

Data Reduction for Direct Geometry Neutron Spectrometers

Michael A. Reuter & Andrei T. Savici

Version 1.1

October 4, 2013

1 Physics

When performing a neutron scattering experiment, one records data as either events (time of flight, wall clock time, and pixel ID tuples) or histograms (time of flight, pixel ID, number of counts tuples). Both of them have advantages and disadvantages, but neither is a format that is easy to understand from a physics point of view. What users like to see is either the *differential cross section* or the *dynamic structure factor*. The process to get to this format is called reduction¹.

For direct geometry spectrometers, the sample is placed in a monochromatic beam, with an incident energy E_i , and a flux (neutrons per time, per unit area) $\Phi(E_i)$. The experiments measure the partial current $\delta J_f(\mathbf{k}_f)$, the number of neutrons scattered per time in a solid angle $d\Omega$, having a final energy in an interval dE around the final energy E_f . The differential cross section is defined as the scattered current density normalized by the solid angle, the flux, and energy interval.

$$\frac{d^2\sigma(\mathbf{Q}, E)}{dEd\Omega} = \frac{\delta J_f(\mathbf{k}_f)}{\Phi(E_i)dEd\Omega} \quad (1.1)$$

The momentum transfer vector for the sample, \mathbf{Q} , is defined as $\mathbf{k}_i - \mathbf{k}_f$, while the energy transfer for the sample is $E = E_i - E_f$. Note that the energy and momentum transfer for the sample have a negative sign compared to the same quantities for the neutron.

¹This is in complexity, not size.

The cross section contains information about the sample physics, but is dependent on the experiment physics as well. The dynamic structure factor, $S(\mathbf{Q}, E)$, is a quantity that is sample dependent only. The relationship between the dynamic structure factor and the differential cross section is given by

$$\frac{d^2\sigma(\mathbf{Q}, E)}{dEd\Omega} = N \frac{k_f}{k_i} \frac{\sigma}{4\pi} S(\mathbf{Q}, E) \quad (1.2)$$

where N is the number of unit cells or molecules, k_i and k_f are the incident and final neutron wave vectors, and σ is the average scattering cross section of the unit cell/molecule that has the dynamical structure factor $S(\mathbf{Q}, E)$. Please note that the quantity that is additive is the differential cross section, meaning that if we have nuclear scattering, with average cross section σ_N , and magnetic cross section, with average cross section σ_M , we can write

$$\begin{aligned} \frac{d^2\sigma(\mathbf{Q}, E)}{dEd\Omega} &= \frac{d^2\sigma_N(\mathbf{Q}, E)}{dEd\Omega} + \frac{d^2\sigma_M(\mathbf{Q}, E)}{dEd\Omega} \\ &= N \frac{k_f}{k_i} \frac{\sigma_N}{4\pi} S_N(\mathbf{Q}, E) + N \frac{k_f}{k_i} \frac{\sigma_M}{4\pi} S_M(\mathbf{Q}, E) \\ &= N \frac{k_f}{k_i} \frac{\sigma_N S_N(\mathbf{Q}, E) + \sigma_M S_M(\mathbf{Q}, E)}{4\pi} \end{aligned}$$

There are four major steps required to obtain the differential cross section or the dynamic structure factor.

1.1 Detector cross-calibration using incoherent scattering

It is customary to check the efficiency variation of the various neutron detectors using incoherent scattering, usually a vanadium sample. For better statistics, when possible, some instruments can perform a measurement using a white (or quasi-white) beam.

An incoherent scatterer, like vanadium, scatters uniformly in all directions. The number of neutron counts should therefore be proportional to the solid angle of the detectors and to their intrinsic efficiency.

In practice, we restrict the number of events to a certain range in the energy domain (or wavelength, or time of flight). This allows elimination of artefacts, such as prompt pulse neutrons (very high energy neutrons, not stopped by the choppers).

Dividing sample data by the cross-calibration data takes care of the division by solid angle in formula 1.1.

1.2 Sample data reduction

Data loaded from the files is in time-of-flight units. The output we are interested in has units of energy transfer. In order to be able to convert between the two, one needs to know the incident energy. This can be found by measuring the neutron flux passing through two monitors. The differences in the peak times, and the positions of the monitors, give the velocity of the neutrons. This is done using the *GetEi* algorithm. This procedure also allows one to calculate an offset time, t_0 . If time=0 is when the proton beam hits the target, this offset time is the time required to moderate the neutrons. Therefore the first step in data processing is an adjustment for this offset. This can be done in two equivalent ways. One can subtract the t_0 from the time-of-flight coordinates for the neutron events, and keep the source of neutrons at the face of the moderator. The second way is to subtract the peak time in the first monitor, and move the origin of the neutrons from the moderator to the first monitor position.

