LEAST-SQUARES AND CHI-SQUARE FITTING FOR THE BUDDING, AND ALSO THE POST-BUDDING, AFICIONADO
July 26, 2001

Carl Heiles

In our never-ending attempt to make your life easier, we present you with this highly instructive, time-saving, and labor-saving informative document! Here we give the derivations and, also, the prescription for doing least-squares the easy way using IDL. This prescription is given as an example in §5, and the budding-aficionado *power-user* can skip the details and go directly there. The sections after the numerical example are for the more advanced post-budding aficionado.

Anyone graduating from Berkeley who claims to understand least square fitting needs to visit §5, 6, 7, and 8. These become progressively more involved; §5 is enough to get started. §5 may be obscure without first going into the introductory material in §3 and 4.

Contents

1. LEAST-SQUARE FITTING FOR TWO PARAMETERS, AS WITH A STRAIGHT LINE.

1.1. The closed-form expressions in Taylor.

First consider the least squares fit to a straight line. Let θ_i be the i^{th} measurement of the observed quantity (e.g., the forward measurement of zenith distance angle); t_i be the time of the i^{th} measurement; N = the total number of observations, i.e. i runs from 1 to N. Remember that in the least-squares technique, quantities such as t_i are regarded to be known with high accuracy while the quantity θ_i has uncertainties in its measurement.

We expect the zenith distance θ_i to change linearly with time as follows:

$$A + Bt_i = \theta_i \; . \tag{1}$$

Given this, one looks at Taylor's equations (8.10) and (8.11) and sees how to solve for the unknowns A and B; these equations are the closed form of the solution for the pair of equations (8.8) and (8.9). Specifically, A and B are obtained by solving the pair of equations

$$AN + B\ \Sigma t_i = \Sigma\theta_i \tag{2a}$$

$$A\ \Sigma t_i + B\ \Sigma t_i^2 = \Sigma t_i\theta_i \tag{2b}$$

1.2. Better is the following generalized notation.

Taylor's closed-form expressions are simply longhand ways to write matrix math. We will express them as matrices below in §3. But to ease into this gently, we first will carry through an intermediate stage of notation.

First rewrite equation (1) in the more general form

$$As_i + Bt_i = \theta_i\ . \tag{3}$$

To correspond to equation (1), $s_i = 1$ for all i, but for a more general application s might be some parameter other than time that varies from one measurement to another. We have N equations like equation (2). They are known as the *equations of condition* because they are the equations that specify the theoretical model to which we are fitting the data.

There are N equations of condition and only two unknowns (A and B). This is too many equations! We have to end up with a system in which the number of equations is equal to the number of unknowns.

To accomplish this, from equation (2) we form the *normal equations.* The number of normal equations is equal to the number of unknowns, so in this case we will have two. To do this, first define the following notation:

$$[ss] = \sum_{i=0}^{N-1} s_i s_i\ , \tag{4a}$$

$$[st] = \sum_{i=0}^{N-1} s_i t_i\ , \tag{4b}$$

$$[s\theta] = \sum_{i=0}^{N-1} s_i \theta_i\ ,\ \text{etc.} \tag{4c}$$

Note: Above and from now on, we consider $i = 0 \rightarrow (N-1)$ instead of $i = 1 \rightarrow N$ for consistency with IDL's convention. Then the two normal equations are

$$[ss]A + [st]B = [s\theta] \; , \tag{5a}$$

$$[ts]A + [tt]B = [t\theta] \; . \tag{5b}$$

These are precisely analogous to Taylor's equations (8.8) and (8.9). You could write the solution in closed form, as in Taylor's equations (8.10) and (8.11).

2. LEAST-SQUARE FITTING FOR MANY PARAMETERS, AS WITH A CUBIC.

With this notation it's easy to generalize to more (M) unknowns: the method is obvious because in each equation of condition (like equation (2)) we simply add equivalent additional terms such as Cc_i, Dd_i, etc; and in the normal equations (equation 4) we have more products and also more normal equations.

Let's take an example with four unknowns ($M = 4$), which we will denote by A, B, C, D; this would be like fitting a cubic. With $M = 4$ we need at least five data points ($N = 5$), so there must be at least five equations of condition. The generalization of equation (2) is the N equations

$$As_i + Bt_i + Cu_i + Dv_i = \theta_i \; , \tag{6}$$

with $i = 0 \rightarrow (N-1)$. Again, the least squares fitting process assumes that the s_i, t_i, u_i, v_i are known with zero uncertainty; all of the uncertainties are in the measurements of θ_i. We then form the four normal equations; the generalization of equation (4) is:

$$\begin{bmatrix} [ss] & [st] & [su] & [sv] \\ [ts] & [tt] & [tu] & [tv] \\ [us] & [ut] & [uu] & [uv] \\ [vs] & [vt] & [vu] & [vv] \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} [s\theta] \\ [t\theta] \\ [u\theta] \\ [v\theta] \end{bmatrix} \tag{7}$$

Note that this equation is really a matrix equation, and that the $M \times M$ matrix on the left is symmetric; this means that you can actually *solve* for A, B, C, D!

3. FAR, FAR BEST AND EASIEST: MATRIX ALGEBRA.

The above equations are terribly cumbersome to solve in a computer code because they require lots of loops. However, it becomes trivial if we use matrices. Here we designate a **matrix** by **boldface** type.

