Science & Technology Facilities Council

Technical Report
RAL-TR-2019-009

Overview of developments in MANTID relating to Indirect Inelastic Spectroscopy July 2018 - August 2019

R Applin

September 2019

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 446677
email: libraryral@stfc.ac.uk

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

ISSN 1358-6254

Overview of Developments in MANTID relating to Indirect Inelastic Spectroscopy July 2018 - August 2019

Robert Applin

August, 2019

Abstract

The Manipulation and Analysis Toolkit for Instrument Data (MANTID) is an open source cross-platform application that provides a framework for data reduction and analysis relating to neutron and muon techniques.

This report presents an overview of the developments to the Indirect Inelastic neutron spectroscopy technique within MANTID between July 2018 and August 2019. Notable changes include an increase in stability within the Indirect Inelastic interfaces and routines, the implementation of automated testing for the Data Analysis interface, and the introduction of the Indirect Inelastic interfaces to MANTID Workbench. Further development is currently ongoing for the implementation of simultaneous fitting of multiple datasets for the Quasi-Elastic Neutron Scattering (QENS) fitting interfaces.

Contents

Acknowledgements

I would like to thank Dr Spencer Howells and Dr Sanghamitra Mukhopadhyay for their assistance and scientific contributions relating to the outlined developments. Additional thanks are given to the Molecular Spectroscopy Group (MSG) at the ISIS Neutron and Muon Source and the Sine2020 project for funding the work detailed in this report. Special thanks are also given to the MANTID team for their contributions and assistance in providing the framework for the routines mentioned in this report.

1 Introduction

The MANTID project is a cross-platform software package which provides a framework used for the manipulation and analysis of neutron scattering and muon spectroscopy data [1].

A large ongoing collaboration between numerous international facilities is taking place for the development of this framework, including the Spallation Neutron Source (SNS) based at Oak Ridge National Laboratory, US, the ISIS Muon and Neutron Source at Rutherford Appleton Laboratory, UK, the Institut Laue-Langevin (ILL), France, and the European Spallation Source (ESS) in Sweden.

MANTID provides advanced data reduction and analysis capabilities for a large number of independent research techniques, including Indirect Inelastic neutron spectroscopy. For this technique, the predominant method of interaction between users and the MANTID framework is through custom-designed interfaces. Such interfaces, if well designed, can facilitate the effortless manipulation of complex processes within a framework. However, the smallest of internal or visual design imperfections within an interface can become a hindrance to someone's work. The problem can be further compounded if inadequate steps are taken to resolve defects before new features are developed. Consequently, as implied by the contents of this report, a balancing act between code maintenance and new feature development must be reached.

This report will begin by introducing what a software flaw (or 'bug') is, before subsequently providing explicit examples of problems detected and corrected within various Indirect Inelastic interfaces and routines between July 2018 and August 2019. The report will then go on to convey the importance of carefully considered and planned software development as a preventative measure to the accidental inclusion of bugs within software. Supplementary to this, the significance of automated testing as a tool to pre-emptively catch bugs, as implemented in the Indirect Data Analysis interface, will be introduced. Building upon these principal ideas, further developments within the Indirect Inelastic interfaces and routines will be described with the purpose of providing an overview of developments in MANTID relating to Indirect Inelastic neutron spectroscopy.

2 Software bugs and stability improvements

2.1 What is a bug?

A bug is an error or flaw within a computer program or software system which causes the system to produce an incorrect or unexpected result, or to behave in an unintended way.

Simple programs with foreseeable behaviours are highly likely to be bug-free, however the inevitability of bugs within large software projects with sophisticated logic, such as MANTID, is undeniable. Such software flaws can cause a large amount of frustration and annoyance to users, and can significantly inhibit their work, so the importance of finding and fixing these bugs cannot be understated. Regrettably for some bugs, this is easier said than done, nevertheless it is still achievable when detailed and unambiguous instructions for their replication are provided. The following sections detail two reproducible MANTID software bugs fixed between July 2018 and August 2019.

2.2 Identifying noisy detectors for a summed data reduction

The reduction of RAW time-of-flight data to energy transfer using the Data Reduction interface is the first step within the Indirect work-flow. To facilitate this calculation, the *ISISIndirectEnergyTransfer* algorithm is executed using a set of well-defined input and output properties specified by a user [2].

A subsidiary collection of these properties are optional and can be employed to specify the manner in which the data is processed. For instance, the 'Sum Files' property can be used to determine a supposed 'average' of a batch of runs before treating it as a single file during the data reduction process. Naturally, the resulting reduced data, hereinafter referred to as a summed reduction, should also resemble an 'average'. However, this was not the case as discovered when reducing a batch of TOSCA runs in MANTID version 3.9. Figure 1 shows four reduced spectra; three of which depict the result of reducing each run in the batch individually, and the remaining (red) spectrum representing the summed reduction.

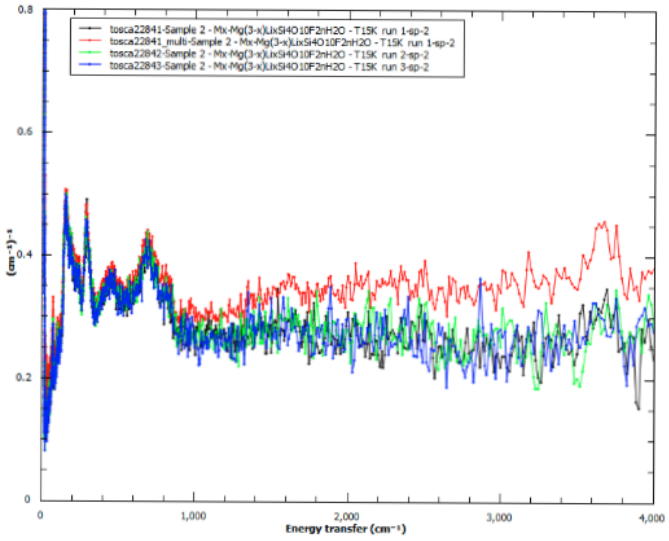

Figure 1: A bug discovered when reducing TOSCA runs using the *ISISIndirectEnergyTransfer* algorithm, MANTID version 3.9.

Evidently, the intensity of the summed reduction (red line) is incorrect. A consequential investigation uncovered that some detectors were masked unnecessarily during this specific reduction case, while other noisy detectors were not masked at all. Thus, the necessity for a different approach when identifying noisy detectors was apparent. This new approach is outlined by Figure 2.

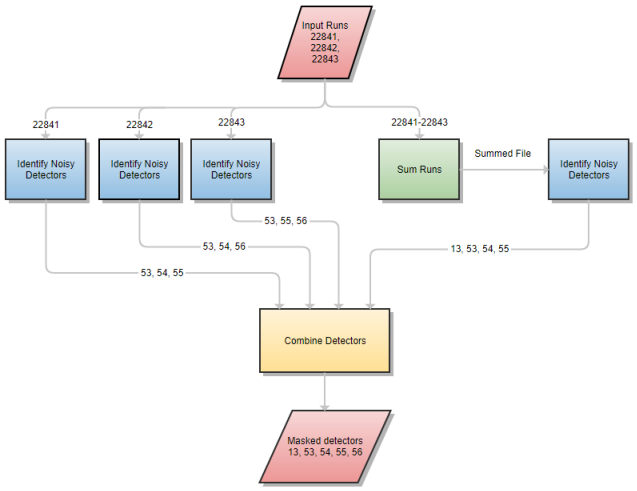

Figure 2: The new approach for identifying noisy detectors when performing a summed reduction.

