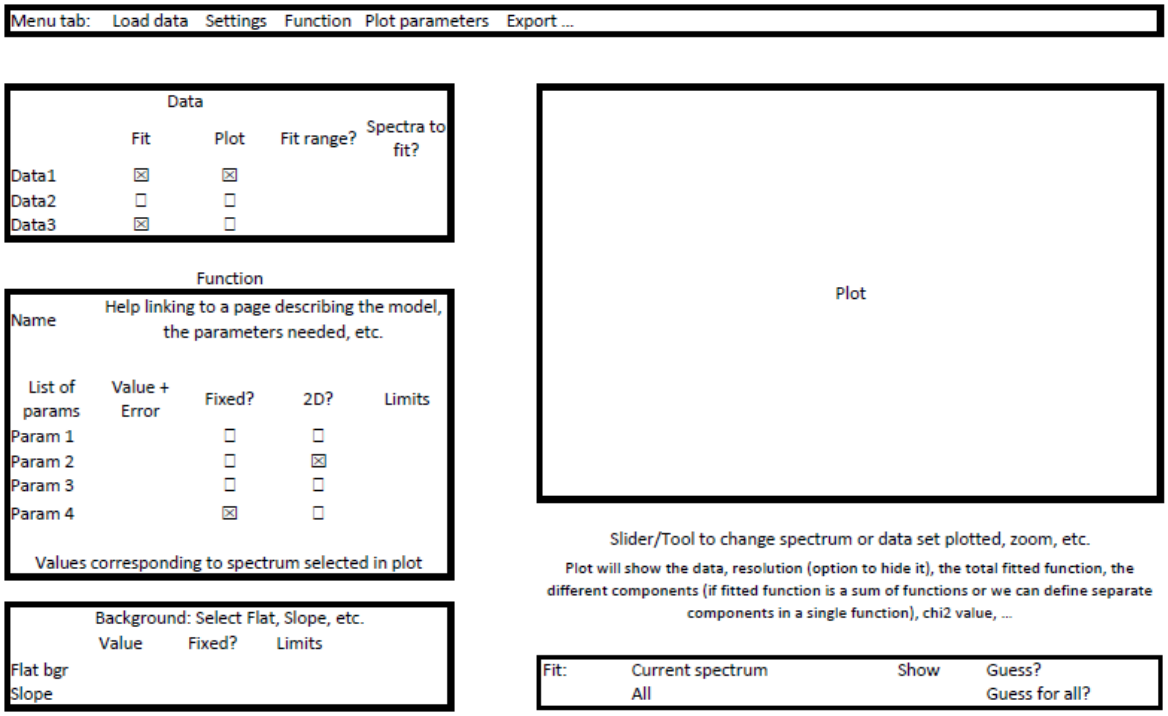
Thoughts on a QENS fitting program/interface

I don't have a particular or very fixed vision of how the interface should look like. Ideally, it should be intuitive enough so that a new user can use it without consulting the documentation or requiring the guidance of an expert user (but this is easier to say than to do). So in the following I describe mainly the functionality that I expect to find, with just some very raw ideas of how it could look like.

I put in blue those extra features that I would like to have, but that I consider that are not essential. I prefer getting first something simple (although having the essential functionality) that works than trying to get the perfect program with all imaginable possibilities.

A very raw picture of how the UI could look like:



And a more detailed description of the functionality that it should have:

Load data: Select between S(Q,ω) or I(Q,t). Possible to add F(x,y).

F(x,y) is only if one wants to create a more general fitting program, so we could load a completely general set of data that depends on one or two arbitrary variables, x and y.

Should be able to load more than a set of data, but on a first approach we can imagine that all of them are of the same type. Loading and fitting together S(Q,ω) and I(Q,t) is highly desirable, but this implies being able to load at least two different functions (in energy and time) with some common parameters and designing a way of linking the right function to the right set of data. Being able to load and fit together data from different techniques (e.g. QENS + NMR + dielectric) would be great, but too ambitious for the moment.

When loading S(Q,ω) data, for each dataset the program should ask the user about the corresponding resolution and suggest 3 possibilities:

1. None: E.g. in case we simply want to load a vanadium dataset and check the instrument resolution.
2. A measured data set. In this case, it should be read, validated (at least check that the number of Q values agrees with the number of spectra in the dataset), and be associated to the corresponding sample dataset. There should be an option to show or hide the resolution. Otherwise, I don't see a special need for showing the resolution in the main window, we can just assume that each sample data has a resolution linked to it.
3. Generate a numerical resolution, e.g. Gaussian or Lorentzian.

For I(Q,t) data, I assume that there is no need for resolution.

