Science & Technology Facilities Council

Technical Report
RAL-TR-2015-007

Developments in MANTID relating to indirect inelastic spectroscopy between July 2014 - July 2015

D Nixon

August 2015

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

Developments in MANTID relating to indirect inelastic spectroscopy between July 2014 - July 2015

Daniel Nixon

August 17, 2015

Abstract

This document aims to describe the recent changes to Manipulation and Analysis Toolkit for Instrument Data (MANTID) relating to the analysis and reduction of data from indirect geometry spectrometers within the MANTID, a cross platform data reduction and analysis framework used by multiple neutron facilities for a range of experimental techniques.

These changes include those carried out between July 2014 and July 2015 and cover MANTID versions 3.3, 3.4 and 3.5.

The majority of the work carried out in this period is attributed to the improvement of the reliability and robustness of the existing data reduction and analysis routines, the extension of the current QENS analysis routines, the extension of the current support for loading and working with simulation data and the implementation of the data analysis workflow for VESUVIO.

Contents

Acknowledgements

The author would like to thank S. Mukhopadhyay and W.S. Howells for their support in writing this report and their contributions to the developments in QENS reduction and analysis within MANTID as well as M. Krzystyniak for his assistance with understanding the theory behind the analysis of VESUVIO data and F. Fernandez-Alonso for his support throughout the year.

1 Introduction

MANTID [9] is a cross platform tool designed to integrate the data reduction and analysis for a range of neutron instruments and techniques into a single software package. Currently it supports instruments at the ISIS, SNS, HFIR and ILL neutron sources.

The current set of reduction and analysis routines within MANTID for indirect geometry spectroscopy was largely based on the existing MODES [6] and IRIS Data Analysis [5] packages as well as the VESUVIO analysis programs [10] that previously used to reduce and analyse data.

This document will give an overview of the current state of the reduction and analysis routines specific to indirect inelastic instruments with emphasis on new features and improvements over the past year of development, which will cover MANTID releases 3.3 to 3.5.

The data reduction and analysis routines at the start of this period were for the majority in the form of Python scripts created to match the processing carried out by MODES in the case of IRIS and OSIRIS. Further details of the data reduction and analysis workflow for QENS data can be found in the MANTID QENS manual [14].

At the start of this period there was little support for VESUVIO within MANTID in terms of data analysis workflow, however the fitting functions that are used in the workflow scripts today had been implemented (see section 10).

There has previously been little effort to include simulation data into the analysis workflows within MANTID, hence as of release 3.2 there was little support for this, with the exception of basic loaders for Sassena [8] and nMOLDYN [15] as well as an algorithm (*DensityOfStates*) to calculate the vibrational density of states from a CASTEP [3] simulation.

A brief overview of further work that is required will be given towards the end of the document, this contains tasks based on both technical and scientific requirements and should not be seen as a fix set of work to be carried out.

2 Data Reduction

This section describes the data reduction interfaces and algorithms within MANTID, this section relates to the functionality found on the *Indirect Data Reduction* (Interfaces ⇒Indirect ⇒Data Reduction) custom interface.

2.1 Conversion to Energy Transfer (ISIS)

The energy transfer reduction routine has now been moved to a MANTID algorithm and replaced the use of reducer classes and reduction steps that were previously used (this is also the case with the diffraction reduction routines 4).

Support for energy transfer reductions for data from the legacy versions of TOSCA was added in release 3.3, this added a new instrument definition for TOSCA which covered TOSCA-1 which was in operation from May 1998 to March 2000 and two definition files for two different iterations of TFXA running from 1985 to October 1993 for the first version then up to May 1998 for the second version.

In the case of TFXA the instrument is still not fully defined within MANTID as for each detector only the L_2 distance is known, however this is sufficient for an energy transfer reduction which is the primary requirement for TFXA data reduction within MANTID.

There has yet to be any work carried out in supporting the simulations reduction/analysis of neutron and Ramen spectroscopy on TOSCA.

The majority of what were reduction steps in the previous reduction framework have been split into Python functions that can be called by any reducer and are kept in the `IndirectReductionCommon.py` Python file. The functions in this file are denoted by a dark blue square in figures 1 and 11.

The reduction workflow is as follows:

Data Loading (`load_files`)
: This step loads the raw sample data, the instrument parameters from the relevant instrument parameter file for the instrument configuration (each analyser and reflection configuration has a separate parameter file) and optionally the sample logs.

  The loading is performed either by *LoadRaw* in most cases and *LoadVesuvio* in the case where the instrument is VESUVIO.

  Depending on the value of the `Workflow.ChopDataIfGreaterThan` parameter in the instrument parameter file the data may also be chopped into multiple frames using the *ChopData* algorithm.

  The raw data may be summed into a single workspace if requested by the user.

Identify Faulty Detectors (`identify_bad_detectors`)
: This step searches for any detectors in the raw data that are deemed to have a significant portion of the signal being noise have failed (providing zero counts).

  This is performed using the *IdentifyNoisyDetectors* algorithm.

Process Monitors (`unwrap_monitor`, `process_monitor_efficiency` and `scale_monitor`)
: This step performs manipulation of the monitor spectrum before it is used later to correct the sample data.

  If required the *UnwrapMonitor* algorithm is used to ensure the monitor workspace has common binning within the maximum theoretical wavelength range.

  The monitor spectrum is then corrected for its thickness, attenuation and scaling factor defined in the instrument parameter file.

Background Removal
: This step optionally calculates and removes a flat background from the sample spectra given a range in time of flight which contains the background signal.

  This is performed using the *CalculateFlatBackground* algorithm.

Apply Calibration
This step divides the sample data by the calibration workspace to correct detectors for changes in relative intensity.

This is performed using the *Divide* algorithm.

Correct by Monitor (`scale_detectors`)
This step divides the sample spectra by that of the monitor to perform normalisation of the data by the monitor intensity.

Convert to Energy Transfer
This step converts the data in time of flight to energy transfer in meV using the *ConvertUnits* algorithm and multiplies by K_i/K_f using the *CorrectKiKf* algorithm.

Rebin (`rebin_reduction`)
This step rebins the reduced data based on the bin parameters provided by the user, this is performed using the *Rebin* algorithm.

Detailed Balance
This step applies an optional exponential correction using the *ExponentialCorrection* algorithm with $C1 = \frac{11.606}{2 \times T}$, where T is the temperature in Kelvin provided by the user.

Scaling
This step optionally scales the sample data by a factor provided by the user using the *Scale* algorithm.

Group Detectors (`group_spectra`)
This step groups the spectra in the workspace based on the grouping policy defined either in the instrument parameter file or by the user if a fixed number of groups were requested.

The grouping is performed using the *GroupDetectors* algorithm.

If any faulty detectors were identified they are masked here using the *MaskDetectors* algorithm.

Fold Chopped Data (`fold_chopped`)
This step converts multiple framed data back to a single workspace if it was chopped during the data loading step, this is done using the *MergeRuns* algorithm and performing a scale based on the average intensity of each frame.

Convert X units
This step converts the X axis of the reduced workspaces to a unit other then energy transfer in meV, typically this is only used to convert to Cm^{-1} which is preferred for instruments such as TOSCA and TFXA.

Rename Workspaces (`rename_reduction`)
This step renames the reduced workspaces as per the naming format define in the instrument parameter file, typically this is in the format `[INST][RUN]_[ANALYSER][REFLECTION]_red` for IRIS and OSIRIS (e.g. `irs26176_graphite002_red`) and `[INST]_[NAME]_red` for TOSCA and TFXA, where `NAME` is the run name.

Save (`save_reduction`)
This step saves the reduced workspaces in the users default save directory in any combination of the following formats, the algorithm used to save the data is given in brackets:

- NeXus (*SaveNexusProcessed*)
- SPE (*SaveSPE*)
- NXSPE (*SaveNXSPE*)
- ASCII (*SaveAscii* v1)
- aClimax (*SaveAscii*)
- DaveGrp (*SaveDaveGrp*)

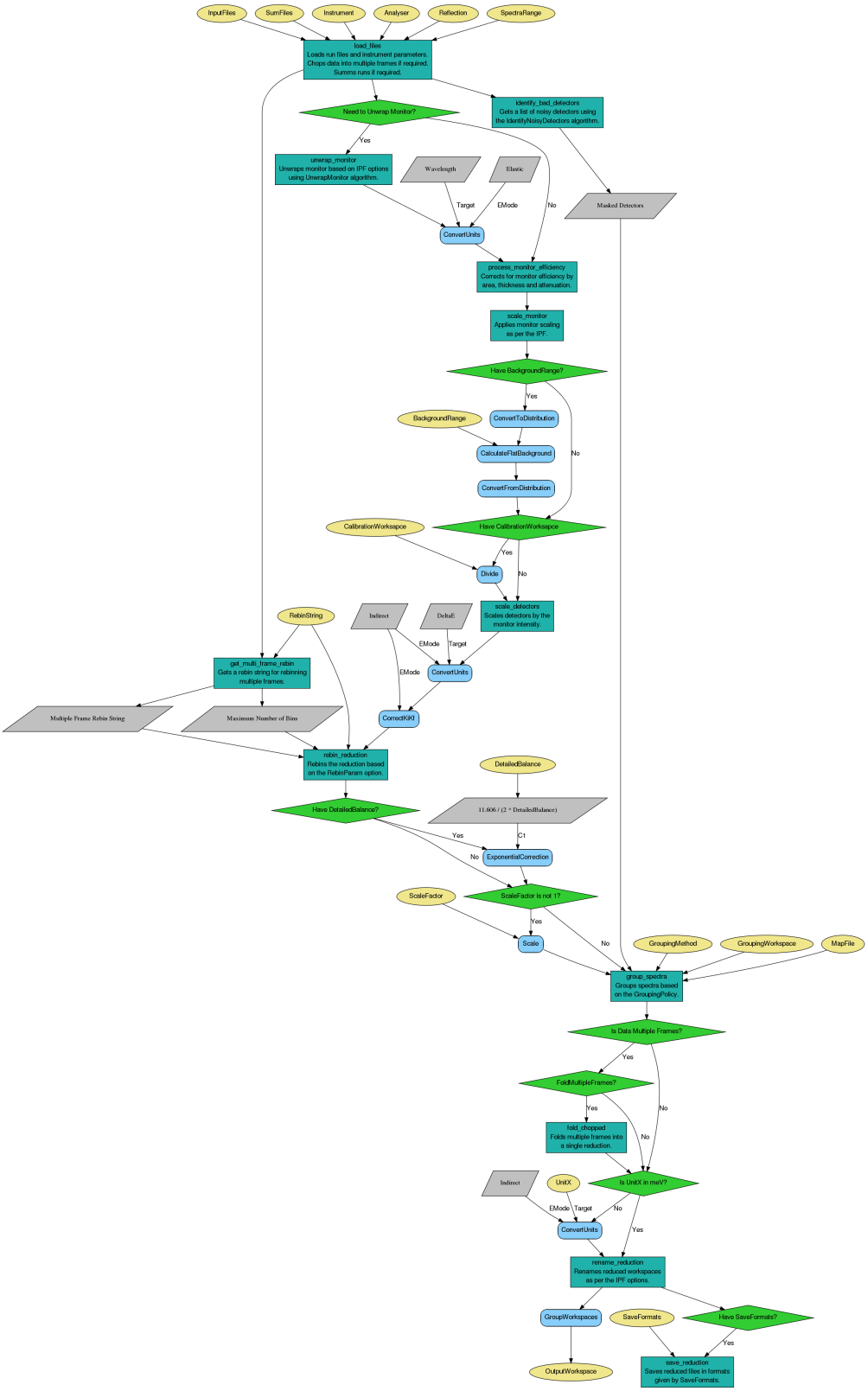


Figure 1: ISISIndirectEnergyTransfer flowchart

A full size version of this diagram is available online at `http://docs.mantidproject.org/algorithms/ISISIndirectEnergyTransfer-v1.html`

2.2 Calibration and Resolution (ISIS)

Calibration of the QENS spectrometers at ISIS is performed using the *IndirectCalibration* algorithm which takes a series of raw files and creates a workspace containing a single value for each detector corresponding to the intensity of the integrated peak region. The full procedure carried out by the algorithm is described in the flowchart figure 2.

The procedure for creating the calibration workspace is for a vanadium sample to first calculate a flat background using the background range (in time of flight) provided by the user then performing an integration over the peak range provided by the user.

If multiple raw files were provided then they are first merged to a single workspace using the *MergeRuns* algorithm before any calculation is done, the intensity of the result is then corrected by multiplying the entire workspace by $1/(I_{total}/(N_{hist}/N_{bad\_dets}))$, where I_{total} is the summed intensity of all spectra in the calibration workspace, N_{hist} is the number of histograms in the calibration workspace and N_{bad_dets} is the number of detectors that are deemed to not be functioning correctly.

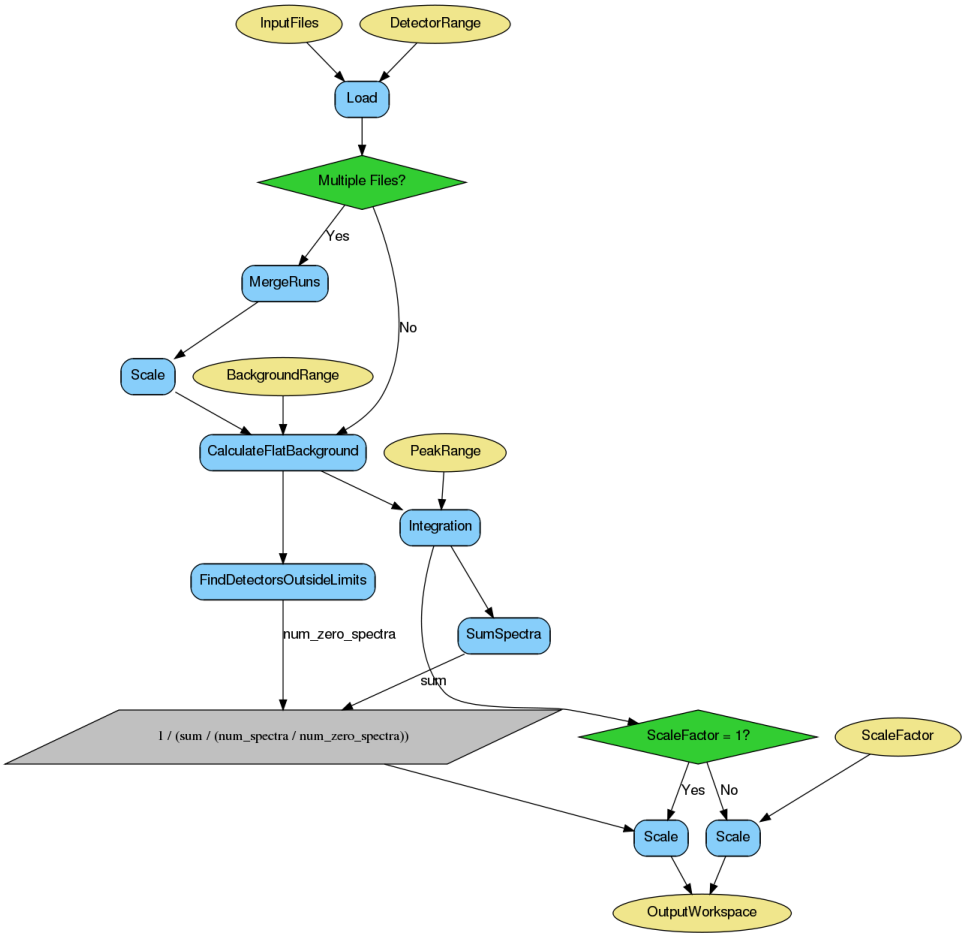


Figure 2: IndirectCalibration flowchart

A full size version of this diagram is available online at `http://docs.mantidproject.org/algorithms/IndirectResolution-v1.html`

The energy resolution of the QENS spectrometers at ISIS is calculated using the *IndirectResolution* algorithm, the exact processing of which is described in the flowchart figure 3.

