Slinky Waves

Chris Compierchio

Abstract:

The purpose of the experiment was to analyze the motion of waves through a hanging slinky in different scenarios. A slinky was suspended by string and waves were driven into it so that the oscillations could be examined.

Introduction:

The main goal of this experiment was to study the behaviour of waves through a hanging slinky. Each ring of the slinky was suspended by a piece of string so that they could behave as mini pendulums. Waves were driven into the slinky so that the rings would begin to oscillate and transfer energy to one another, creating wave-like motion. Different frequencies were tested and combs were also used to determine the wave behaviour with the combs interfering with the motion of the strings. Theory of waves was important for this experiment as the wavenumber, slinky frequency, wave speed and ring frequency were all related through the formula $k = \frac{\sqrt{\omega_o^2 - \omega_o}}{c_o}$.

Equipment:

- **Slinky Setup:** This includes a wooden frame used to hang the slinky with string, as well as the driving motor used to induce waves into the slinky.
- **Meter Stick:** Used to take measurements of length (i.e. the length of the strings)
- **2 Meter Stick**: Used to take longer length measurements (i.e. length of the slinky)
- **Timer:** Used to take time measurements (cellphone used in this case)

Procedure:

This experiment was composed of eleven different parts. In the first part, the slinky had to be disturbed by hand and the wave created was timed as it moved through the slinky. This gave an estimate for the wave speed (c_o). The next task was to find the angular frequency for one ring (ω_o) by finding the vertical length of the strings and using the formula $\omega_o = \sqrt{\frac{g}{L}}$. The two values found in the first two steps were found directly by taking measurements, however in step 3, the same values were calculated, but instead by using the formula shown in the 'Introduction' section. Values for k^2 and ω^2 were found by taking measurements of frequency and distance between nodes. These parameters were plotted in python and the slope of the plot was taken using a linear curvefit to find c_o^2 and the y-intercept was found using the same method. The y-intercept is ω_o^2.

Once all the above parameters were found, further analysis of the wave behaviour in the slinky was conducted. The resonant frequencies were found through trial and error. Different driving frequencies were experimented with until resonance was visibly evident. Further analysis of this will be described in the 'Results' section. Other wave properties examined included the amplitude of the waves in relation to their distance from the driver when and $\omega << \omega_o$ and the relation between amplitude and distance from the fixed end when $\omega = \omega_o$.

Next, the combs were used to "shorten" the strings and change the frequency of the rings in each comb section. Combs 2 and 3 (middle and furthest from the driver) were lowered and a frequency was found so that resonance was evident in section 1 (closest to the driver). Using this same frequency, comb 3 was lifted and the energy transfer from section 1 to 3 was examined. This will be discussed in the 'Discussion' section. Lastly combs

2 and 3 were put in place again. While a wave propagated through the slinky, comb 3 was lifted and the driver was shut off simultaneously. The analysis of this situation leads to the calculation of the beat period, beat frequency, and resonant modes.

Results:

Both the vertical length of the strings (h) and the length of the slinky (L) were found using the meter sticks provided.

h = 0.885m ± 0.001m
L = 1.943m ± 0.001m

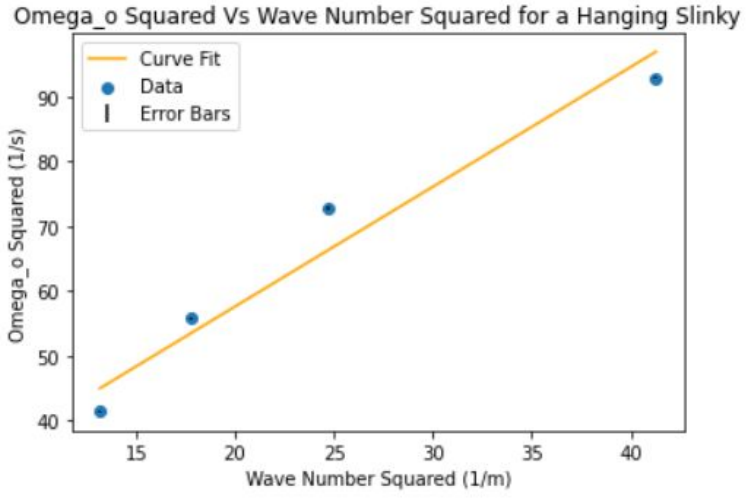
As mentioned above, c_o was found experimentally by sending a wave through the spring and timing its round trip. This means:

$$c_o \simeq \frac{2L}{t} \simeq \frac{2(1.943)}{2.44} \simeq 1.593 m/s \pm 0.002 m/s$$

ω_o was found by using the formula explained in the 'Procedure' section:

$$\omega_o \simeq \sqrt{\frac{g}{L}} \simeq \sqrt{\frac{g}{(1.943)}} \simeq 3.318\ s^{-1} \pm 0.001 s^{-1}$$

