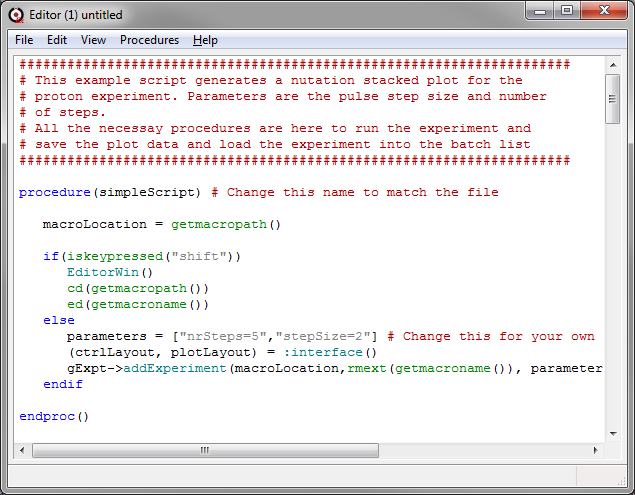
The Spinsolve-Expert Script Editor

The script editor allows sophisticated control of experiment operation:

1. Multiple experiments can be run one after the other automatically.
2. Loops can be defined to run a group of experiments multiple times.
3. The output of one experiment can influence subsequent experiments.
4. Post-processing can be performed on the collected data.

To use the script editor, start by opening the script editor with a basic script template already loaded. Do this by selecting the option *Edit script template* from the *Program* menu.



This displays an editor with a preloaded unnamed script which can then be modified as required. The script is divided into *procedures* the first one; *simpleScript* should not be changed, apart for the procedure name ‘simpleScript’ which should be modified to be the same as the final filename, but without the .mac extension. However the following procedures may need to be updated to suit your application:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Interface ... this defines the parameters which will be exposed in the Expert interface. In this example there are two parameters displayed – the number of steps and step size in a nutation experiment



It also specifies which plots will be visible when loading the saved data. In this case only a single plot is display: pt3



Finally you can define some post processing buttons



The syntax here for each line which corresponds to a post processing button, is the button label, the plot it corresponds to and the macro to run. Notice the extensive escaping of quotes – this is required when quotes appear inside a string.

These three parameters are returned to define the user interface.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

getPlotInfo ... this associates a filename with the plot so we can load and save files. Each plot in the load interface should have a filename next to it



In this case there is only one plot pt3 and it will display the saved nutation curve.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

addCommand ... if uncommented this will over-ride what happens when you press the ‘+’ button next to the batch list. This might be necessary if you need a specific comment. See the StartAtTime macro in the batch menu for an example.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

renameCommand ... if uncommented this will over-ride what happens when you press the rename button next to the batch list. This might be necessary if you need a specific comment. See the StartAtTime macro in the batch menu for an example.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

backdoor ... this is the main entry into the script when the Run button in the main interface is pressed. It includes an argument which will include the parameters defined in the interface procedure.

This example procedure does the following:

It makes all the passed parameters available as local variables



It defines a plot layout with two 1D plots in the first row and a 2D plot in the second row and makes these available locally as the variables pt1, pt2 and pt3

