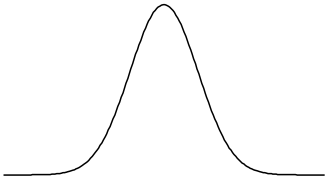
Built-in Curve Fitting Functions

For the most part you will get good results using automatic guesses. A few require additional input beyond what is summarized in the preceding sections. This section contains notes on the fitting functions that give a bit more detail where it may be helpful.

gauss

Fits a Gaussian peak.

$$y_0 + A\exp\left[-\left(\frac{x-x_0}{width}\right)^2\right]$$

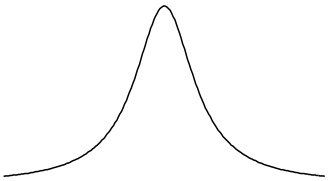


Note that the width parameter is sqrt(2) times the standard deviation of the peak. This is different from the w_i parameter in the Gauss function, which is simply the standard deviation.

lor

Fits a Lorentzian peak.

$$y_0 + \frac{A}{(x-x_0)^2 + B}$$



Voigt

Fits a Voigt peak, a convolution of a Lorentzian and a Gaussian peak. By changing the ratio of the widths of the Lorentzian and Gaussian peak components the shape can grade between a Lorentzian shape and a Gaussian shape.

$$y_0 + \frac{2Area}{W_G}\sqrt{\frac{\ln(2)}{\pi}} \bullet Voigt\left[\frac{2\sqrt{\ln(2)}}{W_G}(x-x_0), Shape \bullet 2\sqrt{\ln(2)}\right]$$

The Voigt function is a normalized Voigt peak shape function. It is the same as the built-in function **VoigtFunc**.

The premultiplier and arguments to the Voigt function result in a peak shape in which fit coefficients are:

Area	The area under the peak excluding the vertical offset
y0	The vertical offset
WG	The Gaussian full width at half maximum (FWHM)
Shape	The ratio of the Lorentzian and Gaussian components, W_L/W_G

There is no analytic expression to compute the height of a Voigt peak.

The FWHM of the Voigt peak can be approximated as

$$W_V = \frac{W_L}{2} + \sqrt{\frac{W_L^2}{4} + W_G^2}$$

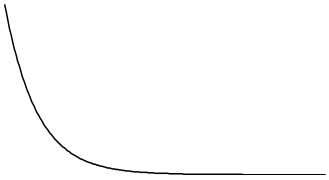
where W_L = Shape * W_G.

Fitting peaks with extreme values of Shape is not recommended as it suffers from numerical instability. Fit the end-point shapes Lor (Shape → ∞) or Gauss (Shape = 0) instead.

exp_XOffset

Fits a decaying exponential.

$$y_0 + A\exp\left(\frac{x-x_0}{\tau}\right)$$



In this equation, x_0 is a constant, not a fit coefficient. During generation of automatic guesses, x_0 will be set to the first X value in your fit data. This eliminates problems caused by floating-point roundoff.

You can set the value of x_0 using the /K flag with the CurveFit operation, but it is recommended that you accept the automatic value. Setting x_0 to a value far from the initial X value in your input data is guaranteed to cause problems.

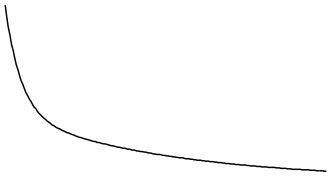
Note: The fit coefficient τ is the inverse of the equivalent coefficient in the exp function. It is actually the decay constant, not the inverse decay constant.

Automatic guesses don't work for growing exponentials (negative τ). To fit a negative value of τ, use Manual Guess on the Coefficients tab, or CurveFit/G on the command line.

dblexp_XOffset

Fits a sum of two decaying exponentials.

$$y_0 + A_1\exp\left(\frac{x-x_0}{\tau_1}\right) + A_2\exp\left(\frac{x-x_0}{\tau_2}\right)$$



In this equation, x_0 is a constant, not a fit coefficient. During generation of automatic guesses, x_0 wall be set to the smallest X value in your fit data. This eliminates problems caused by floating-point roundoff.

You can set the value of x_0 using the /K flag with the CurveFit operation, but it is recommended that you accept the automatic value. Setting x_0 to a value far from the initial X value in your input data is guaranteed to cause problems.

Note: The fit coefficients τ_1 and τ_2 are the inverse of the equivalent coefficients in the dblexp function. They are actual decay constants, not inverse decay constants.

See the notes for **exp_XOffset** on page III-207 for growing exponentials. You will also need to use manual guesses if the amplitudes have opposite signs:

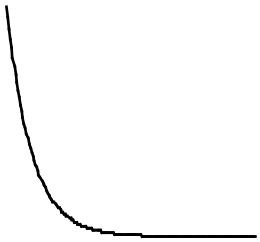
See also the dblexp_peak fit function described below.

If the two decay constants (τ_1 and τ_2) are not quite distinct you may not get accurate results.

exp

Fits a decaying exponential. Similar to exp_XOffset, but not as robust. Included for backward compatibility; in new work you should use exp_Xoffset.

$y_0 + A\exp(-Bx)$



Note that offsetting your data in the X direction will cause changes in *A*. Use exp_XOffset for a result that is independent of X position.

Note: The fit coefficient *B* is the inverse decay constant.

Automatic guesses don't work for growing exponentials (negative *B*). To fit a negative value of *B*, use Manual Guess on the Coefficients tab, or CurveFit/G on the command line.

Floating-point arithmetic overflows will cause problems when fitting exponentials with large X offsets. This problem often arises when fitting decays in time as times are often large. The best solution is to use the exp_XOffset fit function. Otherwise, to fit such data, the X values must be offset back toward zero.

You could simply change your input X values, but it is usually best to work on a copy. Use the Duplicate command on the command line, or the Duplicate Waves item in the Data menu to copy your data.

For an XY pair, execute these commands on the command line (these commands assume that you have made a duplicate wave called `myXWave_copy`):

```
Variable xoffset = myXWave_copy[0]
myWave_copy[0] -= xoffset
```

Note that these commands assume that you are fitting data from the beginning of the wave. If you are fitting a subset, replace `[0]` with the point number of the first point you are fitting. If you are using graph cursors to select the points, substitute `[pcsr(A)]`. This assumes that the round cursor (cursor A) marks the beginning of the data.

If you are fitting to waveform data (you selected _calculated_ in the X Data menu) then you need to set the x0 part of the wave scaling to offset the data. If you are fitting the entire wave, simply use the Change Wave Scaling dialog from the Data menu to set the x0 part of the scaling to zero. If you are fitting a subset selected by graph cursors, it is easier to change the scaling on the command line:

```
SetScale/P x leftx(myWave_copy)-xcsr(A), deltax(myWave_copy), myWave_copy
```

This command assumes that you have used the round cursor (cursor A) to mark the beginning of the data.

Subtracting an X offset will change the amplitude coefficient in the fit. Often the only coefficient of interest is the decay constant (invTau) and the change in the amplitude can be ignored. If that is not the case, you can calculate the correct amplitude after the fit is done:

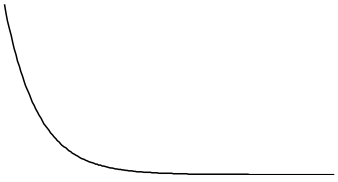
```
W_coef[1] = W_coef[1]*exp(W_coef[2]*xoffset)
```

If you are fitting waveform data, the value of xoffset would be `-leftx(myWave_copy)`.

dblexp

Fits a sum of decaying exponentials. Similar to dblexp_XOffset, but suffers from floating-point roundoff problems if the data do not start quite close to x=0. Included for backward compatibility; in new work you should use exp_Xoffset.

$y_0 + A_1\exp(-B_1x) + A_2\exp(-B_2x)$



Note that offsetting your data in the X direction will cause changes in A_1 and A_2. Use dblexp_XOffset for a result that is independent of X position.

Note: The fit coefficients B_1 and B_2 are inverse decay constants.

See the notes for exp for growing exponentials. You will also need to use manual guesses if the amplitudes have opposite signs:

See also the dblexp_peak fit function described below.

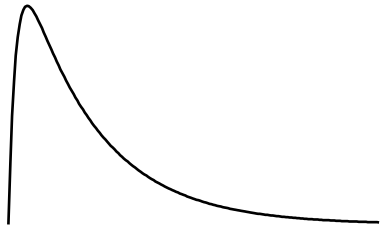
If the two decay constants (B_1 and B_2) are not quite distinct you may not get accurate results.

Fitting data with a large X offset will have the same sort of troubles as when fitting with exp. The best solution is to use the dblexp_XOffset fit function; you can also solve the problem using a procedure similar to the one outlined for **exp** on page III-207.

dblexp_peak

Fits the sum of two exponential terms of opposite sign, making a peak.

$$y_0 + A \cdot \left\{ -exp\left[\frac{-(x-x_0)}{tau_1}\right] + exp\left[\frac{-(x-x_0)}{tau_2}\right] \right\}$$



The location of the peak is given by

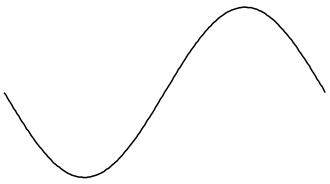
$$X_{peak} = \frac{tau_1\ tau_2\ ln\left(\frac{tau_1}{tau_2}\right)}{tau_1 - tau_2} + x_0$$

If you need individual control of the amplitudes of the two terms, use the dblexp_XOffset function. In that case, you will need to use manual guesses.

sin

Fits a sinusoid.

$$(y_0 + A\sin(fx + \phi))$$



phi is in radians. To convert to degrees, multiply by 180/Pi.

