Phys 205 Lab Report #1

September 9, 2020

1 Data and Error Analysis

Name: Ibrahim Alshehri

ID: 201745170

1.1 Objectives

Review various topics related to data and error analysis such as error propagation, uncertainties, mean, standard deviation, proper graphs and different scales.

[1]:
```
%matplotlib inline

import sympy as sp # for symbolic expressions
import numpy as np # for numerical processing
import pprint      # for good looking printing



from IPython.display import set_matplotlib_formats, display, Math
set_matplotlib_formats('png', 'pdf')

## change the font used for graphs to latex
from matplotlib import rc
rc('text', usetex=True)


import matplotlib.pyplot as plt # for plotting graphs

## print sympy statements using latex
sp.init_printing()
```

1.2 Exercies

---

[2]:
```
# exercise #1
values1 = {
    "mi" : 14.2,      # mass of ice (g)
    "mw" : 72.3,       # mass of water (g)
    "T1" : 25.4,       # initial temperature of water (C)
    "T2" : 7.8         # final temperature of water (C)
}

dmi = 0.1      # uncertainty in mass of ice (g)
dmw = 0.1      # uncertainty in mass of water (g)
dT1 = 0.1      # uncertainty in initial temperature of water (C)
dT2 = 0.1      # uncertainty in final temperature of water (C)

# solving for L
L, mi, mw, T1, T2 = sp.symbols('L mi mw T1 T2')
L = sp.solveset(mi*L + mi*T2 - mw * (T1 - T2), L).args[0]
sp.Eq(sp.Symbol('L'), L).simplify()
```

[2]:

$$L = \frac{-T_2 mi + mw\,(T_1 - T_2)}{mi}$$

[3]:
```
subs_list1 = [(mi, values1["mi"]), (mw, values1["mw"]),
            (T1, values1["T1"]), (T2, values1["T2"])]
values1["L"] = L.subs(subs_list1)
```

[4]:
```
dL = (
    abs(L.diff(mi).subs(subs_list1)) * dmi
    + abs(L.diff(mw).subs(subs_list1)) * dmw
    + abs(L.diff(T1).subs(subs_list1)) * dT1
    + abs(L.diff(T2).subs(subs_list1)) * dT2
)
```

[5]:
```
print(f"L = {values1['L']:4.2f} (cal/g)")
```

```
L = 81.81 (cal/g)
```

[26]:
```
print(f"absolute uncertainty dL = {dL:4.2f} (cal/g)")
```

```
absolute uncertainty dL = 1.87 (cal/g)
```

[27]:
```
print(f"percentage uncertainty dL/L = {(dL/values1['L'] * 100):4.2f}%")
```

```
percentage uncertainty dL/L = 2.29%
```

---

[8]:
```
# exercise 2
# first we convert angels to radians
values2 = {
    "A": 60 * sp.pi/180,            # rads
    "D": (23 + 14/60) * sp.pi/180   # rads
}
```

```

dA = 2/60 * sp.pi / 180;    # rads
dD = 2/60 * sp.pi / 180;    # rads
```

[9]:
```
n, A, D = sp.symbols('n A D')
n = (sp.sin((A+D)/2) / sp.sin(A/2))
n
```

[9]:

$$\frac{\sin\left(\frac{A}{2} + \frac{D}{2}\right)}{\sin\left(\frac{A}{2}\right)}$$

[10]:
```
subs_list2 = [(A, values2['A']), (D, values2['D'])]
values2['n'] = sp.N(n.subs(subs_list2))
```

[11]:
```
dn = (
    abs(n.diff(A).subs(subs_list2)) * dA
    + abs(n.diff(D).subs(subs_list2)) * dD
)
```

[12]:
```
print(f"The values are\n\tn = "
      f"{int(values2['n'] * 180 / np.pi)}\u00B0"
      f"{int(((values2['n'] * 180 / np.pi) % 1) * 60)}'"
      f"\n\tdn = {int(sp.N(dn)*180/sp.N(sp.pi)*60)}'"
      f"{int(((sp.N(dn)*180/sp.N(sp.pi)*60) % 1 ) * 60)}\"")
```

```
The values are
        n = 76°6'
        dn = 2'18"
```

