then *x_value*=0 references trace point 0 which corresponds to wave point 4, and *x_value*=1 references trace point 1 which corresponds to wave point 7.

Moving a cursor in a macro or function does not immediately erase the old cursor. DoUpdate has to be explicitly called.

Examples

```
Display myWave                          // X coordinates from X scaling of myWave
Cursor A, myWave, leftx(myWave)         //cursor A on first point of myWave

AppendToGraph yWave vs xWave            //X coordinates from xWave, not X scaling
Cursor/P B,yWave,numpnts(yWave)-1       //cursor B on last point of yWave
DoUpdate                                // erase any old A or B cursors
```

See Also

Programming With Cursors on page II-321 and the **DoUpdate** operation.

CursorStyle

CursorStyle

CursorStyle is a procedure subtype keyword that puts the name of the procedure in the "Style function" submenu of the Cursor Info pop-up menu. It is automatically used when Igor creates a cursor style function. To create a cursor style function, choose "Save style function" in the "Style function" submenu of the Cursor Info pop-up menu.

See also **Programming With Cursors** on page II-321.

CurveFit

CurveFit [***flags***] ***fitType***, [**kwCWave=*coefWaveName*,**] ***waveName*** [***flag parameters***]

The CurveFit operation fits one of several built-in functions to your data (for user-defined fits, see the **FuncFit** operation). When with CurveFit and built-in fit functions, automatic initial guesses will provide a good starting point in most cases.

The results of the fit are returned in a wave, by default W_coef. In addition, the results are put into the system variables K0, K1 … K*n* but the use of the system variables is limited and considered obsolete

As explained under **Fitting with Complex-Valued Functions** on page III-248, a user-defined fitting function can return complex values. A complex fitting function can be written to require a complex coefficient wave in which case the system variables are not supported.

You can specify your own wave for the coefficient wave instead of W_coef using the kwCWave keyword.

Virtually all waves specified to the CurveFit operation can be a sub-range of a larger wave using the same sub-range syntax as the Display operation uses for graphing. See **Wave Subrange Details** on page V-132.

See Chapter III-8, **Curve Fitting** for detailed information including the use of the Curve Fit dialog.

CurveFit operation parameters are grouped in the following categories: flags, fit type, parameters (kwCWave=*coefWaveName* and *waveName*), and flag parameters. The sections below correspond to these categories. Note that flags must precede the fit type and flag parameters must follow *waveName*.

Flags

/B=*pointsPerCycle*	Used when *type* is sin; *pointsPerCycle* is the estimated number of data points per sine wave cycle. This helps provide initial guesses for the fit. You may need to try a few different values on either side of your estimated points/cycle.
/C	Makes constraint matrix and vector. This only applies if you use the /C=*constraintSpec* parameter to specify constraints (see below). Creates the M_FitConstraint matrix and the W_FitConstraint vector. For more information, see **Fitting with Constraints** on page III-227.
/G	Use values in variables K0, K1 … K*n* as starting guesses for a fit. If you specify a coefficient wave with the kwCWave keyword, the starting guesses will be read from the coefficient wave.

/H="*hhh…*"

Specifies coefficients to hold constant.

h is 1 for coefficients to hold, 0 for coefficients vary.

For example, `/H="100"` holds K0 constant, varies K1 and K2.

/K={*constants*}

Sets values of constants (not fit coefficients) in certain fitting functions. For instance, the exp_XOffset function contains an X offset constant. Built-in functions will set the constant automatically, but the automatic value can be overridden using this flag.

constants is a list of constant values, e.g., /K={1,2,3}. The length of the list must match the number of constants used by the chosen fit function.

This flag is not currently supported by the Curve Fit dialog. Use the To Cmd button and add the flag on the command line.

/L=*destLen*

Sets the length of the wave created by the AutoTrace feature, that is, /D without destination wave (see the /D parameter below, in the section Flag Parameters). The length of the wave fit_*waveName* will be set to *destLen*. This keyword also sets the lengths of waves created for confidence and prediction bands.

/M

Generates the covariance matrix, the waves CM_K*n*, where *n* is from 0 (for K0) to the number of coefficients minus one.

