Science & Technology Facilities Council

Technical Report
RAL-TR-2016-019

Simulation support and further Indirect Inelastic specific improvements within MANTID

E Oram, L McCann

November 2016

Enquiries concerning this report should be addressed to:

RAL Library
STFC Rutherford Appleton Laboratory
Harwell Oxford
Didcot
OX11 0QX

Tel: +44(0)1235 445384
Fax: +44(0)1235 446403
email: libraryral@stfc.ac.uk

Science and Technology Facilities Council reports are available online at: http://epubs.stfc.ac.uk

ISSN 1358-6254

Simulation support and further Indirect Inelastic specific improvements within MANTID

Elliot Oram and Louise McCann

August 15, 2016

Supervisor: Dr. S. Mukhopadhyay

Abstract

The Manipulation and Analysis Toolkit for Instrument Data (MANTID) is an open source cross platform application specialising in the visualisation and analysis of muon and neutron data and is widely used throughout the scientific community worldwide.

This document aims to lead on from the D. Nixon's report "Simulation support within MANTID for molecular spectroscopy" [1] and describe the further improvements to the simulation support within MANTID. This report will also describe the changes made to plotting and saving in MANTID as well as changes made in the data analysis section for indirect inelastic spectroscopy. The report will also briefly touch on minor modifications relevant to indirect inelastic spectroscopy made between MANTID release 3.7 and 3.8. Finally, the report will conclude with a section advising possible future developments with indirect simulation and indirect geometry sections of MANTID.

Contents

Acknowledgements

We would like to thank Dr. Sanghamitra Mukhopadhyay for her support in understanding the simulation techniques that I have amended and implemented as well as Dr. Spencer Howells for his guidance in implementation of the described plotting changes. We would also like to thank ISIS and Molecular Spectroscopy group for the opportunity and funding to carry out the work described in the report below. Finally, we would like to thank the MANTID team for their technical support in solving software engineering problems that occurred during development of these features.

1 Introduction

The Manipulation and Analysis Toolkit for Instrument Data (MANTID) is a cross platform open source software application providing support for the visualisation and analysis of scientific data [2]. MANTID, specialising in muon and neutron data, is built and maintained by a globally distributed software development team with most team members at ISIS (Rutherford Appleton Laboratory, UK), SNS (Oak Ridge National Laboratory, Tennessee, USA), the European Spallation Source (Scandinavia) and a newly establish team at the Institut Laue-Langevin (Grenoble, France). MANTID also has many other contributors that can be found under the partners and contributors section of the MANTID homepage ref [2].

MANTID is written in C++ and Python and offers a graphical user interface (GUI) or script based interface to access it many algorithms. Algorithms can be thought of as isolated routines designed to perform a single operation normally on data stored in MANTID workspaces. A workspace in MANTID is a matrix of x, y and error data that normally represents signals recorded by detectors from experimental data.

This document aims to describe the changes made in the Indirect Simulations area of MANTID as well as some general changes to the indirect geometry section that will be available in Release 3.8.

The document will also conclude with the possible future developments that could be made to further improve the indirect inelastic area of MANTID.

2 Simulations

2.1 Simulated density of states

The SimulatedDensityOfStates algorithm has input of a phonon [3] or castep [4] file containing simulated data which is then parsed by the algorithm and manipulated to allow for comparison with experimental data. SimulatedDensityOfStates can compute partial or complete density of states calculations, infra-red and raman spectra as well as displaying ion and bond analysis tables.

For more information about the SimulatedDensityOfStates algorithm see the online documentation at: http://docs.mantidproject.org/v3.7.1/algorithms/SimulatedDensityOfStates-v1.html

The SimulatedDensityOfStates algorithm can also be operated using the 'Density of States' interface which can be found under Interfaces > Indirect > Simulations.

2.1.1 Isotopes

Castep and Phonon files contain the different contributions from ions to make up a single simulated compound. In the data files each ion is accompanied by its element symbol e.g. H for Hydrogen and a mass number which denotes if the ion is an isotope. Isotopes are suffixed with ':P' so for example a Hydrogen isotope will be written as 'H:P'.

The SimulatedDensityOfStates algorithm has been updated to enable parsing of isotopes. This is done in several steps:

- The isotopes are identified by the existence of the ':P' notation in the ion name.
- The corresponding mass number will be rounded to the nearest integer.
- The mass number is parsed into a single string of the form '(<element><mass>)' for example '(H3)'.
- The MANTID algorithm *SetSampleMaterial* is called and will add the sample material to the workspace including coherent and incoherent scattering cross sections.
- Output workspaces from SimulatedDensityOfStates are renamed to end with the element and isotope (**Figure 1**)

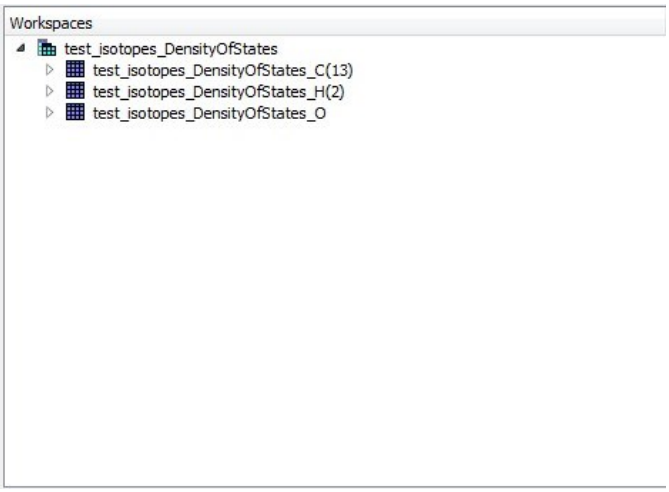

Figure 1: The output from the test isotopes file used in MANTID unit testing. The workspaces show that the 'test_isotope.phonon' file contains Carbon 13, Hydrogen 2 and Oxygen.

2.1.2 Single ion indices

Ion and castep files can potential have various indices for each ion which, whilst they might be the same element, will have differing fractional co-ordinates. To further analyse the data examining each index is required. An option for this has been added in the interface (**Figure 2**). The previous check box options have been replaced with the drop down menu in **Figure 2**.

By selecting "Calculate Individual Indices" from the drop down menu, the individual indices will be created and added to the workspace window (**Figure 3**). Each workspace in the group corresponds to a fractional co-ordinate and the workspaces are sorted by index which is the suffix at the end of the workspace.

