The waves you supply must have the same number of points as the dependent variable data wave. The band intervals will be calculated at the X values of the input data. These waves are not automatically appended to a graph; it is expected that you will display the contour waves as traces or use the error bar waves to make error bars on the model fit wave.

Residual Details

Residuals are calculated only for elements corresponding to elements of waveName that are included in the fit. Thus, you can calculate residuals automatically for a piecewise fit done in several steps.

The automatic residual wave will be appended to the top graph if the graph displays the Y data. It is appended to a new free axis positioned directly above the axis used to display the Y data, making a stacked graph. Other axes are shortened as necessary to make room for the new axis. You can alter the axis formatting later. See **Creating Stacked Plots** on page II-324 for details.

While Igor will go to some lengths to make a nicely formatted stacked graph, the changes made to the graph formatting may be undesirable in certain cases. Use /A=0 to suppress the automatic append to the graph. The automatic residual wave will be created and filled with residual values, but not appended to the graph.

Curve Fitting Screen Updates

A screen update redraws windows displaying data that has changed since the previous update, if any. Updates can take a significant amount of time, so the /N flag allows you to control them during a curve fitting operation.

Igor historically performed a screen update after each curve fitting iteration so that a graph showing the fit destination would display the latest trial solution after each iteration. This was default behavior, equivalent to /N=0.

As processors became faster, updating after every iteration became less useful because, in most cases, the time between iterations is short and the entire fit finishes quickly. Consequently, for Igor Pro 7.00, we changed the default to update only at the end of a fit, equivalent to /N=1.

There are some potential pitfalls to suppressing screen updates.

If you use the /X flag to tell Igor to extrapolate the fit curve to the entire X range of a graph, and if your procedures alter the X axis range, you need to call DoUpdate before calling CurveFit or FuncFit to allow Igor to finalize the change to the axis range.

If you call CurveFit or FuncFit from a user-defined function and use the /D flag, which turns on the auto-destination feature, then the fit curve will not update until all running functions in the call chain return. If it is important that the graph displaying the destination wave reflect the fit result before functions return, you must call DoUpdate.

See Also

Inputs and Outputs for Built-In Fits on page III-212 and **Special Variables for Curve Fitting** on page III-232 as well as **Accessing Variables Used by Igor Operations** on page IV-123.

When fitting to a user-specified function, see **FuncFit**. For multivariate user-specified fitting functions, see **FuncFit** and **FuncFitMD**. See **Confidence Bands and Coefficient Confidence Intervals** on page III-223 for a detailed description of confidence and prediction bands.

References

An explanation of the Levenberg-Marquardt nonlinear least squares optimization can be found in Chapter 14.4 of Press, William H., *et al.*, *Numerical Recipes in C*, 2nd ed., 994 pp., Cambridge University Press, New York, 1992.

CustomControl

`CustomControl` `[`**`/Z`**`]` ***`ctrlName`*** `[`***`keyword = value`*** `[,` ***`keyword = value`*** `…]]`

The CustomControl operation creates or modifies a custom control in the target window. A CustomControl starts out as a generic button, but you can customize both its appearance and its action.

For information about the state or status of the control, use the **ControlInfo** operation.

Parameters

ctrlName is the name of the CustomControl to be created or changed. See **Button** for standard default parameters.

The following keyword=value parameters are supported:

align=*alignment*

Sets the alignment mode of the control. The alignment mode controls the interpretation of the *leftOrRight* parameter to the pos keyword. The align keyword was added in Igor Pro 8.00.

If *alignment*=0 (default), *leftOrRight* specifies the position of the left end of the control and the left end position remains fixed if the control size is changed.

If *alignment*=1, *leftOrRight* specifies the position of the right end of the control and the right end position remains fixed if the control size is changed.

fColor=(*r*,*g*,*b*[,*a*])

Sets color of the button only when picture is not used and frame=1.

r, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**.

focusRing=*fr*

Enables or disables the drawing of a rectangle indicating keyboard focus:

fr=0:	Focus rectangle will not be drawn.
fr=1:	Focus rectangle will be drawn (default).