/M=*doMat*

Generates the covariance matrix. If *doMat* =2, the covariance matrix is put into a 2D matrix wave called M_Covar. If *doMat* =1 or is missing, the covariance matrix is generated as the 1D waves CM_K*n*, where *n* is from 0 (for K0) to the number of coefficients minus one. If *doMat* =0, the covariance matrix is not generated. *doMat* =1 is included for compatibility with previous versions; it is better to use *doMat* =2.

CurveFit does not generate the covariance matrix for line fits. Instead it reports the coefficient of correlation via the output variable V_rab.

/N[=*updateMode*]

Controls updating of windows during a curve fitting operation. Updating windows after each iteration can slow curve fitting down significantly.

updateMode = 0: An update is done every iteration so that you can monitor the progress of a fit. An update is performed when the fit finishes.

updateMode = 1: Suppresses updates after each fit iteration.If you use the /X flag, a screen update is done to ensure that the X range is accurate. /N is the same as /N=1. This is the default update mode.

updateMode = 2: Suppresses all updates including when using the /X flag. In a user-defined function, you must ensure that your graph has been updated by calling **DoUpdate** if you use the /X flag. This mode requires Igor Pro 9.00 or later.

See **Curve Fitting Screen Updates** for details.

/NTHR = *nthreads*

This flag is accepted but is obsolete and does nothing. See **Curve Fitting with Multiple Processors** on page III-249 for further information.

/O

Only generates initial guesses; doesn't actually do the fit.

Unless /ODR=2, this flag is ignored when used with linear fit types (line, poly, poly_XOffset, and poly2D). The **FuncFit** operation also ignores this flag.

/ODR=*fitMethod* Selects a fitting method. Values for *fitMethod* are:

0: Default Levenberg-Marquardt least-squares method using old code.

1: Trust-region Levenberg-Marquardt ordinary least-squares method implemented using ODRPACK95 code. See **Curve Fitting References** on page III-267.

2: Trust-region Levenberg-Marquardt least orthogonal distance method implemented using ODRPACK95 code. This method is appropriate for fitting when there are measurement errors in the independent variables, sometimes called "errors in variables fitting", "random regressor models," or "measurement error models".

3: Implicit fit. No dependent variable is specified; instead fitting attempts to adjust the fit coefficients such that the fit function returns zero for all dependent variables.

Implicit fitting will be of almost no use with the built-in fitting functions. Instead, use **FuncFit** and a user-defined fit function designed for an implicit fit.

/Q[=*quiet*] If quiet = 1, prevents results from being printed in history. /Q is the same as /Q=1.

/TBOX = *textboxSpec* If the top graph displays the fitted data, the /TBOX flag adds an annotation to the graph or updates the annotation if it already exists. Graphs other than the top graph are not affected.

The textbox contains a customizable set of information about the fit. The *textboxSpec* is a bitfield to select various elements to be included in the textbox:

1	Title "Curve Fit Results"
2	Date
4	Time
8	Fit Type (Least Squares, ODR, etc.)
16	Fit function name
32	Model Wave, the autodestination wave (includes a symbol for the trace if appropriate)
64	Y Wave, with trace symbol
128	X Wave
256	Coefficient value report
512	Include errors in the coefficient value report

Request inclusion of various parts by adding up the values for each part you want.

textboxSpec defaults to all bits on.

Setting *textboxSpec* to zero removes the textbox.

/X[=*fullRange*] /X or /X=1 sets the X scaling of the auto-trace destination wave to match the appropriate X axis on the graph when the Y data wave is on the top graph. This is useful when you want to extrapolate the curve outside the range of X data being fit.

If you omit /X or specify /X=0, the X scaling of the auto-trace destination wave is set to the range of X values being fit.

/W=*wait* Specifies behavior for the curve fit results window.

wait=1: Wait till user clicks OK button before dismissing curve fit results window. This is the default behavior from the command line or dialog.

wait=0: Display the curve fit window but do not wait for the user to click the OK button.

wait=2: Do not display the curve fit results window at all. This is the default for fits run from a procedure.

Fit Types

fitType is one of the built-in curve fit function types:

gauss	Gaussian peak: $y = K_0 + K_1 \exp\left[-\left(\frac{x - K_2}{K_3}\right)^2\right]$.
lor	Lorentzian peak: $y = K_0 + \frac{K_1}{(x - K_2)^2 + K_3}$.
Voight	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 = VoigtPeak(W_coef, x).
exp	Exponential: $y = K_0 + K_1 \exp(-K_2 x)$.
dblexp	Double exponential: $y = K_0 + K_1 \exp(-K_2 x) + K_3 \exp(-K_4 x)$.
exp_XOffset	Exponential: $y = K_0 + K_1 \exp(-(x - x_0)/K_2)$. x_0 is a constant; by default it is set to the minimum x value involved in the fit. Inclusion of x_0 prevents problems with floating-point roundoff errors that can afflict the exp function.
dblexp_XOffset	Double exponential: $y = K_0 + K_1 \exp(-(x - x_0)/K_2) + K_3 \exp(-(x - x_0)/K_{4(2)})$. x_0 is a constant; by default it is set to the minimum x value involved in the fit. Inclusion of x_0 prevents problems with floating-point roundoff errors that can afflict the exp function.
dblexp_peak	Double exponential with amplitudes of opposite sign, making a peak: $y = K_0 + K_1 \cdot \left\{-exp\left[\frac{-(x - K_4)}{K_2}\right] + exp\left[\frac{-(x - K_4)}{K_3}\right]\right\}$
sin	Sinusoid: $y = K_0 + K_1 \sin(K_2 x + K_3)$.
line	Line: $y = K_0 + K_1 x$.
poly n	Polynomial: $y = k_0 + K_1 x + K_2 x^2 + \ldots$. n is from 1 to 20. n is the number of terms or the degree plus one. Prior to Igor Pro 9.00, the minimum accepted value for n was 3. n = 1 is equivalent to finding the mean of the Y values, and n = 2 fits a line to the data set, but doesn't output the special statistical information you get if you use the line fit type.
poly_XOffset n	Polynomial: y = K_0 + K_1*(x - x_0) + K_2*(x - x_0)^2 +... n is from 1 to 20. n is the number of terms or the degree plus one. Prior to Igor Pro 9.00, the minimum accepted value for n was 3. n = 1 is equivalent to finding the mean of the Y values, and n = 2 fits a line to the data set, but doesn't output the special statistical information you get if you use the line fit type. x_0 is a constant; by default it is set to the minimum X value involved in the fit. Inclusion of x_0 prevents problems with floating-point roundoff errors when you have large values of X in your data set.

hillequation — Hill's Equation: $y = K_0 + \frac{(K_1 - K_0)}{1 + (K_3/x)^{K_2}}$.

This is a sigmoidal function. X values must be greater than 0.

sigmoid — $y = K_0 + \frac{K_1}{1 + \exp(K_2 - x/K_3)}$.

power — Power law: $y = K_0 + K_1 x^{K_2}$. X values must be greater than 0.

log — Log: $y = K_0 + K_1 \log(x)$. X values must be greater than 0.

lognormal — Log normal: $y = K_0 + K_1 \exp\left[-\left(\frac{\ln(x/K_2)}{K_3}\right)^2\right]$. X values must be greater than 0.

gauss2D — 2D Gaussian: $z = K_0 + K_1 \exp\left[\frac{-1}{2(1-K_6^2)}\left(\left(\frac{x-K_2}{K_3}\right)^2 + \left(\frac{y-K_4}{K_5}\right)^2 - \frac{2K_6(x-K_2)(y-K_4)}{K_3 K_5}\right)\right]$.

The cross-correlation coefficient (K_6) must be between -1 and 1. This coefficient is automatically constrained to lie in that range. If you are confident that the correlation is zero, it may greatly speed the fit to hold it at zero.

poly2D *n* — Two-dimensional polynomial: $z = K_0 + K_1 x + K_2 y + K_3 x^2 + K_4 xy + K_5 y^2 + \ldots$.

where *n* is the degree of the polynomial. All terms up to degree *n* are included, including cross terms. For instance, degree 3 terms are x^3, x^2y, xy^2, and y^3.

For more details on the built-in fit functions, see **Built-in Curve Fitting Functions** on page III-206.

Parameters

kwCWave=*coefWaveName* specifies an optional coefficient wave. If present, the specified coefficient wave is set to the final coefficients determined by the curve fit. If absent, a wave named W_coef is created and is set to the final coefficients determined by the curve fit.

If you use kwCWave=*coefWaveName* and you include the /G flag, initial guesses are taken from the specified coefficient wave.

waveName is the wave containing the Y data to be fit to the selected function *type*. You can fit to a subrange of the wave by supplying (*startX*,*endX*) after the wave name. Though not shown in the syntax description, you can also specify the subrange in points by supplying [*startP*,*endP*] after the wave name. See **Wave Subrange Details** on page V-132 for more information on subranges of waves in curve fitting.

If you are using one of the two-dimensional fit functions (gauss2D or poly2D) either *waveName* must name a matrix wave or you must supply a list of X waves via the /X flag.

Flag Parameters

These flag parameters must follow *waveName*.

/A=*appendResid* — *appendResid* =1 (default) appends the automatically-generated residual to the graph and *appendResid* =0 prevents appending (see /R[=*residwaveName*]). With *appendResid* =0, the wave is generated and filled with residual values, but not appended to the graph.

/AD[=*doAutoDest*] — If *doAutoDest* is 1, it is the same as /D alone. /AD is the same as /AD=1.

/C=*constraintSpec*

Applies linear constraints during curve fitting. Constraints can be in the form of a text wave containing constraint expressions (/C=*textWaveName*) or a suitable matrix and vector (/C={*constraintMatrix*, *constraintVector*}). See **Fitting with Constraints** on page III-227. **Note**: Constraints are not available for the built-in line, poly and poly2D fit functions. To apply constraints to these fit functions you must create a user-defined fit function.

/D [=*destwaveName*]

destwaveName is evaluated based on the equation resulting from the fit. *destwaveName* must have the same length as *waveName*.

If only /D is specified, an automatically-named wave is created. The name is based on the *waveName* with "fit_" as a prefix. This automatically named wave will be appended (if necessary) to the top graph if *waveName* is graphed there. The X scaling of the fit_ wave is set from the range of x data used during the fit.

By default the length of the automatically-created wave is 200 points (or 2 points for a straight line fit). This can be changed with the /L flag.

If *waveName* is a 1D wave displayed on a logarithmic X axis, Igor also creates an X wave with values exponentially spaced. The name is based on *waveName* with "fitX_" as a prefix.

If you fit to a user-defined function that returns complex values, the destination wave will also be complex.

/F={*confLevel*, *confType* [, *confStyleKey* [, *waveName*…]]}

Calculates confidence intervals for a confidence level of *confLevel*. The value of *confLevel* must be between 0 and 1 corresponding to confidence levels of 0 to 100 per cent.

confType selects what to calculate:

1:	Confidence bands for the model.
2:	Prediction bands for the model.
4:	Confidence intervals for the fit coefficients.

These values can be added together to select multiple options. That is, to select both a confidence band and fit coefficient confidence intervals, set *confType* to 5.

Confidence and prediction bands can be shown as waves contouring a given confidence level (use "Contour" for *confStyleKey*) or as error bars (use "ErrorBar" for *confStyleKey*). The default is Contour.

If no waves are specified, waves to contain the results are automatically generated and appended to the top graph (if the top graph contains the fitted data). See **Confidence Band Details** for details on the waves for confidence bands.

Note: Confidence bands and prediction bands are not available for multivariate curve fits.

/I [=*weightType*]

If *weightType* is 1, the weighting wave (see /W parameter) contains standard deviations. If *weightType* is 0, the weighting wave contains reciprocal of the standard deviation. If the /I parameter is not present, the default is /I=0.

/M=*maskWaveName*

Specifies that you want to use the wave named *maskWaveName* to select points to be fit. The mask wave must match the dependent variable wave in number of points and dimensions. Setting a point in the mask wave to zero or NaN (blank in a table) eliminates that point from the fit. The mask wave must be real even when fitting complex data.

/R [=*residwaveName*] Calculates elements of *residwaveName* by subtracting model values from the data values. *residwaveName* must have the same length as *waveName*.

If only /R is specified, an automatically-named wave is created with the same number of points as *waveName*. The name is based on *waveName* with "Res_" as a prefix. If a new wave needs to be created, it is automatically filled with NaN, but if the wave already exists it is simply re-used.

