PyLab - Ohm and Power laws

Fredrik Dahl Bråten, Pankaj Patil

September 23, 2021

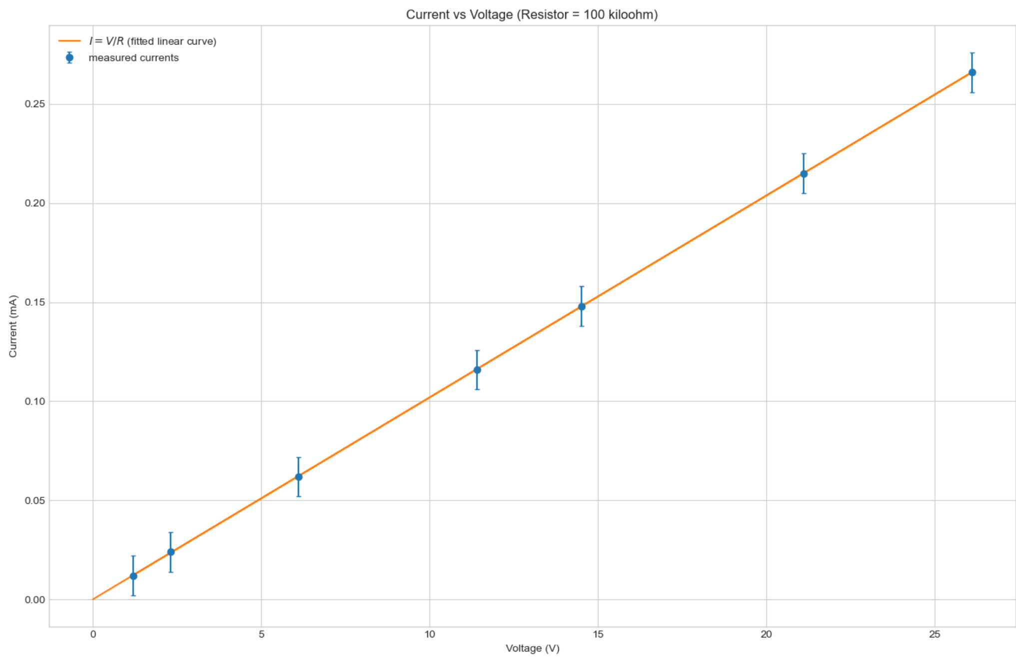
Exercise 1: Introduction to fitting methods