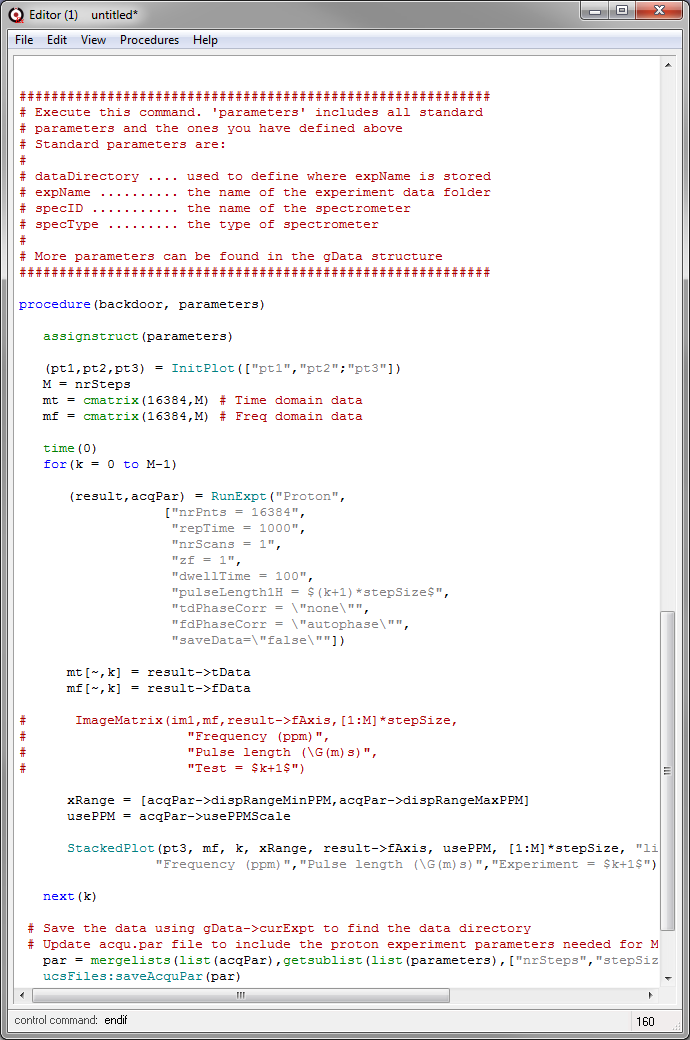


It allocates a 2D data set to store *M* FIDs of 16384 points for both the time and frequency domain data. nrSteps is a variable which was defined and exposed in the user interface.



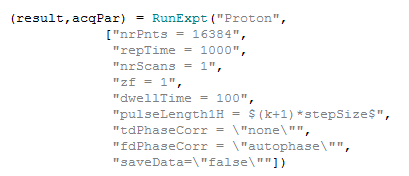
Then *M*, 1Pulse-H experiments are run via the *RunExpt* command.





*The template experiment – a Proton experiment which is repeated M times with a different pulse length for each step*

Each experiment has a pulse length equal to 1 plus the loop counter *k*. So the pulse length will range from 1 to *M* microseconds. A number of other parameters such as the number of points collected (16384) and dwell time (100 s) are also set here. Parameters not listed here are determined by the default and common parameters for the Proton experiment.

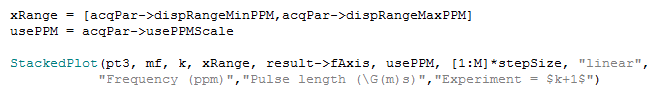


After each experiment has completed, data is returned via the structure *result.* Also the acquisition parameters used by the experiment are also returned as a structure. We used the values in result (the fid; *result->tData* and spectrum; *result->fData*) to fill the kth rows in the time and frequency domain matrices

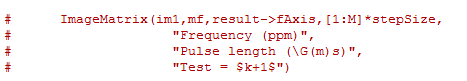


Note that the index order is zero based and [column-x, row-y] in Prospa not [row, column] as in Matlab*.*

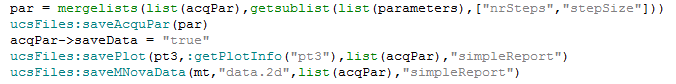
We then display the data in matrix *m* using the StackedPlot command. This pocess is repeated for all *M* scans. Two of the parameters for this command are extracted from the acquisition parameter structure:



An alternative display format using the ImageMatrix command is also given.

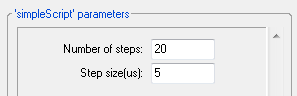


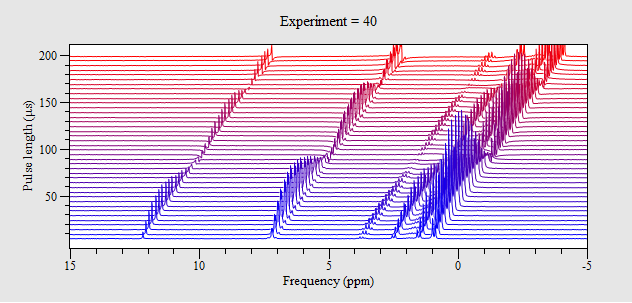
Once FIDs for all M pulse length values have been recorded, the for loop is exited and the data is saved in both Prospa and MNova formats. Also the acqusition parameter file is updated to include the user defined parameters *nrSteps* and *stepSize*. Note that these commands requires the data structures in list format – hence the conversion from structure to list type using the *list* command.



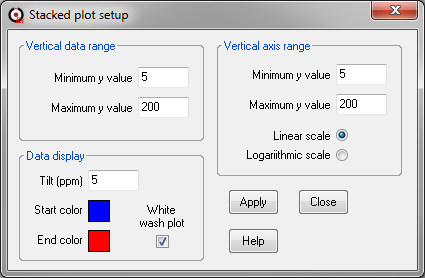
To learn more about the syntax of the special commands like InitPlot, RunExpt and ImageMatrix click on the command and press F1 for help or control double-click on the command to see the script. Details are given later in this document.

Here is the output from simpleScript for 20 steps and a 5 s step size (this Spinolve has a long 90 degree pulse of 47 s)





In this case we have used the post processing button ‘View’ to add a tilt and select the white wash mode.



The Integrate peaks post processing option can be used to integrate specific ppm ranges e.g. for the 3 peaks at 12.5, 7.1 and 1 ppm:

