Conducting a Nonlinear Fit Analysis in MATLAB

Disclaimer: This document is intended as an overview of the MATLAB commands required to use the nonlinear fit function. If you are unfamiliar with nonlinear regression it is recommend that you read *Fitting Curves to Data using Nonlinear Regression*. This provides an overview of how nonlinear regression works and how to analyze the results.

MATLAB can be used to fit an arbitrary model equation to a set of data points. The MATLAB function that accomplishes this is **nlinfit**

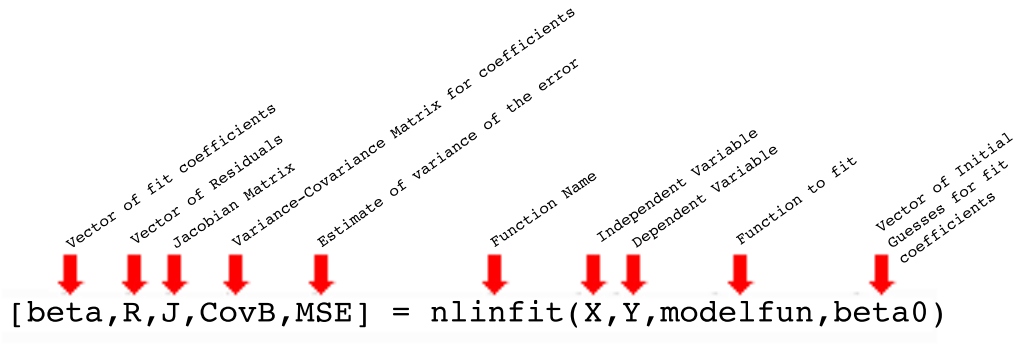

Figure 1: MATLAB's nlinfit Function

Using the Function

To use the **nlinfit** function you must first have independent and dependent variables to pass into the x and y positions in the function shown in Figure 1. You must also input a model equation that you would like to fit to your data. This model will go in the *modelfun* position of the function.

For example, let's say you were trying to fit the data to the Michaelis-Menten equation for enzyme kinetics. The governing equation is: $V = \frac{V_{max}[S]}{Km+[S]}$, where [S] is the independent variable, V is the dependent variable, and V_{max} and K_m are constants (parameters). To write this as a function handle in MATLAB you will need to type:

modelfun=@(b,S) b(1).*S./(b(2)+S)

This line of code will set the variable *modelfun* to a function of b and S, where b is a vector of the constant coefficients corresponding to their positions in the model equation (b(1)=V_{max} and b(2)=K_M) and S is the independent variable (in this case the concentration of substrate).

You must also provide an initial guess for the coefficients: *beta0*. This is a vector where the values correspond to your guess for each coefficient. For example, in the Michaelis-Menten equation there are two unknown coefficients that you must provide an initial guess for. In this case *beta0* is a 1 by 2 vector. It may look like beta0=[1 1], where each guess in this case is 1. It is best to estimate the coefficients and apply the estimates as guesses. Note that it is possible for the solution method to diverge if the wrong guess is chosen.

Understanding the Outputs

Once you run the code, the MATLAB function will return five objects:

- Beta: Vector of the calculated coefficients in the equation you are fitting the data to. Each value corresponds to the value of the specified parameters in the model function in the order in which they are specified.
- R: Vector of the residuals between your fit and the data points. Each value of the vector refers to the numerical difference between the data point and the value of the fit equation for a given value of the independent variable.

- J: The Jacobian matrix of your model function. It is analogous to the model matrix X in linear regression, which has a column of 1's representing the constants and a second column representing the predictors (y=**X**b+e). It is useful for determining 95% confidence intervals for the calculated parameters.
- CovB: The variance-covariance matrix for the coefficients. The diagonal values (upper left to lower right) are the parameter variances, while the other locations are the parameter covariances. The matrix is used to evaluate parameter uncertainty and parameter correlation.
- MSE: An estimate of the variance of the error. It is equal to the residual sum of squares divided by the number of degrees of freedom.

These outputs can be used to calculate descriptive statistics about your fit, most notably 95% confidence intervals for your calculated coefficients and the r^2 value.

95% Confidence Intervals

betaci=nlparci(beta,R,J)

This will return a vector of two columns and n rows, where n is the total number of estimated coefficients. The number in the first column is the lower confidence bound for the nth coefficient, and the number in the second column is the upper confidence bound for the nth coefficient. The methods used by MATLAB to determine these values are advanced and it is best to view this process as a "black box"[1]. If run correctly the output should look like (for n parameters):

Lower Bound for Parameter 1	Upper Bound for Parameter 1
Lower Bound for Parameter 2	Upper Bound for Parameter 2
.	.
.	.
.	.
Lower Bound for Parameter n	Upper Bound for Parameter n

r^2 Value

To calculate the r^2 value, use the formula $r^2 = 1 - (SS_{residual}/SS_{Total})$. $SS_{residual}$ is the sum of squares of the residuals and SS_{Total} is the sum of squares between the data points and their mean[2]. This can be accomplished using the following MATLAB commands:

```
SSresidual=sum(R.^2);
SStotal=sum((y-mean(y)).^2);
rsquare=1-(SSresidual/SStotal);
```

This will return the r^2 value, based on the calculated residuals (R) and the dependent variable (y).

[1]For information on the algorithm used see: *http://www.mathworks.com/support/solutions/en/data/1-194W2/index.html?product=ML&solution=1-194W2*

[2]See *Fitting Curves to Data using Nonlinear Regression* on the BOSS