Data Reduction

This section provides a guide showing how DAWN can be used to reduce SAXS and WAXS data obtained from I22.

As there are such a large number of potential analyses that can be performed on the diffraction images obtained; this guide will predominantly focus on:

- Starting out with DAWN
- The Processing perspective
- Reducing data from I22 using DAWN
- Background subtraction
- Viewing results in DAWN
- Exporting reduced data or detector images

After this introduction it is hoped that you will be able to move on to experimenting with other processing plugins, a list of which is Processing Perspective - List of Processes, to perform more complex analysis on your data. Tutorials on certain common analysis techniques are also covered in the Data Analysis section of this site.

N.B. This version of the guide is intended for use with DAWN 2.7.x.

Manuals for previous versions of DAWN can be found by following Manuals for older DAWN versions, this page serves as an archive, as PDF and Word documents, of this documentation.

In addition to the information hosted here, further information about the primary routes, at Diamond, for the reduction of SAXS data can be found by following these links:

- Starting out with DAWN Science

Next page: Starting out with DAWN

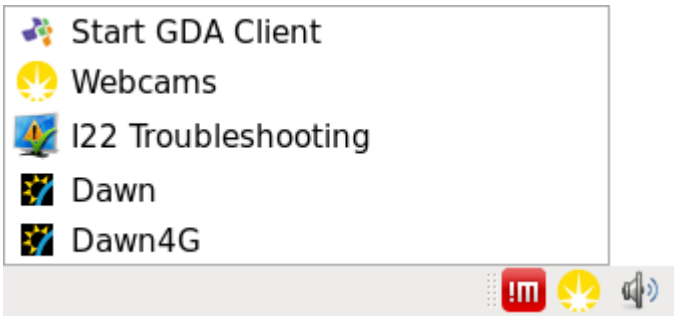
Starting out with DAWN

This page provides information on the following:

After illustration of the installation steps, which are platform specific, all instructions provided thereafter were performed on a Linux version of DAWN 2.7.x. Although there are no specific alterations to the way the program works between the operating systems, certain parts of DAWN might look different on your computer to the illustrations in this guide.

DAWN at Diamond

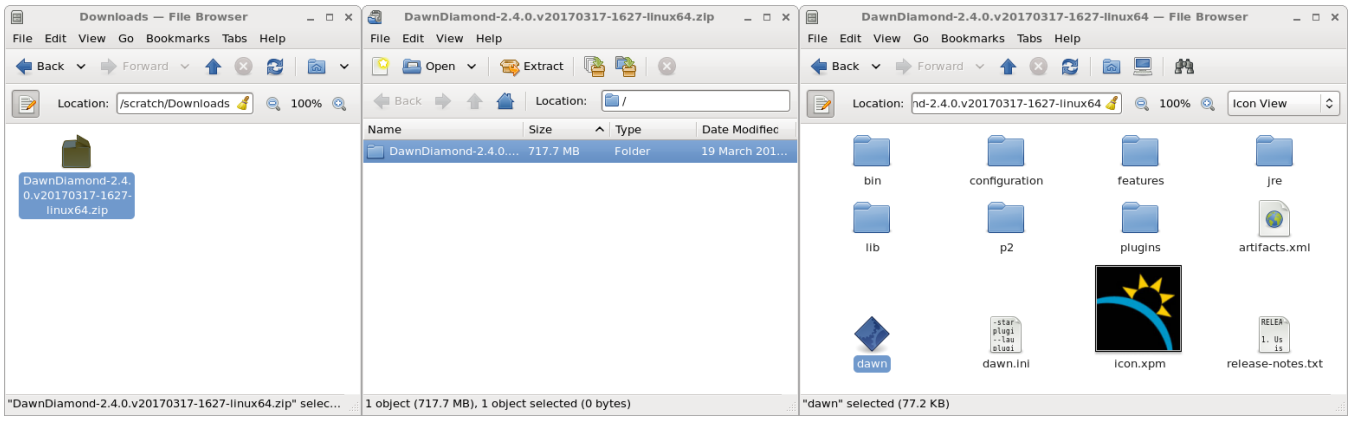
The computers in the I22 control and data rooms come with DAWN pre-installed. Clicking the Diamond icon in the tool bar at the bottom of your screen reveals the Diamond launcher, from here you can launch DAWN, as highlighted below.



There are two versions that are used at Diamond, DAWN and DAWN 4G, these are both the same program with the latter is allocated 4 gigabytes of memory as opposed to the default of 1 gigabyte. Workstation i22-ws002 in the I22 control room has ample memory to run the 4G version, which will likely run more smoothly than the default. However, the workstations in the data room have less memory installed and will not run the higher memory version.

DAWN for Linux

DAWN is principally supported for Red Hat Enterprise Linux (version 6) environments, however, it is anticipated that it should run on a number of other Linux distributions. You can download a copy of DAWN from http://dawnsci.org/downloads/. In addition to the current release of DAWN on this page there is also a link to previous versions, should you require or prefer to use an older version. **Please note**: the images below were captured for version 2.4 of DAWN, despite this the installation methodology remains the same with only the name of the archive changing, it is now DawnDiamond-2.7.0.v20171114-1119-linux64.zip.



For Linux users installation requires you to download the zip file from the dawnsci.org website, unzip this file, place the DAWN folder in your preferred location and then run the executable named *dawn* either from your GUI or from the command line.

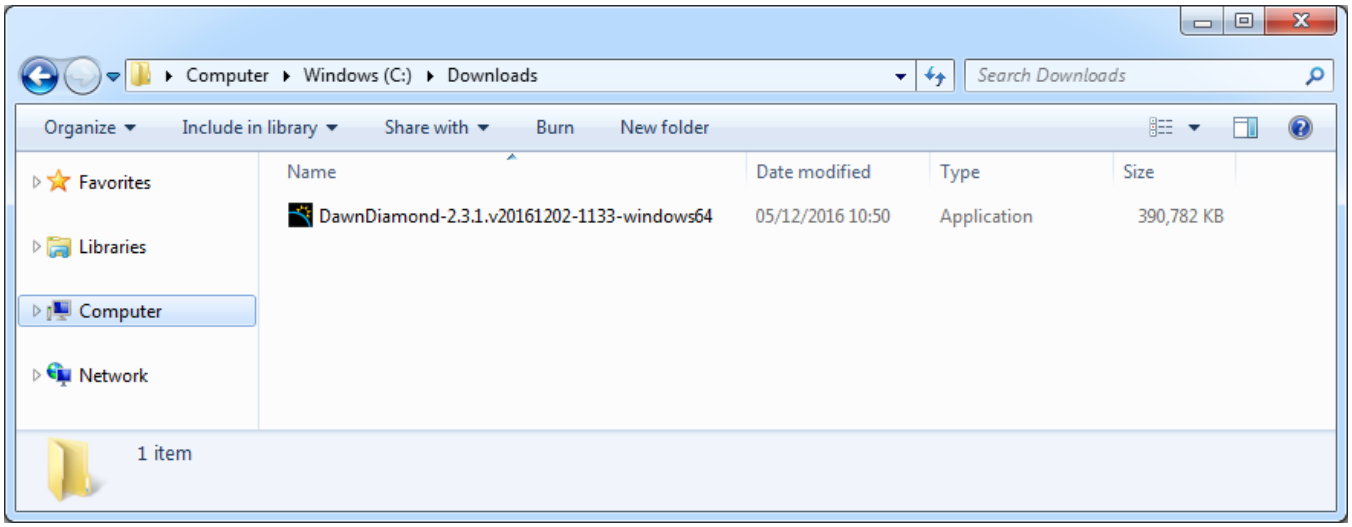
By default DAWN is configured to use, at most, 1 gigabyte of memory. Should you wish to increase this for a single run you will need to launch DAWN from the command line using the following:

Where 4096m is the number of megabytes of memory that DAWN can access, in this case 4 GB. Should you wish to change the amount of memory that DAWN uses for all subsequent launches you will need to edit the dawn.ini file in the DAWN directory. Similarly to running from the command line, the line you will need to edit the line containing the arguement *-Xmx????m*, where you replace the ?'s with the number of megabytes of memory you wish DAWN to be able to access.

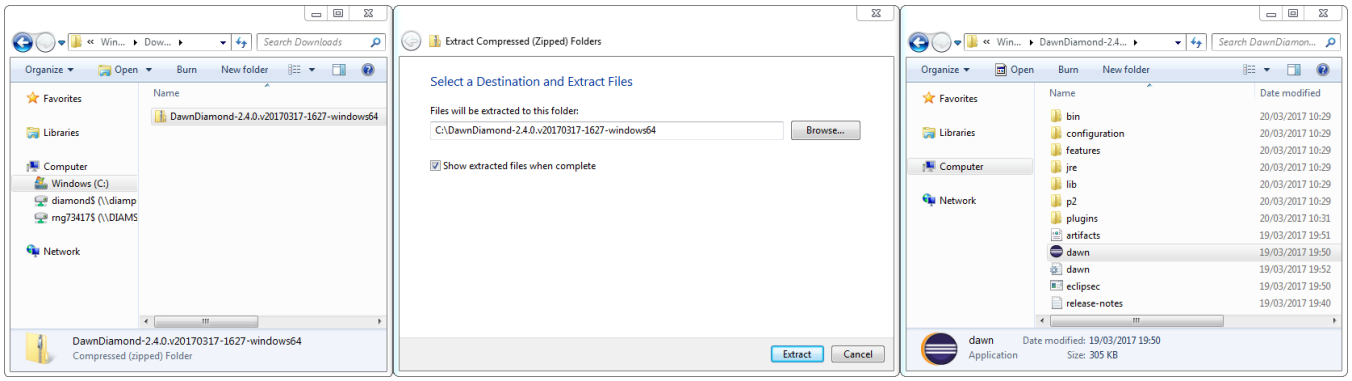
Once DAWN is running please navigate to the getting started section below or click here.

DAWN for Windows

DAWN is principally supported for Windows version 10 & 7. You can download a copy of DAWN from http://dawnsci.org/downloads/. In addition to the current release of DAWN on this page there is also a link to previous versions, should you require or prefer to use an older version. By default, clicking the download button this page will download an executable installer for Windows, as highlighted below, which can be installed in the usual Windows manner. **Pl ease note**: the images below were captured for version 2.3 of DAWN, despite this the installation methodology remains the same with only the name of the archive changing, it is now DawnDiamond-2.7.0.v20171114-1119-windows64.exe.



However, if administrator rights to the desired machine are not held, DAWN can also be downloaded as a self-contained *zip* file which can be downloaded from this location, expanded and run as a straightforward Windows application, as highlighted below.



To run, simply place the DAWN folder in your preferred location and then run the executable named *dawn.exe* from Windows Explorer.

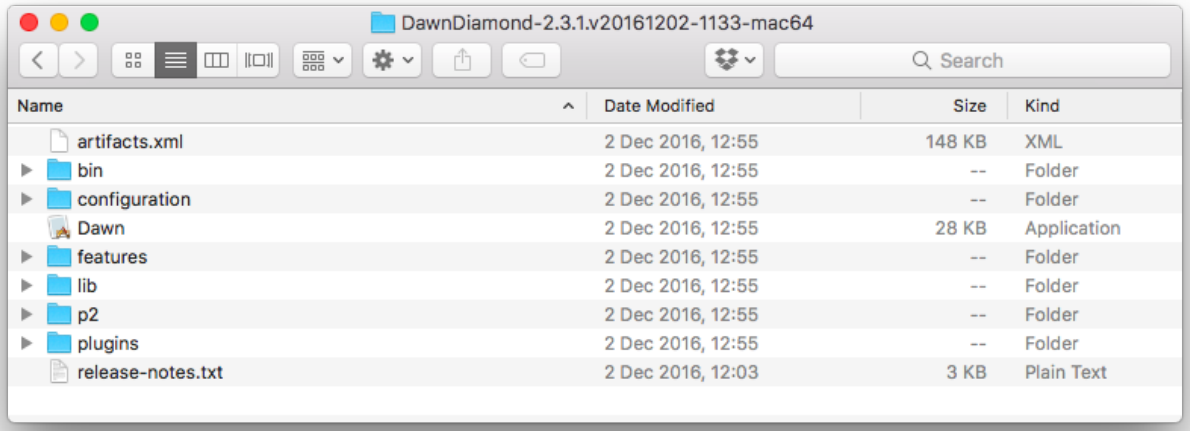
Should you wish to change the amount of memory that DAWN uses for all subsequent launches you will need to edit the *dawn.ini* file in the DAWN directory. Similarly to running from the command line, the line you will need to edit the line containing the arguement *-Xmx????m*, where you replace the ?'s with the number of megabytes of memory you wish DAWN to be able to access.

Once DAWN is running please navigate to the getting started section below or click here.

DAWN for Mac

DAWN is principally supported for Mac OS X versions 10.8-10.11 (Mountain Lion, Mavericks, Yosemite & El Capitan) and macOS 10.12-10.13 (Sierra & High Sierra). You can download a copy of DAWN from http://dawnsci.org/downloads/. In addition to the current release of DAWN on this page there is also a link to previous versions, should you require or prefer to use an older version.

For the Macintosh release it is also necessary to download the Java JDK separately, this can be found on the Oracle website by following this link. From this page it is possible to download the latest version of the Java JDK for Macintosh. **Please note**: the images below were captured for version 2.3 of DAWN, despite this the installation methodology remains the same with only the name of the archive changing, it is now DawnDiamond-2.6.0.v20170904-0753-mac64.



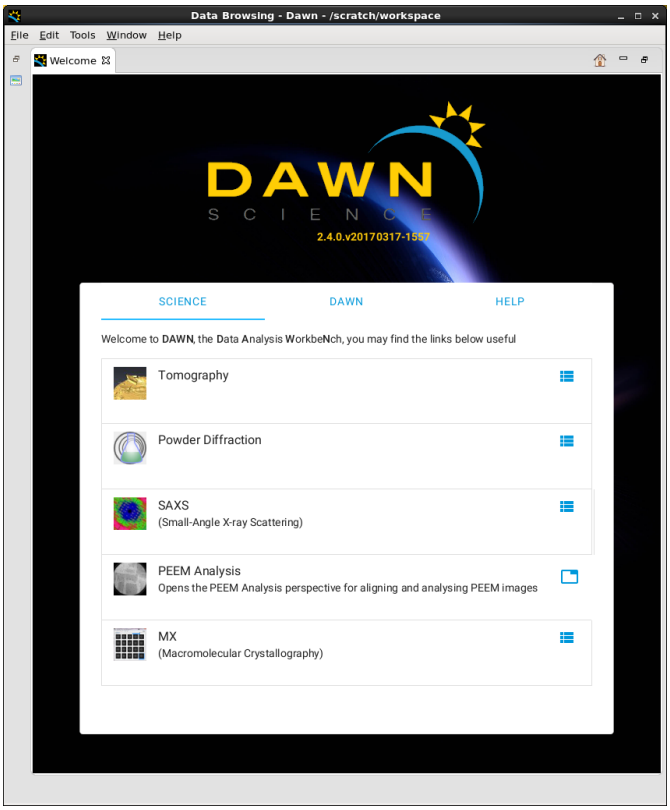
To run, uncompress the zip file, if not done automatically, and simply place the DAWN folder in your preferred location and then run the executable named *Dawn* or *Dawn.app* from the Finder.

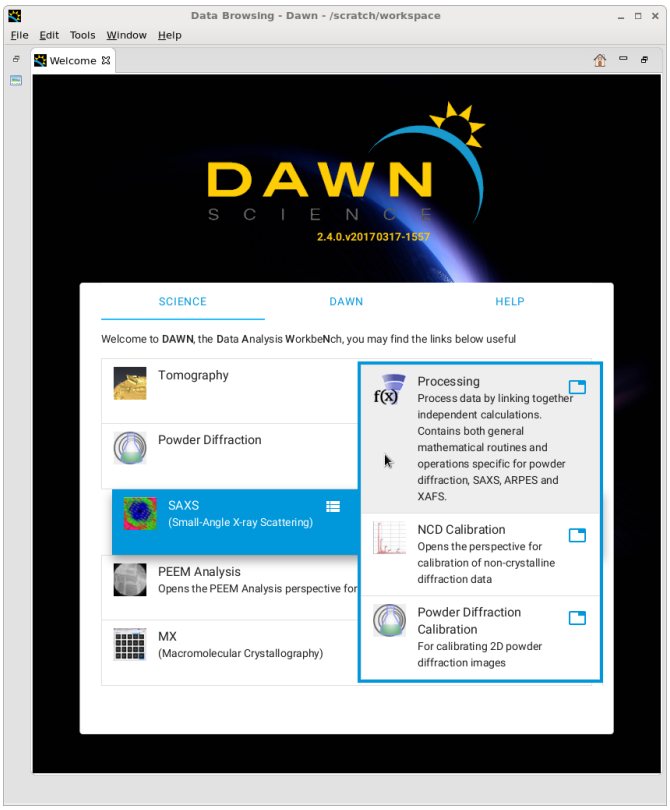
Once DAWN is running please navigate to the getting started section below or click here.

Getting started with DAWN

When you start DAWN for the first time you will be met with the window shown below on the left. Hovering the mouse over the entry marked *SAXS* will cause a pop-up menu to appear showing the functions DAWN presents for small angle x-ray scattering analysis. From this pop-up menu click on the *Processing* entry to enter DAWN. This menu only appears on the first start of DAWN[1].

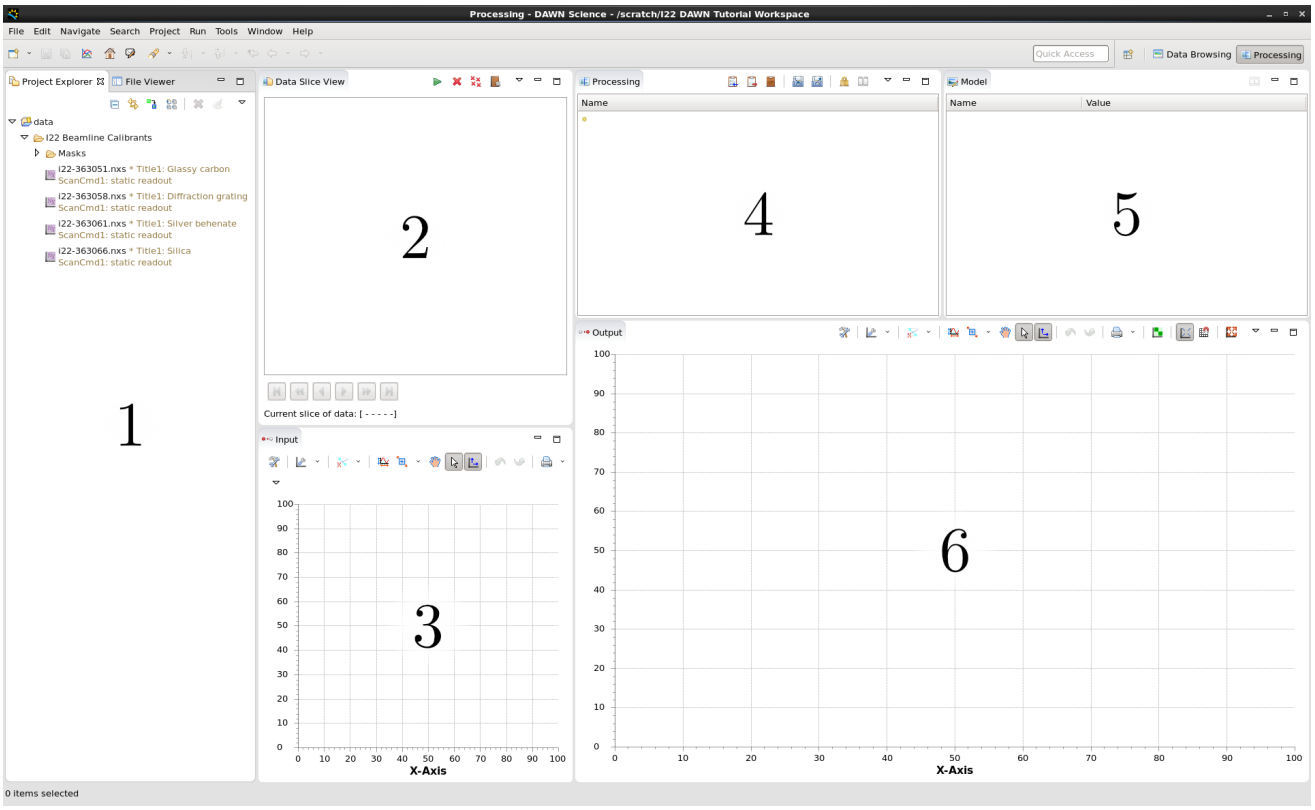
Predominantly the rest of this guide will explore the functionality presented by options found in the *Processing* perspective of DAWN.





Upon launching DAWN you will be presented the default processing perspective, as shown below, containing the following sub-panels:

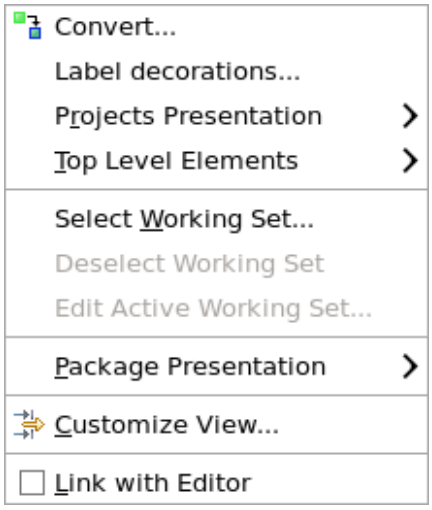
1. **File Explorer** - A representation of the files within the workspace
2. **Data Slice View** - Where files for processing will be placed
3. **Input** - A visual representation of the input data
4. **Processing** - Where the processing 'pipeline' will be represented
5. **Model** - Where options for processing plugins will appear, if appropriate
6. **Output** - A visual representation of the processed (output) data

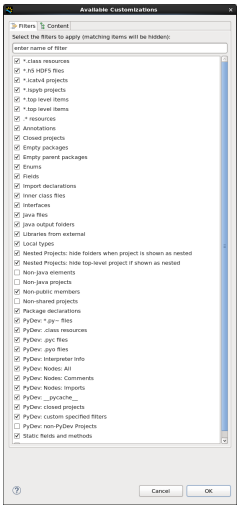


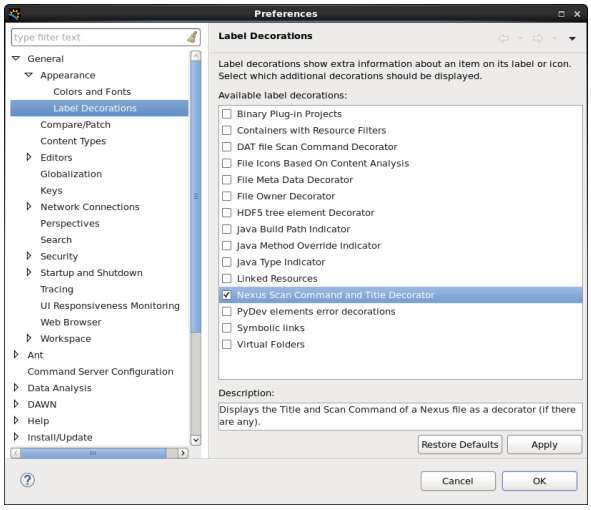
Before progressing further, two pieces of terminology:

- **Perspective** - A setup of smaller windows (sub-panels) put together in a window. These sub-panels can be separated from the main window should it be desired.
- **Workspace** - A location where your preferences and settings for DAWN are stored. In addition either your actual data and results can be stored in this location, or links to your actual data and results are stored here as well.

As DAWN is a multi-purpose tool for data reduction, the default settings show a lot of files and file metadata which may or may not be desired by all users. Options to control this can be found in the *View Menu*, this can be accessed by clicking on the downwards facing triangle found in the list of icons under the Project Explorer tab of sub-panel 1. In this menu two options can be used to streamline the data presented to the user. Firstly, by selecting *Customize View...* it is possible reduce the number of files displayed per scan number to one, and secondly, by selecting *Label decorations...* it is possible to show the title for each scan alongside the command used to gather the data. Pictures of this menu and the settings used in this guide are shown below.

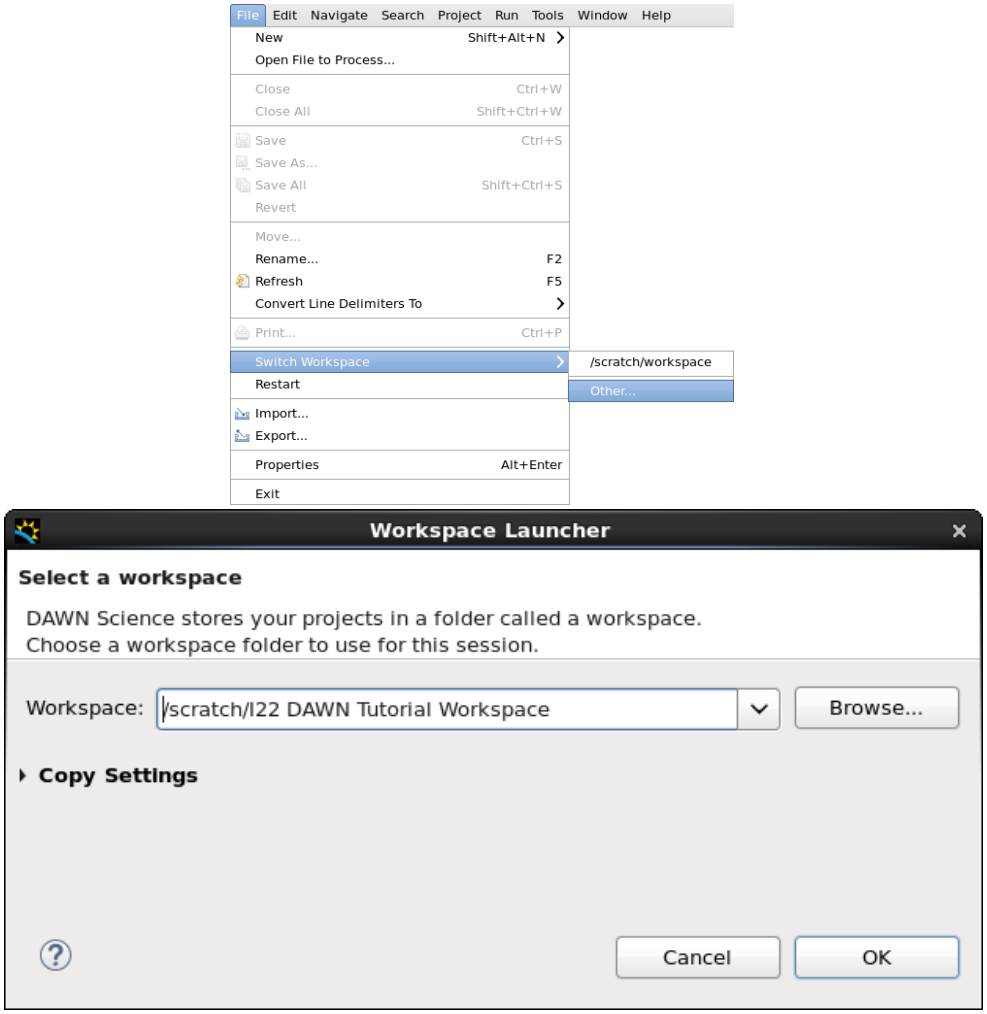






(Click on these images for a larger view)

For convenience, if desired, a workspace for I22, complete with all the settings listed and used in this guide, can be downloaded I22 DAWN Tutorial Workspace.zip. After extracting this zip file to a suitable location, select *Other...* from *Switch Workspace* in the *File* menu, as highlighted below, and then click *Browse...* in the dialog box that appears thereafter. Locate and select the extracted folder and then click *OK*, this will then load the default I22 *Processing* perspective view for DAWN 2.7, complete with some of the example data used in this guide.



[1] - For a given workspace

Overview of features for I22

At this stage DAWN is ready to input your results, as obtained from I22, either for viewing or for reduction and export. However before going into the details of the *Processing* perspective and how to reduce data this section is designed to quickly summarise the capabilities of DAWN for data gathered from I22. Over time it is expected that this list will grow as the feature set becomes richer.

- Reducing data from I22 using DAWN
- Viewing results in DAWN
- Exporting reduced data or detector images
- Mathematical formulae in the Processing pipeline
- Python scripts in the Processing pipeline
- Crystallite parameters

- Orientation calculations

Previous: Data Reduction | **Next:** The Processing perspective

The Processing perspective

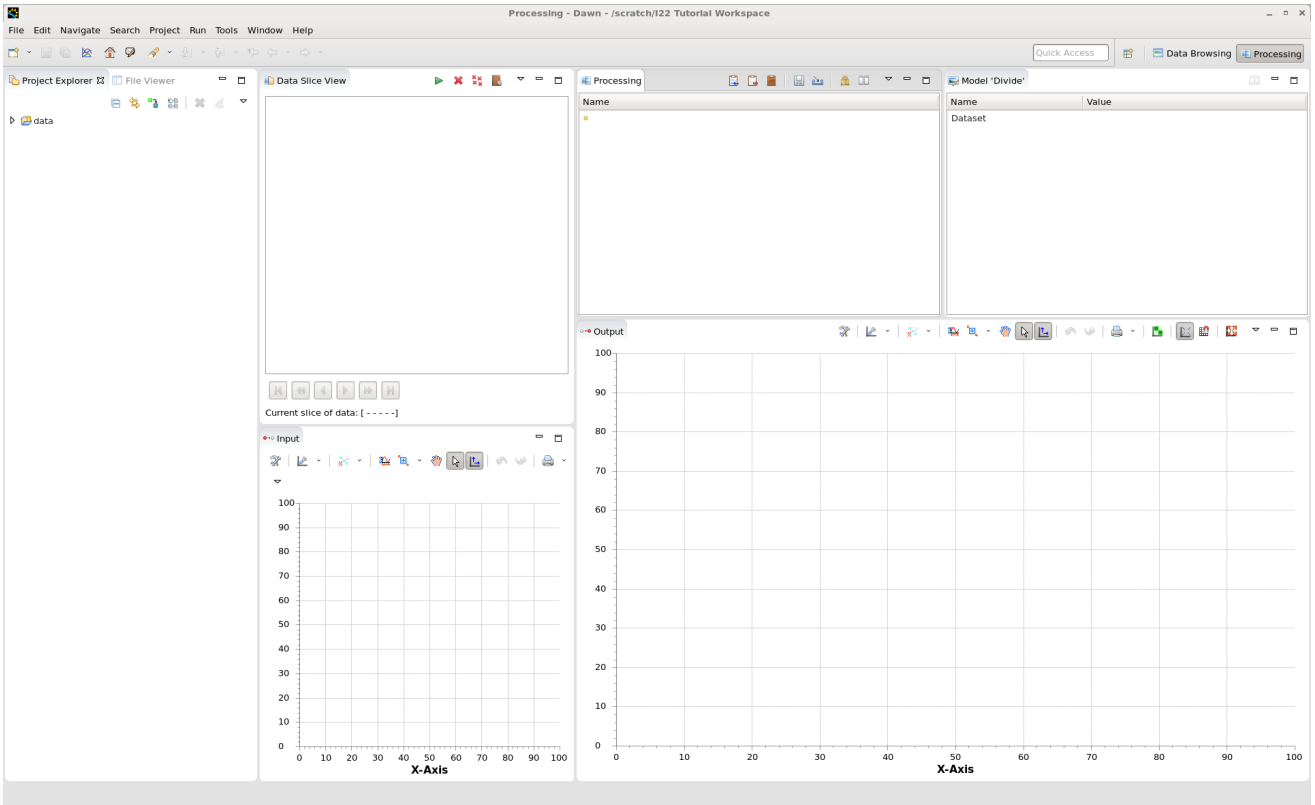
This page provides information on the following:

All instructions provided were performed on a Linux version of DAWN 2.7.x. Although there are no specific alterations to the way the program works between the operating systems, certain parts of DAWN might look different on your computer to the illustrations in this guide. A DAWN workspace accompanying this section of the guide can be downloaded I22 Tutorial Workspace.zip. Alternatively, just the data files highlighted below can be downloaded as a zip file I22 Beamline Calibrants.zip.

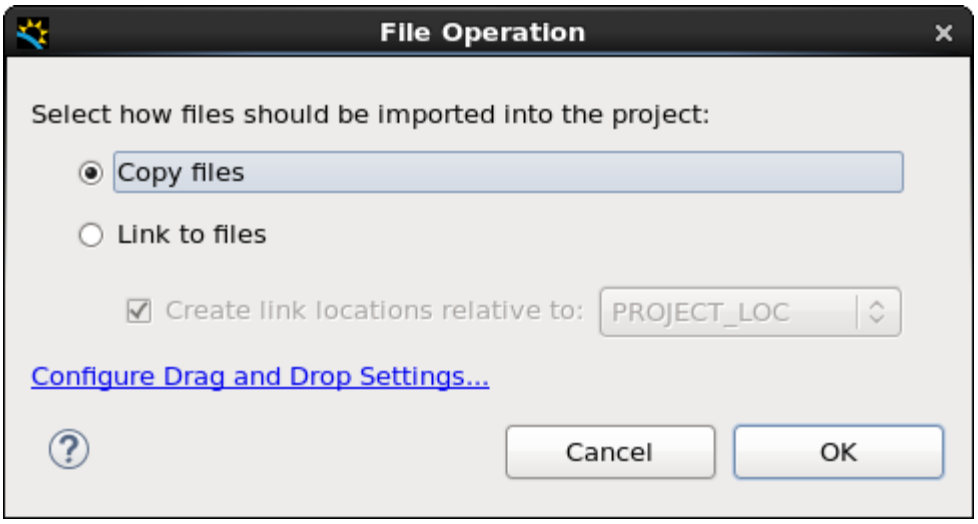
N.B. Whilst at Diamond processed data must be saved to the processing folder within your visit directory which bears the same name as your visit, *e.g.* sm12345-1; this is to prevent the accidental overwriting of experimental data.

Opening data

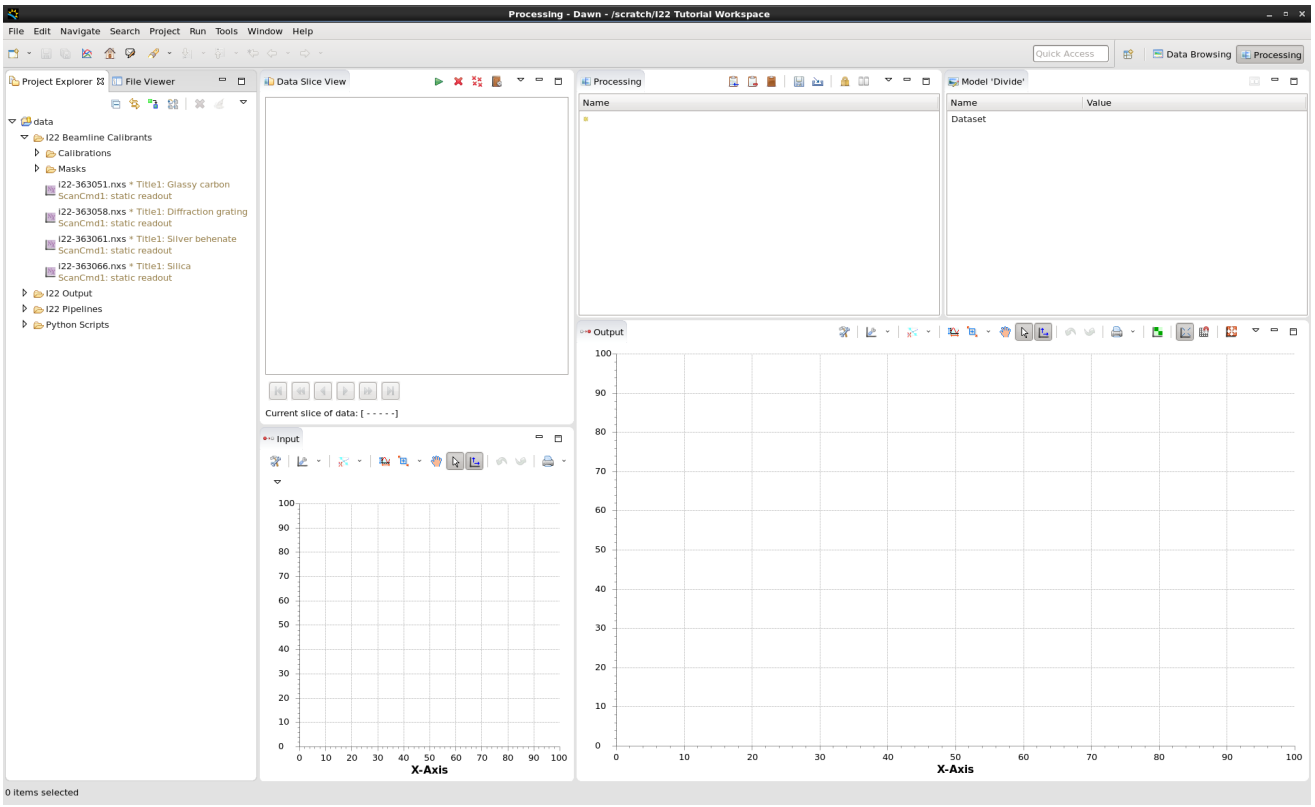
After loading up the processing perspective and customising the view from default, an interface similar to the one illustrated below should be displayed. As this section will rely on knowledge about the different sections of the processing perspective it is assumed that you have read Starting out with DAWN#ProcessingOverview.



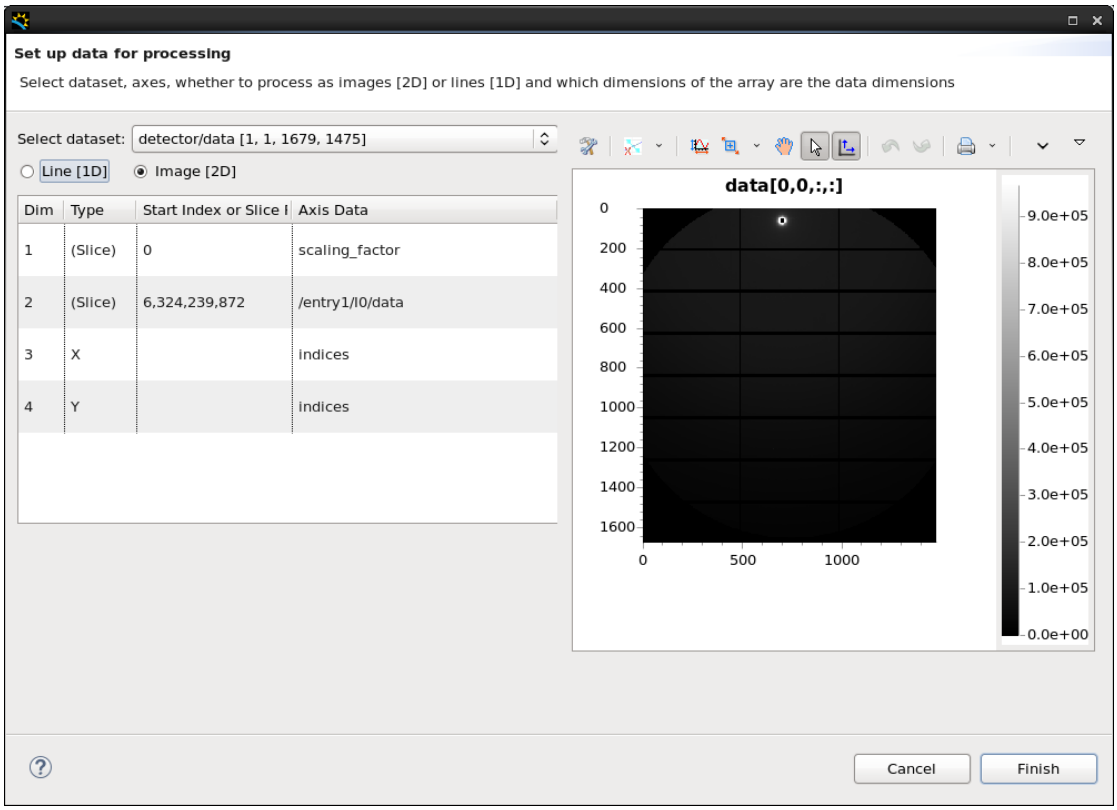
To start, you will need to import some data. This is achieved by dragging and dropping either individual files, or folders of files, into the *Project Explorer* panel. For simplicity, it is recommended that you drag and drop the entire experiment's folder into this panel. This folder will have the title of your experimental run *e.g.* sm12345-1. In these examples the folder of data is entitled *data*.



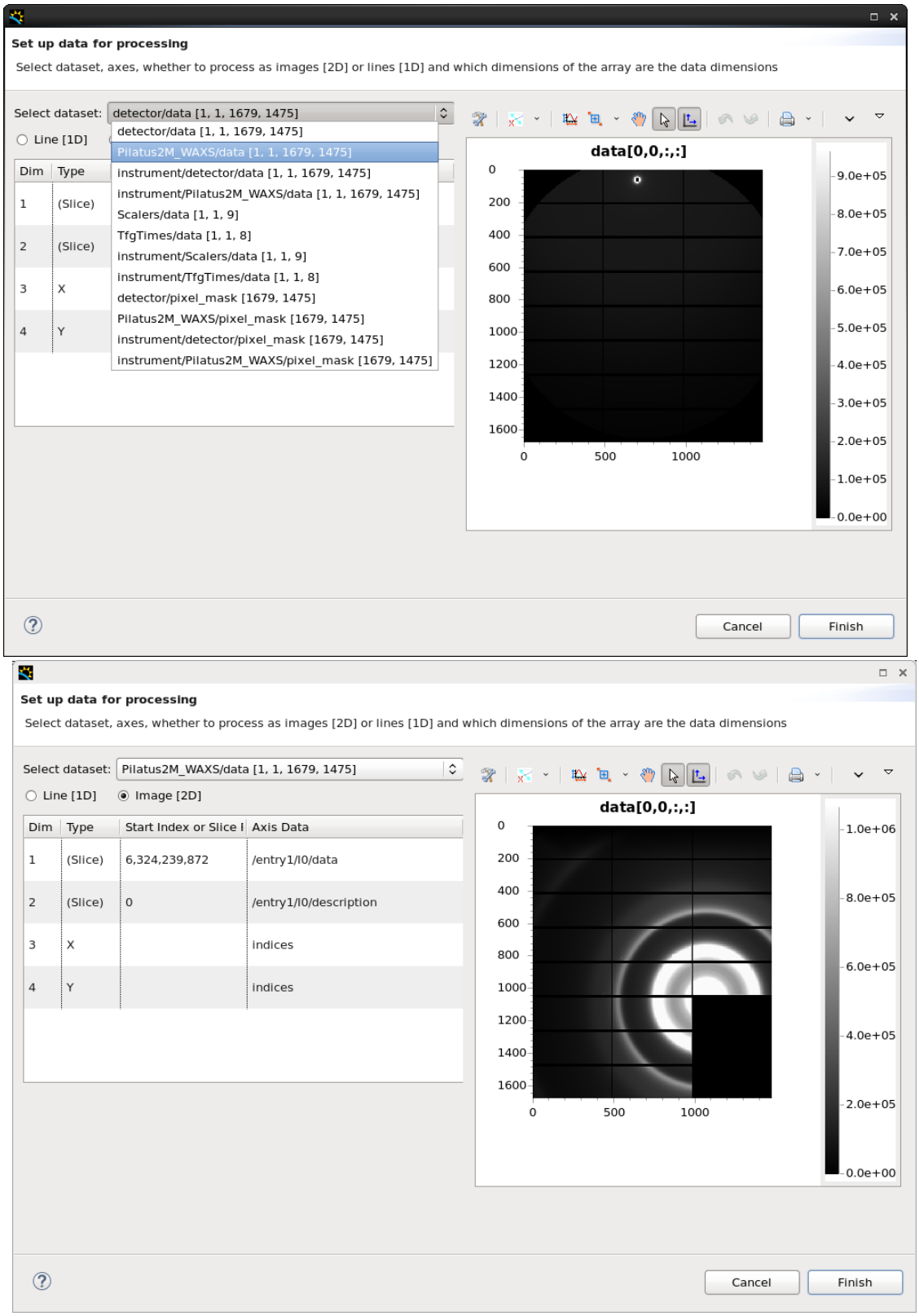
When you have dragged and dropped the folder into the *Project Explorer* panel, the window illustrated above will appear. Depending on where you have elected to store your data it may be practical to pick any of the above options. Typically two scenarios can easily be imagined: one where the computer running DAWN has a very large hard drive and a very slow connection to the data, which would lend itself well to copying the data to the DAWN computer. The other would be where the computer running DAWN has a small hard drive and a very fast connection to the data, which would lend itself well to linking to wherever the data is stored. Regardless, DAWN will represent the data in the same way regardless of whether it's copied or linked.



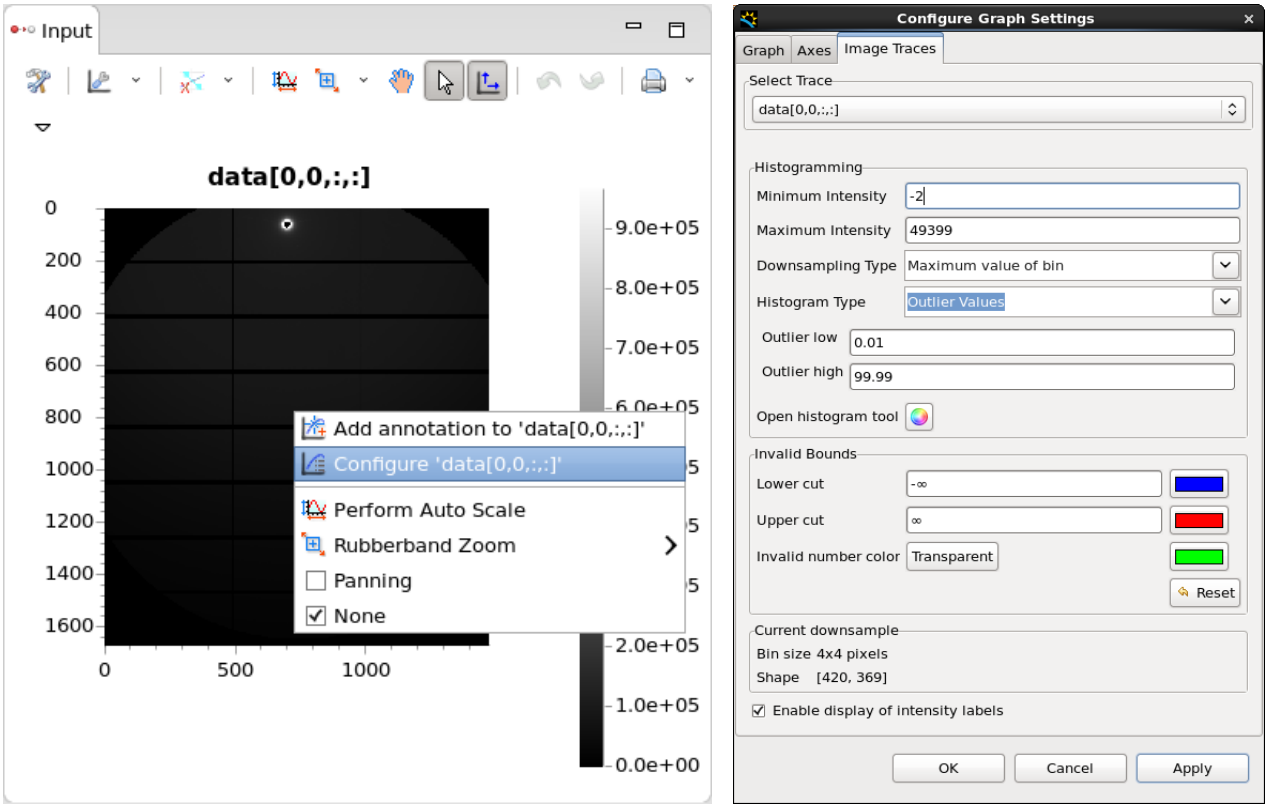
Once you have loaded your data you should have an interface looking somewhat like the above. Now in the *Project Explorer* panel the data is visible, and some information pertaining to the scan conducted is also displayed. Double clicking on a data file will present the following dialog box:



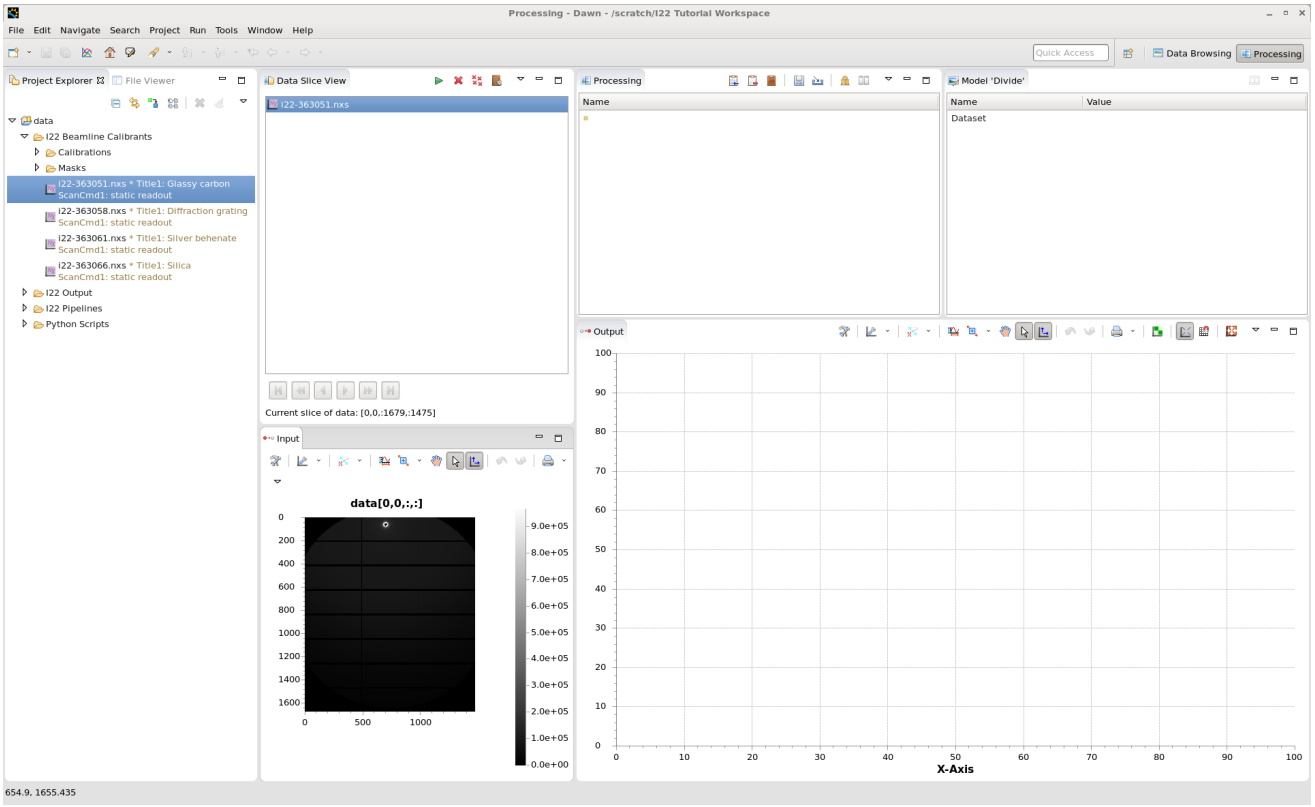
By default, the first dataset displayed is the data gathered from the SAXS Pilatus detector, a visual representation of this is shown on the right hand side of the panel. Clicking on the *Select dataset:* drop down list will present a list of options similar to the ones below, selecting the entry entitled *Pilatus2M_WAXS /data* allows the user to switch between viewing, and processing, either SAXS or WAXS data obtained at I22, the dataset *detector/data* contains the SAXS dataset.