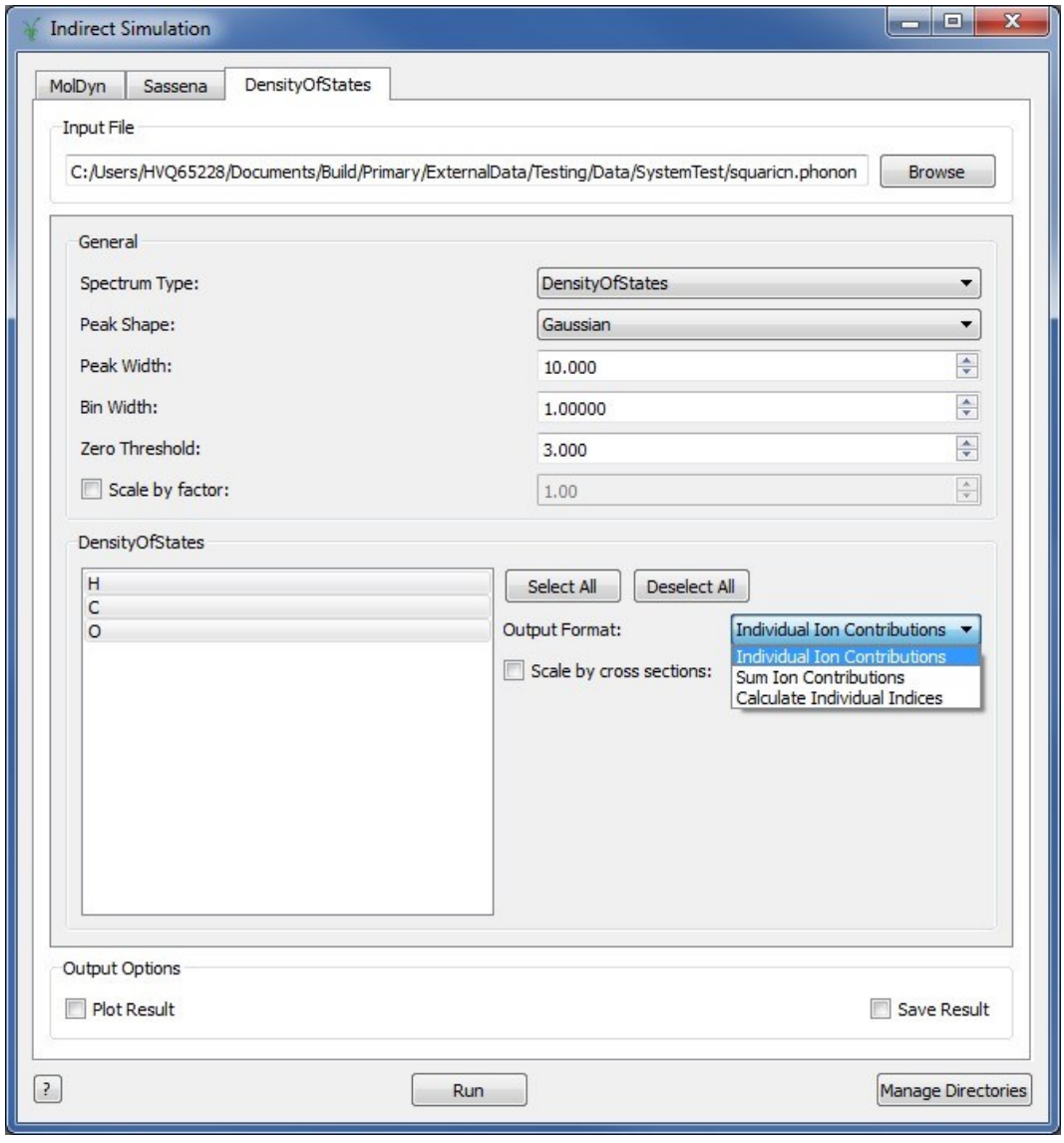

Figure 2: Shows the new drop down menu for Output Format in the DensityOfStates interface (interfaces > Indirect > Simulations > DensityOfStates).

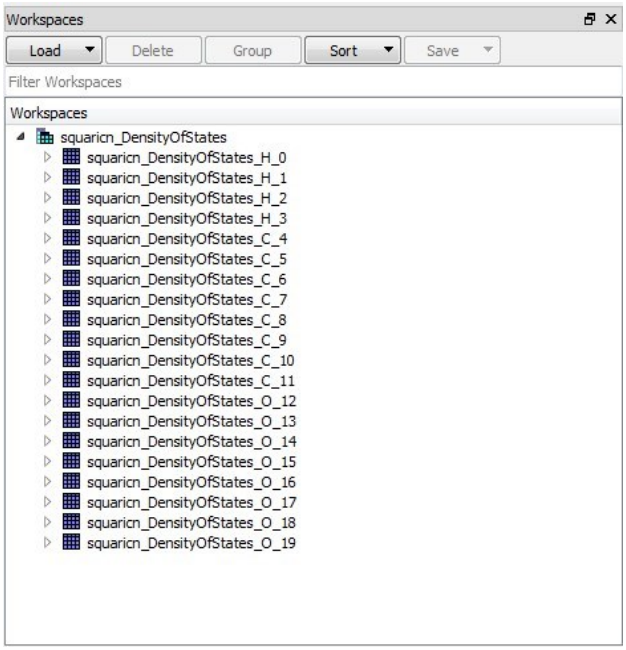

Figure 3: Shows the output from running *SimulatedDensityOfStates* algorithm with the input of the squaricn.phonon test file and the "Calculate Individual Indices" output option selected.

2.1.3 Cross Sections

The simulated density of states algorithm also allows for the scaling of ion contributions to density of states by cross section. The scaling factor can be the coherent cross section, incoherent cross section or a combination of the two. These cross sections are obtained for each ion respectively using the *SetSampleMaterial* algorithm and have units of barns. This option is available for all isotope and ion contribution options.

