```
    *chi2=0.0;                                       Calculate χ².
    *q=1.0;
    if (mwt == 0) {
        for (i=1;i<=ndata;i++)
            *chi2 += SQR(y[i]-(*a)-(*b)*x[i]);
        sigdat=sqrt((*chi2)/(ndata-2));              For unweighted data evaluate typ-
        *siga *= sigdat;                                 ical sig using chi2, and ad-
        *sigb *= sigdat;                                 just the standard deviations.
    } else {
        for (i=1;i<=ndata;i++)
            *chi2 += SQR((y[i]-(*a)-(*b)*x[i])/sig[i]);
        if (ndata>2) *q=gammq(0.5*(ndata-2),0.5*(*chi2));     Equation (15.2.12).
    }
}
```

CITED REFERENCES AND FURTHER READING:

Bevington, P.R. 1969, *Data Reduction and Error Analysis for the Physical Sciences* (New York: McGraw-Hill), Chapter 6.

15.3 Straight-Line Data with Errors in Both Coordinates

If experimental data are subject to measurement error not only in the y_i's, but also in the x_i's, then the task of fitting a straight-line model

$$y(x) = a + bx \tag{15.3.1}$$

is considerably harder. It is straightforward to write down the χ^2 merit function for this case,

$$\chi^2(a,b) = \sum_{i=1}^{N} \frac{(y_i - a - bx_i)^2}{\sigma_{y\,i}^2 + b^2\sigma_{x\,i}^2} \tag{15.3.2}$$

where $\sigma_{x\,i}$ and $\sigma_{y\,i}$ are, respectively, the x and y standard deviations for the ith point. The weighted sum of variances in the denominator of equation (15.3.2) can be understood both as the variance in the direction of the smallest χ^2 between each data point and the line with slope b, and also as the variance of the linear combination $y_i - a - bx_i$ of two random variables x_i and y_i,

$$\text{Var}(y_i - a - bx_i) = \text{Var}(y_i) + b^2\text{Var}(x_i) = \sigma_{y\,i}^2 + b^2\sigma_{x\,i}^2 \equiv 1/w_i \tag{15.3.3}$$

The sum of the square of N random variables, each normalized by its variance, is thus χ^2-distributed.

We want to minimize equation (15.3.2) with respect to a and b. Unfortunately, the occurrence of b in the denominator of equation (15.3.2) makes the resulting equation for the slope $\partial\chi^2/\partial b = 0$ nonlinear. However, the corresponding condition for the intercept, $\partial\chi^2/\partial a = 0$, is still linear and yields

$$a = \left[\sum_i w_i(y_i - bx_i)\right] \Bigg/ \sum_i w_i \tag{15.3.4}$$

where the w_i's are defined by equation (15.3.3). A reasonable strategy, now, is to use the machinery of Chapter 10 (e.g., the routine `brent`) for minimizing a general one-dimensional function to minimize with respect to b, while using equation (15.3.4) at each stage to ensure that the minimum with respect to b is also minimized with respect to a.

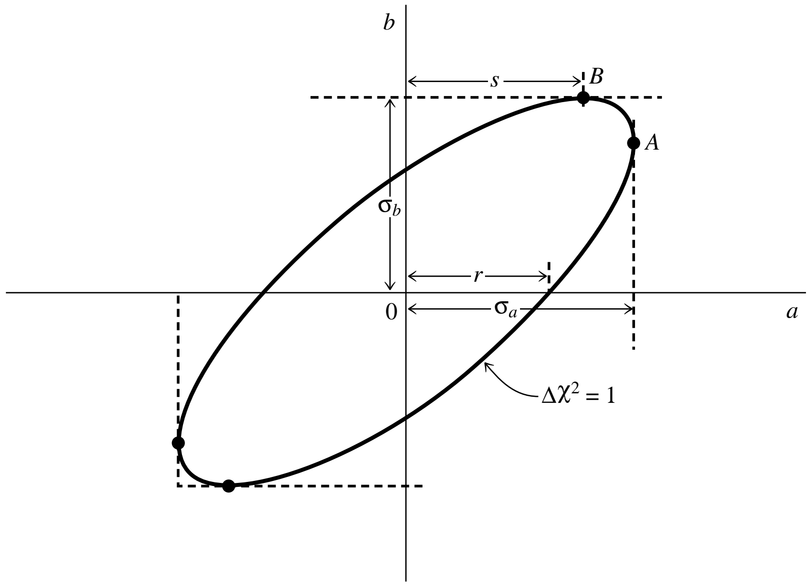

Figure 15.3.1. Standard errors for the parameters a and b. The point B can be found by varying the slope b while simultaneously minimizing the intercept a. This gives the standard error σ_b, and also the value s. The standard error σ_a can then be found by the geometric relation $\sigma_a^2 = s^2 + r^2$.

Because of the finite error bars on the x_i's, the minimum χ^2 as a function of b will be finite, though usually large, when b equals infinity (line of infinite slope). The angle $\theta \equiv \arctan b$ is thus more suitable as a parametrization of slope than b itself. The value of χ^2 will then be periodic in θ with period π (not 2π!). If any data points have very small σ_y 's but moderate or large σ_x 's, then it is also possible to have a maximum in χ^2 near zero slope, $\theta \approx 0$. In that case, there can sometimes be two χ^2 minima, one at positive slope and the other at negative. Only one of these is the correct global minimum. It is therefore important to have a good starting guess for b (or θ). Our strategy, implemented below, is to scale the y_i's so as to have variance equal to the x_i's, then to do a conventional (as in §15.2) linear fit with weights derived from the (scaled) sum $\sigma_{y\,i}^2 + \sigma_{x\,i}^2$. This yields a good starting guess for b if the data are even *plausibly* related to a straight-line model.

Finding the standard errors σ_a and σ_b on the parameters a and b is more complicated. We will see in §15.6 that, in appropriate circumstances, the standard errors in a and b are the respective projections onto the a and b axes of the "confidence region boundary" where χ^2 takes on a value one greater than its minimum, $\Delta\chi^2 = 1$. In the linear case of §15.2, these projections follow from the Taylor series expansion

$$\Delta\chi^2 \approx \frac{1}{2}\left[\frac{\partial^2\chi^2}{\partial a^2}(\Delta a)^2 + \frac{\partial^2\chi^2}{\partial b^2}(\Delta b)^2\right] + \frac{\partial^2\chi^2}{\partial a \partial b}\Delta a \Delta b \qquad (15.3.5)$$

Because of the present nonlinearity in b, however, analytic formulas for the second derivatives are quite unwieldy; more important, the lowest-order term frequently gives a poor approximation to $\Delta\chi^2$. Our strategy is therefore to find the roots of $\Delta\chi^2 = 1$ numerically, by adjusting the value of the slope b away from the minimum. In the program below the general root finder `zbrent` is used. It may occur that there are no roots at all — for example, if all error bars are so large that all the data points are compatible with each other. It is important, therefore, to make some effort at bracketing a putative root before refining it (cf. §9.1).

Because a is minimized at each stage of varying b, successful numerical root-finding leads to a value of Δa that minimizes χ^2 for the value of Δb that gives $\Delta\chi^2 = 1$. This (see Figure 15.3.1) directly gives the tangent projection of the confidence region onto the b axis, and thus σ_b. It does not, however, give the tangent projection of the confidence region onto the a axis. In the figure, we have found the point labeled B; to find σ_a we need to find the point A. Geometry to the rescue: To the extent that the confidence region is approximated by an ellipse, then you can prove (see figure) that $\sigma_a^2 = r^2 + s^2$. The value of s is known from having found the point B. The value of r follows from equations (15.3.2) and (15.3.3) applied at the χ^2 minimum (point O in the figure), giving