The *datasets* and *Axis data* properties will be selected automatically, however, the more curious reader may be interested in more information about these, Loader variables, depending on the analysis you wish to perform selecting the correct axes can be *very* important. After selecting the data of interest, click *Finish* to proceed. If, like this example, the automatic histogramming has provided a dark image, or you would like to view your data in a different colour scheme, right click on the diffraction image as shown in the Input panel selecting the *Configure...* option from this menu. In the subsequent window it is possible to modify the graph as desired.



After customising your view you should now have a view similar to the one below, your data is now loaded into DAWN.



For more information about different ways to visualise your data, either raw or processed, the following links provide a wealth of information on how different perspectives in DAWN will allow you to interact with your data:

- Viewing results in DAWN#dataBrowser
- Viewing results in DAWN#DExplore
- Viewing results in DAWN#Trace

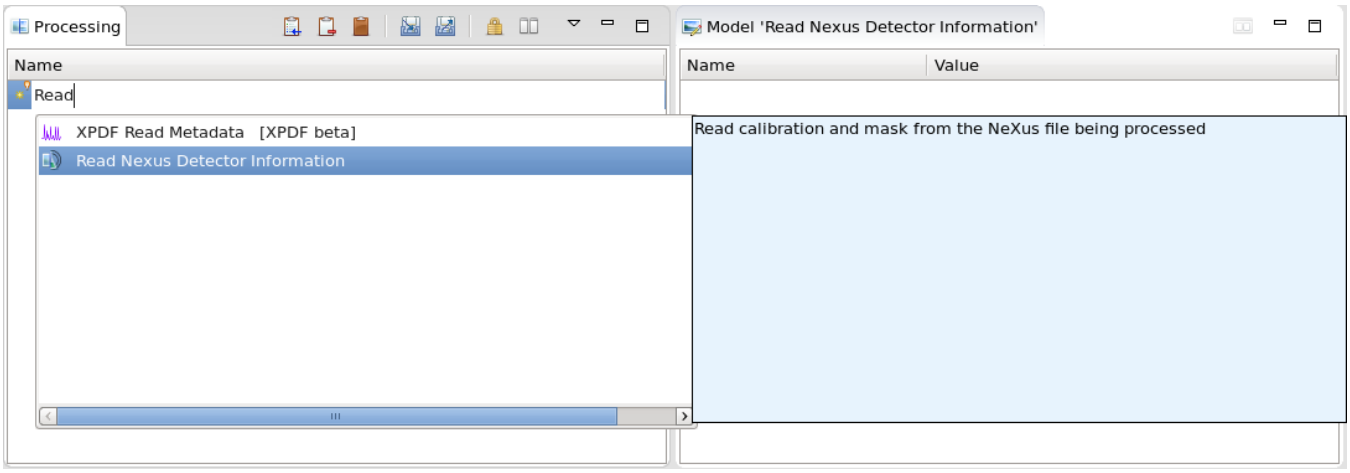
Multi-frame data

Certain measurements conducted on I22 collate together a number detector images into one NeXus file. In the *Data Slice View* panel DAWN facilitates navigation through these frames *via* the use of the buttons found at the bottom of the *Data Slice View* panel. The forward and backward play buttons allow for movement forwards or backwards a single frame, the fast-forward and rewind buttons skip through the data a set number of frames at a time and the skip to end buttons will take you to each respective end of the scan run.

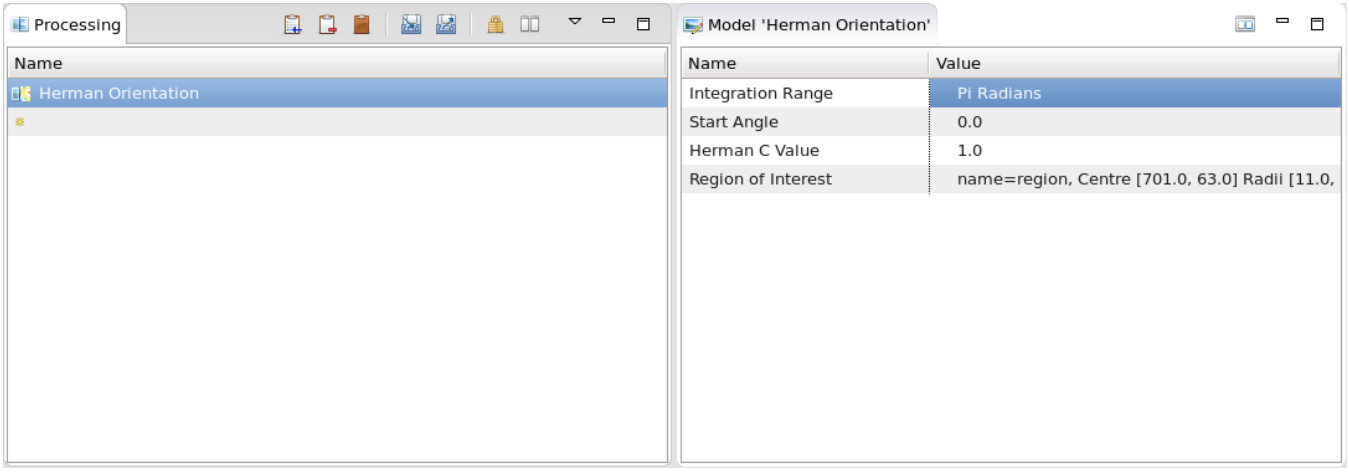
Underneath these buttons a line of text beginning *Current slice of data* will indicate your current location in the NeXus file. The second number inside the square brackets is the frame being currently viewed and/or processed, in the above image frame 0 was being viewed. In this example, as there is only one frame of data, these buttons are greyed out, however, these will play a role in later examples.

The processing pipeline

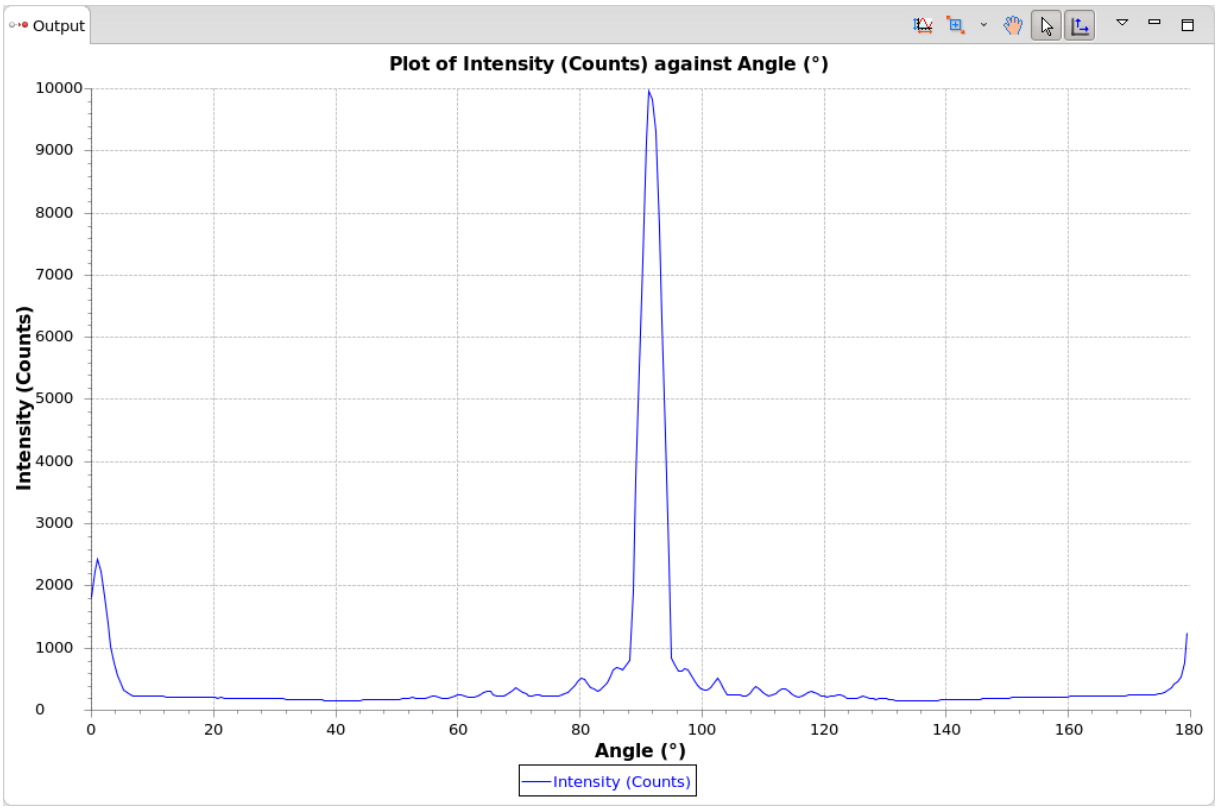
The *Processing* and *Model* panels are the centre of the processing perspective. Clicking the small star icon in the *Processing* panel will open a list of all the potentially available operations that you can currently perform on your data, a full list of which is available here, alongside a short explanation of the process selected. Typing part of, or the whole, name of a particular processing plug-in will search and filter the list of operations. In addition, the list is aware of the number of dimensions of data being passed to it and will also filter the potential operations accordingly.



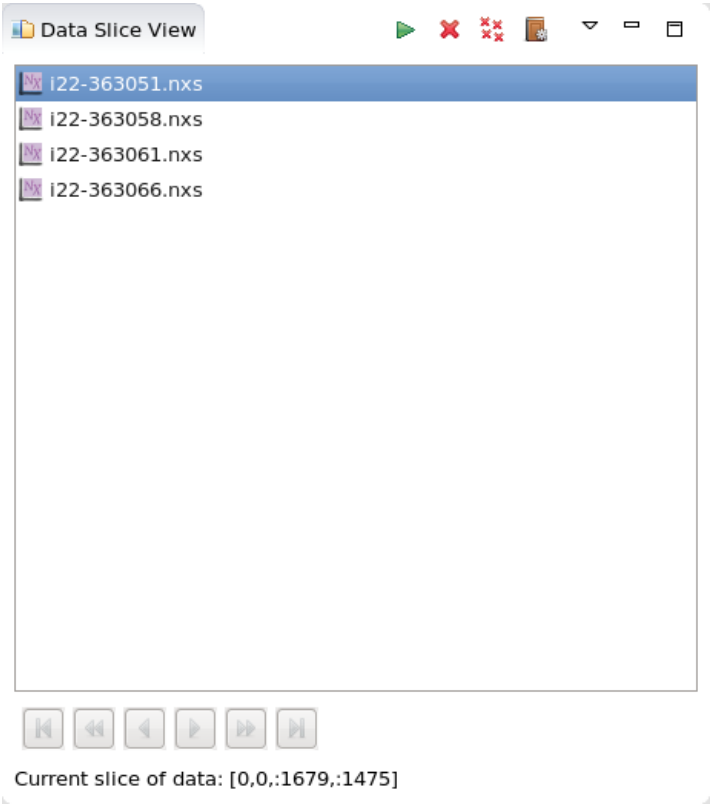
This list is referred to as the *processing pipeline* because you can graphically string together a number of processes to take, for example, a detector image through calibration, detector correction, reduction, normalisation and out to a delimited text file in one operation. These pipelines can then be run on either individual files or number of files, allowing for batch processing.



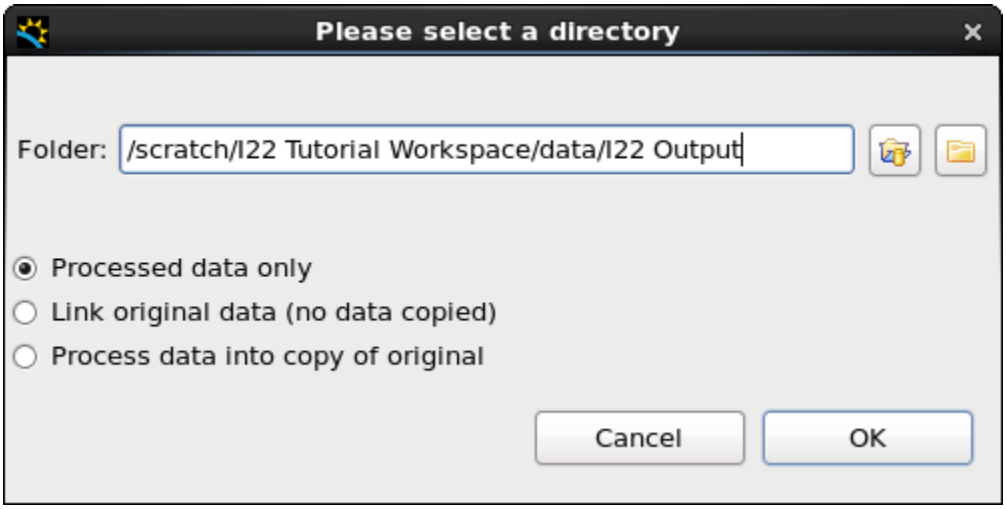
After selecting a process to perform on your data, in the *Model* panel options, if any, for the process will appear. In the example presented above the plug-in requires four pieces of information in the form of a drop-down list, two numerical entry fields and a *Region of Interest* entry field. Defining a region of interest is covered in the next section of this guide. After the required information has been provided, a visualisation of the result is provided below in the *Output* panel.



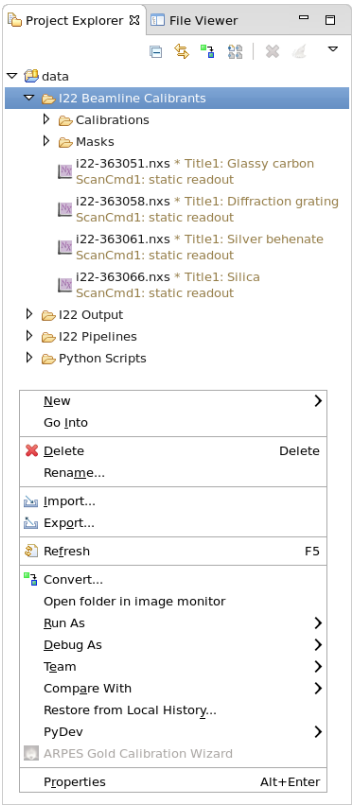
Finally, after a pipeline has been constructed, and any potential values for the process steps defined, it is then possible to process files. Previously, when discussing the *Data Slice View* panel it was omitted that this also is a pipeline, but rather than a pipeline of *processes* it is a pipeline of *files*. After loading in a single file, subsequent files placed in the panel, either by double clicking on the file or drag-and-dropping, will open in the same manner to the first file in the panel, i.e. if the first file is set up for SAXS processing, all subsequent files will be also.



Once both a process and file pipeline is set up it is then possible to run the desired operations on the scan files by clicking the green play button at the top of the panel.



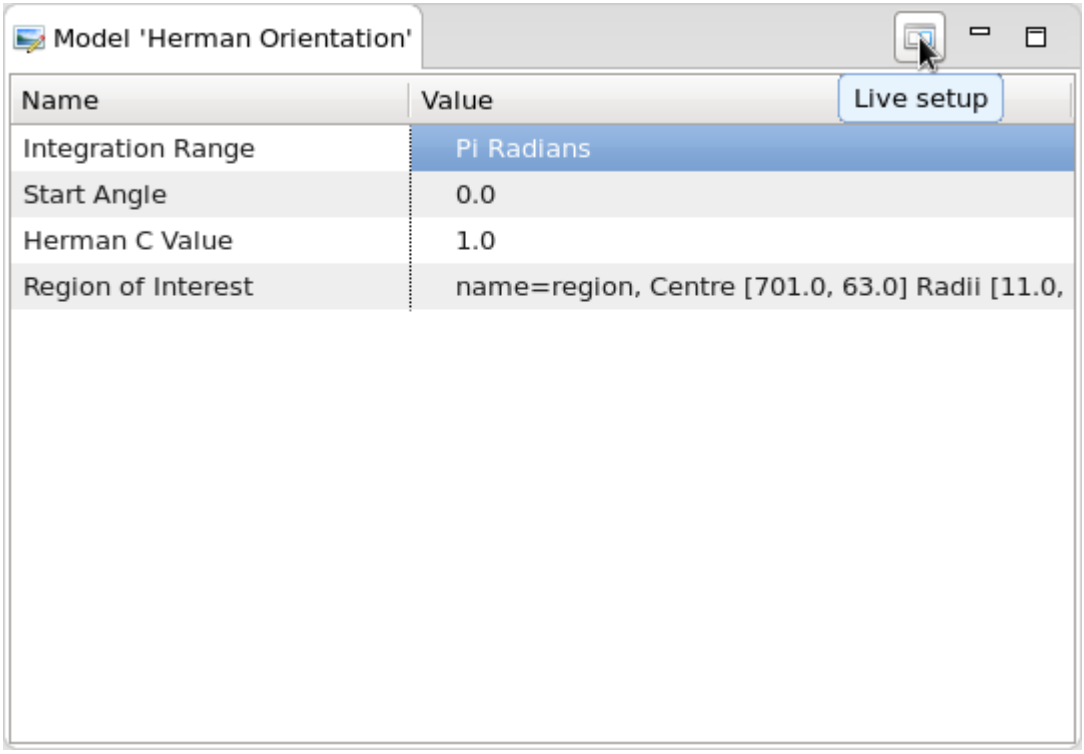
A prompt will then appear to ask where the results should be placed, this folder does not have to be within the DAWN workspace folder, however, if the save location is outside you will have to drag-and-drop the results file back into DAWN manually after processing. If the files are placed within the DAWN workspace, after processing right-clicking on the *Project Explorer* tab and selecting *Refresh* from the contextual menu will show the newly processed files, as shown below.



The newly processed files share the same name as the source file with the suffix *_processed_YYMMDD_HHMMSS*, appended to the file name.

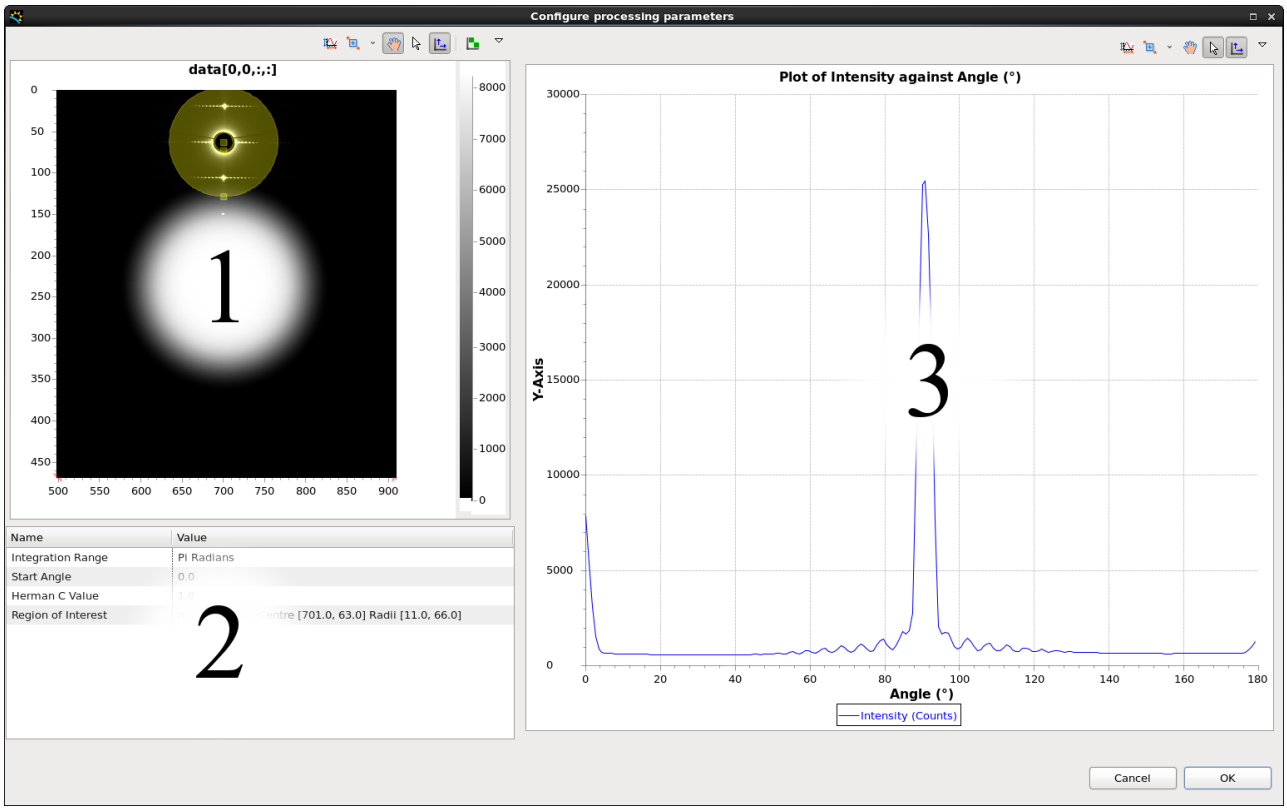
Defining a region of interest

There are occasions where it is desirable to define a region of interest on an image for investigation as opposed to analysing the entire image, to facilitate such analyses DAWN has an interface to click and select a given region of interest. To access this interface click on the *Live setup* icon in the *Model* panel, as illustrated below.

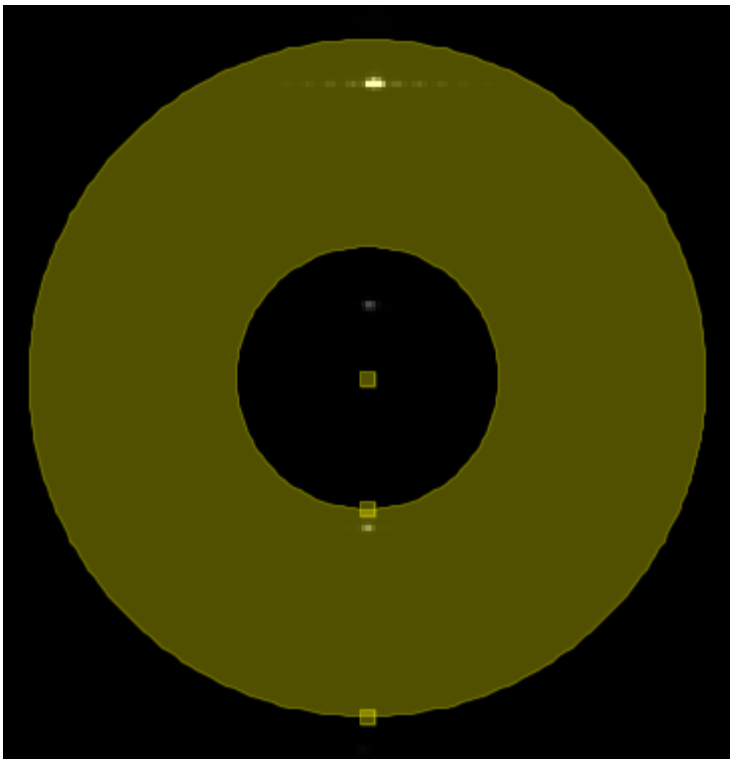


This processing step presents a typical *Live Setup* window consisting of3 main areas, as illustrated below:

1. **Input** - The detector image
2. **Model** - The processing plug-in model panel
3. **Output** - A representation of the output, given the region of interest defined



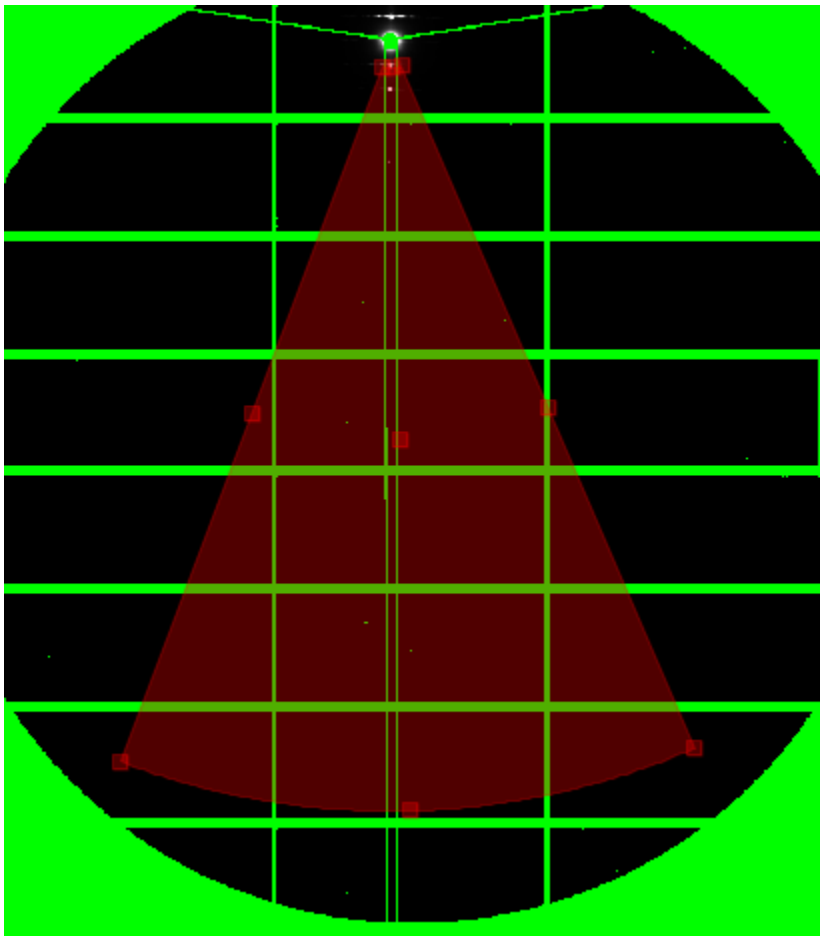
In this example, the plug-in selected makes use of a ring region of interest and so a ring with 3 draggable *handles* is provided. The central handle (draggable square) moves the entire region and handles present at the the inner and outer limits of the ring control the size of the highlighted region of interest, as below.



DAWN also provides other region of interest tools, these are:

- Elliptical
- Linear
- Perimeter box
- Point
- Rectangular
- Ring
- Sector
- X axis box
- Y axis box

Typically the processing plug-in will be limited to the use of one of these, if more than one is usable the plug-in will provide a way of switching between these in the plug-in model panel. Another common region of interest is the *sector*. In a similar fashion to the ring tool, there is a central handle, and a number of other handles allowing manipulation the region of interest as a function of both distance from the beam centre and angle around the beam, as shown below. For the more curious reader, the green areas on the detector image below are areas that have been masked out, this will be covered on the next page of this guide.



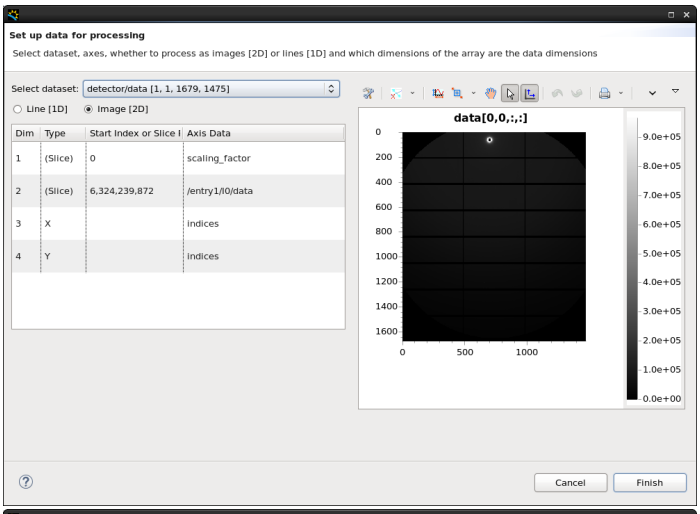
Should a region of interest need to be defined for a particular process, it is encouraged that familiarisation of the different options, and their variables, is undertaken first.

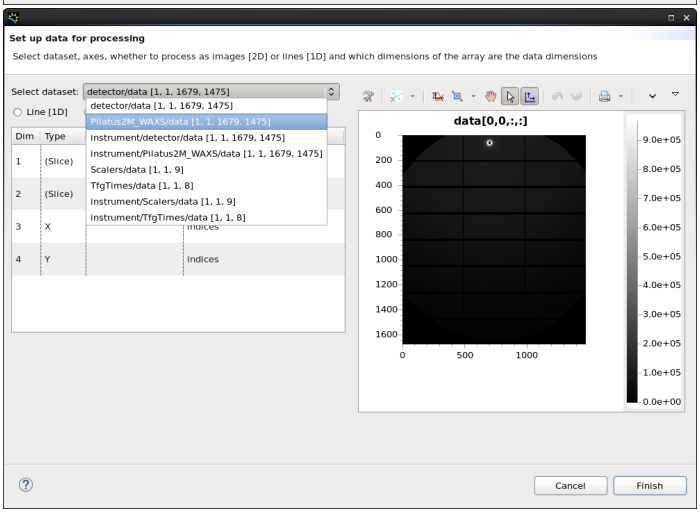
Previous: Starting out with DAWN | **Next:** Reducing data from I22 using DAWN

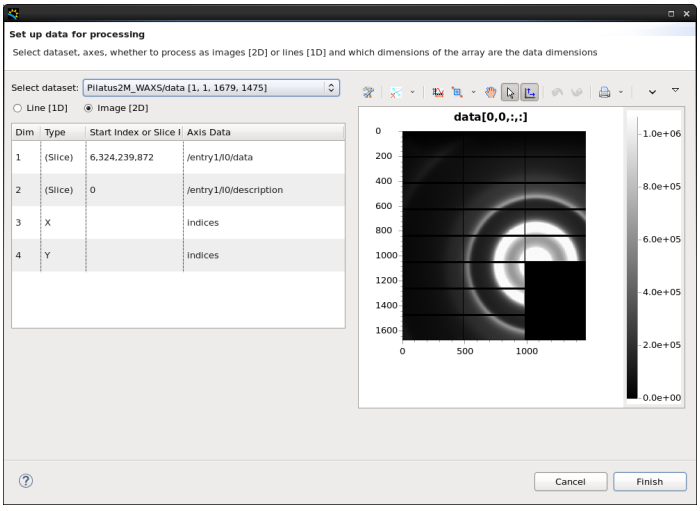
Loader variables

Dataset variations

The images presented on the previous page show a typical list of datasets that would be present for an experimental scan, as shown below, auto population is determined from the file contents, therefore another potential view of this interface could be:







In the drop-down box after the entries for *detector/data* or *Pilatus2M_WAXS/data* that there are a series of numbers inside square brackets such as [1, 1, 1679, 1475], as above, or a larger set of numbers such as [2, 2, 1, 1679, 1475]. Each of the entries in these *arrays* correspond to either physical space co-ordinates, frames taken at those co-ordinates or detector pixel values. Typically a 4-dimensional array (*i.e.* [a, b, c, d]) contains the following information:

- Number of points scanned in one axis of movement - default value of 1
- Number of frames acquired at each point in physical space - default value of 1
- Number of pixels along the detector's y axis - default value, for a Pilatus 2M detector, 1679
- Number of pixels along the detector's x axis - default value, for a Pilatus 2M detector, 1475

For the example at the top of this page, only one point in space was measured, with only one frame taken, using a Pilatus 2M detector. This gives an array of [1, 1, 1679, 1475], which corresponds to the size shown after the *detector/data* or *Pilatus2M_WAXS/data* entries above. If a line scan, of 10 points, was performed along the x or y axis in physical space with 100 frames taken at each point in physical space this would give an array of size [10, 100, 1679, 1475]. If a grid scan was performed, *i.e.* a scan where both x and y in physical space are scanned, then an additional dimension is added to the array giving a five dimensional array (*i.e.* [a, b, c, d, e]).

It is important to note that for the first and second dimensions, in a 4D array, where the size of the dimension is greater than one the entry under the *Type* column should be set to *Range*, if not already set, to allow DAWN to access all of the data in this dimension. In a 5D array this will apply to the first, second and third dimensions.

In addition it is worth noting that for the first two rows in the setup window, by default, in the *Axis Data* column the motor responsible for the movement in that dimension *should* be listed as this will allow you to map the data later on. This is important as mapping produces physical space maps showing where the data was obtained.

Previous: The Processing perspective | **Next:** Reducing data from I22 using DAWN

Reducing data from I22 using DAWN

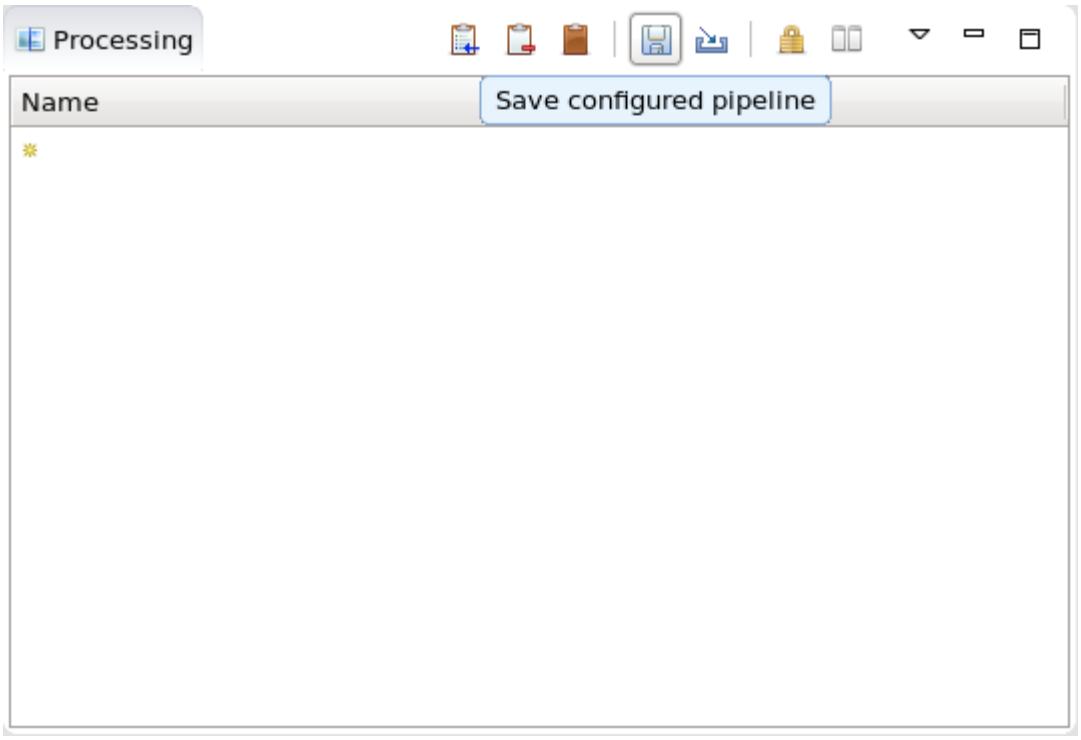
This page provides information on the following:

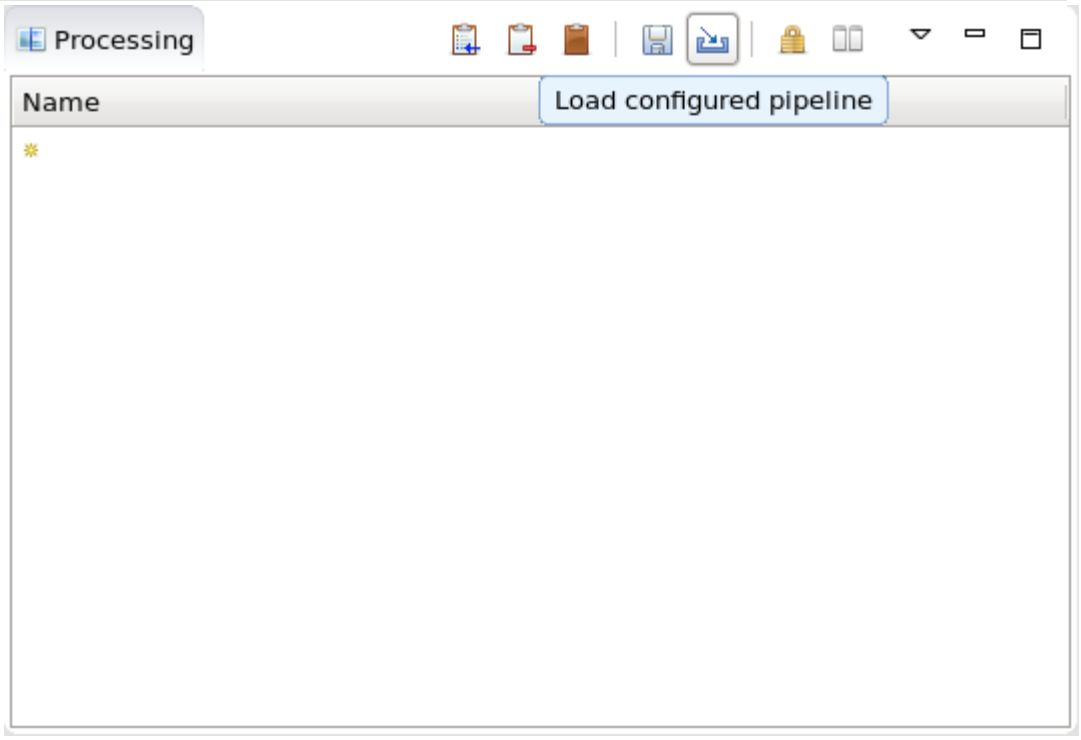
All instructions provided were performed on a Linux version of DAWN 2.7.x. Although there are no specific alterations to the way the program works between the operating systems, certain parts of DAWN might look different on your computer to the illustrations in this guide. A DAWN workspace accompanying this section of the guide can be downloaded I22 Tutorial Workspace.zip. Alternatively, just the data files highlighted below can be downloaded as a zip file I22 Beamline Calibrants.zip.

N.B. It is **assumed** that you have read the guide to this point. If you have not done so it is **highly recommended** as certain assumptions and terminology may appear alien unless you have read the guide to this point.

Setting up the processing pipeline

To streamline the process of creating and running and re-running common, or highly specialised, pipelines in the *Processing* panel two buttons are present that allow a user to either save or load pipelines.

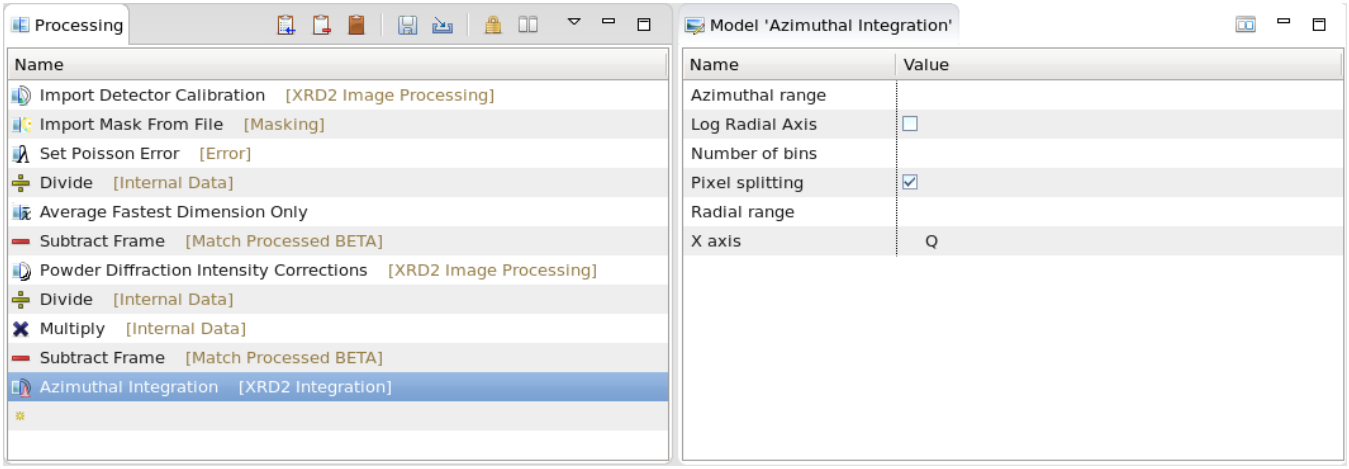




Therefore, if you require a simple reduction of either your SAXS or WAXS data, by far the easiest way to set up the example I22 pipeline is either to load the desired pipeline from the example dataset or to download one of our suggested pipelines from these links:

- Reduction - Absolute Units.nxs
- Reduction - AU - Frame Averaged.nxs
- Reduction - AU - FA - Background Subtracted.nxs
- Reduction - Transmission Corrected.nxs
- Reduction - TC - Frame Averaged.nxs
- Reduction - TC - FA - Background Subtracted.nxs
- Detector Image Correction - Absolute Units.nxs
- Detector Image Correction - Transmission Corrected.nxs
- Reduction - Uncorrected (**Use with caution**)

After downloading these files, loading their respective pipelines can be achieved by either clicking on the *Load configured pipeline* button in the top toolbar of the *Processing* perspective, or by dragging-and-dropping the file into the *Processing* perspective panel. After loading the required file, the *Processing* pipeline should fill with the required processing steps for reducing SAXS or WAXS detector data. Below is shown the default pipeline for data reduction to absolute units with frame averaging and background subtraction. **Please note:** these pipelines have had their q ranges tailored to the examples in this tutorial , this q range is entirely dependent on the sample to detector distance therefore before using these pipelines on your data **check the q range is appropriate**.



After loading the desired pipeline, depending on the reader's level of interest, it is possible to either read the next section to discover how to perform this step manually, with details about each plug-in and the values passed to it, skip to the section delineating how to output results as ASCII files or skip straight to the final section on reducing and exporting data.

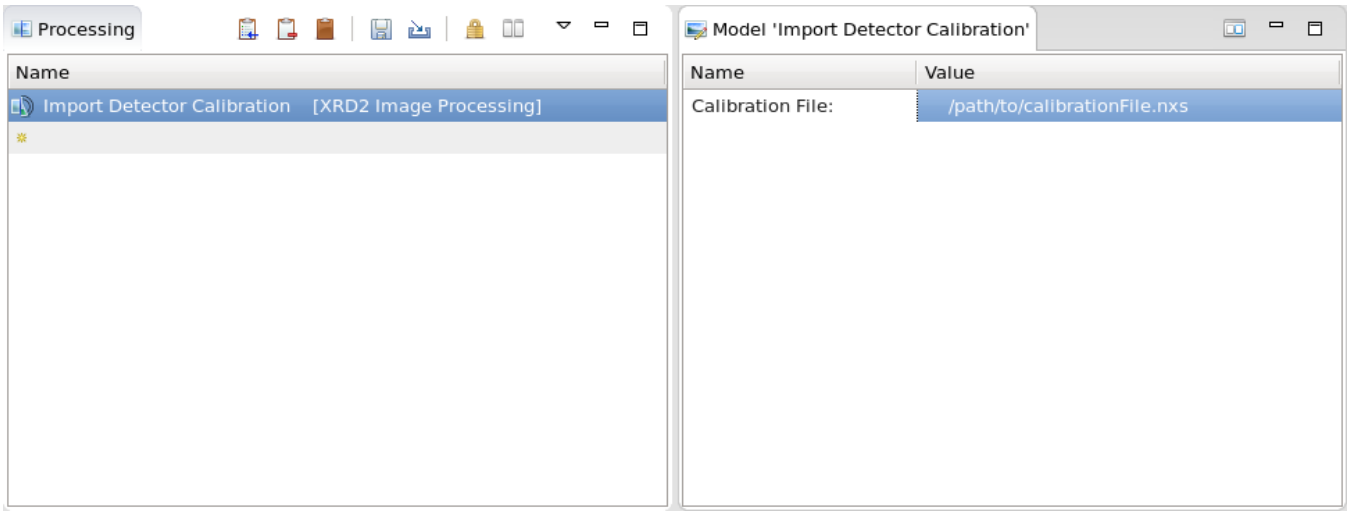
More information on the different methods for background subtraction available *via* DAWN can be found on the next page of this manual, Background subtraction.