---

[13]:
```
# exercise 3
# we will define two numpy arrays to hold the values of the
# tensile forces and wire lengths
T_vals = np.array([2., 4., 6., 8., 10., 12., 14.]) # Nt
l_vals = np.array([0.39, 0.52, 0.64, 0.73, 0.85, 0.93, 1.00]) # m
f_val = 100 # Hz
dT = 0.01 # Nt
dl = 0.05 # m
```

[14]:
```
f, l, T, mu = sp.symbols('f ell T mu')
equ = sp.Eq(f, 1/(2*l) * sp.sqrt(T/mu))
equ
```

[14]:

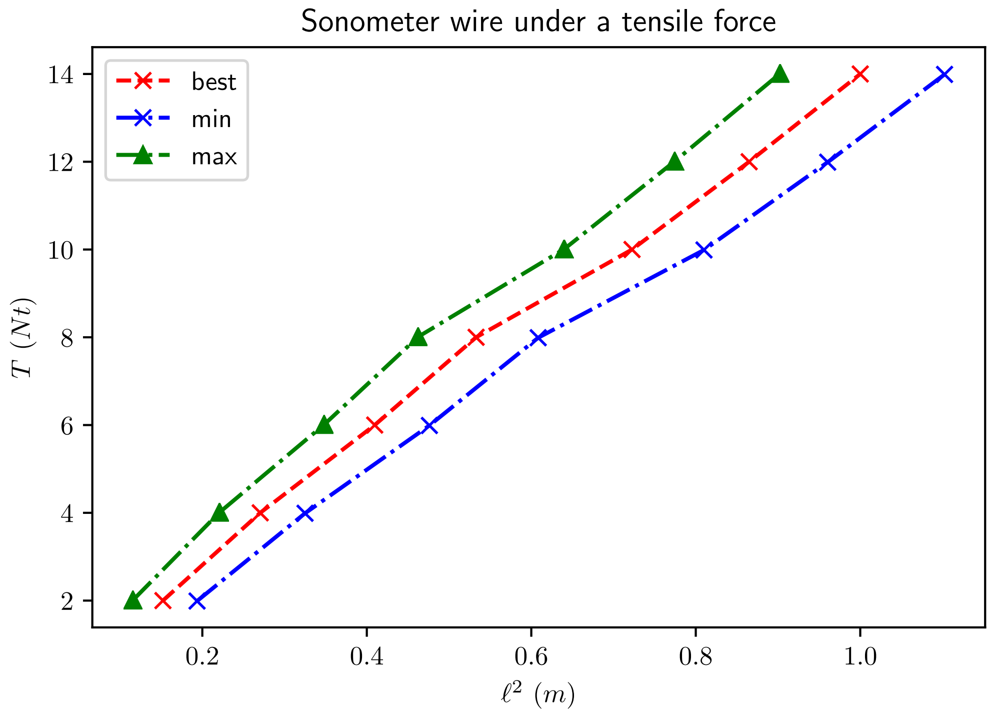
$$f = \frac{\sqrt{\frac{T}{\mu}}}{2\ell}$$

[15]:
```
(sp.Eq(T, sp.solve(equ, T)[0]),
sp.Eq(l, sp.solve(equ, l)[0]),
sp.Eq(mu, sp.solve(equ, mu)[0]))
```

[15]:

$$\left(T = 4\ell^2 f^2 \mu,\ \ \ell = \frac{\sqrt{\frac{T}{\mu}}}{2f},\ \ \mu = \frac{T}{4\ell^2 f^2}\right)$$

[16]:
```
plt.plot(l_vals**2, T_vals, '--xr', mfc='none', label='best')
plt.plot((l_vals + dl)**2, T_vals - dT, '-.bx', label='min')
plt.plot((l_vals - dl)**2, T_vals + dT, '-.g^', label='max')
plt.xlabel('$\\ell^2$ $(m)$')
plt.ylabel('$T$ $(N t)$')
plt.title('Sonometer wire under a tensile force')
plt.legend()
plt.show()
```



Notice that the $slope = 4f^2\mu \implies \mu = \frac{slope}{4f^2}$.

