HELP PAGES FOR PLOTTING TAS DATA WITH NPLOT

With examples

General principle

- nplot analyses the scan command and uses the scanned variable as x-axis by default
- Several scans can be combined and plotted at once
- A number of command line options and switches control the behavior of nplot
- A fitting procedure ("nfit") is integrated. New functions can be created by the user.
- Nplot handles polarized data, as well as data from Flatcone or Imps
- if several variables are scanned, or explicitly chosen, nplot works with arbitrary number of dimensions (for instance, by default [qh,qk,ql,en] for dqh-scans). When combining scans (i.e. for binning, rejection of too distant points, etc.) *all* of these coordinates are taken into account.
- a call to nplot returns a data structure that can be used by other Matlab code

Using nplot

nplot is launched as a single line command from the Matlab command line or a script. In many cases it is sufficient to only give the name of one or several files, nplot then makes assumptions which in most cases will produce a reasonable output. To manually control the behavior of nplot, there are a great number of parameters and switches. Parameters are pairs (name + value), and switches are single keywords. Most of them can be combined in arbitrary number and order.

Syntax

Parameter names and switches are strings, parameter values may be strings, numbers, arrays, etc. depending on the case. They can however be given as strings, nplot will convert in numeric or other format where necessary. Also, the parameter pairs (name, value) can be provided as a single or as two separate strings. Therefore, the following produce the same output:

```
nplot('012345','time',1)
nplot('012345','time','1')
nplot('012345','time 1')
```

Simplified syntax:

When the return values of nplot are ignored, the simplified syntax without parentheses and quotes, which makes typing much more comfortable, can be used:

```
nplot 012345 time 1
```

Note that the command line arguments are separated by spaces; you can use arguments containing space characters by enclosing them in quotes (e.g. `nplot 012345 legend 'data description text'`).

In subsequent calls of nplot, you can use '..' to reuse the full parameter list of the previous call to nplot, which avoids retyping the full command line. That means, after `nplot 012345 time 1`
the command
`nplot 012346 ..`
is equivalent to
`nplot 012346 time 1`

Options and switches

Normalization

By default, data are normalized like the specified in the scan command (first scan in the list if several data files combined). To specify normalization use

```
‘Time’, ti    :  normalize to time ti
‘Monitor’, mn :  normalize to monitor mn
```

Example:

```
nplot 0123[45,46] time 10
nplot 0123[45,46] monitor 1000
```

If plotting several data sets into the same axes by separate calls to nplot, the y-axis label corresponds to the last call of nplot. In some cases of inconsistency a warning is issued, but in general the user has to make sure the normalizations are consistent.

Stepsize, binning, combination, etc

```
'step', [s1, s2, ..] :   Stepsize for binning. (Default: half stepsize of first scan.)
'start', [v1, v2, ..] :  Start point
'end', ... :             End point
'maxdist', [m1, m2, ..]: Maximum distance of points to scanline (discard points with
                         larger distance). In case of more than one coordinate, the
                         scan path is defined either as the connecting line of start
                         and end point (if both given) or by the step size.
‘nobin’:                 Do not perform any binning.
```

For a single scan, by default no binning etc. is performed.
If several scans are combined, by default the first scan defines the stepsize. In case of coordinate with more than one dimension, the stepsize is a vector with n components, and binning and rejection of too distant points take into account all components. This allows to combine scans which run not along the same paths (for instance, Q- and E-scans) taking into account only points that match the resulting scan.
As an example, assume that `000001` is sc qh 2 0 0 0 dqh 0 0 0 1, and `000002` is sc qh 2 0 0 0 dqh .1 0 0 0 :

(1) `nplot 00000[1,2]` produces an Energy scan with the point at (2,0,0,0) averaged from both scans, and the other points from `000002` rejected.
(2) `nplot 00000[2,1]` produces a Q-Scan accordingly.
(3) `nplot 00000[1,2] var EN` uses *only* EN as a coordinate; therefore all points from `000002` have the same coordinate 0 and are averaged to the resulting point at (2,0,0,0)
(4) `nplot 00000[1,2] maxdist [.5, .1, .1, .1]` is like (1), but includes points from `000002` up to a difference 0.5 in qh, i.e. between 1.5 and 2.5 into the resulting point at (2,0,0,0)

