Nonlinear fitting methods I

Radiation and nonlinear circuits

We continue with the curve fitting from the past exercise, but extend it to nonlinear models. Radioactive decay is an example of an exponential relation. We will measure the intensity from a radioactive isotope, and use curve fitting tools to find the half-life.

Background knowledge for exercise 3

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

1 Introduction

You've probabably learned about exponential decay of radioactive materials in the past. The number of emissions (during a sampling period) depends on the number of atoms of that isotope. Because an emission event corresponds to a death of an isotope, it follows that the intensity of radiation decays exponentially,

$$I(t) = I_0 e^{-\frac{t}{\tau}}\,, \tag{1}$$

where $I(t)$ is the intensity of radiation at time t, and τ is the mean lifetime of the isotope. A more intuitive parameter is the half-life, $t_{1/2}$, defined so that

$$I(t) = I_0 \frac{1}{2}^{\frac{t}{t_{1/2}}}\,. \tag{2}$$

Our interest for this experiment is calculating the half-life of an isotope by measuring the intensity (emissions/sample) of radiation.

The sample you will measure today is a metastable Barium (Ba-137m). It is prepared by flowing an acid through a generator containing Cesium (Cs-137). Inside the generator, the Cs-137 beta decays (with a half-life of 30.1 years), into Ba-137m. The acid washes out the Ba-137m which decays by gamma emission (662 keV) with a half-life of 2.6 minutes. We will measure the rate of emissions using a Geiger counter.

2 Analysis with logarithms

Nonlinear relations become more complicated to analyze in general. There are analytical formulae to find the parameters in a linear regression, but typically not for a nonlinear regression.

One method for nonlinear regression is transforming the data into a linear model and using linear regression methods. Logarithms are a particularly useful tool for analyzing exponential relations. Consider the general exponential relation,

$$y = y_0 e^{ax}\,. \tag{3}$$

Taking the logarithm of both sides,

$$\log(y) = ax + \log(y_0)\,. \tag{4}$$

Thus the exponential relation for y became a linear relation for $\log(y)$. Using a semi-log plot, with a logarithmic scale for the y-axis turns an exponential into a straight line, where the slope of the line is the growth/decay rate.

By first transforming the relation, linear regression can be used to find the parameter a. The model function is the same, $f(x) = ax + b$, but now, the data to match is $z_i = \log(y_i)$.

The transformation of the data also requires an error propagation to get more accurate results from the regression,

$$\sigma_{z_i} = \left| \frac{\sigma_{y_i}}{y_i} \right| . \tag{5}$$

This propagation formula is the standard one for logarithms.

3 Nonlinear regression

Traditionally, the method described in Section 2 was used for analyzing exponential relations, because a formula could be used to calculate the parameters. However, there is a problem with performing a linear regression on $z_i = ax_i$.

The maximum likelihood of a χ^2 linear model assumes that the *measurement errors* are normally distributed. When the axes are rescaled, this assumption is *potentially* invalidated. A related side-effect is that certain ranges of y are given more weight in the regression.

With the use of computers and nonlinear optimization, it is now possible to perform a least-squares optimization using a wide range of nonlinear models. Recall that `curve_fit()` uses least squares to minimize

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{y_i - y(x_i)}{\sigma_i} \right)^2 , \tag{6}$$

where y_i is the *dependent* data you measured, x_i is the *independent* data, σ_i is the measurement error in the *dependent* data. For an exponential model, we simply use $y(x) = ae^{bx}$ as the model function.

Computer-aided nonlinear regression brings with it other things to consider like whether the model function is suitable, if the initial guess for parameters is close enough to the true optimum values, and still if the measurement errors fit a normal distribution.

4 The experiment

The radioactive sample used in this experiment will be prepared for you by a lab technologist, as a demonstration.

You will not acquire the data on your computer. Instead youll find the radioactive decay text file at: `/Desktop/2nd Year lab Files/Exercises/Cesium Radioactive Decay.txt`. The instrument used to acquire the data is called a Geiger counter.