[17]:
```
slope_best, _ = np.polyfit(l_vals ** 2, T_vals, 1)

slope_min, _ = np.polyfit((l_vals + dl)**2,
                          T_vals - dT, 1)
```

```
slope_max, _ = np.polyfit((l_vals - dl)**2,
                            T_vals + dT, 1)
```

[18]:
```
mu_vals = np.array([slope_best, slope_min, slope_max])/(4 * f_val**2)
```

[19]:
```
print(f"Best slope = {slope_best:5.3f},"
      f"\nMaximum Slope = {slope_max:5.3f},"
      f"\nMinimum Slope = {slope_min:5.3f}\n"
      f"\u03BC = {mu_vals.mean()} (Kg/m),"
      f"\n\u03B4\u03BC = {mu_vals.std()} (Kg/m)")
```

```
Best slope = 13.797,
Maximum Slope = 14.835,
Minimum Slope = 12.894
  = 0.0003460515543009 42 (Kg/m),
   = 1.982450419495519e-05 (Kg/m)
```

---

[20]:
```
# exercise 4
V_vals = np.array([5., 10., 15., 20., 30., 40., 60.]) # volts
I_vals = np.array([6, 16, 33, 47, 85, 148, 270]) # milliamps

dV_vals = V_vals * 0.1 # volts
dI_vals = np.array([1, 2, 3, 4, 5, 10, 20.0]) # milliamps
```

We have $I = AV^n$, taking the natural logarithm of both sides we get

$$\log I = \log A + n \log V$$

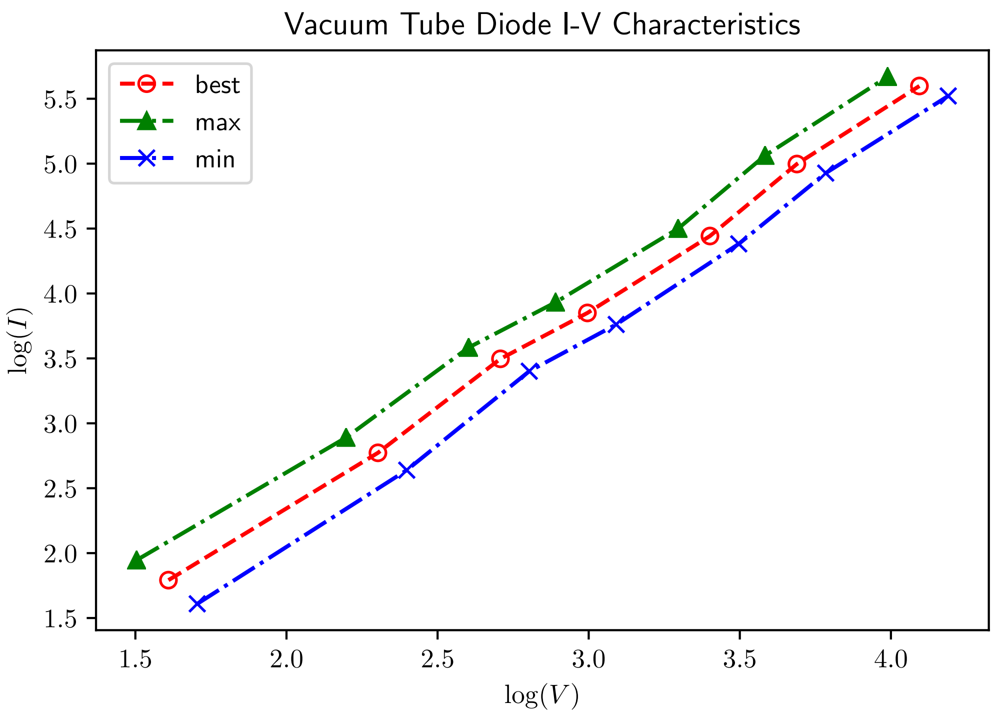
.