In this approach, the noisy detectors for each run within the batch are individually identified using the *IdentifyNoisyDetectors* algorithm before being combined with the summed run's noisy detectors [3]. As a consequence of this, the summed reduction produces the expected result as seen in Figure 3. This implementation can be found in MANTID version 4.0, released on March 25th 2019, and subsequent versions.

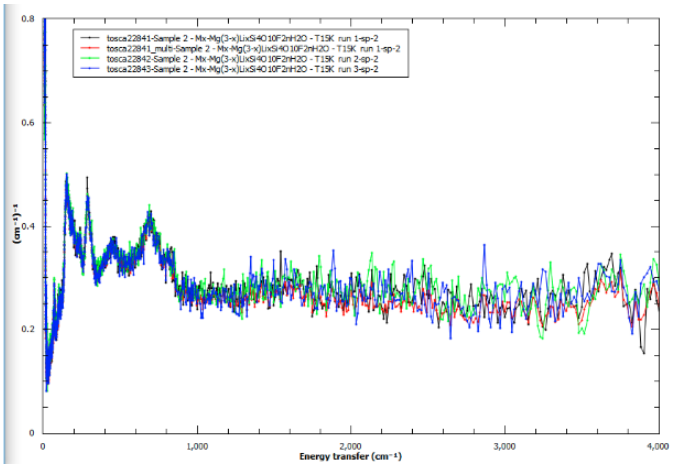

Figure 3: The result of performing a summed reduction using TOSCA runs in MANTID 4.0.

2.3 The calculation of Monte Carlo errors for I(Q, t)

A natural continuation of the indirect work-flow, following from the reduction of data, would be an analysis of this data. The Data Analysis interface provides several custom routines to aid this endeavour, including an interface to compute I(Q, t) using a sample and resolution file. In addition to this, the interface will calculate the error values for this data using a Monte Carlo implementation.

Within this implementation, N workspaces are generated pseudo-randomly with respect to predetermined I(Q, t) data. Accordingly, the standard deviation of these workspaces is then used for the error values. Such a procedure is expected to present a positive correlation between noise within data, and the uncertainty of that data. However, this correlation was not present within the I(Q, t) error calculations of MANTID version 4.0, as implied by Figure 4. Notably, the error bars on the right (noisy) portion of the spectrum show no sizeable increase in magnitude, thusly warranting an investigation.

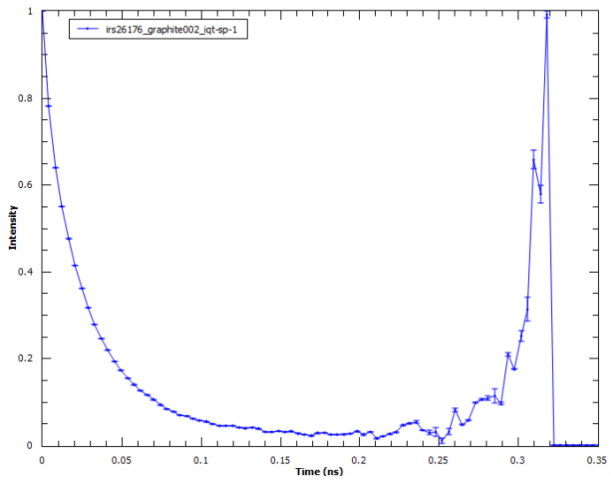

Figure 4: A bug discovered when calculating errors for I(Q, t) data using a Monte Carlo implementation, MANTID version 4.0.

By design, generating numbers using a computer is pseudo-random due to the deterministic and predictable nature of computational devices. As such, a commonly used method for generating "random enough" numbers uses seeds which correspond to a predictable output from a number generating algorithm. Using the same seed while generating multiple sets of supposedly random data will therefore negate any pursuit of randomness. Such an oversight is the cause of the smaller than expected error bars seen in Figure 4. Figure 5 shows the consequence of correcting this bug.

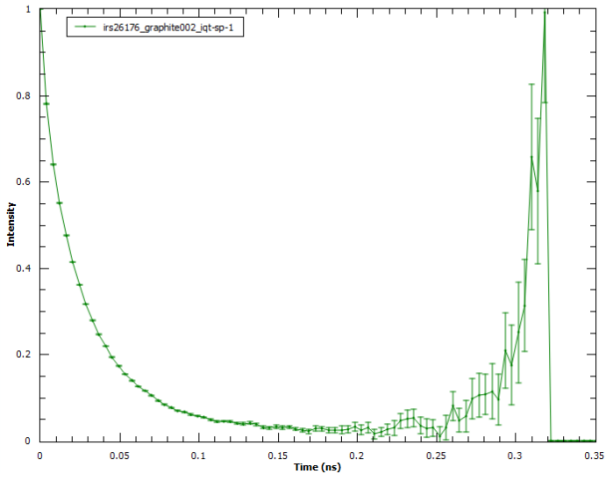

Figure 5: The result of calculating I(Q, t) in MANTID 4.1.

2.4 Additional stability improvements

The two bugs mentioned above only represent a small fraction of the problems fixed within the Indirect interfaces and routines between July 2018 and August 2019.

Further stability improvements have also included an improvement to the validation of input data; a full implementation for the propagation of errors within the *ElasticWindowMultiple* algorithm; and the removal of duplicate workspace history causing large file sizes when reducing a batch of runs on the Data Reduction interface. In addition to this, the ability to reduce antiquated TFXA and TOSCA data has been restored, and a performance improvement when opening the Data Reduction interface has also been accomplished. Lastly to mention, a sizeable effort to harmonize the Data Analysis interface with Project Recovery has been successful.

3 Model-View-Presenter

The effectiveness of a reactive approach to bug fixing is questionable. Undoubtedly, a more proactive approach of carefully considered code design, and a greater reliance on automated testing is greatly preferable. This section details the recent improvements made to code architecture in support of this conviction.

3.1 What is MVP?

Model-View-Presenter (MVP) is a well-established pattern used for the implementation of user interfaces as a means to improve code organisation. Within this pattern, code is separated into three distinct groups with different responsibilities as an intuitive response to its purpose.

The purpose of the *View* is simply to observe and relay actions performed by a user to the *Presenter*, and to give feedback in response to these actions. The *Presenter* acts as a mediator between the *View* and *Model*, and if necessary the higher-level interface set-up. In response to a user action, the *Presenter* will often invoke a state change within the *Model* such as the transformation or alteration of a data member. Such a change within the *Model* will then prompt the *View* to display feedback to the user. Figure 6 provides a visual representation of the MVP pattern.

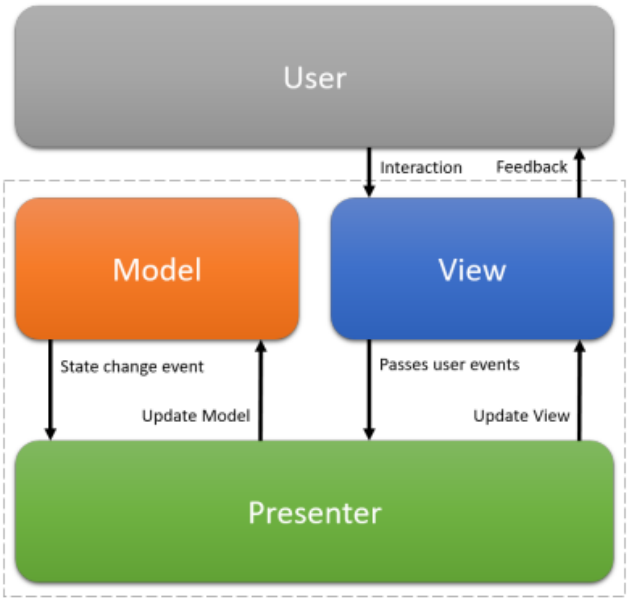

Figure 6: A visual representation of the MVP pattern [4].