The cross section expression in formula 1.1 contains a normalisation of the neutron counts by the incident flux. While we don't have an absolute number for $\Phi(E_i)$, we can use something proportional to it, which is either the proton charge, or the integrated intensity in a monitor.

During experiments, one finds a neutron background that is time independent. It is possible to subtract such background by finding a time-of-flight range that contains only this background, fit it to a constant, expand it to the whole time-of-flight range, and subtract it. A very important observation for this case is that the algorithms to subtract the time-independent background require the output data to be in histogram mode, so all information about individual neutron events is lost. For more informations, see *FlatBackground* algorithm.

While the overall efficiency of the neutron detectors is measured during cross-calibration, there is a variation that depends on the neutron energy - slower neutrons are easier to detect. There are two algorithms that correct for this effect. Data collected at SNS uses the *He3TubeEfficiency* algorithm, while data at ISIS uses *DetectorEfficiencyCor* algorithm.

If we want our data to be proportional to the dynamic structure factor instead of the differential cross section, we can multiply the neutron intensity by k_i/k_f (see formula 1.2).

At this point, we can ensure that the sample data is put on a regular energy transfer grid, using the *Rebin* algorithm.

Once the previous step is performed, it is possible now to divide by the dE term in formula 1.1. This is done by using the *ConvertToDistribution*

algorithm. Note that the output of this algorithm is not an event workspace any more. If events are still useful, make sure you are not using the distribution flag.

In order to take out the solid angle dependence, we mentioned that we can divide by the cross-normalisation incoherent data. Diagnostic tests have to be run on this data, to make sure we don't divide by 0. This means that some pixels will be masked. A more detailed description is found in Section 1.4.

After masking, the last step is an optional grouping of the data, in order to decrease data size.

1.3 Absolute units normalisation

Data analysed according to the previous section is proportional to the differential cross section or the dynamic structure factor. In order to get absolute units, one needs to compare the results to a known standard. This can be done using several methods, for example calculating the scattering cross section of an acoustic phonon. While this is the more accurate method, it is not easy to implement it in a general fashion. *DGSReduction* in MANTID implements an absolute normalisation using monochromatic Vanadium. The cross section of Vanadium as a function of energy ($\sigma(E_i)$) is well known. One measures it in the same conditions as the sample, and uses the same reduction parameters. One can then calculate the scattering per unit formula for the sample by just knowing the relative molecular mass of the sample and vanadium, and the mass of the two. When calculating the ratio of the total scattering intensities from sample and absolute normalisation vanadium, the reduction considers a weighted average. One can limit the use of intensities with very high or very low statistical weight. Since for a monochromatic beam measurement of Vanadium the intensity in each pixel should be constant (the solid angle was considered by cross-normalizing detectors), the corresponding cross section per solid angle is the total (coherent + incoherent) scattering cross section divided by 4π . The DgsReduction absolute normalization units are mbarns/sr/meV.

1.4 Detector diagnostics

Detectors with artificially high or low counting rates can introduce artefacts in the final data. Zero counts in the cross-normalisation data would yield division by zero errors. Therefore it is necessary to perform several diagnostic tests.

1. Once the vanadium data is integrated in the desired range, the first step is applying a hard mask. For example, some prefer to mask the top and bottom eight pixels in a tube with 128 pixels. This takes care of some edge effects.
2. The second step is finding detectors with a count rate too high, or too low.
3. Next, one looks at the median value of the vanadium counts. High or low outliers can be thrown out, then the median is recalculated. The user then sets a limit (as fraction of median count rate) of pixels to keep. There is also a significance test, the number of error bars away from the median so that pixels are kept.
4. If there is a second vanadium run, one can mask pixels for which the variation in intensity is much larger than the overall change.
5. In some cases, tests are run on the sample data as well. One can choose a background range, and mask pixels with too low or too high count rates, in a similar fashion to the vanadium diagnostic tests.
6. One can also integrate the entire sample data range to get the total counts for each detector pixel and mask pixels with too low or too high counts.
7. For position sensitive detectors, one can use the sample data to check if any tube is saturated. If there is a pixel with a raw count rate higher than a certain value, the entire tube is masked. This part is handled by the *CreatePSDBleedMask* sub-algorithm

All these tests are performed using the *DetectorDiagnostic* algorithm

2 MANTID Algorithms

This section will cover how the physics requirements detailed in Section 1 are implemented via algorithms in the MANTID code base.

The top-level workflow algorithm is called *DgsReduction*, and is responsible for the main orchestration of the data reduction process. This algorithm relies on six other workflow algorithms for execution. All the workflow algorithms in turn rely on core MANTID algorithms for the actual calculations. A workflow algorithm is only responsible for orchestrating the setup and execution of core MANTID algorithms. The *DgsReduction* workflow algorithm is

special compared to the other workflow algorithms because it handles the use of either files or workspaces for the various inputs. All other workflow algorithms only handle workspaces for inputs. The following shows the order of execution of the workflow algorithms if all input datasets are given and all corrections are requested. Core **MANTID** algorithms in the listings below are shown in *italics*. Each workflow algorithm will be broken down to show the core algorithms that it runs. Before getting into the breakdown, below is a listing of symbols used to identify the types of data that are being operated on by the algorithms.

