# PyLab - Ohm and Power laws

Fredrik Dahl Bråten, Pankaj Patil

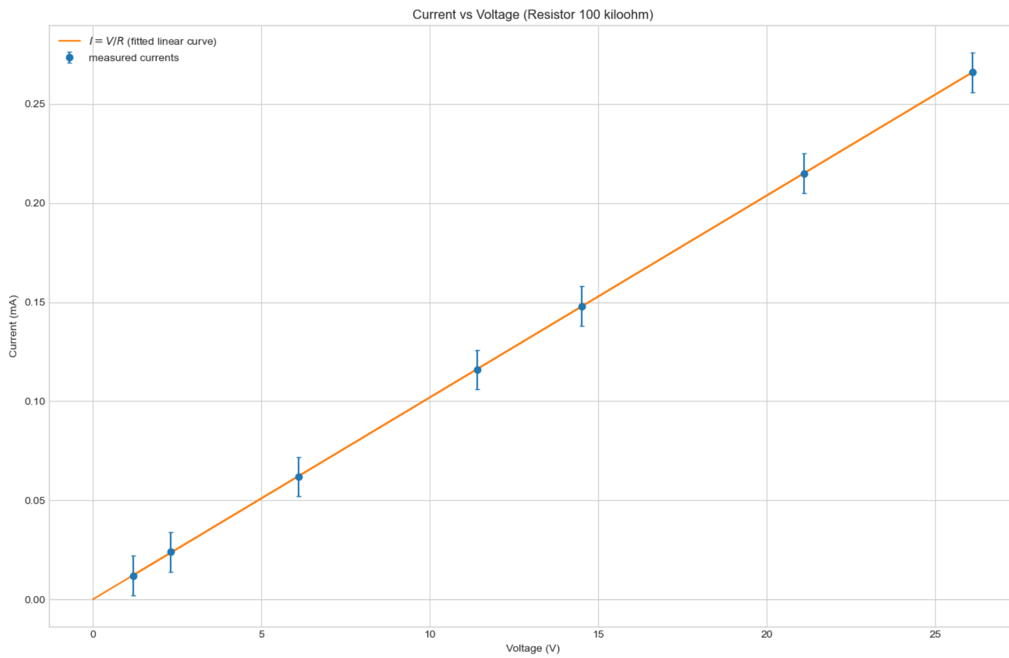
September 23, 2021

## Exercise 1: Introduction to fitting methods

## 1 Procedures

| Voltage (V) | Current (mA) | Resistance (kOhm) |
|---|---|---|
| 1.2 | 0.012 | 100 |
| 2.3 | 0.024 | 95.83333333 |
| 6.1 | 0.062 | 98.38709677 |
| 11.4 | 0.116 | 98.27586207 |
| 14.5 | 0.148 | 97.97297297 |
| 21.1 | 0.215 | 98.13953488 |
| 26.1 | 0.266 | 98.12030075 |

# 2 Analysis




# 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

# filename
filename = "100k.csv"

# 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)

print("linear fit model gives a=%.2f, b=%.2f" % (popt[0], popt[1]))
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 100 kiloohm)")
plt.xlabel("Voltage (V)")
plt.ylabel("Current (mA)")
plt.legend(loc="upper left")
plt.savefig("lab_1_ex_1_plot.png")
```

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

print("fitted (average) resistance = %.3f kiloohm" % (1/popt[0]))
print("error in fitted resistance = %.5f kiloohm" % np.sqrt(pvar[0]))
print("model chi2r = %.3f" % chi2r)
```

## 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

# filename
filename = "100k.csv"

# 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)

print("linear fit model gives a=%.2f, b=%.2f" % (popt[0], popt[1]))
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 100 kiloohm)")
plt.xlabel("Voltage (V)")
plt.ylabel("Current (mA)")
plt.legend(loc="upper left")
plt.savefig("lab_1_ex_1_plot.png")

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

```
print("fitted (average) resistance = %.3f kiloohm" % (1/popt[0]))
print("error in fitted resistance = %.5f kiloohm" % np.sqrt(pvar[0]))
print("model chi2r = %.3f" % chi2r)
```

# References

[1]