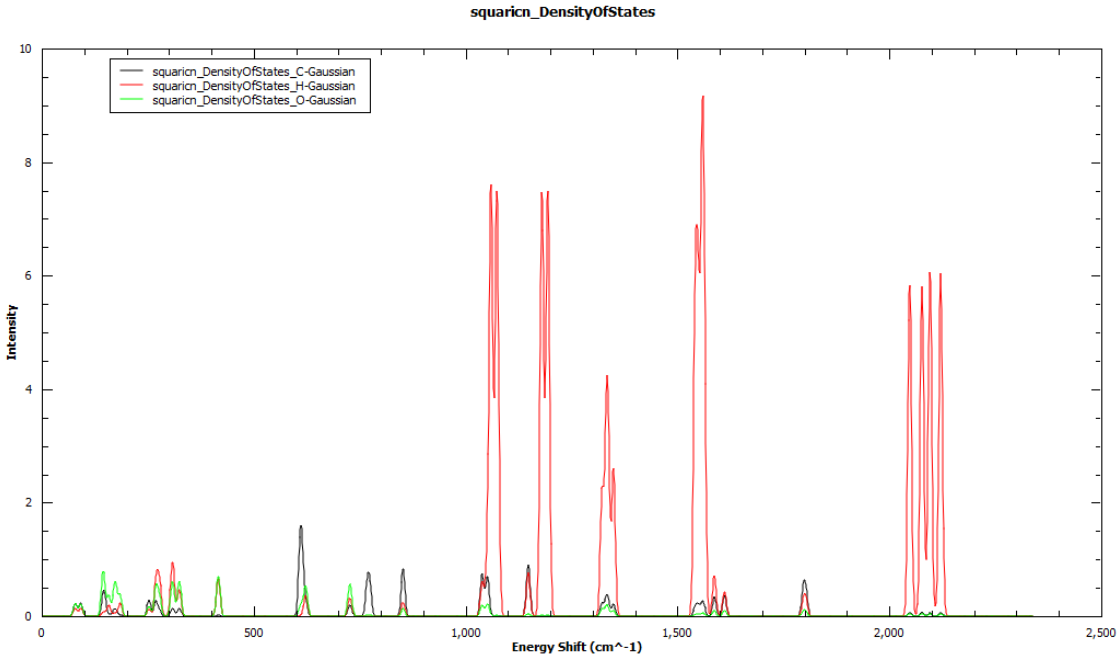

Figure 4: Shows a graphical representation of the simulated density of states of squaric acid given by squarcin.phonon test file, with "Individual Ion Contributions" and "Scale by Total cross section"

2.1.4 Refactor of loading

To improve both the readability and to reduce code duplication several refactoring steps have been carried out. The largest of which is the refactoring of the loading and parsing of the castep and phonon files. Three scripts (load_catep.py and load_phonon.py, load_helper.py) have been created in order to deal with the loading and parsing of the phonon and castep data. These scripts can be found in the scripts/Inelastic/dos directory within the MantidInstall location. **Figure 5** shows in more detail the interaction between the scripts and the algorithm.

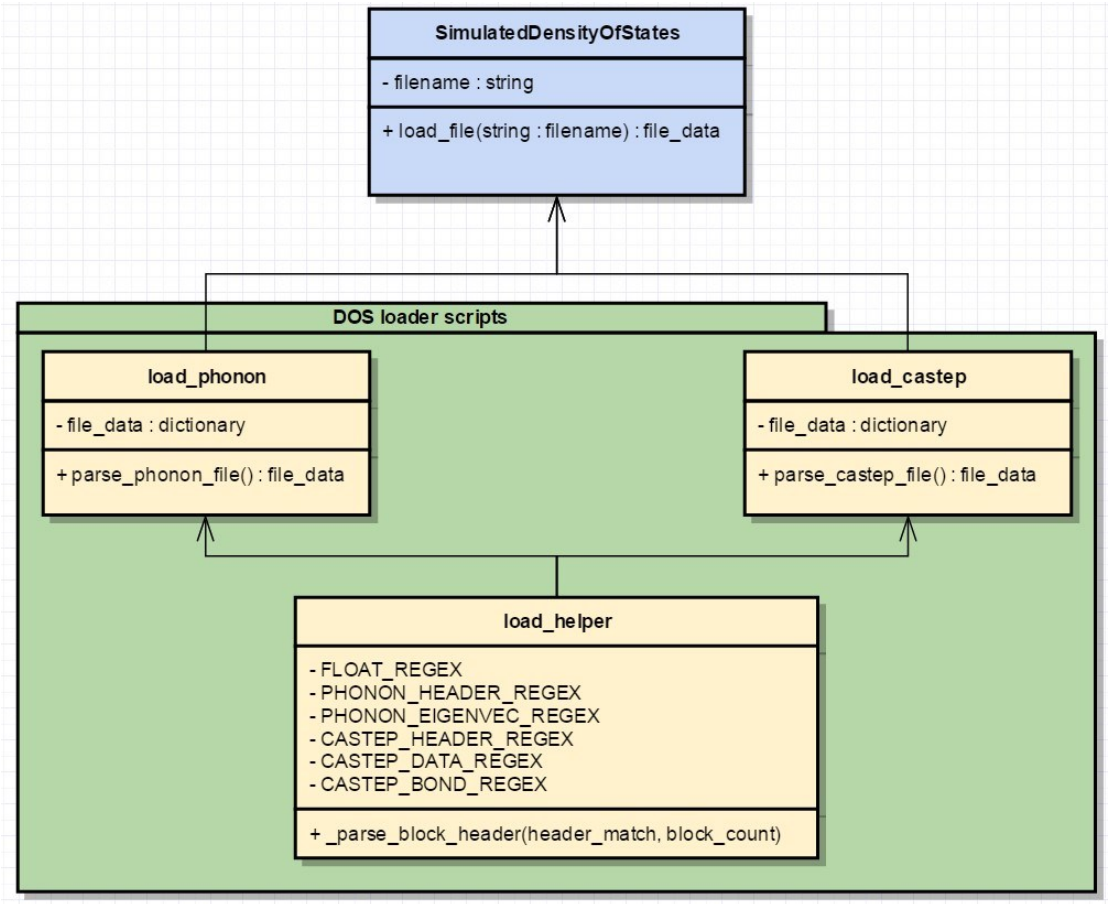

Figure 5: Shows the interaction between the *SimulatedDensityOfStates* algorithm and the refactored DOS loader scripts. *load_helper* stores the regex strings for the phonon and castep code blocks as well as a general function (_parse_block_header) to parse the header of any block.
load_castep and *load_phonon* are used to parse the castep or phonon file (respectively)
SimulatedDensityOfStates calls the correct loader (either *load_phonon* or *load_castep*) depending on the input file.

2.2 NMOLDYN

2.2.1 LoadNMoldyn4Ascii1D

The algorithm *LoadNMoldyn4Ascii1D* imports 1D density of states (dos) and velocity autocorrelation (VAC) functions from an nMoldyn4 [3] ASCII file, with the option to convolve the data with a resolution function. This algorithm was written by A. Phimister and was designed to be a 1D equivalent of the *LoadNMoldyn4Ascii* and was written as such. The parallelisms between the two algorithms will make it easier to refactor functionality in many places (this is discussed in further detail in section 6.2).

The algorithm takes a directory of where the nMoldyn functions are stored as well as the desired functions to load as a string list. Additionally, there is an option to convolve with an instrument resolution. Choosing TOSCA for this option will smear the data with the instrument resolution making comparison with experimental data easier. **Figure 6** shows the data from a dos function with and without the TOSCA resolution convolution.