SD This is the sample data

SDV This is the cross-calibration detector vanadium associated with SD

ISDV This is the integrated (processed) sample detector vanadium

SDV2 This is a second detector vanadium used for diagnostics

AU This is the sample for absolute units correction

ADV This is the detector vanadium associated with AU. Could be same as SDV.

IADV This is the integrated (processed) detector vanadium associated with AU

IAU This is the final integrated absolute units data

s A generic sample, either SD or AU

dv A generic detector vanadium, either SDV, SDV2 or ADV

idv An integrated dataset, either ISDV or IAU

tib A calculated time-independent background

d A generic dataset, possibly any of the capitalized entries

- **DgsReduction**
 - **DgsDiagnose** (SDV, SDV2, SD)
 - **DgsProcessDetectorVanadium** (SDV \rightarrow ISDV)
 - **DgsConvertToEnergyTransfer** (SD, ISDV)
 - **DgsAbsoluteUnitsReduction** (AU, ADV)

- *MaskDetectors* (SD \leftarrow IAU)
- *Divide* (SD, IAU)
- DgsDiagnose
 - DgsProcessDetectorVanadium (dv)
 - DgsProcessDetectorVanadium (SDV2)
 - DgsPreprocessData (s)
 - *Integration* (s \rightarrow total counts)
 - *Integration* (s \rightarrow background)
 - *ConvertUnits* (background)
 - *Divide* (background, dv, SDV2)
 - *DetectorDiagnostic* (dv, SDV2, total counts, background, s)
- DgsProcessDetectorVanadium (dv)
 - DgsPreprocessData
 - *ConvertUnits*
 - *Rebin*
 - *MaskDetectors*
- DgsPreprocessData (d)
 - *NormaliseByCurrent* or *NormaliseToMonitor*
- DgsConvertToEnergyTransfer
 - *GetEi*
 - *ChangeBinOffset* (s)
 - *MoveInstrumentComponent*² (s)
 - *LoadDetectorInfo*² (s)
 - *Rebin*³ (s \rightarrow bg)
 - *ConvertUnits*³ (s)
 - *Rebin*³ (s)
 - *ConvertUnits*³ (s)

²ISIS only

³SNS only, due to event workspaces

- *ConvertToDistribution*³ (s)
- *FlatBackground* (bg or s → tib)
- *ConvertToDistribution*³ (tib)
- *Minus*³ (s, tib)
- *ConvertFromDistribution* (s)
- *DgsPreprocessData* (s)
- *ConvertUnits* (s)
- *Rebin* (s)
- *ConvertUnits*⁴ (s)
- *He3TubeEfficiency*⁴ or *DetectorEfficiencyCor*⁵ (s)
- *ConvertUnits*⁶ (s)
- *CorrectKiKf* (s)
- *Rebin* (s)
- *ConvertToDistribution* (s)
- *Divide* (s, idv)
- *DgsRemap* (s)
- *Multiply*⁷ (s, scale factor)

- **DgsAbsoluteUnitsReduction**

- *DgsProcessDetectorVanadium* (ADV → IADV)
- *DgsConvertToEnergyTransfer* (AU, IADV)
- *Divide* (AU, $\frac{V_{mass}}{V_{rmm}}$)
- *Rebin* (AU)
- *ConvertToMatrixWorkspace* (AU)
- *DetectorDiagnostic* (AU)
- *MaskDetectors* (AU)
- *ConvertFromDistribution* (AU)
- *WeightedMeanOfWorkspace* (AU → IAU)
- *Divide* (IAU, $\sigma(E_i)$)
- *Multiply* (IAU, $\frac{Sample_{mass}}{Sample_{rmm}}$)

- **DgsRemap**⁸ (d)

⁴SNS only

⁵ISIS

⁶SNS only

⁷ISIS only

⁸Algorithms can be run in reverse order

- *MaskDetectors*
- *GroupDetectors* (**NOTE:** This assumes all pixels have same solid angle!)

3 User Interface

In addition to running the *DgsReduction* algorithm in Section 2 directly, a graphical user interface (GUI) is also available, via the *MantidPlot* analysis workbench. The GUI is launched from within *MantidPlot* from the *Interfaces* menu and selecting the *DGS Reduction* menu entry. The GUI can also be used to export a Python script once the appropriate fields are filled for a reduction pass. The GUI also allows one to see the parameters outlined in Section 2 that are read from the instrument parameter file.