Likewise, with `start`, `end`, and `step` you can explicitly define a scan path in n dimensions, and all points within `maxdist` will be binned and averaged to give the resulting data set, while all further points are rejected. The typical application for this is the case when you have a set of Q- and E-scans, and when plotting one scan the equivalent points of other scans (at the crossing points) are taken into account, or when for instance from a set of Q-scans at different dense enough energies an energy scan is to be extracted.

The options start and end may also simply be used to limit the extension of the scan, like in one dimension:

```
nplot 012345 var a4 start 70.4 end 72.1
```

Fitting

You can use fitting within the command line of nplot, or call *nfit* afterwards on an existing nplot window.

```
'fit', 'funcname':        Fit function 'funcname' to data.
'showfit':                Write Fit results in the graphics window
```

```
All parameters which are accepted by nfit can be used (e.g. constraints, start values,
etc). See nfit documentation for help.
```

Examples:

```
nplot 012345 fit gauss
nplot 012345 fit gauss2
nplot 012345 fit fsum(@const,@gaussA,@gaussA) startval [0,5.1,100,1,7.2,50,1]
nplot 012345 fit lorentz2 constraint p2=-p5;p4=p7   (makes x_c^(1) = - x_c^(2) and width^(1)=width^(2))
nplot 012345 fit linear startval [0,3] fitvar [0,1]
[~,fitresult]=nplot('012345','fit linear','noplot');
```

Variables and axes

```
'var', {'v1','v2',..}:  which variables to use as coordinates. If not given, take the
                        scanned variables (of first scan).
'xvar', 'varname':      Variable for x-axis in the plot.
'yvar', 'varname':      If you want to plot sth else than CNTS, give the column name
                        here. You can also plot values of zeros., param., or varia.-
                        sections of the file header. If this option is used, NO
                        normalization and averaging is done.
'offset', offset:       Shift plot on y-axis by [offset]
'xtransform',expr:      Transform x-coordinate by expr, for example expr = '2*x+0.1'
                        (valid Matlab expression)
'ytransform',expr:      Transform y-coordinate by expr, for example expr = 'log(y)+1,
                        expr='y.^2+sqrt(y)', etc.
```

As variable names, you can use any column from the data file(s). When you use another one than CNTS on the y-axis, the y error bars are set to zero.
The options `var` and `xvar` will often have the same effect, unless you work with coordinates in more than one dimension.
There are a few special cases: for Flatcone data, you can use `twotheta` as a variable, which corresponds to the physical 2theta angle of the regarded channel. The variable `realtime` plots date and time on the x-axis.
In addition to columns from the data file, constants from the ZEROS, PARAM, and VARIA section of the data file can be used as y-values (like "zeros.a3", "param.sa", etc).

Examples:

```
nplot 012345 ytransform log(y)
nplot 012345 offset 10   is like   nplot 012345 ytransform y+10
nplot 012345 xvar qk     is like   nplot 012345 var qk
```
(When combining several scans, it can be different; see examples in "binning/combination").
```
nplot 012345 var thwotheta
```
(if powder a4-scan with Flatcone, full powder diffractogram of all channels)
```
nplot 012[300:400] xvar realtime yvar param.ss
```

Options for polarized neutrons

When the data file contains a POLAN section (a pal-file has been used), nplot automatically treats the different polarization separately from each other. It also keeps record of the definition of each channel, and correctly combines equivalent channels of different data files even if the order or the number of pal-channels are not the same in all files.

```
'only', which :              retain only selected pal-states.
'calc', 'a*pal1+b*pal2..':   Calculate linear combination of pal-sets.
```

Examples:

`nplot 012345 only pal[1:3,5]` plots only pal-channels 1,2,3 and 5
`nplot 012345 calc pal3+pal4-2*pal2` the resulting data set is the indicated linear combination of pal's.