During fitting the residual is computed only for points that actually participate in the fit. If you fit to a subrange or use a mask wave to fit to specific points, only those points are stored. See **Residuals Using Auto Trace** on page III-219 for details.

The automatically created residual wave is appended (if necessary) to the top graph if *waveName* is graphed there. The residual wave is appended to a new free axis named by prepending "Res_" to the name of the vertical axis used for plotting *waveName*. To the extent possible, the new free axis is formatted nicely.

If the graph containing the data to be fit has very complex formatting, you may not wish to automatically append the residual to the graph. In this case, use /A=0.

If you fit to a user-defined function that returns complex values, the residual wave will also be complex.

/AR=*doAutoResid* If *doAutoResid* is 1, it is the same as /R alone. /AR is the same as /AR=1.

/W=*wghtwaveName* *wghtwaveName* contains weighting values applied during the fit, and must have the same length as *waveName*. These weighting values can be either the reciprocal of the standard errors, or the standard errors. See the /I parameter above for details.

If you fit to a user-defined function that returns complex values, the weighting wave must also be complex.

/X=*xwaveName* The X values for the data to fit come from *xwaveName*, which must have the same length and type as *waveName*.

If you are fitting to one of the two-dimensional fit functions and *waveName* is a matrix wave, *xwaveName* supplies independent variable data for the X dimension. In this case, *xwaveName* must name a 1D wave with the same number of rows as *waveName*.

When fitting to a complex-valued fit function, the X waves must be real or complex consistent with the X input or parameters declared by your fitting function.

/X={*xwave1*, *xwave2*} For fitting to one of the two-dimensional fit functions when *waveName* is a 1D wave. *xwave1* and *xwave2* must have the same length as *waveName*.

/Y=*ywaveName* For fitting using one of the 2D fit functions if *waveName* is a matrix wave. *ywaveName* must be a 1D wave with length equal to the number of columns in *waveName*.

/NWOK Allowed in user-defined functions only. When present, certain waves may be set to null wave references. Passing a null wave reference to CurveFit is normally treated as an error. By using /NWOK, you are telling CurveFit that a null wave reference is not an error but rather signifies that the corresponding flag should be ignored. This makes it easier to write function code that calls CurveFit with optional waves.

The waves affected are the X wave or waves (/X), weight wave (/W), mask wave (/M) and constraint text wave (/C). The destination wave (/D=wave) and residual wave (/R=wave) are also affected, but the situation is more complicated because of the dual use of /D and /R to mean "do autodestination" and "do autoresidual". See /AR and /AD.

If you don't need the choice, it is better not to include this flag, as it disables useful error messages when a mistake or run-time situation causes a wave to be missing unexpectedly.

Note: To work properly this flag must be the last one in the command.

Flag Parameters for Nonzero /ODR

/XW=*xWeightWave*

/XW={*xWeight1*, *xWeight2*}

/ODR=2 or 3 only.

Specifies weighting values for the independent variables using *xWeightWave*, which must have the same length as *waveName*. When fitting to one of the multivariate fit functions such as poly2D or Gauss2D, you must supply a weight wave for each independent variable using the second form.

Weighting values can be either the reciprocal of the standard errors, or the standard errors. The choice of standard error or reciprocal standard error must be the same for both /W and /XW. See /I for details.

When fitting to a complex-valued fit function, the X weighting waves must be real or complex consistent with the X parameters declared by your fitting function.

/XHLD=*holdWave*

/XHLD={*holdWave1*, *holdWave2*}

/ODR=2 or 3 only.

Specifies a wave or waves to hold the values of the independent variables fixed during orthogonal distance regression. The waves must match the input X data; a one in a wave element fixes the value of the corresponding X value.

/CMAG=*scaleWave* Specifies a wave that indicates the expected scale of the fit coefficients at the solution. If different coefficients have very different orders of magnitude of expected values, this can improve the efficiency and accuracy of the fit.

/XD=*xDestWave*

/XD={*xDestWave1*, *xDestWave2*}

/ODR=2 or 3 only.

Specifies a wave or waves to receive the fitted values of the independent variables during a least orthogonal distance regression.

