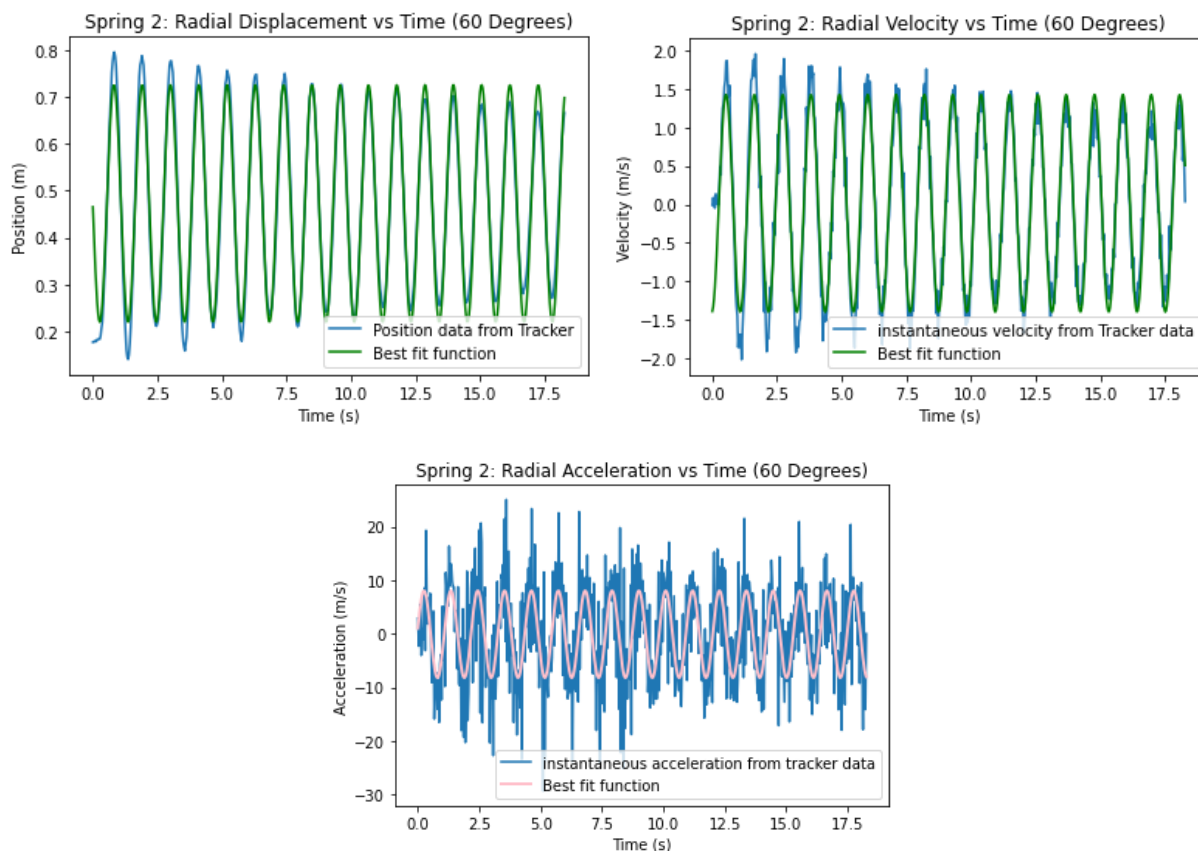


Figure 25: The position, velocity, and acceleration of the Arduino bob in the radial direction was plotted as a function of time from data collected using Tracker. The graphs above show data that was collected for spring 2 after being released from an initial angle of 60 degrees. The positive direction is pointed away from the pivot point of the elastic pendulum system.

The radial position, velocity, and acceleration can all be modeled by a sinusoidal curve of best fit as shown in Figure 25. The raw data for velocity and acceleration were derived from the displacement data discussed earlier. The equations for these models are as follows:

$$r = 0.25\sin(5.73t - 3.11)$$

$$v = 1.41\sin(5.73t - 1.46)$$

$$a = 8.11\sin(5.74t + 0.13)$$

The amplitude of oscillation was found to be approximately 25 centimeters and the angular frequency of the system is approximately 5.73 radians/second. The angular frequency of the system with this initial setup has a larger frequency of oscillation as compared to when we used spring 1, which is to be expected. Spring 2 has a smaller spring constant, so the angular frequency, which is inversely related to the spring constant, is larger in magnitude.