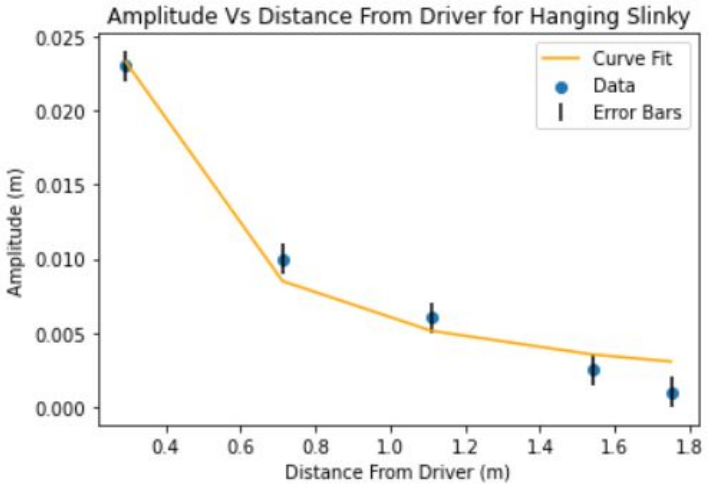
Using Python to plot the relationship between k^2 and ω^2 at $\omega > \omega_o$ resulted in the following plot:



From this plot c_o^2 was found to be about 1.850 and ω_o^2 20.582.

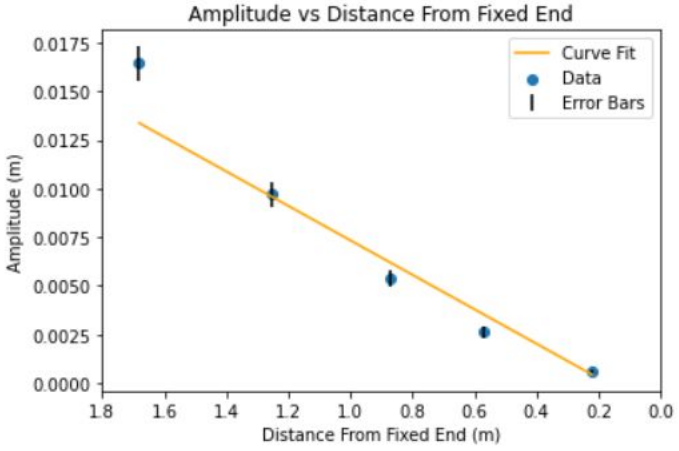
As mentioned before, the resonance frequencies were found through trial and error until resonance was visually present. Through my observations, it seemed the resonant frequencies occurred at integer multiples of 3 which is approximately the calculated frequency found for part 1, although the following formula would be a more accurate calculation of these frequencies: $\omega = \sqrt{\omega_o^2 + \frac{n^2\pi^2 c_o^2}{L^2}}$.

Next, the amplitude of the ring oscillations was plotted vs the distance from the driver for $\omega << \omega_o$:



Further analysis of this plot can be found in the 'Discussion' section.

Now analyzing the situation where $\omega = \omega_o$, the plot below shows the linear decay relationship between amplitude and distance from the fixed end:



For sections 8 to 11, a frequency of $0.907s^{-1}$ +/- $0.001s^{-1}$ was the frequency used as this exhibited resonance while combs 2 and 3 were down. Using this frequency as a reference point, some frequencies close to that value were tested to find the beat frequency and beat period by lifting comb 2 and measuring the frequencies in section 1 and section 3:

$$f_{beat} = |f_3 - f_1| = |1.089 - 0.905| = 0.184s^{-1} \pm 0.001s^{-1}$$

$$T_{beat} = \frac{1}{0.184} = 5.435s \pm 0.001s$$

To find a normal mode, I knew the beat frequency had to be zero, therefore I found a frequency at which $f_3 > f_1$ so that I can get a frequency difference less than 0. With this frequency less than 0 and the one above greater than 0, I could estimate the frequency of

one normal mode as being somewhere in the middle. The two frequencies in this case were $f_1 = 1.033s^{-1} +/- 0.001s^{-1}$ and $f_3 = 1.010s^{-1} +/- 0.001s^{-1}$ The second observed frequency difference was $-0.023s^{-1} +/- 0.001s^{-1}$. Now I knew that the normal mode beat frequency was in between these two values, however, I wanted to find the driving frequency at which a normal mode occurred so multiplying both f_1 values found above by 2π gave me the angular driving frequencies. This told me the angular beat frequency was between these values:

$$f_{d1} < f_{node} < f_{d2} \quad \Rightarrow \quad \omega_{d1} < \omega_{node} < \omega_{d2} \quad \Rightarrow \quad 5.686 < \omega_{node} < 6.347$$

Discussion:

In the first plot above, there is a clear linear relationship between k^2 and ω^2. This means the two values are directly proportional to each other, so changing the angular frequency will change the wavenumber at the same rate. The points are not completely in line with the curvefit, however, this will be discussed with the error analysis below.

