LEAST- AND CHI-SQUARES FOR THE BUDDING AFICIONADO: ART AND PRACTICE

©Carl Heiles June 12, 2002

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 heuristic derivations, discussions, examples, and the prescription for doing least-squares the easy way using matrix techniques generally, and specifically in IDL. This prescription is given as an example in §5, and the *power-user* can skip the details and go directly there.

This document is an update, correction, clarification, and elaboration of a previous one made exclusively for the undergraduate lab class. Here we extend the discussion considerably to cover most of what anyone will need in future professional life. This makes the document longer, but the first parts (§1 to 7) are still accessible at the introductory level because they haven't changed much. We occasionally refer to the books Bevington and Robinson (1992); BR), Cowan (1998), Press et al (2001; Numerical Recipes, NM) and Taylor (1997; T97), and we update the notation to partially conform with NM. We owe sections 11 to 13 to the fantastically excellent website of Stetson, *http://nedwww.ipac.caltech.edu/level5/Stetson/Stetson4.html.*

We begin with least squares in the classic sense, meaning we minimize the sum of squares instead of minimizing χ^2. In astronomy, more often than not you don't have an independent assessment of the intrinsic uncertainty in the data, which means you cannot evaluate χ^2, and the least squares approach is the only option. However, often in astronomy you do want to weight observations differently, e.g. because of integration time, and this requires an approach similar to the χ^2 one. In later sections we generalize to the χ^2 and this other weighted-observations case.

Contents

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

1.1. The closed-form expressions for a straight-line fit

First consider the least squares fit to a straight line. Let θ_m be the m^{th} measurement of the observed quantity (in this example, θ_m is zenith distance; t_m be the time of the m^{th} measurement; $M =$ the total number of observations, i.e. m runs from 0 to $M-1$. Remember that in the least-squares technique, quantities such as t_m are regarded to be known with high accuracy while the quantity θ_m has uncertainties in its measurement.

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

$$A + Bt_m = \theta_m \ . \tag{1}$$

Given this, one does the maximum likelihood (ML) estimate assuming Gaussian statistics. When all measurements have the same intrinsic uncertainty, this is the same as looking for the solution that minimizes the sum of the squares of the residuals (which we will define later). This leads to the pair of equations (Taylor 8.8, 8.9), called the *normal equations*

$$AN + B \ \sum t_m = \sum \theta_m \tag{2a}$$

$$A \ \sum t_m + B \ \sum t_m^2 = \sum t_m \theta_m \tag{2b}$$

Two equations and two unknowns—easy to solve! The closed-form equations for (A, B) are Taylor's equations 8.10 to 8.12.

1.2. Better is the following generalized notation.

We want a way to generalize this approach to include any functional dependence on t and even other variables, and to have an arbitrarily large number of unknown coefficients instead of just the two (A, B). This is very easy using matrix math. We will ease into this matrix technique gently, by first carrying through an intermediate stage of notation.

First generalize the straight-line fit slightly by having two functional dependences instead of one. We have something other than the time t_m; call it s_m. For example, we could have $s_m = \cos(t_m)$ or $s_m = t_m^2$; or we could have $s_m = x_m$, where x_m is the position from which the observation was taken. To correspond to equation 1, $s_m = 1$. Then we rewrite equation 1 to include this extra dependence

$$As_m + Bt_m = \theta_m \ . \tag{3}$$

There are still only two unknown parameters, so this is an almost trivial generalization; later we'll generalize to more parameters.

We have M equations like equation 3, one for each measurement. 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 M 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 3 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. We could carry through the same ML derivation to derive equations equivalent to equation 2; the result is

$$A \sum s_m^2 + B \sum s_m t_m = \sum s_m \theta_m \tag{4a}$$

$$A \sum s_m t_m + B \sum t_m^2 = \sum t_m \theta_m \ . \tag{4b}$$

We can rewrite these equations using the notation $[st] = \sum s_m t_m$, etc.:

$$A[s^2] + B[st] = [s\theta] \tag{5a}$$

$$A[st] + B[t^2] = [t\theta] \ . \tag{5b}$$

This is, of course, precisely analogous to equation 2. And now it's clear how to generalize to more parameters!

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

With this notation it's easy to generalize to more (N) unknowns: the method is obvious because in each equation of condition (like equation 3) we simply add equivalent additional terms such as Cu_m, Dv_m, etc; and in the normal equations (equation 5) we have more products and also more normal equations.

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

$$As_m + Bt_m + Cu_m + Dv_m = \theta_m \ , \tag{6}$$

with $m = 0 \rightarrow (M - 1)$. Again, the least squares fitting process assumes that the s_m, t_m, u_m, v_m are known with zero uncertainty; all of the uncertainties are in the measurements of θ_m. We then form the four normal equations; the generalization of equation 5 written in matrix format is:

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

The $N \times N$ matrix on the left is symmetric. With N equations and N unknowns, 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 $N = 4$ example, and we assume that there are 5 independent measurements ($M = 5$). We first define the matrices

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

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

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

so, in matrix form, the equations of condition (equation 6) reduce to the single matrix equation

$$\mathbf{X}_\sigma \cdot \mathbf{a} = \mathbf{Y}_\sigma \ . \tag{9}$$

The notation for these equations corresponds to NM's. We write them with subscripts σ to emphasize that they are calculated without dividing by σ_{meas}, i.e. that we are doing least squares instead of chi square fitting. For chi square fitting, see §9 and 10.

Our matrix $\mathbf{X}_\sigma$ corresponds to NM's "design matrix" **A** of Figure 15.4.1, except that our elements are not divided by $\sigma_{meas,m}$, and the matrix equation of condition (equation 9) is identical to the expression inside the square brackets of NM's equation 15.4.6. The differences arise because here we are discussing least squares fitting instead of chi square fitting, i.e. we have omitted the factors involving $\sigma_{meas,m}$, the intrinsic measurement uncertainties (§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 7) reduce to the single equation

$$[\alpha_\sigma] \cdot \mathbf{a} = [\beta_\sigma] \ , \tag{10}$$

(NM equation 15.4.10), where

$$[\alpha_\sigma] = \mathbf{X}_\sigma^{\mathbf{T}} \cdot \mathbf{X}_\sigma \tag{11a}$$

$$[\beta_\sigma] = \mathbf{X}_\sigma^{\mathbf{T}} \cdot \mathbf{Y}_\sigma \tag{11b}$$

The matrix $[\alpha_\sigma]$ is known as the *curvature matrix* because each element is twice the curvature of σ^2 (or χ^2) plotted against the corresponding product of variables.

The number of equations is equal to the number of unknowns, so the solution of the matrix equation is easy—just rewrite it by multiplying both sides by the inverse of $[\alpha_\sigma]$ (that is, by $[\alpha_\sigma]^{-1}$), which gives

$$\mathbf{a} = [\alpha_\sigma]^{-1} \cdot [\beta_\sigma] \ . \tag{12}$$

All of these matrix operations are trivially easy in IDL (§5).

4. UNCERTAINTIES IN THE DERIVED COEFFICIENTS.

How about the uncertainties in the derived quantities contained in the matrix **a**?

The first thing to do is derive the *sample* variance s^2 (the square of standard deviation, or mean error, or dispersion, etc) of the individual data points using the generalization of the usual definition for a straight average of x, $s^2 = [\sum_0^{M-1} (x_m - \overline{x_m})^2/(M-1)]$. The generalization is, simply, to replace the $M-1$ in the denominator by $\nu = M - N$. In the straight-average case, $N = 1$ so this fits. Here ν is known as the number of *degrees of freedom* and N, the number of unknown coefficients, is known as the number of *constraints*. So we have

$$s^2 = \frac{1}{M-N} \sum_{m=0}^{M-1} (\theta_m - \overline{\theta_m})^2 \ , \tag{13}$$

where $\overline{\theta_m}$ are the values for θ_m *predicted by the derived quantities* **a**. Note the difference: θ_m are the *observed* values, while $\overline{\theta_m}$ 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 M quantities

$$\delta\theta_m = \theta_m - \overline{\theta_m} \tag{14}$$

are called the *residuals* or *deviations* from the fit.

It's worth reiterating some essentials about s^2, and in particular the denominator $(M - N)$. First consider the case of a single-parameter fit, e.g. $N = 1$. Then we cannot possibly derive a sample variance from only one measurement $M = 1$; but we can from two $M = 2$. So the denominator makes sense from that standpoint. The same goes for $N > 1$.

Next consider the effect of using $(M - N)$ in the denominator: it increases s^2 by the ratio $\frac{M}{M-N}$ over what you'd get if you just took a straight average and used M. This compensates for the fact that you are subtracting $\overline{\theta_m}$, which is derived from the data, instead of the *truly* correct value θ^*. (In formal statistical language, θ^* is the mean of the parent population from which your set of measurements is drawn). If you used the truly correct value θ^*, then the sum would be larger than when using $\overline{\theta_m}$. The use of $M - N$ in the denominator compensates for this larger value in exactly the right way: the expectation value $E_{(s^2)}$ for a large number of experiments is precisely equal to the normal variance σ^2, which you'd get by using [θ^* and M] instead of [$\overline{\theta_m}$ and $(M - N)$] in equation 14; see Cowan equations 5.9 and 5.10. So s^2 is, in fact, exactly the number we want: an unbiased estimate of the true variance of our sample. Why not use [θ^* and M] in equation 14? The reason is obvious: we don't know θ^*! (If we did, we wouldn't be doing this analysis!)

It's easy to calculate the $\overline{\theta_m}$ with matrices. First define the matrix $\overline{\mathbf{Y}_\sigma}$ that contains these values:

$$\overline{\mathbf{Y}_\sigma} = \begin{bmatrix} \overline{\theta_0} \\ \overline{\theta_1} \\ \overline{\theta_2} \\ \overline{\theta_3} \\ \overline{\theta_4} \end{bmatrix} \tag{15}$$

Calculating $\overline{\mathbf{Y}_\sigma}$ is simple:

$$\overline{\mathbf{Y}_\sigma} = \mathbf{X}_\sigma \cdot \mathbf{a} \ . \tag{16}$$

Note that $\mathbf{X}_\sigma$ is already defined (equation 8) and $\mathbf{a}$ was solved for in equation 12. It's convenient to define the residual matrix

$$\delta\mathbf{Y}_\sigma = \mathbf{Y}_\sigma - \overline{\mathbf{Y}_\sigma} \tag{17}$$

so we can write

$$s^2 = \frac{1}{M-N}\delta\mathbf{Y}_\sigma^{\mathbf{T}} \cdot \delta\mathbf{Y}_\sigma \ . \tag{18}$$

This is the sample variance of the datapoints, not the variances in the derived coefficients. We can obtain these as before, by generalizing the results from the two-parameter case like the straight-line fit discussed in §1. We won't go through the derivation here; you can find it in Taylor §8.4 and equation 8.16, 8.17. The result is

$$\mathbf{s_a}^2 = s^2 diag\{[\alpha_\sigma]^{-1}\} \tag{19}$$

Or, to put it simply in words: to get the variance of coefficient n in the matrix $\mathbf{a}$, multiply s^2 by the n^{th} diagonal element of $[\alpha_\sigma]^{-1}$.

Although the above equation for $\mathbf{s_a}^2$ is correct, there is more to the story because of covariances, which are the off-diagonal elements. We return to this topic in §9.

5. A NUMERICAL EXAMPLE AND ITS SOLUTION IN IDL.

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

5.1. Generation of the numerical example

Suppose that we make four measurements of the angle θ and we want to fit to a parabolic function in time t. 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 ($N = 3$). Because there are four independent measurements ($M = 4$) the subscripts run from $m = 0 \rightarrow 3$. Suppose that the four values of time are 5, 7, 9, 11.

First we create the matrix $\mathbf{X}$ in IDL

$$\mathbf{X = fltarr(N, M) = fltarr(3, 4)} \tag{20}$$

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 example, you might manually type them in. After populating the matrix, in direct correspondence with equation 8a we have $s_m = 1$, $t_m = time_m$, $u_m = time_m^2$:

$$\mathbf{X} = \begin{bmatrix} 1 & 5 & 25 \\ 1 & 7 & 49 \\ 1 & 9 & 81 \\ 1 & 11 & 121 \end{bmatrix} \tag{21a}$$

Suppose that the four measured values of θ are (equation 8c)

$$\mathbf{Y} = \begin{bmatrix} 142 \\ 168 \\ 211 \\ 251 \end{bmatrix} . \tag{22a}$$

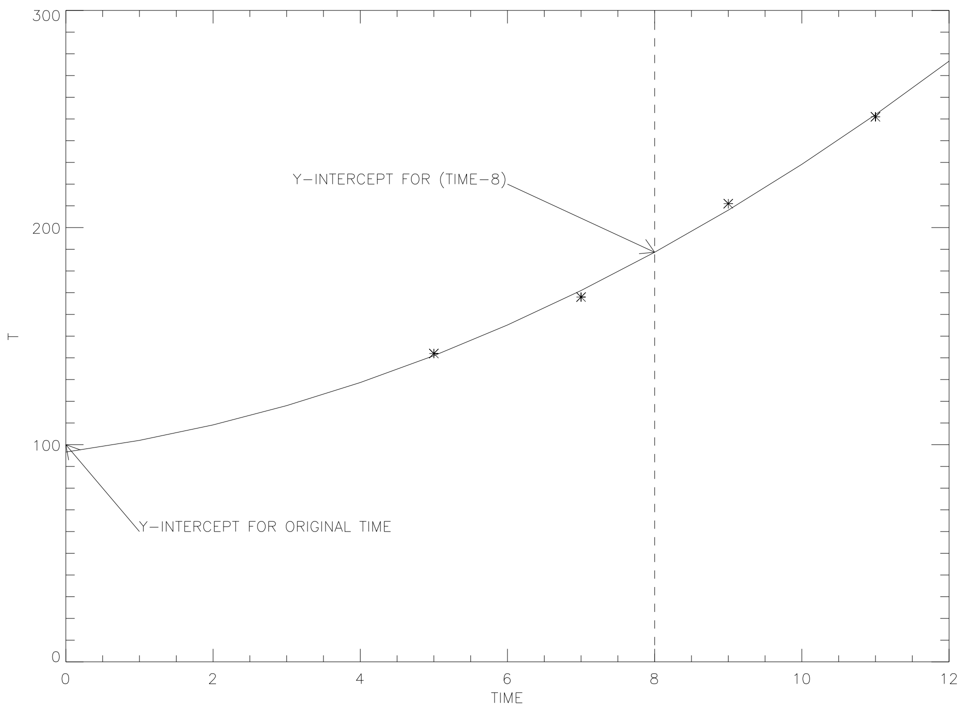

Fig. 1.— Our numerical example. Stars are the four datapoints; the solid line is the fit. We perform two fits: one uses the original definition of time; the other uses ($time - 8$), in effect moving the y-axis to the dashed line. The two fits give the same line but the coefficients and their errors differ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

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

One word of caution here: in IDL, to get these into a column matrix, which is how we've treated $\mathbf{Y}$ above, you have to define $\mathbf{Y}$ as a two-dimensional array because the second dimension represents the column. When working in IDL it's more convenient to define a row vector, which has only one dimension; in IDL you do this by defining $\mathbf{Y} = [142, 168, 211, 251]$; you can make it into the necessary column vector by taking its transpose, i.e. $\mathbf{Y} = \mathbf{transpose(Y)}$.

