FYS2150
Lab Report: Elasticity

Nicholas Karlsen

April 12, 2018

Abstract

Determining the Youngs' modulus of a brass rod by deflection and listening for the root frequency.

1 Introduction

2 Theory

2.1 Euler-Bernoulli beam theory

From the Euler-Bernoulli beam theory, it follows that the deflection of a beam supported by two points of distance l and a load mg halfway between the two knives is given by

$$h(m) = \frac{mgl^3}{48EI} \tag{1}$$

Where E denotes the Young's modulus of the beam and I the second moment area given by

$$I = \frac{\pi d^4}{4 \cdot 2^4} \tag{2}$$

Where d denotes the diameter of the beam.
Further, Eqn. 1 can be rewritten in the following way

$$E = \frac{4l^3 g}{3\pi |A| d^4} \tag{3}$$

Where $A \equiv h(m)/m$, which can be obtained as the gradient from a linear fit on the data gathered when varying the load subjected and on the beam and recording the resulting deflection. Which gives the following relationship

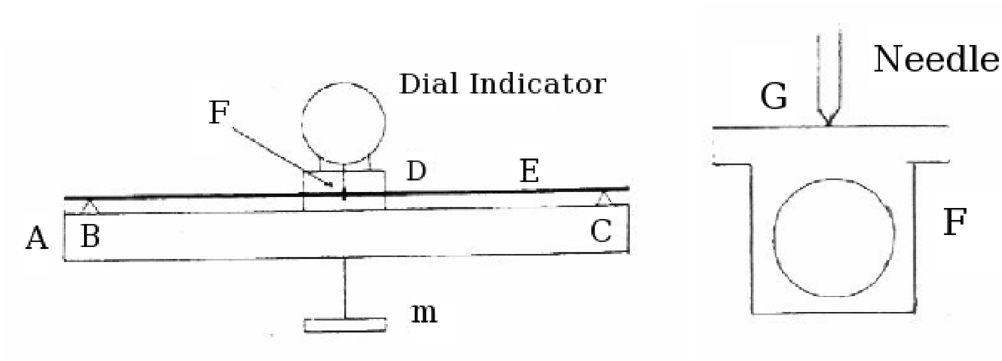
$$h(m) = Am + B \tag{4}$$

2.2 Errors

When performing arithmetic operations on recorded data, the uncertainty in the data must also carry over to the derived results. How these uncertainties are propagated in different operations can be found in Practical Physics [1].

3 Experimental Procedure

3.1 Three-point flexural test



(a) (b)

Figure 1: (a) shows the apparatus used for measuring the deflection of a rod and (b) a cross section of the apparatus at point F.

Using Fig. 1a as a reference; The brass rod, A, was laid on the "knives" B and C, in order to ensure that that the dial gauge, manufactured by Baker[1], was positioned halfway between B and C, we measured the distance from the dial gauge to B and C using a measuring tape of type Lufkin pee wee 2m Y612CM with uncertainty ±0.1cm, adjusting the rod such that the difference in measurements would be sufficiently small. At the middle of the rod, there was a ring attached, as shown in Fig. 1b. The flat surface of the ring was in contact with the needle of the dial gauge at G. In order to ensure that the flat surface of the ring was at right angle with the needle, we turned the rod such that the reading of the dial gauge would be at a minimum, as the skewer the surface, the greater the reading. This process was repeated at the start of every attempt of the experiment.

After having prepared the apparatus, three masses of roughly 0.5g, 1kg and 2kg which we denoted m_a, m_b and m_c respectively. They were placed carefully in the tray denoted

[1] I did not take note of the model number of the particular dial gauge that was used during the lab. While working on this report, i have become aware that each Baker dial gauge is individually calibrated. Therefore, i have no values for the instrumental error in the deflection measurements.

m in Fig. 1a, in different combinations so that we would get readings for the deflection of the rod at $\approx \{0.5, 1, 1.5, 2, 2.5, 3, 3.5\}$kg, recorded by reading the dial gauge.