Once launched, the user is presented with the following layout (Figure 3.1). The current instrument that is being requested for reduction is shown in the title bar of the GUI. To switch the reduction to a different instrument, use the *Tools* menus and select the *Change Instrument...* menu entry. Besides the tab that shows up on launch, there are three (or four)

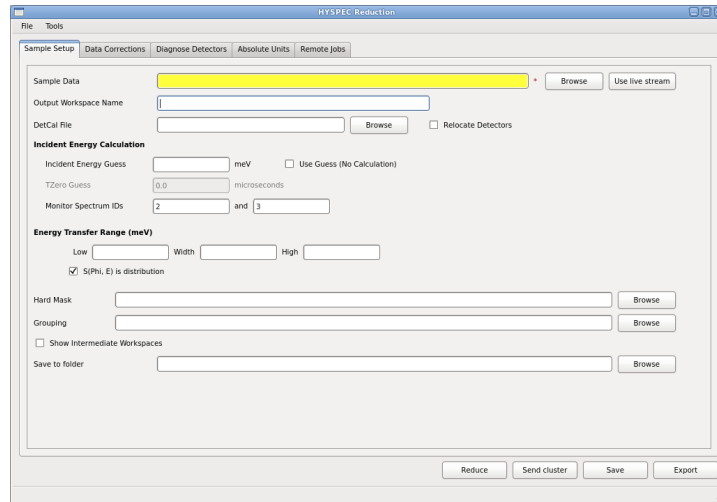


Figure 3.1: Sample Setup tab on the DGS Reduction GUI

other tabs for parameter entry. Each tab will be discussed individually in the next sections. To get start quickly, a reduction can be run with a minimal amount of information entered into the *Sample Setup* tab. A file can be entered into the *Sample Data* field. For *SNS* instruments, this is all that

is required. ISIS instruments will require an entry in the *Incident Energy Guess* field in order to run. Once this minimal information is entered, the *Reduce* button can be clicked to execute the reduction. On some computers, remote job submission is enabled. In that case a *Send cluster* button appears, and can be used instead of the *Reduce* one. More details are given in a later section. A Python script can be saved to a file by clicking the *Export* button and filling in a file name to the subsequent dialog. A configuration file may be save by clicking the *Save* button. This creates a XML file that can be loaded back into the GUI via the *File/Open...* menu.

3.1 Sample Setup

As described earlier, the *Sample Data* entry is the place to put a file (run) for reduction. The *Browse* buttons allows you to locate that file via the standard file finder dialog. If this field is not filled in, the background color is yellow indicating that this field needs to be filled in before running reduction. The interface allows to reduce multiple files simultaneously, either separately or together, or even a combination of the two. For the syntax to use, see <http://www.mantidproject.org/MultiFileLoading>.

On certain computers with adequate privileges, a *Use live stream* button appears for certain instruments. This allows on the fly reduction of the data, as it is collected.

The next field, *Output Workspace Name*, allows you to control the name of the workspace as it appears in the *MantidPlot* workspace listing. If nothing is entered, a default workspace name is created based on the sample file name without extension and the addition of this tag: '_spe'. The *DetCal File* entry is specific to ISIS instruments. Under normal circumstances, the detector calibration file is just the input sample file, but this does not have to be the case. This entry can be used to point to a different detcal file for use in the reduction. The *Relocate Detectors* checkbox controls a parameter that can relocate the detectors in the *LoadDetectorInfo* MANTID algorithm that is run.

Turning to the *Incident Energy Calculation* section of the UI, the first entry field is *Incident Energy Guess*. For ISIS instruments, if no entry is made, the background color is yellow indicating this field needs to be filled in before running reduction. SNS instruments do not require an entry since the reduction algorithm will use the *EnergyRequest* log to determine a guess. If a value is entered, then it will use that one instead of *EnergyRequest*. If you wish to not have the reduction calculate the incident energy, but just use the value provided, you can check the *Use Guess (No Calculation)*

checkbox. Checking this option enables the *TZero Guess* entry, which has a default value of zero microseconds and allows that parameter to be adjusted in accordance with the initial energy. The last set of entries deals with the *Monitor Spectrum IDs* used to perform the initial energy calculation. These two values are read from the instrument parameter file, so by default this is what the reduction algorithm uses. The entries allow you to override the IDs in order to use different monitors to perform the calculation. For the SNS instruments, CNCS and HYSPEC, the *Incident Energy Guess* is automatically used as the incident energy, and the time zero value is calculated via a formula found in the instrument parameter file.

