Treating Uncertainties in Both x- and y-Coordinate

David McCowan

February 10, 2015

Suppose we have N data points of the form $(x_i + dx_i, y_i + dy_i)$ (where dx_i and dy_i are the independent uncertainties on x and y respectively) which we want to describe by a model function $g(x)$. When we perform a least-squares fit, we determine the *best* function $g(x)$ as the one which minimizes the sum of the squares of the weighted residuals of each point. Let us step through this statement more carefully.

First, the residual on the ith point is the difference between the value of the data and the value of the fit function,

$$f(x_i, y_i) = y_i - g(x_i). \tag{1}$$

We wish to weight each of these residuals by some factor which encodes our total uncertainty about that point such that points with smaller uncertainties carry more weight in the sum than those with larger uncertainties. For now, we will call this uncertainty in the residual df_i and our weighted residual will be

$$\chi_i = \frac{f(x_i, y_i)}{df_i}. \tag{2}$$

Our final sum of the squared, weighted residuals is therefore

$$\chi^2 = \sum_{i=1}^{N} \chi_i^2 = \sum_{i=1}^{N} \left(\frac{y_i - g(x_i)}{df_i} \right)^2. \tag{3}$$

Note that we square the weighted residuals so that positive and negative residuals add together rather than cancel out.

If we remember the weighted residuals from the early examples in this class, we made the uncertainty in the residual equal to the given uncertainty in the data point—$df_i = dy_i$—regardless of the form of the fit function. This is true when the uncertainty in the y-values dominates over the uncertainty in the x-values, but what about cases where $dx_i \approx dy_i$?

In such a case, we must compute the uncertainty in $f(x, y)$ by summing partial derivative contributions in quadrature:

$$df_i^2 = \left(\frac{\partial f(x,y)}{\partial y} dy_i \right)^2 + \left(\frac{\partial f(x,y)}{\partial x} dx_i \right)^2. \tag{4}$$

In general, this uncertainty will depend on the fit function $g(x)$. Notice, however, that in the case where we supply only a y-error ($dx_i = 0$) or when the y-uncertainty is much larger than the x-uncertainty ($dy_i \gg dx_i$), this reduces to the more familiar $df_i = dy_i$.

Let's look at some examples.

- Suppose we fit to a **linear** function, $g(x) = Ax + B$. In this case

$$df^2 = dy^2 + (Adx)^2. \tag{5}$$

giving

$$\chi_i = \frac{y - (A + Bx)}{\sqrt{dy^2 + (Adx)^2}} = \frac{y - g(x)}{\sqrt{dy^2 + (Adx)^2}}. \tag{6}$$

- Suppose we fit to an **exponential** function, $g(x) = Ae^{-x/B} + C$. In this case

$$df^2 = dy^2 + [(-Ae^{-x/B}/B)dx]^2 \tag{7}$$

giving

$$\chi_i = \frac{y - (Ae^{-x/B} + C)}{\sqrt{dy^2 + [(Ae^{-x/B}/B)dx]^2}} = \frac{y - g(x)}{\sqrt{dy^2 + [(Ae^{-x/B}/B)dx]^2}} \tag{8}$$

Let's now demonstrate how to use this more general form for the weighted residual in your python code. Consider the following snippet which defines a linear fit function and residual which takes $df_i = dy_i$:

```
def fitfunc(p, x):
    return p[0]*x + p[1]
def residual(p, x, y, yerr):
    return (y - fitfunc(p,x))/yerr
```

If we wish to modify this to incorporate the x-uncertainty, we must do two things. First, we must now pass the x-uncertainty to the residual as an additional argument. Second, we must modify the residual to use the full df_i weight. The code now reads as follows:

```
def fitfunc(p, x):
    return p[0]*x + p[1]
def residual(p, x, y, xerr, yerr):
    return (y - fitfunc(p, x))/numpy.sqrt(yerr**2 + (p[0] *xerr)**2)
```