Due to seemingly disturbing the system significantly when adding masses, we were worried that there might be a significant systematic error in the experiment. So we opted to repeat the readings in this experiment several times in order to investigate if the data in the later readings (when the system had been disturbed multiple times in succession) had an increase in its deviation.

Lastly, the distance between the knives, $l_{B,C}$, was measured using the measuring tape and a micrometer of type Moore & Wright 1965 MI with uncertainty $\pm$0.01mm. The measuring tape was used to measure the distance between the outer edges of the "knives" at B and C, $l_{B\,outer}$, $l_{C\,outer}$. The micrometer was then used to measure the knives thickness l_{knife}, which was needed as the contact points between the rod and the knives are (assumed) at the middle of the identical knives. Since there are two contact points, $l_{B\,outer,C\,outer}$ - $l_{knife} = l_{B,C}$

3.2 Measuring the speed of sound in the rod

The brass rod, with a ring attached to it (same as before), was laid to rest on the flat side of the ring on a solid surface such that the rod is held up by the ring, and nothing else. We also made sure that the rod was not to be disturbed in any way while it was vibrating. When hit with a hammer, it will emit a sound consisting of different frequencies. Following are the two different methods we used for determining the root frequency of the rod. During both experiments, we ensured there were no significant noise pollution during our recording (By which i mean people performing the same experiment as us).

3.2.1 By hearing for beats

A speaker was connected to a signal generator. We started the signal generator at 1200Hz and hit the brass rod with a plastic hammer on the the flat surface on one end of the rod. By ear, there was an audible beat due to the superposition of the two signals. We adjusted the signal generator such that the the frequency of the beat was minimized, and there was essentially no audible difference between the two signals. We did this by trying above and below where we thought the root frequency was, eventually zeroing in on a value.

3.2.2 By Fourier transform

A USB microphone was placed close to the rod, and faced towards it. The microphone was connected to a computer running matlab, with a script that collects audio data from it and Fourier transforms it using FFT. The recordings made were made with a sampling frequency of 8×1024 Hz and varying durations. As before, we hit the rod using a plastic hammer and recorded the data. A total of 7 recordings were made.

3.3 Other measurements

3.3.1 Mass

In order to accurately measure the mass of the rough loads and the rod, the balance scale (Ohaus triple beam balance) which we used had to be calibrated. We did this by weighing a set of three reference weight on the scale, and comparing their measured value to the measured value of the rough loads and the rod using a linear fit. When placing the masses on the scale, we made sure to position the masses in the center of the scale plate and not take a reading until the needle of the balance scale was not sufficiently stable.

3.3.2 Length and thickness of the rod

The length of the rod was measured using the measuring tape, and the thickness using the micrometer. In order to accurately determine the thickness, accounting for any irregularities in the rod due to deformation etc. The thickness was measured several times in different places on the rod, so that we could calculate the mean thickness.

4 Results

4.1 Length and mass measurements

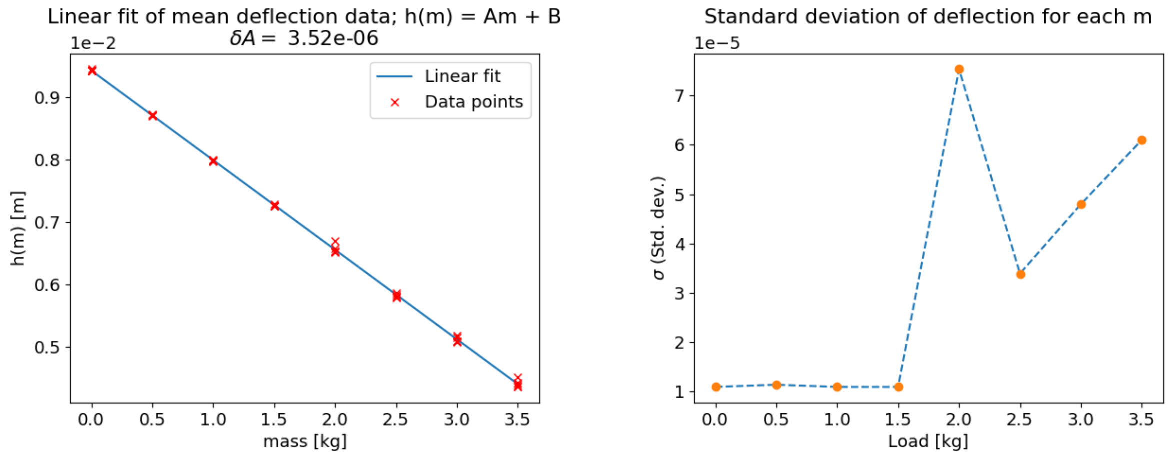
Table 1: Mass of rough load and reference

