Tutorial 6. Non-linear fitting

Non-linear fitting

As we have seen the matrix formula $\mathbf{(X^TX)^{-1}X^TY}$ allows us to calculate the least squares estimates in a variety of models, provided these models are *linear in the parameters* $\boldsymbol{\beta}$. In many cases we cannot linearize our fitting problem. Fortunately you can still minimize the residuals (actually their sum of squares SS) with a very similar formula $\mathbf{(J^TJ)^{-1}J^TY}$.

The difference between the two is that **X** only contains information on where we take our data (our *independent* variables). **J** however also depends on the parameters themselves. In fact **J** contains the derivatives of the fit function f(x; $\boldsymbol{\beta}$) versus each parameter in each chosen data point.
This means that we need to have an idea of what $\boldsymbol{\beta}$ is before we can compute **J**. It also means that $\mathbf{(J^TJ)^{-1}J^TY}$ will only give us a *better* estimate of $\boldsymbol{\beta}$, not the *best*. That's no problem: we can keep applying the process until no more improvement is observed. This iteration process is called ***refinement***.

1. make guess of parameters
2. calculate the **J** matrix based on that guess
3. calculate $\mathbf{(J^TJ)^{-1}J^TY}$ to get better parameters $\boldsymbol{\beta}$,
4. if this is an improvement go to step 1; if not stop process

What refinement does is look for the minimum in the SS function. However this function is now like a landscape with hills and valleys, not a single well. Therefore the procedure will only work ***if*** your initial guess for $\boldsymbol{\beta}$ is close enough to the final minimum. Otherwise the procedure gets lost in the hills.

The final $\mathbf{(J^TJ)^{-1}}$ matrix and Sum of Squares tells you the uncertainties in the final parameters, just like its cousin $\mathbf{(X^TX)^{-1}}$ did. Because you have to terminate the refinement process somewhere (if improvement is less that some criterion) these values are known as *asymptotic standard errors* and $\mathbf{(J^TJ)^{-1}}$ produces an asymptotic variance covariance matrix.

Excel contains an add-in that will do all this for you and we will fit some data with it. The add-in is called the ***solver***. The instructor/TA will help you to make sure it is loaded. Unfortunately the Excel solver does not produce values for the uncertainties, but the book Excel for Chemist by J. Billo has another module on its CD that remedies that.

The data

The data we will be fitting comes from the TGA, in fact it is a decomposition run on Calcium oxalate monohydrate ($CaC_2O_4.H_2O$). It represents a series of decomposition reactions the oxalate as the temperature is ramped to 1000°C.:

$CaOx.H_2O \rightarrow CaOx \rightarrow CaCO_3 \rightarrow CaO$

The first step produces water vapor, the second CO and the third one CO_2.

Open the Non-lin excel spreadsheet. Select the data block and make a graph (line only) of the data in the C column (weight) versus the A column (time). We will work in time, but as the oven was ramped at constant heating rate we could easily translate time into temperature (in the B column).

The Model Function

The biggest problem with fitting data is always to formulate a reasonable *model*. More often than not you do not know '***the'*** model and this is a good example: a good physical model is not known for this kind of data. The graph certainly makes it obvious a straight line will not do! Deviations from straight can often be modeled by adding higher order terms (a polynomial) but this is not recommended for this type of data.
Sigmoidal (S-shaped) step functions are notorious for requiring an infinite number of terms to fit well. This violates the ***parsimony*** principle: always try to retain as many degrees of freedom as you can, or stated differently: is you can fit something with three variables, do not fit it with 300. If you throw in enough parameters you can fit even the kitchen sink! This is why we fit this with a function that already has a sigmoidal shape, the ***logistic function lgt(x):***

$$lgt(x) = \frac{e^x}{1 + e^x}$$

We can fit every decomposition event as:

$$W_o lgt(a - bt)$$

As you see there are three parameters (not 300!) per event: W_o, a and b. W_o stands for the amplitude of the change (the entire weight loss of the event). The time around which the event happens is t_{event}= -a/b and the value of b represents the slope in the weight curve at this time (how sudden the event takes place). As we are losing weight its value is always negative in our case. As we have three events, but do not lose all weight the total fit function becomes

$$W(t) = W_{o,1} lgt(a_1 - b_1 t) + W_{o,2} lgt(a_2 - b_2 t) + W_{o,3} lgt(a_3 - b_3 t) + W_{baseline}$$

As you see we have a total of ten parameters: 3+3+3+1.
It is advisable to build up such a fit job systematically in your sheet and not write out a function like this in one cell as you are bound to make mistakes.