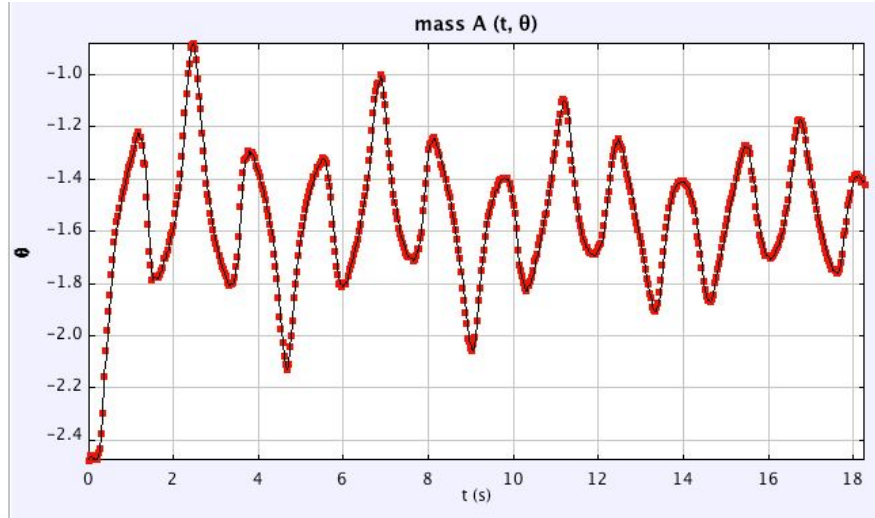


Figure 26: Plot of rotational angle as a function of time for spring 2 released from 60 degrees. Units for the y axis are in radians measured from the positive horizontal axis.

Using the method described in the Analysis section of this report, the equation modeling angular acceleration for Spring 2 released at an initial angle of 60 degrees is:

$$\frac{d^2\theta}{dt^2} = 73.8 - 141\sin(5.73t - 1.46)$$

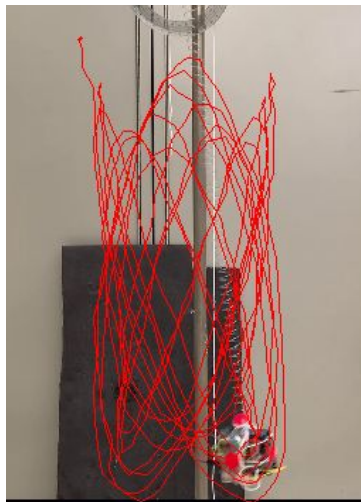


Figure 27: Tracker plot of the motion of the pendulum bob when released at a 60° angle

The trial for Spring 2 at an initial angle of 60 degrees exhibits a path of motion most similar to the idealized path of motion referenced in the Introduction section of this report. As

seen in Figure 27, it has a pretzel-like shape for its path of motion. More interestingly, out of the three springs used in the experiment, Spring 2 was found to have the smallest spring constant.

Spring 3: 60 Degree Angle of Release

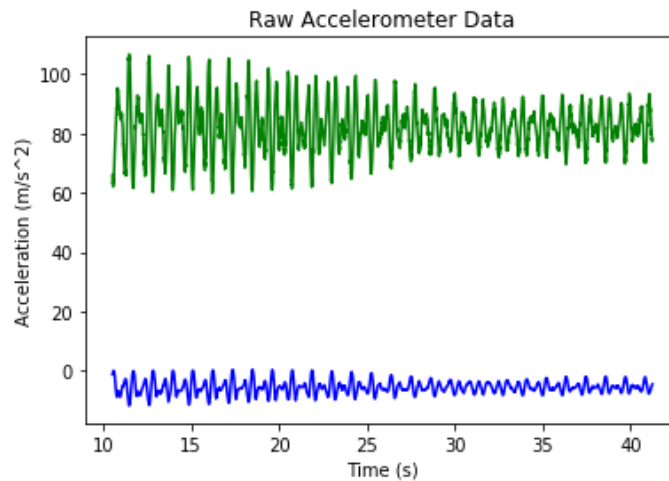


Figure 28: This plot shows the raw accelerometer data in the x- and y- direction. The green line shows the motion of the system in the vertical y-direction, while the blue line reflects the motion of the system in the horizontal x-direction.