5.2. Solution of the Numerical Example in IDL

In IDL we calculate the normal equation matrices and denote the $[\alpha]$ in equation 11a by $\mathbf{XX}$:

$$\mathbf{XX} = \mathbf{transpose(X)\#\#X} \;, \tag{23a}$$

and we denote the $[\beta]$ in equation 11b by $\mathbf{XY}$:

$$\mathbf{XY} = \mathbf{transpose}(\mathbf{X})\#\#\mathbf{Y}\ . \tag{23b}$$

In IDL we take the inverse of $[\alpha]$ (same as $\mathbf{XX}$) by

$$\mathbf{XXI} = \mathbf{invert}(\mathbf{XX})\ . \tag{24}$$

The least-square fitted quantities are in the matrix $\mathbf{a}$ (equation 12), which we obtain in IDL with

$$\mathbf{a} = \mathbf{XXI}\ \#\#\ \mathbf{XY}\ . \tag{25}$$

In IDL we denote the matrix of predicted values $\overline{\theta_m}$ by $\mathbf{YBAR}$, which is

$$\mathbf{YBAR} = \mathbf{X}\ \#\#\ \mathbf{a}\ . \tag{26}$$

and we can also define the residuals in $\mathbf{Y}$ as

$$\mathbf{DELY} = \mathbf{Y} - \mathbf{YBAR} \tag{27}$$

In IDL we denote s^2 in equations 13 and 18 by s_sq and write

$$s\_sq = \mathbf{transpose}(\mathbf{DELY})\#\#\mathbf{DELY}/(M-N)\ . \tag{28a}$$

or

$$s\_sq = total(\mathbf{DELY} \wedge 2)/(M-N)\ . \tag{28b}$$

It is *always* a good idea to plot all three quantities (the measured values $\mathbf{Y}$, the fitted values $\mathbf{YBAR}$, and the residuals $\mathbf{DELY}$) to make sure your fit looks reasonable and to check for bad datapoints.

To get the error in the derived coefficients we need a way to select the diagonal elements of a matrix. Obviously, any $N \times N$ matrix has N diagonal elements; a convenient way to get them is

$$diag\ elements\ of\ \mathbf{XXI} = \mathbf{XXI}[(\mathbf{N}+\mathbf{1}) * \mathbf{indgen}(\mathbf{N})] \tag{29}$$

In IDL, we define the variances of the N derived coefficients by **vardc** (think of "**var**iances of **d**erived **c**oefficients"). You can get this as in equation 19 from[1]

$$\mathbf{vardc} = s\_sq * \mathbf{XXI}[(\mathbf{N+1}) * \mathbf{indgen(N)}] \ . \tag{30}$$

5.3. Discussion of the numerical example

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

$$\mathbf{a} = \begin{bmatrix} 96.6250 \\ 4.5000 \\ 0.8750 \end{bmatrix} \tag{31a}$$

and the errors in the derived coefficients [the square root of the σ^2's of the derived coefficients, i.e. $[\sigma_n^2]^{1/2}$ or, in IDL, $sqrt(\mathbf{vardc})$ in equations 30] are:

$$\sigma_{\mathbf{A}} = \begin{bmatrix} 34.012 \\ 9.000 \\ 0.5590 \end{bmatrix} . \tag{31b}$$

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—certainly not always), without taking more data!

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. Definition of the normalized covariance matrix

The variances in the derived coefficients are obtained from the diagonal elements of **XXI**. The off-diagonal elements represent the *covariances* between the derived coefficients. Covariance means,

[1]If you used equation 28a instead of 28b, then IDL considers s_sq an array and you need to use a # instead of a $*$ in this equation.

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

$$ncov_{ik} = \frac{XXI_{ik}}{\sqrt{XXI_{ii}\ XXI_{kk}}} \tag{32}$$

This is the same normalization that one does with the Pearson linear correlation coefficient. In fact, the elements of the normalized covariance matrix *are* the correlation coefficients. In IDL, you do the following:

$$\mathbf{dc} = \mathbf{XXI}[(\mathbf{N}+\mathbf{1}) * \mathbf{indgen}(\mathbf{N})] \tag{33a}$$

$$\mathbf{ncov} = \mathbf{XXI}/sqrt(\mathbf{dc\#dc}) \tag{33b}$$

In the above, $\mathbf{dc\#dc}$ is an $N \times N$ matrix consisting of products of the diagonals of **XXI**, so dividing **XXI** by $sqrt(\mathbf{dc\#dc})$ generates the normalized version.

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

$$\mathbf{C} = \sigma_{meas}^2 \mathbf{XXI} \tag{34}$$

because **C** is defined for χ^2 instead of σ^2 (see §9).

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 *uncertainties* 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.

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

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 $\overline{A_0}$ and $\overline{A_1}$, which differ from the *true* values (A_0^*, A_1^*) 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. For one, this means that with noisy data power can be shared or passed from one parameter to/from its covariant counterpart(s). As we shall see §10, it also significantly influences the uncertainties 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 sometimes 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. However, sometimes your interest is in a specific set of functions that are *not* orthogonal, and in such cases it makes no sense to convert to orthogonal functions—because you just have to convert back again and do the error propagation after-the-fact instead of letting the least squares process do it for you.

6.2. The covariance in our numerical example

Apply equation 33 to obtain the covariance matrix for our numerical example:

$$\mathbf{ncov} = \begin{bmatrix} 1 & -.989848 & .969717 \\ -.989848 & 1 & -.993808 \\ .969717 & -.993808 & 1 \end{bmatrix} . \tag{35}$$

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 $(time_m = time_m - 8)$. We get the same fitted line, but the derived coefficients are completely different—and amazingly better! We get

$$\mathbf{A} = \begin{bmatrix} 188.625 \\ 18.500 \\ 0.87500 \end{bmatrix} \tag{36a}$$

$$\sigma_{\mathbf{A}} = \begin{bmatrix} 3.58 \\ 1.00 \\ 0.559 \end{bmatrix} . \tag{36b}$$

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. You can see this from the normalized covariance matrix, which has become

$$\mathbf{ncov} = \begin{bmatrix} 1 & 0 & -0.78086881 \\ 0 & 1 & 0 \\ -0.78086881 & 0 & 1 \end{bmatrix} . \tag{37}$$

which is nice, but not perfect: Our step is *partial* because the second-order coefficient A_2 affects A_0 because, over the range of $[(time - 8) = -3 \rightarrow +3]$, the quantity $[A_2 \, \Sigma(time_m - 8)^2]$ is always positive and is thereby correlated with A_0.

We could complete the process of orthogonalization by following the prescription in BR chapter 7.3, which discusses the general technique of orthogonalizing the functions in least square fitting. The general case is a royal pain, so much so that we won't even carry it through for our example.

For some particular cases, standard pre-defined functions are orthogonal. For example, if t_m is a set of uniformly spaced points between $(-1 \rightarrow 1)$ and you are fitting a polynomial, then the appropriate orthogonal set is Legendre polynomials. This is good if your only goal is to represent a bunch of points by a polynomial function, because the coefficients of low-order polynomials are independent of the higher ones. However, it's more work and, moreover, often you are interested in the coefficients for specific functions that don't happen to be orthogonal; in such cases, you should just forge ahead.

But *always* look at the normalized covariance matrix. Suppose one pair of off-diagonal elements departs significantly from zero. Then their corresponding functions are far from being orthogonal and the variances of the derived coefficients will suffer as a result. You might be able to eliminate one of the parameters to make the fit more robust. For example, suppose one function is $t\cos(t)$ and the other is $\sin(t)\cos(t)$. If the range of t is small, these two functions are indistinguishable and have a large covariance; you should eliminate one from the fit. If the range of t is large, there is no problem.

For further discussion of covariance, see §10. Also, you might also want to try out another example in Taylor's §8.5.

7. REJECTING BAD DATAPOINTS I.: CHAUVENET'S CRITERION

Least squares fitting is derived from the maximum likelihood argument assuming the datapoint residuals δy_m have a Gaussian pdf. This means that the errors are distributed as

$$p_{(\delta y;\sigma)} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{\delta y^2}{2\sigma^2}\right)} \tag{38}$$

where σ^2 is the true variance of the datapoints, i.e. s^2 in equation 13 (to be precise, s^2 needs to be averaged over many experiments).

More importantly, the probability of finding datapoints inside the limits $\pm\Delta y$ is

$$P_{(|\delta y|<\Delta y)} = \int_{-\Delta y}^{+\Delta y} p_{(\delta y;\sigma)} d(\delta y) = \mathrm{erf}\left(\frac{\Delta y}{\sqrt{2}\sigma}\right) \tag{39}$$

where we use the commonly-defined error function $\mathrm{erf}_{(X)} = \frac{1}{\sqrt{\pi}} \int_{-X}^{+X} e^{-x^2} dx$. A particularly important value is for $\Delta y = \sigma$, for which

$$P_{(|\delta y|<\sigma)} = 0.683 \tag{40}$$

If we have an experiment with M datapoints, then the number of datapoints we expect to lie outside the interval $\pm\Delta y$ is

$$M_{(outside\ \Delta y)} = M\left[1 - \mathrm{erf}\left(\frac{\Delta y}{\sqrt{2}\sigma}\right)\right] \tag{41}$$

Chauvenet's criterion simply says:

1. Find Δy such that $M_{(outside\ \Delta y)} = 0.5$. This is given by

$$\frac{\Delta y}{\sigma} = \mathrm{erf}^{-1}\left(1 - \frac{1}{2M}\right) \tag{42}$$

2. Discard all datapoints outside this range.

This criterion makes very good sense. It leads to the following rejection criteria

Chauvenet's criterion versus M	
M	$\frac{\Delta y}{\sigma}$
100	2.81
1000	3.48
10^4	4.06
10^5	4.56

which is a moderately interesting set of numbers.

We offer the following important *Comments*:

- This assumes data are Gaussian distributed. In real life this doesn't often happen because of "glitches". Examples of glitches can be interference in radio astronomy, meteors in optical astronomy, and cosmic rays on CCD chips. These glitches produce bad points that depart from Gaussian statistics. They are often called *outliers*.

  It is very important to get rid of the outliers because the least squares process minimizes the *squares* of the residuals. Outliers, being the points with the largest residuals, have a disproportionately evil effect on the result.

  On the other hand, if your data don't follow Gaussian statistics as their *intrinsic* pdf, then you should think twice before using least squares!

- You may wish to relax Chauvenet's criterion by *increasing* the Δx beyond which you discard points. This is being conservative and, in the presence of some nonGaussian statistics, not a bad idea. But think about why you are doing this before you do it. Maybe the intrinsic statistics aren't Gaussian?

  You should *never* make Chauvenet's criterion more stringent by *decreasing* the Δx beyond which you discard points. This rule hardly needs elaboration: it means you are discarding datapoints that follow the assumed pdf!

- Most statistics books (e.g. Taylor, BR) harp on the purity aspect. One extreme: don't throw out any datum without examining it from all aspects to see if discarding it is justified. The other extreme: apply Chauvenet's criterion, but do it *only once* and certainly not repeatedly.

  Being a real-life astronomer, our approach is different. There *do* exist outliers. They increase the calculated value of σ. When you discard them, you are left with a more nearly perfect approximation to Gaussian statistics and the new σ calculated therefrom will be smaller than when including the outliers. Because the original σ was too large, there may be points that should have been discarded that weren't. So our approach is: repeatedly apply Chauvenet's criterion until it converges.

If it doesn't converge, or if it discards an inordinately large number of datapoints, you've got real problems and need to look at the situation from a global perspective.

- Many observers use the 3σ criterion: discard any points with residuals exceeding 3σ. This is definitely *not* a good idea: the limit 3σ is Chauvenet's criterion for $M = 185$ datapoints. Very often M exceeds this, often by a lot.

- To apply Chauvenet's criterion it's most convenient to calculate the inverse error function. For this, you have two choices. One (for sissies like myself), you can use **inverf.pro** from my area ˜ heiles/idl/gen . But the real he-man will want to learn about using a root-finding algorithm such Newton's method (NM §9.4 and 9.6) together with the error function; both procedures exist in IDL as **newton** and **errorf**. You at least ought to skim lightly some of NM's chapter 9 about root finding, because some day you'll need it.