Setting up the processing pipeline manually

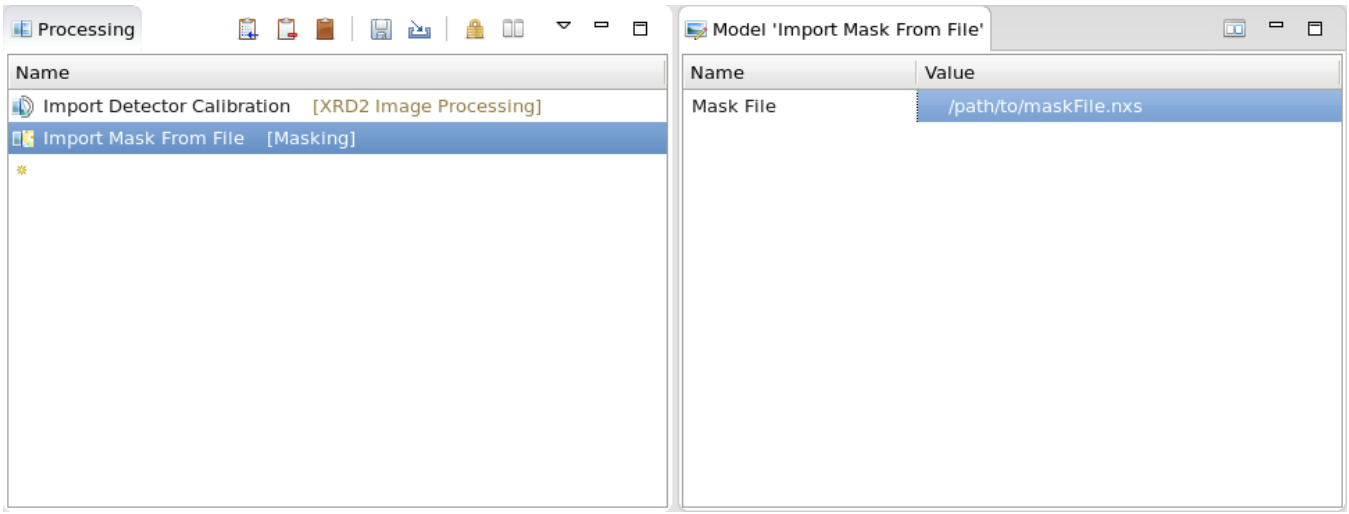
Any detector data obtained at Diamond is, by its very nature, raw information. As such, a number of corrections are required to take such data and transform it from raw information into the most accurate and precise dataset possible. A publication that outlines the subsequent steps that are recommended can be found in the following publication:

> Pauw, B.R., Smith, A.J., Snow, T., Terrill, N.J. & Thunemann, A.F. (2017). J. Appl. Cryst. 50, https://doi.org/10.1107/S1600576717015096

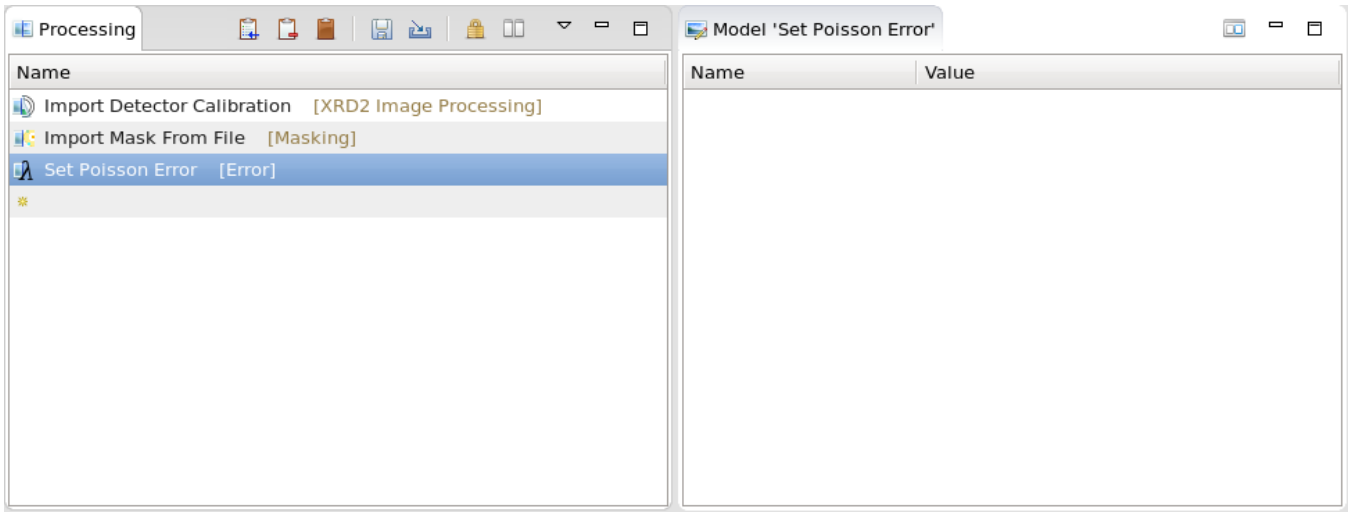
Following on from loading in the scattering pattern(s) to be reduced, the next two steps are to load in the detector calibration and masking information. Clicking on the small golden asterisk at the current end of the processing pipeline will bring up a dialog box of all the currently available processing steps for this dataset. Typing in *Import* at this time will filter this list to steps containing the word *Import* which should filter the list to *Import Detector Calibration* and *Import Mask From File*. Clicking on the *Import Detector Calibration* step will load this into the processing pipeline, as shown below.



With this step loaded, it is now possible to either drag-and-drop the detector calibration file into the *Calibration File*'s *Value* field in the step's *Model* or, alternatively, type in the file path to the calibration file manually. Following on from this, clicking on the asterisk again and filtering in the same manner will allow the reader to place in the *Import Mask From File* step into the pipeline, specifying the mask file in the step's *Model* in the same manner to the detector calibration file as highlighted below.

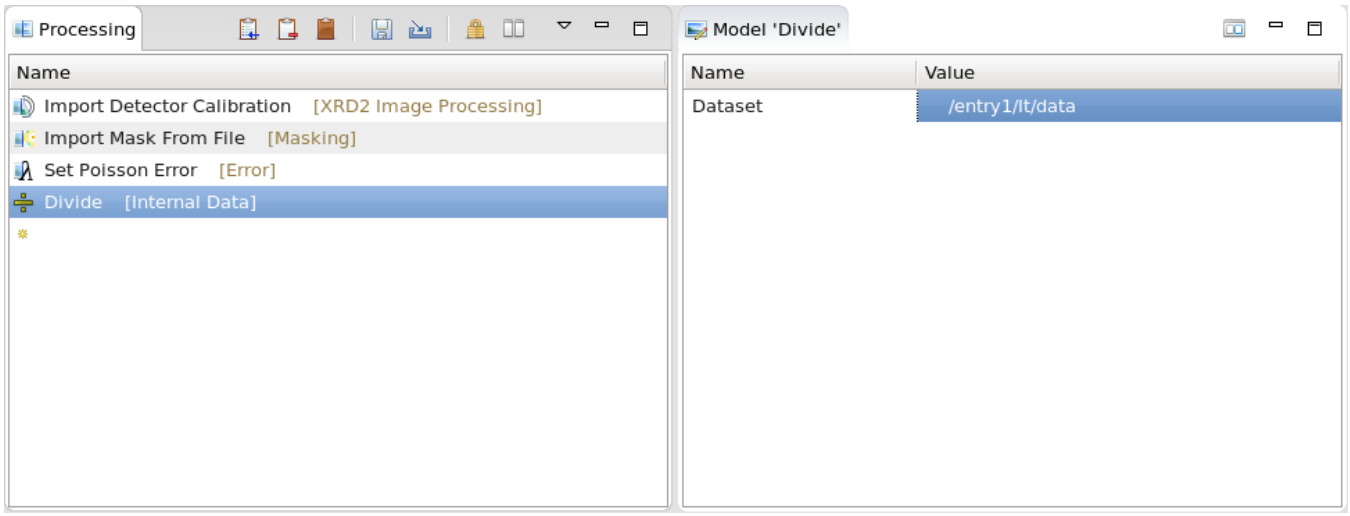


With these files loaded the next step is to calculate the Poission error for each pixel on the detector face, this is achieved *via* the *Set Poission Error* processing step, which has no options and simply has to be placed into the pipeline to calculate the required error values, as shown below.

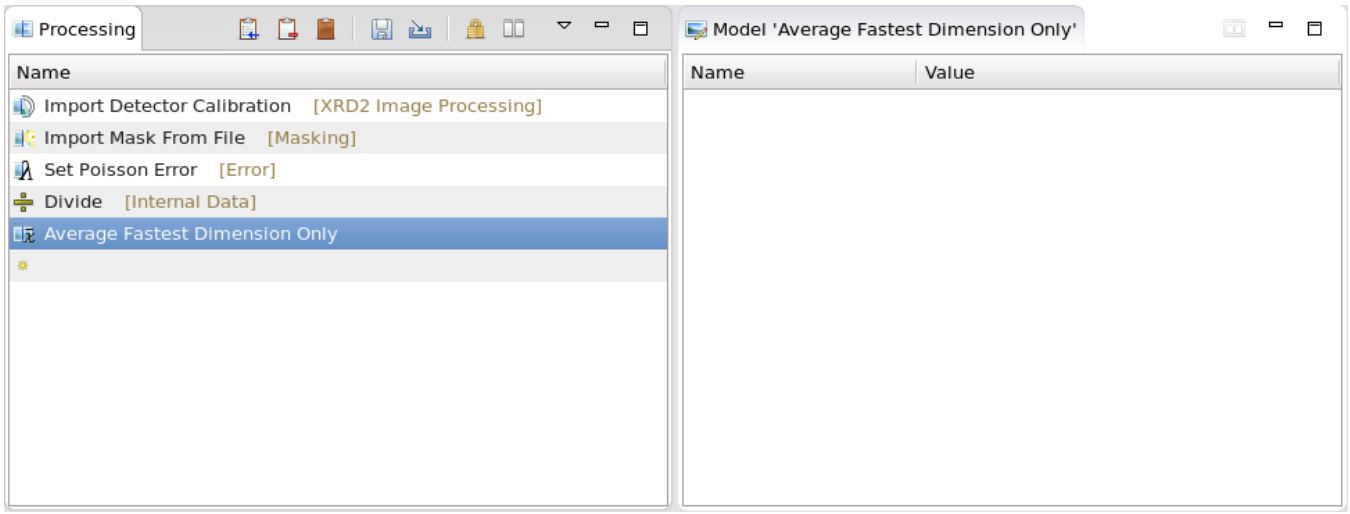


With these files loaded into the pipeline the next steps are to correct for dead time and dark current. The Pilatus software itself, currently, performs a dead time correction on the data collected from the detector so this step can be omitted. Furthermore, dark current measurements conducted on the I22 beamline reveal that for a given *hour* of data collection a dark current of 3 counts per pixel, peak value, is observed. Given that a typical second of data acquisition on I22 usually provides at least 10 counts of intensity per pixel applying this correction has such an infinitesimally small contribution (~8e-5 %) it will be omitted from this pipeline. Should you wish to conduct this correction on your dataset contact your local contact who will be able to provide dark current measurements. It is worth noting that for lab sources this is a much more significant source of error and should be accounted for.

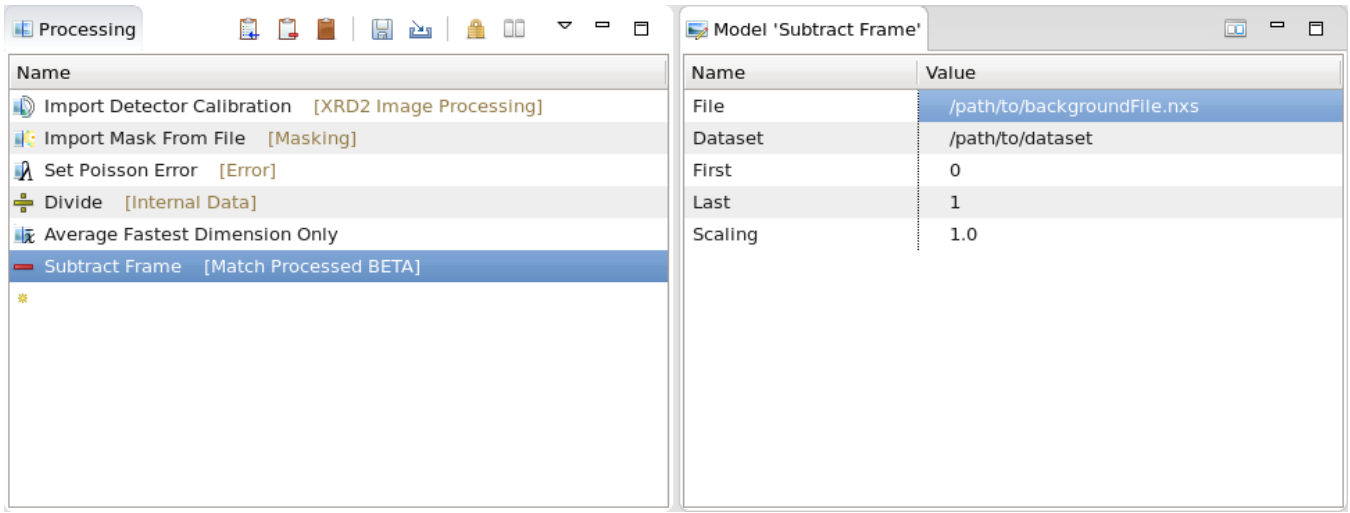
The next three corrections are for time, flux and transmission correction. Inspection of the mathematics that underpins the flux and transmission correction steps reveals that they can be simplified, to a division by the transmitted intensity (this is explained in greater detail in section 4 of the *J. Appl. Cryst.* publica tion). Furthermore, as the transmission diode on I22 records the integral of signal with respect to time, as opposed to a rate, division by the transmission diode reading accounts for time correction as well. As with the dark current correction, this will vary instrument to instrument and so knowledge of how the diode collects its value per frame is needed in order to deduce whether this is valid for data reduction elsewhere. However, for current I22 data all three of these corrections can be applied by placing a single *Divide [Internal Data]* step to the processing pipeline. In the Model panel the internal dataset should be set to */entry1/It/data* as this is where the transmitted intensity information is stored in the NeXus tree.



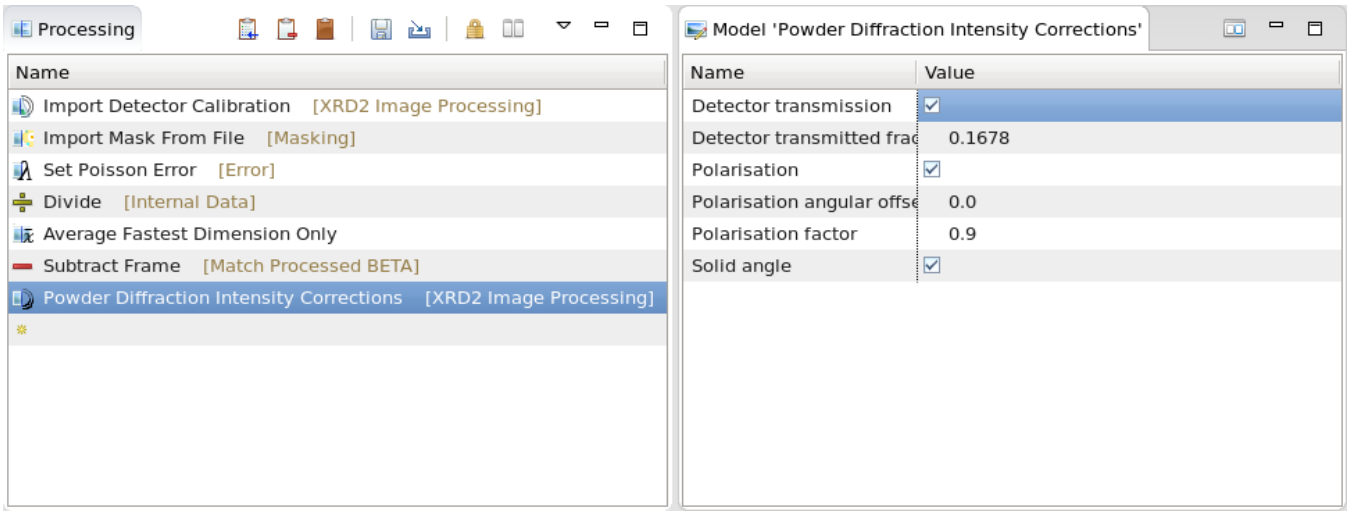
At this point an optional frame averaging step can be inserted, it is optional as the reader might only have obtained one frame of data or might not want to average the data if they anticipate changes with respect to frame (*e.g.* degradation of the sample in the beam). There are several data averaging routines available within DAWN, however, the *Average Fastest Dimension Only* step is recommended for use as this will only average all frames obtained at a given point in physical space (e.g. the x'th dimension in this 4D dataset w, x, y, z). This processing step has no configurable options, as highlighted below.



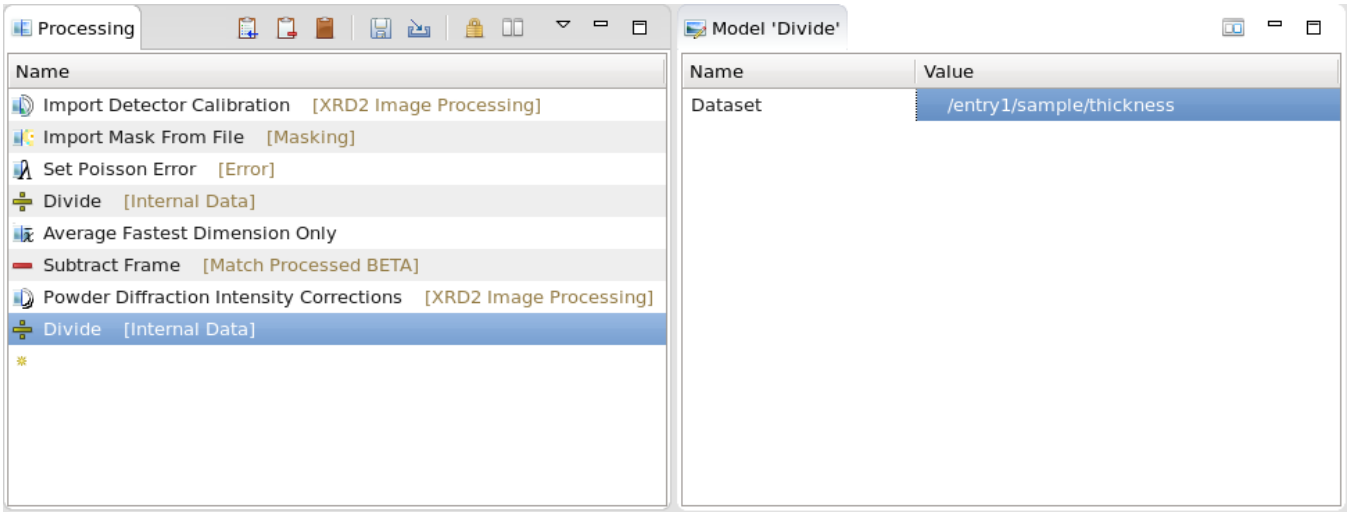
At this stage, should the reader wish to subtract a beamline background, and/or beamline and cell background, the next step required is *Subtract Frame [Match Processed BETA]*, multiple backgrounds can be removed by placing multiple instances of this step into the processing chain. It is worth noting that if this is to be performed the first subtraction should always be a beamline/air background, with a cell background following afterwards. Similar to the frame averaging step, this too is optional if the reader is not interested in subtracting a background, or backgrounds, from their data. Similar to the *Import...* steps previously, into the *File* field it is possible to drag-and-drop the background file, or specify the precise path to the background file. Additionally the dataset within the file that is to be subtracted must also be specified, for I22 this will typically either be */entry1/detector/data* or */entry1/Pilatus2M_WAXS/data*. This step will then apply all of the previous steps in the chain to this file and subtract the result from the currently processed frame. **N.B.** if a number of background subtractions are to be performed, with averaging, on files containing a large number of frames these steps will take *considerable* time to perform.



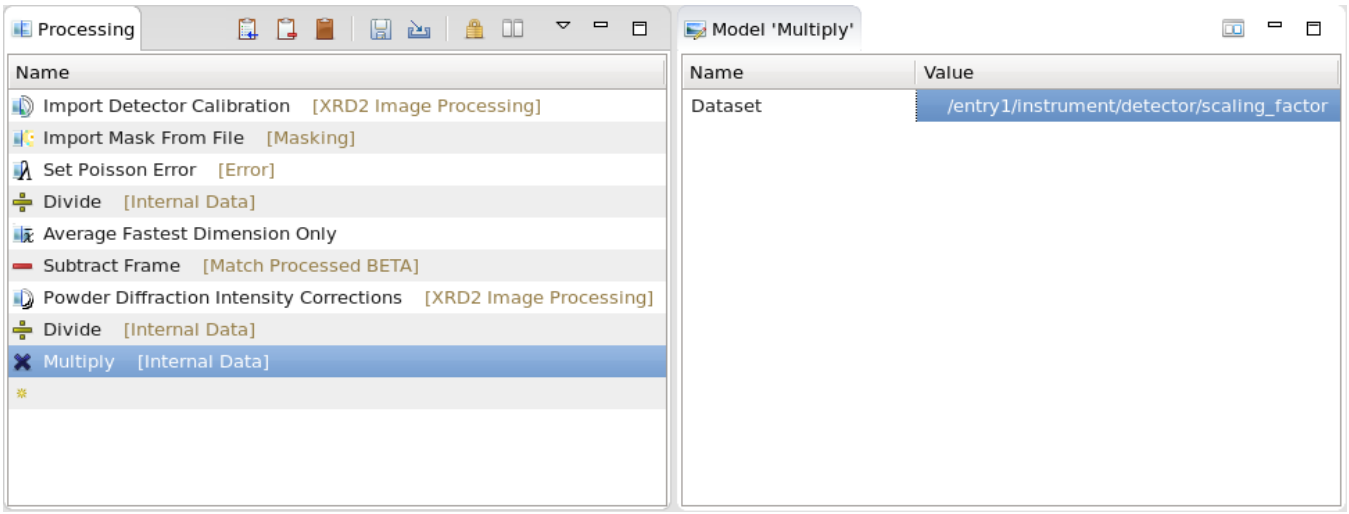
The next step delineated in the literature is a flatfield correction, however, similar to the dead-time correction this is automatically applied by the Pilatus software to the detector images and, therefore, will be omitted. Following on from this, 3 steps: angular efficiency, solid angle correction and polarisation will be applied from the same processing step entitled *Powder Diffraction Intensity Corrections*. These three steps can be applied by ticking the pre-requisite boxes and knowing the required values for both the beam and detector used. All experiments at I22 will present a beam with a polarisation factor of 0.9, however, the detector transmission factor will change with respect to energy, with 0.1678 corresponding to the value for 12.4 keV (1 Angstrom). Factors for other energies can either be calculated (the detector face is made from silicon) or obtained from your local contact.



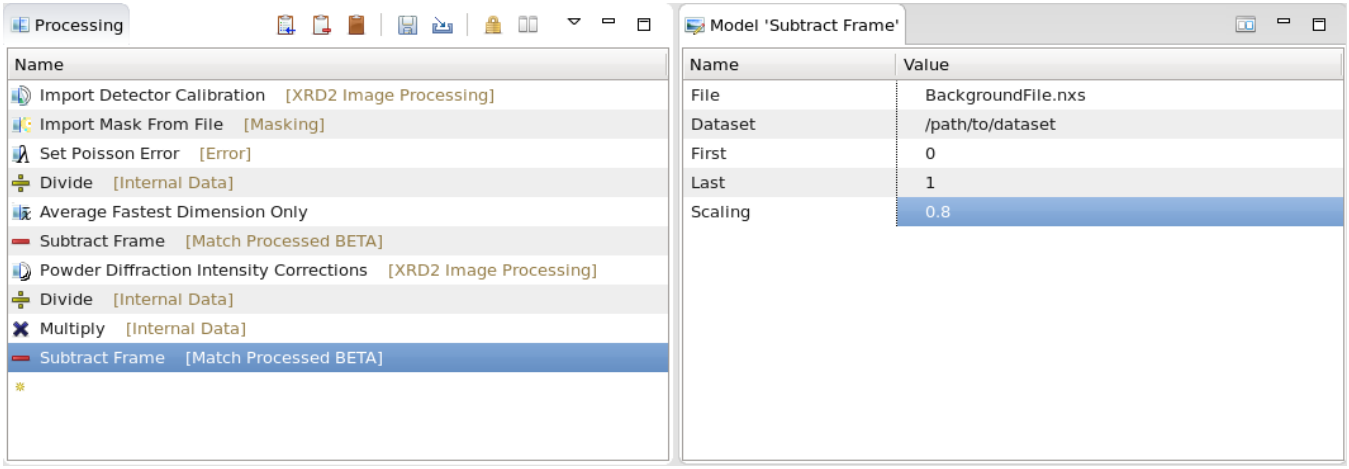
The next two steps are optional as not all readers will want, need, or be able to transition their recorded intensities from arbitrary units to absolute units, primarily because an accurate measure of the thickness of the sample being scanned is required. However, should the sample thickness be known and was entered at the time of scanning adding in a *Divide [Internal Data]* step into the pipeline will allow for scaling of the dataset by the sample thickness. In the Model panel the internal dataset should be set to */entry1/sample/thickness* as this is where the sample thickness information is stored in the NeXus tree. t If the thickness of the sample is known but it was not entered at the time of scanning it is possible to manually enter in the thickness by placing a *Divide by Scalar* step into the processing pipeline, however, employing this technique either requires the input data to be of all the same thickness or the data to be processed file by file.



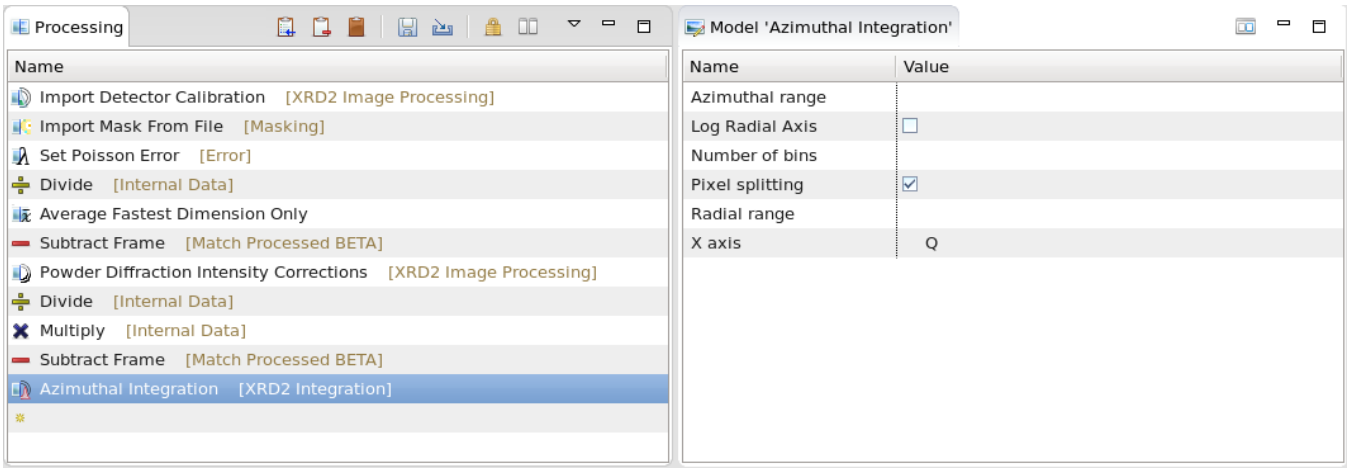
Subsequently it is then necessary to scale the intensity data against a standard calibrant which is typically for data gathered at I22 a glassy carbon sample from NIST. As this scaling factor should have been calculated as part of your experimental setup all that is required is to place a Multiply [Internal Data] step into the processing pipeline to scale the data accordingly. In the Model panel the internal dataset should be set to */entry1/instrument/detector /scaling_factor*.



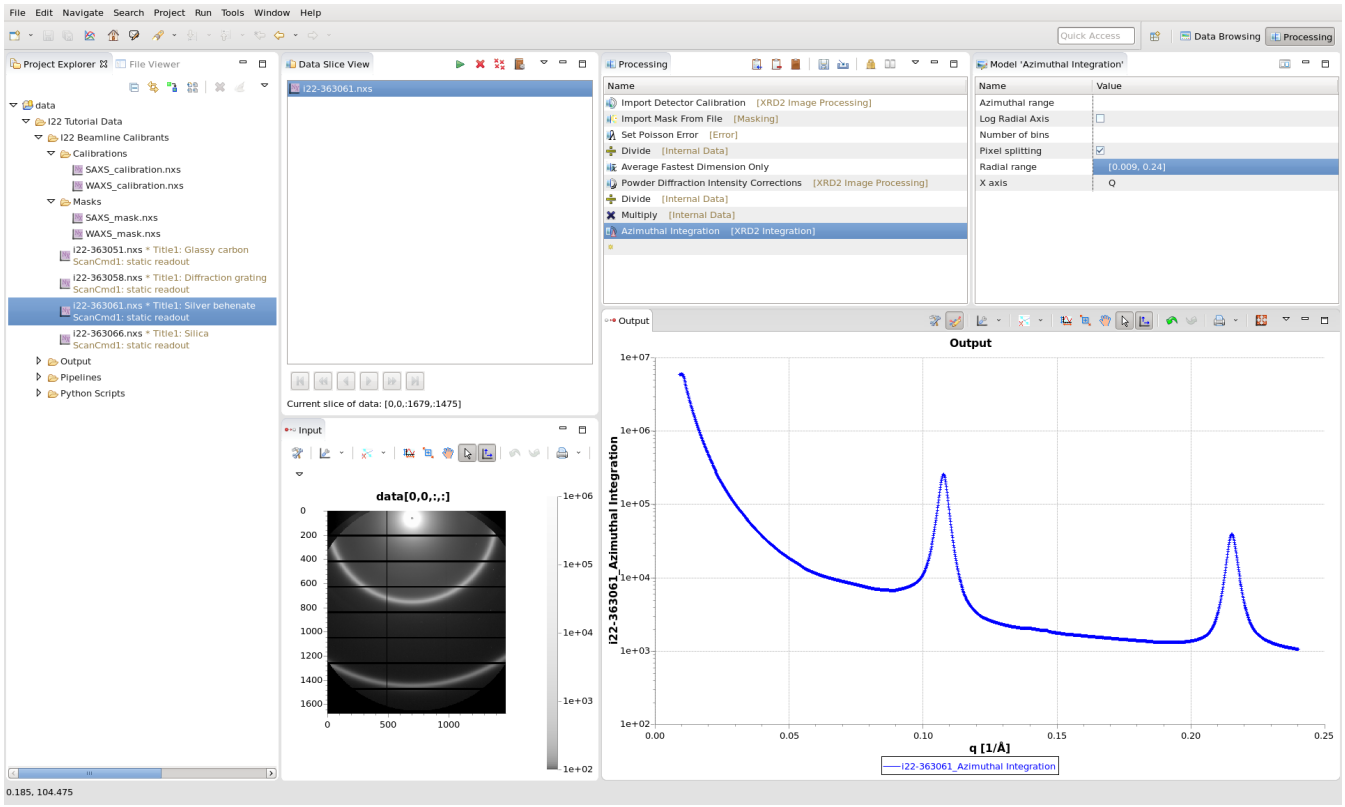
At this point, should the reader have a sample in a solvent, subtraction of this background should be performed here by placing into the pipeline another *Subtract Frame [Match Processed BETA]* step. This works in exactly the same manner as the preceeding subtraction step, however, should the sample under investigation be of a high concentration use of the *Scaling* option will allow for correction of the amount of solvent scattering to be subtracted as depleted volume effects will typically lead to over subtraction if ignored. In the example below a scaling factor of 0.8 is applied which would correlate to a sample where 20 % of the solvent has been displaced by analyte.



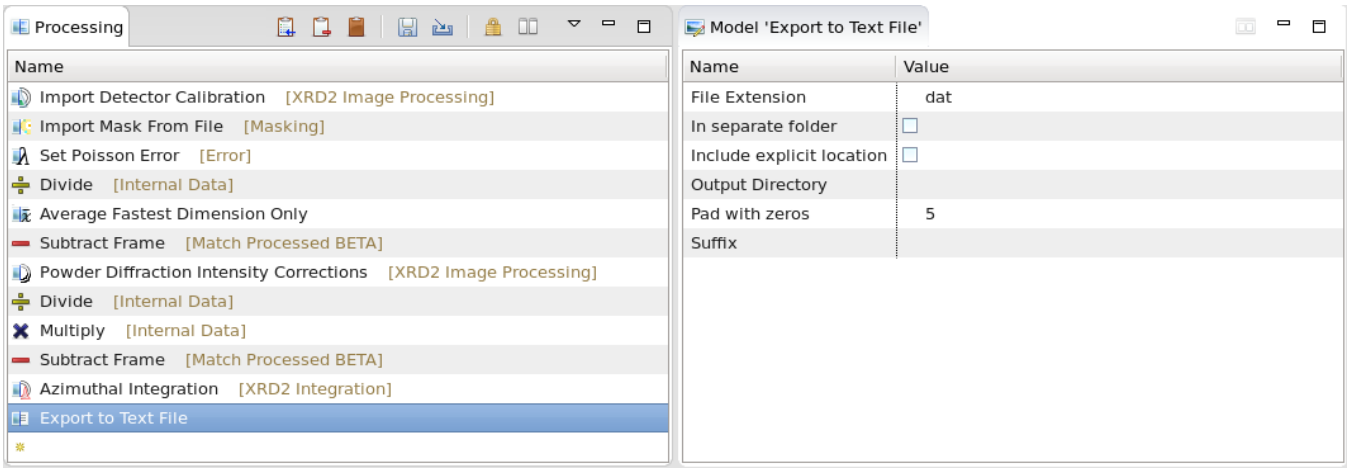
Following on from all of these corrections the final step is to reduce the data from a 2D image to a lineplot *via* either the *Azimuthal Integration [XRD2 Integration]* processing step to produce I vs. q data, or a *Radial Integration [XRD2 Integration]* step to produce I vs. χ data. Below is highlighted the azimuthal integration step.



At this point the data shown in the *Output* pane presents a preview of the reduced data, as highlighted below. Running the *Processing* pipeline on the files in the *Data Slice View* pane, as explained in the next section, at this point will reduce the data and then store the results in NeXus files for further analysis in DAWN or other programs that can read NeXus files. However, it is also anticipated that it will be desired to export the reduced data for use in other programs.



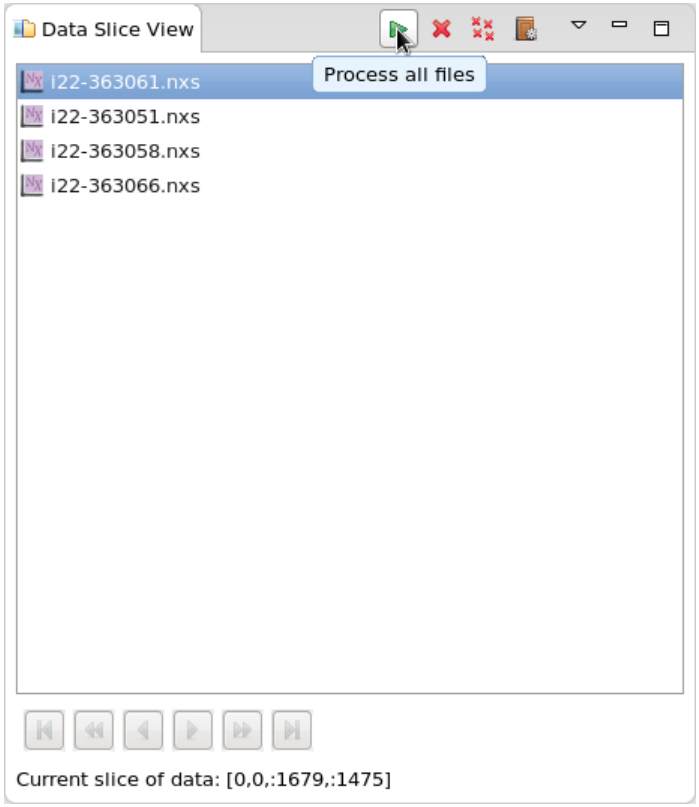
In order to facilitate this the *Export to Text File* plug-in can also be added to the *Processing* pipeline. As before, click on the small star icon in the *Processing* pane and enter in part of, or the whole, plug-in name to filter from the list of available plug-ins and double click on it to enter it into the pipeline. This plug-in will produce a ASCII text file, tab delimited, for each of the reduced frames. Each file contains three columns of data: 1) q (or other requested x-axis); 2) I; 3) I_{err}. In the *Export to Text File Model* pane it is recommended to change the File Extension from *dat* to *txt* and to set an output directory, in order to separate the text files from the NeXus files. The more curious reader may notice that the output plot featured in the above image has the y-axis plotted logarithmically. A quick shortcut to achieve this view is to click on the plot and press the y (or x) key on the keyboard. Alternatively, more options are available by right clicking on the plot and looking through the featureset presented under the *Configure 'SCAN_NUMBER'* option.



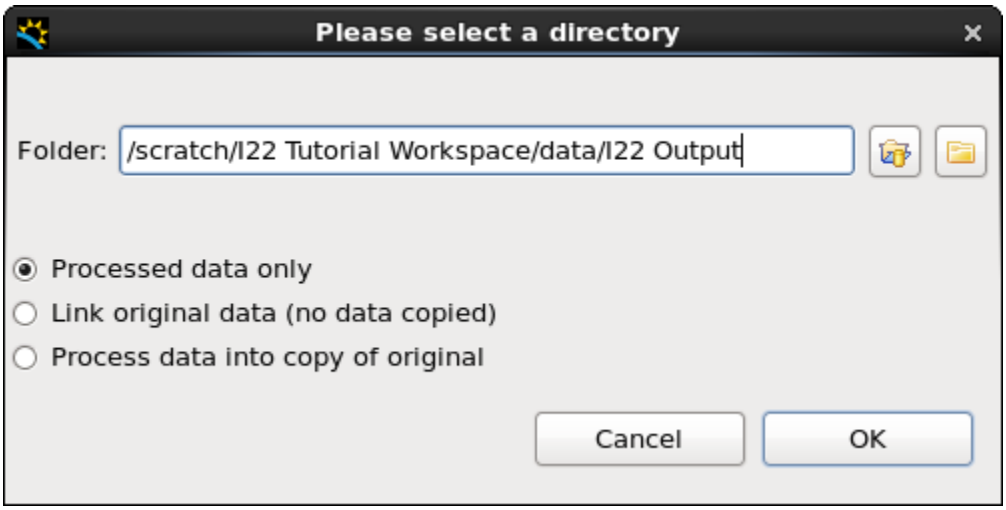
Once the pipeline has been filled and all the potential values in the *Model* pane for each plug-in have been set as desired, proceed to the next section.

Reducing and exporting

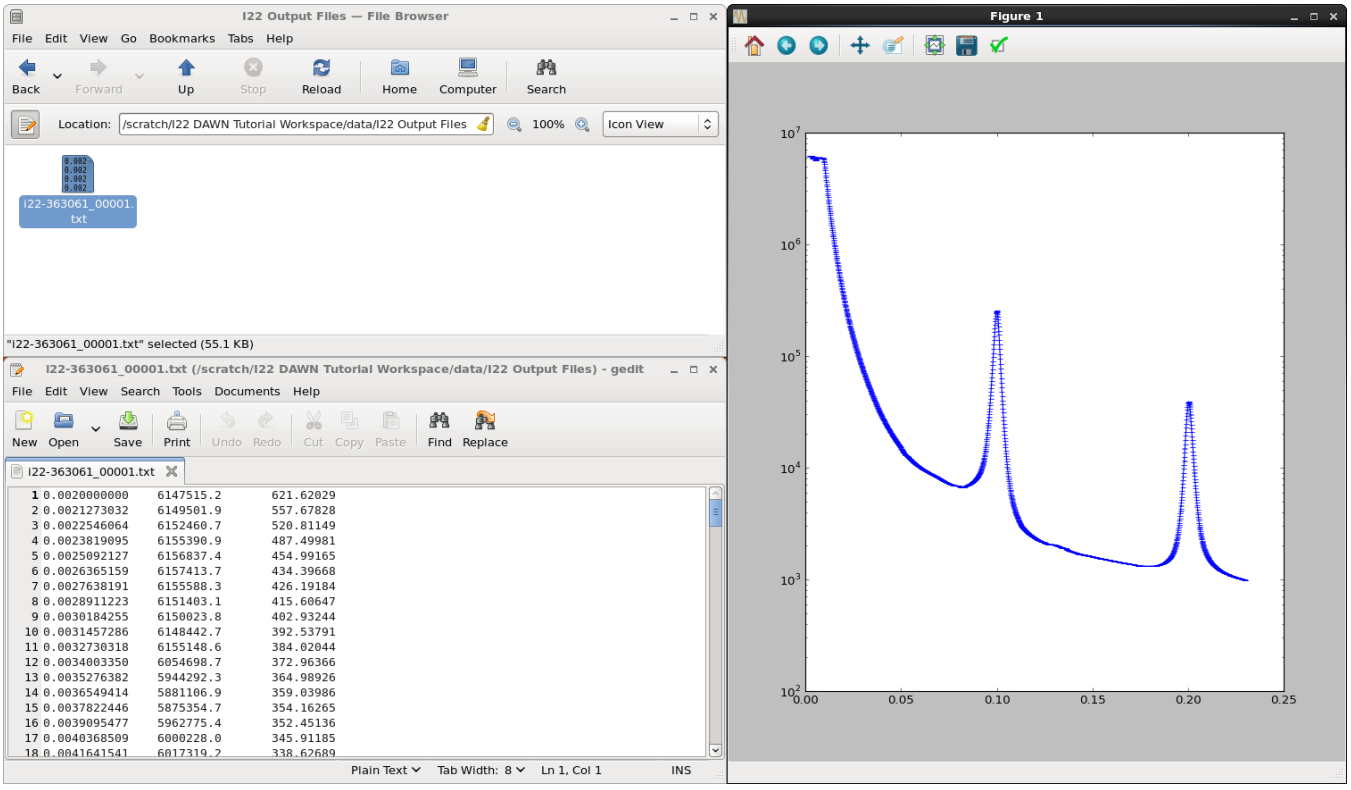
Now that the *Processing* pipeline is populated and ready to reduce data, drag-and-drop all the files that are desired to be reduced into the *Data Slice View* panel. It is important to remember (link) that all the NeXus files placed in this particular processing pipeline will be reduced in the same way, with only SAXS *or* WAXS data analysed per run. When the desired files have been loaded clicking the *Process all files* button, highlighted below, in the *Data Slice Viewer* panel will start the reduction process.



After clicking the start button the below dialog box will appear **please note** this dialog box only applies to any potential NeXus file output, **not** any ASCII file output which is defined in the *Model* panel for the *Export to Text file* plug-in. After clicking *OK*, depending on the number of reductions performed, either a window will appear to inform you of the current progress of the operation or you will be returned to the *Processing* perspective of DAWN. When you are returned to the *Processing* perspective, the reduction process has finished.



Subsequently, refreshing the view in *Project Explorer* perspective should show both the *i22-RUNNUM_processed_YYMMDD_HHMMSS.nxs* file(s) potentially in addition to text files generated, as show below on the left. Navigating to the directory containing any potential text files in a system file explorer will reveal the text files ready for opening in a text editor of choice, or opening in any analysis or plotting package. Below is highlighted some data as seen after processing in the DAWN in an operating system file explorer a text editor and plotted in Python using *matplotlib*, right.



In order to view the reduced data, or to perform additional analysis, in DAWN please read the next section of this guide entitled **Data Analysis**. Alternatively, in order to discover how to export previously reduced NeXus results files, or how to export the detector images for publication please continue on to the Exporting reduced data or detector images of this guide.

Previous: The Processing perspective | **Next:** Background subtraction

Background subtraction

This page provides information on the following:

All instructions provided were performed on a Linux version of DAWN 2.7.x. Although there are no specific alterations to the way the program works between the operating systems, certain parts of DAWN might look different on your computer to the illustrations in this guide. A DAWN workspace accompanying this section of the guide can be downloaded by clicking on this link. Alternatively, just the data files highlighted below can be downloaded as a zip file I22 Beamline Calibrants.zip.

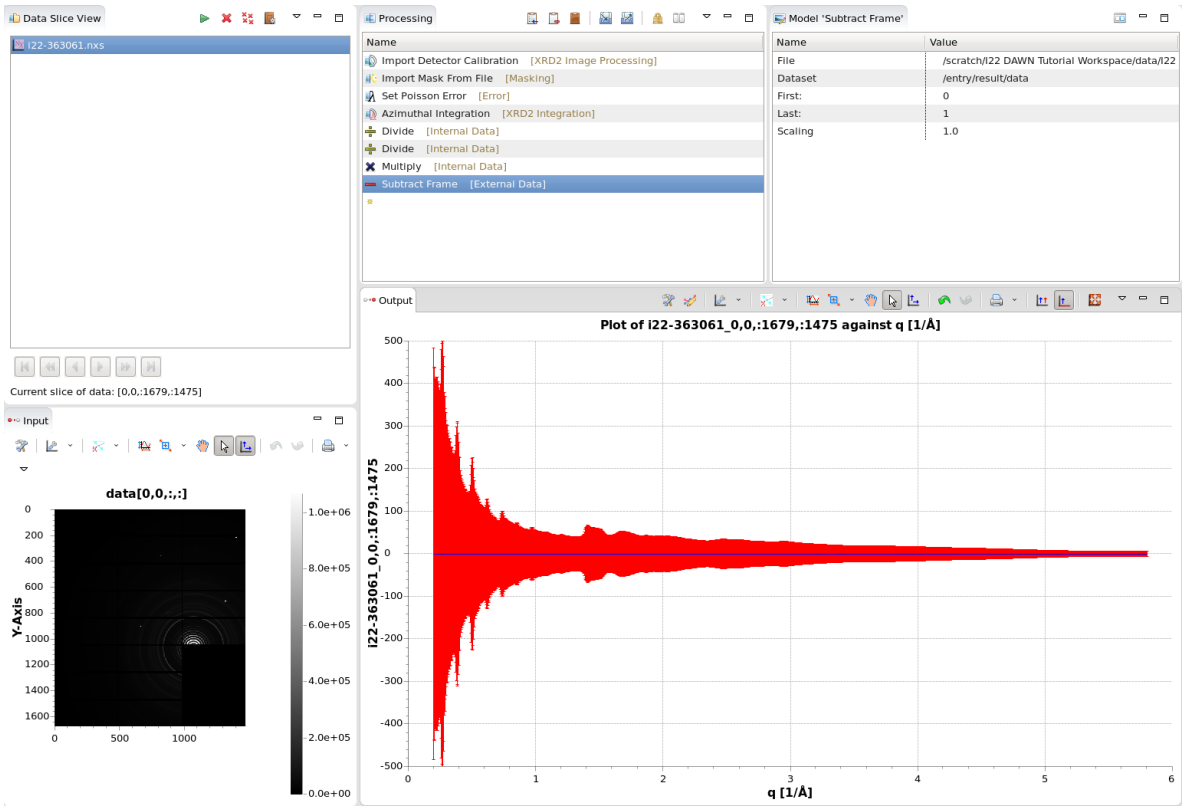
Introduction

As with many scientific endeavours there are many potential approaches that, overall, achieve the same operation. Each approach is typically either uniquely appropriate or specifically tuned for the operation, however, the principle of what is being attempted is the same. Background subtraction for x-ray data analysis is a typical example of such an operation. This section hopes to provide some insight into several different methods that one can adopt in order to perform a background subtraction, however, a degree of *caveat emptor* applies.

Background frame subtraction

In the *Setting up the processing pipeline manually* section of the previous page one approach to background subtraction was covered, that of direct frame subtraction applying any previous corrections to a given dataset. In addition to this it is also possible to subtract a frame from another frame, without applying any preceeding corrections (*via* the *Subtract Frame [Enternal Data]* step), and simply to subtract a constant value from a dataset (*via* the *Subtract Scalar* step), however, as this will subtract a fixed value from the entire dataset this provides a blunt instrument towards background subtraction.

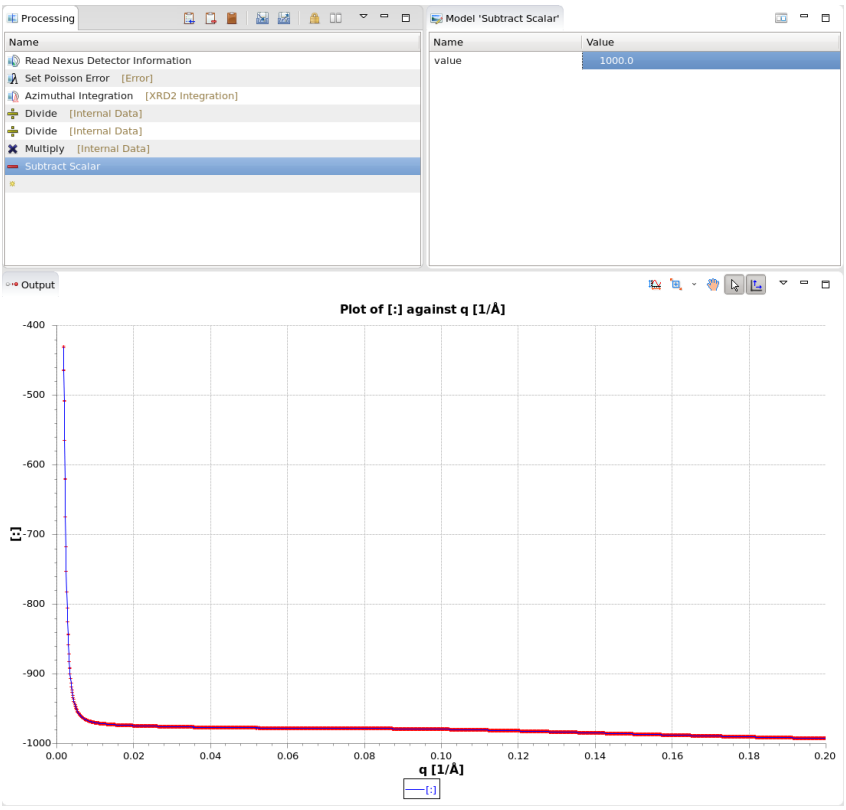
It is anticipated that the most commonly used of these two techniques will be the *Subtract Frame* option, as this will subtract a given frame from the the frame currently being worked on. Typically, a frame that would be used for this subtraction would be an average of a background, such as an air or sample cell background. Loading this into the processing pipeline is achieved in the usual manner from within the *Processing* perspective of DAWN.



