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

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 §4, 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 (§0 to 6) 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, NR) and Taylor (1997; T97), and we update the notation to partially conform with NR. We owe sections 12 and 13 to the excellent website of Stetson, *http://nedwww.ipac.caltech.edu/level5/Stetson/Stetson4.html*. Section 14 treats the case when all measured parameters have errors; we use the very general approach of Jefferys (1980).

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

0. LEAST-SQUAREs FITTING FOR TWO PARAMETERS, AS WITH A STRAIGHT LINE.

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

First consider the least-squares fit to a straight line. Let y_m be the m^{th} measurement of the observed quantity (in this example, y_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 y_m has uncertainties in its measurement.

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

$$A + Bt_m = y_m \; . \tag{0.1}$$

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*

$$AN + B \sum t_m = \sum y_m \tag{0.2a}$$

$$A \sum t_m + B \sum t_m^2 = \sum t_m y_m \; . \tag{0.2b}$$

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

0.2. 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 0.1, $s_m = 1$. Then we rewrite equation 0.1 to include this extra dependence

$$As_m + Bt_m = y_m \; . \tag{0.3}$$

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 0.3, 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 0.3 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 0.2; the result is

$$A \sum s_m^2 + B \sum s_m t_m = \sum s_m y_m \tag{0.4a}$$

$$A \sum s_m t_m + B \sum t_m^2 = \sum t_m y_m \ . \tag{0.4b}$$

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

$$A[s^2] + B[st] = [sy] \tag{0.5a}$$

$$A[st] + B[t^2] = [ty] \ . \tag{0.5b}$$

This is, of course, precisely analogous to equation 0.2. And now it's clear how to generalize to more parameters!

1. LEAST-SQUARES 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 0.3) we simply add equivalent additional terms such as Cu_m, Dv_m, etc; and in the normal equations (equation 0.5) 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 datapoints ($M = 5$), so there must be at least five equations of condition. The generalization of equation 0.4 is the M equations

$$As_m + Bt_m + Cu_m + Dv_m = y_m \ , \tag{1.1}$$

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 y_m. We then form the four normal equations; the generalization of equation 0.5 written in matrix format is:

$$\begin{bmatrix} [ss] & [st] & [su] & [sv] \\ [ts] & [tt] & [tu] & [tv] \\ [us] & [ut] & [uu] & [uv] \\ [vs] & [vt] & [vu] & [vv] \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} [sy] \\ [ty] \\ [uy] \\ [vy] \end{bmatrix} \tag{1.2}$$

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!

2. 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

$$\mathbf{X} = \begin{bmatrix} s_0 & t_0 & u_0 & v_0 \\ s_1 & t_1 & u_1 & v_1 \\ s_2 & t_2 & u_2 & v_2 \\ s_3 & t_3 & u_3 & v_3 \\ s_4 & t_4 & u_4 & v_4 \end{bmatrix} \tag{2.1a}$$

$$\mathbf{a} = \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} \tag{2.1b}$$

$$\mathbf{Y} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \tag{2.1c}$$

so, in matrix form, the equations of condition (equation 1.1) reduce to the single matrix equation

$$\mathbf{X} \cdot \mathbf{a} = \mathbf{Y} \ . \tag{2.2}$$

The notation for these equations corresponds to NR'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 §8 and 9.

Our matrix $\mathbf{X}$ corresponds to NR's "design matrix" $\mathbf{A}$ of Figure 15.4.1, except that our elements are not divided by $\sigma_{meas,m}$, and the matrix equation of condition (equation 2.2) is identical to the expression inside the square brackets of NR'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 (§8).

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 1.2) reduce to the single equation

$$[\alpha] \cdot \mathbf{a} = [\beta] \ , \tag{2.3}$$

(NR equation 15.4.10), where

$$[\alpha] = \mathbf{X^T} \cdot \mathbf{X} \tag{2.4a}$$

$$[\beta] = \mathbf{X^T} \cdot \mathbf{Y} \ . \tag{2.4b}$$

The matrix $[\alpha]$ 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 $[\alpha]$ (that is, by $[\alpha]^{-1}$), which gives

$$\mathbf{a} = [\alpha]^{-1} \cdot [\beta] \ . \tag{2.5}$$

All of these matrix operations are trivially easy in IDL (§4).

3. UNCERTAINTIES IN THE DERIVED COEFFICIENTS

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

The first thing to do is derive the *sample* variance s^2 (the square of standard deviation, or mean error, or dispersion, etc) of the individual datapoints 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

$$s^2 = \frac{1}{M-N} \sum_{m=0}^{M-1} (y_m - \overline{y_m})^2 \ , \tag{3.1}$$

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

$$\delta y_m = y_m - \overline{y_m} \tag{3.2}$$

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 $\overline{y_m}$, which is derived from the data, instead of the *truly* correct value y^*. (In formal statistical language, y^* is the mean of the parent population from which your set of measurements is drawn.) If you used the truly correct value y^*, then the sum would be larger than when using $\overline{y_m}$. 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 [y^* and M] instead of [$\overline{y_m}$ and $(M-N)$] in equation 3.2; 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 [y^* and M] in equation 3.2? The reason is obvious: we don't know y^*! (If we did, we wouldn't be doing this analysis!)

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

$$\overline{\mathbf{Y}} = \begin{bmatrix} \overline{y_0} \\ \overline{y_1} \\ \overline{y_2} \\ \overline{y_3} \\ \overline{y_4} \end{bmatrix} \tag{3.3}$$

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

$$\overline{\mathbf{Y}} = \mathbf{X} \cdot \mathbf{a} \; . \tag{3.4}$$

Note that $\mathbf{X}$ is already defined (equation 2.1) and $\mathbf{a}$ was solved for in equation 2.5. It's convenient to define the residual matrix

$$\delta\mathbf{Y} = \mathbf{Y} - \overline{\mathbf{Y}} \tag{3.5}$$

so we can write

$$s^2 = \frac{1}{M-N}\delta\mathbf{Y}^{\mathbf{T}} \cdot \delta\mathbf{Y} \; . \tag{3.6}$$

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 §0. 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

$$\mathbf{s_a}^2 = s^2 diag\{[\alpha]^{-1}\} \; . \tag{3.7}$$

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 $[\alpha]^{-1}$.

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 §5 and §9.

4. A NUMERICAL EXAMPLE AND ITS SOLUTION IN IDL

If the following sounds like Greek to you, take a look at §2 and 3.

4.1. Generation of the numerical example

Suppose that we make four measurements of the angle y and we want to fit to a parabolic function in time t. In the notation of equation 1.1, 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.

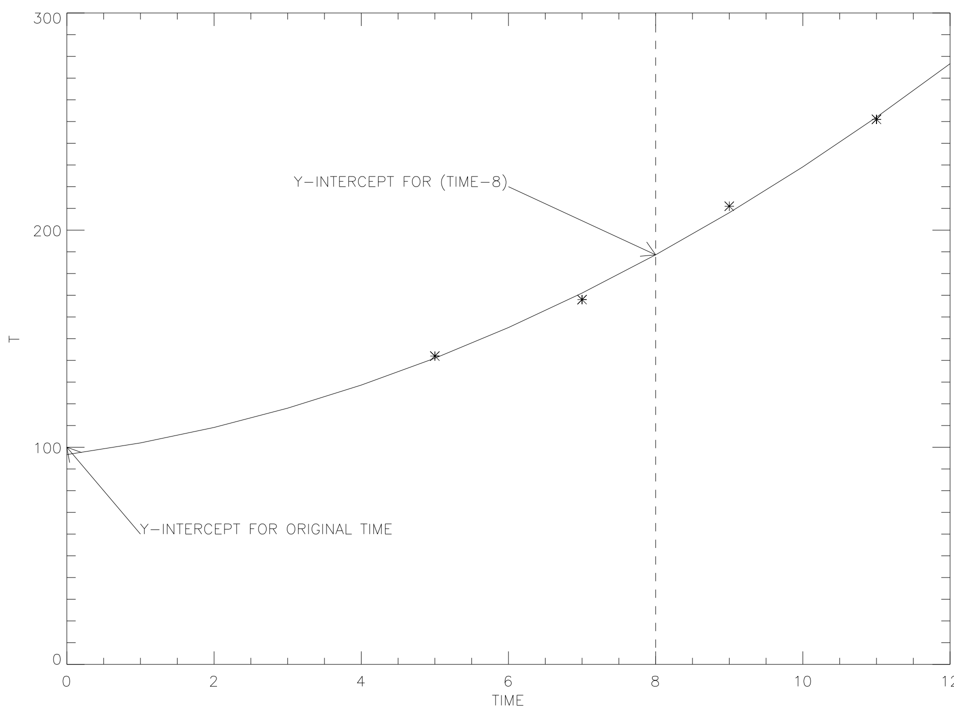

Fig. 4.1.— Our numerical example. Stars are the four datapoints; the solid line is the fit. We perform two fits: one uses the original definition of time; the other uses $(time - 8)$, in effect moving the y-axis to the dashed line. The two fits give the same line but the coefficients and their errors differ greatly.

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

$$\mathbf{X} = \mathbf{fltarr(N, M)} = \mathbf{fltarr(3, 4)} \tag{4.1}$$

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 2.1a we have $s_m = 1$, $t_m = time_m$, $u_m = time_m^2$:

$$\mathbf{X} = \begin{bmatrix} 1 & 5 & 25 \\ 1 & 7 & 49 \\ 1 & 9 & 81 \\ 1 & 11 & 121 \end{bmatrix} . \tag{4.2a}$$

Suppose that the four measured values of y are (equation 2.1c)

$$\mathbf{Y} = \begin{bmatrix} 142 \\ 168 \\ 211 \\ 251 \end{bmatrix} . \tag{4.3a}$$

Figure 4.1 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. $\mathbf{Y} = \mathbf{transpose}(\mathbf{Y})$.

4.2. Solution of the Numerical Example in IDL

In IDL we calculate the normal equation matrices and denote the $[\alpha]$ in equation 2.4a by $\mathbf{XX}$:

$$\mathbf{XX} = \mathbf{transpose}(\mathbf{X})\#\#\mathbf{X} \ , \tag{4.4a}$$

and we denote the $[\beta]$ in equation 2.4b by $\mathbf{XY}$:

$$\mathbf{XY} = \mathbf{transpose}(\mathbf{X})\#\#\mathbf{Y} \ . \tag{4.4b}$$

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

$$\mathbf{XXI} = \mathbf{invert}(\mathbf{XX}) \ . \tag{4.5}$$

The least-squares fitted quantities are in the matrix $\mathbf{a}$ (equation 2.5), which we obtain in IDL with

$$\mathbf{a} = \mathbf{XXI} \ \#\# \ \mathbf{XY} \ . \tag{4.6}$$

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

$$\mathbf{YBAR} = \mathbf{X} \ \#\# \ \mathbf{a} \ , \tag{4.7}$$

and we can also define the residuals in **Y** as

$$\mathbf{DELY} = \mathbf{Y} - \mathbf{YBAR} \ . \tag{4.8}$$

In IDL we denote s^2 in equations 3.1 and 3.6 by s_sq and write

$$s\_sq = \mathbf{transpose}(\mathbf{DELY})\#\#\mathbf{DELY}/(M - N) \ , \tag{4.9a}$$

or

$$s\_sq = total(\mathbf{DELY} \wedge 2)/(M - N) \ . \tag{4.9b}$$

It is *always* a good idea to plot all three quantities (the measured values **Y**, the fitted values **YBAR**, and the residuals **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

$$diag\ elements\ of\ \mathbf{XXI} = \mathbf{XXI}[(\mathbf{N+1}) * \mathbf{indgen}(\mathbf{N})] \ . \tag{4.10}$$

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 3.7 from[1]

$$\mathbf{vardc} = s\_sq * \mathbf{XXI}[(\mathbf{N+1}) * \mathbf{indgen}(\mathbf{N})] \ . \tag{4.11}$$

[1]If you used equation 4.9a instead of 4.9b, then IDL considers s_sq an array and you need to use a # instead of a $*$ in this equation.

4.3. Discussion of the numerical example

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

$$\mathbf{a} = \begin{bmatrix} 96.6250 \\ 4.5000 \\ 0.8750 \end{bmatrix} \tag{4.12a}$$

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 4.11] are:

$$\sigma_{\mathbf{A}} = \begin{bmatrix} 34.012 \\ 9.000 \\ 0.5590 \end{bmatrix} . \tag{4.12b}$$

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!

5. THE COVARIANCE MATRIX AND ITS NORMALIZED COUNTERPART

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

5.1. Definition of the normalized covariance (or correlation) 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, 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 to 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

$$ncov_{ik} = \frac{XXI_{ik}}{\sqrt{XXI_{ii}\ XXI_{kk}}} \ . \tag{5.1}$$

This is the same normalization that one does with the Pearson linear correlation coefficient of two variables. In fact, the elements of the normalized covariance matrix *are* the correlation coefficients. So it makes sense to call this matrix the *correlation matrix*, and many people do. In IDL, you do the following:

$$\mathbf{dc} = \mathbf{XXI}[(\mathbf{N}+\mathbf{1}) * \mathbf{indgen}(\mathbf{N})] \tag{5.2a}$$

$$\mathbf{ncov} = \mathbf{XXI}/sqrt(\mathbf{dc\#\#dc}) \ . \tag{5.2b}$$

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

Because **ncov** is a *normalized* covariance matrix, you might think that its non-normalized parent is **XXI**—and you'd be *almost* right. For the least-squares case we are discussing, the true covariance matrix **C** is[3]

$$\mathbf{C} = s^2\ \mathbf{XXI} \ . \tag{5.3}$$

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 happen to produce a coefficient that differs from its true value by some positive number. If the normalized matrix element is negative, then the other coefficient will tend to differ from its true value by a negative number.