We illustrate the matrix method by carrying through the above $M = 4$ example, and we assume that there are 5 independent measurements ($N = 5$). We first define the matrices

$$\mathbf{S} = \begin{bmatrix} s_0 & t_0 & u_0 & v_0 \\ s_1 & t_1 & u_1 & v_1 \\ s_2 & t_2 & u_2 & v_2 \\ s_3 & t_3 & u_3 & v_3 \\ s_4 & t_4 & u_4 & v_4 \end{bmatrix} \tag{8a}$$

$$\mathbf{A} = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} \tag{8b}$$

$$\mathbf{T} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix} \tag{8c}$$

so, in matrix form, the equations of condition (equation (5)) are simply the single matrix equation

$$\mathbf{S} \cdot \mathbf{A} = \mathbf{T} \,. \tag{9}$$

Again, there are more equations than unknowns so we can't solve this matrix equation directly. So next we form the normal equations from these matrices. In matrix form, the normal equations (equation (6)) reduce to the single equation

$$\mathbf{SS} \cdot \mathbf{A} = \mathbf{ST} \,, \tag{10}$$

where

$$\mathbf{SS} = \begin{bmatrix} [ss] & [st] & [su] & [sv] \\ [ts] & [tt] & [tu] & [tv] \\ [us] & [ut] & [uu] & [uv] \\ [vs] & [vt] & [vu] & [vv] \end{bmatrix} \tag{11a}$$

and

$$\mathbf{ST} = \begin{bmatrix} [s\theta] \\ [t\theta] \\ [u\theta] \\ [v\theta] \end{bmatrix} \tag{11b}$$

Now the number of equations is equal to the number of unknowns, so the solution of the matrix equation is easy (in IDL, that is).

In IDL, we generate the matrix **SS** by

$$\mathbf{SS} = \mathbf{transpose(S)\#\#S} \tag{12a}$$

and the matrix **ST** by

$$\mathbf{ST} = \mathbf{transpose(S)\#\#transpose(T)}\ . \tag{12b}$$

In IDL, ## is the matrix multiplication operator.[1]

To solve the matrix equation (9) for **A**, first rewrite it by multiplying both sides by the inverse of **SS** (that is, by $\mathbf{SS}^{-1}$); this gives

$$\mathbf{A} = \mathbf{SS^{-1}} \cdot \mathbf{ST}\ . \tag{13}$$

To compute $\mathbf{SS^{-1}}$ in IDL we use IDL's invert function and we multiply matrices by typing ##. Thus, in IDL, first calculate the inverse matrix $\mathbf{SS^{-1}}$, which we denote by **SSI**:

$$\mathbf{SSI} = \mathbf{invert(SS)}\ , \tag{14}$$

[1]In equation 12b we could have used **T** instead of **transpose T**; however, in the unusual case in which you are solving for a single unknown, **transpose T** works and **T** doesn't.

and then carry out the multiplication in equation 13:

$$\mathbf{A} = \mathbf{SSI} \ \#\# \ \mathbf{ST} \ . \tag{15}$$

4. UNCERTAINTIES IN THE DERIVED COEFFICIENTS.

How about the uncertainties in the derived quantities contained in the matrix $\mathbf{A}$?

The first thing to do is derive the variance σ^2 (the square of standard deviation, or mean error, or dispersion, etc) of the individual data points using the generalization of Taylor's equation (8.14). The generalization is, simply, to replace the $N-2$ in the denominator by $N-M$. For example, with the fit to a linear line there are 2 parameters, Taylor's A and B, so $M = 2$ and the denominator is $N-2$. So the generalization is

$$\sigma^2 = \frac{1}{N-M} \sum_{i=0}^{N-1} (\theta_i - \bar{\theta}_i)^2 \ , \tag{16}$$

where $\bar{\theta}_i$ are the values for θ_i *predicted by the derived quantities.* Note the difference: θ_i are the *observed* values, while $\bar{\theta}_i$ are the values *predicted by the least squares fit.* The predicted values are those that are computed from the derived coefficients $A, B, C \ldots$ The N quantities $(\theta_i - \bar{\theta}_i)$ are called the *residuals* or *deviations* from the fit (Taylor, page 98).

It's easy to calculate the $\bar{\theta}_i$ with matrices. First define the matrix $\bar{\mathbf{T}}$ that contains these values:

$$\bar{\mathbf{T}} = \begin{bmatrix} \bar{\theta}_0 \\ \bar{\theta}_1 \\ \bar{\theta}_2 \\ \bar{\theta}_3 \\ \bar{\theta}_4 \end{bmatrix} \tag{17}$$

Calculating $\bar{\mathbf{T}}$ is simple:

$$\bar{\mathbf{T}} = \mathbf{S} \cdot \mathbf{A} \ . \tag{18}$$

Note that $\mathbf{S}$ is already defined (equation 7a) and $\mathbf{A}$ was solved for in equation (12) and (14).

Let's do all this in IDL. To do equation (17) in IDL (in IDL we designate $\bar{\mathbf{T}}$ by $\mathbf{BT}$):

$$\mathbf{BT} = \mathbf{S} \ \#\# \ \mathbf{A} \ . \tag{19}$$

To get σ^2 (equation (15)) in IDL:

$$sigsq = \mathbf{total}((\mathbf{T} - \mathbf{BT}) \wedge 2)/(N - M) \ . \tag{20}$$

$(\mathbf{T} - \mathbf{BT})$ is the array of residuals; it is usually a good idea to plot all three quantities $\mathbf{T}$, $\mathbf{BT}$, and $(\mathbf{T} - \mathbf{BT})$ to make sure your fit looks reasonable and to check for bad data points.

OK, now the error of the derived coefficients is given by the analogy to Taylor's equations (8.15) and (8.16). In IDL, let's designate the array that contains the squares of errors of the N derived coefficients by **sigdcsq** (think of "**sig**ma of **d**erived **c**oefficients **sq**uared"); you get this from the diagonal elements of **SSI**, whose indices are given in IDL by[2]

$$\mathbf{sigdcsq} = sigsq * \mathbf{SSI}[(\mathbf{M} + \mathbf{1}) * \mathbf{indgen}(\mathbf{M})] \ . \tag{21}$$

Or, to put it simply in words: to get the square of the "error of coefficient j" in the matrix $\mathbf{A}$, multiply σ^2 (that's the single number $sigsq$ in IDL) by the j^{th} diagonal element of the inverse matrix $\mathbf{SS^{-1}}$. And it should be obvious that the standard deviation of the coefficients is just $sqrt(\mathbf{sigdcsq})$.

Although the above equation is correct, there is more to the story because of covariance. We defer its discussion until the numerical example.

5. A NUMERICAL EXAMPLE AND ITS SOLUTION IN IDL.

If the following sounds like Greek to you, take a look at §3 and 4.