On Macintosh, regardless of this setting, the focus ring appears if you have enabled full keyboard access via the Shortcuts tab of the Keyboard system preferences.

frame=*f*

Sets frame style used only when picture is not used:

f=0:	No frame (only the title is drawn).
f=1:	Default, a button is drawn with a centered title. Set fColor to something other than black to colorize the button.
f=2:	Simple box.
f=3:	3D sunken frame. On Macintosh, when "native GUI appearance" is enabled for the control, the frame is filled with the proper operating system color.
f=4:	3D raised frame.
f=5:	Text well.

help={*helpStr*}

Sets the help for the control.

helpStr is limited to 1970 bytes (255 in Igor Pro 8 and before).

You can insert a line break by putting "\r" in a quoted string.

labelBack=(*r*,*g*,*b*[,*a*]) or 0

Sets background color for the control only when a picture is not used and frame is not 1 and is not 3 on Macintosh.

r, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**. If not set (or labelBack=0), then background is transparent (not erased).

mode=*m*

Notifies the control that something has happened. Can be used for any purpose. See **Details** discussion of the `kCCE_mode` event.

noproc

Specifies that no procedure will execute when clicking the custom control.

picture= *pict*

Uses the named Proc Pictures to draw the control. The picture is taken to be three side-by-side frames, which show the control appearance in the normal state, when the mouse is down, and in the disabled state.

The control action function can overwrite the picture number using the picture={*pict*,*n*} syntax.

The picture size overrides the size keyword.

picture={*pict*,*n*}	Uses the specified Proc Picture to draw the control. The picture is *n* side-by-side frames instead of the default three frames.
pos={*leftOrRight*,*top*}	Sets the position in **Control Panel Units** of the top/left corner of the control if its alignment mode is 0 or the top/right corner of the control if its alignment mode is 1. See the align keyword above for details.
pos+={*dx*,*dy*}	Offsets the position of the control in **Control Panel Units**.
proc=*procName*	Specifies the name of the action function for the control. The function must not kill the control or the window.
size={*width*,*height*}	Sets size of the control in **Control Panel Units** but only when not using a Proc Picture.
title=*titleStr*	Specifies text that appears in the control. Using escape codes you can change the font, size, style, and color of the title. See **Annotation Escape Codes** on page III-53 or details.
userdata(*UDName*)=*UDStr*	Sets the unnamed user data to *UDStr*. Use the optional (*UDName*) to specify a named user data to create.
userdata(*UDName*)+=*UDStr*	Appends *UDStr* to the current unnamed user data. Use the optional (*UDName*) to append to the named *UDStr*.
value=*varName*	Sets the numeric variable, string variable, or wave that is associated with the control. With a wave, specify a point using the standard bracket notation with either a point number (`value=awave[4]`) or a row label (`value=awave[%alabel]`).
valueColor=(*r*,*g*,*b*[, *a*])	Sets initial color of the title for the button drawn only when picture is not used and frame=1. *r*, *g*, *b*, and *a* specify the color and optional opacity as **RGBA Values**. The default is opaque black. To further change the color of the title text, use escape sequences as described for title=*titleStr*.

Flags

/Z	No error reporting.

Details

When you create a custom control, your action procedure gets information about the state of the control using the `WMCustomControlAction` structure. See **WMCustomControlAction** for details on the `WMCustomControlAction` structure.

Although the return value is not currently used, action procedures should always return zero.

When Igor calls your action procedure for the `kCCE_draw` event, the basic button picture (custom or default) will already have been drawn. You can use standard drawing commands such as **DrawLine** to draw on top of the basic picture. Unlike the normal situation when drawing commands merely add to a drawing operation list which only later is drawn, `kCCE_draw` event drawing commands are executed directly. The coordinate system, which you can not change, is **Control Panel Units** with (0,0) being the top left corner of the control. Most drawing commands are legal but because of the immediate nature of drawing, the /A (append) flag of **DrawPoly** is not allowed.

The `kCCE_mode` event, which Igor sends when you execute CustomControl with the mode keyword, can be used for any purpose, but it mainly serves as a notification to the control that something has happened. For example, to send information to a control, you can set a named (or the unnamed) userdata and then set the `mode` to indicate that the control should examine the userdata. For this signaling purpose, you should use a `mode` value of 0 because this value will not become part of the recreation macro.