The raw accelerometer data shown in Figure 28 for Spring 3 could not be properly modeled with a sine curve. Its motion did seem to be very similar to that of the other springs' trials a few seconds after the pendulum bob was released, but exhibited strange behavior after a short period of time. There was a slight decrease in amplitude observed at the 30 second mark but the amplitude increased again at the 35 second mark.

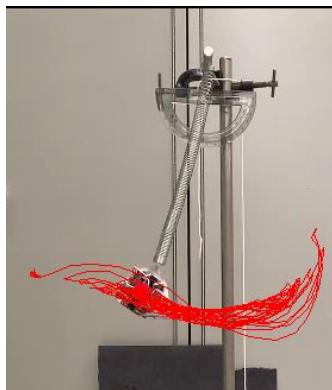


Figure 29: Tracker plot of the motion of the pendulum bob when released at a 60° angle

The trial for Spring 3 closely exhibits the motion of a simple pendulum. The path of motion shown in Figure 29 is difficult to describe but can be seen as a wave type of motion. The spring constant was very high and, when the pendulum bob was released from rest, the spring stretched minimally with very little motion as a spring mass system — the pendulum bob mostly just swayed back and forth as we would expect in a typical simple pendulum system. More interestingly, Spring 3 was found to have the highest spring constant and seems to be the farthest from the path motion for an ideal elastic pendulum of the three springs observed. We can say that since the spring constant for Spring 3 is so high, it is more stiff and acts more like a rod which inhibits motion in the spring-mass component of the system.

VI. Conclusion

The main goal of this lab was to observe how the properties of simple harmonic motion due to a spring-mass system would interact with pendulum motion when they combined to create an elastic pendulum. We have never encountered such a system before, so we wanted to predict and analyze how varying different independent variables would alter the path of motion, velocity, and acceleration for the elastic pendulum system. We changed the angle of release and the spring constants and observed how an elastic pendulum would react to these types of changes. We were able to observe that varying the angle of release and the spring constant actually had significant effects on how the entire system would react. Releasing the pendulum bob from a larger initial angle would increase the amount of time it took for the bob to come back to equilibrium position and there is a wider, more drastic range of horizontal motion tied to the simple pendulum component of the system which can be attributed to the increase in energy due to a larger initial angular displacement. A more detailed explanation can be found in the Results section of this report. We also found that using a spring with a lower spring constant gives a path of motion closer to the idealized path of motion for an elastic pendulum as shown in Figure 2. This is because as the spring constant increases, the stiffness of the spring increases and acts more like a rod — it will not have as much of the simple harmonic motion that results from the spring-mass component of the system as we saw with Spring 3.

In addition, we learned over the course of many trials that as the spring compresses its shorter radius causes the spring to move faster. This is a result of the conservation of angular momentum. Since there was no external torque that was acting on the object, there was no change in angular momentum. According to the angular momentum equation mentioned at the end of the Introduction section, a decrease in the radius would cause an increase in velocity because the mass and angular momentum would always stay constant within the elastic pendulum system.