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 you have a nonlinear dependence, such as for example wanting to solve for A and B with equations of condition that look like

$$\sin(At_m) + Bt_m = \theta_m \; . \tag{43}$$

What do you do here? You linearize the process, 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 $\theta_{0,m}$:

$$\sin(A_0 t_m) + B_0 t_m = \theta_{0,m} \; . \tag{44}$$

Subtract equation 44 from 43, and express the differences as derivatives. Letting $\delta A = A - A_0$ and $\delta B = B - B_0$, this gives

$$\delta A[t_m \cos(A_0 t_m)] + \delta B t_m = \theta_m - \theta_{0,m} \; . \tag{45}$$

This is linear in $(\delta A, \delta B)$ so you can solve for them using standard least squares. Increment the original guessed values to calculate $A_{0,new} = A_0 + \delta A$ and $B_{0,new} = B_0 + \delta B$, These won't be exact because higher derivatives (including cross derivatives) come into play, so you need to use these new values 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.

(0) In linear least squares, the curvature and covariance matrices are set by the values of the independent variable, which here is denoted by t, and are independent of the datapoint values. Here, the matrix elements change from one iteration to the next because they depend on the guessed parameters, and sometimes they even depend on the datapoint values.

(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 multiply 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 45, 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; take a look at NM's section 5.7 on evaluating numerical derivatives.

(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.

When convergence is slow or doesn't occur because your functions are complicated, you might wish to try the Levenberg-Marquardt method (NM §15.5); IDL function **LMFIT**. This involves increasing the diagonal elements of the curvature matrix by a set of suitably chosen factors; when you get close to the minimum, you reset these factors to unity. I have never had to resort to such tactics. In my opinion, if you are in such difficulty that you need such a technique you should look at the problem with an eye to gaining an appreciation of the approximate locations of the true minima and beginning with your initial guess reasonably close to it. Of course, this might be difficult with a large number of parameters.

(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!

(8): Reformulate! Sometimes you can avoid all this by reformulating the problem. There are two cases: the harmless case and the not-so-harmless case.

An example of the harmless case is fitting for the phase ϕ in the function $y = \cos(\theta + \phi)$. This is definitely a nonlinear fit! But its easy to reformulate it in a linear fit using the usual trig identities to write $y = A\cos\theta - B\sin\theta$, where $\frac{B}{A} = \tan\phi$. Solve for (A, B) using linear least squares, calculate ϕ, and propagate the uncertainties.

An example of the not-so-harmless case is in NM's §15.4 example: fit for (A, B) with equations of condition $y_m = Ae^{-Bx_m}$. They suggest linearizing by rewriting as $\log(y_m) = C - Bx_m$, solving for (B, C), and deriving A after-the-fact. This is not-so-harmless because you are applying a nonlinear function to the *observed* values y_m; thus the associated errors $\sigma_{meas,m}$ are *also* affected. This means you have to do weighted fitting, which is discussed in §9 below. Suppose that $A = 1$, your datapoints all have $\sigma_{meas,m} = 0.05$, and the observed y_m ranges from 0.05 to 1. The datapoint with $y_m = 0.05$ has a manageable σ_{meas_m}, but what is the corresponding value of $\sigma_{meas,m}$ for $\log y_m = \log 0.05$? It's ill-defined and asymmetric about the central value. Or even, God forbid, you have an observed y_m that's *negative*??? Even for y_m not near zero, you need to calculate new $\sigma_{meas,m}$ by error propagation; in this case, you need to reassign $\sigma(\log y) = \frac{d\log y}{dy}\sigma(y) = \frac{\sigma(y)}{y}$. This is OK when y_m is large enough so that the linear approximation is accurate, but if not the converted noise becomes nonGaussian.

You should regard your datapoints as sacrosanct and never apply any nonlinear function to them.

9. CHI-SQUARE FITTING, AND WEIGHTED FITTING: DISCUSSION IGNORING COVARIANCE

In least squares fitting the derived parameters minimize the sum of squares of residuals as in equation 13, which we repeat here

$$s^2 = \frac{1}{M-N} \sum_{m=0}^{M-1} (\theta_m - \overline{\theta_m})^2 \ .$$

Chi square fitting is similar except for two differences. One, we divide each residual by its intrinsic measurement error $\sigma_{meas,m}$; and two, we define χ^2 as the sum

$$\chi^2 = \sum_{m=0}^{M-1} \frac{(\theta_m - \overline{\theta_m})^2}{\sigma^2_{meas,m}} \ . \tag{46a}$$

Along with χ^2 goes the *reduced* chi squared $\overline{\chi}^2 = \frac{\chi^2}{M-N}$

$$\overline{\chi}^2 = \frac{1}{M-N} \sum_{m=0}^{M-1} \frac{(\theta_m - \overline{\theta_m})^2}{\sigma^2_{meas,m}} \ . \tag{46b}$$

which is more directly analogous to the definition of s^2.

Chi-square fitting is very much like our least-squares fitting except that we divide each datapoint by its intrinsic measurement uncertainty $\sigma_{meas,m}$. Thus, the reduced Chi-square ($\widehat{\chi}^2$) is equal to the ratio of the *variance of the datapoint residuals* (σ^2) to the *adopted intrinsic measurement variances* (σ^2_{meas}). So it should be obvious that in Chi-square fitting, you must know the measurement uncertainties σ_{meas} of the individual data points beforehand. If you want to give the various datapoints weights based on something other than σ_{meas}, then that is just like Chi-square fitting except that you can adopt an arbitrary scale factor for the uncertainties (section 9.3).

Chi-squared fitting treats uncertainties of the derived parameters in a surprising way. Getting the coefficient uncertainties with chi-squared fitting is a tricky business because

1. With the standard treatments, the errors in the derived parameters don't depend on the residuals of the datapoints from the fit (!).

2. The errors in the derived parameters can depend on their mutual covariances. This discussion requires a separate section, which we provide below in §10.

In this section we treat chi square fitting ignoring covariance. We begin by illustrating the difference between least squares and chi square fitting by discussing the simplest chi-square fitting case of a weighted mean; then we generalize to the multivariate chi square fitting case.

9.1. The weighted mean: the simplest chi-squared fit

First, recall the formulas for an ordinary *unweighted* average in which the value of each point is y_m and the residual of each point from the weighted mean is δy_m:

$$mean = \frac{\sum y_m}{M} \tag{47a}$$

$$s^2 = \frac{\sum \delta y_m^2}{M-1} \tag{47b}$$

$$s_{mean}^2 = \frac{s^2}{M} = \frac{\sum \delta y_m^2}{M(M-1)} \ , \tag{47c}$$

s_{mean}^2 is the variance of the mean and s^2 is the variance of the datapoints around 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 s_{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 $w_{meas,m} = \frac{1}{\sigma_{meas,m}^2} = \frac{1}{\sigma_{meas}^2}$. Applying maximum likelihood, in a nonweighted average the quantity that is minimized is $\sum \delta y_m^2$; in a weighted average the quantity minimized is $\chi^2 = \sum \frac{\delta y_m^2}{\sigma_{meas,m}^2} = \sum w_{meas,m} \delta y_m^2$. So you'd think that the three equations corresponding to the above would become

$$mean_w = \frac{\sum w_{meas,m} y_m}{\sum w_{meas,m}} \rightarrow \frac{\sum y_m}{M} \tag{48a}$$

$$s_w^2 = \frac{M}{M-1} \frac{\sum w_{meas,m} \delta y_m^2}{\sum w_{meas,m}} \rightarrow \frac{\sum \delta y_m^2}{M-1} \tag{48b}$$

$$s_{mean}^2 = \frac{s_w^2}{M} = \frac{\sum w_{meas,m} \delta y_m^2}{(M-1) \sum w_{meas,m}} \rightarrow \frac{\sum \delta y_m^2}{M(M-1)} \ , \tag{48c}$$

where the arrow to the rightmost expressions use our assumption that $\sigma_{meas,m}$ are all identical. In fact, however, the last of these equations is always written (e.g. BR equation 4.19; Taylor equation 7.12)

$$s_{mean}^2 = \frac{1}{\sum w_{meas,m}} \rightarrow \frac{\sigma_{meas}^2}{M} \ , \tag{49}$$

Note the excruciatingly painful difference between equation 48c and equation 49: the former depends on the *variance of the datapoint residuals* s_w^2, as you'd think it should, while the latter depends on only the *adopted intrinsic measurement variance of the data* σ_{meas}^2, which is chosen by the guy doing the fit. If you do an unweighted average, and derive a certain variance, and next do a weighted average in which you choose some value for σ_{meas} that happens to be wrong, the two fits give different results for s_{mean}^2. This is crazy.

To get around this difficulty, we follow the procedure in BR equations 4.20 to 4.26. This introduces an arbitrary multiplicative factor for the weights and goes through the ML calculation to derive, instead of equation 49, the far superior

$$s_{mean}^2 = \frac{\widehat{\chi}^2}{\sum w_{meas,m}} = \frac{\chi^2}{(M-1)\sum w_{meas,m}} \rightarrow \frac{s_w^2}{M} \ , \tag{50}$$

which is precisely the same as our intuitive guess, equation 48c. The difference between equations 50 and 49 is the numerator, which contains the reduced chi-squared $\widehat{\chi}^2$; for this case where all $\sigma_{meas,m}$ are identical, $\widehat{\chi}^2 = \frac{s_w^2}{\sigma_{meas}^2}$. Here

$$\chi^2 = \sum w_{meas,m} \delta y_m^2 = \sum \frac{\delta y_m^2}{\sigma_{meas,m}^2} \tag{51}$$

9.2. The multivariate Chi-square fit

In this case, Chi-square fitting is just like least-squares fitting except for the following:

1. In the matrix $\mathbf{X}_\sigma$ of equation 8a, each row m is a different measurement with a different intrinsic variance $\sigma_{meas,m}$. You are generating a new matrix $\mathbf{X}$, which is identical to $\mathbf{X}_\sigma$ except that each row m is divided by $\sigma_{meas,m}$. This new matrix is the same as NR's *design matrix* (Figure 15.4.1), which they denote by $\mathbf{A}$.

2. Divide each datapoint θ_m in equation 8b by $\sigma_{meas,m}$. You are generating a new matrix $\mathbf{Y}$, which is identical to $\mathbf{Y}_\sigma$ except that each row is divided by σ_{meas}. This new matrix is the same as NR's vector $\mathbf{b}$.

You've divided each row, i.e. the equation of condition for each row m, by a common factor, so the solution of the equation is unchanged.

Now suppose that all datapoints have the same intrinsic measurement uncertainty σ_{meas}. In the full matrix equation 9, which now reads $\mathbf{X} \cdot \mathbf{a} = \mathbf{Y}$, you've divided both sides by the same quantity, σ_{meas}. Obviously, this doesn't change the derived results for $\mathbf{a}$. (If the $\sigma_{meas,m}$ differ from each other, then $\mathbf{a}$ is affected, of course.)

Let's summarize the relationships between the least-squares (σ-subscripted) and χ^2 matrices for this specific case where all $\sigma_{meas,m}$ are identical:

$$\mathbf{X} = \frac{\mathbf{X}_\sigma}{\sigma_{meas}} \tag{52a}$$

$$\mathbf{Y} = \frac{\mathbf{Y}_\sigma}{\sigma_{meas}} \tag{52b}$$

$$\mathbf{X} \cdot \mathbf{a} = \mathbf{Y} \tag{52c}$$

$$[\alpha] = \mathbf{X^T} \cdot \mathbf{X} = \frac{[\alpha_\sigma]}{\sigma^2_{meas}} \tag{52d}$$

$$[\beta] = \mathbf{X^T} \cdot \mathbf{Y} = \frac{[\beta_\sigma]}{\sigma_{meas}} \tag{52e}$$

$$[\alpha]^{-1} = \sigma^2_{meas}[\alpha_\sigma]^{-1} \tag{52f}$$

$$\mathbf{a} = [\alpha]^{-1} \cdot [\beta] \tag{52g}$$

When you calculate the predicted values $\overline{\mathbf{Y}}$ in equation 16, using $\mathbf{X}$ instead of $\mathbf{X}_\sigma$, the values are divided by σ_{meas}. Because $\mathbf{Y}$ is also divided by σ_{meas}, the residuals $\delta\mathbf{Y}$ are as well. Thus when you calculate the square of the sum of the residuals as in equation 18 you get $\frac{s^2}{\sigma^2_{meas}}$, which is the reduced chi-square, denoted as $\widehat{\chi}^2$:

$$\widehat{\chi}^2 = \frac{\delta\mathbf{Y^T} \cdot \delta\mathbf{Y}}{M - N} = \frac{\delta\mathbf{Y}^\mathbf{T}_\sigma \cdot \delta\mathbf{Y}_\sigma}{\sigma^2_{meas}(M - N)} \tag{53}$$

and, obviously, you could calculate the non-reduced χ^2

$$\chi^2 = \delta\mathbf{Y^T} \cdot \delta\mathbf{Y} = \frac{\delta\mathbf{Y}^\mathbf{T}_\sigma \cdot \delta\mathbf{Y}_\sigma}{\sigma^2_{meas}} \tag{54}$$

This is precisely analogous to equation 51 for the weighted mean example above. The reduced chi-square $\widehat{\chi}^2$ is equal to the ordinary chi-square χ^2 except that it is divided by the number of degrees of freedom, ordinarily denoted by ν, which is equal to $(M - N)$.

Finally, we have the analogy of equation 50 expressed in matrix form as in equation 19:

$$\mathbf{s_a}^2 = \widehat{\chi}^2 diag\{[\alpha]^{-1}\} \tag{55}$$

This is in contrast to the result quoted in textbooks (e.g. NM equation 15.4.15, BR equation 7.25), which omits the $\widehat{\chi}^2$ factor:

$$\mathbf{s_a}^2 = diag\{[\alpha]^{-1}\} \tag{56}$$

and, as in the standard textbook solution for the weighted mean case, provides parameter errors that are independent of the datapoint residuals.

