PHYS 229 - Ryan Kaufmann/Experiment 1/Acoustics - Prelab

SIGNED by Ryan Kaufmann Feb 08, 2018 @12:43 PM PST

Ryan Kaufmann Jan 25, 2018 @04:11 PM PST

Acoustics 1: Prelab

Ryan Kaufmann Jan 25, 2018 @04:16 PM PST

In order to analytically calculate the resonant frequencies of a one-dimensional resonator, we must have the speed of propagation of the relevant waves and the length of the resonator. We calculated the height of our resonator (i.e. the insulated box) to be approximately 1.04m. Furthermore, by looking online, we can find that the speed of sound, our wave, is approximately 343 m/s. Then using the equation given to us, we can calculate the resonant frequencies:

0.437m

Ryan Kaufmann Jan 25, 2018 @04:19 PM PST

$$f_n = \frac{n * v_s}{2X}$$

$$f_n = \frac{n * 343m/s}{2 * 1.04m}$$

$$f_n = \frac{343n}{2.08} Hz$$

$$f_n = n164.90Hz$$

Ryan Kaufmann Jan 25, 2018 @04:20 PM PST

Then the first five resonant frequencies are approximately:

Ryan Kaufmann Jan 25, 2018 @04:21 PM PST

$$f_1 = 164.90Hz$$

$$f_2 = 329.81Hz$$

$$f_3 = 494.71Hz$$

$$f_4 = 659.62Hz$$

$$f_5 = 824.52Hz$$

Ryan Kaufmann Feb 07, 2018 @03:04 PM PST

However, this assumes that the lengths in any direction besides the height is negligible. If we consider the x-direction and y-direction, we must consider that the wave propagates in three dimensions instead of one. We can then try to find the resonant frequencies when we consider three dimensions. We measured the dimensions of our insulated box. We found that in addition to the 1.04m height, the box had a width and length of roughly 0.437m. Then we can apply a formula we had found online:

Ryan Kaufmann Feb 07, 2018 @03:04 PM PST

$$\mathrm{f}_{\mathrm{mnq}} = \mathrm{c}_{\mathrm{s}}\left[(\mathrm{m}/2\mathrm{a})^2 + (\mathrm{n}/2\mathrm{b})^2 + (\mathrm{q}/2\mathrm{d})^2\right]^{0.5} \; [\mathrm{Hz}] \quad \text{(resonant frequencies)} \quad (13.2.33)$$

ResonanceEquation.JPG(19.3 KB)

Ryan Kaufmann Feb 07, 2018 @03:11 PM PST

Where m, n, and q are integers; a, b, and d are the dimensions of the box; and Cs is the velocity of propagation (i.e. the speed of sound). This formula was retrieved from a MIT electrical engineering textbook (https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-013-electromagnetics-and-applications-spring-2009/readings/MIT6_013S09_chap13.pdf). Then we can find resonant frequencies by plugging in our dimensions and the speed of sound into our equation. Thus the equation becomes:

Ryan Kaufmann Feb 07, 2018 @03:24 PM PST

$$f_{mnq} = c_s((\frac{m}{2a})^2 + (\frac{n}{2b})^2 + (\frac{q}{2d})^2)^{0.5}$$

$$f_{mnq} = 343m/s * (\frac{m^2}{4*0.437^2} + \frac{n^2}{4*0.437^2} + \frac{q^2}{4*1.04^2})^{0.5}(1/m)$$

$$f_{mnq} = 343 * (\frac{m^2+n^2}{0.763876} + \frac{q^2}{4.3264})^{0.5}Hz$$

Ryan Kaufmann Feb 07, 2018 @03:36 PM PST

We can then plug in various values for m, n, and q to find some small values of resonance frequencies.

Ryan Kaufmann Feb 07, 2018 @03:48 PM PST

$$f_{100} = f_{010} = 392.45Hz$$

$$f_{001} = 164.90Hz$$

$$f_{110} = 555.01Hz$$

$$f_{101} = f_{011} = 425.69Hz$$

$$f_{111} = 578.97Hz$$

PHYS 229 - Ryan Kaufmann/Experiment 1/Acoustics - Lab

SIGNED by Ryan Kaufmann Feb 08, 2018 @07:38 PM PST

Ryan Kaufmann Jan 25, 2018 @04:12 PM PST

Acoustics 2: Categorizing Resonance

Ryan Kaufmann Feb 07, 2018 @04:56 PM PST

We can begin this experiment by looking for the resonant frequencies of the rectangular resonator. We start by opening up software in order to creating and analyze frequencies. In order to create constant frequencies, we downloaded and used SigGenfreeware. SigGenfreeware let us output specific frequencies to be played through a speaker in the resonator. In order to analyze the resonance from the microphone, we downloaded and used trueRTA. When connected to the microphone using a USB drive, trueRTA displays the volume of each of the frequency ranges in millivolts and decibels. Then we can find the resonant frequencies by producing various frequencies and watch where trueRTA gives us the largest expressions of volume in either millivolts or decibels. To get a good range of frequencies and find several resonant frequencies. We picked intervals of 25Hz starting from 100Hz, and additionally 50Hz and 10Hz. When we found a frequency that produced a larger volume than the others, we searched around the frequency of highest volume in smaller and smaller intervals until we found the largest volume produced by a frequency. We labeled this our resonant frequency. We got the following data set:

Ryan Kaufmann Feb 07, 2018 @04:56 PM PST

AcousticsDataP1.csv(605 Bytes)

Ryan Kaufmann Feb 07, 2018 @05:13 PM PST

We isolated four resonant frequencies. From plotting the data, it seems as though there may be additional resonant frequencies in between the ones we found, or that the data did not show the end of its downturn. We continue to have a solid data set able to show us four resonance frequencies. As we are told, the area around a resonant frequency should resemble a Lorentzian distribution. Recall that a Lorentzian distribution has the following form:

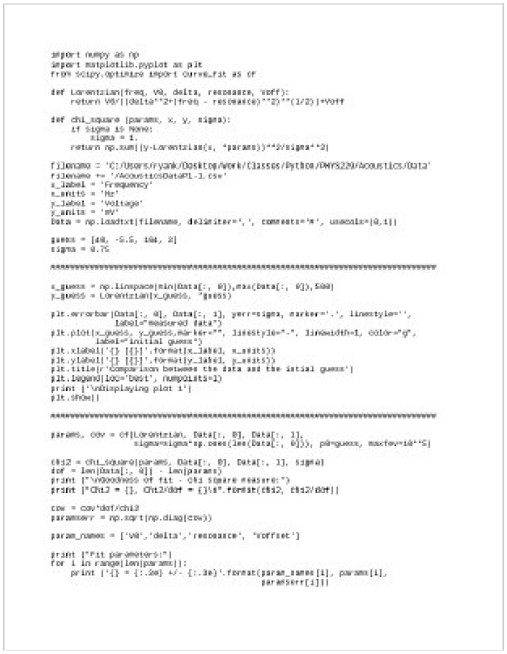
Ryan Kaufmann Feb 07, 2018 @05:17 PM PST

$$V(f) = \frac{\Delta V_0}{\sqrt{\Delta^2 + (f - f_r)^2}}$$

Ryan Kaufmann Feb 07, 2018 @05:51 PM PST

Where V represents volume in volts, f represents frequency, and delta is the width of the curve. We added a constant term to account for any background noise that the microphone in the resonator could have picked up. Then using python we individually fit the area around each resonant frequency to the Lorentzian distribution. We used the following code:

Ryan Kaufmann Feb 07, 2018 @05:52 PM PST



```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit as cf

def Lorentzian(freq, V0, delta, resonance, Voff):
    return V0/((delta**2+(freq - resonance)**2)**(1/2))+Voff

def chi_square (params, x, y, sigma):
    if sigma is None:
        sigma = 1.
    return np.sum((y-Lorentzian(x, *params))**2/sigma**2)

filename = 'C:/Users/ryank/Desktop/Work/Classes/Python/PHYS229/Acoustics/Data'
filename += '/AcousticsDataP1-1.csv'
x_label = 'Frequency'
x_units = 'Hz'
y_label = 'Voltage'
y_units = 'mV'
Data = np.loadtxt(filename, delimiter=',', comments='#', usecols=(0,1))

guess = [40, -5.5, 164, 2]
sigma = 0.75

################################################################################

x_guess = np.linspace(min(Data[:, 0]),max(Data[:, 0]),500)
y_guess = Lorentzian(x_guess, *guess)

plt.errorbar(Data[:, 0], Data[:, 1], yerr=sigma, marker='.', linestyle='',
             label="measured data")
plt.plot(x_guess, y_guess,marker="", linestyle="-", linewidth=1, color="g",
         label="initial guess")
plt.xlabel('{} [{}]'.format(x_label, x_units))
plt.ylabel('{} [{}]'.format(y_label, y_units))
plt.title(r'Comparison between the data and the intial guess')
plt.legend(loc='best', numpoints=1)
print ('\nDisplaying plot 1')
plt.show()

################################################################################

params, cov = cf(Lorentzian, Data[:, 0], Data[:, 1],
                  sigma=sigma*np.ones(len(Data[:, 0])), p0=guess, maxfev=10**5)

chi2 = chi_square(params, Data[:, 0], Data[:, 1], sigma)
dof = len(Data[:, 0]) - len(params)
print ("\nGoodness of fit - chi square measure:")
print ("Chi2 = {}, Chi2/dof = {}\n".format(chi2, chi2/dof))

cov = cov*dof/chi2
paramserr = np.sqrt(np.diag(cov))

param_names = ['V0','delta','resonance', 'Voffset']

print ("Fit parameters:")
for i in range(len(params)):
    print ('{} = {:.3e} +/- {:.3e}'.format(param_names[i], params[i],
                                            paramserr[i]))
```

CurveFitLorentzian.py(4.2 KB)

Ryan Kaufmann Feb 07, 2018 @05:52 PM PST

We found the following for the four resonant frequencies:

Ryan Kaufmann Feb 07, 2018 @06:24 PM PST

$$f_r = 163.9 \pm 0.7 : \chi^2 = 1.67, V_0 = 46.9 \pm 8.6, \Delta = -5.423 \pm 1.044, V_{off} = 3.713 \pm 0.390$$

$$f_r = 302.7 \pm 0.1 : \chi^2 = 10.5, V_0 = 59.7 \pm 7.8, \Delta = -1.013 \pm 0.147, V_{off} = 4.751 \pm 1.511$$

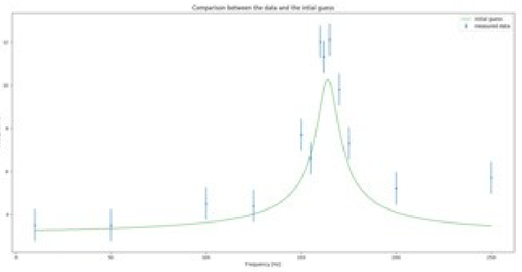
$$f_r = 428.7 \pm 0.9 : \chi^2 = 8.98, V_0 = 213.6 \pm 44.5, \Delta = -8.449 \pm 1.622, V_{off} = 12.16 \pm 1.10$$

$$f_r = 587.0 \pm 0.8 : \chi^2 = 2.84, V_0 = 650.2 \pm 28.9, \Delta = -3.013 \pm 0.138, V_{off} = 2.464 \pm 1.884$$

Ryan Kaufmann Feb 07, 2018 @06:25 PM PST

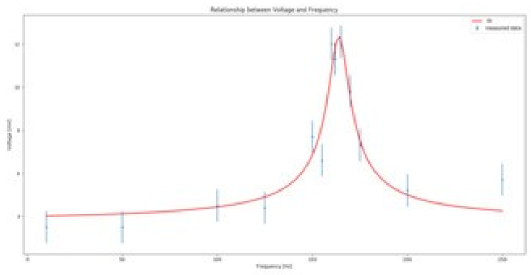
And the following graphs:

Ryan Kaufmann Feb 07, 2018 @06:27 PM PST



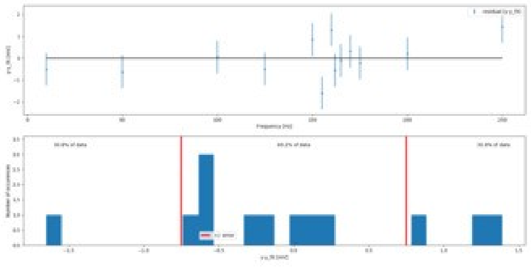
Lorentz1Initial.png(48.5 KB)

Ryan Kaufmann Feb 07, 2018 @06:27 PM PST



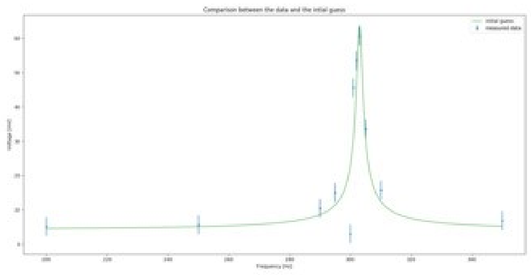
Lorentz1Fitted.png(47 KB)

Ryan Kaufmann Feb 07, 2018 @06:27 PM PST



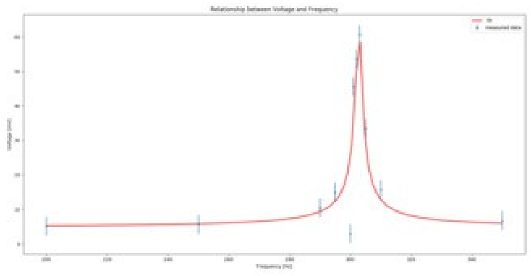
Lorentz1Resids.png(37 KB)