(1) First, generate the numerical example. Suppose that we make four measurements of the angle θ and we want to fit to a parabolic function in time. In the notation of equation 6, s would be unity, t the time, and u the time squared, so the number of unknowns is three ($M = 3$). Because there are four independent measurements ($N = 4$) the subscripts run from $i = 0 \rightarrow 3$. Suppose that the four values of time are 5, 7, 9, 11.

First we create the matrix **S** in IDL

$$\mathbf{S} = \mathbf{fltarr}(\mathbf{M}, \mathbf{N}) = \mathbf{fltarr}(\mathbf{3}, \mathbf{4}) \tag{22}$$

and then we populate it with numbers. In your own work, you would normally do this by reading a data file and transferring the numbers to the matrix using IDL commands; to work through this

[2]Specifically, for any M-sized square array **Q** the vector of its diagonal elements is given by **Q[(M+1)*indgen(M)]**.

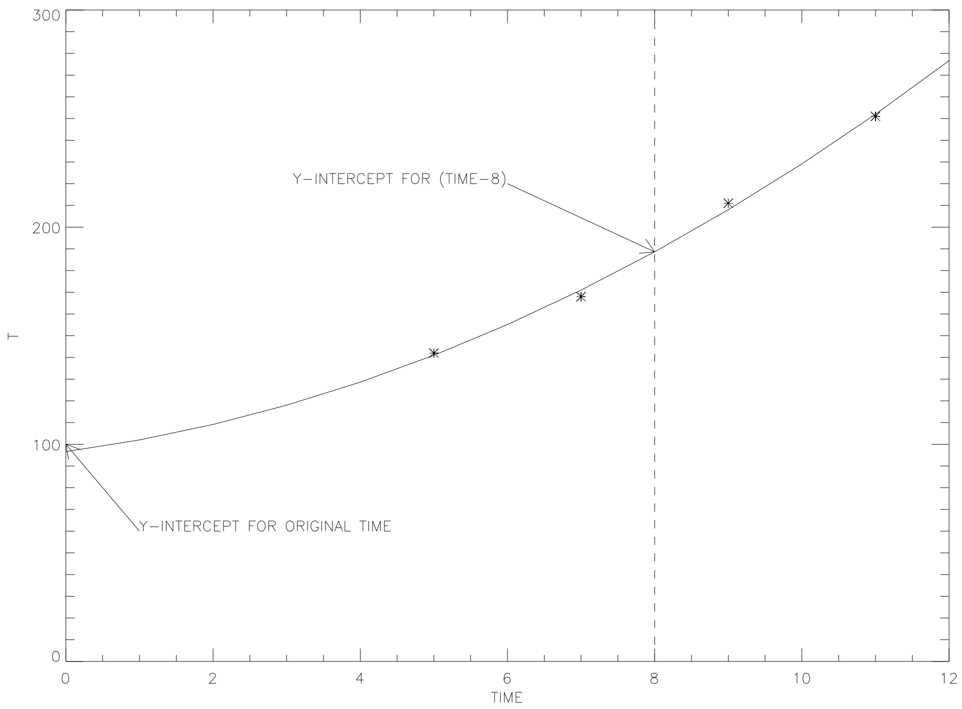

Fig. 1.— Our numerical example. Stars are the four datapoints; the solid line is the fit. We perform two fits: one uses the original definition of time; the other uses $(time - 8)$, in effect moving the y-axis to the dashed line. The two fits give the same line but the coefficients and their errors differ greatly.

example, you might manually type them in. After populating the matrix, in direct correspondence to equation 8a we have $s_i = 1$, $t_i = time_i$, $u_i = time_i^2$:

$$\mathbf{S} = \begin{bmatrix} 1 & 5 & 25 \\ 1 & 7 & 49 \\ 1 & 9 & 81 \\ 1 & 11 & 121 \end{bmatrix} \tag{23a}$$

and suppose that four measured values of θ are (equation 8c)

$$\mathbf{T} = \begin{bmatrix} 142 \\ 168 \\ 211 \\ 251 \end{bmatrix} . \tag{24a}$$

Figure 1 shows the datapoints, together with the fitted curve.

(2) Now solve it in IDL. In IDL we calculate the normal equation matrices (equations 11a, 12a):

$$\mathbf{SS} = \mathbf{transpose(S)\#\#S}\ , \tag{25a}$$

and (equations 11b, 12b)

$$\mathbf{ST} = \mathbf{transpose(S)\#\#transpose(T)}\ . \tag{25b}$$

In IDL we take the inverse of **SS** (equation 14):

$$\mathbf{SSI} = \mathbf{invert(SS)}\ . \tag{26}$$

The least-square fitted quantities are in the matrix **A** (equations 13, 15), which we obtain in IDL with

$$\mathbf{A} = \mathbf{SSI}\ \#\#\ \mathbf{ST}\ . \tag{27}$$

To get the uncertainties, follow the prescription in §4. More specifically, the IDL statements of equations 19, 20, and 21 provide the matrix of squares of the uncertainties, **sigdcsq**.

For this numerical example, the solution (the array of derived coefficients) is

$$\mathbf{A} = \begin{bmatrix} 96.6250 \\ 4.49805 \\ 0.875122 \end{bmatrix} \tag{28a}$$

and the errors in the derived coefficients [the square root of the σ^2's of the derived coefficients, i.e. $(\sigma_j^2)^{1/2}$ or, in IDL, *sqrt*(**sigdcsq**) in equation 21] are:

$$\sigma_{\mathbf{A}} = \begin{bmatrix} 34.0118 \\ 8.99995 \\ 0.559014 \end{bmatrix}\ . \tag{28b}$$

These results look *horrible*: the uncertainties are large fractions of the derived coefficients,

The reason: we have specifically chosen an example with terrible covariance. And the great thing is this can be fixed easily (at least in this case—not always), without taking more data! Read all about this in the next section!

6. THE COVARIANCE MATRIX AND ITS NORMALIZED COUNTERPART

First we provide a general discussion, then we apply it to the above numerical example.

6.1. General discussion

The uncertainties in the derived coefficients are obtained from the diagonal elements of **SSI**. What do the *off-diagonal* elements represent?