Our result, equation 55, is very reasonable. 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 $s^2 \approx \sigma^2_{meas}$ and $\widehat{\chi}^2 \approx 1$. (We write "≈" instead of "=" because different experiments produce somewhat different values of s^2 because of statistical fluctuations; an average gives $\sigma^2 = \langle s^2 \rangle$)). In this situation, though, equations 55 and 56 are identical.

9.3. Persnickety Diatribe on Choosing σ_{meas}

9.3.1. Choosing and correcting $\sigma_{data,m}$

In the previous section, equation 56 taught us that—formally, at least—the variance in the derived fit parameter (or its uncertainty, which is the square root) depends only on the adopted uncertainties $\sigma_{meas,m}$ and not on the *actual variance* of the *datapoints*.

Are you bothered by the fact that the uncertainty of the mean σ_{x0} is independent of the data values? You should be: it is obvious that the data values affect σ_{mean}.

Formally, σ_{x0} depends only on the *adopted* uncertainties $\sigma_{meas,m}$, which are chosen beforehand by you—you're supposed be such a good experimentalist that you really do know the intrinsic uncertainty in your measured values. Moreover, you are assuming that there are no other sources of uncertainty—such as "cosmic scatter" or an inappropriate model to which you are fitting the data. Suppose your adopted values of $\sigma_{meas,m}$ are off by a common scale factor, i.e. if $\sigma_{meas,adopted} = f\sigma_{meas,true}$. Then $\widehat{\chi}^2 \approx f^{-2}$ instead of $\widehat{\chi}^2 \approx 1$. And to obtain the parameter errors from $\delta\chi^2$, you must find the offset δx such that $\Delta\chi^2 = f^{-2} \approx \widehat{\chi}^2$.

You can correct for this erroneous common factor f by dividing your adopted values of $\sigma_{meas,m}$ by f. Of course, you don't know what this factor f is until you do the chi squared fit. Dividing them by f is equivalent to multiplying them by $\widehat{\chi}$. And, of course, the same as multiplying $\sigma^2_{meas,m}$ by $\widehat{\chi}^2$.

To be kosher, after having run through the problem once with the adopted $\sigma_{meas,m}$, calculate the $\widehat{\chi}^2$; multiply all $\sigma_{meas,m}$ by $\widehat{\chi}$; and redo the problem so that the new $\widehat{\chi}^2 = 1$. Then the derived variance σ^2_{x0} is also correct. You can obtain it either as the corresponding diagonal to the covariance matrix (equations 55 and 56, which are identical in this case) or by finding what departure from x_0 is necessary to make $\Delta\chi^2 = 1$. This redoing the fit may seem like unnecessary work, but when we deal with multiparameter error estimation it's the best way to go to keep yourself from getting confused.[3]

9.3.2. Come back to reality

In the case $\widehat{\chi}^2 \approx 1$ the dispersion of the observed points σ is equal to the intrinsic dispersion of the datapoints σ_{meas} and the mathematical model embodied in the least-squares fit is perfect. That, at least, is the *theoretical* conclusion. In practice, however, your obtaining such a low, good value for $\widehat{\chi}^2$ might mean instead that you are using *too large* a value for σ_{meas}: 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_{meas} and you specify that they have $2\sigma_{meas}$ when computing $(\mathbf{X}, \mathbf{Y})$, then you'll find $\widehat{\chi}^2 = 0.25$.

There are, in fact, two ways you can get artificially small values for $\widehat{\chi}^2$. One is by overestimating your adopted intrinsic values of σ_{meas}. The other is if your *measurements are correlated*. Suppose, for example, that by mistake you include the same measurements several times in your fit. Then your measurements are no longer independent; in other words there is covariance in your measured points. Cowan includes this possibility in his equation 7.4 and also example 7.6.1.

High values of $\widehat{\chi}^2$ 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. And in this case, using equation 56 instead of 55 is disastrous.

Think about it.

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

Here the $\sigma_{meas,m}$ are all different. The m^{th} row of the equation-of-condition matrix $\mathbf{X}_\sigma$ and the m^{th} element of the data vector $\mathbf{Y}_\sigma$ get divided by their corresponding $\sigma_{meas,m}$. The equation embodied in each row of the matrix equation 9 remains unchanged, but the different rows are

[3]The $\Delta\chi^2 = 1$ condition only applies for the single-variable case. For more variables, the variances need to make $\Delta\chi^2 > 1$; for example, if you are fitting for two parameters, then the variances make $\Delta\chi^2 = 2.3$, as we discussed in §10.

weighted differently with respect to each other.

Consider two measurements with intrinsic measurement uncertainties $(\sigma_{meas,1}, \sigma_{meas,2})$; suppose $\sigma_{meas,1} < \sigma_{meas,2}$. After being divided by their respective σ_{meas}'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 $\frac{\sigma_{meas,1}}{\sigma_{meas,2}} = \left(\frac{\tau_1}{\tau_2}\right)^{-1/2}$, so the rows of $\mathbf{X}$ are weighted as $\tau^{1/2}$. This means that during the computation of $[\alpha] = \mathbf{X^T} \cdot \mathbf{X}$, the self-products of row 1 are weighted as τ_1. This means that each datapoint is weighted as τ, 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 $\left(\frac{1}{\sigma_{meas}}\right)^2$. We conclude that weighting scheme of the first two steps in section 9.2 agrees with common sense.

Suppose you don't know the intrinsic measurement dispersion σ_{meas}, 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 $\sigma_{meas} \propto \tau^{-1/2}$. In this case, multiply each row by its weight $w \propto \frac{1}{\sigma_{meas}}$ and proceed as above.

9.5. The explicit IDL for the weighted and chi square cases

Here we give the explicit IDL commands required to do chi squared fitting, as we did for least squares fitting in §5.2. First, define the $\mathbf{X}$ and $\mathbf{Y}$ arrays as before in §5. However, here we need to weight the matrices. Let us denote the weighted versions by $\mathbf{XW}$ and $\mathbf{YW}$. Then each row of $\mathbf{XW}$ and element of $\mathbf{YW}$ are divided by the corresponding σ_{meas}. Let us define $\mathbf{W}$ as the diagonal matrix[4] whose elements are equal to $\frac{1}{\sigma_{meas,m}}$, i.e. $w_m = \frac{1.}{\sigma_{meas,m}}$

$$\mathbf{W} = fltarr(M, M) \tag{57a}$$

$$\mathbf{W[indgen(M) * (M + 1)]} = \left[\frac{1}{\sigma_{meas,m}}\right] \tag{57b}$$

Note that w_m is the *square root* of the weights $w_{meas,m}$ that we discussed in §9.1. Specifically, the current weights w_m are *not* those in the sense of the weighted average of §9.1; we use w_m as a

[4]$\mathbf{W}$ has $M \times M$ elements but only M numbers. If M is large and you are impacting machine memory, then you can save memory by: reformulating equations 58 and 64 to use loops; not keeping $\mathbf{XW}$ and $\mathbf{X}$ as separate matrices; and using IDL's **temporary** function wherever possible. The relevant for loop is **for mr= 0, m do xw[*,mr]=w[mr]*x[*,mr]**, etc.

matter of convenience only. Then

$$\mathbf{XW} = \mathbf{W}\ \#\#\ \mathbf{X} \tag{58a}$$

$$\mathbf{YW} = \mathbf{W}\ \#\#\ \mathbf{Y} \tag{58b}$$

Optional comment: At this point you might wish to revert to Singular Value Decomposition method using IDL's **SVDFIT** function; see §14 for a cookbook. It is rare, but sometimes you will encounter a problem for which the inversion of **XXW** is unstable. In this case, $\mathbf{XXW\#\#XXWI} \neq \mathbf{I}$ (**I** is the unit matrix). You can tell this with the **invert** function by calling it with the **status** keyword; if $\mathbf{status} \neq 0$, you've got problems.

All steps up to obtaining the uncertainties follow exactly the same steps as in §5.2, which we repeat here. In IDL we calculate the normal equation matrices and denote the $[\alpha]$ in equation 11a by **XXW**:

$$\mathbf{XXW} = \mathbf{transpose}(\mathbf{XW})\#\#\mathbf{XW}\ , \tag{59a}$$

and we denote the $[\beta]$ in equation 11b by **XYW**:

$$\mathbf{XYW} = \mathbf{transpose}(\mathbf{XW})\#\#\mathbf{YW}\ . \tag{59b}$$

In IDL we take the inverse of $[\alpha]$ (same as **XXW**) by

$$\mathbf{XXWI} = \mathbf{invert}(\mathbf{XXW})\ . \tag{60}$$

The least-square fitted quantities are in the matrix **a** (equation 12), which we obtain in IDL with

$$\mathbf{a} = \mathbf{XXWI}\ \#\#\ \mathbf{XYW}\ . \tag{61}$$

In IDL we denote the matrix of *weighted* predicted values by **YBARW**, which is

$$\mathbf{YBARW} = \mathbf{XW}\ \#\#\ \mathbf{a}\,. \tag{62}$$

and we can also define the *weighted* residuals as

$$\mathbf{DELYW} = \mathbf{YW} - \mathbf{YBARW} \tag{63}$$

The *true* (unweighted) predicted values and residuals are

$$\mathbf{YBAR} = \mathbf{invert(W)}\ \#\#\mathbf{XW}\ \#\#\ \mathbf{a} = \mathbf{X}\ \#\#\ \mathbf{a}\ . \tag{64a}$$

$$\mathbf{DELY} = \mathbf{invert(W)}\ \#\#\ (\mathbf{YW} - \mathbf{YBARW}) = \mathbf{Y} - \mathbf{YBAR} \tag{64b}$$

To calculate the chi square we want the weighted sum $\sum \delta w_m^2 y_m^2$:

$$chi\_sq = \mathbf{transpose(DELYW)}\ \#\#\ \mathbf{DELYW}\ . \tag{65a}$$

and the reduced chi square

$$red\_chi\_sq = \mathbf{transpose(DELYW)}\ \#\#\ \mathbf{DELYW}/(M - N)\ . \tag{65b}$$

The full covariance matrix is just **XXWI** and, corresponding to equation 30, the variances of the derived coefficients are the diagonal elements—unless the reduced chi squared differs significantly from unity, in which case you should seriously look at your assumed weights derived from $\sigma_{meas,m}$ and/or the quality with which the model fits the data. That is,

$$\mathbf{vardc\_official} = \mathbf{XXWI}[(\mathbf{M+1}) * \mathbf{indgen(M)}] \tag{66a}$$

or, better,

$$\mathbf{vardc\_realworld} = red\_chi\_sq * \mathbf{XXWI}[(\mathbf{M+1}) * \mathbf{indgen(M)}] \tag{66b}$$

Finally, the normalized covariance matrix is of course independent of chi squared and is the same as equation 33

$$\mathbf{ncov} = \mathbf{XXWI}/sqrt(\mathbf{vardc\_official\#vardc\_official}) \tag{67}$$

You should, of course, always look at the residuals from the fit. But here you need to look at the weighted residuals **DELYW**.

10. CHI-SQUARE FITTING, AND WEIGHTED FITTING: DISCUSSION INCLUDING COVARIANCE

10.1. Phenomenological description

Consider the first two coefficients in our above example, which we discussed a bit above in §6.2. In this example, the fit gives $y = A_0 + A_1 t + A_2 t^2$, where the numerical values are given in vector form by equation 31. The coefficient A_0 is the y-intercept and A_1 is the slope. They have derived values $A_0 = 96 \pm 34$ and $A_1 = 4 \pm 9$.

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.

Suppose you are interested in knowing about A_0 *without regard to* A_1. By this we mean that as A_0 is varied from its optimum value of 96, χ^2 increases from its minimum value. As we vary A_0, if we allow A_1 to take on whatever value it needs to for the purpose of minimizing χ^2, then this is what we mean by "knowing about A_0 without regard to A_1". For this case, the uncertainty of A_0 is indeed 34. Ditto for A_1. In other words, equations 19, and 55 apply.

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 $\delta A_0 = +34$ and $\delta A_1 = +9$ are acceptable, you *cannot* conclude that the *pair* of values $(\delta A_0, \delta A_1) = (+34, +9)$ is acceptable, because this pair has both positive. In contrast, what *is* acceptable here would be something like $(\delta A_0, \delta A_1) = (+34, -9)$.

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 is to construct the ellipses that define the loci of constant $\Delta\chi^2$ and present them on a graph with axes $(\delta a_0, \delta a_1)$ as in BR Figure 11.2 or NR Figure 14.5.4. For three variables, these ellipses become ellipsoids; for four, they become four-dimensional volumes, etc.

