Experiment 214: Non-linear Curve Fitting

Aim

The aim of this experiment is to fit curves to sets of data described by functions which are non-linear with respect to their parameters, and to analyse the results of the fitting. In particular, you will learn how to use the PYTHON function `curve_fit`.

Note: You must complete Experiment 213 "Data Analysis With PYTHON" before attempting this experiment so that you are familiar with the theory of linear curve fitting. It will be assumed that you have learnt how to use PYTHON to a sufficient level to have completed the "Data Analysis With PYTHON" experiment.

Reference

1. *Table of Isotopes,* C.M. Lederer, J.M. Hollander & I. Perlman, John Wiley

Introduction

`scipy.optimize.curve_fit` is an error weighted general non-linear least squares fitting function returning the fitted parameters and their standard errors `p` and `sp`, and the normalized chi-squared value `nchi2` by means of the function:

```
p,cov=curve_fit(model,x,y,sigma=sy,p0=pguess)
```

where

`model`	is the name of the PYTHON function file which defines the function to be fitted,
`x`	is the independent variable; `x`, `y` and `sy` are vectors
`y`	is the dependent variable
`sy`	is the standard error of `y`
`pguess`	is a vector containing the parameter estimates

In this pamphlet, sections which require you to obtain some results are numbered.

A pamphlet about how to use both linear and non-linear fitting functions in PYTHON is available in the laboratory. You should read the pamphlet before starting this experiment. The function `polyfit`is part of the `pylab` library, that should be loaded by typing

```
from pylab import *
```

Non-linearity and the chi-squared Value

A function is linear with respect to its parameters if it is of the form:

$$y = a\,f\,(x) + b\,g\,(x) + c\,h\,(x) + d\,k\,(x) + \ldots$$

where the parameters of the function are $a, b, c, d, \ldots$, *i.e.* a function is linear with respect to its parameters if all of the parameters appear only as coefficients of functions of x. A function which is not linear in one or more of its parameters is called non-linear with respect to its parameters, and will be referred to in the rest of this experiment just as 'non-linear'.

Example:

$y = a\,x^2 + b\,x + c$ is linear, but $y = \exp(b\,x) + c$ is not. In the second equation the function is non-linear, because it is not linear in the parameter b.

All polynomials are linear functions in their parameters. In particular, fitting a straight line or a quadratic function is a linear fitting.

The curves that will be fitted to experimental data in this experiment are non-linear. Non-linear fitting is done by minimising the chi-squared (or misfit) function, given by:

$$\chi^2(p) = \sum_i \frac{(\widehat{y}_i - y_i)^2}{\sigma_i^2}$$

where

σ_i	is the standard error in data point i
y_i	is the i^{th} experimental value of y
$\widehat{y}_i$	is the i^{th} fitted value of y
p	$= (p_1, p_2, p_3)$ is the vector of parameter values.

The chi-squared function produces a single number, called the chi-squared value, which measures how well the fitted curve corresponding to the parameter set p fits the data. The lower the chi-squared value, the better the fit. The parameters corresponding to the fitted curve that has the minimum value of chi-squared are the best fit parameters. The number of degrees of freedom for the fit is given by:

$$\text{number of degrees of freedom} = \text{number of data points used for the fit} - \text{number of fitted parameters.}$$

Clearly as the number of data points being fitted rises, the chi-squared value will increase due to the effect of the additional data points (which introduce more misfit), and the number of degrees of freedom will rise.

The Normalised chi-squared Value

It is convenient to define a measure of goodness of fit that is independent of the number of data points. This is the normalised chi-squared value, which is given by:

$$\text{normalised chi-squared value} = \frac{\text{chi-squared}}{\text{number of degrees of freedom for the fit}}.$$

The expectation value of the normalised chi-squared value is 1; more importantly, the normalised chi-squared value has at least a 90% probability of being in the range between 0.5 and 1.5 for greater than 20 degrees of freedom. It is highly unlikely that the fit is a good one if the normalised chi-squared value lies outside this range. One cause for the normalised chi-squared to be unusually large or small is that the errors in the data points are over– or under-estimated. In the exercise below, we calculate the normalised chi-squared value of a randomly generated set of data and confirm that the value lies near 1.

(1) We want to write a script file to generate a set of data which can be modelled by the equation $y = 2\,x+3$, for x between 0 and 40. To do this, first create the `x` and `y` vectors using the commands:

```
x = arange(0,41,1);    % generates a row vector
y = 2*x+3;      % y is also a row vector
```

Add Gaussian noise with standard deviation of 2 to this data using the line:

```
y = y + normal(0.0,2.0,41);
```

Fit the data using `polyfit`, assuming a standard error of 2 in all data points. Plot the data as discrete points together with the fitted line. Add error bars using the function `errorbar`. Calculate the normalised chi-squared value for the fit. Run the program a few times and check that the normalised chi-squared value generally lies within the range 0.5 to 1.5. Remember to obtain the parameter values by using the call

```
p,cov=polyfit(x,y,1,w=1/sy,cov=True).
```

Linearisable Problems

In some cases we may model a set of data with a function that is non-linear in its coefficients, but which can be linearised through the use of logarithms. A common example is the equation:

$$y = A\,\exp\left(-b\,t\right) \tag{1}$$

This equation describes many physical processes, including radioactive decay. In order to make the relationship linear, we can take the natural logarithm of both sides of the equation to end up with:

$$\ln\left(y\right) = \ln\left(A\right) - b\,t \tag{2}$$

from which we see that a graph of $\ln(y)$ versus t will be a straight line, *i.e.* $\ln(y)$ is linear in the two parameters $\ln(A)$ and b.

