Mantid DGS Data Reduction Guide

M. B. Stone and A. T. Savici

Neutron Sciences Directorate

February-April 2012

v 5.0 May 2012

v 5.01 September 2012

v 5.02 December 2012

v 5.03 February 2013

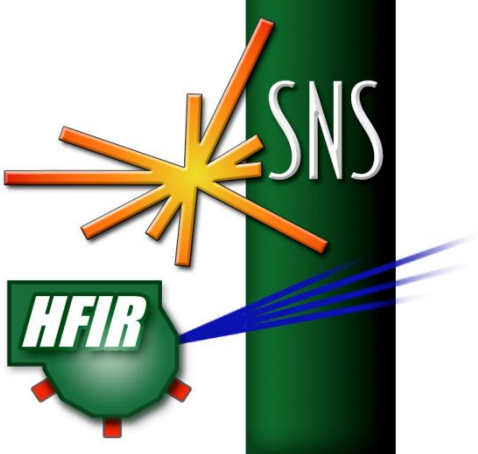

TABLE OF CONTENTS

Quick Start

If you are working on the SNS analysis machines, you will need certain paths set in your .bashrc file before you are able to use the reduction routines. Before you begin, please go through these Quick Start steps.

Navigate to the directory to which you would like to have the Mantid scripts copied.

Run the user_setup shell script from the terminal command line.

```
[XXX@arcs2 bin]$ /SNS/ARCS/shared/scripts/user_setup.sh
```

That is one types `/SNS/ARCS/shared/scripts/user_setup.sh` at the command prompt. The script will then copy the files in /SNS/ARCS/shard/mantidscripts to the current directory and will either output:

```
[XXX@arcs2 bin]$ /SNS/ARCS/shared/scripts/user_setup.sh
mantidnightly not in .bashrc file. Adding /opt/mantidnightly/bin to the
PYTHONPATH in the .bashrc file
```

Or

```
[XXX@arcs2 bin]$ /SNS/ARCS/shared/scripts/user_setup.sh
OK!
```

In the event that mantidnightly was added, the script will also source the ~/.bashrc file so that the user does not have to login/logout.

Introduction and Overview

This document describes how to use a standardized time-of-flight direct-geometry spectrometer (dgs) data reduction routine based upon functions within the Mantid software application. This routine is written to be used with the dgs instruments at the Oak Ridge National Laboratory's Spallation Neutron Source: ARCS, CNCS, HYSPEC, and SEQUOIA. The intention of this routine is to provide a consistent and expandable cross-instrument mechanism for data reduction. Future developments will include additional time-of-flight spectrometers at the SNS, and potentially at other facilities, as well as additional features as requested by the users of this software. A GUI is also being planned for development within the Mantid platform.

We recommend using the example files and working with the local instrument contact as the best mechanism for learning how to use the data reduction routine.

The reduction routines described here are executed from within the Mantid software application, from a Linux command line, or from a Python command line. The user generates an XML text file that sets keywords and variables for data reduction. This text file is referred to as the "reduction.xml" file in this document and provides a mechanism for batch processing of data from time-of-flight histograms to energy transfer histograms appropriate for further data visualization and analysis.

This document is organized to describe how to generate and use a reduction.xml file. Section one describes the reduction.xml file and how to generate it. Section two describes how to evaluate this file to reduce ones

data, and Section three describes the available output of the data reduction routines. In section four, we describe in further detail the single-crystal capabilities available in these reduction routines. In section five, we describe additional capabilities in the routine which are under development. Appendix 1 describes all of the available keyword parameters in the reduction routine. Standard reduction.xml files and examples are included in appendix 2. A typical default.py file is shown in appendix 3 for the ARCS, CNCS, HYSPEC, and SEQUOIA instruments. In Appendix 4, we describe known bugs with the current software.

If additional features are requested or programming errors are found, please email these to Matthew Stone and Andrei Savici at (5us@ornl.gov and 3y9@ornl.gov).

Updates in V 5.0

The mantid user's .bashrc file

Please use the export statement:

```
export PYTHONPATH=$PYTHONPATH:/opt/mantidnightly/bin/
```

in your .bashrc file. This command should be placed after any `if..fi` statements in your .bashrc file.

Grouping

Standard grouping files for '8-pack' style instruments are no longer need to be made prior to running the reduction routines. Instead, these files are made during the reduction process if they are required. Standard values for the grouping keyword are 'powder' and 'yvalXxval' where yval is in the set {1,2,4,8,16,32,64,128} and xval is in the set {1,2,4,8}. For more complicated detector grouping, there is also the option of generating a grouping file and then using the name of this file for the keyword value.

Orientation vectors/matrices

A lattice and ub keyword have been added to the reduction routines. These keywords allow one to include the crystal orientation in the reduction process. In addition, the mdnxs file type is now included under the save keyvalue. This will save a multi-dimensional nexus file. The lattice and ub keywords are described further in Appendix 1 and Section 4.

Updates in V 5.01

Powder Grouping

Powder grouping now makes use of the GenerateGroupingPowder command available in Mantid. Keywords for binning remain the same.

Updates in V 5.02

iofq and iofe output files now account for NaN's in the powder datasets correctly.

Time synching of logs

Logs of numeric series values are now synchronized to the accumulating proton charge. This is done using the ShiftTime command

```
def ShiftTime(WName,lg_name):
    """
    shift the time in a given log (lg_name) of a workspace (WName) to match the time in the
proton charge log"
    """
    H_IN = mtd[WName]
    PC =  H_IN.getRun()['proton_charge'].firstTime()
    #print "P="+str(PC)+"\n"
    P =  H_IN.getRun()[lg_name].firstTime()
    #print "P="+str(P)+"\n"
    Tdiff = PC-P
    Tdiff_num = Tdiff.total_milliseconds()*1E-3
    #print "Tdiff="+str(Tdiff_num)+"\n"
    ChangeLogTime(InputWorkspace=WName, OutputWorkspace = WName, LogName = lg_name, TimeOffset
= Tdiff_num)
```

Updates in V 5.03

Accounting for changing configuration files

During the lifetime of an instrument, the configuration file can potentially change. This information is stored in the NeXus files associated with each measurement. Therefore, the instrument configuration is automatically accounted for when examining individual datafiles or vanadium calibration files. There are times when one wants to use no vanadium calibration file, see example 1b, Generation of a unitary calibration file. For this case, one must load the correct configuration file for the data that will be reduced. A date field, "datadate" has been added to allow the user to enter the date of the measurements and thus choose the correct configuration file. An example of this is shown in example 1c at the end of this document. The format of datadate is a hyphenated string of year-month-day: datadate="2012-10-30". If the datadate keyword is not used, then the most recent configuration of the instrument is used. The configuration file which is used in this situation is also written to the summary text file.

Multiple filters of log values

An individual or a set of log values can be set with acceptable ranges. Data outside of these ranges will not be included in the reduced output file. This is accomplished using the keywords filternames, filtermin and filtermax. Example 21 illustrates how to use these keywords.

1. The reduction.xml file

The reduction.xml file uses keywords and instrument defaults to generate a vanadium calibration file, generate a detector mask, and reduce a series of data files into energy transfer histograms appropriate for further analysis. This file is an xml file and follows the conventions of use for this type of file.