Ryan Kaufmann Feb 07, 2018 @06:27 PM PST



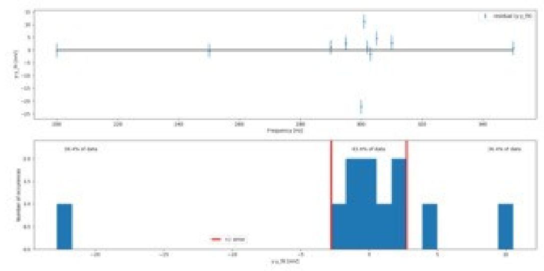
Lorentz2Initial.png(47.6 KB)

Ryan Kaufmann Feb 07, 2018 @06:27 PM PST



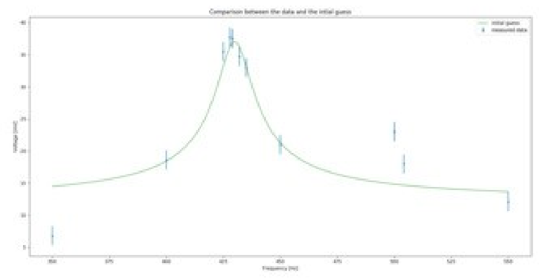
Lorentz2Fitted.png(44.1 KB)

Ryan Kaufmann Feb 07, 2018 @06:27 PM PST



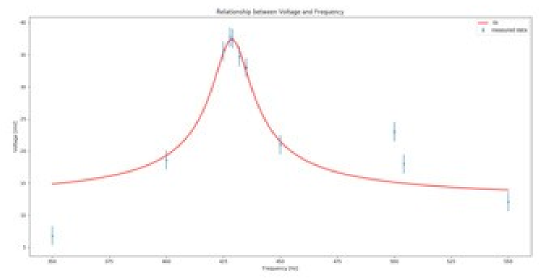
Lorentz2Resids.png(37.6 KB)

Ryan Kaufmann Feb 07, 2018 @06:27 PM PST



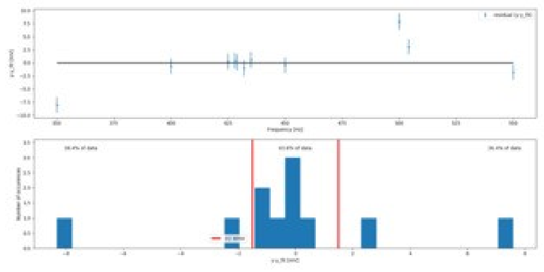
Lorentz3Initial.png(52.2 KB)

Ryan Kaufmann Feb 07, 2018 @06:27 PM PST



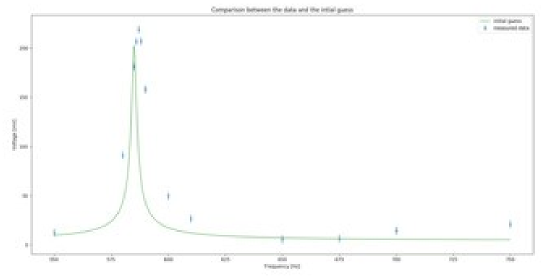
Lorentz3Fitted.png(49.6 KB)

Ryan Kaufmann Feb 07, 2018 @06:27 PM PST



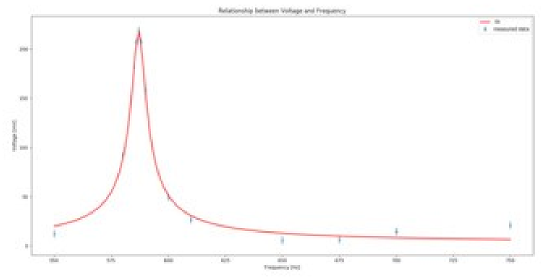
Lorentz3Resids.png(41.7 KB)

Ryan Kaufmann Feb 07, 2018 @06:27 PM PST



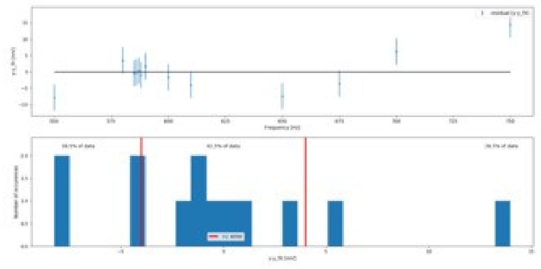
Lorentz4Initial.png(45.5 KB)

Ryan Kaufmann Feb 07, 2018 @06:27 PM PST



Lorentz4Fitted.png(47.2 KB)

Ryan Kaufmann Feb 07, 2018 @06:27 PM PST

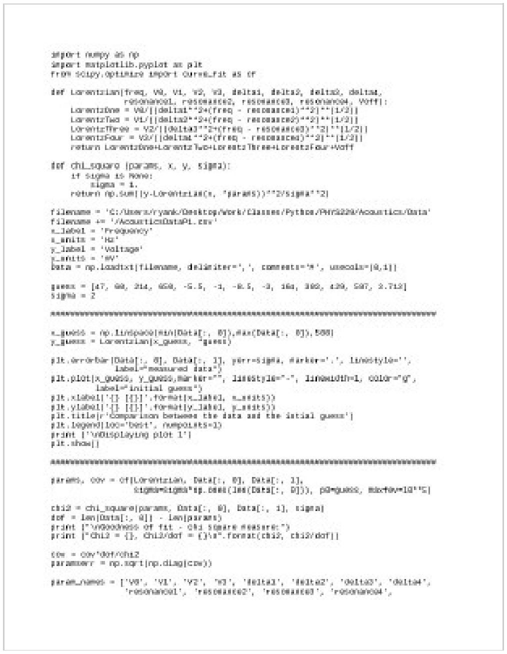


Lorentz4Resids.png(35.9 KB)

Ryan Kaufmann Feb 07, 2018 @06:30 PM PST

We can also graph this four Lorentzians together to see if there is a better fit. Thus we created another script in order to fit the four Lorentzians:

Ryan Kaufmann Feb 07, 2018 @06:30 PM PST



```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit as cf

def Lorentzian(freq, V0, V1, V2, V3, delta1, delta2, delta3, delta4,
               resonance1, resonance2, resonance3, resonance4, Voff):
    LorentzOne = V0/((delta1**2+(freq - resonance1)**2)**(1/2))
    LorentzTwo = V1/((delta2**2+(freq - resonance2)**2)**(1/2))
    LorentzThree = V2/((delta3**2+(freq - resonance3)**2)**(1/2))
    LorentzFour = V3/((delta4**2+(freq - resonance4)**2)**(1/2))
    return LorentzOne+LorentzTwo+LorentzThree+LorentzFour+Voff

def chi_square (params, x, y, sigma):
    if sigma is None:
        sigma = 1.
    return np.sum((y-Lorentzian(x, *params))**2/sigma**2)

filename = 'C:/Users/ryank/Desktop/Work/Classes/Python/PHYS229/Acoustics/Data'
filename += '/AcousticsDataP1.csv'
x_label = 'Frequency'
x_units = 'Hz'
y_label = 'Voltage'
y_units = 'mV'
Data = np.loadtxt(filename, delimiter=',', comments='#', usecols=(0,1))

guess = [47, 60, 214, 650, -5.5, -1, -0.5, -3, 164, 302, 429, 587, 2.712]
sigma = 2

################################################################################

x_guess = np.linspace(min(Data[:, 0]),max(Data[:, 0]),500)
y_guess = Lorentzian(x_guess, *guess)

plt.errorbar(Data[:, 0], Data[:, 1], yerr=sigma, marker='.', linestyle='',
             label="measured data")
plt.plot(x_guess, y_guess,marker="", linestyle="-", linewidth=1, color="g",
         label="initial guess")
plt.xlabel('{} [{}]'.format(x_label, x_units))
plt.ylabel('{} [{}]'.format(y_label, y_units))
plt.title(r'Comparison between the data and the intial guess')
plt.legend(loc='best', numpoints=1)
print ('\nDisplaying plot 1')
plt.show()

################################################################################

params, cov = cf(Lorentzian, Data[:, 0], Data[:, 1],
                 sigma=sigma*np.ones(len(Data[:, 0])), p0=guess, maxfev=10**5)

chi2 = chi_square(params, Data[:, 0], Data[:, 1], sigma)
dof = len(Data[:, 0]) - len(params)
print ("\nGoodness of fit - chi square measure:")
print ("Chi2 = {}, Chi2/dof = {}\n".format(chi2, chi2/dof))

cov = cov*dof/chi2
paramserr = np.sqrt(np.diag(cov))

param_names = ['V0', 'V1', 'V2', 'V3', 'delta1', 'delta2', 'delta3', 'delta4',
               'resonance1', 'resonance2', 'resonance3', 'resonance4',
```

CurveFitFourLorentzians.py(4.8 KB)

Ryan Kaufmann Feb 07, 2018 @06:34 PM PST

Which gives us the following values and graph:

Ryan Kaufmann Feb 07, 2018 @06:47 PM PST

$$\chi^2 = 9.933$$

$$f_{r1} = 163.6 \pm 1.7Hz, V_1 = 34.74 \pm 15.6mV, \Delta_1 = -4.327 \pm 2.326$$

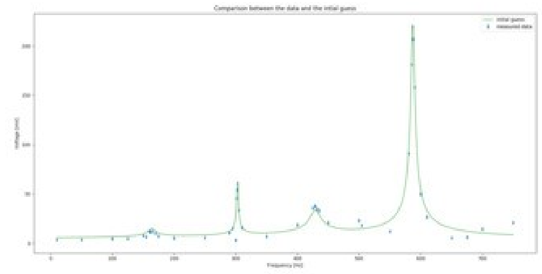
$$f_{r2} = 302.70 \pm 0.07Hz, V_2 = 52.49 \pm 3.91mV, \Delta_2 = -0.9009 \pm 0.0891$$

$$f_{r3} = 427.9 \pm 1.5Hz, V_3 = 394.7 \pm 59.2mV, \Delta_3 = -12.63 \pm 1.84$$

$$f_{r4} = 587.00 \pm 0.04Hz, V_4 = 640.2 \pm 12.3mV, \Delta_4 = -2.983 \pm 0.064$$

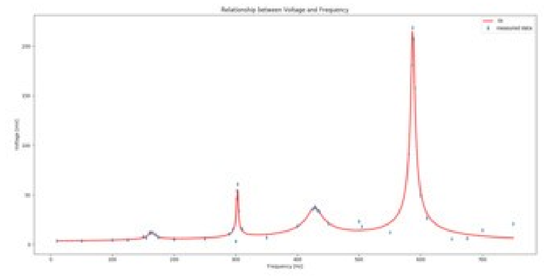
$$V_{off} = 1.123 \pm 0.787mV$$

Ryan Kaufmann Feb 07, 2018 @06:47 PM PST



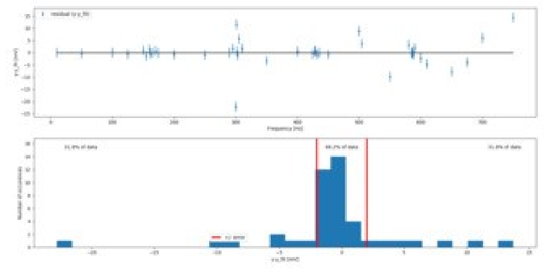
AllLorentzInitial.png(54.6 KB)

Ryan Kaufmann Feb 07, 2018 @06:47 PM PST



AllLorentzFitted.png(50.9 KB)

Ryan Kaufmann Feb 07, 2018 @06:47 PM PST



AllLorentzResids.png(41.2 KB)

Ryan Kaufmann Feb 07, 2018 @06:58 PM PST

We also noticed from the data of the resonant frequencies are around some of the frequencies we predicted in the prelab. Namely the predicted 164, 578, and 425 are close to three of the four resonant frequencies we received.

Ryan Kaufmann Feb 07, 2018 @06:27 PM PST

Acoustics 3: Perturbations by Solid Sphere

Ryan Kaufmann Feb 08, 2018 @10:40 AM PST

Now that we have gotten a good idea of the resonant frequencies, let us focus on a single one. Since we got the best reaction from the 587Hz resonant frequency, we will use it to test how the resonant frequency changes as we adjust the properties of the resonator.

In this next section, we will be adding a hollow glass sphere into the resonator. The glass sphere will be suspended in the resonator above the speaker by a brass rod connected to the top of the resonator. The brass rod will be held in place using a clamp so that it is also capable of being lowered and raised at our will. We began with the sphere at the highest point we could set it to, a measure of 14cm on the brass rod.

Once we had the sphere placed at our 14cm mark, we found an approximate resonant frequencies and took the value of the volume, in mV. Then we looked on either side of our frequency to find other frequencies that were half the value of our recorded volume, in mV. Once we received these two frequencies, we can average them to get a more accurate value of the resonant frequency. Then we lowered the sphere 7cm and repeated this measurement, until we had reached just below 80cm in total. We feared going over 80cm since we might have broken the speaker.