For more information on the *LoadNMoldyn4Ascii* algorithm see the online documentation at: http://docs.mantidproject.org/nightly/algorithms/LoadNMoldyn4Ascii1D-v1.html

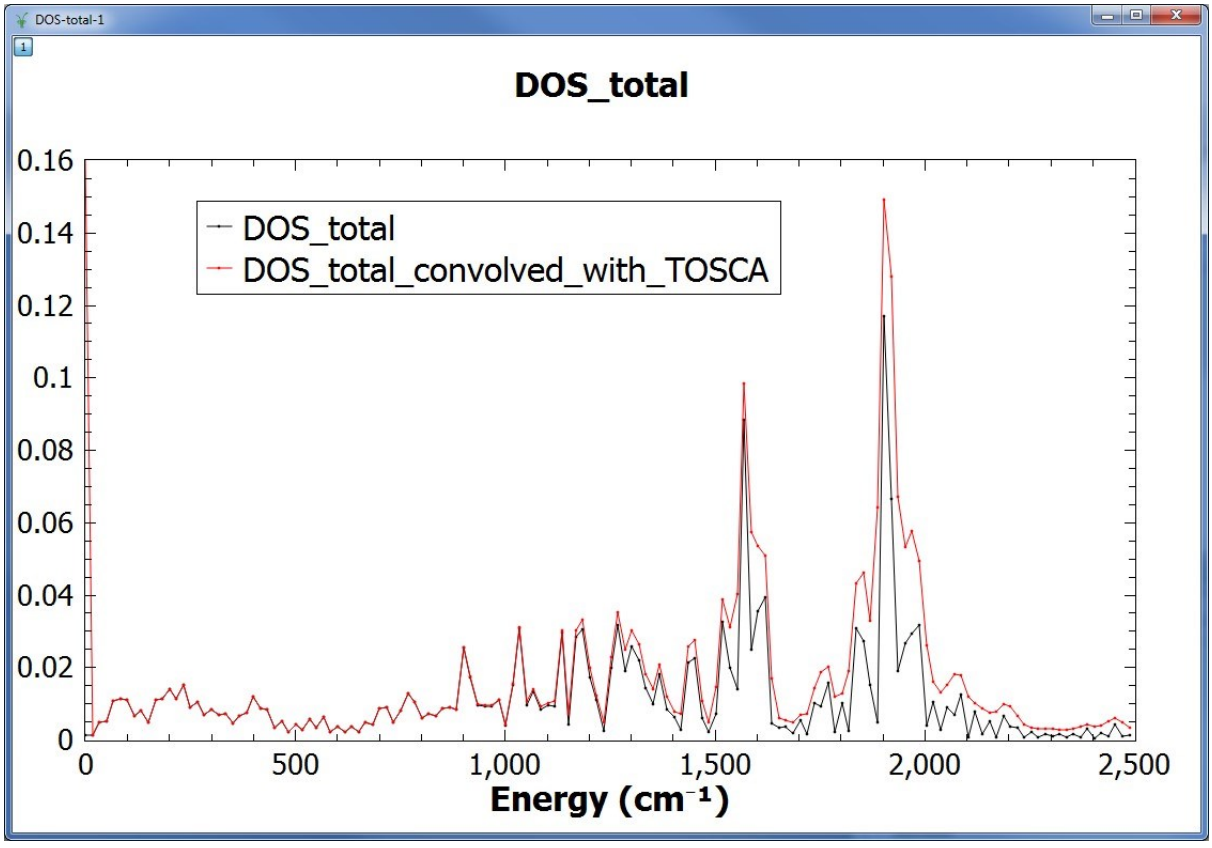

Figure 6: Plot showing the total density of states simulated using nMoldyn 4 both as raw data from nMoldyn 4 (black) and convoluted with the TOSCA resolution (red).

3 Plotting and Saving

When many of the original scripts were ported from the original MODES [3] and IRIS Data Analysis package [4], the plotting and saving of workspaces these algorithms produced was handled within the algorithm itself. To reduce the amount of duplicated code used to plot and save for every algorithm, this functionality has been refactored from the algorithms and moved to the interface code that creates and handles the GUIs in the indirect section of MANTID. The simplest form of refactoring was used in this case which allowed the plotting and saving options to be defined before the algorithm was executed.

For most cases, pre-definition of plotting options is not a problem, as plots only need to be created once. However, with an interface such as ConvFit (see section 3.1) many fitting models are available to apply to the data. In this case, users would often want to make the decision to plot and save data after running the algorithm and assessing the quality of the data and fit.

3.1 Post plotting and saving in ConvFit

Post plotting and saving (or plotting and saving after algorithm execution) has been implemented in ConvFit to resolve the problem mentioned above. The previous "Save Result" check box has been replaced with a button (**Figure 7**). Whilst the "Plot Output" drop down menu still exists, it is now accompanied by a button that is enabled after the algorithm is run to plot the desired option see **Figure 7**. Additionally, another button has been added to export the preview plot to a Mantid plotting window, which allows for further manipulation of that data than that offered by the preview plot (this button is highlighted with a red ring in **Figure 8**).

Under the previous pre-plotting and saving system, very few checks were required as plotting and saving were performed directly after execution. With the post plotting scenario, some time may pass before the plot or save button is pressed and therefore, it is important to validate that the workspace still exists in the Analysis Data Service (a key-value map of workspace names and workspaces themselves). If the workspace is no longer present when plot or save is called a message box will appear stating that the workspace no longer exists, this is shown in **Figure 9**.

This has also been implemented for other interfaces. As of the MANTID 3.8 release, all interfaces including: Bayes, Corrections, Data Analysis, Data Reduction, Diffraction, and Simulations now use the post plotting and saving functionality where applicable.

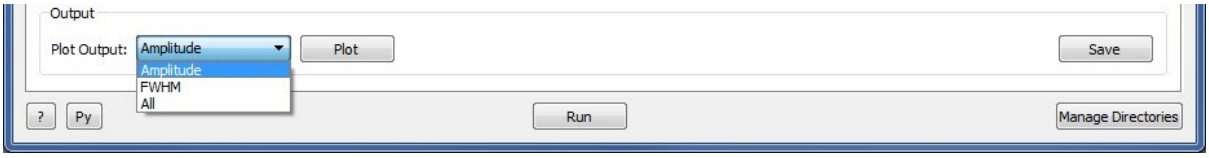

Figure 7: Shows the 'Plot' and 'Save' buttons in the ConvFit interface.

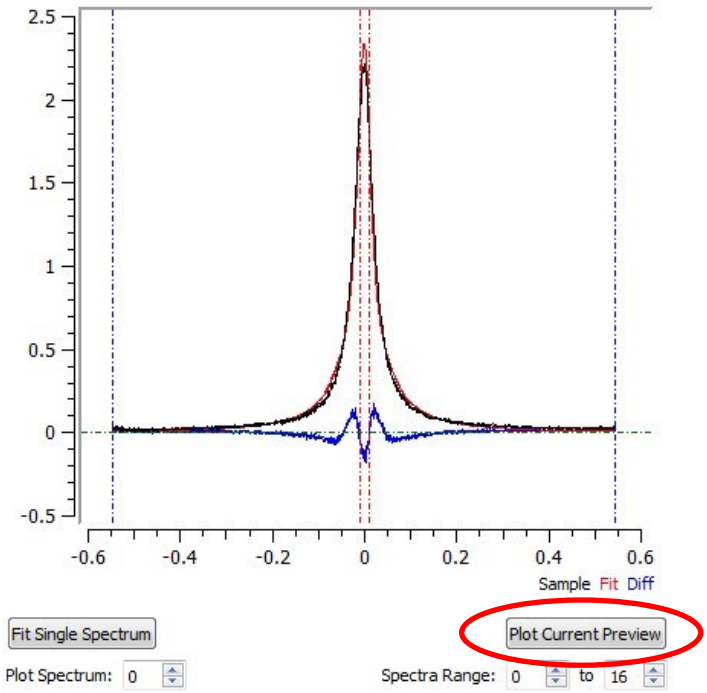

Figure 8: Shows the button to export the current Preview Plot to a MANTID plotting window highlighted by a red ring.