- First copy the entire data block (three columns) and open a new sheet and paste it in cell A8, B8 and C8. It consists of 1452 points so that the three columns should be filled down to line 1460.
- Now put the following initial guesses for the parameters in the range C1:G4 including the labels. The numerical parameters should be in D2:G4.

	first	second	Third	final
W	2	3	5	7
A	15	40	35	
B	2	2	1	

1. Type in C6: t-event
2. Type in D6: -D3/D4
3. Copy D6 over D6:F6
4. Type in D8: =D$3-D$4*$A8 (this calculates *a-bt* for the first event)
5. Copy D8 over D8:F8
6. Type in G8: =EXP(D8)/(1+EXP(D8)) (The lgt function)
7. Copy G8 over G8:I8
8. Type in J8: =$D$2*G8+$E$2*H8+$F$2*I8+$G$2 (This computes the W(t) fit function)
9. For plotting purposes we will copy the relevant columns:
10. Type in K8: =A8; (time)
11. Type in L8: =C8 (the measured weight)
12. Type in M8: =J8 (the fit function)
13. Type in N8: =L8-M8 (the residual)
14. Now select D8:N8 and use the double click on the + symbol that appears on the bottom right corner of N8 to double click and fill all your calculations over the whole data set.
15. Select the K,L and M functions and make a chart with only lines of the measured and calculated data.
16. You can now change the values in the parameter block to make the function fit a bit better. It is useful to change them by hand to get a feel for what they do. If the new guess is really bad, just hit Ctrl+z to correct your mistake. Here is a trial example using the input data set from the website.
17. n e.g. M4 type =SUM(N8:N1459^2)/1450/STDEV^2. To activate this formula use <Shift> <ctrl> <Enter>. This calculates the a value known as chi-squared,

$$\chi^2 = \frac{\sum(y_i - y_{model})^2}{N\sigma^2}$$

Chi-squared is the sum of the squares (SS) divided by the number of data points and STDEV, where STDEV is the standard deviation in your data. You may estimate STDEV using a flat section of the data to generate a series of numbers. Then you can calculate the STDEV for this section. Remember that STDEV is equal to,

$$\sigma = \sqrt{\frac{\sum(y_i - \langle y \rangle)}{M}}$$

where M is the number of data points in the short section you are using to calculate the STDEV. M is not at all the same as the value of N you used above, which consists of all of the points in the data set.

18. Try a change in one of the parameters and look at the value of SS. It should get lower as the fit gets better. When you have obtained an initial guess that is reasonably close to the shape of the data then you can use the solver to get a better fit.
19. Now run the solver. It should be under Data.
20. On the pop up click the icon with the red dot of the *set target cell* option and select the cell that contains the SS (M4)
21. Then click the *Min* option of the *Set equal to* set. (We want to minimize SS!!)
22. Click the red dot icon of the *By changing cells* block an select the cell that contains the final weight. Then click solve.

You can choose which parameters you want to run first. Often it is wise not to take too many parameters at once, but once you are close enough to a good fit you can select then all at once. E.g. you can click the red dot icon and select D2:F4 then type a comma and then click G2 to get them all. The fit is not bad but not ideal either. Make a plot of the residuals to see how bad it is!

In programs that use non-linear least squares fitting the errors in the parameters can be output using the second derivative of the $\mathbf{J^TJ}$ matrix to estimate the root-mean-square error in each parameter. It is important to have the correct weights for the estimate (i.e. the error should be calculated for displacements of the fitted curve relative to the σ you calculated above for a flat portion of the data). This type of analysis has numerous assumptions that may be incorrect. The error in the data may be larger than the σ you calculate. The curvature of the $\mathbf{J^TJ}$ matrix may have significant distortions from a quadratic shape. And so on. For this reason we will use another method to estimate the quality of the fit using a comparison of the the parameters to a physical model. This is often a useful approach and has more significance for the scientist than a number that comes from a statistical analysis that has possible numerical inaccuracies.