If we were to redo this experiment and further our investigations, there are some improvements that we would make to this lab in order to obtain better results. We would first

find and experiment with a more varied selection of springs to verify the trends we observed in this experiment. We would also try to experiment with different angular displacements. An issue we had was securing the transmitter Arduino as the pendulum bob perfectly so that it would not spin at all during its motion. This did not really affect the data-acquisition part of the lab, but it did make the process of plotting the path of motion using Tracker slightly more difficult since the red dot would sometimes fade out of view of the camera. We resolved this issue by replacing the red sticker with an orange table tennis ball. We taped the ball to the bottom of the pendulum bob and then recorded video of the path of motion. Since the ball could be visible even if the pendulum bob turned slightly, the process of creating a plot with Tracker was significantly simplified. Another path of analysis that we wanted to pursue other than just simply changing the angle and spring constants for our elastic pendulum system was adding a driving force to the experiment. We wanted to have a motor that would provide a driving force in the direction of the pendulum bob's motion to see if we could find a resonant frequency and observe any effects it would have on the system. However, the lab only had motors that would provide a driving force in the vertical direction. This would not have worked very well in our experiment because we needed a driving force that would match the angle of the pendulum as it oscillated and swung side to side. If a motor that could provide that type of driving force had been provided, we could have delved much deeper into the study of elastic pendulums. We could also expand our analysis to varying initial displacements of the spring and observe its effect on the motion of the system.

VII. Appendix

Sample code for analysis of accelerometer data:

```
# Run this cell to mount your Google Drive.
from google.colab import drive
drive.mount('drive')

# numpy has a function that helps to read data files so we import it
import numpy as np
import matplotlib.pyplot as plt

read_in_array = np.loadtxt('/content/drive/My Drive/Physics 4AL/Final
Project/littlespring30degrees.csv',skiprows=1,delimiter=',')

# This dataset has five columns
# Take all of the elements in the 0th column to create your x-axis array
x_axis = read_in_array[:,0]
# Elapsed time is typically always going to be the x-axis
# We are converting it from units of ms to s.
elapsed_time = (x_axis-x_axis[0])/1000.

# We create an array to store the acceleration in x - the axis along which
simple harmonic motion takes place
acc_data_unrefined_y = read_in_array[:,1]
acc_data_unrefined_x = read_in_array[:,3]

# The units are cm/s^2, we are converting it to m/s^2.
# The data is collected from a vertical spring mass system.
# We add 9.8 to the array to centre the acceleration around 0 since the
smaller mass in the accelerometer always experiences -g.
acc_data_y = (acc_data_unrefined_y/100.)+9.8
acc_data_x = (acc_data_unrefined_x/100.)+9.8

start_time = 2000
end_time = 6000

# Starting the array from the 20th element instead of 0
acc_window_y = acc_data_y[start_time:end_time]
```

```

acc_window_x = acc_data_x[start_time:end_time]
elapsed_time_window = elapsed_time[start_time:end_time]

# Create a scatter plot
plt.plot(elapsed_time_window, acc_window_y, color="blue")
plt.plot(elapsed_time_window, acc_window_x, color="green")

# Provide a title to the plot
plt.title('Raw Accelerometer Data')

# Label the y-axis
plt.ylabel('Acceleration (m/s^2)')

# Label the x-axis
plt.xlabel('Time (s)')
from scipy.optimize import least_squares

# Make reasonable guesses for parameters like amplitude, omega, offset and
phi based on the plot above
guess_amplitude = 7
guess_omega = 4.99
guess_offset = 0
guess_phi = 3

# Store all the guessed parameters in an array
guess_parameters = [guess_amplitude, guess_omega, guess_offset, guess_phi]

def sin_fit_fun(parameters, time):
    a = parameters[0]
    omega = parameters[1]
    offset = parameters[2]
    phi = parameters[3]
    y = a * np.sin(omega * time + phi) + offset
    return y

def get_residuals(parameters, data, x):
    residuals = np.abs(data - sin_fit_fun(parameters, x))

