Optimisation of multi-parameter empirical fitting functions

By David Knight

Version 1.02, 1st April 2013.

Check the author's website to ensure that you have the most recent version of this document: http://www.g3ynh.info/ .

First version created in March 2010. v1.01→1.02: Minor typographical corrections only.

Introduction

For some types of data reduction problem, the 'least-squares' fitting criterion is inappropriate. This is the case (for example) when determining an approximate empirical formula to be used in place of a more complicated calculation procedure, where it is better to minimise the peak error (the runout) than it is to minimise the RMS error. There is also a choice over whether to define the fitting residuals as absolute (observed minus calculated) or as proportionate errors; and the average used in determining the proportion can be weighted anywhere between the observed and the calculated value. This article discusses some of the possibilities and their effects in the context of non-linear empirical modelling using downhill search methods.

Downhill search methods

When fitting data to a theoretical model, and presuming that there is no restriction on the number of variable parameters; the usual procedure for determining the optimal parameter values is to configure the problem as a set of simultaneous equations that can be solved numerically by inverting a matrix[1]. One reason for using this approach is that it provides intermediate results for the estimation of parameter uncertainties and correlation coefficients. If the model is empirical however, then the parameters have no physical significance and so cannot be used in other calculations; which means that the full statistical analysis is not required. The matrix inversion procedure can still be used of course, and has the advantage of computational efficiency; but for one-off fitting problems, such as developing approximate formulae or converting tabulated data into functions, a great deal of programming can be avoided by using a search method.

For the solution of any fitting problem, it is first necessary to define the fitting criterion. This is usually embodied in a composite error-function that produces a single number called the 'goodness of fit' or "Goof". The Goof is calculated from the fitting residuals in such a way that it takes every residual of finite weight into account and diminishes smoothly (i.e., without discontinuities) as the parameters are shifted towards their optimum values.

When solving by means of a search method, the variable parameters can be considered as the co-ordinates of a point on an N-dimensional hypersurface (this is easiest to imagine when N, the number of variable parameters, is 2, in which case the surface is simply a sheet). The Goof provides an extra dimension, the height of the terrain, and the job of the search algorithm is to walk downhill, using information obtained by applying increments to the parameters, until it finds itself at the bottom of a basin. Any minimum it finds is a possible solution; but note that for complicated non-linear systems, it could be just a local minimum, and there might be a better solution elsewhere. Hence, for many problems, there should be an initial hand optimisation phase (to get close to a good solution), before the parameters are passed over to the search routine. It is also sometimes useful to

1 The procedure is described in numerous references. See for example: **Data Reduction and Error Analysis for the Physical Sciences**, Philip R Bevington. McGraw-Hill, 1969. Library of Congress cat. card # 69-16942.
A brief explanation and Fortran code is given in **D W Knight PhD 1985,** Appendix 6. [www.g3ynh.info].

give the system a kick, to see if it can be made to jump out of one minimum and find another; and that can be done by adjusting the initial parameter shifts, or by changing parameters manually between fitting runs.

The simplest search strategy is that of cycling through the parameters one at a time and adjusting them to minimise the Goof. If the problem is non-linear (i.e., the parameters are not orthogonal), then, after one parameter has been adjusted, all of the others may require re-adjustment, and so it is necessary to repeat the cycle a large number of times until no further significant improvement can be obtained. This is known as a 'grid search', and the process can be painfully slow. The inefficiency can be overcome, to some extent, by using methods that involve shifting all (or, at least, some) of the parameters simultaneously.

There are numerous ways in which searches involving simultaneous parameter shifts can be carried out. One that works well and is popular for its reliability is the Nelder-Mead downhill simplex method[2] [3]. The Nelder-Mead algorithm is not necessarily the most efficient in terms of speed, but it has the ability to expand and contract the size of the parameter perturbations used, enabling it to find narrow wells in the hypersurface. A good reason for recommending it also is that it has been turned into an Open-Document spreadsheet macro function by Robert Weaver[4]. This makes it possible to attack elaborate non-linear optimisation problems using the free Open Office[5] software package.