The reduction.xml file consists of three sections: defaults, calibration and mask, and scan(s). We describe the three portions of the file in the corresponding sections of this document. The three sections are separated with xml style labels as shown in the following example.

```
<?xml version="1.0" ?>
<dgsreduction>
  <!-- DEFAULTS section -->
  <defaults
    Instrument="ARCS"
    filterbadpulses = "False"
  />
  <!-- CALIBRATION AND MASKING section -->
  <calibration processedfilename = "vanadium_23101_3mono.nxs" units="wavelength"
normalizedcalibration = "True">
    <vanruns>23101,23102-23104</vanruns>
    <vanmin>0.35</vanmin>
    <vanmax>0.75</vanmax>
    <mask algorithm="BankTubePixel" pixel="1,2,3,4,5,6,7"/>
    <mask algorithm="angle" twothetamax="2.5"/>
    <mask algorithm="MedianDetectorTest" SignificanceTest="4.0" LowThreshold="0.3"
HighThreshold="2.0"/>
  </calibration>
  <!--scan(s) section -->
  <scan IPTS="5824" runs="22831" t0="4.0" Efixed="90" kioverkf="1" grouping="2X1"
AngleMotorOffset="5.6" save="nxspe4,nxs5,qjpg,phx" friendlyName="test1"/>
</dgsreduction>
```

These sections can be in any order, although for clarity we suggest the standard order shown here. The sections cannot be nested within one another. Information within these sections consists of a series of keywords and values separated by an "=" sign. The format of the defaults and calibration and mask sections is to have these values listed on single lines and separated by carriage returns. Each <scan> call in the scan(s) section corresponds to one call to the data reduction routines. There are different types of scans which are available for a given ascan. These are described in the description of the scantype keyword in Appendix 1.

There is a fourth location for keyword parameters. These are located in a file called XXXdefault.py where XXX is the instrument name (i.e. arcsdefault.py, cncsdefault.py, sequoiadefault.py, and hyspecdefault.py). An example of this default file is shown in Appendix 3. These parameters are typically set by instrument scientists of the respective instruments. These files are typically stored in the /SNS/XXX/shared/ directory where XXX is the instrument name. This file may also be located in the directory you are working in for the reduction. If you are running the reduction code using an external machine (i.e. not one of the instrument or reduction Linux machines within the neutron sciences directorate), you will need to copy this file to the local directory within which you are working.

Keyword values can be reset or set in any of the sections. Values are overwritten as the reduction code traverses from the XXXdefault.py file, to the defaults section, to the calibration and mask section to the scan section(s). A comparison of the examples in Appendix 2 illustrates how defaults can be set for all of

the individual data reductions. If a keyword value is not set, then the reduction routine will use the default value.

Formatting

The reduction.xml is an ascii xml file. There are portions of this file which are case sensitive. Keywords are case sensitive, and are typically lowercase. Keyvalues are also case sensitive. Examples of case sensitive keyvalues are: instrument, goniometermotor, any filenames, and binning values. Calls to the <scan> element of the xml file within the scans section can be set up as multiple lines. Each individual <scan> line is treated as a separate reduction process. Lines with no text, or only whitespace are ignored by the reduction routines and can be used to improve the layout of the reduction.xml file. One can use the <!-- --> elements to enclose commented text within the reduction.xml file. Single or double quotation marks may be used in setting keyvalues.

The <defaults> section

The defaults section is used to set default keywords to be used for all of the data being processed as well as the vanadium calibration files being generated. For example, one may set the kioverkf keyword in the defaults section, and its value will be applied to all data reduction calls which do not have this parameter set.

Required input

One must define the instrument with the "instrument" keyword, i.e.

```
<defaults>
instrument = "ARCS"
</defaults>
```

Optional input

Optional input in this section includes being able to set all of the keyword parameters to be used throughout your reduction routine. For example, `filterbadpulses = "False"` is commonly included in this section such that it would be input as:

```
<defaults>
instrument = "ARCS"
filterbadpulses = "False"
</defaults>
```

The <calibration> section

The calibration section is used to generate the vanadium calibration and masking file (this information is stored in a single file). If the 'processedfilename' already exists, then the routine will simply load this file for use in the masking and calibration of data. If the 'processedfilename' does not exist, then the routine will generate this file. The data from 'vanruns' are loaded, and integrated between 'vanmin' and 'vanmax' for the given 'units'. The mask algorithms are then applied. Current available mask algorithms allow one to mask over banks, tubes, and pixels ('BankTubePixel'), ranges of scattering angle in two theta ('angle'), and the 'MedianDetectorTest' available in Mantid. If the 'normalizedcalibration' keyword is set to True, then the vanadium calibration file will be normalized to fluctuate about a value of one. Please note that the order of masking will affect the resulting mask. We suggest masking detectors using the MaskBTP algorithm AFTER other algorithms are called.

Required input

There is no required input in the calibration and mask section. With no input a calibration/mask file will not be used in the reduction of data. If you would like to have the data masked and calibrated appropriately, then you must set the "processedfilename" keyword to indicate which calibration/mask file should be used in the reduction of the data. When no calibration/mask file is applied, then the solid angle correction will not be included in the reduced data.

The <scan> section

The scan section allows one to set details of the data reduction process. Each scan element will describe one distinct set of commands for the runs listed by the runs keyword. Lines of this section may be long if all of the keywords are set to their non-default values. Three scan types are available for a scan. These are single (default), step, and sweep. The use of these scan types is illustrated in the examples listed in Appendix 2.

- Single mode will combine all runs in a single data reduction.
- Step mode will reduce each individual run of the runs listed.
- Sweep mode will combine all of the runs together and separate and reduce the combined dataset according to the independent variable and its range chosen.

Required input

There is no required input in the data section. However, without any input, no data will be reduced.

2. Evaluation of the data reduction routines

The data reduction routines can be run from within a Python terminal or from a Linux command line.

Evaluation within a Python terminal of an analysis Linux machine

Set-up

In order to access correct Mantid directories, one must add a path line to their .bashrc file. This typically only needs to be done once.

- Open a text editor (Applications -> Accessories -> editor of your choice) and open your .bashrc file. This file is located in the directory /SNS/users/youruserid, where youruserid is your user id.
- Add the following text at the end of the .bashrc file:
  - `export PYTHONPATH=$PYTHONPATH:/opt/mantidnightly/bin/`
- Save the file and close the text editor.

Execution

- Open a new command shell (right click on desktop and choose 'Open in Terminal' or Applications-> System -> Terminal)
- Navigate to the shared directory of the IPTS directory you are working within:
  - `cd /SNS/INSTRUMENT/IPTS-XXXX/shared`

- The local contact will place any needed files (calibration files and python code) within this shared directory.
- Open a text editor and generate the reduction.xml file. Save this file within the shared directory.
- Open a python terminal from the shell. The terminal ipython has been working well: type `ipython` in the command shell.
- Run the reduction code within the python shell: `run -i dgsreductionmantid reduction.xml` , where reduction.xml is your reduction xml filename.
- If files are not being accessed properly, then one may need to tell python to add the appropriate subdirectories to the path using the config.appendDataSearchDir command e.g.
    - `from mantid import *`
    - `config.appendDataSearchDir('/SNS/SEQ/shared/2012_A/V_files')`

Evaluation within a Linux shell of an analysis Linux machine

Set-up

In order to access correct Mantid directories, one must add a path line to their .bashrc file. This typically only needs to be done once.

- Open a text editor (Applications -> Accessories -> editor of your choice) and open your .bashrc file. This file is located in the directory /SNS/users/youruserid, where youruserid is your user id.
- Add the following text at the end of the .bashrc file:
    - `export PYTHONPATH=$PYTHONPATH:/opt/mantidnightly/bin/`
- NOTE: mantidnightly is updated nightly. If code does not run properly, one may need to use the latest full release of mantid which is set instead with
    - `export PYTHONPATH=$PYTHONPATH:/opt/Mantid/bin/`
- Save the file and close the text editor.

