MULTIPLE SITE BINDING

Gary D. Knott, Ph.D.

Civilized Software Inc.
12109 Heritage Park Circle
Silver Spring, MD 20906 USA
Tel: (301)962-3711
Email: csi@civilized.com
URL: http: //www.civilized.com

Introduction

The study of how a ligand material, such as a hormone or antibody, binds to one or more kinds of molecular complexes, called sites, is of fundamental importance in biochemistry. Sites are often embedded in cell membranes, and the binding serves to control the behavior of the cell itself. Typically we are interested in the number of distinct kinds of sites and their frequency of occurrence, and also the equilibrium constants for each ligand-site binding reaction which indicates the absolute strength of each such binding reaction.

Quantitative analysis of hormone-receptor binding data can be performed using appropriate software such as MLAB. MLAB is a computer program whose name is an acronym for "modeling laboratory"; it is an interactive system for mathematical modeling, originally developed at the National Institutes of Health. Detailed information about MLAB is available at the web-site *www.civilized.com*. MLAB can fit multiple non-linear models to data points obtained from standard direct-binding or competitive displacement assays. Typical assays involve measuring the competition between radiolabelled and cold ligand in detergent-solubilized membrane preparations or on whole cells. Affinity constants and limit values of binding protein concentrations for single or multiple sites can be computed by fitting saturation curves in MLAB. Output can include Scatchard plots obtained by a suitable transformation.

In addition to ligand binding analyses, MLAB can be used in many other curve fitting applications such as kinetic modeling, Michaelis-Menten

enzyme kinetics, ultracentrifuge data analysis, and various statistical analyses. Kinetic analysis involves fitting differential equation models, which is an important core capability of MLAB.

The MLAB Mathematical Modeling System

MLAB has hundreds of useful functions, e.g., the discrete Fourier transform function `dft` and the parametric spline interpolation function `splinep`. One of the central components of the system is a curve fitting program which will adjust the parameters of a model function to minimize the weighted sum of the errors raised to a specified power. A repertoire of mathematical operators and functions, routines for solving differential equations, a collection of routines for onscreen drawing and for hardcopy plotting, and mechanisms for saving data between sessions provide a powerful and convenient environment for data manipulation, arithmetic calculations, and for building and testing models.

One of MLAB's main uses is to fit models to data via its curve-fitting facilities. Curve-fitting is a useful analytical tool in many diverse disciplines. The basic notion is easily described. Given data, say various points in the plane (x_1, y_1), (x_2, y_2), ..., (x_n, y_n), and a function $y = f(x)$ where f involves some parameters, say a and b, as for example $f(x) = ax^b + 1$, we may wish to calculate values for the parameters a and b so that the function f well-predicts the observed data, that is, so that $f(x_i) = y_i$ for $1 \leq i \leq n$. In this case, we say we have fit the model f to the data by estimating the parameters a and b. The end result is merely the values obtained for the initially unknown parameters. The same principles apply in higher dimensions with arbitrarily many parameters. MLAB can simultaneously fit multiple non-linear model functions, some or all of which may be implicit functions, or may even be defined by a system of differential equations.

The curve-fitting and graphics display facilities of MLAB make it an ideal tool for the estimation of equilibrium constants from ligand binding data, which typically consist of observed amounts of ligand bound for various specified amounts of ligand provided for binding.

The Mathematics of Multiple Site Binding

Suppose we have a ligand, F, which binds to each of N *independently-acting* sites, S_1, S_2, ..., S_N, which are present in the various concentrations of $S_{10}\,\mu$M, $S_{20}\,\mu$M, ..., and $S_{N0}\,\mu$M respectively. Each reaction $F + S_i \rightleftharpoons B_i$ forms a bound complex B_i, and we shall define κ_i to be the associated equilibrium constant. Let F_0 denote the concentration in μM of ligand

present initially, and let F be the concentration in μM of free ligand at equilibrium. Similarly, let S_i denote the concentration in μM of free sites of type i at equilibrium, and let B_i denote the concentration of bound site i complex at equilibrium. Note we use the symbols F, S_i, and B_i for the numerical quantities in μM (micromolar) units of these materials, as well as for the names of the materials themselves. Then:

$$\begin{aligned} \kappa_i &= B_i/(FS_i) \quad \text{and} \quad B_i + S_i = S_{i0} \quad \text{for} \quad 1 \le i \le N, \quad \text{and} \\ F &= F_0 - B_1 - B_2 - \ldots - B_N. \end{aligned}$$