The second plot shows the exponential decay of the oscillation amplitude for each ring as the distance of the rings from the driver increased. This relationship is due to dissipation. As the wave travels through the rings, it loses energy due to internal friction in the slinky so the energy near the end of the slinky is much lower resulting in a lower frequency. It is possible to choose a frequency that causes exponential decay near the driven end. For example, if omega is close to 0 (about 0-10), the wave dies out rapidly and the decay is quick due to the lack of input energy in the slinky.

The third plot shows the linear decay between the amplitude and the distance from the fixed end. This function can be thought of as the envelope function for the motion. The wave position will take a sinusoidal shape, however, the loss of energy will cause the amplitudes to decay linearly from the fixed end.

With combs 1 and 2 lowered, it is clear that section 3 can exhibit resonance as the energy seems to tunnel throughout the combs and reach section 3. This also results in large amplitudes in section 3 compared to that of section 1 and 2 since the strings in the first two sections are heavily restricted in motion due to the combs, while section 3 is free to oscillate.

Lastly, the beat frequency of tunneling occurs when only section 2 has the combs blocking it while 1 and 3 oscillate at the same rate, but in different directions. This is similar to a standing wave where the wave appears to be stationary. In this situation, the wave also appears to be stationary as the free sections swing like pendulums. This is not necessarily the same as the resonance frequency can occur when the frequencies in sections 1 and 3 oscillate at the same rate but in the same direction.

In terms of errors for this experiment, there were a lot of errors and uncertainties to take into account. Each piece of equipment had its own error measurement or even a deformity. For example, the slinky was hung by several pieces of string and it is almost impossible to get each piece of string to hang at the same length. Obviously, this would cause an error in values dependent on the string length like ω_o. Another noticeable error was the gap between values in the velocity knob of the driver. The ticks around the knob went up in increments in 5. Those are large increments and a slight reading error could result in different experimental outcomes than expected. These errors are most likely the cause for the discrepancies in the plots above as well as the error in the measurements and calculated values.

Conclusion:

The results of this experiment turned out as expected and all observed relationships fit their theoretical descriptions.

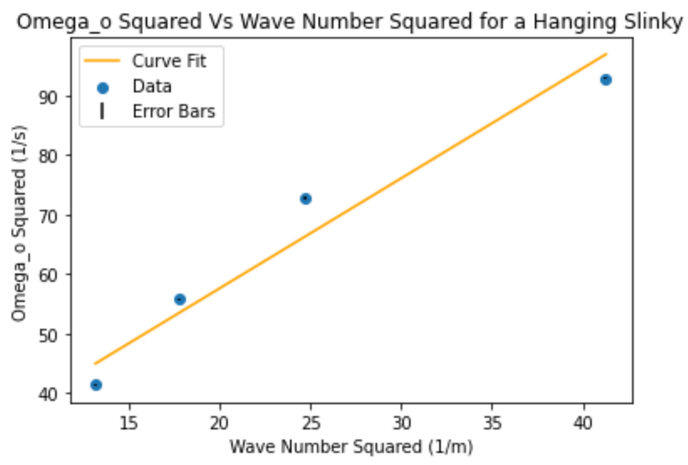
In [10]:
```
#import libraries
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
%matplotlib inline
```

In [69]:
```
#Import motor velocity, time, frequency, node distance, angular frequency, k, and the errors for each
v, t, f, d, omega, k, t_err, f_err, d_err, omega_err, k_err = np.loadtxt("Slinky part 3.csv"
```

In [70]:
```
#define linear and exponential functions for curvefit
def f(x, a, b):
    return (a*(x) + b)
def g(x, m, b):
    return m*(x**b)
```

In [71]:
```
#Plot k^2 vs omega^2, fit the data and plot error bars
plt.scatter(k**2, omega**2, label = "Data")

sig = (omega**2)*2*(omega_err/omega)

p_opt, p_cov = curve_fit(f, k**2, omega**2, p0 = (1,2), sigma = sig, absolute_sigma = True)

c_sq = p_opt[0]
omega0_sq = p_opt[1]

plt.plot(k**2, f(k**2, c_sq, omega0_sq), color = "orange", label = "Curve Fit")

plt.ylabel("Omega_o Squared (1/s)")
plt.xlabel("Wave Number Squared (1/m)")
plt.title( "Omega_o Squared Vs Wave Number Squared for a Hanging Slinky")

plt.errorbar(k**2, omega**2, yerr=sig, color = "black", ls="none", label = "Error Bars")
plt.legend(loc="best")

#Print the found values for c squared and omega0 squared
print("c squared is:",c_sq)
print("omega0 squared is:",omega0_sq)
```