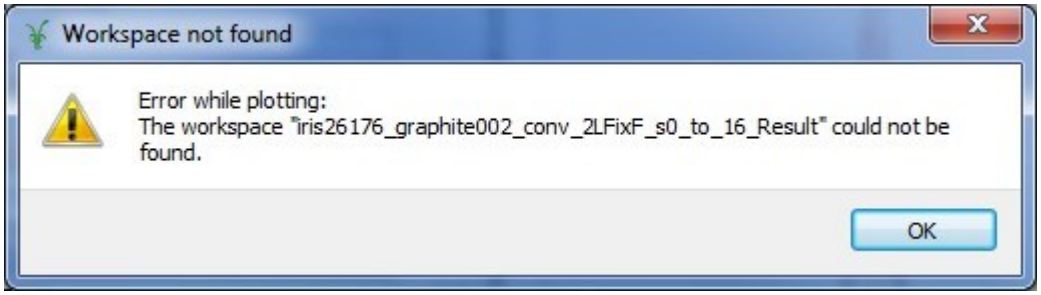

Figure 9: The error message that is produced when the plot button is clicked but the workspace cannot be found in the ADS.

4 Further developments for Data Analysis

4.1 Bayes Stretch for Bayesian Analysis

BayesStretch is a new algorithm that has replaced the Quest.py script. The original Quest.py script acted as a wrapper to the IndirectBayes.py script which in turn called the precompiled FORTRAN code currently used for MANTID Bayesian analysis. With these changes it should now be possible to remove the vast majority of the code from the IndirectBayes.py script as it now exists in three algorithms: *BayesQuasi, BayesStretch* and *ResNorm*. These algorithms are based on the Bayesian analysis described by D.S. Sivia [8].

Whilst BayesStretch still calls the FORTRAN codes, it has the advantage of removing the plotting and saving functionality from the algorithm code. Refactoring the plotting and saving to the interface has the advantage of making the transition to post plot and save functionality (see section 3) much easier. In addition, as it is now a MANTID algorithm, features such as progress reporting and algorithm cancelling can be implemented, automatically generated dialog boxes for alternative algorithm input (see **Figure 10**) as well as automated testing and integrated documentation pages. The 3 Bayes analysis scripts are implemented as interfaces under Indirect > Bayes as ResNorm, Quasi and Stretch.

The code for the BayesStretch algorithm can be found online at:
https://github.com/mantidproject/mantid/blob/master/scripts/Inelastic/IndirectBayes.py

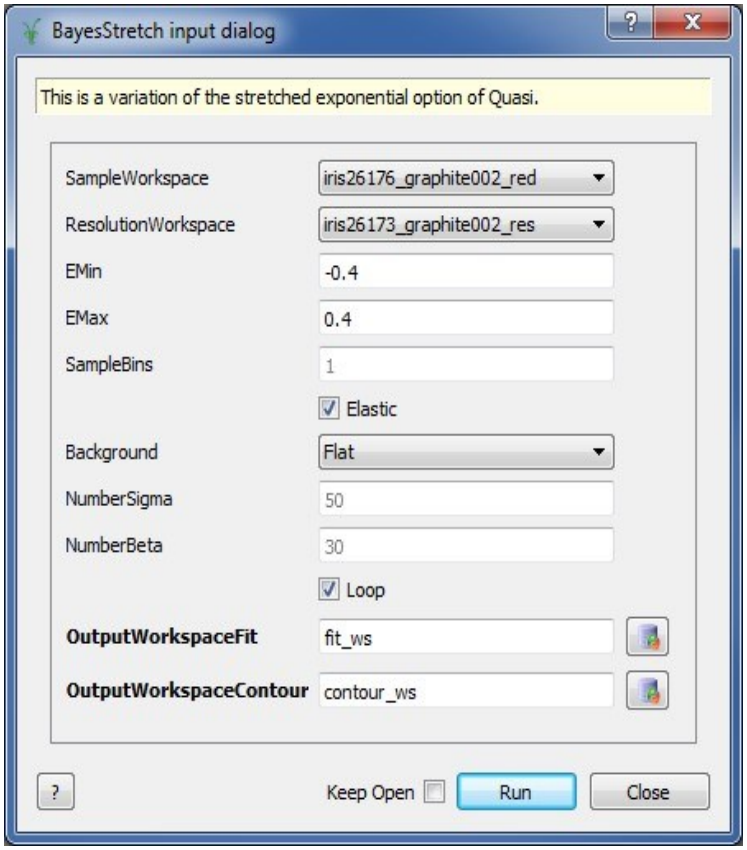

Figure 10: The BayesStretch dialog showing input for iris26176 data set

4.1.1 Fitting Algorithm for Bayesian Analysis of Data, FABADA

An alternative option for Basyian analysis is the FABADA minimizer, which can be accessed using the MANTID fitting wizard [9] , as shown in **Figure 11**. As of MANTID 3.8, the minimizer has been improved to include ergodicity and ties between parameters. It also has improvements for dealing with false convergences. An option to use FABADA as the fitting minimizer is included in the I(Q, t)Fit and ConvFit interfaces, as shown in **Figure 12.**

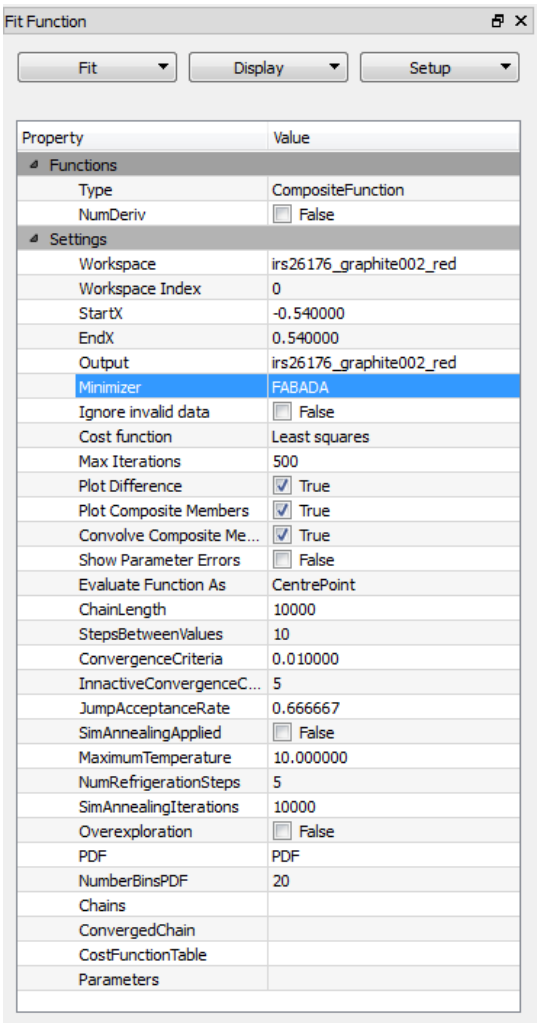

Figure 11: The MANTID Fit Wizard showing the FABADA minimizer highlighted with associated parameters and options