Firstly an energy transfer reduction of a vanadium sample is carried out with the detector grouping set to group all detectors into a single spectrum, from which a flat background is removed based on a background range in energy provided by the user, finally the resolution spectrum is rebinned using parameters provided by the user.

Note that in the user interface the rebin parameters are provided automatically based on the instrument resolution defined in the instrument parameter file.

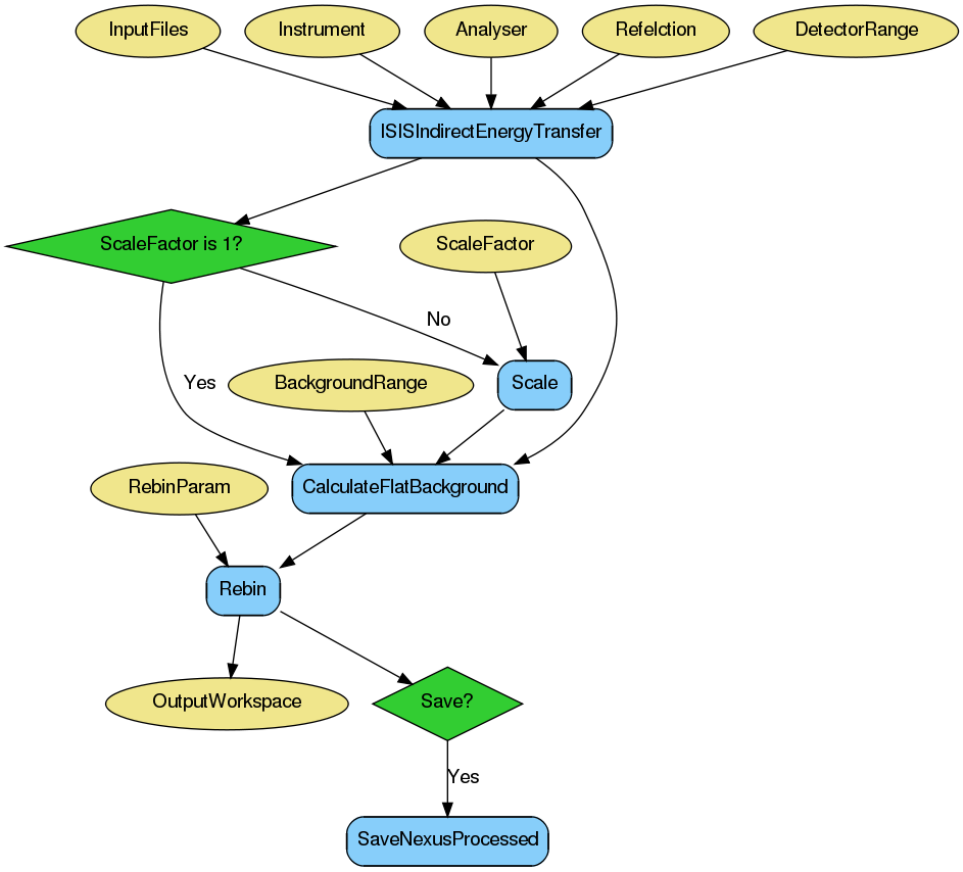


Figure 3: IndirectResolution flowchart

A full size version of this diagram is available online at `http://docs.mantidproject.org/algorithms/IndirectResolution-v1.html`

2.3 Sample Transmission

The Transmission tab allows calculation of the sample transmission as a function of wavelength, this uses the *IndirectTransmissionMonitor* algorithm, the processing of which is described in the flowchart figure 4.

This routine no longer depends on the fact that the two monitors are always the first two spectra (or that the instrument has two monitors) by obtaining the spectra numbers of the monitors from the instrument parameter file, this has extended instrument support from this algorithm to include TOSCA and VESUVIO.

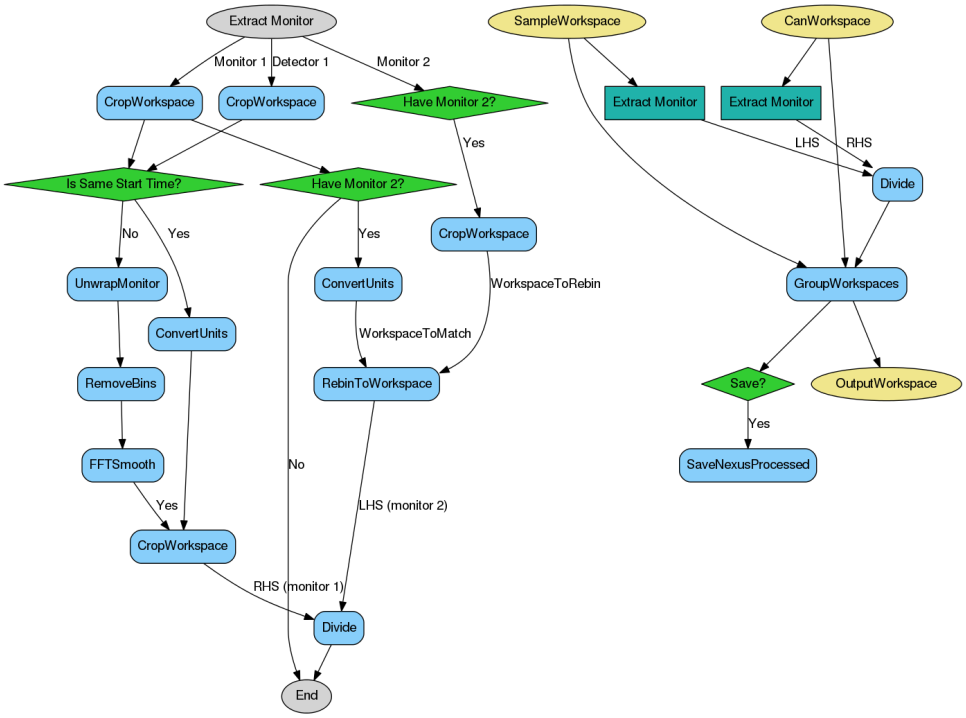


Figure 4: IndirectTransmissionMonitor flowchart

A full size version of this diagram is available online at `http://docs.mantidproject.org/algorithms/IndirectTransmissionMonitor-v1.html`

2.4 Peak Symmetrise

The *Symmetrise* algorithm provides a method of performing a partial reflection of a curve about the Y axis given a range to be reflected.

This would typically be used to correct an asymmetrical peak caused by physical issues with instrument setup.

An example of the usage of the *Symmetrise* algorithm is shown in figure 5, in this case the peak was symmetrised between 0 and 0.02 meV.

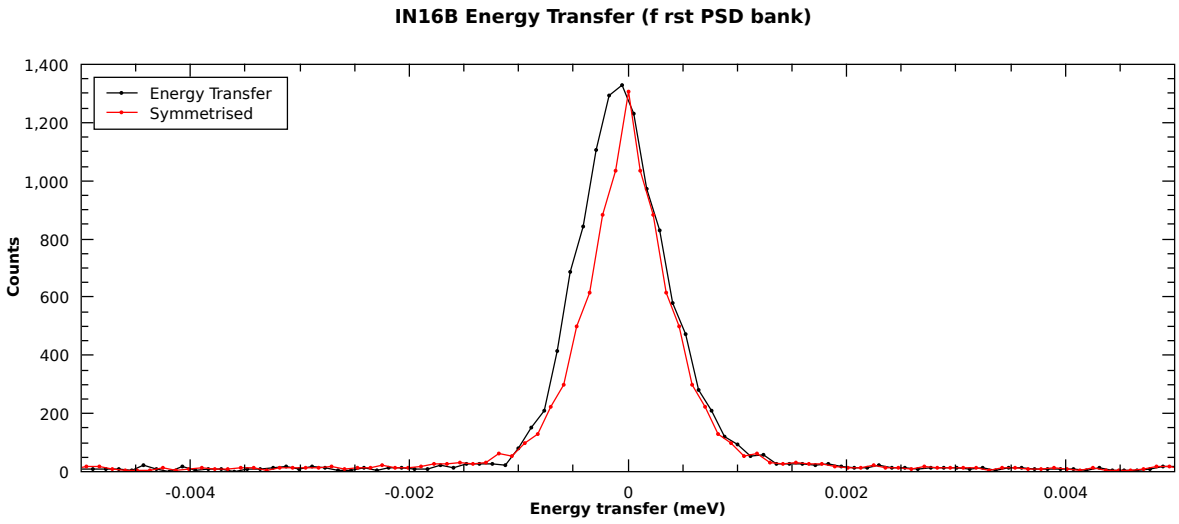


Figure 5: Running *Symmetrise* on an energy transfer reduction

2.5 S(Q, w)

The S(Q, w) tab provides a simplified interface to the collection of algorithms withing MANTID for conversion to $S(Q, \omega)$.

MANTID now has a single algorithm (*SofQW*) that can be used to call any of the existing $S(Q, \omega)$ algorithms given by the value of the Method property, this is now the algorithm that is used within the tab.

The names of the rebin types have now been changed to match the values that can be given to the Method property, the names have changes as follows, the name of the algorithm called by the *SofQW* algorithm is given in brackets:

- Parallelpiped ⇒Polygon (*SofQWPolygon*)
- Parallelpiped / Fractional Area ⇒NormalisedPolygon (*SofQWNormalisedPolygon*)

The option to use centre point rebinning has been removed.
A comparison of the two binning methods used withing the interface is given in figure 6.

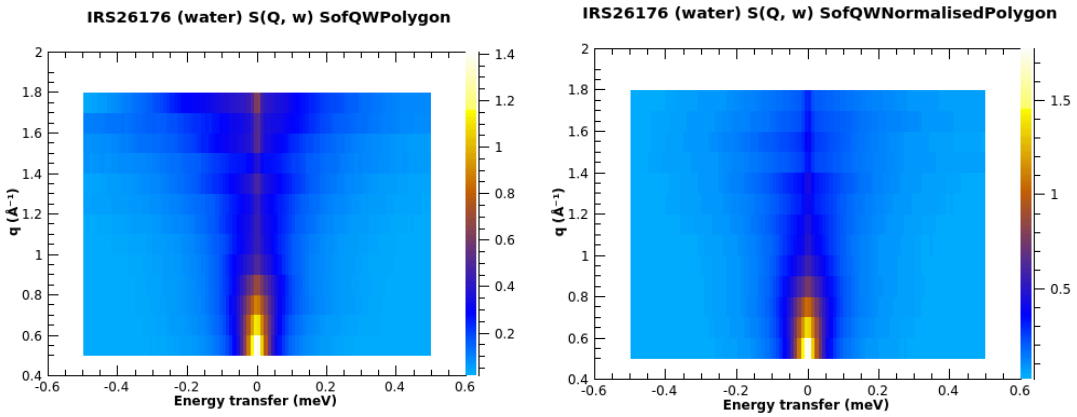


Figure 6: Comparison of rebinning methods used on *SofQW** algorithms

2.6 S(Q, w) Moments

This interface provides a simple means of calculating the first 4 moments of an $S(Q, \omega)$, currently this is implemented in a workflow algorithm (SofQWMoments) which calculates the moments using integration as described in flowchart figure 7, however there is scope to replace this large section of code with use of the statistics module within MANTID.

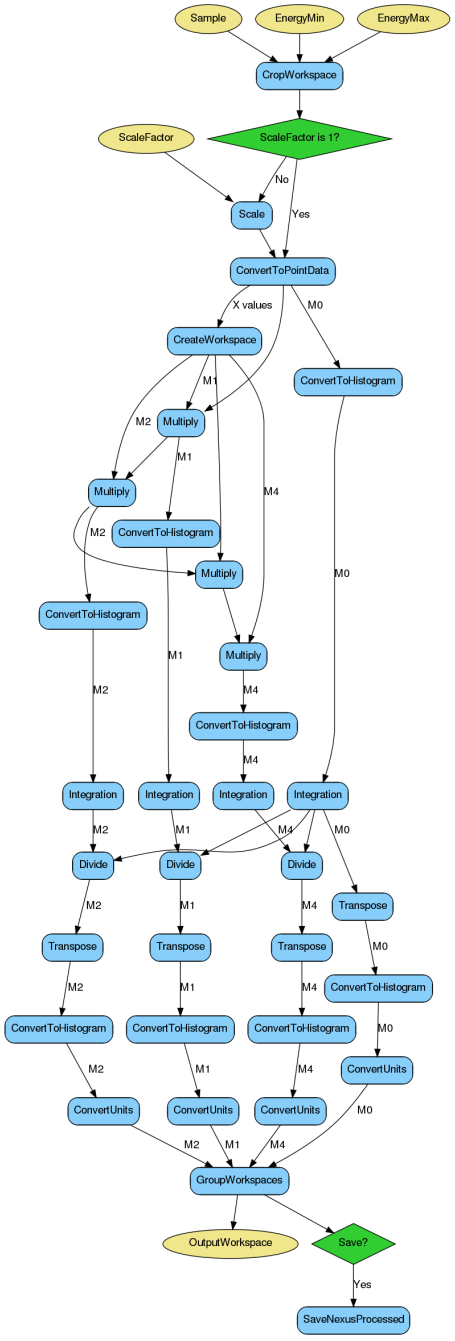


Figure 7: SofQWMoments flowchart

A full size version of this diagram is available online at `http://docs.mantidproject.org/algorithms/SofQWMoments-v1.html`

3 Multiple facility support

A significant amount of work has gone into supporting multiple facilities in the data reduction and analysis workflows and interfaces used with ISIS instruments, primarily this has involved supporting the BASIS instrument at the SNS in the QENS analysis routines and the IN16B instrument at the ILL in the Indirect Data Reduction interface.

Initial work was carried out to enable the Indirect Data Reduction interface to display different tabs based on the current facility, this would allow multiple tabs performing the same task to be created and only the relevant ones shown.

Energy transfer reductions for the IN16B spectrometer were already supported using the *IndirectILLReduction* algorithm which is used on the ILL Energy Transfer tab of Indirect Data Reduction (see figure 8).

