

# Introduction to fitting methods

## Ohm's law

The function `curve_fit()` is introduced for fitting linear data. You will perform voltage and current measurements on a resistor to find the resistance via Ohm's law.

This lab contains explicit questions for you to answer labelled with **Q**. However, you still need to maintain a notebook for the lab, make plots and write Python code.

## Background knowledge for exercise 2

**Python** arrays, numpy, pyplot, scipy, `curve_fit()`

**Error Analysis** Chi squared ( $\chi^2$ ), goodness of fit

**Physics** Ohm's law. Review this from your first year text.

## 1 Introduction

In this exercise, we will introduce how to *fit* experimental data to a certain function, and get *quantitative* observations from experiment.

The package “Notes on Error Analysis” can be found at [http://www.physics.utoronto.ca/~phy225h/web-pages/Notes\\_Error.pdf](http://www.physics.utoronto.ca/~phy225h/web-pages/Notes_Error.pdf). It introduces linear and non-linear regression methods, aimed at finding the values of parameters which give the closest agreement between the data and the theoretical model. You should read the section on least-squares fitting.

Suppose our data consisted of  $N$  measurements  $(y_i, x_i)$  with  $i = 1, 2, \dots, N$ . A first step is to plot the data with error bars representing the reading error (or the standard deviation  $\sigma_i$ ) on each point. This gives a visual check for what relation our data has.

Next, assume we know a *fitting function* which is close to the experimental data. The function depends on unknown parameters,  $p_1, p_2, \dots$ . We shall calculate the parameters  $p_i$  so that the function  $\tilde{y}(x_i)$  would be in closest agreement with  $(y_i, x_i)$ . Note that  $\tilde{y}$  is probably not the *true* value of the relation, but rather the *best* value we can determine from the data.

The least-squares method is a *maximum likelihood* estimation using the minimization of chi-squared,

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

The function  $y(x_i)$  that minimizes  $\chi^2$  is the curve of best fit.

In this exercise, we shall look at a *linear fitting method*, which uses the fitting function  $y(x) = ax + b$ , where  $a$  and  $b$  are model parameters to be determined.

We shall not attempt a complete analysis of the goodness of the fit. This will be done in exercise 3.

## 2 Curve fitting in Python

The `scipy` package in Python contains functions for performing a least-squares curve fit. We will use the `curve_fit()` function from the `scipy.optimize` module.

The inputs/arguments to `curve_fit()` are: the model function,  $x$  data,  $y$  data, and an initial guess at the model parameters. The reading errors  $\sigma_i$  can also be included. In linear fitting, the model function is  $f(x; a, b) = ax + b$ , which can be defined in Python as,

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

Note that the dependent variable `x` must be the first argument.

The return values are the optimal parameter values, and the covariance matrix for those parameters. Given `xdata` and `ydata`, the full call for a linear fit to find the parameters would be,

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

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

The slope of the line of best fit is `p_opt[0]`. Passing the value `True` instructs `curve_fit()` to use the absolute magnitudes of `sigma` to calculate covariance; the value `p_opt` is unaffected by it.

There are many keyword arguments to `curve_fit()`, which can be found in the [documentation](#). Commonly, you may use `maxfev` to control the maximum number of function calls. The default value is 200.

### 3 The experiment

We will be testing Ohm's Law – the linear relation between voltage and current in electrical circuits. The voltage is our independent variable, current is the dependent variable, and resistance is the parameter we'd like to measure. There are two parts to the experiment: testing a known resistor, and testing a potentiometer.

#### 3.1 Uncertainty from multimeters

Our later analysis requires an idea of the uncertainty in the measurements. Digital multimeters have two types of uncertainty:

**Error of accuracy** which depends on function (DCV, ACV, DCA, ACA and Ohm ()). This error is provided by the instrument manufacturer. This error is expressed as a percentage of the reading. Please check the link below for complete instrument specifications. The reading can be done on different scales of the same function, which changes the number of digits on display and also the error of accuracy.

**Error of precision** which is due to fluctuations at the level of the last digit. Take one digit (with position on the last digit and value 1) to be the error of precision.

**The uncertainty is taken to be the largest of the two errors above.**

For example, using a multimeter with an error of accuracy of 3%, you read 2.06 mA of the instrument. Error of accuracy is 3% of reading:

$$2.06\text{mA} \times 0.03 = 0.06\text{mA}$$

Error of precision is 0.01 (fluctuations in the last digit). Therefore, by choosing the largest of the two errors, we state that the uncertainty in reading 2.06mA is 0.06mA. The instrument specifications for the multimeter you'll use can be found at <http://www.upscale.utoronto.ca/specs/tegam130a.html>

Be sure to track the uncertainty in measurements from the multimeter. Also, keep in mind that there are other sources of uncertainty that are harder to measure, like the quality of electrical connections.

#### 3.2 The experiment

You will be provided with two multimeters, a board with electrical components (see Figure 1), and enough wires to connect everything. Choose one resistor, and one potentiometer for the experiment.

