User Macros in PAN

Rob Dimeo
(Updated by Larry Kneller, 09/16/09, 12/03/09, 02/19/10)

This document describes the functionality in PAN in which the user can type in customized macros in IDL syntax for fitting their data.
(See note below on integer usage in macros.)

The user macro interface

Once data is loaded into PAN then you can select the user macro function selection from the select function drop-down menu as shown in figure 1. Note that you can define and use as many fit macros as you wish for your overall fit function.

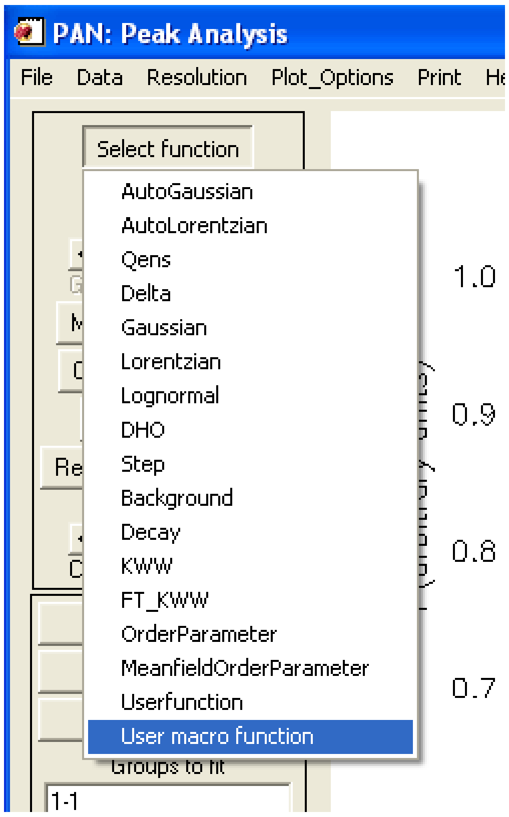
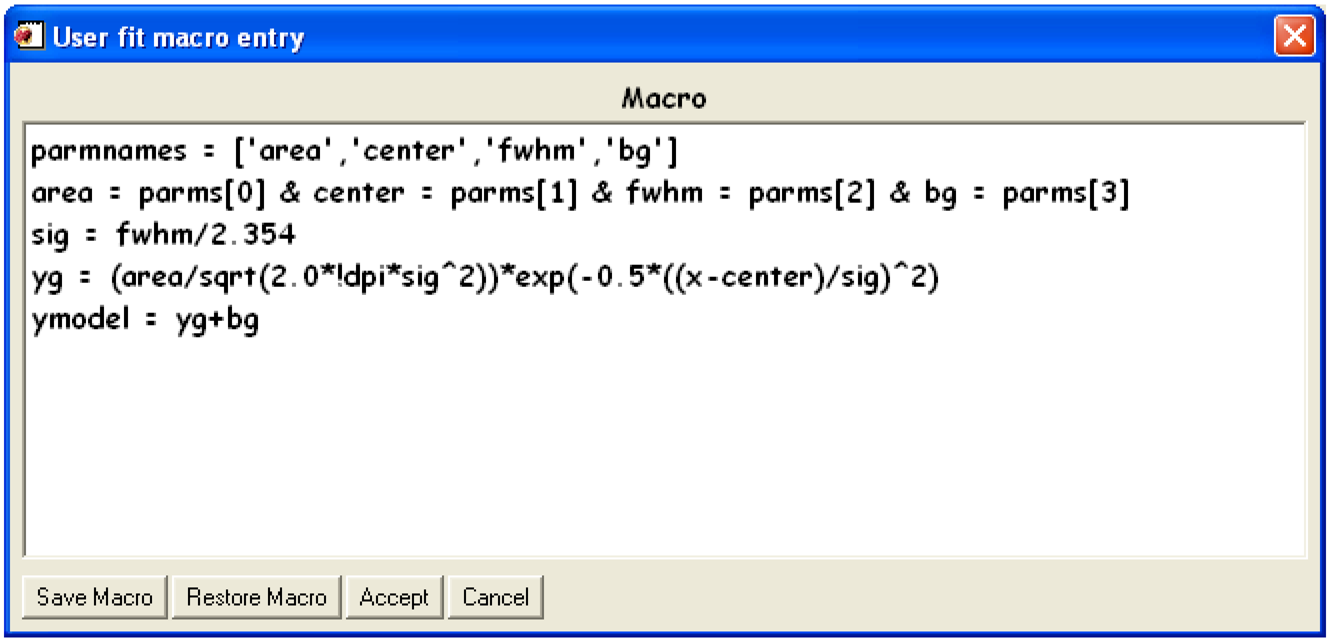


Fig. 1 User macro function selection from the select-function menu.