Stated mass	Measured reference	Measured rough
500g	500.0g	500.1g
1000g	999.9g	1000.3g
2000g	2000.1g	2000.5g

4.2 Results from Three-point flexural test

Table 2: Deflection of rod

Attempt no.	h(0kg) [mm]	h(0.5kg) [mm]	h(1kg) [mm]	h(1.5kg) [mm]	h(2.0kg) [mm]	h(2.5kg) [mm]	h(3.0kg) [mm]	h(3.5kg) [mm]
1	9.44	8.72	8.00	7.28	6.58	5.84	5.15	4.43
2	9.42	8.70	7.98	7.26	6.53	5.80	5.09	4.39
3	9.42	8.71	7.98	7.26	6.53	5.80	5.09	4.37
4	9.41	8.69	7.97	7.25	6.52	5.79	5.08	4.36
5	9.42	8.70	7.98	7.26	6.70	5.87	5.19	4.51



(a) Deflection of rod (b) Standard deviation of data

Figure 2: (a) Shows the deflection of the brass rod measured by the dial gauge. (b) Shows the standard deviation of the data points in (a) at their respective masses

Table 2 contains the deflection data recorded with the dial gauge where the loads listed are from the rough, uncalibrated masses. Their corrected value is listed in table 1.

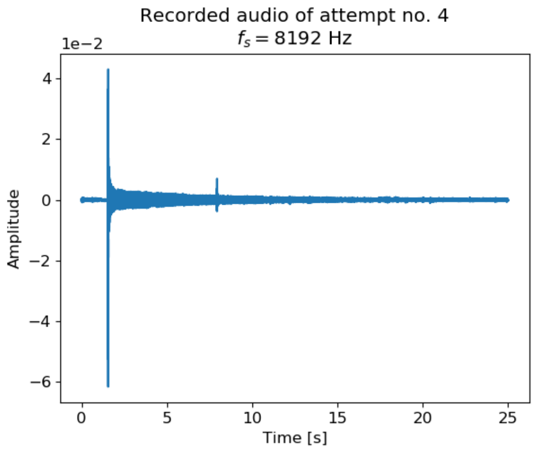
Fig. 2a contains all the recorded data, as well as a linear fit on the mean deflection for each load using corrected values for the mass, m. The error of the linear fit, $h(m) = Am + B$, $dA = 3.52e - 06$. Fig. 2b contains the standard deviation of the deflection values for each load.

From the stated data and their given uncertainties, using Eqn. 3 as well as summing the error, the Young's modulus determined by deflection is as follows

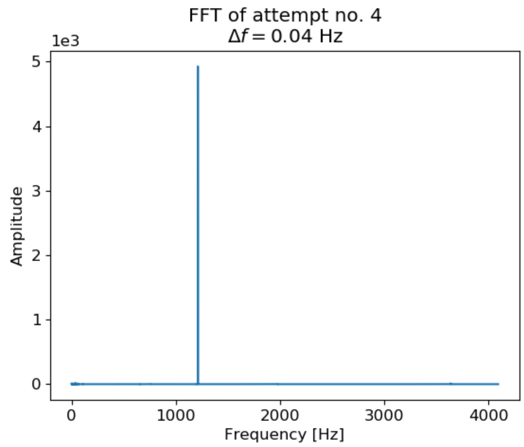
$$E_{deflection} = 105.8 \pm 0.4\% \, \text{GPa} \tag{5}$$