When used to describe radioactive decay, the symbols in the equation have the following meanings:

t	is the time at which the decay rate has been measured
y	is the number of counts recorded in each time interval
b	is the decay constant
A	is the number of counts recorded in the first time interval

A set of data from a radioactive decay modeling experiment will be provided for you. The aim of this section is to linearise the data and make a linear fit in order to obtain the value of the radioactive decay constant. This result will then be compared to the results of a non-linear fit for the same problem done in the following section.

Radioactive counts obey a Poisson distribution. As a result, the standard deviation in each recorded count is equal to the square root of the recorded count. For instance, if 50 counts were recorded in an interval, then the standard deviation in this count total is $\sqrt{50} = 7.07$. The standard deviation will be used as an estimate of the standard error (or randomness) in each data point.

(2) Create a script file and enter the following data:

```
t = arange(0,6.0,0.2)
y = array([398,344,306,252,250,220,159,154,140,131,126,108,85,91,73,54,66,58,54,
34,44,33,23,24,28,24,17,12,13,13])
```

Define the vector of standard errors `sy` in the `y` values to be equal to the square root of the `y` values with the following line:

```
sy = sqrt(y);
```

Now when we take the logarithm of equation (1) in order to linearise it, we end up with $\ln(y)$ on the left hand side of the equation. It is now necessary to calculate the error in $\ln(y)$ for each of the y data points. We apply the general formula for the propagation of errors.

If we have a function w, which is a function of other variables $a, b, c, \ldots$, *i.e.*

$$w = w(a, b, c, \ldots)$$

then the error σ_w in w can be calculated if we know the errors in $a, b, c, \ldots$ using the following formula:

$$\sigma_w^2 = \left(\frac{\partial w}{\partial a}\right)^2 \sigma_a^2 + \left(\frac{\partial w}{\partial b}\right)^2 \sigma_b^2 + \left(\frac{\partial w}{\partial c}\right)^2 \sigma_c^2 + \ldots$$

Here we have $w = \ln(y)$. Differentiate w with respect to y and use the formula above to obtain the code necessary to calculate the errors in the `ln(y)` vector from the errors in the `y` vector. Add this code to your script file, defining new variables `w = ln(y)` and `sw` as the standard error in `w` calculated using your error formula.

(3) Now add the following lines of code to your file:

```
p,sp = polyfit(t,w,1,w=sw,cov=True)
wfit = p[0]*t + p[1]
plot(t,w,'x',t,wfit)
errorbar(t,w,sw);
```

This will carry out the linear regression and plot the experimental data as points together with the fitted line. Error bars will be added to the data points. The first element in the vector `p` is the quantity `b`. The error in this quantity is the first element of the vector `sp`. Record the value of `b` and `ln(A)` together with their errors. Use the error formula above to calculate `A` with its error. Calculate the chi-squared value.

Non-linear Fitting

We wish to check the result we found above by using a non-linear fitting function available in the laboratory. The function we will use is `curve_fit`. Import it into your PYTHON session by typing

```
from scipy.optimize import curve_fit
```

There are several non-linear fitting algorithms which can be used for problems such as this one; all of these algorithms are centred around the idea of minimising chi-squared. A non-linear fit is much more complicated than a linear fit, and the final solution cannot be calculated directly. Instead, an iterative method is used, with each iteration improving on the previous solution until the optimum final solution is found. Due to the iterative nature of these methods, an initial guess of the parameter values is required as a 'starting point' for the algorithm. Usually the parameter values are known (or can be guessed) to the correct order of magnitude which is all that is required of a starting guess. If the guess is of the wrong order of magnitude then the algorithm may still converge to the correct solution, but will be slower and convergence becomes less likely.

Also required by a non-linear fitting function is a function we wish to fit to the data. It should take at least two arguments, the first is the independent or x variable array, and the following parameters are to be fitted. For instance, to fit a sine wave with variable amplitude and frequency, we would use:

```
def myfitfn(x,A,k):
    return A*sin(k*x)
```

(4) Write a file defining the function $y = A\exp(-b\,t)$ that we wish to fit to the data. .