Note that κ_i is specified in liters/μMole units (which is $1/\mu$M units), so that $10^6\kappa_i$ is the corresponding equilibrium constant in liters/Mole units. Sometimes κ_i is called the *equilibrium association* constant, in contrast to the value $1/\kappa_i$ which is called the *equilibrium dissociation* constant. The unit of measurment for material amounts of F, S_i, B_i, S_{i0} and F_0 may be chosen to be any convenient unit convertable to μM units, such as *cpm* (counts per minute.) In general, the equilibrium constant κ_i will be specified in the reciprocal of that unit. (Of course, an amount of non-labeled material measured in *cpm* units is a fiction to be interpreted *as if* it were labeled.)

The fraction of S_i-sites occupied by ligand molecules is $r := F/(F+1/\kappa_i)$. When $F \ll 1/\kappa_i$, the occupancy fraction r is small, and when $F \gg 1/\kappa_i$, r is nearly 1; when $F = 1/\kappa_i$, fifty percent of the S_i-sites are occupied.

Often there is a fictitious $(N+1)$-st site, X, which binds F molecules, which is introduced to describe the *non-specific* binding of F molecules with weak affinity to many locations, other than the N specific sites of interest. Let κ_0 be the equilibrium constant for the fictitious non-specific binding reaction $F + X \rightleftharpoons Y$. Then we have:

$$\begin{aligned} \kappa_i &= B_i/(FS_i) \quad \text{and} \quad B_i + S_i = S_{i0} \quad \text{for} \quad 1 \le i \le N, \\ \kappa_0 &= Y/(FX) \quad \text{and} \quad Y + X = X_0, \quad \text{and} \\ F &= F_0 - B_1 - B_2 - \ldots - B_N - Y, \end{aligned}$$

where Y is the concentration of non-specifically bound ligand, X_0 is the concentration of the fictitious non-specific binding site, and X is the concentration of the free fictitious site at equilibrium. Then $B_i = \kappa_i S_{i0} F/(1 + \kappa_i F)$ for $1 \le i \le N$, and $Y = \kappa_0 X_0 F/(1 + \kappa_0 F)$.

Now, to capture the notion of non-specific binding as a weak sticking of F molecules almost everywhere, let κ_0 tend to zero and let X_0 tend to infinity such that $\kappa_0 X_0 = c$, where c is a fixed constant. Then $Y \to cF$, and we have:

$$B_i = \kappa_i S_{i0} F/(1+\kappa_i F) \quad \text{for} \quad 1 \le i \le N, \quad Y = cF, \quad \text{and} \quad F = F_0 - B_1 - \ldots - B_N - Y.$$

Note that the mathematical form $B_i = S_{i0}F/(1/\kappa_i + F)$ is difficult to handle when κ_i approaches 0, so that the form given above is more convenient.

Usually values of F (free ligand) and/or $B_1 + B_2 + \ldots + B_N + Y$ (total bound ligand) are measured for different values of F_0 and we wish to use this data, which is generally of the form (F_0, F) or $(F_0, B_1 + \ldots + B_N + Y)$, to estimate κ_1, κ_2, ..., κ_N, c, and, if not already known, S_{10}, S_{20}, ..., S_{N0}. When only the total bound concentration $B_1 + B_2 + \ldots + B_N + Y$ is measured, rather than B_1, B_2, ..., B_N and Y separately, only a few sites can be distinguished by curve-fitting. In order to determine how many kinds of sites appear to be present, we must try each of the 1-site, 2-site, etc. models and choose among them on the basis of how well the data is fit. Note if the equilibrium constants obtained for two species of sites are close, then there are no grounds for considering them to be distinct species, based on this analysis alone.