The off-diagonal elements represent the *covariance* between the derived coefficients. Covariance means, specifically, the degree to which the *uncertainty* in *one* derived coefficient affects the uncertainty in *another* derived coefficient. Because the covariance matrix elements relate pairwise the various coefficients, the units of the matrix elements are all different. This makes it convenient to reduce all the matrix elements to a standard set of units—namely, no units at all. So before discussing the covariance matrix *per se*, we first discuss its normalized counterpart.

The normalized covariance matrix[3] **ncov** is derived from **SSI** by dividing each element by the square root of the product of the corresponding diagonal elements. Let **ncov** be the normalized covariance matrix; then

$$ncov_{ik} = \frac{SSI_{ik}}{\sqrt{SSI_{ii}\ SSI_{kk}}} \tag{29}$$

Or, in IDL, do the following:

$$\mathbf{dc} = \mathbf{SSI}[(\mathbf{M}+\mathbf{1}) * \mathbf{indgen}(\mathbf{M})] \tag{30a}$$

$$\mathbf{ncov} = \mathbf{SSI}/sqrt(\mathbf{dc}\#\mathbf{dc}) \tag{30b}$$

In the above, **dc#dc** is an $M \times M$ matrix consisting of products of the diagonals of **SSI**, so dividing **SSI** by *sqrt*(**dc#dc**) generates the normalized version.

[3]It is a pleasure to thank Doug Finkbiener for introducing me to this concept.

Because **ncov** is a *normalized* covariance matrix, you might think that it's non-normalized parent is **SSI**—and you'd be *almost* right. The true covariance matrix **C** (as defined in *Numerical Recipes* for example) is

$$\mathbf{C} = \sigma_{data}^2 \mathbf{SSI} \tag{31}$$

because **C** is defined for χ^2 instead of σ^2 (see §7).

In **ncov**, the diagonal elements are all unity and the off-diagonal elements reflect the interdependence of the derived coefficients on each other. The off-diagonal elements can range from $-1 \rightarrow 1$. Each matrix element is the correlation coefficient between the uncertainty of its two parameters. In particular, suppose that the data happens to produce a coefficient that differs from its true value by some positive number. If the normalized matrix element is negative, then the other coefficient will tend to differ from its true value by a negative number.

Here's a more detailed discussion of what the covariance means. Suppose you are least-squares fitting for two derived coefficients (A_0 and A_1). When you do a least-squares fit to a set of data, you are fitting one set of data out of a possible infinity of possible sets that you'd get by repeating the experiment, and your particular set of data happens to produce specific values of A_0 and A_1, which differ from the *true* values by amounts δA_0 and δA_1. If their covariance is zero, then in the infinity of data sets you'd find that δA_0 is uncorrelated with δA_1. But if it is nonzero, these two quantities would be correlated.

A high covariance is bad because the derived variables depend on each other. As we shall see in the next subsection, it also can greatly increase the uncertainty in derived coefficients. Often a high covariance results from a poor choice of functions that you are fitting or even a bad choice of the zero point of the independent variable—as in our numerical example (see the next subsection). And, as in that example, you can often (not always!) eliminate the bad covariance by reformulating the problem—you don't even need to take more data! The best reformulation involves using a set of orthonormal functions.

6.2. The covariance in our numerical example

Apply equation 30 to obtain the covariance matrix for our numerical example:

$$\mathbf{ncov} = \begin{bmatrix} 1 & -.989848 & .969717 \\ -.989848 & 1 & -.993808 \\ .969717 & -.993808 & 1 \end{bmatrix} . \tag{32}$$

The off-diagonal elements are *huge*. This is the reason why our solution is so bad.

In this seemingly innocuous example we have an excellent case of a poor choice of zero point for the independent variable (the time). The reason is clear upon a bit of reflection. We are fitting for $y = A_0 + A_1 t + A_2 t^2$. The coefficient A_0 is the y-intercept and A_1 is the slope. Inspection of Figure 1 makes it very clear that an error in the slope has a big effect on the y-intercept.

Now we retry the example, making the zero point of the time equal to the mean of all the times, that is we set $(time_i = time_i - 8)$. We get the same fitted line, but the derived coefficients are completely different—and amazingly better! We get

$$\mathbf{A} = \begin{bmatrix} 188.625 \\ 18.500 \\ 0.87500 \end{bmatrix} \tag{33a}$$

$$\sigma_{\mathbf{A}} = \begin{bmatrix} 3.58 \\ 1.00 \\ 0.559 \end{bmatrix} . \tag{33b}$$

In redefining the origin of the independent variable, we have made the zero intercept completely independent of the slope: changing the slope has no affect at all on the intercept. We are taking a partial step towards fitting coefficients of the orthogonal set of Legendre polynomials—it's always a good idea to fit orthogonal functions if you can express the problem in those terms. Our step is *partial*: the second-order coefficient A_2 affects A_0 because, over the range of $[(time - 8) = -3 \rightarrow +3]$, the quantity $[A_2 \, \Sigma(time_i - 8)]$ is nonzero and thereby has the same effect as A_0.

For further discussion of covariance, see §7.4. Also, you might also want to try out another example in Taylor's §8.5.

7. CHI-SQUARE FITTING, AND WEIGHTED FITTING

Chi-square fitting is very much like our least-squares fitting above, with one additional wrinkle. Chi-square (χ^2) is equal to the ratio of the variance of the *fit* (σ^2) to the intrinsic variances of the datapoints (σ^2_{data}). So it should be obvious that In Chi-square fitting, you must know the measurement uncertainties of the individual data points beforehand. If you want to give the various datapoints different weights, then that is just like Chi-square fitting except that you can adopt an arbitrary scale factor for the uncertainties (section 7.2).

7.1. The Chi-square when all datapoints have identical dispersions

Suppose, first, that all datapoints have the same intrinsic measurement uncertainty σ_{data}.[4] This is a function *only* of the *instrument* and the *data-taking procedure*: if you know the properties of the instrument, and if you understand how it works, and if there are no systematic errors, then you should be able to specify σ_{data}. Specifically, σ_{data} is most certainly *not* the same thing as the dispersion of the points from the least-squares fit.