Upon loading in the *Subtract Frame* plug-in the following options are available in the *Model* panel:

- **File** - The file containing the reduced background data
- **Dataset** - The internal NeXus path to the reduced data, typically */entry/result/data* or */entry1/detector/data*, upon typing in the first / a list will appear delineating the file's contents
- **First** - In a multiframe results file, the first frame to average, leave blank to average over all frames
- **Last** - In a multiframe results file, the last frame to average, leave blank to average over all frames
- **Scaling** - A scaling factor to apply to the data before subtracting

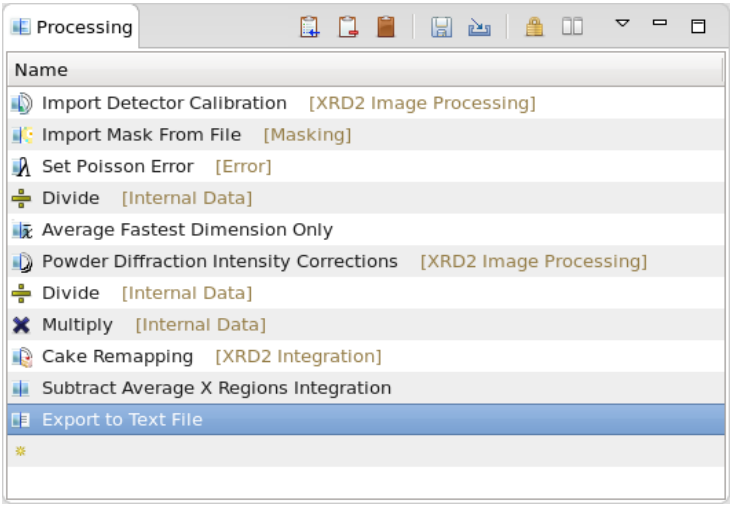
In the above example, as the dataset being reduced and the subtracted values come from the same image the subtracted intensity is zero, however, as we propagate errors there is uncertainty in this result, hence the red error bars dominating the *Output* plot.



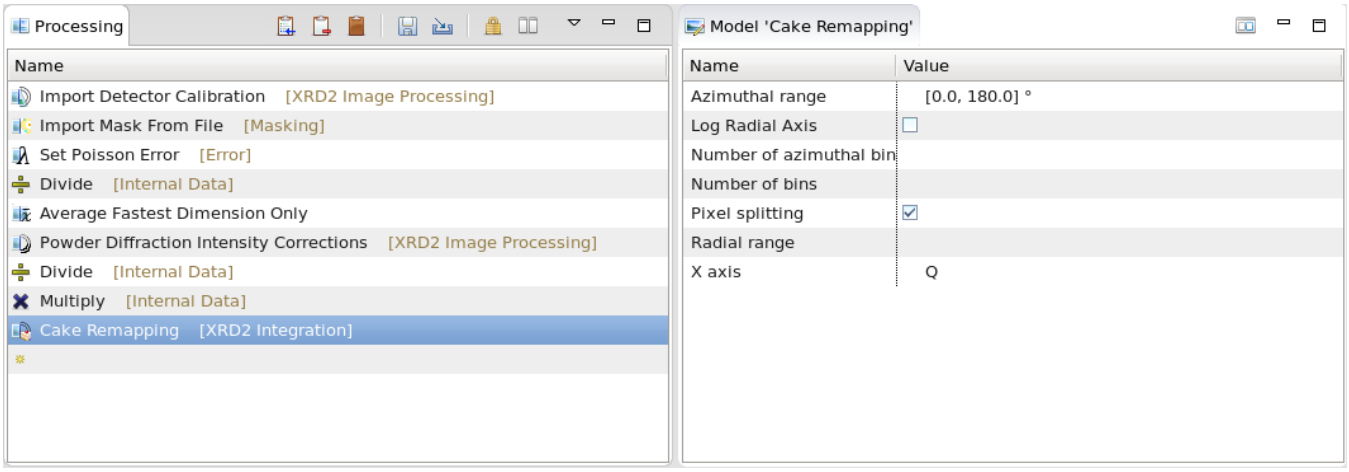
Finally, using the *Subtract Scalar* plug-in it is possible, **although not recommended**, to subtract a constant value, this can be useful for certain *specific* analyses. In the case highlighted above 1000 has been subtracted uniformly from the entire dataset.

Using cake plotting

Cake plotting, or *cake remapping,* in DAWN allows for the plotting of detector images as a function of *azimuthal angle* vs q rather than x and y pixel values. Using this plotting mode rings of intensity become vertical lines, with respect to the y axis, allowing for visually clearly defined subtraction on an image between diffraction features, as a function of q. As this reduction and background subtraction approach is different from before a different processing pipeline is also required. By the end of this section a pipeline, similar to the one highlighted below, will be created and customised to background subtract your data.



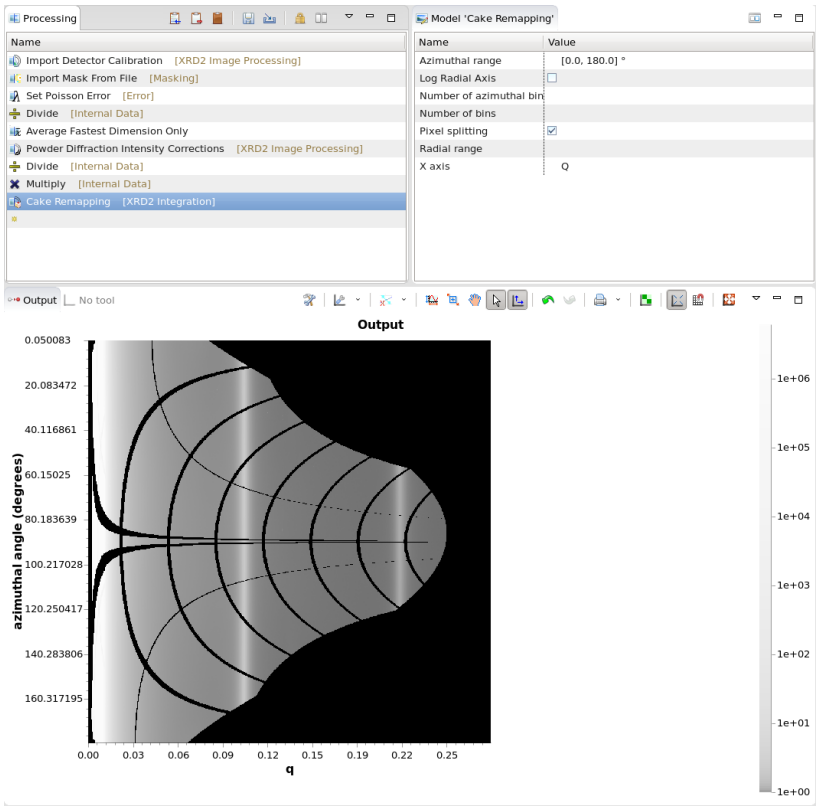
To start, load in an image of interest into the *Data Slice View*, selecting to view either SAXS or WAXS images. After this has been loaded, depending on whether it was elected to view SAXS or WAXS data, it is then necessary to load in a standard reduction pipeline, omitting the final reduction step (which can be removed using the *Delete Selected Operation* button above the processing pipeline), such pipelines are available in the reducting data from I22 using DAWN section. Proceeding this, load in the Cake Remapping [XRD2 Integration] into the pipeline.



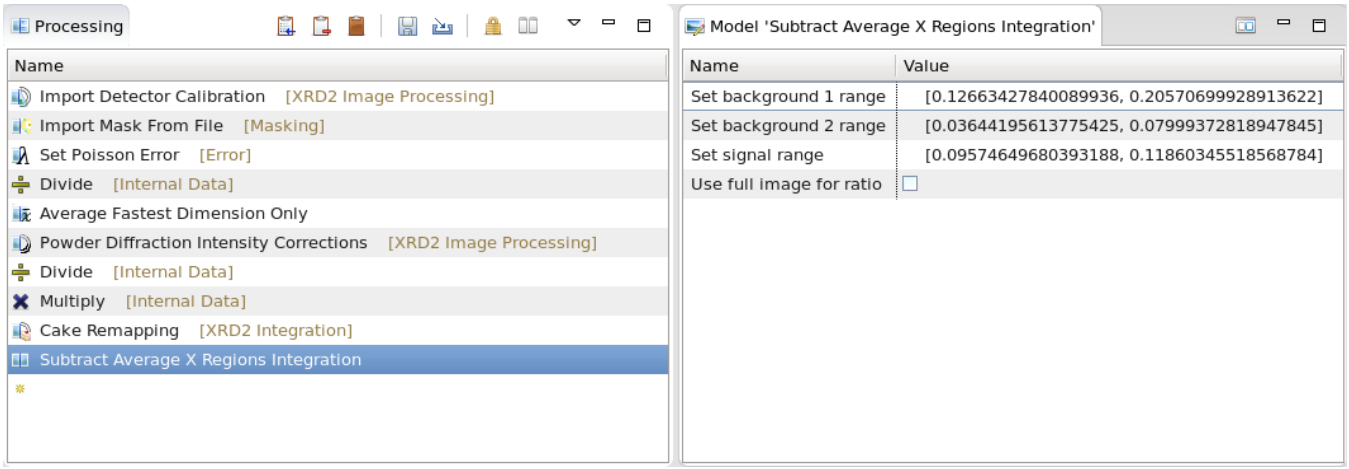
As is highlighted in the image above, there are a number of configurable options for the *Cake Remapping* tool these are:

- **Azimuthal range** - This adjusts the azimuthal region of interest, if the beamstop is an extreme of the image it is advisable to change this to remove zero intensity regions. Zero degrees is horizontally right (3 o'clock) from the beamstop turns clockwise.
- **Log Radial Axis** - A checkbox asking whether the x axis should be on a log scale.
- **Number of azimuthal bins** - The number of bins, or pixels, that the data should be resampled to along the y axis, if blank the native resolution will be used.
- **Number of bins** - The number of bins, or pixels, that the data should be resampled to along the x axis, if blank the native resolution will be used.
- **Pixel splitting** - A checkbox allowing for the intensity of a pixel either to be directly mapped onto the cake plot or split the intensity into the corresponding bins (data smoothing).
- **Radial range** - This adjusts the q region of interest.
- **X axis** - A drop-down box to select whether the x axis should be displayed as *angle*, *pixel*, *resolution* or q.

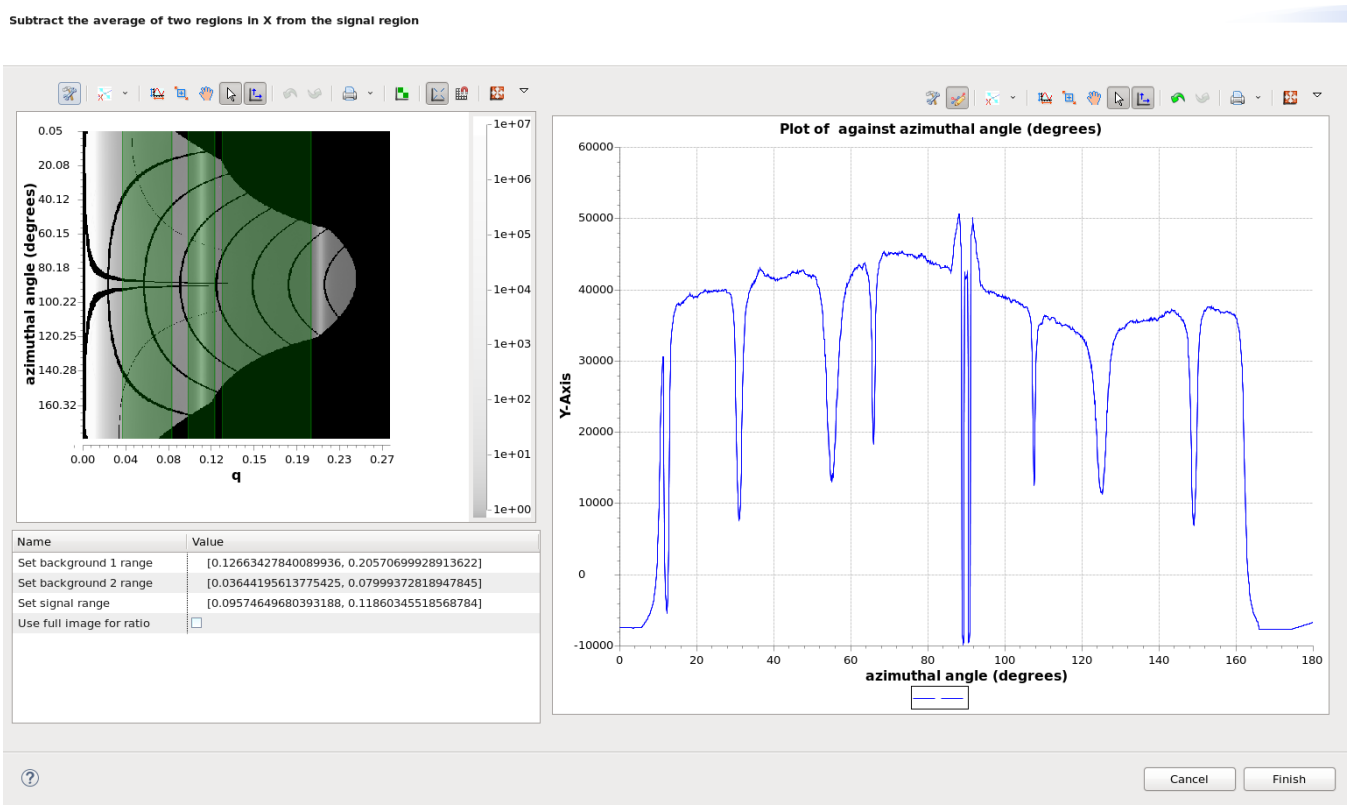
For the example SAXS image *i22-363061*, the above values should produce the following display in the *Input* and *Output* fields. You will have to adjust the intensity range of the plot to get the result below.



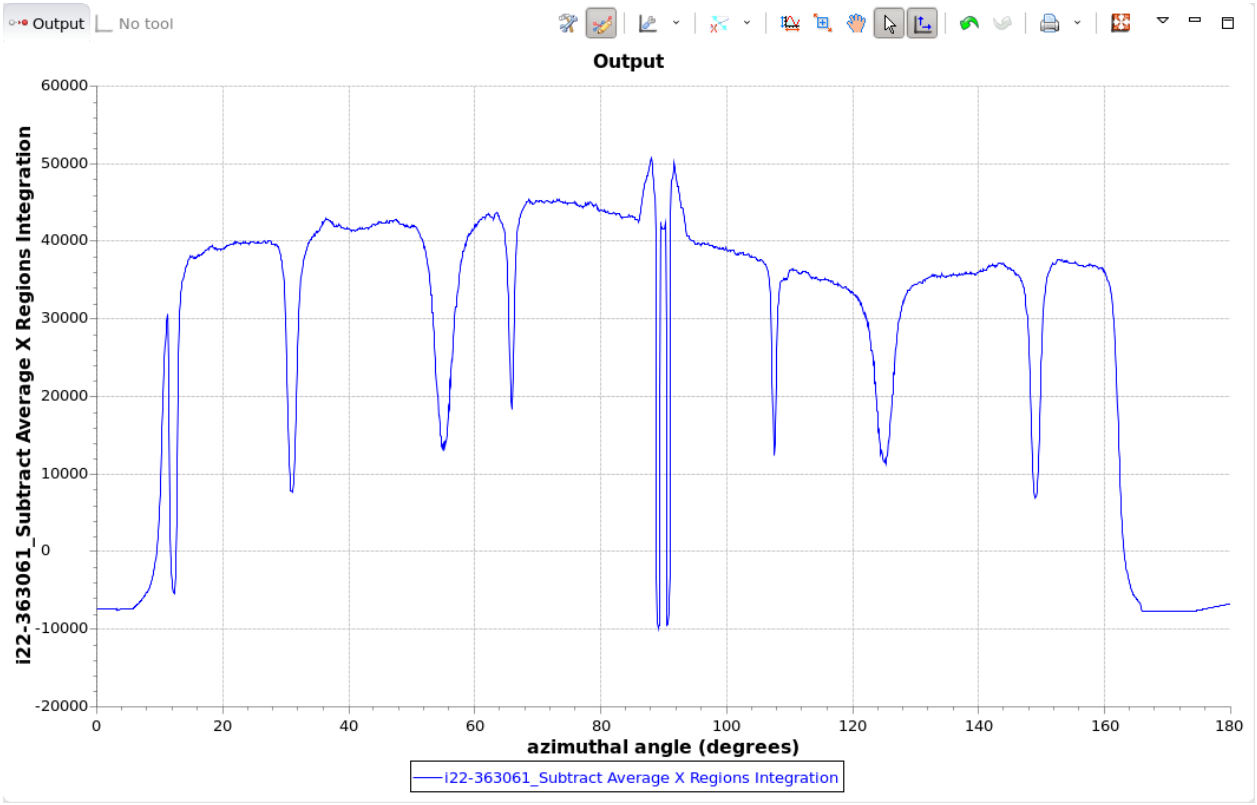
When this has been configured as desired, next load the *Subtract Average X Regions Integration* plug-in into the pipeline. This plug-in will perform the background subtraction on the remapped image by taking the average intensity of two defined background regions, also averaged, and subtracting the corresponding azimuthal background value from the averaged azimuthal value, over the q range available or specified.



Although, as highlighted above, it is possible to directly enter in values for the background subtraction areas and the overall q area of interest. However, clicking the *Live setup* option from the top button bar in the *Model* panel presents a user-friendly and visual interface for the area selection.



Using the ROI tools, The Processing perspective#ROI , it is possible to click and drag the handles on the regions of interest for the first and second background ranges as well as the signal range to integrate over. After selecting the regions of interest to and clicking *Finish* the interface will return to the *P rocessing* perspective with a plot of the result in the *Output* panel, as displayed below.



Depending on what is desired it is now either possible to click the *Process all files* button in the *File Slice View* panel, or add the *Export to Text file* plug-in into the *Processing* pipeline in order to export results as NeXus files as well as tab delimited text. Instructions for using the *Export to Text file* plug-in can be Exporting reduced data or detector images.

Previous: **Reducing data from I22 using DAWN** | **Next:** **Viewing results in DAWN**

Viewing results in DAWN

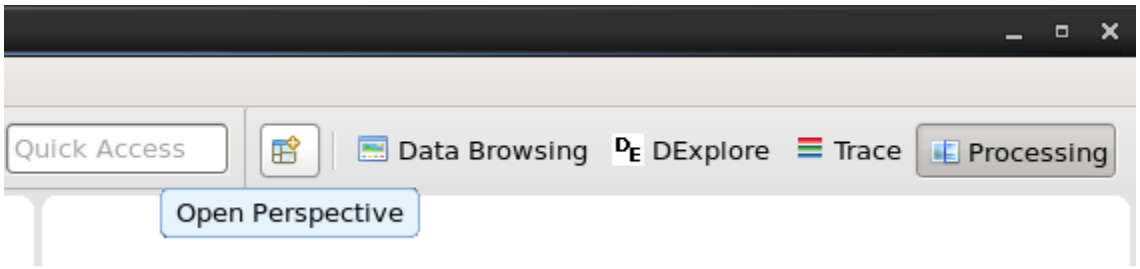
This page provides information on the following:

All instructions provided were performed on a Linux version of DAWN 2.7.x. Although there are no specific alterations to the way the program works between the operating systems, certain parts of DAWN might look different on your computer to the illustrations in this guide. A DAWN workspace accompanying this section of the guide can be downloaded by clicking on this link. Alternatively, just the data files highlighted below can be downloaded as a zip file I22 Beamline Calibrants.zip.

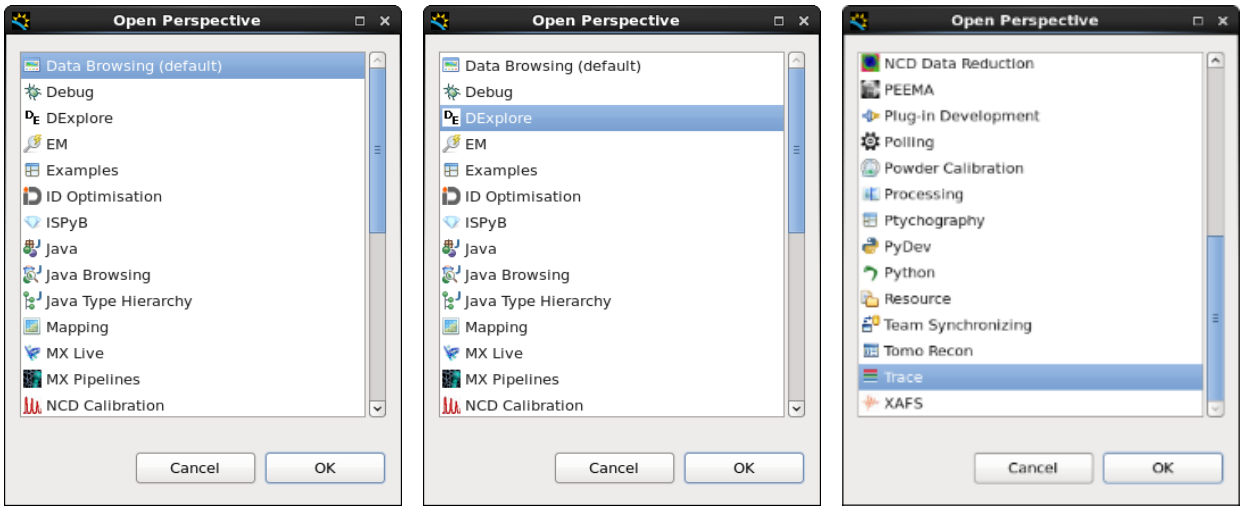
N.B. We are aware that having three ways to visualise data within DAWN is *far* from ideal. To simplify this we are currently working on a new perspective, *DataVis*, which will incorporate all of the functionality presented by the existing perspectives, alongside some new features. However, due to it's current developmental nature it is not necessarily as mature as the existing data visualisation perspectives. For more information on *DataVis* and how to use it please browse the documentation Quick Introduction to DataVis.

Changing perspective

Throughout these examples we shall be moving between perspectives in DAWN, depending on activities to date in the workspace used these may or may not already be activated to switch between. In this section we shall be using the *Data Browsing, DExplore* and *Trace* perspectives. In order to determine if a perspective is currently active look towards the top right hand side of the screen, in the example highlighted below the *Data Browsing, DExplore, Trace* an *Processing* perspectives are currently active. If the desired perspective is not currently active click on the *Open Perspective* button, as highlighted below.



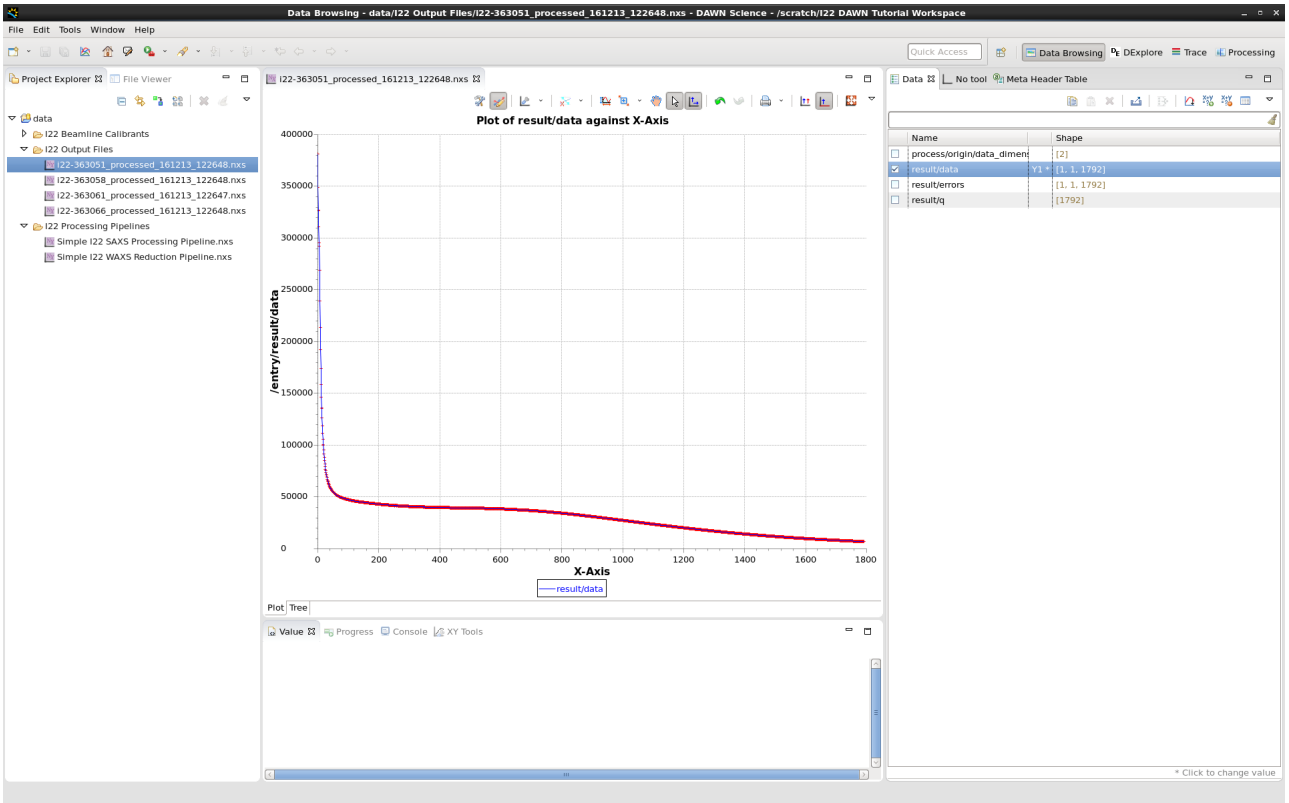
After clicking this button, select the desired perspective from the list provided and click OK, this will add the selected perspective to the list in the top right hand corner of the screen. This process can be repeated as desired, for this section of the guide the *Data Browsing, DExplore* and *Trace* perspectives have been loaded.



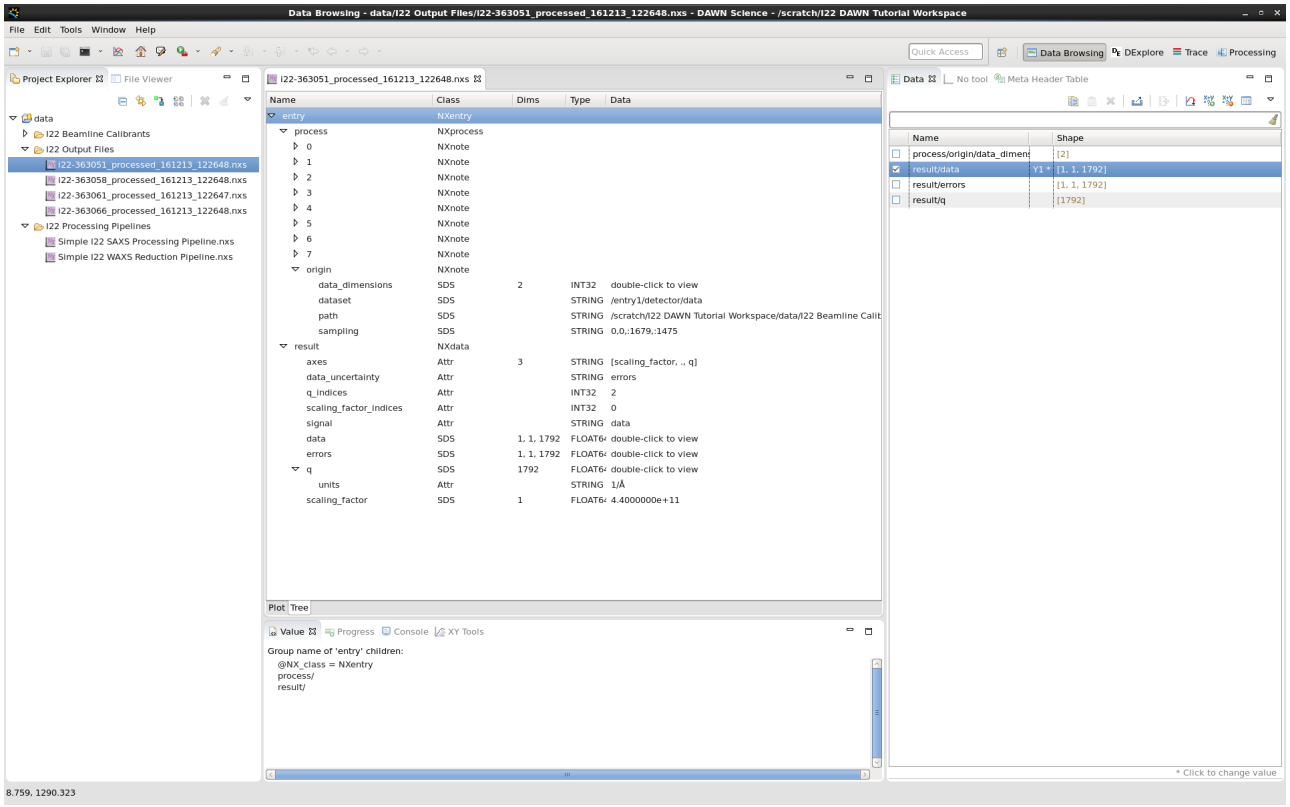
To remove a perspective no longer desired, right-click on the perspective's button at the top right hand corner of the screen and select *Close*. Closed perspectives can be re-opened later, however, any work/progress made in the perspective will be lost.

The *Data Browsing* perspective

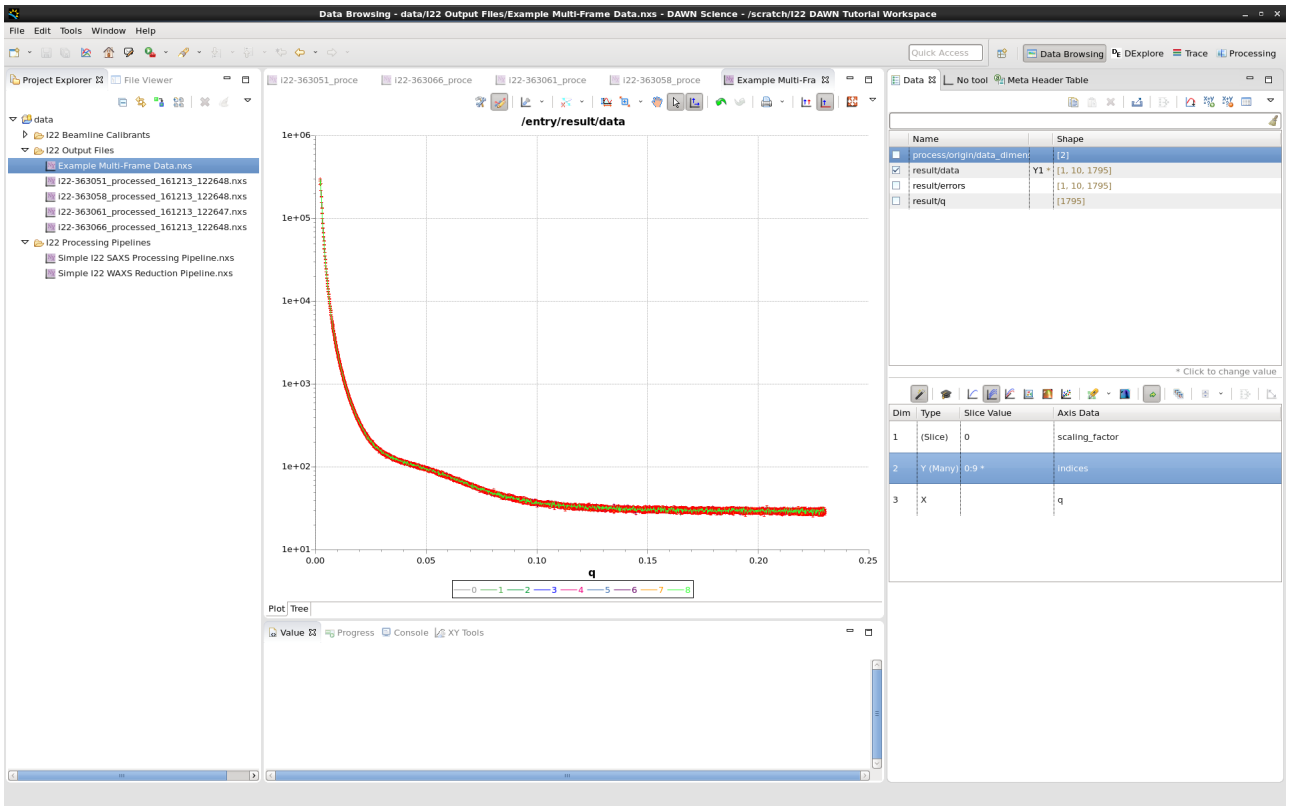
The simplest perspective for viewing data is the *Data Browsing* perspective. This perspective is useful for viewing plots from one file, that is to say, only one file can be open at any one time. If a NeXus file has several sets of reduced data in it, these can be plotted, however it is not possible with this perspective to view scans between files. The *Trace* perspective is best for viewing multiple results from multiple files.



In order to view a plot from a file containing a single reduction, double click on the file in the *Project Explorer* panel and from the *Data* panel on the far right check the tick-boxes for *result/q* and *result/data*. In the third column of that table you can select, from drop-down menus, on which axis to display which dataset. As results are typically I vs q, the recommended default would be to display *result/q* on the x axis and *result/data* on the y axis, as highlighted above. Naturally, selecting other datasets that might be available and selecting the relevant axes will display this data as well.



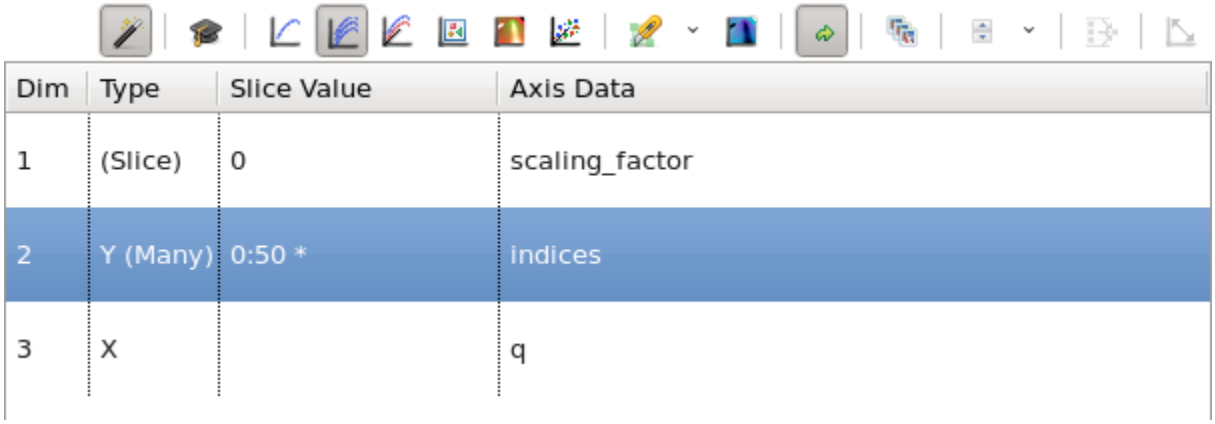
If however a single, or auxiliary, value is required, clicking the tab entitled *Tree* under the main plot will bring up the NeXus file tree. From here all data contained within the file is explorable and inspectable. If greater detail, or finer granularity, is required when viewing this data, or datasets, then it is recommended to use the *DExplore* perspective.



Commonly, multi-frame files will be placed into the data browser, here a multi-frame example has been bought in from a later tutorial page. These files are handled in much the same way as single-frame files, however, more plotting features become available. As highlighted in the above image, the right hand panel splits to reveal more information about the dataset being plotted. From within this perspective it is possible to visualise multi-frame data as:

- Singular line plots (Slice as line plots)
- Overlaid line plots (Slice as a stack of line plots)
- 3D overlaid line plots (Slice as a stack of line plots in 3D)
- As a 2D image (Slice as image)

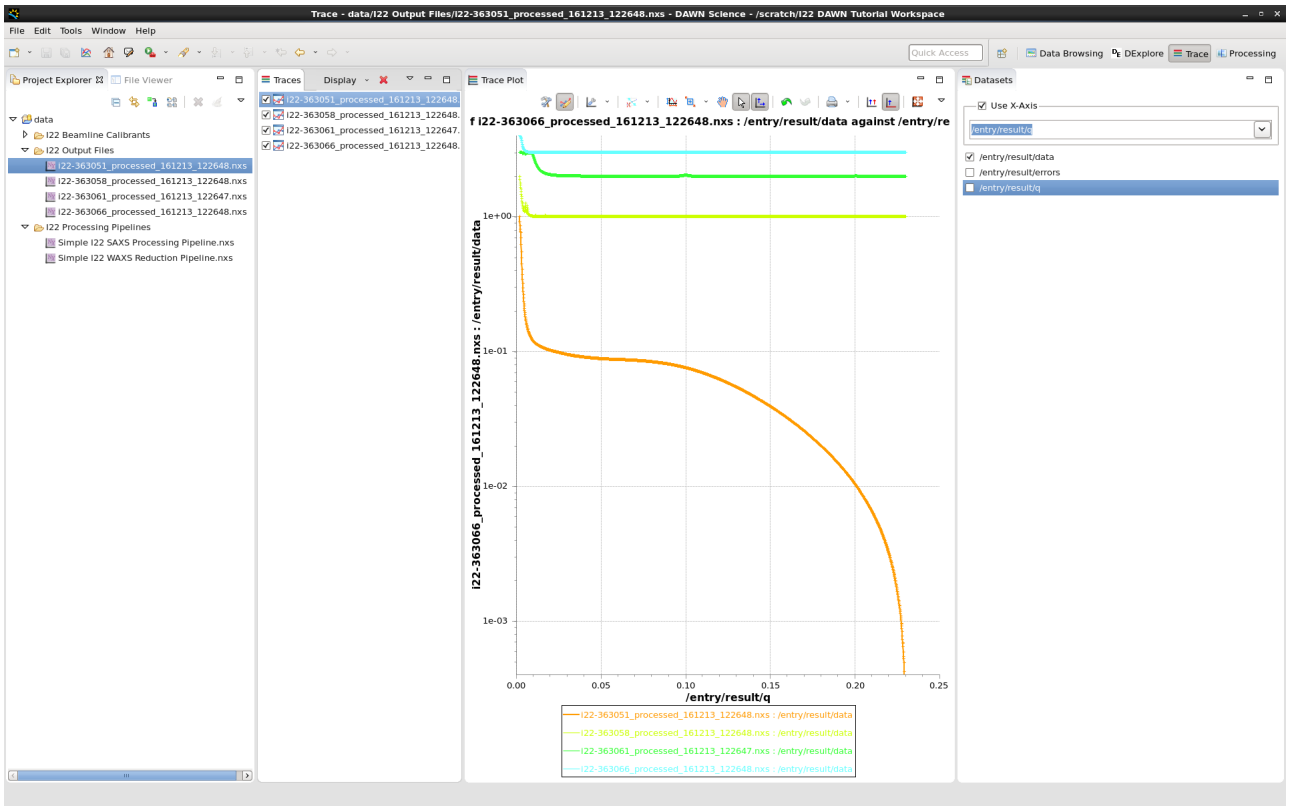
In addition to these most commonly used views, additional plotting modes are available for the user to explore. Switching between these modes is possible by clicking on the buttons below the dataset names in the right hand panel, as highlighted at the top of the image below. Hovering over these buttons will reveal a tooltip, the names of which correspond to the names in brackets above.



As mentioned previously, the *Trace* perspective offers many more features beyond this perspective for visualising results.

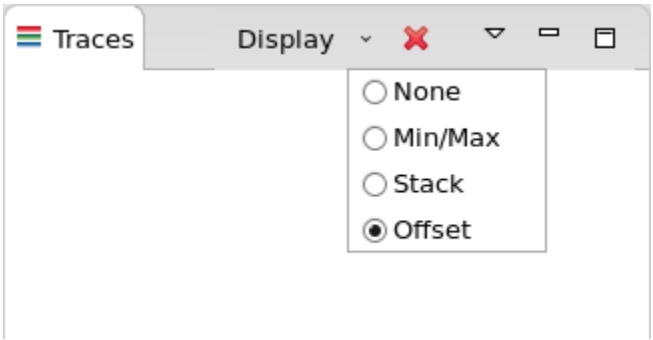
The *Trace* perspective

The *Trace* perspective offers a way to view multiple plots from multiple files either overlaying, offsetting or stacking plots on top of each other to compare and contrast datasets. Double clicking on a file, or dragging-and-dropping it from the *Project Explorer* panel into the adjacent *Traces* panel will load a dataset into *Trace*. After loading in a file it is possible to view the different one dimensional dataset by clicking the checkboxes next to the different datasets within the file. However, selecting only one dataset will display the data without a calibrated x axis, to display I vs q data as would be expected, first the checkbox entitled *Use X-axis* must be ticked, with */entry/result/q* picked to load the q axis scale, and then */entry/result/data* checked from the remaining options underneath.



As highlighted above, repeating this process multiple times, or by loading in several files, highlighting them and subsequently selecting which datasets to view, will plot multiple datasets. At the top of the *Traces* panel is a small drop-down list under the title *Display*, from here the following options are available:

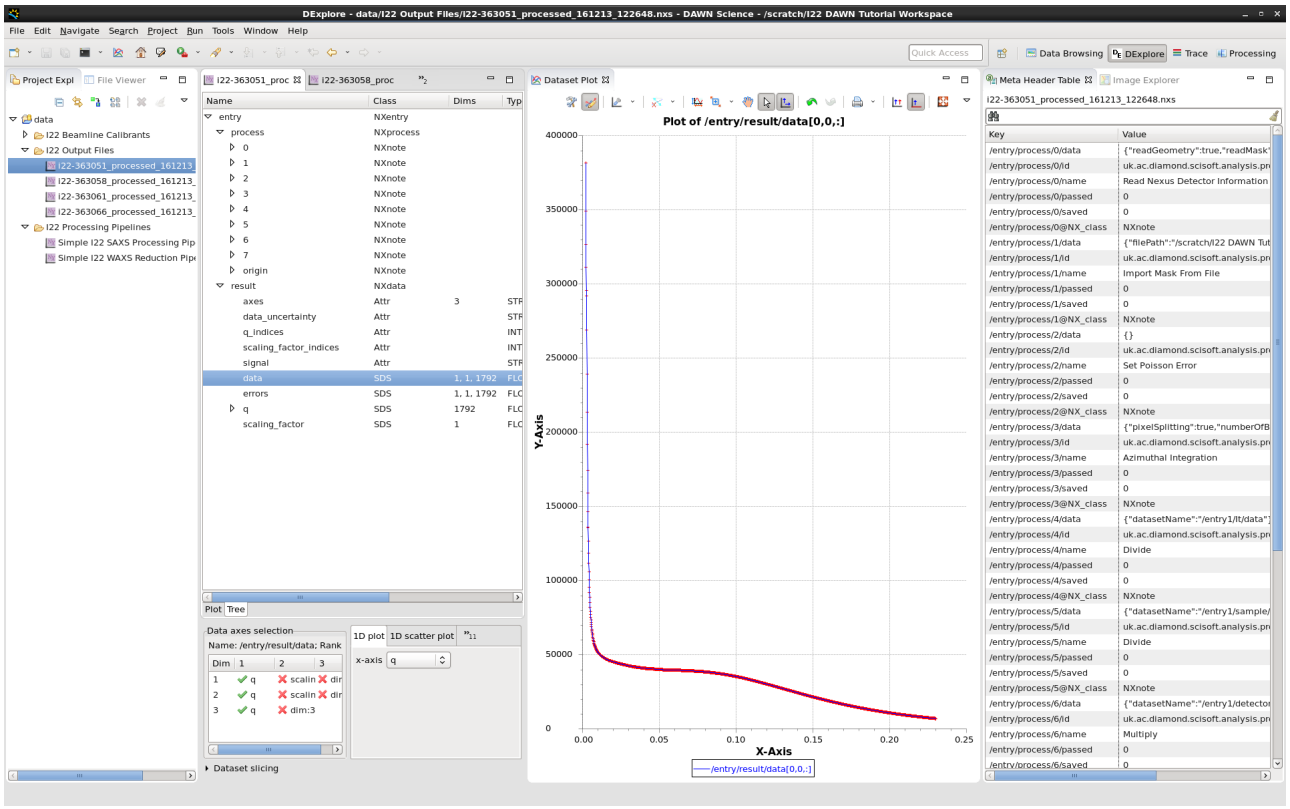
- **None** - Overlay the individual line plots without alteration
- **Min/Max** - All plots are normalised between zero and one and overlaid on top of each other
- **Stack** - Each line plot is stacked on top of each other
- **Offset** - Working down the *Traces* list each plot is offset from previous zero value by 1



This allows for easy side-by-side comparison of data gathered in separate files. However, if a NeXus file is loaded that contains multiple datasets then the sum of the datasets contained within that file will be averaged and the overall result loaded. *Trace* has been designed in this manner as it has been designed for analysis of results gathered that might contain a high signal to noise ratio and therefore multiple acquisition passes will improve this ratio. More information about *Trace* can be Session E. DAWN Training - Plotting Single Line Plots from Multiple Files [Trace Perspective].

The *DExplore* perspective

DExplore offers the greatest control over the data viewable, however, it also does so by providing the most complex perspective towards viewing the contents of a given NeXus file. As can be seen in the below overview of the perspective on the left of the *Project Explorer* the NeXus file tree can be navigated through providing an overview of the contents of the file. Double clicking on any dataset that posses more than a single value will then load this in the main plotting window. Underneath the tree view is a panel containing greater information about which datasets are being plotted on which axes, and by selecting the plot type (1D plot is highlighted below) the plot type can be changed to any of DAWN's potential plotting systems (**CAUTION**: even if the data will not display DAWN will attempt to display the data, this may temporarily cause the plotting to go awry, closing all files are reloading them will be required). Underneath this panel is the *Dataset slicing* panel which provides a number of sliders to navigate through the different frames present in your data (if any).



On the right hand side of the screen is displayed more information (the metadata) about all the values held in the NeXus file, typically will not be of use, unless you are deliberately looking for them at which point it is anticipated you will know what you are doing.

