



An overview of the development of Indirect Inelastic Data Reduction and Analysis in MANTID between July 2015-July 2016

E Oram

August 2016

©2016 Science and Technology Facilities Council



This work is licensed under a [Creative Commons Attribution 3.0 Unported License](https://creativecommons.org/licenses/by/3.0/).

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

Neither the Council nor the Laboratory accept any responsibility for loss or damage arising from the use of information contained in any of their reports or in any communication about their tests or investigations.

An Overview of the development of Indirect Inelastic Data Reduction and Analysis in MANTID between July 2015 - July 2016

Elliot Oram

July 11, 2016

Supervisor: Dr. S. Mukhopadhyay

Abstract

The Manipulation and Analysis Toolkit for Instrument Data (MANTID) [1] is an open source cross platform application that provides a framework for supporting computing and visualisation of scientific data, specialising in neutron and muon analysis. MANTID is currently used at multiple neutron facilities worldwide.

This document aims to describe the development carried out between July 2015 to July 2016 across version 3.5, 3.6 and 3.7 of MANTID.

Development during this period was focused on the full integration of support for the VESUVIO instrument at ISIS as well as improving the reliability and robustness of all graphical user interfaces (GUIs) and the supporting data reduction and analysis routines that these GUIs use. In addition, areas for future development are outlined at the end of this document.

Abstract	2
Acknowledgements	5
1 Introduction.....	6
2 Data Reduction.....	7
2.1 ISIS Energy Transfer	7
2.2 ISIS Calibration	8
3 Corrections	10
3.1 Container Subtraction.....	10
4 Data Analysis.....	13
4.1 I(Q, t)	13
4.2 ConvFit.....	15
4.2.1 Quasi Elastic Fit Function	15
4.2.2 Interface parameter table.....	16
4.2.3 Delta Function centre and height	16
4.4 Conversion of Python Scripts.....	17
5 Bayesian Analysis	18
5.1 Quasi.....	19
5.2 FABADA.....	20
6 VESUVIO.....	22
6.1 Integration within MANTID	22
6.1.1 Algorithms.....	22
6.1.1.1 VesuvioPreFit.....	23
6.1.1.2 VesuvioTOFFit.....	24
6.1.1.3 VesuvioCorrections.....	26
6.1.2 Scripts.....	29
6.2 Data loading and VESUVIO operation	31
6.2.1 Periods	31
6.2.2 Differencing modes.....	33
6.2.3 Difference Mode vs. Period validation.....	33
6.2.4 Load Monitors.....	34
6.3 Vesuvio Thickness.....	35
6.4 Notable bug fixes for Vesuvio.....	36
6.4.1 Fitting data in VesuvioCorrections	36
6.4.2 Intensity constraints bug.....	37
7 Simulations.....	38
7.1 Rebinning in Simulated Density of States.....	38
8 Consistency and Robustness.....	39

8.1 Progress tracking	39
8.2 Validation improvements.....	40
9 Documentation.....	41
9.1 Release Notes.....	41
9.2 Documentation improvements	42
10 Testing	44
10.1 Test best practices	44
10.2 Types of tests in MANTID	45
10.3 Indirect specific testing	46
10.3.1 IRIS	46
10.3.1.1 IRIS testing files.....	46
10.3.2 OSIRIS and TOSCA	47
10.3.2.1 OSIRIS testing files	47
10.3.2.2 TOSCA testing files.....	47
10.3.3 VESUVIO.....	48
10.3.3.1 VESUVIO testing files	48
10.3.3.2 VesuvioCorrections system test	48
10.3.3.3 Testing meaningful values	48
10.3.3.4 Tolerance values and justification	50
10.3.4 Simulations.....	51
10.3.4.1 Simulations testing files.....	51
10.4 Modifying Mantid Workflow	52
11 Future Planning	56
11.1 Vesuvio	56
11.1.1 LoadVesuvio	56
11.1.2 Vesuvio Commands.....	57
11.1.3 Documentation and testing	57
11.2 Data Analysis	57
11.3 Bayesian Analysis.....	58
11.4 FABADA.....	58
11.5 Simulations.....	59
11.5.1 aCLIMAX.....	59
Bibliography.....	60

Acknowledgements

I would like to thank Professor Felix Fernandez-Alonso for his continued support throughout the year. I would also like to thank Dr Sanghamitra Mukhopadhyay and Dr Spencer Howells for their guidance and support with learning and implementing scientific technique within the MANTID framework. Finally I would like to thank Dr Matthew Krzystyniak and Dr Giovanni Romanelli for their help in the implementation of, and aid in my understanding of, the VESUVIO data analysis and data reduction in MANTID.

1 Introduction

MANTID is an open source, cross platform, scientific software package offering support for data reduction, analysis and visualisation [1]. MANTID is currently in use at many of the large neutron and muon sources across the world. The main partners of the project are ISIS (Rutherford Appleton Laboratory, UK), SNS (Oak Ridge National Laboratory, Tennessee, USA) and the European Spallation Source (Scandinavia) as well as a newly established team at the Institut Laue-Langevin (Grenoble, France). The majority of the development team is also distributed across the main partners however the MANTID project has also had notable contributions from the Paul Scherrer Institut (Villigen, Switzerland) and ANSTO (Sydney, Australia) have also contributed to the project.

MANTID offers both user interface driven and python based scripting solutions for the manipulation of data. MANTID GUIs originally stemmed from the QtiPlot project [2] and have been further extended using the Qt library. MANTID uses the concept of workspaces (a matrix of x, y and error data) for handling data that is extracted from raw data files. Algorithms, which can be thought of as isolated routines or functions, perform operations on the data. These algorithms are written in either C++ or Python with the bulk of the MANTID framework written in C++.

The indirect geometry section of MANTID is largely created with the original MODES [3] application as a template as well as the IRIS Data Analysis package [4]. VESUVIO functionality has been modelled on the VESUVIO analysis programs originally hosted on the VMS cluster.

This document aims to describe the current state of Indirect geometry support within MANTID as well and the changes over the course of Release 3.5 to 3.7 that have been made.

Much has been done to support the VESUVIO instrument at ISIS in terms of data reduction and analysis. The vast majority of the Vesuvio workflow has been fully incorporated into MANTID which includes up to date and fully automated tests for all VESUVIO algorithms and scripts (see section 6 for details).

The general consensus from feedback given to the MANTID team at the Scientific Steering Committee, held at the Oak Ridge National laboratory January 20th – 21st 2016 [5], was to ensure that a focus was given to robustness and stability of the software package over the addition of new features. As such, much of the focus with release 3.6 and 3.7 was put on improving the stability of the software through validation, testing and consistency.

This document also includes a brief overview of further development task that can be carried out in future releases. This is not designed to be a task list but could act as guidance for future development.

2 Data Reduction

In order to analyse data from an experiment, the data collected must be converted from the raw time of flight data to energy transfer. This can be done by using the parameters of the instrument. After the reduction of energy transfer data, it can then be converted to the instrument independent function which describes the proportion of incident neutrons scattered with a given momentum and energy transfer known as $S(Q, \omega)$. The full procedure for reducing data in this way is detailed in Ref [6].

MANTID already has facilities in place to perform data reduction which are outline in more detail in Ref [7]. The follow describes the additional changes made to the data reduction interfaces within MANTID (Interfaces \Rightarrow Indirect \Rightarrow Data Reduction).

2.1 ISIS Energy Transfer

In order to improve consistency throughout the indirect section of MANTID a new naming convention has been implemented that ensures that during the data reduction stage, all workspaces are appropriately named. The new naming convention is:

<Full instrument name><run number>_<analyser><reflection>_<red/res/sqw/iqt>

Every character in the new naming convention is lower case. An example of this could be the IRIS run 26176 which is reduced using the graphite analysers and 002 reflection. This would be given the name:

iris26176_graphite002_red

At the end of the workspace names, the suffix red, res, sqw or iqt is used to represent the data in different formats. red is the reduced raw file, res is a resolution file, sqw being $S(Q, \omega)$ and iqt being $I(Q, t)$. These suffixes are used to help restrict where this data can be used within the interfaces which acts as an additional form of validation as well as helping to guide users with file input. Other suffixes are also used in indirect but those mentioned above are the most common.

2.2 ISIS Calibration

Many of the interfaces in the indirect geometry section of MANTID have preview plots that display the input and, after a successful run of the algorithm, output data. In order to aid users to visualise the input parameters for algorithms relative to the data, many preview plots also have range bars that define the FWHM, background contribution or fitting range for the data. These range bars can be changed by clicking and dragging them to a new location or updating the corresponding value in the property table. **Figure 1** shows an example of property table and corresponding range bars.

In an effort to make MANTID more user friendly, range bars were changed to be automated in many cases where a good guess can be made. In the case of the ISIS Calibration Interface, the peak range and background can be approximated for each instrument. These approximations are based on the resolution of the instrument and correlate to the background and peak range values as shown in **Table 1**.

Table 1 is true for experimental data, with units of energy, as the elastic peak is at 0meV. In order to translate this data to be correct for time of flight, these values are added to a workspace with instrument data also being loaded into that workspace. The instrument data includes replacing the spectrum number of the workspace to match the first non-monitor spectrum for spectroscopy data (For the IRIS instrument, this is spectrum number 3). After this data is loaded, the data is then converted to time of flight using the *ConvertUnits* algorithm.

During the development period leading up to release 3.7, a bug was discovered that was causing these values to be incorrect for OSIRIS data. This was tracked down to the spectrum number being incorrectly set for OSIRIS data. Where, as described above, the first IRIS spectrum number of interest is 3, for OSIRIS this is 963. This led to the spectrum number being for a diffraction detector on OSIRIS causing an offset in the approximation for the range bar locations for OSIRIS data. **Figure 1** shows the correct range bars in the interface.

Approximate parameter for spectra	Peak minimum boundary	Peak centre position	Peak maximum boundary	Background start boundary	Background end boundary
Resolution scale factor	-2	0	2	5	6

Table 1: Shows the scale factor used in product with the instrument resolution to obtain approximate parameters for a single spectrum in the ISIS Calibration interface

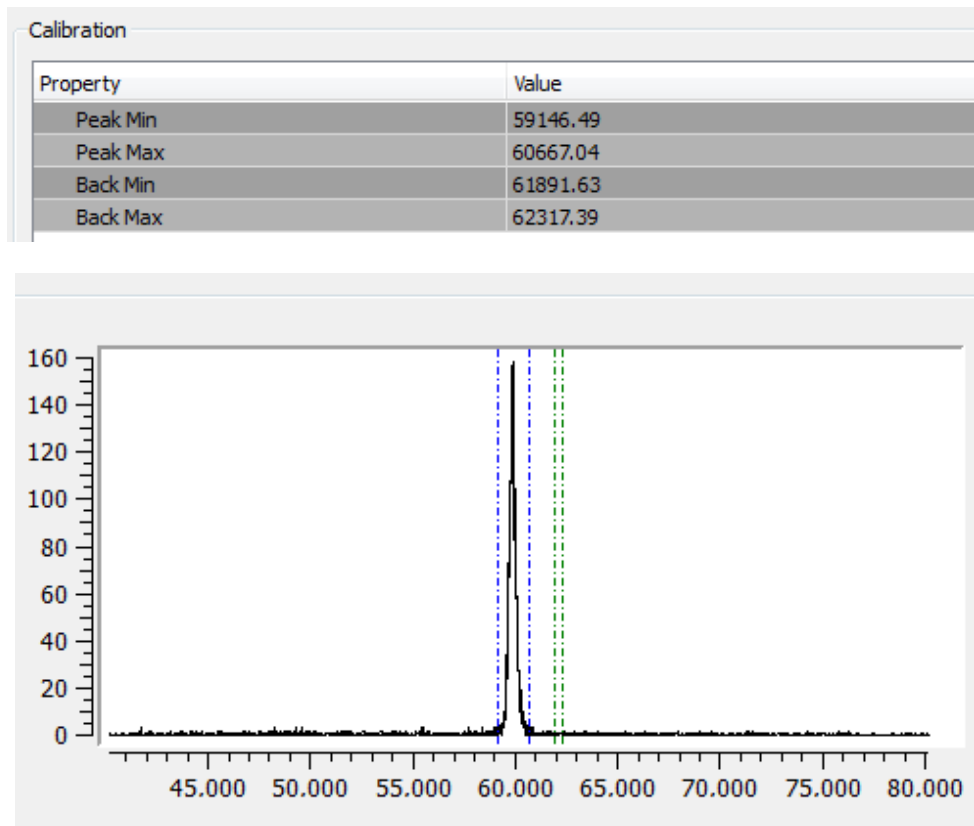


Figure 1: ISIS Calibration interface showing the preview plot range bars, and corresponding values, for OSIRIS data approximated by the process described in section 2.2.

3 Corrections

Before data is analysed, corrections should be applied to better model the data and to account for attributes such as sample shape and the contribution to the spectra from the container. Currently MANTID supports absorption correction using the method formalised by Paalman and Pings to describe attenuation factors of a sample and container [8]. Additionally, a basic method for Container Subtraction is supported in the interface, see section 3.1.

The changes described below have been made to the corrections interface (Interfaces ⇒ Indirect ⇒ Corrections).

3.1 Container Subtraction

The Container Subtraction interface enables the user to mathematically subtract the contribution of a container from an experimental run of the sample. This will ensure that further operations and visualisation of the data will be performed on the observations of the sample alone.

This interface uses the *ApplyPaalmanPingsCorrection* algorithm which, in this case, is only used to perform the container scale and subtraction. In this simple mode of operation the algorithm takes sample workspace, container (can) workspace and a Container Scale Factor as inputs. The *ApplyPaalmanPingsCorrection* algorithm then applies the Scale Factor to the container and then uses the *Minus* algorithm to remove the container contributions from the sample workspace, see **Figure 2**.

Further to the scaling of the container, the ability to shift the container in the X axis was also added to account for a potential shift in peak centre. This uses the *ScaleX* algorithm and is currently handled by the interface code as opposed to the *ApplyPaalmanPingsCorrection* algorithm which is a planned development for the future. In addition to enabling a shift in the X axis in the Container Subtraction interface, this functionality has also been implemented in the Apply Paalman Pings interface.

The GUI allows the visualisation of the sample relative to the container as well as the magnitude of the shift and scale. This is with an aim to aid in the selection of inputs for the algorithm. The interface provides a preview plot which shows curves that represent the sample (black) and container (red). The subtracted (green) is displayed after the algorithm has completed, see **Figure 3**.

In order to alter the shift or scale of the container, the options (“Scale Container by factor” and Shift x-values of container by adding”, see **Figure 4**) must be checked in the interface and then the spinners allow the user to change the value by typing or clicking the up and down arrows. After loading a container workspace into the interface, changing values for shift and scale will automatically update the curve in the preview plot. Currently the maximum values for container shift are bound by the x range of the sample. Additional validation that has been added to interface to ensure the input workspaces should be end in ‘_red.nxs’ or ‘_sqw.nxs’.

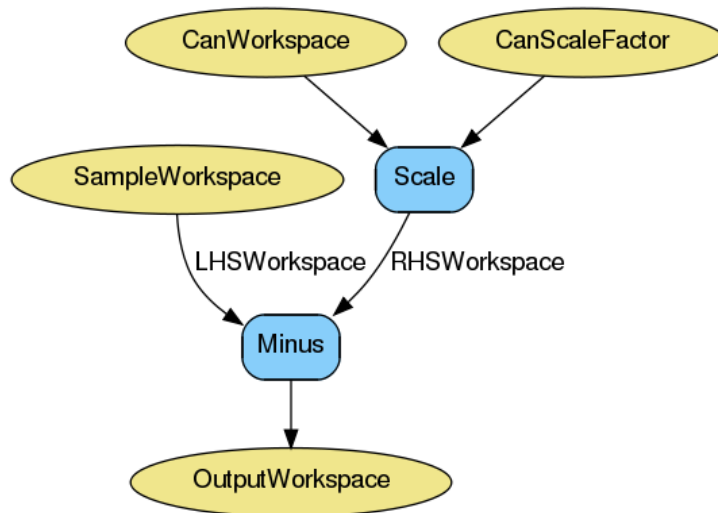


Figure 2: ApplyPaalmanPingsCorrection Container Scale and Subtraction Workflow
 A full size version of this diagram and additional information regarding the algorithm can be found at <http://docs.mantidproject.org/nightly/algorithms/ApplyPaalmanPingsCorrection-v1.html>

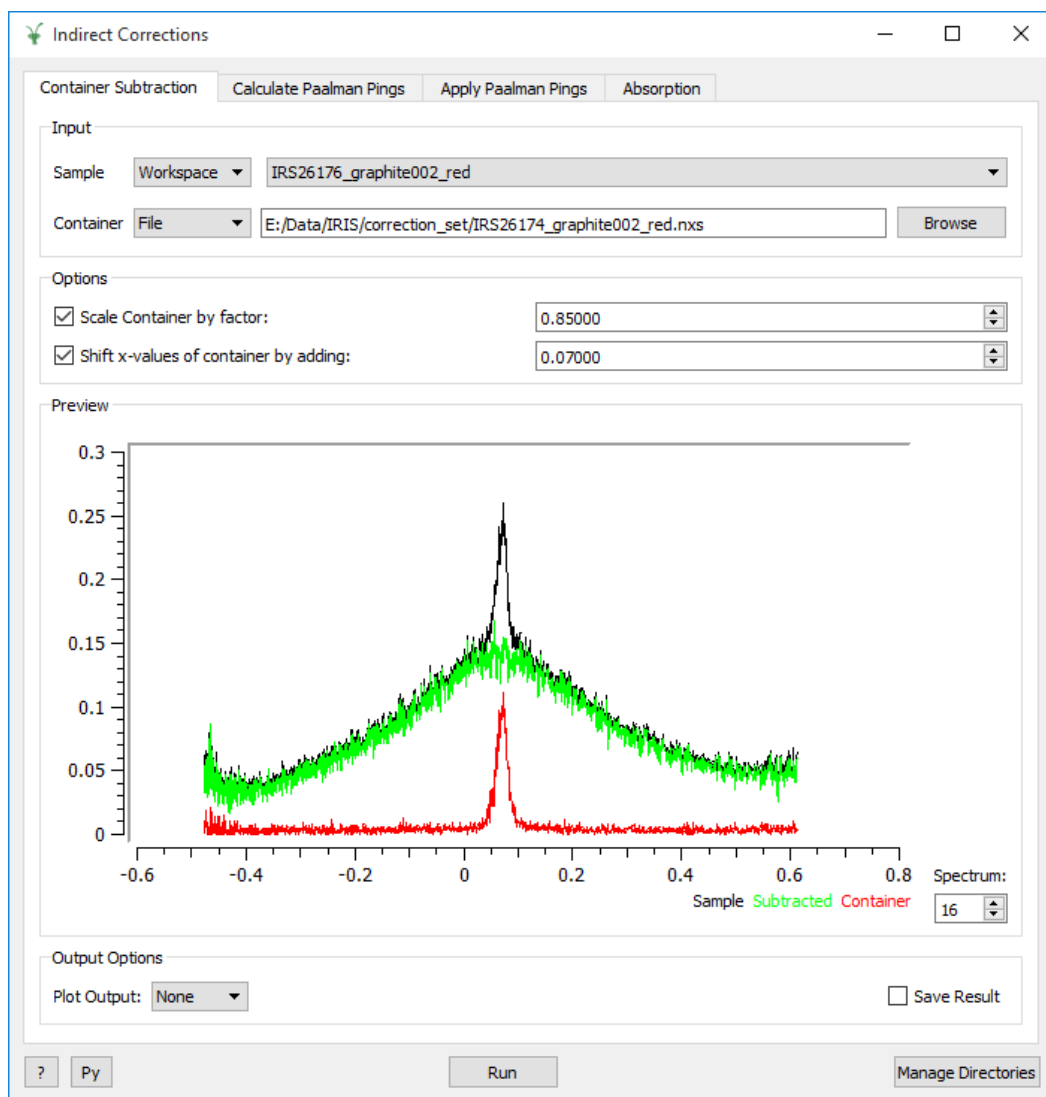


Figure 3: Container Subtraction interface post run with shift and scale for IRIS data

Options

<input checked="" type="checkbox"/> Scale Container by factor:	1.00000
<input checked="" type="checkbox"/> Shift x-values of container by adding:	0.00000

Figure 4: Scale Container and Shift x-value options in the ContainerSubtraction interface

4 Data Analysis

The Indirect Data Analysis interface (Interfaces > Indirect > Data Analysis) in MANTID allows users to easily perform common processes and fitting to reduced data. The Elwin and I(Q, t) interfaces perform transformations of the data whilst the MSD fit, I(Q, t) Fit, ConvFit and JumpFit tabs act as a wrapper to the *Fit* algorithm in MANTID. The interfaces designed for fitting, offer the users many commonly used fit functions in an easy to manipulate format. More detailed descriptions of these can be found in Ref [7].

The indirect data analysis interfaces allows processing of both energy transfer reduction workspaces (suffixed with `_red`) as well as $S(Q, \omega)$ workspaces (suffixed with `_sqw`).

4.1 I(Q, t)

The I(Q, t) interface uses the *TransformTolqt* workflow algorithm which, in brief, by means of a Fourier transform, converts reduced spectrum data into the intermediate scattering function I(Q, t). A more detailed description of the algorithm can be found in the MANTID documentation online at <http://docs.mantidproject.org/nightly/algorithms/TransformTolqt-v1.html>

The interface is designed to take input of a sample and resolution workspace as well as an energy range and binning preference for the data in order to create a `_iqt` output workspace. Each spectra in the sample workspace is treated independently and therefore the I(Q, t) workspace will contain the same number of spectra as the sample.

To obtain a quick overview of the data to ensure it is of good quality, a tiled plot has been introduced which enables a user to see all the spectra in the output workspace in a single window. This option exists as a checkbox in the I(Q, t) interface, see **Figure 5**, and should be checked before running the algorithm. Once the algorithm concludes, a plotting window similar to **Figure 6** will be displayed showing the plots for all spectra in the workspace. A scroll bar on the right hand side of the window gives access to the remaining plots not shown in **Figure 6**. The legend details the spectrum number using the notation `iris26176_graphite002_iqt_tiled-sp-1` where “sp-1” denotes spectrum number 1.



Figure 5: I(Q, t) interface Output section Tiled Plot option

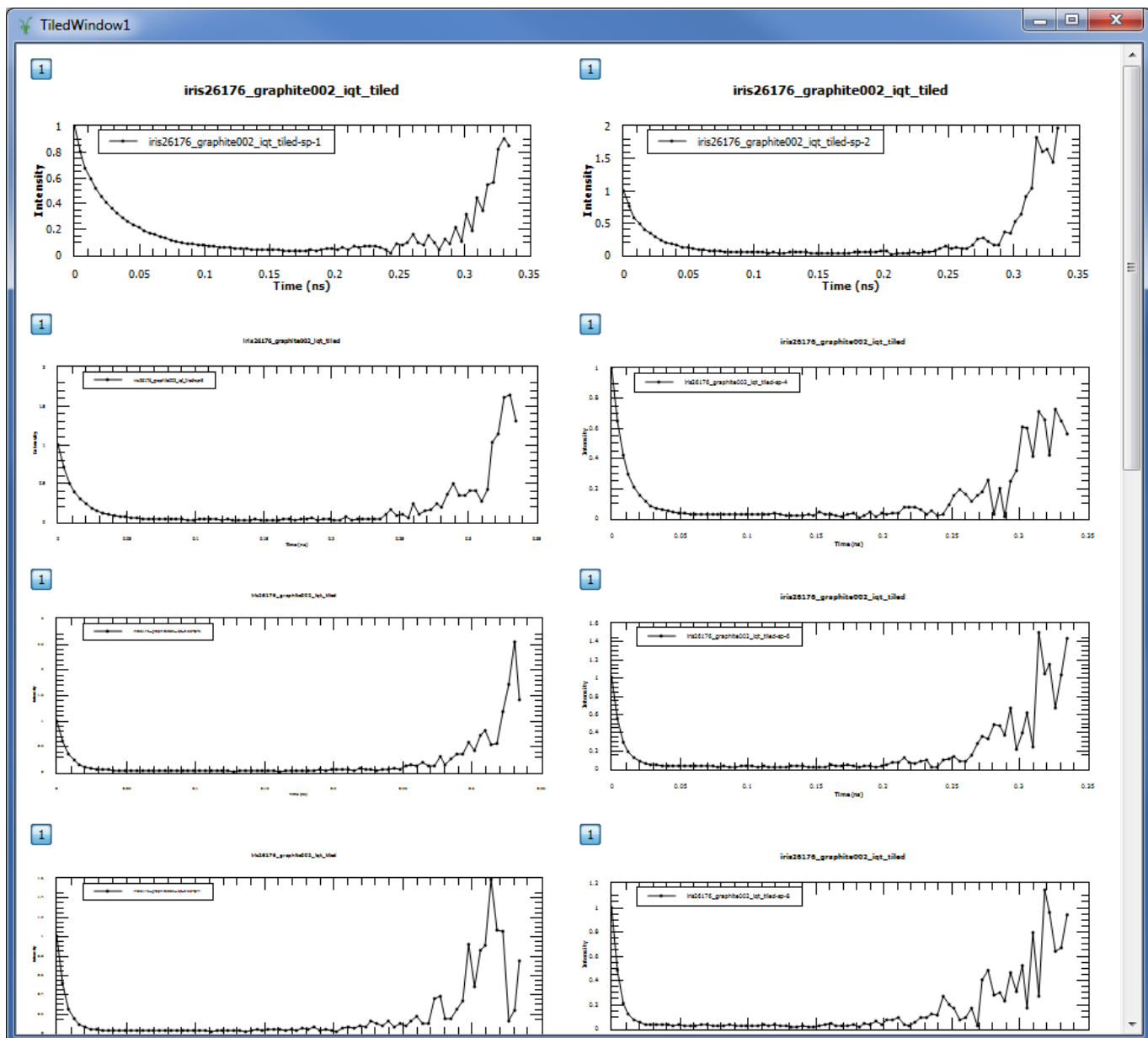


Figure 6: I(Q, t) interface Tiled Plot produced at the end of run in the I(Q, t) interface if the option is checked (see **Figure 5**). The data above is iris26176_graphite002_red.

4.2 ConvFit

4.2.1 Quasi Elastic Fit Function

The ConvFit interface provides a simple way to fit a convolution of the resolution and a model to reduced data. This action can be mirrored in the fit wizard, but the interface offers an easier way to fit commonly used models as well as a define ties, constraints and temperature correction.

Four QuasiElastic models have been added to both the underlying algorithm and the interface:

- Inelastic Diffusion Sphere (InelasticDiffSphere)
- Inelastic Diffusion Rotation Discrete Circle (InelasticDiffRotCircle)
- Elastic Diffusion Sphere (ElasticDiffSphere)
- Elastic Diffusion Rotation Discrete Circle (ElasticDiffRotCircle)

These fit functions are available from the Fit Type drop down menu in the interface, see **Figure 7**.

Further information on all fit function available in the ConvFit interface can be found online at http://docs.mantidproject.org/nightly/interfaces/Indirect_DataAnalysis.html#fitting-model

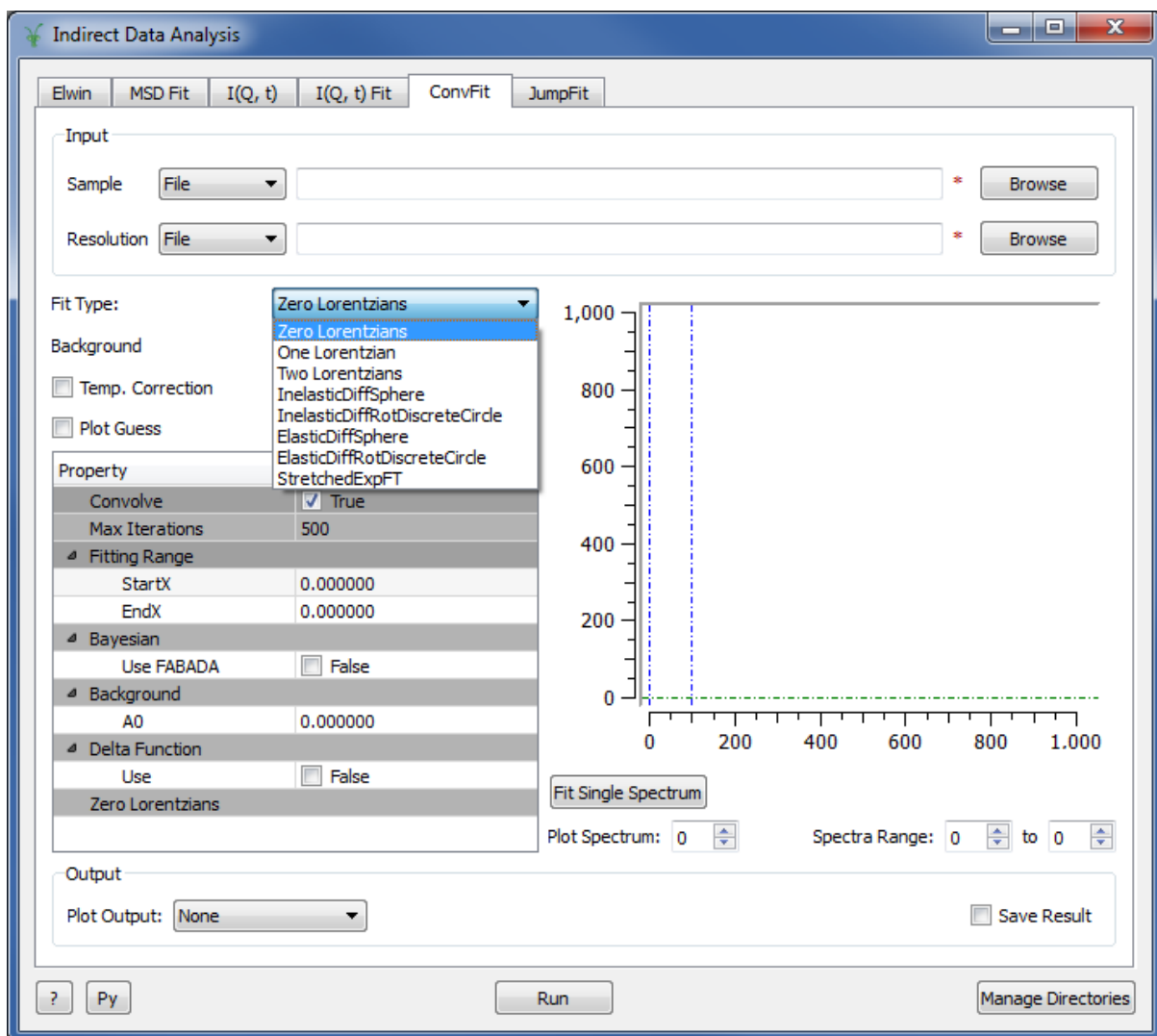


Figure 7: ConvFit interface Fit Type drop down menu displaying all currently implemented models

4.2.2 Interface parameter table

Each model has its own initial starting parameters that should be set in order to obtain a better fit for data. The parameter table on the left of the interface, see **Figure 8**, allows for entry of these values and these are then parsed into a function string which is given to the *Fit* algorithm along with the data.

The parameter table automatically updates to display the relevant parameters for the desired model when it is changed in the drop down menu. Furthermore, when the interface is run, the parameters values in the table are updated with the output parameters calculated by the *Fit* algorithm.

In the more specific case, after a single Lorentzian fit is completed, which will cause the values in the parameter table to update automatically, if the user then selects the Two Lorentzian model in the interface, the parameters from the one Lorentzian fit are passed as the input for the Two Lorentzian fit.

InelasticDiffSphere	
Intensity	1.000000
Radius	1.000000
Diffusion	1.000000
Shift	0.000000

Figure 8: ConvFit interface parameter table showing required parameters for Inelastic Diffusion Sphere (InelasticDiffSphere)

4.2.3 Delta Function centre and height

Some data sets used in the ConvFit interface have an additional contribution from the container that should be represented in model. This contribution can be modelled as a single delta function in the elastic window.

The functionality to allow the delta function centre and height to be adjusted has been added. This enables a better guess for the peak height to improve the output from the fit. Furthermore, the peak centre of the delta function can now be shifted along the x axis which is required if the peak from the container contribution is off centre (possibly due to experimental inaccuracies). This option is shown in **Figure 9**.

Delta Function	
Use	<input checked="" type="checkbox"/> True
Height	1.079261
Centre	-0.001254

Figure 9: Delta Function centre and height options in ConvFit interface

4.4 Conversion of Python Scripts

The origin of many of the data analysis programs was the MODES package [3] the IRIS data analysis package [4], therefore, some of the functionality still remains in the form of python scripts. In order to improve maintainability and consistency throughout MANTID many of these scripts have been converted to MANTID algorithms.

Conversion of these scripts gives the advantage of progress tracking (see section 7.1), automatically generated dialogue box allowing the algorithm to be run separately from the interface more easily. In addition, logging, workspace handling and being consistent with MANTID architecture are also benefits of this change.

The largest script that has been converted in this periods is the `IndirectDataAnalysis.py` script which contained the functionality for the ConvFit and I(Q, t) Fit interfaces.

The core algorithm that has been created from this change is the *ProcessIndirectFitParameters* algorithm. This is used by all of the algorithms created from the script (See below for description of other algorithms). This algorithm is designed to turn the output of the *PlotPeakByLogValue* algorithm to a MatrixWorkspace in the form expected by the post processing stages of the parent algorithm. The algorithm allows for the definition of the column containing the x axis values, names of the parameters that should be put into the output workspace and the target unit for the workspace. The full documentation including a usage example for this algorithm can be found online at <http://docs.mantidproject.org/v3.7.1/algorithms/ProcessIndirectFitParameters-v1.html>

ConvolutionFitSequential is a C++ algorithm that controls the fitting performed in the ConvFit interface. The algorithm performs a sequential fit, given a model, for a convolution workspace. This algorithm also offers the ability to change the fitting range, spectra to be fitted, the minimizer and max iterations of the underlying call to *PlotPeakByLogValue*. The full documentation including a usage example for this algorithm can be found online at <http://docs.mantidproject.org/v3.7.1/algorithms/ConvolutionFitSequential-v1.html>

lqtFitSequential and *lqtFitMultiple* are python algorithms that perform the data analysis in the I(Q, t) Fit interface. These algorithms, originally part of the MODES package [3], are designed to take the *_lqt workspace produced from the I(Q, t) interface and perform a fit using a model of either one or two exponentials, a stretched exponential or a combination of an exponential and a stretched exponential. The full documentation including a usage example for this algorithm can be found online at <http://docs.mantidproject.org/v3.7.1/algorithms/lqtFitSequential-v1.html> and <http://docs.mantidproject.org/v3.7.1/algorithms/lqtFitMultiple-v1.html>

All the algorithms listed above are accompanied by automated unit tests to ensure robustness.

The implementation of these algorithms has allowed for the removal of the `IndirectDataAnalysis.py` script from the MANTID package and this will no longer be shipped with MANTID as of release 3.8.

5 Bayesian Analysis

The Bayesian analysis tools have proven to be popular within MANTID. Currently housed in the Bayes interface (Interfaces ⇒ Indirect ⇒ Bayes), the analysis tools exist in three tabs: ResNorm, Quasi and Stretch. All of these tools, via use of probability theory, aid in the assessment of the likelihood of 1-3 quasi elastic components in the data. Bayesian analysis also gives the advantage of providing an optimal estimate for the parameters of these components [9].

ResNorm provides a facility to normalise the input data by fitting the resolution. This fitted data from the ResNorm algorithm, namely the ResNorm group, see **Figure 10**, can then be used as input for the BayesQuasi algorithm (see section 5.1)

BayesQuasi, takes a sample, resolution and optional ResNorm file, see **Figure 10**, and through Bayesian analysis assesses the likelihood of 1-3 peaks being present in the data [9]. This algorithm does not require input parameters as these are instead optimised by the algorithm itself. After the algorithm the _Result workspace displays the parameters for 1-3 quasi elastic peaks, see **Figure 11**. BayesQuasi was originally called Q Lines in the MODES package [3].

Stretch, originally Quest in the MODES package [3], operates in an almost identical manner to that of *BayesQuasi* with the exception of using Stretched Exponentials rather than lorentzians. This algorithm, like BayesQuasi, is also a wrapper for the underlying FORTRAN code (see section 5.1 for more detail).

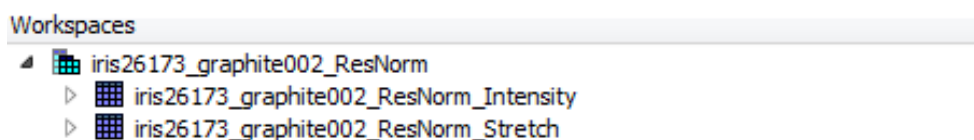


Figure 10: ResNorm group for the iris26173_graphite002 reduction and resolution files

	0 0.483618Å ⁻¹	1 0.60787Å ⁻¹	2 0.729145Å ⁻¹	3 0.847025Å ⁻¹	0.000000Å ⁻¹
f1.f0.Height	7.14508	4.20554	2.94479	2.00381	1.46000
f1.f1.Amplitude	73.9856	74.4935	71.983	63.0661	55.5000
f1.f1.FWHM	0.0523066	0.0758276	0.104449	0.135821	0.160000
f1.f1.EISF	0.0880688	0.0534383	0.0393017	0.0307948	0.020000
f2.f0.Height	2.74177	2.02035	1.51968	1.462	1.11000
f2.f1.Amplitude	13.7681	16.3149	25.0279	20.1999	26.8000
f2.f1.FWHM	0.259771	0.260927	0.221157	0.262463	0.250000
f2.f1.EISF	0.166069	0.110189	0.0572434	0.0674916	0.040000
f2.f2.Amplitude	69.2764	64.767	51.8529	46.1986	31.2000
f2.f2.FWHM	0.0397088	0.060001	0.0773818	0.109798	0.120000
f2.f2.EISF	0.0380705	0.0302504	0.028473	0.0306751	0.030000
f3.f0.Height	1.64781	1.87833	0.09676	1.48954	0.78000
f3.f1.Amplitude	12.8392	14.6921	17.8255	17.9603	23.5000

Figure 11: Shows the optimal parameters for the iris26176_graphite002 data set with the resolution iris26173_graphite002. The initial *fx* number in the parameter name refers to the peak the parameter references. Height is highlighted here for peaks 1 to 3.

5.1 Quasi

The algorithm *BayesQuasi* has been created from the QLines script (which has since been removed). For information on why this script has been changed, see the advantages listed in section 4.4.

This particular script is slightly more complex than those mentioned in section 4.4 as it contains code originally written in the FORTRAN coding language. To allow this to run in MANTID, a pre-compiled (.pyc) python module is built and shipped with MANTID that performs the same operations as the FORTRAN routines but as a python implementation. This is a temporary solution and not ideal as the implementation of the FORTRAN routine (and its compiled python counterpart) is relatively unknown. However, there are replacements for this that can be introduced in future releases (see section 11.3).

In addition to the algorithm implementation, there have also been updates to the validation to ensure the FORTRAN routine is given data in an acceptable format.

A problem was observed when performing Bayesian analysis (using the Bayes interface) with a data set produced from the instrument IRIS. The dataset in question contained a single peak at 0 energy, as expected from IRIS, but also had a large number of 0 count values at the extremities of the range. Using this data over the full fitting range produced a crash in MANTID. This crash was tracked down to the underlying FORTRAN code. Due to the unsustainability of the code (see above), additional validation was added to ensure data does not contain zeros. This validation does not prevent the algorithm from running or producing a result but instead will update the fitting range of the data if it contains zeros in the fitting range specified by the user. If this value is updated, the change will be displayed in the results log while the algorithm is running, see **Figure 12**.

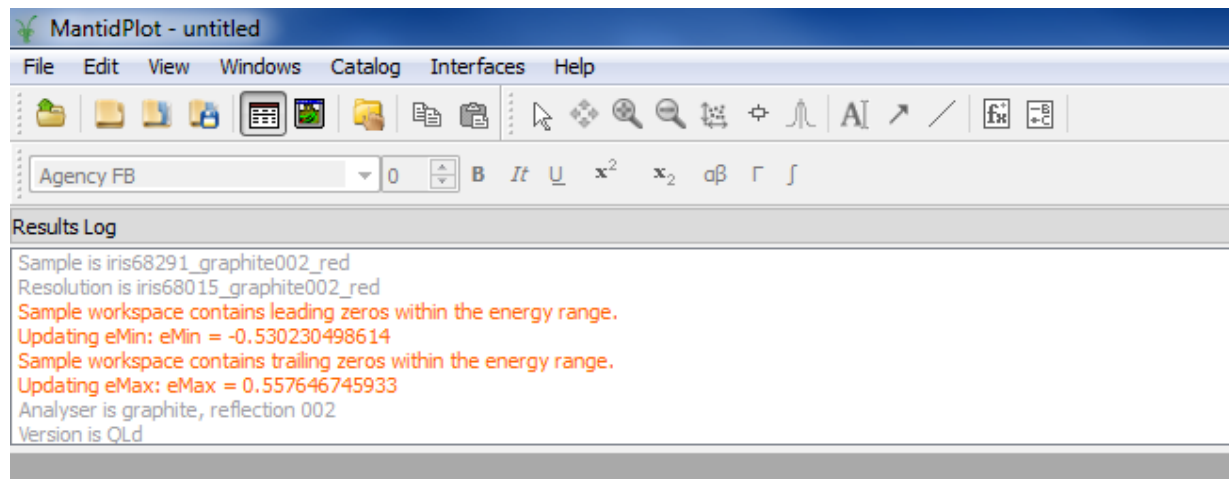


Figure 12: Updated fitting range in results log of Bayes > Quasi for data containing zeros

5.2 FABADA

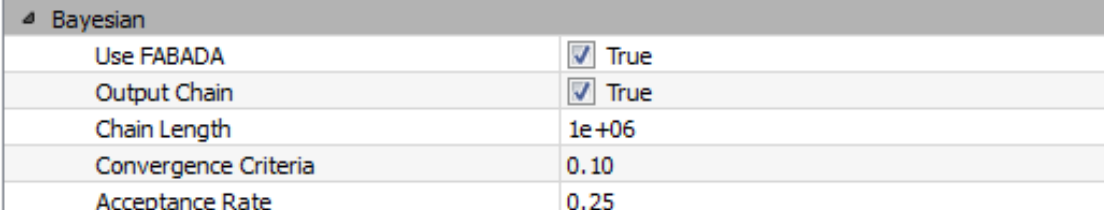
The FABADA minimiser (Fitting Algorithm for Bayesian Analysis of Data) is a powerful tool adapted from the Markov chain Monte Carlo method to perform fitting of experimental data [10]. It is currently implemented in MANTID and is being propagated through the interfaces to allow FABADA minimiser use in GUIs.

Currently in the indirect geometry interfaces, FABADA has been implemented within ConvFit and I(Q, t) Fit with plans to implement it in the ResNorm interface as well (see section 11.4). **Figure 13** shows the additional section added to these interfaces in order to better control FABADA, these options are described in more detail online at <http://docs.mantidproject.org/v3.7.1/concepts/FABADA.html>

When complete, FABADA will output the Probability Density Function (PDF) for each fitted parameter and the cost function as well as an optional plot for the chain showing parameter values for each iteration of the chain.

The usage example described in <http://docs.mantidproject.org/v3.7.1/concepts/FABADA.html>, uses the irs26176_graphite002 data set. After execution, both the PDF and chain are output as Matrix Workspaces. The chain for the amplitude parameter, **Figure 14**, shows the initial sharp increase while exploring the parameter space until it converges to a final value of ~ 2.45 . The PDF agrees in **Figure 15**, showing the probability of the amplitude being 2.45 being the most likely.

Whilst the example in the documentation runs well, due to the complexity of the FABADA minimiser the execution time when used in ConvFit, which performs sequential fitting over many spectra, the run time is significantly longer. This has been reported to occasionally cause crashes and is believed to be related to the converge criteria. This is being addressed (see section 11.4 for more details).



Bayesian	
Use FABADA	<input checked="" type="checkbox"/> True
Output Chain	<input checked="" type="checkbox"/> True
Chain Length	1e+06
Convergence Criteria	0.10
Acceptance Rate	0.25

Figure 13: FABADA minimiser options in the ConvFit and I(Q, t) Fit interface

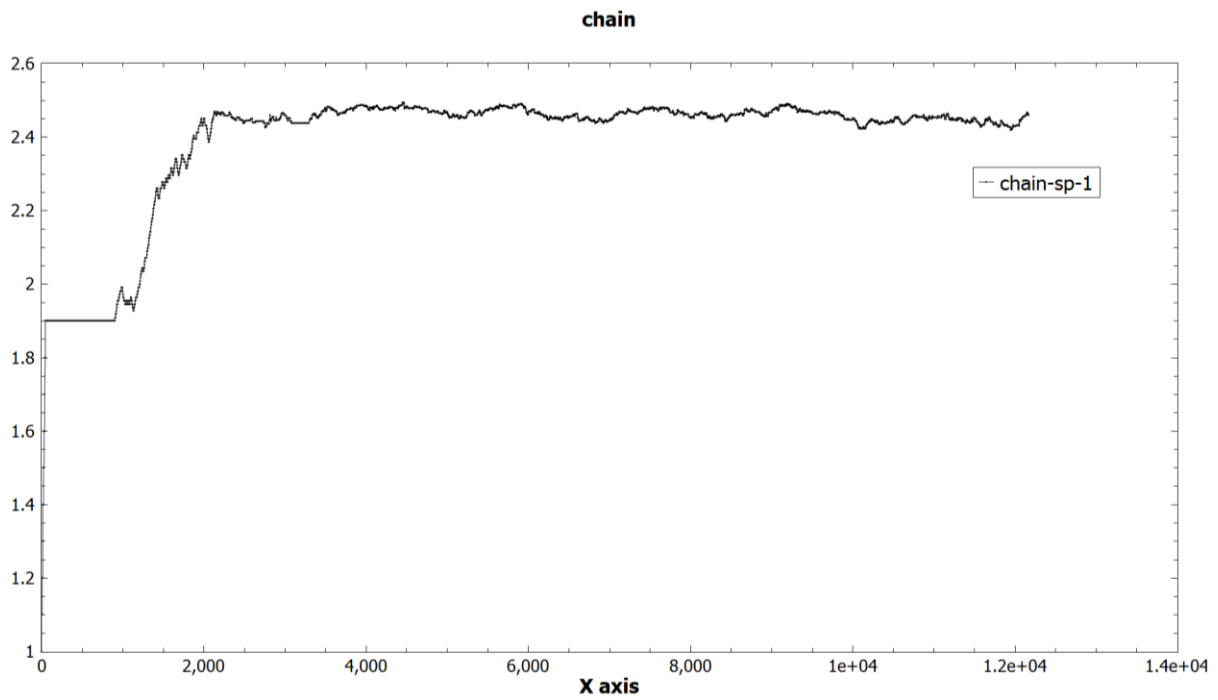


Figure 14: The FABADA chain for the fitting parameter amplitude for the irs26176_graphite002 data set.

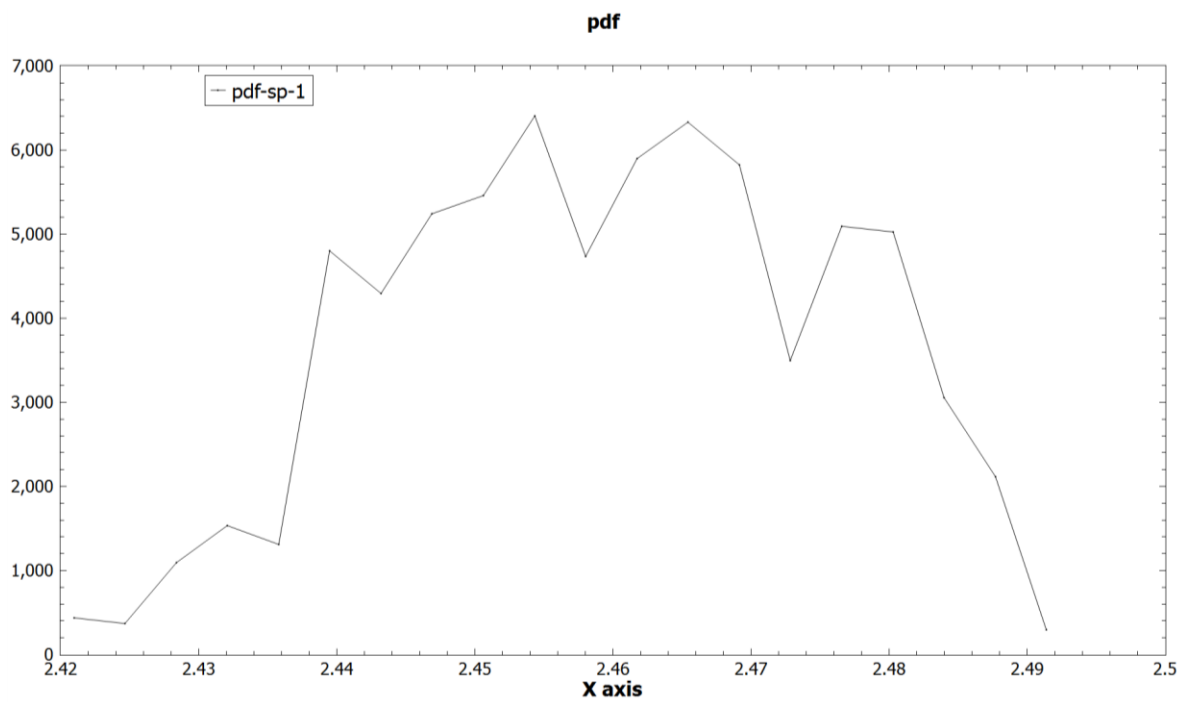


Figure 15: the FABADA PDF for the fitting parameter amplitude for the irs26176_graphite002 data set.

6 VESUVIO

VESUVIO, housed at the ISIS pulsed neutron and muon source, is an electron volt neutron spectrometer and is primarily used for the measurement of proton distributions in condensed matter system. Additionally, VESUVIO can be used to measure kinetic energies of heavier masses and sample compositions [11].

6.1 Integration within MANTID

Previous versions of MANTID had basic support for the workflow of VESUVIO. Much of this was in a separate repository and not automatically tested. The VESUVIO workflow consisted of multiple scripts and algorithms which were all required to be moved into the MANTID framework in order to allow Reduction and Analysis to take place for VESUVIO data. Alongside this, tests were also required to ensure that the reduction and analysis workflow was working as expected but also to ensure that further development did not have unexpected effects on the output.

6.1.1 Algorithms

VESUVIO algorithms are housed in the Framework > PythonInterface > Plugins > Algorithms section of the MANTID code base. These already used the MANTID style syntax for algorithm definitions.

Figure 16 shows the example data set of run numbers 15039-15045. Initially the data is loaded using the *LoadVesuvio* algorithm. In this example a single spectrum number, 135, is loaded (the first of the forward scattering detectors) using the single difference mode with the instrument parameter file 'Vesuvio_IP_file_test.par'. Further information on the operation of and improvements to *LoadVesuvio* can be found in section 6.2.

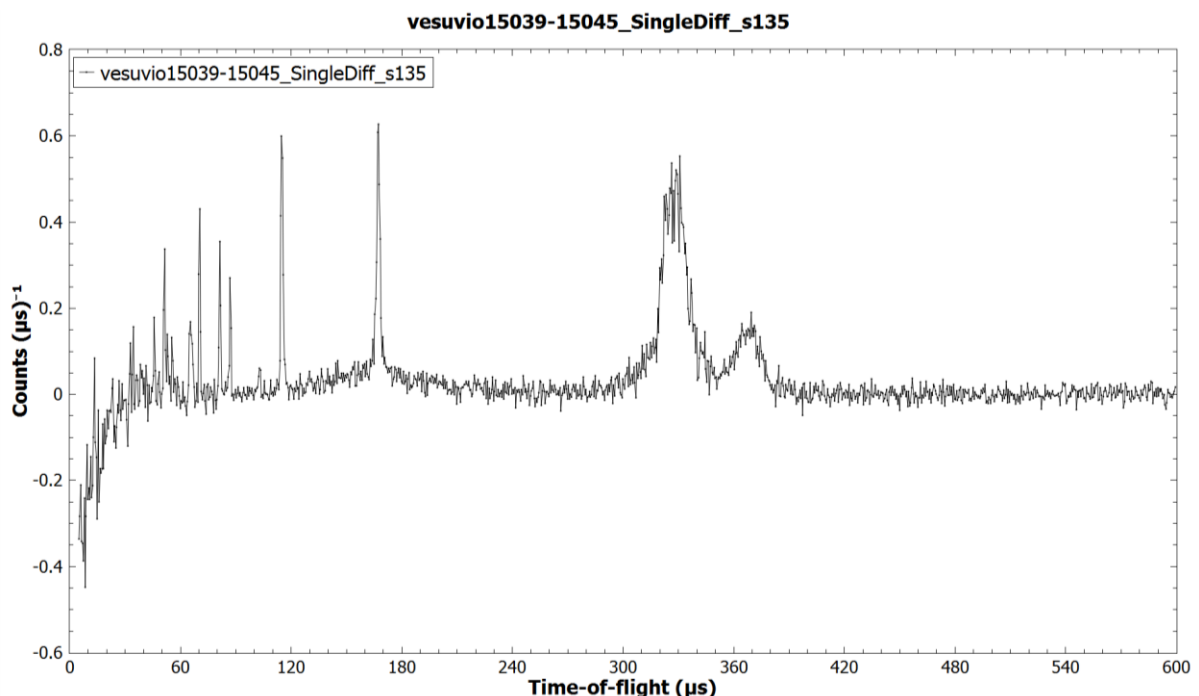


Figure 16: Spectrum 135 of loaded VESUVIO data for run numbers 15039-15045 using the single difference mode

6.1.1.1 VesuvioPreFit

VesuvioPreFit performs initial pre-processing steps on reduced VESUVIO data obtained from *LoadVesuvio*, see **Figure 16**. First the data is smoothed using the *SmoothData* algorithm in MANTID, smoothing by a number of points defined by the user. Following this, any bad data, which is classified as data that has a higher error than the user defined threshold, is masked. The purpose of this algorithm is to make the data easier to fit and less prone to reaching incorrect local minima as well as removing any obvious outliers in the data.

Figure 17 shows the difference between the loaded data in **Figure 16** and the outcome of running that data through the *VesuvioPreFit* algorithm. The red curve (output of *VesuvioPreFit*) shows smoother data with fewer sharp peaks. In the workflow, this algorithm would then pass the output to the *VesuvioTOFFit* algorithm (see section 6.1.1.2).

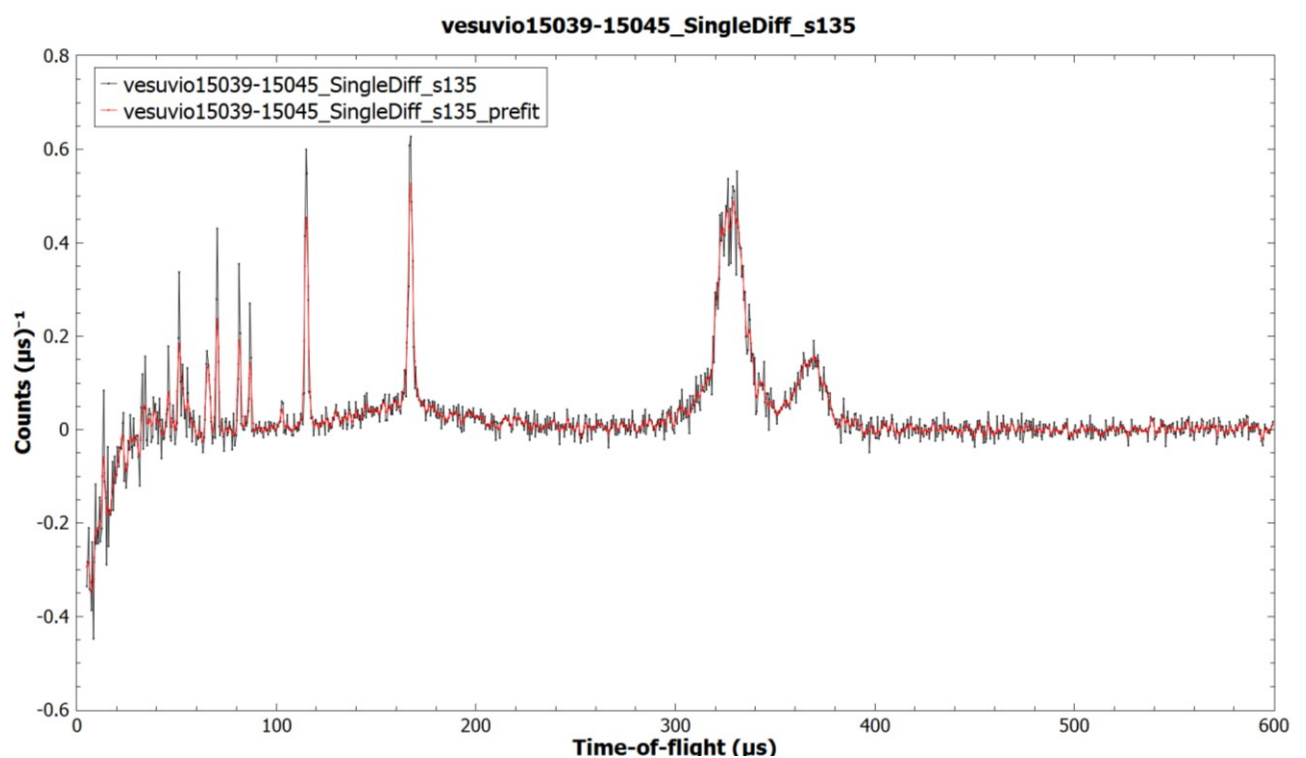


Figure 17: Shows the difference between the Loaded Vesuvio data from **Figure 15** (black) and the *VesuvioPreFit* data (red)

6.1.1.2 *VesuvioTOFFit*

VesuvioTOFFit performs the main fitting of the expected model to the experimental data. The algorithm takes the Input workspace in time of flight, the individual masses to be fitted (as a list) and a function for each mass that is used to approximate the profile of the mass (as a list). The output of the algorithm is a workspace containing the contribution of each mass in the sample (as determined by the *Fit* algorithm), the fit parameters and the chi squared value for the fit. Additionally, *VesuvioTOFFit* allows for the optional declaration of function for the background, additional intensity constraints, an option for fitting by bank or spectra as well as options to set the maximum iterations for the fit and the minimizer to use when fitting.

Additional information about the parameters can be found at

<http://docs.mantidproject.org/v3.7.1/algorithms/VesuvioTOFFit-v1.html>

During execution, the masses, mass profiles, background function and intensity constraints are parsed into a single fit string. Two fits are then performed on the data. The first uses the number of iterations defined by the user in the input parameters and produces fit parameters from this. These parameters are then used in a second fit to calculate the chi squared value for the fit function (how well the function fits the data). This secondary fit only has a single iteration.

Figure 18 shows the output of the *VesuvioTOFFit* algorithm. The algorithm will output the contribution from each mass that was defined in the mass profiles. In this case there are four: the first a GramCharlierComptonProfile (shown in green in **Figure 18**), and three GaussianComptonProfiles that all contribute to the final peak in the sample.

Figure 19 shows the fitting parameters per mass that are also an output from the *VesuvioTOFFit* algorithm. The f_x values in the parameter names denote which mass the following parameter references, where x is (in this case) value between 0 and 3 to represent the 4 different masses.

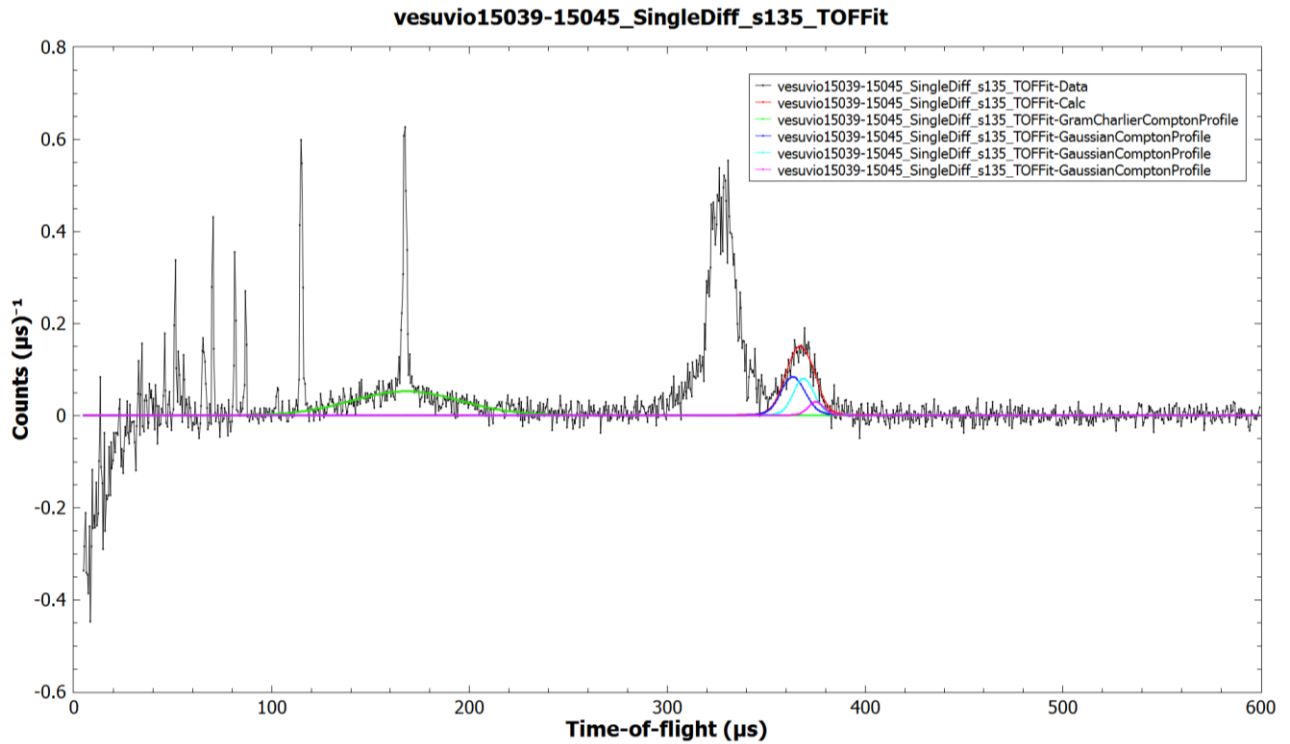


Figure 18: The contribution from the Masses of interest in the sample as calculated by *VesuvioTOFFit* algorithm

	Name[L]	Value[Y]	Error[yEr]
1	f0.Mass	1.0079	0
2	f0.Width	3.7029109391033885	0.18188063043875158
3	f0.FSECoeff	0.4368286314341705	0
4	f0.C_0	21.261604304612487	0
5	f1.Mass	16	0
6	f1.Width	10	0
7	f1.Intensity	4.0306943033309777	0
8	f2.Mass	27	0
9	f2.Width	13	0
10	f2.Intensity	3.2381726871013714	0
11	f3.Mass	133	0
12	f3.Width	30	0
13	f3.Intensity	0.88263710029253195	0
14	Cost function value	3.1849817166533243	0

Figure 19: The fit parameters produced by the *VesuvioTOFFit* algorithm

6.1.1.3 VesuvioCorrections

VesuvioCorrections performs the corrections to fitted VESUVIO data. These corrections include a scaling factor for the fit, background contribution, total scattering contribution and multiple scattering contribution. The algorithm has input of a reduced VESUVIO workspace in time of flight and also requires the mass, mass profiles and intensity constraints used for fitting the data and the fit parameters calculated for the workspace. Additionally, the algorithm also requires details about the gamma background, the container and the sample material and beam in order to calculate multiple scattering. A full list of inputs can be found online at <http://docs.mantidproject.org/v3.7.1/algorithms/VesuvioCorrections-v1.html> .

VesuvioCorrections performs the corrections for one spectrum, as defined by the user, in the supplied workspace. The calculated fit parameters are used to construct a fit string which is then fit to the workspace containing the time of flight data to obtain the scaling values.

Gamma background corrections, **Figure 20**, are then calculated using the *VesuvioCalculateGammaBackground* algorithm which produces two workspaces, a corrected workspace, containing the time of flight data with the gamma background corrections applied to it, and the Background workspace which represents the corrections applied to the workspace. More information on this algorithm can be found online at <http://docs.mantidproject.org/v3.7.1/algorithms/VesuvioCalculateGammaBackground-v1.html> .

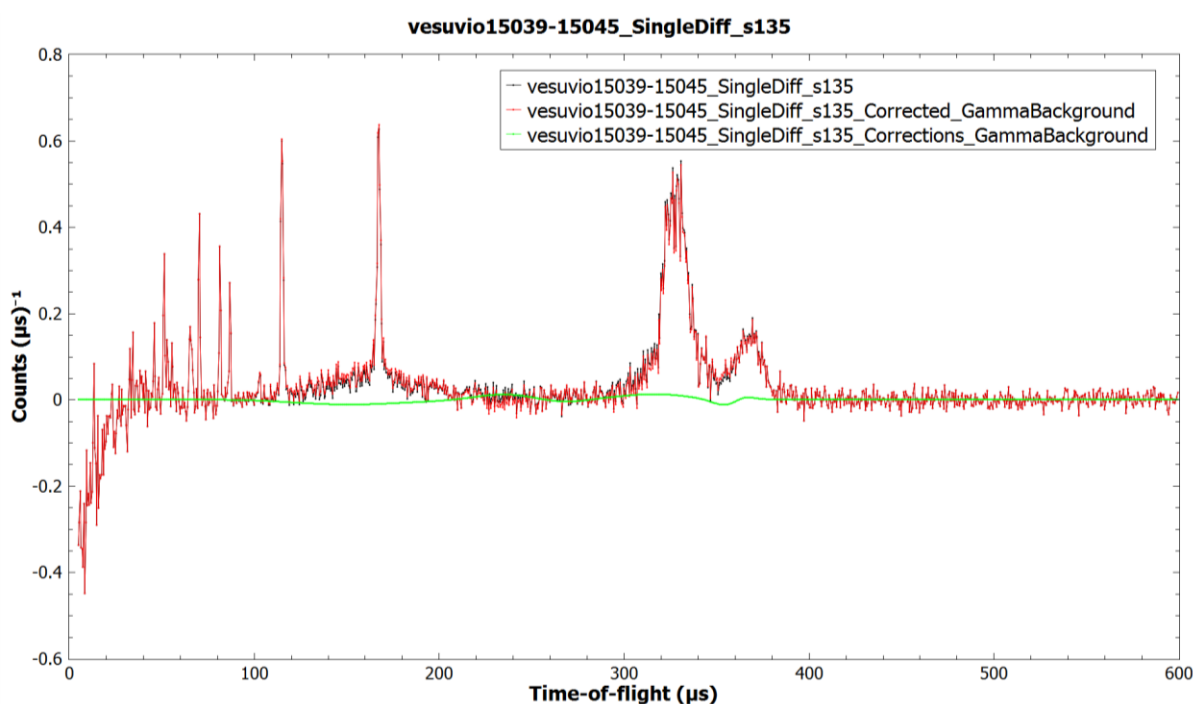


Figure 20 shows the gamma background contribution that is calculated by the *VesuvioCorrections* algorithm. The contribution (green) is the subtracted from the original data from **Figure 16** (black) in order to obtain the corrected data (red).

The next corrections are for multiple scattering and total scattering, **Figure 21**, which is calculated using the algorithm *VesuvioCalculateMS*. This algorithm follows the procedures to calculate multiple scattering for VESUVIO as defined by J. Mayers et al [12].

In order to run *VesuvioCalculateMS*, a sample shape must be defined for the input workspace. This is done using an xml definition that represents a cuboid with vertices that are formed by the user input for sample height, width and depth. This is applied to the workspace using the *CreateSampleShape* algorithm. The workspace is then passed to *VesuvioCalculateMS* which performs the multiple scattering corrections for the data. This will produce two workspaces, the total scattering workspace containing the counts for the total scattering, **Figure 22**, and the multiple scattering workspace containing the summed contributions of multiple scattering for all orders, **Figure 21**. After the execution is complete both the total scattering and multiple scattering workspaces are smoothed to a number of points defined by the users initial input. More information on *VesuvioCalculateMS* can be found online at <http://docs.mantidproject.org/v3.7.1/algorithms/VesuvioCalculateMS-v1.html> .

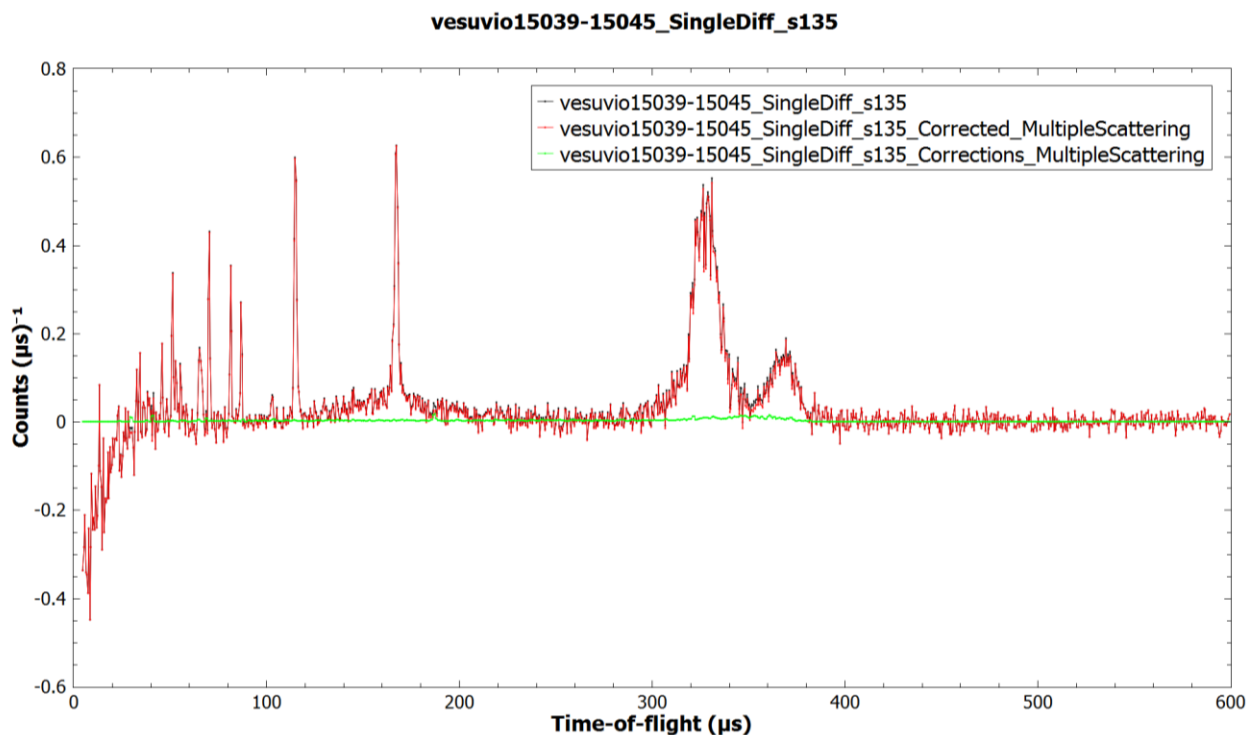


Figure 21: shows the multiple scattering contribution as calculated by the *VesuvioCorrections* algorithm. The multiple scattering (green) is subtracted from the original data from **Figure 16** (black) in order to show the corrected data (red).

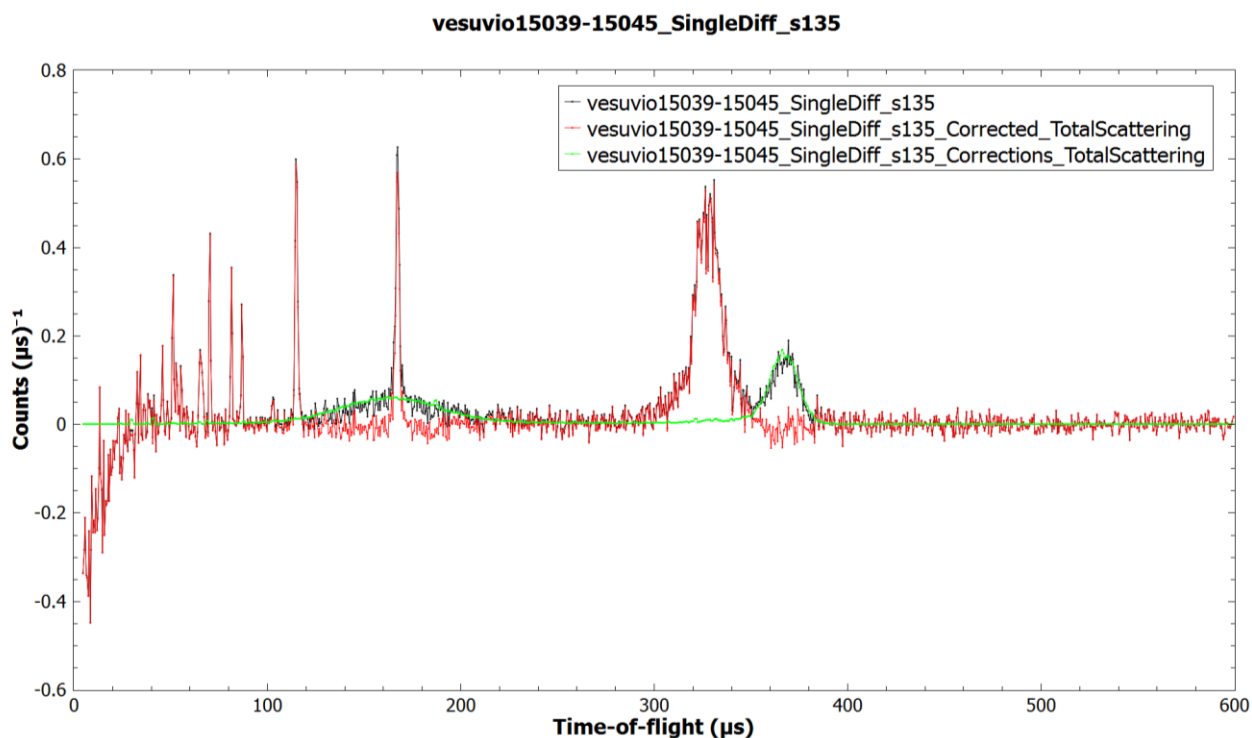


Figure 22: shows the Total scattering contribution as calculated by the VesuvioCorrections algorithm. The total scattering (green) is subtracted from the original data from **Figure 16** (black) in order to show the corrected data (red).

The VesuvioCorrections algorithm returns two group workspaces upon completion, one containing the corrections that will be applied to the data (workspaces in the ‘_corrections’ group workspace in **Figure 23**) and the other, the data after the corrections have been applied to it (workspaces in the ‘_corrected’ group workspace in **Figure 23**). Furthermore, the algorithm returns the parameters used to linearly fit the corrections data as well as the workspace containing all the corrections (total scattering, gamma background and multiple scattering) applied to the initial time of flight input workspace. **Figure 23** Shows these workspaces.

- vesuvio15039-15045_SingleDiff_s135_Corrected
 - vesuvio15039-15045_SingleDiff_s135_Corrected_GammaBackground
 - vesuvio15039-15045_SingleDiff_s135_Corrected_TotalScattering
 - vesuvio15039-15045_SingleDiff_s135_Corrected_MultipleScattering
 - vesuvio15039-15045_SingleDiff_s135_Corrections
 - vesuvio15039-15045_SingleDiff_s135_Corrections_GammaBackground
 - vesuvio15039-15045_SingleDiff_s135_Corrections_TotalScattering
 - vesuvio15039-15045_SingleDiff_s135_Corrections_MultipleScattering
 - vesuvio15039-15045_SingleDiff_s135_LinearFit
 - vesuvio15039-15045_SingleDiff_s135_Out

Figure 23: All workspaces that are outputs of the VesuvioCorrections algorithm displayed in the workspace windows of MANTID.

6.1.2 Scripts

Several VESUVIO specific scripts are now being shipped with MANTID. These scripts include common functionality used within the VESUVIO algorithms and are designed to support fit string parsing, the definition and comprehension of mass profiles and helper functions for the definition of complex background functions. All the scripts can be found in the `MantidInstall\scripts\Inelastic\Vesuvio` directory.

Additionally the main commands algorithm (`commands.py`) is contained in the same folder. This script contains the main workflow for the VESUVIO data reduction and is normally operated by a driver script. The driver script sets multiple flags that will be used as input for the function. These flags will include all the values mentioned in the description of the algorithms (see section 6.1.1) such as mass profiles and intensity constraints. The `commands.py` file also allows for multiple iterations of the fit to be carried out and this can be set in the driver script but defaults to a single iteration.

The full workflow of VESUVIO data reduction will include the algorithms *LoadVesuvio*, *VesuvioPreFit*, *VesuvioTOFFit* and *VesuvioCorrections*. The output of the main function in the `commands.py` script (`fit_tof`) can be by bank see **Figure 24**, or by spectra see **Figure 25**.

Figure 24 shows all the workspaces produced by a single iteration of the `fit_tof` function in `commands.py`, all workspaces are prefixed with the run numbers used to create them and then the remaining name describing what data is contained within it. **Table 2** shows the contents of each workspace using **Figure 24** as a reference.

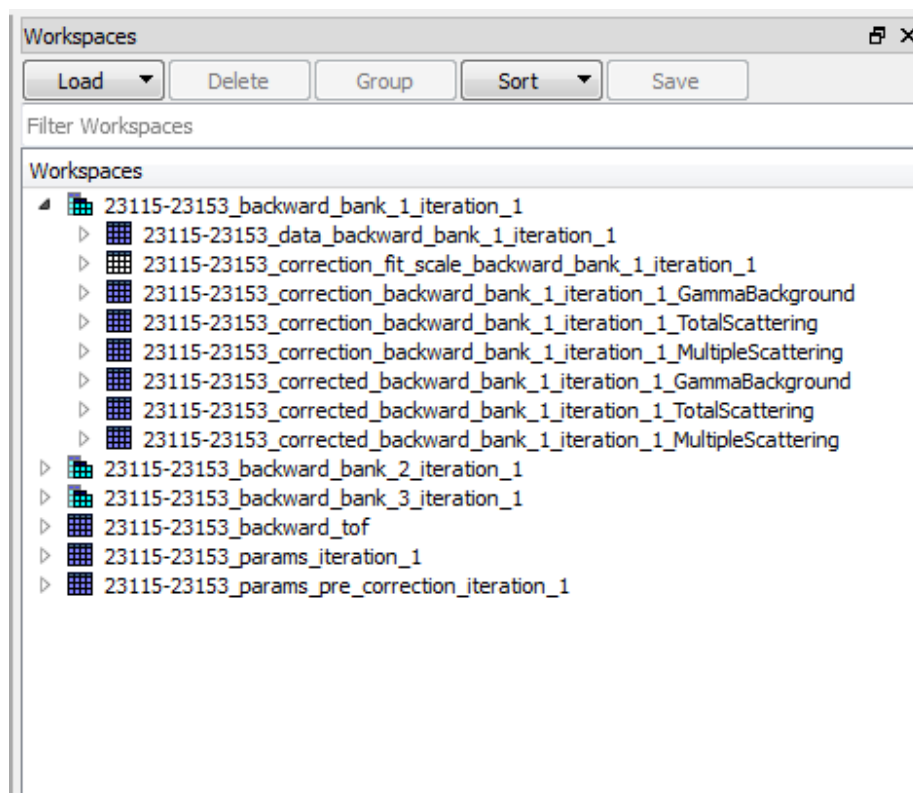


Figure 24: Output from a single iteration of the VESUVIO reduction workflow by bank

Workspace name	Workspace type	Contains
23115-23153_backward_bank_1_iteration_1	Workspace Group	Results of fitting the first of the 3 banks in back scattering. This workspace group contains the data, scaling parameters for the fit the correction data for the background, the total scattering and the multiple scattering contributions as well as the corrected fit curves for the background, total scattering and multiple scattering contributions. 23115-23153_backward_bank_2_iteration_1 and 23115-23153_backward_bank_3_iteration_1 contain the same as bank_1 but for each of the banks in back scattering.
23115-23153_backward_tof	Matrix Workspace	The reduced time of flight data for the backwards banks on VESUVIO for runs 23115-23153
23115-23153_params_iteration_1	Matrix Workspace	The fit parameters produced by the VesuvioTOFFit algorithm for each bank in back scattering after applying corrections
23115-23153_params_pre_correction_iteration_1	Matrix Workspace	The fit parameters produced by the VesuvioTOFFit algorithm for each bank in back scattering before corrections

Table 2: Shows the contents of each workspace produced by the VESUVIO reduction workflow

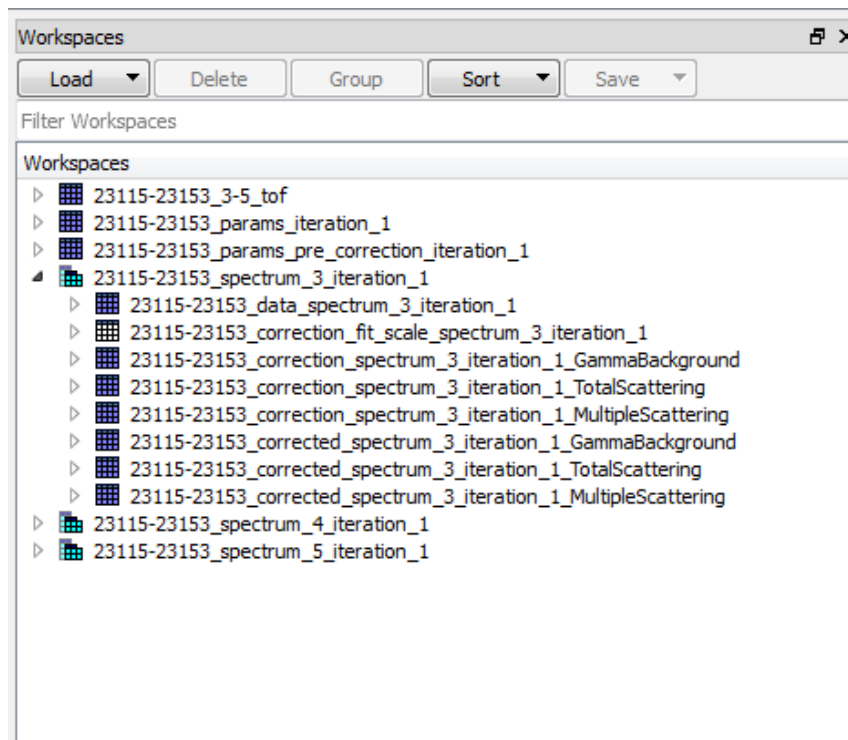


Figure 25: Output from a single iteration of the VESUVIO reduction workflow by spectra

6.2 Data loading and VESUVIO operation

Vesuvio data is loaded using the *LoadVesuvio* algorithm. *LoadVesuvio* is designed to take the raw VESUVIO data and convert it into time of flight spectra that, after corrections are applied, can be used for analysis. This algorithm takes, as input, a range of run numbers, a range of spectra, a differencing mode (which deliberates how the periods are treated – see section 6.2.2 for more detail), an optional instrument parameter file to adjust for the calibration of the instrument as well as options to sum the spectra defined in the list as well as loading monitor data (see section 6.2.4).

6.2.1 Periods

The VESUVIO instrument is capable of measuring data using different periods. The gold foils on the instrument that are situated between the sample and the detectors can be moved in and out to allow various orientations of data collection. This varies depending on the bank of detectors (forward scattering or back scattering) in use.

Forward scattering has 2 foil changers, left and right hand side of the beam, which have 2 sets of foils each for the corresponding 4 detector banks. This allows for 2 foil set ups, *foil-in* and *foil-out* which mean data is collected in 2 periods. If only the forward scattering detectors are used, detectors labelled S135-S198 in **Figure 26**, this data will have only exactly 2 periods.

Back scattering has a single foil changer with 3 possible states: *foil-out*, *foil-in* and *foil-thick*. *Foil-thick* is a double thickness foil is placed between the detectors and the sample as opposed to the single thickness foil observed in *foil-in*. This allows for the collection of 3 period data which only includes data that is collected from the back scattering detectors labelled S3-S134 in **Figure 26**.

VESUVIO is also capable of collecting 6 period data. This is a combination of collecting both 2 and 3 period data at the same time. The orientation of the foils in forward and back scattering is shown in **Table 3**.

The pre-existing *LoadVesuvio* algorithm present in MANTID has been adapted to correctly deal with all periods used on VESUVIO as well as the differencing modes (see section 6.2.2)

Period	Forward scattering Foil	Back scattering Foil
1	In	Thick
2	Out	In
3	In	Out
4	Out	Thick
5	In	In
6	Out	Out

Table 3: VESUVIO Foil orientation for 6 period data collection

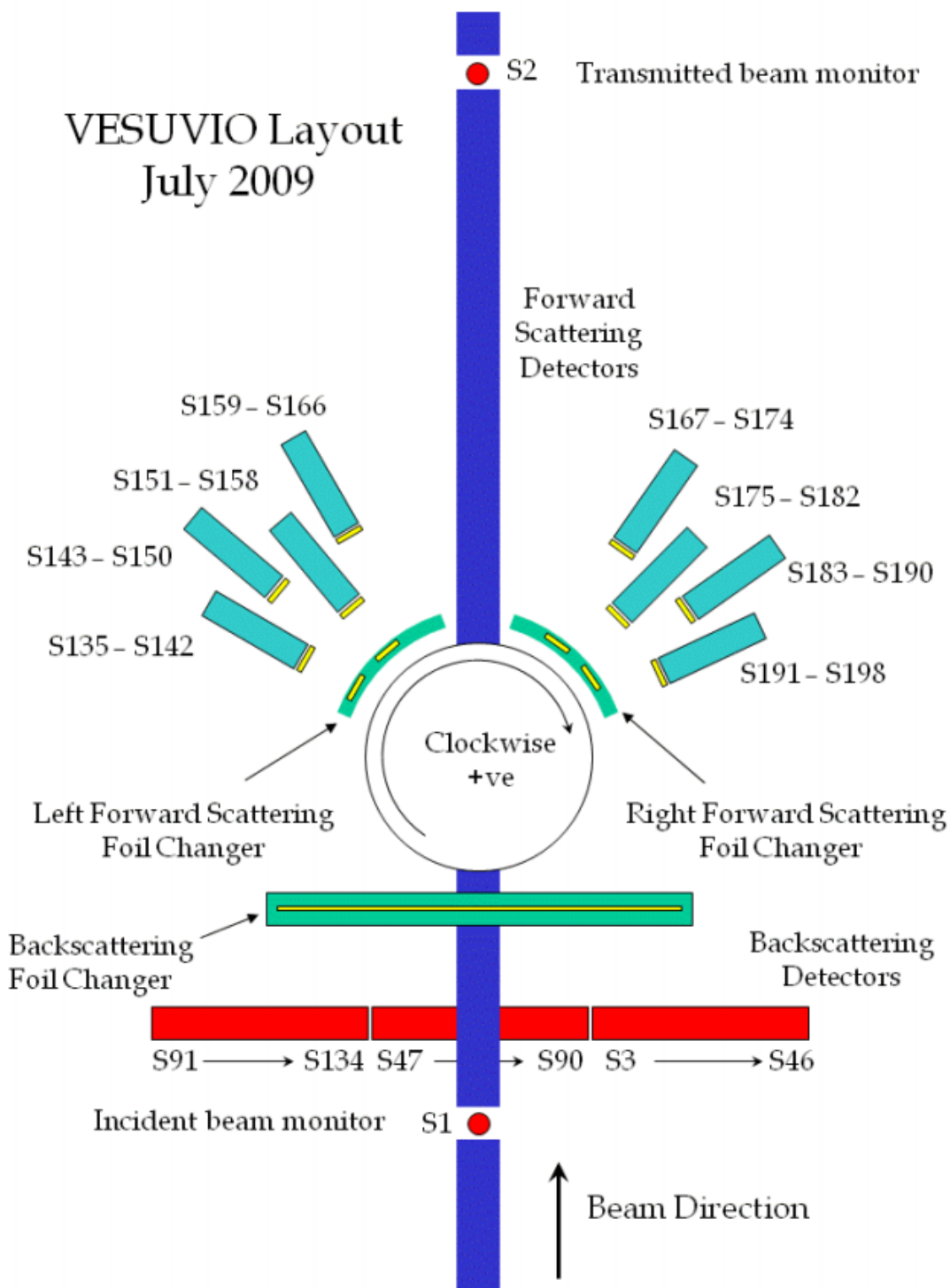


Figure A1.1 shows a schematic diagram of the VESUVIO spectrometer.

Figure 26: VESUVIO instrument Layout July 2009. Taken from User guide to VESUVIO data analysis [13]

6.2.2 Differencing modes

As VESUVIO has the ability to collect data with foils in different positions (see section 6.2.1), these can be treated in different ways to obtain differing results. VESUVIO currently has 3 main differencing options available and these are stated in **Table 4**. **Table 4** also details the equations that used in order to calculate the final time of flight spectra for Vesuvio that is the output of *LoadVesuvio*.

Differencing mode	Equation
Single Difference	$fi - fo$
Thick Difference	$ft - fo$
Double Difference	$fi - fo$ AND $ft - fo$

Table 4: Table showing the VESUVIO differencing modes and the equations relating to periods. Period positions are referenced as *foil-out (fo)*, *foil-in (fi)* and *foil-thick (ft)*. See section 6.2.1 for an explanation of foil states.

6.2.3 Difference Mode vs. Period validation

The *LoadVesuvio* algorithm will automatically determine the period of the data when it is loading based on the experiment information stored with the raw data. Given the number of periods, the spectra numbers that are being loaded (provided by the user upon execution) and the equations for differencing modes (see **Table 4**) it is possible to work out that if the current period/difference mode/scattering banks are of a valid combination. As such additional validation has been added to the *LoadVesuvio* algorithm to ensure that incorrect combinations of these variables are not possible to select, see **Figure 27**.

2 period

Gold foils only move in forward scattering, so only differencing in forward scattering is valid.

Scattering	Single Difference	Thick Difference	Double Difference
Forward	Valid	NEVER	NEVER
Back	Not Valid	Not Valid	Not Valid

3 period

Gold foils only move in back scattering, so only differencing in back scattering is valid.

Scattering	Single Difference	Thick Difference	Double Difference
Forward	Not Valid	NEVER	NEVER
Back	Valid	Valid	Valid

6 period

All are valid (with the exclusion of Thick Difference and Double difference in forward scattering - never valid)

Scattering	Single Difference	Thick Difference	Double Difference
Forward	Valid	NEVER	NEVER
Back	Valid	Valid	Valid

Figure 27: Valid combination of period/scattering/Differencing modes for VESUVIO data. The full version and accompanying documentation can be found online at <http://docs.mantidproject.org/v3.7.1/algorithms/LoadVesuvio-v1.html>

6.2.4 Load Monitors

The transmission spectrum for a sample on VESUVIO can be calculated and will show the probability of a neutron to be scattered by the sample as a function of wavelength. This is calculated by dividing the monitor spectra from the sample by the monitor spectra of the empty container, converting to wavelength and dividing the counts of the first monitor (labelled S1 in **Figure 26**) by the counts from the second monitor (labelled S2 in **Figure 26**). In order to make this operation easier for users the *LoadVesuvio* algorithm now has the option of loading the monitor workspaces along with the in a separate workspace, see **Figure 28**.

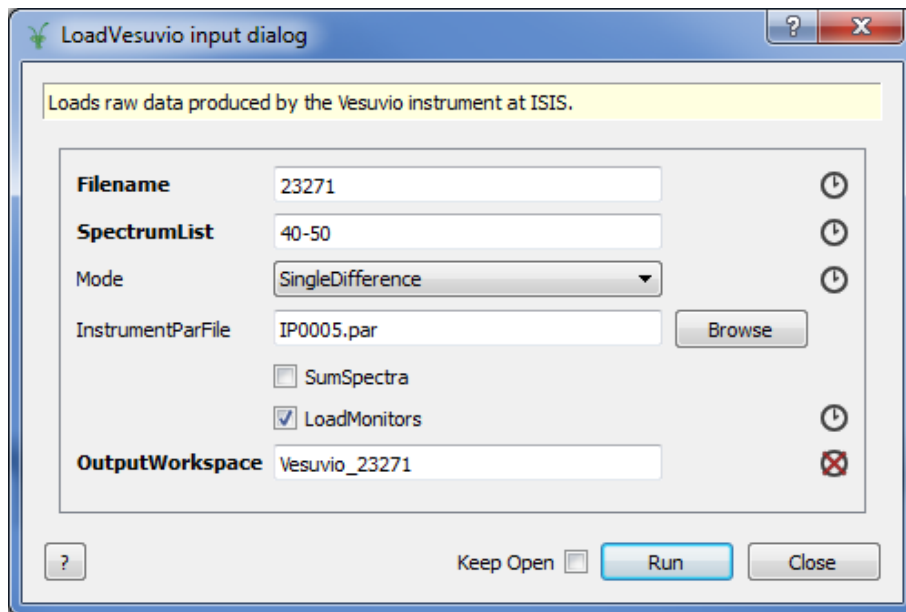
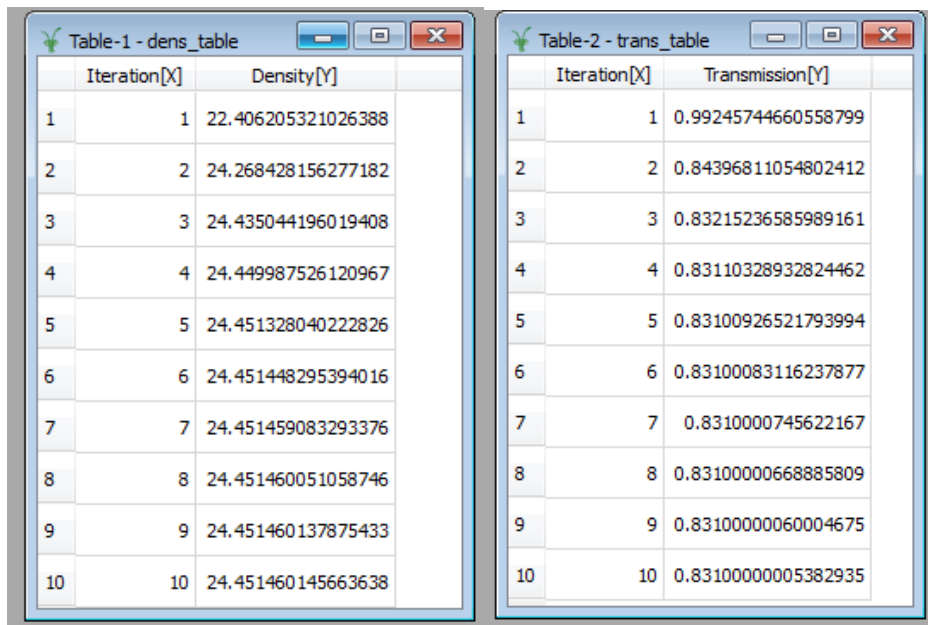


Figure 28: LoadVesuvio dialogue showing load monitor option

6.3 Vesuvio Thickness

The *VesuvioThickness* algorithm was originally designed to mirror the THICK routine on the VMS and calculates the sample density and transmission using a list of known masses within the sample and their corresponding amplitudes, the thickness of the sample (in cm), the number density of the sample material and an initial guess for the Transmission value.

The algorithm iteratively improves values for sample density and transmission over 10 iterations and then outputs the iterations of both values. These values are returned in two table workspaces, see **Figure 29**. The final value from the sample density iteration is then used as input to the *VesuvioCorrections* algorithm which uses the value to aid in calculating the multiple scattering contribution.



The figure displays two side-by-side table workspaces. The left workspace, titled 'Table-1 - dens_table', shows the iterative improvement of sample density. The right workspace, titled 'Table-2 - trans_table', shows the iterative improvement of transmission. Both tables have 10 rows corresponding to iterations 1 through 10.

Iteration[X]	Density[Y]
1	22.406205321026388
2	24.268428156277182
3	24.435044196019408
4	24.449987526120967
5	24.451328040222826
6	24.451448295394016
7	24.451459083293376
8	24.451460051058746
9	24.451460137875433
10	24.451460145663638

Iteration[X]	Transmission[Y]
1	0.99245744660558799
2	0.84396811054802412
3	0.83215236585989161
4	0.83110328932824462
5	0.83100926521793994
6	0.83100083116237877
7	0.8310000745622167
8	0.83100000668885809
9	0.83100000060004675
10	0.83100000005382935

Figure 29: Table workspace showing the iterative improvement of sample density (left) and transmission (right) over the 10 iterations of *VesuvioThickness*

6.4 Notable bug fixes for Vesuvio

During the implementation of VESUVIO in MANTID it was necessary to address some of the bugs that had arisen in the older algorithms.

6.4.1 Fitting data in VesuvioCorrections

The VesuvioCorrections algorithm (see section 6.1.1.3 for more detail) was showing unexpected behaviour for fitting multiple spectra in a reduced workspace containing VESUVIO data. It appeared that the fit from the first spectra was the same as all the following spectra and therefore the parameters were not propagated between fits.

The algorithm takes each workspace and extracts the spectrum to work on (this is a value obtained from the algorithm properties declared as "WorkspaceIndex"). The cause of the bug was discovered to be by the extracted spectrum not being passed to the next stage of the algorithm. Instead, the whole workspace containing the full list of spectra was being fitted and this resulted in the first spectrum in the workspace being fitted for every iteration of the algorithm.

In order to guard against similar problems in the future, the unit test for the algorithm (VesuvioCorrectionsTest.py) as well as the system test for the over-arching Vesuvio workflow (VesuvioCommandsTest.py) were updated. These tests now ensure that both the first and second spectrum of an input workspace are used in the algorithm. In addition, the values of the output workspaces are now tested. Previously only checks to ensure the expected workspaces were created and were of the correct type and size were in place.

6.4.2 Intensity constraints bug

When modelling data obtained on VESUVIO, in many cases, the composition of the sample will be known before any analysis takes place. With this, the relative intensities of each mass that contribute to the observed spectra from the sample can be calculated. In order to improve fitting, these relative intensities can be entered in the fit string (the input to the *Fit* algorithm that describes the model) as constraints between intensities. Given how VESUVIO defines masses to be used in fitting the constraint can be defined as follows:

```
flags['intensity_constraints'] = list([0,0,0,1, -0.25], [1,-0.5,0,0,0])
```

The above shows the `intensity_constraints` flag that is parsed as an additional argument when the fit string is constructed. In the above case, 5 masses are present in the sample. The constraints string describes two constraints on the intensity of each Compton profile (contribution from mass), the first that the final mass is 0.25 times smaller than the penultimate mass and the second is that the second mass is half the intensity of the first. The third mass has no constraints and is stated as 0 in both constraints.

In more complex intensity constraint scenarios, an issue was discovered causing intensity constraints to not be correctly applied to the fitting of the data. The results produced by the *VesuvioTOFFit* algorithm (see section 6.1.1.2 for more details) did not take into account the intensity constraints that were specified. This was however not noticed in simple cases as the *Fit* algorithm was able to make a close to accurate approximation to the expected simple constraints. However in the complex case, the *Fit* algorithm struggled to meet the constraints and was far more likely to choose the result of another acceptable local maximum.

The *VesuvioTOFFit* algorithm performs 2 fits; the first is a fit using the user specified number of iterations (defaulting to 500) and the second, a single iteration fit using the fit parameters of the first in order to assess the Chi squared value of the fit parameters. The bug is introduced here where the fit parameters from the second fit are used as the output of the *VesuvioTOFFit* algorithm. These parameters are created from running *Fit* using a fit string generated from the parameters of the initial fit but do not explicitly consider the constraints applied in the first fit. As such, the output fit parameters are liable to change even over a single iteration in the case of complex constraints on the data.

This problem is resolved by making the fit parameters that are returned by the *VesuvioTOFFit* algorithm the fit parameters that are obtained from the first call to *Fit* (the algorithm run that uses multiple iterations rather than the second call that only uses a single iteration). This is because the first set of fit parameters still contains the constraints that were specified by the user.

This *VesuvioTOFFit* unit test has been updated to validate this behaviour. The data previously used in the test was artificially generated and the origin of the fit parameters being provided was unknown. The test itself had already proven to be unstable and the decision was made to update the test to use better data which has since improved the stability of the test.

7 Simulations

There are several tools designed to support Simulations currently implemented in MANTID. These can be found in the simulation interface (Interfaces ⇒ Indirect ⇒ Simulation) and include MolDyn, Sassena and DensityOfStates.

MolDyn is designed to aid in the loading and visualisation of nMoldyn functions. The algorithms that support loading functions from nMoldyn version 3 and 4 are named *LoadNMoldyn3Ascii* and *LoadNMoldyn4Ascii*. Documentation for these algorithms can be found online at <http://docs.mantidproject.org/v3.7.1/algorithms/LoadNMoldyn3Ascii-v1.html> and <http://docs.mantidproject.org/v3.7.1/algorithms/LoadNMoldyn4Ascii-v1.html>.

Similar to the MolDyn interface, the Sassena interface is design to load simulations from the Sassena simulation software. The interface uses the *LoadSassena* algorithm, the documentation for this can be found at <http://docs.mantidproject.org/v3.7.1/algorithms/LoadSassena-v1.html>.

The DensityOfStates interface is used to load CASTEP data in both the CASTEP and PHONON file formats using the *SimulatedDensityOfStates* algorithm. The algorithm supports loading full and partial densities of states, Raman and IR spectroscopy. The documentation for this algorithm can be found online at <http://docs.mantidproject.org/v3.7.1/algorithms/SimulatedDensityOfStates-v1.html>.

7.1 Rebinning in Simulated Density of States

A bug was found with the rebinning parameter for the *SimulatedDensityOfStates* algorithm causing the output workspace to be scaled in the X axis rather than changing the binning.

In order to obtain the expected outcome from the *SimulatedDensityOfStates* algorithm in DOS mode, the peaks have to be created. The creation of peaks is handled by the *_draw_peaks()* function within the algorithm. Previously this had been using the user defined bin width as a scaling factor in X to draw the peaks whereas the width should have been used instead. After changing these variables, a call to the *Rebin* algorithm is used to set the binning of the output workspace to the binning expected by the user. This change also ensured that the interface for Simulated Density of States (the location the bug was original found in) worked as expected, see **Figure 30**. **Figure 30** shows the resulting workspace after running the algorithm with a binning of 3.0 (default binning is 1.0).

In order to validate this change, an additional test case was added to *SimulatedDensityOfStatesTest.py* to check that the rebinning had been correctly applied and the binning of the output workspace is that which is expected.

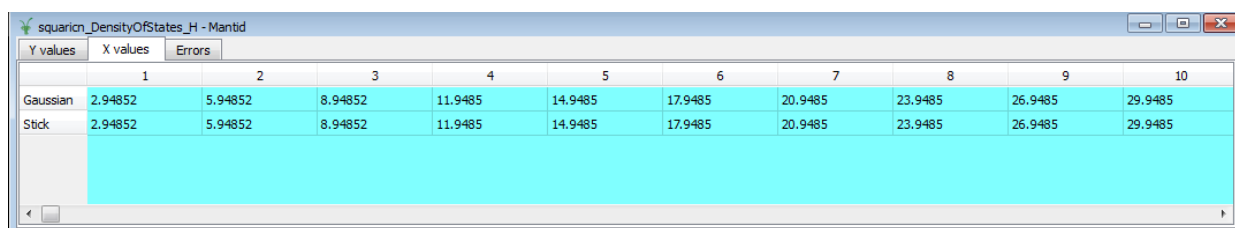


Figure 30: Shows the X axis binning for the H contribution in the suaricn data set.

8 Consistency and Robustness

Much of the user feedback for MANTID was to reduce the focus on the addition of new features in favour of improved consistency and robustness. This request has led to updates in the indirect interfaces to be more consistent with the rest of the project and to improve validation.

8.1 Progress tracking

To aid users, MANTID offers the facility to show the current completion of an algorithm as a percentage with optional additional information such as the current task being performed, see **Figure 31**. This has now been implemented in almost all indirect specific algorithms.

Progress tracking is implemented using a polling style system in which a Progress object is initialised to report within a percentage range of the algorithms completion. An initial guess for the number of times the progress percentage should be updated is also given. The progress is then updated by calling the report function on the progress object with an optional additional message to give further information.

The progress tracking bar is normally located in the bottom right hand corner of the MANTID GUI and has an accompanying button labelled 'Details'. When clicked, the 'Details' button shows further information regarding the algorithm that is currently being executed, see **Figure 32**. These details include input parameters, percentage completion and current task as well as the option to cancel the execution of the algorithm. The 'Cancel' button will abort the algorithm during its execution when it reaches the next point at which it was expecting to report its progress.



Figure 31: Progress Tracking bar during execution

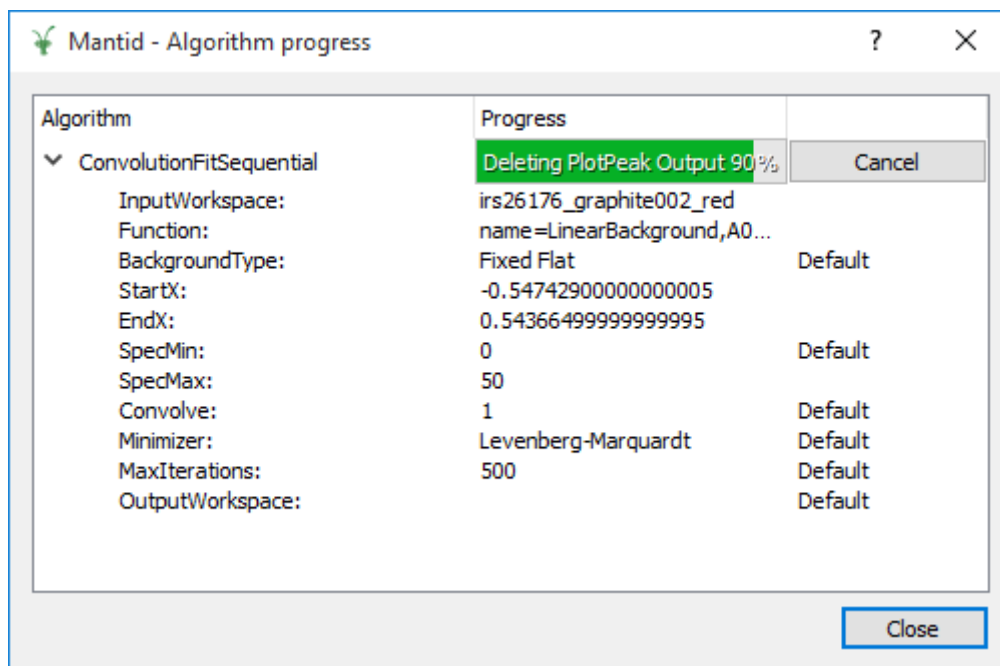


Figure 32: Progress tracking interface giving more details for *ConvolutionFitSequential* with IRIS data

8.2 Validation improvements

Many MANTID interfaces, including those in the indirect section, have known values or configurations that it would be impossible or illogical to compute. A significant effort has been made in indirect interfaces to ensure that they are stopping users from entering these values by the means of validation. This not only helps to warn users that they are entering invalid inputs but also directs them as to why the inputs are invalid.

Most indirect interfaces perform operations on spectra within a specified range of X values or a fitting range. This potentially allows a user to enter values that are either the same for the low and high fields or the incorrect way round, with low being more than high. Previously this would result in an uninformative error message in the results log. Now an error message is displayed stating the problem, see **Figure 33**. In **Figure 33**, the EMin has been set to a larger value than EMax and when the tab is run, a message box is produced and the algorithm is not executed until the problem is resolved.

When numeric spinners are used, it is possible to restrict the upper and lower bounds to ensure that values entered are expected. **Figure 34** shows an example from the Data Reduction interface where this is used. The instrument has been set to IRIS with graphite analysers and a reflection of 002. In this configuration it is known that spectra of interest are only between a minimum of 3 (1 and 2 are monitors) and a maximum 53. In order to stop potentially invalid user input, the numerical spinners are bounded making it impossible to enter values outside of this range.

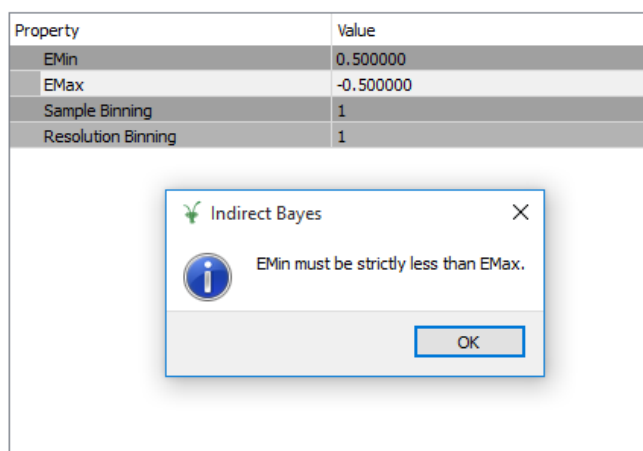


Figure 33: Error message displayed with invalid EMin/EMax range in Indirect Bayes

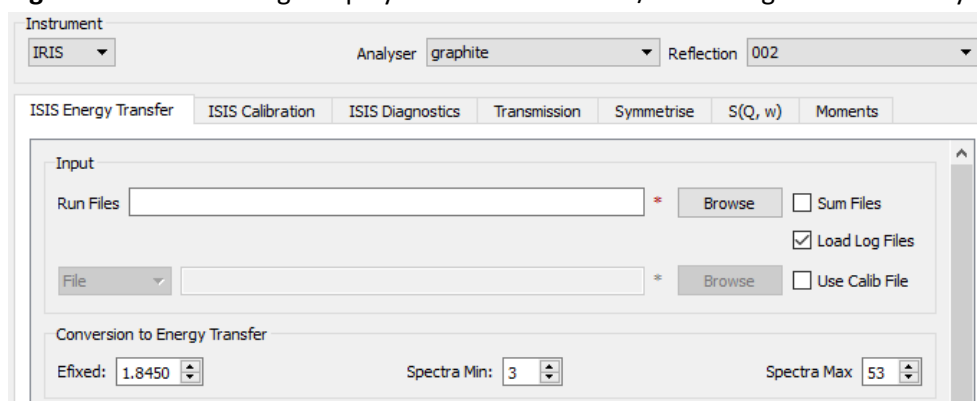


Figure 34: Bounded Spectra Min/Max spinners in ISIS Energy Transfer for IRIS graphite 002

9 Documentation

9.1 Release Notes

The release notes, previously stored as a standalone set of pages on the mantidproject.org webpage are now integrated with the build. This ensures that within our MANTID workflow (see section 10.4) the initial reviewer checks also include reading and reviewing the release notes as well as checking if embedded links work as expected.

Figure 35 shows the indirect release notes for the MANTID release 3.7.1, however, a full set of release notes for all scientific areas of MANTID are available online at <http://docs.mantidproject.org/v3.7.1/release/>

Table of Contents

- New features
 - Algorithms
 - Data Analysis
 - Jump Fit
 - Diffraction
 - Vesuvio
- Improvements
- Bugfixes

New features

Algorithms

- *IndirectNormSpectra* algorithm is designed to normalise all spectra in a MatrixWorkspace so that the maximum value for any spectra is 1.
- *IqtFitSequential* algorithm has been added to sequential Iqt Fit data. This algorithm will be mainly used in the IqtFit interface.

Data Analysis

Jump Fit

- The interface now has the option to plot a guess of what the fit will look like before running the algorithm.
- The Plot button is no longer present in the interface as it is no longer used.

Diffraction

- OSIRIS Diffraction DiffOnly interface and the *OSIRISDiffractionReduction* algorithm now support the use of multiple container runs. Additional validation also ensures you have the same number of sample/vanadium/container runs.

Vesuvio

- **The following Mantid algorithms used for Vesuvio have been added:**
 - *VesuvioPreFit*
 - *VesuvioTOFFit*
 - *VesuvioCorrections*
- The script used to process data for Vesuvio has also been added. This used to be called *VesuvioWorkflow.py*, but is now named *VesuvioCommands.py*.
- **The following Vesuvio specific algorithms have been updated to have their name prefixed by Vesuvio:**
 - *VesuvioCalculateGammaBackground* previously *CalculateGammaBackground*
 - *VesuvioCalculateMS* previously *CalculateMSVesuvio*
 - *VesuvioDiffractionReduction* previously *EVSDiffractionReduction*
- *LoadVesuvio* now has the option to load the monitor data in addition to its normal operation. This is loaded as an additional separate workspace.
- Added a fit function to fit a multivariate Gaussian profile (*MultivariateGaussianComptonProfile*)

Figure 35: MANTID Release notes for the Indirect Inelastic changes. The full list of indirect inelastic specific changes can be found online at http://docs.mantidproject.org/v3.7.1/release/v3.7.1/indirect_inelastic.html

9.2 Documentation improvements

Much work in this period has gone towards improving the standard of documentation for indirect algorithms, in particular the workflow algorithm. It has become standard practice in MANTID that to aid both the software developers and the users understanding of complex algorithms, workflow diagrams are used in the documentation to explain the flow of execution of the algorithm.

Workflow diagrams are written in the DOT plain text graph description language and then compiled with the documentation and displayed graphically. An example of a simple workflow diagram used in MANTID can be seen in **Figure 36**. **Figure 37** shows a simple dot graph and the accompanying definition used as an example in Ref [14].

Almost all indirect workflow algorithms used in MANTID now have documentation accompanied by a workflow diagram. The only notable exceptions are the newer VESUVIO algorithms for which workflow diagrams have yet to be created. All the updated workflow algorithms can be found at the links provided in **Table 5**.

Algorithm	Documentation link
ConvolutionFitSequential	http://docs.mantidproject.org/v3.7.1/algorithms/ConvolutionFitSequential-v1.html
FlatPlatePaalmanPingsCorrection	http://docs.mantidproject.org/v3.7.1/algorithms/FlatPlatePaalmanPingsCorrection-v1.html
IndirectAnnulusAbsorption	http://docs.mantidproject.org/v3.7.1/algorithms/IndirectAnnulusAbsorption-v1.html
IndirectCylinderAbsorption	http://docs.mantidproject.org/v3.7.1/algorithms/IndirectCylinderAbsorption-v1.html
IndirectFlatPlateAbsorption	http://docs.mantidproject.org/v3.7.1/algorithms/IndirectFlatPlateAbsorption-v1.html
IqtFitMultiple	http://docs.mantidproject.org/v3.7.1/algorithms/IqtFitMultiple-v1.html
IqtFitSequential	http://docs.mantidproject.org/v3.7.1/algorithms/IqtFitSequential-v1.html
OSIRISDiffractionReduction	http://docs.mantidproject.org/v3.7.1/algorithms/OSIRISDiffractionReduction-v1.html
ProcessIndirectFitParameters	http://docs.mantidproject.org/v3.7.1/algorithms/ProcessIndirectFitParameters-v1.html
SimulatedDensityOfStates	http://docs.mantidproject.org/v3.7.1/algorithms/SimulatedDensityOfStates-v1.html
TOSCABankCorrection	http://docs.mantidproject.org/v3.7.1/algorithms/TOSCABankCorrection-v1.html
VesuvioDiffractionReduction	http://docs.mantidproject.org/v3.7.1/algorithms/VesuvioDiffractionReduction-v1.html

Table 5: Table showing indirect specific algorithms with accompanying links to their updated documentation pages.

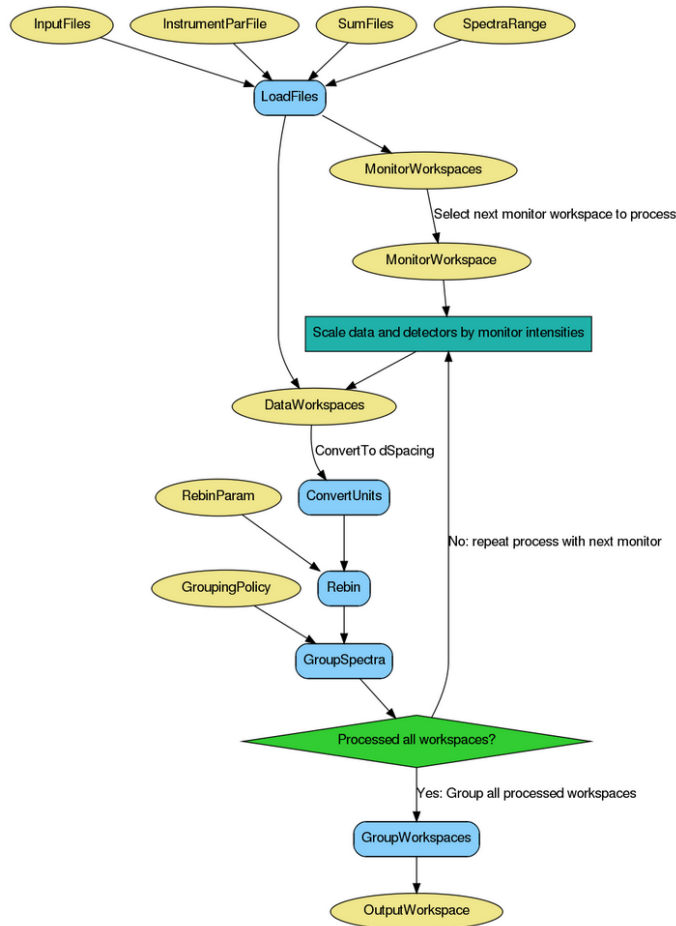


Figure 36: Workflow diagram for VesuvioDiffractionReduction. A size version is available at: <http://docs.mantidproject.org/v3.7.1/algorithms/VesuvioDiffractionReduction-v1.html>

```

1: digraph G {
2:   main -> parse -> execute;
3:   main -> init;
4:   main -> cleanup;
5:   execute -> make_string;
6:   execute -> printf;
7:   init -> make_string;
8:   main -> printf;
9:   execute -> compare;
10: }

```

Figure 1: Small graph

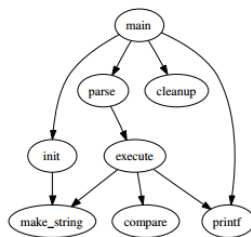


Figure 2: Drawing of small graph

Figure 37: Simple DOT graph used as an example in the *dot User's Manual* [14]

10 Testing

10.1 Test best practices

To ensure that development changes made by the MANTID team do not have unexpected effects on the code base, a suite of system and unit tests are in place that run automatically when a new change is proposed in a Pull Request (see section 10.4). Unit tests, as well as system test cases (see below); can be separated into two sections, success cases and failure cases.

Success cases are designed to exercise the algorithm in the way they are expected to be used and test the functionality of the algorithm. These will cause the algorithm to run without failure and produce an outcome that can be verified by comparison of data values or, more often done in system tests, comparing the data to a reference file. Normal data and data that can be considered an edge case, extreme example, are tested in success case tests.

Failure cases are used to test how the algorithm copes with unexpected input. These tests are designed to fail when they are run and they should catch an exception that is thrown by the algorithm. In many cases, when testing unexpected input, this will be a `ValueError` either thrown by the property validator or by the `validate()` function in the algorithm. Failure cases normally do not test the execution of the algorithm workflow as they should fail before that point.

10.2 Types of tests in MANTID

Unit tests are small isolated tests that are designed to test the functionality of a single segment of code. In most cases these are written for a single algorithm. The aim is to ensure that, ideally, all execution paths of the code are covered by the test including as many edge cases as possible. Their execution time should be kept a short.

System tests are more often used to test a number of steps a user takes to reach a certain goal for example the `IRISIndirectInelasticFit` test found in the `IRISIndirectInelastic` system test performs the same operations as the `I(Q, t)` and `I(Q, t) Fit` interface sequentially with the standard IRIS data set (see section 10.3.1). These tests will normally have a longer run time and are a more realistic representation of how users interact with MANTID. Full version of `IRISIndirectInelastic` system test available from:

<https://github.com/mantidproject/mantid/blob/master/Testing/SystemTests/tests/analysis/IRISIndirectInelastic.py>

Most documentation contains a usage example which is a simple example of how to use an algorithm in a python script format. Usage examples may include a test to ensure the expected output is obtained. This output as well as an external links to documentation, such as workflow diagrams, are checked by the documentation tests.

Several static analysis tools, designed to ensure code is syntactically correct and properly formatted, check the whole code base. There are four internal project tools that are used regularly to checked for consistency:

- Clang format – Ensures C++ code is formatted according to a C++ standard of your choosing (the default for the MANTID project being LLVM) more information for LLVM can be found here <http://llvm.org/docs/CodingStandards.html>
- Pylint – Ensures python code is formatted correctly in line with the PEP8 style guide including restrictions on line length, variable names and module imports [15]. Pylint also offers help in error detection with validating function names and implementation.
- Doxygen – Ensures that C++ style comments are correctly formatted and use the correct syntactical tags to define metadata.
- CppCheck – Performs checks on C++ code for errors, warnings, style, performance and portability.

10.3 Indirect specific testing

In order to test the functionality of MANTID algorithms and scripts, instrument specific data files are used and stored in the MANTID test data directory. As many of the algorithms are independent of instrument, the IRIS data set (see 10.3.1.1) is used to perform the bulk of the testing with the OSIRIS (see 10.3.2.1) and TOSCA (see 10.3.2.2) data sets are used to test only instrument specific algorithms. As VESUVIO operation differs from the other indirect geometry instruments, the IRIS data set cannot be used to test its algorithms, as such it has its own data set (see section 10.3.3.1).

10.3.1 IRIS

10.3.1.1 IRIS testing files

- Sample: 26176
- Container: 26174
- Resolution: 26173

Reduced versions of these files are already stored in the Mantid test data directory. The test data directories contain these files in various states, see **Table 6**.

Run number	Name	Description
IRS26176 <i>Sample</i>	IRS26176.raw	Raw experimental data used to test loading and workflow (system tests).
	irs26176_graphite002_red.nxs	Reduced raw file. Used as the input for the sample data.
	irs26176_graphite002_cyl_Abs.nxs	Cylinder absorption corrections workspace used in unit tests for corrections algorithms.
	irs26176_graphite002_conv_1LFixF_s0_to_9_Result.nxs	A single Lorentzian model fitted to the irs26176 data set in the ConvFit interface. This is used in further analysis such as JumpFit.
	iris26176_graphite002_iqt.nxs	An I(Q, t) file for use in the <i>IqtFitMultiple</i> and <i>IqtFitSequential</i> testing. <i>Note: This file is the first to use the new naming convention ('iris' as opposed to 'irs'). Other files will be updated to this convention in the future.</i>
IRS26173 <i>Resolution</i>	IRS26173.raw	Raw experimental data used to test loading and in workflow system tests.
	irs26173_graphite002_res.nxs	Reduced raw file of the resolution.
	irs26173_graphite002_red.nxs	Reduced raw file of the resolution as a reduction. Used in the unit test for ResNorm.
	irs26173_graphite002_ResNorm.nxs	Output of the ResNorm algorithm. Used for testing Bayesian analysis

Table 6: The standard testing data used for iris

10.3.2 OSIRIS and TOSCA

OSIRIS and TOSCA files are only used for specific OSIRIS and TOSCA algorithms and in most cases this is for the loaders. Once data has been reduced, the majority of indirect algorithms will be executed in the same way regardless of the instrument definition. Therefore the only files stored in the MANTID test data directory are the raw files. These file are used for unit, system and doc tests.

10.3.2.1 OSIRIS testing files

Run Number	Name	Description
OSI10203-10204 <i>Sample</i>	OSI10203-10204.raw	Raw files containing the sample run data.
OSI10241-10242 <i>Container</i>	OSI10241-10242.raw	Raw files containing the container run data.
OSI10156-10157 <i>Vanadium</i>	OSI10156-10157.raw	Raw files containing the vanadium run data.
Osiris_041-RES10 <i>Calibration</i>	Osiris_041-RES10.cal	The aerial calibration file containing the offsets for the detector positions as well as any masking for spectra.

Table 7: OSIRIS standard testing data

10.3.2.2 TOSCA testing files

Run Number	Name	Description
TSC14007 <i>Sample</i>	TSC14007.raw	An older TOSCA raw data file containing sample information used for testing compatibility with older formats.
TSC15352-15354 <i>Sample</i>	TSC15352-15354.raw	Newer TOSCA raw data file containing sample information. These are the more commonly used files now.

Table 8: TOSCA standard testing data

10.3.3 VESUVIO

10.3.3.1 VESUVIO testing files

VESUVIO unit tests use artificially generated data where possible. This is to enforce encapsulation of the test by restricting its access to *LoadVesuvio* as well as reduce the number of raw files stored in the test directory and reduce the execution time of the tests. The files are generated from the script *testing.py* stored in the *inelastic/Vesuvio/* directory of the code base. The script has two functions, *create_test_ws()* which returns a workspace containing data and instrument meta data replicating a sample and *create_test_container_ws()* which returns a workspace containing data and instrument meta data replicating a container. The raw data used to create these is taken from the reduced data used in the system tests which is EVS14188-15045 using single difference and spectra numbers 134-135 (the first two back scattering detectors).

In order to test the workflow of VESUVIO data reduction and analysis, the system test *VesuvioCommands.py* uses all the algorithms required for a full reduction and analysis run from their unprocessed (.raw) state. The files used are EVS14188-15045 reduced in the single differencing mode and using spectra numbers 134-135. Additionally, the system tests use the *Vesuvio_IP_file_test.par*. This is a mock version of an actual Vesuvio IP file and is used in tests to ensure that adjustments are made relative to its contents.

10.3.3.2 VesuvioCorrections system test

Due to the complex operations performed by the *VesuvioCorrections* algorithm the execution time is longer than most algorithms. Unit tests are designed to be short and to test the algorithm in question in isolation but due to a long run time, the decision was made to move this test to be a system test. This has not affected performance or the values checked by the test itself, but the location and layout of the test file *VesuvioCorrectionsTest.py* has changed. This is an internal change to MANTID with no effect on the user.

10.3.3.3 Testing meaningful values

Many of the unit and system tests used previously for VESUVIO required updating as their expected values were not those being produced by the algorithms. This was due to the age of the tests and changes that had taken place to the algorithms that had already been implemented in MANTID, such as *LoadVesuvio*.

Additionally to updating the values, the aspect of the data being tested was also changed. Originally, the test checked the extremities of the data (the first and last bin). Whilst this is a reasonable testing practice in the majority of cases, due to the noisy background of VESUVIO data, see **Figure 38**, testing a more meaningful value such as the maximum height of the peak and its corresponding bin index is more appropriate.

Whilst most changes were made during the development period leading up to the 3.7 release, some of the final changes were made just after the release. As such, all tests will be using the peak height and bin index test values in 3.8.

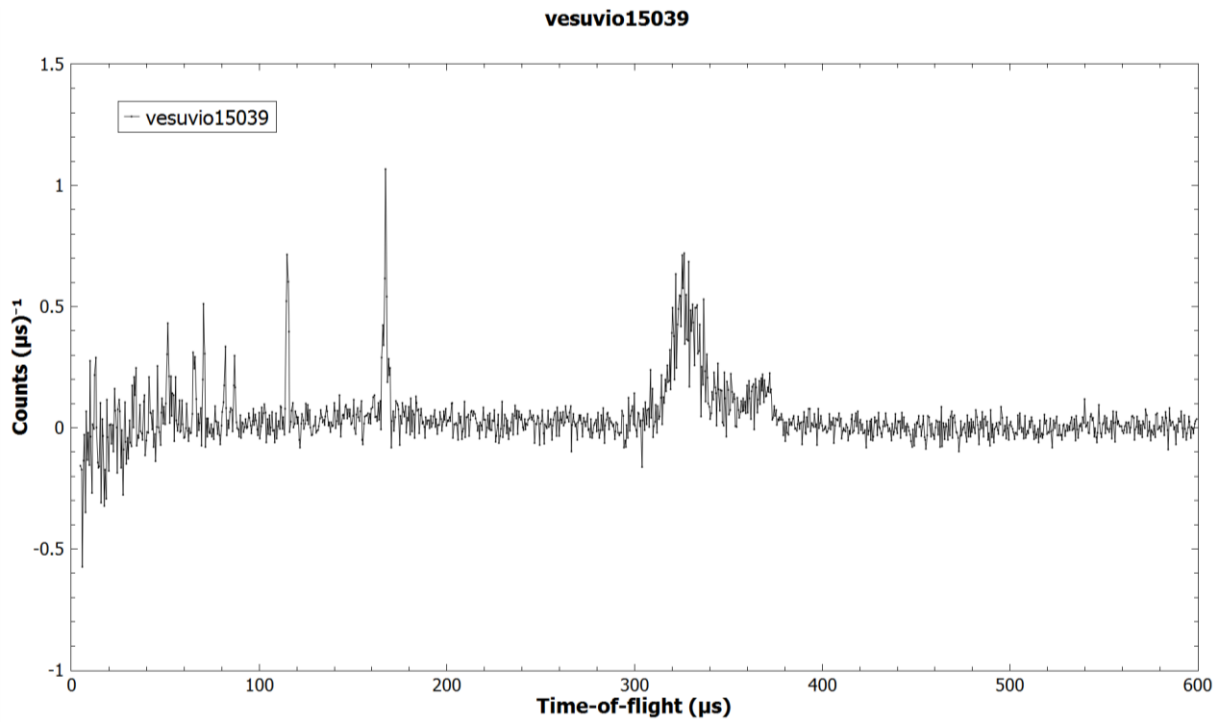


Figure 38: Example of VESUVIO data from run 15039 loaded with the LoadVesuvio algorithm and single difference mode highlighting noise in the initial bin.

10.3.3.4 Tolerance values and justification

During test migration, some Linux systems were producing marginally different values to Windows and more recent distributions of linux. This minor difference is within scientific error, but was causing an issue in the comparison of test data. More specifically, unit tests that were expecting specific values were failing due to the small amount of difference in the values.

This difference in values has been found to be caused by a change in the boost library, upon which MANTID depends. In more recent versions of the boost library, those newer than 1.56, the random number generation used in `boost::variate_generator` was altered.

The algorithm *VesuvioCalculateMS* uses the `variate_generator` and, as such, this the algorithm will produce marginally different values depending on the version of boost that is standard in the operating system. Due to this, the algorithms that run *VesuvioCalculateMS* are now required to have a tolerance on certain values for testing. This was deemed a more acceptable solution than having multiple lists of values that are expected between version as the difference between version is small. The ideal scenario would be to check the version of boost in use but unfortunately this operation is not easily performed in python.

These tolerance values will no longer be required when all operating systems are using newer versions of boost. When all supported platforms running MANTID have upgraded, it will be possible to remove the tolerance.

10.3.4 Simulations

10.3.4.1 Simulations testing files

Simulations algorithms require a different set of data as they are designed to take input from external simulation packages such as Sassena and nMoldyn. The files used to test simulations algorithms are shown in **Table 9**.

Type	Name	Description
nMoldyn	NaF_DISF.cdl	nMoldyn version 3 .cdl file format. Contains a function called 'Sqw-total'
	WSH_test.dat	nMoldyn version 3 .dat file format.
	/nmoldyn4_data/	Directory containing a set of nMoldyn version 4 functions.
Sassena	outputSassena_1.4.1.h5	An output from the Sassena simulations package used to test the <i>LoadSassena</i> algorithm
SimulatedDensityOfStates	squaricn.phonon	A PHONON file used to test the <i>SimulatedDensityOfStates</i> algorithm
	squaricn.castep	A CASTEP file used to test the <i>SimulatedDensityOfStates</i> algorithm

Table 9: Describes the files used to test simulations algorithms in MANTID.

10.4 Modifying Mantid Workflow

MANTID uses Git and GitHub for issue tracking, bug reports as well as new development and the workflow for fixing these bugs has changed in the 3.6 release development period, see **Figure 39**. The workflow remains similar in many respects to its description by D. Nixon (*figure 27 [7]*) with the exception of the introduction of Gatekeepers. Gatekeepers, or second reviewers, are more senior members of the development team who perform a second review of the proposed changes. Functionality is rarely checked in the second review phase but instead coding style and best practices are reviewed to attempt to find points in which code can be better optimised or more clear.

After every change that is made to the proposed solution, all tests are run and if any fail then the code cannot be merged in the main repository. The way in which tests execute has been changed in order to streamline the build servers. Previously all static analysis, unit and system tests ran simultaneously but in order to reduce the amount of redundant tests being run these are now run in a sequence. Clang Format is run as it is the fastest and has the shortest fix time. In the event of this test failing, all subsequent tests will be cancelled and no other tests will be run on the build server until the Clang Format issue is fixed. **Figure 40** shows the order of the remaining tests as well as their dependencies on one another.

To ensure all changes are correctly tested and a good test description is supplied for the reviewer, a new pull request template has been added to ensure consistency across all pull requests, see **Figure 41**. When changes are made to the code base locally, a developer must submit a pull request in order to have their code reviewed and, providing it is of good quality and the tests pass, merged with the main code for the project. This template encourages the developer to fill in a description of the changes, a guide on how to test the changes, a link to any release notes that may have been updated as well as which issue number the pull request will resolve. The most important section is the guide to testing the changes, which should mention any files that are required and where they can be found, as well as a step by step guide of how to test the code, as shown in **Figure 42**. It is expected that tester also attempt to break the change to ensure robustness. This is normally done by using unexpected values and observing how the newly developed code handles this.

The pull request template also contains a section below that is used by the first reviewer. The first reviewer should follow these steps and ensure that the changes made by the initial developer adhere to specifications stated (where applicable). Once the first reviewer is satisfied with the changes they add a comment detailing the steps they took to review the code as well as any additional comments and end this with the shipit emoji, as mentioned at the end of **Figure 41**. The Gatekeepers are then alerted that the changes are ready for a second review.

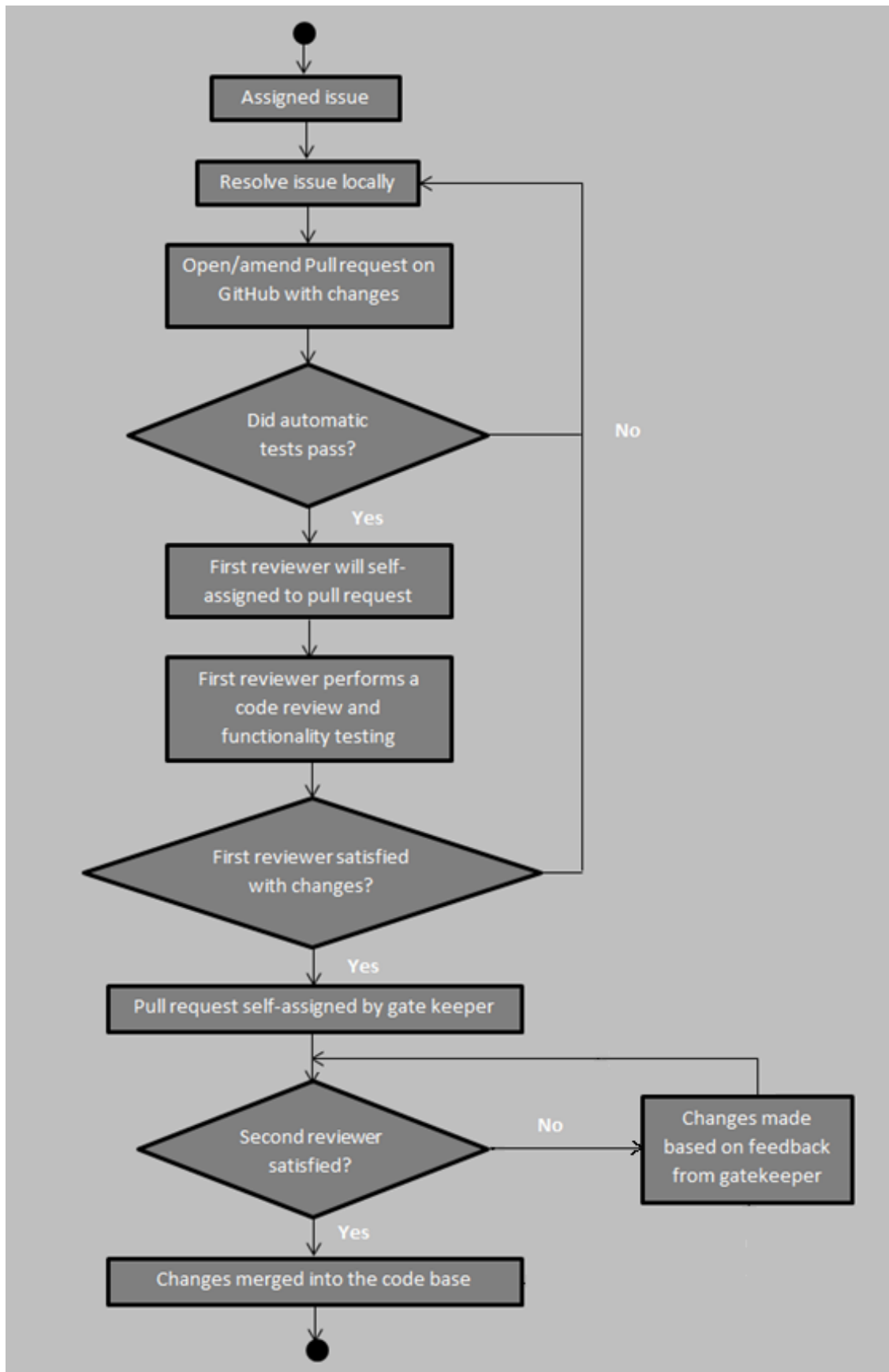


Figure 39: Workflow for changing code in the MANTID project

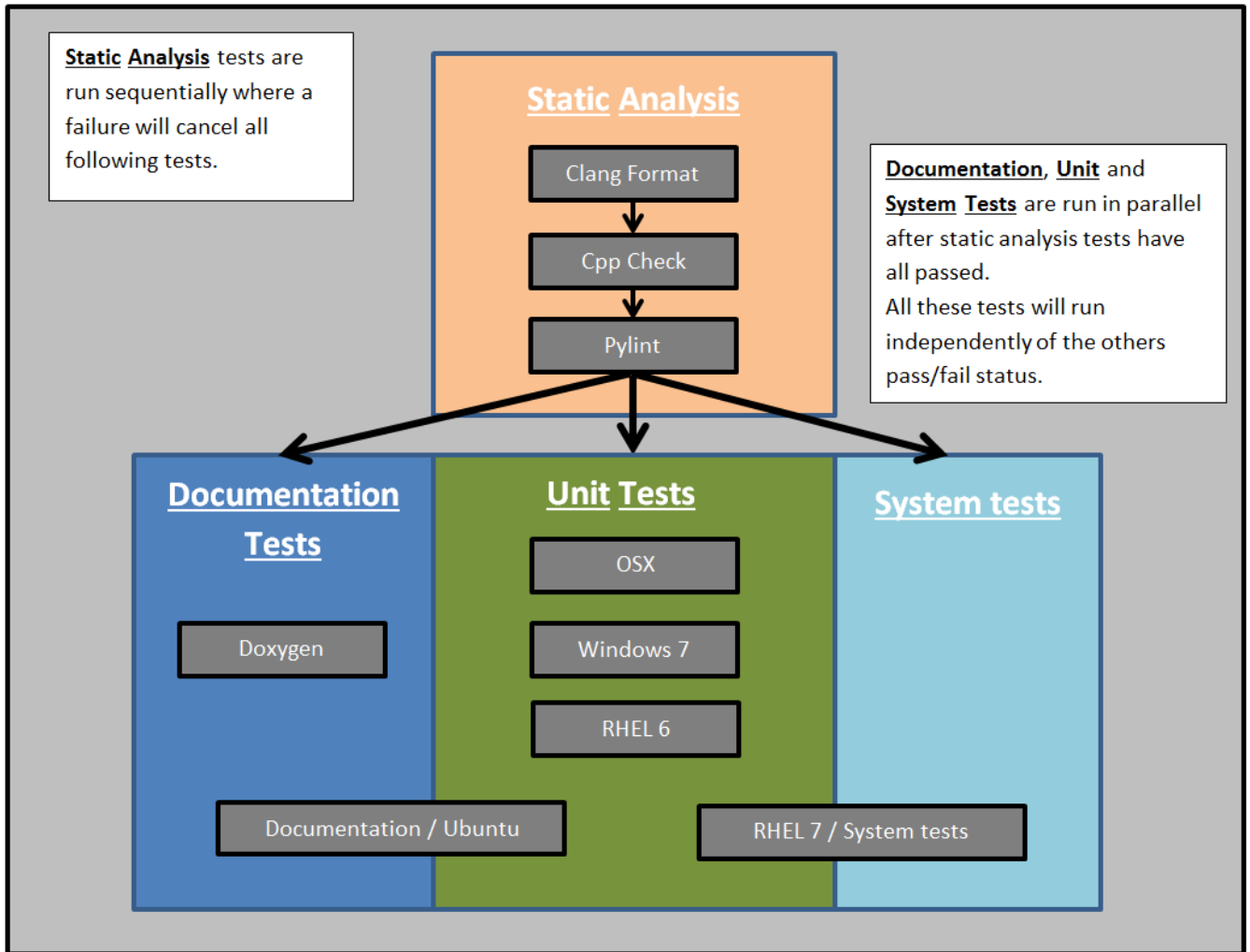


Figure 40: Shows the order in which tests run on the MANTID build servers as well as their dependencies on one another. Ubuntu unit tests and Documentation tests are bundled into a single test and similarly, RHEL 7 and system tests are bundled.

Pull Request Template ⚙

Write
Preview

Description of work.

To test:

Fixes #xxxx.

Reviewer

Please comment on the following ([full description](#)):

Code Review

- Is the code of an acceptable quality?
- Does the code conform to the [coding standards](#)? Is it well structured with small focussed classes/methods/functions?
- Are there unit/system tests in place? Are the unit tests small and test the a class in isolation?
- If there are changes in the release notes then do they describe the changes appropriately?

Functional Tests

- Do changes function as described? Add comments below that describe the tests performed?
- How do the changes handle unexpected situations, e.g. bad input?
- Has the relevant documentation been added/updated?
- Is user-facing documentation written in a user-friendly manner?
- Has developer documentation been updated if required?
- Does everything look good? Comment with the ship it emoji but don't merge. A member of [@mantidproject/gatekeepers](#) will take care of it.

📖 Styling with Markdown is supported
Create pull request

Labels ⚙

None yet

Milestone ⚙

No milestone

Assignees ⚙

No one—assign yourself

Figure 41: A blank preview view of the Mantid Project GitHub pull request template

The I(Q, t) Fit interface should now properly update the plot output options when switching between Fit types

To test:

- Open `I(Q, t) Fit` (Interfaces > Indirect > Data Analysis > `I(Q, t) Fit`)
- Change the `Plot output` to `All`
- Change the `Fit Type` to `2 Exponential`
- Ensure the Plot Output is still `All`
- Change to Fit type to `1 stretched exponential`
- Change Plot Output to `Beta`
- Change Fit type to `1 exponential`
- Ensure the Plot Output is now `None`
 - `Beta` does not exist as a plot output for `1 exponential` hence defaults to `None`

Fixes #16385

[RELEASE NOTES](#)

Figure 42: An example of a completed Mantid Project GitHub pull request

11 Future Planning

11.1 Vesuvio

Whilst the main workflow of the VESUVIO data reduction and analysis has been implemented into MANTID, some additional functionality has been requested from the VESUVIO users and instrument scientists.

11.1.1 LoadVesuvio

Given the operations performed and the validation in place for *LoadVesuvio* (see section 6.2) it should be possible to have mixed period loading (for example, loading 6 period and 3 period data at the same time providing the spectra, or banks, are in back scattering).

It should be possible use *LoadVesuvio* without using a difference mode. Instead, all periods in the data would be summed together and normalised by the monitors.

LoadVesuvio operations are clearly partitioned by differencing mode. As such, refactoring the differencing modes into separate scripts would make the algorithm far more comprehensible. The suggestion would be to have multiple scripts in a *LoadVesuvio* directory alongside the algorithm that handle the implementation of the differencing modes. It is possible that actions such as data loading, normalisation and validation can remain in the top level of the algorithm as these will be common across all difference modes, see **Figure 43**.

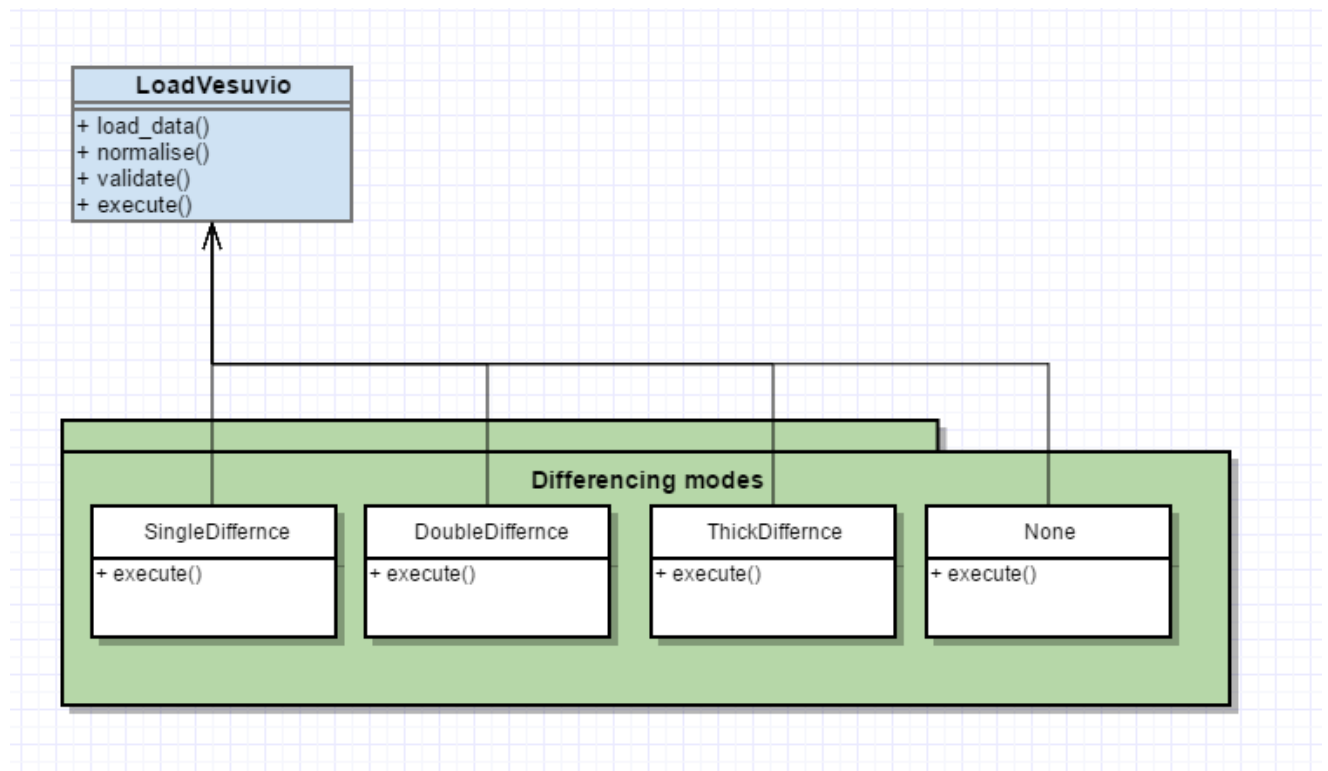


Figure 43: Possible refactoring method for LoadVesuvio.
created using tool provided by www.glify.com

11.1.2 Vesuvio Commands

The current script containing the main workflow for VESUVIO (`commands.py`), see section 6.1.2 for more details, has a long execution time. This is for the most part due to the calculations of multiple scattering contributions which needs to be done for every fit. During the execution of the workflow, spectra, or banks depending on the mode of operation, are analysed sequentially in a loop. As the analysis of a single spectra, or bank, is independent of another, this process could be parallelised to allow multiple spectra to be processed simultaneously. This should result in a noticeable increase in run time for the algorithm.

11.1.3 Documentation and testing

The VESUVIO documentation is of varying quality throughout. Whilst most documentation pages contain usage examples, very few contain detailed descriptions of how the algorithm functions and currently there are no workflow diagrams for algorithms. In order to actually be useful to both developers and users, these should be updated to a good and consistent standard throughout.

Almost all the functionality in VESUVIO is being tested although there is currently no testing for the 2 and 3 period case of *LoadVesuvio* due to not having any data at present. This data would also be required in order to allow for testing of the mixed period case example discussed in section 11.1.1.

11.2 Data Analysis

In order to obtain a quick overview of a data set an algorithm that takes a set of run numbers and reduces them (if required) then runs performs the same process as the Elwin interface could be beneficial. This algorithm should be capable of handling data files from any instrument as well as files that consist of multiple runs. The algorithm should also handle an input consisting of a mixture of NeXUS and/or raw files. The simple workflow for this algorithm is shown in **Figure 44**.

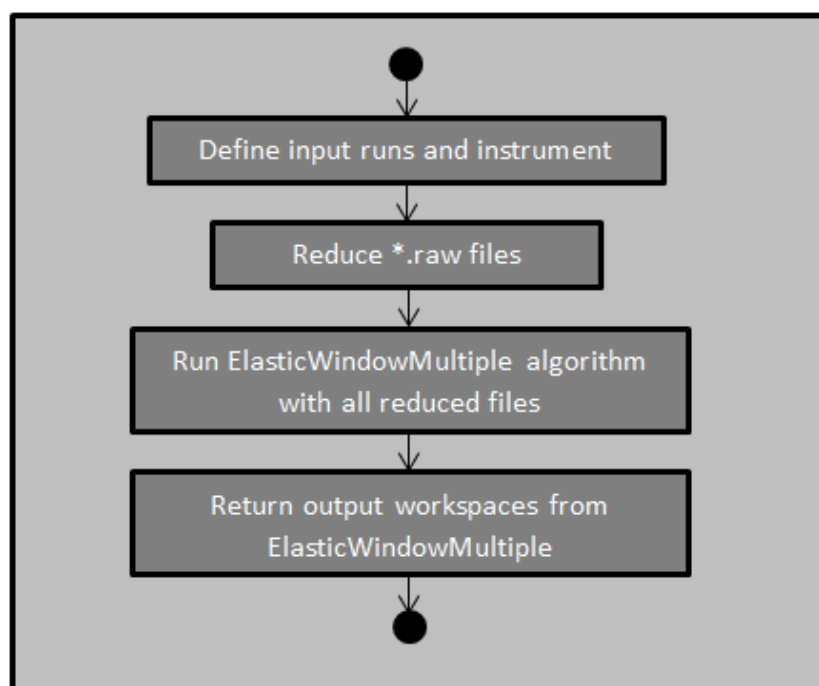


Figure 44: Basic workflow for IndirectQuickRun algorithm

11.3 Bayesian Analysis

Bayesian analysis in MANTID is currently performed by FORTRAN routines that have been converted, using the F2Py library to Python and is currently only distributed on windows. The main algorithm that runs Bayesian analysis, *BayesQuasi*, has the advantage of outputting the relative probabilities of the fit as well as not requiring any fit parameters initially. However, due to the age and structure of this code, as well as the lack of documentation, it is becoming increasingly difficult to maintain and is acting as a black box for Bayesian analysis in MANTID.

A new alternative has been proposed in order to replace the existing FORTRAN code using a fitting process similar to *ConvolutionFitSequential* in order to obtain similar results to *BayesQuasi*. However, a method was required to determine the start parameters automatically. The current trial implementation uses the moments of the input data to determine the input parameters where background is taken from the data, the intensity is the 0th moment and width is the $\sqrt{2}$ nd moment.

This case works for single peak fitting however in two peak fitting; the parameters for the second peak is calculated by taking the 0th and 2nd moments of the fitted difference (Calculation – data) of the first peak. In order to guard against poor fits, several constraints are also used:

- Background > 0
- Peak centres are tied
- Peak amplitude and FWHM > 0

The first iteration of this algorithm has been created by W.S. Howells and is currently being tested. Once feedback has been obtained, this algorithm can be implemented in MANTID accompanied by automatic testing.

11.4 FABADA

The FABADA minimizer is a very powerful tool for Bayesian analysis within MANTID. However, it does still require some improvements in order to bring it up to standard with the other minimizers in MANTID.

Currently FABADA is unable to handle using ties in the function. Whilst this does not cause crashing, it does mean that it is not possible to use the minimiser to full affect if you are already aware of parameters that are related.

By nature, FABADA is very sensitive to initial parameters which can lead to a poor chance of convergence. FABADA has shown some instability in previous releases which appears to be due to convergence. When the minimiser is set to have a large tolerance for convergence FABADA will normally perform a reasonable fit but, due to the easily met convergence criteria, it can also get stuck at a local minima producing a poor fit. By contrast setting the convergence criteria too low will normally result in a crash as the minimizer is unable to converge within the number of steps.

Both the issues above are currently being investigated by B. Llopis Vidal during his summer internship with the Molecular Spectroscopy group at ISIS, Rutherford Appleton Laboratory, UK.

Once FABADA is fully operational, work can continue on implementing it into the Indirect Interfaces. The next interface that should use FABADA is ResNorm.

11.5 Simulations

11.5.1 aCLIMAX

aCLIMAX is a software package designed to interpret Inelastic Neutron Scattering from, ab initio, TOSCA-like spectrometer data. Originally an unsupported software package in Microsoft Excel [16], by version 4, the package was converted to the Visual Basic programming language and a GUI was also developed [17]. aCLIMAX is not open source therefore it can only be treated as a black box. Furthermore, in its current state it is not optimised for the indirect instruments housed as ISIS.

In order to maintain support for aCLIMAX in the future, a new python version of the aCLIMAX code, current named ABINS is being developed by K. Dymkowski at the Rutherford Appleton Laboratory. ABINS, when complete, will be distributed with MANTID and therefore open source. Furthermore, this will allow automated testing of the algorithms.

Bibliography

- [1] "About Us," Mantid Project, [Online]. Available: <http://www.mantidproject.org/MantidProject>About>. [Accessed 06 2016].
- [2] "QtiPlot - Data Analysis and Scientific Visualisation," QtiPlot, [Online]. Available: <http://www.qtiplot.com/>. [Accessed 06 2016].
- [3] 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.
- [4] W. Howells, "IDA - IRIS Data Analysis.," Technical Report RAL-TE-96-006, Rutherford Appleton Laboratory, 1996.
- [5] "Scientific Steering Committee 2016," Mantid Project, [Online]. Available: http://www.mantidproject.org/Scientific_Steering_Committee_2016. [Accessed 06 2016].
- [6] S. Mukhopadhyay, "How to use Mantid for low energy inelastic neutron scattering data analysis on indirect geometry instruments.," Technical Report RAL-TR-2014-005, Rutherford Appleton Laboratory, 2014.
- [7] D. Nixon, "Developments in MANTID relating to indirect inelastic spectroscopy between July 2014 - July 2015.," Technical Report RAL-TR-2015-007, Rutherford Appleton Laboratory, 2015.
- [8] H. H. P. a. C. J. Pings, "Numerical Evaluation of XRay Absorption Factors for Cylindrical Samples and Annular Sample Cells," *Journal of Applied Physics*, 1962.
- [9] D. Sivia, C. Carlile, W. Howells and S. König, "Bayesian analysis of quasielastic neutron scattering data.," *Physica B: Condensed Matter*, 182(4):341-348, 1992.
- [10] D. M. López, "Implementation of a Bayesian algorithm into Mantid for the analysis of neutron scattering data to reveal molecular movements," UNIVERSITAT POLITÈCNICA DE CATALUNYA, 2016.
- [11] J. Mayers and G. Reiter, "The VESUVIO electron volt neutron spectrometer," *Measurement Science and Technology*, 23(4):045902, 2012.
- [12] J. Mayers, A. Fielding and R. Senesi, "Nuclear Instruments and Methods in Physics Research," Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 2002.
- [13] J. Mayers, "User guide to VESUVIO data analysis.," Technical Report RAL-TR-2011-003, Rutherford, 2010.
- [14] E. Gansner and E. K. a. S. North, "Drawing graphs with dot," 2006. [Online]. Available: www.graphviz.org. [Accessed 06 2016].

- [15] "PEP Standards," python, [Online]. Available: <https://www.python.org/dev/peps/pep-0008/> . [Accessed 06 2016].
- [16] D. Champion and J. T. a. G. Kearley, "a-CLIMAX: a new INS analysis tool," *Appl. Phys. A* 74, S1302–S1304, 2002.
- [17] A. Ramirez-Cuesta, "aCLIMAX 4.0.1, The new version of the software for analyzing and interpreting INS spectra," *Computer Physics Communications* 157, 2004.