An additional algorithm, *ILLIN16BCalibration*, was added which generated a calibration workspace for IN16B data by integrating the elastic peak for each detector group (much the same as the *IndirectCalibration* algorithm used at ISIS but working in energy rather than time of flight).

This new algorithm was then used on the ILL Calibration tab of Indirect Data Reduction to provide full support for IN16B in the data reduction workflow.

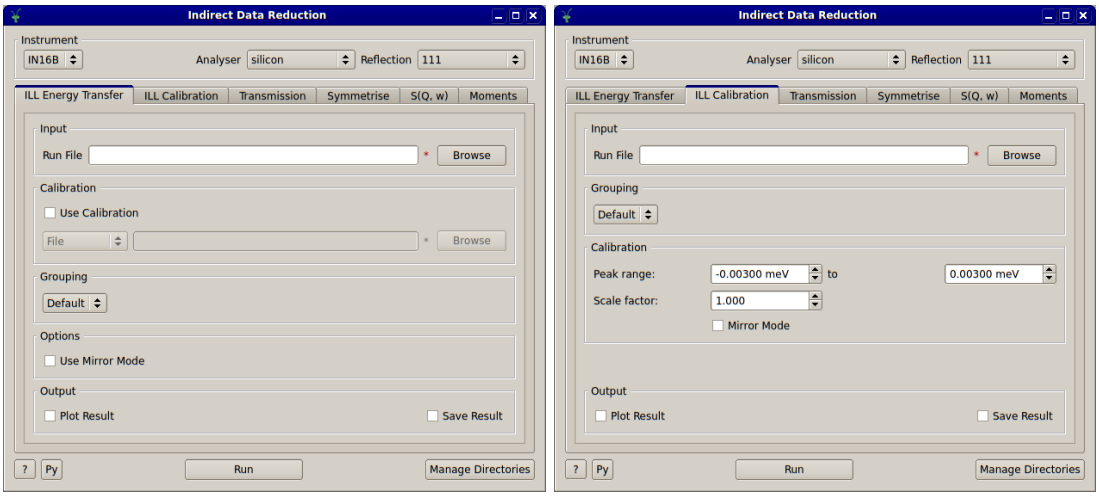


Figure 8: Support for IN16B at the ILL in Indirect Data Reduction

Although the task of supporting BASIS in the existing analysis routines was at the time specific to BASIS, it helped to both refine the routines such that they can be used with a wider range of instruments and to define the requirements for an instrument to be supported by the routines. For example BASIS often ran into issues as it did not have its detectors attached to an instrument component representing the crystal analyser which is assumed given that is what is done with ISIS instrument.

3.1 Conversion to Energy Transfer (ILL IN16B)

The theory behind the energy transfer reduction routine for IN16B at the ILL is mostly similar to that for ISIS given that there are many similarities between this instrument and IRIS (see section 2.1), however there is added support for the mirror mode on IN16B in which the acceleration and deceleration phases of the Doppler drive are recorded individually.

In this case the spectra from each phase are simply overlaid by cropping the sample workspace to obtain a workspace for the each phase, converting each phase to energy transfer separately and summing each phase.

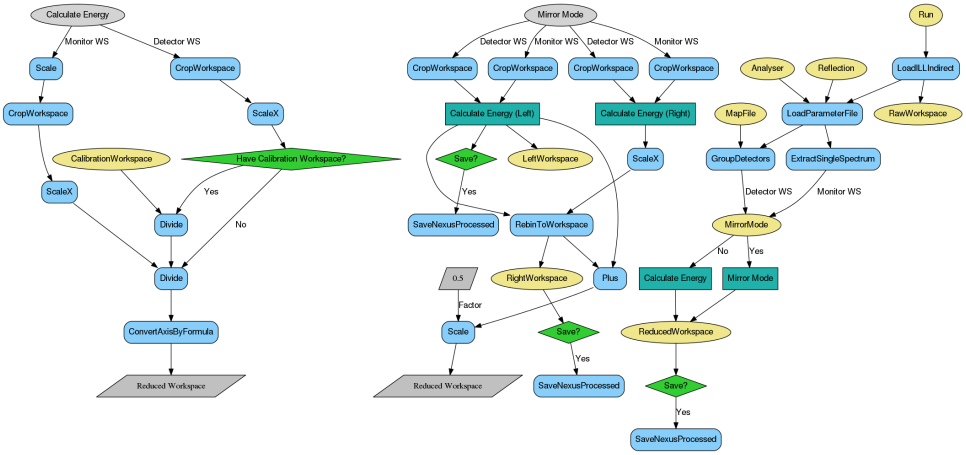


Figure 9: IndirectILLReduction flowchart

A full size version of this diagram is available online at `http://docs.mantidproject.org/algorithms/IndirectILLReduction-v1.html`

3.2 Calibration (ILL IN16B)

The calibration algorithm for the IN16B spectrometer at the ILL follows the same method as the *Indirect-Calibration* algorithm does for QENS spectrometers at ISIS (section 2.2), however the notable difference is that the algorithm for IN16B is in energy rather than time of flight as per the *IndirectCalibration* algorithm.

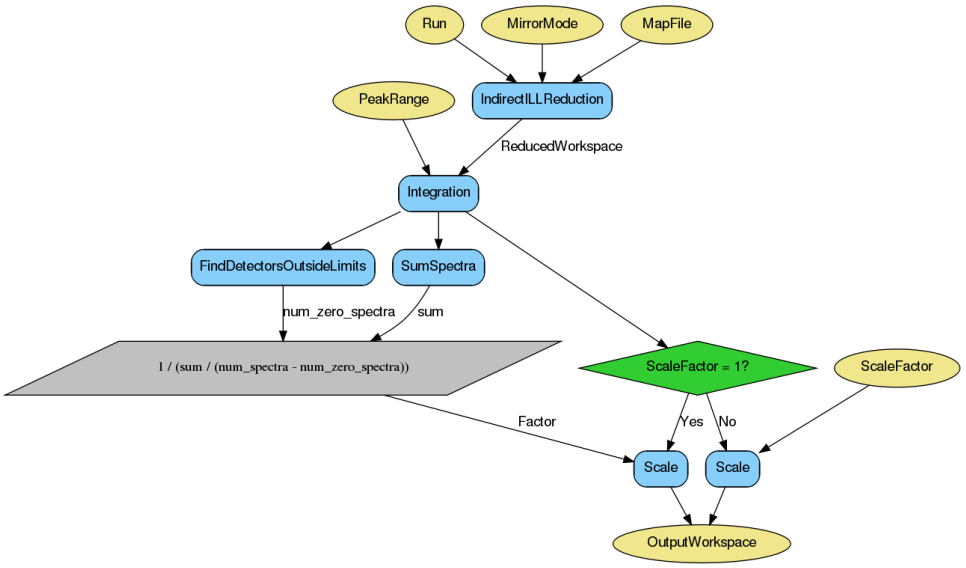


Figure 10: ILLIN16BCalibration flowchart

A full size version of this diagram is available online at `http://docs.mantidproject.org/algorithms/ILLIN16BCalibration-v1.html`

4 Diffraction Reduction

This section describes the algorithms used for diffraction reductions on the ISIS indirect inelastic instruments with diffraction banks, the functionality covered is available via the *Indirect Diffraction* (Indirect ⇒Diffraction) custom interface.

4.1 General purpose reduction

Data reduction for diffraction is performed using the *ISISIndirectDiffractionReduction* algorithm in most cases (the one exception being for OSIRIS reductions in diffonly mode). A workflow diagram for this algorithm is shown in figure 11.

This diagram shows the complete steps taken by the algorithm, however the basic concept of this algorithm can be broken down into three key steps:

Scale by monitor
Scales the detector intensity by that of the monitors.

Convert to dSpacing
Conversion from time of flight to dSpacing using the *ConvertUnits* algorithm as per equations 1 and 2.

$$\lambda = \frac{h \times t}{M_N \times L} \tag{1}$$

where t is time of flight, L is the total flight path, M_N is the neutron mass and h is the Planck constant.

$$d = \frac{\lambda}{2sin(\theta)} \tag{2}$$

Group detectors
Groups all detector spectra into a single spectra using the *GroupDetectors* algorithm, excluding any detectors that have significant variation from the intensity of other detectors.

There is also the option to not perform this grouping step, this was added mainly as a diagnostic tool for detector positions on VESUVIO.

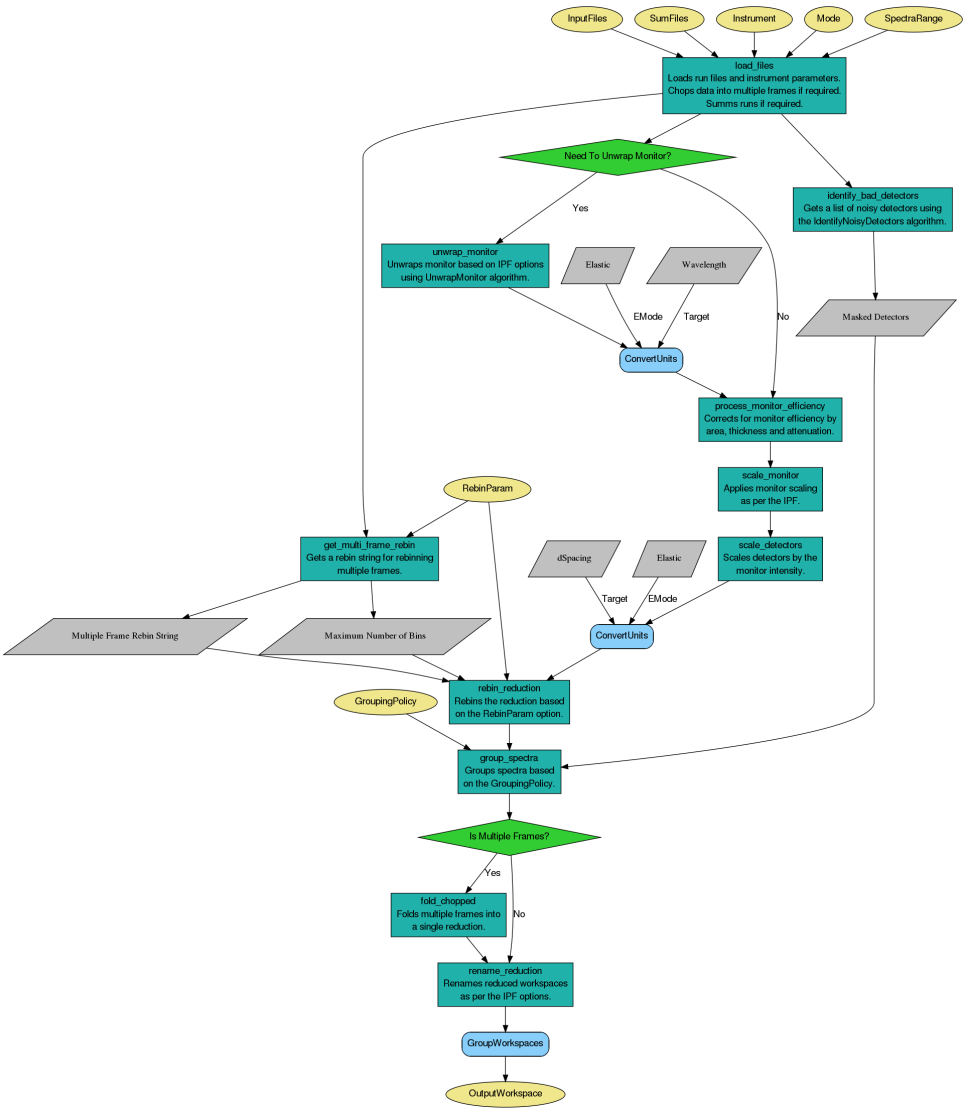


Figure 11: ISISIndirectDiffractionReduction flowchart

A full size version of this diagram is available online at `http://docs.mantidproject.org/algorithms/ISISIndirectDiffractionReduction-v1.html`

4.2 OSIRIS diffonly

In the case of OSIRIS reductions in diffonly mode the *OSIRISDiffractionReduction* algorithm is used, in this case instead of the monitor intensity the detector intensities are corrected by a vanadium sample.

This algorithm can also allow stitching of up to five individual runs in order to increase the measured dRange, a comparison of this with the *ISISIndirectDiffractionReduction* algorithm can be seen in figures 12 and 13.

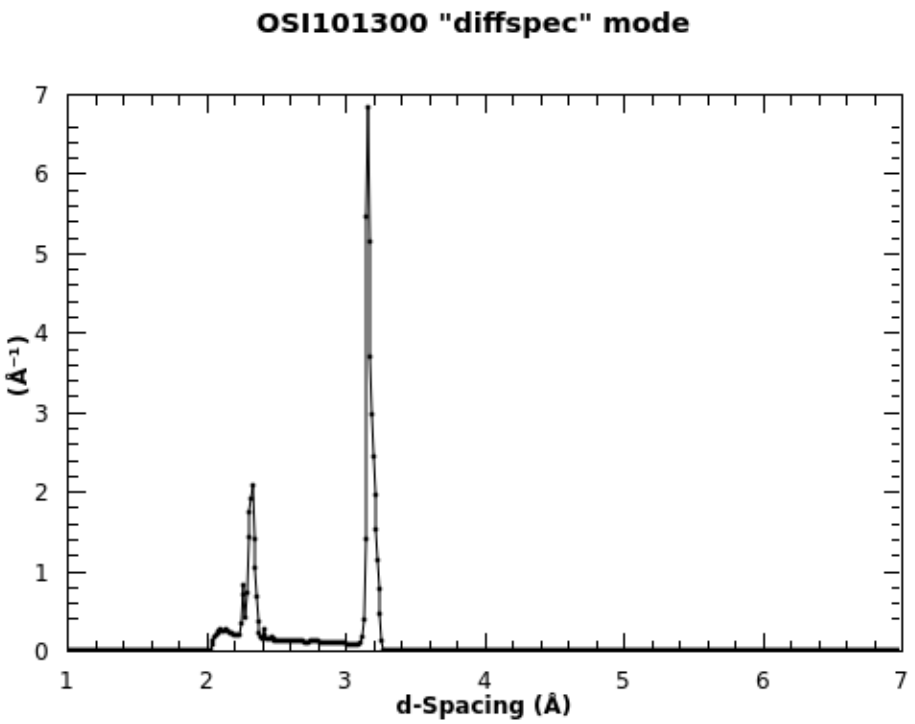


Figure 12: OSIRIS diffraction reduction in "diffspec" mode

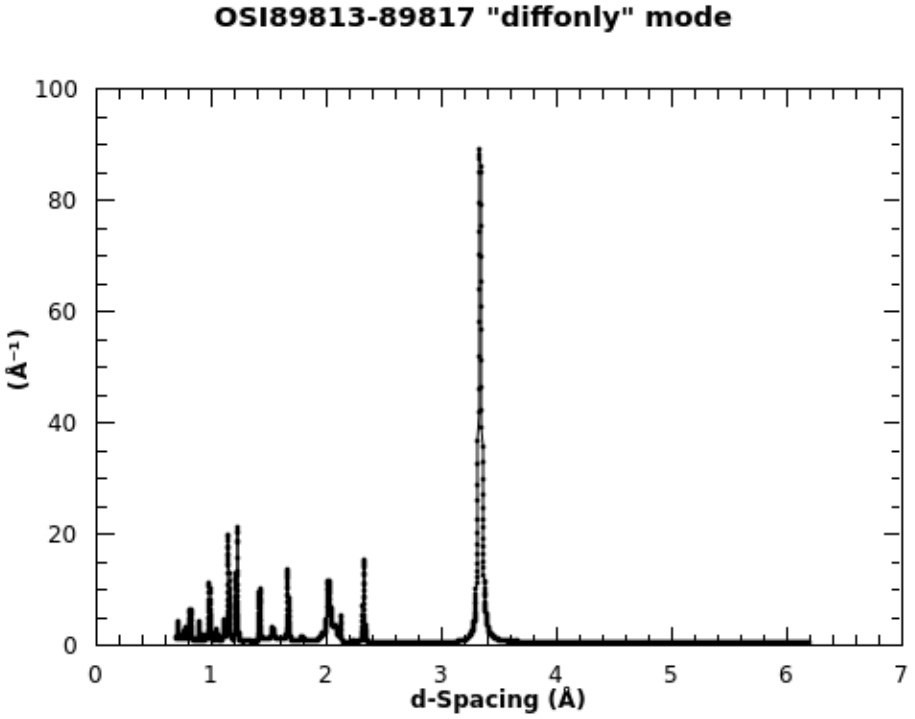


Figure 13: OSIRIS diffraction reduction in "diffonly" mode

5 Data Analysis

The *Indirect Data Analysis* (Interfaces ⇒Indirect ⇒Data Analysis) interface contains analysis routines that operate over either the energy transfer reductions (workspaces ending in `_red`) or $S(Q, \omega)$ workspaces (ending in `_sqw`). The majority of the routines can also be used with diffraction reductions created with the *Indirect Diffraction Reduction* interface and associated algorithms.