Here's a more detailed discussion of what the covariance means. Suppose you are least-squares fitting for two derived coefficients (A_0 and A_1). When you do a least-squares fit to a set of data, you are fitting one set of data out of a possible infinity of possible sets that you'd get by repeating the experiment, and your particular set of data happens to produce specific values of $\overline{A_0}$ 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,

---

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

[3]For chi-square, you use σ^2_{meas} instead of s^2; see §8.

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 in §9, 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 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.

5.2. The covariance in our numerical example

Apply equation 5.2 to obtain the covariance matrix for our numerical example:

$$\mathbf{ncov} = \begin{bmatrix} 1 & -.989848 & .969717 \\ -.989848 & 1 & -.993808 \\ .969717 & -.993808 & 1 \end{bmatrix} . \tag{5.4}$$

The off-diagonal elements are *huge*. This is the reason why our derived coefficients have such large uncertainties. Note, however, that the fitted predicted fit is a good fit even with these large uncertaintis.

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 4.1 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

$$\mathbf{A} = \begin{bmatrix} 188.625 \\ 18.500 \\ 0.87500 \end{bmatrix} \tag{5.5a}$$

$$\sigma_{\mathbf{A}} = \begin{bmatrix} 3.58 \\ 1.00 \\ 0.559 \end{bmatrix} . \tag{5.5b}$$

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

$$\mathbf{ncov} = \begin{bmatrix} 1 & 0 & -0.78086881 \\ 0 & 1 & 0 \\ -0.78086881 & 0 & 1 \end{bmatrix} , \tag{5.6}$$

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-squares fitting. The general case is a royal pain, analytically speaking, so much so that we won't even carry it through for our example. But for numerical work you accomplish the orthogonalization using Singular Value Decomposition (SVD), which is of course trivial in IDL (§11).

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 §9. Also, you might also want to try out another example in Taylor's §8.5.

6. 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

$$p_{(\delta y;\sigma)} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{\delta y^2}{2\sigma^2}\right)} \ , \tag{6.1}$$

where σ^2 is the true variance of the datapoints, i.e. s^2 in equation 3.1 (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

$$P_{(|\delta y|<\Delta y)} = \int_{-\Delta y}^{+\Delta y} p_{(\delta y;\sigma)} d(\delta y) = \text{erf}\left(\frac{\Delta y}{\sqrt{2}\sigma}\right) \ , \tag{6.2}$$

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

$$P_{(|\delta y|<\sigma)} = 0.683 \ . \tag{6.3}$$

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

$$M_{(outside\ \Delta y)} = M\left[1 - \text{erf}\left(\frac{\Delta y}{\sqrt{2}\sigma}\right)\right] \ . \tag{6.4}$$

Chauvenet's criterion simply says:

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

$$\frac{\Delta y}{\sigma} = \sqrt{2}\ \text{erf}^{-1}\left(1 - \frac{1}{2M}\right) \ . \tag{6.5}$$

   This criterion leads to the numbers in the associated table, which is a moderately interesting set of numbers. Many astronomers adopt 3σ, which is clearly inappropriate for large N!

2. Discard all datapoints outside this range.

We offer the following important *Comments*:

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

- 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! (Like, maybe you should try the median fitting discussed in §13.)

- 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 non-Gaussian 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 real-life astronomers, 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 as Newton's method (NR §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 NR's chapter 9 about root finding, because some day you'll need it.

7. NONLINEAR LEAST SQUARES

The least-squares formulation requires that the data values depend *linearly* on the unknown coefficients. For example, in equation 0.1, the unknown coefficients A and B enter linearly.

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

$$\sin(At_m) + Bt_m = y_m \ . \tag{7.1}$$

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

First, assume 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 $y_{0,m}$:

$$\sin(A_0 t_m) + B_0 t_m = y_{0,m} \ . \tag{7.2}$$

Subtract equation 7.2 from 7.1, and express the differences as derivatives. Letting $\delta A = A - A_0$ and $\delta B = B - B_0$, this gives

$$\delta A[t_m \cos(A_0 t_m)] + \delta B t_m = y_m - y_{0,m} \ . \tag{7.3}$$

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 its 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 7.3, 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 NR's section 5.7 on evaluating numerical derivatives.

(6) Canned nonlinear least squares (particularly Levenberg-Marquardt, and var-

ious Gaussian fit routines): Packages like IDL offer canned nonlinear least squares routines. They are designed to work well for a wide range of different problems. However, for the specific problem at hand you can often do better by tailoring things (such as the factor $\mathcal{F}$ and convergence criteria above). A good example is Gaussian fitting: IDL's fitting program doesn't converge for multiple overlapping Gaussians, while for many of these cases the program that I wrote myself works fine; and converseley, my program doesn't work well for single Gaussians with a small number of datapoints, in which case IDL's `GAUSSFIT` is much better..

When convergence is slow or doesn't occur because your functions are complicated, you might wish to try the Levenberg-Marquardt method (NR §15.5); IDL function **LMFIT**. This technique involves increasing the diagonal elements of the curvature matrix by a set of suitably chosen factors; when you get close to the minimum, it resets these factors to unity. LM is the gold standard for nonlinear least-squares fitting because it is supposed to converge faster than other methods. Because of its sterling reputation, many people think it's the panacea. How many times have I seen journal articles saying that the LM method was used—as if that's all one needs to know—but without saying anything about the important stuff, such as how parameter space was explored to determine uniqueness of the solution! See the discussion in NR. I've done lots of nonlinear fits and have never had to resort to any tactic other than the simple, straightforward linearization process discussed above.

(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 NR'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 §8 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 non-Gaussian.

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

8. 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 3.1, which we repeat here:

$$s^2 = \frac{1}{M-N} \sum_{m=0}^{M-1} \delta y_m^2 \ .$$

where the m^{th} residual $\delta y_m = (y_m - \overline{y_m})$. 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

$$\chi^2 = \sum_{m=0}^{M-1} \frac{\delta y_m^2}{\sigma_{meas,m}^2} \ . \tag{8.1a}$$

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

$$\widehat{\chi^2} = \frac{1}{M-N} \sum_{m=0}^{M-1} \frac{\delta y_m^2}{\sigma_{meas,m}^2} \ , \tag{8.1b}$$

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* (s^2) to the *adopted intrinsic measurement variances* ($\sigma_{meas,m}^2$). So it should be obvious that in chi-square fitting, you must know the measurement uncertainties $\sigma_{meas,m}$ of the individual datapoints beforehand. If you want to give the various datapoints weights based on something other than $\sigma_{meas,m}$, then that is just like chi-square fitting except that you can adopt an arbitrary scale factor for the uncertainties (section 8.5).

Chi-square fitting treats uncertainties of the derived parameters in a surprising way. Getting the coefficient uncertainties with chi-square 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 §9.

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.

8.1. The weighted mean: the simplest chi-square 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:

$$mean = \frac{\sum y_m}{M} \tag{8.2a}$$

$$s^2 = \frac{\sum \delta y_m^2}{M-1} \tag{8.2b}$$

$$s_{mean}^2 = \frac{s^2}{M} = \frac{\sum \delta y_m^2}{M(M-1)} \ , \tag{8.2c}$$

where 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 $w_{meas,m} = \frac{1}{\sigma_{meas,m}^2}$. Applying maximum likelihood, in an unweighted average the quantity that is minimized is $\sum \delta y_m^2$; in a weighted average the quantity minimized is $\chi^2 = \sum \frac{\delta y_m^2}{\sigma_{meas,m}^2} = \sum w_{meas,m} \delta y_m^2 \rightarrow w_{meas,m} \sum \delta y_m^2$, where to the right of the arrow we assume all $w_{meas,m}$ are identical. So your intuition says that the three equations corresponding to the above would become

$$mean_{w,intuit} = \frac{\sum w_{meas,m} y_m}{\sum w_{meas,m}} \rightarrow \frac{\sum y_m}{M} \tag{8.3a}$$

Again, to the right of the arrow we assume all $w_{meas,m}$ are identical and the subscript *intuit* means "intuitive". For the variances the intuitive expressions are

$$s_{w,intuit}^2 = \frac{M}{M-1} \frac{\sum w_{meas,m} \delta y_m^2}{\sum w_{meas,m}} = \frac{\widehat{\chi^2}}{(\sum w_{meas,m}/M)} \rightarrow \frac{\sum \delta y_m^2}{M-1} = s^2 \tag{8.3b}$$

$$s^2_{w,mean,intuit} = \frac{s^2_{w,intuit}}{M} = \frac{\sum w_{meas,m}\delta y_m^2}{(M-1)\sum w_{meas,m}} = \frac{\widehat{\chi^2}}{\sum w_{meas,m}} \rightarrow \frac{\sum \delta y_m^2}{M(M-1)} = \frac{s^2}{M} \ . \qquad (8.3c)$$

In fact, after a formal derivation, the first two equations (8.3a and 8.3b) are correct, so we will drop the additional subscipts *intuit* and *formal* on *mean* and s_w^2. However, after a formal derivation, the last of these equations becomes, and is always written (e.g. BR equation 4.19; Taylor equation 7.12)

$$s^2_{w,mean,formal} = \frac{1}{\sum w_{meas,m}} \rightarrow \frac{\sigma^2_{meas}}{M} \ . \qquad (8.4)$$

This is a problem, for the following reason.

Note the excruciatingly painful difference between the intuitive equation 8.3c and the formally correct equation 8.4: on the right-hand side of the arrows, the intuitive one depends on $\frac{s^2}{M}$ (the variance of the *datapoint residuals*) , as you'd think it should, while the formal one depends on $\frac{\sigma^2_{meas}}{M}$ (the *adopted* intrinsic measurement variances of the data), which are 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 values for σ_{meas} that happen to be wrong, the two fits give different results for $s^2_{w,mean}$. 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 8.4, the *far superior*

$$s^2_{w,mean,BR} = \frac{\widehat{\chi^2}}{\sum w_{meas,m}} \rightarrow \frac{s_w^2}{M} \ , \qquad (8.5)$$

which is precisely the same as our intuitive guess, equation 8.3c. The difference between the formal equation 8.5 and the intuitive equations 8.3b and 8.4 is the numerator, which contains the reduced chi-square $\widehat{\chi^2}$; for the case where all $\sigma_{meas,m}$ are identical, $\widehat{\chi^2} = \frac{s_w^2}{\sigma^2_{meas}}$. Note that χ^2 and $\widehat{\chi^2}$ are defined in equations 8.1.

8.2. The multivariate chi-square fit

Here we generalize §8.1, which dealt with the weighted average, to the multivariate case. In this case, chi-square fitting is just like least-squares fitting except for the following:

1. In the least-squares matrix $\mathbf{X}$ of equation 2.1a, each row m is a different measurement with a different intrinsic variance σ_m. For chi-square fitting you generate a new matrix $\mathbf{X}_\chi$, which

is identical to $\mathbf{X}$ except that each row m (which contains a particular equation of condition) is divided by σ_m. This new matrix is the same as NR's *design matrix* (Figure 15.4.1), which they denote by $\mathbf{A}$.

2. For chi-square fitting, divide each datapoint y_m in equation 2.1b by σ_m. You are generating a new data vector $\mathbf{Y}_\chi$, which is identical to $\mathbf{Y}$ except that each datapoint is divided by σ_m. This new data vector is the same as NR's vector $\mathbf{b}$.

3. *Note* that the above two steps can be accomplished matrixwise by defining the $M \times M$ diagonal matrix $[\sigma]$ in which the diagonal elements are σ_m.

$$[\sigma] = \begin{bmatrix} \sigma_0 & 0 & \dots & 0 \\ 0 & \sigma_1 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \sigma_{M-1} \end{bmatrix} \tag{8.6}$$

in which case we can write

$$\mathbf{X}_\chi = [\sigma]^{-1} \cdot \mathbf{X} \tag{8.7a}$$

$$\mathbf{Y}_\chi = [\sigma]^{-1} \cdot \mathbf{Y} \ . \tag{8.7b}$$

4. Carry through the matrix calculations in equations 8.8 below (using the matrices subscripted with χ).

You've divided each row, i.e. the equation of condition for each row m, by a common factor, so the solution of that particular equation of condition is unchanged. However, in the grand scheme of things—i.e. the normal equations—it receives a greater or lesser weight by a factor $\frac{1}{\sigma_m^2}$.