Execution

- Open a new command shell (right click on desktop and choose 'Open in Terminal' or Applications-> System -> Terminal)
- Navigate to the shared directory of the IPTS directory you are working within:
    - `cd /SNS/INSTRUMENT/IPTS-XXXX/shared`
- The local contact will place any needed files (calibration files and python code) within this shared directory.
- Open a text editor and generate the reduction.xml file. Save this file within the shared directory.
- Run the reduction code from the Linux sell by typing at the command line:
    - `python dgsreductionmantid.py reduction.xml.`

3. Available output files

Several output files available can be generated by the reduction routines. These files are chosen by setting the save keyword in the data section. We describe here the available output and references to the formatting of these files.

The summary file

The summary file is a text file which is created during the course of data reduction. It contains a description in prose of what was done during the data reduction process. If the vanadium calibration file was also generated during the data reduction process, then this information is also included in the summary text file. There are some options as to how summary files are named (please see friendlyname and friendlynamelogs keywords descriptions), but they always contain the text _summary.txt. Summary files for data processed using step or sweep mode are cumulative, that is, the summary file is saved for each individual reduced data file, but it gets appended to as one traverses each of the runs being reduced. The summary file is saved in the firendlyname subdirectory. The following is an example of a summary file.

```
-----------VANADIUM CALIBRATION AND MASKING-----------
Calibration loaded from
/SNS/users/5us/mantidscripts/v1/test1/vanadium21106_yesmediantest5_normcalib.nx5
-----------DATA REDUCTION-----------------------------
Loaded data run from /SNSlocal/ARCS/IPTS-5368/0/22663/NeXus/ARCS_22663_event.nxs
Incident energy is calculated from monitor data.
Ei (sum of monitors) = 90.4334027692 meV, t0 = 19.0718124122 microseconds
No time-independent background subtraction performed.
Data normalized by proton charge (555.970871253 micro-Ah)
Data corrected for He3 Tube Efficiency.
ki/kf factor has been applied to the data.
Data binned with emin=-86.5, emax=86.5, ebin=1.0meV.
Data converted to differential cross section by dividing by the energy bin width.
Data have been normalized and masked by the calibration file.
No goniometer set.
Data have been saved as a .nxspe file, FILENAME=TiO2_90meV_400C/ARCS22663_TiO2_90meV_400C.nxspe
```

The vanadium calibration and mask output file

If within the calibration and masking section, the file corresponding to the processedfilename keyword exists, then the data reduction routine will simply read this file. If it does not exist, then the data reduction routine will generate this file based upon the other key values set in this section. The calibration file is a Nexus file. The Nexus file is saved using the SaveNexus command in Mantid (http://www.mantidproject.org/SaveNexus). The vanadium calibration file can be loaded and viewed using the MantidPlot application.

The nxspe file

One of the file types available for writing the reduced data to is an .nxspe file. The .nxspe file is saved using the SaveNXPSE command in Mantid (http://www.mantidproject.org/SaveNXSPE). The values of fixed, psi, ki_over_kf_scaling, and detector information (as requested by the SaveNXSPE command) are included with this file. The nxspe file can be read using DAVE-Mslice software.

The nxs file

One of the file types available for writing the reduced data to is a .nxs file. This is a NeXus file type that is saved using the SaveNexus command in Mantid (http://www.mantidproject.org/SaveNexus). These data can be loaded and visualized using the MantidPlot software application. The complete history of the data reduction is also included in this file. One can use this with MantidPlot to generate additional data reduction scripts in MantidPlot.

The spe file

One of the file types available for writing the reduced data to is a .spe file. This is a text file which describes the scattering intensity and error as a function of energy transfer and detector position. This filetype is used with Mslice, DAVE-Mslice, and in generation of Horace files.

The phx file

One of the file types available for writing the reduced data to is a .phx file. This is a text file which describes the detector positions. This filetype is used along with the .spe file in Mslice, and DAVE-Mslice.

The par file

One of the file types available for writing the reduced data to is a .par file. Par files are used in generation of Horace files.

The iofe file

The iofe file is a text file of Inensity as a function of energy transfer generated directly after the data are reduced. The data are integrated over all wave vector transfer, q, values. NOTE: If you use any detector grouping other than grouping='powder', it will take a very long time to generate the iofe file.

The iofq file

The iofq file is a text file of Intensity as a function of wave vector transfer generated directly after the data are reduced. The data are integrated over all energy transfer values. NOTE: If you use any detector grouping other than grouping='powder', it will take a very long time to generate the iofq file.

The mdnxs file

The mdnxs file is a nexus file which includes the vector orientation of the crystal. This file type can be loaded into Mantid and merged with other similar files for navigating a multi-dimensional workspace of data.

4. Single crystal details and capabilities

The single crystal data file of choice currently is the .nxspe, and phx and spe files. These files can be visualized using DAVE-Mslice (http://www.ncnr.nist.gov/dave/)and Horace (http://horace.isis.rl.ac.uk/Main_Page) respectively. No further generation of files is needed to visualize data in DAVE-Mslice beyond the creation of the .nxspe files. One must generate appropriate .SQW files from the .spe and .phx files to use Horace. Typical single crystal measurements are made by rotating the crystal through multiple angles with all other parameters fixed. To reduce these data, one may use either step, sweep, or single for the scantype. Note that step mode does not average multiple files together, and sweep mode does not yet work when the motor is held in position for an entire measurement. One can read the motor angle directly from the motor logs, or one can set this angle to a default value. Up to 6 goniometer angles can be given. The goniometer angles are set using the SetGoniometer function in Mantid. The keywords available for setting the goniometer are goniometermotor, goniometermotoroffset, goniometermotordirection, goniometermotoraxis. Often the XXXdefault.py instrument default file will contain default information for the goniometermotoraxis and goniometermotordirection. If no offset and no

motor name are listed, than no goniometer will be set. If no motor name is listed, and the number of elements of offset, direction and axis are the same, then those values will be set for the goniometer. If the number of elements of motor, direction, and axis are all >= 1 and all equal, then the angle set for the goniometer is the sum of the offset angle listed and the angle value in the motor log. Further details regarding the motor keywords are listed in Appendix 1 and in the example files.

The multi-dimensional workspace option within Mantid can now be used with these reduction routines. One must choose 'mdnxs' for the save file type. One must also input either the UB matrix or the lattice parameters and dave-mslice orientation vectors. This is done with either the 'lattice' or 'ub' keywords as described in Appendix 1.

5. Binning

There are two choices one must make when binning (i.e. histogramming) data. We describe these options and their consequences here

Energy binning

Energy bins are set from emin to emax with a step size of ebin. The emin value is the minimum energy of the lowest energy bin, the emax value is the maximum energy of the highest energy bin, and ebin is the width of each individual energy bin. For example, emin='-10.5' emax='10.5' ebin='1' would generate histrogramed data at energy transfer values of -10, -9, -8, …,0,…8,9,10 meV. Figure 1 illustrates how the binning is performed. There is no check for if the chosen binning values are appropriate.