We get the following data sets for the perturbations from the sphere:

Ryan Kaufmann Feb 08, 2018 @10:41 AM PST

DataSolidSphereP2.csv(424 Bytes)

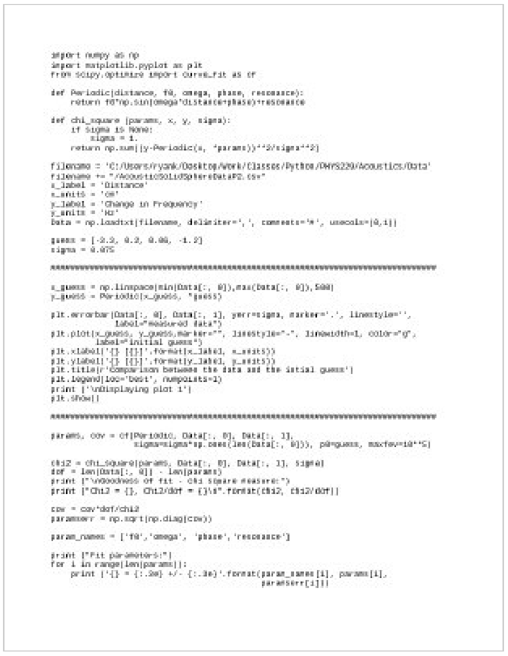
Ryan Kaufmann Feb 08, 2018 @10:41 AM PST

AcousticSolidSphereDataP2.csv(125 Bytes)

Ryan Kaufmann Feb 08, 2018 @11:06 AM PST

When we plot the data we received above as resonant frequency versus distance recorded from the brass rod, we notice a sinusoidal pattern in the data points. This coincides with what we were expecting in the data. It makes sens that the sphere would cause more perturbation if it is farther from a node then when it is on a node, creating a wave pattern that we see in this data. We can then fit the data to a sinusoidal curve around the no perturbed resonant frequency. We set up our script as so:

Ryan Kaufmann Feb 08, 2018 @11:07 AM PST



```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit as cf

def Periodic(distance, f0, omega, phase, resonance):
    return f0*np.sin(omega*distance+phase)+resonance

def chi_square (params, x, y, sigma):
    if sigma is None:
        sigma = 1.
    return np.sum((y-Periodic(x, *params))**2/sigma**2)

filename = 'C:/Users/ryank/Desktop/Work/Classes/Python/PHYS229/Acoustics/Data'
filename += "/AcousticSolidSphereDataP2.csv"
x_label = 'Distance'
x_units = 'cm'
y_label = 'Change in Frequency'
y_units = 'Hz'
Data = np.loadtxt(filename, delimiter=',', comments='#', usecols=(0,1))

guess = [-2.2, 0.2, 0.08, -1.2]
sigma = 0.075

################################################################################

x_guess = np.linspace(min(Data[:, 0]),max(Data[:, 0]),500)
y_guess = Periodic(x_guess, *guess)

plt.errorbar(Data[:, 0], Data[:, 1], yerr=sigma, marker='.', linestyle='',
             label="measured data")
plt.plot(x_guess, y_guess,marker="", linestyle="-", linewidth=1, color="g",
         label="initial guess")
plt.xlabel('{} [{}]'.format(x_label, x_units))
plt.ylabel('{} [{}]'.format(y_label, y_units))
plt.title(r'Comparison between the data and the intial guess')
plt.legend(loc='best', numpoints=1)
print ('\nDisplaying plot 1')
plt.show()

################################################################################

params, cov = cf(Periodic, Data[:, 0], Data[:, 1],
                 sigma=sigma*np.ones(len(Data[:, 0])), p0=guess, maxfev=10**5)

chi2 = chi_square(params, Data[:, 0], Data[:, 1], sigma)
dof = len(Data[:, 0]) - len(params)
print ("\nGoodness of fit - chi square measure:")
print ("Chi2 = {}, Chi2/dof = {}\n".format(chi2, chi2/dof))

cov = cov*dof/chi2
paramserr = np.sqrt(np.diag(cov))

param_names = ['f0','omega', 'phase','resonance']

print ("Fit parameters:")
for i in range(len(params)):
    print ('{} = {:.3e} +/- {:.3e}'.format(param_names[i], params[i],
                                           paramserr[i]))
```

CurveFitSpherePerturbation.py(4.3 KB)

Ryan Kaufmann Feb 08, 2018 @11:29 AM PST

When we attempt to fit the function, we get the following parameters and graphs:

Ryan Kaufmann Feb 08, 2018 @12:39 PM PST

$$\chi^2 = 2.003$$

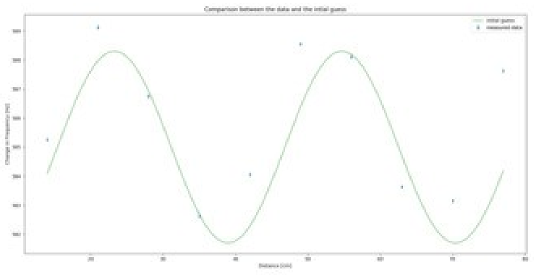
$$f_0 = -3.397 \pm 0.035$$

$$\omega = 0.2103 \pm 0.0004$$

$$\phi = 0.06253 \pm 0.02209$$

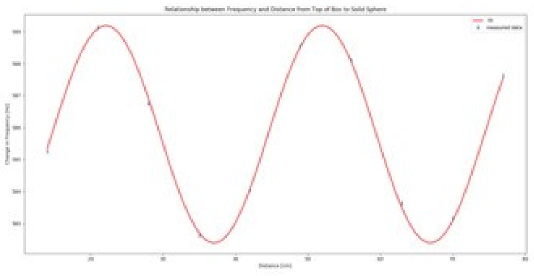
$$f_r = 585.80 \pm 0.02$$

Ryan Kaufmann Feb 08, 2018 @12:41 PM PST



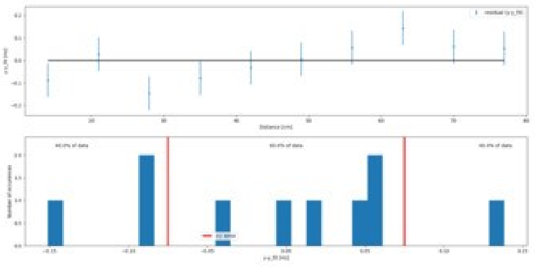
SolidSphereInitial.png(74.5 KB)

Ryan Kaufmann Feb 08, 2018 @12:41 PM PST



SolidSphereFitted.png(75.1 KB)

Ryan Kaufmann Feb 08, 2018 @12:41 PM PST



SolidSphereResids.png(35.7 KB)

Ryan Kaufmann Feb 08, 2018 @03:19 PM PST

Thus our function becomes:

Ryan Kaufmann Feb 08, 2018 @03:21 PM PST

$$f(x) = f_0 * \sin(\omega x + \phi) + f_r$$

$$f(x) = -3.397 * \sin(0.2103x + 0.06253) + 585.8$$

Ryan Kaufmann Feb 08, 2018 @02:46 PM PST

It seems like the data is fitted well by the sine curve. The chi-squared and the residuals indicated that the fit is not exact and could use improvement. We will take more data to better get a sense of how well the sine wave fits.

Otherwise, the data makes sense with our predictions.

Ryan Kaufmann Jan 25, 2018 @04:12 PM PST

Acoustics 4: Perturbations by Slotted Sphere

Ryan Kaufmann Feb 08, 2018 @12:12 PM PST

In this second portion, we will instead be looking at a slotted metal sphere rather than a solid glass sphere. Furthermore, instead of testing how the resonant frequency changes with height, we will be observing how it changes with rotation. In the next section we define zero degrees as the point where the slots on the sphere are vertical. Then we repeat the procedure from the last section.