To perform the chi-square fit, we first generate the weighted versions of $\mathbf{X}$ and $\mathbf{Y}$

$$\mathbf{X}_\chi = [\sigma]^{-1} \cdot \mathbf{X} \tag{8.8a}$$

$$\mathbf{Y}_\chi = [\sigma]^{-1} \cdot \mathbf{Y} \ . \tag{8.8b}$$

Then the equations of condition are

$$\mathbf{X}_\chi \cdot \mathbf{a} = \mathbf{Y}_\chi \tag{8.8c}$$

and the rest of the solution follows as with an unweighted fit, as before

$$[\alpha_\chi] = \mathbf{X_\chi^T} \cdot \mathbf{X_\chi} \tag{8.8d}$$

$$[\beta_\chi] = \mathbf{X_\chi^T} \cdot \mathbf{Y_\chi} \tag{8.8e}$$

$$\mathbf{a} = [\alpha_\chi]^{-1} \cdot [\beta_\chi] \ . \tag{8.8f}$$

Having calculated the derived coefficients $\mathbf{a}$, we can calculate the residuals. In doing so we must recall that $\mathbf{X_\chi}$ and $\mathbf{Y_\chi}$ contain factors of $\frac{1}{\sigma_m}$ and $[\alpha_\chi]^{-1}$ contains factors of σ_m^2. With all this, we can write the chi-square fit predicted data values as

$$\overline{\mathbf{Y_\chi}} = \mathbf{X_\chi} \cdot \mathbf{a} \tag{8.8g}$$

and the chi-square residuals as

$$\delta\mathbf{Y_\chi} = \mathbf{Y_\chi} - \overline{\mathbf{Y_\chi}} \tag{8.8h}$$

Because the data vector $\mathbf{Y_\chi}$ contains factors of $\frac{1}{\sigma_m}$, so do the residuals $\delta\mathbf{Y_\chi}$. You should, of course, always look at the residuals from the fit, so *remember these scale factors affect the residual values!* For example, if all σ_m are identical and equal to σ, then $\mathbf{Y_\chi} = \frac{\mathbf{Y}}{\sigma}$. If they don't, then when you plot the residuals $\delta\mathbf{Y_\chi}$ *each one will have a different scale factor!*

Moving on, we have

$$\chi^2 = \delta\mathbf{Y_\chi^T} \cdot \delta\mathbf{Y_\chi} \tag{8.8i}$$

$$\widehat{\chi^2} = \frac{\delta\mathbf{Y_\chi^T} \cdot \delta\mathbf{Y_\chi}}{M - N} \ . \tag{8.8j}$$

Finally, we have the analogy of equations 8.3c and 8.5 expressed in matrix form as in equation 3.7:

$$\mathbf{s_{a,intuit}}^2 = \widehat{\chi^2} \ diag\{[\alpha_\chi]^{-1}\} \ . \tag{8.9}$$

This *intuitively-derived* result is in contrast to the result derived from a *formal derivation*, which is the analogy to equation 8.4; again, it omits the $\widehat{\chi^2}$ factor:

$$\mathbf{s_{a,formal}}^2 = diag\{[\alpha_\chi]^{-1}\} \; . \tag{8.10}$$

This formally-derived result is what's quoted in textbooks (e.g. NR equation 15.4.15, BR equation 7.25). It provides parameter errors that are independent of the datapoint residuals, and leads to the same difficulties discussed above for the weighted mean case.

8.3. Which equation—8.9 or 8.10?

In most cases—but not all—we recommend that you use equation 8.9. Equation 8.9 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 "$\approx$" instead of "$=$" because different experiments produce somewhat different values of s^2 because of statistical fluctuations; an average over zillions of experiments gives $\sigma^2 = \langle s^2 \rangle$.) In this situation, though, equations 8.9 and 8.10 are identical. However, if the least-squares fit model is *not correct*, meaning that it doesn't apply to the data, then the residuals will be larger than the intrinsic measurement errors, which will lead to larger values of χ^2 and $\widehat{\chi^2}$—which is the indicator of a poor fit.

However, equation 8.9 is not a panacea. The numerical value of $\widehat{\chi^2}$ is subject to statistical variation. If the number of datapoints M is small (or, more properly, if the number of degrees of freedom $(M - N)$ is small), then the fractional statistical variation in $\widehat{\chi^2}$ is large and this affects the normalization inherent in equation 8.9. Alternatively, if you *really do know* the experimental errors equation 8.10 is appropriate.

Use your head!

8.4. Datapoints with known *relative* but unknown *absolute* dispersions

Here the σ_m are all different. The m^{th} row of the equation-of-condition matrix $\mathbf{X}$ and the m^{th} element of the data vector $\mathbf{Y}$ get divided by their corresponding σ_m. The equation embodied in each row of the matrix equation 2.2 remains unchanged, but the different rows are weighted differently with respect to each other.

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

Suppose that the above two measurements were taken under identical conditions except that measurement 1 received more integration time than measurement 2; we have $\frac{\sigma_1}{\sigma_2} = \left(\frac{\tau_1}{\tau_2}\right)^{-1/2}$, so the rows of $\mathbf{X}_\chi$ are weighted as $\tau^{1/2}$. This means that during the computation of $[\alpha_\chi] = \mathbf{X}_\chi^{\mathbf{T}} \cdot \mathbf{X}_\chi$, 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 $\left(\frac{1}{\sigma_m}\right)^2$. We conclude that the weighting scheme of the first two steps in section 8.2 agrees with common sense.

Suppose you don't know the intrinsic measurement dispersion σ_m, 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 $\sigma_m \propto \tau^{-1/2}$. In this case, multiply each row by its weight $w \propto \frac{1}{\sigma_m}$ and proceed as above. (The factors $\frac{1}{\sigma_m}$ in the equations of condition become $\frac{1}{\sigma_m^2}$ in the normal equations.)

8.5. Persnickety Diatribe on Choosing σ_m

8.5.1. Choosing and correcting σ_m

In the previous section, equation 8.10 taught us that—formally, at least—the variances in the derived fit parameters (or their uncertainties, which are the square roots) depend only on the adopted uncertainties σ_m and not on the *actual variance* of the *datapoints*.

Are you bothered by the fact that the variances of the derived parameters $\mathbf{s_a}$ are independent of the data residuals? You should be: it is obvious that the residuals should affect $\mathbf{s_a}$.

Formally, $\mathbf{s_a}$ depends only on the *adopted* uncertainties σ_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 σ_m are off by a common scale factor, i.e. if $\sigma_{m,adopted} = f\sigma_{m,true}$. 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 σ_m by f. Of course, you don't know what this factor f is until you do the chi square fit. Dividing them by f is equivalent to multiplying them by $\widehat{\chi}$. And, of course, the same as multiplying σ_m^2 by $\widehat{\chi^2}$.

8.5.2. When you're using equation 8.9...

To be kosher, after having run through the problem once with the adopted σ_m, calculate the $\widehat{\chi^2}$; multiply all σ_m by $\widehat{\chi}$; and redo the problem so that the new $\widehat{\chi^2} = 1$. Then the derived variance

$\mathbf{s_a}$ is also correct. You can obtain it either as the corresponding diagonal to the covariance matrix (equations 8.9 and 8.10, which are identical in this case) or by finding what departure from x_0 is necessary to make $\Delta\chi^2 = 1$.[4] This redoing the fit may seem like unnecessary work, but when we deal with multiparameter error estimation in §9 it's the best way to go to keep yourself from getting confused.

8.5.3. Think about your results!

In the case $\Delta\chi^2 \approx 1$ (and $\widehat{\chi^2} \approx 1$) the dispersions of the observed points s_m are equal to the intrinsic dispersions of the datapoints σ_m 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 values for σ_m: you are ascribing more error to your datapoints than they really have, perhaps by not putting enough faith in your instrument.

But there is *another way* you can get artificially small values for $\widehat{\chi^2}$. This will occur 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. Cowan discusses this possibility in his §7.6.

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 8.10 instead of 8.9 is disastrous.

Think about your results.

8.5.4. When your measurements are correlated...

One more point, a rather subtle one. There are circumstances in which your datapoints are not independent. Then the formulation of chi-square fitting (and least-squares fitting, for that matter) is more complicated. You need to calculate the covariance matrix for the measured values y_m; call this covariance matrix $\mathbf{V}$. If this matrix is not unitary, then χ^2 is no longer given by equation 8.8i. Rather, it is given by

$$\chi^2 = \delta\mathbf{Y}_\chi^{\mathbf{T}} \cdot \mathbf{V}^{-\mathbf{1}} \cdot \delta\mathbf{Y}_\chi \ . \tag{8.11a}$$

Of course, this leads to a different expression for $\mathbf{a}$, which replaces equation 8.8f,

[4]To understand this comment about $\Delta\chi^2 = 1$, see §9.

$$\mathbf{a} = (\mathbf{X_\chi^T} \cdot \mathbf{V^{-1}} \cdot \mathbf{X_\chi})^{-1} \cdot \mathbf{X_\chi^T} \cdot \mathbf{V^{-1}} \cdot \mathbf{Y}_\chi \ , \tag{8.11b}$$

and also to a different equation for the covariance matrix,

$$[\alpha_\chi]^{-1} = (\mathbf{X_\chi^T} \cdot \mathbf{V^{-1}} \cdot \mathbf{X_\chi})^{-1} \ . \tag{8.11c}$$

Correlated datapoints can occur when the measured y_m are affected by systematic errors or instrumental effects. Cowan §7.6 discusses this case. For example, suppose you take an image with a known point-spread function (psf) and want to fit an analytic function to this image. Example: a background intensity that changes linearly across the field plus a star. Here the independent variables in the function are the (x, y) pixel positions and the data are the intensities in each pixel. You'd take the intensity in each individual pixel and fit the assumed model. But here your data values are correlated because of the psf. Because you know the psf, you know the correlation between the various pixels. Such a formulation is required for CBR measurements because of the sidelobes of the radio telescope (which is just another way of saying "psf").

Another case of correlated measurements occurs when your assumed model is incorrect. This is the very definition of correlation, because the residual δy_m is correlated with the data value y_m. But how do you calculate $\mathbf{V}$? If you could do a large number J of experiments, each with M datapoints producing measured values $y_{m,j}$, each measured at different values of x_m, then each element of the covariance matrix would be $V_{mn} = \sum_j (y_{m,j} - \overline{y_m})(y_{n,j} - \overline{y_n})$. You don't normally have this opportunity. Much better is to look at your residuals; if the model doesn't fit, use another one!

Normally, and in particular we assume everywhere in this tutorial, the measurements are uncorrelated, so one takes $\mathbf{V} = \mathbf{I}$ (the unitary matrix).

9. CHI-SQUARE FITTING AND WEIGHTED FITTING: DISCUSSION INCLUDING COVARIANCE

9.1. Phenomenological description

Consider the first two coefficients in our example of §5.2. 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 4.12. 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 -5 and 13.

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 3.7, and 8.9 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.

We illustrate these concepts for the (a_1, a_2) parameters in 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 §9.5 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 3.7, 4.11, 8.9, and 8.10. If the datapoints were projected downward, i.e. look at the marginal pdf of δa_1 by taking small strips of δa_1 and integrating over δa_2, the marginal 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

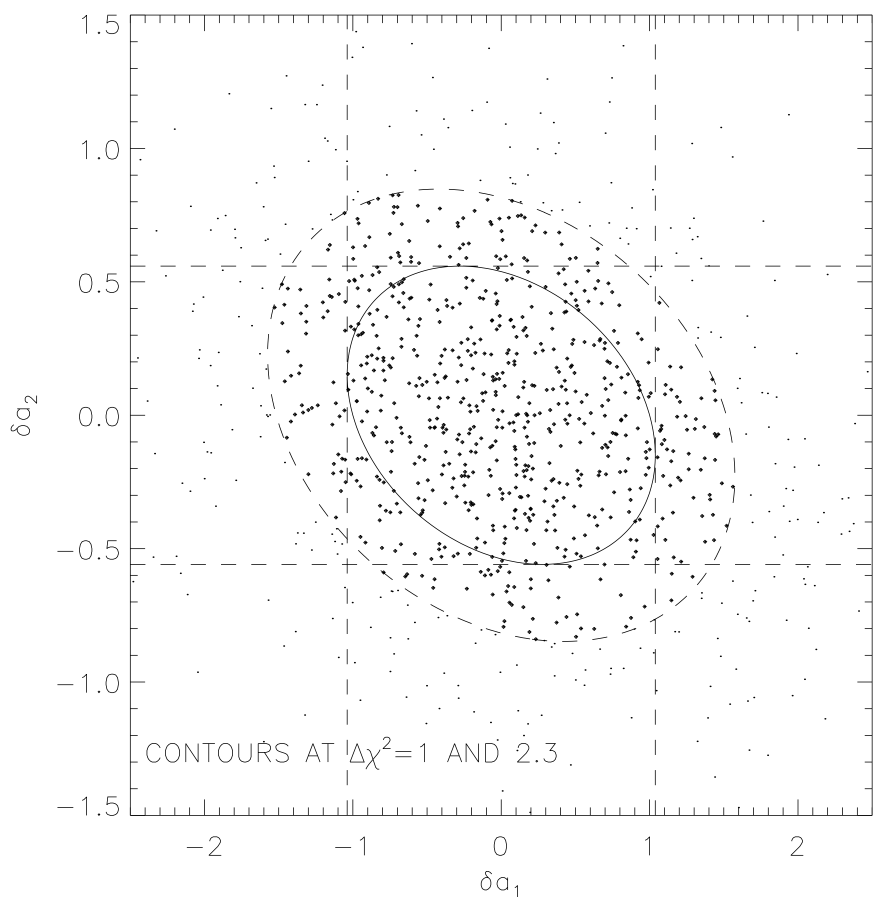

Fig. 9.1.— Illustrating the meaning of variance and covariance between $(a1, a2)$ for our numerical example. See text for discussion.