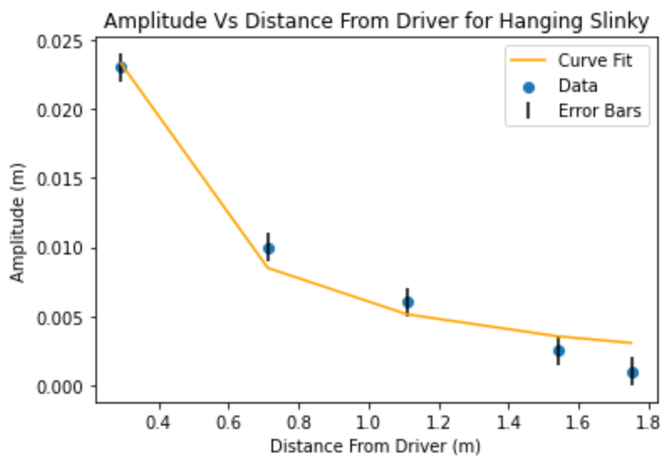
```
c squared is: 1.8500351979046554
omega0 squared is: 20.581941970142296
```



In [35]:
```
#Import distance from driver, amplitude and their errors
D, A, D_err, A_err = np.loadtxt("Slinky part 5.csv", skiprows = 1, delimiter = ',', unpack =
```

In [36]:
```
#Print Distance from driver vs Amplitude, fit the data and plot errorbars
plt.scatter(D, A, label = "Data")

sig2 = A_err

p_opt2, p_cov2 = curve_fit(g, D, A, p0 = (0.5,0.5), sigma = sig2, absolute_sigma = True)

plt.plot(D, g(D, p_opt2[0], p_opt2[1]), color = "orange", label = "Curve Fit")

plt.errorbar(D, A, yerr=sig2, color = "black", ls="none", label = "Error Bars")

plt.ylabel("Amplitude (m)")
plt.xlabel("Distance From Driver (m)")
plt.title( "Amplitude Vs Distance From Driver for Hanging Slinky")
plt.legend(loc="best")
```

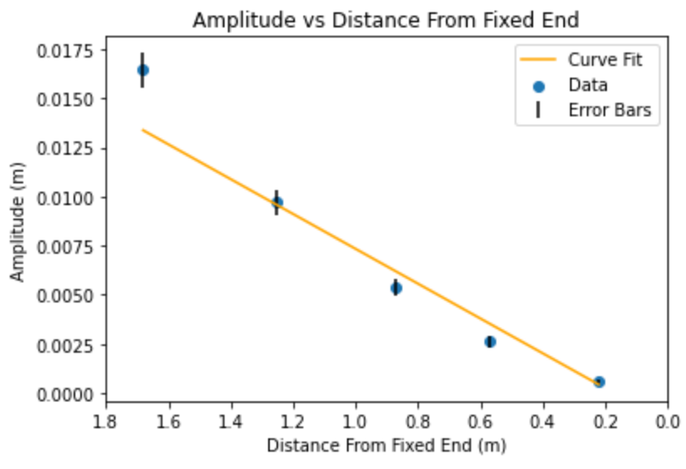
Out[36]:
```
<matplotlib.legend.Legend at 0x2cbc2a632e0>
```



In [56]:
```
x, yd, x_err, yd_err = np.loadtxt("Slinky part 6.csv", skiprows = 1, delimiter = ',', unpack
```

In [67]:
```
#Define the length of the slinky
L = 1.943
L_err = 0.001

#compute amplitude
y = yd*(x/L)

#plot the Amplitude vs Distance from the fixed end
plt.scatter(x, y, label = "Data")

#compute error in amplitude
sig3 = ((yd*x)*(np.sqrt((yd_err**2/yd**2)+(x_err**2/x**2))))
sig4 = (yd*(x/L))*np.sqrt((sig3**2/(yd*x)**2)+(L_err**2/L**2))

#plot curvefit and error bars
p_opt3, p_cov3 = curve_fit(f, x, y, p0 = (1,1), sigma = sig4, absolute_sigma = True)

plt.plot(x, f(x, p_opt3[0], p_opt3[1]), color = "orange", label = "Curve Fit")

plt.ylabel("Amplitude (m)")
plt.xlabel("Distance From Fixed End (m)")
plt.xlim(1.8,0)
plt.title( "Amplitude vs Distance From Fixed End")

plt.errorbar(x, y, yerr=sig4, color = "black", ls="none", label = "Error Bars")
plt.legend(loc="best")
```

Out[67]:
```
<matplotlib.legend.Legend at 0x2cbc41781c0>
```



In [ ]: