15.2 Fitting Data to a Straight Line

A concrete example will make the considerations of the previous section more meaningful. We consider the problem of fitting a set of $N$ data points $(x_i, y_i)$ to a straight-line model

$$y(x) = y(x; a, b) = a + bx$$  \hspace{1cm} (15.2.1)
Chapter 15. Modeling of Data

This problem is often called linear regression, a terminology that originated, long ago, in the social sciences. We assume that the uncertainty $\sigma_i$ associated with each measurement $y_i$ is known, and that the $x_i$'s (values of the dependent variable) are known exactly.

To measure how well the model agrees with the data, we use the chi-square merit function (15.1.5), which in this case is

$$\chi^2(a, b) = \sum_{i=1}^{N} \left( \frac{y_i - a - bx_i}{\sigma_i} \right)^2$$  \hspace{1cm} (15.2.2)

If the measurement errors are normally distributed, then this merit function will give maximum likelihood parameter estimations of $a$ and $b$; if the errors are not normally distributed, then the estimations are not maximum likelihood, but may still be useful in a practical sense. In §15.7, we will treat the case where outlier points are so numerous as to render the $\chi^2$ merit function useless.

Equation (15.2.2) is minimized to determine $a$ and $b$. At its minimum, derivatives of $\chi^2(a, b)$ with respect to $a, b$ vanish.

$$0 = \frac{\partial \chi^2}{\partial a} = -2 \sum_{i=1}^{N} \frac{y_i - a - bx_i}{\sigma_i^2}$$

$$0 = \frac{\partial \chi^2}{\partial b} = -2 \sum_{i=1}^{N} x_i (y_i - a - bx_i)$$  \hspace{1cm} (15.2.3)

These conditions can be rewritten in a convenient form if we define the following sums:

$$S \equiv \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \quad S_x \equiv \sum_{i=1}^{N} \frac{x_i}{\sigma_i^2} \quad S_y \equiv \sum_{i=1}^{N} \frac{y_i}{\sigma_i^2}$$

$$S_{xx} \equiv \sum_{i=1}^{N} \frac{x_i^2}{\sigma_i^2} \quad S_{xy} \equiv \sum_{i=1}^{N} \frac{x_i y_i}{\sigma_i^2}$$  \hspace{1cm} (15.2.4)

With these definitions (15.2.3) becomes

$$a S + b S_x = S_y$$

$$a S_x + b S_{xx} = S_{xy}$$  \hspace{1cm} (15.2.5)

The solution of these two equations in two unknowns is calculated as

$$\Delta = SS_{xx} - (S_x)^2$$

$$a = \frac{S_{xx} S_y - S_x S_{xy}}{\Delta}$$

$$b = \frac{S S_{xy} - S_x S_y}{\Delta}$$  \hspace{1cm} (15.2.6)

Equation (15.2.6) gives the solution for the best-fit model parameters $a$ and $b$. 
We are not done, however. We must estimate the probable uncertainties in
the estimates of \( a \) and \( b \), since obviously the measurement errors in the data must
introduce some uncertainty in the determination of those parameters. If the data
are independent, then each contributes its own bit of uncertainty to the parameters.
Consideration of propagation of errors shows that the variance \( \sigma_j^2 \) in the value of
any function will be

\[
\sigma_j^2 = \sum_{i=1}^{N} \sigma_i^2 \left( \frac{\partial f}{\partial y_i} \right)^2
\]  

(15.2.7)

For the straight line, the derivatives of \( a \) and \( b \) with respect to \( y_i \) can be directly
evaluated from the solution:

\[
\frac{\partial a}{\partial y_i} = \frac{S_{zz} - S_x x_i}{\sigma_i^2 \Delta} \\
\frac{\partial b}{\partial y_i} = \frac{S_x i - S_z}{\sigma_i^2 \Delta}
\]  

(15.2.8)

Summing over the points as in (15.2.7), we get

\[
\sigma_a^2 = \frac{S_{zz}}{\Delta} \\
\sigma_b^2 = \frac{S}{\Delta}
\]  

(15.2.9)

which are the variances in the estimates of \( a \) and \( b \), respectively. We will see in
§15.6 that an additional number is also needed to characterize properly the probable
uncertainty of the parameter estimation. That number is the covariance of \( a \) and \( b \),
and (as we will see below) is given by

\[
\text{Cov}(a, b) = -\frac{S_x}{\Delta}
\]  

