Nonlinear fitting methods II

Nonlinear circuits

We continue with the curve fitting from the previous exercises, but extend it to powerlaw models. A lightbulb demonstrates various power laws; resistance and radioactive power are both dependent on temperature for black bodies. In this lab, we will examine these relations with the Voltage-Current curve for a lightbulb.

Background knowledge for exercise 4

Python lists, arrays, numpy, SciPy, PyPlot, `curve_fit()`
Error Analysis Chi squared (χ^2), goodness of the fit
Physics Ohm's law. Review this from your first year text.

1 Introduction

An incandescent lightbulb works by heating a tungsten filament until it glows. All matter emits electromagnetic radiation, called *thermal radiation*, when it has a temperature above absolute zero. The simple theoretical material, a *blackbody*, is one that absorbs all radiation and follows a strict powerlaw:

$$P = A\sigma T^4 \,, \tag{1}$$

where σ is the Stefan-Boltzmann constant ($5.670373 \times 10^{-8}\,\mathrm{W\,m^{-2}K^{-4}}$).

Of course, real materials are not ideal blackbodies. The relation for these "grey bodies" adds an emissivity, $\varepsilon < 1$,

$$P = A\sigma\varepsilon T^4 \,. \tag{2}$$

The emissivity itself is typically not constant, and depends on temperature through a power law. For tungsten this relation is,

$$\varepsilon(T) = 1.731 \times 10^{-3} T^{0.6632} \,. \tag{3}$$

Including the emissivity leads to a power law between *power* and *temperature*. There is also a power law for the resistance of the bulb. In the simplest case, it is treated as linear ($R \propto T$), but for tungsten it is more accurately,

$$R(T) \propto T^{1.209} \,. \tag{4}$$

Combining the dependencies on temperature with $P = VI$ and $V = IR$, a power law can be found between current and voltage. For an ideal black body with a linear relation between resistance and temperature,

$$I \propto V^{\frac{3}{5}} \,. \tag{5}$$

For a more physically accurate model of tungsten, we expect

$$I \propto V^{0.5882} \tag{6}$$

For most of this experiment, we are not concerned with the constant of proportionality. However, when comparing the theoretical curve to your data you might need to define a suitable value.

2 Analysis of power laws with logarithms

The tool for analyzing power laws is, again, logarithms. Start with the general power law,

$$y = ax^b \,, \tag{7}$$

and take the logarithm of both sides,

$$\log(y) = b\log(x) + \log(a)\,. \tag{8}$$

Thus, $\log(y)$ depends linearly on $\log(x)$, with slope b. We can view this linear relation on a log-log plot (logarithmic scales for both x and y).

Linear regression works again, with the same linear model function, except using $\log(x_i)$ and $\log(y_i)$ as the input data.

Note: The same downfalls of using transformations apply as outlined in Exercise 3 (exponential decay). Again, using `curve_fit()`, we can use a nonlinear function directly as our model function.

3 The experiment

We will observe the power law for blackbody radiation through a lightbulb's voltage-current graph. Set up the Ohm's Law experiment as in Exercise 2, but with a lightbulb instead of a resistor.

After setting up the apparatus do the following:

1. Adjust the power supply voltage to its lowest value.
2. Wait for the voltage and current to stabilize (it shouldn't take long).
3. Record the voltage and current values.
4. Also record the measurement error.
5. Repeat this process for at least 15 different values of voltage.
6. Save the values in a text file (`.txt`) and save it to your memory stick

4 The Python program

You will use your programs to compare the transformation method and the non-linear least-squares method. The best parameters will be found with both methods, and used to plot best fit curves over the data.

The program should be organized as follows:

- Import the required functions and modules (`numpy`, `scipy.optimize.curve_fit()`, `matplotlib.pyplot`).
- Define the model functions (linearized: $f(x, a, b) = ax + b$, non–linear: $g(x, a, b) = a\,x^b$)
- Load the data and uncertainty measurements using `loadtxt()`.
- Perform the linear regression on $(\log(x_i), \log(y_i))$ using f as the model function.
- Perform the nonlinear regression on (x_i, y_i) using g as the model function.
- Output both of the power law relations you calculated.
- Plot the errorbars, both curves of best fit, and the theoretical curve. Include a legend for the different curves.
- Plot all the same things on a log–log plot. One way of doing this is with the `pylab.loglog()` function, another way is to call `pylab.yscale('log')` and `pylab.xscale('log')` after you make the plot.

The `a` and `b` parameters in the two model functions represent different parameters in the original, untransformed, model function. Make sure you can convert between the functions correctly to compare the parameters accurately.

Write the program, and run it using the data gathered in the experiment. Save all plots and the parameters you determined.

Which regression method gave an exponent closer to the expected value? Can you see the difference on the plots?

5 Analyzing the quality of the fit

As covered in previous exercises, there are two ways that we can assess the quality of the fit of our model: variance of the calculated parameters, and the reduced chi-squared statistic.

5.1 Variance of parameters

The variance of the parameters is returned by `curve_fit()` as the diagonal entries in the covariance matrix, `p_cov`. Recall that the error in measurements is understood as the standard deviation,

$$a = \tilde{a} \pm \sigma_a \,, \tag{9}$$

where $\sigma_a = \sqrt{\mathrm{Var}(a)}$. In the python program, the variance of the first parameter in your model function is given in `p_cov[0, 0]`, the variance of the second parameter in `p_cov[1, 1]`, and so on.

Modify your program from Section 4 to calculate the standard deviation of the parameters. What values did you find? Does the value of your fitted exponent fall within the range of the blackbody values ($\frac{3}{6}$) with your calculated standard deviation. What about in comparison to the expected value for tungsten?

5.2 Reduced chi-squared

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

Add a function to your program to calculate χ^2_{red}. What values were computed? How would you interpret your results?