/Z	Errors are not fatal. Will not abort procedure execution if an error occurs. Your procedure can inspect the V_flag variable to see if the transfer succeeded. V_flag will be zero if it succeeded, -1 if the user canceled in an interactive dialog, or another nonzero value if an error occurred.

Examples

Upload a file using a full local path:

```
String url = "ftp://ftp.wavemetrics.com/pub/test/TestFile1.txt"
String localPath = "hd:Test Folder:TestFile1.txt"
FTPUpload url, localPath
```

Upload a file using a local symbolic path and file name:

```
String url = "ftp://ftp.wavemetrics.com/pub/test/TestFile1.txt"
String pathName = "Igor"      // Igor is the name of a symbolic path.
String fileName = "TestFile1.txt"
FTPUpload/P=$pathName url, fileName
```

Upload a directory using a full local path:

```
String url = "ftp://ftp.wavemetrics.com/pub/test/TestDir1"
String localPath = "hd:Test Folder:TestDir1"
FTPUpload/D url, localPath
```

See Also

File Transfer Protocol (FTP) on page IV-272.

FTPCreateDirectory, **FTPDelete**, **FTPDownload**, **URLEncode**.

FuncFit

`FuncFit` `[`***`flags`***`]` ***`fitFuncName`***, ***`cwaveName`***, ***`waveName`*** `[`***`flag parameters`***`]`

`FuncFit` `[`***`flags`***`]` **`{`*`fitFuncSpec`*`}`**, ***`waveName`*** `[`***`flag parameters`***`]`

The FuncFit operation performs a curve fit to a user defined function, or to a sum of fit functions using the second form (see **Fitting Sums of Fit Functions** on page V-275). Fitting can be done using any method that can be selected using the /ODR flag (see **CurveFit** for details).

FuncFit operation parameters are grouped in the following categories: flags, parameters (*fitFuncName*, *cwaveName*, *waveName* or {*fitFuncSpec* }, *waveName*), and flag parameters. The sections below correspond to these categories. Note that flags must precede the *fitFuncName* or `fitFuncSpec` and flag parameters must follow *waveName*.

In Igor Pro 9.00 and later, your fitting function can return complex values. The sections below indicate requirements for complex fitting functions. For details, see **Fitting with Complex-Valued Functions** on page III-248.

Flags

See **CurveFit** for all available flags.

Parameters

fitFuncName	The user-defined function to fit to, which can be a function taking multiple independent variables (see also **FuncFitMD**). Multivariate fitting with FuncFit *requires* /X=*xwaveSpec*. To fit complex data waves, your fitting function must be declared complex (Function/C).
cwaveName	Wave containing the fitting coefficients. If the fitting function is declared with a complex wave for the coefficient wave, then this wave must be complex.
waveName	The wave containing the dependent variable data to be fit to the specified function. For functions of just one independent variable, the dependent variable data is often referred to as "Y data". You can fit to a subrange of the wave by supplying (*startX*,*endX*) or [*startP*,*endP*] after the wave name. See **Wave Subrange Details** below for more information on subranges of waves in curve fitting. If your fitting function returns complex values then this wave must be complex.

fitFuncSpec	List of fit functions and coefficient waves, with some optional information. Using this format fits a model consisting of the sum of the listed fit functions. Intended for fitting multiple peaks, but probably useful for other applications as well. See **Fitting Sums of Fit Functions** on page V-275. Does not support complex fitting functions.

Flag Parameters

These flag parameters must follow *waveName*.

/*E=ewaveName*
A wave containing the epsilon values for each parameter. Must be the same length as the coefficient wave. If your fitting function requires a complex coefficient wave then your epsilon wave must be complex. Not supported with complex fitting functions.

/STRC=*structureInstance*

Used only with **Structure Fit Functions** on page III-261. When using a structure fit function, you must specify an instance of the structure to FuncFit. This will be an instance that has been initialized by a user-defined function that you write in order to invoke FuncFit.

/X=*xwaveSpec*
An optional wave containing X values for each of the input data values. If the fitting function has more than one independent variable, *xwaveSpec* is required and must be either a 2D wave with a column for each independent variable, or a list of waves, one for each independent variable. A list must be in braces: /X={*xwave0*, *xwave1*,…}. There must be exactly one column or wave for each independent variable in the fitting function. If your fitting function takes complex X values then your X waves must be complex.

/NWOK
Allowed in user-defined functions only. When present, certain waves may be set to null wave references. Passing a null wave reference to FuncFit is normally treated as an error. By using /NWOK, you are telling FuncFit 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 FuncFit with optional waves.