Data can come from different instruments and reduction software. I assume that the user provides fully corrected data representing the dynamic structure factor of the sample as a function of the momentum and energy transfers, i.e. S(Q,ω). I think that all the models can be made completely agnostic of the units, so in this case we can accept any units for Q ($Å^{-1}$ or nm^{-1}) and ω (meV, μeV, THz). However, this implies that in some cases the user will need to think twice about the units of some parameters. For example, in a model where the fitted parameter is the diffusion constant, the output would be D in $Å^2$meV if S(Q,ω) was given in $Å^{-1}$ and meV or in nm^2μeV if the input was in nm^{-1} and μeV. In any case, we need to propose several formats for the input data (at least one in ASCII and one in NeXus or HDF, and considering also the errors). We also need to agree on how to identify the points that contain no data, as the (Q,ω) grid is not rectangular. Can we assume that a value of 0 in the data set means that this corresponds to a (Q,ω) point that is out of the range of the spectrometer and therefore should not be considered in the fit?

Is there any NeXus standard for S(Q,ω) data? On the long-term, one could think of doing something similar to what the SANS community has done (see www.cansas.org), but to start with I'm happy with using Mantid format + any other reasonable format that we can agree.

I think that dealing with multiple scattering has been mentioned/requested, but this requires a knowledge of the instrument and sample geometry, so I would not consider it.

Settings: To select the minimizer (e.g. Levenberg-Marquardt, genetic algorithm, Bayesian, etc.) and the corresponding parameters (tolerance, max number of cycles …).

It can be used also to set other general options, such as use the weights (computed from the errors, this should be the default) or not.

Perhaps give here also the possibility of selecting the useful range of the resolution function, in order to avoid making the convolution with the full dataset. If there is more than one dataset with different resolutions, then one needs to provide different ranges for each of them. Perhaps this can be done automatically without asking the user?

Function: Select a function from the library of models. It could be a sum of functions, but we need to find a mechanism to be able to combine them. E.g. imagine that we have two populations of atoms and we want to fit the first one with model M1 and the second one with M2 (M1 and M2 being models in the library). In this case, the user cannot simply select M1 and M2, but she will need to be able to write something like Y = p*M1 + (1-p)*M2, where p would become an extra parameter to fit in addition to the parameters of M1 and M2.

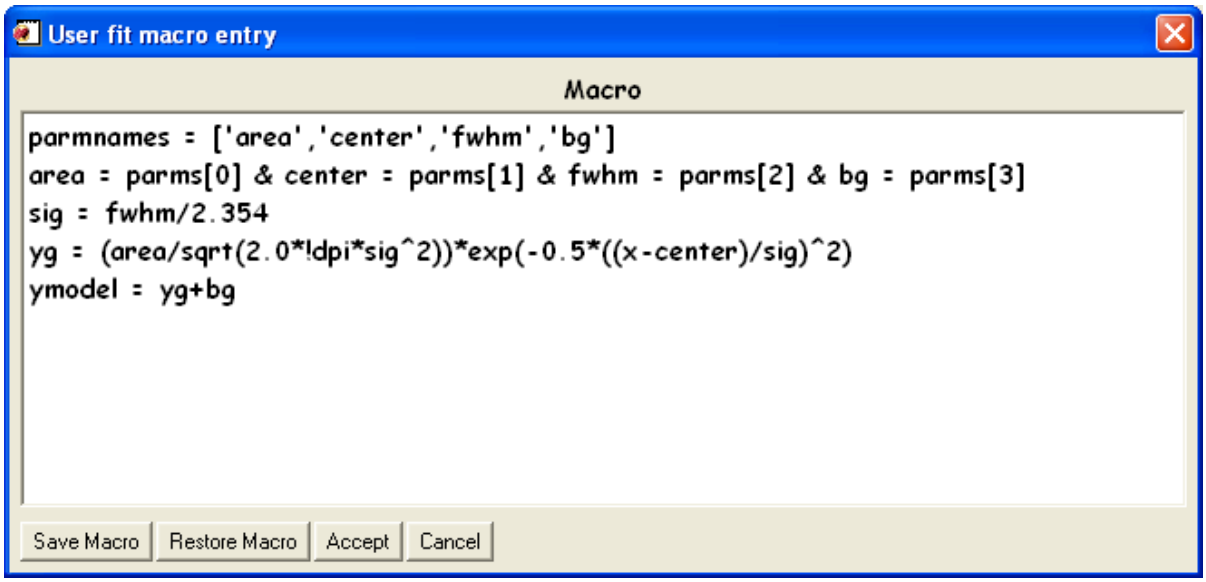
The models will have parameters that are independent for each Q value and common parameters that do not depend on Q (e.g. the diffusion coefficient). At the moment, I have simply considered that once that a function is created, all the parameters will appear in the main interface (being independent by default) and the user will select the 2D button for those that are shared. It could be good if the models already identify the Q-independent parameters and this is done automatically.

The background could be part of the list of models, but in this case it should not be convoluted with the resolution, so for the moment I put it apart in the interface.