As alluded to previously, the possibility of bugs within MVP code is greatly

diminished due to the nature of its carefully considered and planned organization. Such an organization will also ease the process of bug fixing while simultaneously hastening the development of new interface features. Further to this, the separation of code into distinct groups also allows the creation of good-quality automated tests, as described in section 4. Therefore, the importance and relevance of MVP to the continued development of the Indirect Inelastic interfaces is undeniable. The following sections give an overview of MVP-related developments to the Indirect Inelastic interfaces between July 2018 and August 2019.

3.2 A new widget for general plotting

When converting an interface to MVP, it is often beneficial to split the interface into different parts, called "widgets". These widgets are manufactured to be self-standing and are consequently easy to re-use, as is the case for the new general plotting widget. This widget is used to plot output data on an interface, and is seen in Figure 7 and Figure 8.

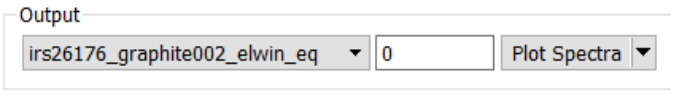

Figure 7: A close-up of the general plotting widget.

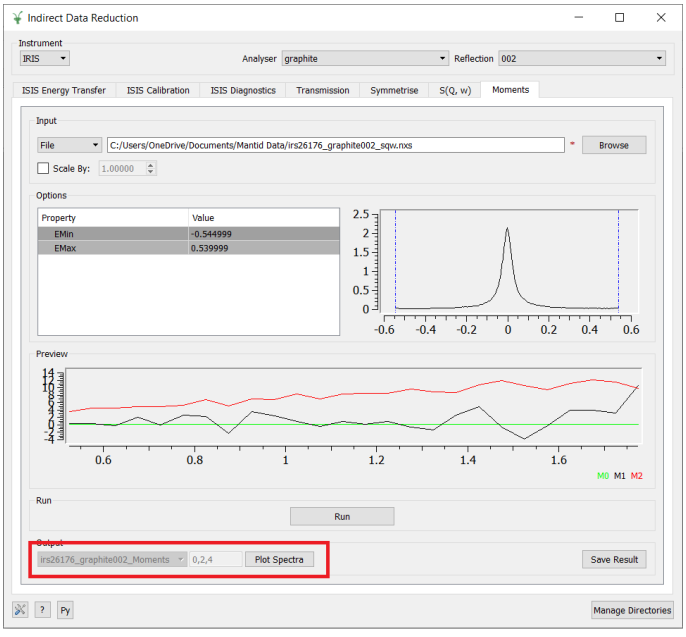

Figure 8: The plotting widget on the Moments tab of Data Reduction.

The general plotting widget is used extensively throughout the Data Corrections, Data Reduction, Simulations and Diffraction interfaces; and is used sparingly on the Data Analysis interface. Figure 8 clarifies the location of this widget on the majority of these interfaces (on the bottom left). The widget consists of three parts:

Workspace ComboBox

- This is on the left of the widget, and is used to select which workspace to plot.

Workspace Indices LineEdit

- This is in the middle of the widget, and is used to select workspace/bin indices.
- Entering 0-2,4,6-7 is equivalent to selecting indices 0, 1, 2, 4, 6 and 7.
- A red asterisk is displayed when an index doesn't exist in the workspace.
- It has an auto-complete feature which will suggest previously entered indices.

Plot Spectra Button

- This is on the right of the widget, and is used to trigger a plot event.
- Plot events include Plot Spectra, Plot Bins, Plot Contour and Plot Tiled.
- The available plot events will depend on the interface being used.
- Figure 9 shows how to use Plot Bins, Plot Contour or Plot Tiled (via the arrow).

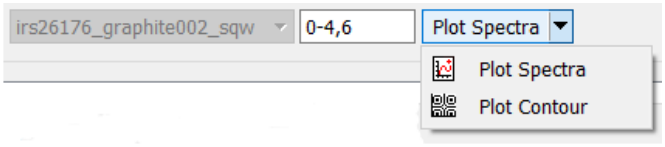

Figure 9: The plotting widget on the S(Q, ω) tab of Data Reduction.

Significant improvements have been made in comparison to the old plotting widgets such as the ability to select multiple workspace indices in a discontinuous manner. Other improvements are not related to the functionality of the widget, but rather to its usability. For example, the *Plot Spectra Button* now encompasses all types of plotting in an attempt to avoid a messy collection of plot options. Additionally, the *Workspace Indices LineEdit* comes with an auto-complete feature. For these reasons, it is hoped that there will be a slight improvement to interface usability.

3.3 A new widget for fit data

A widget used for plotting, editing and saving output data from a fit has been implemented using the MVP pattern for the Data Analysis interface (Figure 10). This widget is used on the MSDFit, IqtFit, ConvFit and F(Q)Fit tabs, and adds to previous MVP developments made between July 2017 and June 2018 [4].

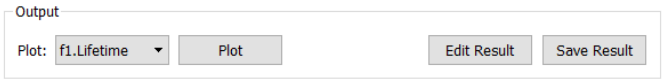

Figure 10: A widget used for plotting, editing and saving output data from a fit.

The appearance and function of the widget is relatively unchanged, however the internal code has been greatly enhanced. The widget is used to perform three tasks:

Plotting data

- Select a fit parameter before clicking Plot.

Editing data

- Facilitates the replacement of fit data as described in section 7.2.
- Only available in IqtFit and ConvFit.

Saving data

- Saves the result of the current fit.

Although the consolidation of this widget has no discernible usability improvements, it's conversion to MVP is assuredly worthwhile. Such an endeavour enables the creation of automated tests, thereby safeguarding the widget from unintended and fallacious alterations in the future. Section 4 will begin by introducing the purpose and importance of automated testing, before subsequently outlining related advancements within the Indirect area between July 2018 and August 2019.

4 Automated testing

4.1 What is automated testing?

Test automation is the use of software, separate from the software being tested, which asserts the equality of predicted outcomes to actual outcomes [5]. A greater reliance on automated testing is an ongoing objective within the Indirect Inelastic interfaces and routines, and within MANTID as a whole, because of its bug prevention capabilities. To facilitate this aim, various types of automated testing are used to examine different behavioural aspects of a software system. The following sections outline recent advancements to *unit* testing and *system* testing within the Indirect area.

4.2 Developments related to unit testing

Simply put, unit tests are responsible for testing fine-details within a program through the examination of individual units of functionality. Suitably, this type of testing is often used for testing interface code which has been systemically broken-up into separate units of functionality. Therefore, as may be reasonably assumed, the new widgets described in section 3 are automatically tested using unit tests. In addition to this, the vast majority of the Data Analysis interface is also unit tested, with the exception of the *FitPropertyBrowser*. Figure 11 attempts to illustrate the compartmentalised testing of the Data Analysis interface, where each coloured box represents a separate unit test implementation (labelled 1, 2, 3 and 4).

