Pylab2

All the data and codes are stored on https://github.com/travelwithwind/PHY224

Experiment design

1. Connect the ameter, voltmeter, and power supply to the resistor
2. Vary the voltage on the power supply.
3. Record the voltage and current from the multimeters, along with uncertainty.
4. Change the voltage, and record the new values.
5. Save the data in two columns: the first column being the independent variable (voltage).
6. After performing all the above measurements, disconnect the power, and switch the voltmeter to resistance. This will serve as a reference value to compare against.

Units

Voltage: V

Current: mA

Data collection

Sample size

For resistor (99.1 ohm): 15

For potentiometer (34.6 ohm):10

The resistance is measured by multimeter.

Uncertainty measures

The uncertainty is chosen to be the greater of the error of accuracy and error of precision.

For voltage:

DC Voltage Accuracy ±(0.25% of reading)

Error of precision is 0.01V

For current:

DC Current Accuracy ±(0.75% of reading)

Error of precision is 0.1mA

Regression Method

minimization of chi-squared

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - y(x_i)}{\sigma_i} \right)^2$$

where $y(x_i) = ax + b$

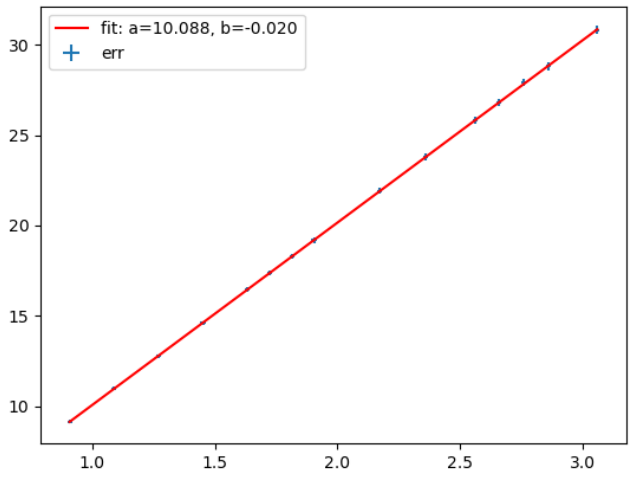
Result

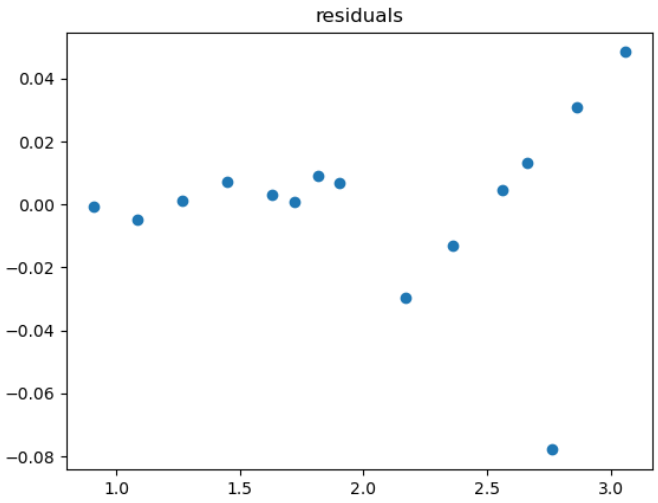
For curve fitting on resistor data,

$$popt = [10.088, -0.020]$$

$$\text{p\_cov} = \begin{bmatrix} 0.004, -0.006 \\ -0.006, 0.011 \end{bmatrix}$$

Below is the plot based on resistor data.





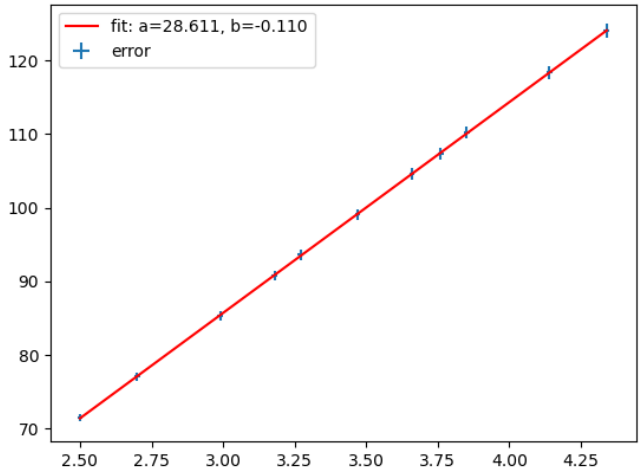
The fit looks great. But the residuals are concerning, Residuals seem to be positively correlated. However, the sample size is too small to say anything definitely.