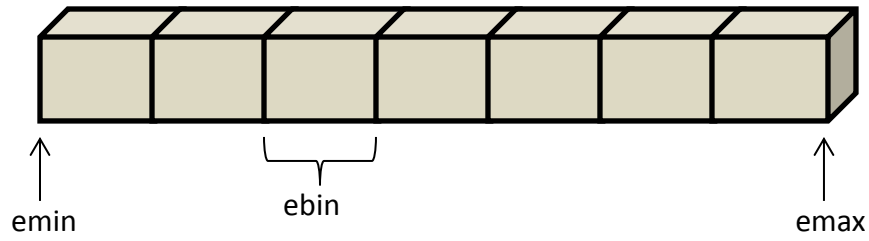

Figure 1 Illustration of how energy binning is performed within the reduction software. Returned energy values are the bin centers.

The default values for emin, emax and ebin are based upon the incident energy, Ei, and are respectively emin = -0.5*Ei, emax=Ei, ebin = .015Ei.

Detector binning

Detector binning refers to how one may average over neighboring pixels in detector space during the data reduction. Two types of binning are allowed in the data reduction routine: powder binning and position binning. Please note that the overall intensity scale will change when using different types of binning.

Powder binning

Powder binning averages over common values of 2theta in detector space. This type of binning dramatically reduces output file size for nxspe, spe and phx files. This type of binning is used when grouping='powder' is set within the defaults or scan sections of the input xml file. There is a default value of the binning in two-theta detector space. One may adjust this parameter by using the powderanglestep

keyword. The intensity scale for powder binning is different than that used for position binning. The intensity scale for powder binning does not change as one adjusts the powderanglestep parameter.

Position binning

Position binning averages over neighboring pixels vertically and horizontally for a given detector pack. Currently this type of binning is used for 8-pack detector style instruments. This type of binning can reduce the output file size by reducing the number of effective detectors. One needs to use the grouping keyword to choose a grouping file. The keyvalue for this keyword refers to the type of grouping. For example grouping='2X1' refers to a grouping file with 2 vertical pixels being averaged in each detector tube (from 128 pixels to 64 pixels), and no horizontal grouping. These standard grouping files for '8pack' style instruments are generated by the reduction routine. Grouping files are named using the convention INSTRUMENT_YbyX_grouping.xml, where INSTRUMENT is the instrument name listed under the instrument keyword and YbyX is the grouping in detector space written as "2X1", "2X2", "4X2", etc. For example ARCS_2X1_grouping.xml is a grouping file name. One can also generate a grouping file for more complicated binning and then use the name of this grouping file as the keyvalue for the grouping keyword. For ARCS, SEQUIOA, and CNCS, it is recommended to only use binning of 1X1, 2X1 or 4X1. 1X1 is the default binning.

Logvalue Binning

For the 'sweep' scantype, one can bin data using logvalues. For example, this can be used when one acquires data while temperature or a motor angle is being changed. One can then bin by the log values and reduce these events into separate reduced datafiles. The keywords logvaluemin, logvaluemax, logvaluebin and logvalue are described in AppendixI and an example of this capability is shown in Example 12. The logvalue binning follows the same rules as described in Figure 1 for energy binning. When one uses logvalue binning one must use the friendlynamelogs keyword set to the same logvalue so that the written filenames will be different for each of the binned and reduced output files.

6. Capabilities under development

Masking based upon data files

Currently masking is only set based upon the vanadium calibration file. We would like to extend this to include a union of this mask with a mask based upon the individual data files being reduced. A user defined hard mask in addition to the vanadium and sample masks is also usefull.

Other file types

Other file types being considered to be made available for writing include a text file for powder data as S(Q,w).

Automatic or description of how to convert to absolute units.

A keyword to change between normalization per Coulomb OR normalization per micro-Amp hour. Currently normalization is to the number of micro-Amp hours.

For iofq and iofe text files, one should add code to first convert single crystal grouping to powder grouping, and then generate the iofq and iofe datasets. This is because these routines use the Sofqw3 mantidplot routine which takes a very long time for anything but powder data.

Appendix 1 – keywords

Keywords and keyvalues are case sensitive. They are set as a space delimeted list within the defaults, calibration, and scan sections of the xml file. They are set using a keyword=”keyvalue” sequence. The quotation marks can be single or double quotes.

Keyvalues for Boolean keywords can be set by, yes, true, t, 1, y, tru, tr, or y (case insensitive). Everything else is false.

calce – Set to true to fit the beam monitors for the incident energy. If the efixed keyvalue is set, then calce will use that value as the starting point for the fitting. If efixed is not set, then calce will use the value that was saved to the data file during the measurement as the starting point for the fitting. Set typically line by line in the data section, in the defaults section, or in the instrument defaults file. If calce is set to false, one must set the efixed value. Examples: `calce='true'` or `calce='false'`.

datadate – This keyword is set if one needs to choose an older configuraition file when NOT using a vanadium calibration measurement. The date should be enetered as hyphenated year-month-day. If no datadate keyword is given when not using a vanadium calibration file, then the most recent instrumemt configuration file is used. The datadate keyword is enetered in the <calibration> portion of the calibration and masking section as shown in Example 1c. Example: `datadate='2012-10-30'`.

datapath – Add this keyword and value to the scan element to add a particular path to search for data files. Example: `datapath = "/SNS/users/5us/tempfiles/newdata"`.

ebin – Bin size in energy transfer in meV units for energy binning. Default value in the code is set to choose ebin based upon 100 steps between emin and emax. Set typically line by line in the data section or in the defaults section. Example, `ebin='0.5'`.

efixed – The incident energy in meV to be used for the data reduction. If calce is set to false, then efixed must be set and will be used without fitting the beam monitors for the incident energy. Set typically line by line in the data section or in the defaults section. Exapmle: `efixed='12.7'`.

emax – The maximum energy transfer in meV for the energy binning. The default value in the code is set to choose emax as 1.0 times the incident energy. Set typically line by line in the data section or in the defaults section. Example, `emax='10'`.

emin –The minimum energy transfer in meV for the energy binning. The default value in the code is set to choose emin as -0.5 times the incident energy. Set typically line by line in the data section or in the defaults section. Example, `emin='-10'`.

filterbadpulses – Setting this keyword to true will apply the filterbadpulses mantid function to the data. Set typically in the defaults section. the filterbadpulses function removes any events that were measured at

a beam power that is less than 95% of the average beam power for a given dataset. Examples: `filterbadpulses='False'` or `filterbadpulses='True'`.

Filtermax – This value is used with filternames and filtermax to set a bandpass filter for the filternames parameter. The order of values for filtermax must match those listed in filternames. See example 21. Example `filtermax='102,2302.04'`.

Filtermax - This value is used with filternames and filtermin to set a bandpass filter for the filternames parameter. The order of values for filtermin must match those listed in filternames. See example 21. Example `filtermin='100,1802.04'`.

filternames – This keyword consists of a comma delimeted list of log value names. The keywords filtermin and filtermax set the range of the band-pass filter. Any value stored in a sample log can be set with this keyword. See example 21. Example `filternames='SensorB,Phase3'`.

friendlyname – This will be the name of the generated subdirectory. This subdirectory will contain the files that were generated as listed in the file keyvalues. The text for friendlyname must follow the conventions for an operating systems directory names. If no friendlyname is given then no subdirectory is created, and the files are stored in a subdirectory with the name of the instrument. This value is typically set line by line in the data section. Example, `friendlyname='rotation_T50K'`. Examples of the use of friendlyname and friendlynamelogs are given in Appendix 2.