Let us make a hypothesis: we have obtained four weights from the regression, let us assume they correspond to the compounds H_2O, CO, CO_2 and finally CaO.

1. Compute the molar masses for these four compounds and plot the weights against the molar masses. This should give a straight line. The weight of the CaO is only correct if the instrument was properly calibrated.
2. Do a linear regression of the weights against the M_w. How many moles of Ca did the sample contain?
3. Using the slope and intercept compute an estimated weight for each compound and determine the absolute value of the residual from the regression line. These residuals should correspond to the asymptotic standard errors in the weights, which we did not calculate (it can be obtained from the covariance matrix $\mathbf{J^TJ}$ as discussed above). To get a better feeling for how important the residuals for each point (i.e. $\sqrt{(y_i - \langle y \rangle)^2}$) you could divide the residual by the standard deviation for the straight line fit.
4. What does the result say about the chemistry?

Assignment in peak fitting

Go to sheet 2 of the nonlin spreadsheet.

I generated some data for you. They consist of two partially overlapping Gaussian peaks. I made a version with two different noise levels. As these data are *generated* ones you may expect the final residuals to be random noise only. (Check residuals. Real data may not always be so nice.)

The data represent a problem that is often encountered in science, that of ***peak resolution***. In many techniques we get a pattern consisting of a series of signals each in the form of a peak. This is true for spectra, for thermograms, for chromatograms and many other types of data alike.

The peaks can have a variety of shapes. Gaussians are the simplest one and we will only do those today. Unfortunately peaks may *overlap*. The more they do the harder it is to separate the two signals and derive independent information from them. *Limited* overlap can be overcome by peak fitting.

Caution:

Peak fitting works reasonably well ***if***:

- the overlap is not too large,
- there are not too many peaks involved,
- you know how many there should be,
- you know what shape they should have (Gaussian, Lorentzian etc.)
- the noise level is low
- the peaks are not too broad
- the peaks are (more or less) symmetrical

If any of these requirements are violated, peak fitting notoriously yields various different solutions that are fit equally well but are ***quite incorrect***. In other words: you can get out what you want by changing the model….

This is why in e.g. chromatography people try to ***avoid*** overlap, e.g. by choosing a different internal standard that does not overlap. Instrument builders also try to make their broadening factor as small as possible to avoid or diminish overlap trouble. However overlap cannot always be avoided and fitting may be all you can do. The ***parsimony*** principle applies here: always fit with as few parameters as possible while demonstrating the fit is as perfect as random noise allows. (Look at residuals!)

There are other statistical methods that ***do*** allow you to use overlapping data, but they typically require that you do not have one spectrum, but a series of them, e.g. taken as a function of time

The data

For each of the 'spectra' that I generated, use non-linear fitting to determine the *amplitude*, the *position* and the *width* of the two overlapping peaks, i.e. six parameters in total. In Excel a Gaussian function f(x) is easily generated by the function:

=*amplitude**NORMDIST(x,*position*,*width*,0)

Add two of these terms to generate a fit model. Note that in Excel the NORMDIST function is ***normalized***. This means that it integrates to unity. Thus the *amplitude* parameter is automatically also the integration value of each peak.

In order to use solver you will also need to generate a target. Our target is chi-squared (χ^2), which is defined as above.

I used integer values to generate the data for all six parameters. How large is the bias, i.e. the difference between the parameters you find and the nearest integer? How does the noise level of the data influence the bias? What would happen if the noise level would get larger? What if the peaks got broader? Closer together?

Literature

Salvador Naya, Ricardo Cao, Ignacio Lopex de Ullibarri, Ramon Artiaga, Fernando Arbadillo and Ana Garcia, *Journal of Chemometrix* 2006 20:158-163