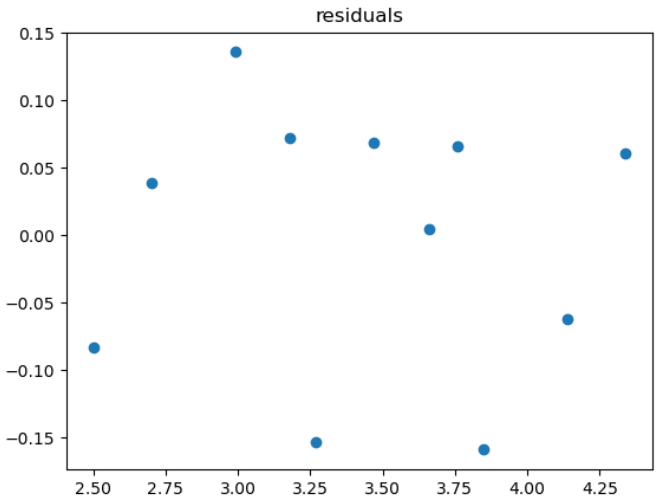
For curve fitting on potentiometer data,

$$popt = [28.611, -0.110]$$

$$\text{p\_cov} = \begin{bmatrix} 0.149, -0.486 \\ -0.486, 1.626 \end{bmatrix}$$

Here are the plots based on potentiometer data:





The residuals plot on this data set is much more pleasing as it displays no particular pattern.

Q1

Linear fit does not pass through zero.

The regression on resistor data has Intercept is -0.2

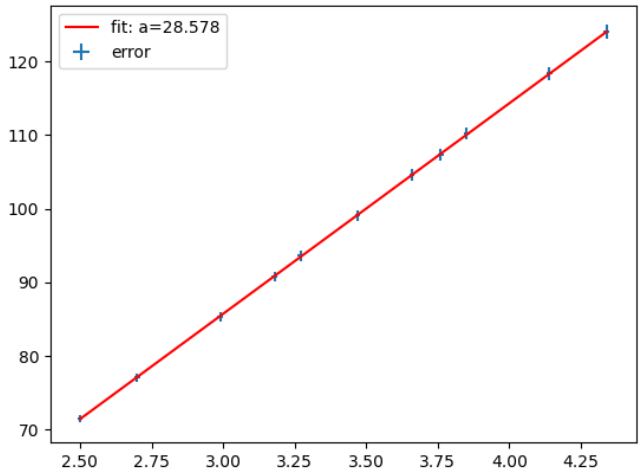
The regression on potentiometer data has Intercept is -0.11

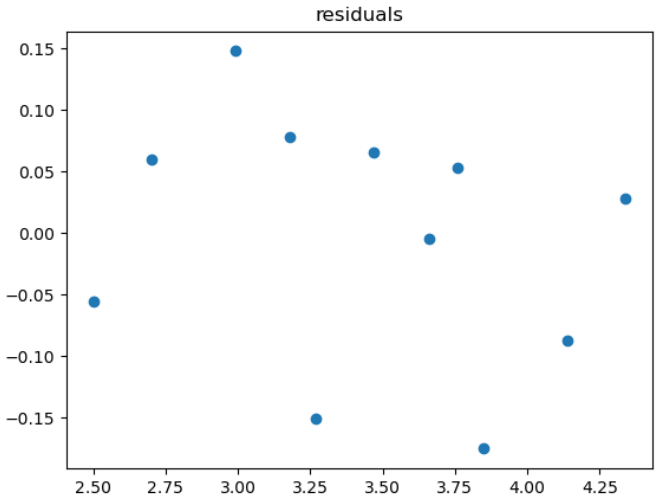
It could happen because the equipments are not perfect. There are accuracy and precision issue associated with it.