The next section is the *Energy Transfer Range* entries. The first three entries allow you to specify the binning strategy for the x-axis of the output $S(\theta, \phi, E)$. If no values are entered, a default binning strategy is constructed by the reduction algorithm. Using the initial energy guess, E_i^{guess} , the low, width and high values are $-0.5E_i^{guess}$, $0.01E_i^{guess}$, $0.99E_i^{guess}$. If you want to fill in values, all the entries must be filled. The last option is the *S(Phi, E) is distribution* checkbox. This option causes the resulting data to be converted to a $\frac{d^2\sigma(\theta, \phi, E)}{dEd\Omega}$ or $S(\theta, \phi, E)$ histogram workspace by dividing by the bin width from the x-axis binning. Disabling this option stops this division and will keep an event workspace as the final result if the input was an event workspace and no time-independent background subtraction was requested.

The first of the last set of options allows one to apply a hard mask to the data by setting a file in the *Hard Mask* entry. A grouping file (sometimes know as a mapping file) which aggregates data into large blocks can be applied to the data by setting a file in the *Grouping* entry. The *Show Intermediate Workspaces* can be used to display workspaces created during the reduction process in addition to the output workspace.

Processed NeXus files are saved for each set of files added together in the directory specified by *Save to folder*. No files are saved for live reduction. In case of remote cluster submission, users should not specify folders where the cluster cannot write output files.

3.2 Data Corrections

This tab (Figure 3.2) deals with corrections that are applied to the sample data. The first set of entries deals with the *Incident Beam Normalisation*. This group of radio buttons allows one to pick the type of incident beam normalisation to perform. The default, *None*, performs no normalisation on the data. The *Current* option divides the data by its corresponding value of

the proton charge. The *Monitor 1* option uses the monitor specified in the instrument parameter file to integrate the time-of-flight range given in the *Integration Range* entries. The values in these entries are also read from the instrument parameter file. The radio buttons enforce that only one type of incident beam monitor normalisation can be performed on all input datasets.

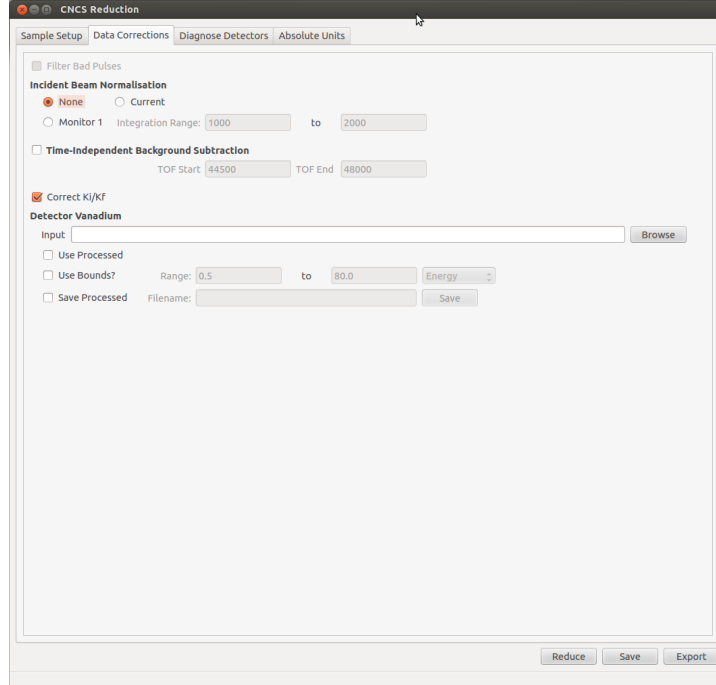


Figure 3.2: Data Corrections tab on the DGS Reduction GUI

The next section is deals with the *Time-Independent Background Subtraction* parameters. Here, the two entries represent the time-of-flight range in between which the background value will be determined. These parameters are also read from the instrument parameter file. The determined background will then be subtracted from the sample workspace.

The *Correct Ki/Kf* converts $\frac{d^2\sigma(\theta,\phi,E)}{dEd\Omega}$ to $S(\theta,\phi,E)$ and is applied by default.

The last section deals with the *Detector Vanadium* that is associated with the sample input data. The *Input* entry is where you can input a detector vanadium file (run) for use in the reduction. The *Browse* button will allow you to search for the file via the normal file dialog. The *Incident Beam Normalisation* chosen earlier will be applied if you do not use a pre-

processed file. The *Use Processed* checkbox allows you to tell the reduction that the file you are providing has already been processed and just needs to be used as is in the reduction. How you save a processed detector vanadium will be covered soon. If *Used Processed* is checked, all the following entries are cleared and disabled. Otherwise, the detector vanadium is processed according to the procedure in Section 2. The data is integrated according to the integration range given in the instrument parameter file. However, you can request alternative bounds by checking the *Use Bounds?* checkbox. This activates the *Range* entries, in which you can specify the bounds of the integration for the detector vanadium. The units on the bounds are assumed to be energy in the instrument parameter file. You can use the combobox at the end of the *Range* entries to set the units on the integration bounds. The available options are Energy, Wavelength and TOF (time-of-flight). Setting the units to anything other than time-of-flight will cause a unit conversion to occur on the detector vanadium data. The last options are used for if you are processing the detector vanadium for the first time. Checking the *Save Processed* checkbox allows you to save a NeXus file containing the processed detector vanadium. Using this checkbox activates the Filename entry so that you can provide an output file name for the saved file. If you do not put anything into the entry, a default file name will be created and looks like “<OutputWorkspaceName>_idetvan.nxs” and the file will be saved in the MANTID default output directory.