friendlynamelogs – Using this keyword allows one to include the mean value of a parameter of a logfile as a portion of the filename. The mean value is included to two decimal places, and decimal points are changed to the character 'p' for use with Horace. Multiple values can be listed for friendlynamelogs. For example, one can include the rotation angle position of a motor, or the mean temperature of a measurement or both. Any variable which is stored in the sample logs. Note that friendlynamelogs is often set in the instrument defaults file. In order to override this default set `friendlynamelogs=''`. If one is using the 'sweep' scantype with logvalue filtering, then one should use friendlynamelogs with the same logvalue name. This will prevent separate reduced files from having the same filename and being continuously overwritten as additional files are produced. Example, `friendlynamelogs='CCR13VRot'`. NOTE: if you use the motor angle in the filename and you add an offset, then the filename will be written with the motor angle, not the offset+motorangle.

goniomtermotor– if multiple goniometer angles are used, then each of the keyvalues should be input in the same order for the three individual keyword parameters. Set typically line by line in the data section or in the defaults section. This is the name of the goniomtermotor which is to be read by the data reduction routines. This will have the same name as that used by the DAS. Set typically line by line in the data section or in the defaults section. Example, `goniomtermotor='CCR13VRot'`.

goniometermotoraxis - These are the vector directions of the goniometer angles. Coordinate system is the NeXus standard. The default vector is the vertical axis of the sample being the axis of rotation. This would be input as goniometerangleaxis = YYYY. Multiple goniometerangleaxes can also be input as a series. Set typically line by line in the data section or in the defaults section.

goniometermotordirection – This is the direction of positive rotation for a given goniometer motor. 1 is used for counter clockwise rotation designated as positive. Example: `goniometermotordirection ='1'`, and for example, for multiple goniometermotors, `goniometermotordirection='1,-1,1,1'.`

goniometermotoroffset – This is an angle in degrees to be used for the offset angle for the goniometer angle motor. This value is added to the value read from the goniometermotor value read from the data file. If no offset is given, then the angle is read directly from the datafile. Set typically line by line in the data section or in the defaults section. Example `goniomtermotoroffset='13.12'` or for multiple motor offsets `goniometermotoroffset='[-44,-12,0]'.`

grouping – This is the name of the grouping file to be used in the data reduction. Pre-generated grouping files can be placed in the current directory within which one is running the reduction code. If no grouping value is given, then the default value is 1X1. The VXH grouping corresponds to V pixels grouped vertically and H pixels grouped horizontally. For instruments with "8pack" detectors V can have the values of (1,2,4,8,16,32,64,128), and H can have the values (1,2,4,8). These '8pack' style grouping files are generated automatically by the reduction routines. It is not recommended to bin the data with V>4 or H>1. The grouping can also be set to `grouping='powder'`. In this case, a powder grouping of the detectors based upon the anglestep value will be used. This keyword is set typically line by line in the data section or in the defaults section. Example: `grouping='2X1'.`

instrument - the instrument keyword should correspond to the instrument used for the measurements. Current allowed values are ARCS, SEQUIOA, HYSPEC, and CNCS. The instrument keyword value is case sensitive. The instrument keyword must be set in the defaults section. Example: instrument="SEQUIOA"

ipts – integer for the IPTS number. This is currently a bookkeeping device, and is not required. Set typically line by line in the data section or in the defaults section. Example: `ipts='1234'.`

kiokf – Set this to Boolean true to include this correction. Set typically line by line in the data section or in the defaults section or in the instrument defaults. `Example kiokf='true'.`

lattice – This keyword uses lattice parameters and the DAVE-Mslice style of U and V vectors to input the crystal orientation into the reduction routines. It is suggested that one sets this value in the defaults section of the xml file. This is currently used with 'mdnxs' file types. The key value is a set of comma delimited numbers in the order "a,b,c,alpha,beta,gamma,ux,uy,uz,vx,vy,vz" where a,b,c,alpha, beta and gamma are the real space lattice constants in angstroms and degrees. ux, uy, uz, vz, vy, and vz are the DAVE-Mslice style U and V vectors.

logvalue – This keyword sets the logvalue to be used with sweep mode of data reduction. It is set to the name of a log value in the sample logs. Examples: `logvalue='CCR12Rot'` and `logvalue='SensorD'.`

logvaluebin – This keyword sets the binning of logvalues for sweep mode of datareduction. Example: `logvaluebin='0.5'.`

logvaluemax – This keyword sets the maximum value of binning for the logvalue used in sweep mode. Example: `logvaluemax='125'.`

logvaluemin - This keyword sets the minimum value of binning for the logvalue used in sweep mode. Example: `logvaluemin='12'.`

mask(algorithm="angle",twothetamin= X, twothetamax = Y) – keywords within the mask function are case sensitive. The angle algorithm will mask between twothetamin and twothetamax. if no minimum value is given, then the minimum value is set to 0. Set within the vanadium and calibration section. For multiple angular ranges, algorithm should be called multiple times. Please note that the order of masking will affect the resulting mask. We suggest masking detectors using the MaskBTP algorithm AFTER other algorithms are called. The mask algorithms are currently only set in the calibration and mask section. Example, `<mask algorithm="angle" twothetamax="2.5"/>`

mask(algorithm="BankTubePixel", bank = "", tube = "", pixel="") - keywords within the mask function are case sensitive. The BankTubePixel algorithm will mask ranges of banks tubes and pixels for 8pack detector instruments. The algorithm is to be called multiple times for multiple ranges of bank, tubes and pixels. Please note that the order of masking will affect the resulting mask. We suggest masking detectors using the MaskBTP algorithm AFTER other algorithms are called. Examples of these calls are shown and commented in the example sections of this document. The mask algorithms are currently only set in the calibration and mask section. Example, `<mask algorithm="BankTubePixel" bank="59" tube="6"/>`

mask(algorithm="FindDetectorsOutsideLimits") - This algorithm is used to eliminate detectors outside of set limits. The default is currently set to mask any detectors with zero counts.

mask(algorithm="MedianDetectorTest", Significancetest=X, LowThreshold=Y, HighThreshold=Z) - keywords within the mask function are case sensitive. The mediandetectortest calls the MantidPlot algorithm of the same name. Please see Mantid help files for further details regarding this algorithm. Please note that the order of masking will affect the resulting mask. We suggest masking detectors using the MaskBTP algorithm AFTER other algorithms are called. The mask algorithms are currently only set in the calibration and mask section. All keywords available to this algorithm are available for use in the reduction routine. Example `<mask algorithm="MedianDetectorTest" SignificanceTest="3.0" LowThreshold="0.2" HighThreshold="5.0"/>`

normalizedcalibration – True or False. When set to true, this will normalize the vanadium calibration file to fluctuate about a value of one. This keyword is only set in the calibration and mask section. Example `normalizedcalibration='true'.`

powderanglestep – This is the step size in scattering angle with which the data are binned for powder grouping. Example: `powderanglestep='0.15'.`

processedfilename – This is the filename of the vanadium calibration file. The calibration file also contains the mask information. If the processedfilename file already exists, then the vanadium information will not be reprocessed, and the existing file will be used for calibration and masking. If the processedfilename file does not exist in the current directory you are operating within, then the vanruns vanadium file information will be processed. Example `<processedfilename='vanadium_15884_white.nxs'>`

qstep – This is the bin size for q in inverse Angstroms for the binning of iofq text files. Example: `qstep='0.01'.`

runs – This is a listing of runs to combine together in a single reduction of data. If more than one run is listed runs must be specified within quotation marks. multiple runs are specified in a comma delimeted list. hyphenated pairs of run numbers are considered to be a series of consecutive runs. Example, `runs='19888-19919, 19922'`.

save – This is a listing of the file types one wants to use in saving the data. Currently supported file types are .nxspe, spe, phx, summary, nxs, iofq, mdnxs, and iofe. Values are set in a comma delimited list. Example, `save= 'spe,summary,par,nxspe,nxs'`.