Often people use the model $B(F) = \kappa_1 S_{10}F/(1+\kappa_1 F)+\cdots+\kappa_N S_{N0}F/(1+\kappa_N F) + cF$ and fit it to data points of the form (F, B) where F is the free ligand concentration at equilibrium and B is the total bound ligand concentration at equilibrium (measured with error), with one such point for each experiment. *The difficulty with this approach is that we must compute F as $F_0 - B$ so that there is correlated error in* **both** *the dependent and independent values used to form the data points.* Our approach uses data points of the form (F_0, B), and since F_0 can generally be accurately determined, we avoid the aforementioned difficulty of error in the independent variable.

As an explicit example of the multiple-site binding model developed above, the general two-site model with non-specific binding for a single ligand type can be defined in MLAB with the following dialog. Here and hereafter, the text shown following the MLAB prompt asterisk is an MLAB command statement entered by the user. (In practice, an MLAB do-file would be used to permit repetitive use when desired.)

```
* FUNCTION B(F)  = k1*S10*F/(1+k1*F) + k2*S20*F/(1+k2*F)
* FUNCTION F(F0) = ROOT(Z,0,F0,F0-B(Z)-Z*(1+c))
* FUNCTION Y(F0) = c*F(F0)
```

Here $A(F_0) = B_1 + \ldots + B_N$ for $N = 2$.

These commands exemplify the MLAB `FUNCTION` statement, which is used to define a function or differential equation. Note that arguments of functions must be explicitly specified. Variables, such as `k1` and `k2`, which appear in the body of a function, but not in its argument list, are

called parameters. Parameters must be assigned values before an associated function can be evaluated.

`ROOT` is an operator which is built-in in MLAB. `ROOT(Z,A,B,E)` is a value between `A` and `B` which, when taken as the value of the dummy variable, `Z`, makes the expression, `E`, which involves `Z`, equal to zero. Thus `ROOT(Z,A,B,E)` is a solution of $E(Z) = 0$. The model given above, involving a so-called *implicit* function, deserves careful study; it is easily extendible to more than two sites. The amount F of free-ligand at equilibrium as a function of F_0 satisfies $F_0 - F = B(F) + c \cdot F$.

An Example

Suppose we have measured F in μM units as a function of F_0 in μM units, as follows:

F_0	F
.58668	.036
1.1734	.096
2.3467	.385
2.9334	.61
4.1068	1.15
4.6934	1.46
5.8668	2.11
7.0402	2.73

and we have $S_{10} = S_{20} = 1.7121\,\mu$M. (If S_{10} and S_{20} were unknown, we could include them as fitting parameters.)

Then, we can introduce the appropriate constraints (which should always be used for this particular model), guess K_1, K_2, and c and then estimate them, as follows.

```
* constraints q ={k1>=0,k2>=0,c>=0,S10>=0,S20>=0}
```

The `CONSTRAINTS` statement permits the user to specify successive linear inequalities or equations involving the parameters (or potential parameters). Now we may specify values for the parameters, guessing when necessary.

```
* k1 = 10; k2 = 1; c = 0; S10 = 1.7121; S20 = S10;
```

Here the `ASSIGNMENT` statement is exemplified. In this case all the above assignments are assigning values to scalar variables, but the assignment statement is used to assign values to matrices as well. This can be seen in the next assignment statement which defines a matrix, D.

```
* D = Kread(8,2)
.58668 .036
1.1734 .096
2.3467 .385
2.9334 .61
4.1068 1.15
4.6934 1.46
5.8668 2.11
7.0402 2.73
```

The KREAD operator takes optional array size arguments; in this case an 8 row by 2 column matrix is specified, and reads in numbers from the keyboard to construct an appropriately-dimensioned matrix as the result. An analogous function is used to read numbers from a file, which is preferred manner in practice. This matrix is then, in this case, assigned to D. Note the entry of the numbers which follow. Now we may examine D by "typing it out" using the TYPE statement.

```
* type D
D:  8 by 2 matrix
1: .58668 .36E1
2: 1.1734 .96E-1
3: 2.3467 .385
4: 2.9334 .61
5: 4.1068 1.15
6: 4.6934 1.46
7: 5.8668 2.11
8: 7.0402 2.73
```