Options for Flatcone data

```
'only', which :     retain only selected channels
```

Examples:

```
nplot 012345 only chan10
nplot 012345 only chan[10:15,20:25] var twotheta
```

Options for Imps data

```
'reintegrateimps', rois: Reintegrate the IMPS multidetector with the new ROI's (rois=
                         9x4 matrix). Needs to access the corresponding ".multi" file.
'only', which :          retain only selected ROIs.
```

Example:

```
nplot 012345 only roi5
nplot 012345 reintegrateimps ROIS
```
(where `ROIS=[1,30,3,220; 4,30,6,220; 8,30,10,220; …]` (9x4))

Output and graphics

```
'plotstyle', {'s1',..}:   Strings defining the marker style for each pal-series (e.g.
                          'or', '*b', etc.)
'plotaxes', axhandle :    Give either valid axes handle or 'none' to suppress plot. A
                          new window is created, if not given.
'overplot'    :      adds the plot to the currently active axes.
'showfit'     :      Write Fit results in the graphics window
'nolegend'    :      do not put a legend
'noplot'      :      do not plot the results. (like 'plotaxes none')

'details'     :      show detailed information on the data (in command window).
'nooutput'    :      suppress all text output in command window except errors.
```

Examples:

`nplot 012345 plotaxes gca plotstyle ob` (plotstyle like in Matlab "plot", plus "f" for filled symbol)
`nplot 012345 only chan[1:3] plotstyle {'or','fr','fb'}`

Using the return values:

(Remember: to store the return values in a workspace variable, you need to use the syntax with ( ) and ‘ ’)

In `[avgdata,fitresult] = nplot(files, varargin),`

`avgdata` is a structure containing all resulting data.
Its most essential fields are `coordlist`, `valuelist` and `monitorlist`.
Each row in these lists corresponds to one data point of the resulting data set.

`.monitorlist` contains the true measured monitor (mon counts or time/seconds) for each point.
`.valuelist` is the list of pairs [y, dy] after normalization.
`.coordlist` are the coordinates of the data points (variable number of dimensions).
The other fields may depend on the case, and/or are mostly self-explanatory.

The structure `fitresult`, which contains the results of a fit to a function, should be self-explanatory, too.
`fitresult.line` contains the calculated graph of the fitted function.

If something does not work as expected, try this first:

1.) Use the switch “details”. This often contains the information what is going wrong.
2.) If plotting multiple files, start by plotting them separately. Also, for nplot’s guesses for many parameters, it can matter which file is the first in the order.
3.) Open the data file(s) and check for anything unusual.

PS 02/2015

HELP FOR DATA FITTING WITH NFIT

With examples

NFit can be used for any data fitting. User-defined functions can be added. The input data are of type (x,y,dy) of any source and do not have to be produced by nplot or other specific programs.

Nfit can be used both from the command line (or a script), and with a graphical interface.

In its script/command line mode, there are two typical ways to use it:

- Assign the nfit object to a Matlab variable, and do all manipulations on this variable. In this way, several instances of nfit can be used in parallel. Example:
  ```
  f=nfit('gauss');   (creates a workspace variable f, containing all data on this fitting process)
  f.setparam([1,55,.1]);
  f.fix(1);
  f.fit;        etc.    (Operations on the nfit instance called f)
  ```

- Interactive mode (convenient syntax for quick use). Example:
  ```
  nfit gauss; (creates new nfit instance with specified parameters (gauss function))
  nfit setparam [1,55,.1];
  nfit fix 1;
  nfit fit      etc.    Subsequent calls act on previous nfit instance which is stored in memory
  ```

With its graphical interface, the parameters of nfit can be controlled interactively. To open this interface on the currently active nfit instance, type

```
nfit *
```

Remark: by adding "*" to a call from the command line, this window opens automatically after execution of the command, e.g.

```
nfit gauss *
```

To open such a window for a particular instance f stored in a workspace variable, type

```
nfitgui(f)
```

Data assignment

When creating a new instance of nfit and without explicitly providing data, it will attempt to take the x, y, and dy data automatically from the currently active axes. This is in general possible for plots produced by nplot or with the Matlab "errorbar" command.