Fitting Criteria

Since the only dynamic information available to the fitting algorithm is a single number (the Goof), the type and acceptability of the solution is determined by the way in which this number is calculated. If a least-squares fit is required, the correct choice for the Goof is Chi-squared[6], i.e:

$$\chi^2 = \Sigma(w_i\, e_i^2)$$

where e_i is the i^th^ error or 'residual', and is usually defined as:

$$e_i = y_{i(obs)} - y_{i(calc)}$$

$y_{i(obs)}$ is an observed value of the dependent variable, and $y_{i(calc)}$ is the corresponding value calculated using the fitting function. The weight w_i is strictly analytically proportional to the reciprocal of the variance of the observation (i.e., $w_i = 1/\sigma_i^2$ or some multiple thereof, making χ^2 dimensionless), but may sometimes be set manually to exclude bad data or otherwise gerrymander the result.

For some types of problem however, the least-squares criterion is inappropriate. In particular, when adjusting a function to agree with an exact or reference dataset (such as when developing an approximation formula), the residuals will not be normally distributed, and so the only valid criterion is that which guarantees that the maximum error will lie within strict limits. In other words, we do not want to minimise the RMS error (which is what a least-squares fit does), we want to minimise the maximum error. Also we need to consider the definition of error (should it be absolute; or should it be proportionate so that the overall error can be expressed as a percentage or in parts-per-million, etc.), and perhaps consider the relative authority of the reference data in comparison to the fitting function. To explain these issues, let us start with a generalised composite error function "eta to the 2m":

2 http://en.wikipedia.org/wiki/Nelder-Mead_method

3 **Numerical Recipes**. W H Press, B P Flannery, S A Teukolsky, W T Vetterling. CUP 1986. ISBN 0521 30811 9. [newer editions exist] Section 10.4: Downhill Simplex method in multidimensions.

4 http://electronbunker.sasktelwebsite.net/

5 http://www.openoffice.org/

6 See, for example: **Scientific Data Analysis**, D W Knight [www.g3ynh.info].

$\eta^{2m} = \Sigma(w_i \, e_i^{2m})$

If m=1, and the weight and error are as defined above, then this is the same as χ^2 and a least-squares-type fit will result. If m is a much larger positive integer however, then large errors will have vastly greater effect than small ones and the parameter adjustment process will tend to crush the larger errors. The result will then be the desired minimisation of maximum error (minimisation of runout). The power 2m is limited only by the need to to condition the problem so that the error terms do not cause floating-point underflow or overflow, in which case η^{2m} will become discontinuous and the fitting process could become erratic. Note that the factor of 2 in the power is to ensure an even number, the point being to keep all summation terms positive and prevent error cancellation.

For the individual residuals; if we want to minimise the absolute error, then $y_{i(obs)} - y_{i(calc)}$ should be used as the definition. Otherwise, we might want to minimise the proportionate difference between $y_{i(obs)}$ and $y_{i(calc)}$, and that difference needs to be defined in a way that not only produces the composite error function required by the fitting algorithm, but also produces a rigorous measure of the goodness of the fitting process.

For a least-squares fit, the correct overall measure of goodness is 'reduced chi-squared', i.e.; χ^2/ν , where ν (Greek "nu") is the number of degrees of freedom in the data, i.e., the number of observations of finite weight minus the number of variable parameters used in the fit. Since ν is a constant for a particular fitting run, it makes no difference to a search routine whether χ^2 is divided by ν or not (provided that floating-point errors do not occur). For minimum runout problems however, the statistic required is often the worst-case error; and that depends on both the error in the fitting process and the error in the reference data.

In the most general case, the proportionate difference between two numbers is the absolute difference divided by the weighted average. The weighted average is:

$(w_{(obs)} \, y_{(obs)} + w_{(calc)} \, y_{(calc)}) \, / \, (w_{(obs)} + w_{(calc)})$

where the weights are proportional to the squares of the respective reciprocal uncertainties (i.e., $w \propto 1/\sigma^2$ for normally distributed errors). Note that when both weights are the same, this reverts to $(y_{(obs)}+y_{(calc)})/2$, which is the familiar straight average. The proportionate difference is thus (in general):

$e_i = (y_{i(obs)} - y_{i(calc)}) \, (w_{i(obs)} + w_{i(calc)}) \, / \, (w_{i(obs)} \, y_{i(obs)} + w_{i(calc)} \, y_{i(calc)})$