4.3 Results from measuring the speed of sound in the rod

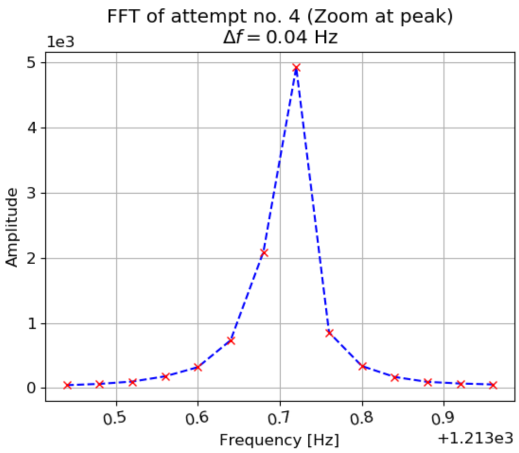
When hearing for beats, me and my lab-partner decided that the root frequency was ≈ 1240 Hz.



(a) Time domain



(b) Frequency domain



(c) Zoomed frequency domain

Figure 3: All of the plots generated for attempt no. 4

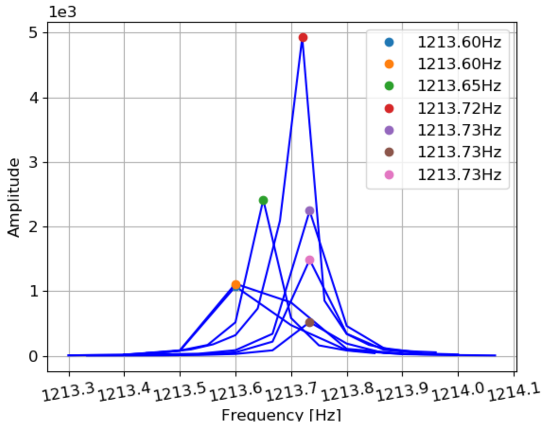


Figure 4: Zoomed frequency plot for all 7 attempts.

Fig. 3 contains the data and derived results from our fourth attempt of the experiment. We performed a total of 7 attempts, all of which yielded in similar results to attempt no. 4. The data yielded from all of the attempts is summarized in Fig. 4 which shows the peaks in the frequency domain in one plot. Table 3 contains all of the relevant numbers related to each attempt, where f denotes the root frequency, Δf the resolution of the frequency domain, t the time of the recording and f_s the sampling frequency.]

Table 3: FFT data

Attempt no.	f [Hz]	Δf [Hz]	t [s]	f_s [Hz]
1	1213.60	0.10	10	8192
2	1213.60	0.10	10	8192
3	1213.65	0.05	20	8192
4	1213.72	0.04	25	8192
5	1213.72	0.04	25	8192
6	1213.72	0.07	15	8192
7	1213.73	0.07	15	8192

Using the root frequency gathered from attempt 4 and 5 (which are identical), the Young's modulus, using Eqn. **??** is

$$E_{sound} = 103.7 \pm 0.2\% \,\mathrm{GPa} \tag{6}$$

5 Discussion

Note STD.DEV of deflection increases with m (system is disturbed). Assume the disturbance is normally distributed, therefore error -> given by STD. DEV.

6 Conclusion

References

[1] G. L. Squires. *Practical Physics 4th Edition.* Cambridge University Press, 2001.