```

```

    return -residuals

# Create a sin function based on original guess parameters
theoretical_guess_function = sin_fit_fun(guess_parameters,
elapsed_time_window)

# Plot the guess function
plt.plot(elapsed_time_window,
theoretical_guess_function,color="green",label="Guessed sine function")

# Plot the original data
plt.plot(elapsed_time_window, acc_window_y, label="Accelerometer Data Y")

plt.xlabel("Time (s)")
plt.ylabel("Acceleration (m/s^2)")
plt.title("Accelerometer data with the initial guess sin function")
plt.legend()

# The imported least_squares function minimizes the residuals
res_lsq = least_squares(get_residuals, guess_parameters,
args=(acc_window_y,elapsed_time_window))

# We store the values of best possible parameters obtained for amplitude,
omega, offset and phase in best_parameters
best_parameters = res_lsq['x']

# The best fit sin function will have the best_parameters as an input to
the sin function that we created
acc_fitted_y = sin_fit_fun(best_parameters, elapsed_time_window) -
best_parameters[2]
acc_window_y_after_fit = acc_window_y - best_parameters[2]

# Plot the original data and the best fit function on the same plot
plt.plot(elapsed_time_window, acc_window_y_after_fit,label="Accelerometer
data X")
plt.plot(elapsed_time_window, acc_fitted_y, color = 'green', label="Best
fit function")
plt.xlabel("Time (s)")

```

```

plt.ylabel("Acceleration (m/s^2)")
plt.legend()
plt.title("Accelerometer data with the best fit sin function")

# Print the values of the best parameters
print('Best Amplitude: ' + str(best_parameters[0]))
print('Best Omega: ' + str(best_parameters[1]))
print('Best offset: ' + str(best_parameters[2]))
print('Best Phi: ' + str(best_parameters[3]))
plt.plot(elapsed_time_window, acc_fitted_y, color = 'green', label="Best
fit function")
plt.xlabel("Time (s)")
plt.ylabel("Acceleration (m/s^2)")
plt.title("Spring 1: Best Fit Acceleration in y-Direction 30 Degrees")

# Print the values of the best parameters
print('Best Amplitude: ' + str(best_parameters[0]))
print('Best Omega: ' + str(best_parameters[1]))
print('Best offset: ' + str(best_parameters[2]))
print('Best Phi: ' + str(best_parameters[3]))
from scipy.optimize import least_squares

# Make reasonable guesses for parameters like amplitude, omega, offset and
phi based on the plot above
guess_amplitude = 10
guess_omega = 5
guess_offset = 32
guess_phi = 2

# Store all the guessed parameters in an array
guess_parameters = [guess_amplitude, guess_omega, guess_offset, guess_phi]

def sin_fit_fun(parameters, time):
    a = parameters[0]
    omega = parameters[1]
    offset = parameters[2]
    phi = parameters[3]

```

```

y = a * np.sin(omega * time + phi) + offset
return y

def get_residuals(parameters, data, x):
    residuals = np.abs(data - sin_fit_fun(parameters, x))
    return -residuals

# Create a sin function based on original guess parameters
theoretical_guess_function = sin_fit_fun(guess_parameters,
elapsed_time_window)

# Plot the guess function
plt.plot(elapsed_time_window,
theoretical_guess_function,color="green",label="Guessed sine function")

# Plot the original data
plt.plot(elapsed_time_window, acc_window_x, label="Accelerometer Data Y")

plt.xlabel("Time (s)")
plt.ylabel("Acceleration (m/s^2)")
plt.title("Accelerometer data with the initial guess sin function")
plt.legend()

# The imported least_squares function minimizes the residuals
res_lsq = least_squares(get_residuals, guess_parameters,
args=(acc_window_x,elapsed_time_window))

# We store the values of best possible parameters obtained for amplitude,
omega, offset and phase in best_parameters
best_parameters = res_lsq['x']

# The best fit sin function will have the best_parameters as an input to
the sin function that we created
acc_fitted_x = sin_fit_fun(best_parameters, elapsed_time_window) -
best_parameters[2]
acc_window_x_after_fit = acc_window_x - best_parameters[2]