Previous: Background subtraction | **Next:** Custom detector masks

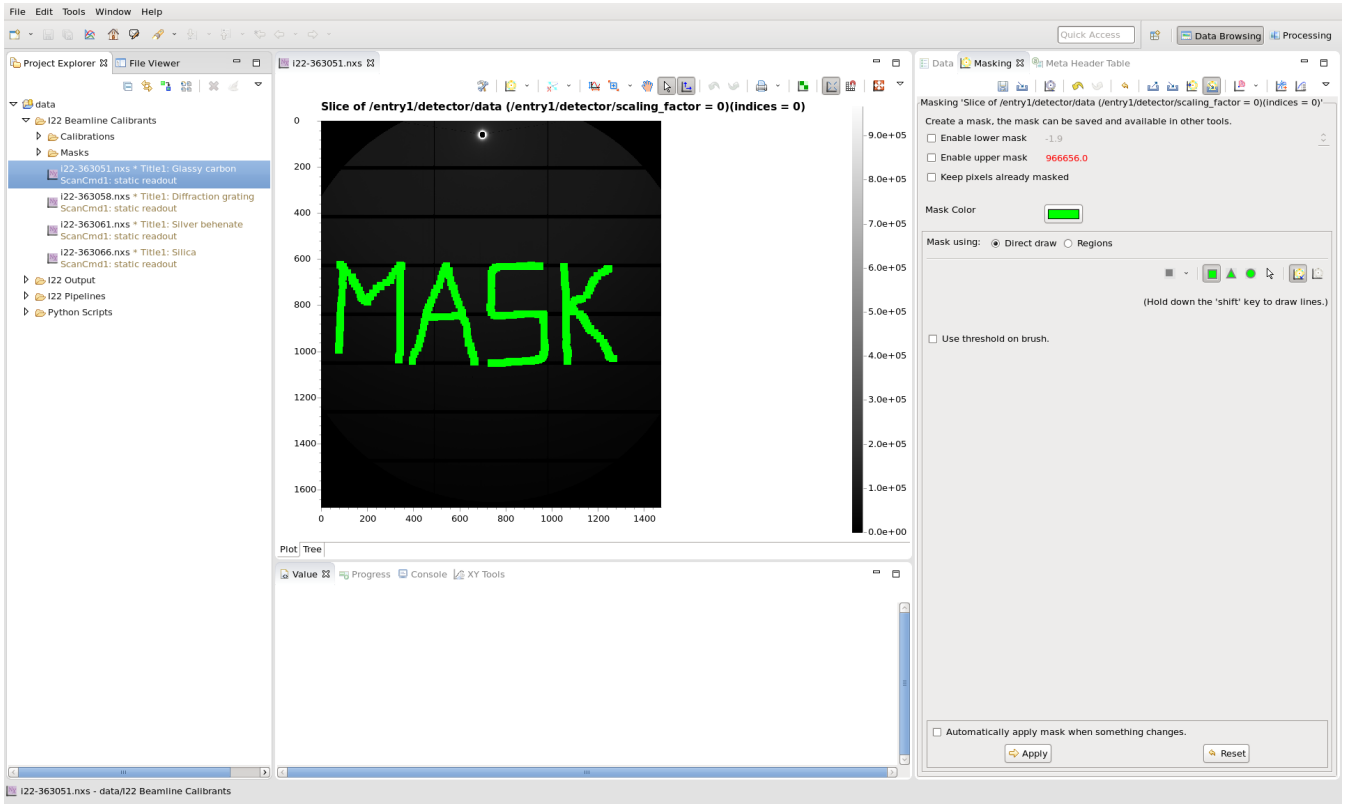
Custom detector masks

This page provides information on the following:

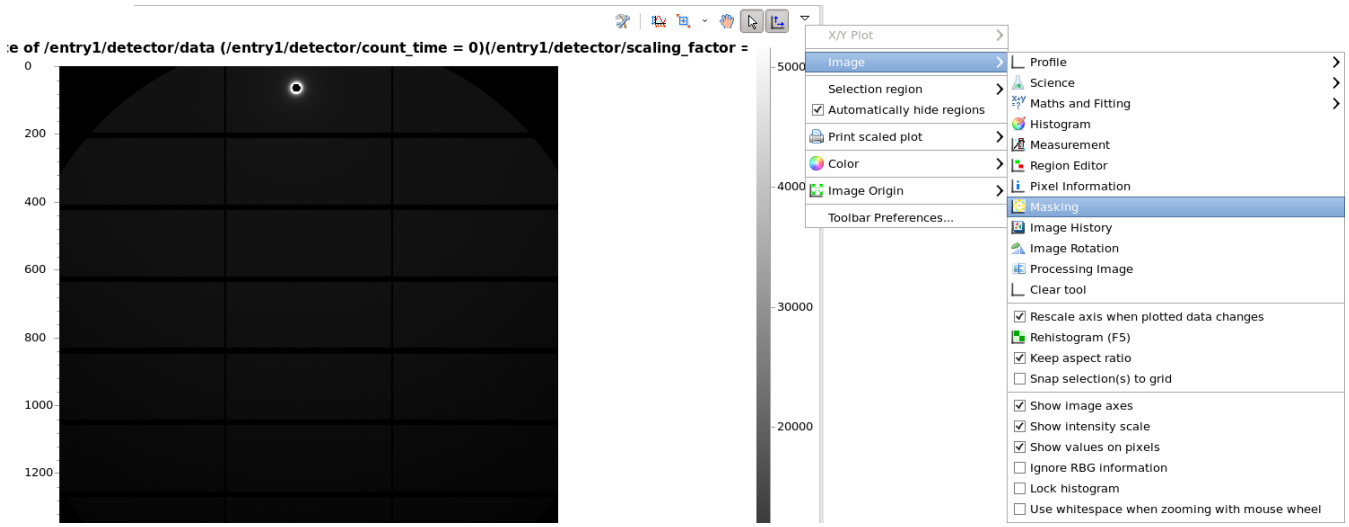
All instructions provided were performed on a Linux version of DAWN 2.7.x. Although there are no specific alterations to the way the program works between the operating systems, certain parts of DAWN might look different on your computer to the illustrations in this guide. A DAWN workspace accompanying this section of the guide can be downloaded by clicking on this link. Alternatively, just the data files highlighted below can be downloaded as a zip file I22 Beamline Calibrants.zip.

Creating a custom mask

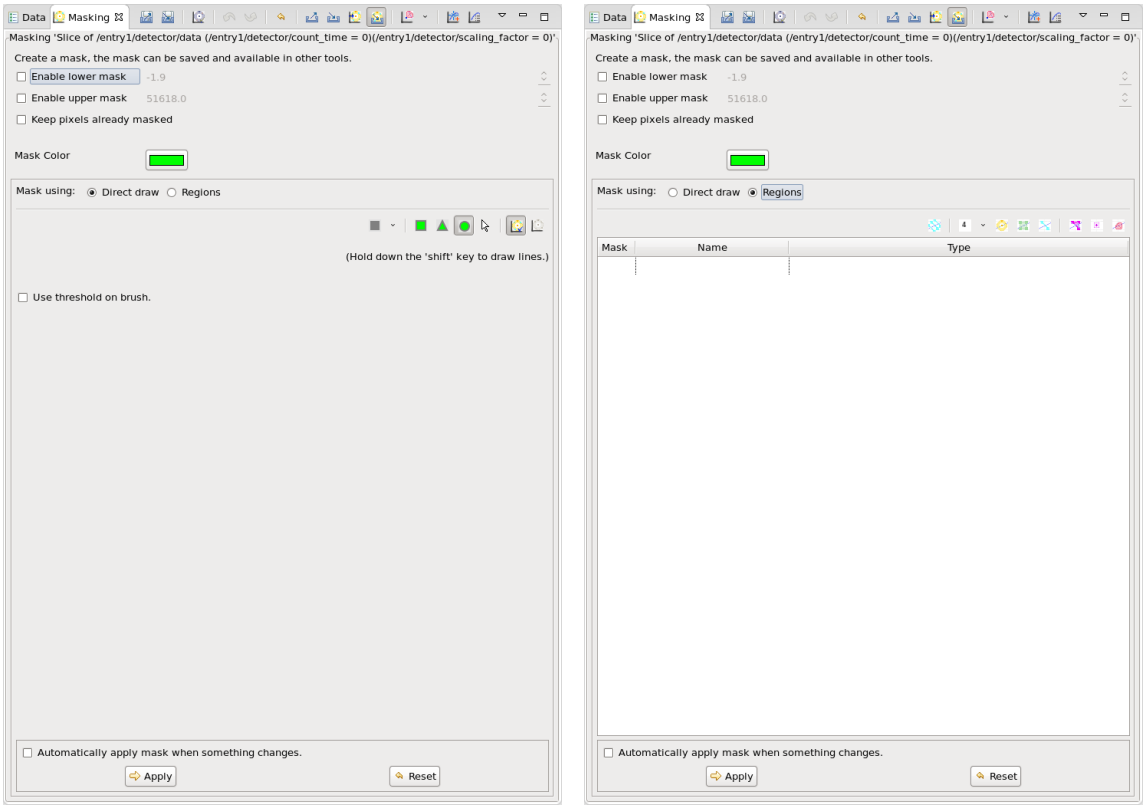
It is recommended that custom image masks are created in the *Data Browsing* perspective, although technically they can be made in almost any perspective the *Data Browsing* perspective affords more screen space to this task than most others. If not already read, more information about loading and switching to the *Data Browsing* perspective can be found under the *changing perspectives* section of the previous page .



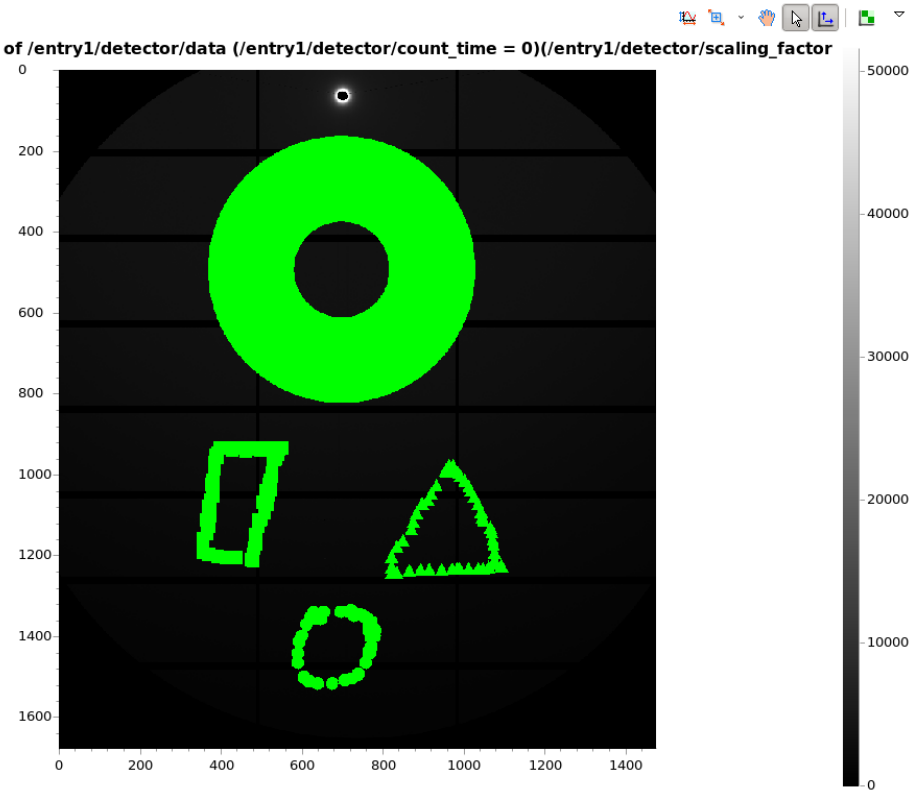
When in the *Data Browsing* perspective double click on the file from which you wish to make the mask and from the options on the right hand panel of the screen click on the checkbox for either *detector/data* to create a SAXS mask, as highlighted above, or *Pilatus2M_WAXS/data* to create a WAXS mask. Depending on what kind of mask is desired it is worth noting that it is possible to create a mask that will be *added* to the existing detector mask, which takes care of the detector grid and any dead pixels, or it is possible to create a *replacement* mask, which will only mask the features created in this mask. It is **recommended** that, without good reason, an additive mask is created principally to catch dead detector pixels.



After loading the desired image to create the mask from, click on the small downwards facing triangle in the image panel, selecting *Image > Masking* from the drop-down menu that comes up, as highlighted above. This will bring up the *Masking* panel in the right hand panel. To start with this should bring up the panel highlighted below on the left. The options marked *Enable lower/upper mask* allow a quick way of applying thresholding to the image and will accept integers when enabled. The *Mask Color* option allows for the display of the masked pixels to be changed in colour, this is only for display. Approximately a third of the way down the panel on the right hand side are three buttons, a square, a triangle and a circle, selecting one of these will allow you to draw directly onto the detector image using the corresponding shape, furthermore the 'greyed out' square furthest on the left allows for tuning the size of the draw tool. To the right of the tools are buttons to turn the pencil into an eraser and *vice versa*. Hovering over these tools will display their corresponding tooltips.



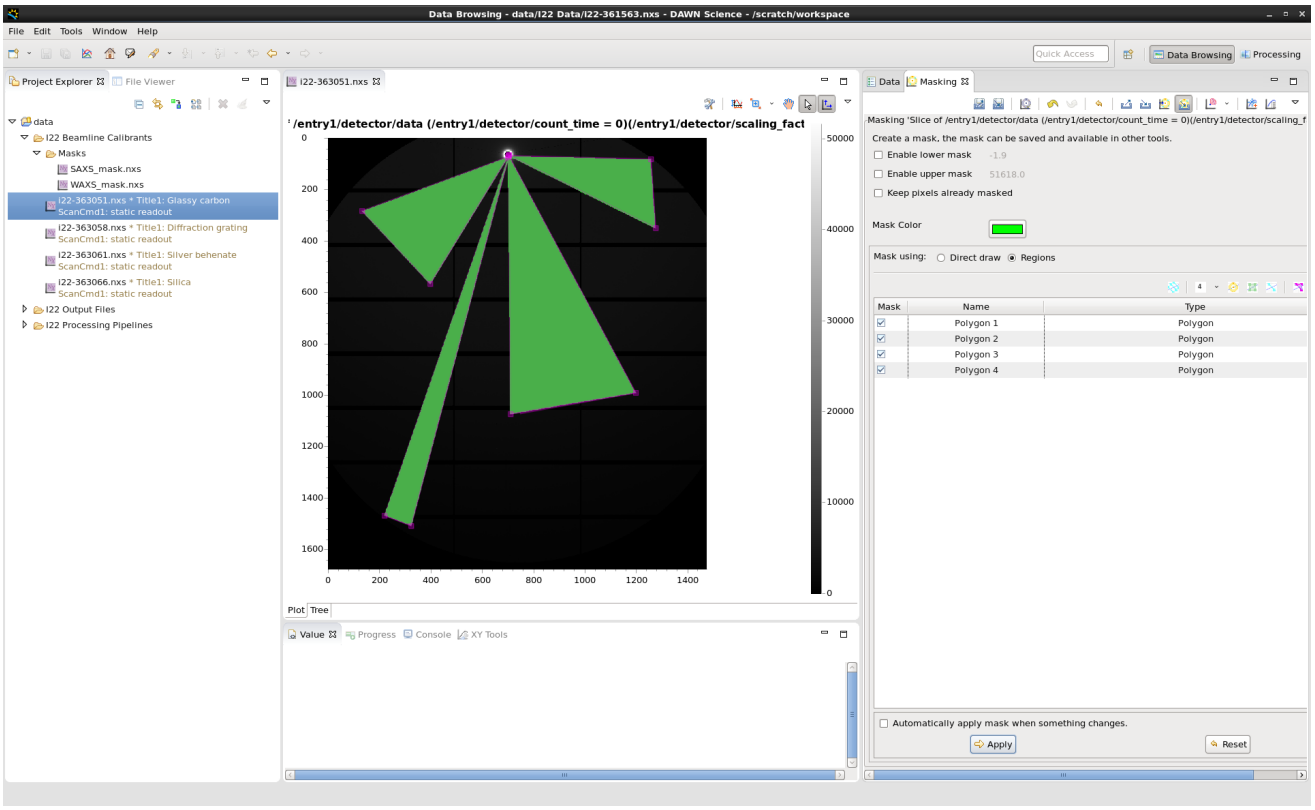
However, the keener observer will also see the radio buttons approximately a third of the way down the panel on the left hand side of the panel, here it is possible to switch between the panels shown on the left and right above. Using the region of interest tools, as covered in the The Processing perspective#ROI, it is also possible mask sections on to the detector image. After the section has been drawn, click the *Apply* button on the bottom left of the panel to draw the region on to the image.



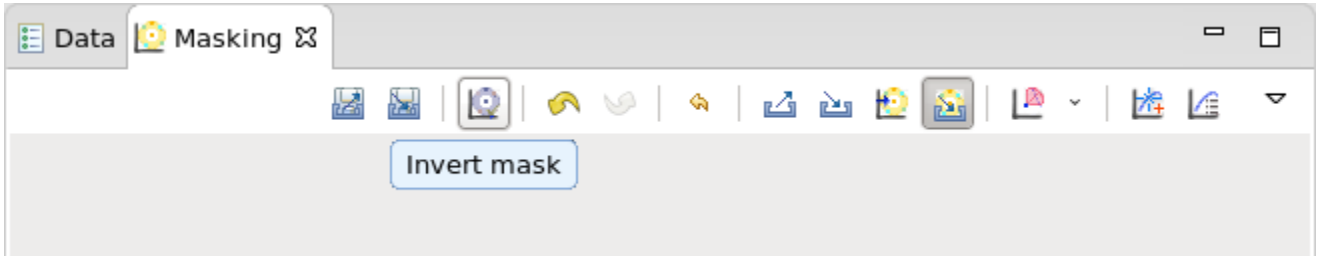
Naturally, as can be seen from the above image, using the hand drawing tools can be combined with region of interest tools to generate a highly customised mask.

Masks as multiple sector integrators

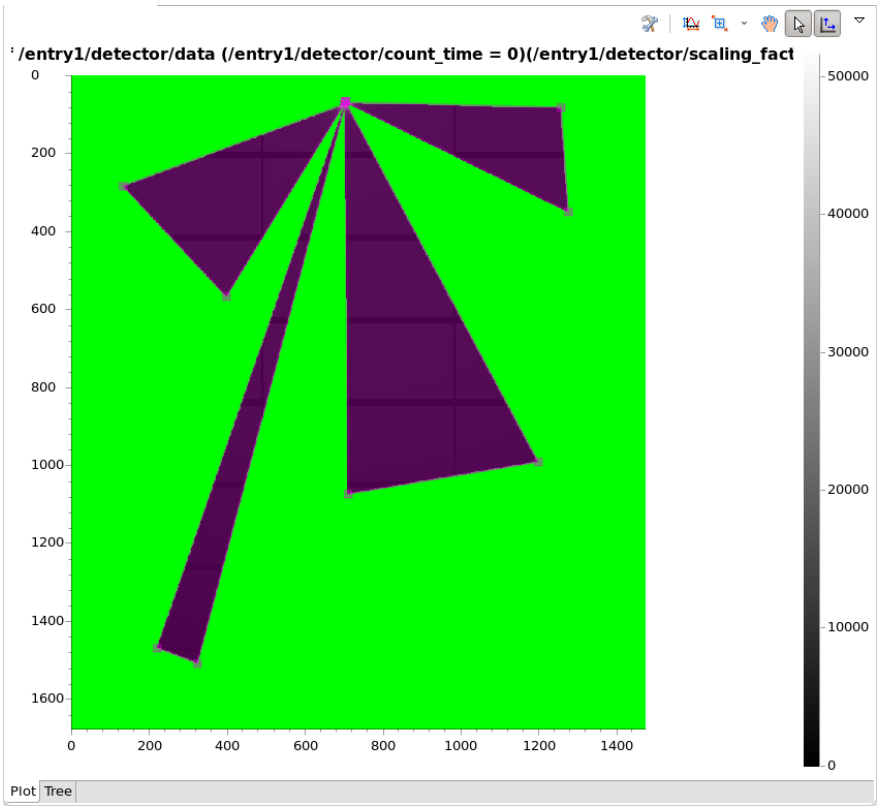
As one can imagine, by using the drawing tools for masking it is possible to utilise masking to create, in essence, multiple sector integrators. To start, bring up the masking tool on the detector image of interest and draw masks in the areas that are desired for integration, as highlighted below.



Then click on the *Invert mask* button from the *Masking* toolbar, as highlighted below.



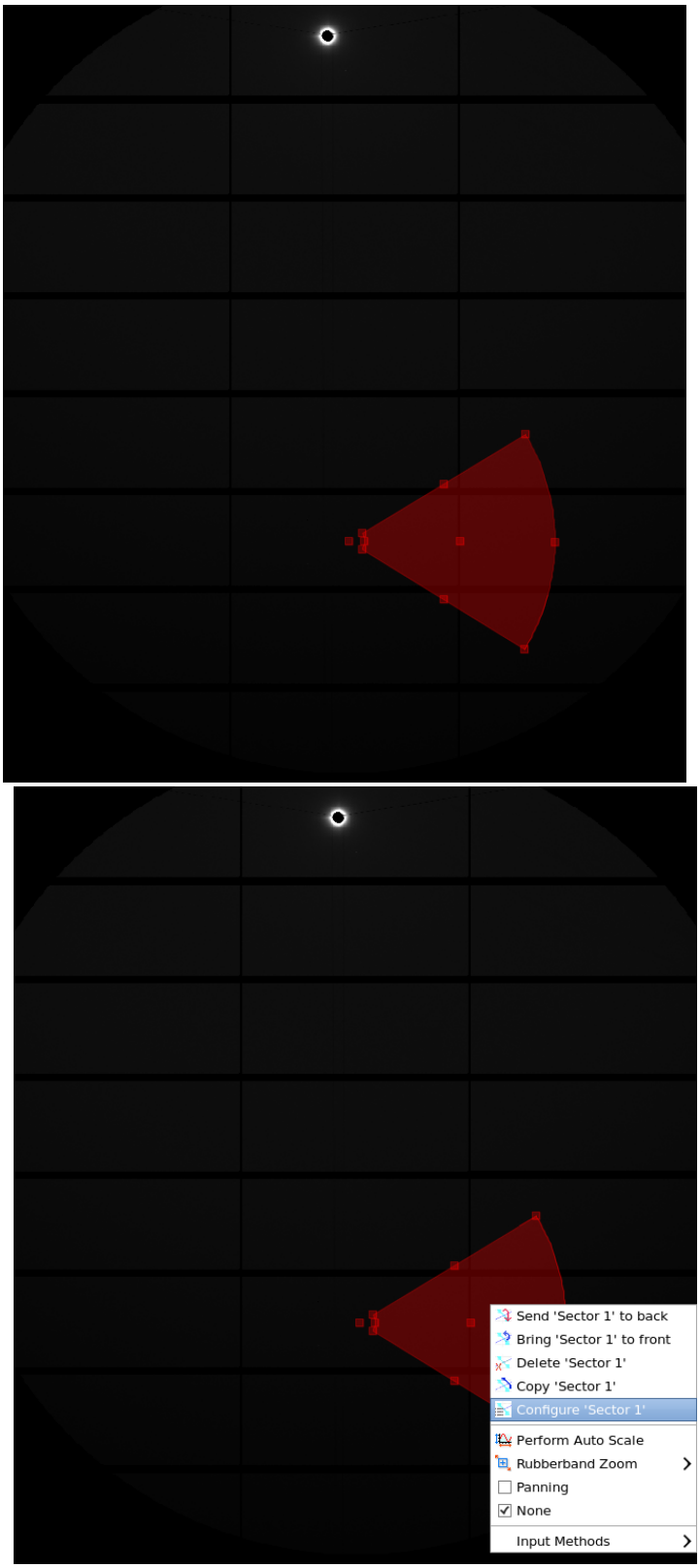
This will turn the previously masked sections 'transparent' or unmasked, whilst masking the rest of the image, as highlighted below.



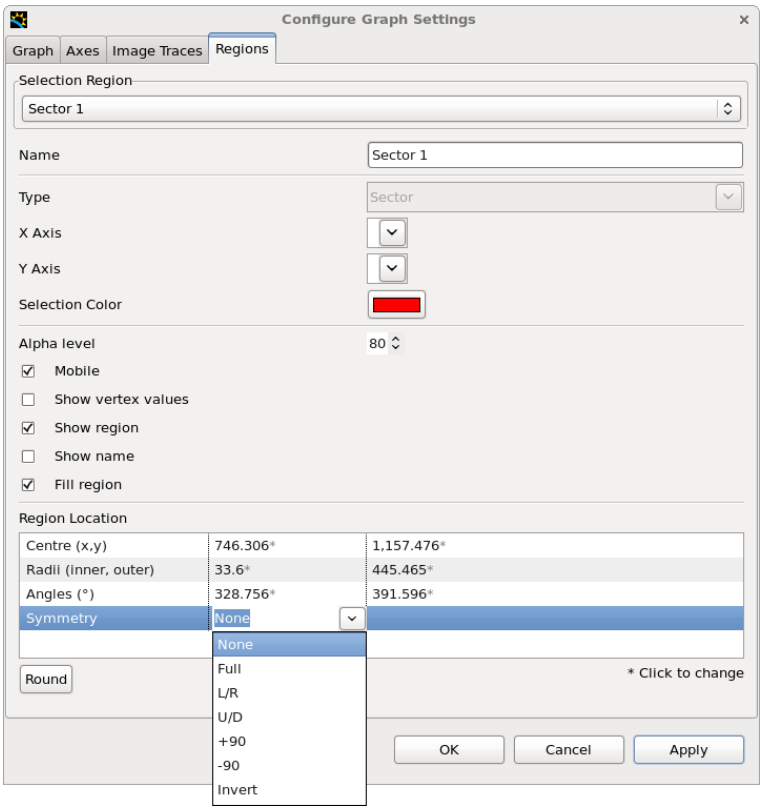
Should it be further desired, additional custom masks can then be created from the shape or free drawn tools and further added within the unmasked sections as well.

Masking regions with symmetry

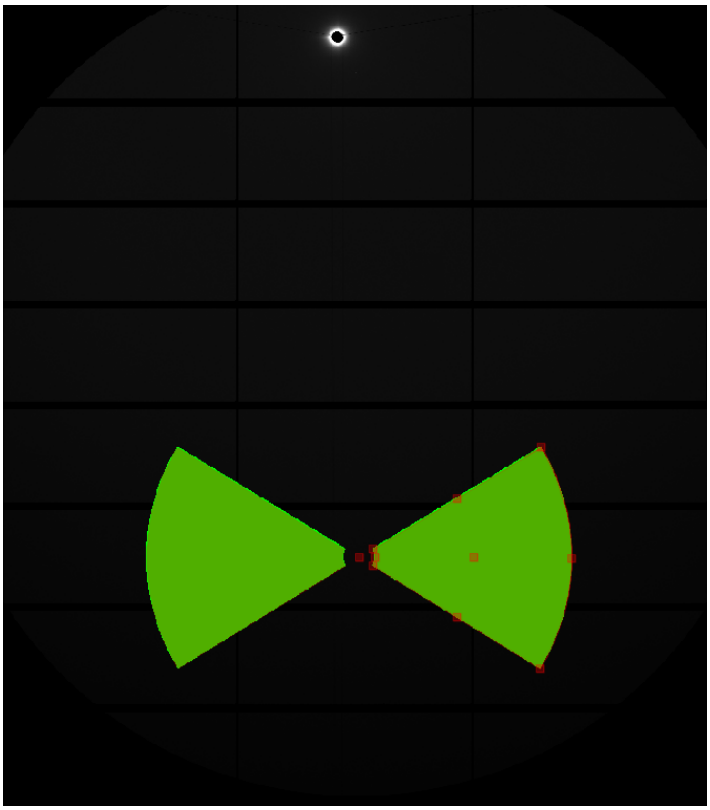
Certain masking or integration operations might benefit or require symmetrical masks to be required, a typical example being a 'butterfly' or symmetrical sector mask. Creation of such masks is possible by first drawing the sector of interest on to the image, as highlighted below, and then right clicking on the sector drawn, from the contextual menu that appears the option *Configure 'Sector x'* should be selected (x will take the region's number in the example below it is the only sector and so it is Sector 1).

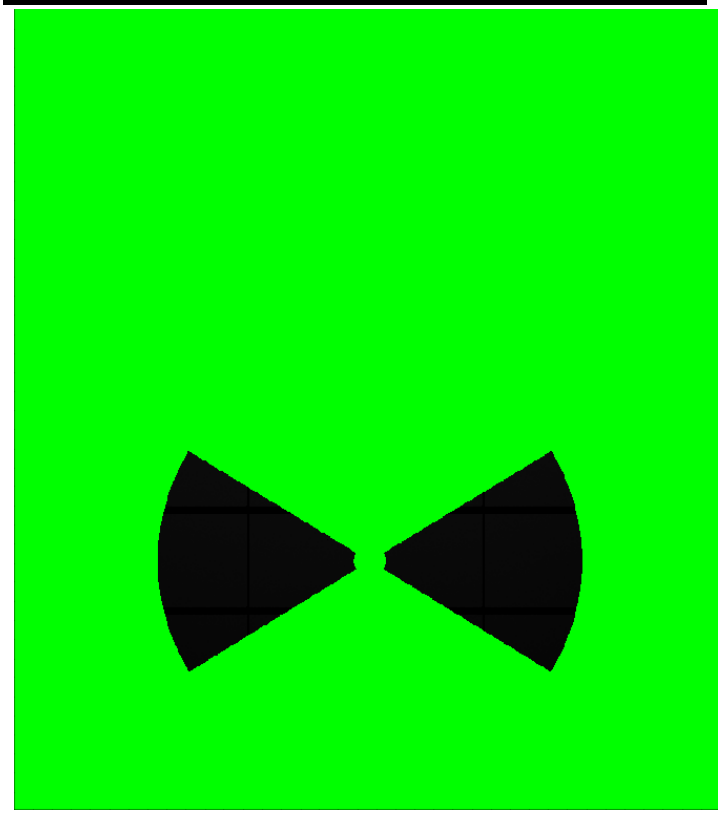


From the window that appears when the configure option is selected navigate to the *Regions* tab, if not already selected, and in the *Region Location* table click on the downward arrow in the column next to the *Symmetry* entry. This will display the list of available symmetry operators that you can select for this region, for this example L/R (Left/Right) will be selected.



After selecting the desired symmetry operation clicking OK will return the image being masked. Should it be desired to mask out a symmetrical feature, such as beam flare, then at this stage saving the mask for subsequent loading would be sufficient, as highlighted below. However, if it is desired to only evaluate the sector that is currently masked, clicking the *Invert Mask* button in the Masking panel will mask the rest of the image only allowing for reduction of the region initially masked, as shown below.

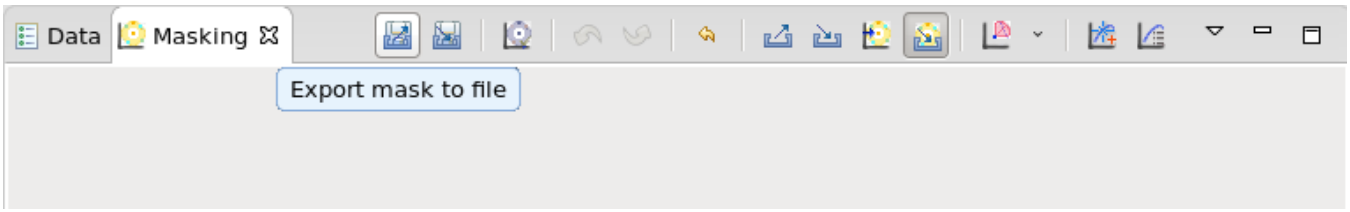




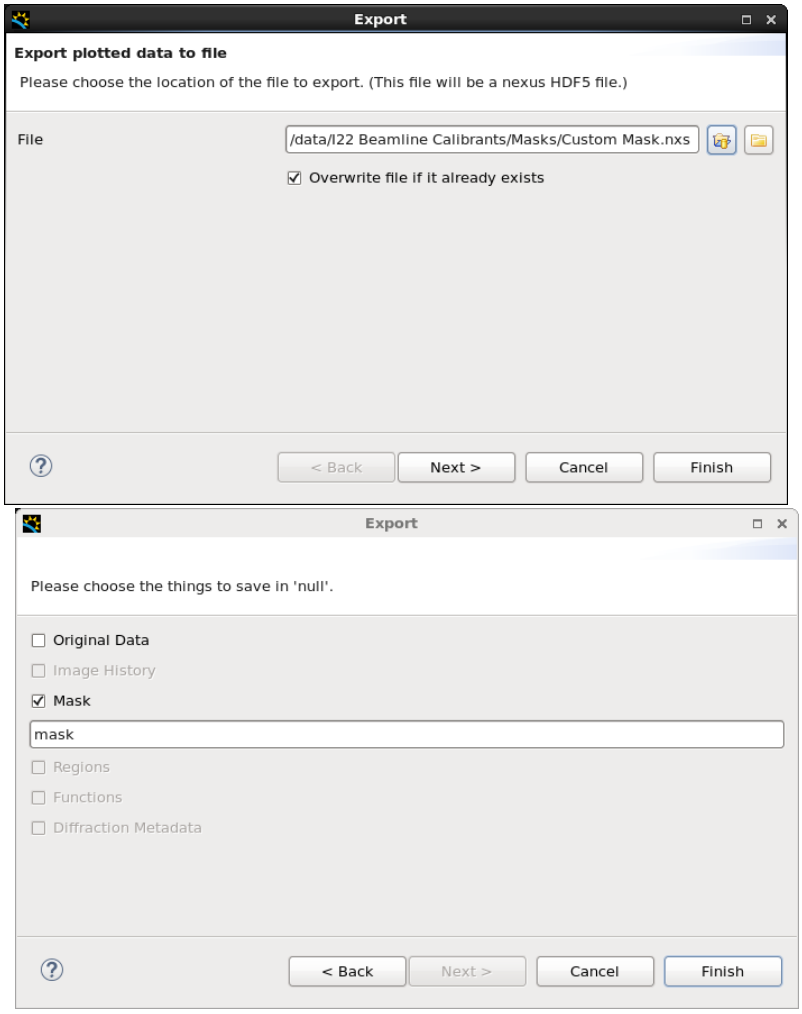
After a satisfactory mask has been please proceed to discover how to save and load these mask files.

Saving a mask

After drawing this it is necessary to click the *Export mask to file* button, as highlighted below, from the right hand side *Masking* panel.



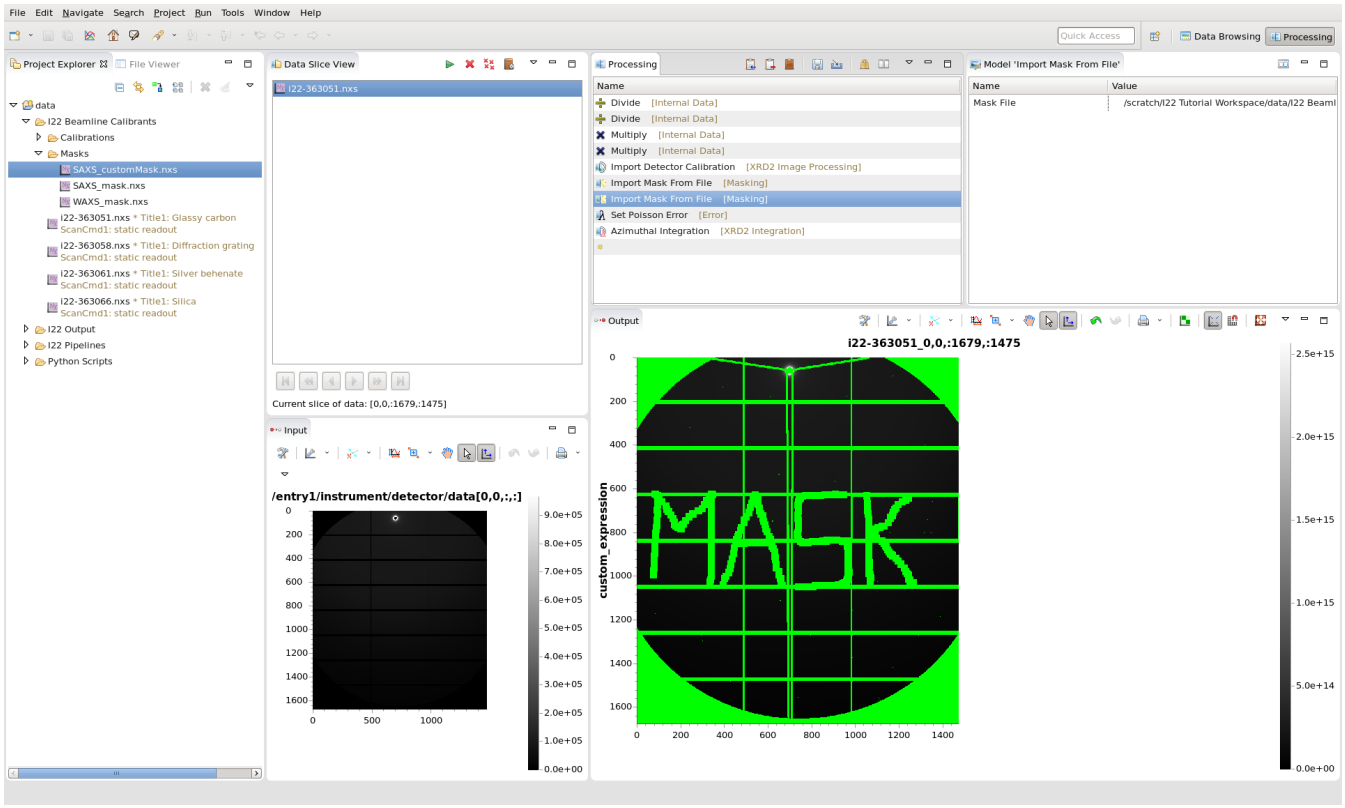
This will bring up the following save dialog, which will need to be filled out in the usual way, first by selecting a save path and name for the mask, then (after clicking next to progress from the first to the second window) by giving the mask a name.



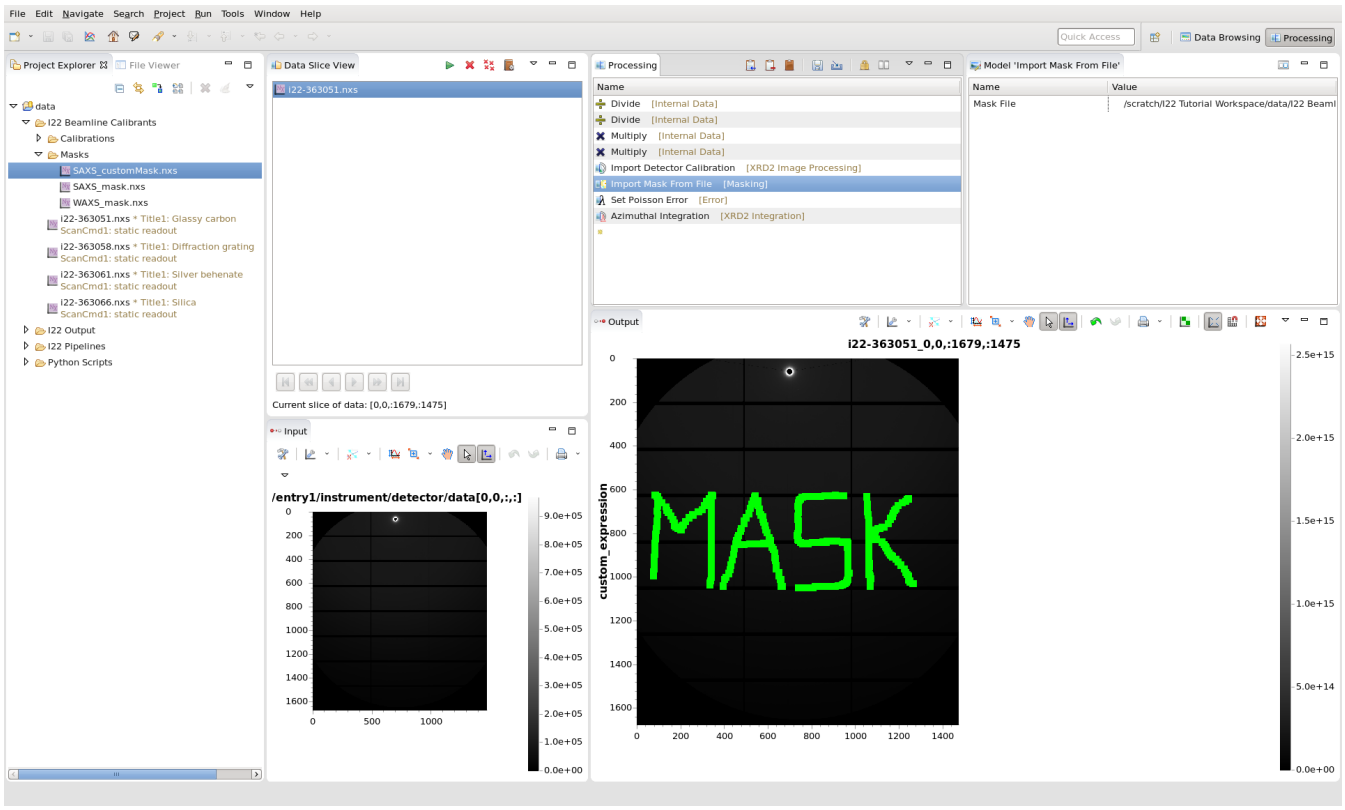
Clicking *Finish* on the second window will save the mask into the desired location.

Loading a mask into a processing pipeline

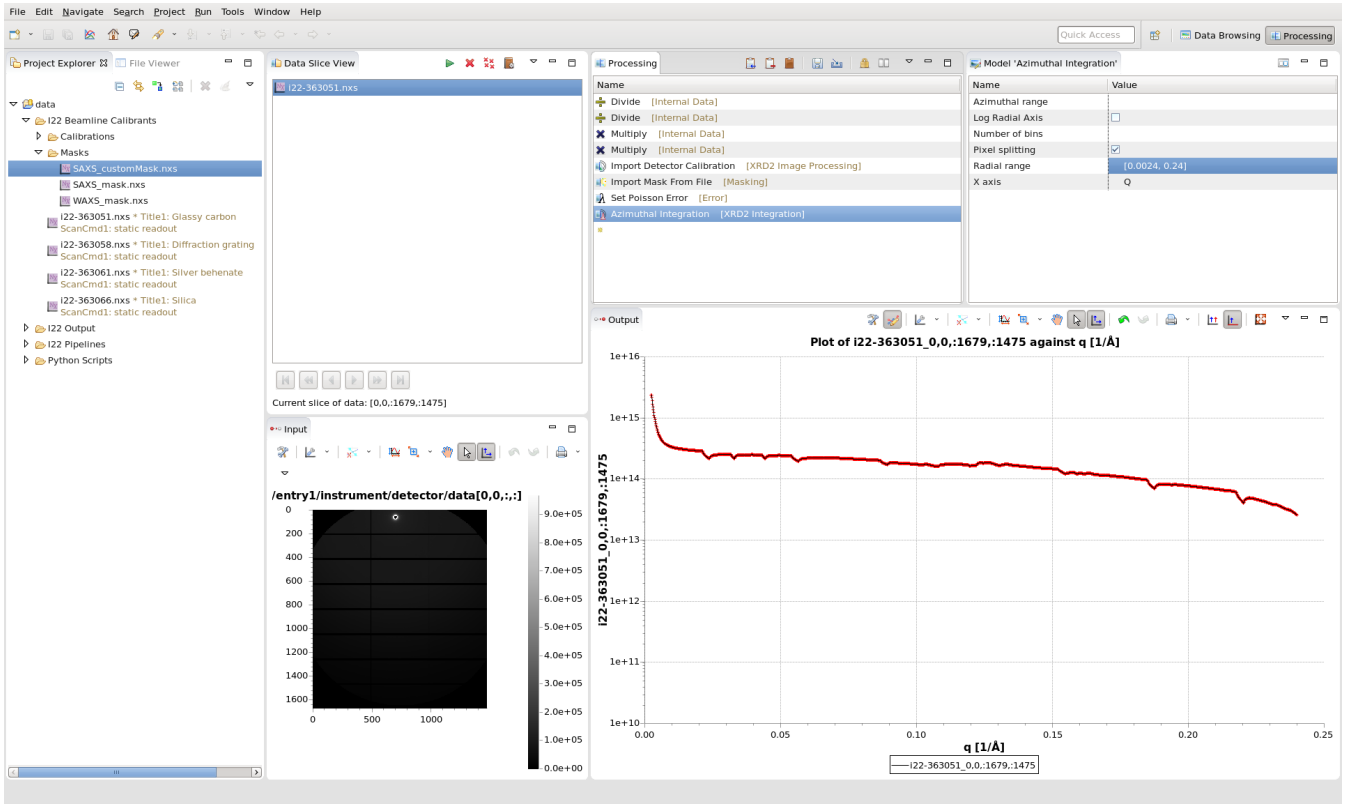
With the mask saved, the final step is to load it into the processing pipeline after the *Import Mask From File [Masking]* plug-in has been loaded, but before the reduction step. It is recommended that the second *Import Mask From File [Masking]* plug-in is loaded into the pipeline straight after the first instance, as highlighted below.



After loading in the plug-in in the appropriate location, drag-and-drop the mask file into *Mask File* field in the *Model* panel. This should then show the mask added on to the existing detector mask. In the above example, the word 'MASK' has been added to the basic detector mask. All subsequent steps in the pipeline will then inherit this information. Should you wish to completely override the beamline mask, simply remove the first *Import Mask From File [Masking]* plug-in, as shown above.



Now when the custom mask is loaded, it is loaded alone without the beamline mask being added to it, as highlighted by the word 'MASK' now appearing alone in the above image without the detector, flight tube and beamstop mask present as well.



Regardless of which type of masking is performed running the subsequent pipeline will display the reduced line plot from the masked image in the *Output* panel. In the above image, the output displayed is with just the custom mask loaded as without the grid mask clearly visible dips appear in the reduced data, producing a clear result for demonstration purposes.

Previous: Viewing results in DAWN | **Next:** Exporting reduced data or detector images

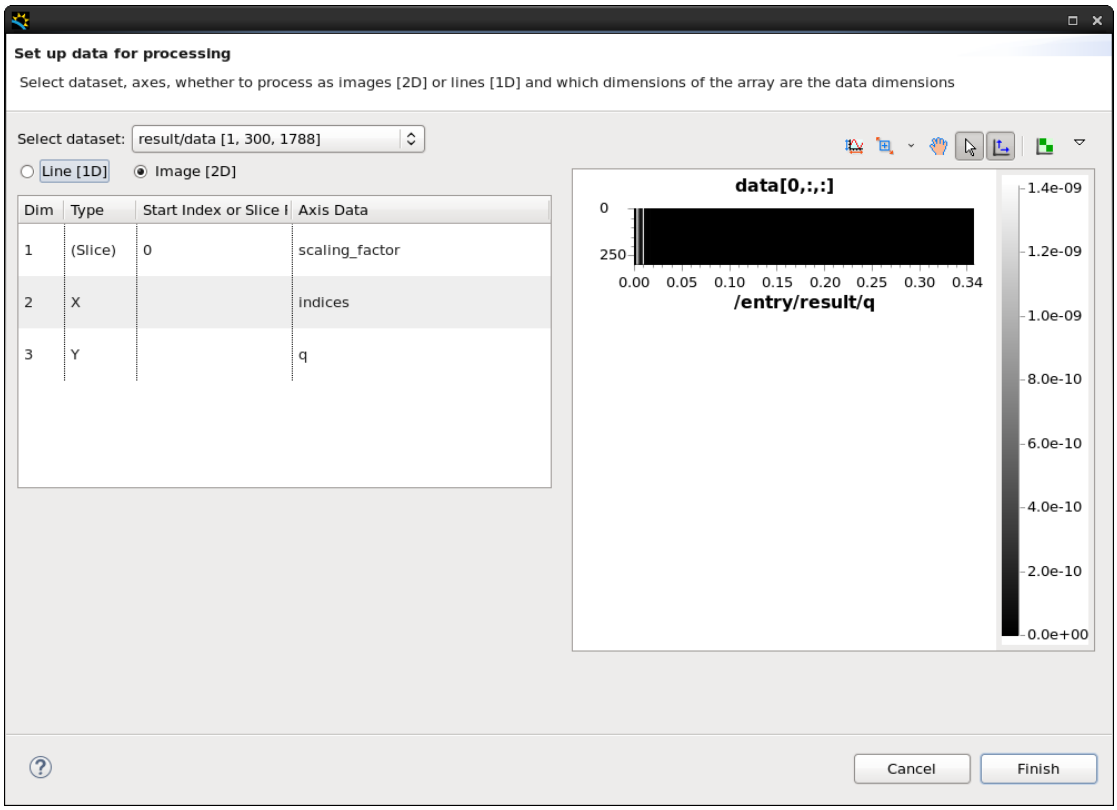
Exporting reduced data or detector images

This page provides information on the following:

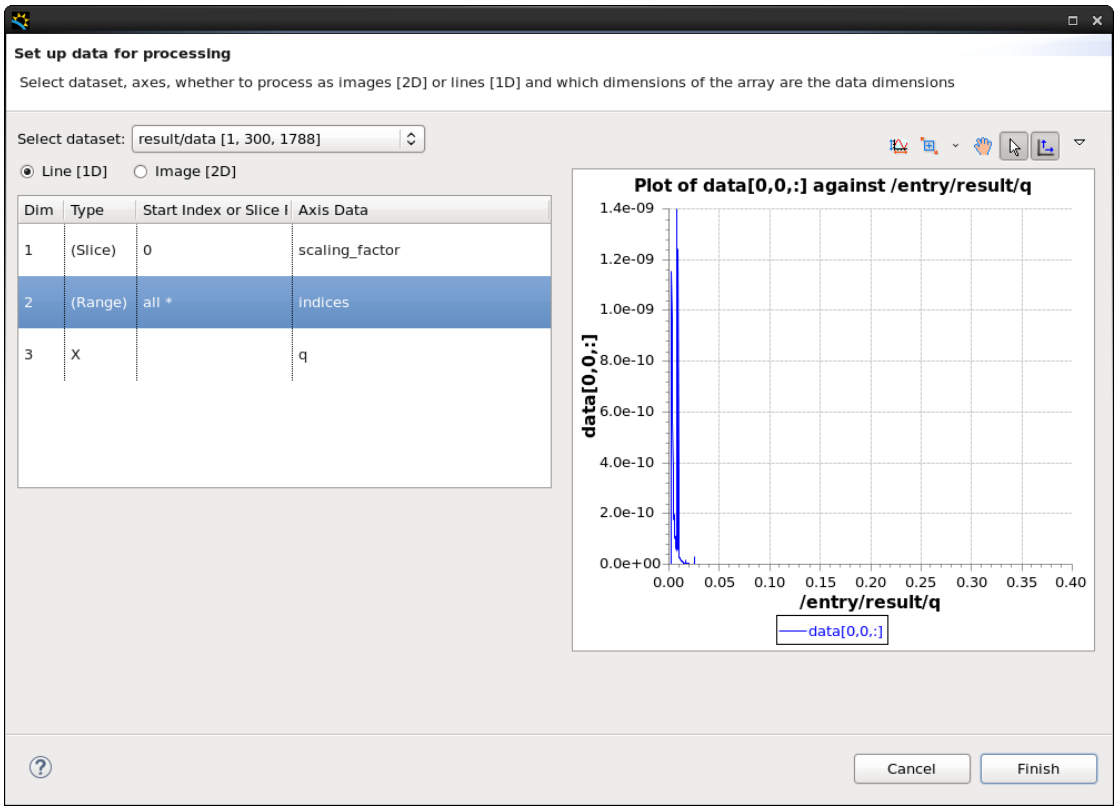
All instructions provided were performed on a Linux version of DAWN 2.7.x. Although there are no specific alterations to the way the program works between the operating systems, certain parts of DAWN might look different on your computer to the illustrations in this guide. A DAWN workspace accompanying this section of the guide can be downloaded by clicking on this link. Alternatively, just the data files highlighted below can be downloaded as a zip file I22 Beamline Calibrants.zip.

