Generalized Linear Least Squares Curve Fitting

Jack Merrin

February 12, 2017

1 Summary

Least squares curve fitting can be generalized to work with any linear model to determine unique parameter estimates and uncertainties in terms of matrix equations. Linear models follow a general form. All other models are classified as nonlinear. While nonlinear models do not have a general analytic solution, some nonlinear models can be transformed into linear models. We give two examples of transforming an exponential model and a transformation of the enzymatic rate equation to straight line models using propagation of error. Polynomials form a class of linear models which we implement in MATLAB.

2 Linear and nonlinear models

In least square curve fitting, there is a step where we take derivatives to find the parameter estimates. If those equations are too complicated then we will have a hard time solving them. It is straightforward to solve these equations for so called linear models. Linear models are of the form

$$f(x_i;\mathbf{a}) = a_1X_1(x_i) + a_2X_2(x_i) + \ldots + a_MX_M(x_i)$$

One example of a linear model with M parameters is a polynomial.

$$f(x_i;\mathbf{a}) = a_1 + a_2x_i + \ldots + a_Mx_i^{M-1}$$

Other linear models might have the x_i replaced with general functions $X_j(x_i)$. This gives a lot of different possible functions, but some are unnatural and do not occur in usual data sets.

$$f(x_i;\mathbf{a}) = a_1\sin(x_i) + a_2e^{x_i} + \ldots + a_M\ln(x_i)$$

Linear means the parameter is to the first power out in front. The following model is not linear.

$$f(x_i;\mathbf{a}) = a_1\sin(a_2x_i) + a_3e^{a_4x_i}$$

For models of the nonlinear form, the optimization problem is usually quite difficult to solve. We will consider what to do with nonlinear models later.

To performa curve fit on a model with M parameters you need at least M different points. A good practice is to have many more measured points than parameters. When models are bloated with a lot of parameters you can usually fit a wide variety of functions. Keep it as simple as possible but not simpler. Use only as few parameters are necessary.

3 The generalized least squares problem

Now with our linear model of M parameters we would like to derive the best parameter estimates and their uncertainties in the least squares weighted framework. The model is.

$$f(x_i; a_1, a_2, ..., a_M) = f(x_i; \mathbf{a}) = \sum_{k=1}^{M} a_k X_k(x_i)$$

where $X_k(x_i)$ are some generally defined continuous functions.

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

To minimizise χ^2, we set each derivative equal to zero.

$$\frac{\partial}{\partial a_k}\chi^2 = 0$$

This gives a $k = 1, 2, \ldots, M$ linear system of equations. These "normal equations" give the optimal fit parameters $\mathbf{a}$ which we can write as a vector.

$$\sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{M} a_j X_j(x_i) \right) \frac{X_k(x_i)}{\sigma_i^2} = 0$$

Taking terms on each side of the equal sign.

$$\sum_{i=1}^{N} \frac{y_i X_k(x_i)}{\sigma_i^2} = \sum_{i=1}^{N} \sum_{j=1}^{M} a_j X_j(x_i) \frac{X_k(x_i)}{\sigma_i^2}$$

It is useful to do some linear algebra to solve these equations for the parameters and their uncertainties. The "design matrix," A_{ij}, is defined as the N row and M column matrix

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

Second we define the column vector $\mathbf{B}$ of M rows.

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

A third $M \times M$ matrix, D_{jk} has inverse C_{jk}.

$$D_{jk} = \sum_{i=1}^{N} \frac{X_j(x_i)X_k(x_i)}{\sigma_i^2} \rightarrow \mathbf{D} = \mathbf{A}^T\mathbf{A}$$
$$\mathbf{C} = \mathbf{D}^{-1}$$

Now the normal equations can be rewritten as

$$\mathbf{Da} = \mathbf{B}$$

To solve for the fit parameters $\mathbf{a}$ it is easy to numerically compute the inverse in MATLAB.

$$\mathbf{a} = \mathbf{D}^{-1}\mathbf{B} = \mathbf{CB}$$
$$a_j = \sum_{k=1}^{M} C_{jk} \left[\sum_{i=1}^{N} \frac{y_i X_k(x_i)}{\sigma_i^2} \right]$$

4 General parameter uncertainties

Now, we can find the uncertainty in the parameters by the propagation of error formula.

$$\sigma_{a_j}^2 = \sum_{i=1}^{N} \left(\frac{\partial a_j}{\partial y_i} \right)^2 \sigma_i^2$$
$$\frac{\partial a_j}{\partial y_i} = \sum_{k=1}^{M} \frac{C_{jk}X_k(x_i)}{\sigma_i^2}$$
$$\sigma_{a_j}^2 = \sum_{k=1}^{M}\sum_{l=1}^{M} C_{jk}C_{jl} \left(\sum_{i=1}^{N} \frac{X_k(x_i)X_l(x_i)}{\sigma_i^2} \right)$$
$$\sigma_{a_j}^2 = \sum_{k=1}^{M}\sum_{l=1}^{M} C_{jk}C_{jl}D_{lk}$$

But $\sum_{l=1}^{M} C_{jl}D_{lk} = \delta_{jk}$ the Kroenecker delta because $\mathbf{C}$ is the inverse of $\mathbf{D}$.