We have established a model function, F, and entered data, D. We expect that $f(D[i,1]) \approx D[i,2]$ would hold, if only the parameters κ_1, κ_2, and c were set to their "correct" values. The following FIT statement requests MLAB to estimate κ_1, κ_2, and c by assigning them values which minimize the sum-of-squares objective function $S(\kappa_1, \kappa_2, c) = \sum_{i=1}^{8} (F(D[i,1]) - D[i,2])^2$.

```
* maxiter = 30; TOLSOS = .001
* fit(k1,k2,c), F to d, constraints q
final parameter values
value           error           dependency      parameter
```

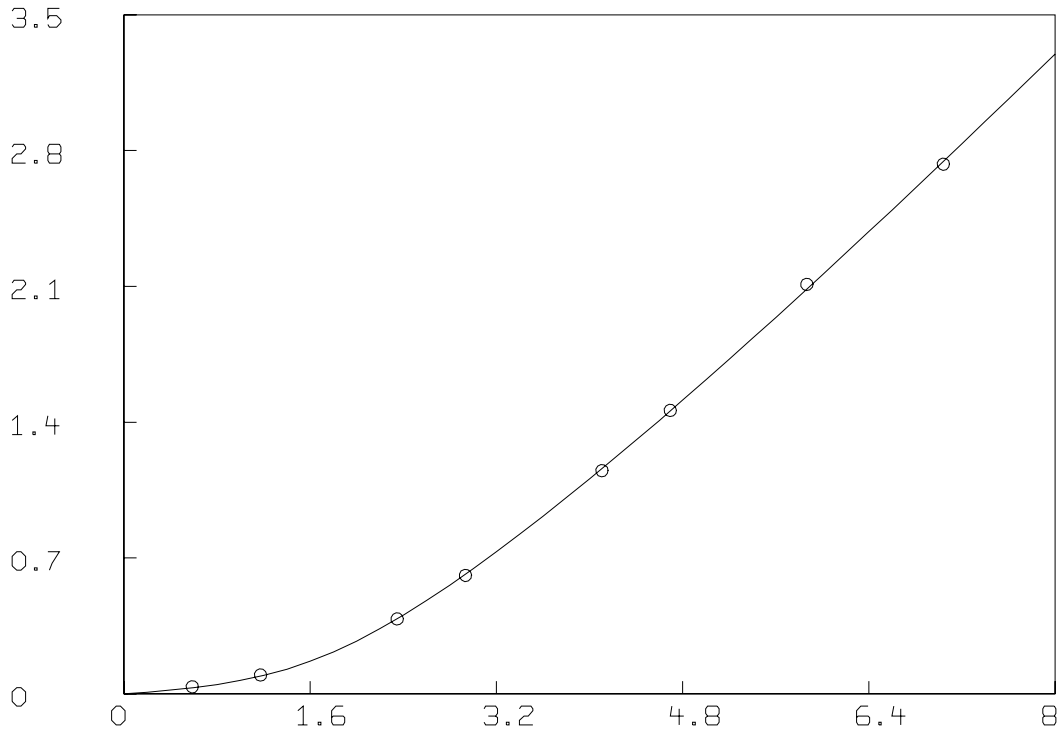
```
13.86352736     2.331373145     0.665073064     K1
0.5321372226    0.06602915638   0.9432600296    K2
0.5874015019    0.02229743988   0.9171690302    C
5 iterations
CONVERGED
best sum of squares = 9.78763e-04
root mean square error = 1.39912e-02
deviation fraction = 6.82654e-03
R squared = 9.99854e-01
no active constraints
```

The behavior of the fit statement depends upon the supplied constraints q, as well as upon the MLAB control variables: maxiter, the maximum number of iterations and tolsos, the requested convergence factor. Many multiple-site binding models are sensitive to the initial guesses used, and care must be taken to obtain a physically-meaningful fit. Trial and error fitting may be required to find the necessary guesses.