At each measure of degree, we search for an approximate resonant frequency and record it’s value of volume in mV. Then we search of either side of the proposed resonant frequency for frequencies that are both approximately half the measured volume. Then the resonant frequency is more precisely around the average of the two half-volume frequencies.

We then performed this method for measures of 60 degrees until we completed a circle. Thus we get the following data:

Ryan Kaufmann Feb 08, 2018 @01:02 PM PST

DataSlottedSphereP3.csv(267 Bytes)

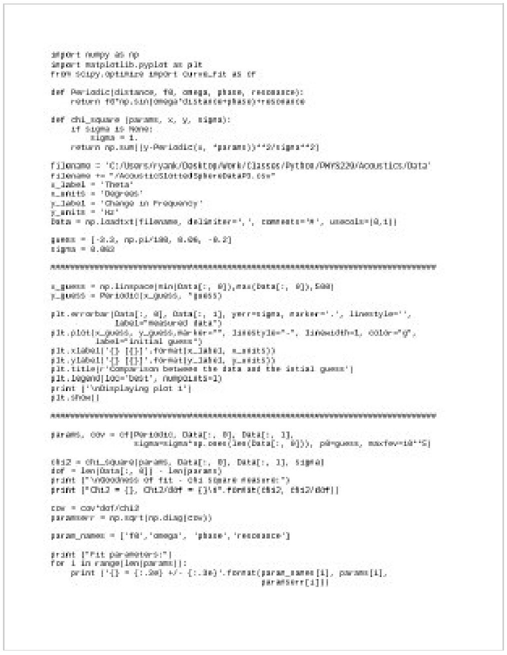
Ryan Kaufmann Feb 08, 2018 @01:02 PM PST

AcousticSlottedSphereDataP3.csv(78 Bytes)

Ryan Kaufmann Feb 08, 2018 @01:19 PM PST

We need another script for this data. We notice that as the data is at 0 and 180 degrees, the wave almost acts as if it wasn't there. Furthermore, the data is the same between 0 and 180 degrees and 180 to 360 degrees. This makes sense as the sphere simply repeats its orientation over those angles. Between 0 and 180 degrees, the data seems to represent a sine curve again. It seems that as we turn the sphere, it begins to act more and more like a solid sphere at that location, which is reasonable due to the definition and orientation of the slots. We then used the following script:

Ryan Kaufmann Feb 08, 2018 @01:19 PM PST



```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit as cf

def Periodic(distance, f0, omega, phase, resonance):
    return f0*np.sin(omega*distance+phase)+resonance

def chi_square (params, x, y, sigma):
    if sigma is None:
        sigma = 1.
    return np.sum((y-Periodic(x, *params))**2/sigma**2)

filename = 'C:/Users/ryank/Desktop/Work/Classes/Python/PHYS229/Acoustics/Data'
filename += "/AcousticSlottedSphereDataP3.csv"
x_label = 'Theta'
x_units = 'Degrees'
y_label = 'Change in Frequency'
y_units = 'Hz'
Data = np.loadtxt(filename, delimiter=',', comments='#', usecols=(0,1))

guess = [-2.2, np.pi/180, 0.06, -0.2]
sigma = 0.002

################################################################################

x_guess = np.linspace(min(Data[:, 0]),max(Data[:, 0]),500)
y_guess = Periodic(x_guess, *guess)

plt.errorbar(Data[:, 0], Data[:, 1], yerr=sigma, marker='.', linestyle='',
             label="measured data")
plt.plot(x_guess, y_guess,marker="", linestyle="-", linewidth=1, color="g",
         label="initial guess")
plt.xlabel('{} [{}]'.format(x_label, x_units))
plt.ylabel('{} [{}]'.format(y_label, y_units))
plt.title(r'Comparison between the data and the intial guess')
plt.legend(loc='best', numpoints=1)
print ('\nDisplaying plot 1')
plt.show()

################################################################################

params, cov = cf(Periodic, Data[:, 0], Data[:, 1],
                 sigma=sigma*np.ones(len(Data[:, 0])), p0=guess, maxfev=10**5)

chi2 = chi_square(params, Data[:, 0], Data[:, 1], sigma)
dof = len(Data[:, 0]) - len(params)
print ("\nGoodness of fit - chi square measure:")
print ("Chi2 = {}, Chi2/dof = {}\n".format(chi2, chi2/dof))

cov = cov*dof/chi2
paramserr = np.sqrt(np.diag(cov))

param_names = ['f0','omega', 'phase','resonance']

print ("Fit parameters:")
for i in range(len(params)):
    print ('{} = {:.3e} +/- {:.3e}'.format(param_names[i], params[i],
                                           paramserr[i]))
```

CurveFitSlottedSpherePerturbation.py(4.3 KB)

Ryan Kaufmann Feb 08, 2018 @01:25 PM PST

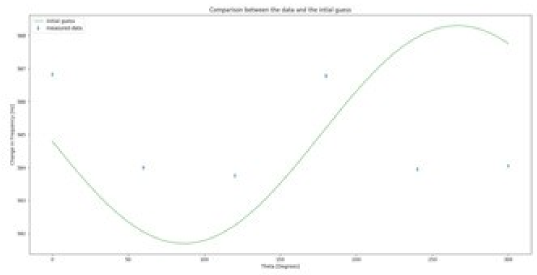
We then fit both the full data set and the data set up to 180 degrees:

Ryan Kaufmann Feb 08, 2018 @03:19 PM PST

$$Full: \chi^2 = 1.529$$
$$f_0 = -1.933 \pm 0.046$$
$$\omega = 0.03546 \pm 0.00024$$
$$\phi = -1.692 \pm 0.050$$
$$f_r = 584.9 \pm 0.03$$

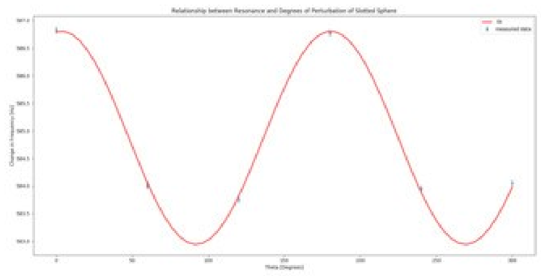
$$Half: \chi^2 = 3.26 * 10^{-24}$$
$$f_0 = -1.913$$
$$\omega = -0.03304$$
$$\phi = 4.622$$
$$f_r = 584.9$$

Ryan Kaufmann Feb 08, 2018 @03:23 PM PST



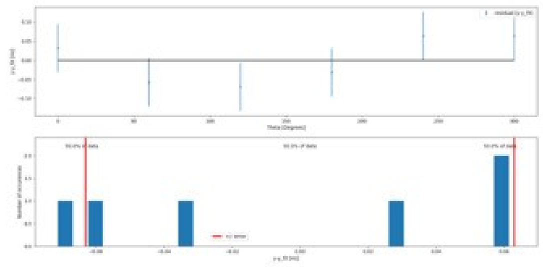
SlottedSphereInitial.png(60.5 KB)