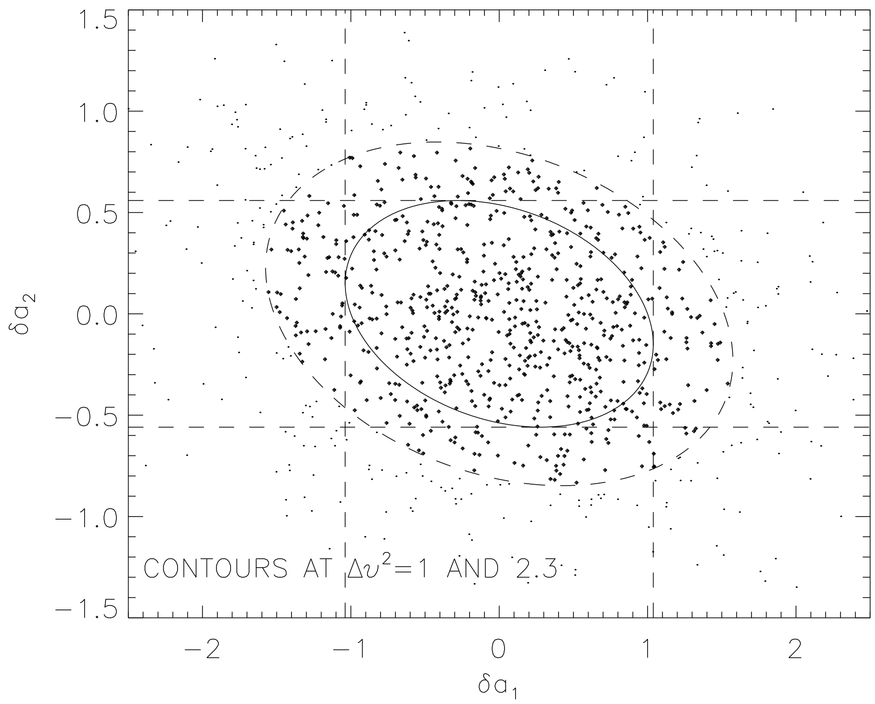

Fig. 2.— Illustrating the meaning of variance and covariance between $(a1, a2)$ for our numerical example. See text for discussion.

We illustrate these concepts for the (a_1, a_2) parameters for our numerical example. We subtracted 7.75 from all times so that the covariance would be small enough to illustrate the difference between the tangents to the ellipses and the end points of the ellipses. Contours are calculated as described in §10.5 and are at $\Delta\chi^2 = 1$ and 2.3. The dashed horizontal and vertical lines are at $\delta_a = \pm\sigma_a$.

First consider the pair of vertical lines, which are drawn at $\delta a_1 = \pm\sigma_{a_1}$, where σ is the square root of the variance of the parameters as described in equations 19, 30, 55, 56, and 66. If the datapoints were projected downward, i.e. if we take small strips δa_1 and integrate over δa_2, the pdf of δa_1 is Gaussian; ditto for the other coordinate. Thus, 68% of the points lie between these dashed lines. This is what we mean by the phrase "being interested in knowing about a_1 without regard to a_2". If we allow a_2 to vary so as to minimize χ^2 as we consider departures δa_1, then the pdf of δa_1 has dispersion σ_{a_1}. Alternatively, we can say that in a large number of experiments, the pdf of δa_1 follows a chi squared pdf with one degree of freedom if we don't care what happens to δa_2.

If, however, we are concerned about the pair, then we must look not at the projection down one axis or the other, but rather at the two-dimensional distribution. This is characterized by the tilted ellipses. Here, for a large number of experiments, the pair (a_1, a_2) follows a chi square distribution with 2 degrees of freedom (if we don't care about a_0; if we do, it's 3 degrees of freedom and the ellipse becomes an ellipsoid, but this is very hard to plot!). For $\nu = 2$, 68.3% of the points lie within $\Delta\chi^2 = 2.3$, where we have drawn the outer contour in Figure 2. The points inside this ellipse are darker; 68.3% of the points lie within that ellipse.

The best description of the specifics of calculating these ellipsoids is in BR §11.5 (Confidence Intervals, Confidence Levels for Multiparameter Fits). To describe it, we'll talk specifically about our numerical example, which has $M = 4$ measurements and $N = 3$ unknowns. The unknowns are $\mathbf{a} = [a_0, a_1, a_2]$. We'll first begin by discussing the case of single parameter; then we'll generalize.

10.2. Calculating the uncertainties of a single parameter

First, suppose we want to know the value σ_{a_0} without regard to the values of a_1 and a_2. Having already done the solution, we know the chi-square value of a_0 so we consider variations δa_0 around this best value.

Pick a value of δa_0 and redo the least squares solution for $[a_1, a_2]$. This gives a new value for χ^2 which is, of course, larger than the minimum value that was obtained with $\delta a_0 = 0$. Call this difference $\Delta\chi^2_{\delta a_0}$. Determine the dependence of $\Delta\chi^2_{\delta a_0}$ upon δa_0 and find the value of δa_0 such that $\Delta\chi^2_{\delta a_0} = 1$. This is the desired result, namely the value σ_{a_0} without regard to the values of a_1 and a_2.

This value is $\sigma^2_{a_0} = [\alpha]^{-1}_{00}$, the same result quoted in equation 56.

Consider now what you've done in this process. For each least squares fit you used a trial value of δa_0. In specifying δa_0 you had exactly one degree of freedom because you are fixing one and only one parameter. Having done this, you could do a large number of experiments (or Monte Carlo trials) to determine the resultant distribution of $\Delta\chi^2_{\delta a_0}$. It should be clear that this distribution follows a chi squared distribution with one degree of freedom ($\nu = 1$). So the uncertainty σ_{a_0} is that value for which $\Delta\chi^2_{\delta a_0} = 1$. (The chi square fit for the other two parameters has $M - 2$ degrees of freedom, but this is irrelevant because—by hypothesis—you don't care what happens to those variables.)

10.3. Calculating the uncertainties of two parameter

Suppose we want to know the value $(\sigma_{a_0}, \sigma_{a_2})$ without regard to the value of a_1. Now we consider variations $(\delta a_0, \delta a_2)$ around the best values (a_0, a_2).

Pick values for $(\delta a_0, \delta a_2)$ and redo the least squares solution for a_1. This gives a new value for

χ^2 which is, of course, larger than the minimum value that was obtained with $(\delta a_0, \delta a_2) = 0$. Call this difference $\Delta\chi^2_{(\delta a_0, \delta a_2)}$. As above, this follows a chi square distribution, but now with $\nu = 2$. Determine the dependence of $\Delta\chi^2_{(\delta a_0, \delta a_2)}$ upon $(\delta a_0, \delta a_2)$ and find the set of values of $(\delta a_0, \delta a_2)$ such that $\Delta\chi^2_{(\delta a_0, \delta a_2)} = 2.3$. This is the desired result, namely the ellipse within which the actual values $(\delta a_0, \delta a_2)$ lie with a probability of 68.3%, without regard to the value of a_1.

These values can be defined in terms of the curvature matrix $[\alpha]$, as we discuss below.

Consider now what you've done in this process. For each least squares fit you used trial values of $(\delta a_0, \delta a_2)$. In specifying them you had exactly two degrees of freedom because you are fixing two parameters. This distribution follows a chi squared distribution with two degree of freedom ($\nu = 2$). So the uncertainty σ_{a_0} is that value for which $\Delta\chi^2_{\delta a_0} = 2.3$, which follows from the integrated probability for the chi squared distribution for $\nu = 2$. (The chi square fit for the third parameter a_1 has $M - 1$ degrees of freedom, but again this is irrelevant.)

One can expand this discussion in the obvious way. Consider finally...

10.4. Calculating the uncertainties of three parameter

Suppose we want to know the values of all three parameters (or, generally, all N parameters). Then we pick trial values for all three. There is no least squares fit for the remaining parameters, because there are none. For each combination of the three (or N) parameters we obtain $\Delta\chi_{\mathbf{a}}$, which defines a 3- (or N-) dimensional ellipsoid. This follows a chi squared distribution with $\nu = 3$ (or N). We find the (hyper)surface such that $\Delta\chi_{\mathbf{a}}$ is that value within which the integrated probability is 68.3%. This defines the (hyper)surface of $\sigma_{\mathbf{a}}$.

10.5. Doing these calculations the easy way

The obvious way to do the calculations described above is to set up a grid of values in the parameters of interest (δa_n); perform the chi squared fit on the remaining variables, keeping track of the resulting grid of χ^2; and plot the results in terms of a contour plot (for two parameters of interest) or higher dimensions.

There's an easier way which is applicable unless you are doing a nonlinear fit and the parameter errors are large. The curvature matrix ($[\alpha]$ of equation 52d, the same as **XXW**) contains the matrix of the second derivatives of χ^2 with respect to all pairwise combinations of δa_n, evaluated at the minimum χ^2; it's known as the curvature matrix for this reason. Clearly, as long as the Taylor expansion is good we can write

$$\Delta\chi^2_{\mathbf{a}} = \delta\mathbf{a}^{\mathbf{T}} \cdot [\alpha] \cdot \delta\mathbf{a} \tag{68}$$

Knowing the curvature matrix, we don't have to re-do the fits as we described above. Rather, we can use the already-known matrix elements. Use the following procedure [see NM §15.6 (Probability Distribution of Parameters in the Normal Case)]. Follow the steps

1. Decide which set of parameters you are interested in; call this number N_i and denote their vector by $\mathbf{a_i}$. Here we use the above example and consider $N_i = 2$ and $\mathbf{a_i} = [a_0, a_2]$.

2. From the $N \times N$ covariance matrix $[\alpha]^{-1}$, extract the rows and columns corresponding to the N_i parameters and form a new $N_i \times N_i$ covariance matrix $[\alpha]_i^{-1}$; in our case the original covariance matrix is

$$[\alpha]^{-1} = \mathbf{XXI} = \left[\begin{array}{ccc} 1156.8125 & -303.000 & 18.4375 \\ -303.000 & 81.000 & -5.000 \\ 18.4375 & -5.000 & 0.31250 \end{array} \right] \tag{69a}$$

and it becomes

$$[\alpha]_i^{-1} = \mathbf{XXI} = \left[\begin{array}{cc} 1156.8125 & 18.4375 \\ 18.4375 & 0.31250 \end{array} \right] . \tag{69b}$$

3. Invert this new covariance matrix to form a new curvature matrix $[\alpha]_i$. The elements differ from the those in the original curvature matrix.

4. As usual, we have

$$\Delta\chi^2_{\mathbf{a_i}} = \delta\mathbf{a_i^T} \cdot [\alpha]_{\mathbf{i}} \cdot \delta\mathbf{a_i} \tag{70}$$

so find the locus of $\mathbf{a_i}$ such that the integrated probability of $\Delta\chi_{\mathbf{a_i}}$ for $\nu = N_i$ contains 68.3% of the space; e.g. for $\nu = 2$ this is $\Delta\chi_{\mathbf{a_i}} = 2.3$.

You may well wonder why, in steps 2 and 3, you need to derive a new curvature matrix from the extracted elements of the original covariance matrix. Why not just use the extracted elements of the original curvature matrix? To understand this, read the NM's discussion surrounding equation (15.6.2). Good luck.

10.6. Important comments about uncertainties

Having said all the above, we offer the following important *Comments*:

- The easiest way to calculate these (hyper)surfaces is to set up a grid in N_i-dimensional space of trial values for $\delta\mathbf{a_i}$ and use a contour plot or volume plot package to plot the loci of constant $\Delta\chi^2_{\mathbf{a_i}}$.

- The procedure described in §10.5 works well for linear fits, or nonlinear fits in which the $\sigma_{\mathbf{a}}$ are small so that $\Delta\chi^2$ is well-approximated by the second derivative curvature matrix. This is not necessarily the case; an example is shown in BR Figure 11.2. Here, the higher-order curvature terms are important and it's better to actually redo the fit for the grid of trial values of $\mathbf{a_i}$ as described above in §10.2, 10.3, and 10.4.

- The variance (i.e., uncertainty squared) of the derived parameters $\mathbf{a}$ depends *only on the elements in the covariance matrix* $[\alpha]^{-1}$. These, in turn, depend only on the curvature matrix $[\alpha]$. These, in turn, depend only on $\mathbf{X}$. This matrix $\mathbf{X}$ is the matrix of the quantities that are *known exactly*. For example, we began with the example in which the elements of $\mathbf{X}$ were the times at which the measurements were taken.

  Generally, then, the curvature and covariance matrix elements depend on the *locations* of the datapoints (the ensemble of t_m in equation 1) but not on the *measured values* (the ensemble of θ_m in equation 1). And on your adopted values for $\sigma_{meas,m}$. Because of this...

- Think *before* making your measurements about the covariance matrix and how to minimize the off-diagonal elements. By taking measurements at *well-chosen* times, or *well-chosen* values of the independent variable x_m whatever it is, you can really optimize the accuracy-to-effort ratio! For example, in our numerical example if you can get a few measurements at negative times you'll efforts will be repaid in terms of much better accuracy for the y-intercept.

11. REJECTING BAD DATAPOINTS II.: STETSON'S METHOD PLUS CHAUVENET'S CRITERION

Chauvenet's criterion is an on-off deal: either you include the datapoint or you don't. This makes sense from a philosophical point of view: either a datapoint is good or not, so you should either include it or exclude it.

However, when doing a nonlinear fit this presents a problem. As you iterate, the solution changes, and a given datapoint can change from being "bad" to "good". Or vice-versa. You can imagine being in a situation in which the iteration oscillates between two solutions, one including a particular datapoint and the other excluding it; the solution never converges, it just keeps chasing its tail.

Enter Stetson's beautiful technique[5]. Stetson reasons that we shouldn't have an on-off criterion. Rather, it should relieve a datapoint of its influence adiabatically: as its residual gets larger and

[5]Stetson is one of those anomalies, a *true* expert on fitting. He invented many of the stellar photometry routines

larger, its weight gets smaller and smaller. With this, in nonlinear fitting all datapoints are always included and their weights automatically adjust as the fit parameters home into their correct values. And you can't get into the chasing-tail syndrome that can happen with the strict on-off inclusion.