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

so the sum reduces to

$$\sigma_{a_j}^2 = C_{jj}$$

These are the uncertainties in the fitting parameters. The off diagonal terms of $\mathbf{C}$ also give the covariances. If two different groups use the least squares theory on a linear model they should arrive at the same results for the fitting parameters and their uncertainties which are derived here. To summarize the results for linear least squares models

$$a_j = \sum_{k=1}^{M} C_{jk} \left[\sum_{i=1}^{N} \frac{y_i X_k(x_i)}{\sigma_i^2} \right] \qquad \sigma_{a_j} = \sqrt{C_{jj}}$$

5 Linearizing nonlinear models

Sometimes data following a nonlinear relationship may be transformed to a linear model. One can then estimate the transformed parameters exactly in conjunction with propagation of error. Here is a well known example from biochemistry for the velocity of a chemical reaction illustrating linearization.

$$V = \frac{V_{max}[S]}{K_m + [S]}$$

This equation can be rewritten as

$$\frac{1}{V} = \frac{K_m}{V_{max}}\frac{1}{[S]} + \frac{1}{V_{max}}$$

We start with $([S]_i, V_i, \sigma_{V_i})$ as the set of information. σ_{V_i} is the standard deviation of the mean of the repeated measurement of V_i. The transformed variables are

$$x_i = 1/[S]_i \quad y_i = 1/V_i \quad \sigma_i = \frac{\sigma_{V_i}}{V_i^2}$$
$$a_2 = \frac{K_m}{V_{max}} \qquad a_1 = \frac{1}{V_{max}}$$

Propagation of error for a single variable has been used to find σ_i. One can then weighted fit to a line to find a_1, a_2, σ_{a_1}, and σ_{a_2}. K_m and V_{max} can then be found also by propagation of error. This transformation generates a linear plot called the Lineweaver-Burke plot.

Another straightforward example of linearization involves exponential decay.

$$N(t) = N_0 e^{-kt}$$

One has the set of information (t_i, N_i, σ_{N_i}) where σ_{N_i} are the standard deviations of the mean. Now take the logarithm of both sides.

$$\log(N_i) = \log(N_0) - kt_i$$

Now we transform

$$x_i = t_i \quad y_i = \log(N_i) \quad \sigma_i = \sigma_{N_i}/N_i$$
$$a_1 = \log(N_0) \quad a_2 = -k$$

If the error bars were originally similar for most points on the linear plot they will be stretched out at longer times after transformation. After finding a_1 and σ_{a_1}, you can use error propagation to find N_0 and its uncertainty.

$$\sigma_{N_0} = e^{a_1}\sigma_{a_1}$$

Example 5.1. *Fit some simulated exponentially decaying data with linear least squares by linearizing the model.*

Solution 5.1. *We generate some model data which are the average of five points of* $y = 100e^{-t}$ *+ normrnd(0,4).*

```
clear;
ti = 0.0:0.1:2.5
N0 = 100;
L = 1;
Ni = 5;
for i = 1:length(ti)
    temp = [];
    for j = 1:Ni
        temp(j) = N0*exp(-L*ti(i))+normrnd(0,4);
    end
    yi(i) = mean(temp);
    ei(i) = std(temp)/sqrt(Ni);
end

figure(1);clf;
errorbar(ti,yi,ei,'.k');
axis([-0.5 3 0 125]);
xlabel('x');
ylabel('N');

figure(2);clf;
zi = log(yi);
si = ei./yi;
errorbar(ti,zi,si,'.k');
axis([-0.5 3 0 5])
xlabel('t');
ylabel('z');
hold on;

[A sigA B sigB chi2] = lineABx(ti, zi, si);
XX = 0:0.1:3
YY = A + XX.*B;
plot(XX,YY,'k');

estN = exp(A)
estsigN = exp(A)*sigA
estL = -B
estsigL = sigB
```

```
figure(1);
hold on;
XX = 0:0.01:3
YY = estN.*exp(-estL.*XX);
plot(XX,YY,'k');
```