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-square 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 9.1. 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 a single parameter; then we'll generalize.

9.2. Calculating the uncertainties of a single parameter—gedankenexperiment

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 nonzero value of δa_0 and redo the least-squares solution for $[a_1, a_2]$; because of the covariance, these adopt values different from those when $\delta a_0 = 0$. 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 $\sigma^2_{a_0} = [\alpha_\chi]^{-1}_{00}$, the same result quoted in equation 8.10.

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-square 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.)

9.3. Calculating the uncertainties of two parameters—gedankenexperiment

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 $[\alpha_\chi]$, 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-square 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-square 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...

9.4. Calculating the uncertainties of three parameters—gedankenexperiment

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-square 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 $\sigma_{\mathbf{a}}$.

9.5. Doing these calculations the non-gedanken easy way

The obvious way to do the gedanken calculations described above is to set up a grid of values in the parameters of interest (δa_n); perform the chi-square 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[5]. The curvature matrix $[\alpha_\chi]$ of equation 8.8d 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

$$\Delta\chi^2_{\mathbf{a}} = \delta\mathbf{a}^{\mathbf{T}} \cdot [\alpha_\chi] \cdot \delta\mathbf{a} \ . \tag{9.1}$$

Knowing the curvature matrix, we don't have to redo the fits as we described above. Rather, we can use the already-known matrix elements.

[5]In which case you use the gedanken technique!

Suppose, however, that you are interested in an N_i (for "$N_{interested}$") subset of the N parameters, and you want to know their variances (and covariance matrix) *without regard to the values of the other* $(N - N_i)$ *parameters.* You could use the gedanken technique, but you can also use the "non-gedanken easy way" by using the following procedure [see NR §15.6 (Probability Distribution of Parameters in the Normal Case)].

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

$$[\alpha_\chi]^{-1} = \mathbf{XXI} = \begin{bmatrix} 1156.8125 & -303.000 & 18.4375 \\ -303.000 & 81.000 & -5.000 \\ 18.4375 & -5.000 & 0.31250 \end{bmatrix} \tag{9.2a}$$

and it becomes

$$[\alpha_\chi]_i^{-1} = \mathbf{XXI} = \begin{bmatrix} 1156.8125 & 18.4375 \\ 18.4375 & 0.31250 \end{bmatrix} . \tag{9.2b}$$

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

4. As usual, we have

$$\Delta\chi^2_{\mathbf{a_i}} = \delta\mathbf{a_i^T} \cdot [\alpha_\chi]_\mathbf{i} \cdot \delta\mathbf{a_i} \ , \tag{9.3}$$

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 NR's discussion surrounding equation (15.6.2); this is not very clear, in my opinion. I find it easier to recall the way covariance matrices propagate errors according to the derivatives of the quantities derived, as in Cowan's equation 1.54. I could explain this here but, quite frankly, don't have the time; maybe in the next edition of these notes!

9.6. 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 §9.5 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 §9.2, 9.3, and 9.4. In other words, use the gedanken technique.

- The variance (i.e., uncertainty squared) of the derived parameters $\mathbf{a}$ depends *only on the elements in the covariance matrix* $[\alpha_\chi]^{-1}$. These, in turn, depend only on the curvature matrix $[\alpha_\chi]$—which depends only on $\mathbf{X}_\chi$. This matrix $\mathbf{X}_\chi$ is the matrix of the quantities that are *known exactly*. For example, we began with the example in which the elements of $\mathbf{X}_\chi$ 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 0.1) but not on the *measured values* (the ensemble of y_m in equation 0.1). 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 your efforts will be repaid in terms of much better accuracy for the y-intercept.

10. BRUTE FORCE CHI-SQUARE AND THE CURVATURE MATRIX

10.1. 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 χ^2 or s^2 for each combination of parameters, and find the minimum. This provides the best-fit parameter values.

How about the parameter uncertainties? Simple: use the concept of $\Delta\chi^2$ from §9. Here we describe the case for a single parameter fit; call this parameter a. Generalizing to more parameters is straightforward.

For a chi-square fit, getting the uncertainty is easy. Calculate χ^2 as a function of the guessed values of a. As usual, define $\Delta\chi^2$ as χ^2 minus its minimum value; the minimum value gives the best estimate of a. The uncertainty in a is that offset where $\Delta\chi^2 = 1$. In other words: $\Delta\chi^2 = [\alpha_{\chi,00}]\Delta a^2$, so the uncertainty in a is $\frac{1}{\sqrt{[\alpha_{\chi,00}]}} = \sqrt{[\alpha_{\chi,00}^{-1}]}$ (See §9).

For a least-squares fit, it's exactly the same idea. A least-squares fit implies that the measurement uncertainties σ_m are all identical, equal to σ. Thus, the sample variance $s^2 = \frac{1}{M-1}\sum \Delta y_m^2$ is equal to $\frac{\chi^2\sigma^2}{(M-1)}$. In other words, $\chi^2 = \frac{(M-1)}{\sigma^2}s^2$, which has expectation value $(M-1)$. Therefore, the uncertainty in a is that offset where $\Delta\chi^2 = 1$, i.e. where $\Delta s^2 = \frac{\sigma^2}{(M-1)}$.

To be totally explicit: For the fitted value of a_{fit}, the sample variance is

$$s_{min}^2 = \frac{1}{M-1}\sum(y_m - a_{fit})^2 \tag{10.1}$$

As a is moved from its fitted value, s^2 increases, so we can speak of the minimum sample variance s_{min}^2. As we move a from its fitted value by amounts Δa, the uncertainty in a is that value of Δa for which s^2 increases by $\frac{s_{min}^2}{M-1}$, i.e. that value of Δa for which

$$\Delta s^2 = s^2 - s_{min}^2 = \frac{s_{min}^2}{M-1} \tag{10.2}$$

11. USING SINGULAR VALUE DECOMPOSITION (SVD)

Occasionally, a normal-equation matrix $[\alpha] = \mathbf{X^T} \cdot \mathbf{X}$ is degenerate, or at least 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 $[\alpha^{-1}]\cdot[\alpha] = \mathbf{I}$, the unitary matrix). In these cases, Singular Value Decomposition (SVD) comes to the rescue.

First, we reiterate the least-squares problem. Least squares begins with equations of condition (equation 2.2), which are expressed in matrix form as

$$\mathbf{X} \cdot \mathbf{a} = \mathbf{y} \tag{11.1}$$

In our treatments above we premultiply both sides by $\mathbf{X^T}$, on the left generating the curvature matrix $[\alpha]$ to obtain the normal equations (equation 2.3 or its equivalent derived from equations 8.8)

$$[\alpha]\cdot\mathbf{a} = \mathbf{X^T} \cdot \mathbf{y} \ , \tag{11.2}$$

for which the solution is, of course,

$$\mathbf{a} = \left([\alpha]^{-1} \cdot \mathbf{X^T}\right) \cdot \mathbf{y} \tag{11.3}$$

We need to find the inverse matrix $[\alpha]^{-1}$. Above in this document we did this using simple matrix inversion, which doesn't work if $[\alpha]$ is degenerate.

SVD provides a bombproof and interestingly informative way do least squares. Perhaps surprising, with SVD you don't form normal equations. Rather, you solve for the coefficients directly. In essence, SVD provides the combination $\left([\alpha]^{-1} \cdot \mathbf{X^T}\right)$ without taking any inverses. By itself, this doesn't prevent blowup for degenerate cases; however, SVD provides a straightforward way to eliminate the blowup and get reasonable solutions.

For a discussion of the details of SVD, see NR §2.6; for SVD applied to least squares, see NR §15.4; if you are rusty on matrix algebra, look at NR §11.0. Below, we provide a brief description of SVD. Implementing SVD in our least-squares solutions is trivially easy, and we provide the IDL prescription below in §11.5 and §11.5.2. *Be sure* to look at §11.3!!

11.1. Phenomenological description of SVD

The cornerstone of SVD is that our (or any) $M \times N$ (M rows, N columns) matrix $\mathbf{X}$, where $M \geq N$, can be expressed as a product of three matrices:

$$\mathbf{X} = \mathbf{U} \cdot [w] \cdot \mathbf{V^T} \ , \tag{11.4}$$

where

1. $\mathbf{U}$ is $M \times N$, $[w]$ is $N \times N$ and diagonal, and $\mathbf{V}$ is $N \times N$; and
2. the columns of $\mathbf{U}$ and $\mathbf{V}$ are unit vectors that are orthonormal. Because $\mathbf{V}$ is square, its rows are also orthonormal so that $\mathbf{V} \cdot \mathbf{V^T} = \mathbf{I}$. Recall that, for square orthonormal vectors, the transpose equals the inverse so $\mathbf{V^T} = \mathbf{V^{-1}}$. Similarly, because $\mathbf{U}$ has columns that are orthonormal, the matrix product $\mathbf{U^T} \cdot \mathbf{U} = \mathbf{I}_{N \times N}$ (the $N \times N$ unitary matrix).
3. The columns of $\mathbf{V^T}$, which are orthonormal vectors, are the eigenvectors of $[\mathbf{X^T} \cdot \mathbf{X}]$. That is, in our case, they are the eigenvectors of the curvature matrix of $\mathbf{X}$. That is, they define the principal axes of the error ellipsoid $\Delta\chi^2$; see NR §14.5.
4. The eigenvalues of $[\mathbf{X^T} \cdot \mathbf{X}]$ are $[w^2]$, i.e. the squares of the diagonal elements in $[w]$.

So what?

Consider a single *row* (measurement number m) of the matrix $\mathbf{X}$. This row has a single value of x_m, with an associated value of y_m. For this row, we can consider the entry of each column n of $\mathbf{X}$ to be a basis function $f_n(x_m)$, evaluated at x_m. The N columns contain N basis functions. And, of course, we have N coefficients in the vector $\mathbf{a}$. These N basis functions and associated coefficients represent the entire set of M measurements y_m taken at positions x_m.

Now consider a single *column* (function and coefficient number n) of the matrix $\mathbf{X}$. This column consists of the entire set of M values of $f_n(x_m)$ for a single value of n. We can regard the M values of x_m to be a vector of length M. Similarly, we can regard each column n of $\mathbf{X}$ to be an M-length vector consisting of the elements $f_n(x_m)$.

Finally, consider the *set of N columns* of the matrix $\mathbf{X}$. Each column n is an M-element vector with elements $f_n(x_m)$. Now, there is no reason for these N column vectors of $\mathbf{X}$ to have any special property, such as being orthogonal. In fact, consider a typical least-squares polynomial fit; this basis set is certainly *not* orthogonal, as we illustrated in our simple numerical example of §4. What SVD does is to replace the set of N original nonorthogonal vectors $\mathbf{X}$ by an orthogonal basis set that consists of the N rows of $\mathbf{V}$—in fact, they are not only orthogonal, but orthonormal.

11.2. Using SVD for Least Squares

Some of the original nonorthogonal column vectors of $\mathbf{X}$ might not only be nonorthogonal, they might be degenerate! This happens if the particular set of the M values x_m make some of the original nonorthogonal vectors linear combinations of others. This leads to the unfortunate situation of the curvature matrix α not having an inverse. In such cases, SVD comes to the rescue.

With SVD you *don't form normal equations*, so you don't explicitly calculate $[\alpha]^{-1}$. We apply SVD to $\mathbf{X}$, as in equation 11.4:

$$\mathbf{X} = \mathbf{U}\cdot[w]\cdot\mathbf{V^T} \ , \tag{11.5}$$

If we express the covariance matrix in these SVD terms, we get

$$[\alpha^{-1}] = \mathbf{V}\cdot\left[\frac{1}{w^2}\right]\cdot\mathbf{V^T} \tag{11.6}$$

Here's the proof:

$$[\alpha] = \mathbf{X^T}\cdot\mathbf{X} \tag{11.7a}$$

Using equation 11.4 or 11.5 for the SVD representation of $\mathbf{X}$, we have

$$[\alpha] = [\mathbf{U}\cdot[w]\cdot\mathbf{V^T}]^{\mathbf{T}}\cdot[\mathbf{U}\cdot[w]\cdot\mathbf{V^T}] \tag{11.7b}$$

Remembering that $\mathbf{w}$ is diagonal so that $\mathbf{w^T} = \mathbf{w}$, that $\mathbf{U^T} \cdot \mathbf{U} = \mathbf{I}_{N\times N}$, and the matrix identity $[\mathbf{a} \cdot \mathbf{b}]^T = \mathbf{b^T} \cdot \mathbf{a^T}$, we have

$$[\alpha] = \mathbf{V} \cdot [w^2] \cdot \mathbf{V^T} \tag{11.7c}$$

Finally, take the inverse (using the matrix identity $[\mathbf{a} \cdot \mathbf{b}]^{-1} = \mathbf{b^{-1}} \cdot \mathbf{a^{-1}}$ and also $\mathbf{V^T} = \mathbf{V^{-1}}$); this gives the result of equation 11.6.