The macro entry interface is then launched as shown in figure 2. A default function is loaded in automatically in order to illustrate the required syntax for the user macro. In the default case the macro defines a Gaussian plus a flat background. You can save your custom macros or you can restore previously-saved macros using the buttons at the bottom of the macro entry interface. Note that if a resolution function is loaded into

PAN then the default behavior is to fit the result of the convolution of your fit function with the resolution function to the data.



```
parmnames = ['area','center','fwhm','bg']
area = parms[0] & center = parms[1] & fwhm = parms[2] & bg = parms[3]
sig = fwhm/2.354
yg = (area/sqrt(2.0*!dpi*sig^2))*exp(-0.5*((x-center)/sig)^2)
ymodel = yg+bg
```

Fig. 2 The user macro function entry interface.

Macro Requirements

The requirements for the macros are as follows:

1. The macro must begin with a string array defining the parameter names. You can be as creative as you like with the names but the array must adhere to IDL syntax. The parameter names for the Gaussian are defined as follows:

```
parmnames = ['area','center','fwhm','bg']
```

2. The parameter values are stored in a vector called parms and must be referenced as such. The parameter values must be stored in the same order as the parameter names in the parmnames array, or,

```
parms = [area,center,fwhm,bg]
```

In IDL syntax this is the same thing as:

```
area = parms[0]
center = parms[1]
fwhm = parms[2]
 bg = parms[3]
```

3. The independent variable is always x.
4. The value of each group q value can be referenced as Qgroup. When the macro is executed the current Q value is substituted anywhere this variable is present.
5. The output for the function which does not include δ-functions must be a variable named ymodel. This is the last line in the code shown in fig. 2. See point 6 below for how to incorporate δ-functions into your fit function.
6. The code must be written like an IDL macro. This means that you have access to all of the high level functionality of IDL but you are limited to a single executable line within a control loop. Examples of macros are shown in the next section.
7. When using δ-functions in your fit function the resolution function must be loaded in. Furthermore the δ-functions are specified by an array of areas and centers using the variable name delta_params. For example, to represent the following equation,

$$y_{deltas} = a_1\delta(x - c_1) + a_2\delta(x - c_2),$$

you would enter the following

delta_params = [[a_1,c_1],[a_2,c_2]].

Special note on δ-functions: Note that when δ-functions are used in a macro, no other components of the macro will be evaluated. This special treatment of the delta function is done to prevent double-convolution of the resolution function during evaluation. Thus, if you want to include delta functions and other functions in your model using a macro, then you must either add each δ-function separately or include the deltas in one or more separate macro functions.

Macro Examples

The syntax for the macros is standard IDL syntax. Therefore you may want to consult the IDL help for more information on the functionality available.

Sloping line (simple):

$$y_{model} = b + m \cdot x$$

```
parmnames = ['offset','slope']
offset = parms[0] & slope = parms[1]
ymodel = offset+slope*x
```

Note that you could have also written this more compactly as:

```
parmnames = ['offset','slope']
ymodel = parms[0]+parms[1]*x
```

Quasielastic lineshape (with Gaussian resolution):

$$y_{model} = eisf \bullet \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{x-x_o}{\sigma}\right)^2\right) + (1-eisf)\bullet \frac{\Gamma}{\pi}\frac{1}{(x-x_o)^2+\Gamma^2}$$

```
parmnames = ['amp','eisf','gcenter','gfwhm','lfwhm']
amp = parms[0] & eisf = parms[1] & center = parms[2] & gfwhm = parms[3]
lfwhm = parms[4]
sig = gfwhm/2.354
gamma = lfwhm/2.0
yg = (1.0/sqrt(2.0*!dpi*sig^2))*exp(-0.5*((x-center)/sig)^2)
yl = (gamma/!pi)/((x-center)^2+gamma^2)
ymodel = amp*(eisf*yg+(1.0-eisf)*yl)
```