(5) Create a new script file containing the `t` and `y` data from the previous section. Define the vector `sy` as before, and call the non-linear fitting function with the line:

```
p,cov = curve_fit(model,t,y,sigma=sy,p0=pguess);
% In your calling program you will have given your estimates of A and b
% in pguess[0] and pguess[1] respectively
```

The function returns the fitted values of the parameters, and the covariance matrix. Plot the data together with the fitted line (use the `semilogy` command). The variances of the parameters are given by the diagonal elements of the covariance matrix. Compare the value of the parameters `A` and `b` and their error to the result from the previous section. Note that exact agreement is not expected; what is expected is that the parameter values agree to within their error range, and that the errors agree to at least 1 (preferably 2) significant figures.

Experimental Data Fitting

The results obtained from many analytical instruments often take the form of a histogram obtained by plotting 'counts per channel' ('counts') versus 'channel number' ('chan'). It generally takes the form of peaks standing on a background. The peaks may be isolated or partially overlapping. In many cases this is referred to as a 'spectrum'. For instance in this section you will analyse the low energy gamma and X-ray spectrum of the radiation emitted by the radioactive isotope ^{57}Co. In such a spectrum, the peak due to a mono-energetic photon beam is expected to be well described by a gaussian distribution. A convenient measure of the width of a peak is the 'full width at half maximum' ('FWHM'). The energy, E, of the photon varies linearly with the channel number, $E = a + b*$ chan, while the width of the peak increases smoothly as the energy of the photon beam increases. Often a linear dependence is appropriate, FWHM $= c + d * E$.

(6) Load the ^{57}Co spectrum from this file: http://advlabsvr.phy.auckland.ac.nz/data/co57.mat. More details on loading data can be found in this document: http://advlabsvr.phy.auckland.ac.nz/pamphlets/e301.pdf

The data appears in two vectors:

chan	– contains the channel numbers
counts	– contains the number of counts for each channel

Plot the spectrum. You will see that it consists of three main peaks superimposed on a varying background. The left hand peak is due to the characteristic iron K_α X-ray of energy 6.40 keV. The prominent right hand peak is also a K_α X-ray from a certain element. It has an energy of 21.12 keV. (Actually, in both cases, other peaks are due to the close-lying $K_{\alpha 1}$ and $K_{\alpha 2}$ X-rays.)

(7) In this section we will fit a Gaussian peak on a linearly sloping background to the middle peak in the spectrum. The Gaussian should be of the form:

```
y = A*exp(-(x-x0)**2/d**2)
```

N.B: FWHM = 1.67*d

Remember that a radioactive count is Poisson distributed with a standard error equal to the square root of the count.

Select the central part of the spectrum by creating new vectors 'midchan' and 'midcounts' that contain only the data from channels 61 to 120:

```
midchan = chan[60:120];
midcounts = counts[60:120];
```

Plot out this central part of the spectrum. Work out what each of the constants `A`, `x0` and `d` represent in the Gaussian formula and estimate values for them from the plot.

Write script files to fit a Gaussian peak and linear background to this part of the spectrum. Comment on your results:

Is the fit a good one ?

Which parameter values can be determined precisely, and which are imprecise ?

Is this what we expect ?

(8) Modify your script files to fit the entire spectrum using three Gaussian peaks and a linear background. Comment on the results.

(9) Plot the entire spectrum with a log scale on the vertical axis. You will notice that there is a very small, noisy fourth peak overlapping with the end of the third peak. Refit the spectrum with four Gaussian peaks, and comment on the change in the normalised chi-squared value.

(10) Record chisq/degrees of freedom for each fit and the values of `A`, `x0` and `d` for each fitted peak.

Questions

1. Show that, for a Gaussian peak, FWHM = 1.67*d.
2. Determine the energy of the central peak.
3. What is the element which gives rise to the 21.12 keV peak? To answer this question, you need to understand that, when suitably excited, the atoms of all elements emit X-rays of a characteristic energy. Your first year Physics textbook will have a section on these 'characteristic X-rays'. You will next need to consult a reference book which lists these for each element. Useful tables are to be found in Kaye and Laby, American Institute of Physics Handbook, Table of Isotopes, etc. (It arises from the material in which the ^{57}Co source is embedded.)

Damped Simple Harmonic Motion

In this section we will fit a theoretical curve to data taken from an exponentially damped sinusoidal oscillator.

(11) Load the damped oscillator data from this file: http://advlabsvr.phy.auckland.ac.nz/data/dampshm.mat

The data appears in two vectors:

time	– the time at which each measurement was taken
position	– the position of the oscillator (in cm)

Plot a graph of position against time to see what the data looks like. You will notice that the sinusoid has been displaced from zero so that its equilibrium position is around 20.

(12) The experimenter estimates that the error in any position measurement is 0.1 cm, the same for all data points. Choose a suitable formula to fit to the data. Estimate initial parameters by looking at the graph of position against time, and write script files to fit the data with your formula. Record the fitted parameter values and their errors, as well as the normalised chi-squared value. Comment on the results.

Equipment

1. PC computer

M.I.J. Fleming

A.R. Poletti

Last updated: July 24, 2015