This case, in which all datapoints have the same intrinsic measurement uncertainty σ_{data}, occurs frequently, for example when the datapoints are taken under identical conditions. Least-squares assumes a Gaussian distribution of datapoints around the true value, which is characterized by the dispersion σ_{data}. In this case, Chi-square fitting is just like least-squares fitting except for the following:

(1) Divide each equation of condition by σ_{data}. You are generating a new matrix $\mathbf{S_X}$, which is identical to $\mathbf{S}$ except that each row is divided by σ_{data}. This new matrix is the same as NR's *design matrix.*

(2) Divide each datapoint by σ_{data}. You are generating a new matrix $\mathbf{T_X}$, which is identical to $\mathbf{T}$ except that each row is divided by σ_{data}. This new matrix is the same as NR's vector $\mathbf{b}$.

(3) In our equation 20, obtain the *reduced* observed chi-square $\widehat{\chi_{obs}}^2$ by replacing $\mathbf{T}$ by $\mathbf{T_X}$ and $\mathbf{BT}$ by $\mathbf{BT_X} = \mathbf{S_X}\#\#\mathbf{A_X}$ (cf equation 19). That is, in IDL

$$\widehat{\chi^2_{obs}} = \mathbf{total}((\mathbf{T_X} - \mathbf{BT_X}) \wedge 2)/(N - M) \ . \tag{34}$$

The reduced chi-square $\widehat{\chi_{obs}}^2$ is equal to the ordinary chi-square χ^2_{obs} except that it is divided by the number of degrees of freedom, ordinarily denoted by ν, which is equal to $(N - M)$.

Look at the effect of all these three steps somewhat carefully. In steps **(1)** and **(2)**, you've taken our entire matrix equation 9 and divided both sides by the same quantity, σ_{data}. Obviously, this doesn't change the derived results for $\mathbf{A}$—as it shouldn't. So $\mathbf{A_X} = \mathbf{A}$. However, in equation 34 the vectors $\mathbf{T_X}$ and $\mathbf{BT_X}$ are multiplied by the factor $\frac{1}{\sigma_{data}}$, so equation 34 calculates $\widehat{\chi^2_{obs}} = \frac{\sigma^2}{\sigma^2_{data}}$.

This is exactly what we *should* get! Suppose, for example, that the least-squares fit model is perfect and the only deviations from the fitted curve result from measurement error. Then by necessity we have $\sigma = \sigma_{data}$ and $\widehat{\chi^2_{obs}} = 1$.

In the case $\widehat{\chi^2_{obs}} = 1$ the dispersion of the observed points σ is equal to the intrinsic dispersion of the datapoints σ_{data} and the mathematical model embodied in the least-squares fit is perfect. That, at least, is the *theoretical* conclusion. In practice, however, you're obtaining such a low, good

[4] Our discussion follows *Numerical Recipes* §14.4.3, *Solution by Use of the Normal Equations*, which we denote by NR.

value for χ^2 might mean instead that you are using *too large* a value for σ_{data}: you are ascribing more error to your datapoints than they really have, perhaps by not putting enough faith in your instrument. Specifically, if the datapoints really have intrinsic measurement dispersion σ_{data} and you specify that they have $2\sigma_{data}$ when computing $(\mathbf{S_X}, \mathbf{T_X})$, then you'll find $\widehat{\chi^2_{obs}} = 0.25$.

High values of $\widehat{\chi^2_{obs}}$ indicate that the model is not perfect and could be improved by the use of a different model, such as the addition of more parameters—or, alternatively, that you think your equipment works better than it really does and you are ascribing *less* error to your datapoints than they really have.

We don't discuss the interpretation of χ^2 here, because that is done very adequately in textbooks on statistics. In particular, we recommend Taylor's chapter on Chi-square (§12), which provides a nice introduction and discussion.

7.2. The case in which datapoints have different dispersions, or different weights: like a weighted average

In this case, each row of the equation-of-condition matrix $\mathbf{S}$ and each element of the data vector $\mathbf{T}$ get divided by their corresponding σ_{data}. The equation embodied in each row of the matrix equation 9 remains unchanged, but the different rows are weighted differently with respect to each other.

Consider two measurements with intrinsic measurement uncertainties (σ_1, σ_2); suppose $\sigma_1 < \sigma_2$. After being divided by their respective σ_{data}'s, all of the numbers in row 1 are larger than those in row 2. In all subsequent matrix operations, these larger numbers contribute more to all of the matrix-element products and sums. Thus, the measurement with smaller uncertainty has more influence on the final result, as it should.

Suppose that the above two measurements were taken under identical conditions except that measurement 1 received more integration time than measurement 2; we have $\frac{\sigma_1}{\sigma_2} = \left(\frac{t_1}{t_2}\right)^{-1/2}$, so the rows of $\mathbf{S_X}$ are weighted as $t^{1/2}$. This means that during the computation of $\mathbf{SS_X}$, the self-products of row 1 are weighted as t (and each row multiplies *only* itself; see equations 4 and 7). This means that each datapoint is weighted as t, which is exactly what you'd expect! Note that this is also exactly the same weighting scheme used in a weighted average, in which the weights are proportional to $\left(\frac{1}{\sigma_{data}}\right)^2$. We conclude that weighting scheme of the three steps in section 7.1 agrees with common sense.

7.3. Relative, but not absolute, weights

Suppose you don't know the intrinsic measurement dispersion σ_{data}, but you *do* know the *relative* dispersion of the various measurements. For example, this would be the case if the datapoints were taken under identical conditions except for integration time; then $\sigma_{data} \propto t^{-1/2}$. This case is neither fish nor fowl: that is, it is neither the case in which you obtain the dispersions directly nor is it the chi-squares case. It needs a somewhat special discussion, which I obtained by considering the case of a weighted average and then extending it to the case of a multiparameter least squares fit.

As a preliminary, we remind you of the formulae for an ordinary *unweighted* average in which the value of each point is y_i and the residual of each point from the weighted mean is r_i; we also neglect the difference between N and $N-1$ to make the comparison to the weighted case clearer:

$$mean = \frac{\sum y_i}{N} \tag{35a}$$

$$\sigma^2 = \frac{\sum r_i^2}{N} \tag{35b}$$

$$\sigma_{mean}^2 = \frac{\sum r_i^2}{N^2} = \frac{\sigma^2}{N} \ , \tag{35c}$$

σ_{mean} is the error of the mean. Recall that in this case the mean is the least-squares fit to the data, so to use least-squares jargon we can also describe σ_{mean} as the error in the derived coefficient for this single-parameter least-squares fit.