Exporting previously reduced data

It is anticipated that, under a number of different scenarios, a NeXus file containing reduced data is created without generating an ASCII text output and later on having the data in a text format becomes crucial. Although there are several ways to do this in DAWN, the easiest way in the *Processing* perspective is to use the *Export to Text File* plug-in.



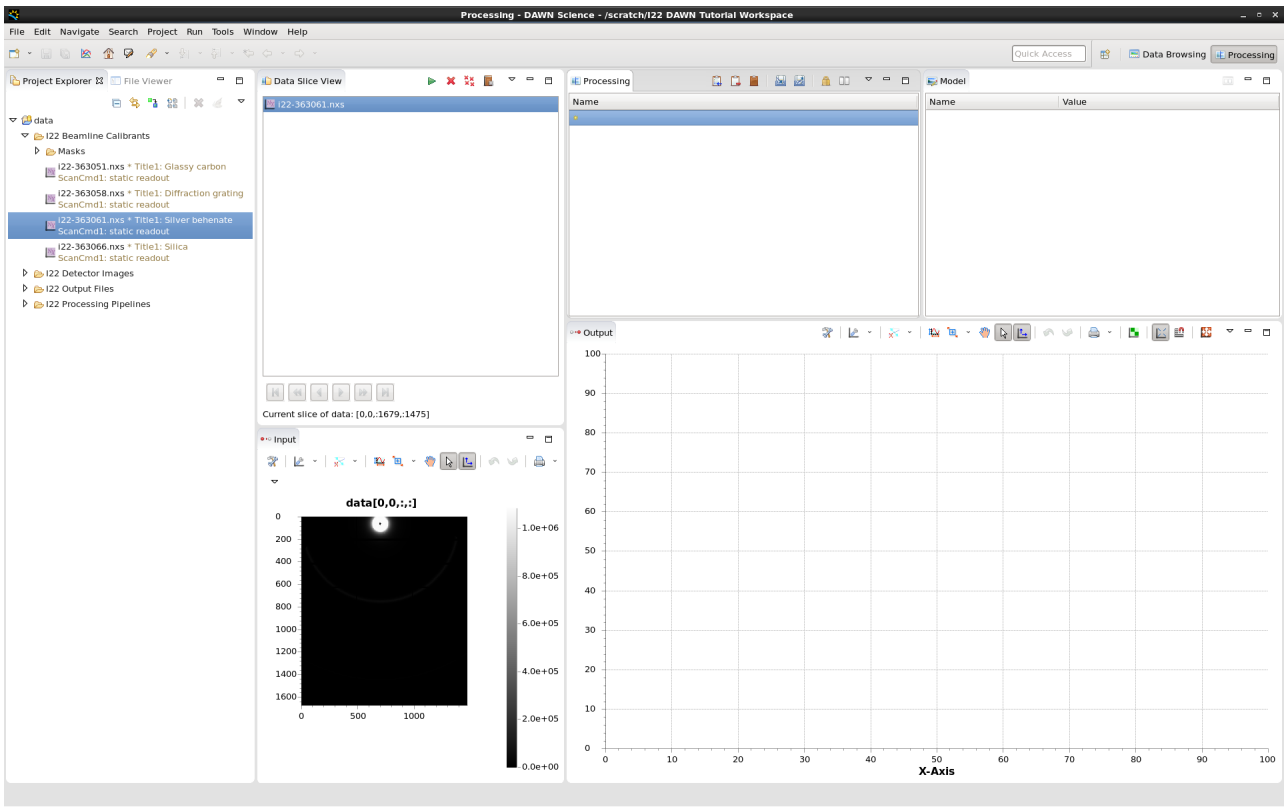
First, drag-and-drop the desired processed NeXus file, or files, for export into the *Data Slice View* panel. Doing so will open up the *set up data for processing* wizard, as shown above. If the NeXus file contains more than one reduced frame a 2D representation of the data will be shown, otherwise it will show a line plot of the reduced frame. In either scenario, change the value of the radio button underneath *Select dataset:* from *Image [2D]* to *Line [1D]* and from the drop-down box in the *Type* column, change the entry for *Dim 2* from *(Slice)* to *(Range)*, as shown in the example below.



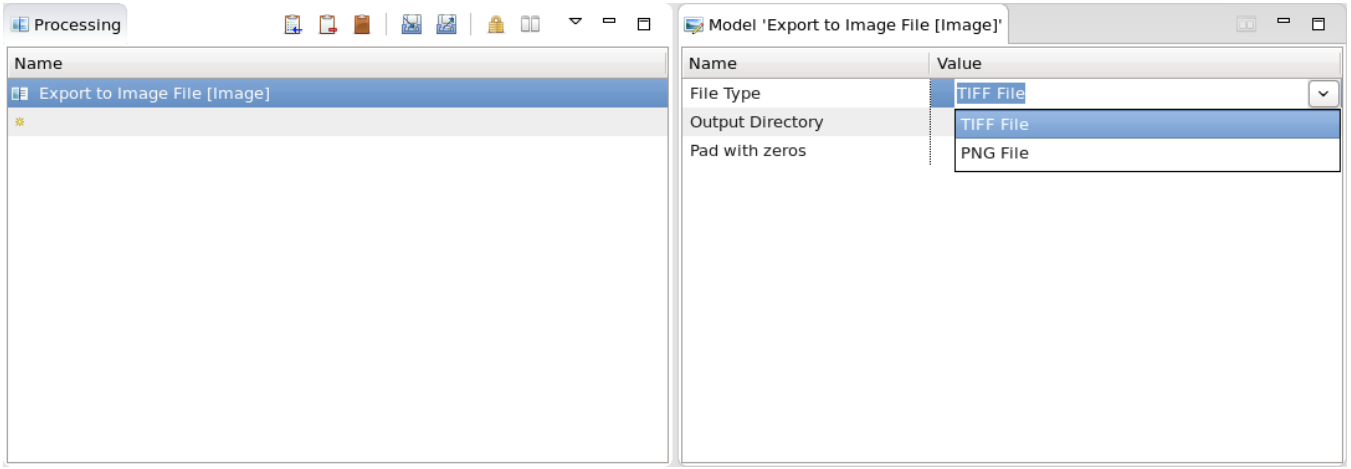
After changing these values, add the *Export to Text File* plug-in into the *Processing* pipeline by clicking on the small star icon in the *Processing* pane and type in 'export', this should filter down the list of plug-ins so that *Export to Text File* is now visible, double click on this plug-in to enter it to the processing pipeline. As highlighted on Reducing data from I22 using DAWN#ASCIIOutput, set up the options in the *Model* panel as desired and then click on the *Process all files* button in the *Data Slice View* panel to export out all the reduced SAXS or WAXS data located inside the processed NeXus file.

Exporting detector images

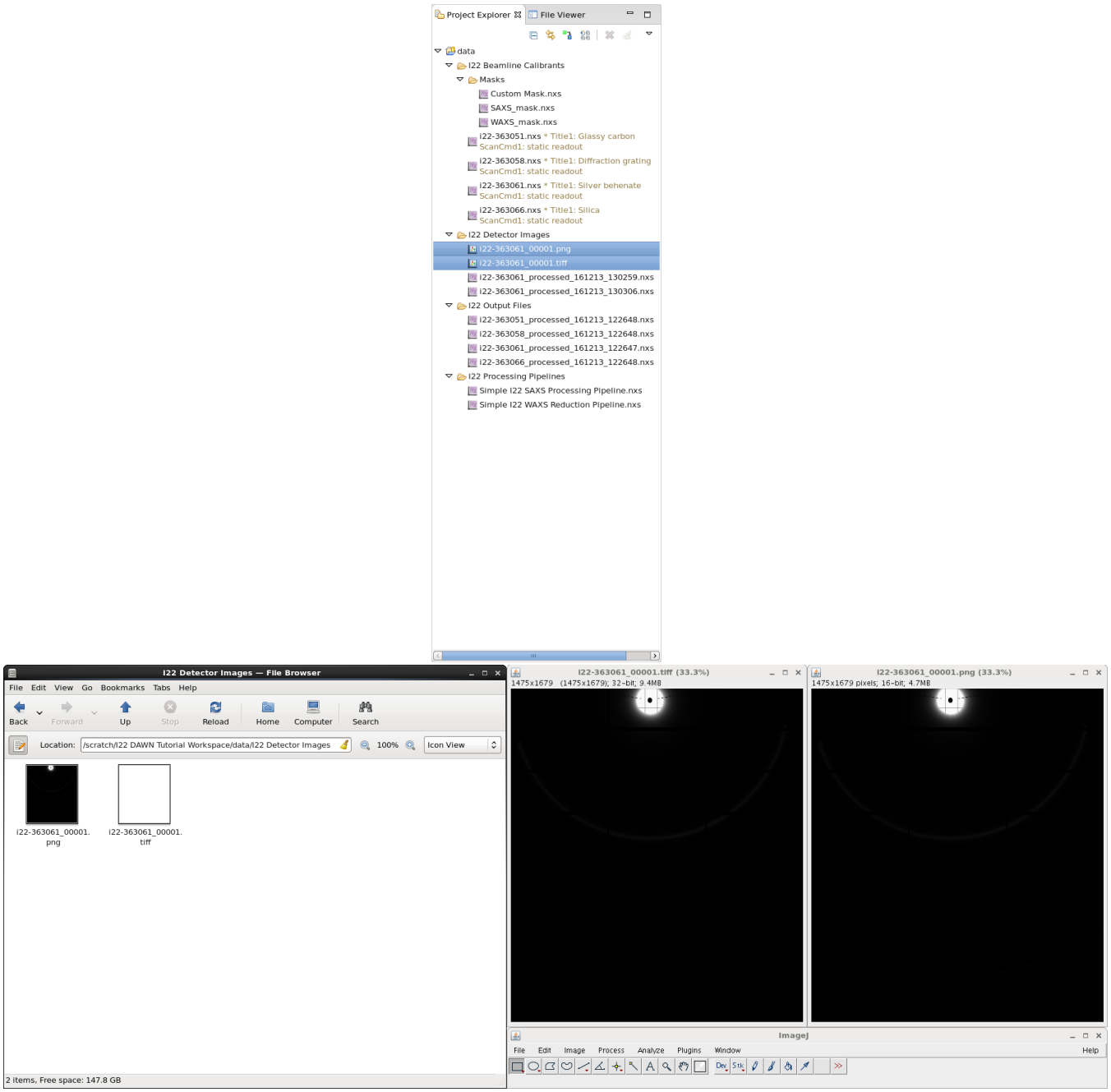
After reducing and processing experimental results, it may be of use to export select images from one or more detector images to present in articles, posters or presentations. Although there are several ways to do this in DAWN, the easiest way in the *Processing* perspective is to use the *Export to Image file [Image]* plug-in. Currently this plug-in will allow for either the export of 32-bit TIFF files or 16-bit PNG files. **Please note: saving detector images as PNG files will result in data loss. The PNG format should only be used for publishing images and not data analysis, see the section PNG data loss below for more information.**



First, load the desired NeXus file, or files, as desired for export into the *Data Slice View* panel as highlighted above. Then click on the small star icon in the *Processing* pane and type in 'export', this should filter down the list of plug-ins so that *Export to Image File [Image]* is now visible, double click on this plug-in to enter it to the processing pipeline.



In the *Model* panel for the plug-in will appear the export options, there is a drop-down list option for file type, giving TIFF and PNG as possible outputs, an option to select a location for the images to be saved and an option to pad the output file name with zeros, useful for sorting multi-frame images. Taking heed of the notice at the beginning of this section, select the options desired and then click on the *Process all files* button in the *Data Slice View* panel.



After refreshing the *Project Expolrer* panel, or navigating to the export folder in the operating system, the exported files should be appear. In the above example both a TIFF and PNG file were exported from the same NeXus file and opened in ImageJ, note that although the operating system has only recognised the PNG file in the file explorer, both files are viewable in ImageJ.

PNG data loss

When recording detector images each pixel represents a physical electronic unit which counts the number of x-rays that interact with it per unit time. Current generation detectors are capable of recording values on the order of one million counts per second, meaning that each pixel must be able to store a number between zero and one million. In decimal one million is a seven digit number (1000000), however, in binary the smallest length of number that can record one million would be 20 digits long or, in computing terminology, 20 bits.

The PNG file format has a limitation of only being able to record digits of up to 16 bits (or digits) long, in decimal this would represent integer values between 0 and 65536. Therefore if we are to record values above this value there are two options:

1. Assign any value above 65536 as 65536
2. Scale and round the numbers to fit the new range

DAWN reverts to using the second option. However, this means that *resolution* of your results is decreased as rather than representing the true values it becomes necessary to scale the result, potentially by up to a factor of 16. In essence this can be thought of as binning, or compressing, the raw result which is rarely desired.

Therefore, if you truly wish to export the raw detector images for analysis outside of DAWN it is **highly recommended** that you export the images as 32-bit TIFF files as this will preserve all the data recorded by the detector (as well as making finding the detector mask easier as these pixels have a value of -1). The drawback of this is that whilst 16-bit PNG files are recognised and openable by almost all applications and systems, 32-bit TIFF files are a speciality and are less well recognised. However, both ImageJ and a number of Python packages, in addition to a number of other scientific packages, are fully capable of reading 32-bit TIFF files.

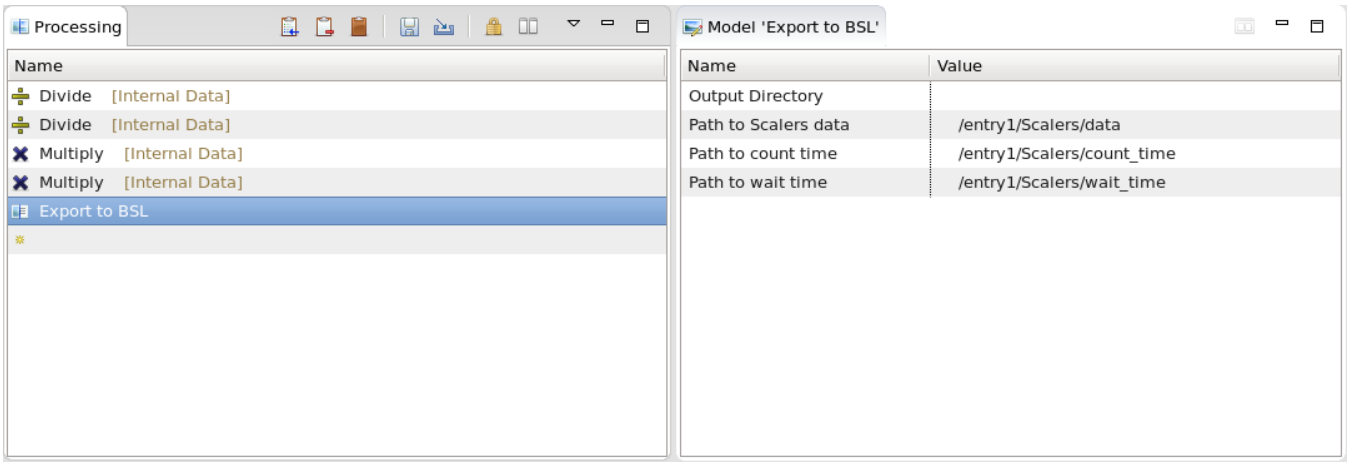
Previous: **Custom detector masks** | **Next:** **Data Analysis**

BSL files

Exporting to BSL from DAWN

The default file format at Diamond is NeXus HDF5 files. However, we are aware of a number of applications where exporting files into other formats, such as TIFF or BSL is either useful, or crucial, depending on the analysis that is to be performed. It is hoped that in time the functionality provided by programs such as FibreFix and Fit2D are either implemented in DAWN or other, more contemporary, programs such as SasView, for which options to export data to are present within DAWN.

However, in the interim there is a processing plug-in entitled *Export to BSL* which will allow detector images to be exported from DAWN in the BSL format.



Depending on the analysis that is required, it may or may not be beneficial to perform the sample thickness normalisation, transmission correction and absolute scale multiplication, as shown above, before exporting to BSL. If these steps are not required, simply just place the *Export to BSL* plugin in the processing pipeline.

If it is anticipated that only a few files from an experimental run are required, it would be appreciated if this methodology were to be followed for those frames, rather than exporting the whole experimental run to BSL files as, typically, the lossless compression provided by HDF5 reduces a diffraction image from 10 MB to 2.5 MB, a functionality *not* provided by binary BSL files.

However, if all the data obtained from an experimental run is to be analysed using software that requires BSL files, the following section provides information on how to enable this for your visit.

Creating BSL files during an experimental run

As of shutdown 2 2015, BSL files will not be automatically created for every scan file. They are large and, for most visits, not required.

To enable those users who do wish to take BSL versions of their files away with them a script is available to enable it. This is typed at the command prompt in GDA. This function runs the script nexus2otoko.py. The behaviour of the nexus2otoko script is exactly the same as before. If the file it's converting is in a visit directory, it will be put in a dedicated BSL folder (something like /dls/i22/data/2015/ab1234-1/bsl/), otherwise it will be in the same place as the original file. Files in commissioning visits will still be ignored, even if passed to the convertToBsl function. **It should be noted that these scripts run from inside the GDA**

- To automatically convert all new files as they are created (previous behaviour)

```
bslUtils.createBslFiles(True)
```

- To disable the conversion again

```
bslUtils.createBslFiles(False)
```

- To convert a single file to BSL

```
bslUtils.convertFileToBsl("/path/to/file.nxs")
```

- To convert all the nexus files (.nxs extension) in a directory (eg at the end of a visit)

```
bslUtils.convertAllFilesToBsl("/path/to/directory")                #This will only convert files in
the specified directory.
bslUtils.convertAllFilesToBsl("/path/to/directory", recursive=True)  #This will also convert files in
the subdirectories.
```

- To check which command is currently being used

```
bslUtils.bslConversionCommand()
```

For this help, you can run "`help bslUtils`" from the Jython command line in GDA.

Headless DAWN Operation

In addition to operating DAWN through its user interface it is also possible to process data using DAWN, headlessly on a command line. This functionality is used by GDA at Diamond in order to reduce and analyse user data. This page is designed to describe how it would be possible for end users, or other facilities, to set up such functionality.

Three elements are required in order for DAWN to process data headlessly:

1. A dataset
2. A processing pipeline NeXus file
3. A JSON configuration file

For the purposes of this guide it is assumed that the reader is in possession of a dataset which can be both read and processed by DAWN's processing perspective. Additionally it is also assumed that the reader has read the previous section of this manual dedicated to the creation and saving of a processing pipeline, the most pertinent page can be Reducing data from I22 using DAWN.

Building on this foundation the reader is encouraged to JSON file walkthrough in order to understand what will be required for setting up a DAWN JSON file. In addition to the information laid out on that page it is worth noting that the "filePath" and "outputFilePath" entries will need to be populated with the source file path and output file path, respectively, before using DAWN headlessly.

N.B. This does mean that for each dataset that requires reduction a *unique* JSON file is required. Whereas multiple datasets can be processed using the same pipeline file a unique JSON file is required to facilitate data processing as the contain the source and target file paths.

With these three elements in hand it is then possible to invoke DAWN from the command line using the following:

In the above example, the /path/to/work/area/.eclipse can be located in a temporary directory (/tmp/.eclipse is a reliable choice) as no data will be saved there, just temporary information.

Configuring a script, or custom program, to generate the required JSON file and invoking the required shell command will then allow the reader to process data using DAWN headlessly.

Data Analysis

This section provides a guide showing some of the ways that DAWN can be used to analyse SAXS and WAXS data obtained from I22.

As there are such a large number of potential analyses that can be performed on the diffraction images obtained; this guide will predominantly focus on:

- Mathematical formulae in the *Processing* pipeline
- Python scripts in the *Processing* pipeline
- Guinier analysis
- Kratky analysis
- Porod analysis
- Crystallite parameters
- Orientation calculations

After reading this section of the guide it is hoped that you will be able to perform a number of analyses on results obtained, have a grounding in how DAWN works allowing you to explore and use other functions in DAWN and be able to run externally sourced, or custom written, Python scripts from within DAWN directly on results, raw or reduced, obtained at I22.

N.B. This version of the guide is intended for use with DAWN 2.4.x.

Listed below are direct links to the home pages of some of the software mentioned in this section and Diamond's Data reduction package for SAXS data:

- EDNA Pipeline for BioSAXS
- ATSAS
- Scatter
- SasView
- Fibres
- CCP-SAS (Collaborative Computational Project for advanced analysis of structural data in Chemical Biology and Soft Condensed Matter. This is an SI2-CHE Cyberinfrastructure Project addressing Grand Challenges in the Chemical Sciences)
- Legacy CCP13 (A now ceased CCP for develoing primarily fibre diffraction data analysis software)

These programs and more are also available for download *via* the SAS Portal

---

Next page: Introduction

Introduction

SAXS data comes in many shapes and sizes and there are a range of tools available to help the user analyse their data once corrected for experiment and beamline conditions. Some of these can be accessed *via* the Diamond computing servers, including DAWN derived tools for data reduction, peak fitting and other data analysis techniques.

This section of the guide is designed primarily to demonstrate different data analysis techniques made available, primarily, through using DAWN. As such, it will focus on several in-built routines, as well as highlighting routes that have been designed to allow users to perform their own mathematical manipulations in-line whilst performing data reductions and analysis and ways to implement Python scripts or programs in-line as well.

However, DAWN is only one piece of software. There are many other software packages which have been written to analyse small angle scattering data and so this section will also touch upon some of these pieces of software. A quick list of some prominent scattering software packages include:

- ATSAS - From the Svergun group for Biological Solution Scattering
- BornAgain - A tool for the analysis of grazing incidence scattering patterns
- McSAS - A tool for the analysis of SAS patterns
- SASfit - A tool for the analysis of SAS patterns
- SasView - For a number of small angle fitting routines
- Sctter - SAXS & BioSAXS fitting routines

Software available whilst at Diamond

To find out what modules are available whilst here at Diamond, using our Linux workstations:

1. Open a command prompt and type:

   This will provide an exhaustive list of the software available for that machine. Looking, or searching, through this list will show you whether the programs you want to use are available.
2. Subsequently, typing in:

   This will load the software package, identified, from the list obtained in step 1.
3. Finally, typing in the name of the module just loaded into the command prompt will now load this package:

Previous: Data Analysis | **Next:** Mathematical formulae in the Processing pipeline

Analysis software at Diamond

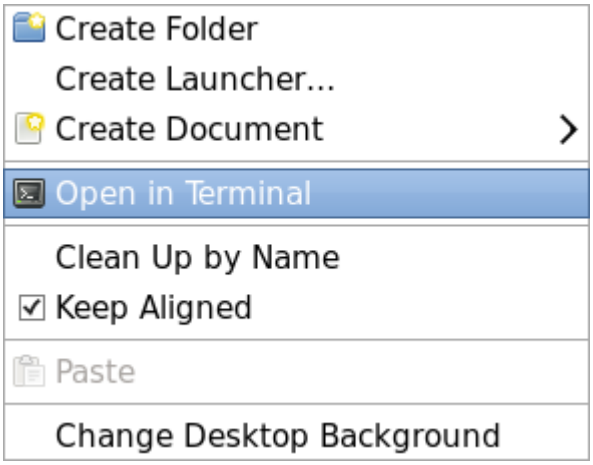
DAWN provides access to number of analysis and curve fitting tools, primarily *via* the Processing perspective, with the majority of the SAS specific tools described in the following pages of this manual. However, as mentioned in the Introduction, there are a number of specialised software packages for the analysis of SAS data a number of which are installed and available for use at Diamond on our Linux machines.

The packages available are (click to jump to instructions):

BornAgain

BornAgain is a software package to simulate and fit small-angle scattering at grazing incidence. It supports analysis of both X-ray (GISAXS) and neutron (GISANS) data. Its name, BornAgain, indicates the central role of the distorted wave Born approximation in the physical description of the scattering process. The software provides a generic framework for modeling multilayer samples with smooth or rough interfaces and with various types of embedded nanoparticles.

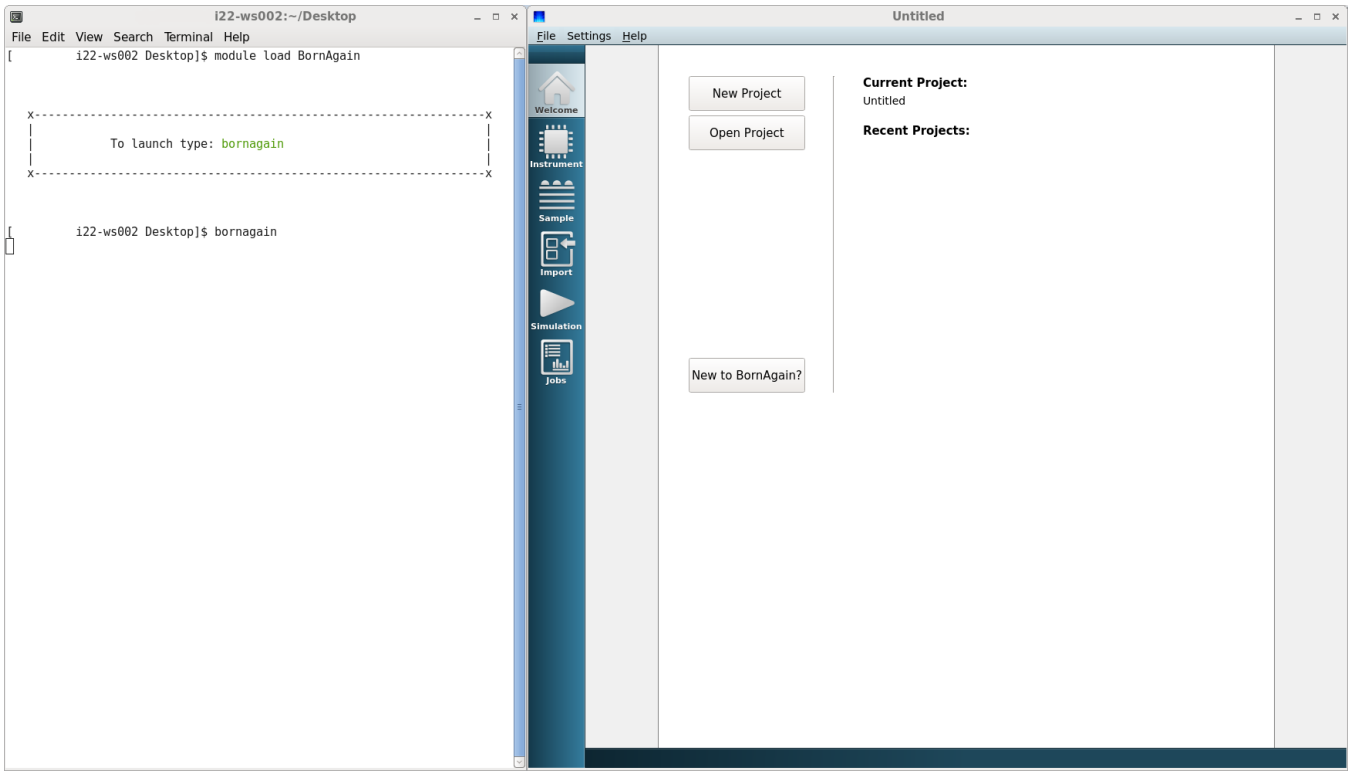
To load BornAgain, right click on the desktop to open a contextual menu and select 'Open in Terminal', as highlighted below, to open a Terminal session.



In the Terminal session that opens you will then need to type in the following two commands, **N.B.** capitalisation is important. After typing in each line, press return.

```
module load BornAgain
bornagain
```

As highlighted below, after typing in these two commands into the Terminal session (left hand side) BornAgain will load (right hand side).



For further information about BornAgain and how to use this software package, please do read the documentation on the BornAgain website which can be found here. For more information on the theory and background that underpins BornAgain there is a PDF manual available here.

The authors of the software request that you include the following reference in publications/presentations that made use of BornAgain for data analysis:

C. Durniak, M. Ganeva, G. Pospelov, W. Van Herck, J. Wuttke (2015), BornAgain — Software for simulating and fitting X-ray and neutron small-angle scattering at grazing incidence, version <...>, http://www.bornagainproject.org .

McSAS

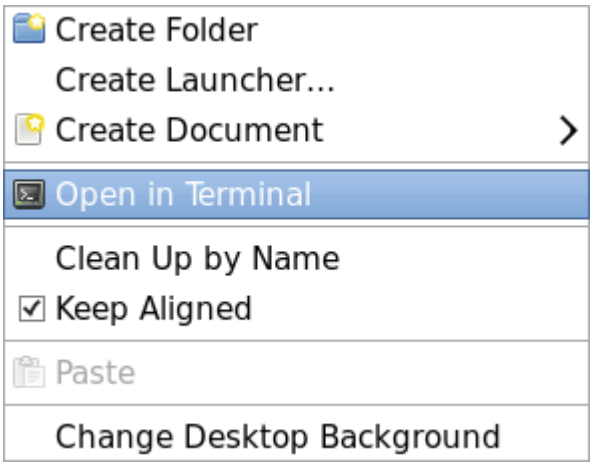
McSAS is a tool for analysis of SAS patterns. This tool can extract form-free size distributions from small-angle scattering data using the Monte-Carlo method as described in:

Brian R. Pauw, Jan Skov Pedersen, Samuel Tardif, Masaki Takata, and Bo B. Iversen. "Improvements and Considerations for Size Distribution Retrieval from Small-angle Scattering Data by Monte Carlo Methods." Journal of Applied Crystallography 46, no. 2 (February 14, 2013). DOI:10.1107/S0021889813001295 .

The GUI and latest improvements are described in:

I. Breßler, B. R. Pauw, A. F. Thünemann, "McSAS: A package for extracting quantitative form-free distributions". Journal of Applied Crystallography 48: 962-969, DOI: 10.1107/S1600576715007347 .

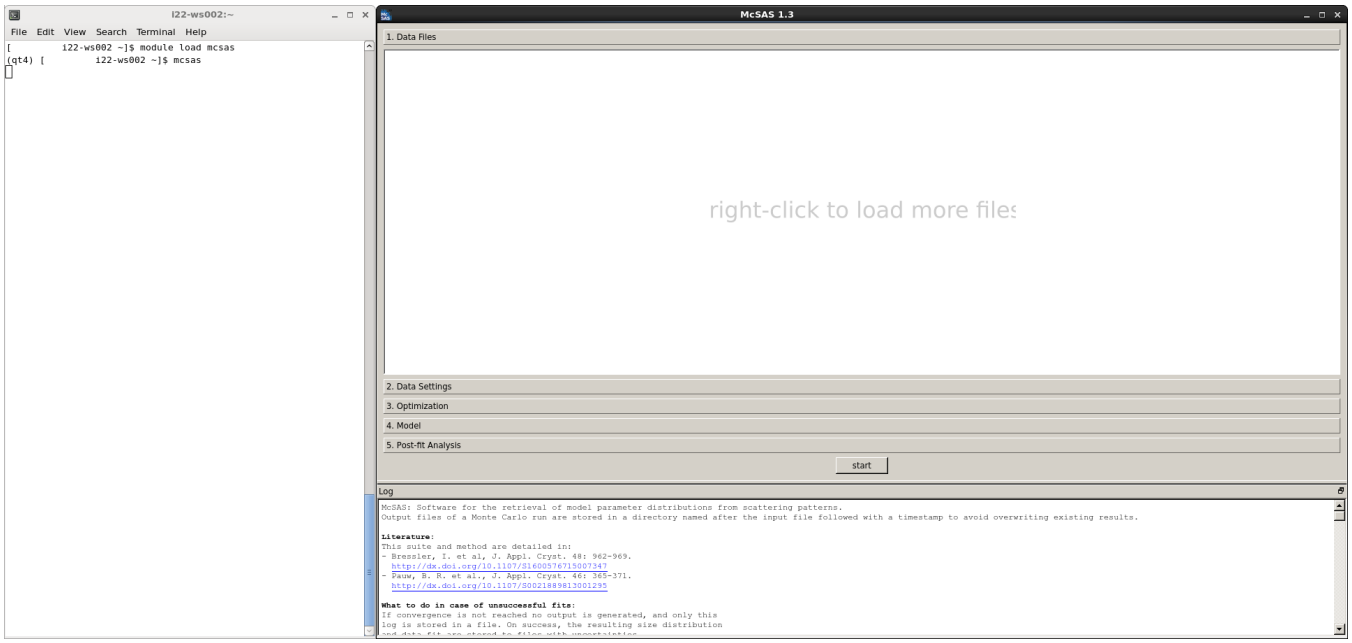
To load McSAS, right click on the desktop to open a contextual menu and select 'Open in Terminal', as highlighted below, to open a Terminal session.



In the Terminal session that opens you will then need to type in the following two commands, **N.B.** capitalisation is important. After typing in each line, press return.

```
module load mcsas
mcsas
```

As highlighted below, after typing in these two commands into the Terminal session (left hand side) McSAS will load (right hand side).



For further information about McSAS and how to use this software package, please do read the documentation on the McSAS website which can be found here.

The authors of the software request that you include the following references in publications/presentations that made use of McSAS for data analysis:

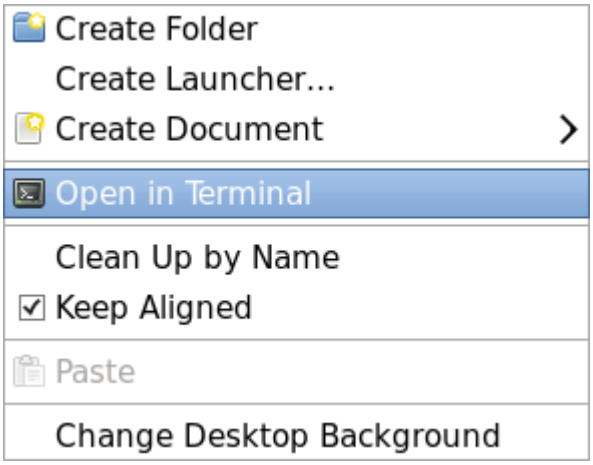
B. R. Pauw, J. S. Pedersen, S. Tardif, M. Takata, and B. B. Iversen, Journal of Applied Crystallography, 2013, 46, 365-371. DOI:10.1107/S0021889813001295 .

I. Breßler, B. R. Pauw, A. F. Thünemann, Journal of Applied Crystallography, 2015, 48, 962-969. DOI: 10.1107/S1600576715007347.

SASfit

SASfit has been written for analyzing and plotting small angle scattering data. It can calculate integral structural parameters like radius of gyration, scattering invariant, Porod constant. Furthermore it can fit size distributions together with several form factors including different structure factors. Additionally an algorithm has been implemented, which allows to simultaneously fit several scattering curves with a common set of (global) parameters. This last option is especially important in contrast variation experiments or measurements with polarised neutrons. The global fit helps to determine fit parameters unambiguously which by analyzing a single curve would be otherwise strongly correlated. The program has been written to fulfill the needs at the small angle neutron scattering facility at PSI (kur.web.psi.ch). The numerical routines have been written in C whereas the menu interface has been written in tcl/tk and the plotting routine with the extension blt.

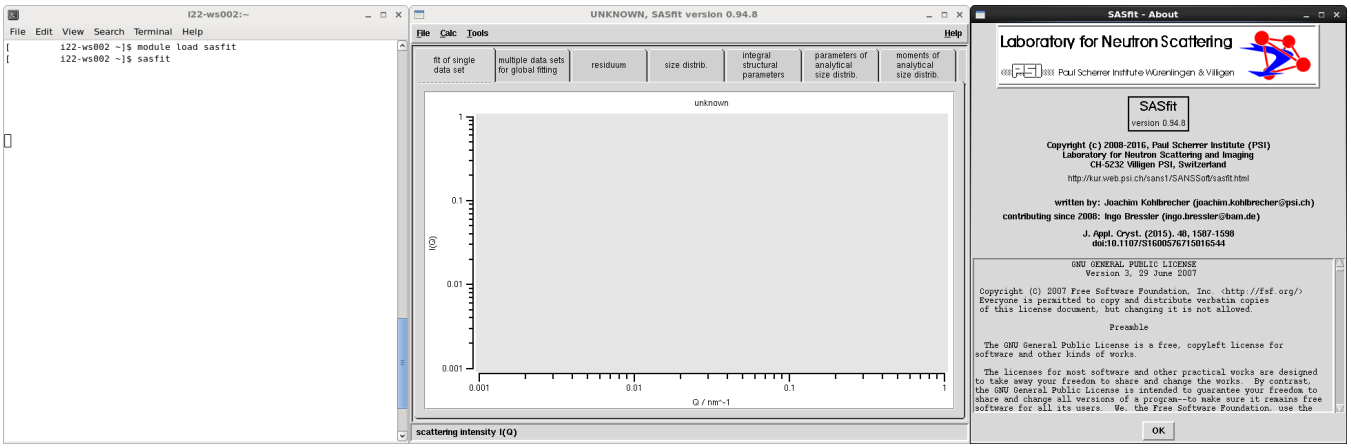
To load SASfit, right click on the desktop to open a contextual menu and select 'Open in Terminal', as highlighted below, to open a Terminal session.



In the Terminal session that opens you will then need to type in the following two commands, **N.B.** capitalisation is important. After typing in each line, press return.

```
module load sasfit
sasfit
```

As highlighted below, after typing in these two commands into the Terminal session (left hand side) SASfit will load (right hand side).



For further information about SASfit and how to use this software package, please do read the documentation on the SASfit website which can be found here.

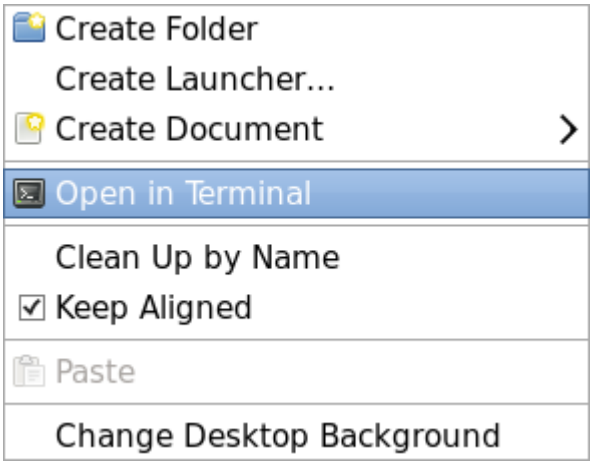
The authors of the software request that you include the following reference in publications/presentations that made use of SASfit for data analysis:

> *I. Breßler, J. Kohlbrecher and A. F. Thünemann, Journal of Applied Crystallography, 2015, 48, 1587-1598. DOI: 10.1107/S1600576715016544.*

SasView

SasView is a Small Angle Scattering Analysis Software Package, originally developed as part of the NSF DANSE project under the name SansView, now managed by an international collaboration of facilities. Feedback and contributions are welcome and encouraged.

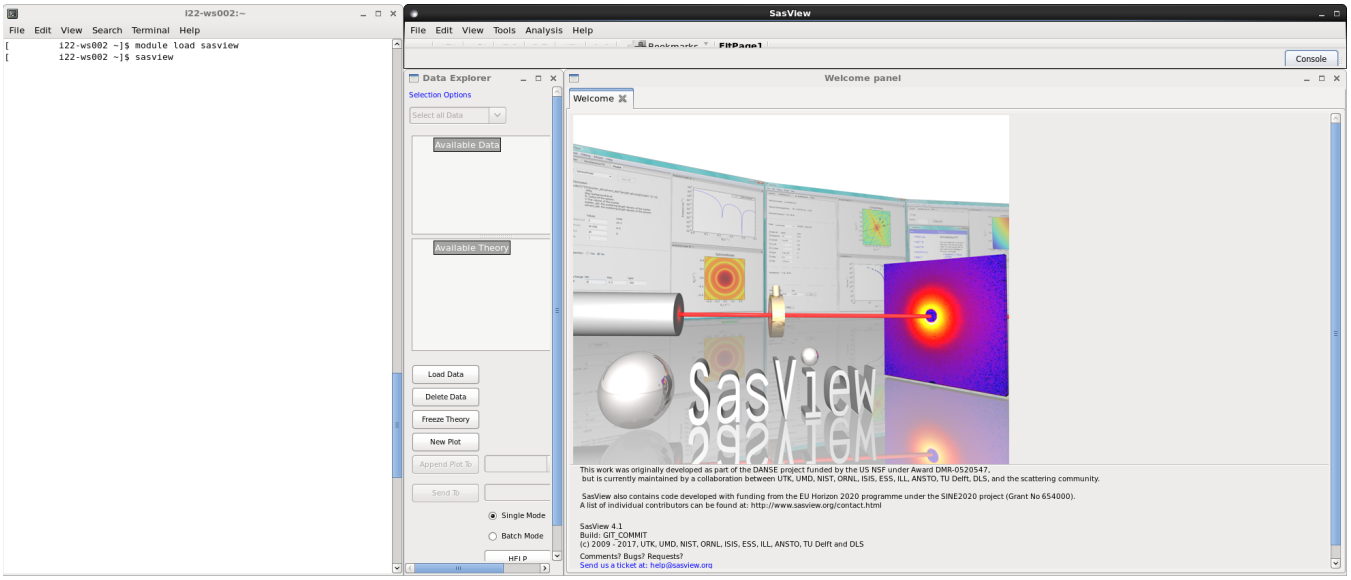
To load SasView, right click on the desktop to open a contextual menu and select 'Open in Terminal', as highlighted below, to open a Terminal session.



In the Terminal session that opens you will then need to type in the following two commands, **N.B.** capitalisation is important. After typing in each line, press return.

```
module load sasview
sasview
```

As highlighted below, after typing in these two commands into the Terminal session (left hand side) SasView will load (right hand side).



For further information about SasView and how to use this software package, please do read the documentation on the SasView website which can be found here. A PDF version of this manual can be found here. Tutorials on how to use SasView can be found here.

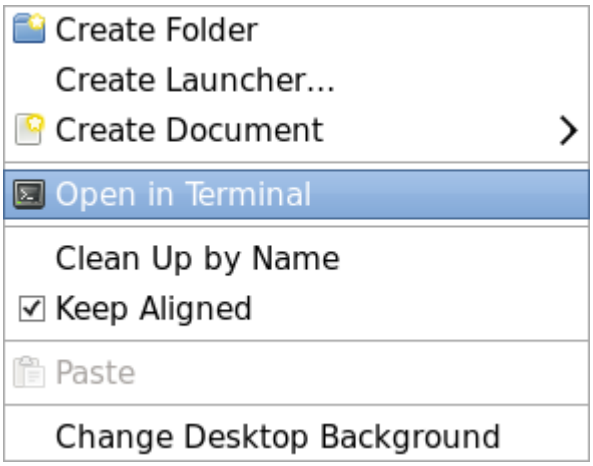
The authors of the software request that you include the following acknowledgement in publications/presentations that made use of SasView for data analysis:

> *This work benefited from the use of the SasView application, originally developed under NSF award DMR-0520547. SasView contains code developed with funding from the European Union's Horizon 2020 research and innovation programme under the SINE2020 project, grant agreement No 654000.*

Scåtter

Scåtter has been completely re-written and utilizes several new structures for handling datasets. PDB files can be dropped in like *.dat files which will then create corresponding P(r)-distribution function and Intensity files. This is useful for comparing models in real-space against experimental data. There is also a new archiving feature for taking selected data and writing them to a separate directory with associated image files. The release has been tested for the past 4 months and should be stable.

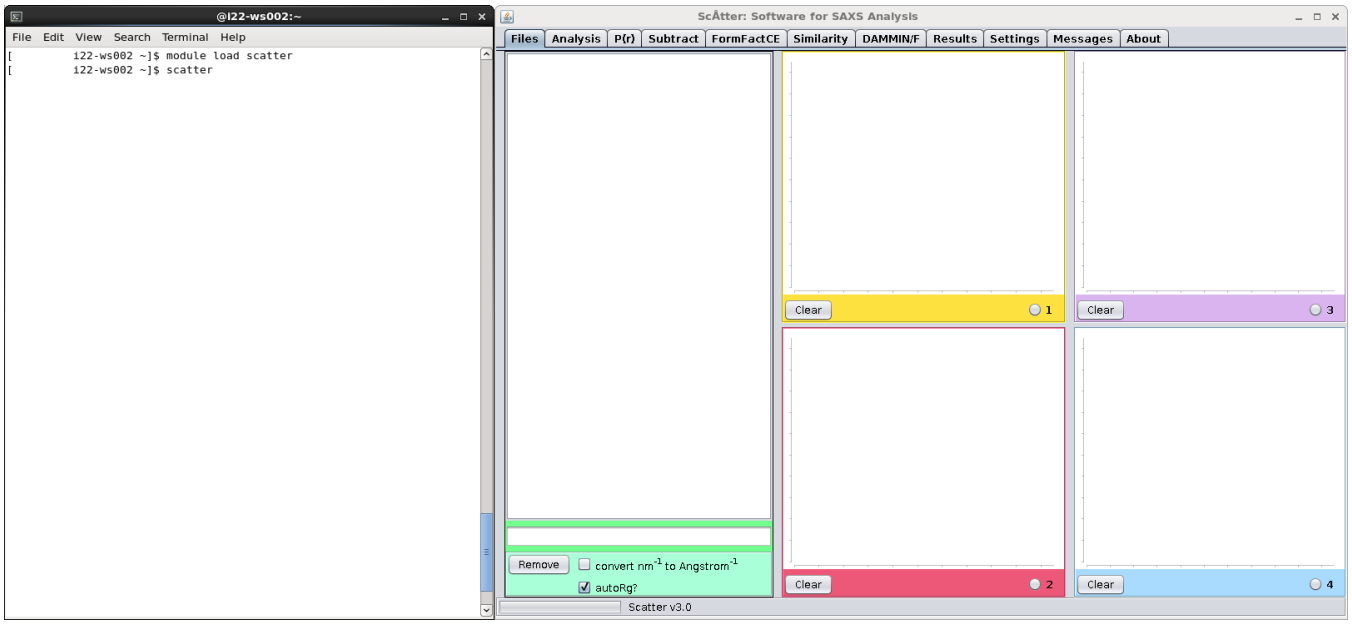
To load Scåtter, right click on the desktop to open a contextual menu and select 'Open in Terminal', as highlighted below, to open a Terminal session.



In the Terminal session that opens you will then need to type in the following two commands, **N.B.** capitalisation is important. After typing in each line, press return.

```
module load scatter
scatter
```

As highlighted below, after typing in these two commands into the Terminal session (left hand side) Scåtter will load (right hand side).



For further information about Scåtter and how to use this software package, please do read the documentation on the Scåtter website which can be found here.

The authors of the software request that you include the following reference in publications/presentations that made use of Scåtter for data analysis:

R. P. Rambo and J. A. Tainer, Nature, 2013, 496, 477-481. DOI: 10.1038/nature12070.

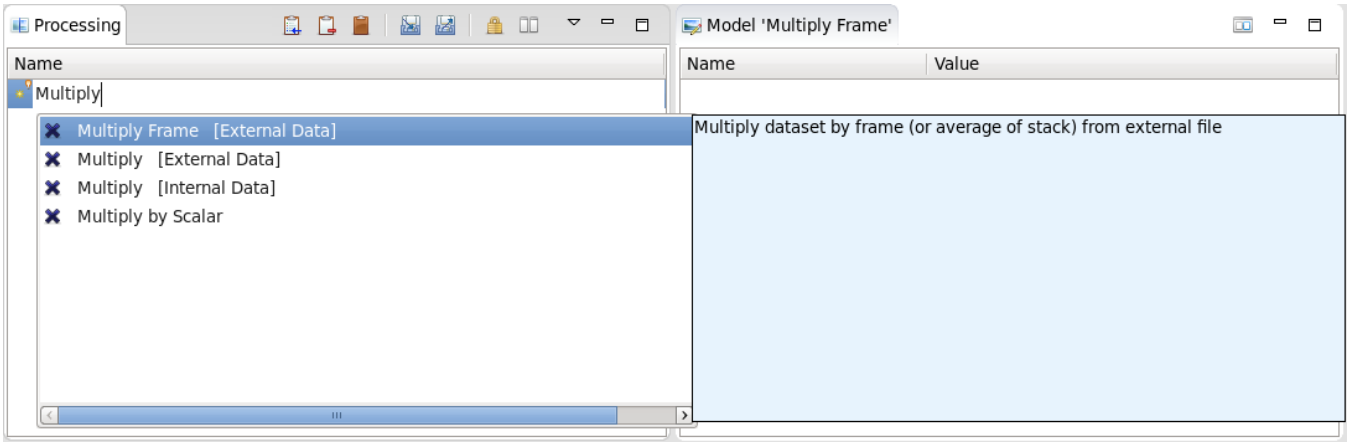
Mathematical formulae in the Processing pipeline

This page provides information on the following:

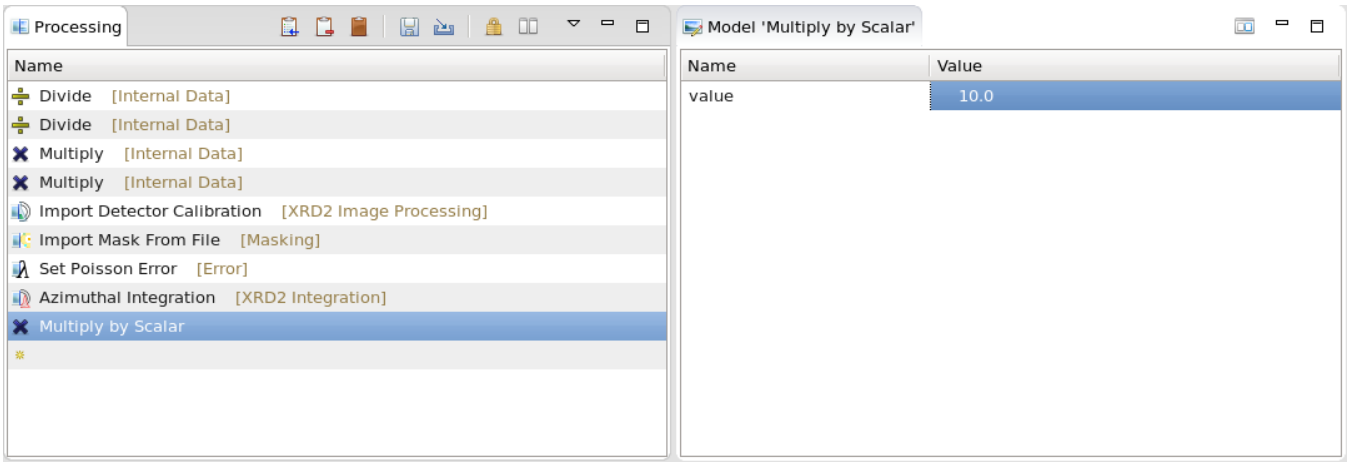
All instructions provided were performed on a Linux version of DAWN 2.7.x. Although there are no specific alterations to the way the program works between the operating systems, certain parts of DAWN might look different on your computer to the illustrations in this guide.

Simple mathematical operators

As alluded to when discussing setting up a standard reduction pipeline, DAWN provides processing plug-ins to facilitate addition, subtraction, multiplication and division by either a fixed value, a value (or values) held within the data file, a value (or values) held in another file or an entire frame of data. Adding any of these to a processing pipeline is achieved by clicking on the small star icon in the *processing* panel and type in the name of the operator desired, this should filer the list sufficiently that the processing plug-in is visible, subsequently double-clicking on the desired plug-in will place it into the pipeline.



If in doubt, the tooltips provided for each plug-in, as highlighted above, should provide a reasonable indication as to what the plug-in will do. After loading the desired plug-in entering in the required values into the *Model* panel will then manipulate the data as requested.

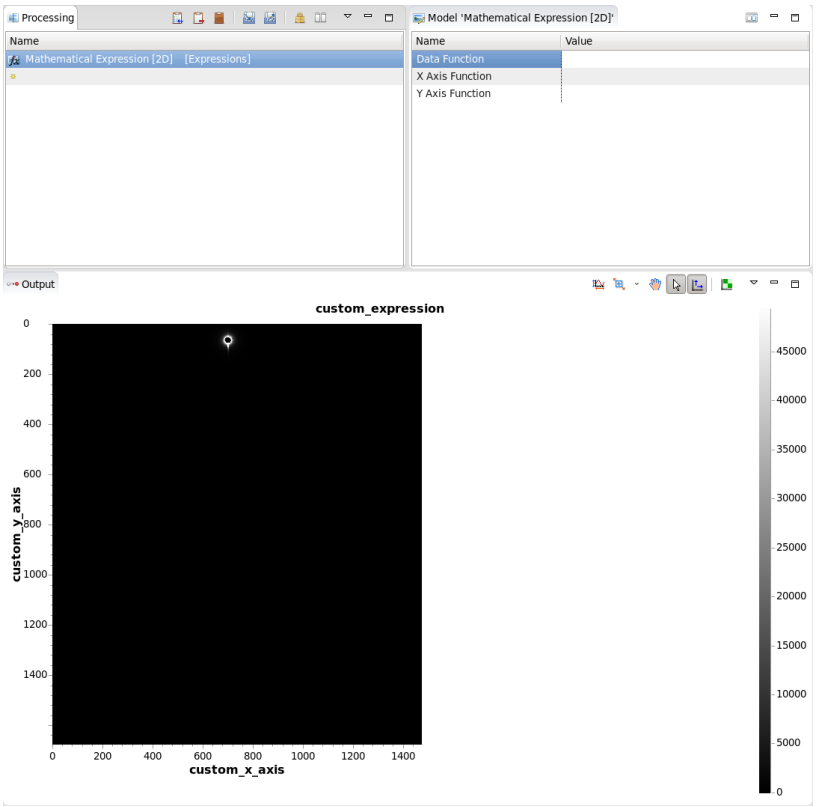


Formulae

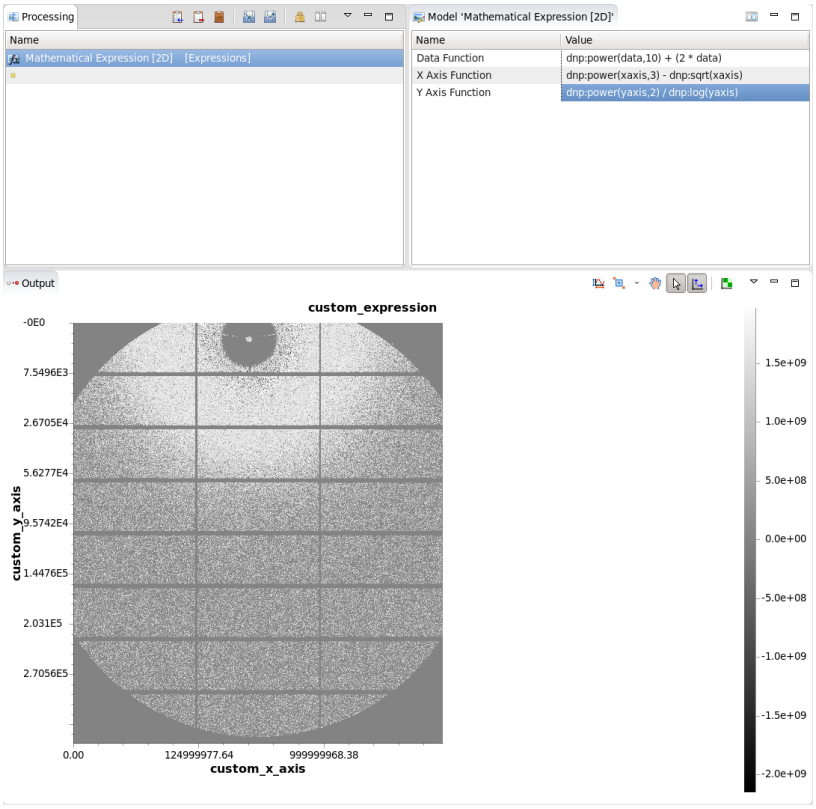
There are a wide range of mathematical functions available to include in any given processing pipeline through using the *Mathematical Expression [2D] [Expressions]* and *Mathematical Expression [2D] [Expressions]* plug-ins. A full list of these functions can be found by following these links:

- Data commands (dat:)
- Scisoftpy commands (dnp:)

In order to add these into a pipeline, first consider whether the data is 2D (images) or 1D (lineplots) at the current stage of the pipeline, as this will affect the plug-in required. For this example an image will be manipulated. First, clicking on the small star icon in the *processing* panel and type in the name of the operator desired, this should filer the list sufficiently that the processing plug-in is visible, subsequently double-clicking on the desired plug-in will place it into the pipeline.



After loading the plug-in it is then possible to mathematically modify the data, the x-axis values and for 2D data the y-axis values, as desired. The example below shows two operations applied to the data and its axes.



It is worth noting that if two datasets are of equal length, they can be operated on with each other, this will be demonstrated later in this section.

Previous: Introduction | **Next:** Python scripts in the Processing pipeline

The Expression Engine in DAWN

The expression engine can be used in the 'Function Fitting' tool and the 'Mathematical Expression 1D' Operation in the Processing Perspective.

Syntax

The expression engine uses the typical computer syntax for the four basic arithmetic operations (+-*/). Special functions are accessed by namespaces (described below). The syntax for separating namespace components is the colon (:), rather than the usual dot (.). For example, the correct syntax to access the function `exp()` in the namespace `dnp`, is shown below.

```
dnp:exp()
```

If a dot is entered by mistake, the expression editor will show an error. This can be cleared by right clicking on the list of expressions to refresh the list and then deleting the erroneous expression.

Namespaces

The expression engine incorporates a set of names spaces. These are:

Namespace	Purpose
dat	Data manipulation functions, treating the array as a matrix or retrieving statistics
dnp	Mathematical functions operating on scalars or arrays of data
func	Function fitting functions
im	Image processing functions
lz	Lazy evaluation functions

Clicking on a namespace of interest in this table will jump you to the appropriate section on this page.

---

dat

The 'dat' namespace exposes the functions defined in /org.dawnsci.jexl/src/org/dawnsci/jexl/internal/JexlGeneralFunctions.java. These are:

- arange(dataset)
- max(dataset, int)
- mean(dataset, int)
- median(dataset, int)
- min(dataset, int)
- peakToPeak(dataset, int)
- product(dataset,int)
- reshape(dataset, int[])
- rootMeanSquare(dataset, int)
- slice(dataset, int[], int[], int[])
- slice(Dataset, start, stop, step)
- slice(Dataset, String)
- squeeze(dataset)
- stdDev(dataset, int)
- sum(dataset, int)
- tile(dataset, int[])
- transpose(dataset, int[])

---

dnp

The 'dnp' namespace exposes the functions defined in /org.dawnsci.jexl/src/org/dawnsci/jexl/internal/JexlMaths.java. These are:

- arccos(data)
- arccosh(data)
- arcsin(data)
- arcsinh(data)
- arctan(data)
- arctanh(data)
- cbrt(data)
- ceil(data)
- cos(data)
- cosh(data)
- exp(data)
- expm1(data)
- floor(data)
- log(data)
- log10(data)
- log1p(data)
- log2(data)
- maxium(data, data)
- minimum(data, data)
- pow(data, data)
- power(data)
- signum(data)
- sin(data)
- sinh(data)
- sqrt(data)
- square(data)
- tan(data)
- tanh(data)
- toDegrees(data)
- toRadians(data)

---

func

The 'func' namespace exposes fitting functions, these are:

- Gaussian(dataset, double, double, double)
- Lorentzian(dataset, double, double, double)
- PearsonVII(dataset, double, double, double, double)
- PseudoVoigt(dataset, double, double, double, double, double)

---

im

The 'im' namespace exposes the functions defined in /org.eclipse.dawnsci.analysis.dataset/src/org/eclipse/dawnsci/analysis/dataset/impl/Image.java. These are:

- adaptiveGaussianThreshold(dataset, int, boolean)
- adaptiveSauvolaThreshold(dataset, int, boolean)
- adaptiveSquareThreshold(dataset, int, boolean)
- align(dataset)
- convolutionFilter(dataset, dataset)
- derivativeSobelFilter(dataset, boolean)
- extractBlob(dataset, int)
- fanoFilter(dataset, int, int)
- findTranslation2D(dataset, dataset, IRectangularROI)
- flip(dataset, boolean)
- gaussianBlurFilter(dataset, int)
- globalEntropyThreshold
- globalMeanThreshold(dataset, boolean)
- globalOtsuThreshold(dataset, boolean)
- globalThreshold(dataset, float, boolean, boolean)
- globalThreshold(dataset, float, boolean)
- maxFilter(dataset, int[])
- maxRectangleFromEllipticalImage(dataset, double, double, int, int)
- maxRectangleFromEllipticalImage(dataset, IROI)
- meanFilter(dataset, int)
- meanFilter(dataset, int[])
- meanSummedAreaFiltermeanFilter(dataset, int)
- medianFilter(dataset, int)
- minFilter(dataset, int[])

- pseudoFlatFieldFilter(dataset, int)
- regrid_kabsch(dataset, dataset, dataset, dataset, dataset)
- regrid(dataset, dataset, dataset, dataset, dataset)
- rotate(dataset, double, boolean)
- setImageFilterService(IImageFilterService)
- setImageTransformService(IImageTransform)
- sobelFilter(dataset)

---

lz

The 'lz' namespace exposes the functions defined in /org.dawnsci.jexl/src/org/dawnsci/jexl/internal/JexlLazyFunctions.java. These are:

- rmean(dataset, int)
- rsum(dataset, int)
- slice (Dataset, int)
- slice(Dataset, int[], int[], int[])
- slice(Dataset, String)

Common methods

All of the namespaces listed above also have access to methods inherited from their Java superclasses, these are:

- equals(data)
- getClass()
- hashCode()
- notify()
- notifyAll()
- toString()
- wait(long)
- wait(long, int)

Python scripts in the Processing pipeline

This page provides information on the following:

All instructions provided were performed on a Linux version of DAWN 2.7.x. Although there are no specific alterations to the way the program works between the operating systems, certain parts of DAWN might look different on your computer to the illustrations in this guide.

Introduction

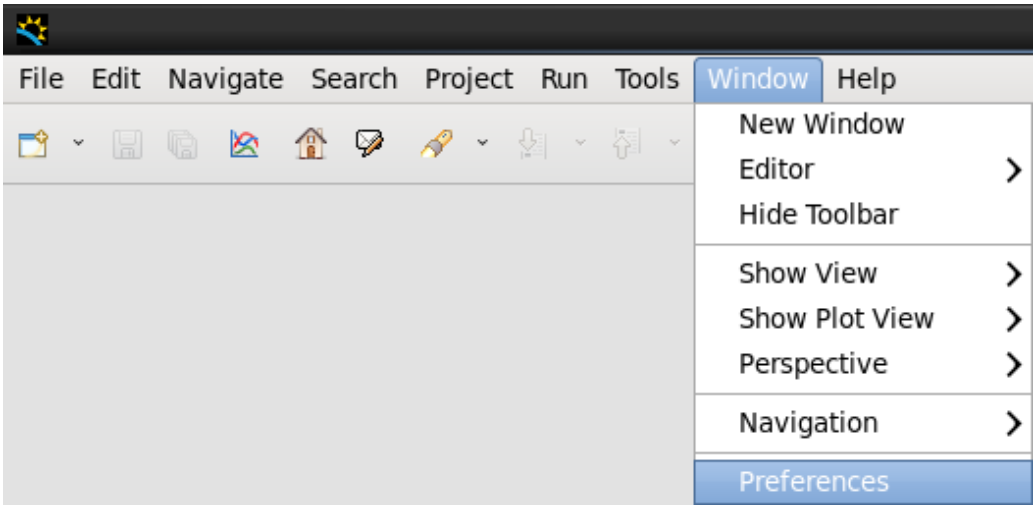
So far this guide has discussed the use of built-in methods, albeit with user defined variables, in order to manipulate obtained data. However, through the use of the *Processing* pipeline and the *Python Script* plug-ins it is also possible to take advantage of the power of the Python programming language from within DAWN itself.

For more of an introduction to the Python programming language many primers and guides can be found here: https://www.python.org/about/gettingstarted/

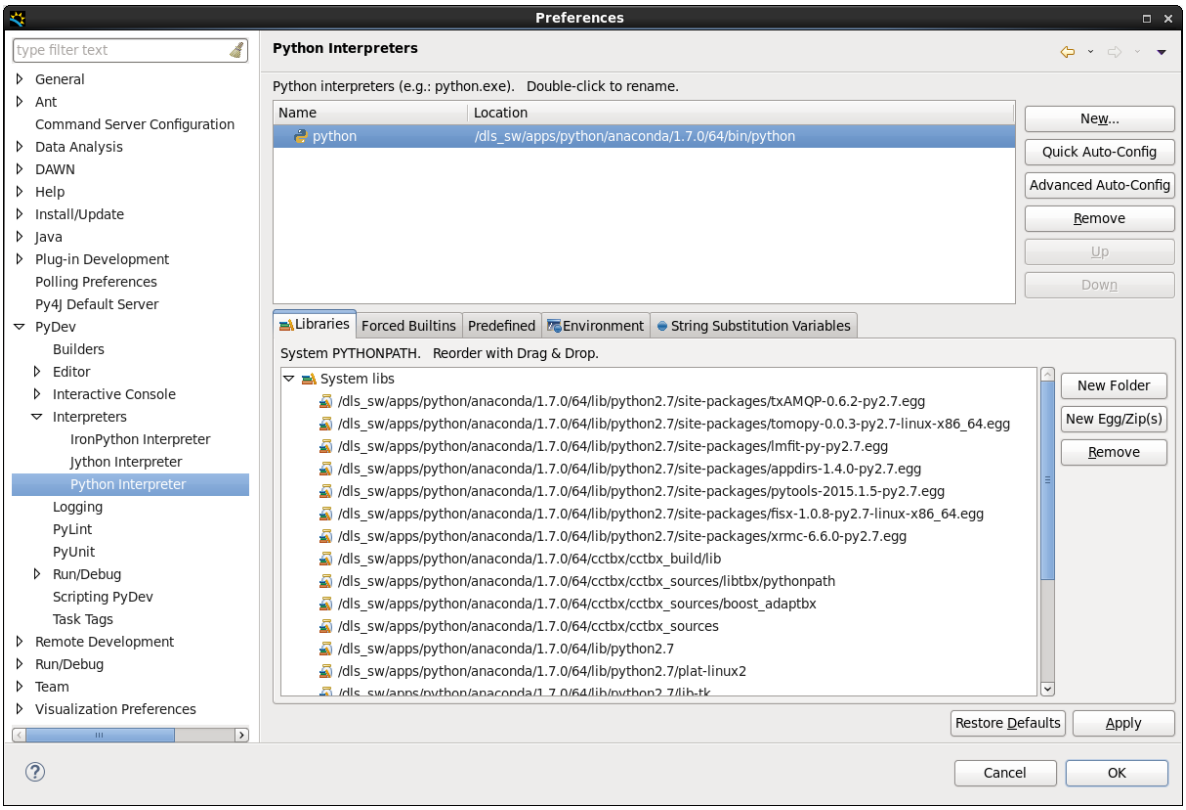
For more of an introduction to Python for use in a scientific setting a guide, containing many further links, can be found here: http://www.scipy-lectures.org/

Configuring Python for use in DAWN

If Python has yet to be used in combination with DAWN it is recommended that it is checked that DAWN is aware of the location of the Python installation on your computer. To do this, first navigate to the DAWN preferences, this can be found at the bottom of the *Window* menu in the main window menubar under *Preferences* as highlighted below



When inside the preferences window navigate to the *Python Interpreter* preferences *via* the sidebar on the left hand side of the window. These preferences can be found under *PyDev > Interpreters > Python Interpreter*, as highlighted below.

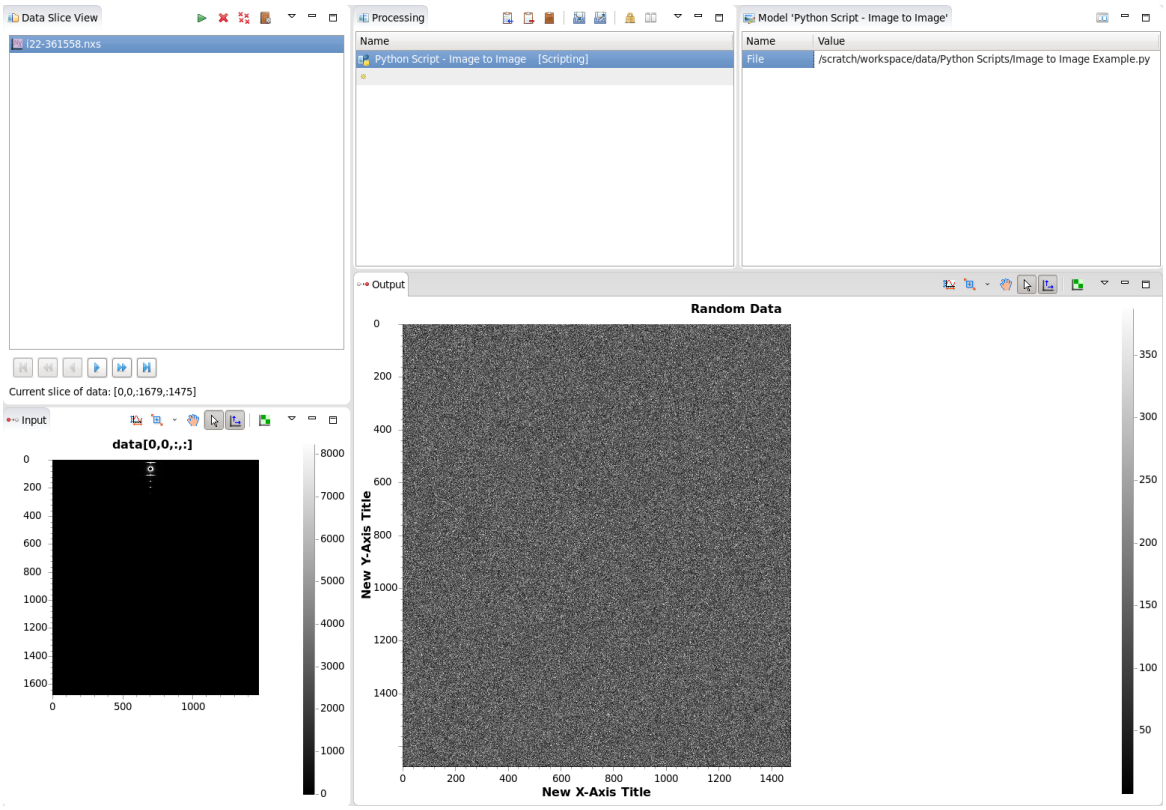


If Python has never been used with DAWN before it is likely that this window is emptier than illustrated above. If a distribution of Python is already installed on the computer clicking *Quick Auto-Config* from the options at the top-right hand side of the window should find and configure it for use with DAWN. However, if a distribution of Python is not currently installed on the computer rather than downloading the default installation from python.org, it is recommended that Anaconda is downloaded instead as this distribution of Python has a large number of scientific packages pre-installed and will make the process of writing and running scripts within DAWN far simpler. Either way after installation, simply click on *Quick Auto-Config*. Depending on the number of packages installed this process can a while but it will only be required to be done once.

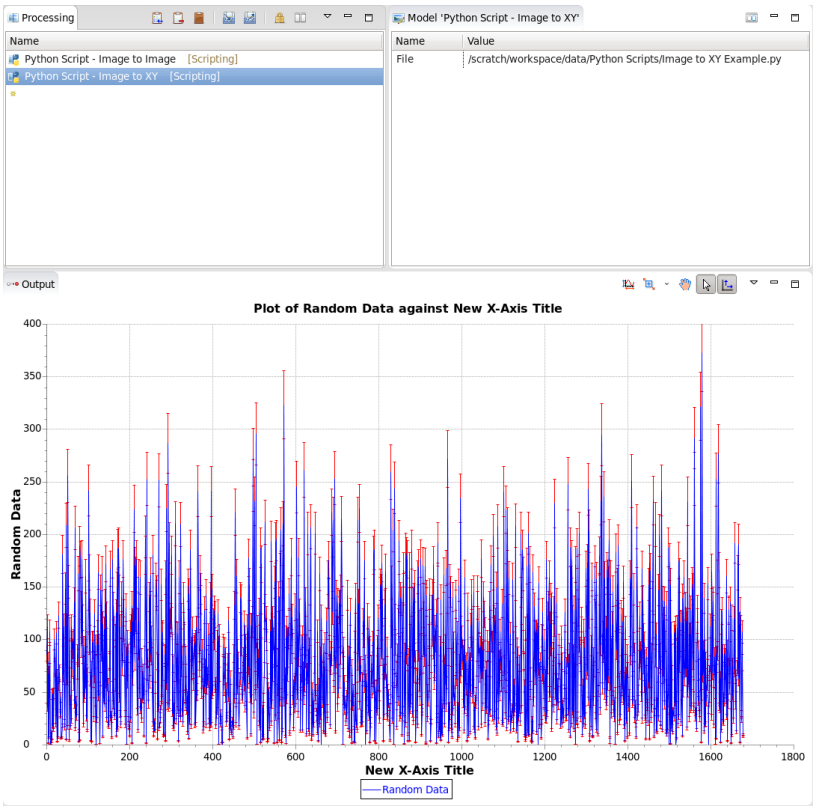
Examples

There are three processing plug-ins to facilitate running Python scripts inside a pipeline these are *Python Script - Image to Image [Scripting]*, for 2D data input and output, *Python Script - Image to XY [Scripting]*, for 2D data input and 1D output and *Python Script - XY to XY [Scripting]*, for 1D data input and output.

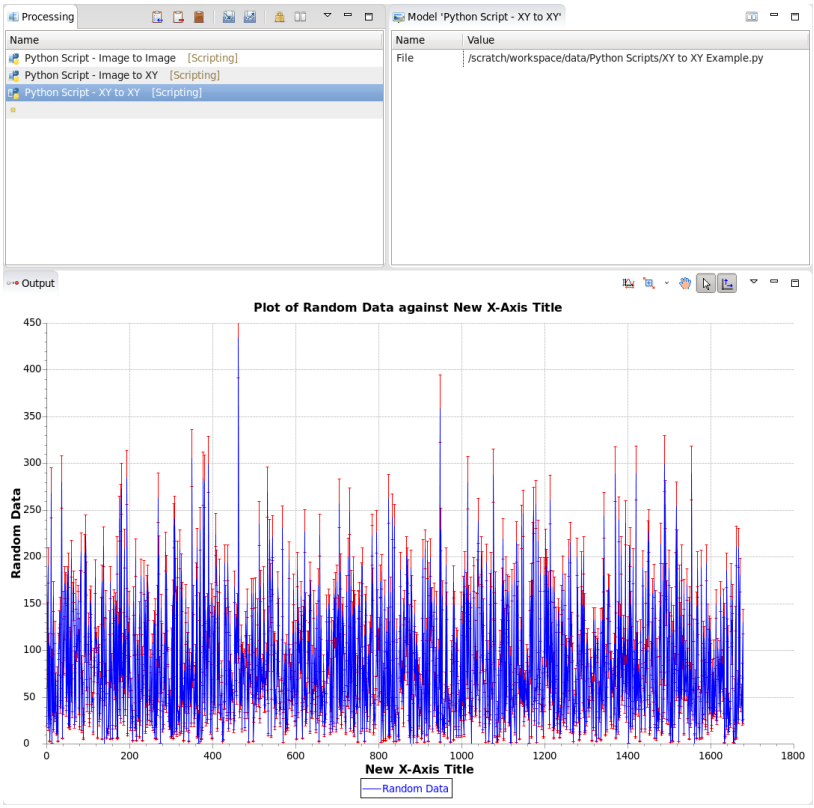
Example scripts delineating the given inputs and required outputs for each of the three *Processing* plug-ins that handle Python I/O can be Python Script Examples, three of the scripts found on that page will be used in this section. This section of the guide will show how these example scripts can be loaded into a *Processing* pipeline within DAWN and used to manipulate either detector images or reduced data as either the only, or an intermediate, step in an analysis pipeline. First, depending on the input and intended output click on the small star icon in the *processing* panel and type in the name of the plug-in desired, this should filer the list sufficiently that the processing plug-in is visible, subsequently double-clicking on the desired plug-in will place it into the pipeline.



In the above example, an image was taken as an input and an image was returned. As this is a processing *pipeline* as well as chaining together Python scripts with other built-in plug-ins it is also possible to chain Python scripts together.



The above example shows an image taken as input and a 1D array of results returned, displayed as a line plot.



Finally, the above example completes the set of Python processing plug-ins, showing a one dimensional input and a one dimensional output. Although these examples are abstract, more functional Python scripts will be implemented in some of the data analysis examples presented in this section to show real-world examples where this functionality can be used.

Previous: Mathematical formulae in the Processing pipeline | **Next:** Guinier analysis

Python Script Examples

This page provides example Python scripts for:

- Processing plug-ins:
  - Python Script - Image to Image [Scripting]
  - Python Script - Image to XY [Scripting]
  - Python Script - XY to XY [Scripting]

Unknown macro: 'widget'

Python Script - Image to Image [Scripting]

1) Take an image (with axes) and return an image (with axes)

```
import numpy as np

def run(xaxis,yaxis, data, **kwargs):
    #generate random noise the same shape as the data
    noise = np.random.rand(*data.shape)
    #add it to the output dictionary
    script_outputs = {"data":noise}
    #also return the x-axis
    script_outputs["xaxis"] = xaxis
        #and the y-axis
    script_outputs["yaxis"] = yaxis

    return script_outputs
```

2) Take an image and return an image of random noise of the same dimensions, overwriting axes, data and titles (click here to download)

```
# Example script for use with the 'Python Script - Image to Image [Scripting]' processing plug-in.
#
# Last updated 2016-11-02 - For DAWN 2.3
#
# Copyright (c) 2016 Diamond Light Source Ltd.
#
#
# The DAWN Python PyDev actor will execute this script, invoking a 'run' method in this file.
# This actor will pass the data and any associated variables from the dataset, which we will
# catch as a dictionary.
#
# The dictionary keys passed for the XY to XY plugin are always:
#
# 'current_slice' = The current slice of the data, typically [0, Frame Number, X Length]
# 'data' = The data, as a 2D array
# 'data_dimensions' = The dimensionality of the data
# 'data_title' = The title of the data
# 'dataset_name' = By default, the NeXus path to the dataset, or a given title of the dataset
# 'file_path' = The path to the file
# 'total' = Number of frames passed
```

```
# 'xaxis' = The x axis values, i.e. the x axis scale
# 'xaxis_title' = The title of the x-axis values/scale
# 'yaxis' = The y axis values, i.e. the y axis scale
# 'yaxis_title' = The title of the y-axis values/scale
#
# with potentially one or more of the following, depending on the dataset passed:
#
# 'auxiliary' = Any auxiliary data associated with the dataset
# 'error' = Any error values for the data, as a 2D array
# 'mask' = A mask, indicating if there are any values not to evaluate, as a 2D boolean array
#
# The PyDev actor will expect, at least, all of the first set of keys to be returned.
# However, it will accept any of the second set of keys being inserted or removed to/from
# the returned dictionary.
#
# An error will be thrown back to the processing perspective GUI if the script does not execute.
#
# Below is a simple example where random data, of the same size to the input is returned
# complete with all the potential additional keys added as well.

# Always handy to have numpy
import numpy


# The method the the PyDev actor will call, we'll catch any and all arguements as a dictionary
def run(**kwargs):
    # Extract out the data
    data = kwargs['data']
    # and find it's length
    x, y = data.shape

    # Generate some random data the same length as the input
    # N.B. The output data length can be different to the input but must be 2D
    randomData = numpy.random.randn(x, y)

    # Generate some error values for the plot
    randomError = numpy.abs(randomData) * 10
    # Generate some data values as well
    randomData = numpy.abs(randomData) * 100

    # Input the error values into the input dictionary
    kwargs['data'] = randomData
    # Input the data values into the input dictionary
    kwargs['error'] = randomError

    # Delineate some axes so that the axis titles plot
    # (No data in the xaxis/yaxis variables means no titles!)
    kwargs['xaxis'] = numpy.arange(0, x)
    kwargs['yaxis'] = numpy.arange(0, y)

    # Give new titles for everything
    kwargs['data_title'] = "Random Data"
    kwargs['xaxis_title'] = "New X-Axis Title"
    kwargs['yaxis_title'] = "New Y-Axis Title"

    # Return the dictionary to DAWN
    return kwargs
```

Python Script - Image to XY [Scripting]

1) Take an image (with axes) and return XY data (with axes) converting data to noise

```
import numpy as np

def run(xaxis,yaxis, data, **kwargs):
    #generate random noise the same shape as the data
    noise = np.random.rand(data.shape[1])
    #add it to the output dictionary
    script_outputs = {"data":noise}
    #also return the x-axis
    script_outputs["xaxis"] = xaxis

    return script_outputs
```

2) Take a 2D image and return 1D XY data of random noise of the same length as the image y-axis, overwriting axes, data and titles (click here to download)

```
# Example script for use with the 'Python Script - Image to XY [Scripting]' processing plug-in.
#
# Last updated 2016-11-02 - For DAWN 2.3
#
# Copyright (c) 2016 Diamond Light Source Ltd.
#
#
# The DAWN Python PyDev actor will execute this script, invoking a 'run' method in this file.
# This actor will pass the data and any associated variables from the dataset, which we will
# catch as a dictionary.
#
# The dictionary keys passed for the XY to XY plugin are always:
#
# 'current_slice' = The current slice of the data, typically [0, Frame Number, X Length]
# 'data' = The data, as a 2D array
# 'data_dimensions' = The dimensionality of the data
# 'data_title' = The title of the data
# 'dataset_name' = By default, the NeXus path to the dataset, or a given title of the dataset
# 'file_path' = The path to the file
# 'total' = Number of frames passed
# 'xaxis' = The x axis values, i.e. the x axis scale
# 'xaxis_title' = The title of the x-axis values/scale
# 'yaxis' = The y axis values, i.e. the y axis scale
# 'yaxis_title' = The title of the y-axis values/scale
#
# with potentially one or more of the following, depending on the dataset passed:
#
# 'auxiliary' = Any auxiliary data associated with the dataset
# 'error' = Any error values for the data, as a 2D array
# 'mask' = A mask, indicating if there are any values not to evaluate, as a 2D boolean array
#
# The PyDev actor will expect, at least, all of the first set of keys to be returned.
# However, it will accept any of the second set of keys being inserted or removed to/from
# the returned dictionary.
#
# For scripts made for this plug-in the returned data should be reduced from a 2D to a 1D array,
# furthermore removing the yaxis title, whilst not essential, will not result in an error
#
# An error will be thrown back to the processing perspective GUI if the script does not execute.
#
# Below is a simple example where random data, of the same size to the input is returned
# complete with all the potential additional keys added as well.

# Always handy to have numpy
import numpy


# The method the the PyDev actor will call, we'll catch any and all arguements as a dictionary
```

```
def run(**kwargs):
    # Extract out the data
    data = kwargs['data']
    # and find it's length
    x, y = data.shape

    # Generate some random data the same length as the input
    # N.B. The output data length can be different to the input but must be 1D
    randomData = numpy.random.randn(x)

    # Generate some error values for the plot
    randomError = numpy.abs(randomData) * 10
    # Generate some data values as well
    randomData = numpy.abs(randomData) * 100

    # Input the error values into the input dictionary
    kwargs['data'] = randomData
    # Input the data values into the input dictionary
    kwargs['error'] = randomError

    # Delineate some axes so that the axis titles plot
    # (No data in the xaxis/yaxis variables means no titles!)
    kwargs['xaxis'] = numpy.arange(0, x)

    # Give new titles for everything
    kwargs['data_title'] = "Random Data"
    kwargs['xaxis_title'] = "New X-Axis Title"

    # Return the dictionary to DAWN
    return kwargs
```

Python Script - XY to XY [Scripting]

Quick links:

- XY Noise
- XY Noise - Annotated
- Guinier Plot
- Kratky Plot
- Porod Plot

1) Take XY data, and return XY data converting data to noise

```
import numpy as np

def run(xaxis, data, **kwargs):
        #generate random noise the same shape as the data
        noise = np.random.rand(*data.shape)
        #add it to the output dictionary
        script_outputs = {"data":noise}
        #also return the x-axis
        script_outputs["xaxis"] = xaxis
        return script_outputs
```

2) Take XY data and return XY data of random noise of the same length, overwriting axes, data and titles (click here to download)

```
# Example script for use with the 'Python Script - XY to XY [Scripting]' processing plug-in.
#
# Last updated 2016-11-02 - For DAWN 2.3
#
# Copyright (c) 2016 Diamond Light Source Ltd.
#
#
# The DAWN Python PyDev actor will execute this script, invoking a 'run' method in this file.
# This actor will pass the data and any associated variables from the dataset, which we will
# catch as a dictionary.
#
# The dictionary keys passed for the XY to XY plugin are always:
#
# 'current_slice' = The current slice of the data, typically [0, Frame Number, X Length]
# 'data' = The data, as a 1D array
# 'data_dimensions' = The dimensionality of the data
# 'data_title' = The title of the data
# 'dataset_name' = By default, the NeXus path to the dataset, or a given title of the dataset
# 'file_path' = The path to the file
# 'total' = Number of frames passed
# 'xaxis' = The x axis values, i.e. the x axis scale
# 'xaxis_title' = The title of the y-axis values/scale
#
# with potentially one or more of the following, depending on the dataset passed:
#
# 'auxiliary' = Any auxiliary data associated with the dataset
# 'error' = Any error values for the data, as a 1D array
# 'mask' = A mask, indicating if there are any values not to evaluate, as a 1D boolean array
# 'yaxis' = The y axis values, i.e. the y axis scale
#
# The PyDev actor will expect, at least, all of the first set of keys to be returned.
# However, it will accept any of the second set of keys being inserted or removed to/from
# the returned dictionary.
#
# An error will be thrown back to the processing perspective GUI if the script does not execute.
#
# Below is a simple example where random data, of the same size to the input is returned
# complete with all the potential additional keys added as well.

# Always handy to have numpy
import numpy


# The method the the PyDev actor will call, we'll catch any and all arguements as a dictionary
def run(**kwargs):

    # Extract out the data
    data = kwargs['data']
    # and find it's length
    x, = data.shape

    # Generate some random data the same length as the input
    # N.B. The output data length can be different to the input but must be 1D
    randomData = numpy.random.randn(x)

    # Generate some error values for the plot
    randomError = numpy.abs(randomData) * 10
    # Generate some data values as well
    randomData = numpy.abs(randomData) * 100

    # Input the error values into the input dictionary
    kwargs['data'] = randomData
    # Input the data values into the input dictionary
    kwargs['error'] = randomError

    # Delineate some axes so that the axis titles plot
```

```
    # (No data in the xaxis/yaxis variables means no titles!)
    kwargs['xaxis'] = numpy.arange(0, x)

    # Give new titles for everything
    kwargs['data_title'] = "Random Data"
    kwargs['xaxis_title'] = "New X-Axis Title"

    # Return the dictionary to DAWN
    return kwargs
```

3) Take small angle scattering I vs q data and transform this to produce a Guinier plot (click here to download)

```
# Example script for use with the 'Python Script - XY to XY [Scripting]' processing plug-in.
#
# Last updated 2016-11-02 - For DAWN 2.3
#
# Copyright (c) 2016 Diamond Light Source Ltd.
#
#
# The DAWN Python PyDev actor will execute this script, invoking a 'run' method in this file.
# This actor will pass the data and any associated variables from the dataset, which we will
# catch as a dictionary.
#
# Below is a simple example where reduced SAXS data is taken in and converted to display a Guinier plot

# Always handy to have numpy
import numpy as np


# The method the the PyDev actor will call, we'll catch any and all arguements as a dictionary
def run(**kwargs):

    # Extract out the data and xaxis from the dictionary
    data = kwargs['data']
    xaxis = kwargs['xaxis']

    # Do some 'error' handling
    if 'error' in kwargs:
        del kwargs['error']

    # Do the required mathematics on the data
    guinier = np.log(data)
    guinier_x = np.power(xaxis,2)

    # Set the plot titles and data values
    kwargs['data_title'] = 'ln(I)'
    kwargs['data'] = guinier
    kwargs['xaxis_title'] = 'q^2'
    kwargs['xaxis'] = guinier_x

    return kwargs
```

4) Take small angle scattering I vs q data and transform this to produce a Kratky plot (click here to download)

```
# Example script for use with the 'Python Script - XY to XY [Scripting]' processing plug-in.
#
# Last updated 2016-11-02 - For DAWN 2.3
#
# Copyright (c) 2016 Diamond Light Source Ltd.
#
#
# The DAWN Python PyDev actor will execute this script, invoking a 'run' method in this file.
# This actor will pass the data and any associated variables from the dataset, which we will
# catch as a dictionary.
#
# Below is a simple example where reduced SAXS data is taken in and converted to display a Kratky plot

# Always handy to have numpy
import numpy as np


# The method the the PyDev actor will call, we'll catch any and all arguements as a dictionary
def run(**kwargs):

 # Extract out the data and xaxis from the dictionary
 data = kwargs['data']
 xaxis = kwargs['xaxis']

 # Do some 'error' handling
 if 'error' in kwargs:
 del kwargs['error']

 # Do the required mathematics on the data
 kratky = np.power(xaxis,2)*data

 # Set the plot titles and data values
 kwargs['data_title'] = 'I*q^2'
 kwargs['data'] = kratky
 kwargs['xaxis_title'] = 'q'

 return kwargs
```

5) Take small angle scattering I vs q data and transform this to produce a Porod plot (click here to download)

```
 # Example script for use with the 'Python Script - XY to XY [Scripting]' processing plug-in.
#
# Last updated 2016-11-02 - For DAWN 2.3
#
# Copyright (c) 2016 Diamond Light Source Ltd.
#
#
# The DAWN Python PyDev actor will execute this script, invoking a 'run' method in this file.
# This actor will pass the data and any associated variables from the dataset, which we will
# catch as a dictionary.
#
# Below is a simple example where reduced SAXS data is taken in and converted to display a Porod plot

# Always handy to have numpy
import numpy as np


# The method the the PyDev actor will call, we'll catch any and all arguements as a dictionary
def run(**kwargs):

    # Extract out the data and xaxis from the dictionary
    data = kwargs['data']
    xaxis = kwargs['xaxis']

    # Do some 'error' handling
    if 'error' in kwargs:
        del kwargs['error']

    # Do the required mathematics on the data
    porod = np.power(xaxis, 4) * data
    porod_x = np.power(xaxis, 4)

    # Set the plot titles and data values
    kwargs['data_title'] = 'I*q^4'
    kwargs['data'] = porod
    kwargs['xaxis_title'] = 'q^4'
    kwargs['xaxis'] = porod_x

    # Return the data
    return kwargs
```

Guinier analysis

To accompany this section of the guide, please download the workspace that can be found Data Analysis Examples Workspace.zip. This workspace contains example data, all the pre-requisite pipelines and Python scripts to follow this section of the guide and it will be used in the illustrations for this section as well. The Python script used in this example is also listed Python Script Examples as well.

When converting a standard, reduced, I vs q plot to a Guinier plot the following transformations are applied:

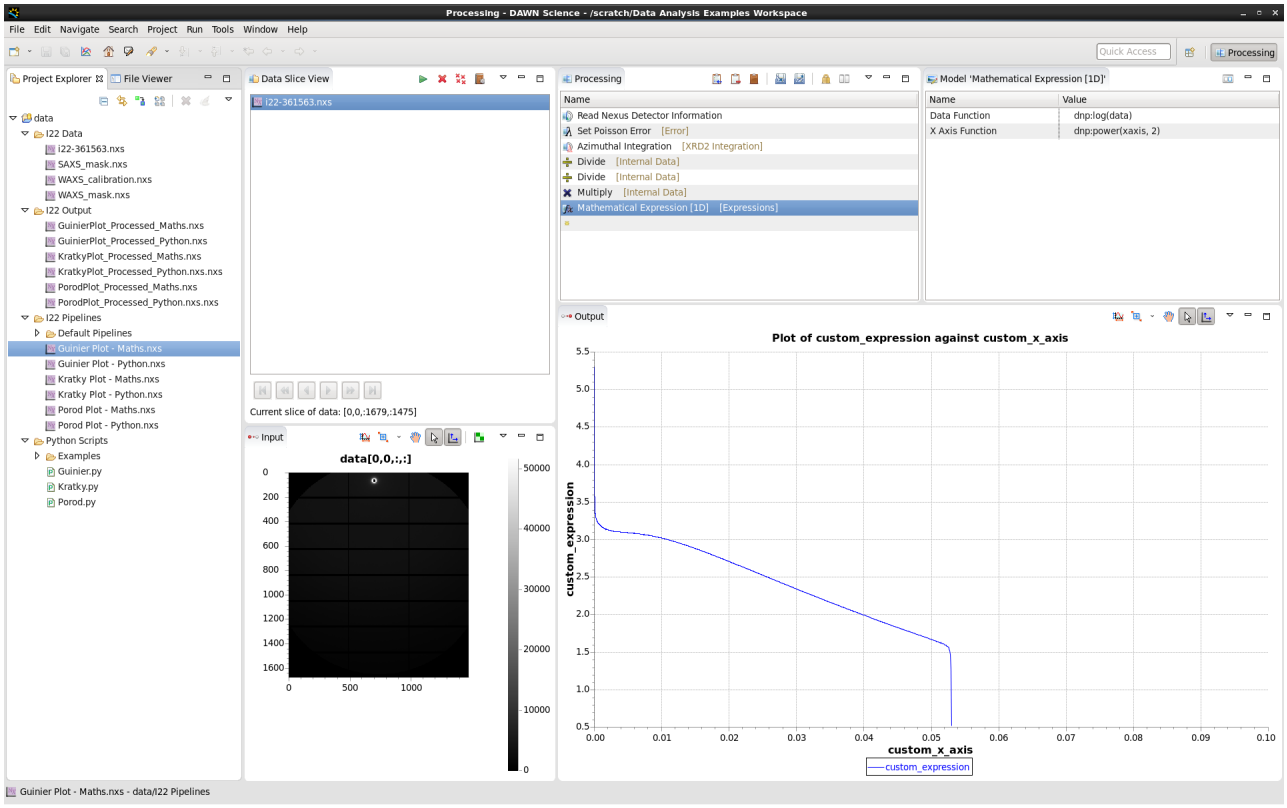
- $I_g = \log(I)$
- $q_g = q^2$

Using DAWN, these relatively straightforward transformations can be applied in the *Processing* perspective either using the in-built *Mathematical Expression [1D]* plug-in or using the *Python Script - XY to XY [Scripting]* plug-in. Naturally, as the complexity of the operation desired increases the limitations of the mathematical expression plug-in will quickly become apparent, however, for this example either approach can be taken. The following links will skip to the appropriate section of this page:

- Mathematical expression approach
- Python script approach

Mathematical expression approach

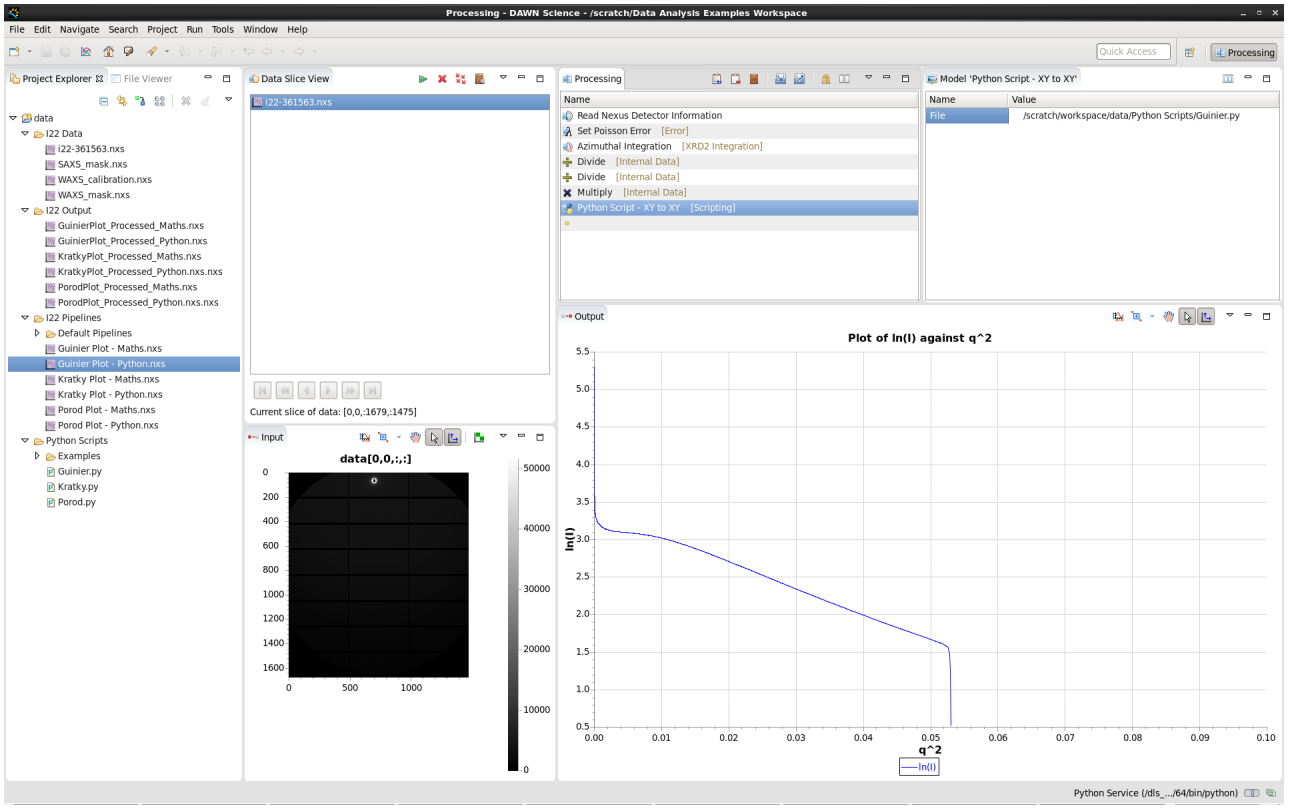
If you have downloaded the workspace above, after loading the workspace and expanding all the collapsible folders in the *Project Explorer* panel it should be relatively straightforward to dragging-and-dropping the scan file *i22-361563.nxs* into the *Data Slice View*, selecting to view the SAXS detector image, and dragging-and-dropping the pipeline entitled *Guinier Plot - Maths.nxs* into the *Processing* panel. After performing these steps a view, similar to the one presented below, should be on screen. At this point a plot in the *Output* panel is presented that shows the Guinier modification to the reduced data.