$$r^2 = 1 \Big/ \sum_i w_i \tag{15.3.6}$$

Actually, since b can go through infinity, this whole procedure makes more sense in (a,θ) space than in (a,b) space. That is in fact how the following program works. Since it is conventional, however, to return standard errors for a and b, not a and θ, we finally use the relation

$$\sigma_b = \sigma_\theta / \cos^2\theta \tag{15.3.7}$$

We caution that if b and its standard error are both large, so that the confidence region actually includes infinite slope, then the standard error σ_b is not very meaningful. The function `chixy` is normally called only by the routine `fitexy`. However, if you want, you can yourself explore the confidence region by making repeated calls to `chixy` (whose argument is an angle θ, not a slope b), after a single initializing call to `fitexy`.

A final caution, repeated from §15.0, is that if the goodness-of-fit is not acceptable (returned probability is too small), the standard errors σ_a and σ_b are surely not believable. In dire circumstances, you might try scaling all your x and y error bars by a constant factor until the probability is acceptable (0.5, say), to get more plausible values for σ_a and σ_b.

```
#include <math.h>
#include "nrutil.h"
#define POTN 1.571000
#define BIG 1.0e30
#define PI 3.14159265
#define ACC 1.0e-3

int nn;                                          Global variables communicate with
float *xx,*yy,*sx,*sy,*ww,aa,offs;                  chixy.

void fitexy(float x[], float y[], int ndat, float sigx[], float sigy[],
    float *a, float *b, float *siga, float *sigb, float *chi2, float *q)
```