There are two main categories of analysis that are performed on the interface; transformations and fitting. Transformations include *Elwin* and *I(Q, t)* and are used to transform data for fitting. Fitting includes *MSD Fit*, *I(Q, t) Fit* and *ConvFit* and are used as wrappers for the MANTID fitting routines with specific model configurations.

5.1 Elastic Window

The *Elwin* tab is used as an interface for the *ElasticWindowMultiple* algorithm which is used to perform elastic scans over a series of energy transfer reductions.

The *ElasticWindowMultiple* algorithm does this by making repeated calls to the *ElasticWindow* algorithm which creates two workspaces containing the integral of each spectrum, transposed one of which has the axis converted to Q and the other to Q^2. If desired the *ElasticWindow* algorithm can also perform a background subtraction given an energy range defining the background signal.

The *ElasticWindowMultiple* then collects these results into two workspaces (one in Q and the other in Q^2) and converts the vertical axis to the run temperature if it is available from the sample logs otherwise it will use the last three digits of the run number.

An output workspace (*OutputELF*) is then given which is the Q workspace transposed.

If the temperature logs are available then the option to normalise the workspace to the lowest temperature and given as an additional output workspace (*OutputELT*).

The workflows of the *ElasticWindow* and *ElasticWindowMultiple* algorithms are shown in figures 14 and 15 respectively.

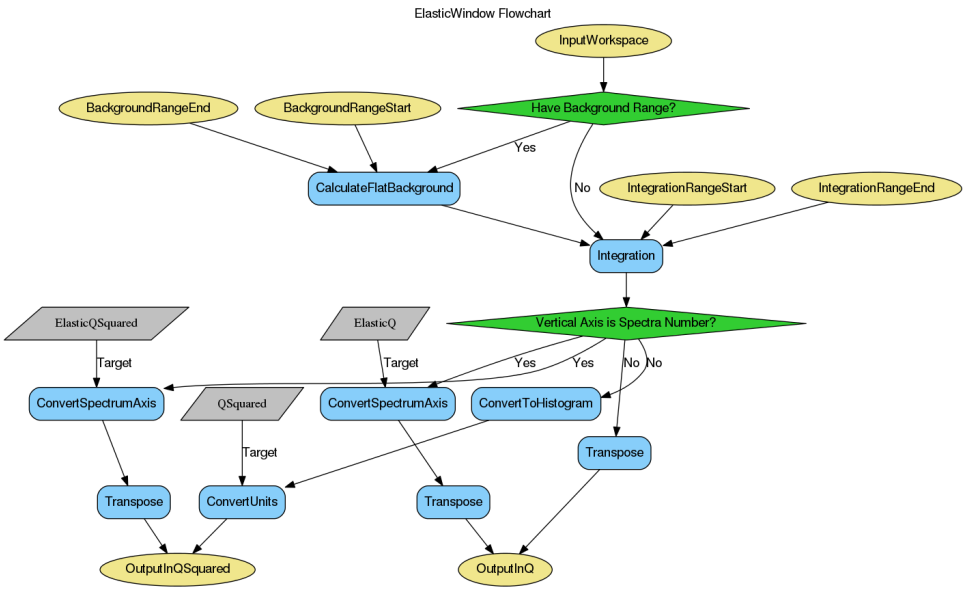


Figure 14: ElasticWindow flowchart

A full size version of this diagram is available online at `http://docs.mantidproject.org/algorithms/ElasticWindow-v1.html`

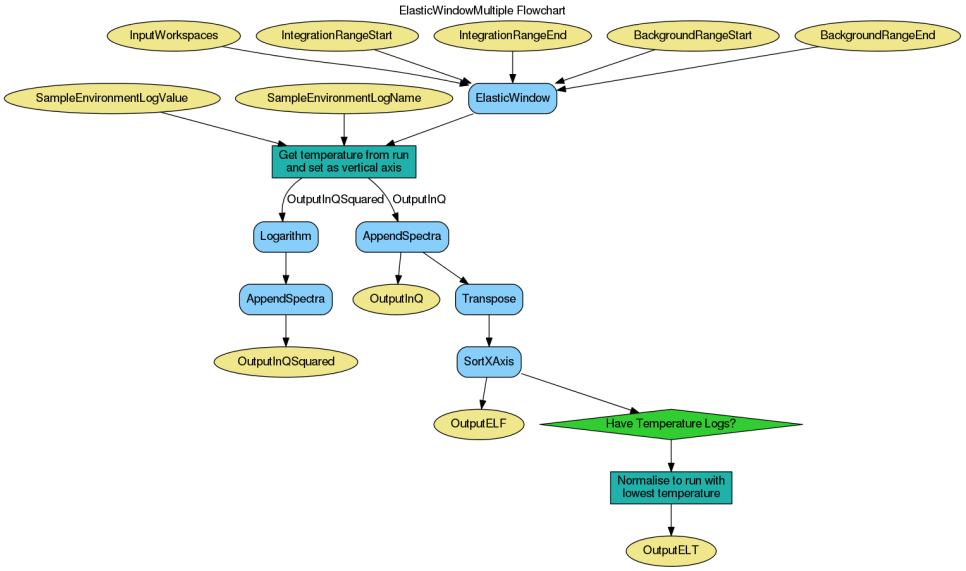


Figure 15: ElasticWindowMultiple flowchart

A full size version of this diagram is available online at `http://docs.mantidproject.org/algorithms/ElasticWindowMultiple-v1.html`

5.2 Mean Squared Displacement fitting

The *MSD Fit* tab provides an interface to the *MSDFit* algorithm which is used to calculate the mean squared displacement using the results of *Elastic Window* (see section 5.1).

This is done by fitting a *LinearBackground* function to each of the spectra in either the *OutputELF* or *OutputELT* workspaces created by the *ElasticWindowMultiple* algorithm, the mean squared displacement is then given by the gradient of the fit (fitting parameter A1).

A workspace is then created that contains the mean squared displacement as a function of run number or temperature, depending on th workspace that was provided to the *MSDFit* algorithm.

The full workflow of the *MSDFit* algorithm is given in the diagram in figure 16.

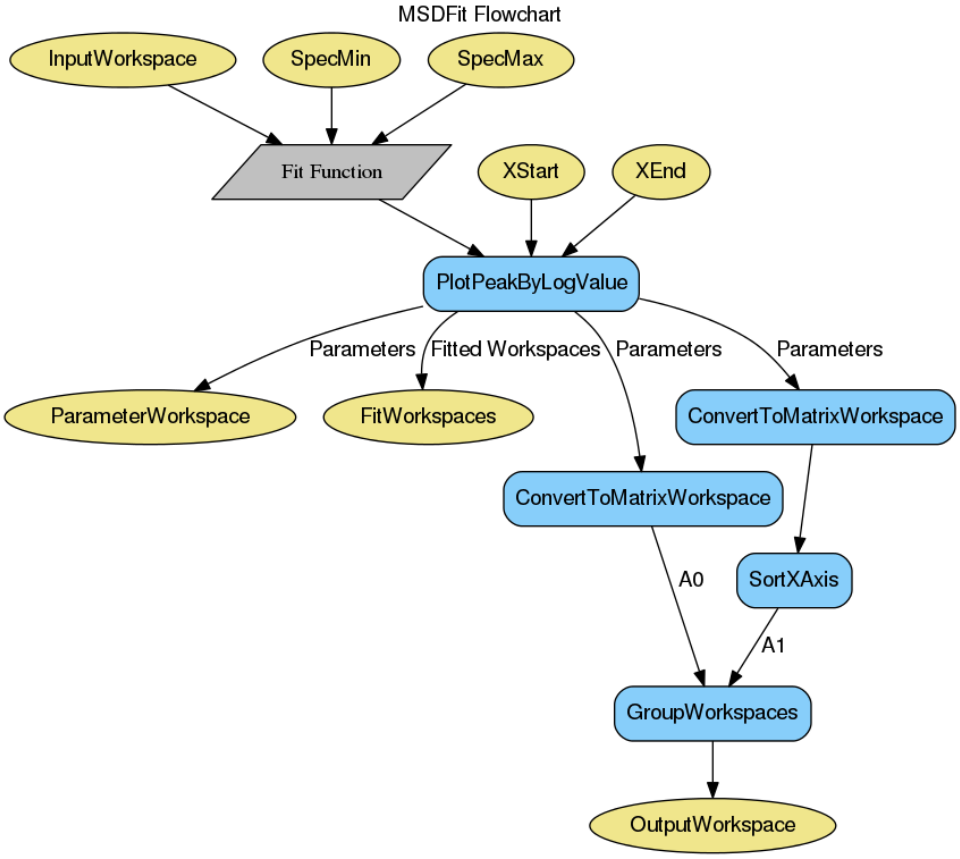


Figure 16: MSDFit flowchart

A full size version of this diagram is available online at `http://docs.mantidproject.org/algorithms/MSDFit-v1.html`

5.3 Transformation to I(Q, t)

The *I(Q, t)* tab provides a means of converting reduced data to the intermediate scattering function $I(Q,t)$ by means of Fourier transformation. This transformation is carried out by the *TransformToIqt* workflow algorithm.

The transformation is performed by obtaining the Fourier transform using the *ExtractFFTSpectrum* algorithm and dividing the resulting spectrum by the integral of the spectrum. This is performed for all spectra of the sample and the resolution, after which the sample workspace is divided by that of the resolution.

This transformation is based on the theory that the measured data $I(Q,\omega)$ is proportional to the convolution of the scattering function $S(Q,\omega)$ and the instrument resolution $R(Q,\omega)$ as per equation 3.

$$I(Q,\omega) = S(Q,\omega) \otimes R(Q,\omega) \tag{3}$$

Equation 3 can be deconvoluted in Q and ω by performing a Fourier transformation to obtain equation 4.

$$I(Q,t) = S(Q,t) \otimes R(Q,t) \tag{4}$$

The full workflow for the *TransformToIqt* algorithm is given in figure 17.

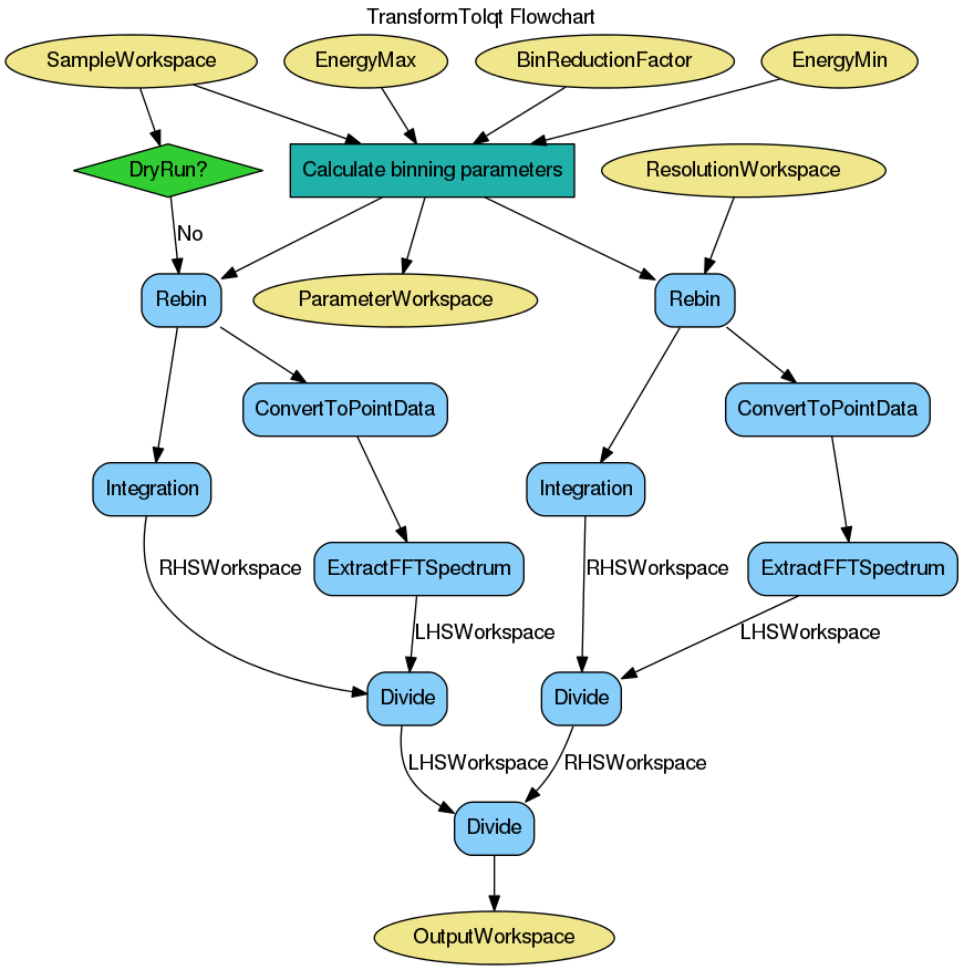


Figure 17: TransformToIqt flowchart

A full size version of this diagram is available online at `http://docs.mantidproject.org/algorithms/TransformToIqt-v1.html`

5.4 I(Q, t) fitting

The *I(Q, t) Fit* interface provides an interface in which a workspace in $I(Q,t)$ (obtained from the *I(Q, t)* tab 5.3) is fitted with a range of decaying exponential functions.

The current fitting models supported are:

1. One exponential
2. Two exponentials
3. One stretched exponential
4. One exponential and stretched exponential

There is the option to constrain the intensity of all fitted exponentials to the same value for all spectra.

If the option to use a single stretched exponential is selected then there is the option to constrain the beta parameters to be the same for all spectra in the workspace (i.e. the same for all Q).

The fitting results are given as both a series of fit workspaces containing both the original spectrum, fitted spectrum and difference between the two and a table workspaces containing the fitted parameters for each spectrum.

If desired there is also an option to use the FABADA [17] minimiser to perform basic Bayesian analysis by means of the probability distribution function output workspace. When this option is enabled the maximum number of fitting iterations is increased as a requirement for successful fitting using FABADA.

The default options for the *ChainLength*, *ConvergenceCriteria* and *JumpAcceptanceRate* are set to default options that are likely to give a successful fit and are exposed on the interface to allow a user to modify them if needed.

5.5 Convolution fitting

The *ConvFit* tab provides an interface to the *PlotPeakByLogValue* fitting algorithm with a predefined set of fitting models.

The interface allows fitting either:

- A DeltaFunction alone
- One Lorentzian (optionally with a DeltaFunction)
- Two Lorentzians (optionally with a DeltaFunction)
- DiffSphere
- DiffRotDiscreteCircle

An optional temperature correction can be applied, this is given by equation 5 and applied alongside the *Lorentizian*, *DiffSphere* and *DiffRotDiscreteCircle* fit functions using a *ProductFunction*.

$$f(x) = \frac{11.606 \times x}{T} / (1 - exp(\frac{11.606 \times x}{T}) \tag{5}$$

where x is the value of energy transfer and T is the temperature in Kelvin provided by the user.

The full fitting model is defined below:

- *CompositeFunction*
  - *LinearBackground*
  - *Convolution*
    - *Resolution*
    - Fitting Model
      - *DeltaFunction*
      - *ProductFunction*
        - *Lorentzian*
        - Temperature Correction
      - *ProductFunction*
        - *Lorentzian*
        - Temperature Correction
      - *ProductFunction*
        - *DiffSphere*
        - Temperature Correction
      - *ProductFunction*
        - *DiffRotDiscreteCircle*
        - Temperature Correction

As with the *I(Q, t) Fit* interface 5.4 *ConvFit* has the option to use the FABADA minimiser, the options here are identical to those available in *I(Q, t) Fit*.

5.6 JumpFit

The *JumpFit* tab provides a simple interface to the set of jump diffusion fitting functions within MANTID, this operates over the output created by either the *ConvFit* 5.5 or *Quasi* 7.2.

The interface allows fitting of a series of FWHMs that have been fitted to the quasi-elastic peaks of spectra at increasing values of Q, this is done with one of four fit functions which fit the residence time, τ and jump length, L.

ChudleyElliot
The *ChudleyElliot* function implements the diffusion model for liquids as described by C.T. Chudley and R.J. Elliott [2], this function takes the form of equation:

$$\Gamma(Q) = \frac{1 - sin(Ql)/Ql}{\tau} \tag{6}$$

HallRoss
The *HallRoss* function implements the model described by P.L. Hall and D.K. Ross [4], this model is can be expressed by the equation:

$$\Gamma(Q) = \frac{1 - exp(-l \times Q^2)}{\tau} \tag{7}$$

FickDiffusion
The *FickDiffusion* function uses Fick's diffusion laws and is given by the equation:

$$\Gamma(Q) = DQ^2 \tag{8}$$

where

$$D =< l^2 > \frac{Q^2}{6}\tau \tag{9}$$

TeixeiraWater
The *TeixeiraWater* function implements the diffusion model for water described by Teixeira et al in [19].

This model is given by the equation:

$$\Gamma(Q) = \frac{DQ^2}{1 + DQ^2\tau} \tag{10}$$

where

$$D =< l^2 > \frac{Q^2}{6}\tau \tag{11}$$

6 Corrections

The *Corrections* interface contains correction algorithms for experimental data, currently this is limited to absorption corrections.

The absorption corrections currently available for indirect inelastic instruments is provided in two implementations; the legacy MODES implementation, and the implementation that uses the existing MANTID absorption correction algorithms.

6.1 MODES

This implementation is available via the *Calculate Corrections* and *Apply Corrections* tabs of the *Indirect Data Analysis* interface and provides corrections for a flat, annular and cylindrical sample.

This uses one of two algorithms (*FlatPlatePaalmanPingsCorrection* and *CylinderPaalmanPingsCorrection*) to calculate a set of Paalman & Pings [16] correction factors which are then applied to a sample using the *ApplyPaalmanPingsCorrection* algorithm.

Note that the *CylinderPaalmanPingsCorrection* technically calculates corrections for an annulus, therefore in the *Calculate Corrections* interface when calculating corrections for a cylinder the *SampleInnerRadius* property is fixed to 0.

These algorithms are based on the routines that were previously used within the MODES [6] package and are the ones typically used by default by instrument scientists.

6.2 MANTID

MANTID has a set of general purpose absorption correction algorithms that are used is a set of workflow algorithms (*IndirectFlatPlateAbsorption*, *IndirectAnnulusAbsoprtion* and *IndirectCylinderAbsorption*) that are used to calculate corrections for indirect inelastic instruments.

These algorithms correct the data directly rather than outputting factors to be used in a further step to apply the corrections, however they can optionally output the A_{ss} and A_{cc} factors as per the Paalman & Pings format.

Further improvements to these algorithms should be carried out to allow them to replace the existing MODES implementation within MANTID, this is discussed further is section 13.3.

7 Bayesian Analysis

The *Indirect Bayes* (Interfaces ⇒Indirect ⇒Bayes) interface provides a set of Bayesian analysis routines for indirect inelastic in either an energy transfer reduction (workspaces ending in `_red`) or $S(Q, \omega)$ workspaces (ending in `_sqw`).

7.1 ResNorm

The *ResNorm v2* algorithm provides a method of fitting the energy resolution of an instrument to the peaks in an energy transfer reduction of a vanadium sample.

Version 2 of this algorithm is a port of the legacy FORTRAN routine previously used to perform this fitting which replaces the majority of the processing with use of the MANTID fitting functions and algorithms.

This is done by first normalising the instrument resolution to 1 then fitting the resolution using a *TabulatedFunction* which will fit a scale factor in the X and Y axis.

The fitted parameters are then used to provide the Intensity and Stretch output from the algorithm which is then used in the *Quasi* 7.2 tab.

Version 1 of the algorithm is a wrapper around the F2Py routine and will still be included in MANTID.

Figure 18 describes the workflow of version 2 of the *ResNorm* algorithm.

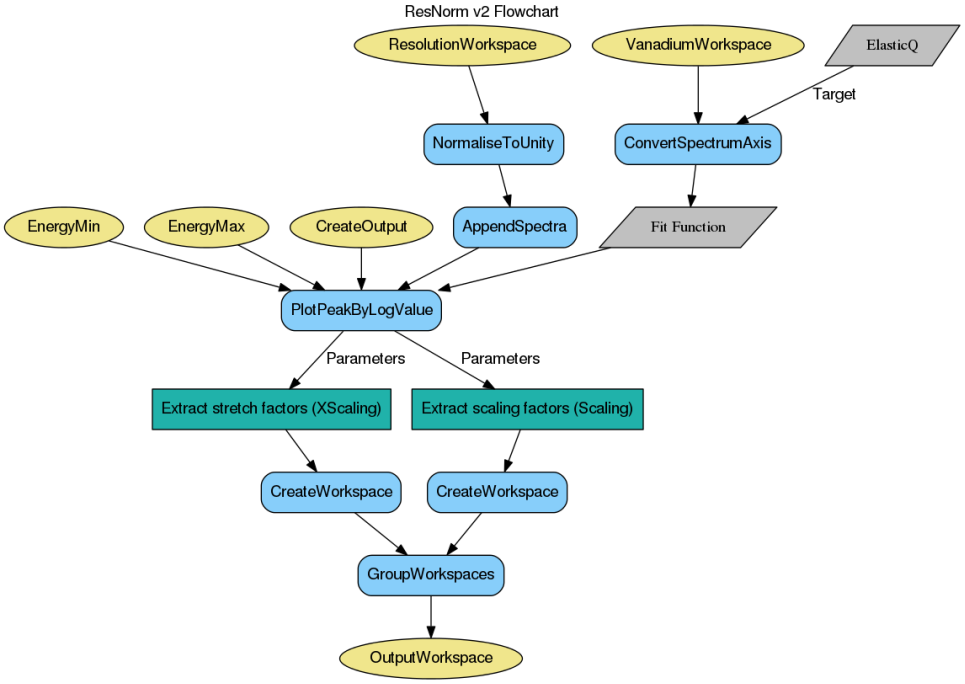


Figure 18: ResNorm v2 flowchart

A full size version of this diagram is available online at `http://docs.mantidproject.org/algorithms/ResNorm-v2.html`

7.2 Quasi

The Quasi routines are used to fit a selection of Lorentzians to an energy transfer reduction, the number of Lorentzian components is chosen using Bayesian analysis methods described in [18].

The program runs by fitting a δ-function plus the sum of several Lorentzians or a stretched exponential.

These routines are still implemented as FORTRN code wrapped in Python modules using F2Py and are only available in Windows builds of MANTID. There are three modules related to the Quasi routines:

`QLres`
: This module is used to fit Lorentzians, optionally using input from ResNorm.

`QLdata`
: This module is used when the resolution is more than one spectrum (i.e. the energy transfer reduction of the vanadium run), in this case the ResNorm input is ignored.

`QLse`
: This module is used to fit stretched exponentials.

The program runs by first refining parameters for the δ-function and linear background. Then refines the parameters for one, two and three Lorentzians.

7.3 Stretch

The Stretch routine is fundamentally very similar to the Quasi routines, however this fits the grid of β and σ values to each spectrum.

This routines uses a single module; `Quest`.

7.4 FABADA

In MANTID release 3.3 the FABADA minimiser was added to allow the MANTID fitting framework to perform Bayesian analysis by outputting the PDF for each fitted parameter and the cost function.

The option to use this as the minimiser for the fitting performed by the ConvFit and I(Q, t) Fit interfaces is already present in their respective user interfaces (see sections 5.4 and 5.5).

The theory of FABADA is described by L.C. Padro et al in [17].

8 Modelling and Simulation

The *Indirect Simulation* interface (Interfaces ⇒Indirect ⇒Simulation) has existed within MANTID since release 3.2, however at that time it could only be used for loading data from nMoldyn [15].

Since then it has been expanded to also include loaders for data from Sassena [8] and CASTEP [3] as well as several improvements to the preexisting loader for nMoldyn.

8.1 nMoldyn

The loader for simulation data from the nMoldyn package has been expanded to allow data with an X axis in energy to be processed in order to make it more comparable to experimental data, this is done in three steps (all of which are optional):

Peak Symmetrisation A theoretical energy transfer produced by nMoldyn would typically only have the positive energy transfer, this step allows the spectra to be reflected in the Y axis to obtain a peak around the elastic line.

Crop Energy The energy range of a theoretical energy transfer may greatly exceed that of an instrument, this step removes data above a given energy threshold.

Resolution Convolution This step convolves the theoretical data with the instrument resolution to obtain a spectra which is directly comparable to measured spectra from an instrument.

This uses the relation in equation 3.

Currently the algorithm supports loading data in ASCII formats (`.cdl` and `.dat`) and is limited to loading spectra corresponding to $S(Q, \omega)$ (those ending in `Sqw`) and $I(Q, t)$ (those ending in `Fqt`).

8.2 CASTEP

Data from the CASTEP software can currently be loaded through the *DensityOfStates* algorithm, currently this can load data for calculation of a full or partial density of states as well as IR and Raman simulation data.

Currently the algorithm has four modes of operation:

DOS
In the density of states mode the algorithm calculates the density of states based on the phonon trajectories read from a *.phonon* file.

There is also the option to specify a subset of atoms to be used in the calculation and to weight the individual atom contributions by the cross sections.

IonTable
This mode outputs a table containing the number of each atom contained in a `.phonon` file, this is mainly used on within the *Indirect Simulation* interface to show a list of atoms for calculation of a partial density of states.

IR
This mode extracts infrared spectra from a `.phonon` or `.castep` file.

Raman
This mode extracts Raman spectra from a `.phonon` or `.castep` file.

Work is currently in progress to have this algorithm extended to calculate the structure factor $S(Q, \omega)$ from the loaded data.

8.3 Sassena

There is basic support for loading data from the Sassena using the *LoadSassena* algorithm, this functionality is also available via the *Sassena* tab of the *Indirect Simulation* interface.

9 Indirect Tools

The *Indirect Tools* interface (Interfaces ⇒Indirect ⇒Tools) is used for any interfaces that do not fit into the category of the other indirect interfaces but are still worth having, at the moment this includes the sample transmission calculator (*Transmission* tab) and legacy ILL data loader (*Load ILL* tab).

This interface was added in release 3.3 and contributed to the removal of the *Indirect Load Ascii* interface which previously contained the *Load ILL* tab which was moved to this interface with little other modification.

The *Transmission* tab performs an approximate calculation of the sample transmission and scattering at the elastic line for IRIS, OSIRIS, TOSCA, VISION and BASIS, this can be useful for estimating the required sample size for an experiment.

The *Load ILL* tab holds the legacy loaders for data from the indirect geometry spectrometers at the ILL, the use of this has been made partially redundant by the work carried out to support ILL instruments within the *Indirect Data Reduction* interface.

The interface supports loading files from IN13 and IN16B (via the LoadILLIndirect algorithm) as well as any data in the *.asc* and `.inx` file formats, this functionality is provided by the Python scripts in `scripts/Inelastic/IndirectNeutron.py`.

10 VESUVIO data analysis

The analysis of VESUVIO data has moved from the `ncs.py` scripts to two more robust and user friendly scripts, namely one for performing calibration of the instrument intended for use by the instrument scientists only and another for performing analysis of sample data to be used by both instrument scientists and users.

Both sets of scripts are currently being kept separate from the main MANTID code and are available at `https://github.com/mantidproject/scripts/tree/master/development/inelastic`.

10.1 Calibration

The `CalibrateVesuvio.py` Python file contains two algorithms that are can be used within MANTID; *EVSCalibrationAnalysis* and *EVSCalibrationFit*, the first of which is the workflow for performing calibration on the instrument, the second is the algorithm that is used to fit peaks defined by the instrument parameter file to spectra in the workspace.

The calibration of the scattering angle (Φ), final energy (E_1), incident flight path (L_0) and final flight path (L_1) are performed as described by J. Mayers at al [11].

Currently the *EVSCalibrationAnalysis* algorithm saves a parameter (`.par`) file that contains the calibrated instrument parameters which is then loaded in the reduction and analysis routines where it is used to correct the default instrument parameters using the *UpdateInstrumentFromFile* algorithm.

Work is currently in progress to extend this algorithm to allow the resolution of these instrument parameters to also be calculated.

10.2 Data Analysis

The VESUVIO user scripts are used to provide a simple interface to the range of fitting and correction routines for VESUVIO and currently operate in time of flight using the `tof_fit.py` script which defines options for the data treatment workflow.

The data reduction and analysis workflow for VESUVIO can be split into several stages: data loading, initial fit, calculate corrections, apply corrections and final fit.

The workflow is described in figure 19.

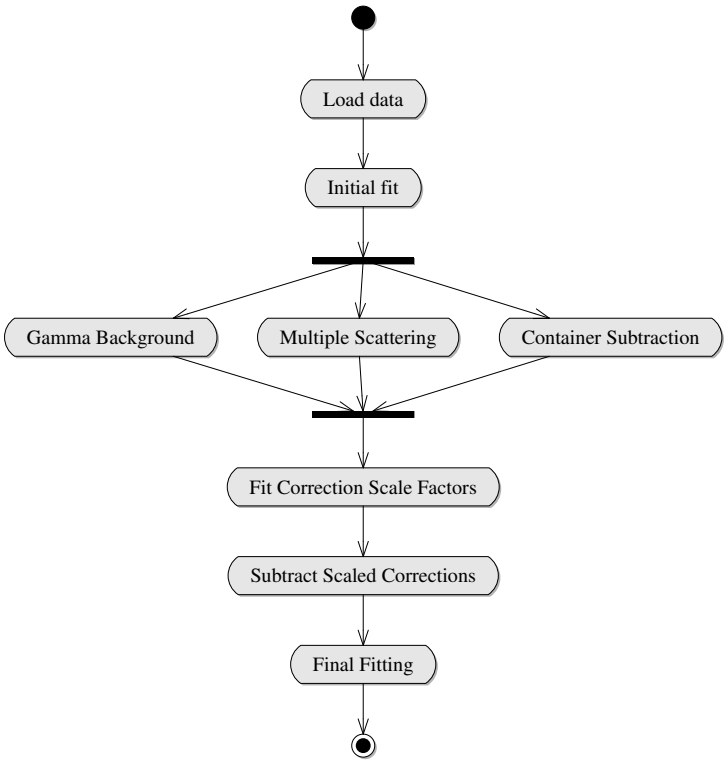


Figure 19: VESUVIO data analysis workflow

The initial fitting and final fitting stages are pretty much identical in the fitting that is performed, the only difference is the input data; for the initial fit this is the data unmodified after the data loading step, for the final fit this is the data after the corrections have been applied.

The initial fit is required in order to calculate the multiple scattering and gamma background corrections that require a set of fitted parameters in their calculation.

10.2.1 Data Loading

The data loading step consists of loading the raw files that are defined by the `runs` variable in the `tof_fit.py`, this then crops the workspaces to either the range for spectra in the case of spectra fitting mode or crops to spectra ranges for given banks in the case that bank fitting is used.

If the user specified any rebinning options via the `flags['bin_parameters']` flag then that rebinning is performed here.

If any container runs were provided using the `flags['container_runs']` flag then they are loaded in the same way as sample data.

10.2.2 Fitting

The fitting for VESUVIO data is provided by the *VesuvioTOFFit* workflow algorithm which takes a series of parameters which are generated by the helper classes in the *vesuvio* Python module based on the values of the flags in the *tof_fit.py* script.

The mass peaks to be fitted are defined using the `flags['masses']` flag, this is a list of Python dictionaries that contain properties of the mass, namely the mass value, peak function (currently either `Gaussian` using the *GaussianComptonProfile* or `GramCharlier` using the *GramCharlierComptonProfile*) and peak width. An example of such a mass peak setup is given in listing 1.

The *GaussianComptonProfile* is used for simple peaks that can be approximated to a Gaussian peak profile, the *GramCharlierComptonProfile* is used for fitting peaks where the atoms in the sample are affected by anisotropy and anharmonicity. The theory behind these fit functions is explained in [13] and [1].

```
mass1 = {'value': 7, 'function': 'Gaussian', 'width': [4,7,10]}
mass2 = {'value': 9, 'function': 'Gaussian', 'width': [4,7,20]}
mass3 = {'value': 19.0, 'function': 'Gaussian', 'width': [5,10,20]}
mass4 = {'value': 27.0, 'function': 'Gaussian', 'width': 13.42554}
flags['masses'] = [mass1, mass2,mass3,mass4]
```

Listing 1: `tof_fit.py` mass peak definition

The mass profiles in listing 1 define the following mass peaks:

- Lithium with a Gaussian profile, width constrained between 4 and 10 width default of 7
- Beryllium with a Gaussian profile, width constrained between 1 and 10 with default of 7
- Florine with a Gaussian profile, width constrained between 5 and 20 with default of 10
- Aluminium (container) with a Gaussian profile, width fixed at 13.42554

The intensity of fitted mass peaks may also be constrained using the `flags['intensity_constraints']` flag, here a number of constraints can be provided that describe the relation of one peak to another. An example of this is shown in listing 2.

```
flags['intensity_constraints'] = [
        [1, 0, -0.1,  0],
        [0, 1, -0.66, 0]
    ]
```

Listing 2: `tof_fit.py` mass peak intensity constraints

In this example the intensity of the first peak is constrained to be 0.1 times the intensity of the third and the intensity of the second is constrained to be 2/3 times the intensity of the third.

There is also an option to fit a background to the sample data, currently this only supports a polynomial background using the *Polynomical* fit function. This option is set using the `flags['background']` flag.

```
flags['background'] = {'function': 'Polynomial', 'order': 2}
```

Listing 3: `tof_fit.py` background

10.2.3 Calculate Corrections

The calculate correction stage takes the data and the result of the initial fitting and calculates a set of correction workspaces for multiple scattering, gamma background (optional) and the empty container (optional).

For multiple scattering calculation the *CalculateMSVesuvio* algorithm is used which performs a Monte Carlo simulation as per the theory described by J. Mayers et al [12].

The gamma background is computed by a simple simulation of the expected count across all of the foils. The corrected workspace counts are computed by calculating a ratio of the expected counts at the detector to the integrated foil counts (β), the final corrected count rate c_f is then defined by equation 12.

$$c_f = c_i - \beta c_b \tag{12}$$

No processing is performed on the container workspace other than extracting a single spectrum for the sample spectrum being corrected, this spectrum extracted form the raw container data is the correction workspace in this case.

10.2.4 Apply Corrections

The corrections are applied by first scaling each of them to a coefficient defining the proportion of the uncorrected sample data they represent, such coefficients are calculated by performing a linear fit of each correction to the sample data using several *TabulatedFunction* fit functions inside a *CompositeFunction*.

In the case of multiple scattering it is the total scattering spectrum that is used in the fit and the multiple scattering spectrum that is scaled.

Once each of the corrections have been scaled they are simply subtracted from the input spectrum.

There are options to fix the parameters for the scale factor for the container and gamma background spectra, these are provided by means of the `flags['fixed_gamma_scaling']` and `flags['fixed_container_scaling']` flags in `tof_fit.py`.

Note that since all masses are present on both the initial fit and final fit then the reduction in intensity of the mass peak for the container material can be used as a measure of the quality of the container subtraction.

10.2.5 Output

Figure 20 shows the output of the VESUVIO workflow script with a sample of Lithium, Beryllium and Florine in an Aluminium container for spectra 153 (the first detector in forward scattering.

In this case the spectra contains corrections for the gamma background and multiple scattering.

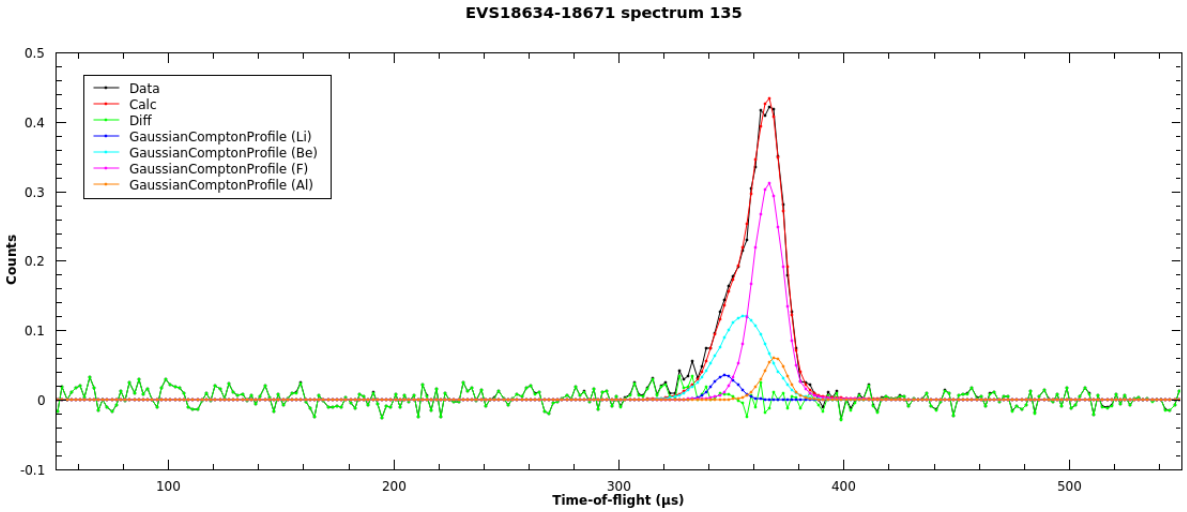


Figure 20: VESUVIO data analysis for Li, Be, F sample

11 User Interfaces

During release 3.3 of MANTID significant work was carried out to improve the quality and reliability of both the indirect custom interfaces and the scripts that performed the data treatment. Previously this was carried out by a set of Python scripts that were called by the user interface this had several disadvantages in that it reduced the scope for automated testing of the routines, made documenting the routines difficult as the only options were either the MANTID Wiki pages or in the code its self and made the routines dependant on the user interface which meant that a fault in the user interface side meant greatly reduced access to the routines.

The approach taken to resolve this was to move such Python scripts to MANTID Python algorithms which instantly makes them available via the algorithm browser within the MantidPlot interface and provides access to them via the MANTID Python API, therefore also allowing users to use the routines in their own scripts. This also allowed them to be documented using the rST file based documentation built into MANTID which ensures that the documentation for each routine stays up to date with any changes in its parameters or data treatment.

Moving the data treatment code to MANTID algorithms has allowed the user interface code to simply call the algorithms on the input data, essentially turning the interfaces into a facade for the algorithms (or a series of algorithms in some cases), this separation between the user interface and the algorithms allows for more automated test coverage for the algorithms and helps to reduce the complexity of the user interface code.

The following is a list of the new workflow algorithms (that have been converted from existing Python scripts) and the interfaces they are used in:

ISISIndirectEnergyTransfer
Indirect ⇒Data Reduction ⇒ISIS Energy Transfer

IndirectCalibration
Indirect ⇒Data Reduction ⇒ISIS Calibration

IndirectResolution
Indirect ⇒Data Reduction ⇒ISIS Calibration

TimeSlice
Indirect ⇒Data Reduction ⇒Diagnostics

IndirectTransmissionMonitor
Indirect ⇒Data Reduction ⇒Transmission

ISISIndirectDiffractionReduction
Indirect ⇒Diffraction (IRIS, OSIRIS diffspec, TOSCA & VESUVIO)

ElasticWindowMultiple
Indirect ⇒Data Analysis ⇒Elwin

MSDFit
Indirect ⇒Data Analysis ⇒MSDFit

TransformToIqt
Indirect ⇒Data Analysis ⇒I(Q, t)

FlatPlatePaalmanPingsCorrection
Indirect ⇒Data Analysis ⇒Calculate Corrections

CylinderPaalmanPingsCorrection
Indirect ⇒Data Analysis ⇒Calculate Corrections

ApplyPaalmanPingsCorrection
Indirect ⇒Data Analysis ⇒Apply Corrections

The data reduction interface has now been split from the direct interface ensuring that any work carried out for the indirect side does not affect the direct reduction, this also helps to keep the code cleaner especially since all of the indirect data reduction tabs have moved away from using Python scripts to perform their processing.

There is now a single base class for tabs on the indirect custom interfaces (`IndirectTab`) which contains common functionality for plotting, loading, saving and workspace name operations that are used in the majority of tabs across all indirect interfaces.

11.1 Code Structure

Over releases 3.3 and 3.4 the structure of the indirect user interfaces was drastically changed to reduce the amount of code duplication and to enable common functionality between the interfaces (for instance, the Python script export).

Much of this duplicated code was relating to the creation and manipulation of spectra plots, which has now been moved to a specialised Qt widget, `PreviewPlot`, which handles plotting a MatrixWorkspace, manipulating the plot using tools similar to those available in the standard MANTID plots and displaying a legend for multiple plot curves.

Figure 21 shows a UML diagram showing the structure of the indirect custom interface code, note that for simplicity some functions and member variables have been omitted form the diagram.

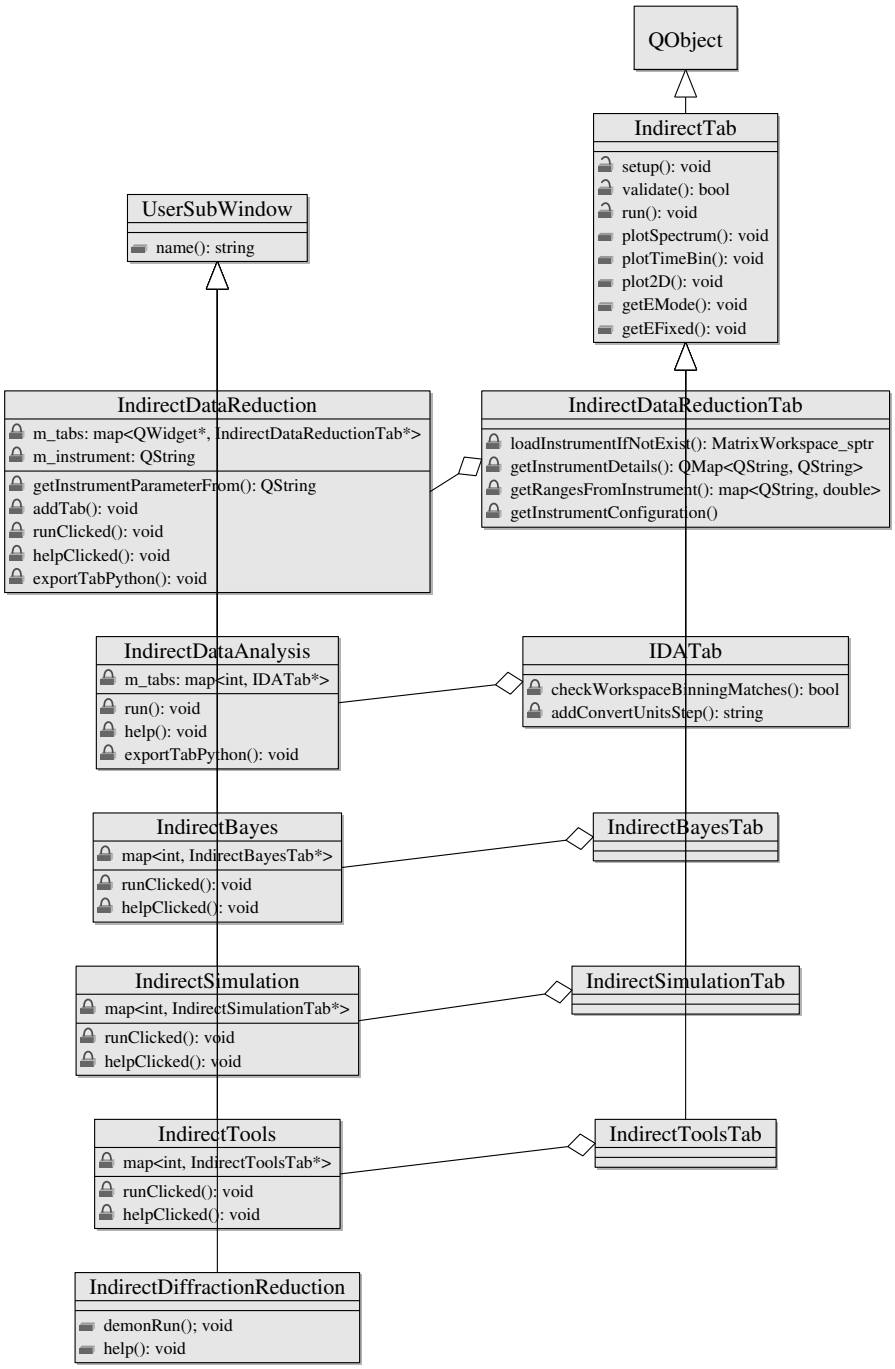


Figure 21: Indirect custom interface code structure

11.2 Plotting in interfaces

In release 3.4 a new Qt widget (`PreviewPlot`) was added to provide plotting functionality to custom interfaces within MantidPlot, this included all of the functionality that was commonly used with the mini plots included on many of the indirect custom interfaces such as multiple curves and drag and drop selectors for values and ranges as well as several new pieces of functionality including pan and zoom tools (as can be used in the MantidPlot plots), error bars and selectable axis scales.

All of this functionality is configured through the context menu of the plot which can be accessed by right clicking on the plot area as show in figure 22.

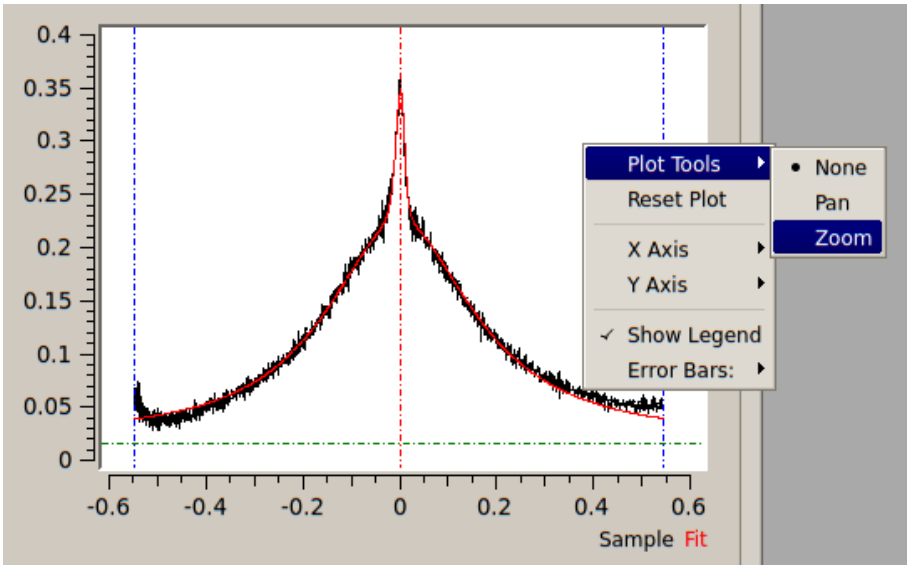


Figure 22: Preview plot options in Indirect Data Analysis

11.3 Python script export

In release 3.3 the option to export a Python script from the Indirect Data Reduction and Indirect Data Analysis interfaces was added, this allows a simple means by which to obtain a Python script which will reproduce the processing done by a given tab on the interface.

This was intended both as a means for minor modification of the existing functionality in the interface or as a means to obtain a template reduction or analysis script which can then be used for batch processing multiple data sets.

A Python script can be exported by clicking on the *Py* button in the bottom left hand corner of either the Indirect Data Reduction or Indirect Data Analysis interface (figure 23).

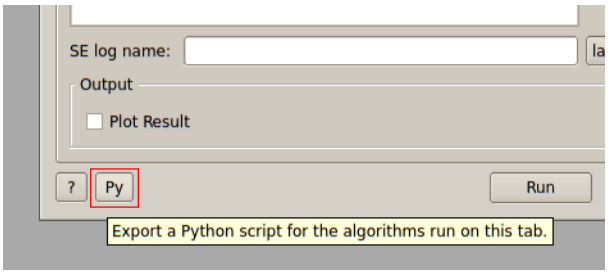


Figure 23: Python script export button

When this option is selected a dialog box is shown (see figure 24) which allows you to change where the generated Python script will be saved and the name of the file. The option to select the workspace that the history will be taken from is also provided, this is set to a default value by the interface when it is executed but can be changed to be any workspace that is created by the interface.

This dialog box is simply the automatically generated algorithm dialog for the *GeneratePythonScript* algorithm which is used to generate a Python script from the history of a given workspace.

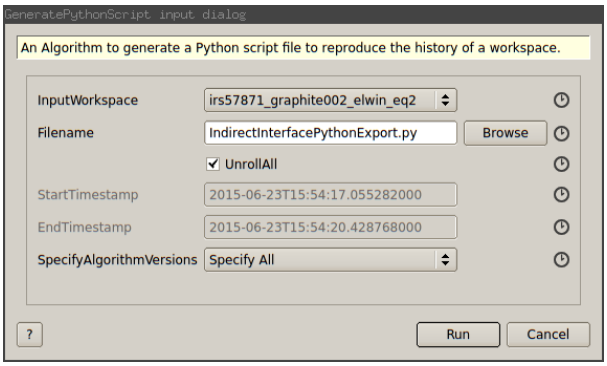


Figure 24: Python script export dialog

Once executed you will have a Python script that will reproduce the processing for a given interface with the assumption that any data it uses is already loaded into MANTID (the exceptions to this are when data loading is part of the processing, for example in the ISIS Energy Transfer tab of Indirect Data Reduction).

An example of such a script is given in listing 4. This shows the Python script exported after running the Elwin tab of Indirect Data Analysis on three data sets.

```
##########################################################################
#Python Script Generated by GeneratePythonScript Algorithm
##########################################################################
GroupWorkspaces(InputWorkspaces='irs57871_graphite002_red,irs57872_graphite002_red,irs57873_graphite002_red',
                OutputWorkspace='IDA_Elwin_Input', Version=1)

# Child algorithms of ElasticWindowMultiple
ElasticWindow(InputWorkspace='irs57871_graphite002_red', OutputInQ='__irs57871_graphite002_red_q',
              OutputInQSquared='__irs57871_graphite002_red_q2', IntegrationRangeStart=-0.017500000000000002,
              IntegrationRangeEnd=0.017500000000000002, Version=1)
Logarithm(InputWorkspace='__irs57871_graphite002_red_q2', OutputWorkspace='__irs57871_graphite002_red_q2', Version=1)
ElasticWindow(InputWorkspace='irs57872_graphite002_red', OutputInQ='__irs57872_graphite002_red_q',
              OutputInQSquared='__irs57872_graphite002_red_q2', IntegrationRangeStart=-0.017500000000000002,
              IntegrationRangeEnd=0.017500000000000002, Version=1)
Logarithm(InputWorkspace='__irs57872_graphite002_red_q2', OutputWorkspace='__irs57872_graphite002_red_q2', Version=1)
ElasticWindow(InputWorkspace='irs57873_graphite002_red', OutputInQ='__irs57873_graphite002_red_q',
              OutputInQSquared='__irs57873_graphite002_red_q2', IntegrationRangeStart=-0.017500000000000002,
              IntegrationRangeEnd=0.017500000000000002, Version=1)
Logarithm(InputWorkspace='__irs57873_graphite002_red_q2', OutputWorkspace='__irs57873_graphite002_red_q2', Version=1)
AppendSpectra(InputWorkspace1='__irs57871_graphite002_red_q', InputWorkspace2='__irs57872_graphite002_red_q',
              OutputWorkspace='irs57871_graphite002_elwin_eq', Version=1)
AppendSpectra(InputWorkspace1='__irs57871_graphite002_red_q2', InputWorkspace2='__irs57872_graphite002_red_q2',
              OutputWorkspace='irs57871_graphite002_elwin_eq2', Version=1)
AppendSpectra(InputWorkspace1='irs57871_graphite002_elwin_eq', InputWorkspace2='__irs57873_graphite002_red_q',
              OutputWorkspace='irs57871_graphite002_elwin_eq', Version=1)
AppendSpectra(InputWorkspace1='irs57871_graphite002_elwin_eq2', InputWorkspace2='__irs57873_graphite002_red_q2',
              OutputWorkspace='irs57871_graphite002_elwin_eq2', Version=1)
DeleteWorkspace(Workspace='__irs57871_graphite002_red_q', Version=1)
DeleteWorkspace(Workspace='__irs57872_graphite002_red_q', Version=1)
DeleteWorkspace(Workspace='__irs57873_graphite002_red_q', Version=1)
DeleteWorkspace(Workspace='__irs57871_graphite002_red_q2', Version=1)
DeleteWorkspace(Workspace='__irs57872_graphite002_red_q2', Version=1)
DeleteWorkspace(Workspace='__irs57873_graphite002_red_q2', Version=1)
Transpose(InputWorkspace='irs57871_graphite002_elwin_eq', OutputWorkspace='irs57871_graphite002_elwin_elf', Version=1)
SortXAxis(InputWorkspace='irs57871_graphite002_elwin_elf', OutputWorkspace='irs57871_graphite002_elwin_elf', Version=1)
# End of child algorithms of ElasticWindowMultiple
```

Listing 4: Exported Python script

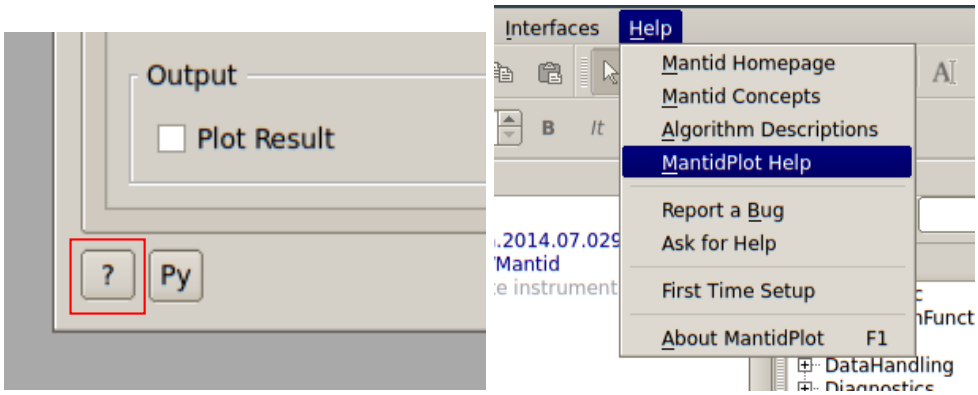
The script is generated by extracting the algorithm history entries from a workspace that was processed by the interface, the time window in which entries are extracted is determined by how long the algorithms that are executed by the user running the interface take to execute. Currently the history is taken between 1 second before the user pressed the Run button and 1 second after that last algorithm has finished execution.

11.4 Interface Documentation

In release 3.4 the documentation for all of the indirect interfaces was moved to the rST based documentation pages that are accessible both offline within MantidPlot and online via `http://docs.mantidproject.org/interfaces`.

This has the advantage that it goes further in enforcing that the documentation is updated alongside the interface as the screenshots used in this documentation are automatically captured when the documentation is generated.

The interface help pages can be accessed via either the help (*?*) button in the bottom left hand corner of each indirect interface (figure 25a) which will open the help page for the interface you are currently on (figure 26) or via the MantidPlot Help option of the Help menu within the MantidPlot main window (figure 25b) which will open the main MANTID help pages, from there you can select the Interfaces category.



(a) Via the help button in the bottom left of the interface

(b) Via the MantidPlot Help option in the Help menu

Figure 25: Means of accessing the help pages

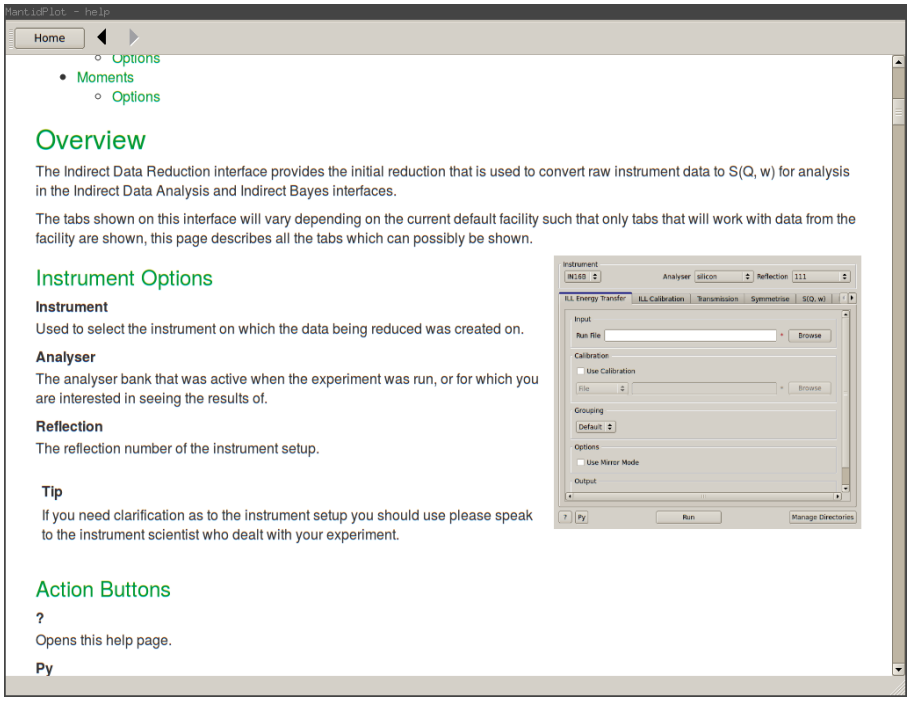


Figure 26: Help page for the Indirect Data Reduction interface

12 Automated testing

When the majority of the reduction and analysis scripts were converted to workflow algorithms a set of unit tests for each algorithm was added which aim to test that the algorithm behaves as expected with respect to input validation, output format, etc.

Algorithms are also expected to have a documentation test which tests the usage examples given to users through the algorithm documentation pages, this is typically a bare minimum example and does not provide enough checking to guarantee an algorithm is functioning.

The system test coverage has only marginally increased over the year and further work sill needs to be undertaken to ensure compatibility between reduction and analysis algorithms in a workflow.

To an extent this task could also be accomplished by more in depth unit testing for key pieces of information that further reduction is known to rely on in sample logs, the instrument attached to a workspace or the sample information. As a very basic example, *ApplyPaalmanPingsCorrection* requires that all input workspacea are MatrixWorkspaces that have X axis units of wavelength and the unit tests for *FlatPlatePaalmanPingsCorrection* and *CylinderPaalmanPingsCorrection* check that the algorithms are outputting MatrixWorkspaces that meet this criteria.

This approach has the advantage that unit tests can be run on sample generated data sets that are relatively small which in turn vastly reduces the time taken to run a full test suite. System tests by comparison take a significant amount of time and are currently the longest running job on the build server, therefore if it is possible to do this testing in a unit test it should be the preferred method.

12.1 Tests

The automated tests that are performed can be split into several categories:

Unit Tests

Unit tests are designed to cover all possible sets of code execution paths and test a variety of cases that are known to cause issues with the code under test.

These tests are designed to cover as much code as possible and be very fast to execute, as such they typically only use very small sets of generated data rather than real data.

These tests can be found in various location throughout MANTID, the most relevant locations are:

- C++ Algorithms: `https://github.com/mantidproject/mantid/tree/master/Code/Mantid/Framework/Algorithms/test`
- C++ Workflow Algorithms: `https://github.com/mantidproject/mantid/tree/master/Code/Mantid/Framework/WorkflowAlgorithms/test`
- Python Algorithms: `https://github.com/mantidproject/mantid/tree/master/Code/Mantid/Framework/PythonInterface/test/python/plugins/algorithms`

Performance tests

These tests are designed to obtain a rough estimate of how fast aspects of MANTID are and to gauge change is performance over time. These tests only test a small subsection of all functionality within MANTID.

The code for these tests are kept in the same locations as the standard unit tests.

System Tests

System tests are used to test the output of a know workflow with known correct data to ensure that the data treatment is correct and has not been altered by new changes.

These tests tend to use real data and can take up to 5 minutes each to execute.

These tests can be found at `https://github.com/mantidproject/mantid/tree/master/Code/Mantid/Testing/SystemTests/tests/analysis`.

Documentation Tests

Documentation tests are used to verify that the usage examples used on the algorithm documentation pages will actually run if a user tried to execute them, they typically only test a very simple use case.

These test are part of the documentation pages which can be found at `https://github.com/mantidproject/mantid/tree/master/Code/Mantid/docs/source/algorithms`.

Static Analysis

There are several tools used to check code quality that can be used to identify possible areas where code can fail, this does not rely on the code in question being covered by another type of test.

Currently the checkers in use include:

- C++ compiler warnings (C++)
- Coverity (C++)
- cppcheck (C++)
- PyLint (Python)
- Doxygen (for finding issues in documentation)

Note that the C++ compiler varies depending on the platform MANTID is being built on.

12.2 Workflow

Figure 27 shows the workflow for making changes to MANTID, in this case the tester is another member of the development team.

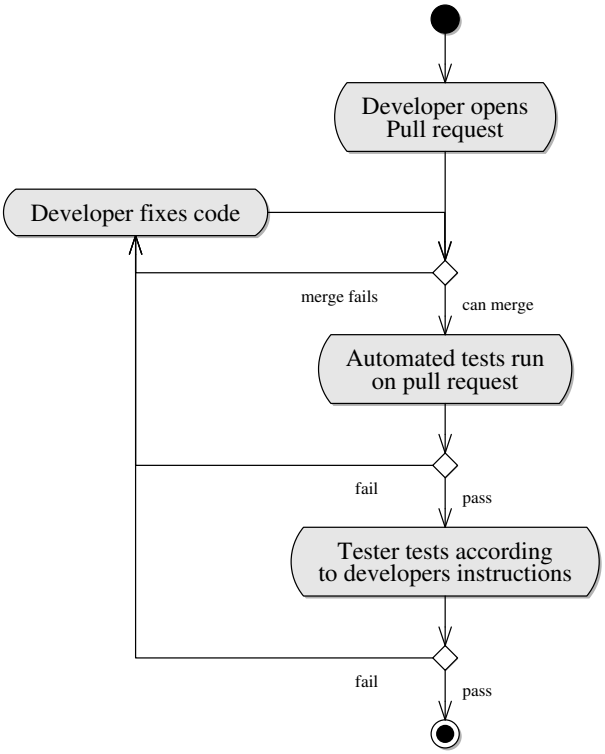


Figure 27: Workflow for testing changes to MANTID

The automated tests carried out in figure 27 currently consist of:

- Compile (RHEL6, RHEL7, OSX, Ubuntu, Windows 7)
- Unit tests (RHEL6, RHEL7, OSX, Ubuntu, Windows 7)
- System tests (RHEL7)
- Documentation tests (Ubuntu)

- Pylint
- cppcheck
- Doxygen

The full suite of tests then run on all platforms regularly on the master branch when the change has been approved.

13 Future planning

This section outlines the planned future work to improve support for indirect geometry instruments within MANTID relating both to data reduction & analysis and simulation & modelling.

13.1 Bayesian analysis

Currently there is a requirement by the scientific steering committee to remove the remaining FORTRAN code used in the indirect analysis routines and a further requirement to improve Bayesian analysis for indirect data sets, both of these requirements are partially solved with the replacement of the remaining Bayesian fitting routines used in the Indirect Bayes interface.

Currently the plan on replacing the remaining FORTRAN fitting routines is to make use of the FABADA minimizer withing MANTID as the probability distribution function that is output from the fitting when using FABADA will be roughly equivalent to the output of the existing routines.

13.2 Modelling and simulation support

There is a requirement within MANTID to extend the range of tools for loading and analysing the output of modelling and simulation tools (mostly using density functional theory) for use with multiple scientific techniques.

There are also several requirements specificly from the MSG:

- Ability to calculate an $S(Q, \omega)$ from data loaded by the *DensityOfStates* algorithm.
- Support for the LAMMPS [7] simulation package.

13.3 Absorption and multiple scattering corrections

The current Paalman & Pings absorption corrections used in indirect are fixed to a given sample shape based on the way they are coded, however it would be better to have a general purpose algorithm that would allow any sample shape to be used for th sample, similar to what is done with the Monte Carlo absorption correction algorithms currently in MANTID.

However these correction algorithms do not perform corrections for the container which is some in the current indirect absorption corrections.

The ideal solution to this would be to extend the current absorption correction algorithms within MANTID to perform corrections for both the sample and container, outputting in the Paalman & Pings format.

This would require a method of defining unique compositions for different parts of the sample shape in order to differentiate between the sample its self and the container and any other sample environment that may be in place. Doing so would also automatically allow absorption corrections for samples with several distinct compositions.

As with the multiple scattering there is a scientific steering committee requirement to overhaul the current absorption correction framework within MANTID and these requirements will be taken into account when the new functionality is designed.

13.4 Further work on user interfaces

Although the indirect custom interfaces have been significantly improved over the last year there are still several changes that could be made to further improve the code quality.

One issue is that there is still code duplication in interfaces that have a property tree widget (such as ConvFit and I(Q, t) Fit), one possible way this could be resolved is to create a new Qt widget that either wrapped or inherited from the `QtTreepropertyBrowser` widget and provided a simple interface for storage of the properties as currently each interface that uses a property tree also needs to use a boolean and double manager and have a map of `QtProperty` objects.

Whilst some of this functionality has been moved to `IndirectTab` to help reduce the amount of duplicated code, this remains one of the main areas where the indirect interfaces still require work.

The I(Q, t) Fit and ConvFit tabs on the Indirect Data Analysis interface still partially rely Python scripts for their processing, however since the majority of what the scripts do is calling the *PlotPeakByLogValue* fitting algorithm this code can be moved to the user interface to reduce code complexity.

The only complication here is handling the processing of the fitted parameters which convert the table workspace into a MatrixWorkspace and calculate the elastic incoherent structure factor, however an algorithm could be added that takes the parameter table and does this separately from the interface.

There is also a requirement to allow the ConvFit interface to work with any quasi-elastic fitting function, this should simply be a case of replacing the code the populates the property tree with the fitting parameters with code that will first query the fit function to determine what parameters are required, then having the code the builds the fit function parse the contents of the property tree to find parameters rather than using hard coded parameter names as it does currently.

One issue that may arise here is handling the fitting modes that do not simply use a single function, for example the Delta function and Lorentzian combinations that are currently possible, these functions could still need to be hard coded into the user interface however an alternative solution would be to define additional fit functions under the quasi-elastic category that cover all of the current possible Delta function and Lorentzian peak combinations.

13.5 VESUVIO

The VESUVIO reduction and analysis is still confined to the user scripts and have yet to be moved to a GUI, however the current user script now makes it quite clear roughly what the interface layout should given the original requirement to have separate tabs for data loading, corrections and fitting.

There are still several additional fitting options that have been requested by the VESUVIO instrument scientists including the option to perform a fit in y-Space after a single mass peak is extracted from the time of flight spectra and fitting a multivariate Gaussian.

13.6 Conversion of FORTRAN routines

Work is still in progress to convert the remaining FORTRAN routines into MANTID algorithms, namely the Bayesian fitting routines Quasi and Stretch remain to be converted, this is described further in section 13.1.

The `cylabs` routine that previously provided the absorption corrections for cylinder samples has been converted to a Python algorithm: *CylinderPaalmanPingsCorrection* v2, version 1 of this algorithm is a wrapper around the FORTRAN routine that was converted using F2Py and has been kept for reference.

The `ResNorm` routine has been replaced by using standard MANTID fitting to fit the intensity and stretch factor in the instrument resolution to the vanadium reduction, this new routine outputs the intensity and stretch factors in the same format as the FORTRAN routine so maintains compatibility with the other Bayes routines, however the format of the fitted peak spectra differs but this is only intended as a check for the user that the fitting is working as intended.

This new fitting routine has been added as the *ResNorm* v2 algorithm, version 1 is an algorithm that simply wraps the existing FORTRAN code using F2Py.

References

[1] C. Andreani et al. "Measurement of momentum distribution of lightatoms and molecules in condensed matter systems using inelastic neutron scattering". In: *Advances in Physics* 54.5 (2005), pp. 377–469. DOI: 10.1080/00018730500403136. eprint: http://dx.doi.org/10.1080/00018730500403136. URL: http://dx.doi.org/10.1080/00018730500403136.

[2] C T Chudley and R J Elliott. "Neutron Scattering from a Liquid on a Jump Diffusion Model". In: *Proceedings of the Physical Society* 77.2 (1961), p. 353. URL: http://stacks.iop.org/0370-1328/77/i=2/a=319.

[3] S. J. Clark et al. "First principles methods using CASTEP". In: *Zeitschrift fuer Kristallographie* 220 (2005), pp. 567–570.

[4] Peter L. Hall and D.K. Ross. "Incoherent neutron scattering functions for random jump diffusion in bounded and infinite media". In: *Molecular Physics* 42.3 (1981), pp. 673–682. DOI: 10.1080/00268978100100521. eprint: http://dx.doi.org/10.1080/00268978100100521. URL: http://dx.doi.org/10.1080/00268978100100521.

[5] WS Howells and MTF Telling. "IRIS Data Analysis". In: *Technical Report RAL-TR-2000-004* (Jan. 2000).

[6] WS Howells et al. "The MODES user guide v3". In: *Technical Report RAL-TR-2010-006* (2010).

[7] *LAMMPS*. http://lammps.sandia.gov/. accessed 10/07/2015.

[8] Benjamin Lindner and Jeremy C. Smith. "Sassena X-ray and neutron scattering calculated from molecular dynamics trajectories using massively parallel computers". In: *Computer Physics Communications* 183.7 (July 2012), pp. 1491–1501. DOI: 10.1016/j.cpc.2012.02.010. URL: http://dx.doi.org/10.1016/j.cpc.2012.02.010.

[9] "Mantid: Manipulation and Analysis Toolkit for Instrument Data." In: (2013). DOI: 10.5286/Software/Mantid.

[10] J Mayers. "User guide to VESUVIO data analysis programs for powers and liquids". In: *Technical Report RAL-TR-2011-003* (Jan. 2003).

[11] J. Mayers and M.A. Adams. "Calibration of an electron volt neutron spectrometer". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 625.1 (Jan. 2011), pp. 47–56. DOI: 10.1016/j.nima.2010.09.079.

[12] J. Mayers, A.L. Fielding, and R. Senesi. "Multiple scattering in deep inelastic neutron scattering: Monte Carlo simulations and experiments at the {ISIS} eVS inverse geometry spectrometer". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 481.13 (2002), pp. 454–463. ISSN: 0168-9002. DOI: http://dx.doi.org/10.1016/S0168-9002(01)01335-3. URL: http://www.sciencedirect.com/science/article/pii/S0168900201013353.

[13] J Mayers and G Reiter. "The VESUVIO electron volt neutron spectrometer". In: *Measurement Science and Technology* 23.4 (2012), p. 045902. URL: http://stacks.iop.org/0957-0233/23/i=4/a=045902.

[14] S Mukhopadhyay. "How to Use Mantid for Low Energy Inelastic Neutron Scattering Data Analysis On Indirect Geometry Instruments". In: *Technical Report RAL-TR-2014-005* (2014).

[15] *nMoldyn*. http://dirac.cnrs-orleans.fr/plone/software/nmoldyn/. accessed 10/07/2015.

[16] H. H. Paalman and C. J. Pings. In: *Journal of Applied Physics* 625.1 (Aug. 1962), pp. 47–56.

[17] L C Pardo et al. "FABADA: a Fitting Algorithm for Bayesian Analysis of DAta". In: *Journal of Physics: Conference Series* 325.1 (2011), p. 012006. URL: http://stacks.iop.org/1742-6596/325/i=1/a=012006.

[18] D.S. Sivia et al. "Bayesian analysis of quasielastic neutron scattering data". In: *Physica B: Condensed Matter* 182.4 (1992). Quasielastic Neutron Scattering, pp. 341–348. ISSN: 0921-4526. DOI: http://dx.doi.org/10.1016/0921-4526(92)90036-R. URL: http://www.sciencedirect.com/science/article/pii/092145269290036R.

[19] J. Teixeira et al. “Experimental determination of the nature of diffusive motions of water molecules at low temperatures”. In: *Phys. Rev. A* 31 (3 Mar. 1985), pp. 1913–1917. DOI: 10.1103/PhysRevA.31.1913. URL: http://link.aps.org/doi/10.1103/PhysRevA.31.1913.