Now for a weighted average in which the weight of each point is w_i. In a nonweighted average the quantity that is minimized is $\sum r_i^2$; in a weighted average the quantity minimized is $\sum (w_i r_i)^2$. So the three equations corresponding to the above become

$$mean_w = \frac{\sum w_i y_i}{\sum w_i} \tag{36a}$$

$$\sigma_w^2 = \frac{\sum w_i^2 r_i^2}{\sum w_i^2} \tag{36b}$$

$$\sigma_{mean}^2 = \frac{\sum w_i^4 r_i^2}{(\sum w_i^2)^2} \ , \tag{36c}$$

The last equation looks strange, but is straightforward to derive by the usual minimization procedure. Let's do a reality check. Note that taking all $w_i = 1$ reduces to the nonweighted case, as it should. Moreover, taking all w_i equal to *any* constant reduces to the nonweighted case.

However, for the general case when the w_i are all different. there is a *crucial difference* between equations 35c and 36c. It's the surprising presence of w^4 in equation 36c! This means that the variance of the mean σ^2_{mean} *cannot* be expressed in terms of σ^2_w!!

These differences carry over into the general least squares case for this case of arbitrary weights. There are three crucial items:

(1) Multiply each row of the equation-of-condition matrix $\mathbf{S}$ and each element of the data vector $\mathbf{T}$ by the corresponding weight w_i. This is just what you do with chi-square fitting.

(2) Calculate the weighted dispersion according to equation 36b. In IDL or any matrix formulation, this means that you use the equivalent of equation 34

$$sigsq = \mathbf{total}((\mathbf{T_X} - \mathbf{BT_X}) \wedge 2)/\mathbf{total}(\mathrm{w} \wedge 2)\ . \tag{37}$$

i.e., you don't include the weights in the residuals)$\mathbf{T_X} - \mathbf{BT_X}$) because they already have the weights included.

(3) Calclatate a modified $sigsq_{mod}$ using equation 36c. To be explicit:

$$sigsq_{mod} = \mathbf{total}(\mathbf{w^2} * (\mathbf{T_X} - \mathbf{BT_X}) \wedge 2)/\mathbf{total}(\mathrm{w} \wedge 2) \wedge 2\ . \tag{38}$$

Finally, use this modified $sigsq_{mod}$ instead of $sigsq$ in equation 21.

7.4. The uncertainties of the derived coefficients

The above two subsections deal with the Chi-square of the fitted points, and provide information about how well the model fits the data. It says nothing about the uncertainties in the derived coefficients. To get this[5], you re-express equation 21 in terms of the new parameters, namely

$$\mathbf{sigdcsq} = \widehat{\chi^2_{obs}} * \mathbf{SSI_X}[(\mathbf{M+1}) * \mathbf{indgen}(\mathbf{M})]\ , \tag{39}$$

where $\widehat{\chi^2_{obs}}$ is from equation 34. However, *this is most definitely not the whole story* because you are doing a multiparameter fit, and generally speaking the derived parameters have nonzero covariance.

Consider the example of the first two coefficients in our above example, which we discussed a bit above in §6.2. In this example, the fit gives $y = A_0 + A_1t + A_2t^2$, where the numerical values are given in vector form by equation 28. The coefficient A_0 is the y-intercept and A_1 is the slope. They have derived values $A_0 = 96 \pm 34$ and $A_1 = 4 \pm 9$.

[5]Our discussion follows *Numerical Recipes* §14.5, the first three italicized subsections, which we denote by NR.

Remember what these uncertainties really mean: in an infinity of similar experiments, you'll obtain an infinity of values of (A_0, A_1) that are normally distributed with dispersions (34,9). Loosely speaking, this means that A_0 lies between $(96 - 34 = 62)$ and $(96 + 34 = 130)$ and A_1 lies between -4.5 and 13.5. If you are interested in knowing about A_0 *without regard to* A_1, then its uncertainty is indeed 96; ditto for A_1. In other words, equations 21 and 39 apply. Thus, if you don't care about A_1, then you can conclude $A_0 = 130$; and if you don't care about A_0, then you can conclude $A_1 = 13.5$ (both with 68.3% probability).

However, if you are interested in knowing about *both*, you must include their covariance. In our example, the large negative covariance follows logically just from looking at a graph: if you fit some points, all of which lie at positive t, then a more negative derived slope will raise the y-intercept.

Specifically, the large negative covariance means that positive departures of A_0 are associated with negative departures of A_1. So even though the *individual* values $A_0 = 130$ and $A_1 = 13.5$ are acceptable, you *cannot* conclude that the *pair* of values $(A_0, A_1) = (130, 13.5)$ is acceptable, because this pair has large positive departures of both. In contrast, what *is* acceptable here would be something like $(A_0, A_1) = (130, -4.5)$.

We stress that the acceptable ranges of values depend on what you are interested in. This is sort of like the observer's influence in quantum mechanics. If you are interested in A_1 alone, then you can say $A_1 = 4 \pm 9$ and, in making this statement, you have to realize that, as A_1 varies over this range, A_0 can vary over (formally, at least) the range $(\infty \rightarrow -\infty)$: you just don't give a damn *what* happens to A_0 because you're not interested. But the moment you become interested and restrict its possible range, that influences the possible range for A_1, too.

There is no simple relationship between the covariance matrix elements and the acceptable ranges. For two variables, the best way to express this matters is to construct the ellipses that define the loci of constant $\Delta\chi^2$ and present them on a graph as in NR's Figure 14.5.4. For three variables, these ellipses become ellipsoids; for four, they become four-dimensional volumes, etc.