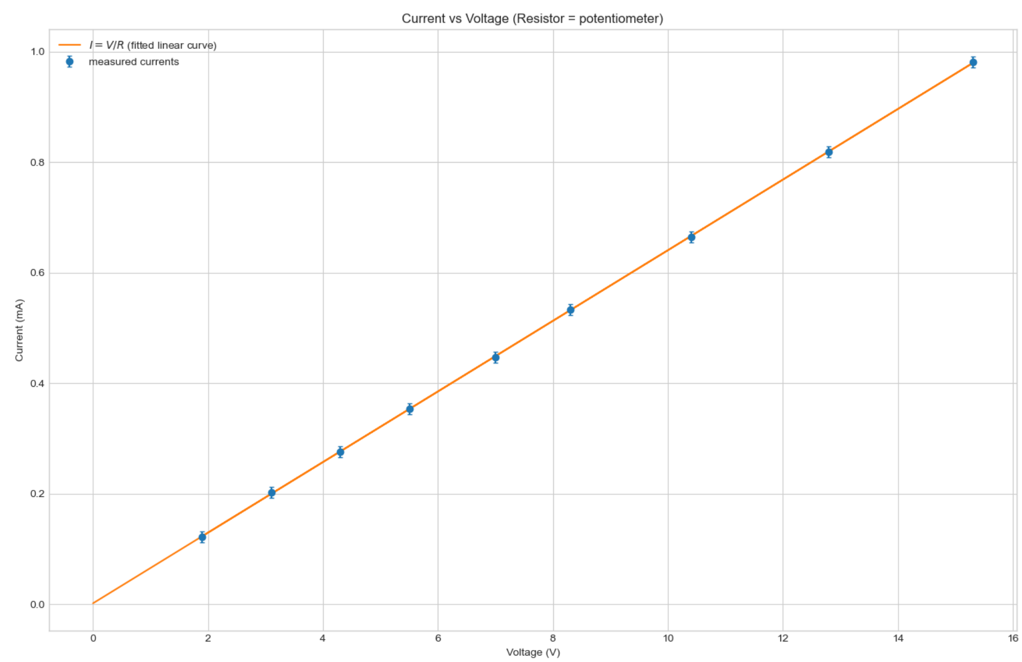
1 Procedures

Voltage (V)	Current (mA)
1.2	0.012
2.3	0.024
6.1	0.062
11.4	0.116
14.5	0.148
21.1	0.215
26.1	0.266

Voltage (V)	Current (mA)
1.9	0.122
3.1	0.202
4.3	0.276
5.5	0.354
7	0.447
8.3	0.533
10.4	0.665
12.8	0.818
15.3	0.981

2 Analysis


Current vs Voltage (Resistor = 100 kiloohm)
$I = V/R$ (fitted linear curve)
measured currents
Current (mA)
0.00
0.05
0.10
0.15
0.20
0.25
0
5
10
15
20
25
Voltage (V)


Current vs Voltage (Resistor = potentiometer)
$I = V/R$ (fitted linear curve)
measured currents
Current (mA)
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
10
12
14
16
Voltage (V)

3 Conclusions

Exercise 3: Nonlinear fitting methods II

Appendix

Python Code: Exercise 1

```
import numpy as np
import scipy.optimize as optim
import matplotlib.pyplot as plt

def chi2(y_measure,y_predict,errors):
    """Calculate the chi squared value given a measurement with errors and
    prediction"""
    return np.sum( np.power(y_measure - y_predict, 2) / np.power(errors, 2) )

def chi2reduced(y_measure, y_predict, errors, number_of_parameters):
    """Calculate the reduced chi squared value given a measurement with errors
    and prediction, and knowing the number of parameters in the model."""
    return chi2(y_measure, y_predict, errors)/(y_measure.size - number_of_parameters)

# we have constant voltage uncertainity which is 0.1 V
voltage_uncertainity = 0.1

def current_uncertainity(current):
    """return the uncertainity in current for given values of current"""
    if current > 100:
        return 1
    elif current > 10:
        return 0.1
    else:
        return 0.01

#  model function
def linear_model_function(x, a, b):
    return a*x + b

def analyse_file(filename, title):
    # load the csv file as txt
    measured_voltages, measured_currents = np.loadtxt(filename,
                                                    skiprows=1,
                                                    usecols=(0,1),
                                                    delimiter=",",
                                                    unpack=True)

    # create error array for the voltage
```

```
voltage_errors = np.ones_like(measured_voltages) * voltage_uncertainity

# create error array for the current
current_errors = np.vectorize(current_uncertainity)(measured_currents)

# do the curve fitting
popt, pcov = optim.curve_fit(linear_model_function,
                             measured_voltages,
                             measured_currents,
                             absolute_sigma=True,
                             sigma=current_errors)

pvar = np.diag(pcov)

# new figure for this file
plt.figure(figsize=(16, 10))
plt.style.use("seaborn-whitegrid")

# plot the error bar chart
plt.errorbar(measured_voltages,
            measured_currents,
            yerr=current_errors,
            marker="o",
            label="measured currents",
            capsize=2,
            ls="")

# plot the fitted curve
# add 0 to the measured data set
measured_voltages_with_0 = np.append(measured_voltages, 0)
plt.plot(measured_voltages_with_0,
        linear_model_function(measured_voltages_with_0, popt[0], popt[1]),
        label='$I = V/R$ (fitted linear curve)')

# legend and title
plt.title("Current vs Voltage (Resistor = %s)" % title)
plt.xlabel("Voltage (V)")
plt.ylabel("Current (mA)")
plt.legend(loc="upper left")
plt.savefig("lab_1_ex_1_plot_%s.png" % filename[:-4].lower())

chi2r = chi2reduced(measured_currents,
                    linear_model_function(measured_voltages,
                                          popt[0],
                                          popt[1]),
```

```
                        current_errors,
                        2)

    print("filename %s" % filename)
    print("\tlinear fit model gives a=%.2f, b=%.2f" % (popt[0], popt[1]))
    print("\tfitted (average) resistance = %.3f kiloohm" % (1/popt[0]))
    print("\terror in fitted resistance = %.5f kiloohm" % np.sqrt(pvar[0]))
    print("\tmodel chi2r = %.3f" % chi2r)

# files to analyse
file_titles ={
    "100k.csv": "100 kiloohm",
    "Potentiometer.csv": "potentiometer"
}

for filename, title in file_titles.items():
    analyse_file(filename, title)
```