Given equation 11.6, the fact that $\mathbf{V^T} \cdot \mathbf{V} = \mathbf{I}$, and the general matrix identity $(\mathbf{a} \cdot \mathbf{b})^\mathbf{T} = \mathbf{b^T} \cdot \mathbf{a^T}$, it's straightforward to show that

$$\mathbf{a} = \left(\mathbf{V} \cdot \left[\frac{1}{w} \right] \cdot \mathbf{U^T} \right) \cdot \mathbf{y} \tag{11.8a}$$

or

$$\mathbf{a} = \left(\mathbf{V} \cdot \left[\frac{1}{w} \right] \right) \cdot \left(\mathbf{U^T} \cdot \mathbf{y} \right) \tag{11.8b}$$

Here in equation 11.8a, we see that $\left(\mathbf{V} \cdot \left[\frac{1}{w}\right] \cdot \mathbf{U^T}\right)$ is identical to $\left([\alpha]^{-1} \cdot \mathbf{X^T}\right)$ in equation 11.3. In the rewritten equation 11.8b we see that we can regard the coefficient vector being defined by the orthogonal vectors of $\mathbf{V}$. Moreover, the covariance matrix of equation 11.6 is diagonal, so that the principal axes of the χ^2 ellipse are defined by the orthonormal column vectors of $\mathbf{V}$ with lengths proportional to $\left[\frac{1}{w^2}\right]$. See NR Figure 15.6.5 and the associated discussion.

The N orthonormal vectors in $\mathbf{V}$ define an N-dimensional space for $\mathbf{a}$ with N orthogonal directions. Suppose that a particular value w_n is small. In equation 11.5, this means that the associated orthonormal vector $\mathbf{v}_n$—i.e., the associated direction n—in $\mathbf{V}$ is not well-represented by the set of original nonorthogonal vectors in $\mathbf{X}$. This, in turn, means that you can't represent that direction in $\mathbf{a}$ of equation 11.8a without amplifying that orthonormal vector by a large factor; these amplification factors $\propto w_n^{-1}$. This is an unsatisfactory situation because it means some combinations of the original m measurements are highly weighted. As $w_n \rightarrow 0$ this situation becomes not only unsatisfactory, but numerically impossible.

Consider the limiting case, $w_n = 0$. In this case, the original x_m values do not have any projection along associated orthonormal vector $\mathbf{v}_n$ in $\mathbf{V}$. (These $\mathbf{v}_n$ are called "null" orthonormal vectors.) In equation 11.8a, you can add any multiple of the null $\mathbf{v}_n$ to the solution for $\mathbf{a}$ and it won't change $\mathbf{a}$ at all (!) because it has absolutely no effect on the fit to the data (because of the particular set of values x_m).

What multiple is appropriate? Common sense says that, because these $\mathbf{V}$ vectors have no meaning for the solution, the multiple should be *zero*. So in equation 11.8a, instead of trying, in vain, to include this null vector $\mathbf{v}_n$ by using a huge multiple, you toss in the towel and eliminate it altogether by replacing its corresponding $w_n^{-1} = \infty$ by $w_n^{-1} = 0$. So we have the rule: *wherever* $w_n = 0$ *(or sufficiently small), replace* w_n^{-1} *by 0!* This replacement provides the minimum length for $\mathbf{x}$ and thereby constitutes the least-squares solution.

11.3. Important Conclusion for Least Squares!!!

Suppose you have degeneracy, or near-degeneracy. *This, in turn, means that the formulation of the least-squares model is faulty*: some of the original basis functions represent (or *nearly*) represent the same physical quantity, or at least one of the functions is (nearly) a linear combination of others. Sometimes you can discover the problem and fix it by imposing additional constraints, or by finding two unknown coefficients that are nearly identical. If so, you can reformulate the model and try again.

Even if you can't discover the root cause(s) and remove the degeneracy, the SVD solution allows you to bypass problems associated with degeneracy and provides reasonable best-fit parameter values.

11.4. How Small is "Small"?

When looking at w_n, just exactly how small is "small"?

11.4.1. Strictly Speaking...

Strictly speaking, this is governed by numerical machine accuracy. NR suggests measuring this by the ratio of the largest w_n to the smallest. This ratio is the *condition number*. Single precision 32-bit floats have accuracy $\sim 10^{-6}$, so if the condition number is larger than 10^6 you certainly have problems.

11.4.2. Practically Speaking...

Practically speaking it's not machine accuracy that counts. Rather, it's the accuracy of your data and the ability of those uncertain data to define the parameters. In any real problem, plot the vector w. If the values span a large range, then the data with small w_n might not be defining the associated vectors well enough. If so, then *practically* speaking, these vectors are degenerate.

How to tell what constitutes "small w_n"? As far as I know, and as far NR says, it's an art.

11.5. Doing SVD in IDL

11.5.1. *IDL's SVD routine* `la_svd`

IDL'S `la_svd` procedure[6] provides the SVD decomposition. Thus for equation 11.4, the IDL SVD decomposition is given by

$$\mathbf{la\_svd}, \mathbf{X}, \mathbf{w}, \mathbf{U}, \mathbf{V}\ ; \tag{11.9}$$

the notation is identical to that in equation 11.4.

11.5.2. *My routine lsfit_svd*

Implementing a least-squares SVD fit requires the ability to modify the weights. I've written an IDL routine `lsfit_svd` that makes the above process of dealing with the weights easy. When used without additional inputs, it returns the standard least-squares results such as the derived coefficients and the covariance matrix; it also returns $[w]$, $\mathbf{U}$, and $\mathbf{V}$. It allows you to input those matrices and, also, the $\left[\frac{1}{w}\right]$ matrix so that you can tailor the weights to your heart's content.

12. 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[7]. 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

---

[6]Old versions of IDL (before 5.6) had a significantly worse algorithm called *svdc*. Don't use this unless you have to.

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

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.

12.1. Stetson's sliding weight

Stetson recommends using a sliding weight. To explain this, we review the ML concept of chi-square fitting. We define chi-square as

$$\chi^2 = \sum_{m=0}^{M-1} \frac{(y_m - \mathbf{a} \cdot \mathbf{f}(\mathbf{x_m}))^2}{\sigma_m^2} \ , \tag{12.1a}$$

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

$$\frac{d\chi^2}{da_n} = -2 \sum_{m=0}^{M-1} \frac{f_n(x_m) \Delta y_m}{\sigma_m^2} \ . \tag{12.1b}$$

Here $\Delta y_m = (y_m - \mathbf{a} \cdot \mathbf{f}(\mathbf{x_m}))$. For each coefficient a_n setting this to zero gives

$$\sum_{m=0}^{M-1} \frac{f_n(x) \Delta y_m}{\sigma_m^2} = 0 \ . \tag{12.1c}$$

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:

$$\sum_{m=0}^{M-1} \frac{w_{(|\Delta y_m|)} f_n(x_m) \Delta y_m}{\sigma_m^2} = 0 \ . \tag{12.2}$$

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.

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

Stetson recommends

$$w_{(|\Delta y_m|)} = \frac{1}{1 + \left(\frac{|\Delta y|}{\alpha\sigma}\right)^{\beta}} \ . \tag{12.3}$$

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 $\beta \rightarrow \infty$ 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 6.5, this occurs by setting

$$\alpha = \sqrt{2}\ \mathrm{erf}^{-1}\left(1 - \frac{1}{2M}\right) \tag{12.4}$$

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 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 non-Gaussian problems such as cosmic ray hits, it seems to me only reasonable to increase α above Stetson's recommended values by using equation 12.4.

12.2. Implementation of the weight in our matrix equations

Clearly, implementing Stetson's method requires a weighted fit, so you have to use the chi-square technique discussed in §8. There equation 8.6 defines a matrix of weights (which is diagonal) in which

$$\mathbf{W}_{m,m} = \frac{1}{\sigma_m} \ . \tag{12.5}$$

Comparing this with equation 12.1c, it's clear what to do: we modify this equation to read

$$\mathbf{W}_{m,m} = \frac{w_m^{1/2}}{\sigma_m} \ , \tag{12.6}$$

where the weight w_m is defined in equation 12.3.

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.

13. MEDIAN/MARS, INSTEAD OF LEAST-SQUARES, FITTING

Least-squares fitting minimizes the squares 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 between outliers and good datapoints. Or you might have non-Gaussian statistics. In these cases, using least squares might not be 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 the median is often a good solution. The median is the solution for which there are as many positive as negative residuals, *irrespective of how big they are.* The median works especially well when the discrepant datapoints are asymmetrically distributed with respect to sign.

The median isn't always appropriate: for example, for pdfs that are symmetrically dominated by large residuals and have few small ones, datapoints near the expectation value are few and far between so the median has high error. Also, if the statistics are Gaussian 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).

If you have non-Gaussian statistics, then apply to your situation these somewhat contradictory introductory remarks both extolling and trashing the median. If you decide that a median fit is appropriate, read on!

13.1. The Median versus the MARS

Consider ARS, the sum of the absolute values of the residuals (the Absolute Residual Sum) and suppose we want to minimize this. This is the MARS: the *Minimum Absolute Residuals Sum.* For the standard median (that takes the "average" of a bunch of numbers), the MARS is identical to the median. This isn't true for more complicated functions, as we briefly discuss in the next two subsections.

13.1.1. For the Standard Median—it's the MARS

When we take the median of a bunch of points that's like taking a biased average of the points. In this case the residual $\Delta y_m = y_m - y_{MARS}$, where y_{MARS} is the derived median value. Normally we take the median by finding that datapoint for which the number of positive residuals is equal to the number of negative ones.

Here we take a more general approach. First, write the ARS as a function of the a trial value y_{ARS}; we will find that the median value is y_{MARS}, the one that gives the minimum of the ARS. So write

$$ARS(y_{ARS}) = \sum_{m=0}^{M-1} |(y_m - y_{ARS})| = \sum_{m=0}^{M-1} |\Delta y_m| \tag{13.1a}$$

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

$$ARS(y_{ARS}) = \sum_{\Delta y_m > 0} (y_m - y_{ARS}) - \sum_{\Delta y_m < 0} (y_m - y_{ARS}) \tag{13.1b}$$

To find the minimum of the ARS we take its the derivative with respect to y_{ARS} and set it equal to zero. The derivative of each term is equal to -1, so we get

$$\frac{dARS}{dy_{ARS}} = \sum_{\Delta y > 0} 1 - \sum_{\Delta y < 0} 1 = M(\Delta y > 0) - M(\Delta y < 0) \tag{13.2}$$

Setting this equal to zero requires choosing $M(\Delta y > 0) = M(\Delta y < 0)$. In plain English: y_{MARS} is that value of y_{ARS} which provides equal numbers of positive and negative residuals.

This is the very definition of the median! Thus, for the "average" of a bunch of numbers, the median is the same as the MARS.

The median is defined unambiguously only if M is odd: the median datapoint then has equal numbers with positive as negative residuals. If M is even, then the median lies anywhere between the two middle points. Similarly, the ARS doesn't change between the two middle points because as you increase the residual from one point the residual from the other decreases by an identical amount, so the sum of the absolute values doesn't change.

13.1.2. For an arbitrary function, e.g. the slope—it's a weighted MARS

Things change if we are fitting a function other than the "average" (i.e., the standard median). Suppose, for example, that we want to fit the slope s using MARS. Then $y_m = s_{MARS} x_m$ so the analog to equation 13.1a is

$$ARS(s_{ARS}) = \sum_{m=0}^{M-1} |(y_m - x_m s_{ARS})| = \sum_{m=0}^{M-1} |\Delta y_m| \tag{13.3}$$

Carrying out the same steps, the analog for equation 13.2 becomes

$$\frac{dARS}{ds_{ARS}} = \sum_{\Delta y>0} x_m - \sum_{\Delta y<0} x_m \tag{13.4}$$

This is no longer the median. Rather, it's a *weighted ARS*, or WARS. In calculating the ARS, each datapoint is weighted by its x value.

This weighting is exactly the same weighting that would occur in a standard least-squares fit for the slope. To show this, carry out the steps in equations 12.1 for a single function $f(x) = sx$. This makes sense because points with large x are intrinsically able to define the slope better than points with small x.