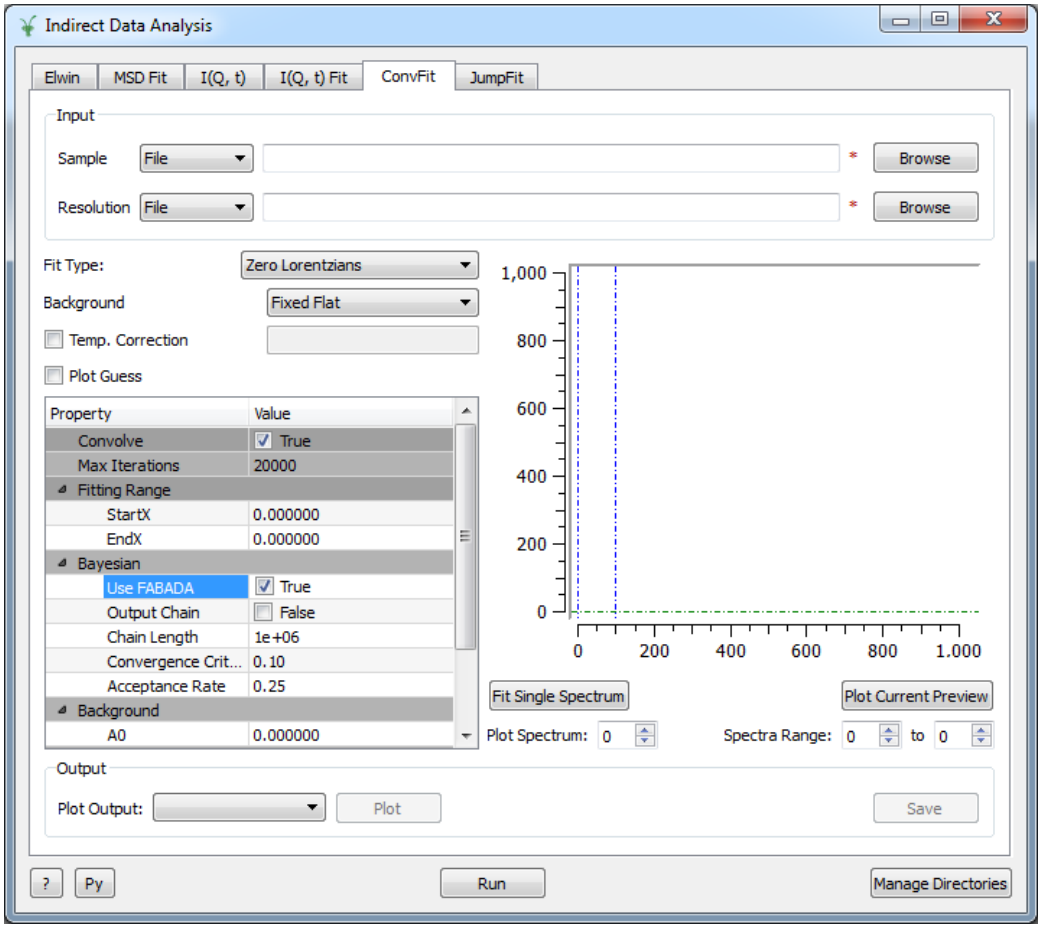

Figure 12: The ConvFit interface with the option to use the FABADA fitting minimizer highlighted.

4.2 Internal function ties for VESUVIO

With the introduction of the MultivariateGaussianComptonProfile fit function in release 3.7 of MANTID, the need to add more ties to functions when fitting VESUVIO data became more prevalent. Whilst Mass is normally a fixed parameter for the MultivariateGaussianComptonProfile, SigmaX, SigmaY and SigmaZ are parameters that are left free or constrained to within a certain boundary. However, with the addition of the tie *SigmaX = SigmaY,* the MultivariateGaussian can be made in made into a BivariateGaussian.

To add this tie, the driver script implementation as well as the underlying code has been updated. The tie is defined alongside the definition of the mass profiles in the driver script. For the example of the BivariateGaussian this could be done with:

flags['masses'] = [{'value':1.0079, 'function': 'MultivariateGaussian', 'SigmaX': 5, 'SigmaY': 5, 'SigmaZ': 5, 'ties': 'SigmaX=SigmaY'}]

The *'ties'* entry in the mass profile defines that the SigmaX value is tied to the SigmaY value for this mass profile. This ties is local only to this function and the parser in the underlying code will treat it as such as it will translate to f0.SigmaX = f0.SigmaY in the ties for all fitting (f0 assuming the MultivariateGaussian is the first function).

So far only local ties (as described above) have been implemented for VESUVIO however a future development (see section 6.4) would be for ties between different mass profiles.

For more information on the MultivariateGaussianComptonProfile see the online documentation at: http://docs.mantidproject.org/v3.7.1/fitfunctions/MultivariateGaussianComptonProfile.html#func-multivariategaussiancomptonprofile

5 Additional changes

5.1 Refactoring ApplyPaalmanPings interface

The ApplyPaalmanPingsCorrection algorithm is used in the Indirect Corrections interfaces to apply corrections to the data with a simple example case of this is container subtraction. During release 3.7, the option to apply a shift in the x axis of the container was added to the interface to allow for container subtraction with off centre peaks. This is explained in more detail in reference [5] section 3.1. This change was applied to both the Container Subtraction interface as well as Apply Paalman Pings interface. To reduce code duplication, this functionality has now been refactored from the interface code into the algorithm in a more generic form. The GUIs still perform the same operation as before however now the underlying algorithm ApplyPaalmanPingsCorrection performs the shift in the x axis of the container. The updated documentation for this algorithm can be found here: http://docs.mantidproject.org/v3.8.0/algorithms/ApplyPaalmanPingsCorrection-v1.html

5.2 ASCII Data reading

When saving ASCII data, normally, unless otherwise stated, the line ending characters may vary between different application and more often between different operating systems. Whilst MANTID uses the universal line ending character when saving ASCII data, an example was found in the IndirectNeutron.py helper script that was using the windows specific line ending to read in data. This has since been rectified and the line ending characters from files from any operating system can now be read successfully in IndirectNeutron.py.