When fitting to a complex-valued fit function, the X destination waves must be real or complex consistent with the X parameters declared by your fitting function.

/XR=*xResidWave*

/XR={*xResidWave1*, *xResidWave2*}

/ODR=2 or 3 only.

Specifies a wave or waves to receive the differences between fitted values of the independent variables and the starting values during a least orthogonal distance regression. That is, they will be filled with the X residuals.

When fitting to a complex-valued fit function, the X residual waves must be real or complex consistent with the X parameters declared by your fitting function.

Details

CurveFit gets initial guesses from the K*n* system variables when user guesses (/G) are specified, unless a coefficient wave is specified using the kwCWave keyword. Final curve fit parameters are written into a wave name W_coef, unless you specify a coefficient wave with the kwCWave keyword.

Other output waves are M_Covar (see the /M flag), M_FitConstraint and W_FitConstraint (see /C parameter and **Fitting with Constraints** on page III-227) and W_sigma.

For compatibility with earlier versions of Igor, the parameters are also stored in the system variables Kn. This can be a source of confusion. We suggest you think of W_coef as the **output** coefficients and Kn as **input** coefficients that get overwritten.

Other output waves are M_Covar (see the /M flag), M_FitConstraint and W_FitConstraint (see /C parameter and **Fitting with Constraints** on page III-227), W_sigma. If you have selected coefficient confidence limits using the /F parameter, a wave called W_ParamConfidenceInterval is created with the confidence intervals for the fit coefficients.

CurveFit stores other curve fitting statistics in variables whose names begin with "V_". CurveFit also looks for certain V_ variables which you can use to modify its behavior. These are discussed in **Special Variables for Curve Fitting** on page III-232.

When fitting with /ODR=nonzero, fitting with constraints is limited to simple "bound constraints." That is, you can constrain a fit coefficient to be greater than or less than some value. Constraints involving combinations of fit coefficients are supported only with /ODR=0. The constraints are entered in the same way, using an expression like `K0>1`.

Wave Subrange Details

Almost any wave you specify to CurveFit can be a subrange of a wave. The syntax for wave subranges is the same as for the Display command (see **Subrange Display Syntax** on page II-321 for details). However, the Display command allows only one dimension to have a range (multiple elements from the dimension); if a multidimensional wave is appropriate for CurveFit, you may use a range for more than one dimension.

Some waves must have the same number of points as other waves. For instance, a one-dimensional Y wave must have the same number of points as any X waves. Thus, if you use a subrange for an X wave, the number of points in the subrange must match the number of points being used in the Y wave (but see **Subrange Backward Compatibility** on page V-133 for a complication to this rule).

A common use of wave subranges might be to package all your data into a single multicolumn wave, along with the residuals and model values. For a univariate fit, you might need X and Y waves, plus a destination (model) wave and a residual wave. You can achieve all of that using a 4 column wave. For example:

```
Make/D/N=(100, 4) Data
... fill column zero with X data and column one with Y data ...
CurveFit poly 3, Data[][1] /X=Data[][0]/D=Data[][2]/R=Data[][3]
```

Note that because all the waves are full columns from a single multicolumn wave, the number of points is guaranteed to be the same.

The number of points used for X waves (*xwaveName* or {*xwave1*, *xwave2*, …}), weighting wave (*wghtwaveName*), mask wave (`maskWaveName`), destination wave (*destwaveName*) and residual wave (*residwaveName*) must be the same as the number of points used for the Y wave (*waveName*). If you specify your own confidence band waves (/F flag) they must match the Y wave; you cannot use subranges with confidence band waves. If you set /ODR = nonzero, the X weight, hold, destination and residuals waves must match the Y wave.

The total number of points in each wave does not need to match other waves, just the number of points in the specified subrange.