Python Code: Exercise 3

```
import numpy as np
import scipy.optimize as optim
import matplotlib.pyplot as plt

def chi2(y_measure,y_predict,errors):
    """Calculate the chi squared value given a measurement with errors and
    prediction"""
    return np.sum( np.power(y_measure - y_predict, 2) / np.power(errors, 2) )

def chi2reduced(y_measure, y_predict, errors, number_of_parameters):
    """Calculate the reduced chi squared value given a measurement with errors
    and prediction, and knowing the number of parameters in the model."""
    return chi2(y_measure, y_predict, errors)/(y_measure.size - number_of_parameters)

# we have constant voltage uncertainity which is 0.1 V
voltage_uncertainity = 0.1

def current_uncertainity(current):
    """return the uncertainity in current for given values of current"""
    if current > 100:
        return 1
    elif current > 10:
        return 0.1
    else:
        return 0.01

#  model function
def linear_model_function(x, a, b):
    return a*x + b

def analyse_file(filename, title):
    # load the csv file as txt
    measured_voltages, measured_currents = np.loadtxt(filename,
                                                       skiprows=1,
                                                       usecols=(0,1),
                                                       delimiter=",",
                                                       unpack=True)

    # create error array for the voltage
    voltage_errors = np.ones_like(measured_voltages) * voltage_uncertainity

    # create error array for the current
    current_errors = np.vectorize(current_uncertainity)(measured_currents)
```

```
# do the curve fitting
popt, pcov = optim.curve_fit(linear_model_function,
                             measured_voltages,
                             measured_currents,
                             absolute_sigma=True,
                             sigma=current_errors)

pvar = np.diag(pcov)

# new figure for this file
plt.figure(figsize=(16, 10))
plt.style.use("seaborn-whitegrid")

# plot the error bar chart
plt.errorbar(measured_voltages,
             measured_currents,
             yerr=current_errors,
             marker="o",
             label="measured currents",
             capsize=2,
             ls="")

# plot the fitted curve
# add 0 to the measured data set
measured_voltages_with_0 = np.append(measured_voltages, 0)
plt.plot(measured_voltages_with_0,
        linear_model_function(measured_voltages_with_0, popt[0], popt[1]),
        label='$I = V/R$ (fitted linear curve)')

# legend and title
plt.title("Current vs Voltage (Resistor = %s)" % title)
plt.xlabel("Voltage (V)")
plt.ylabel("Current (mA)")
plt.legend(loc="upper left")
plt.savefig("lab_1_ex_1_plot_%s.png" % filename[:-4].lower())

chi2r = chi2reduced(measured_currents,
                    linear_model_function(measured_voltages,
                                          popt[0],
                                          popt[1]),
                    current_errors,
                    2)

print("filename %s" % filename)
```

```
    print("\tlinear fit model gives a=%.2f, b=%.2f" % (popt[0], popt[1]))
    print("\tfitted (average) resistance = %.3f kiloohm" % (1/popt[0]))
    print("\terror in fitted resistance = %.5f kiloohm" % np.sqrt(pvar[0]))
    print("\tmodel chi2r = %.3f" % chi2r)

# files to analyse
file_titles ={
    "100k.csv": "100 kiloohm",
    "Potentiometer.csv": "potentiometer"
}

for filename, title in file_titles.items():
    analyse_file(filename, title)
```

References

[1]