Ryan Kaufmann Feb 08, 2018 @03:23 PM PST



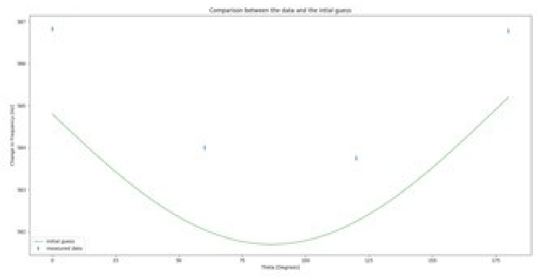
SlottedSphereFitted.png(74.2 KB)

Ryan Kaufmann Feb 08, 2018 @03:23 PM PST



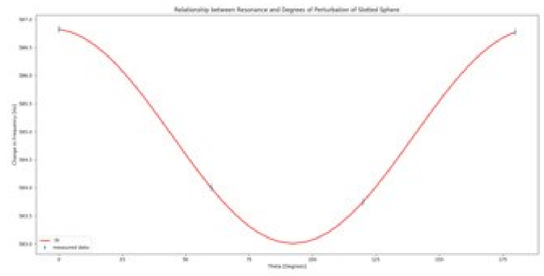
SlottedSphereResids.png(36.9 KB)

Ryan Kaufmann Feb 08, 2018 @03:23 PM PST



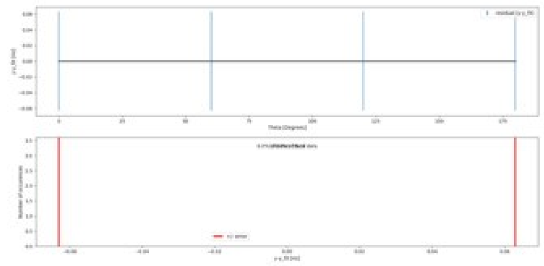
HalfSlottedSphereInitial.png(57.6 KB)

Ryan Kaufmann Feb 08, 2018 @03:23 PM PST



HalfSlottedSphereFitted.png(66.9 KB)

Ryan Kaufmann Feb 08, 2018 @03:23 PM PST



HalfSlottedSphereResids.png(41.8 KB)

Ryan Kaufmann Feb 08, 2018 @03:10 PM PST

Thus our equation for the half the sphere becomes:

Ryan Kaufmann Feb 08, 2018 @03:21 PM PST

$$f(\theta) = f_0 \sin(\omega\theta + \phi) + f_r$$

$$f(\theta) = -1.913 \sin(-0.03304 * \theta + 4.622) + 584.9$$

Ryan Kaufmann Feb 08, 2018 @03:08 PM PST

While the residuals and the chi-squared for the half may look good, there is not enough points to justify this conclusion. In fact, the number of points is equal to the number of parameters, thus no conclusion can be made. Otherwise, we notice what we expected. The slotted sphere acts like a solid sphere when turned 90 degrees, and produces a small to none perturbation when it is 0 degrees.

This agrees with our predictions and makes sense. As the slots are parallel with the direction the wave propagates, it can pass through the body of the sphere easier than if it were a solid sphere. Next in our improvements, we want to take more data points to see how closely the behavior matches our predictions, and get more accurate numbers for our parameters

Ryan Kaufmann Jan 25, 2018 @04:13 PM PST

Acoustics 5: Improvements

Ryan Kaufmann Feb 08, 2018 @04:36 PM PST

We re-performed our perturbation analysis by taking more data points then refitting our data. For the solid sphere, we resolved to take twice as many data points. After re-configuring our resonator to find its resonant frequency, we continued our data set by measuring points 3 cm above each of our previous points. For the slotted sphere, we continued to take frequencies at values in between the ones we had taken, at 30 degree intervals.

Then for the solid sphere, we get the following data:

Ryan Kaufmann Feb 08, 2018 @04:54 PM PST

DataSolidSphereP4.csv(800 Bytes)

Ryan Kaufmann Feb 08, 2018 @04:54 PM PST

AcousticSolidSphereDataP4.csv(225 Bytes)

Ryan Kaufmann Feb 08, 2018 @04:59 PM PST

And our slotted sphere has the following data:

Ryan Kaufmann Feb 08, 2018 @05:07 PM PST

DataSlottedSphereP5.csv(508 Bytes)

Ryan Kaufmann Feb 08, 2018 @05:07 PM PST

AcousticSlottedSphereDataP5.csv(148 Bytes)

Ryan Kaufmann Feb 08, 2018 @05:34 PM PST

Then using the above script, we get the following:

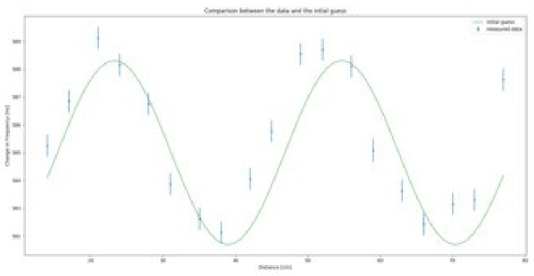
Ryan Kaufmann Feb 08, 2018 @06:14 PM PST

$$Solid: \chi^2 = 1.48$$
$$f_0 = -3.347 \pm 0.135$$
$$\omega = 0.2084 \pm 0.0018$$
$$\phi = 0.1622 \pm 0.0911$$
$$f_r = 585.50 \pm 0.09$$
$$f(x) = -3.347 \sin(0.2084 * x + 0.1622) + 585.5$$

$$SlottedFull: \chi^2 = 4.23$$
$$f_0 = -2.279 \pm 0.155$$
$$\omega = 0.03595 \pm 0.00074$$
$$\phi = -1.717 \pm 0.148$$
$$f_r = 584.6 \pm 0.1$$

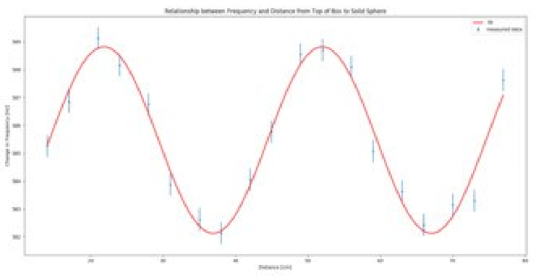
$$SlottedHalf: \chi^2 = 2.62$$
$$f_0 = 1.890 \pm 0.080$$
$$\omega = 0.03616 \pm 0.00151$$
$$\phi = -11.22 \pm 0.41$$
$$f_r = 584.90 \pm 0.09$$
$$f(\theta) = 1.890 \sin(0.03616\theta - 11.22) + 584.9$$