This might seem complicated, but if we import some knowledge about the data, it will revert to its most familiar form. The most common assumption is that one of the y values is absolutely authoritative, and the other has no authority by comparison. This situation applies (correctly) when we have an exact (or at least extremely accurate) way of calculating say $y_{i(obs)}$. Such would be the case, for example, when trying to develop a simple fitting function to replace a much more complicated calculation procedure. In that case, $w_{i(obs)} = 1$ and $w_{i(calc)} = 0$, and we get:

$e_i = (y_{i(obs)} - y_{i(calc)}) \, / \, y_{i(obs)}$

or

$e_i = 1 - y_{i(calc)} \, / \, y_{i(obs)}$

which can be multiplied by 100 to give the familiar formula for calculating the error in $y_{i(calc)}$ as a percentage, or multiplied by 10^6 to give the error in ppM, etc.. If this type of error definition is put into η^{2m} (with $w_i = 1$), then the search process (if successful, and presuming that m is large) will minimise the proportionate runout. Furthermore, the worst case runout can be determined as: Max($|e_i|$), and this is the statistic that will be required by people who intend to use the fitted formula.

Sometimes however, we might want to refer the proportionate error to the fitting function rather than to the supplied data, or we might want some intermediate form using relative weights. Every situation should be considered separately, but one that crops up frequently is that of wishing to fit data that are accurate but not precise to a function that is precise but not accurate. Such is the case, for example, when fitting data that are of theoretical origin (accurate), but which have been rounded to a fixed number of decimal places or exhibit significant floating-point machine error (and are therefore imprecise). The fitting process in this case is a type of smoothing; the proper objective being to find a curve that averages the noise in the data. The end result will be that the smoothing function has a better knowledge of the exact y-values than does the original data, and so the average used in obtaining the proportionate error (should we decide to fit the data in that way) should be weighted to the calculated values. Thus we get:

$e_i = (y_{i(obs)} - y_{i(calc)}) / y_{i(calc)}$

i.e.,

$e_i = (y_{i(obs)} / y_{i(calc)}) - 1$

and the error function given to the fitting algorithm is:

$\eta^{2m} = \Sigma(w_i\, e_i^{2m})$

as it was before, but with the definition of e_i altered. For a fitting exercise in which all of the observations are taken to have the same uncertainty (absolute, or proportional, depending on how the residuals are defined), the weight of an observation, w_i , is usually set to 1 initially; but if we find that the dataset contains illegitimate errors (typographic errors, etc), or if the model breaks down in some regions, we can exclude individual observations from the fit by setting the corresponding w_i to zero. The problem of how to deal with data that do not all have the same uncertainty will be examined later.

Progressive weighting

When fitting data to minimise runout; it will often be found, after a first attempt, that the error increases or diminishes as the independent variable (x) is increased. Consider that we have a smooth non-linear fitting function:

$y_{(calc)} = f(x)$

which has a reasonably large number of parameters to be adjusted to bring it into near coincidence with a smooth reference function:

$y_{(obs)} = g(x)$

After fitting using a runout criterion, a plot of the error curve (e_i vs x_i) will usually show a series of undulations, but the peaks of the undulations will not necessarily be of the same height over the entire range of x. Instead, they will typically either increase or decrease in magnitude with increasing x, and if lines are drawn across the tops of the positive and negative peaks, it will be seen that the fitting function has developed a wedge-shaped pair of error boundaries. Now, if the problem is well-conditioned, it will be possible to trade peak height in one location for peak height in another, and so this is not the optimum fit. The solution is to apply progressive weighting. Recall that we have an error function:

$$\eta^{2m} = \Sigma(w_i \, e_i^{2m})$$

where the weights are usually set to 1. All we have to do to apply progressive weighting is use weighting coefficients defined thus:

$$w_i = x_i^u$$

For the first fitting run, we set u=0, so that all of the weights are unity. Then if we find that the error increases with x, we set u to a positive number (say 0.5), and run the fitting routine again. The fitting weight will then increase as x increases; the error for large x will decrease slightly, and the error for small x will increase slightly. With a little judicious adjustment of u between fitting runs, the error boundaries can be made parallel, and the maximum runout will then be less than it was before. Similarly, if the error decreases with increasing x, we make u negative; and the converse applies.