3.3 Diagnose Detectors

This tab (Figure 3.3) deals with performing diagnostic tests on the detector vanadium data being used in the reduction as well as the sample data for the reduction. On this tab, almost all of the entries and checkbox states are read from the instrument parameter file. The only exception is the *Detector Van 2* entry. These parameters are the ones that the reduction algorithm will use in the diagnostic tests unless the values are changed in the GUI.

The first section, *Detector Vanadium Tests*, deals with the parameters used for the two tests (*FindDetectorsOutsideLimits* and *MedianDetectorTest*) run on the detector vanadium data, which may already have a hard mask applied. All of the parameters in this section are read from the instrument parameter file. The *High counts* and *Low counts* parameters are used in the *FindDetectorsOutsideLimits* algorithm to determine hot and cold pixels in the data. All of the other parameters in this section are used in the *MedianDetectorTest* algorithm. Once the median is calculated, the

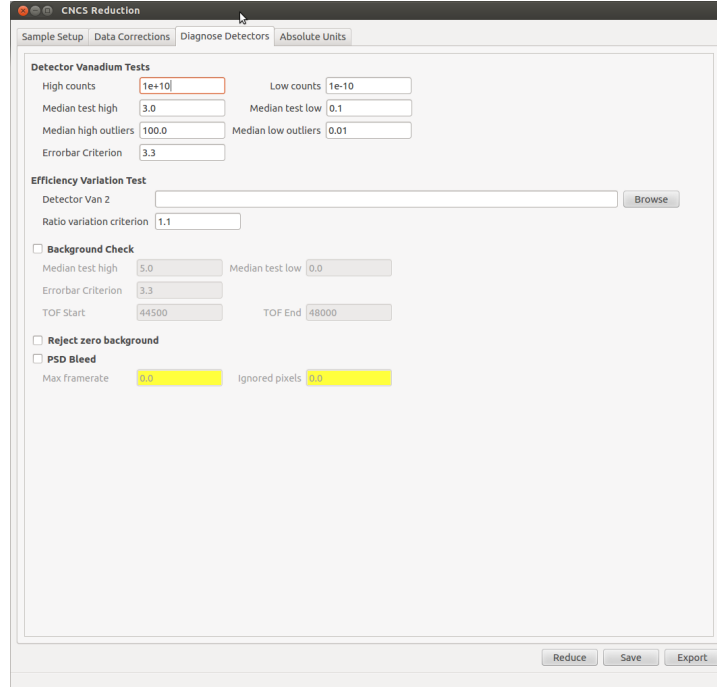


Figure 3.3: Diagnose Detectors tab on the DGS Reduction GUI

Median high outliers and *Median low outliers* entries are used as multipliers on the median and any pixel outside the bounds is masked and then the median is recalculated. Next, the *Median test high* and *Median test low* are used as multipliers on the median and any pixel outside the bounds is masked. The *Errorbar Criterion* entry is used in conjunction with previous two parameters, but checks the pixel signal minus the median.

The *Efficiency Variation Test* section uses a second set of detector vanadium data to compare against the original detector vanadium data. The *Detector Van 2* entry is where you specify this second detector vanadium file (run). The *Browse* button can help you find a file via the normal dialog. The *Ratio variation criterion* entry is the parameter used to check the consistency between the two detector vanadium datasets. This is used by the *DetectorEfficiencyVariation* test which is run after the two previously described tests are run on the second detector vanadium.

The *Background Check* checkbox activates the parameters below it and will perform a *MedianDetectorTest* on a sample workspace that has been incident beam normalised and integrated over the background range given

by the *TOF Start* and *TOF End* entry parameters. The other parameters are just in conjunction with the *MedianDetectorTest* are described previously. All the parameters in this section are read from the instrument parameter file.

The *Reject zero background* checkbox activates a *FindDetectorsOutsideLimits* test on a sample workspace that has been incident beam normalised and integrated over the entire range of data present (one number per detector pixel or total counts). This test is hard-wired to check for detector pixels having their total counts between 1×10^{-10} and 1×10^{100} .

The *PSD Bleed* checkbox triggers the activation of the *CreatePSDBleedMask* test. This test uses the *Max Framerate* and *Ignored pixels* parameters to determine which tubes are counting above the maximum frame rate. These tubes will be masked if they fail this check. The ignored pixels parameter tells how many pixels around the central region are ignored in the check.

3.4 Absolute Units

This tab (Figure 3.4) handles collecting the data and parameters for calculating the absolute units normalisation data. Checking the *Perform Absolute Normalisation* checkbox will activate this reduction procedure.

The first section, *Run Files*, handles setting the appropriate data files for the absolute units calculation. The *AbsUnits Vanadium* entry allows you to set the appropriate file (run) containing the absolute units sample data. As usual, the *Browse* button can be used to navigate to a file. The *Grouping File* entry allows you to group (map) the absolute units sample data during processing. This grouping is usually the same as the sample and sample detector vanadium processing, but this is not enforced. The *Detector Vanadium (AbsUnits)* entry allows you to specify a detector vanadium for the absolute units processing. Again, this detector vanadium is usually the one used for the sample data, but does not have to be.

The next section, *Integration*, deals with setting the parameters for the integration of the absolute units sample data. The *Incident Energy* entry allows you to specify the incident energy for the absolute units sample data. This is usually the same as the incident energy for the sample data, but does not necessarily have to be. The *EMin* and *EMax* entries for the *Energy Range* are where you specify the range over which to accumulate the values that will go into the final value for the absolute units normalisation. These two parameters are read from the instrument parameter file.

The following section, *Masses for Absolute Units*, manages setting con-

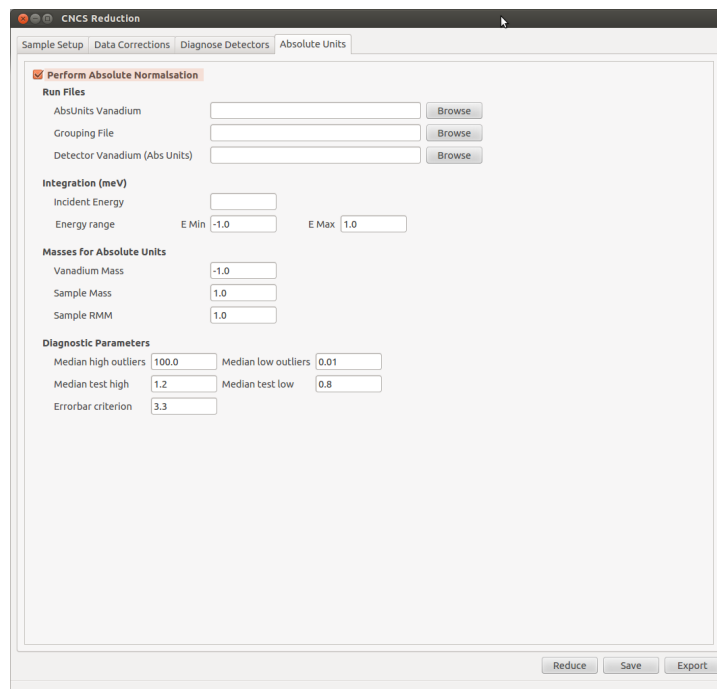


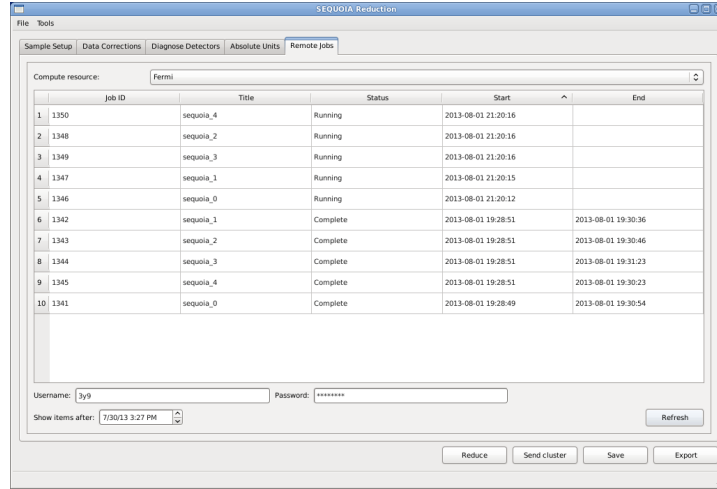
Figure 3.4: Absolute Units tab on the DGS Reduction GUI

stants that will be applied to the final absolute units normalisation result. The *Vanadium Mass* entry is where you specify the amount of vanadium (in grams) used as the absolute unit sample. This parameter is read from the instrument parameter file and if a -1.0 is shown, no good value is provided in the file. Since the absolute units is applied against the sample data, the number of moles of sample is calculated via the ratio of the *Sample Mass* and *Sample RMM* parameters. The first entry is the amount of the sample (in grams) being used and the second entry is the relative molecular mass of the sample. The default for both parameters is 1.0 which means there is no a priori knowledge of the sample data. For calculation of the number of moles of Vanadium, the atomic mass is hard-coded and the *Vanadium Mass* parameter is used.