[21]:
```
plt.plot(np.log(V_vals), np.log(I_vals),
         '--ro', mfc='none', label='best')

plt.plot(np.log(V_vals - dV_vals),
         np.log(I_vals + dI_vals), '-.g^', label='max')

plt.plot(np.log(V_vals + dV_vals),
         np.log(I_vals - dI_vals), '-.bx', label='min')

plt.title('Vacuum Tube Diode I-V Characteristics')
plt.ylabel('$\\log(I)$')
plt.xlabel('$\\log(V)$')
plt.legend()
plt.show()
```



[22]:
```
slope, intercept = np.polyfit(np.log(V_vals), np.log(I_vals), 1)
# slope = n
# intercept = log(A)

slope_min, intercept_min = np.polyfit(np.log(V_vals + dV_vals),
                            np.log(I_vals - dI_vals), 1)

slope_max, intercept_max = np.polyfit(np.log(V_vals - dV_vals),
                            np.log(I_vals + dI_vals), 1)

n_mean = 1/3 * (slope_min + slope + slope_max)
dn = np.sqrt(((slope_min - n_mean)**2 + (slope - n_mean)**2 + (slope_max -␣
 ↪n_mean)**2)/2)

A = np.exp(intercept)
A_mean = 1/3 * np.sum(np.exp([intercept, intercept_min, intercept_max]))
dA = np.sqrt((np.sum(np.square(np.exp([intercept, intercept_min,␣
 ↪intercept_max]) - A_mean)))/2)

(
    sp.Eq(sp.Symbol('n'), round(slope, 3)),
    sp.Eq(sp.Symbol('A'), round(A, 3)),
    sp.Eq(sp.Symbol('\\bar n'), round(n_mean, 3)),
    sp.Eq(sp.Symbol('\\bar A'), round(A_mean, 3)),
    sp.Eq(sp.Symbol('\\delta n'), round(dn, 3)),
    sp.Eq(sp.Symbol('\\delta A'), round(dA, 3))
)
```

[22]:

$$\left(n = 1.537, \quad A = 0.489, \quad \bar{n} = 1.54, \quad \bar{A} = 0.506, \quad \delta n = 0.041, \quad \delta A = 0.184\right)$$

1.2.1 Comments regarding δA.

It is difficult to determine the uncertainty in δA due to the relativly high uncertainty in V^n. Also once we take the $\log$ of both sides and fit the graph to find the value $\log A$, we would have also a small error term ϵ in the value we get from fitting so that the intercept will be equal to $\log A \ + \ \epsilon$. But to get the value of δA we will have to take the exponential of the intercept so that we have $A \pm \delta A = \exp(\log A + \epsilon) = Ae^{\epsilon}$. Hence if the value of ϵ is far from zero the error δA will become large quickly.

---

[23]:
```
# exercise 5
a = np.array([0.085, 0.087, 0.085, 0.086, 0.085,
              0.087, 0.086, 0.085, 0.086, 0.085])
a_mean = np.sum(a)/len(a)
a_rms  = np.sqrt(np.mean(np.square(a)))
# in numpy the square method is applied element wise
a_error = np.sqrt(np.sum(np.square(a - a.mean()))/(len(a) - 1))
# we can use a.std() to calculate the error, but it will use
# len(a) in the denominator instead of len(a) - 1
```

[24]:
```
print(f"The average is {a_mean:.4f}\n"
      f"the r.m.s is {a_rms:.4f}\n"
      f"the error is {a_error:.6f}")
```

```
The average is 0.0857
the r.m.s is 0.0857
the error is 0.000823
```

---

2 Conclusion

So far we have analysed the errors that appear in different data sets using means and standard deviations. We've seen how errors in our measurements and in our data could propagate to our results. And we have plotted different graphs to aid us in visualizing and understanding the relationships between the datasets we have in our hands.