(15.2.10)

The coefficient of correlation between the uncertainty in \( a \) and the uncertainty
in \( b \), which is a number between \(-1\) and \(1\), follows from (15.2.10) (compare
equation 14.5.1),

\[
r_{ab} = -\frac{S_x}{\sqrt{S_{zz}}}
\]  

(15.2.11)

A positive value of \( r_{ab} \) indicates that the errors in \( a \) and \( b \) are likely to have the
same sign, while a negative value indicates the errors are anticorrelated, likely to
have opposite signs.

We are still not done. We must estimate the goodness-of-fit of the data to the
model. Absent this estimate, we have not the slightest indication that the parameters
\( a \) and \( b \) in the model have any meaning at all! The probability \( Q \) that a value of
chi-square as poor as the value (15.2.2) should occur by chance is

\[
Q = \text{gammq} \left( \frac{N - 2}{2}, \frac{\chi^2}{2} \right)
\]  

(15.2.12)
Here gamma is our routine for the incomplete gamma function \(Q(a, x)\), §6.2. If \(Q\) is larger than, say, 0.1, then the goodness-of-fit is believable. If it is larger than, say, 0.001, then the fit may be acceptable if the errors are nonnormal or have been moderately underestimated. If \(Q\) is less than 0.001 then the model and/or estimation procedure can rightly be called into question. In this latter case, turn to §15.7 to proceed further.

If you do not know the individual measurement errors of the points \(\sigma_i\), and are proceeding (dangerously) to use equation (15.1.6) for estimating these errors, then here is the procedure for estimating the probable uncertainties of the parameters \(a\) and \(b\): Set \(\sigma_i \equiv 1\) in all equations through (15.2.6), and multiply \(\sigma_a\) and \(\sigma_b\), as obtained from equation (15.2.9), by the additional factor \(\sqrt{\chi^2/(N-2)}\), where \(\chi^2\) is computed by (15.2.2) using the fitted parameters \(a\) and \(b\). As discussed above, this procedure is equivalent to assuming a good fit, so you get no independent goodness-of-fit probability \(Q\).

In §14.5 we promised a relation between the linear correlation coefficient \(r\) (equation 14.5.1) and a goodness-of-fit measure, \(\chi^2\) (equation 15.2.2). For unweighted data (all \(\sigma_i = 1\)), that relation is

\[
\chi^2 = (1 - r^2) \text{NVar}(y_1 \ldots y_N)
\]

(15.2.13)

where

\[
\text{NVar}(y_1 \ldots y_N) \equiv \sum_{i=1}^{N} (y_i - \bar{y})^2
\]

(15.2.14)

For data with varying weights \(\sigma_i\), the above equations remain valid if the sums in equation (14.5.1) are weighted by \(1/\sigma_i^2\).

The following function, \textit{fit}, carries out exactly the operations that we have discussed. When the weights \(\sigma\) are known in advance, the calculations exactly correspond to the formulas above. However, when weights \(\sigma\) are unavailable, the routine assumes equal values of \(\sigma\) for each point and assumes a good fit, as discussed in §15.1.

The formulas (15.2.6) are susceptible to roundoff error. Accordingly, we rewrite them as follows: Define

\[
t_i = \frac{1}{\sigma_i} (x_i - \frac{S_x}{S}) , \quad i = 1, 2, \ldots, N
\]

(15.2.15)

and

\[
S_{tt} = \sum_{i=1}^{N} t_i^2
\]

(15.2.16)

Then, as you can verify by direct substitution,

\[
b = \frac{1}{S_{tt}} \sum_{i=1}^{N} t_i y_i / \sigma_i
\]

(15.2.17)

and

\[
a = \frac{S_y - S_x b}{S}
\]

