FYS2150
Lab Report: Elasticity

Nicholas Karlsen
(Dated: April 1, 2018)

A study on two different methods to determine the Young's modulus of a brass rod.

I. INTRODUCTION

II. THEORY

A. Euler-Bernoulli beam theory

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

$$E = \frac{4l^3 g}{3\pi |A| d^4} \tag{2}$$

[1]

B. Errors

[2]

III. EXPERIMENTAL PROCEDURE

A. Three-point flexural test

B. Measuring the speed of sound in the rod

IV. RESULTS

V. DISCUSSION

VI. CONCLUSION

[1] Wikipedia contributors. Eulerbernoulli beam theory — wikipedia, the free encyclopedia, 2018. [Online; accessed 31-March-2018].

[2] G. L. Squires. *Practical Physics 4th Edition.* Cambridge University Press, 2008.

CODE

All of the code used to produce this report. Anything noteworthy should already be mentioned in the main body of the report.

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
import FYS2150lib as fys

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

    if set == sets[1]:
        return

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


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
```

```

    m, c, dm, dc = fys.linfit(mass_dat, h_mean)

    mass = linspace(0, 3.5, 8)
    h_mass = m * mass + c  # 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", label="data points")
        xlabel("mass [kg]")
        ylabel("h(m) [m]")
        plt.legend()
        show()
    plotdata()

    # lengde mellom yttersidene til festepunktene til knivene
    # PEE WEE 2m Y612CM LUFKIN +- 0.01cm
    l_AB = 133.9 * 1e-2  # [m]
    # diameter til festepunkter
    # Moore & Wright 1965 MI +- 0.01mm
    l_AB_diameter = 4.09 * 1e-3  # [mm]
    # anta festepunktet er p  midtden s  trekk fra diameter totalt sett
    l = l_AB - l_AB_diameter

    #M linger av stangens diameter d p  forskjellige punkter
    # Moore & Wright 1965 MI +- 0.01mm
    d = array([15.98, 15.99, 15.99, 16.00, 15.99, 15.99, 15.98, 15.99, 15.99, 15.99]) * 1e-3  # [m]
    d_m = mean(d); #m

    A = abs((h_mass - c) / mass)

    E = mean(4.0 * l**3 * const.g / (3 * pi * A * d_m**4)[1:-1])
    print E



if __name__ == "__main__":
    experiment1_data()
```

scripts/FYS2150lib.py

```
#!/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
```