The material typed out above shows that the vector of parameters $(\mathrm{K}_1, \mathrm{K}_2, c)$ has been estimated to be $(13.8633 \pm 2.3315, .532151 \pm .0660296, .587396 \pm .0222973)$, with reasonably small dependency values, and with an RMS error of about .014, which should be comparable with the experimental error in our data, D. The sum-of-squares was reduced from an initial value of 2.64314 to .000978763 at the final parameter values. The standard-errors of the parameters are normal-theory estimates and are often unreliable, although they are of suggestive value.

In order to visually see how our model with its parameters set to their best-estimated values corresponds to the data, we may draw a graph of the data points and the model function. Although we are drawing only the simplest and most direct kind of picture here, it should be noted that MLAB provides facilities for many types of point-symbols and types of lines, axes with arbitrarily-placed numeric labels in various formats, titles in the form of text strings in arbitrary sizes and various fonts with subscripts and superscripts, color, and a number of other special features. It is quite possible to prepare more or less elaborate publication-quality graphs with a modest amount of effort. Indeed this is one of MLAB's most-used facilities. The desired graph can be constructed as follows.

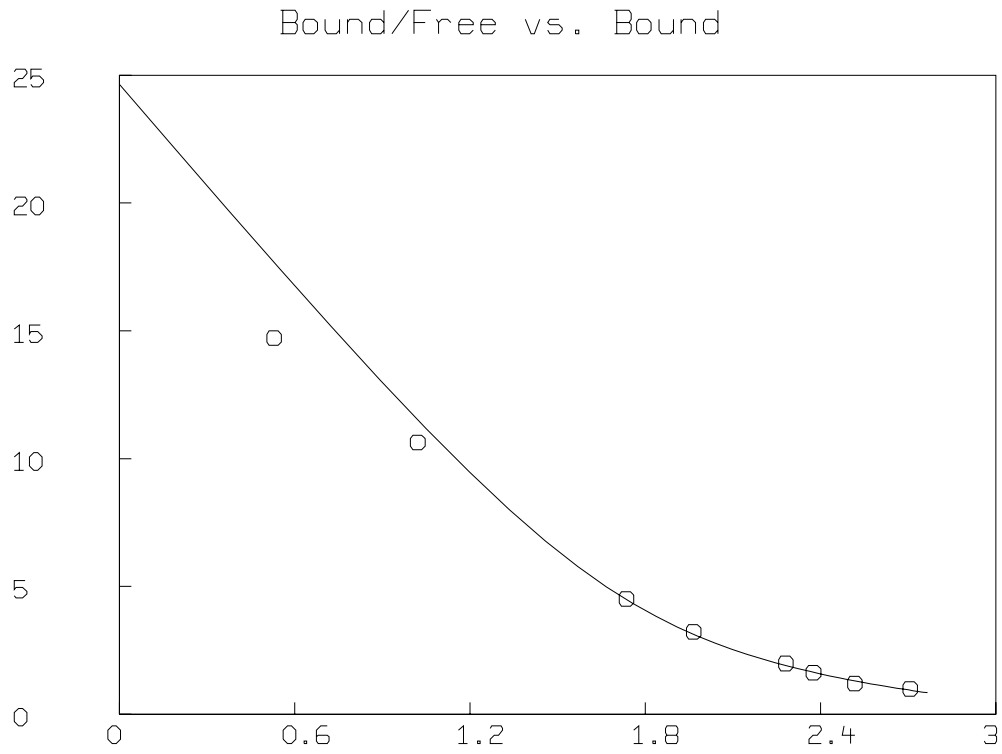
```
* draw D, linetype none,pointtype circle
* draw c2 = points(F,0:8:.2)
* VIEW
```



The first `DRAW` command above plots the data points, while the second draw command constructs a curve called `C2`, which is a graph consisting of solid straight-lines connecting the points which are the rows of the 2-column matrix which is the value of the expression `POINTS(F,0:8:.2)`. This matrix has the values 0 through 8 in steps of .2 in its first column and corresponding values of the function F evaluated at 0 through 8 in steps of .2 in the second column. The `POINTS` operator is very useful for graphing functions. Both these curves are drawn in the default MLAB 2D-window called `W` (since no other window is specified) which has predefined labeled axes already present. The picture finally appears when its display is requested with a view statement and a plot can be obtained if desired using the `PLOT` statement.