5.3 ISIS Calibration log files

The ISIS Calibration interface is used to load in calibration files for data collected on specific instruments. The interface works by specifying a run number (or range of run numbers) to be loaded as well as the matching calibration file. The runs are loaded using the LoadRaw algorithm and this, by default loads the log files that contain additional data about the run along with the raw data files. Some users have expressed that for speed and data access reasons, it is not always suitable to load log files.

An additional option in the ISIS Calibration interface has been added to specify if you wish to load the log files with the raw data or not. The update has also been made in the IndirectCalibration algorithm and the default for loading log files has been set to false. The updated documentation, as of release 3.8, for this algorithm can be found here:

http://docs.mantidproject.org/v3.8.0/algorithms/IndirectCalibration-v1.html?highlight=indirectcalibration

6 Future

6.1 General Refactoring

6.1.1 Importing mantid.simpleapi

The mantid.simpleapi is a python API used within MANTID to access algorithms from both C++ and python. Unlike the mantid.api, the mantid.simpleapi only loads in the algorithm definitions for calling each algorithm and the implementation is then loaded in later on when the algorithm is accessed. In many of the python algorithms in indirect, the following import statement is used:

*from mantid.simpleapi import **

In isolation, this is not overly problematic other than the slight increase in overheads to import all the algorithm definitions rather than just those that are required. There can be such a case when an algorithm calls a script which also imports the simpleapi. When this happens, as the algorithm has yet to be registered the to the MANTID AlgorithmFactory (this action is performed at the subscribe statement at the end of the algorithm), the result can be that the import statements disagree on the number of algorithm definition there are. In this scenario, the code will not compile and will be unrunnable.

This error is caught at compile time so does not affect users as non-compiling code would never be distributed; however, it is still bad practice to import in this way. There is however a simple solution to this which is to import the simpleapi as a module and access it as such. This would be done with the import statement:

import mantid.simpleapi as s_api

When imported as a module, the simpleapi replaces itself if it re-imported elsewhere and this means that the number of algorithm definitions will always be correct. This change also has a minor impact on the code as, with this import statement, algorithms should be called from the module e.g.:

s_api.CloneWorkspace(…)

As this error is found at compile time, it is a reasonably low priority to make this change in the code base but is worth thinking about when resolving refactoring issues and writing new algorithms.

6.1.2 ADS access

The ADS or Analysis Data Service is a global map in MANTID that has keys of "Workspace Name" and values of "Workspace pointer". A visualisation of this concept is most commonly seen in the MANTID workspace window (**Figure 13**). To following python code will assign the variable "workspace" to a handle to a workspace from the ADS called "Sample":

workspace = mtd['Sample']

Due to the ADS being a map, this call is a small, but noticeable, increase in overheard and should ideally be avoided where possible. This overheard will become more noticeable in the case where the ADS has many entries (workspaces) currently being stored.

Most algorithms in MANTID have workspaces as their input and output as defined by the properties in the *init* function of every algorithm. Each property has a Direction which defines whether a property is an input variable or output variable (Direction defaults to Input when not explicitly stated). When calling an algorithm, all the properties that are output properties (as defined by the Direction) are returned by the algorithm call. This means that for algorithm like *CloneWorkspace* (a simple algorithm which creates an exact copy of a workspace in the ADS) the following can be done to obtain a handle to the clone of the input workspace that it returns:

cloned_workspace = CloneWorkspace(InputWorkspace=input_workspace, OutputWorkspace='ws')

The above code will assign the cloned_workspace variable to an exact copy of the InputWorkspace only with the name 'ws'. Using code like this will remove the need to call *mtd['ws']* to pick up the workspace that is created by the *CloneWorkspace* algorithm from the ADS.

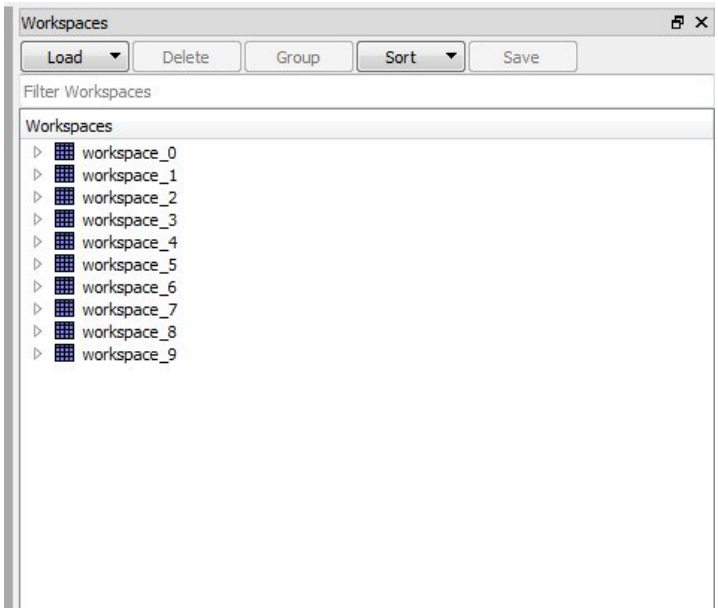

Figure 13: The MANTID workspace window (a visualisation of the Mantid ADS).

6.2 Simulation refactoring

Currently, three algorithms exist that are used for loading NMoldyn data from ascii into Mantid: *LoadNMoldyn3Ascii, LoadNMoldyn4Ascii, LoadNMoldyn4Ascii1D*. These algorithms all share similar workflows and, particularly in the case of LoadNMoldyn4Ascii and the 1D variant, similar code. A two-stage refactoring process should be used to simplify the loading of NMoldyn data across all variants and versions.

The first stage is to identify common elements in the code that can be generalised to fit more cases and refactor this functionality into a separate helper script that can be accessed by all loaders. This will include the code that is used to read in blocks of data. For the LoadNMoldyn4Ascii and the 1D variant, the parsing of the code blocks are very similar and only differ by way of the layout of the data (where 1D data is described in a single column and other version 4 data is a multiple rows of data).

The next stage is to make access to these loaders more seamless. A *LoadNMoldynAscii* algorithm could be created that will determine the type of data that it is given and run the appropriate version of the ascii Loader. **Figure 14** describes these changes from an architectural view.