(15.2.18)
\[ \sigma_a^2 = \frac{1}{S} \left( 1 + \frac{S_y^2}{SS_{tt}} \right) \]  
(15.2.19)

\[ \sigma_b^2 = \frac{1}{S_{tt}} \]  
(15.2.20)

\[ \text{Cov}(a, b) = -\frac{S_y}{SS_{tt}} \]  
(15.2.21)

\[ r_{ab} = \frac{\text{Cov}(a, b)}{\sigma_a \sigma_b} \]  
(15.2.22)

```c
#include <math.h>
#include "nrutil.h"

void fit(float x[], float y[], int ndata, float sig[], int nwt, float *a, float *b, float *sig_a, float *sig_b, float *chisq, float *q) {
  
  Given a set of data points x[1..ndata], y[1..ndata] with individual standard deviations sig[1..ndata], fit them to a straight line \( y = a + bx \) by minimizing \( \chi^2 \). Returned are \( a, b \) and their respective probable uncertainties \( \sigma_a \) and \( \sigma_b \), the chi-square \( \text{chisq} \), and the goodness-of-fit probability \( q \) (that the fit would have \( \chi^2 \) this large or larger). If \( nwt=0 \) on input, then the standard deviations are assumed to be unavailable: \( q \) is returned as 1.0 and the normalization of \( \text{chisq} \) is to unit standard deviation on all points.

  
  float gamma(float a, float x); 
  int i; 
  float wt, sx, sy, ssx, ssy, ss, sigdat;

  float *w = (float *) malloc(sizeof(float) * nwt);
  float wtsum = 0.0;
  float sxsum = 0.0;
  float sysum = 0.0;

  for (i=1;i<=nndata;i++) {
    sx += x[i];
    sy += y[i];
  }
  sxsum = sx;
  sysum = sy;

  ss = sxsum * sxsum;
  if (nwt) {
    for (i=1;i<=nndata;i++) {
      wt = 1.0 / sqrt(sig[i]);
      sx += wt * x[i];
      sy += wt * y[i];
    }
    wtsum = sxsum;
  } else {
    for (i=1;i<=nndata;i++) {
      sx += x[i];
      sy += y[i];
    }
  }
  
  sxsum = sxsum - ssx;
  sysum = sysum - ssy;

  if (nwt) {
    for (i=1;i<=nndata;i++) {
      t = x[i] - sxsum / sig[i];
      st2 += t * t;
      *b += t * y[i] / sig[i];
    }
  } else {
    for (i=1;i<=nndata;i++) {
      t = x[i] - sxsum;
      st2 += t * t;
      *b += t * y[i];
    }
  }
  
  *b /= st2;
  *ax = (sy - sx * (*b)) / ss;
  *sig_a = sqrt(((1.0 + sx * sx / (ss * st2)) / ss));
  *sig_b = sqrt(1.0 / st2);

  Solve for \( a, b, \sigma_a, \) and \( \sigma_b \).

  Accumulate sums ... 
  ...with weights
  ...or without weights.
  
  \( \text{chisq} \) and \( q \) are returned.
}
```
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 \(1, x, x^2, \ldots, x^{M-1}\), in which case their general linear combination,

\[
y(x) = a_1 + a_2 x + a_3 x^2 + \cdots + a_M x^{M-1}
\]

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

\[
y(x) = \sum_{k=1}^{M} a_k X_k(x)
\]

(15.4.2)

where \(X_1(x), \ldots, X_M(x)\) 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

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

(15.4.3)

As before, \(\sigma_i\) is the measurement error (standard deviation) of the \(i\)th 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 \(\chi^2\). 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 \(\sigma_i\), by the prescription

\[
A_{ij} = \frac{X_j(x_i)}{\sigma_i}
\]

(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

\[
b_i = \frac{y_i}{\sigma_i}
\]

(15.4.5)

and denote the \(M\) vector whose components are the parameters to be fitted, \(a_1, \ldots, a_M\), by \(a\).
Figure 15.4.1. Design matrix for the least-squares fit of a linear combination of $M$ basis functions to $N$ data points. The matrix elements involve the basis functions evaluated at the values of the independent variable at which measurements are made, and the standard deviations of the measured dependent variable. The measured values of the dependent variable do not enter the design matrix.