Often a Scatchard graph of Bound/Free vs. Bound (*i.e.*, $B(F(F_0))/F(F_0)$ vs. $B(F(F_0))$) is desired. Although such a formulation should *not* be used for curve-fitting due to the non-normal error introduced by computing Bound/Free, it is quite straightforward to draw the data and model Scatchard plots as follows. Note the current picture in `W` is saved and restored.

```
* SAVE W IN GW
* DELETE W
* FUNCTION R(X,Y)=IF Y=0 THEN (k1*s10+k2*s20) ELSE X/Y
* MF = F ON 0:8:.2
* M = B ON MF
* M = M&'(R ON M&'MF)
* DRAW M;
* MF = D COL 2
* M = (D COL 1)-(1+C)*MF
* M = M&'(R ON M&'MF)
* DRAW M, LINETYPE NONE, POINTTYPE "o"
* TOP TITLE "Bound/Free vs. Bound"
* VIEW
```




```
* DELETE W,MF,M
* USE GW
```

The &óperator denotes column concatenation, while the `ON` operator, as in `F ON H`, applies the function `F` to each row of the matrix `H` treated as an argument list for `F` and returns the column vector of result values.

It is an enlightening exercise to fit the Scatchard model to the corresponding transformed data and to then draw a graph of the original data points and the function f using the parameter values obtained. An example of this comparison is given in the following section.

In general, we should use weights which specify the relative accuracy of each data point. For each data matrix, a vector of weight values may be given as a clause in the `FIT` statement. The weight-value, g_i, that corresponds to an observation (x_i, y_i) should be $1/\operatorname{var}(y_i)$. Of course, guesses or estimates must generally suffice. The value $1/y_i^2$ for y_i corresponds approximately to a constant percentage error in the observed value y_i. Weight functions are permitted in MLAB, and an often more accurate device is to use the reciprocal of the square of the model function itself as the weighting function. This results in an iterative reweighting process.

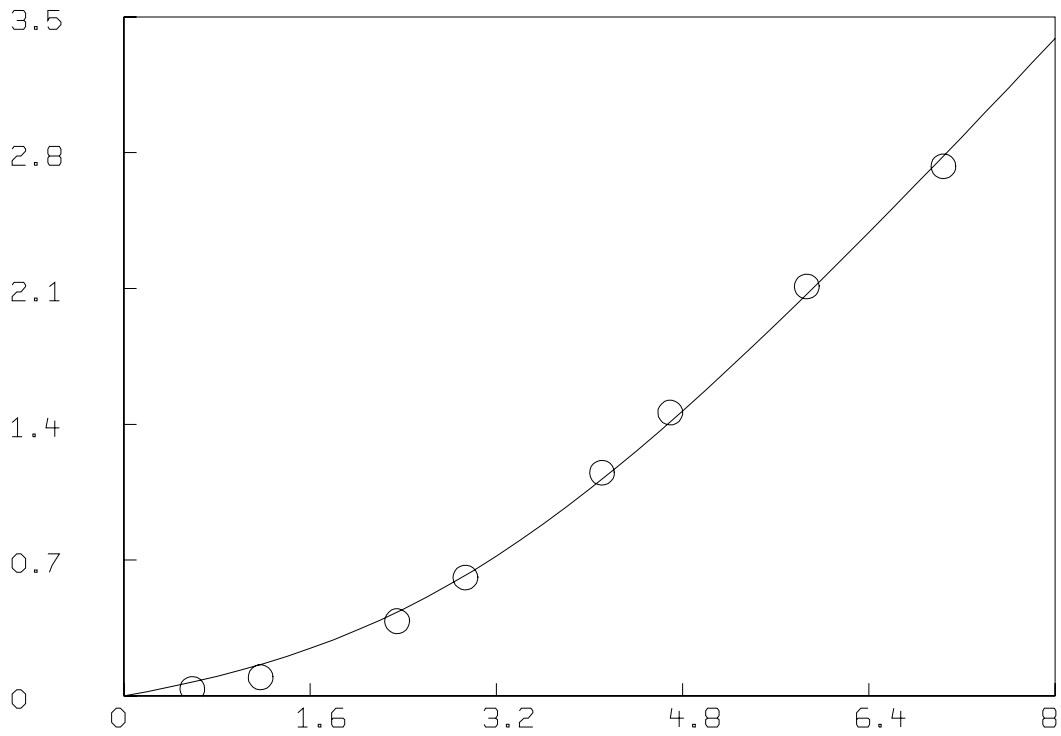
The use of weights permits data points of high reliability to exert greater influence on the parameter estimates than data points of lesser reliability. When simultaneous fitting is being done, weights have the additional function of compensating for differing units or magnitudes of the observations from various data sets. Without weights, the curve-fitting process would tend to favor one model component, fitting it at the expense of others, if the deviations there were much larger than the others, even though this may be an artifact of the use of different units.