It would be good to have a mechanism allowing to redefine a parameter as a function of another parameter. E.g. I just want to fit a Lorentzian, but instead of letting the FWHM free I would like to force it to be $D*Q^2$, with D again common for all Qs.

When fitting at the same time two or more datasets, which typically will correspond to different energy ranges and resolutions, I think that one can work with a single function (or sum of functions) that should be able to fit all the datasets and that any parameter that is common for all Qs should also be common for all the datasets. If we do not find some cases where this is not true, then fitting one or multiple datasets is equivalent. If there is some exception, then we possibly need to have a mechanism that allows the user to set constraints such as 'Dataset2.Param2 = 3*Dataset1.Param1'. If things become too complex, this could be available only in script mode.

The user should be able to add easily new functions. For simple cases, it could just be an interpreter allowing to do things like that (this is an example in PAN):



For more complex cases, there should be a template and/or some guidelines allowing to write a new model and add it to the library. I think that python is the preferred option for this. Our own models could be also written directly in python or otherwise in C++ if needed for efficiency.

Plot and data: After loading some data, the plot window should show the 1st spectrum of the 1st dataset and the associated resolution. For the resolution, there should be a button or an option in the menu to hide/show it. In the plot or in a box close to it should appear the corresponding Q value. For a single dataset is easy to have a slider or a box to give the number of the spectrum one wants to see. For multiple datasets is not so clear. Perhaps a button to select the dataset and another to

select the spectrum in it? Or constraint the plot window to show only 1 dataset at a time? In this case, the user would select in the data window the dataset that she wants to plot (only one) and to fit (and this could be one or several).

The plot window should have the standard capability expected for scientific plots: zoom, select ranges, log/linear scales, etc.

In the data window, for each dataset the user can give a fitting range. I assume that there is no need to give different fitting ranges for different spectra. She can also give the list of spectra to fit (all by default).

The selection of the spectrum to be shown should be linked to the list of parameters, so when a new value spectrum is selected the corresponding parameters are updated.

A show button will show the model function for the current spectrum. If the function is a sum of functions it will also show each individual function. Then there should be an option to show/hide the different functions, as well as to show them convoluted or not with the resolution. Note that in Lamp/str_fit we can also generate a function that is a sum of models, but in some cases we found that the fitting was unstable and that it was preferable to write everything in a single function, e.g:

$$Y = P_1 \cdot e^{\left(-\frac{P_4 Q^2}{3}\right)} \cdot \begin{bmatrix} P_3 \cdot \delta(\omega) + \\ P_5 \cdot S_1^{\mathrm{RotTrans}}\left(Q, \omega, D_{w1}, \tau_{w1}, R_{w1}, D_{R,w1}\right) + \\ P_{10} \cdot S_2^{\mathrm{RotTrans}}\left(Q, \omega, D_{w2}, \tau_{w2}, R_{w2}, D_{R,w2}\right) + \\ (1 - P_3 - P_5 - P_{10}) \cdot \frac{1}{\pi} \cdot \frac{P_{15}}{(\omega - P_2)^2 + \mathrm{P}_{15}^2} \end{bmatrix}$$

Hopefully, this is not needed, so with a limited number of functions we can generate all the possibilities. In my example the user could write:

Y = DW x (P1 x Delta + P2 x RotoTrans + P3 x RotoTrans + (1-P1-P2-P3) x Lorentz),

Where DW, Delta, RotoTrans, Lorentz would be the general models in the library, each with its own parameters, and P1, P2, and P3 additional parameters needed to handle the different contributions of each model to the total. But if this results difficult to implement and we need to write full functions like this one, it would be useful to have a mechanism to identify different components in a function (here Delta, RotoTrans1, RotoTrans2, Lorentz) in order to be able to plot them.

The user should be able to load and plot a model function even if there are no datasets. This requires that one can select an energy range to plot.

When a model function is shown together with the data, the reduced chi2 should be shown in the plot or in a box.

Each model in the library should have some default parameters. If data exist, it would be good to have a guess button that makes a quick search to propose some values of the parameters bringing the model close to the data.

Fitting: The function name used in the fitting should be shown clearly, together with all the parameters and corresponding values. For each parameter the user must have the possibility of modifying the initial value, fixing the value, setting it as a common parameter for all the spectra and datasets, and giving a range of limits.

The user can fit a single spectrum (the one currently active in the plot) or all of them. The fit should return the parameters and an estimation of the errors.

When the fit is done, there should be a button or a menu to plot the parameters, probably in a new window.

Export: Need to export the fitted parameters and the model curves. For the model, the user needs to have the same options than in the interactive plot: export full model, separate components, convoluted with resolution or not.

It should be possible to reload the fitted parameters from this file.

It should be possible to save and load the current session.

It should be possible to do the fitting without the interface, using a script. Ideally once that the user has done a fit, the program could generate a script able to redo it. Then the user can share it with other colleagues or modify to automatically fit N temperatures or do more complex things not foreseen by the interface.