**Solution by Use of the Normal Equations**

The minimum of (15.4.3) occurs where the derivative of $\chi^2$ with respect to all $M$ parameters $a_k$ vanishes. Specializing equation (15.1.7) to the case of the model (15.4.2), this condition yields the $M$ equations

$$0 = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[ y_i - \sum_{j=1}^{M} a_j X_j(x_i) \right] X_k(x_i) \quad k = 1, \ldots, M \quad (15.4.6)$$

Interchanging the order of summations, we can write (15.4.6) as the matrix equation

$$\sum_{j=1}^{M} \alpha_{kj} a_j = \beta_k \quad (15.4.7)$$

where

$$\alpha_{kj} = \sum_{i=1}^{N} \frac{X_j(x_i) X_k(x_i)}{\sigma_i^2}$$

or equivalently

$$[\alpha] = A^T \cdot A \quad (15.4.8)$$

an $M \times M$ matrix, and

$$\beta_k = \sum_{i=1}^{N} \frac{y_i X_k(x_i)}{\sigma_i^2}$$

or equivalently

$$[\beta] = A^T \cdot b \quad (15.4.9)$$
a vector of length \( M \).

The equations (15.4.6) or (15.4.7) are called the normal equations of the least-squares problem. They can be solved for the vector of parameters \( \mathbf{a} \) by the standard methods of Chapter 2, notably LU decomposition and backsubstitution, Choleksy decomposition, or Gauss-Jordan elimination. In matrix form, the normal equations can be written as either

\[
[\alpha] \cdot \mathbf{a} = [\beta] \quad \text{or as} \quad (\mathbf{A}^T \cdot \mathbf{A}) \cdot \mathbf{a} = \mathbf{A}^T \cdot \mathbf{b}
\]

(15.4.10)

The inverse matrix \( C_{jk} = [\alpha]^{-1}_{jk} \) is closely related to the probable (or, more precisely, standard) uncertainties of the estimated parameters \( \mathbf{a} \). To estimate these uncertainties, consider that

\[
a_j = \sum_{k=1}^{M} [\alpha]_{jk}^{-1} \beta_k = \sum_{k=1}^{M} C_{jk} \left[ \sum_{i=1}^{N} \frac{y_i X_k(x_i)}{\sigma_i^2} \right]
\]

(15.4.11)

and that the variance associated with the estimate \( a_j \) can be found as in (15.2.7) from

\[
\sigma^2(a_j) = \sum_{i=1}^{N} \sigma_i^2 \left( \frac{\partial a_j}{\partial y_i} \right)^2
\]

(15.4.12)

Note that \( \alpha_{jk} \) is independent of \( y_i \), so that

\[
\frac{\partial a_j}{\partial y_i} = \sum_{k=1}^{M} C_{jk} X_k(x_i) / \sigma_i^2
\]

(15.4.13)

Consequently, we find that

\[
\sigma^2(a_j) = \sum_{k=1}^{M} \sum_{l=1}^{M} C_{jk} C_{jl} \left[ \sum_{i=1}^{N} \frac{X_k(x_i) X_l(x_i)}{\sigma_i^2} \right]
\]

(15.4.14)

The final term in brackets is just the matrix \([\alpha]\). Since this is the matrix inverse of \([C]\), (15.4.14) reduces immediately to

\[
\sigma^2(a_j) = C_{jj}
\]

(15.4.15)

In other words, the diagonal elements of \([C]\) are the variances (squared uncertainties) of the fitted parameters \( \mathbf{a} \). It should not surprise you to learn that the off-diagonal elements \( C_{jk} \) are the covariances between \( a_j \) and \( a_k \) (cf. 15.2.10); but we shall defer discussion of these to \S 15.6.

We will now give a routine that implements the above formulas for the general linear least-squares problem, by the method of normal equations. Since we wish to compute not only the solution vector \( \mathbf{a} \) but also the covariance matrix \([C]\), it is most convenient to use Gauss-Jordan elimination (routine gaussj of \S 2.1) to perform the linear algebra. The operation count, in this application, is no larger than that for \( LU \) decomposition. If you have no need for the covariance matrix, however, you can save a factor of 3 on the linear algebra by switching to \( LU \) decomposition, without
computation of the matrix inverse. In theory, since \( A^T \cdot A \) is positive definite, Cholesky decomposition is the most efficient way to solve the normal equations. However, in practice most of the computing time is spent in looping over the data to form the equations, and Gauss-Jordan is quite adequate.