```



```

# Plot the original data and the best fit function on the same plot
plt.plot(elapsed_time_window, acc_window_x_after_fit, label="Accelerometer
data X")
plt.plot(elapsed_time_window, acc_fitted_x, color = 'green', label="Best
fit function")
plt.xlabel("Time (s)")
plt.ylabel("Acceleration (m/s^2)")
plt.legend()
plt.title("Accelerometer data with the best fit sin function")

# Print the values of the best parameters
print('Best Amplitude: ' + str(best_parameters[0]))
print('Best Omega: ' + str(best_parameters[1]))
print('Best offset: ' + str(best_parameters[2]))
print('Best Phi: ' + str(best_parameters[3]))
plt.plot(elapsed_time_window, acc_fitted_x, color = 'orange',
label="x-direction")
plt.plot(elapsed_time_window, acc_fitted_y, color = 'pink',
label="y-direction")
plt.xlabel("Time (s)")
plt.ylabel("Acceleration (m/s^2)")
plt.title("Spring 1: Best Fit Acceleration 30 Degrees")
plt.legend()

# Print the values of the best parameters
print('Best Amplitude: ' + str(best_parameters[0]))
print('Best Omega: ' + str(best_parameters[1]))
print('Best offset: ' + str(best_parameters[2]))
print('Best Phi: ' + str(best_parameters[3]))

```

Sample Code for analysis of Tracker data:

```

from google.colab import auth
auth.authenticate_user()
from oauth2client.client import GoogleCredentials
!pip install --upgrade -q gspread
!pip install gspread-dataframe
import numpy as np

```

```

import pandas as pd
import gspread
import matplotlib.pyplot as plt
from gspread_dataframe import get_as_dataframe, set_with_dataframe

gc = gspread.authorize(GoogleCredentials.get_application_default())
sheet =
gc.open_by_url('https://docs.google.com/spreadsheets/d/1Fc-K8IsLCcSTaw9_Zt
6mOFskGpaUJIeC8GosTQis4QQ/edit?usp=sharing')
sheet_df = get_as_dataframe(sheet.worksheet('Sheet1'))
time = sheet_df.loc[0:476 , "Time (s)"]
position = sheet_df.loc[0:476 , "position (m)"]
velocity = sheet_df.loc[0:476 , "velocity (m/s)"]
acceleration = sheet_df.loc[0:476 , "acceleration (m/s^2)"]
from scipy.optimize import least_squares

# Make reasonable guesses for parameters like amplitude, omega, offset and
phi based on the plot above
guess_amplitude = 0.25
guess_omega = 5.
guess_offset = 0.4
guess_phi = -1.9

# Store all the guessed parameters in an array
guess_parameters = [guess_amplitude, guess_omega, guess_offset, guess_phi]

def sin_fit_fun(parameters, time):
    a = parameters[0]
    omega = parameters[1]
    offset = parameters[2]
    phi = parameters[3]
    y = a * np.sin(omega * time + phi) + offset
    return y

def get_residuals(parameters, data, x):
    residuals = np.abs(data - sin_fit_fun(parameters, x))
    return -residuals

```

```

# Create a sin function based on original guess parameters
theoretical_guess_function = sin_fit_fun(guess_parameters, time)

# Plot the guess function
plt.plot(time, theoretical_guess_function, color="green", label="Guessed
sine function")

# Plot the original data
plt.plot(time, position, label="radial position")

plt.xlabel("Time (s)")
plt.legend()

# The imported least_squares function minimizes the residuals
res_lsq = least_squares(get_residuals, guess_parameters, args=(position,
time))

# We store the values of best possible parameters obtained for amplitude,
omega, offset and phase in best_parameters
best_parameters = res_lsq['x']

# The best fit sin function will have the best_parameters as an input to
the sin function that we created
fitted_function = sin_fit_fun(best_parameters, time)

# Plot the original data and the best fit function on the same plot
plt.plot(time, position, label="Position data from Tracker")
plt.plot(time, fitted_function, color = 'green', label="Best fit
function")
plt.xlabel("Time (s)")
plt.ylabel("Position (m)")
plt.legend()
plt.title("Spring 1: Radial Displacement vs Time (15 Degrees)")

# Print the values of the best parameters
print('Best Amplitude: ' + str(best_parameters[0]))
print('Best Omega: ' + str(best_parameters[1]))