*

A Code

All of the code used to produce this report. Anything noteworthy should already be mentioned in the main body of the report. Note that when this code was written, readability was not a huge concern, so some of it may not be very easy to interpret.

scripts/FFTlyd.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Generates the same figures as FFTlyd.m
author: Nicholas Karlsen
"""
import scipy.io as sio
import matplotlib.pyplot as plt
import numpy as np


# Sets font size of matplot
plt.rcParams.update({'font.size': 12})


def import_matlab(filename):
    # Opens .mat file
    mfile = sio.loadmat(filename)
    # Fetches data
    data = mfile.get("data")
    energi = mfile.get("energi")
    fut = mfile.get("fut")
    L = mfile.get("L")
    t = mfile.get("t")

    return data, energi, fut, L, t


rel_path = "data/"
```

```python
n = 1
mat_file = "forsok%i.mat" % n


def raw_fig(filename):
    data, energi, fut, L, t = import_matlab(filename)
    plt.plot(t, data)
    plt.xlabel("Time [s]")
    plt.ylabel("Amplitude")
    plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))


raw_fig(rel_path + "forsok1.mat")
plt.title("Recorded audio of attempt no. 1\n$f_s = 8192$ Hz")
plt.savefig("raw_exp2_1.png")
plt.close()

raw_fig(rel_path + "forsok4.mat")
plt.title("Recorded audio of attempt no. 4\n$f_s = 8192$ Hz")
plt.savefig("raw_exp2_4.png")
plt.close()


def figure1(filename):
    data, energi, fut, L, t = import_matlab(filename)
    fut = np.transpose(fut)
    fh = int(len(energi) / 2.0)  # half lenght of data
    # Only plot first half of data, as FF mirrors in half-way point.
    plt.plot(fut[:fh], energi[:fh])
    plt.xlabel("Frequency [Hz]")
    plt.ylabel("Amplitude")
    plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))


figure1(rel_path + "forsok1.mat")
plt.title("FFT of attempt no. 1\n$\Delta f=0.10$ Hz")
plt.savefig("energy_exp2_1.png")
plt.close()

figure1(rel_path + "forsok4.mat")
plt.title("FFT of attempt no. 4\n$\Delta f=0.04$ Hz")
plt.savefig("energy_exp2_4.png")
plt.close()

eigenfreqs = []


def figure2(filename, style="-", cross=0):
    data, energi, fut, L, t = import_matlab(filename)
    fut = np.transpose(fut)

    fh = int(len(energi) / 2.0)  # half lenght of data
    ipeak = np.argmax(energi[:fh])
```

```python

    eigenfreqs.append(fut[ipeak])

    i = ipeak
    while energi[i] > np.amax(energi[:fh]) * 0.01:
        i -= 1

    j = ipeak
    while energi[j] > np.amax(energi[:fh]) * 0.01:
        j += 1

    plt.plot(fut[i:j], energi[i:j], color="blue", linestyle=style)
    if cross == 1:
        plt.plot(fut[i:j], energi[i:j], "rx")
    else:
        plt.plot(fut[ipeak], energi[ipeak], "o", label="%.2fHz" % fut[ipeak])

    plt.grid("on")

figure2(rel_path + "forsok1.mat", style="--", cross=1)
plt.xlabel("Frequency [Hz]")
plt.ylabel("Amplitude")
plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
plt.xticks(rotation=10)
plt.title("FFT of attempt no. 1 (Zoom at peak)\n$\Delta f=0.10$ Hz")
plt.savefig("freq_exp2_1.png")
plt.close()


figure2(rel_path + "forsok4.mat", style="--", cross=1)
plt.xlabel("Frequency [Hz]")
plt.ylabel("Amplitude")
plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
plt.xticks(rotation=10)
plt.title("FFT of attempt no. 4 (Zoom at peak)\n$\Delta f=0.04$ Hz")
plt.savefig("freq_exp2_4.png")
plt.close()


for i in range(1, 8):
    figure2(rel_path + "forsok%i.mat" % i)