scantype – The scantype keywords determines how runs are combined in a given call of the <scan> reduction line. scantype can be, single, step, or sweep. single will combine all of the data together for a single reduction. step will individually reduce all of the given runs listed. sweep will combine all of the data together, and then bin them according to the log parameter chosen by the keywords logvalue, logvaluemin, logvaluemax, and logvaluestep. Note that currently there must be more than one value in the logvalue for sweep mode to work correctly (see Appendix 4). Example: `scantype='step'`.

t0 – This is the time in microseconds that the peak in the neutron pulse takes to leave the moderator. If the incident energy is fitted, than the value of t0 from this fit is used, regardless if t0 is set or not. The t0 value is typically set line by line in the data section or in the defaults section. Example: `t0='12.75'`.

tibg – This is a boolean keyword to set if one is using a time-independent-background subtraction or not. This is typically set in the instrument default file or line by line in the data section. Example: `tibg='False'`.

tibgstart – This is the starting time in microseconds to use for averaging and creation of the time-independent background subtraction. The time-independent background is generated based upon the data within runs. Exapmle: `tibgstart = '1200'`.

tibgstop - This is the stopping time in microseconds to use for averaging and creation of the time-independent background subtraction. The time-independent background is generated based upon the data within runs. Exapmle: `tibgstpo = '1500'`.

ub – This keyword uses the ub matrix to input the crystal orientation into the reduction routines. This is currently used with 'mdnxs' file types. It is suggested that one sets this value in the defaults section of the xml file. The key value is a set of comma delimited numbers in the order of first moving across the matrix and then moving down to the next line and subsequently across. Example, if the ub matrix is given by

$$ub = \begin{matrix} 0 & 1 & 1 \\ 2 & 3 & 5 \\ 8 & 13 & 21 \end{matrix}$$

then the keyword is set by `ub='0,1,1,2,3,5,8,13,21'`.

units – This is the units of integration that are used in the creation of the vanadium calibration file. Currently available units are DeltaE', 'DeltaE_inWavenumber', 'Energy', 'Energy_inWavenumber', 'Momentum', 'MomentumTransfer', 'QSquared', 'TOF', 'Wavelength', and 'dspacing'. The units value is set in the calibration and mask section. Example, `units='wavelength'`.

vanmax – This is the maximum in the integration range used for generation of the vanadium calibration file. If no vanmax value is given, than the maximum in the available range of integration is used. The vanmax value is set in the calibration and mask section. Example, `<vanmax>1.2</vanmax>`.

vanmin - This is the minimum in the integration range used for generation of the vanadium calibration file. If no vanmin value is given, than the minimum in the available range of integration is used. The vanmin value is set in the calibration and mask section. Example, `<vanmin>0.3</vanmin>`.

vanpath – Set this keyword and value in the calibration section to add a particular path to find the location of vanadium files. `vanpath='/SNS/users/5us/mantidscripts/v3'`.

vanruns – These are the numbers of the vanadium measurements to be used for generation of the calibration and mask file. The vanruns value is set in the calibration and mask section. If more than one run is listed runs must be specified within quotation marks. multiple runs are specified in a comma delimeted list. hyphenated pairs of run numbers are considered to be a series of consecutive runs. The vanruns value is set in the calibration and mask section. Example, `<vanruns>15884-15885</vanruns>`

Appendix 2 - Example files

The following example files illustrate how to set up the reduction text file. Note that line wrapping makes the long lines of the data section appear on separate lines of this document.

---Example 1 --- Generation of a vanadium calibration file

For ARCS and SEQUOIA, the vanadium calibration file is generated via a T0 (no Fermi chopper) chopper only measurement of a vanadium sample. Local contacts are the best resource for determining the appropriate settings for generating of a calibration file.

```
<dgsreduction>
  <!-- DEFAULTS section -->
  <defaults
    Instrument="ARCS"
    filterbadpulses = "False"
    save='summary'
  />

  <!-- CALIBRATION AND MASKING section -->
  <calibration processedfilename = "vanadium_24510_white.nxs" units="wavelength"
normalizedcalibration = "True">
    <vanruns>24510</vanruns>
    <vanmin>0.35</vanmin>
    <vanmax>0.75</vanmax>

    <mask algorithm="BankTubePixel" pixel="1,2,3,4,5,6,7"/>
    <mask algorithm="BankTubePixel" pixel="120-128"/>
    <mask algorithm="angle" twothetamax="2.5"/>
    <mask algorithm="FindDetectorsOutsideLimits" />
    <mask algorithm="MedianDetectorTest" SignificanceTest="3.0" LowThreshold="0.4"
HighThreshold="1.6" ExcludeZeroesFromMedian="1" LevelsUp="1" CorrectForSolidAngle="1"/>
  </calibration>
</dgsreduction>
```

Once the nexus vanadium calibration file is obtained, it is a good idea to examine it using MantidPlot (Load the nexus file, and use view-instrument option). One may need to mask additional detectors and iterate through the process of generating a properly masked vanadium calibration file. If the processedfilename already exists, than it will simply be opened and used for the reduction of data. So when one is iterating

through generation of the calibration file, one needs to delete or rename the previously generated vanadium calibration file.

One may also generate a false calibration files of all unity values. This is done by

---Example 1b --- Generation of a unitary calibration file

One may also generate a false calibration file. This file is composed entirely of unity values. This is done identically to Example 1, but no file information is included. It is usefull for examining data with no sensitivity correction. One may include masking in this type of file also.

```
dgsreduction>
  <!-- DEFAULTS section -->
  <defaults
    Instrument="ARCS"
    filterbadpulses = "False"
    save='summary'
  />

  <!-- CALIBRATION AND MASKING section -->
  <calibration processedfilename = "noVan.nxs" normalizedcalibration = "True">

    <mask algorithm="BankTubePixel" pixel="1,2,3,4,5,6,7"/>
    <mask algorithm="BankTubePixel" pixel="120-128"/>
    <mask algorithm="angle" twothetamax="2.5"/>

  </calibration>
</dgsreduction>
```

---Example 1c --- Generation of a unitary calibration file using an older detector configuration

The default detector configuration for a unitary calibration file is the most current configuration. One may choose older configuration by using the datadate keyword as shown here. The keyword value is a hyphenated string of year-month-day, datadate=”2012-10-30”.

```
dgsreduction>
  <!-- DEFAULTS section -->
  <defaults
    Instrument="ARCS"
    filterbadpulses = "False"
    save='summary'
  />

  <!-- CALIBRATION AND MASKING section -->
  <calibration processedfilename = "noVan.nxs" normalizedcalibration = "True" datadate=”2012-10-
30”>

    <mask algorithm="BankTubePixel" pixel="1,2,3,4,5,6,7"/>
    <mask algorithm="BankTubePixel" pixel="120-128"/>
    <mask algorithm="angle" twothetamax="2.5"/>

  </calibration>
</dgsreduction>
```

The remainder examples only show individual <scan/> lines assuming that the top of the file is identical to the above example.

---Example 2 --- Simplest reduction line

```
<scan runs="24511" />
```

The run 24511 is reduced using all default values.

---Example 3 --- Saving a data file

```
<scan runs="24512" save='nxspe,summary'/>
```

The run 24512 is reduced and an nxspe and summary file are saved. All other values are default values.