To construct these curves/surfaces/volumes/etc., follow the prescription defined by the six bullets at the end of the section NR's *Probability Distribution of Parameters in the Normal Case.* This description is a bit curt, but it is accurate. To make sure that you understand it, take a numerical example and try applying the prescription to the case of deriving the uncertainties of a single coefficient (the case $\nu = 1$); that case is the case covered by equations 21 and 39 above. In this case, you have the usual normal distribution function for that variable, in which its formal error (as defined in equations 21 and 39 above) defines a range within which the derived value will fall 68.3% of the time in the infinity of experiments—it's just like any other normal distribution.

8. NONLINEAR LEAST SQUARES

The least squares formulation requires that the data values depend *linearly* on the unknown coefficients. For example, in equation 1, the unknown coefficients A and B enter linearly.

Suppose one had a nonlinear dependence, such as for example

$$\sin(At_i) + Bt_i = \theta_i \ . \tag{40}$$

What do you do here? You linearize the equation, using the following procedure.

First, assume a trial values for A and B; call these A_0 and B_0. You should pick values that are close to the correct ones. In our example you wouldn't need to do this for B, but it's easier to treat all coefficients identically. These trial values produce *predicted* values $\theta_{0,i}$:

$$\sin(A_0 t_i) + B_0 t_i = \theta_{0,i} \ . \tag{41}$$

Subtract equation 41 from 40, and express the differences as derivatives. Letting $\delta A = A - A_0$ and $\delta B = B - B_0$, this gives

$$\delta A[t_i \cos(A_0 t_i)] + \delta B t_i = \theta_i - \theta_{0,i} \ . \tag{42}$$

Now solve for δA and ΔB. These won't be exact because higher derivatives (including cross derivatives) come into play, so you need to repeat the process. This is an iterative procedure and you keep going until the changes become "small". The generalization to an arbitrarily large number of unknown coefficients is obvious.

We now offer some cautionary and practical remarks.

(1): Multiple minima: Nonlinear problems often have multiple minima in σ^2. A classical case is fitting multiple Gaussians to a spectral line profile. Gaussians are most definitely not orthogonal functions and in some cases several solutions may give almost comparably good values of σ^2, each one being a local minimum. For example, for the case of two blended Gaussians, one can often fit two narrow Gaussians or the combination of a wide and narrow Gaussian, the two fits giving almost equal σ^2. The lower of these is the real minimum but, given the existence of systematic errors and such, not necessarily the best solution. The best solution is often determined by physical considerations; in this case, for example, you might have physical reasons to fit a broad plus narrow Gaussian, so you'd choose this one even if it's σ^2 weren't the true minimum.

(2): The Initial Guess: When there are multiple minima, the one to which the solution converges is influenced by your initial guess. To fully understand the range of possible solutions, you should try different initial guesses and see what happens. If the solutions always converge to the same answer, then you can have some confidence (but not *full* confidence) that the solution is unique.

(3): Iterative stability: If your initial guess is too far from the true solution, then the existence of higher derivatives means that the computed corrections can be too large and drive the iterative solution into instability. It is often a good idea to multiiply the derived correction factors (δA and δB above) by a factor $\mathcal{F}$ less than unity, for example $\mathcal{F} = 0.5$ or 0.75. This increases the number of iterations required for convergence but often allows convergence instead of producing instability.

(4): Convergence criteria: How do you know when the solution has converged? One way: for each iteration, calculate the uncertainties in the derived coefficients. If the uncertainty exceeds the correction, then you are getting close. An alternate way, which I usually use: if the fractional correction (e.g. $\frac{\delta A}{A_0}$) decreases below some threshold, say 1%, you're close (some parameters, such as angles, need a threshold that is absolute instead of fractional). At this point, if you are using $\mathcal{F} \neq 1$, set $\mathcal{F} = 1$, do a few more iterations, and you're done.

(5): Numerical derivatives: Sometimes the equations of condition are so complicated that taking the derivatives, as in obtaining equation 42, is a huge job and subject to mistakes. So you can take numerical derivatives instead of analytic ones. Be careful, though; it's safest to use double precision and think a bit about numerical accuracy.

(6): Canned nonlinear least squares: Packages like IDL offer canned nonlinear least squares routines. They are designed to work well for a wide range of different problems. However, often you can do better by tailoring things (such as the factor $\mathcal{F}$ and convergence criteria above) for the specific problem at hand. A good example is Gaussian fitting: IDL's fitting program doesn't converge for some input data, while for many of these cases the program that I wrote myself works fine.

(7): Be careful and LOOK at the solution before accepting it! These nonlinear problems can produce surprising results, sometimes completely meaningless results. Don't rely on them to be automatic or foolproof!

9. BRUTE FORCE LEAST SQUARES AND THE CURVATURE MATRIX

9.1. Parameter Uncertainties in Brute Force Least Squares for a Single Variable

There are times when "brute" least squares is appropriate. For example, if you have a nonlinear problem in which taking derivatives is complicated, and if the number of unknown coefficients is small, then it might be easier to search through the coefficient parameter space, calculate the σ^2 for each combination of parameters, and find the minimum. That procedure certainly provides the least squares solution. It is particularly easy if there is only one coefficient. Let us consider this case.

How do you determine the uncertainties in the derived coefficient with this brute force technique? Consider the case of a simple average (which is the least squares solution for a constant). Suppose the least-squares value for the unknown coefficient (i.e., the average) is x_0 and that its uncertainty is Δx_0. Now consider offsets from x_0, namely $x_0 + \delta x$. It is easy to show that

$$\sigma_{\delta x}^2 = \sigma_{x_0}^2 + \delta x^2 \tag{43}$$

This is a parabola. Intuitively, one expects that the "narrower" the parabola, i.e. the smaller its curvature at x_0, the smaller Δx_0. The curvature $\mathcal{C}$ is just half the second derivative of this parabola evaluated at x_0, i.e.

$$\mathcal{C} = \frac{1}{2}\frac{d^2\sigma_{\delta x}^2}{d\delta x^2} \tag{44}$$

In this particular case, $\mathcal{C} = 1$ and *the curvature is independent of the values of the datapoints!* This independence of the data *values* is *generally the case.* Generally, $\mathcal{C}$ depends on the *locations* of the datapoints (the ensemble of t_i in equation 1) but not on the *measured values* (the ensemble of t_i in equation 1); you can see this because the curvature is proportional to **SS**, whose elements depend only on the locations.