MLAB includes an operator, `EWT`, which computes a vector of weight-values for a given set of data points, `M`, by estimating the standard deviation at each point by the difference between the data curve and a smoothed form of it, which effectively assumes the error is due to white noise. These standard-deviation estimates are, in turn, smoothed, and then used to form reciprocal variance weight values. In our example, we shall use `EWT` to obtain the necessary weight vectors, although this is not totally appropriate for chemical binding data, which is often taken in a range where the measurement accuracy improves as the amount of bound ligand increases. Also, `EWT` is unreliable for small numbers of data points. Nevertheless, we shall proceed as follows.

```
* DW = EWT(D); DN = EWT(N)
* TYPE D&'DW,N&'DN
: a  8 by 3 matrix
1:  0.58668    0.036    340.092919
```

```
2:  1.1734     0.096    340.092919
3:  2.3467     0.385    271.140209
4:  2.9334     0.61     256.511957
5:  4.1068     1.15     193.017413
6:  4.6934     1.46     171.441311
7:  5.8668     2.11     93.14747
8:  7.0402     2.73     93.14747
 a  5 by 3 matrix
1:  1.1734     0.3      159.445918
2:  2.5        0.5      159.445918
3:  3          0.3      156.341715
4:  3.98       0.7      134.665471
5:  6.5        1.6      134.665471

* fit (k1,k2,c), F to d with weight dw, Y to n with weight dn, CONSTRAINTS q
Begin iteration 1 bestsosq=5.86711e+01
Begin iteration 2 bestsosq=2.76531e+01
Begin iteration 3 bestsosq=2.69881e+01
final parameter values
value           error           dependency      parameter
1.681292363     0.3984166729    0.6190759098    K1
1.905413519e-17 0.06987097833   0.6367292361    K2
0.6401358429    0.057666687     0.09280130233   C
3 iterations
CONVERGED
best sum of squares = 2.69879e+01
root mean square error = 1.64280e+00
deviation fraction = 6.88260e-02
lagrange multiplier[1] = -0
lagrange multiplier[2] = -73.23991748
lagrange multiplier[3] = -0
* delete w
* draw d line none pt circle
* draw points(F,0:8:.2)
* draw n line NONE pointtype "+"
* draw points(Y,0:8:.2) linetype dashed
* VIEW
```



As mentioned above, MLAB also allows weights to be the reciprocals or other functions of the actual squared deviations themselves. This device is called curve-fitting with iterative reweighting and is invoked in MLAB by specifying an appropriate weight function. Moreover MLAB employs an iterative reweighting technique to allow the general form of a sum of *pth* powers to be minimized, rather than just a sum-of-squares. This so-called L_p-norm fitting is sometimes useful for $p = 1$ or 1.5 when the data is not normally-distributed, and tends to contain some outliers. The value p is specified as the value of the MLAB control variable `fitnorm`.

You can see that MLAB is an extremely flexible and general tool for curve-fitting. Moreover, it has a broad range of other useful functions, only a few of which have been alluded to here. (A more complete version of this paper is available from the author.)