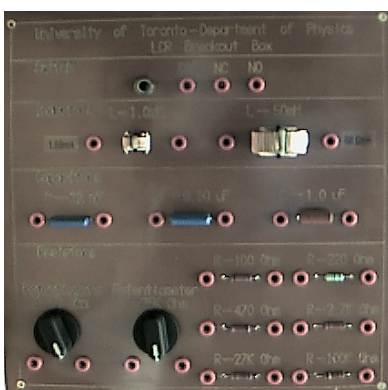


Figure 1: The electrical components board.

For the known resistor, perform the following.

1. Connect the ammeter, voltmeter, and power supply to the resistor (Figure 2). *Ask the TA if you're in doubt*
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 – repeat.
5. Save the data on a memory stick as a text file (extension `.txt`) with 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 give you a reference value to compare against.

Repeat these measurements for the potentiometer, making sure not to move the dial between measurements.

**Resistors are marked with coloured bars to indicate their nominal resistance through a code (there are many calculators for this online). As part of this exercise, you may also compare your results against the nominal resistance  $\pm$  the tolerance.**

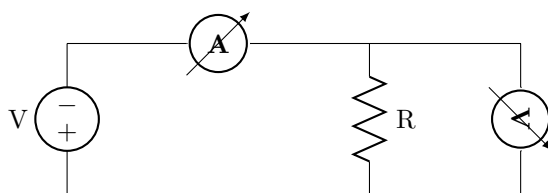


Figure 2: Circuit diagram of the experimental setup.

## 4 Analyzing the data

We will analyze the dependency of the current on the voltage using a linear fitting program. Ohm's law for resistors states

$$I = \frac{1}{R} V. \quad (2)$$

Thus, the resistance of a linear component can be determined by plotting  $I$  vs.  $V$ , and measuring the slope.

There are other electrical components that have nonlinear voltage-current relations, like diodes, transistors, and amplifiers. Using a linear model for these would result in a bad fit. In a later exercise, we will perform a nonlinear regression with a lightbulb.

## 5 Build a linear fitting program

Linear fitting is done via the Levenberg-Marquardt algorithm (an advanced least-squares method), as implemented in the `scipy.optimize` module. Today, you will use the `curve_fit()` function (i.e. `scipy.optimize.curve_fit`). Calculation of the statistics from the output of `curve_fit()` will be done in the next exercise.

Here is the outline of the Python program for Exercise 2a:

- import the necessary modules and functions (e.g. `numpy` and `curve_fit()`)
- Read the data file into an array with the `loadtxt()` function.
- Extract the variables from the array columns.
- Define the model function  $f(x) = ax + b$ .
- Call `curve_fit()` with the function, data, and initial guess for the parameters.
- Output the calculated parameters.
- Create relevant plots.

Write a program to perform a linear fit for the data you collected in Section 3. It should create a plot of current vs. voltage with errorbars and the line of best fit, and output the calculated value of resistance (the slope of the line of best fit). Run this program using the data for the resistor, and again using the data for the potentiometer.

**Q1** In Ohm's law,  $V = IR$ , the line should pass through  $I = V = 0$ . Did your linear fit pass through zero as well? If it didn't, why do you think that happened?

**Q2** What result do you find for resistance if you force the line to pass through zero? (i.e. try the model function  $f(x, a) = ax$ )

**Q3** How does your resistance from using `curve_fit()` compare to the value measured with the multimeter?

## 6 Goodness of fit - reduced chi-squared

Recall that the  $\chi^2$  distribution gives a measure of how unlikely a measurement is. The more a model deviates from the measurements, the higher the value of  $\chi^2$ . But if  $\chi^2$  is too small, it's also an indication of a problem: usually that there weren't enough samples.

This best (most likely) value of  $\chi^2$  depends on the number of degrees of freedom of the data,  $\nu = N - n$ , where  $N$  is the number of observations and  $n$  is the number of parameters. This dependence is removed with the *reduced chi-squared* distribution,

$$\chi_{\text{red}}^2 = \frac{1}{\nu} \sum_{i=1}^N \left( \frac{y_i - y(x_i)}{\sigma_i} \right)^2, \quad (3)$$

where  $y_i$  is the *dependent* data you measured,  $x_i$  is the *independent* data,  $\sigma_i$  is the measurement error in the *dependent* data.

For  $\chi_{\text{red}}^2$ , the best value is 1:

- $\chi_{\text{red}}^2 \gg 1$  indicates that the model is a poor fit;
- $\chi_{\text{red}}^2 > 1$  indicates an incomplete fit or underestimated error variance;
- $\chi_{\text{red}}^2 < 1$  indicates an over-fit model (not enough data, or the model somehow is fitting the noise).

**Q4** Add a function to your program to calculate  $\chi_{\text{red}}^2$ . What values were computed? How would you interpret your results?

## Your submission for this exercise should include:

- Answers to questions 1-4,
- plots of current vs. voltage for each of the resistor and potentiometer,
- the calculated resistance values,
- the final version of your program, and
- the recorded data from the experiment.

**Resistors for electrical circuits are designed to be very linear and reliable, so the three measurements for resistance will likely be very close. The lines may even be indistinguishable on the plot.**