Straight-line fit to input data `x[1..ndat]` and `y[1..ndat]` with errors in both x and y, the respective standard deviations being the input quantities `sigx[1..ndat]` and `sigy[1..ndat]`. Output quantities are `a` and `b` such that $y = a + bx$ minimizes χ^2, whose value is returned as `chi2`. The χ^2 probability is returned as `q`, a small value indicating a poor fit (sometimes indicating underestimated errors). Standard errors on `a` and `b` are returned as `siga` and `sigb`. These are not meaningful if either (i) the fit is poor, or (ii) b is so large that the data are consistent with a vertical (infinite b) line. If `siga` and `sigb` are returned as `BIG`, then the data are consistent with *all* values of b.

```
{
    void avevar(float data[], unsigned long n, float *ave, float *var);
    float brent(float ax, float bx, float cx,
        float (*f)(float), float tol, float *xmin);
    float chixy(float bang);
    void fit(float x[], float y[], int ndata, float sig[], int mwt,
        float *a, float *b, float *siga, float *sigb, float *chi2, float *q);
    float gammq(float a, float x);
    void mnbrak(float *ax, float *bx, float *cx, float *fa, float *fb,
        float *fc, float (*func)(float));
    float zbrent(float (*func)(float), float x1, float x2, float tol);
    int j;
    float swap,amx,amn,varx,vary,ang[7],ch[7],scale,bmn,bmx,d1,d2,r2,
        dum1,dum2,dum3,dum4,dum5;

    xx=vector(1,ndat);
    yy=vector(1,ndat);
```

```
    sx=vector(1,ndat);
    sy=vector(1,ndat);
    ww=vector(1,ndat);
    avevar(x,ndat,&dum1,&varx);                    Find the x and y variances, and scale
    avevar(y,ndat,&dum1,&vary);                        the data into the global variables
    scale=sqrt(varx/vary);                             for communication with the func-
    nn=ndat;                                           tion chixy.
    for (j=1;j<=ndat;j++) {
        xx[j]=x[j];
        yy[j]=y[j]*scale;
        sx[j]=sigx[j];
        sy[j]=sigy[j]*scale;
        ww[j]=sqrt(SQR(sx[j])+SQR(sy[j]));         Use both x and y weights in first
    }                                                  trial fit.
    fit(xx,yy,nn,ww,1,&dum1,b,&dum2,&dum3,&dum4,&dum5);     Trial fit for b.
    offs=ang[1]=0.0;                               Construct several angles for refer-
    ang[2]=atan(*b);                                   ence points, and make b an an-
    ang[4]=0.0;                                        gle.
    ang[5]=ang[2];
    ang[6]=POTN;
    for (j=4;j<=6;j++) ch[j]=chixy(ang[j]);
    mnbrak(&ang[1],&ang[2],&ang[3],&ch[1],&ch[2],&ch[3],chixy);
    Bracket the χ2 minimum and then locate it with brent.
    *chi2=brent(ang[1],ang[2],ang[3],chixy,ACC,b);
    *chi2=chixy(*b);
    *a=aa;
    *q=gammq(0.5*(nn-2),*chi2*0.5);                Compute χ2 probability.
    for (r2=0.0,j=1;j<=nn;j++) r2 += ww[j];        Save the inverse sum of weights at
    r2=1.0/r2;                                         the minimum.
    bmx=BIG;                                       Now, find standard errors for b as
    bmn=BIG;                                           points where Δχ2 = 1.
    offs=(*chi2)+1.0;
    for (j=1;j<=6;j++) {                           Go through saved values to bracket
        if (ch[j] > offs) {                            the desired roots. Note period-
            d1=fabs(ang[j]-(*b));                      icity in slope angles.
            while (d1 >= PI) d1 -= PI;
            d2=PI-d1;
            if (ang[j] < *b) {
                swap=d1;
                d1=d2;
                d2=swap;
            }
            if (d1 < bmx) bmx=d1;
            if (d2 < bmn) bmn=d2;
        }
    }
    if (bmx < BIG) {                               Call zbrent to find the roots.
        bmx=zbrent(chixy,*b,*b+bmx,ACC)-(*b);
        amx=aa-(*a);
        bmn=zbrent(chixy,*b,*b-bmn,ACC)-(*b);
        amn=aa-(*a);
        *sigb=sqrt(0.5*(bmx*bmx+bmn*bmn))/(scale*SQR(cos(*b)));
        *siga=sqrt(0.5*(amx*amx+amn*amn)+r2)/scale;      Error in a has additional piece
    } else (*sigb)=(*siga)=BIG;                              r2.
    *a /= scale;                                   Unscale the answers.
    *b=tan(*b)/scale;
    free_vector(ww,1,ndat);
    free_vector(sy,1,ndat);
    free_vector(sx,1,ndat);
    free_vector(yy,1,ndat);
    free_vector(xx,1,ndat);
}
```

```
#include <math.h>
#include "nrutil.h"
#define BIG 1.0e30

extern int nn;
extern float *xx,*yy,*sx,*sy,*ww,aa,offs;

float chixy(float bang)
Captive function of fitexy, returns the value of (χ² − offs) for the slope b=tan(bang).
Scaled data and offs are communicated via the global variables.
{
    int j;
    float ans,avex=0.0,avey=0.0,sumw=0.0,b;

    b=tan(bang);
    for (j=1;j<=nn;j++) {
        ww[j] = SQR(b*sx[j])+SQR(sy[j]);
        sumw += (ww[j] = (ww[j] < 1.0/BIG ? BIG : 1.0/ww[j]));
        avex += ww[j]*xx[j];
        avey += ww[j]*yy[j];
    }
    avex /= sumw;
    avey /= sumw;
    aa=avey-b*avex;
    for (ans = -offs,j=1;j<=nn;j++)
        ans += ww[j]*SQR(yy[j]-aa-b*xx[j]);
    return ans;
}
```

Be aware that the literature on the seemingly straightforward subject of this section is generally confusing and sometimes plain wrong. Deming's [1] early treatment is sound, but its reliance on Taylor expansions gives inaccurate error estimates. References [2-4] are reliable, more recent, general treatments with critiques of earlier work. York [5] and Reed [6] usefully discuss the simple case of a straight line as treated here, but the latter paper has some errors, corrected in [7]. All this commotion has attracted the Bayesians [8-10], who have still different points of view.