Sampling interval

When fitting experimental data to a mathematical model, the size of the dataset is limited by practical considerations. When comparing two smooth functions however, the dataset is potentially infinite, and we must select a finite number of points at which to make comparisons. There are no hard and fast rules on how to do that, but there is a simple test that will tell if the number of samples is insufficient. Always plot a graph of the residuals (e_i vs. x_i). The plotting software will draw straight lines between the points. If straight sections are visible in the peaks of the residual function, and especially if it obvious that a peak lies between two points, then the maximum runout will not be reported correctly if that is the highest (positive or negative) peak. The solution is to increase the number of data.

Since the size of the dataset can become enormous in some cases, the problem can often be made more manageable by noting that it is not necessary to use a constant sampling interval when generating argument values. Say we have a reference function:

$$y_{(obs)} = g(x)$$

The obvious thing to do is create a spreadsheet column for x and then populate it with x_i values using the 'Fill down' tool. Frequently however, we will find that the residual function has much more detail for low values of x than it has for high values, especially if the fitting function is an asymptotic form. The solution is to create a log(x) column (with a constant interval between values), and calculate x_i values from it using $x=10^{\log(x)}$.

In general; the interval generator can be any function that bunches the data where the residual function is changing rapidly and spreads it out where nothing much is happening. In this way the total number of samples can be reduced to a few hundred (say), rather than the thousands it might have taken to do the job well using a linear interval generator.

Dimensionless Goof

From the nature of the foregoing discussion, it might appear that the process of deciding how to go about fitting a given set of data is somewhat vaguely defined. Certainly, it can seem like that when trying to deal with unusual problems; but in fact there is an underlying logic that, if applied specifically, will indicate what should be done. It is a matter of addressing two issues:

- Are the errors normally distributed?
- Is the Goof (effectively) dimensionless?

If the errors are normally distributed (such as when fitting experimental data) then the data should be subjected to a least-squares fit. In that case, the Goof is:

$\chi^2 = \Sigma(w_i\, e_i^2)$

and if the residuals are defined as

$e_i = y_{i(obs)} - y_{i(calc)}$

then the weights must be defined as proportional to reciprocal variances.
We can re-write the expression for chi-squared as follows.

$\chi^2 = k\, \Sigma[\, w_{i(fit)}\, (e_i/\sigma_i)^2\ ]$

Now we have separated the fitting weight into three parts, i.e.;

$w_i = w_{i(fit)}\, k / \sigma_i^{\,2}$

$w_{i(fit)}$ is a fitting flag, which can be set to 1 of we want to include the ith observation, or set to 0 if it is suspected that a logging mistake or a breakdown of the model prevents this particular observation from being fitted. The number of observations of finite weight (used for calculating reduced chi-squared) is then simply $\Sigma w_{i(fit)}$. k is a global constant, which should be 1 if the standard deviations of the observations have been correctly scaled, but we might need to re-scale the standard deviations after fitting in order to get k=1.

Now notice that both e_i and σ_i have the same dimensions. Hence the effect of all of the decisions that were made in constructing the composite error function is to make the Goof dimensionless.

Now consider the problem of fitting data that do not have normally distributed errors. We can tell whether or not the errors are normally distributed by asking the question: 'Will the apparent standard deviation of fit change if the data are sampled in a different way?' In the case where one smooth function is being fitted to another (for example); the degree of agreement between the two functions will depend on the chosen range of x-values. Hence the concept of standard deviation becomes meaningless, because the differences are not random and depend on choices made by the investigator. Hence we must first try to make the dataset comprehensive; by taking plenty of samples overall, and particularly in regions where the disagreement is greatest. This has the effect of maximising the composite error (however we might choose to define it). Then, still unable to define a standard deviation, we must use a runout criterion. Thus we define the Goof as:

$\eta^{2m} = \Sigma(w_i\, e_i^{2m})$

Now the weighting coefficients become empirical parameters, initially set to 1, but later used to adjust the final error boundaries. They cannot be defined as reciprocal variances because the errors are not random. Finally, we need to decide on a definition for the residuals. In most cases involving runout minimisation, the error is defined as a proportion, in which case the residuals are dimensionless. Indeed, it is not possible to define a meaningful single-valued statistic for the accuracy of the fitting function unless the error is expressed as a proportion (%, ppM, etc.). Thus, unless we intend to supply users of the function with a comprehensive table of differences; the residuals must be defined according to a criterion that makes the Goof dimensionless.