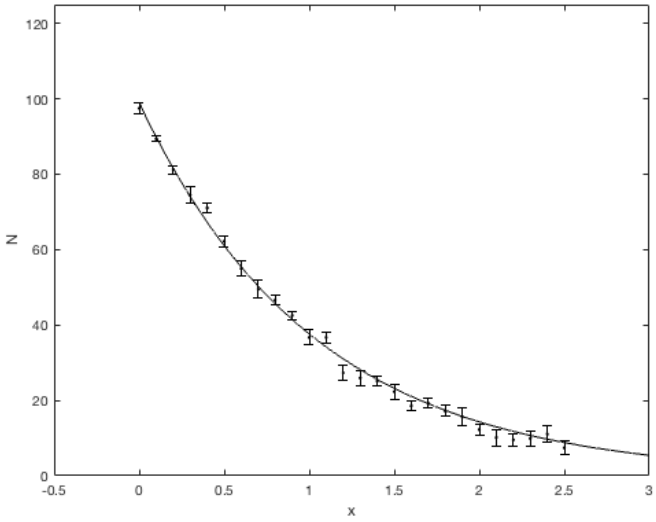

Figure 1: Exponential decay with best fit

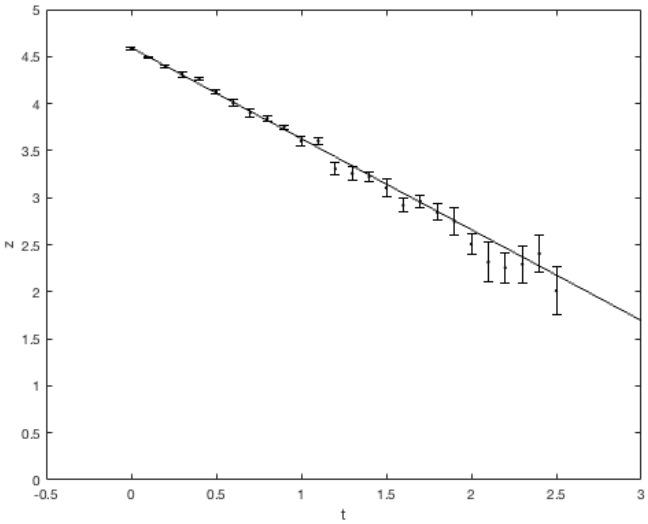

Figure 2: Linearization and weighted fit

6 Least squares polynomial

Example 6.1. *Implement a polynomial least squares fitting function in MATLAB.*

Solution 6.1. *Here is the MATLAB function we will use.*

```
function [ A sigA ] = genpolyfit( M, xi, yi, sigi )
%   [A sigA ] = genpolyfi( M, xi, yi, sigi)
%   Least sqaures polynomial fit to degree M-1
%

for k = 1:M
    X(k,:) = xi.^(k-1);
end

for j = 1:M
    for k = 1:M
        D(j,k) = sum(sigi.^(-2).*X(j,:).*X(k,:));
    end
end

for k = 1:M
    B(k) = sum(X(k,:).*yi.*sigi.^(-2));
end

A = inv(D)*B';
C = inv(D);
for k = 1:M
    sigA(k) = sqrt(C(k,k));
end
```

Example 6.2. *Simulate some data that can be fit by a parabola.*

Solution 6.2. *We can use the least square polynomial function.*

```
clear;figure(1);clf;
A = 1;
B = 2;
C = 3;
Ni = 5;
xi = -10:1:10;
for i = 1:length(xi)
    temp = [];
    for j = 1:Ni
```

```
        temp(j) = A + B*xi(i) + C*xi(i).^2 + normrnd(0,10);
    end
    yi(i) = mean(temp);
    si(i) = std(temp)/sqrt(Ni);
end
figure(1);clf;
errorbar(xi,yi,si,'.k', 'markersize',10);
xlabel('x')
ylabel('y')
hold on;
[A sigA] = genpolyfit(3,xi,yi,si);
x1 = -11:0.1:11;
y1 = A(3).*x1.^2 + A(2).*x1 + A(1);
plot(x1,y1, 'r');
axis( [ -11 11 -50 1.1*max(y1)]);'
```

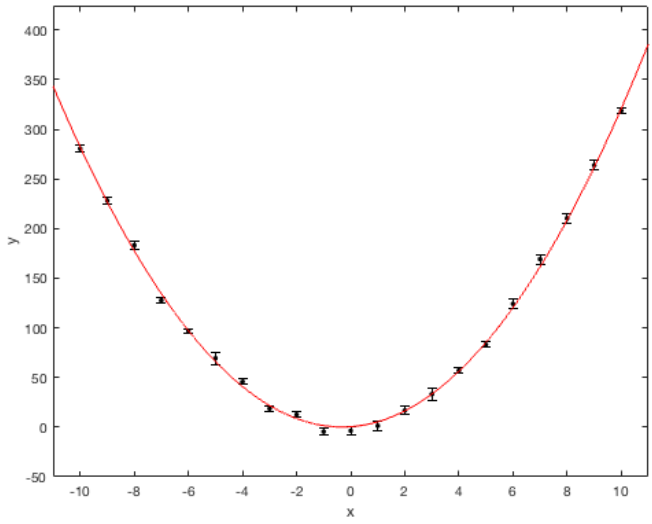

Figure 3: Curve fit to a parabola