You will also be provided with a text file with data for background radiation: `/Desktop/2nd Year lab Files/Exercises/Radioactive Background 20min 20secdwell.txt`

The radioactive energy is quite low and safe. We are using a single demonstration Geiger counter because of the complexity of setting up 30 identical experiments before the sample has completely decayed. You still need to document the experiment in the lab–book as if you were conducting it.

4.1 Uncertainty in the Geiger counter

The Geiger counter follows *counting statistics* for determining the uncertainty. This is much like counting the number of heads in a series of coin-flip trials (the $\sqrt{N}$ rule). The rate is the number of events N divided by the sample time Δt. The best estimate of the count rate is given by,

$$R = \frac{N}{\Delta t} \pm \frac{\sqrt{N}}{\Delta t} . \tag{7}$$

There is also background radiation to consider; you should have heard the counter beeping with no sample present. The counts from the two sources of radiation add together for our final reading,

$$N_\mathrm{s} = N_\mathrm{T} - N_\mathrm{b}\,. \tag{8}$$

The uncertainty can be calculated via standard error propagation, so

$$\begin{aligned}\sigma_\mathrm{s} &= \sqrt{\sigma_\mathrm{T}^2 + \sigma_\mathrm{b}^2} \\ &= \sqrt{N_\mathrm{T} + N_\mathrm{b}}\,.\end{aligned}$$

Finally, because the background radiation cannot be measured at the same time, we assume it has constant statistics, and use its mean in the calculations ($N_\mathrm{b} = \bar{N}_\mathrm{b}$). The mean background radiation must be subtracted from each data point before analysis.

5 The Python program

We will use our programs to compare the transformation method (Section 2) and the non-linear least-squares method (Section 3). The best parameters will be found with both methods, and used to plot best fit curves over the data. For this experiment, we have the luxury of comparing our calculated half-time to the theoretical value.

The program should be organized as follows:

- Import the required functions and modules (`numpy`, `scipy.optimize.curve_fit()`, `matplotlib.pyplot`).
- Define the model functions ($f(x, a, b) = ax + b$, $g(x, a, b) = b\,e^{ax}$)
- Load the data using `loadtxt()`.
- Subtract the mean background radiation from the data.
- Calculate the standard deviation for each data point.
- Convert the count data into rates (divide by Δt).
- Perform the linear regression on $(x_i, \log(y_i))$ 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 half-life values you calculated.
- Plot the errorbars, both curves of best fit, and the theoretical curve. Include a legend for the different curves.
- Make a second plot of the same data, but using a logarithmic y axis. One way of doing this is to call the `pylab.semilogy` function. Another way is to call `pylab.yscale('log')` after you make the plot.

An alternate form of the non–linear g function can be used where the exp is replaced by $\frac{1}{2}$ or 2. You can also treat the linearized model f as a natural logarithm (ln), or a base–2 logarithm ($\log_2$). These transforms might change the way you process your data for input into the function, or change the way you interpret the errors from the `curve_fit`.
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 a half-life closer to the expected half-life of 2.6 minutes? Can you see the difference on the plots?

6 Analyzing the quality of the fit

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.

6.1 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,

$$t_{1/2} = \tilde{t}_{1/2} \pm \sigma_{t_{1/2}}\,, \tag{9}$$

where $\sigma_{t_{1/2}} = \sqrt{\mathrm{Var}(t_{1/2})}$. The variance of the parameter `a` is `p_cov[0, 0]`.

To get $\sigma_{t_{1/2}}$ from $\sigma_{\mathtt{a}}$, you will need to use the error propagation formula for a reciprocal,

$$\sigma\left(\frac{1}{a}\right) = \frac{\sigma_a}{a^2}\,. \tag{10}$$

Modify your program from Section 5 to calculate the standard deviation of the half-life. What values did you find? Does the nominal half-life (2.6 minutes) fall in the range?

6.2 Reduced χ^2

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.

This best (most likely) value of χ^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,

$$\chi^2_{\mathrm{red}} = \frac{1}{\nu}\sum_{i=1}^{N}\left(\frac{y_i - y(x_i)}{\sigma_i}\right)^2\,. \tag{11}$$

For χ^2_{red}, the best value is 1:

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

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

Prepared by John Ladan and Ruxandra Serbanescu, 2016. Edited by Christopher Lee, 2017.