The last section, *Diagnostic Parameters*, handles setting the quantities used in the same diagnostic tests that were discussed in Section 3.3. In this case, only the *FindDetectorsOutsideLimits* and *MedianDetectorTest* are run on the absolute units sample after detector vanadium correction (if applicable). All of the parameters in this section are read from the instrument

parameter file.

3.5 Remote jobs



	Job ID	Title	Status	Start	End
1	1350	sequoia_4	Running	2013-08-01 21:20:16	
2	1348	sequoia_2	Running	2013-08-01 21:20:16	
3	1349	sequoia_3	Running	2013-08-01 21:20:16	
4	1347	sequoia_1	Running	2013-08-01 21:20:15	
5	1346	sequoia_0	Running	2013-08-01 21:20:12	
6	1342	sequoia_1	Complete	2013-08-01 19:28:51	2013-08-01 19:30:36
7	1343	sequoia_2	Complete	2013-08-01 19:28:51	2013-08-01 19:30:46
8	1344	sequoia_3	Complete	2013-08-01 19:28:51	2013-08-01 19:31:23
9	1345	sequoia_4	Complete	2013-08-01 19:28:51	2013-08-01 19:30:23
10	1341	sequoia_0	Complete	2013-08-01 19:28:49	2013-08-01 19:30:54

Figure 3.5: Remote jobs tab on the DGS Reduction GUI

The *DGS Reduction* interface allows submission of jobs in parallel to compute clusters (currently only Fermi at ORNL). One job is created for each set of files that are added together. On first *Send cluster* use, the user is prompted to select number of nodes and cores, and to provide username and password. Once the jobs are submitted, one can check the status in the *Remote jobs* tab (Figure 3.5). User can select the most recent jobs by setting the value for *Show items after* and clicking the *Refresh* button.

4 Example scripts

The GUI described in Section 3 provides a way to export a script from the gathered information. However, one can put a script together by hand. MANTID uses Python as its scripting interface. The scripts described in this section will be constructed using the version 2 Python API and oriented towards running them via *MantidPlot*. Documentation on using the version 2 API can be found at http://www.mantidproject.org/Python_In_Mantid.

The simplest script that can be assembled for SNS:

```
config['default.facility'] = "SNS"
```

```
ws_name = "my_ws"
output = DgsReduction(
    SampleInputFile="CNCS_7860_event.nxs",
    OutputWorkspace=ws_name
)
```

The simplest script that can be assembled for ISIS is:

```
config['default.facility'] = "ISIS"
ws_name = "my_ws"
output = DgsReduction(
    SampleInputFile="MER06398.raw",
    OutputWorkspace=ws_name,
    IncidentEnergyGuess=18.0
)
```

Both scripts assume that the provided file is in the default search path for MANTID since full paths are not provided. The first line in both scripts is necessary to ensure correct functioning of the facility based switches in the reduction code and will override whatever has been set in the MANTID user preferences. From here, more options can be passed to the script to perform more reduction steps. The parameters are documented here: <http://www.mantidproject.org/DgsReduction>.

It is possible to just pass run numbers or an instrument name / run number combination to the script. In order for automatic file finding to take place, a couple of modifications need to be made to the script.

```
config['datasearch.searcharchive'] = "On"
config['default.instrument'] = "MERLIN"
```

This allows you to pass just run numbers to the *SampleInputFile* parameter as well as other **InputFile* parameters. Alternatively, you can omit the second line and hand something like *CNCS7860* to the *SampleInputFile* parameter.

You may run into the case where you have a dataset that needs some correction applied to it before handing it off for reduction. This system allows for this eventuality. In this case, you pass the appropriate workspace to the *SampleInputWorkspace* parameter. For SNS, this case will require that the monitors be loaded and passed to the *SampleInputMonitorWorkspace* parameter. For ISIS, this will depend on how/if *LoadRaw* is invoked. The following is an example taken from the *SEQUOIA* instrument at the SNS.

```

config['default.facility'] = "SNS"
config['datasearch.searcharchive'] = 'On'
dataset = "SEQ30541"
output = LoadEventNexus(dataset, OutputWorkspace=dataset,
                        LoadMonitors=True)
monitor = output[1]
valC3 = output[0].getRun()['Phase3'].getStatistics().median
output = FilterByLogValue(output[0],
                        OutputWorkspace=dataset+"_filt",
                        LogName='Phase3', MinimumValue=valC3-0.15,
                        MaximumValue=valC3+0.15)
ws_name = "my_ws"
output = DgsReduction(
    SampleInputWorkspace=output,
    SampleInputMonitorWorkspace=monitor,
    OutputWorkspace=ws_name,
    IncidentBeamNormalisation="ByCurrent")
summed = SumSpectra(output[0], OutputWorkspace=ws_name+"_s")
plt = plotSpectrum([summed], 0)

```