Ryan Kaufmann Feb 08, 2018 @06:35 PM PST



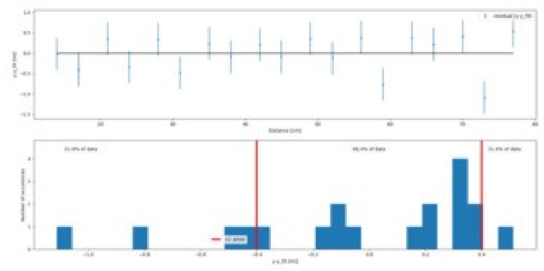
SolidSphereRedoInitial.png(75.6 KB)

Ryan Kaufmann Feb 08, 2018 @06:35 PM PST



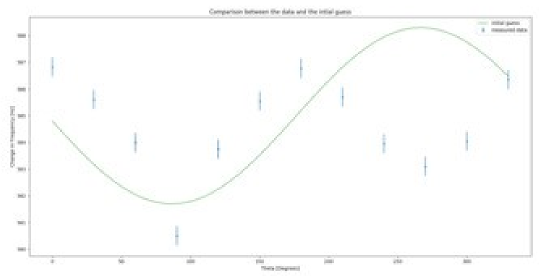
SolidSphereRedoFitted.png(76.5 KB)

Ryan Kaufmann Feb 08, 2018 @06:35 PM PST



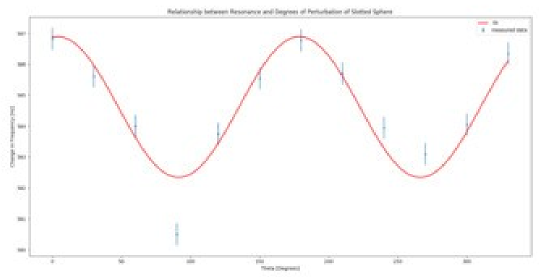
SolidSphereRedoResids.png(36.7 KB)

Ryan Kaufmann Feb 08, 2018 @06:35 PM PST



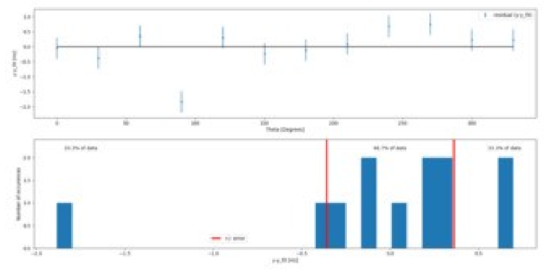
SlottedSphereRedoInitial.png(61.8 KB)

Ryan Kaufmann Feb 08, 2018 @06:35 PM PST



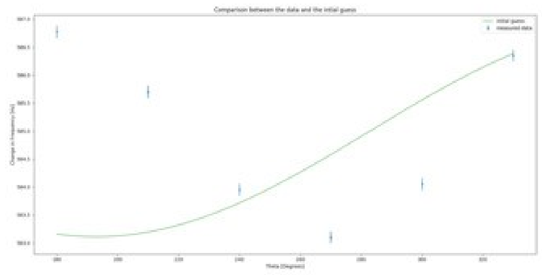
SlottedSphereRedoFitted.png(67.2 KB)

Ryan Kaufmann Feb 08, 2018 @06:35 PM PST



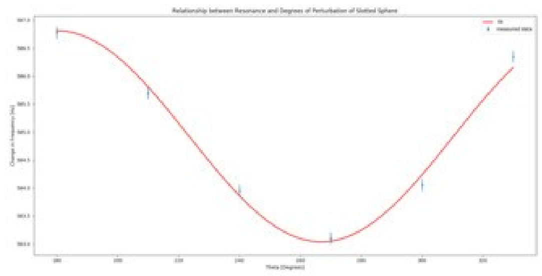
SlottedSphereRedoResids.png(37.5 KB)

Ryan Kaufmann Feb 08, 2018 @06:35 PM PST



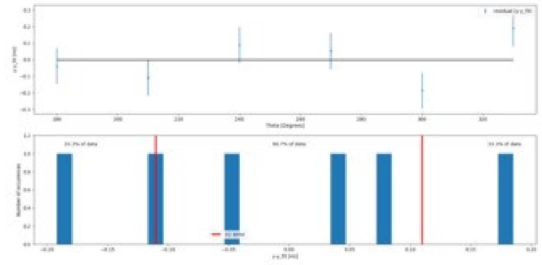
HalfSlottedSphereRedoInitial.png(56.5 KB)

Ryan Kaufmann Feb 08, 2018 @06:35 PM PST



HalfSlottedSphereRedoFitted.png(66.1 KB)

Ryan Kaufmann Feb 08, 2018 @06:35 PM PST



HalfSlottedSphereRedoResids.png(39.6 KB)

Ryan Kaufmann Feb 08, 2018 @07:25 PM PST

Although the chi-squareds aren't better in this set of data, our residuals don't show as many patterns. Furthermore, our predictions of each has continued in our improvements. All of our parameters seem close between our improvements and our first set.

Ryan Kaufmann Feb 08, 2018 @06:42 PM PST

Acoustics 6: Conclusion

Ryan Kaufmann Feb 08, 2018 @07:38 PM PST

This experiment has been aimed at analyzing a resonator and how it is affected by different perturbations. The two perturbations include a vertically positioned solid glass sphere and a rotated slotted sphere. The first part of the experiment gave us the experience with the software we are using and an understanding of the placements of the resonant frequencies. The second part focused on how the solid sphere causes perturbations as a function of depth, while the third part focused on how the slotted sphere causes perturbations as a function of angle. Lastly, we took more data points on the topics of the last two sections to try and get improved data.

While most of the first section was about getting used to our equipment and generally exploring the laboratory set up, there was one part that we sent our to confirm. We took many data points near resonant frequencies that we found as the largest volume the microphone picked up. Using these points, we wanted to see if the area around resonant frequencies resembled a Lorentzian curve. We found that all four of our resonant frequencies fit the Lorentzians fairly well. Thus we concluded the the Lorentzian curve fit the expression of volume in mV as a function of frequency.

In our second and fourth part, we look at perturbations of a solid sphere. We wanted to analyze how the resonant frequency behaves as a function of the depth of the sphere. We proposed that the displacement of the resonant frequency would be best categorized as a sine function of the depth of the sphere. We found that this was reasonable and our data points were fit well with the curve. When we expanded into more points, this did not change. Many of the values we received were very similar between the two trials. Overall, we concluded that our model was accurate.

Finally, in parts three and four, we looked at perturbations of a slotted sphere. We again wanted to analyze the behavior of the resonant frequency, but now as a function of angle of the slotted sphere. We believed that when the slots were in parallel with the axis, the resonant frequency would not change from the empty resonator very much. Meanwhile, as the sphere rotated, it would begin to act more and more like a solid sphere rather than a slotted one. We also expected to see a symmetry about 180 degrees. While the symmetry was very apparent between both trials, the shape of the data points wasn't as much. We settled on a sine curve to describe the half of the data. We found that this worked the best between all values. Furthermore, the data seems to agree between the two sections as well, concluding for us that as the sphere turns, the change in frequency obeys a sinusoidal pattern.