plt.xlabel("Frequency [Hz]")
plt.ylabel("Amplitude")
plt.legend()
plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
plt.xticks(rotation=10)
plt.savefig("freq_exp2_all.png")
plt.close()
```

scripts/FYS2150lib.py

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
A collection of commonly used functions in FYS2150.
author: Nicholas Karlsen
"""
import numpy as np


def stddev(x):
    """
    Finds the standard deviation, and standard deviation of
    a 1D array of data x.
    See. Eqn D. Page 24 squires
    """
    n = len(x)
    sigma = np.sqrt((np.sum(x**2) - 1.0 / n * np.sum(x)**2) / (n - 1))
    sigma_m = np.sqrt((np.sum(x**2) - 1.0 / n * np.sum(x)**2) / (n * (n - 1)))

    return sigma, sigma_m


def linfit(x, y):
    """
    Finds the line of best-fit in the form y=mx+c given two
    1D arrays x and y.
    """
    n = np.size(y)
    D = np.sum(x**2) - (1.0 / n) * np.sum(x)**2
    E = np.sum(x * y) - (1.0 / n) * np.sum(x) * np.sum(y)
    F = np.sum(y**2) - (1.0 / n) * np.sum(y)**2

    dm = np.sqrt(1.0 / (n - 2) * (D * F - E**2) / D**2)
    dc = np.sqrt(1.0 / (n - 2) * (float(D) / n + np.mean(x)) *
                 ((D * F - E**2) / (D**2)))
    m = float(E) / D
    c = np.mean(y) - m * np.mean(x)

    return m, c, dm, dc
```

scripts/lab_data.py

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Contains all of the data collected in the
Elacticity lab, module 2 of FYS2150
author: Nicholas Karlsen
"""

from pylab import *
import scipy.constants as const
```

```python
import FYS2150lib as fys


rcParams.update({'font.size': 13})  # Sets font size of plots

def weight_data(set=1):
    "set decides which data set the function returns."
    set = set.lower()   # Forces lowercase
    sets = ["masses", "rod"]
    # Mass of weights measured with balance
    m_a_balance = 500.1e-3
    m_b_balance = 1000.3e-3
    m_c_balance = 2000.5e-3

    # Mass of reference weights
    m_reference = array([0.5, 1.0, 2.0])
    m_reference_balance = array([500.0e-3, 999.9e-3, 2000.1e-3])  # Weighed

    # Using linear fit to correct for error in balance
    a, b, da, db = fys.linfit(m_reference, m_reference_balance)
    # Corrected masses
    m_a = (m_a_balance - b) / a     # approx 500g
    m_b = (m_b_balance - b) / a     # approx 1000g
    m_c = (m_c_balance - b) / a     # approx 2000g

    if set == sets[0]:  # Return corrected masses
        return m_a, m_b, m_c

    m_rod_ring = np.array([2482.7, 2482.5, 2482.1]) * 1e-3
    m_ring = 34.4 * 1e-3    #kg
    m_rod = (mean(m_rod_ring) - m_ring - b) / a  #kg


    if set == sets[1]:
        return m_rod

    if set not in sets:
        print "Invalid set"
        print "List of valid sets:", sets
        print "exiting..."
        exit()


def E_sound(f, L, d, M):
    '''
    Returns youngs modulus given
    f = root frequency
    L = lenght between knives
    d = diameter of rod
    M = mass of rod
    '''
    return (16.0 * M * L * f**2) / (np.pi * d**2)

```

```

def E_sound_error(E, sd, sf, sL, sM, d, f, L, M):
    return E * np.sqrt((2 * sd / d)**2 + (2 * sf / f)**2 +
                       (2 * sL / L)**2 + (2 * sM / M)**2)


d = np.array([15.98, 15.99, 15.99, 16.00,
              15.99, 15.99, 15.98, 15.99,
              15.99, 15.99]) * 1e-3
d_mean = np.mean(d)
d_err = np.sqrt(fys.stddev(d)[1]**2 + (0.01e-3)**2) # Std dev of mean + instrumentation error

f_root = 1213.72
f_err = 0.04  # resolution of FFT
M_err = 9.8974331835e-05  # from linfit above (da)

l_rod = 144.4e-2     # m
l_rod_err = 0.1e-2

E_sound = E_sound(f=f_root,
                  L=l_rod,
                  d=d_mean,
                  M=weight_data("rod"))

print "E from root f = %e" % E_sound

E_sound_err = E_sound_error(E=E_sound,
                            sd=d_err,
                            sf=f_err,
                            sL=l_rod_err,
                            sM=M_err,
                            d=d_mean,
                            f=f_root,
                            L=l_rod,
                            M=weight_data("rod"))
print "E_err root = %e" % E_sound_err

print "error percentage = %.3f percent" % ((E_sound_err / E_sound) * 100)


def experiment1_data():
    m_a, m_b, m_c = weight_data("masses")
    mass_dat = array(
        [0, m_a, m_b, m_a + m_b, m_c, m_a + m_c,
         m_b + m_c, m_a + m_b + m_c])                # [Kg]