We need to warn you that the solution of a least-squares problem directly from the normal equations is rather susceptible to roundoff error. An alternative, and preferred, technique involves QR decomposition (§2.10, §11.3, and §11.6) of the design matrix \( A \). This is essentially what we did at the end of §15.2 for fitting data to a straight line, but without invoking all the machinery of QR to derive the necessary formulas. Later in this section, we will discuss other difficulties in the least-squares problem, for which the cure is singular value decomposition (SVD), of which we give an implementation. It turns out that SVD also fixes the roundoff problem, so it is our recommended technique for all but "easy" least-squares problems. It is for these easy problems that the following routine, which solves the normal equations, is intended.

The routine below introduces one bookkeeping trick that is quite useful in practical work. Frequently it is a matter of "art" to decide which parameters \( a_k \) in a model should be fit from the data set, and which should be held constant at fixed values, for example values predicted by a theory or measured in a previous experiment. One wants, therefore, to have a convenient means for "freezing" and "unfreezing" the parameters \( a_k \). In the following routine the total number of parameters \( a_k \) is denoted \( m \) (called \( M \) above). As input to the routine, you supply an array \( a[1..ma] \), whose components are either zero or nonzero (e.g., 1). Zeros indicate that you want the corresponding elements of the parameter vector \( a[1..ma] \) to be held fixed at their input values. Nonzeros indicate parameters that should be fitted for. On output, any frozen parameters will have their variances, and all their covariances, set to zero in the covariance matrix.

```c
#include "nrutil.h"

void lfit(float x[], float y[], float sig[], int ndat, float a[], int ia[],
          int ma, float **covar, float *chiq, void (*func)(float, float [], int))
  Given a set of data points \( x[1..ndat], y[1..ndat] \) with individual standard deviations \( sig[1..ndat] \), use \( \chi^2 \) minimization to fit for some or all of the coefficients \( a[1..ma] \) of a function that depends linearly on \( a, y = \sum a_i x^i \). The input array \( ia[1..ma] \) indicates by nonzero entries those components of \( a \) that should be fitted for, and by zero entries those components that should be held fixed at their input values. The program returns values for \( a[1..ma], \chi^2 = \text{chiq}, \text{and the covariance matrix covar[1..ma][1..ma].} \) (Parameters held fixed will return zero covariances.) The user supplies a routine \( \text{func}(x,a[1..ma]) \) that returns the \( ma \) basis functions evaluated at \( x = x \) in the array \( a[1..ma] \).

  void convert(float **covar, int ma, int ia[], int mfit);
  void gaussj(float **a, int n, float **b, int m);
  int i,j,k,l,m,mfit=0;
  float zm,wt,sum,sig21, *beta,*afunc;

  beta=matrix(1,ma,1,1);
  afunc=vector(1,ma);
  for (j=1;j<ma;j++)
    if (ia[j]) mfit++;
  if (mfit == 0) nrerror("lfit: no parameters to be fitted");
  for (j=1;j<mfit;j++)
    initialize the (symmetric) matrix.
    for (k=1;k<mfit;k++)
      covar[j][k]=0.0;
      beta[j][j]=0.0;
  for (i=1;i<ndat;i++)
    Loop over data to accumulate coefficients of
    normal equations.
```
(*func) (x[i], afunc, ma);

ym=y[i];
if (mfit < ma) {
  for (j=1; j<ma; j++)
    if (ia[j]) ym -= a[j]*afunc[j];
}

sig2i=1.0/SQR(sig[i]);
for (j=0, l=1; l<ma; l++) {
  if (ia[l])
    wt=afunc[l]*sig2i;
  for (j++, k=0, m=1; m<l; m++)
    if (ia[m]) covar[j][l+k] += wt*afunc[m];
  beta[j][l] += ym*wt;
}
}

for (j=2; j<mfit+1; j++)
  for (k=1; k<j; k++)
    covar[k][j]=covar[j][k];

gaussj(covar, mfit, beta[i]);
for (j=0, l=1; l<ma; l++)
  if (ia[l]) a[l]=beta[i+j][l];
*chisq=0.0;
for (i=1; i<ndat+1; i++)
  (*func) (x[i], afunc, ma);
  for (sum=0.0, j=1; j<ma; j++) sum += a[j]*afunc[j];
*chisq += SQN((y[i]-sum)/sig[i]);

covr(covar, ma, ia, mfit);
free_vector(afunc, ma);
frees_matrix(beta, ma, 1, 1);"