A sinusoidal fit takes an additional parameter that sets the approximate frequency of the sinusoid. This is entered in terms of the approximate number of data points per cycle. When you choose `sin` from the Function menu, a box appears where you enter the expected number of points per cycle.

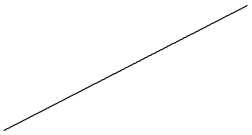
If you enter a number less than 6, the default value will be 7. It may be necessary to try various values to get good results. You may want to simply use manual guesses.

The nature of the sin function makes it impossible for a curve fit to distinguish phases that are different by 2π. It is probably easier to subtract 2nπ than to try to get the fit to fall in the desired range.

line

Fit a straight line through the data.

$$(a + bx)$$



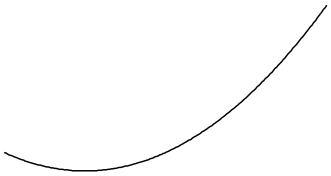
Never requires manual guesses.

If you want to fit a line through the origin, in the Coefficients tab select the Hold box for coefficient *a* and set the Initial Guess value to zero.

poly n

Fits a polynomial with n terms, or order n-1.

$$(K_0 + K_1 x + K_2 x^2 + \ldots)$$



A polynomial fit takes an additional parameter that sets the number of polynomial terms. When you choose `poly` from the Function menu, a box appears where you enter that value.

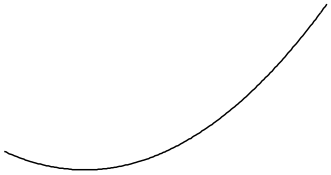
The minimum value of n is 1 corresponding to taking the mean of the Y value. Most Igor users won't need anything less than 3, a quadratic polynomial.

A polynomial fit never requires manual guesses.

poly_XOffset n

Fits a polynomial with n terms, or order n-1. The constant x_0 is not an adjustable fit coefficient; it allows you to place the polynomial anywhere along the X axis. This would be particularly useful for situations in which the X values are large, for instance when dealing with date/time data.

$$(K_0 + K_1(x - x_0) + K_2(x - x_0)^2 + \ldots)$$



The poly_XOffset fit function takes additional parameters that set the number of polynomial terms and the value of x_0. When you select poly_XOffset from the Function menu, boxes appear where you enter these values.

The minimum value of n is 1 corresponding to taking the mean of the Y value. Most Igor users won't need anything less than 3, a quadratic polynomial.

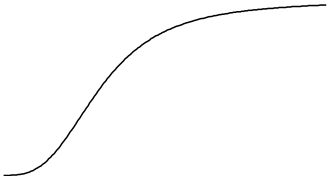
When Set Constant X0 is set to Auto, x_0 is set to the minimum X value found in the data you are fitting. You can set x_0 to any value you wish. Values far from the X values in your data set will cause numerical problems.

A polynomial fit never requires manual guesses.

HillEquation

Fits Hill's Equation, a sigmoidal function.

$$base + \frac{(max - base)}{1 + \left(\frac{x_{1/2}}{x}\right)^{rate}}$$



The coefficient *base* sets the y value at small X, *max* sets the y value at large X, *rate* sets the rise rate and $x_{1/2}$ sets the X value at which Y is at (*base* + *max*)/2.

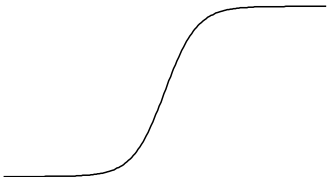
Note that X values must be greater than 0. Including a data point at $X \le 0$ will result in a singular matrix error and the message, "The fitting function returned NaN for at least one X value."

You can reverse the values of *base* and *max* to fit a falling sigmoid.

sigmoid

Fits a sigmoidal function with a different shape than Hill's equation.

$$base + \frac{max}{1 + \exp\left(\frac{x_0 - x}{rate}\right)}$$

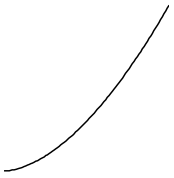


The coefficient *base* sets the Y value at small X. The Y value at large X is *base*+*max*. x0 sets the X value at which Y is at (*base*+*max*/2) and *rate* sets the rise rate. Smaller *rate* causes a faster rise, specifically, the slope at x=x0 is *max*/(4**rate*).

power

Fits a power law.

$$(y_0 + Ax^{pow})$$



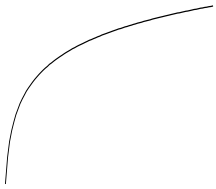
May be difficult to fit, requiring good initial guesses, especially for *pow* > 1 or *pow* close to zero.

Note that X values must be greater than 0. Including a data point at $X \le 0$ will result in a singular matrix error and the message, "The fitting function returned NaN for at least one X value."

log

Fits a logarithmic curve.

$$a + b\log(x)$$



log is very forgiving of bad initial guesses. It was added in Igor Pro 9.00.

X values must be greater than 0. Including a data point with X<=0 results in a singular matrix error and the message "The fitting function returned NaN for at least one X value."