11.1. Stetson's sliding weight

Stetson recommends using a sliding weight. To explain this, we review the ML concept of chi square fitting. We write equations for the case of a single parameter to keep things simple, but it works for multiparameter fits too. In chi square fitting, we define chi square as

$$\chi^2 = \sum_{m=0}^{M-1} \frac{(y_m - af_{(x_m)})^2}{\sigma_m^2} \tag{71a}$$

and we minimize χ^2 by setting its derivative with respect to each parameter a equal to zero:

$$\frac{d\chi^2}{da} = -2 \sum_{m=0}^{M-1} \frac{f_{(x_m)} \Delta y_m}{\sigma_m^2} \tag{71b}$$

Here $\Delta y_m = (y_m - af_{(x_m)})$. This gives

$$\sum_{m=0}^{M-1} \frac{f_{(x)} \Delta y_m}{\sigma_m^2} = 0 \tag{71c}$$

Now we wish to modify this equation by introducing a weight $w_{(|\Delta y_m|)}$ that makes datapoints with large $|\Delta y_m|$ contribute less, so it reads like this:

$$\sum_{m=0}^{M-1} \frac{w_{(|\Delta y_m|)} f_{(x_m)} \Delta y_m}{\sigma_m^2} = 0 \tag{72}$$

It's clear that we need the following properties for $w_{(\Delta y_m)}$:

1. $w_{(\Delta y_m)} = w_{(|\Delta y_m|)}$, meaning simply that it should depend on the absolute value of the residual and not bias the solution one way or the other.

2. $w_{(|\Delta y_m|)} \rightarrow 1$ as $|\Delta y_m| \rightarrow 0$, meaning that datapoints with small residuals contribute their full weight.

---

used in *daophote*, all of which use least squares techniques. He provides a lively, engaging discussion of many fascinating and instructive aspects in his website: *http://nedwww.ipac.caltech.edu/level15/Stetson/Stetson4.html.*

3. $w_{(|\Delta y_m|)} \to 0$ as $|\Delta y_m| \to \infty$, meaning that datapoints with large residuals contribute nothing.

Stetson recommends

$$w_{(|\Delta y_m|)} = \frac{1}{1 + \left(\frac{|\Delta y|}{\alpha\sigma}\right)^{\beta}} \tag{73}$$

This function $w_{(|\Delta y_m|)}$ has the desired properties. Also, for all β it equals 0.5 for $|\Delta y_m| = \alpha\sigma$. As $\beta \to \infty$ the cutoff gets steeper and steeper, so in this limit it becomes equivalent to a complete cutoff for $|\Delta y_m| > \alpha\sigma$.

Stetson recommends $\alpha = 2$ to 2.5, $\beta = 2$ to 4 on the basis of years of experience. Stetson is a true expert and we should take his advice seriously; he provides a vibrant discussion to justify these choices in real life, including an interesting set of numerical experiments.

However, for large M I see a problem with the choice $\alpha = 2$ to 2.5. For large β, for which the cutoff is sharp, it seems to me that the cutoff should duplicate Chauvenet's criterion. Referring to equation 42, this occurs by setting

$$\alpha = \mathrm{erf}^{-1}\left(1 - \frac{1}{2M}\right) \tag{74}$$

and I recommend making this change, at least for problems having reasonably large M; this makes α larger than Stetson's choice. I'm more of a purist than Stetson, probably because I'm a radio astronomer and often fit thousands of spectral data datapoints that are, indeed, characterized mainly by Gaussian statistics. Stetson is an optical astronomer and probably sees a lot more departures from things like cosmic rays. Nevertheless, in a CCD image with millions of pixels, of which only a fraction are characterized by nonGaussian problems such as cosmic ray hits, it seems to me only reasonable to increase α above Stetson's recommended values by using equation 74.

11.2. Implementation of the weight in our matrix equations

Clearly, implementing Stetson's method requires a weighted fit, so you have to use the chi squared technique discussed in §9.5. There equation 57 defines a matrix of weights (which is diagonal) in which

$$\mathbf{W}_{m,m} = \frac{1}{\sigma_m} \tag{75}$$

Comparing this with equation 71c, it's clear what to do: we modify this equation to read

$$\mathbf{W}_{m,m} = \frac{w_m^{1/2}}{\sigma_m} \tag{76}$$

where the weight w_m is defined in equation 73.

Now you must not forget here that the solution depends on the weights w_m, which in turn depend on the solution. Thus when you implement this technique you must iterate until the solution converges by not changing.

12. MEDIAN, INSTEAD OF LEAST SQUARE, FITTING

Least square fitting minimizes the square of the residuals. This means that datapoints having large residuals contribute importantly to the fit. If these datapoints are really bad, you've got problems; this is why it's important to get rid of outliers! Sometimes you're faced with a set of datapoints that look bimodal: most datapoints have a Gaussian-like pdf, and many lie outside the main distribution; sometimes it's difficult to decide "where to draw the line". Or you might have nonGaussian statistics. In these cases, using least squares is not a great idea because least squares gives greatest weight to the datapoints having the largest residuals, but you don't know what the residuals are until after you've done the fit—and the fit is influenced, and maybe even dominated, by the outliers!

In these cases most astronomers use the median[6]. The median is the solution for which there as many positive as negative residuals, *irrespective of how big they are*. As long as the discrepant datapoints are symmetrically distributed with respect to sign, the median works well. In fact, for *any* pdf that is symmetrically distributed with respect to sign, the median is a good solution because it doesn't depend at all on the details of the pdf. If the statistics are nonGaussian then the error of the median is somewhat greater than that of the mean (by a factor of something like $\frac{\pi}{2}$; I forget, but it's straightforward to calculate) but, when the datapoints are contaminated by nonGaussian statistics, this is often less than the error introduced by the least squares technique.

12.1. The median and the double-sided exponential pdf

In fact, there is a specific pdf for which the median is the *theoretically correct* solution: the double-sided exponential. Here the pdf of the measured datapoints is

$$p_{(\Delta y_m)} = \frac{e^{-|\Delta y_m|/\sigma_m}}{2\sigma_m} \tag{77}$$

[6]Stetson recommends, instead of the median, his sliding weight technique. You should read his website

where, again, $\Delta y_m = (y_m - af_{(x_m)})$. For this, the logarithm of the likelihood function is (we exclude the term involving $\log \Pi_{m=0}^{M-1} \frac{1}{\sigma_m}$ for simplicity)

$$\mathcal{L}_{(\Delta y_m)} = \log(L_{(\Delta y_m)}) = -\sum_{m=0}^{M-1} \left[\frac{|y_m - af_{(x_m)})|}{\sigma_m} \right] \tag{78a}$$

The absolute value signs are horrible to deal with, so we rewrite this as

$$\mathcal{L}_{(\Delta y_m)}) = \sum_{\Delta y_m > 0} \frac{y_m - af_{(x_m)}}{\sigma_m} - \sum_{\Delta y_m < 0} \frac{y_m - af_{(x_m)}}{\sigma_m} \tag{78b}$$

Now we take the derivative of $\mathcal{L}$ with respect to a and set it equal to zero to find the maximum. This gives

$$\frac{dL}{da} = \sum_{\Delta y > 0} \frac{f_{(x_m)}}{\sigma_m} - \sum_{\Delta y < 0} \frac{f_{(x_m)}}{\sigma_m} = 0 \tag{79}$$

Amazing—neither Δy_m nor a appears!

Consider first the case $f_{(x_m)} = 1$; this is like an average. If all σ_m are identical, this says: the proper solution is the median, for which half the datapoints have positive residuals! More generally, we have to define a generalized median where each datapoint is weighted by its $\frac{f_{(x_m)}}{\sigma_m}$. Can you show that the median of the residuals is zero?sss

12.2. Doing a "chi square" fit for the double-sided exponential

We can use our standard chi square technique to do a median fit by adjusting the weights. To see how, we rewrite equation 72 here; this is the equation that has Stetson's additional weight $w_{(|\Delta y_m|)}$

$$\sum_{m=0}^{M-1} \frac{w_{(|\Delta y_m|)} f_{(x_m)} \Delta y_m}{\sigma_m^2} = 0 \tag{80}$$

We need to devise weight $w_{(|\Delta y_m|)}$ so that this equation looks like equation 79. By inspection, we require

$$w_m = \frac{\sigma_m}{|\Delta y_m|} \tag{81}$$

so that in equations 72 and 80 we have

$$\frac{w_m}{\sigma_m^2} = \frac{1}{\sigma_m |\Delta y_m|} \tag{82}$$

and in weight matrix **W** of equations 57 and 76 we use the square root of the above quantity.

This works for multivariate fits using any set of functions $f_{(x_m)}$. However, we caution again that the weights w_m depend on the parameters **a** and the parameters depend on the weights, so you must iterate. In the experiments I've performed the iteration is slow (but sure). And at the end you will certainly find that the reduced chi square $\widehat{\chi}^2$ will be nowhere near unity; nevertheless, you should use errors calculated χ^2, *not* forcing $\widehat{\chi}^2$ to equal unity (so use equation 56 instead of 55). [**THIS STATEMENT IS ALMOST CERTAINLY CORRECT BUT NEEDS TO BE CHECKED WITH NUMERICAL EXPERIMENTS...**].

12.3. IDL's resources for median fitting

There are two cases for which IDL provides median fitting. One is IDL's **median** function, which takes the median of a bunch of datapoints. It's great if there's no functional dependence on x_m. The other is **ladfit** ("least absolute deviation fit"). These are quick and you don't have to do any iterations. And they give a slightly more accurate answer than the above technique; this is because IDL is careful about the exact definition of the median, e.g. when there is an even number of datapoints.

However, with IDL's routines there are no errors given; there is no possibility for differing σ_m; there is no covariance matrix. And, of course, you can't fit anything more complicated than a straight line.

13. FITTING WHEN ALL VARIABLES 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. If you go ahead with a standard least squares fit when there are errors in both coordinates, the slope will be systematically too small, as we discuss briefly below.

Thanks to Stetson, the ML formulation of this problem is straightforward. Nevertheless, as far as I know the proper formulation is rather recent; Stetson's treatment is the first I've seen. Before reviewing Stetson's formulation, I want to warn you about some *incorrect* formulations:

1. Taylor §8.4 discusses the case incorrectly, arguing that you can account for x-variance σ_{xm}^2 by increasing the y-variance by the usual error propagation, i.e. define an equivalent y-variance $\sigma_{ym}^2(equiv) = [\sigma_{ym}^2 + (a_1\sigma_{xm})^2]$, where a_1 is the slope. As we mentioned, this procedure leads to a slope that is systematically too small.

2. Isobe et al (1990, ApJ 364, 104) discuss the case incorrectly. Look in particular at their Section V, where they make 5 numbered recommendations. Two of these are incorrect:

   (a) Number 3 says, in essence, that if you have measurement errors in y but not in x, and want to predict x from y in some future dataset, that you should least squares fit the x values (which have no errors) to the y. This is *flat wrong.* Again, it leads to a slope that is systematically too small. The proper procedure is to fit y to x in the standard way, which is consistent with the ML formulation and gives the right answer; then use the resulting parameters, whose errors you know about, to predict x from y in the future.

   (b) Number 4 says that if both x and y have errors, and your main focus is finding the true slope, you should use their "bisector" method. I won't explain this because this concept is wrong.

Stetson has a beautiful discussion of the problem, which is obviously correct—both because it makes sense and because numerical experiments show that the derived parameters don't depend on whether you consider x or y the independent variable. With his technique you are not restricted to fitting a straight line (a one-degree polynomial). NM §15.3 provides another method, which is more complicated and is restricted to a straight line. Here we review Stetson's method.

13.1. A preliminary: Why the slope is systematically small

Why is the derived slope systematically too small if you use the standard least squares technique when both variables have errors? To see this, take a look back at equation 5, where we explicitly write the normal equations for fitting a straight line of the form $As_m + Bt_m = \theta_m$. To focus the discussion and make it easy, replace that problem with a single-parameter solution for only the slope B, and use the usual variables (x, y) in place of (t, θ). Then we are fitting the set of M equations

$$Bx_m = y_m \tag{83a}$$

the set of two normal equations becomes just the single equation

$$B[x^2] = [xy] \tag{83b}$$

or, writing out the sums explicitly,

$$B = \frac{\sum_{m=0}^{M-1} x*_m y_m}{\sum_{m=0}^{M-1} x*_m^2} \tag{83c}$$

Here we use the star to designate the perfectly-known independent variable $x*_m$. It is important to realize that the x_m that appear in this equation are the perfectly-known ones $x*_m$; this is a fundamental tenet of least squares fitting, which comes from the concept and principle of maximum likelihood ML.

Because B is defined by the $x*_m$ and we are asking what happens when we use the imperfectly known x_m instead, let us reduce the problem to the essence and imagine that y_m is perfectly known, i.e. $y_m = y*_m$; and that

$$x*_m = x_m - \delta x_m \tag{84}$$

where δx_m is the observational error in point m. If we do standard least squares on this situation, as recommended by Taylor and Isobe et al, then we (incorrectly) rewrite equation 83c to read

$$B = \frac{\sum_{m=0}^{M-1} x_m y_m}{\sum_{m=0}^{M-1} x_m^2} \tag{85a}$$

that is, using x_m instead of $x*_m$ (because we don't know what $x*_m$ is!). Substituting equation 84, and remembering that $y_m = y*_m = Bx*_m$, we have

$$B = \frac{\sum_{m=0}^{M-1} x*_m (x*_m + \delta x_m)}{\sum_{m=0}^{M-1} (x*_m^2 + 2x*_m \delta x_m + \delta x_m^2)} \tag{85b}$$

Now all terms having δx_m sum to zero because the errors are distributed symmetrically around zero. But the denominator contains δx_m^2. The denominator is irrevocably increased by this term, which decreases the derived value of B from its true value. Yes, this is only a second-order effect, but it matters—after all, χ^2 is a second-order quantity! Try some numerical experiments!

13.2. Stetson's method

13.2.1. Philosophy

To make things clear, we discuss the case of a straight line. Thus we assume

$$y* = a_0 + a_1 x* \tag{86}$$

where the $*$ indicates the true quantity without errors. The measured points differ from the true ones:

$$\delta x_m = x_m - x*_m \quad ; \quad \delta y_m = y_m - y*_m \tag{87}$$

and we define, as usual, the apparent residual in y_m as

$$\Delta y_m = y_m - a_1 x_m \tag{88}$$

Note that this differs from the *true* residual in y_m, which is δy_m; the apparent residual Δy_m assumes that x_m has no error.