---Example 4 --- friendlyname and friendlynamelogs.

```
<scan runs="24513" save='nxs,phx,spe,summary,par,garbage' friendlyname="diamond"
friendlynamelogs="EnergyRequest"/>
```

Scan 24513 is reduced, and several file types are saved. Note that unknown filetypes can be listed, but they are just ignored. A friendlyname is used for the subdirectory within which the data are saved, and the log values for EnergyRequest are included in the file names. If a friendlynamelogs value is listed which does not exist, then it is ignored.

---Example 5 --- set the energy binning.

```
<scan runs="24515" emin="-10" emax="35" ebin="0.5" save='nxspe,summary' friendlyname="diamond_bin1"
friendlynamelogs=""/>
```

The incident energy is automatically set based upon a fit to monitor data (if set this way in default parameters), but the energy binning is set from -10 to 35 meV in 0.5 meV steps. No friendlynamelogs information is used.

---Example 6 --- Set the efixed value, but it is too far from true value.

```
<scan runs="24516" efixed="30" emin="-10.25" emax="22.25" ebin="0.5" save='nxspe,summary'
friendlyname="diamond_bin2" />
```

This efixed value is very far from the true value. A fit is performed, but it fails and issues a warning "No peak found in data that satisfieds prominence criterion."

---Example 7 --- Set the efixed value, and Ei fit works correctly.

```
<scan runs="24516" efixed="27" emin="-10.25" emax="22.25" ebin="0.5" save='nxspe,summary'
friendlyname="diamond_bin2" />
```

This sets the efixed value as the starting point for the incident energy fit, and successfully sets the incident energy and T0 values after the fit.

---Example 8 --- Do not fit the incident energy or T0 value (calce='false').

```
<scan runs="24516" efixed="27" t0='3.725' calce='false' emin="-10.25" emax="22.25" ebin="0.5"
save='nxspe,summary' friendlyname="diamond_bin3" />
```

---Example 9 --- Combine and average two runs together.

```
<scan runs="24517,24532"  save='nxspe,summary' friendlyname="diamond_bin5" />
```

---Example 10 --- Set grouping to powder mode and change powder binning.

```
<scan runs="24521"  grouping='powder' powderanglestep='0.1' save='nxspe,summary'
friendlyname="diamond_bin6" />
```

powder binning file is 443 kB, and 1X1 file is 185 MB.

---Example 11 --- Multiple files to be reduced individually (step mode).

```
<scan runs="24511-24514"  save='nxspe,summary' grouping='powder' powderanglestep="0.2"
friendlyname="diamond_bin8" scantype='step'  friendlynamelogs =’run_number’/>
```

Scans are saved in powder mode in the diamond_bin8 directory with file names set by the run number. Each individual scan is separately reduced. These scans are actually at different incident energies, and this is automatically accounted for by the default parameters set. The command does not contain an efixed value since the incident energy of these scans is varying

--- Example 12 --- Splitting up a series of runs by temperature (sweep mode).

```
<scan runs="23985-23989"  save='nxs,nxspe,summary' grouping='powder' powderanglestep="0.2"
friendlyname="CrCl2_temptest1" scantype='sweep' logvalue="SensorD" logvaluemin="6" logvaluemax="51"
logvaluestep="5" friendlynamelogs='SensorD'/>
```

Temperature was varying throughout these runs. The data are all loaded together, combined, and then separated according to SensorD values with a binning of 5 K between 5 and 51 K. The logvalue of SensorD is used in the filenames.

--- Example 13 --- Several measurements separately reduced.

```
<scan runs="25131" save='nxspe,summary'/>
<scan runs="25132" save='nxspe,summary'/>
<scan runs="25133" save='nxspe,summary'/>
<scan runs="25134" save='nxspe,summary'/>
<scan runs="25135" save='nxspe,summary'/>
```

--- Example 14 --- single orientation of sample, directly fixing the motor angle.

```
<scan runs="25317" save='nxspe,summary'  goniometermotoroffset='-46'/>
```

the offset for the default axis (set in the default files), is set to -46 degrees, no reading of the actual motor angle is performed.

---Example 15 --- single orientation of sample, using logged motor angle

```
<scan runs="25131" save='nxspe,summary' friendlynamelogs='CCR12Rot' goniometermotor='CCR12Rot'/>
```

This will only work if there is a goniomtermotorffset, axis, and direction set in the defaults file or defaults section. The goniomtermotor setting is read from the sample logs and used in writing the nxspe file, it is also used in generating the fielnames.

---Example 16 --- single orientation, but adding an offset to the motor angle

```
<scan runs="25323" save='nxspe,summary'  goniometermotor='CCR12Rot' goniometermotoroffset='-46'/>
```

Adds -46 to the actual goniometer angle from CCR12Rot Log and uses this in the .nxspe file. Also notes this shfirt in the summary file. Only works is goniomtermotoraxis and direction are also set elsewhere.

---Example 17 --- Step mode with angles

```
<scan runs="25131-25135" save='nxspe,summary'  goniometermotor='CCR12Rot'
friendlynamelogs='CCR12Rot' scantype='step'/>
```

Generates one file for each run number, saves them with the angle in the filename, the summary file gets appended to as one moves through the file list.

---Example 18 --- Sweep mode with angles 1

```
<scan runs="25136-25140" save='nxspe,summary'  goniometermotor='CCR12Rot'
friendlynamelogs='CCR12Rot' scantype='sweep'/>
```

does not work since no log values listed

---Example 19 --- Sweep mode with angles 2

```
<scan runs="25136-25140" save='nxspe,summary' logvalue='CCR12Rot' logvaluemin='-26.5'
logvaluemax='26.5' logvaluestep='1.0' goniometermotor='CCR12Rot' friendlynamelogs='CCR12Rot'
scantype='sweep'/>
```

Loads all data, combines it together, then trys to split it appart over the range listed. This will only work if the motor was continuously moving and updataing the log file for these runs. If the motor was stationary for each of the individual runs, then this will not work, please see Appendix 4.

---Example 20 --- Sweep mode with temperature

```
<scan runs="25792" efixed='80' t0='29.23' save='nxspe,summary' logvalue='SampleTemp'
logvaluemin='0.5' logvaluemax='110.5' logvaluestep='10.0' friendlynamelogs='SampleTemp'
scantype='sweep'/>
```

Makes a set of temperature delimited files, and uses the mean temperature within each of the temperature bins to name the files.

---Example 21 --- Filtering by log values

```
<scan runs="25792" efixed='80' t0='29.23' save='nxspe,summary' friendlynamelogs='SampleTemp'
scantype='single' filternames='SampleTemp,Phase3' filtermin='96,2301.84' filtermax='102,2302.04'/>
```

The filtered parameters are SampleTemp and Phase3. Data acquired outside of the range of SampleTemp between 96 and 102 will be discarded. In addition, data will be discarded for any Phase3 values outside of the range 2301.84 to 2302.04. The order of the values listed in filtermin, filtermax and filternames must be the same for this to work properly.

Appendix 3 – The instrument default file

The instrument default file is a function building up a dictionary of keywords and their default values. The file is saved in the /SNS/XXX/shared/ directory where XXX is the instrument name. It can also be saved in the local directory one is running the reduction code from.