The waves affected are the X wave or waves (/X), weight wave (/W), epsilon wave (/E) and mask wave (/M). 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.

Other parameters are used as for the **CurveFit** operation, with some exceptions for multivariate fits.

Details for Multivariate Fits

The dependent variable data wave, *waveName*, must be a 1D wave even for multivariate fits. For fits to data in a multidimensional wave, see **FuncFitMD**.

For multivariate fits, the auto-residual (/R with no wave specified) is calculated and appended to the top graph if the dependent variable data wave is graphed in the top graph as a simple 1D trace. Auto residuals are calculated but not displayed if the data are displayed as a contour plot.

The autodest wave (/D with no wave specified) for multivariate fits has the same number of points as the data wave, with a model value calculated at the X values contained in the wave or waves specified with /X=*xwaveSpec*.

Confidence bands are not supported for multivariate fits.

Wave Subrange Details

Almost any wave you specify to FuncFit 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). See **Wave Subrange Details** on page V-132 for a discussion of the use of subranges in curve fitting.

The backwards compatibility rules for CurveFit apply to FuncFit as well.

In addition to the waves discussed in the CurveFit documentation, it is possible to use subranges when specifying the coefficient wave and the epsilon wave. Since the coefficient wave and epsilon wave must have the same number of points, it might make sense to make them two columns from a single multicolumn wave. For instance, here is an example in which the first column is used as the coefficient wave, the second is used as the epsilon wave, and the third is used to save a copy of the initial guesses for future reference:

```
Make/D/N=(5, 3) myCoefs
myCoefs[][0] = {1,2,3,4,5}            // hypothetical initial guess
myCoefs[][1] = 1e-6                   // reasonable epsilon values
myCoefs[][2] = myCoefs[p][0]          // save copy of initial guess
FuncFit myFitFunc, myCoefs[][0] myData /E=myCoefs[][1] …
```

You might have a fit function that uses a subset of the coefficients that are used by another. It might be useful to use a single wave for both. Here is an example in which a function that takes four coefficients is used to fit a subset of the coefficients, and then that solution is used as the initial guess for a function that takes six coefficients:

```
Make/D/N=6 Coefs6={1,2,3,4,5,6}
FuncFit Fit4Coefs, Coefs6[0,3] fitfunc4Coefs …
FuncFit Fit6Coefs, Coefs6 ...
```

Naturally, the two fit functions must be worked out carefully to allow this.

Fitting Sums of Fit Functions

If Igor encounters a left brace at the beginning of the fit function name, it expects a list of fit functions to be summed during the fit. This is useful for, for instance, fitting several peaks in a data set to a sum of peak functions.

The fit function specification includes at least the name of the fitting function and an associated coefficient wave. A sum of fit functions requires multiple coefficient waves, one for each fit function. Any coefficient wave-related options must be specified in the fit function specification via keyword-value pairs.

The syntax of the sum-of-fit-functions specification is as follows:

`{{`*`func1`*`, `*`coef1`*`, `*`keyword`*`=`*`value`*`},{`*`func2`*`, `*`coef2`*`, `*`keyword`*`=`*`value`*`}, …}`

or

`{`*`string=fitSpecStr`*`}`

Within outer braces, each fit function specification is enclosed within inner braces. You can use one or more fit function specifications, with no intrinsic limit on the number of fit functions.

The second format is available to overcome limitations on the length of a command line in Igor. This format is just like the first, but everything inside the outer braces is contained in a string expression (which may be just a single string variable).

You can use any fit function that can be used for ordinary fitting, including the built-in functions that are available using the CurveFit operation. If you should write a user-defined fitting function with the same name as a built-in fit function, the user-defined function will be used (this is strongly discouraged).

Every function specification must include an appropriate coefficient wave, pre-loaded with initial guesses.

The comma between each function specification is optional.

The keyword-value pairs are optional, and are used to communicate further options on a function-by-function basis. Available keywords are:

HOLD=*holdstr*	Indicates that a fit coefficient should be held fixed during fitting. *holdstr* works just like the hold string specified via the /H flag for normal fitting, but applies only to the coefficient wave associated with the fit function it appears with. If you include HOLD in a string expression (the `{string=fitSpecStr}` syntax), you must escape the quotation marks around the hold string. If you use the command-line syntax `{{func1,coef1,HOLD=`*`holdStr`*`}, ...}`, *holdStr* may be a reference to a global variable acquired using SVAR, or it may be a quoted literal string. If you use `{string=`*`fitSpecStr`*`}`, *fitSpecStr* is parsed at run-time outside the context of any running function. Consequently, you cannot use a general string expression. You can use either `HOLD="quotedLiteralString"` or `HOLD=root:globalString`.