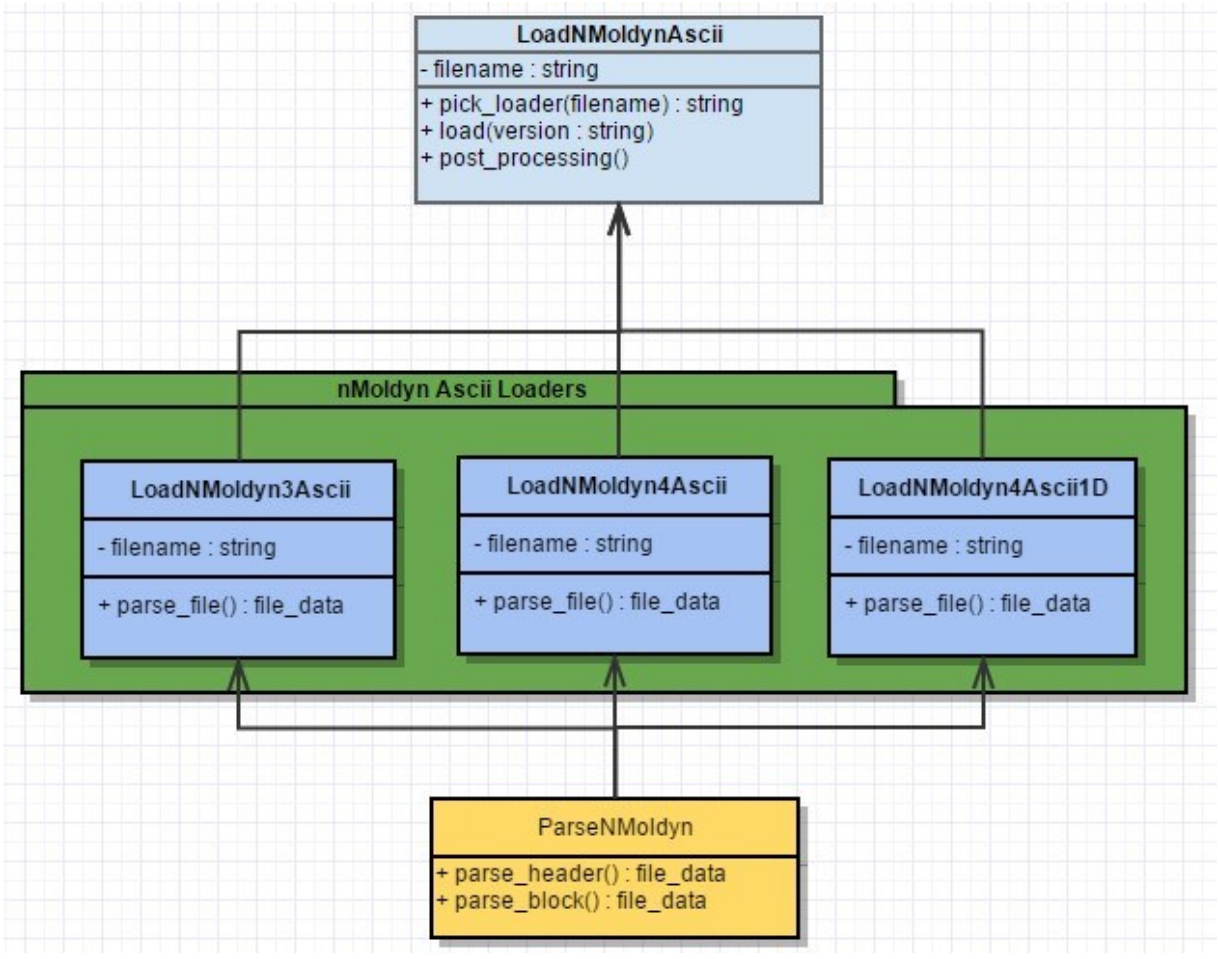

Figure 14: A possible further refactoring design for LoadNMoldynAscii. ParseNMoldyn is a script with functionality for parsing a block of ascii.

6.3 Post save and plot

With the plotting and saving changes made in ConvFit, it is important to begin making this consist with the rest of the indirect interfaces. It should be the case that all interfaces going forward use the post plotting and saving system. However, to improve the user experience, and improve the customisation of option that can be checked when plotting, a multi check combo box could be introduced.

The multi check combo box (example shown in **Figure 15**) would allow a user to select multiple options from a drop down menu. This will improve the current plotting options as they are presently: one parameter, or all parameters. MANTID uses Qt for graphical interfaces and widgets and Qt currently does not have a multi check combo box widget as standard. However, this can be implemented as a new custom widget in MANTID to facilitate this need.

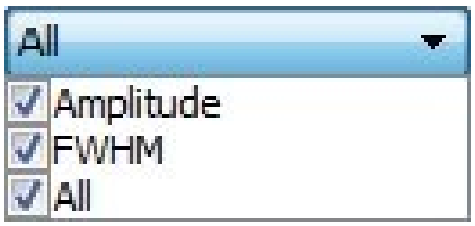

Figure 15: Possible design for a MultiCheckComboBox widget. Checkboxes can be checked and unchecked to select various options at once. Checking 'All' checks/unchecks all options.

6.4 VESUVIO cross mass profile ties

Section 5.1 discusses the implementation of mass profile internal ties for VESUVIO fitting and this simple case could be further improved to facility for ties between differing mass profiles on VESUVIO. A possible implementation for this would be to add a new flag in the driver script called 'ties'. This could take this form:

flags['ties'] = 'f0.width = f1.width'

The aim of the above statement would be to tie the width parameter of the first mass profile to the width of the second mass profile. This implementation is reliant on users understanding the *fx* notation used in MANTID fitting but is otherwise simple to define.

With regards to the implementation, the value from the ties flag should first be parsed and validated to ensure it is a valid tie. Following this, the parsed ties string becomes an input property of the *VesuvioTOFFit* algorithm which will apply the ties when *Fit* is called.

Bibliography

[1] D. Nixon, “Simulation support within MANTID for molecular spectroscopy,” Technical Report RALTR-2015-012, Rutherford Appleton Laboratory, 2015.

[2] MantidProject, “MantidProject.org,” [Online]. Available: http://www.mantidproject.org/Main_Page [Accessed 08 2016].

[3] Róg, Tomasz, et al. "nMoldyn: a program package for a neutron scattering oriented analysis of molecular dynamics simulations," J. Comp. Chem , **24**, 657, 2003.

[4] S. J. Clark, M. D. Segall, C. J. Pickard, P. J. Hasnip, M. J. Probert, K. Refson, M. C. Payne, "First principles methods using CASTEP", Zeitschrift fuer Kristallographie ,2005.

[5] W. Howells, V. Garcia-Sakai, F.Demmel, M. Telling and F. Fernandez-Alonso, “MODES User Guide, Version 3.,” Technical Report RAL-TR-2010-006, Rutherford Appleton Laboratory, 2010.

[6] W.S.Howells, “IDA - IRIS Data Analysis,” Technical Report RAL-TR-96-006, Rutherford Appleton Laboratory, 1996.

[7] E. Oram, “An Overview of the development of Indirect Inelastic Data Reduction and Analysis in MANTID between July 2015 - July 2016” Technical Report RAL-TR-2016-011, Rutherford Appleton Laboratory, 2016.

[8] Monserrat, D., et al. "FABADA Goes MANTID to Answer an Old Question: How Many Lines Are There?." Journal of Physics: Conference Series. Vol. 663. 012009, 2015.

[9] Sivia, Devinderjit, and John Skilling.” Data analysis: a Bayesian tutorial”. OUP Oxford, 2006.