CITED REFERENCES AND FURTHER READING:

Deming, W.E. 1943, *Statistical Adjustment of Data* (New York: Wiley), reprinted 1964 (New York: Dover). [1]

Jefferys, W.H. 1980, *Astronomical Journal*, vol. 85, pp. 177–181; see also vol. 95, p. 1299 (1988). [2]

Jefferys, W.H. 1981, *Astronomical Journal*, vol. 86, pp. 149–155; see also vol. 95, p. 1300 (1988). [3]

Lybanon, M. 1984, *American Journal of Physics*, vol. 52, pp. 22–26. [4]

York, D. 1966, *Canadian Journal of Physics*, vol. 44, pp. 1079–1086. [5]

Reed, B.C. 1989, *American Journal of Physics*, vol. 57, pp. 642–646; see also vol. 58, p. 189, and vol. 58, p. 1209. [6]

Reed, B.C. 1992, *American Journal of Physics*, vol. 60, pp. 59–62. [7]

Zellner, A. 1971, *An Introduction to Bayesian Inference in Econometrics* (New York: Wiley); reprinted 1987 (Malabar, FL: R. E. Krieger Pub. Co.). [8]

Gull, S.F. 1989, in *Maximum Entropy and Bayesian Methods*, J. Skilling, ed. (Boston: Kluwer). [9]

Jaynes, E.T. 1991, in *Maximum-Entropy and Bayesian Methods, Proc. 10th Int. Workshop*, W.T. Grandy, Jr., and L.H. Schick, eds. (Boston: Kluwer). [10]

Macdonald, J.R., and Thompson, W.J. 1992, *American Journal of Physics*, vol. 60, pp. 66–73.

15.4 General Linear Least Squares

An immediate generalization of §15.2 is to fit a set of data points (x_i, y_i) to a model that is not just a linear combination of 1 and x (namely $a + bx$), but rather a linear combination of *any* M specified functions of x. For example, the functions could be $1, x, x^2, \dots, x^{M-1}$, in which case their general linear combination,

$$y(x) = a_1 + a_2 x + a_3 x^2 + \cdots + a_M x^{M-1} \tag{15.4.1}$$

is a polynomial of degree $M - 1$. Or, the functions could be sines and cosines, in which case their general linear combination is a harmonic series.

The general form of this kind of model is

$$y(x) = \sum_{k=1}^{M} a_k X_k(x) \tag{15.4.2}$$

where $X_1(x), \dots, X_M(x)$ are arbitrary fixed functions of x, called the *basis functions*.

Note that the functions $X_k(x)$ can be wildly nonlinear functions of x. In this discussion "linear" refers only to the model's dependence on its *parameters* a_k.

For these linear models we generalize the discussion of the previous section by defining a merit function

$$\chi^2 = \sum_{i=1}^{N} \left[\frac{y_i - \sum_{k=1}^{M} a_k X_k(x_i)}{\sigma_i} \right]^2 \tag{15.4.3}$$

As before, σ_i is the measurement error (standard deviation) of the ith data point, presumed to be known. If the measurement errors are not known, they may all (as discussed at the end of §15.1) be set to the constant value $\sigma = 1$.

Once again, we will pick as best parameters those that minimize χ^2. There are several different techniques available for finding this minimum. Two are particularly useful, and we will discuss both in this section. To introduce them and elucidate their relationship, we need some notation.

Let **A** be a matrix whose $N \times M$ components are constructed from the M basis functions evaluated at the N abscissas x_i, and from the N measurement errors σ_i, by the prescription

$$A_{ij} = \frac{X_j(x_i)}{\sigma_i} \tag{15.4.4}$$

The matrix **A** is called the *design matrix* of the fitting problem. Notice that in general **A** has more rows than columns, $N \geq M$, since there must be more data points than model parameters to be solved for. (You can fit a straight line to two points, but not a very meaningful quintic!) The design matrix is shown schematically in Figure 15.4.1.

Also define a vector **b** of length N by

$$b_i = \frac{y_i}{\sigma_i} \tag{15.4.5}$$

and denote the M vector whose components are the parameters to be fitted, $a_1, \dots, a_M$, by **a**.