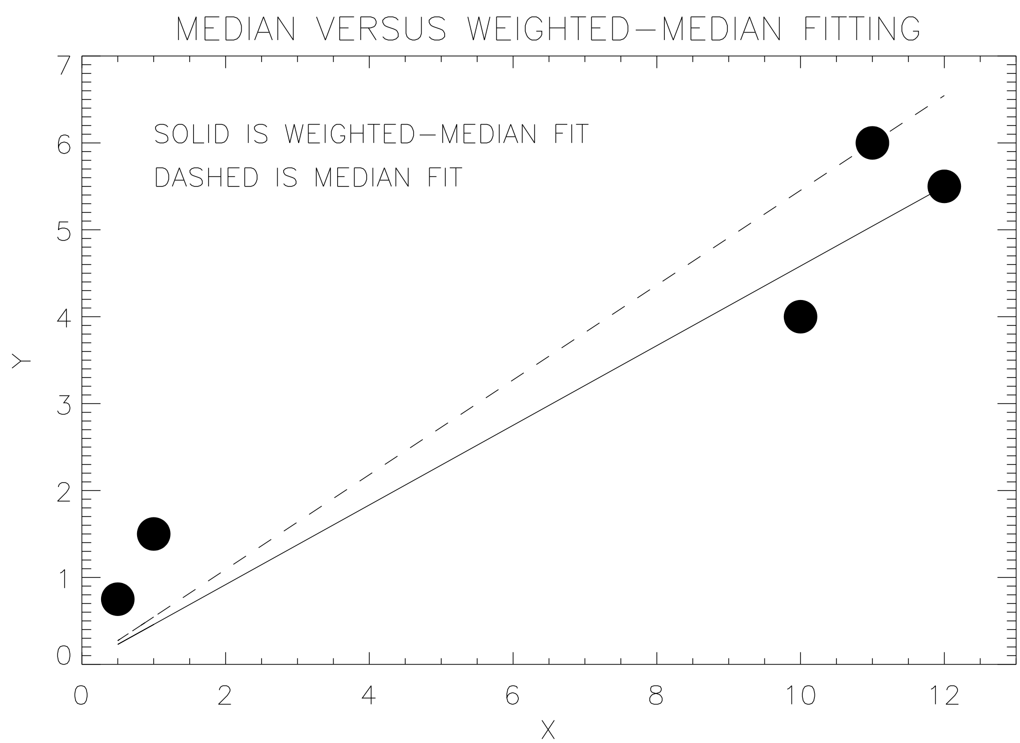

Fig. 13.1.— Illustrating the difference between the *median* and *weighted MARS* fits for a slope.

Let's explore the effect of this weeighted MARS by considering a simple example, shown in Figure 13.1. Five datapoints are shown with black circles. The dashed line is the median fit; there are two points above and two below the line, so it is the true median. But this fit is unsatisfying because it's the datapoints at large x that matter for determining the slope.

The solid line is the weighted *MARS* fit. This is satisfying because the points at large x cluster around it instead of lying to one side, and the points at small x (which have comparable residuals to the others) don't (and intuitively shouldn't) matter as much.

Our intuition, as well as the math in equation 13.4, tells us that the weighted *MARS* is the appropriate choice. This is valid for not only this example, but for *any* combination of functions such as the general case formulated in equations 12.1.

13.2. The General Technique for Weighted MARS Fitting

By being clever with weights w_m we can formulate a general technique for weighted MARS fitting[8]. Consider chi-square fitting the function $y = \mathbf{a} \cdot \mathbf{f}(\mathbf{x})$, in which $\mathbf{a}$ is an N-long vector of unknown parameters and $\mathbf{f}(\mathbf{x})$ a set of N basis functions. In §12.1 we reviewed the definition of χ^2 and the resulting equations for minimization; this discussion culminated in equation 12.2 in which we included a weight w_m, specified by the user to accomplish some particular objective. We repeat that equation here:

$$\sum_{m=0}^{M-1} \frac{w_m f_n(x_m) \Delta y_m}{\sigma_m^2} = 0 \ . \tag{13.5}$$

Here our objective is reproduce the spirit of equation 13.4, in which all dependence Δy_m (but not on f_n or σ_m^2) is removed so that the sum is the multifunction equivalent of equivalent of equation 13.4. To accomplish this, we simply choose

$$w_m = \frac{1}{|\Delta y_m|} \ . \tag{13.6}$$

So weighted MARS fitting is just another version of the sliding weight technique of §12.1.

[8]It is fortunate that weighted MARS, instead of median, fitting is what we want. Otherwise we could not formulate the solution using w_m because, when y depends on more than one function of x, one would need a separate weight for each function—and that is completely incompatible with the general least-squares approach.

13.3. Implementation, a Caution, and When To Stop Iterating

Implementation: To implement MARS fitting, you include a factor $w_m^{1/2}$ in the diagonal elements of equation 8.6. Begin with $w_m = 1$ and iterate.

A caution: You might get too close to one point, making its $\Delta y_{m,central} \rightarrow 0$. Then in equation 13.6 $w_{m,central} \rightarrow \infty$ and your solution goes crazy. You can build in a limit to take the lesser value, i.e. something like $w_m = (10^9 < 1/|\Delta y_m|)$ ($<$ means "whichever is smaller").

When to stop? You might think that all you need to do is keep track of the minimum value of $|\Delta y_m|$ and stop when this minimum value stops changing by very much. In my experience, this doesn't work very well. It's far better to keep tract of the corrections to each and every parameter that are derived on successive iterations. When you reach convergence, these corrections will asymptotically approach zero.

13.4. Errors in the Derived Parameters

In conventional least squares, you use the covariance matrix and the variance of the datapoints, as in equation 3.7. This is definitely *not* what you want here. For example, suppose you have a singly wildly discrepant datapoint and evaluate σ^2 in the usual way. Then the sum $\sum \Delta y_m^2$ reflects that single datapoint and this colors all the derived errors. This isn't fixed by using the reduced chi-square $\widehat{\chi^2}$, which will be nowhere near unity. Replacing the sum of residuals squared by the ARS seems reasonable until you realize that the MARS coefficient values are completely independent of the residuals—yet, you'd expect the errors to be smaller for better data! A similar concern holds for replacing the ARS by its weighted counterpart.

My current—but untested—recommendation is this. Suppose you have M datapoints. Conventionally, we define the $+1\sigma$ dispersion by the boundary where 34.2% of the points lie outside the limit, and ditto for the -1σ boundary. These boundaries are defined only by the *number* of daapoints outside the boundaries, not how big the residuals are. We can do the same here. Define the sample variance not by the sum-of-squares of residuals as in equation 3.1, but rather by how far away from the MARS fitted line you need to go before 34.2% of the points lie outside the plus-and-minus boundaries. Then use this fake variance in equation 3.7.

13.5. Pedantic Comment: The MARS and the Double-sided Exponential pdf

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

$$p_{(\Delta y_m)} = \frac{e^{-|\Delta y_m|/\sigma_m}}{2\sigma_m} \ . \tag{13.7}$$

where, again, $\Delta y_m = (y_m - \mathbf{a} \cdot \mathbf{f(x_m)})$. 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)

$$\mathcal{L}_{(\Delta y_m)} = \log(L_{(\Delta y_m)}) = -\sum_{m=0}^{M-1} \left[\frac{|y_m - \mathbf{a} \cdot \mathbf{f(x_m)})|}{\sigma_m} \right] . \tag{13.8a}$$

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

$$\mathcal{L}_{(\Delta y_m)}) = \sum_{\Delta y_m > 0} \frac{y_m - \mathbf{a} \cdot \mathbf{f(x_m)}}{\sigma_m} - \sum_{\Delta y_m < 0} \frac{y_m - \mathbf{a} \cdot \mathbf{f(x_m)}}{\sigma_m} . \tag{13.8b}$$

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

$$\frac{dL}{da_n} = \sum_{\Delta y > 0} \frac{f_n(x_m)}{\sigma_m} - \sum_{\Delta y < 0} \frac{f_n(x_m)}{\sigma_m} = 0 , \tag{13.9}$$

which is the MARS fit.

13.6. IDL's related resources

IDL provides the `median` function, which uses sorting and is much faster than our general weighted-MARS technique—but of course cannot deal with functional forms. IDL also provides `ladfit` ("least absolute deviation fit"), which does a weighted-MARS fit for a straight line. My routine, `polyfit_median`, does a weighted-MARS fit for an arbitrary polynomial; it is slightly less accurate as `ladfit` for an odd number of points but is slightly better for an even number.

14. FITTING WHEN MORE THAN ONE MEASURED PARAMETERS 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 y, as in equation 0.1. This essential assumption means that t is known with high precision and all the uncertainty is in y, and you are minimizing the squares of the residuals in the y-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.

Thanks to Jefferys (1980), the ML formulation of this problem is straightforward. Nevertheless, there is a lot of confusion on such fitting and not-inconsiderable propagation of myth. Before reviewing Jefferys' formulation, let's see two approaches:

1. Taylor §8.4 argues that you can account for x-variance $\sigma^2_{x_m}$ by increasing the y-variance by the usual error propagation, i.e. define an equivalent y-variance $\sigma^2_{y_m}(equiv) = [\sigma^2_{y_m} + (a_1\sigma_{x_m})^2]$, where a_1 is the slope. This is equivalent to our results below.

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.

14.1. 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 0.5, where we explicitly write the normal equations for fitting a straight line of the form $As_m + Bt_m = y_m$. 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, y). Then we are fitting the set of M equations

$$B x_m = y_m \; . \tag{14.1a}$$

The set of two normal equations becomes just the single equation

$$B[x^2] = [xy] \; , \tag{14.1b}$$

or, writing out the sums explicitly,

$$B = \frac{\sum_{m=0}^{M-1} {x^*}_m y_m}{\sum_{m=0}^{M-1} {x^*}_m^2} \; . \tag{14.1c}$$

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

$${x^*}_m = x_m - \delta x_m \; , \tag{14.2}$$

where δx_m is the observational error in point m. If we do standard least squares on this situation, then we (incorrectly) rewrite equation 14.1c to read

$$B = \frac{\sum_{m=0}^{M-1} x_m y_m}{\sum_{m=0}^{M-1} x_m^2} \; , \tag{14.3a}$$

that is, using x_m instead of ${x^*}_m$ (because we don't know what x^*_m is!). Substituting equation 14.2, and remembering that $y_m = {y^*}_m = B{x^*}_m$, we have

$$B = \frac{\sum_{m=0}^{M-1} {y^*}_m ({x^*}_m + \delta x_m)}{\sum_{m=0}^{M-1} ({x^*}_m^2 + 2{x^*}_m \delta x_m + \delta x_m^2)} \; . \tag{14.3b}$$

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!

14.2. Jefferys' Method: Introduction

Elsewhere we have regarded y as the dependent variable with measured errors and x (and its multiple counterparts) as independent variables that have no measured errors. But sometimes *all* of the measured parameter's have uncertainties. In this case the distinction between "dependent" and "independent" variables disappears and we need to treat them symmetrically. Moreover, we can have an arbitrarily large number of variables. This means we need to generalize our notation—and our mathematical technique. Our treatment follows Jefferys (1980) and we will adopt his notation in some measure. To explain the method we will follow his example and present the general case and also show the application to a specific example.

Figure 14.1 compares a conventional fit to $y = a_0 + a_1 t$ with a proper fit when both variables have uncertainties. The right-hand panel is the conventional fit in which the measurement uncertainties in t are set equal to zero; the left-hand panel includes the uncertainties in t. Some important differences are apparent:

1. Errors on the left panel are completely specified by errorbars. On the right panel we could use vertical errorbars for y and horizontal ones for t, but these would be sufficiently only if the errors were uncorrelated (as they are for points 0, 2, and 4). However, when they are correlated we must use error *ellipses* that specify $\chi^2 = 1$.

2. For the left panel, the best-fit points have the same t values as the datapoints but different y values. For the right panel the best-fit values differ from the data values for *both* variables.

3. If you look carefully, you'll see that the fitted slope on the left is a bit smaller than that on the right. This systematic bias arises from ignoring the errors in t.

14.3. The Data Matrix and Vector

First, a word on notation. We distinguish between scalars, vectors, and matrices as follows: for Roman letters, vectors are **lower-case bold** and matrices are **UPPER-CASE BOLD**. For Greek letters, scalars are the letter itself (e.g. α), vectors are single-underlined (e.g. $\underline{\phi}$), and matrices are double-underlined (e.g. $\underline{\underline{\sigma}}$).