(Note: to prevent nfit from automatically searching suitable data and to create an empty instance without data, use the 'nodata' switch. Also, if data are explicitly provided, no search on open graphics windows is performed.)

Data can be changed for an existing nfit instance by directly assigning the xdata, ydata, and yerror properties: `f.xdata = [...];`

Calling nfit

`nfit`	Creates new nfit class
`nfit(parameters)`	Creates new nfit class with property-value pairs(see below)
`nfit(axes,...)`	Take data from axes (valid axes handle)
`nfit(function,...)`	Use specified fit function
`nfit(axes,function,...)`	...

Possible parameters:

```
xdata, [xdata]     : x-data
ydata, [ydata]     : y-data
yerror, [yerror]   : y error bars
startval, [vals]   : start parameters for fit
fitvar, [1/0]      : array containing "1" for variable and "0" for fixed parameters
fitfunction, [func]: function used for fitting
background, [func] : function describing baseline
constraint, [cstr] : linear constraints (like "p1=2*p2;p9=10;p7+p8=p6" etc.)
```

Switches :

```
nodata           : Do not perform automatic search for data in open windows
*                : open interactive window
```

nfit methods:

```
addfunc(func,varargin)    : add a new function
rmfunc(ncomp)             : remove the function ncomp from the sum of functions
addconstraint(constr)     : add (one or several) constraint strings
fix(n)                    : fix one or several parameters
clear(n)                  : clear one or several parameters
rmconstraint(ncon)        : remove one or several constraints
fit                       : perform fit
setparam(varargin)        : set parameter values
plot                      : plot the function with the current parameters
```

(Type 'help nfit.[methodname]' for help on individual methods)

To call a method on a particular instance f of nfit, use f.methodname, e.g. `f.fit;`
To call a method on the nfit object in memory (last call), use nfit methodname, e.g. `nfit fix 1;`

Examples:

`nfit gauss`	Fit a Gaussian to data in currently open window (if possible)
`nfit gauss *`	same, and open interactive window after fitting
`nfit set p1=1`	set first parameter value
`nfit fix 1`	fix first parameter
`nfit plot`	
`nfit fit`	
`f=nfit(gca):`	create new object f with data in current axes

```
f=nfit('xdata',xdat,'ydata',ydat,'yerror',err,'fitfunction','linear')
f.addfunc(@gaussA);            add a gaussian to function definition in f
f.setparam([1,1,5,30,.5])      set parameter values
f.plot
```

Fit Functions

You can obtain a list of available functions by typing `fitfunctions.`

For a particular function, type `showfunction [name]` for information on parameters etc.

You can define new functions by using any existing one as template; you should also have in the same folder an empty file "template.m". In general, the syntax for the function is
`[val paramnames paramnum description] = myfunction(param,x,opt)`
where `val` is the function value, `paramnames` and `paramnum` are the names and number of the function's parameters, and `description` returns the name or a description of the function.
(For problems in higher dimensions, the function `x`→`val` can be a function $\Re^n \to \Re$.)

For a few particular functions (gauss, lorentz, linear) an automatic guess of start parameters is performed; for all other functions you must provide starting values using `startval`.

By default, all parameters are varied; use `fitvar` or `fix` to fix some of them, or `constraint` to introduce linear constraints.

For "gauss" and "lorentz" you can specify the number of peaks by adding an integer ("gauss3", "lorentz2", etc.). For general combination of functions, you can use `fsum` (for instance, "gauss2" is equivalent to "`fsum(@const,@gaussA,@gaussA)`"), or add several functions with consecutive calls to `addfunc`.

P.S. 02/2015

HELP FOR POWDER ALIGNMENT : POWXBU AND POWFIT

With examples

The script powxbu prepares a job file for a4-scans.

The script powfit analyses a given set of scans and determines the spectrometer zeros.

Powfit works on any set of a4-scans (i.e. not necessarily performed by a powxbu-jobfile) under the condition that***

P.S. 02/2015