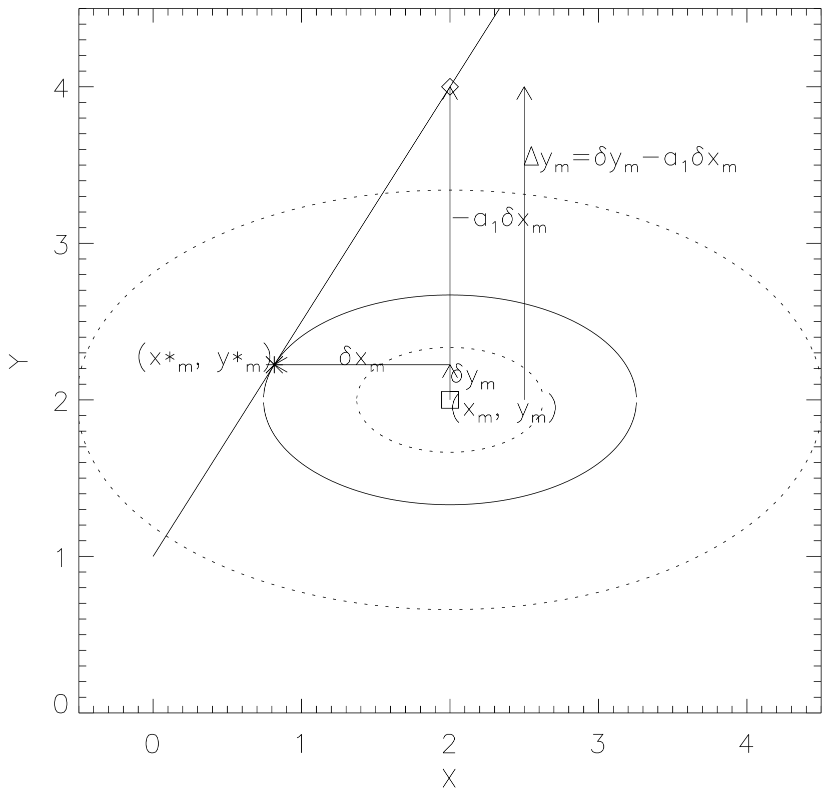

Fig. 3.— Illustrating the discussion of fitting when both variables have errors. Here, as usual in this document, differences are “measured minus true”, e.g. $\Delta y_m = y_m - y*_m$, etc.; in the figure, $\Delta y_m < 0$, $\delta y_m < 0$, $\delta x_m > 0$.

Look at Figure 3. There (x_m, y_m) is a measured point. The ellipse centered on it has (x, y)

axial ratio (σ_x, σ_y) and traces out a the locus where the point's contribution to χ^2 is constant; call this contribution χ^2_m. The straight line is the fitted straight line, assumed to be perfectly correct, i.e. the line given by equation 86. The most probable place where the observed point (x_m, y_m) would lie if it had no errors is $(x*_m, y*_m)$; it lies on the smallest ellipse centered on (x_m, y_m), as pictured. This is the ellipse having the smallest possible χ^2_m. The true differences between the observed and most probable points are $(\delta x_m, \delta y_m)$ given by equation 87 above, where these are measured from the center to the starred point on the ellipse.

It's easy to calculate $(\delta x_m, \delta y_m)$ in terms of Δy_m. These are the distances from the measured to the starred point. We realize that the starred point must satisfy two criteria:

1. The straight line fit is tangent to the ellipse, so the slopes much match. The ellipse has equation

$$\frac{\delta x_m^2}{\sigma_{xm}^2} + \frac{\delta y_m^2}{\sigma_{ym}^2} = const = \chi^2_m \tag{89}$$

so the slope is

$$\frac{d\delta y_m}{d\delta x_m} = -\frac{\sigma_{ym}^2}{\sigma_{xm}^2}\frac{\delta x_m}{\delta y_m} \tag{90}$$

and this, evaluated at $(x*_m, y*_m)$ must equal a_1.

2. The actual values must match. Rather than solve for these, we can solve in terms of Δy_m directly.

From the figure it's clear that

$$\Delta y_m = \delta y_m - a_1 \delta x_m \tag{91}$$

so combining equations 90 and 91 we get

$$\delta x_m = \frac{-\frac{\Delta y_m}{a_1}}{1 + \frac{\sigma_{ym}^2}{a_1^2 \sigma_{xm}^2}} \tag{92a}$$

and

$$\delta y_m = \frac{\Delta y_m}{1 + \frac{a_1^2 \sigma_{xm}^2}{\sigma_{ym}^2}} \tag{92b}$$

You could minimize χ^2 by brute force. You would begin with the standard least squares solution to get an estimate of the correct parameter values. Then you'd make up a grid in the (a_0, a_1) plane to find the location of the minimum in χ^2, which is

$$\chi^2 = \sum_{m=0}^{M-1} \chi_m^2 = \sum_{m=0}^{M-1} \frac{\delta x_m^2}{\sigma_{x_m}^2} + \frac{\delta y_m^2}{\sigma_{y_m}^2} \tag{93}$$

which gives the parameters (a_0, a_1). The quantities $(\delta x_m^2, \delta y_m^2)$ are given by equation 92. The errors are given by the usual $\Delta\chi^2$ contours as discussed in §10.

13.2.2. Direct solution

Using the principle of maximum likelihood (ME), we find the minimum χ^2 by taking the derivative of χ^2 with respect to each parameter a_n and setting the derivative equal to zero. To illustrate, we work this through for the slope. We are fitting for the slope in the equation

$$y_m = a x_m \tag{94}$$

for which χ^2 is given by equation 93. Express $(\delta x_m, \delta y_m)$ in terms of Δy_m, as in equations 92, and rewrite equation 93 to read

$$\chi^2 = \sum_{m=0}^{M-1} \frac{(y_m - a x_m)^2}{a^2 \sigma_{x_m}^2 + \sigma_{y_m}^2} = \sum_{m=0}^{M-1} \frac{\Delta y_m^2}{a^2 \sigma_{x_m}^2 + \sigma_{y_m}^2} \tag{95}$$

Differentiate with respect to a and set equal to zero. After cancelling extraneous factors, and using equation 92a, we get

$$\sum_{m=0}^{M-1} \frac{(y_m - a x_m)(x_m - \delta x_m)}{a^2 \sigma_{x_m}^2 + \sigma_{y_m}^2} = 0 \tag{96}$$

The term $a x_m (x_m - \delta x_m)$ defines the relevant diagonal matrix element in the curvature matrix, and corresponds to the lower right element $(B[t^2])$ in equation 5; the term $y_m (x_m - \delta x_m)$ corresponds to $[t\theta]$ in that equation. In other words, if we had standard least squares and *no* errors in x, equation 96 would read

$$\sum_{m=0}^{M-1} \frac{(y_m - a x_m)(x_m)}{\sigma_{y_m}^2} = 0 \tag{97}$$

Looking at these and, also, referring back to §3, it's easy to see how to generalize this and write it in matrix notation. First, note that each datapoint has a net total error given by the denominator of equation 95, and in general—i.e., when we fit something other than a straight line—the slope a varies from point to point so that, even if the (σ_x, σ_y) are constant, the net total error is not. This means you need to use the weighted fit technique of §9.2 and 9.5 with

$$\sigma_{meas,m}^2 = (a_m^2 \sigma_{x_m}^2 + \sigma_{y_m}^2) \tag{98a}$$

where the local slope a_m is evaluated at $(x_m - \delta x_m)$

$$a_m = \left.\frac{\partial y}{\partial x}\right|_{(x_m - \delta x_m),(y_m - \delta y_m)} \tag{98b}$$

We repeat all this to be perfectly explicit:

1. You are fitting

$$\mathbf{X} \cdot \mathbf{a} = \mathbf{Y} \tag{99}$$

   In $\mathbf{X}$, each column n contains a different function of x called $f_n(x)$. The entry for $(col, row) = (n, m) = f_n(x_m)$.

2. In our normal equations we have the factor $(x_m - \delta x_m)$ instead of x_m, which means that in forming the matrix normal equations we need to define a new matrix

$$\mathbf{X\_MOD} = \mathbf{X} - \delta\mathbf{X} \tag{100}$$

   where the elements of $\delta\mathbf{X}$ are evaluated using equation 92. That is, $\delta\mathbf{X}_{m,n} = f_n(x_m - \delta x_m)$. [Here we use IDL's convention (m, n) instead of the conventional mathematical one (n, m).] If it's a straight-line fit, the slope is the same everywhere. If it's a more general fit, the slope is the local slope derived using equation 98; this slope can be calculated either analytically or numerically.

3. Define the weighted versions of $\mathbf{X}$, $\mathbf{X\_MOD}$, and $\mathbf{Y}$

$$\mathbf{XW} = \mathbf{W}\ \#\#\ \mathbf{X} = \mathbf{W} \cdot \mathbf{X} \tag{101a}$$

$$\mathbf{X\_MODW} = \mathbf{W}\ \#\#\ \mathbf{X\_MOD} = \mathbf{W} \cdot \mathbf{X\_MOD} \tag{101b}$$

$$\mathbf{YW} = \mathbf{W}\ \#\#\ \mathbf{Y} = \mathbf{W} \cdot \mathbf{Y} \tag{101c}$$

   where $\mathbf{W}_{m,m} = \frac{1}{\sigma_{meas,m}}$ (from equation 98a).

4. Then the matrix normal equation is

$$[\alpha\_mod] \cdot \mathbf{a} = [\beta\_mod] \tag{102}$$

where

$$[\alpha\_mod] = \mathbf{X\_MODW^T} \cdot \mathbf{X} \tag{103a}$$

$$[\beta\_mod] = \mathbf{X\_MODW^T} \cdot \mathbf{Y} \tag{103b}$$

But, you ask, how can we form the matrix $\delta\mathbf{X}$ if we don't know the slope a? The answer: iterate. That is,

1. Solve $y_m = a_0 + a_1 x_m$ using standard least squares or chi square; or use any other technique to get an initial guess for the parameters $\mathbf{a}$. Calculate the local slopes using (x_m, y_m).
2. Determine the Δy_m and calculate δx_m using equation 92a.
3. Calculate the matrices **X_MODW**, $[\alpha\_mod]$, and $[\beta\_mod]$ as above.
4. Solve for new parameters $\mathbf{a}$.
5. Calculate the local slopes using $(x_m - \delta x_m, y_m - \delta y_m)$.
6. Loop back and repeat steps 2, 3, 4,and 5 until convergence.
7. After convergence, calculate the uncertainties, as usual, by the equivalent of equation 55, i.e.

$$\mathbf{s_a^2} = \widehat{\chi}^2 diag\{[\alpha\_mod]\} \tag{104}$$

13.3. Commentary on Stetson's solution

13.3.1. This solution is general and makes sense

This solution makes sense. Consider standard least squares in which $\sigma_{x_m} \rightarrow 0$; here we should revert to standard least squares. Also, when the slope is zero it's like taking an average of y_m, so it shouldn't matter what σ_x is. Equation 92a satisfies these expectations because the denominator $\rightarrow \infty$: the ellipse becomes tall and thin so $\delta x_m \rightarrow 0$.

Similarly, if $\sigma_{y_m} \rightarrow 0$ or $a \rightarrow \infty$ then the ellipse becomes very short and fat: $\delta x_m \rightarrow \frac{\Delta y_m}{a_1}$, meaning that *all* of the correction goes to δx_m. This is equivalent to y being the independent variable, and you can solve it this way, but of course it is much more efficient to use standard least squares with y *actually* being the independent variable.

Finally, there's no reason why we had to use a straight-line fit. It could have been any function; in an arbitrary case, we replace the slope by the *local* slope, namely the derivative of the function evaluated at the current value $x_m + \delta x_m$. So this technique is *general*. In his website, Stetson provides an example of fitting a *circle* to points in (x, y)!

13.3.2. Yet another point I do not understand

First, I found Stetson's website discussion charming and stimulating. However, I found it very difficult to glean just exactly how he suggests solving the problem! His formulation involves using differences from one iteration to the next and differs from the one I've given above.

Second, Stetson spends several pages discussing the importance of the difference between two quantities. One is Δy_m. The other is

$$\epsilon = -\delta F_m = \left.\frac{\partial F}{\partial x}\right|_{x*,y*} \delta x_m + \left.\frac{\partial F}{\partial y}\right|_{x*,y*} \delta y_m \tag{105}$$

where

$$F(x, y) = a_0 + a_1 x - y \tag{106}$$

You could put contours of constant F on Figure 3. The zero contour is where $F(x*, y*) = 0$, which is what you get with $(\delta x_m, \delta y_m) = (0, 0)$. This zero-level contour is the straight line solution of the problem. Moving away from zero gives contours of constant ϵ; these are offset from the zero contour and have the same shape.

Stetson emphasizes that the proper thing to do is use ϵ_m in place of Δy_m in equation 92a. However, after scratching my head for quite a while, I conclude that in fact these two quantities are always equal, i.e. $\epsilon_m = \Delta y_m$. I do not understand Stetson's emphasis on this point.