But it is obvious that the data values affect Δx_0. In fact, from our experience with least squares, we expect $\Delta x_0 \propto \frac{\sigma_{x_0}}{\sqrt{N}}$, where N is the number of datapoints. So we expect something like the following, which is, in fact, the correct expression:

$$\Delta x_0^2 = \frac{\sigma_{x_0}^2}{N\mathcal{C}} \tag{45}$$

9.2. The Curvature Matrix

One can show that the diagonal elements of our matrix **SS** are the curvatures, i.e. $\mathcal{C}_n = \frac{\mathbf{SS}_{nn}}{N}$. This makes us think that we might denote **SS** by the moniker *curvature matrix.* This is close to the usual definition, for example in *Numerical Recipes*, where their curvature matrix is $\alpha = \frac{\mathbf{SS}}{\sigma_{data}^2}$. This makes $\mathcal{C}_n = \frac{\sigma_{data}^2 \alpha_{nn}}{N}$.

If there were only a single variable, or if **SS** were diagonal, then the diagonal elements of its inverse are just the reciprocals. In other words, for this special case, $\mathbf{SSI_{nn}} = \mathbf{SS_{nn}^{-1}}$ and $\mathcal{C}_n = \frac{1}{N\ SSI_{nn}}$; also, equation 45 holds. More generally, *for the particular case* in which **SS** is diagonal, then the undertainty in derived coefficient n is

$$\sigma_n^2 = \frac{\sigma^2}{N\mathcal{C}_n} \tag{46}$$

However, this is *not* true for more than one coefficient and a nondiagonal covariance matrix. In that case, I *think* it's true that $\mathbf{SSI_{nn}} > \mathbf{SS_{nn}^{-1}}$; this means that $\mathcal{C}_n > \frac{1}{N\ SSI_{nn}}$. Thus, equation 46 *under*estimates the error so that

$$\sigma_n^2 > \frac{\sigma^2}{N\mathcal{C}_n} \; . \tag{47}$$

Fundamentally this is because the off-diagonal terms add uncertainty to the derived coefficient because of the correlations among the coefficient uncertanties.

To derive the coefficient error in brute force least squares, you can use the curvature and equation 45 if you have only a single variable. Otherwise, you need to evaluate the covariance matrix; if it is diagonal, you could use equation 47; but having already evaluated the covariance matrix, you might as well do the problem using the usual least squares technique.

10. YOU WANT MORE? YOU GOT IT!

10.1. The true expert would...

In our numerical example, a true expert (there aren't many of these, even among professional astronomers) would often do something more involved, namely to fit an orthogonal set of polynomials (i.e., in this case the first three terms of Legendre polynomials) instead of just a cubic. Because this set is orthogonal, the off-diagonal elements of **ncov** would automatically equal zero.

10.2. Comparison with Numerical Recipes

There are two "bibles" of least-square fitting. Most important is Press *et al.*'s *Numerical Recipes*, which provides a wide-ranging treatment of computational techniques. Our treatment follows theirs quite closely (with one important exception—see next paragraph), but I wrote this before knowing of their treatment so didn't use their notation.

The exception: their approach is based on χ^2, while ours is based on σ^2. Their approach is more flexible because σ_{data} can vary from one measurement to another; ours implicitly assumes σ_{data} is the same for all measurements. They describe our approach in the discussion of equation (14.1.6) and warn that it is "dangerous" because you aren't comparing the residuals to the intrinsic inaccuracies of the data. In astronomy, though, more often than not you don't have an independent assessment of σ_{data} and our approach is the only option. In any case, our matrices correspond to theirs pairwise as follows (left of the double arrow is ours, to the right is theirs):

$$\mathbf{S} \longleftrightarrow \sigma_{data}\mathbf{A} \tag{48a}$$

$$\mathbf{T} \longleftrightarrow \sigma_{data}\mathbf{b} \tag{48b}$$

$$\mathbf{S^T} \cdot \mathbf{S} = \mathbf{SS} \longleftrightarrow \sigma_{data}^2 \mathbf{A^T} \cdot \mathbf{A} = \sigma_{data}^2[\alpha] = \sigma_{data}^2 \alpha \tag{48c}$$

$$\mathbf{SS}^{-1} = \mathbf{SSI} \longleftrightarrow \frac{1}{\sigma_{data}^2}\alpha^{-\mathbf{1}} = \frac{1}{\sigma_{data}^2}[C] = \frac{1}{\sigma_{data}^2}\mathbf{C} \tag{48d}$$

10.3. Chauvenet

An ancient (1863!—and in this case, age is beauty!) treatise by William Chauvenet, the Appendix of *A Manual of Spherical and Practical Astronomy*, treats complicated cases such as least-square fitting in the presence of known constraints (e.g. if three angles of a triangle are measured, their sum is constrained to be 180°). But it doesn't use matrices.

11. ADDENDUM: FITTING A LINE WHEN BOTH X AND Y HAVE UNCERTAINTIES

We've mentioned that one of the essential assumptions of least squares is that the independent variables are known with high precision and the errors occur only in the measured data. Suppose you're fitting two variables, t and θ, as in equation 1. This essential assumption means that t is known with high precision and all the uncertainty is in θ, and you are minimizing the squares of the residuals in the θ-direction only.

If *both* variables have uncertainties, then you have to be careful because the essential assumption is violated. Before doing anything, read (lightly) Isobe, Feigelson, Akritas, and Babu (1990, ApJ 364, 104). Then adopt one of the two following procedures:

(1) You may be able to reformulate the problem by defining two new variables, one of which has much less fractional uncertainty than the other. The most straightforward reformulation is to define $x = (\theta + t)$ and $y = (\theta - t)$. The sum, x, has smaller fractional uncertainty than y so its dispersion *might* not matter as much. In such a redefinition, you should include a scaling factor on one of the original variables to make the two original dispersions about equal.

Before proceeding with this option, re-read Isobe et al!

(2) More generally, use the prescription in Isobe et al. Lucky for me, I haven't had to delve into this method—but their formulation looks good!