For the general case we have M experiments. In each experiment we measure J parameters. We want to combine these $(M \times J)$ measurements to derive N coefficients. We denote the matrix of measured parameters by $\mathbf{X}$, which is $(M \times J)$ [using conventional matrix notation, not IDL notation, in which in the vertical dimension (number of rows) is M and the horizontal dimension (number of columns) is J]. We will want to concatenate this matrix to a one-dimensional vector $\mathbf{x}$ of length (MJ). Specifically, the $M \times J$ datapoint matrix is

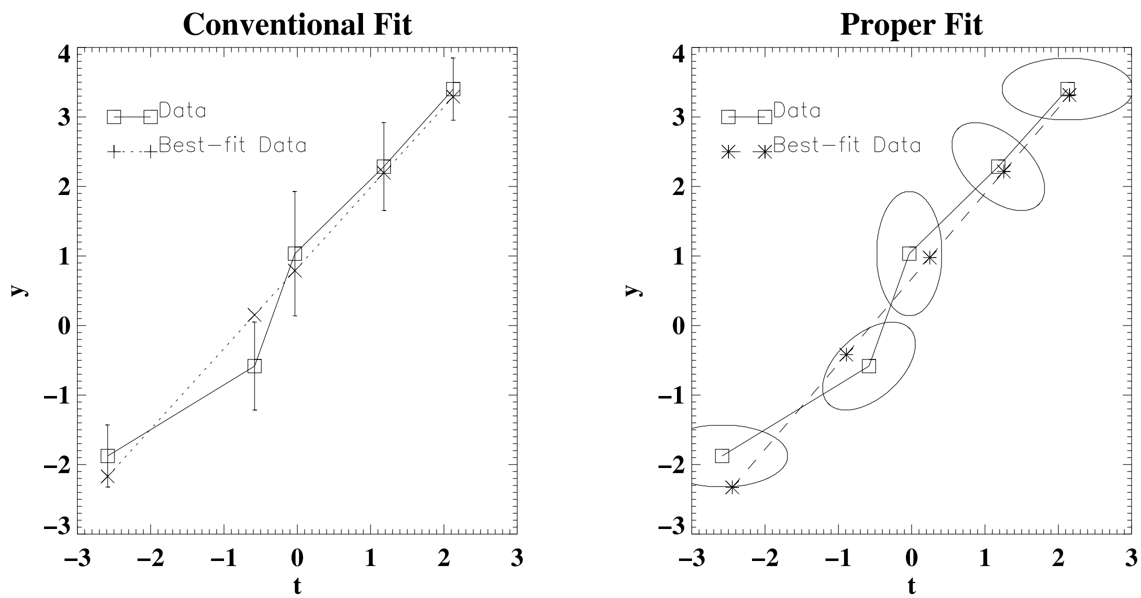

Fig. 14.1.— Comparing a conventional fit for to $y = a_0 + a_1 t$ (left panel) to a proper one when both measured variables have errors. On the right, the ellipses denote the correlated errors in (t, y); these are the generalization of the errorbars on the left. The right-hand slope is a bit steeper than the left-hand one.

$$\mathbf{X} = \begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} & \cdots & x_{0,J-1} \\ x_{1,0} & x_{1,1} & x_{1,2} & \cdots & x_{1,J-1} \\ x_{2,0} & x_{2,1} & x_{2,2} & \cdots & x_{2,J-1} \\ . & . & . & \cdots & . \\ . & . & . & \cdots & . \\ . & . & . & \cdots & . \\ x_{M-1,0} & x_{M-1,1} & x_{M-1,2} & \cdots & x_{M-1,J-1} \end{bmatrix} \tag{14.4a}$$

We don't use this big matrix in the solution. Instead, we turn it into a vector in which the first J elements are the data for $m = 0$, the second J elements are for $m = 1$, etc. So the vector has dimensions $(MJ) \times 1$, like this:

$$\mathbf{x} = \begin{bmatrix} x_{0,0} \\ x_{0,1} \\ x_{0,2} \\ . \\ . \\ . \\ x_{0,J-1} \\ x_{1,0} \\ x_{1,1} \\ x_{1,2} \\ . \\ . \\ . \\ x_{1,J-1} \\ x_{2,0} \\ x_{2,1} \\ x_{2,2} \\ . \\ . \\ . \\ x_{2,J-1} \\ \dots \\ \dots \\ \dots \\ x_{M-1,0} \\ x_{M-1,1} \\ x_{M-1,2} \\ . \\ . \\ . \\ x_{M-1,J-1} \end{bmatrix} \tag{14.4b}$$

One important reason for writing the whole set of data as a vector instead of a matrix is to make it possible to write the covariance matrix for all measured data in the conventional form, as we now discuss.

14.4. The Data Covariance Matrix and Defining Chi-Square

The whole object of fitting is to minimize the chi-square. When all measured parameters have uncertainties, their uncertainties can be correlated. We have to generalize the definition of chi-square accordingly.

First, suppose that the observational errors in the datapoints are uncorrelated. Then the intrinsic variance of each datapoint is described by a single number. In our example, for uncorrelated observational errors we'd have the variances in the y values be $(\sigma_{y0}^2, \sigma_{y1}^2, \ldots)$, and similarly for the t values; this would give

$$\chi^2 = \sum_m \frac{\delta y_m^2}{\sigma_{ym}^2} + \frac{\delta t_m^2}{\sigma_{tm}^2} \tag{14.5}$$

However, it is common that errors are correlated. For example, if we were fitting y to a polynomial in t, then the errors in the various powers of t would certainly be correlated. More generally, then, the *covariances* among the different measured values are nonzero. These covariances are the off-diagonal terms in the covariance matrix. Thus, if we denote the covariance matrix for the measured $\mathbf{x}$ values by the $(MJ \times (MJ)$ matrix $\underline{\underline{\sigma}}$, then the most general case has $\underline{\underline{\sigma}}$ with no nonzero elements.

Less general, but much more common, is the situation shown in Figure 14.1. Here, the covariances among the J measured parameters nonzero are for a *particular* experiment m, but the covariances from one experiment m to another are zero; in other words, each experiment is completely independent of the others. In this less general but very common case, the covariance matrix looks like this. (*Note:* we denote the covariance matrix by $\underline{\underline{\sigma}}$, but it contains *variances*, not dispersions.)

$$\underline{\underline{\sigma}} = \begin{bmatrix} \underline{\underline{\sigma_0}} & \mathbf{0} & \mathbf{0} & \ldots \\ \mathbf{0} & \underline{\underline{\sigma_1}} & \mathbf{0} & \ldots \\ \mathbf{0} & \mathbf{0} & \underline{\underline{\sigma_2}} & \ldots \\ . & & & \\ . & & & \\ . & & & \end{bmatrix} \tag{14.6}$$

Here, each element (including the $\mathbf{0}$ elements) is itself a $J \times J$ matrix. For our specific example of §14.7, $J = 2$ so $\underline{\underline{\sigma_0}}$ is a covariance matrix of the form

$$\underline{\underline{\sigma_0}} = \begin{bmatrix} \sigma_{yy} & \sigma_{yt} \\ \sigma_{yt} & \sigma_{tt} \end{bmatrix} \tag{14.7}$$

Generally, the chi-square is given by (e.g. Cowan equation 2.39)

$$\chi^2 = \delta\mathbf{x}^{\mathbf{T}} \cdot \underline{\underline{\sigma^{-1}}} \cdot \delta\mathbf{x} \tag{14.8}$$

14.5. Formulation of the Problem and its Solution with Lagrange Multipliers

We will be referring to various versions of the data parameters $\mathbf{x}$ and derived parameters $\mathbf{a}$: *measured*, *best-fit*, and (for the iterative solution) *guessed*. The subscript d denotes the set of *measured datapoints*, of which there are (JM). The subscript $*$ denotes the set of *best-fit* quantities; these parameters include not only the datapoints $\mathbf{x}$, but also the derived parameters $\mathbf{a}$. We will be doing an iterative fit using *guessed* values of both the data and derived parameters, represented by the subscript g.

We begin by writing *exact* equations for each measurement. The fitted vales, subscripted with stars, satisfy the *exact* equations of condition

$$\boldsymbol{f}(\boldsymbol{x}_*, \boldsymbol{a}_*) = \mathbf{0} \tag{14.9a}$$

This is an M-long vector of functions $\mathbf{f}(\mathbf{x}, \mathbf{a}) = \mathbf{0}$ (one row for each measurement). This set of M equations doesn't do us much good because we don't know the best-fit (starred) values. Consequently, for the datapoints we define the difference between the best-fit and measured data values

$$\delta\mathbf{x} = \mathbf{x_d} - \mathbf{x}_* \tag{14.9b}$$

This is the *negative* of Jefferys' definition of the corresponding quantity $\hat{\mathbf{v}}$ in his section II. With this, the equation 14.9a becomes

$$\mathbf{f}(\mathbf{x_d} - \delta\mathbf{x}, \mathbf{a}_*) = \mathbf{0} \ . \tag{14.9c}$$

Our goal is to solve these M equations for the (MJ) differences $\delta\mathbf{x}$ and the N parameters $\mathbf{a}_*$ and, simultaneously, minimize χ^2.

This is a classic minimization problem: we minimize χ^2 with respect to the $(MJ+N)$ values of $\delta\mathbf{x}$ and $\mathbf{a}$, subject to the M constraints of equation 14.9c. Such problems are solved using Lagrange multipliers. Here, the M Lagrange multipliers form the vector $\underline{\lambda}$. We define the Lagrangian $\mathcal{L}$ as

$$\mathcal{L} = \left[\frac{1}{2}\delta\mathbf{x}^{\mathbf{T}} \cdot \underline{\underline{\sigma^{-1}}} \cdot \delta\mathbf{x}\right] + \left[\mathbf{f}^{\mathbf{T}}(\mathbf{x_d} - \delta\mathbf{x}, \mathbf{a}) \cdot \underline{\lambda}\right] \ ; \tag{14.10}$$

the $\frac{1}{2}$ arises because, for a Gaussian pdf for the errors, the residuals are distributed as $e^{-\frac{\chi^2}{2}}$ (e.g. Cowan equation 2.28). We differentiate $\mathcal{L}$ with respect to each of the unknowns $\delta\mathbf{x}$ and $\mathbf{a}$. The solution provides $\mathbf{a}$, the vector of N derived coefficients (so $\mathbf{a}$ is defined as elsewhere in this tutorial) together with the MJ fitted datapoints. To proceed, we need the derivatives of $\mathbf{f}$ in equation 14.9a with respect to the vectors $\mathbf{x}$ and $\mathbf{a}$, as we now discuss.

14.6. The Derivative Matrices

For the analysis we will need the derivatives of $\mathbf{f}$ with respect to the vectors $\mathbf{x}$ and $\mathbf{a}$. The derivatives will always be evaluated at the *guessed* values $\mathbf{x_g}$ and $\mathbf{a_g}$. The derivative with respect to $\mathbf{x}$ is a $M \times MJ$ matrix and looks like this [We take the case $(M = 4, J = 3)$ for transparency; subscripts of x are in the order (m, j)]:

$$\left.\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{a})}{\partial \mathbf{x}}\right|_{\mathbf{x_g},\mathbf{a_g}} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_{0,0}} & \frac{\partial \mathbf{f}}{\partial x_{0,1}} & \frac{\partial \mathbf{f}}{\partial x_{0,2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial \mathbf{f}}{\partial x_{1,0}} & \frac{\partial \mathbf{f}}{\partial x_{1,1}} & \frac{\partial \mathbf{f}}{\partial x_{1,2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial \mathbf{f}}{\partial x_{2,0}} & \frac{\partial \mathbf{f}}{\partial x_{2,1}} & \frac{\partial \mathbf{f}}{\partial x_{2,2}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial \mathbf{f}}{\partial x_{M-1,0}} & \frac{\partial \mathbf{f}}{\partial x_{M-1,1}} & \frac{\partial \mathbf{f}}{\partial x_{M-1,2}} \end{bmatrix} \tag{14.11}$$

where all the derivatives are evaluated at $(\mathbf{x_g}, \mathbf{a_g})$. Much easier is the derivative with respect to $\mathbf{a}$, which is a $M \times N$ matrix and looks like

$$\left.\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{a})}{\partial \mathbf{a}}\right|_{\mathbf{x_g},\mathbf{a_g}} = \begin{bmatrix} \left.\frac{\partial \mathbf{f}}{\partial \mathbf{a_0}}\right|_{\mathbf{x_{g,0}},\mathbf{a_g}} & \left.\frac{\partial \mathbf{f}}{\partial \mathbf{a_1}}\right|_{\mathbf{x_{g,0}},\mathbf{a_g}} & \left.\frac{\partial \mathbf{f}}{\partial \mathbf{a_2}}\right|_{\mathbf{x_{g,0}},\mathbf{a_g}} & \cdots & \left.\frac{\partial \mathbf{f}}{\partial \mathbf{a_{N-1}}}\right|_{\mathbf{x_{g,0}},\mathbf{a_g}} \\ \left.\frac{\partial \mathbf{f}}{\partial \mathbf{a_0}}\right|_{\mathbf{x_{g,1}},\mathbf{a_g}} & \left.\frac{\partial \mathbf{f}}{\partial \mathbf{a_1}}\right|_{\mathbf{x_{g,1}},\mathbf{a_g}} & \left.\frac{\partial \mathbf{f}}{\partial \mathbf{a_2}}\right|_{\mathbf{x_{g,1}},\mathbf{a_g}} & \cdots & \left.\frac{\partial \mathbf{f}}{\partial \mathbf{a_{N-1}}}\right|_{\mathbf{x_{g,1}},\mathbf{a_g}} \\ \left.\frac{\partial \mathbf{f}}{\partial \mathbf{a_0}}\right|_{\mathbf{x_{g,2}},\mathbf{a_g}} & \left.\frac{\partial \mathbf{f}}{\partial \mathbf{a_1}}\right|_{\mathbf{x_{g,2}},\mathbf{a_g}} & \left.\frac{\partial \mathbf{f}}{\partial \mathbf{a_2}}\right|_{\mathbf{x_{g,2}},\mathbf{a_g}} & \cdots & \left.\frac{\partial \mathbf{f}}{\partial \mathbf{a_{N-1}}}\right|_{\mathbf{x_{g,2}},\mathbf{a_g}} \\ & & \cdot & & \\ & & \cdot & & \\ & & \cdot & & \end{bmatrix} \tag{14.12}$$

where again, all the derivatives are evaluated at $(\mathbf{x_g}, \mathbf{a_g})$.