Q2

Forcing the intercept to be zero does not affect the result much.

Here are the results based on the potentiometer data:





The slope has changed only a tiny bit. The impact on resistor data curve fitting is similar, so I will omit it here.

I thought the slope could become unstable with larger standard error. But the variance of the slope is 0.0042 with no intercept. With intercept is the covariance matrix is:

$$\begin{bmatrix} 0.149, -0.486 \\ -0.485, \ 1.626 \end{bmatrix}$$

Q3

Resistance measured by multimeter is 34.6 ohm for potentiometer and 99.1 ohm for resistor.

Now we will calculate resistance based on $curve\_fit()$ results.

For resistor

The slope a $= 0.088$. Resistance implied is $\frac{1000}{a} = \frac{1000}{10.088} = 99.1$ ohm. We need 1000 on the numerator because current is measured in mA. The uncertainty is $\sqrt{\frac{1000}{a^2} var(a)} = \sqrt{\frac{1000}{10.088^2} 0.004} = 0.2$. Thus, resistance is 99.1 ± 0.2 ohm

For potentiometer

The slope a $= 0.088$. Resistance implied is $\frac{1000}{a} = \frac{1000}{28.611} = 35.0$ ohm. The uncertainty is $\sqrt{\frac{1000}{a^2} var(a)} = \sqrt{\frac{1000}{28.611^2} 0.149} = 0.4$. Thus, resistance is 35.0 ± 0.4 ohm.

Q4

Reduced chi-squared on the resistor and potentiometer data are 0.0200 and 0.0215 respectively.

This suggests that they both have very good fit.

Codes

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

xdata, ydata = np.loadtxt("ohm data.txt", unpack=True, delimiter="\t", skiprows=1)


accuracy_x = xdata * 0.0025  # DC Voltage Accuracy ±(0.25% of reading)
precision_x = 0.01  # Error of precision is 0.01V
sigma_x = np.maximum(accuracy_x, precision_x)

accuracy_y = ydata * 0.0075  # DC Current Accuracy ±(0.75% of reading)
precision_y = 0.1  # Error of precision is 0.1mA
sigma_y = np.maximum(accuracy_y, precision_y)
```

```
# Q1
def model_function(x, a, b):
    return a * x + b

p_opt, p_cov = curve_fit(model_function, xdata, ydata, (1, 0), sigma_y, True)

# plt.scatter(xdata,ydata, label="data")
fig, ax = plt.subplots()
ax.plot(xdata, model_function(xdata, *p_opt), 'r-',
        label='fit: a=%5.3f, b=%5.3f' % tuple(p_opt))
ax.errorbar(xdata, ydata, xerr=sigma_x, yerr=sigma_y, linestyle='None',
label="error")
ax.legend()
plt.show()

fig, ax = plt.subplots()
ax.set_title('residuals')
ax.scatter(xdata, model_function(xdata, *p_opt) - ydata)
plt.show()

# Q2

def model_function2(x, a):
    return a * x

p_opt, p_cov = curve_fit(model_function2, xdata, ydata, (1), sigma_y, True)

# plt.scatter(xdata,ydata, label="data")
fig, ax = plt.subplots()
ax.plot(xdata, model_function2(xdata, *p_opt), 'r-',
        label='fit: a=%5.3f' % tuple(p_opt))
ax.errorbar(xdata, ydata, xerr=sigma_x, yerr=sigma_y, linestyle='None',
label="error")
ax.legend()
plt.show()

fig, ax = plt.subplots()
ax.set_title('residuals')
ax.scatter(xdata, model_function2(xdata, *p_opt) - ydata)
plt.show()


# Q4

def chi2(xdata, ydata, measurement_error, model_function):
    p_opt, p_cov = curve_fit(model_function, xdata, ydata, (1, 0), measurement_error,
True)
    n = len(p_opt)
    N = len(xdata)
    v = N - n
    expected = model_function(xdata, *p_opt)
    statistics = sum(((ydata - expected) / measurement_error) ** 2) / v
    return statistics
```

```
chi2(xdata, ydata, sigma_y, model_function)
```