Structure Fit Functions

Sometimes you may need to transmit extra information to a fitting function. This might include constants that need to be set to reflect conditions in a run, but should not be fit; or a wave containing a look-up table or a measured standard sample that is needed for comparison. Perhaps your fit function is very complex and needs some sort of book-keeping data.

If you use a basic fit function or an all-at-once function, such information must be transmitted via global variables and waves. That, in turn, requires that this global information be looked up inside the fit function. The global data requirement makes it difficult to be flexible: the fit function is tied to certain global variables and waves that must be present for the function to work. In addition to adding complexity and difficulty to management of global information, it adds a possible time-consuming operation: looking up the global information requires examining a list of waves or variables, and comparing the names to the desired name.

Structure fit functions are ideal for such problems because they take a single parameter that is a structure of your own design. The first few members of the structure must conform to certain requirements, but the rest is up to you. You can include any kind of data in the structure. An added bonus is that you can include members in the structure that identify for the fit function which fit coefficient is being perturbed when calculating numerical derivatives, that signal when the fit function is being called to fill in the auto-destination wave, and identify when a new fit iteration is starting. You can pass back a flag to have FuncFit abandon fitting.

To use a structure fit function, you must do three things:

1. Define a structure containing certain standard items at the top.
2. Write a fitting function that uses that structure as its only parameter.
3. Write a wrapper function for FuncFit that creates an instance of your structure, initializes it and invokes FuncFit with the /STRC parameter flag.

You should familiarize yourself with the use of structures before attempting to write a structure fit function (see **Structures in Functions** on page IV-99).

Structure fit functions come in basic and all-at-once variants; the difference is determined by the members at the beginning of the structure. The format for the structure for a basic structure fit function is:

```
Structure myBasicFitStruct
   Wave coefw
   Variable x
   …
EndStructure
```

The name of the structure can be anything you like that conforms to Igor's naming rules; all that is required is that the first two fields be a wave and a variable. By convention we name the wave coefw and the variable x to match the use of those members. These members of the structure are equivalent to wave and variable parameters required of a basic fit function.

You may wish to use an all-at-once structure fit function for the same reason you might use a regular all-at-once fit function. The same concerns apply to all-at-once structure fit functions; you should read and understand **All-At-Once Fitting Functions** on page III-256 before attempting to write an all-at-once structure fit function.

The structure for an all-at-once structure fit function is:

```
Structure myAllAtOnceFitStruct
   Wave coefw
   Wave yw
   Wave xw
   …
EndStructure
```

The first three members of the structure are equivalent to the pw, yw, and xw parameters required in a regular all-at-once fit function.

A simple example of fitting with a structure fit function follows. More examples are available in the File menu: select appropriate examples from Examples→Curve Fitting.

Basic Structure Fit Function Example

As an example of a basic structure fit function, we will write a fit function that fits an exponential decay using an X offset to compensate for numerical problems that can occur when the X range of an exponential function is small compared to the X values. The X offset must not be a fit coefficient because it is not mathematically distinguishable from the decay amplitude. We will write a structure that carries this constant as a custom member of a structure. This function will duplicate the built-in exp_XOffset function (see **Built-in Curve Fitting Functions** on page III-206).

First, we create fake data by executing these commands:

```
Make/D/O/N=100 expData,expDataX
expDataX = enoise(0.5)+100.5
expData = 1.5+2*exp(-(expDataX-100)/0.2) + gnoise(.05)
Display expData vs expDataX
ModifyGraph mode=3,marker=8
```

Now the code. Copy this code and paste it into your Procedure window:

```
// The structure definition
Structure expFitStruct
   Wave coefw           // required coefficient wave
   Variable x           // required X value input
   Variable x0          // constant
EndStructure

// The fitting function
Function fitExpUsingStruct(s) : FitFunc
   Struct expFitStruct &s

   return s.coefw[0] + s.coefw[1]*exp(-(s.x-s.x0)/s.coefw[2])
End

// The driver function that calls FuncFit:
Function expStructFitDriver(pw, yw, xw, xOff)
   Wave pw     // coefficient wave- pre-load it with initial guess
   Wave yw
   Wave xw
   Variable xOff
   Variable doODR

   // An instance of the structure. We initialize the x0 constant only,
   // Igor (FuncFit) will initialize coefw and x as required.
   STRUCT expFitStruct fs
   fs.x0 = xOff   // set the value of the X offset in the structure

   FuncFit fitExpUsingStruct, pw, yw /X=xw /D /STRC=fs

   // no history report for structure fit functions. We print our own
   // simple report here:
   print pw
   Wave W_sigma
   print W_sigma
End
```

Finally, make a coefficient wave loaded with initial guesses and invoke our driver function:

```
Make/D/O expStructCoefs = {1.5, 2, .2}
expStructFitDriver(expStructCoefs, expData, expDataX, 100)
```