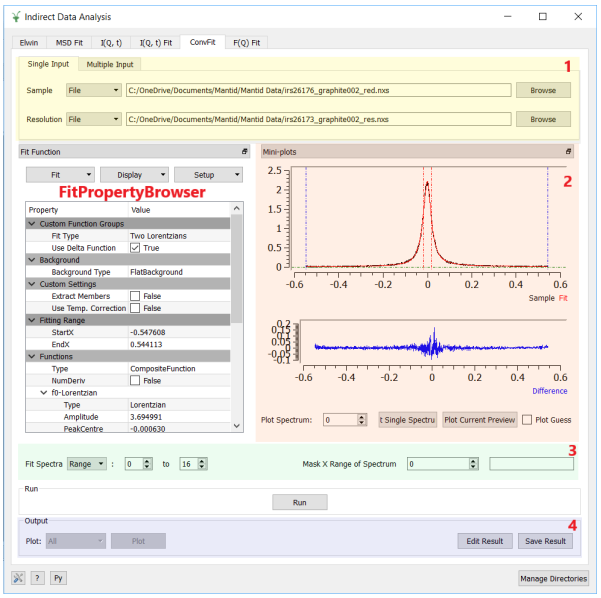

Figure 11: The automated test coverage for the Data Analysis interface.

As seen on the left of Figure 11, the *FitPropertyBrowser* is the only widget which isn't automatically tested. This widget will be replaced upon the advent of Simultaneous fitting (described in section 10.1) with a different widget which is better-designed and unit tested. An example of a unit test is provided in Appendix A.1.

4.3 Developments related to system testing

Contrary to unit tests, system tests are responsible for testing big-picture behaviours of a software system comprising of many units of functionality. Within MANTID, system tests are commonly used to test algorithms by comparing predicted outputs to actual outputs when certain input properties are set. As such, system tests can be devised to prevent the re-appearance of bugs previously fixed within an algorithm, thereby protecting the algorithm from an unintended regression. Accordingly, the following system tests have been added:

- A system test to cover the noisy detectors bug described in section 2.2.
- A system test to cover the MC error calculation bug described in section 2.3.
- A system test to cover a bug discovered in the *SofQWMoments* algorithm (used by the Moments tab of the Data Reduction interface).
- A system test to guarantee the compatibility of old TOSCA runs with the *ISISIndirectEnergyTransfer* algorithm.
- A multitude of existing system tests have also been updated in response to other bug fixes not previously mentioned. An example of a system test is provided in Appendix A.2.

Evidently, the importance of automated testing to the continued maintenance and advancement of the Indirect interfaces and algorithms cannot be understated. It is therefore unfortunate that many Indirect interfaces, including the data reduction interface, do not yet employ unit testing due to their disregard of the MVP pattern. Consequently, further development in relation to MVP and automated testing should be considered a high priority.

5 Data Reduction

The data reduction interface is primarily used to produce reduced data by performing a conversion of raw time-of-flight data to energy transfer on the Energy Transfer interface. In addition to this, the Calibration interface can be used to produce calibration and resolution workspaces, while the S(Q,ω) interface is used to calculate the dynamic structure factor. This section aims to give a brief overview of previously unmentioned developments within the data reduction interface.

5.1 Developments in ISIS Energy Transfer

The ISIS Energy Transfer interface is used to perform a conversion from raw time-of-flight data to energy transfer using the *ISISIndirectEnergyTransfer* algorithm, and provides a wide range of options including options for the grouping of detectors, and for the re-binning of data. The following subsections outline the changes made to this interface.

Group Output

Version 4.1 of MANTID introduced the 'Group Output' option, which is used for the grouping of output reduced workspaces (Figure 12). This option can be used to prevent congestion in the 'Workspaces List' while reducing a large batch of runs.

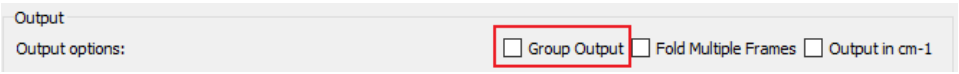

Figure 12: The 'Group Output' option in ISIS Energy Transfer.

Detailed balance

As of MANTID version 4.1, the 'Detailed balance' is automatically loaded from the sample logs of the first run entered into the interface. If the relevant sample log cannot be found, a default value of $300K$ is used.

5.2 Developments in S(Q, ω)

The S(Q, ω) interface is used to calculate the dynamic structure factor of reduced data produced on the Energy Transfer tab. The following subsections outline the changes made to the S(Q, ω) interface.

Embedded contour plot

Version 4.0 of MANTID introduced an embedded contour plot for the S(Q, ω) interface (Figure 13) which is used to automatically plot the selected reduced data.

This feature makes it easier to determine the Q and Energy binning to use when executing the *SofQW* algorithm.

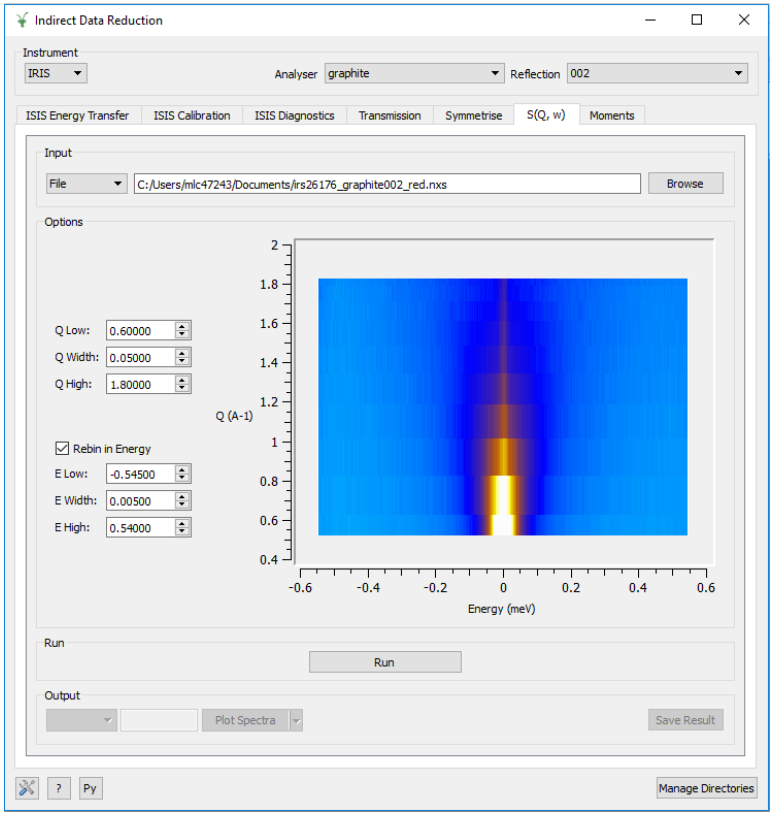

Figure 13: The contour plot on the S(Q, ω) tab of Data Reduction.

Automatic determination of Q and Energy Binning

The $Q_{low/high}$ and $E_{low/high}$ binning properties are now automatically calculated upon loading reduced data in MANTID version 4.1. For example, to calculate Q_{low} and Q_{high} using a pre-determined value for Q_{width}, the limits on the Q axis of the contour plot are used as an initial guess for Q_{low} and Q_{high}, before they are adjusted to satisfy Equation 1:

$$Z = \frac{Q_{high} - Q_{low}}{Q_{width}} \tag{1}$$

where Z is any integer number. This process is repeated when calculating E_{low} and E_{high}.

6 QENS Corrections

6.1 Calculating absorption corrections

During quasi-elastic neutron scattering, some neutrons are absorbed by the container and sample, and consequently it becomes necessary to calculate corrections for the reduced data produced from each run. This can be done on the Indirect Corrections interface using either a Paalman-Pings or Monte-Carlo implementation first introduced in MANTID version 3.11. Table 1 outlines the absorption factors used for these calculations [4].

Symbol	Scatter From	Absorbed By
$A_{s,s}$	Sample	Sample
$A_{s,sc}$	Sample	Sample, Container
$A_{c,c}$	Container	Container
$A_{c,sc}$	Container	Sample, Container

Table 1: Description of the absorption factors.

The Paalman-Pings method utilizes all four factors as seen in Equation 2 [4]:

$$I_s = \frac{1}{A_{s,sc}} \left(I_{sc}^E - K_c I_c^E \frac{A_{c,sc}}{A_{c,c}} \right) \tag{2}$$

where I_s is the corrected sample data, K_c is the container scale factor, I_{sc}^E is the sample data before corrections are applied and I_c^E is the container data. Meanwhile, the Monte Carlo method currently only utilizes two of the absorption factors [4]:

$$I_s = \frac{I_{sc}^E}{A_{s,s}} - K_c \frac{I_c^E}{A_{c,c}} \tag{3}$$

6.2 Using sample and container cross sections

MANTID version 4.1 introduced an alternative method for defining the sample and container's configuration on the Calculate Paalman-Pings and Calculate Monte-Carlo Absorption interfaces. The alternative is to specify the coherent, incoherent and attenuation cross-sections instead of a chemical formulae in the 'Sample Details' or 'Container Details' section of the interfaces as seen in Figure 14.

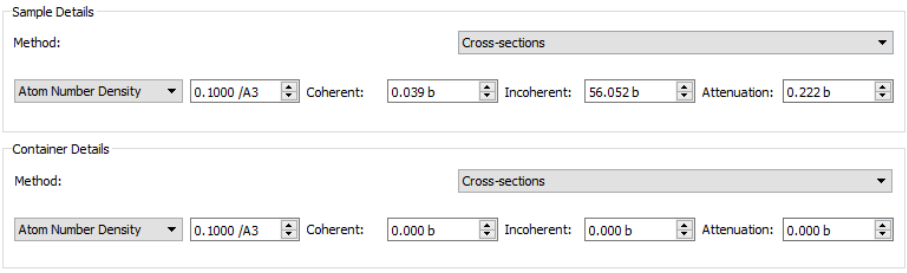

Figure 14: The coherent, incoherent and attenuation cross-sections in the Calculate Paalman Pings interface, measured in barns.

To switch between using the chemical formulae or cross-sections, the desired method is selected in the box at the top of the 'Sample Details' or 'Container Details' sections. Figure 14 illustrates the options that are visible when the cross-sections method is selected.

7 QENS Data Analysis

The Data Analysis interface is an instrument-independent interface which provides a wide range of data analysis capabilities for reduced and $S(Q, \omega)$ file formats. These abilities include the option to interact with the *ElasticWindow* algorithm, and the ability to calculate $I(Q, t)$. In addition to this, it can also be used for four different types of fitting: Mean Square Displacement fitting, fitting $I(Q, t)$, convolution fitting and jump diffusion fitting. Further information about how to use the Data Analysis interface can be found online [6, 7]. This section aims to give an overview of previously unmentioned developments within the Data Analysis interface.

7.1 Developments within the fitting tabs

Several improvements to the fitting tabs (i.e. MSDFit, IqtFit, ConvFit and F(Q)Fit) have been made with the aim of improving their usability. The mini-plots within the fitting tabs can now be floated to allow them to be moved and resized separate to the interface. Supplementary to this, the relative sizes of the top and bottom mini-plots can be adjusted by dragging a 'handle' located between the two plots as demonstrated in Figure 15. In general, this improvement is useful when using devices with small screens.

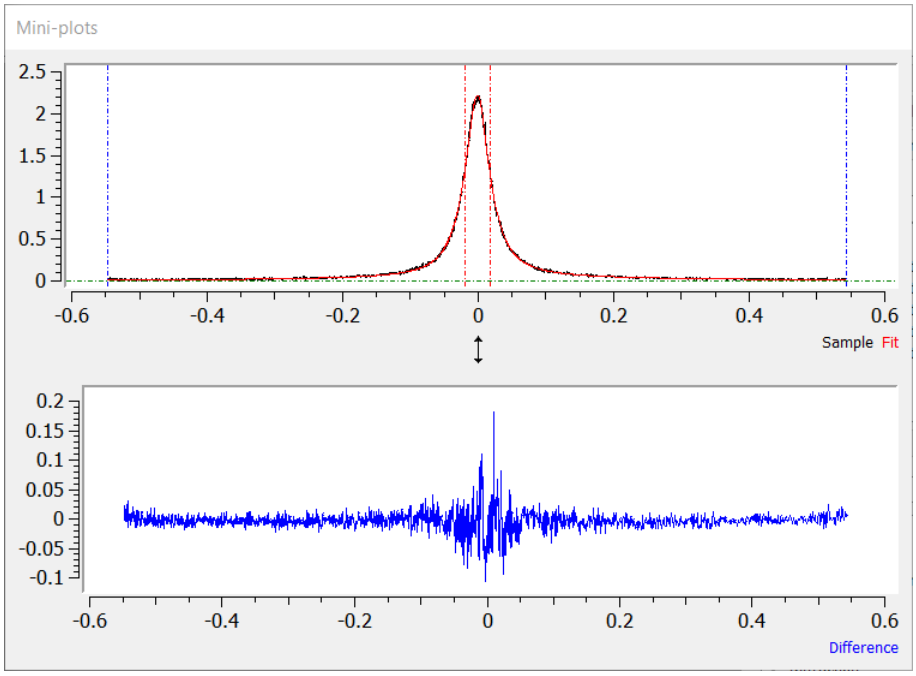

Figure 15: Adjusting the relative sizes of the mini-plots on the fitting tabs.

Further usability improvements such as the rearrangement of features within the tabs, and the disabling of various buttons and options for the duration of a fit have also been enforced. Other improvements are not related to usability, and instead provide new capabilities. For instance, it is now possible to plot χ^2 using the output plotting options after performing a sequential fit. Figure 16 depicts a plot of χ^2 following a convolution fit using two lorentzians and a flat background.

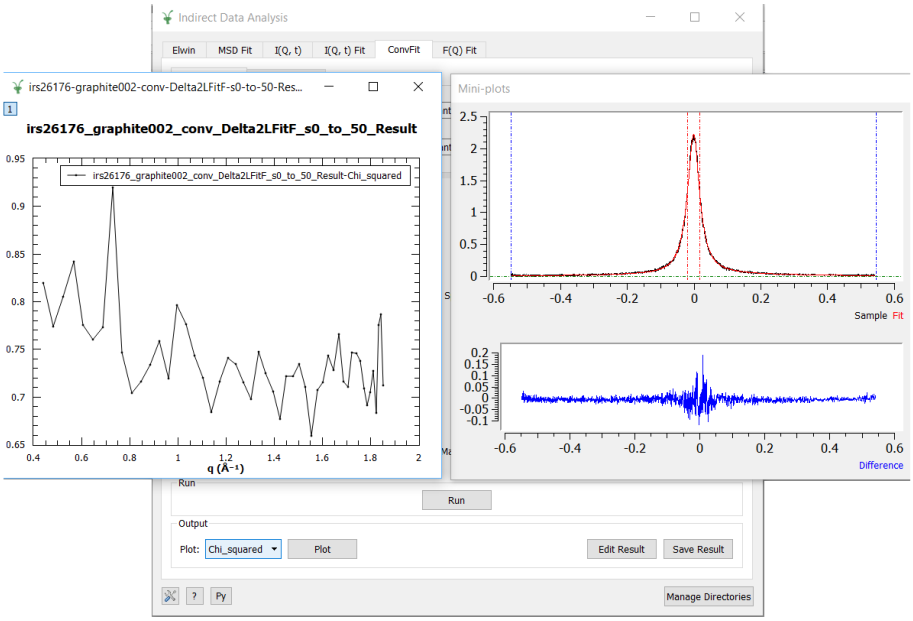

Figure 16: Plotting χ^2 using the output options on ConvFit.

In addition to these shared improvements seen across all of the fitting tabs, several minor improvements relating to individual tabs have also been made:

- A new temperature unit (in Kelvin) has been introduced to MANTID for use in the output plots in MSDFit.
- IqtFit and ConvFit provide plotting options for PDF data after using a FABADA minimizer.
- ConvFit allows the fitting of dave ASCII files.

7.2 Replacing a poor fit result

During a sequential fit in QENS Data Analysis, the output parameters from fitting one spectrum are used as the starting values for the next. In general, this leads to better fits for the latter spectra because the starting values for the parameters get progressively more accurate, however a poor fit is still possible if two consecutive

spectra are substantially different. Moreover, unreasonable starting values for the first spectrum in a fit can also cause a poor fit, and therefore an option to replace this fit result would be useful. Consequently, an option to replace a poor fit result on IqtFit and ConvFit was introduced in MANTID version 4.0.

To perform this replacement, the poorly fitted spectrum should first be re-fitted (to get the replacement fit) by selecting the poor fit using the 'Plot Spectrum' option, and then clicking 'Fit Single Spectrum'. The Edit Result dialog box, as seen in Figure 17, should then be opened by clicking 'Edit Result'. This dialog box asks for the result workspace containing the poor fit result (labelled 1), and the result workspace containing the replacement fit (labelled 2). Clicking 'Replace Fit Result' will locate and replace the corresponding fit data in workspace (1) with the single fit result found in workspace (2). Figure 17 depicts the replacement of a spectrum with index 1 within a set of fit results containing 51 spectra.

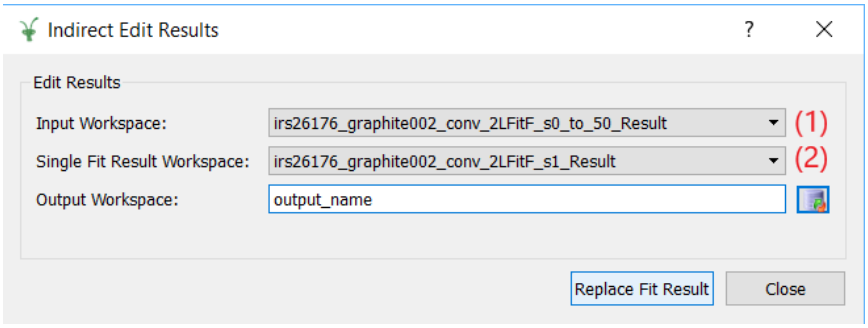

Figure 17: The Edit Result dialog box.

The *IndirectReplaceFitResult* algorithm is executed to perform this operation [8]. A more general algorithm called *CopyDataRange* can also be used to replace a block of data in one workspace with a block of data from another workspace [9].

7.3 Additional Improvements

The developments to the QENS Data Analysis interface mentioned above only highlight a small number of improvements made to the interface between July 2018 and August 2019. Further improvements have also been made to the I(Q, t) interface including the addition of a progress bar to track the progress of the Monte Carlo error calculations; the option to skip the calculation of Monte Carlo errors; and the ability to select an asymmetric energy range. In addition to this, the ability to avoid loading workspace history, and a full implementation for the propagation of errors within Elwin has been added. Lastly to mention, a sizeable effort to harmonize the Data Analysis interface with Project Recovery has been successful.

8 Indirect Settings GUI

MANTID version 4.1 saw the introduction of the Indirect Settings graphical user interface (GUI), seen in Figure 18. The Indirect Settings GUI provides a selection of options which can be used to specify a users preferences relating to common functionality within the indirect interfaces.

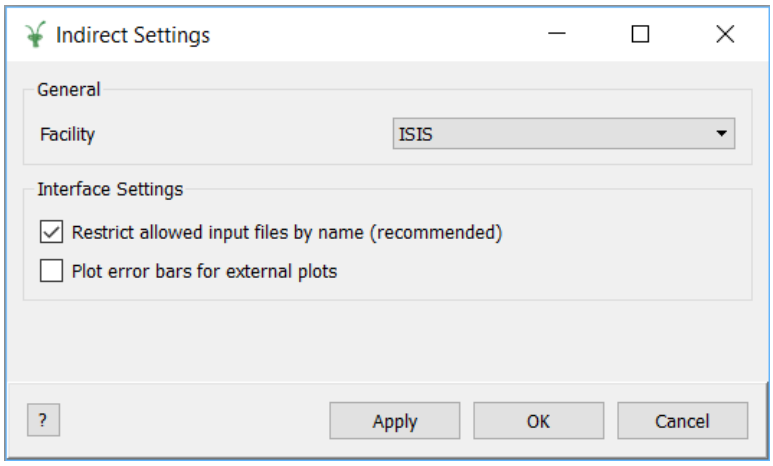

Figure 18: The Indirect Settings GUI.

Facility
Select the facility (e.g. ISIS, ILL, etc.).

Restrict allowed input files by name
If unchecked, this option allows input files with any name to be loaded into an interface. Previously, only files with certain name endings, such as '_red', could be loaded. Further information, including a glossary of data restrictions, is provided in the online documentation for the Indirect Settings interface:
`https://docs.mantidproject.org/v4.1.0/interfaces/Indirect%20Settings.html`

Plot error bars for external plots
If checked, error bars will be plotted for external plots produced by the indirect interfaces. Figure 19 shows the result of plotting I(Q, t) with and without error bars on MantidPlot.

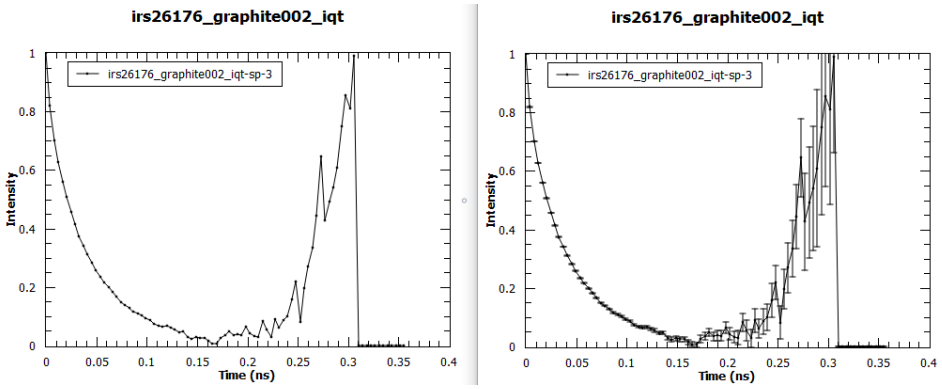

Figure 19: A plot of I(Q, t) with and without error bars.

Traditionally in the indirect interfaces, a naming convention is often used to restrict the data files that can be loaded into an interface with the purpose of preventing the selection of invalid data. For instance, a common restriction would be to only allow files with names ending in '_red' (meaning a reduced file) into an interface used to manipulate reduced data. This data restriction is important for preventing unexpected problems in a programs execution if an invalid file is selected, however it can also prevent the use of an interface if users do not follow these naming conventions. Consequently, one motivation for the creation of this interface was to allow users to turn off this restriction.

This interface also now allows the extraction of common functionality seen across all of the indirect interfaces thereby enabling the re-use of code, and improving the consistency of interfaces. Therefore, the continued development of this GUI is important for the advancement of the Indirect interfaces.

9 Moving to the Workbench

The Mantid Workbench, which was released as part of MANTID 4.0, will eventually replace MantidPlot becoming the primary interface between the user and the mantid framework. The Workbench has been systematically built from scratch with the aim to improve usability, stability and test coverage within MANTID. Further to this, the Workbench utilizes the plotting functionality provided by matplotlib, allowing publication-quality plots with increased flexibility for customization [10].

At the present time, all of the Indirect algorithms have been transferred to the Workbench, as well as the Bayes, Corrections, Data Reduction, Diffraction, Settings, Simulation and Tools interfaces. The functionality of these interfaces is largely unchanged, with the main differentiation coming from visual differences for the embedded and external plots. Figures 20 and 21 show what the Data Reduction and Corrections interfaces look like in the Workbench. Finally, it is anticipated that the Data Analysis interface, seen in Figure 22, will be transferred to the Workbench before the release of MANTID version 4.2.

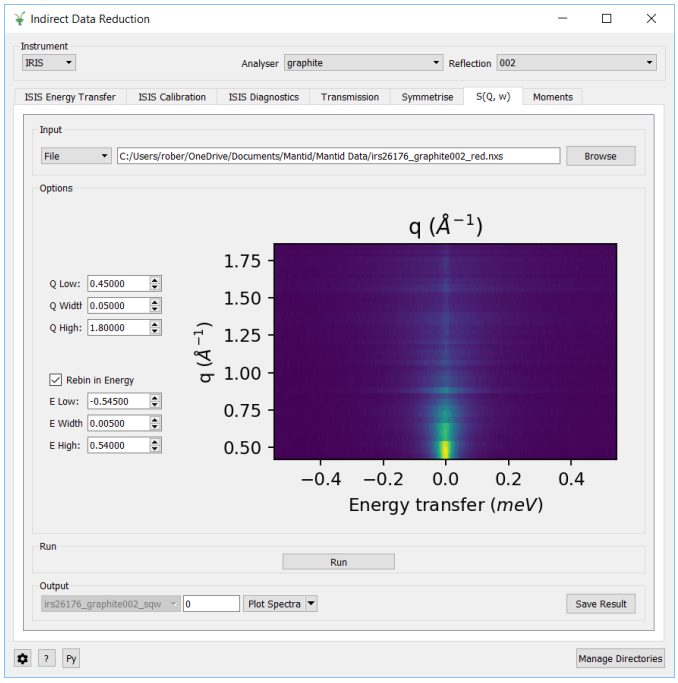

Figure 20: The Data Reduction interface in the Workbench.

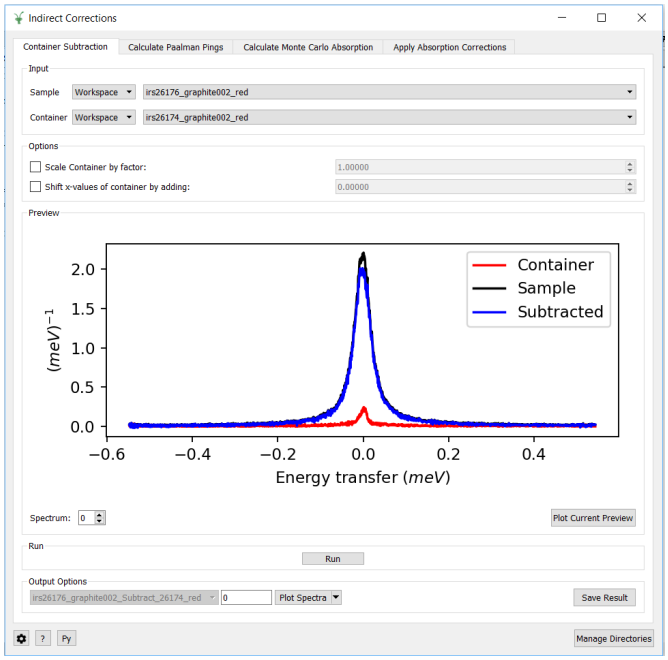

Figure 21: The Corrections interface in the Workbench.

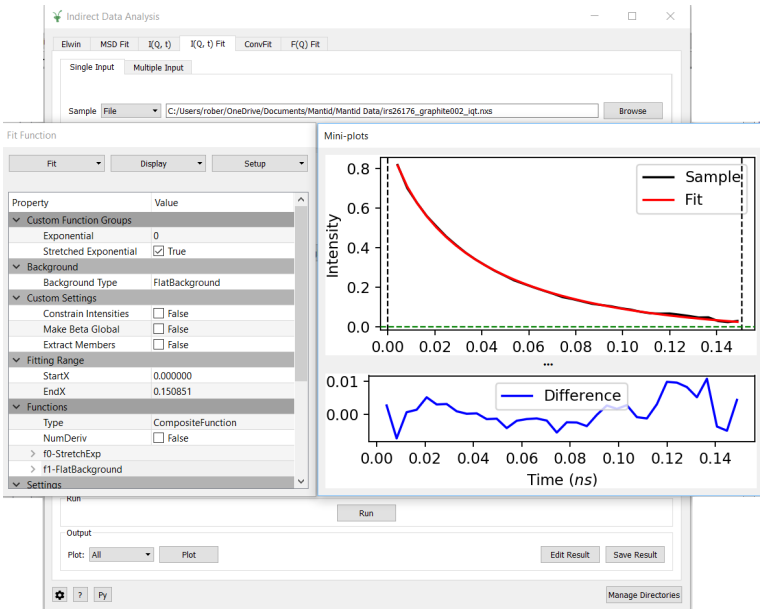

Figure 22: The Data Analysis interface in the Workbench.

10 Prospective developments

10.1 Simultaneous fitting

It is an ongoing objective to implement simultaneous fitting for the QENS Data Analysis interface. In contrast to sequential fitting, simultaneous fitting is where all of the spectra selected for a fit are fitted concurrently as opposed to one after another. This allows the selection of 'global' fit parameters which are shared across all of the spectra being fitted. This development warrants the replacement of the *FitPropertyBrowser* with the *FunctionBrowser* and the *FitOptionsBrowser* as seen in Figure 23. Figure 23 shows the old *FitPropertyBrowser* alongside what the new *FunctionBrowser* and *FitOptionsBrowser* will look like.

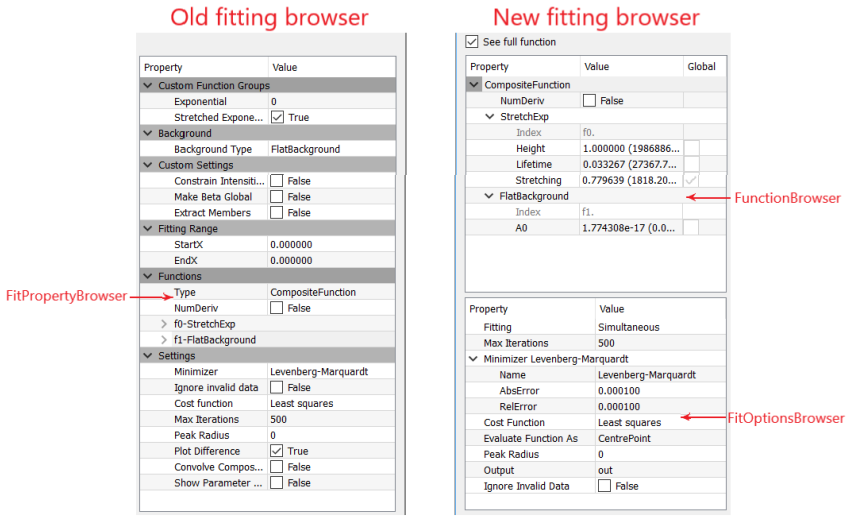

Figure 23: The FitPropertyBrowser, FunctionBrowser and FitOptionsBrowser.

FunctionBrowser

- Used to select the fit functions you want to use for a fit.
- Displays the fit functions in two different formats (toggled by ticking the 'See full function' check box).
- Used to specify whether a fit parameter is global (by ticking the right-most check box).

FitOptionsBrowser

- Used to choose between a sequential or simultaneous fit.
- Used to select the minimizer and various other fitting options.

10.2 Rearranging the reduction and analysis interfaces

Rearranging the Data Reduction and Data Analysis interfaces will allow an MVP approach (described in section 3) to be adopted on the reduction interface, and will also improve the usability of both interfaces. These improvements are made possible because the new arrangement, which includes the creation of the Data Manipulation interface, will group together the tabs as a natural response to their purpose. For instance, the Data Analysis interface will comprise of the fitting tabs only, while the Data Reduction interface will exclusively contain tabs which operate on run numbers. The new arrangement of these interfaces might follow the plan outlined in Table 2:

Interface	**Current arrangement**	**After rearrangement**
Data Reduction	ISISEnergyTransfer ISISCalibration ISISDiagnostics Transmission Symmetrise $S(Q, \omega)$ Moments	ISISEnergyTransfer ISISCalibration ISISDiagnostics Transmission
Data Analysis	Elwin MSDFit I(Q, t) I(Q, t)Fit ConvFit F(Q)Fit	MSDFit I(Q, t)Fit ConvFit F(Q)Fit
Data Manipulation	-	Symmetrise $S(Q, \omega)$ Moments Elwin I(Q, t)

Table 2: A plan for rearranging the Data Reduction and Data Analysis interfaces.

Following the rearrangement of these interfaces, and the enhancement of the *IndirectQuickRun* and *IndirectSampleChanger* algorithms, two new tabs can be added to the Data Reduction interface (i.e. QuickRun and SampleChanger).

10.3 Other prospective developments

Other future developments include an improvement to the Simulations interface to allow the new version of MDANSE output, and also the inclusion of transmission calculations for the VESUVIO instrument from calculated VDOS.

References

[1] Mantid: Manipulation and Analysis Toolkit for Instrument Data, Mantid Project. (2013) Available at: `https://www.mantidproject.org/Main_Page` (Accessed: 20-06-2019).

[2] ISISIndirectEnergyTransfer v1. (2019) Available at: `https://docs.mantidproject.org/nightly/algorithms/ISISIndirectEnergyTransfer-v1.html` (Accessed: 05-08-2019).

[3] IdentifyNoisyDetectors v1. (2019) Available at: `https://docs.mantidproject.org/nightly/algorithms/IdentifyNoisyDetectors-v1.html` (Accessed: 20-05-2019).

[4] Hewer, B. (2018) *Overview of Developments in MANTID relating to Indirect Inelastic Spectroscopy July 2017 - June 2018*, Technical Report RAL-TR-2018-010, STFC.

[5] Kolawa, A. and Huizinga, D. (2007) *Automated Defect Prevention: Best Practices in Software Management*. Wiley-IEEE Computer Society Press. p. 74. ISBN 978-0-470-04212-0

[6] Science and Technology Facilities Council : ISIS Neutron and Muon Source (2019) *Mantid QENS User Manual online*. Available at: `https://www.isis.stfc.ac.uk/Pages/Mantid-QENS-Manual-online.aspx` (Accessed: 15-09-2019).

[7] Mukhopadhyay, S., Hewer, B., Howells, S. and Markvardsen, A. (2019) *A modern approach to QENS data analysis in Mantid*, Physica B: Condensed Matter, 563, pp. 41-49.

[8] IndirectEditFitResult v1. (2019) Available at: `https://docs.mantidproject.org/nightly/algorithms/IndirectReplaceFitResult-v1.html` (Accessed: 20-05-2019).

[9] CopyDataRange v1. (2019) Available at: `https://docs.mantidproject.org/nightly/algorithms/CopyDataRange-v1.html` (Accessed: 20-05-2019).

[10] Hunter, J. D. (2007) *Matplotlib: A 2D graphics environment*, Computing In Science & Engineering, 9(3), pp. 90-95.

A Appendices

A.1 Unit testing example

The following provides a simple example for how unit tests may be implemented for a function written in C++:

```
// Divides number a by number b and returns the result
double divide(double a, double b) {
  if (b != 0.0)
    return a/b;
  throw std::invalid_argument("Argument b cannot be zero.");
}
```

This function might have the following unit tests:

```
// 1. Tests that the correct answer is achieved when b is not zero
void testThatDivideReturnsTheCorrectAnswerWhenBIsNotZero() {

  // Performs the calculation
  double result = divide(4.0, 2.0);

  // Asserts that the result is equal to 2.0
  TS_ASSERT_EQUALS(result, 2.0);
}


// 2. Tests that an error is thrown when b is zero
void testThatDivideThrowsAnErrorWhenBIsZero() {

  // Asserts that a std::invalid_argument error occurs
  TS_ASSERT_THROWS(divide(4.0, 0.0), const std::invalid_argument &);
}
```

A.2 System testing example

The following example outlines the general structure of a typical system test used to test an unspecified MANTID algorithm which takes two workspaces as input before producing an output workspace. For the sake of simplicity, unimportant code has been excluded.

```
// Creates a matrix workspace using the data provided
MatrixWorkspace_sptr createBasicWorkspace(
                     std::vector<double> const &data) {
  ...
}

// Executes the algorithm being tested and returns the result
MatrixWorkspace_const_sptr executeAlgorithm(
                            MatrixWorkspace_sptr workspace1,
                            MatrixWorkspace_sptr workspace2) {
  ...
}

// Manually creates the expected workspace
MatrixWorkspace_const_sptr createExpectedWorkspace() {
  ...
}

// Returns true if the two workspaces are identical
bool compareWorkspaces(MatrixWorkspace_const_sptr workspace1,
                       MatrixWorkspace_const_sptr workspace2) {
  ...
}

// The system test
void testThatAnAlgorithmProducesTheExpectedOutputWorkspace() {
  // Creates two workspaces which contain different data
  auto ws1 = createBasicWorkspace({1.0,2.0,3.0});
  auto ws2 = createBasicWorkspace({4.0,5.0,6.0});

  // Executes the algorithm and returns the resulting workspace
  auto const resultWorkspace = executeAlgorithm(ws1, ws2);
  // Manually constructs the expected result
  auto const expectedWorkspace = createExpectedWorkspace();

  // Asserts that the result and expected workspaces are identical
  TS_ASSERT(compareWorkspaces(resultWorkspace, expectedWorkspace));
}
```