```

```

print('Best offset: ' + str(best_parameters[2]))
print('Best Phi: ' + str(best_parameters[3]))

# Make reasonable guesses for parameters like amplitude, omega, offset and
phi based on the plot above
guess_amplitude = 1.5
guess_omega = 5.
guess_offset = 0.4
guess_phi = -1.9

# Store all the guessed parameters in an array
guess_parameters = [guess_amplitude, guess_omega, guess_offset, guess_phi]

def sin_fit_fun(parameters, time):
    a = parameters[0]
    omega = parameters[1]
    offset = parameters[2]
    phi = parameters[3]
    y = a * np.sin(omega * time + phi) + offset
    return y

def get_residuals(parameters, data, x):
    residuals = np.abs(data - sin_fit_fun(parameters, x))
    return -residuals

# Create a sin function based on original guess parameters
theoretical_guess_function = sin_fit_fun(guess_parameters, time)

# Plot the guess function
plt.plot(time, theoretical_guess_function, color="green", label="Guessed
sine function")

# Plot the original data
plt.plot(time, velocity, label="radial position")

plt.xlabel("Time (s)")
plt.legend()

```

```

# The imported least_squares function minimizes the residuals
res_lsq = least_squares(get_residuals, guess_parameters, args=(velocity,
time))

# We store the values of best possible parameters obtained for amplitude,
omega, offset and phase in best_parameters
best_parameters = res_lsq['x']

# The best fit sin function will have the best_parameters as an input to
the sin function that we created
fitted_function = sin_fit_fun(best_parameters, time)

# Plot the original data and the best fit function on the same plot
plt.plot(time, velocity, label="instantaneous velocity from Tracker data")
plt.plot(time, fitted_function, color = 'green', label="Best fit
function")
plt.xlabel("Time (s)")
plt.ylabel("Velocity (m/s)")
plt.legend()
plt.title("Spring 1: Radial Velocity vs Time (15 Degrees)")

# Print the values of the best parameters
print('Best Amplitude: ' + str(best_parameters[0]))
print('Best Omega: ' + str(best_parameters[1]))
print('Best offset: ' + str(best_parameters[2]))
print('Best Phi: ' + str(best_parameters[3]))

# Make reasonable guesses for parameters like amplitude, omega, offset and
phi based on the plot above
guess_amplitude = 10
guess_omega = 5.
guess_offset = 0.4
guess_phi = 0

# Store all the guessed parameters in an array
guess_parameters = [guess_amplitude, guess_omega, guess_offset, guess_phi]

```

```

def sin_fit_fun(parameters, time):
    a = parameters[0]
    omega = parameters[1]
    offset = parameters[2]
    phi = parameters[3]
    y = a * np.sin(omega * time + phi) + offset
    return y

def get_residuals(parameters, data, x):
    residuals = np.abs(data - sin_fit_fun(parameters, x))
    return -residuals

# Create a sin function based on original guess parameters
theoretical_guess_function = sin_fit_fun(guess_parameters, time)

# Plot the guess function
plt.plot(time, theoretical_guess_function, color="green", label="Guessed
sine function")

# Plot the original data
plt.plot(time, acceleration, label="radial acceleration")

plt.xlabel("Time (s)")
plt.legend()

# The imported least_squares function minimizes the residuals
res_lsq = least_squares(get_residuals, guess_parameters,
args=(acceleration, time))

# We store the values of best possible parameters obtained for amplitude,
omega, offset and phase in best_parameters
best_parameters = res_lsq['x']

# The best fit sin function will have the best_parameters as an input to
the sin function that we created
fitted_function = sin_fit_fun(best_parameters, time)

```

```
# Plot the original data and the best fit function on the same plot
plt.plot(time, acceleration, label="Instantaneous Acceleration from Tracker
Data")
plt.plot(time, fitted_function, color = 'pink', label="Best fit function",
linewidth = 2)
plt.xlabel("Time (s)")
plt.ylabel("Acceleration (m/s)")
plt.legend()
plt.title("Spring 1: Radial Acceleration vs Time (15 Degrees)")

# Print the values of the best parameters
print('Best Amplitude: ' + str(best_parameters[0]))
print('Best Omega: ' + str(best_parameters[1]))
print('Best offset: ' + str(best_parameters[2]))
print('Best Phi: ' + str(best_parameters[3]))
```