When fitting to a univariate fit function (that includes almost all the fit types) the Y wave must have effectively one dimension. That means the Y wave must either be a 1D wave, or it must have a subrange that makes the data being used one dimensional. For instance:

```
Make/N=(100,100) Ydata           // 2D wave
CurveFit gauss Ydata[][0]        // OK- a single column is one-dimensional
CurveFit gauss Ydata[2][]        // OK- s single row is one-dimensional
CurveFit gauss Ydata             // not OK- Ydata is two-dimensional
CurveFit gauss Ydata[][0,1]      // not OK- two columns makes 2D subrange
```

When fitting a multivariate function (**poly2D** or **Gauss2D**) you have the choice of making the Y data either one-dimensional or two-dimensional. If it is one-dimensional, then you must be fitting XYZ (or Y,X1,X2) triplets. In that case, you must provide a one-dimensional Y wave and two one-dimensional X waves, or 2 columns from a multicolumn wave. For instance:

These are OK:

```
Make/N=(100,3) myData
CurveFit Gauss2D myData[][0] /X={myData[][1],myData[][2]}
CurveFit Gauss2D myData[][0] /X=myData[][1,2]
```

These are not OK:

```
CurveFit Gauss2D myData /X={myData[][1],myData[][2]}// 2D Y wave with 1D X waves
CurveFit Gauss2D myData[][0] /X=myData            // too many X columns
```

If you use a 2D Y wave, the X1 and X2 data can come from the grid positions and the Y wave's X and Y index scaling, or you can use one-dimensional waves or wave subranges to specify the X1 and X2 positions of the grid:

```
Make/N=(20,30) yData
CurveFit Gauss2D yData        //OK- 2D Y data, X1 and X2 from scaling
Make/N=20 x1Data
Make/N=30 x2Data
// OK: 2D effective Y data, matching 1D X and Y flags
CurveFit Gauss2D yData[0,9][0,19] /X=x1Data[0,9]/Y=x2data[10,29]
// OK: effective 2D Y data
Make/N=(10,20,3) Y data
CurveFit Gauss2D yData[][][0]
```

There are, of course, lots of possible combinations, too numerous to enumerate.

Subrange Backward Compatibility

Historically, a Y wave could have a subrange. The same subrange applied to all other waves. For backward compatibility, if you use a subrange with the Y wave only, and other waves lack a subrange, these other waves must have either: 1) The same total number of points as the total number of points in the Y wave in which case the Y wave subrange will be applied; or 2) The same total number of points as the Y wave's subrange.

In addition, the Y wave can take a subrange in parentheses to indicate that the subrange refers to the Y wave's scaled indices (X scaling). If you use parentheses to specify an X range, you must satisfy the old subrange rules: All waves must have the same number of points. Subrange is allowed for the Y wave only. The Y wave subrange is applied to all other waves.

Confidence Band Details

Automatic generation of confidence and prediction bands occurs if the /F={…} parameter is used with no wave names. One to four waves are generated, or you can specify one to four wave names yourself depending on the *confKind* and *confStyle* settings.

Waves auto-generated by /F={*confLevel*, *confKind*, *confStyle*}:

confKind	*confStyle*	What You Get	Auto Wave Names
1	"Contour"	upper and lower confidence contours	UC_*dataName*, LC_*dataName*
2	"Contour"	upper and lower prediction contours	UP_*dataName*, LP_*dataName*
3	"Contour"	upper and lower confidence contours and prediction contours	UC_*dataName*, LC_*dataName*, UP_*dataName*, LP_*dataName*
1	"ErrorBar"	confidence interval wave	CI_*dataName*
2	"ErrorBar"	prediction interval wave	PI_*dataName*
3	"ErrorBar"	confidence and prediction interval waves	CI_*dataName*, PI_*dataName*

Note that *confKind* may have 4 added to it if you want coefficient confidence limits calculated as well.

The contour waves are appended to the top graph as traces if the data wave is displayed in the top graph. The wave names have *dataName* replaced with the name of the wave containing the Y data for the fit.

Waves you must supply for /F={*confLevel*, *confKind*, *confStyle*, *wave*, *wave*…}:

confKind	*confStyle*	You Supply
1	"Contour"	2 waves to receive upper and lower confidence contours.
2	"Contour"	2 waves to receive upper and lower prediction contours.
3	"Contour"	4 waves to receive upper and lower confidence and upper and lower prediction contours.
1	"ErrorBar"	1 wave to receive values of confidence band width.
2	"ErrorBar"	1 wave to receive values of prediction band width.
3	"ErrorBar"	2 waves to receive values of confidence and prediction band widths.