Diffusion within a plane (somewhat more complicated):

$$y_{model} = \frac{1}{\pi}\frac{B}{x^2+B^2}\bullet K$$

$$K = 1+\sum_{n=1}^{\infty}\frac{1}{2n+1}\cos[(n+1)\varphi], \quad \tan\varphi = \frac{x}{B}$$

```
parmnames = ['amp','width']
amp = parms[0] & b = parms[1]
nx = n_elements(x)
nn = 500
nindex = 1.0+findgen(nn)
phi = atan(x/b)
ux = 1+bytarr(nx)
t1 = (1.0/(2.0*nindex+1.0))#ux
t2 = (cos((nindex+1.0)#phi))
k = 1.0+total(t1*t2,1)
ymodel = amp*((k*b/!pi)/(x^2+b^2))
```

QENS lineshape (with resolution function):

$$y_{model} = area\left[eisf \bullet \delta(x) + (1-eisf)\bullet \frac{\Gamma}{\pi}\frac{1}{x^2+\Gamma^2}\right]\otimes R(x)$$

```
parmnames = ['area','eisf','fwhm','bg']
area = parms[0] & eisf = parms[1] & fwhm = parms[2]
bg = parms[3]
gamma = 0.5*fwhm
ylor = area*(1.0-eisf)*(gamma/!dpi)/(x^2+gamma^2)+bg
delta_params = [[area*eisf,0.0]]
ymodel = ylor
```

Notes/Warnings on integer usage and IDL Reserved words in macro and function definitions:

1) When defining a "User Function" or "User Macro", take care not to use integer arithmetic unless you intend to do so. For example, use 0.5 instead of $\frac{1}{2}$ and 0.25 instead of $\frac{1}{4}$. The ratios $\frac{1}{2}$ and $\frac{1}{4}$ will be evaluated by IDL using integer arithmetic and yield zero (0) as the result in each case. This may effectively eliminate part of your function or macro definition.

2) Also, when defining a "User Function" or "User Macro", you may not use IDL reserved words as variable names. This will produce a syntax error message, and it may be very confusing to track down the issue. The IDL reserved words are shown here in uppercase, but IDL is not case sensitive:
AND, BEGIN, BREAK, CASE, COMMON, COMPILE_OPT, CONTINUE, DO, ELSE, END, ENDCASE, ENDELSE, ENDFOR, ENDIF, ENDREP, ENDSWITCH, ENDWHILE, EQ, FOR, FORWARD_FUNCTION, FUNCTION, GE, GOTO, GT, IF, INHERITS, LE, LT, MOD, NE, NOT, OF, ON_IOERROR, OR, PRO, REPEAT, SWITCH, THEN, UNTIL, WHILE, XOR.
(While most of these words are unlikely to appear as variables, here are some possible reasonable examples of this type of bug:
If = Io*exp(-x/Do), Do for D_zero, Lt for Lorentzian, Of for Ofinal, repeat for a repeat spacing, etc.)

Using PAN's Built-in Fit Functions

You can use the functions that are already built-in to PAN in your own user macro-function. Simply find the name of the function from the drop-down menu and add the prefix pan_ to get the function name. For instance, say you want to call the Lorentzian. Then you would call the function pan_lorentzian. The arguments for the function are the independent variable,x, and the parameters, parms. The order for the parameters for both the Gaussian and Lorentzian are [area,center, full-width at half maximum].

The following example shows how you can use functions already built-in to PAN to create your own composite user macro function:

Quasielastic lineshape (with Gaussian resolution):

$$y_{\text{model}} = \text{eisf} \bullet \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{x-x_o}{\sigma}\right)^2\right) + (1-\text{eisf}) \bullet \frac{\Gamma}{\pi} \frac{1}{(x-x_o)^2+\Gamma^2}$$

parmnames = ['amp','eisf','gcenter','gfwhm','lfwhm']

```
amp = parms[0] & eisf = parms[1] & center = parms[2] & gfwhm = parms[3]
lfwhm = parms[4]
yg = pan_gaussian(x,[1.0,center,gfwhm])
yl = pan_lorentzian(x,[1.0,center,lfwhm]
ymodel = amp*(eisf*yg+(1.0-eisf)*yl)
```