Using the preceding logic, we can, if so desired, use a runout criterion for fitting data with normally distributed (or quasi-normally distributed) errors. The typical reason for wanting to do that is to fit the data using the minimum possible number of variable parameters; i.e., to keep the fitting formula as simple as possible but still get a good fit.

A commonly encountered situation is that of wanting to produce a formula to reproduce theoretical data given as a table. The numbers in the table will have rounding errors; i.e., the errors will not be strictly normally distributed, but there is a relationship between rounding error and apparent standard deviation that gives important information about the fitting function. If the errors are randomly distributed, then the most probable error in an observation is σ_i. If the errors are due to rounding, then the most probable error is half the worst-case rounding error. Hence, if the fitting function has sufficient variable parameters to smooth the data, the estimated 'standard deviation' of fit will be very close to half the worst-case rounding error. Thus, for example, if we have data rounded to 4 decimal places, we should get

$$\sigma_{\mathbf{fit}} = \sqrt{[\ (1/\nu)\ \Sigma(w_{\mathbf{i(fit)}}\ e_i^2)\ ]} = 0.000025$$

(where $w_{\mathbf{i(fit)}}$ is a fitting flag, 0 or 1). If the ESD is much smaller than that, then there are too many parameters and the empirical function is fitting the noise. If the ESD is much larger than that, then the function does not fit the data.

Nevertheless, despite the diagnostic value of $\sigma_{\mathbf{fit}}$, we still want to fit the data using a runout criterion. The reason is that the true errors have a 'brick-wall' tolerance, i.e., there should be no errors significantly outside the maximum rounding error, and a runout criterion will enforce that condition. Hence the success criterion is different from the fitting criterion. The Goof used by the fitting algorithm is:

$$\eta^{2m} = \Sigma(w_{\mathbf{i(fit)}}\ e_i^{2m})$$

but note that all of the residuals have the same uncertainty (all of the numbers are rounded to the same number of decimal places). Hence, the residuals can all be notionally divided by an arbitrary number having the same dimensions, and the only effect will be to scale the Goof. Scaling the Goof (multiplying the whole thing by a constant) has no effect on the fit (provided that the operation does not lead to floating-point errors). Hence the Goof is effectively dimensionless.

Now, having spotted that the trick in fitting data properly is to make the Goof either actually or effectively dimensionless, we can use this requirement as a test of correctness. It is impossible to envisage all of the situations that might arise; but take, for example, the problem of fitting accurate tabulated data that do not have a constant number of decimal places. In that case, the most probable error differs between observations. The solution is to divide each residual by its most probable error. Thus, if we call the most probable error $e_{\mathbf{i(mp)}}$, then we have:
for 2 decimal places: $e_{\mathbf{i(mp)}} = 0.0025$
for 3 decimal places: $e_{\mathbf{i(mp)}} = 0.00025$
for 4 decimal places: $e_{\mathbf{i(mp)}} = 0.000025$

etc.
The fitting criterion is then:

$$\eta^{2m} = \Sigma[w_{\mathbf{i(fit)}}\ (e_{\mathbf{i}}\ /e_{\mathbf{i(mp)}})^{2m}\ ]$$

which is dimensionless.
The success criterion is reduced chi-squared:

$$\chi^2/\nu = (1/\nu)\ \Sigma[w_{\mathbf{i(fit)}}\ (e_{\mathbf{i}}\ /e_{\mathbf{i(mp)}})^2] \approx 1$$

That this statistic should come out to be approximately 1 may not be immediately obvious, but note that an error divided by the most probable error should, on average, be 1. For a statistically representative dataset, the number of degrees of freedom (ν) should tend towards the number of observations. Hence, if the number of observations is n; the sum of n quantities that should be 1 on average, divided by a number that tends towards n, should tend towards 1 in the limit of large n if the data reduction has been carried out correctly.

*   *   *

A demonstration of the Nelder-Mead downhill simplex search method, using some of the fitting optimisation techniques discussed in this article is given in the accompanying Open Document spreadsheet: **Nelder_demo.ods** .