Example 1: arcsdefault.py

```
def instrumentparameters():

    globaldict = dict()
    #instrument items
    globaldict['instrument']='ARCS'
    globaldict['filterbadpulses'] = False

    #calibration and mask items
    globaldict['vanruns'] = None
    globaldict['units']   = 'TOF'
    globaldict['vanmin']  = None
    globaldict['vanmax']  = None
```

```
    globaldict['processedfilename'] = 'vanadium.nx5'
    globaldict['maskfilename'] = None
    globaldict['mask'] = [] #the mask parameters are stored as a list of dictionaries. (check this)
    globaldict['normalizedcalibration'] = False

    #data items
    globaldict['ipts'] = None  #optional, need to make work for ISIS too.
    globaldict['runs'] = []  #returns a list of ints
    globaldict['efixed']= None #number double
    globaldict['t0'] = 0.0 #number double
    globaldict['calce'] = True
    globaldict['ei-mon1-spec'] = 1
    globaldict['ei-mon2-spec'] = 2
    globaldict['emin']=None #number double
    globaldict['emax']=None #number double
    globaldict['ebin']=None #number double
    globaldict['qstep']=None #number double
    globaldict['kiokf']=True #can only use boolean here, not 1 or 0.
    globaldict['tibg']=False
    globaldict['tibgstart']=None
    globaldict['tibgstop']= None
    globaldict['grouping']=None
    globaldict['powderanglestep'] = 0.5
    globaldict['goniometermotor']=None  #script takes care of default situations.
    globaldict['goniometermotoroffset']=None
    globaldict['goniometermotoraxis']=["0,1,0"]
    globaldict['goniometermotordirection']=[1]
    globaldict['save']=[]
    globaldict['friendlyname']='ARCS'
    globaldict['friendlynamelogs'] = 'run_number' #will get the run_number from the logs, first run
number only.

    return globaldict
```

Example 2: hyspecdefault.py

```
import math

def instrumentparameters():

    globaldict = dict()


    #instrument items
    globaldict['instrument']='HYSPEC'
    globaldict['filterbadpulses'] = False

    #calibration and mask items
    globaldict['vanruns'] = None
    globaldict['units']   = 'TOF'
    globaldict['vanmin']  = None
    globaldict['vanmax']  = None
    globaldict['processedfilename'] = 'vanadium.nx5'
    globaldict['maskfilename'] = None
    globaldict['mask'] = [] #the mask parameters are stored as a list of dictionaries. (check this)
    globaldict['normalizedcalibration'] = False


    #data items
    globaldict['ipts'] = None  #optional, need to make work for ISIS too.
    globaldict['runs'] = []  #returns a list of ints
    globaldict['efixed']= None #number double
    globaldict['t0'] = None #number double
    globaldict['calce'] = False
    globaldict['ei-mon1-spec'] = 1
    globaldict['ei-mon2-spec'] = 2
    globaldict['emin']=None #number double
    globaldict['emax']=None #number double
    globaldict['ebin']=None #number double
    globaldict['qstep']=None #number double
    globaldict['kiokf']=True #can only use boolean here, not 1 or 0.
    globaldict['tibg']=False
    globaldict['tibgstart']=None
    globaldict['tibgstop']= None
    globaldict['grouping']=None
    globaldict['powderanglestep'] = 0.5
    globaldict['goniometermotor']=None  #script takes care of default situations.
```

```
    globaldict['goniometermotoroffset']=None
    globaldict['goniometermotoraxis']=["0,1,0"]
    globaldict['goniometermotordirection']=[1]
    globaldict['save']=[]
    globaldict['friendlyname']='HYS'
    globaldict['friendlynamelogs'] = 'run_number' #will get the run_number from the logs, first run
number only.

    return globaldict


def t0fromei(ei):
    return 1.0*(25.0 + 85.0 / (1+math.pow((ei/27.0),4.0)))
```

Example 3: sequoiadefault.py

```
def instrumentparameters():
    globaldict = dict()

    #instrument items
    globaldict['instrument']='SEQUOIA'
    globaldict['filterbadpulses'] = False

    #calibration and mask items
    globaldict['vanruns'] = None
    globaldict['units']   = 'TOF'
    globaldict['vanmin']  = None
    globaldict['vanmax']  = None
    globaldict['processedfilename'] = 'vanadium.nx5'
    globaldict['maskfilename'] = None
    globaldict['mask'] = [] #the mask parameters are stored as a list of dictionaries. (check this)
    globaldict['normalizedcalibration'] = False


    #data items
    globaldict['ipts'] = None  #optional, need to make work for ISIS too.
    globaldict['runs'] = []  #returns a list of ints
    globaldict['efixed']= None #number double
    globaldict['t0'] = 0.0 #number double
    globaldict['calce'] = True
    globaldict['ei-mon1-spec'] = 1
    globaldict['ei-mon2-spec'] = 2
    globaldict['emin']=None #number double
    globaldict['emax']=None #number double
    globaldict['ebin']=None #number double
    globaldict['qstep']=None #number double
    globaldict['kiokf']=True #can only use boolean here, not 1 or 0.
    globaldict['tibg']=False
    globaldict['tibgstart']=None
    globaldict['tibgstop']= None
    globaldict['grouping']=None
    globaldict['powderanglestep'] = 0.5
    globaldict['goniometermotor']=None  #script takes care of default situations.
    globaldict['goniometermotoroffset']=None
    globaldict['goniometermotoraxis']=["0,1,0"]
    globaldict['goniometermotordirection']=[1]
    globaldict['save']=[]
    globaldict['friendlyname']='SEQ'
    globaldict['friendlynamelogs'] = 'run_number' #will get the run_number from the logs, first run
number only.


    return globaldict
```

Example4: cncsdefault.py

```
import math

def instrumentparameters():

    globaldict = dict()

    #instrument items
    globaldict['instrument']='CNCS'
```

```
    globaldict['filterbadpulses'] = False

    #calibration and mask items
    globaldict['vanruns'] = None
    globaldict['units']   = 'TOF'
    globaldict['vanmin']  = None
    globaldict['vanmax']  = None
    globaldict['processedfilename'] = 'vanadium.nx5'
    globaldict['maskfilename'] = None
    globaldict['mask'] = [] #the mask parameters are stored as a list of dictionaries. (check this)
    globaldict['normalizedcalibration'] = False

    #data items
    globaldict['ipts'] = None  #optional, need to make work for ISIS too.
    globaldict['runs'] = []  #returns a list of ints
    globaldict['efixed']= None #number double
    globaldict['t0'] = None #number double
    globaldict['calce'] = False
    globaldict['ei-mon1-spec'] = 1
    globaldict['ei-mon2-spec'] = 2
    globaldict['emin']=None #number double
    globaldict['emax']=None #number double
    globaldict['ebin']=None #number double
    globaldict['qstep']=None #number double
    globaldict['kiokf']=True #can only use boolean here, not 1 or 0.
    globaldict['tibg']=False
    globaldict['tibgstart']=None
    globaldict['tibgstop']= None
    globaldict['grouping']=None
    globaldict['powderanglestep'] = 0.25
    globaldict['goniometermotor']=None  #script takes care of default situations.
    globaldict['goniometermotoroffset']=None
    globaldict['goniometermotoraxis']=["0,1,0"]
    globaldict['goniometermotordirection']=[1]
    globaldict['save']=[]
    globaldict['friendlyname']='CNCS'
    globaldict['friendlynamelogs'] = 'run_number' #will get the run_number from the logs, first run
number only.

    return globaldict


def t0fromei(ei):
    return 1000.0*(-0.1982*math.pow(1.0+ei,-0.84098))
```

Appendix 4 – Known bugs and issues.

- Use of the scan=’sweep’ for a series of runs at different motor rotation angles when the rotation motor is stationary for each run. This does not work because only a single motor angle is stored in the log files. Mantid is not able to separate the file into multiple steps, if there is only a single value in the log file.

Acknowledgements

Thanks to Doug Abernathy for suggestions for improvement. Thanks to Jennifer Niedziela for assistance with coding the filtering by log value portion of the code.