14.7. The Specific Example

We illustrate the above with this specific example, for which we fit a first-order polynomial to (t, y) of the form $[y = a_0 + a_1 t]$; the coefficients are (a_0, a_1) and $[\mathbf{f}(\mathbf{x}, \mathbf{a}) = y - a_0 - a_1 t]$. For this example, the matrix of measured parameters is

$$\mathbf{X} = \begin{bmatrix} y_{d0} & t_{d0} \\ y_{d1} & t_{d1} \\ y_{d2} & t_{d2} \\ . & \\ . & \\ . & \end{bmatrix} \tag{14.13}$$

and the concatenated vector version is

$$\mathbf{x} = \begin{bmatrix} y_{d0} \\ t_{d0} \\ y_{d1} \\ t_{d1} \\ y_{d2} \\ t_{d2} \\ . \\ . \\ . \end{bmatrix} \tag{14.14}$$

The exact vector-of-functions equation $\mathbf{f}(\mathbf{x}_*, \mathbf{a}_*) = \mathbf{0}$ uses the starred (best-fit) values, and is

$$\mathbf{f}(\mathbf{x}_*, \mathbf{a}_*) = \begin{bmatrix} y_{*0} - a_{*0} - a_{*1}t_{*0} \\ y_{*1} - a_{*0} - a_{*1}t_{*1} \\ . \\ . \\ . \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ . \end{bmatrix} \tag{14.15}$$

The derivative matrix of $\mathbf{f}$ with respect to the vector $\mathbf{x}$ is

$$\left.\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{a})}{\partial \mathbf{x}}\right|_{\mathbf{x_g}, \mathbf{a_g}} = \begin{bmatrix} 1 & -a_{g1} & 0 & 0 & 0 & 0 & \ldots \\ 0 & 0 & 1 & -a_{g1} & 0 & 0 & \ldots \\ 0 & 0 & 0 & 0 & 1 & -a_{g1} & \ldots \\ & & & . & & & \\ & & & . & & & \\ & & & . & & & \end{bmatrix}, \tag{14.16}$$

always evaluated at the *guessed* values of the parameters (subscript g). The derivative matrix of $\mathbf{f}$ with respect to the vector $\mathbf{a}$ is

$$\left.\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{a})}{\partial \mathbf{a}}\right|_{\mathbf{x_g},\mathbf{a_g}} = \begin{bmatrix} -1 & -t_{g0} \\ -1 & -t_{g1} \\ -1 & -t_{g2} \\ . & \\ . & \\ . & \end{bmatrix}, \tag{14.17}$$

again always evaluated at the *guessed* values of the parameters (subscript g).

14.8. The Solution to the Lagrangian: Two Matrix Equations

Jefferys does all this this[9] for us and, after some algebraic manipulation, provides the two following matrix equations (from the set of four in his equation 10):

$$\underline{\underline{\sigma}}^{\mathbf{-1}} \cdot \delta\mathbf{x} + \left.\frac{\partial \mathbf{f^T}(\mathbf{x},\mathbf{a})}{\partial \mathbf{x}}\right|_{\mathbf{x_d},\mathbf{a_*}} \cdot \underline{\lambda} = \mathbf{0} \tag{14.18a}$$

This single matrix equation embodies MJ individual equations. Here we write $\frac{\partial \mathbf{f^T}}{\partial \mathbf{x}}$ instead of $\frac{\partial \mathbf{f^T}}{\partial \delta\mathbf{x}}$ for clarity, and use the fact that they are the negative of each other.

$$\left.\frac{\partial \mathbf{f^T}(\mathbf{x},\mathbf{a})}{\partial \mathbf{a}}\right|_{\mathbf{x_d},\mathbf{a_*}} \cdot \underline{\lambda} = \mathbf{0} \tag{14.18b}$$

This matrix equation embodies N individual equations. These two matrix equations embody $MJ+N$ individual equations, which is equal to the number of unknowns, so we can solve for them! In both of these equations, the dimensions of the factors are (*Note the transposes!*):

$$\underline{\underline{\sigma}} : (MJ) \times (MJ) \tag{14.19a}$$

$$\delta\mathbf{x} : MJ \times 1 \tag{14.19b}$$

$$\mathbf{f^T} : 1 \times M \tag{14.19c}$$

$$\frac{\partial \mathbf{f^T}(\mathbf{x},\mathbf{a})}{\partial \mathbf{x}} : (MJ) \times M \tag{14.19d}$$

---

[9]And more: he includes the possibility for additional constraints, the second line in his equation (7). This introduces additional complications and we ignore this possibility in the interest of pedagogical simplification. For example, his set of equations (10) has four equations, not two as we write here.

$$\underset{\sim}{\lambda} : M \times 1 \tag{14.19e}$$

$$\frac{\partial \mathbf{f^T}(\mathbf{x}, \mathbf{a})}{\partial \mathbf{a}} : N \times M \tag{14.19f}$$

Note that in 14.19d above the dimension is $(MJ) \times M$: M elements in $\mathbf{f}$, differentiated by (MJ) different variables $\mathbf{x}$; similarly for 14.19f above.

14.9. Solving Equations 14.18a and 14.18b Iteratively

Generally, these equations are nonlinear and we solve them iteratively using guessed solutions. We denote the guessed values with subscript g, so we have

$$\mathbf{a_g} = \text{guessed values for } \mathbf{a}_* \tag{14.20a}$$

$$\mathbf{x_g} = \text{guessed values for } \mathbf{x}_* \tag{14.20b}$$

ITERATION STEP 1: We define the difference between the measured data quantities and their currently-guessed counterparts

$$\mathbf{\Delta x_g} \equiv \mathbf{x_d} - \mathbf{x_g} \tag{14.20c}$$

Above, our $\mathbf{\Delta x_g}$ is the *negative* of Jefferys' $\hat{\mathbf{v}}$. The nonlinear fit solves for corrections to these guesses, which we denote by $\mathbf{\Delta x_{new}}$ (the negative of Jefferys' $\hat{\mathbf{v}}_{\mathbf{new}}$) and $\mathbf{\Delta a_{new}}$ (which is identical to Jefferys' $\hat{\underset{\sim}{\delta}}$).

ITERATION STEP 2: We define the following:

1. The $M \times M$ weight matrix $\mathbf{W}$ is from Jefferys' equation (15):

$$\mathbf{W} \equiv \left[\left. \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{a})}{\partial \mathbf{x}} \right|_{\mathbf{x_g}, \mathbf{a_g}} \cdot \underline{\underline{\sigma}} \cdot \left. \frac{\partial \mathbf{f^T}(\mathbf{x}, \mathbf{a})}{\partial \mathbf{x}} \right|_{\mathbf{x_g}, \mathbf{a_g}} \right]^{-1} \tag{14.21a}$$

2. The $[0, 0]$ element of his equation (17), which is equivalent to our $\underline{\underline{\alpha}}$ (the curvature matrix) elsewhere in this document:

$$\underline{\underline{\alpha}} \equiv \left. \frac{\partial \mathbf{f^T}(\mathbf{x}, \mathbf{a})}{\partial \mathbf{a}} \right|_{\mathbf{x_g}, \mathbf{a_g}} \cdot \mathbf{W} \cdot \left. \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{a})}{\partial \mathbf{a}} \right|_{\mathbf{x_g}, \mathbf{a_g}} \tag{14.21b}$$

$\underline{\underline{\alpha}}$ is $N \times N$.

3. The modified equations of condition from his equation (18) (we have a + instead of his − because $\mathbf{\Delta x_g} = -\hat{\mathbf{v}}$)

$$\underline{\phi_{\mathbf{g}}} \equiv \mathbf{f}(\mathbf{x_g}, \mathbf{a_g}) + \left(\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{a})}{\partial \mathbf{x}} \bigg|_{\mathbf{x_g}, \mathbf{a_g}} \right) \cdot \mathbf{\Delta x_g} \tag{14.21c}$$

$\underline{\phi_{\mathbf{g}}}$ is $M \times 1$.

ITERATION STEP 3: The solutions follow directly. The matrix equation for $\mathbf{\Delta a_{new}}$ is Jefferys equation (17)

$$\underline{\underline{\alpha}} \cdot \mathbf{\Delta a_{new}} = - \frac{\partial \mathbf{f^T}(\mathbf{x}, \mathbf{a})}{\partial \mathbf{a}} \bigg|_{\mathbf{x_g}, \mathbf{a_g}} \cdot \mathbf{W} \cdot \underline{\phi_{\mathbf{g}}} \tag{14.22}$$

which is solved for $\mathbf{\Delta a_{new}}$ in the conventional way by premultiplying both sides by $\underline{\underline{\alpha}}^{-1}$. The matrix equation for the new, corrected $\mathbf{\Delta x_g}$ is Jefferys equation (19)

$$\mathbf{\Delta x_{new}} = -\underline{\underline{\sigma}} \cdot \frac{\partial \mathbf{f^T}(\mathbf{x}, \mathbf{a})}{\partial \mathbf{x}} \bigg|_{\mathbf{x_g}, \mathbf{a_g}} \cdot \mathbf{W} \cdot \left(\underline{\phi_{\mathbf{g}}} + \frac{\partial \mathbf{f}}{\partial \mathbf{a}} \bigg|_{\mathbf{x_g}, \mathbf{a_g}} \cdot \mathbf{\Delta a_{new}} \right) \tag{14.23}$$

ITERATION STEP 4: The previous two equations provide corrections to the values $\mathbf{x_d}$ and $\mathbf{a_g}$. One applies them (Jefferys equation 20):

$$\mathbf{a_{g,new}} = \mathbf{a_g} + \mathbf{\Delta a_{new}} \tag{14.24a}$$

$$\mathbf{x_{g,new}} = \mathbf{x_d} + \mathbf{\Delta x_{new}} \tag{14.24b}$$

Note that $\mathbf{\Delta x_{new}}$ is applied to $\mathbf{x_d}$, not to $\mathbf{x_g}$.

ITERATION STEP 5: Return to **Iteration Step 1**, using these new values $\mathbf{a_{g,new}}$ and $\mathbf{x_{g,new}}$ in place of $\mathbf{a_g}$ and $\mathbf{x_g}$. Iterate until convergence. Convergence occurs when all derived corrrections $\mathbf{\Delta a_{new}}$ and $\mathbf{\Delta x_{new}}$ become negligible. [How do you define "negligible"...]

14.10. Taking all those derivatives!

Why not use numerical derivatives? It's easier with complicated functions!

14.11. The Initial Guess

Getting started is the hard part unless the problem is relatively easy. One way is to use conventional least squares to derive the initial guesses $\mathbf{a_g}$. To do this, you designate one particular variable as the dependent variable and all the others as the independent ones (for which you assume no errors). Do the conventional least squares solution for the parameters $\mathbf{a}$ and use these as the initial guesses. A good choice for the dependent variable is the one with the largest errors, if there is one; otherwise the choice is more-or-less arbitrary. For the initial guesses of the data parameters $\mathbf{x_g}$, just use the measured data values $\mathbf{x_d}$. If this scheme doesn't work, you're on your own!

14.12. The Covariance Matrix (and errors) of the Derived Parameters

The matrix $\underline{\underline{\alpha}}$, which is defined above in equation 14.21b, is the conventionally-defined curvature matrix and its inverse $\underline{\underline{\alpha^{-1}}}$ is the conventionally-defined covariance matrix. Because we have formulated this problem as a chi-squared one, the elements of this matrix give the covariance directly. *Thus, the errors in the derived parameters can be taken as the square-root of the diagonal elements of* $\underline{\underline{\alpha^{-1}}}$.

One usually wants to calculate the chi-squared of the solution. This involves not only the best-fit parameters $\mathbf{a}_*$ but also the best fit datapoints $\mathbf{x}_*$. To do this, use equation 14.8 using $\mathbf{\Delta x_g}$ in place of $\delta\mathbf{x}$. This is the same as Jefferys' equation (43).

The expectation value of χ^2 is the number of degrees of freedom. In one sense this is just like the usual least-squares solution: it's equal to the number of datapoints minus the number of derived parameters. Here the number of datapoints is not the number of experiments M; rather, it's JM. So the number of degrees of freedom is $JM - N$.

15. 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 §0 and 1 follows his development quite closely. I wrote the first version of this document before knowing of NR'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 §8.

On the other hand, NR 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_m. 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 §8.

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

$$\mathbf{X} \longleftrightarrow \mathbf{A} \tag{15.1a}$$

$$\mathbf{Y} \longleftrightarrow \mathbf{b} \tag{15.1b}$$

$$\mathbf{X^T} \cdot \mathbf{X} = \mathbf{XX} = [\alpha] \longleftrightarrow \mathbf{A^T} \cdot \mathbf{A} = [\alpha] \tag{15.1c}$$

$$\mathbf{XX}^{-1} = \mathbf{XXI} = [\alpha]^{-1} \longleftrightarrow [\alpha]^{-1} = [C] = \mathbf{C} \tag{15.1d}$$

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

$$N \longleftrightarrow M \tag{15.1e}$$

$$M \longleftrightarrow N \tag{15.1f}$$

Confusing, hey what?

It is a great pleasure to thank Tim Robishaw for his considerable effort in providing detailed comments on several aspects of this paper. These comments led to significant revisions and improvements. He also fixed bad formatting and manuscript glitches. Also, I am deeply appreciative to Berkeley undergraduate students (Spring 2006) Ashley Bacon and Tiffany Ussery for their persistence and care in discovering small but crucial errors in the original version of §14.

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.

Jefferys, W.H. 1980, AJ, 85, 177.

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

Stetson, P.B., *http://nedwww.ipac.caltech.edu/level5/Stetson/Stetson4.html.*

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

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