    # Round 1: (in order)
    h_1 = array([9.44, 8.72, 8.00, 7.28, 6.58, 5.84, 5.15, 4.43]) * 1e-3  # [m]
    # Round 2: (in order)
    h_2 = array([9.42, 8.70, 7.98, 7.26, 6.53, 5.80, 5.09, 4.39]) * 1e-3  # [m]
```

```
    # Round 3: (in order)
    h_3 = array([9.42, 8.71, 7.98, 7.26, 6.53, 5.80, 5.09, 4.37]) * 1e-3  #
     [m]
    # Round 4: (in order)
    h_4 = array([9.41, 8.69, 7.97, 7.25, 6.52, 5.79, 5.08, 4.36]) * 1e-3  #
     [m]
    # Round 5: (in order)
    h_5 = array([9.42, 8.70, 7.98, 7.26, 6.70, 5.87, 5.19, 4.51]) * 1e-3  #
     [m]

    h_mean = (h_1 + h_2 + h_3 + h_4 + h_5) / 5.0

    A, B, dA, dB = fys.linfit(mass_dat, h_mean)

    mass = linspace(0, 3.5, 8)
    h_mass = A * mass + B  # h(m)


    def plotdata():
        h_sets = [h_1, h_2, h_3, h_4, h_5]
        plot(mass, h_mass, label="Linear fit")
        # errorbar(mass, m * mass + c, yerr=dm, color='blue', fmt='o',
    label='Error Range')

        for dat in h_sets:
            plot(mass_dat, dat, "x", color="r")
        plot(NaN, NaN, "xr", label="Data points")
        xlabel("mass [kg]")
        ylabel("h(m) [m]")
        ticklabel_format(style='sci', axis='y', scilimits=(0,0))
        legend()
        title("Linear fit of mean deflection data; h(m) = Am + B\n$\delta A
     =$ %.2e" % dA)
        savefig("figs/h_m_fig.png")
        close()
    plotdata()

    def plot_stddev():
        """Plots the standard deviation of h(m)
        as m is increased"""
        deviation = np.zeros(len(h_1))
        for i in xrange(len(h_1)):
            deviation[i] = fys.stddev(array([h_1[i],
                                             h_2[i],
                                             h_3[i],
                                             h_4[i],
                                             h_5[i]]))[0]
        plot(mass_dat, deviation, linestyle="--")
        plot(mass_dat, deviation, "o")
        ticklabel_format(style='sci', axis='y', scilimits=(0,0))
        title("Standard deviation of deflection for each m\n")
        xlabel("Load [kg]")
        ylabel("$\sigma$ (Std. dev.)")
```

```
        savefig("figs/h_m_deviation.png")
        close()
    plot_stddev()


    l_BC_outer = 133.9 * 1e-2
    l_knife_diameter = 4.09 * 1e-3
    l_BC = l_BC_outer - l_knife_diameter
    s_l_BC = np.sqrt((0.1e-2)**2 + (0.01e-3)**2)


    E = (4.0 * l_BC**3 * const.g / (3 * pi * abs(A) * d_mean**4))
    print "\nE from deflection = %e"%E
    S_E = E * np.sqrt((dA / A)**2 + (4.0 * d_err / d_mean)**2 +(3.0 * s_l_BC / l_BC)**2)
    print "error in deflection E = %e" % S_E

    print "percentage error in deflection = %.3f percent" %(100 * S_E / E)

def experiment_2():
    ''' Data pertaining to the audio exp.'''


experiment1_data()
```