13.4. Generalization to Multivariate Case

The subsections above deal with the simple case of two variables. It is straightforward to generalize to the multivariate case. As in section 13.2.1, suppose you are fitting the multivariate analogy of equation 86

$$z* = a_0 + a_1 x* + a_2 y* + \ldots \tag{107a}$$

which is equivalent to

$$dz* = a_1 dx* + a_2 dy* + \ldots \tag{107b}$$

Then the discussion follows the same track except that here, instead of the surface of χ^2 being an ellipse on a two-dimensional space, it's an ellipsoid in a three-dimensinal space. And for the case of N variables it's an N-dimensional ellipsoid. We can't draw this, but the ideas embodied in Figure 3 remain: we match the slopes and values to find the smallest ellipse, i.e. the smallest χ^2, for each measured point. Equations 90 and 91 become

$$d\delta z_m = -\frac{\sigma_{z_m}^2}{\sigma_{x_m}^2}\frac{\delta x_m}{\delta z_m} d\delta x_m - \frac{\sigma_{z_m}^2}{\sigma_{y_m}^2}\frac{\delta y_m}{\delta z_m} d\delta y_m + \ldots \tag{108}$$

and

$$\Delta z_m = \delta z_m - a_1 \delta x_m - a_2 \delta y_m + \ldots \tag{109}$$

Comparing equations 107b and 109, we see that

$$a_1 = -\frac{\sigma_{z_m}^2}{\sigma_{x_m}^2}\frac{\delta x_m}{\delta z_m} \rightarrow \delta z_m = -\frac{\sigma_{z_m}^2}{a_1 \sigma_{x_m}^2}\delta x_m \tag{110a}$$

$$a_2 = -\frac{\sigma_{z_m}^2}{\sigma_{y_m}^2}\frac{\delta y_m}{\delta z_m} \rightarrow \delta z_m = -\frac{\sigma_{z_m}^2}{a_2 \sigma_{y_m}^2}\delta y_m \tag{110b}$$

$$\vdots \tag{110c}$$

There are N such equations. We then generate N equations that express Δz_m in terms of $\delta x_m, \delta y_m, \ldots$ by successively substituting for δz_m the right-hand-side versions of equations 110 into equation 109:

$$\Delta z_m = -a_1 \left[\frac{\sigma_{z_m}^2}{(a_1 \sigma_{x_m})^2} + 1\right] \delta x_m - a_2 \delta y_m - \ldots \tag{111a}$$

$$\Delta z_m = -a_1 \delta x_m - a_2 \left[\frac{\sigma_{z_m}^2}{(a_2 \sigma_{y_m})^2} + 1\right] \delta y_m - \ldots \tag{111b}$$

$$\vdots \tag{111c}$$

or, in matrix form,

$$\begin{bmatrix} -a_1\left[\frac{\sigma_{z_m}^2}{(a_1\sigma_{x_m})^2}+1\right] & -a_2 & \cdots \\ -a_1 & -a_2\left[\frac{\sigma_{z_m}^2}{(a_2\sigma_{y_m})^2}+1\right] & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \delta x_m \\ \delta y_m \\ \vdots \end{bmatrix} = \begin{bmatrix} \Delta z_m \\ \Delta z_m \\ \vdots \end{bmatrix} \tag{112a}$$

or, symbolically,

$$\mathbf{A}_m \cdot \delta_m = \boldsymbol{\Delta} z_m \tag{112b}$$

so that, as in equation 92a, we obtain $(\delta x_m, \delta y_m, \ \ldots)$ in terms of Δz_m from

$$\delta_{\mathbf{m}} = \mathbf{A}_m^{-1} \cdot \Delta z_m \tag{113}$$

Finally, as in equations 93 and 95, we compute χ^2 from

$$\chi^2 = \sum_{m=0}^{M-1} \frac{(z_m - a_1 x_m - a_2 y_m - \ldots)^2}{\sigma_{z_m}^2 + a_1^2\sigma_{x_m}^2 + a_2^2\sigma_{y_m}^2 + \ldots} = \sum_{m=0}^{M-1} \frac{\Delta z_m^2}{\sigma_{z_m}^2 + a_1^2\sigma_{x_m}^2 + a_2^2\sigma_{y_m}^2 + \ldots} \tag{114}$$

and equation 98a becomes

$$\sigma_{meas,m}^2 = \sigma_{z_m}^2 + a_1^2\sigma_{x_m}^2 + a_2^2\sigma_{y_m}^2 + \ldots \tag{115}$$

which is used, as above, to calculate $\mathbf{W}_{m,m} = \frac{1}{\sigma_{meas,m}}$.

14. USING SINGULAR VALUE DECOMPOSITION (SVD) IN IDL

Occasionally, a normal-equation matrix $[\alpha] = \mathbf{XX} = \mathbf{XXW}$ is sufficiently ill-posed that inverting it using standard matrix inversion doesn't work. In its **invert** function, IDL even provides a keyword called **status** to check on this (although I find that it is not perfectly reliable; the best indicator of reliability is to check that the matrix product $\mathbf{XX\#\#XXI} = \mathbf{I}$, the unitary matrix). In these cases, Singular Value Decomposition (SVD) comes to the rescue. For a discussion of the details of SVD, see NM §15.4.

You don't need to read the following material to use the SVD technique in IDL. Rather, you can just use IDL's documentation for **svdfit**. The only problem is that you need to fit to a function that you define in an IDL function. If you are fitting to tabular values, or to a function that is not easily expressed analytically, then you can still use **svdfit**: read on.

First let us briefly describe the technique[7]. We are fitting

$$y_m = \sum_{n=0}^{N-1} a_n f_n(x_m) \ ; \tag{116}$$

The matrix element $\mathbf{X}_{nm} = f_n(x_m)$, i.e. it contains the values of f_n evaluated at x_m. The idea is to replace the set of functions $f_n(x_m)$ with the set of matrix elements $\mathbf{X}_{nm} = f_n(m)$, i.e. to replace the independent variable x_m by its index m and the function $f_n(x_m)$ by its matrix-element counterpart $\mathbf{X}_{nm}$.

We will denote the **X** matrix by the symbol **X_SVD** because we need to put it in a common block; however, remember that **X_SVD** = **X**. You populate this matrix yourself, Often with values that are not easily expressible analytically. Here's the cookbook:

14.0.1. Step 0: Introduce a common block that will contain the **X_SVD** *matrix*

Let's call this common block **svdinput**. The relevant IDL command is **common svdinput,x_svd**. The only catch is that, in IDL, you can't use a variable before defining a common block that contains it. So it is probably best to exit IDL and re-enter it, defining this common block as the first statement.

14.0.2. Step 1: Define the function **svdfcn**

This function references the **x_svd** matrix, which you've put in common. This name of this function is an input parameter to **svdfit**. It must have two input parameters, x_m and N, the number of parameters to fit; and it must provide the N values of the parameters' functions $f_n(x_m)$ for each measurement location x_m. So we replace the set of x_m by the the indices m and $f_n(x_m)$ by the set of $\mathbf{X\_SVD}_{*,m}$, where the * means all N values.

```
function svdfcn, dataindex, N
common svdinput
return, x_svd[ *, dataindex]
end
```

[7]It is a pleasure to thank Jason Wright for telling me about this technique.

14.0.3. *Step 2: Populate the* **X_SVD** *matrix and* **Y** *vector almost as before*

Here you populate the **X_SVD** and **Y** matrices just as before in §5.2 and 9.5, with some small differences. One, the measurement uncertainties are an independent input to **svdfit** so we don't create the weighted versions of **X_SVD** and **Y**. Two, we must make **Y** a row vector instead of a column vector. Here we repeat all the IDL commands as in §9.5, but without the verbal commentary, and include this slight modification for **Y**. We use the numbers in the numerical example of §5.2 to be absolutely unambiguous about what to do. In this example, the number of unknowns $N = 3$ and the number of measurements $M = 4$.

$$\mathbf{X\_SVD} = \begin{bmatrix} 1 & 5 & 25 \\ 1 & 7 & 49 \\ 1 & 9 & 81 \\ 1 & 11 & 121 \end{bmatrix} \tag{117a}$$

$$\mathbf{Y} = [142, 168, 211, 251] \ . \tag{117b}$$

and we define **ME**, the vector of measurement errors, to be

$$\mathbf{ME} = [5., 3., 7., 4.] \ . \tag{117c}$$

where we have invented a four arbitrary values for illustrative purposes. You can, of course, use unity for all of the elements of **ME**; also, you can call **svdfit** without specifying any measurement errors, in which case it becomes a standard least squares fit. In the following we assume that you are doing a chi-square fit, i.e. following §9.5.

14.0.4. *Step 3: Call* **svdfit**

Now in in §9.5 we replace 59 to 67 (and in §5.2, equation 23 to 30) by the following:

$$dataindex = indgen(M) \tag{118a}$$

where M, we remind you, is the number of measurements; and

$$\mathbf{a = svdfit(dataindex, Y, N, measure\_errors = ME, function\_name =' svdfcn',}$$

$$\mathbf{yfit} = \mathbf{ybar}, \mathbf{chisq} = chi\_sq, \mathbf{variance} = \mathbf{vardc\_official}, \mathbf{sigma} = \mathbf{sigdc\_official},$$
$$\mathbf{covar} = \mathbf{XXWI}, \mathbf{singular} = \mathbf{singular}) \tag{118b}$$

$$\mathbf{ncov} = \mathbf{XXWI}/sqrt(\mathbf{vardc\_official\#vardc\_official}) \tag{118c}$$

If you are doing least squares fitting and setting all elements of **ME** equal to unity, then of course **chisq** is the same as the sample variance s^2.

This technique is simpler because it requires fewer IDL statements. However, you have to define the additional function and, if you are using the **X_SVD** matrix, the common block, which must be done at the beginning of the IDL session. This technique is good in those rare instances where inversion of the curvature matrix fails, but it works for all cases, of course. NM §15.4 extols its advantages and mentions its one disadvantage, which is requiring more memory (and, I'm guessing, more computer time); they recommend that you use it always. I got along without it for 40 years, perhaps because I never did any truly challenging problems, but I'll probably change my default technique to this. Note, however, you can't use **svdfit** for errors in both variables discussed in §13.2.2.

15. BRUTE FORCE CHI SQUARED AND THE CURVATURE MATRIX

15.1. Parameter Uncertainties in Brute Force Chi square fitting

There are times when "brute force" 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.

This procedure certainly provides the least squares solution. However, getting the uncertainties requires a bit more work. You can get the full set of uncertainties if you know the curvature matrix, which is the matrix of second derivatives of the pairwise parameters evaluated at the minimum χ^2. Having found the minimum, you can derive these curvatures (equal to half the second derivatives) numerically. Then, to find the uncertainties in the parameters, follow the procedure in §10.5.

15.2. Example: the single-parameter case

It is particularly easy to discuss this if there is only one parameter. Let us consider this case and assume that it is a simple weighted average (which is the least squares solution for a constant). Suppose the best fit χ^2 value for the unknown coefficient (i.e., the weighted average) is x_0 and that its variance is $\sigma_{x_0}^2$. Now consider offsets from x_0, namely $x_0 + \delta x$. Because χ^2 is minimized at x_0, it is clear that

$$\Delta\chi^2 = \chi^2_{x_0+\delta x} - \chi^2_{x_0} = \frac{1}{2}\frac{d^2\chi^2}{d\delta x^2}\delta x^2 \tag{119}$$

Generally, the curvature is the relevant element of the curvature matrix $[\alpha]$. In the particular case of a weighted average, there is just one element so

$$\frac{1}{2}\frac{d^2\chi^2}{d\delta x^2} = [\alpha_\sigma] = [\alpha_\sigma]_{00} = \sum\left(\frac{1}{\sigma^2_{meas,m}}\right) \tag{120}$$

Because we have a single parameter x_0, its variance σ_{x0}^2 is the offset δx_0^2 from x_0 that gives $\Delta\chi^2 = 1$. If we had more than one variable, we'd define ellipsoids in the manner discussed in §10.5.

Are you bothered by the fact that the curvature is independent of the data values? Then go back and read §9.3.

16. NOTATION COMPARISON WITH NUMERICAL RECIPES

I learned least squares from Appendix A of Chauvenet (1863). He didn't use χ^2 and didn't use matrix techniques, but §1 and 2 follows his development quite closely. I wrote the first version of this document before knowing of NM's treatment, which explains my orientation towards least squares instead of chi-square. I'm fortunate in this approach because it made me realize the pitfalls one can get into with chi-square, as I discuss in §9.

On the other hand, NM describe the least squares approach with some disdain in the discussion of equation (15.1.6) 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_{meas}. But you might know the relative weights, and this is a plus for Chi-square fitting. In any case, heed our warnings about chi-square fitting in §9.

In this writeup I have revised my old notation to agree, partially, with NM's. This effort wasn't completely successful because I didn't read NM very carefully before starting. To make it easier to cross-reference this document with NM, I provide the following table of correspondences (left of the double arrow is ours, to the right is theirs):

$$\mathbf{X} \longleftrightarrow \mathbf{A} \tag{121a}$$

$$\mathbf{Y} \longleftrightarrow \mathbf{b} \tag{121b}$$

$$\mathbf{X^T} \cdot \mathbf{X} = \mathbf{XX} = [\alpha] \longleftrightarrow \mathbf{A^T} \cdot \mathbf{A} = [\alpha] \tag{121c}$$

$$\mathbf{XX}^{-1} = \mathbf{XXI} = [\alpha]^{-1} \longleftrightarrow [\alpha]^{-1} = [C] = \mathbf{C} \tag{121d}$$

I use M for the number of measurements and N for the number of unknown coefficients; NM uses the opposite, so we have

$$N \longleftrightarrow M \tag{121e}$$

$$M \longleftrightarrow N \tag{121f}$$

REFERENCES

Bevington, P.R. & Robinson, D. 1992, Data Reduction and Error Analysis for the Physical Sciences (WCB/McGraw-Hill).

Chauvenet, W. 1863, A Manual of Spherical and Practical Astronomy, Dover Press.

Cowan, G. 1998, Statistical Data Analysis, Clarendeon Press.

Press, W.H., Flannery, B.P., Teukolsky, S.A., & Vetterling, W.T. 2001, Numerical Recipes (second edition), Cambridge University Press.

Taylor, J.R. 1997, An Introduction to Error Analysis, University Science Books.

This preprint was prepared with the AAS LaTeX macros v5.0.