It is hoped that you have read the Data Reduction part of this guide first, assuming this it should become apparent that the only operation that has been applied, after reducing the diffraction image, is that a *Mathematical Expression [1D] [Expressions]* plug-in has been placed in the pipeline after the data reduction plug-ins. Looking in the plug-in's *Model* panel reveals that the transformations described in the introduction have been applied to the *Data Functio* and *X Axis Function* as appropriate. Clicking the *Process all files* button in the *Data Slice View* at this point will output files containing Guinier plotting data. As with the data reduction, adding the *Export to Text File* plug-in at the end of this pipeline will also export these results as tab-delimited text.

Python script approach

If you have downloaded the workspace above, after loading the workspace and expanding all the collapsible folders in the *Project Explorer* panel it should be relatively straightforward to dragging-and-dropping the scan file *i22-361563.nxs* into the *Data Slice View*, selecting to view the SAXS detector image, and dragging-and-dropping the pipeline entitled *Guinier Plot - Python.nxs* into the *Processing* panel. After performing these steps a view, similar to the one presented below, should be on screen. At this point a plot in the *Output* panel is presented that shows the Guinier modification to the reduced data. It is possible that DAWN will not find the Python script, as the path has changed, in order to rectify this drag-and-drop the *Guinier.py* file from the *Project Explorer* panel into the *Value* field in the *Name* entry of the plug-in's *Model* panel.



It is hoped that you have read the Data Reduction part of this guide first, assuming this it should become apparent that the only operation that has been applied, after reducing the diffraction image, is that a *Python Script - XY to XY [Scripting]* plug-in has been placed in the pipeline after the data reduction plug-ins. Looking in the plug-in's *Model* panel reveals the script used to alter the data, opening this file in a text editor will reveal the manipulations performed. Clicking the *Process all files* button in the *Data Slice View* at this point will output files containing Guinier plotting data. As with the data reduction, adding the *Export to Text File* plug-in at the end of this pipeline will also export these results as tab-delimited text.

Previous: Python scripts in the Processing pipeline | **Next:** Kratky analysis

Kratky analysis

To accompany this section of the guide, please download the workspace that can be found Data Analysis Examples Workspace.zip. This workspace contains example data, all the pre-requisite pipelines and Python scripts to follow this section of the guide and it will be used in the illustrations for this section as well. The Python script used in this example is also listed Python Script Examples as well.

When converting a standard, reduced, I vs q plot to a Kratky plot the plot intensity is transformed thusly:

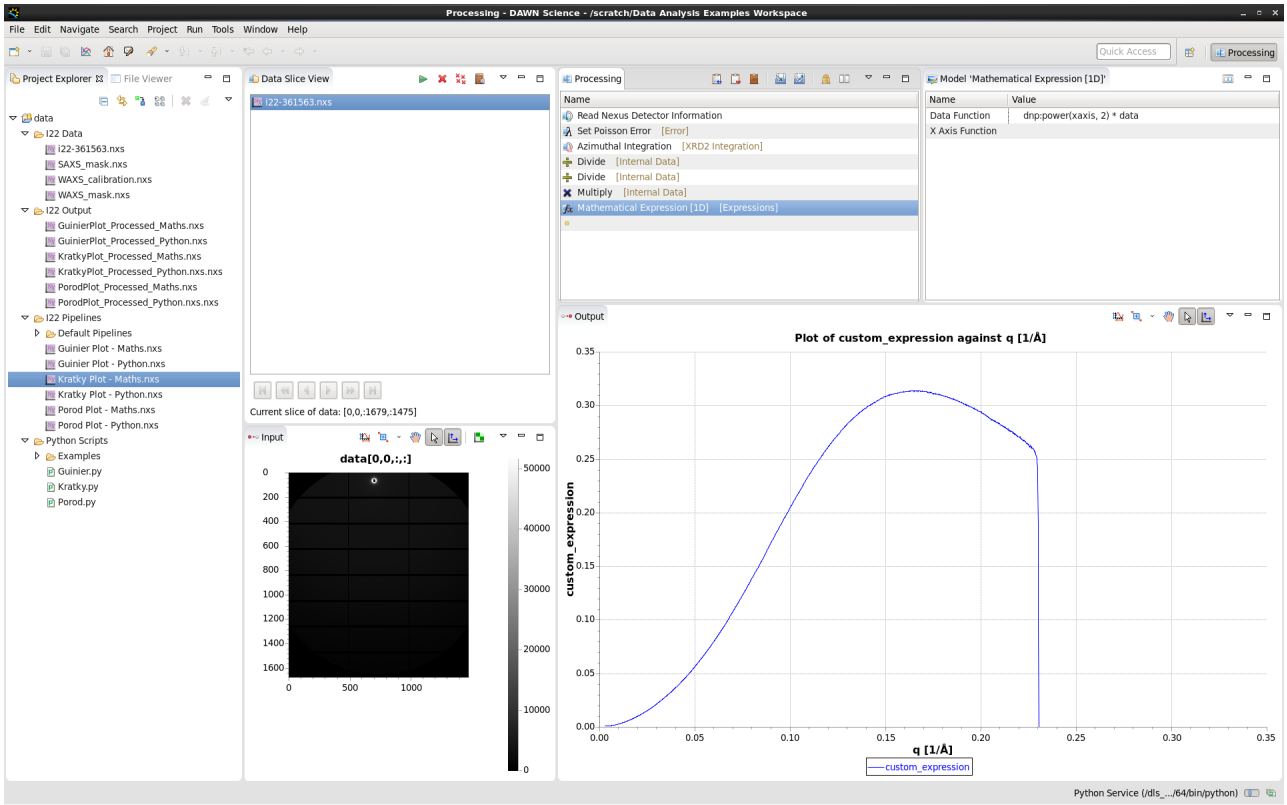
- $I_k = I \times q^2$

Using DAWN, these relatively straightforward transformations can be applied in the *Processing* perspective either using the in-built *Mathematical Expression [1D]* plug-in or using the *Python Script - XY to XY [Scripting]* plug-in. Naturally, as the complexity of the operation desired increases the limitations of the mathematical expression plug-in will quickly become apparent, however, for this example either approach can be taken. The following links will skip to the appropriate section of this page:

- Mathematical expression approach
- Python script approach

Mathematical expression approach

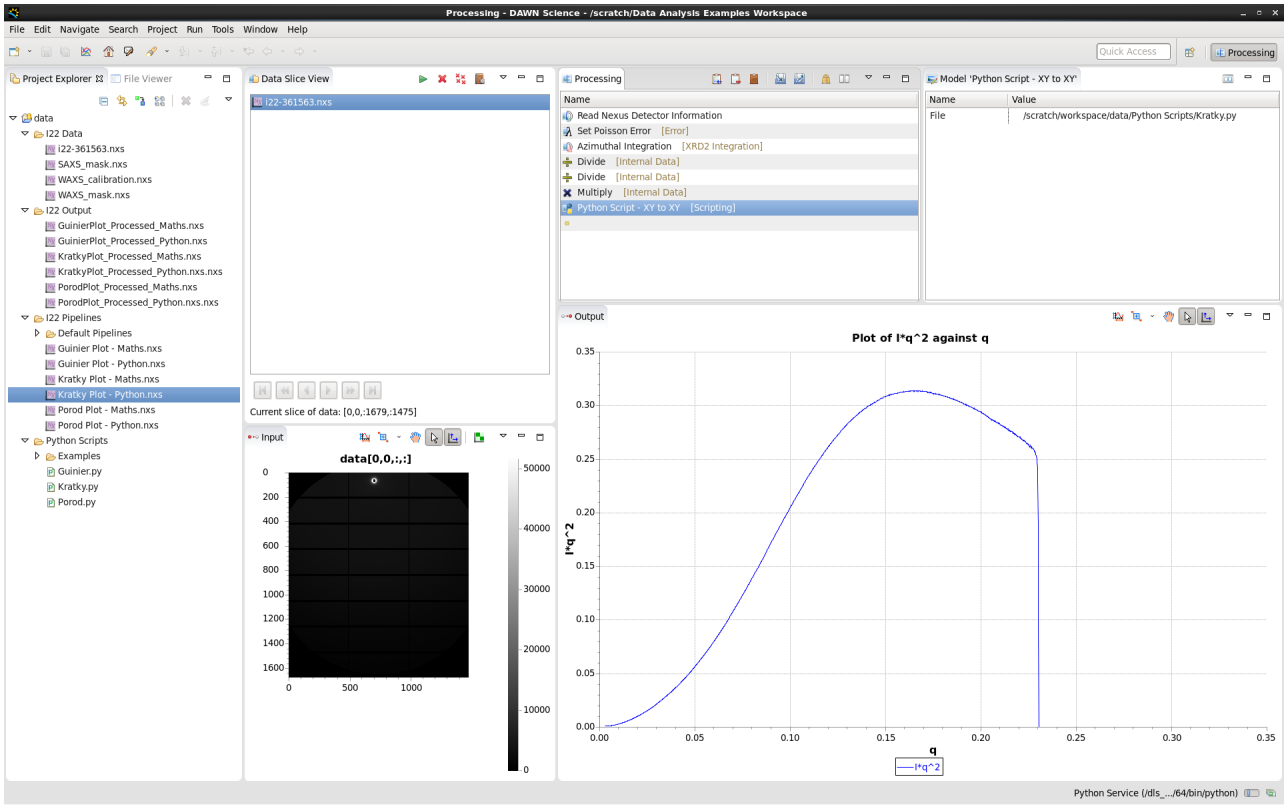
If you have downloaded the workspace above, after loading the workspace and expanding all the collapsible folders in the *Project Explorer* panel it should be relatively straightforward to dragging-and-dropping the scan file *i22-361563.nxs* into the *Data Slice View*, selecting to view the SAXS detector image, and dragging-and-dropping the pipeline entitled *Kratky Plot - Maths.nxs* into the *Processing* panel. After performing these steps a view, similar to the one presented below, should be on screen. At this point a plot in the *Output* panel is presented that shows the Kratky modification to the reduced data.



It is hoped that you have read the Data Reduction part of this guide first, assuming this it should become apparent that the only operation that has been applied, after reducing the diffraction image, is that a *Mathematical Expression [1D] [Expressions]* plug-in has been placed in the pipeline after the data reduction plug-ins. Looking in the plug-in's *Model* panel reveals that the transformations described in the introduction have been applied to the *Data Functio* and *X Axis Function* as appropriate. Clicking the *Process all files* button in the *Data Slice View* at this point will output files containing Kratky plotting data. As with the data reduction, adding the *Export to Text File* plug-in at the end of this pipeline will also export these results as tab-delimited text.

Python script approach

If you have downloaded the workspace above, after loading the workspace and expanding all the collapsible folders in the *Project Explorer* panel it should be relatively straightforward to dragging-and-dropping the scan file *i22-361563.nxs* into the *Data Slice View*, selecting to view the SAXS detector image, and dragging-and-dropping the pipeline entitled *Kratky Plot - Python.nxs* into the *Processing* panel. After performing these steps a view, similar to the one presented below, should be on screen. At this point a plot in the *Output* panel is presented that shows the Kratky modification to the reduced data. It is possible that DAWN will not find the Python script, as the path has changed, in order to rectify this drag-and-drop the *Kratky.py* file from the *Project Explorer* panel into the *Value* field in the *Name* entry of the plug-in's *Model* panel.



It is hoped that you have read the Data Reduction part of this guide first, assuming this it should become apparent that the only operation that has been applied, after reducing the diffraction image, is that a *Python Script - XY to XY [Scripting]* plug-in has been placed in the pipeline after the data reduction plug-ins. Looking in the plug-in's *Model* panel reveals the script used to alter the data, opening this file in a text editor will reveal the manipulations performed. Clicking the *Process all files* button in the *Data Slice View* at this point will output files containing Kratky plotting data. As with the data reduction, adding the *Export to Text File* plug-in at the end of this pipeline will also export these results as tab-delimited text.

Previous: Guinier analysis | **Next:** Porod analysis

Porod analysis

To accompany this section of the guide, please download the workspace that can be found Data Analysis Examples Workspace.zip. This workspace contains example data, all the pre-requisite pipelines and Python scripts to follow this section of the guide and it will be used in the illustrations for this section as well. The Python script used in this example is also listed Python Script Examples as well.

When converting a standard, reduced, I vs q plot to a Porod plot the following transformations are applied:

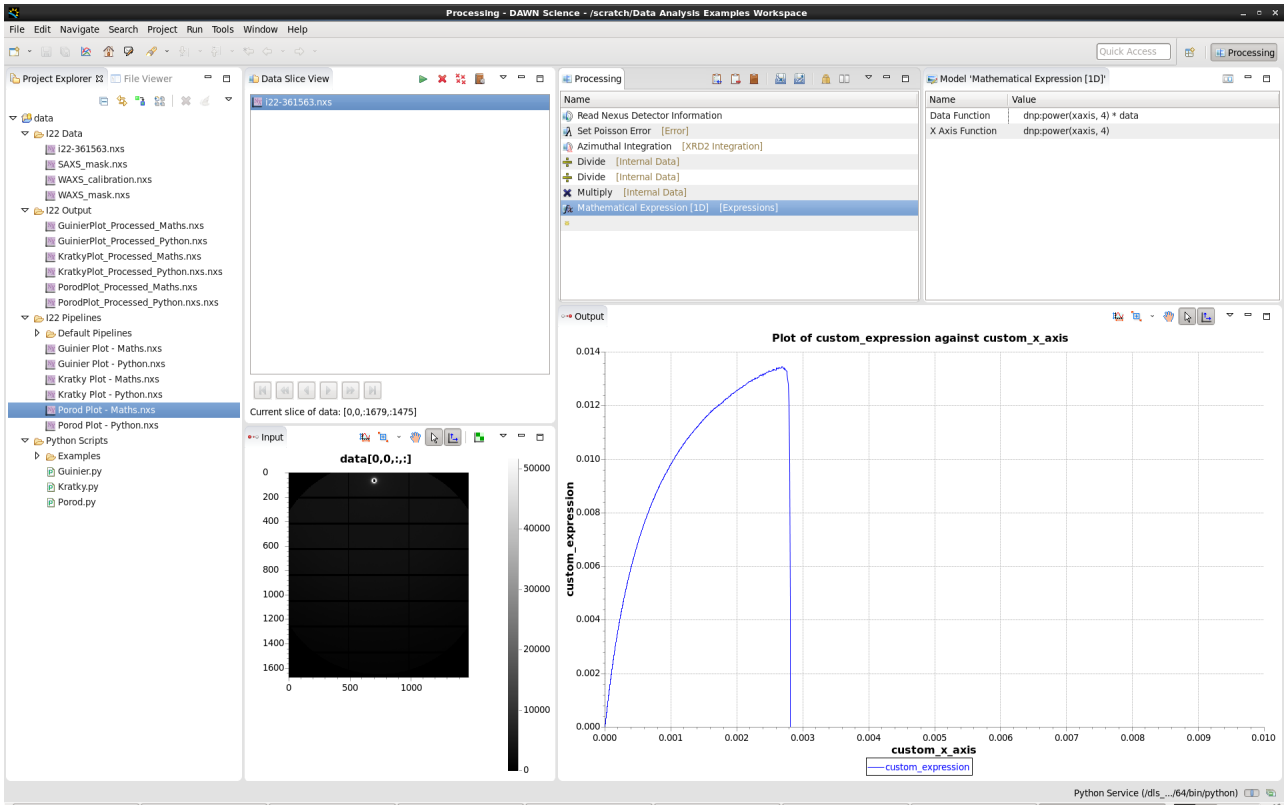
- $I_p = I \times q^4$
- $q_p = q^4$

Using DAWN, these relatively straightforward transformations can be applied in the *Processing* perspective either using the in-built *Mathematical Expression [1D]* plug-in or using the *Python Script - XY to XY [Scripting]* plug-in. Naturally, as the complexity of the operation desired increases the limitations of the mathematical expression plug-in will quickly become apparent, however, for this example either approach can be taken. The following links will skip to the appropriate section of this page:

- Mathematical expression approach
- Python script approach

Mathematical expression approach

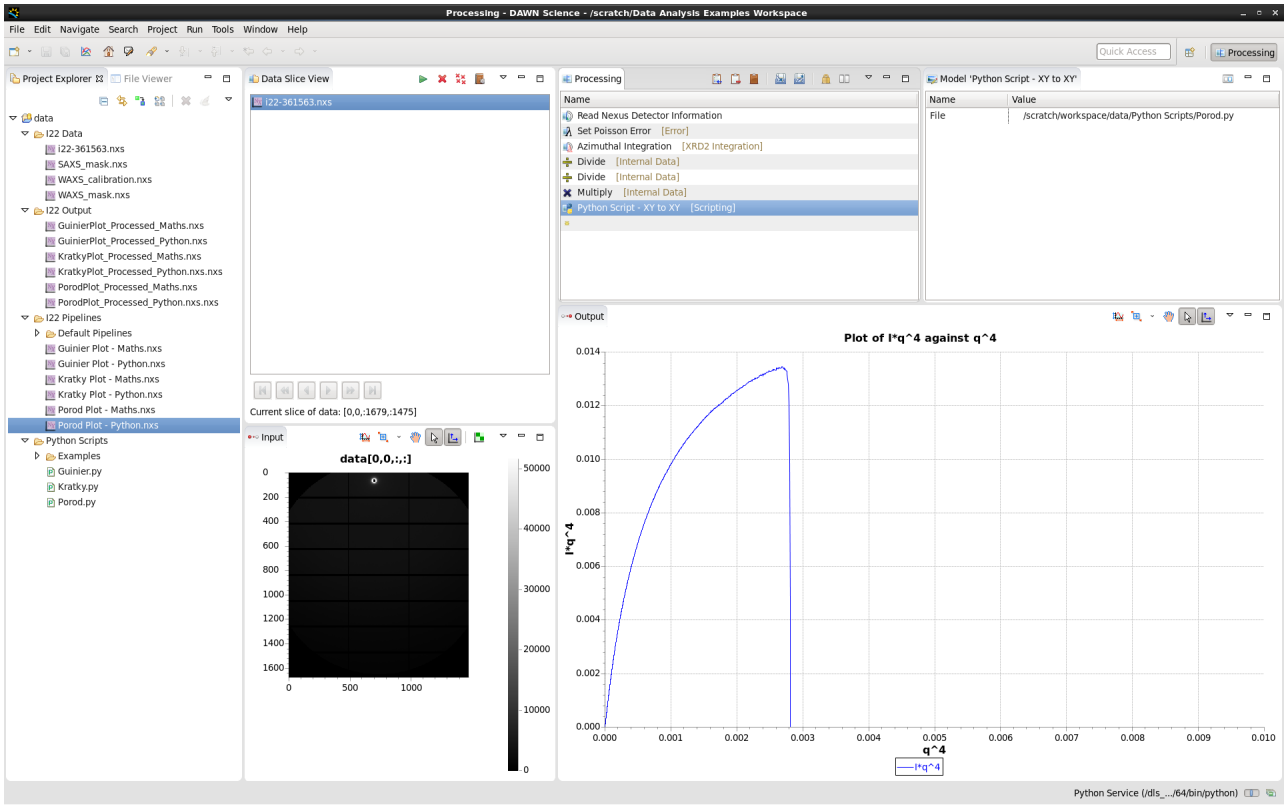
If you have downloaded the workspace above, after loading the workspace and expanding all the collapsible folders in the *Project Explorer* panel it should be relatively straightforward to dragging-and-dropping the scan file *i22-361563.nxs* into the *Data Slice View*, selecting to view the SAXS detector image, and dragging-and-dropping the pipeline entitled *Porod Plot - Maths.nxs* into the *Processing* panel. After performing these steps a view, similar to the one presented below, should be on screen. At this point a plot in the *Output* panel is presented that shows the Porod modification to the reduced data.



It is hoped that you have read the Data Reduction part of this guide first, assuming this it should become apparent that the only operation that has been applied, after reducing the diffraction image, is that a *Mathematical Expression [1D] [Expressions]* plug-in has been placed in the pipeline after the data reduction plug-ins. Looking in the plug-in's *Model* panel reveals that the transformations described in the introduction have been applied to the *Data Functio* and *X Axis Function* as appropriate. Clicking the *Process all files* button in the *Data Slice View* at this point will output files containing Porod plotting data. As with the data reduction, adding the *Export to Text File* plug-in at the end of this pipeline will also export these results as tab-delimited text.

Python script approach

If you have downloaded the workspace above, after loading the workspace and expanding all the collapsible folders in the *Project Explorer* panel it should be relatively straightforward to dragging-and-dropping the scan file *i22-361563.nxs* into the *Data Slice View*, selecting to view the SAXS detector image, and dragging-and-dropping the pipeline entitled *Porod Plot - Python.nxs* into the *Processing* panel. After performing these steps a view, similar to the one presented below, should be on screen. At this point a plot in the *Output* panel is presented that shows the Porod modification to the reduced data. It is possible that DAWN will not find the Python script, as the path has changed, in order to rectify this drag-and-drop the *Porod.py* file from the *Project Explorer* panel into the *Value* field in the *Name* entry of the plug-in's *Model* panel.

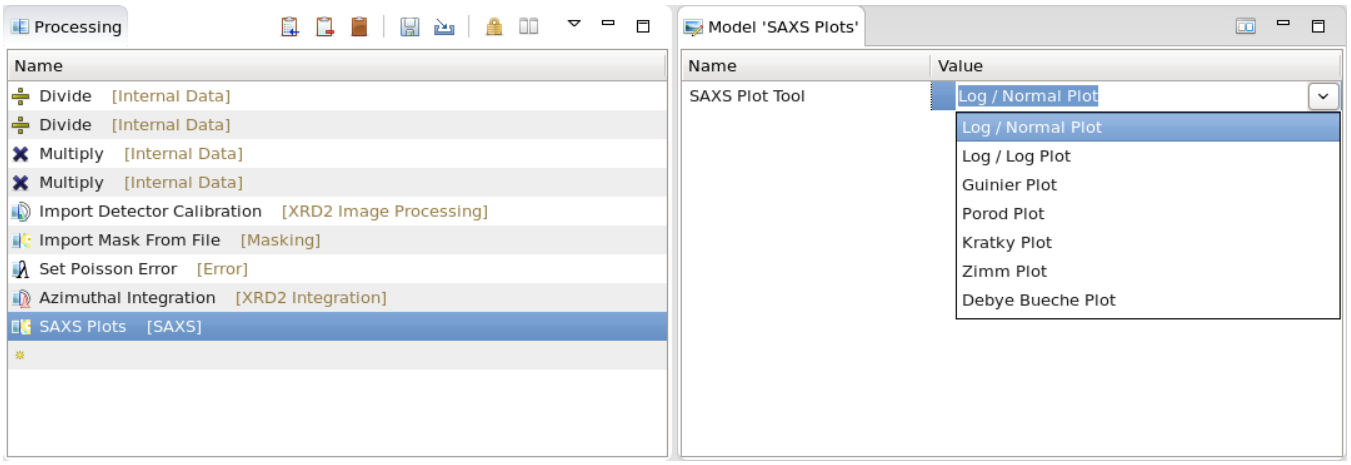


It is hoped that you have read the Data Reduction part of this guide first, assuming this it should become apparent that the only operation that has been applied, after reducing the diffraction image, is that a *Python Script - XY to XY [Scripting]* plug-in has been placed in the pipeline after the data reduction plug-ins. Looking in the plug-in's *Model* panel reveals the script used to alter the data, opening this file in a text editor will reveal the manipulations performed. Clicking the *Process all files* button in the *Data Slice View* at this point will output files containing Porod plotting data. As with the data reduction, adding the *Export to Text File* plug-in at the end of this pipeline will also export these results as tab-delimited text.

Previous: Kratky analysis | **Next:** SAXS Tools plug-in

SAXS Tools plug-in

Within the processing plug-ins distributed by DAWN are a number of simple operations that can be applied to reduced 1D data. These modify either the output plot, e.g. log-log plot, or perform simple operations on the data to display commonly used plot formats. Adding the *SAXS Plots [SAXS]* plug-in to a processing pipeline will allow the user to select from a number of different plot types by selecting from the list provided from the in the *Model* panel.



N.B. It is worth nothing that although this will alter the plot view and provide data for further steps in the processing pipeline, having this plug-in as the last step in the processing pipeline and clicking *Process all files* from the Data Slice View panel will only give the data in the current form requested *not* as the original I vs. q, or other, form.

Previous: Porod analysis | **Next:** Crystallite parameters

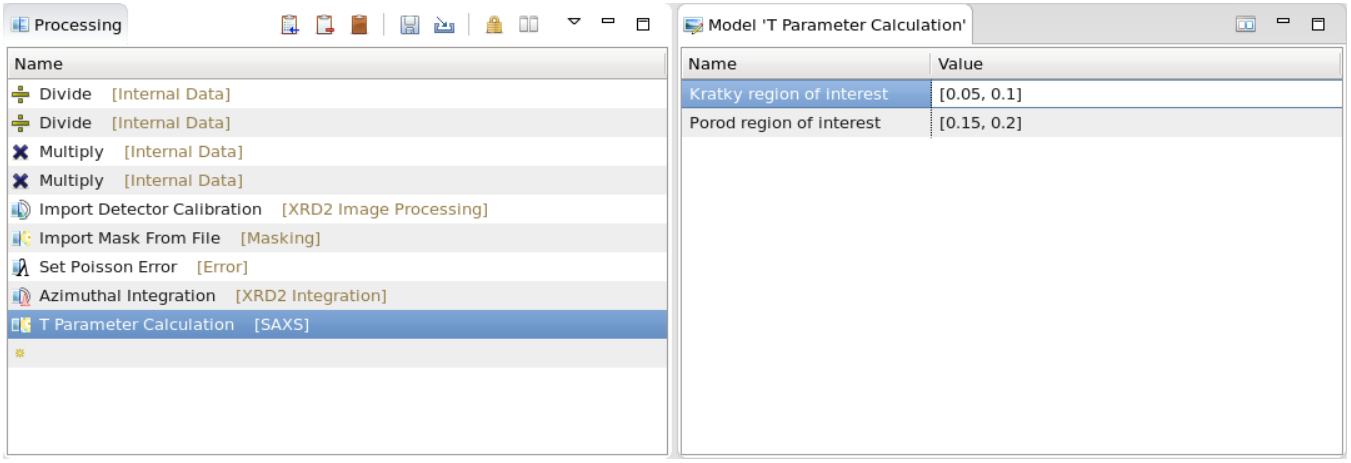
Crystallite parameters

To accompany this section of the guide, please download the workspace that can be found Data Analysis Examples Workspace.zip. This workspace contains example data, all the pre-requisite pipelines and Python scripts to follow this section of the guide and it will be used in the illustrations for this section as well.

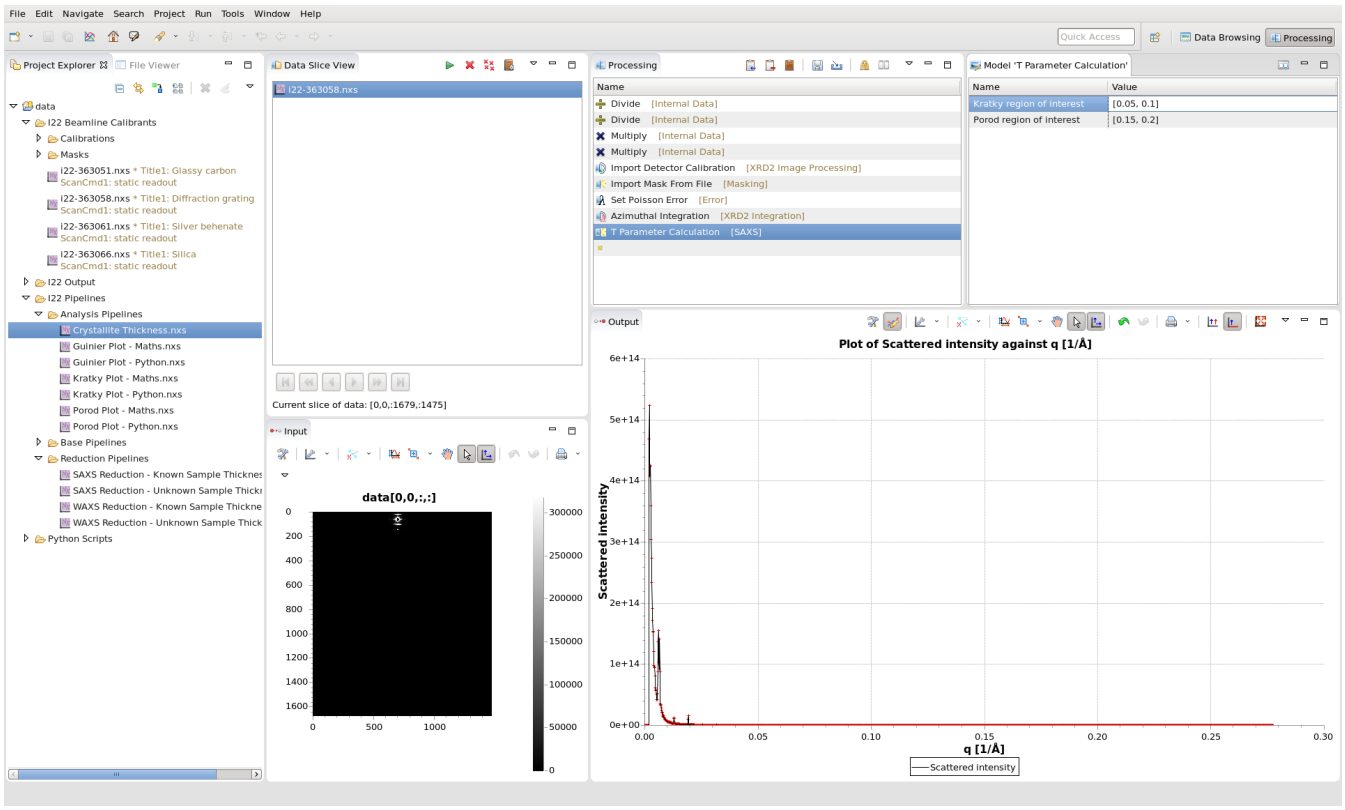
This is an example of a more complex processing pipeline utilising several different plug-ins together, each with differing inputs required, in order to reduce an obtained image to a single value.

The methodology and calculations for this plug-in been based on the published work by: P. Fratzl, S. Schreiber, and K Klaushofer, *Connective Tissue Research*, 1996, **34**, 247-54 DOI: 10.3109/03008209609005268.

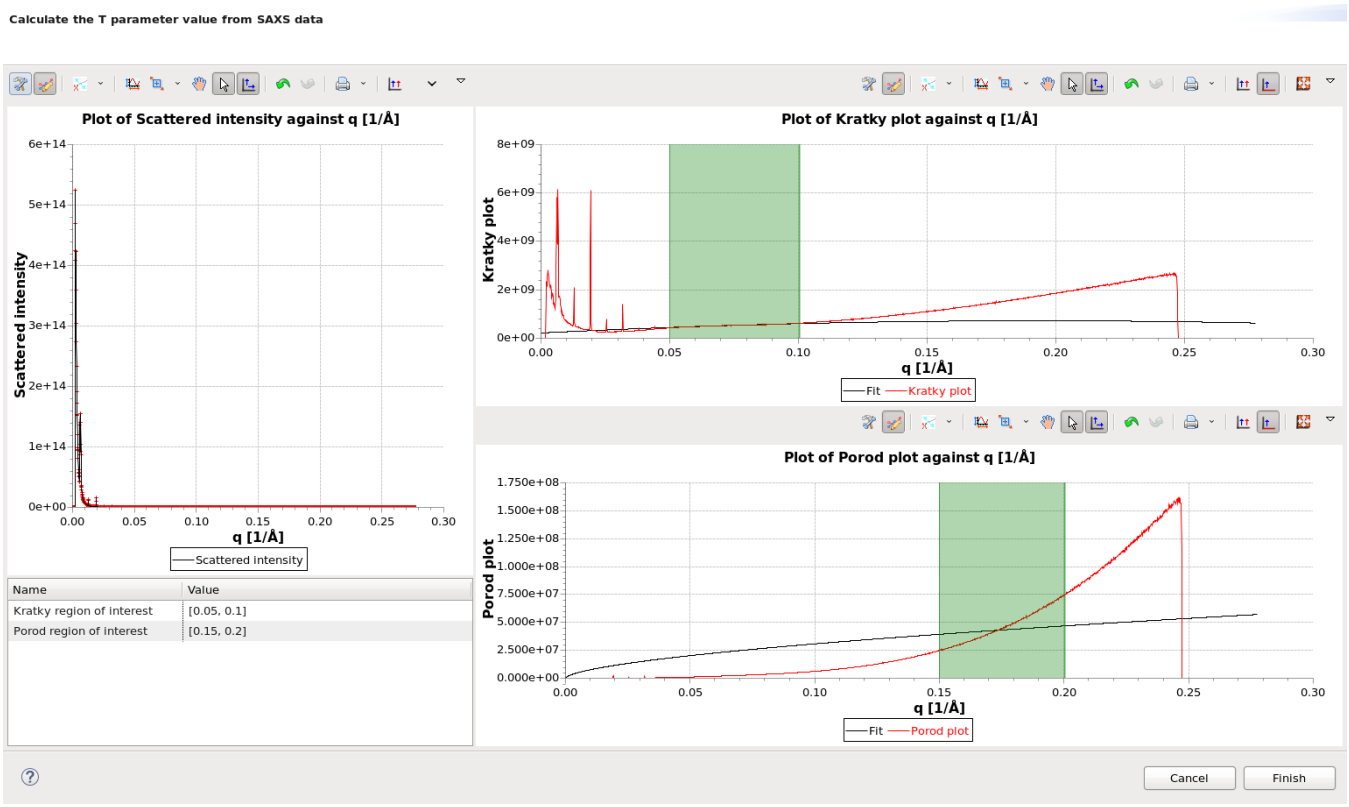
Calculating the T parameter



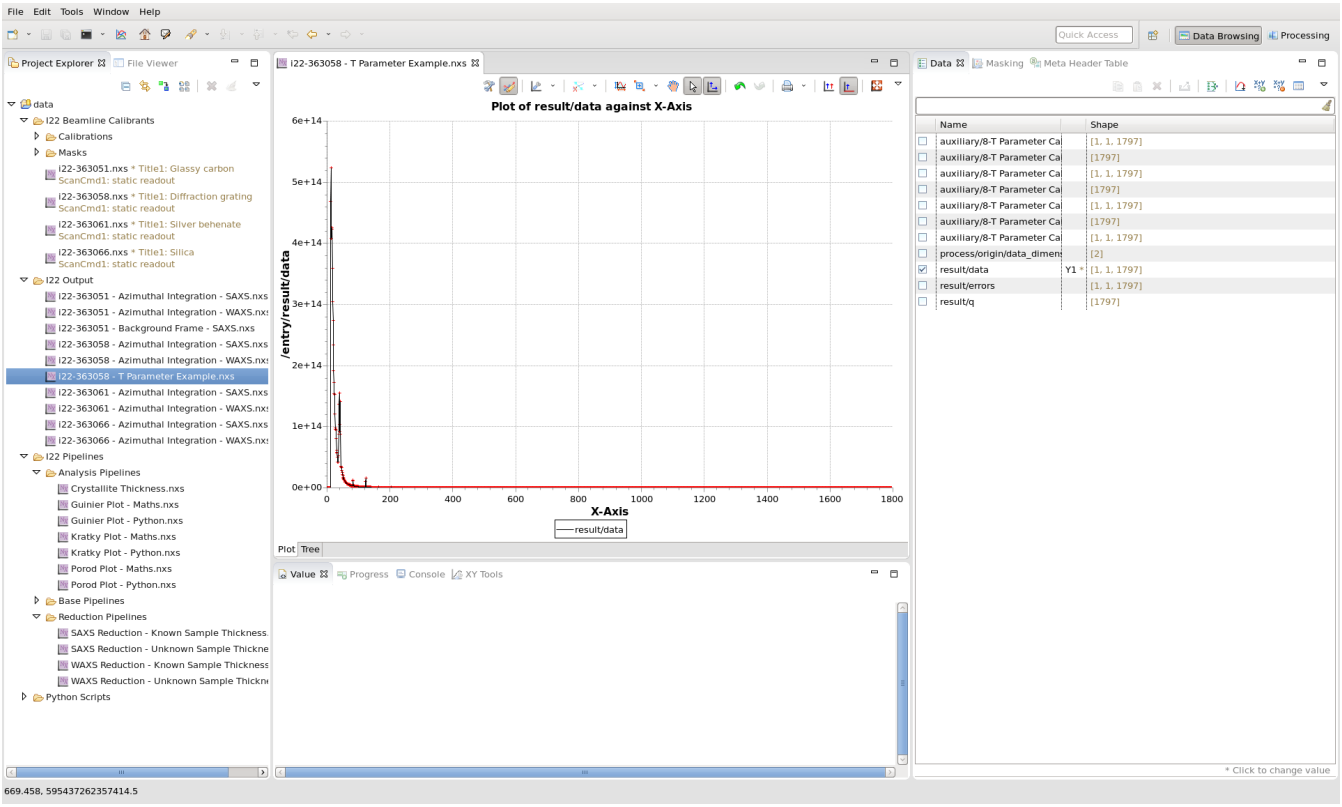
The image above shows the pipeline for calculating the crystallite T value, this pipeline can be loaded by dragging-and-dropping the *Crystallite Thickness.nxs* file from the *Analysis Pipelines* subfolder in the *I22 Pipelines* folder in the *Project Explorer* panel into the *Processing* panel or by adding in the step *T Parameter Calculation [SAXS]* on top of a standard azimuthal reduction pipeline.



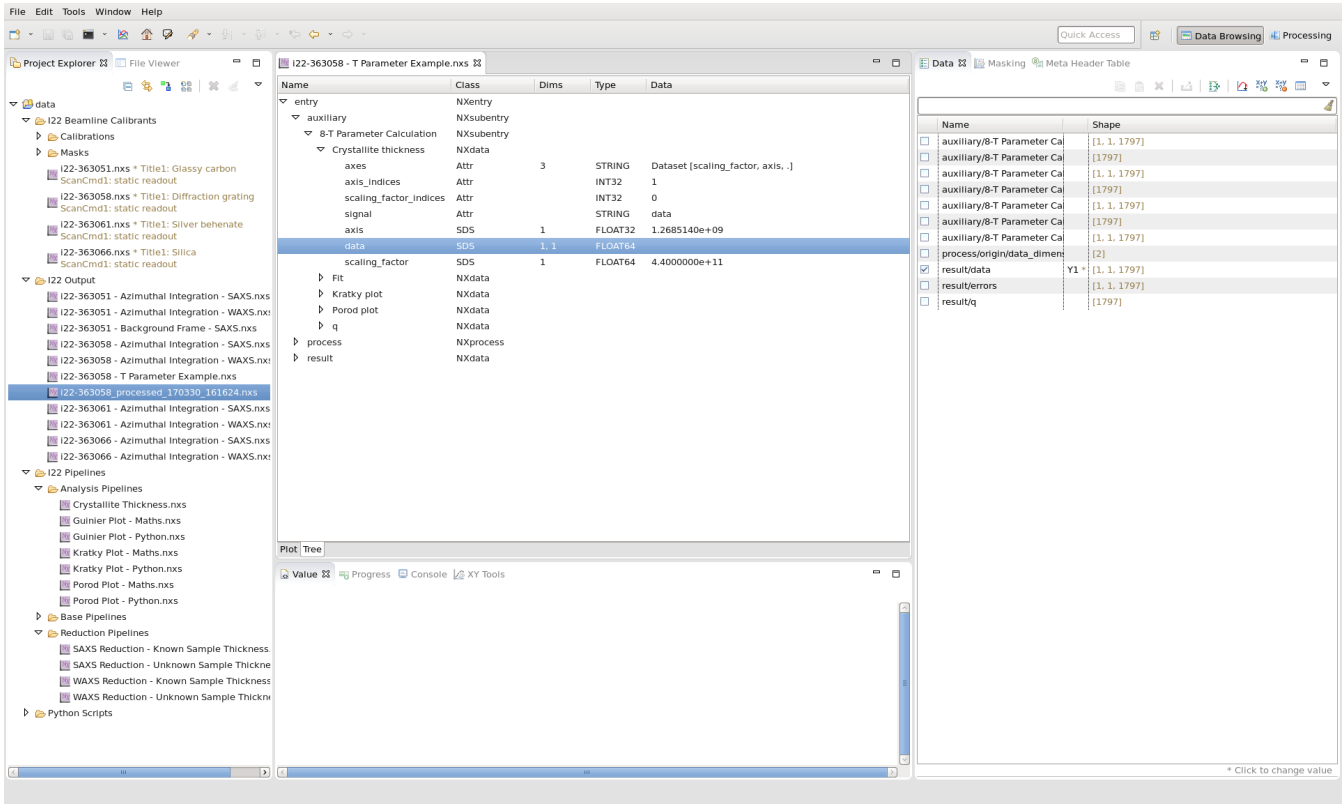
When loaded a DAWN should look similar to the view highlighted above. From here click on the *Live Setup* button in the *Model* panel to bring up the panel highlighted below.



By clicking and dragging on the edges of the green areas of the plot, adjust the Kratky (low *q*) and Porod (high *q*) fit regions as appropriate to best fit the data presented. When finished click *Finish* to be returned to the *Processing* perspective window and click *Process all files*. After running the pipeline, click on the *Data Browsing* button, just above the *Model* panel, this will switch the DAWN perspective.



So far, this guide has focussed only on the *Processing* perspective, principally as this perspective provides a simple way to reduce and analyse data obtained at I22. However, inspection of some values requires the use of the *Data Browsing* perspective to look at the values stored in NeXus files. Double clicking on the NeXus file generated from running the T parameter pipeline, in the *Project Explorer* panel, should present a view similar to the one above. In the centre of the perspective is a plot window, under which are two tabs, one entitled *Plot* and the other *Tree*, select *Tree*. At this point the plot window will change to a drop down list, expanding either all of the options, or just the ones highlighted below, reveals the contents of the NeXus file. More information about the *Data Browsing* perspective can be found from Tutorials 1 & 2 on the DAWN Tutorials - Diamond Light Source.



As highlighted above, under *entry -> auxiliary -> T parameter calculation -> Crystallite thickness* can be found the variables produced by performing the T parameter calculation. This data directory has 7 variables associated with it, with the variable of interest being the number associated with the *data* tag, as highlighted above. Naturally, as this is a raw result, the accuracy of this number is far greater than reasonable and a degree of rounding will be required.

The keener reader will notice that in the above image that the T parameter calculation is listed as *8-T parameter calculation*, this is because the T parameter calculation was the 8th step in the processing pipeline.

Previous: SAXS Tools plug-in | **Next:** Orientation calculations

Orientation calculations

This page provides information on the following orientation calculations that can be performed using DAWN:

- Cinader & Burghardt
- Herman

All instructions provided were performed on a Linux version of DAWN 2.7.x. Although there are no specific alterations to the way the program works between the operating systems, certain parts of DAWN might look different on your computer to the illustrations in this guide.

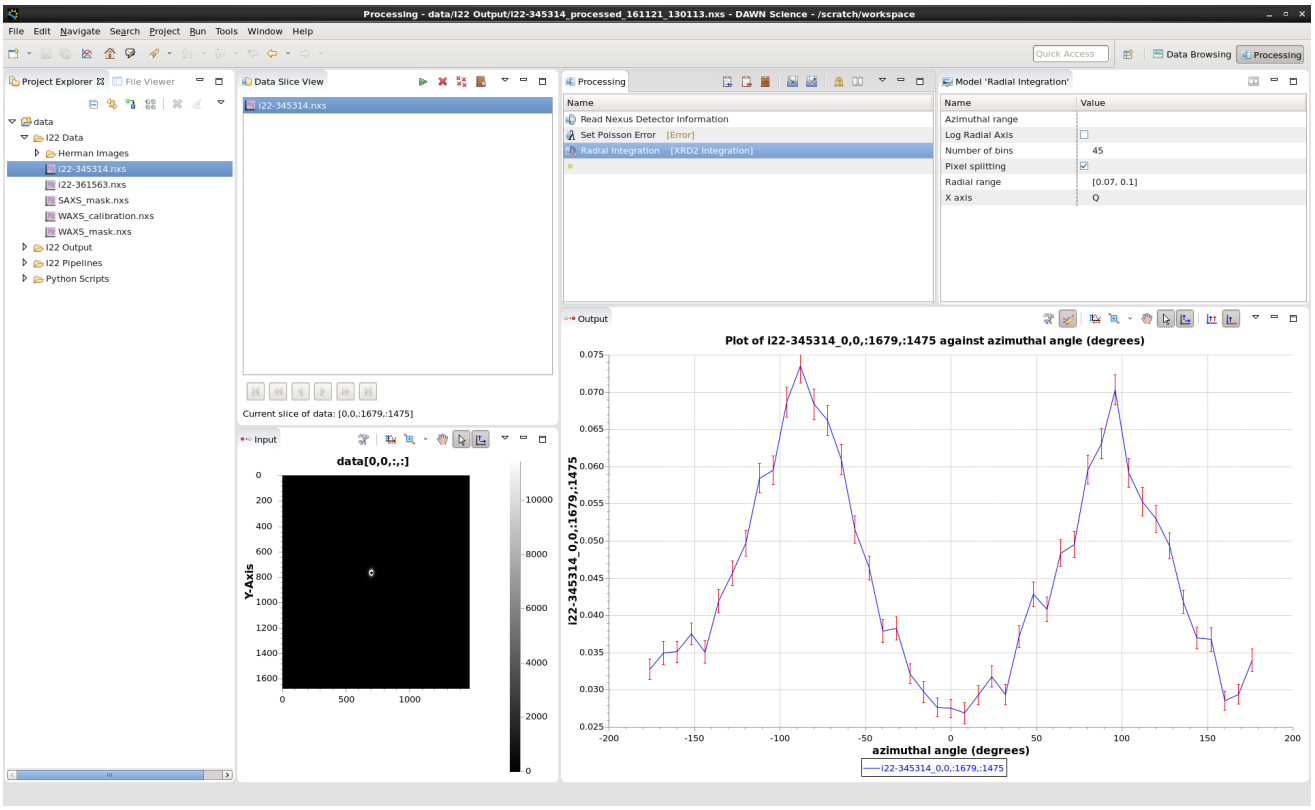
Cinader & Burghardt

The calculations provided through the use of this plug-in come from: D. K. Cinader and W. R. Burghardt, *Macromolecules*, 1998, **31**, 9099-9102, DOI: 10.1021/ma981139l.

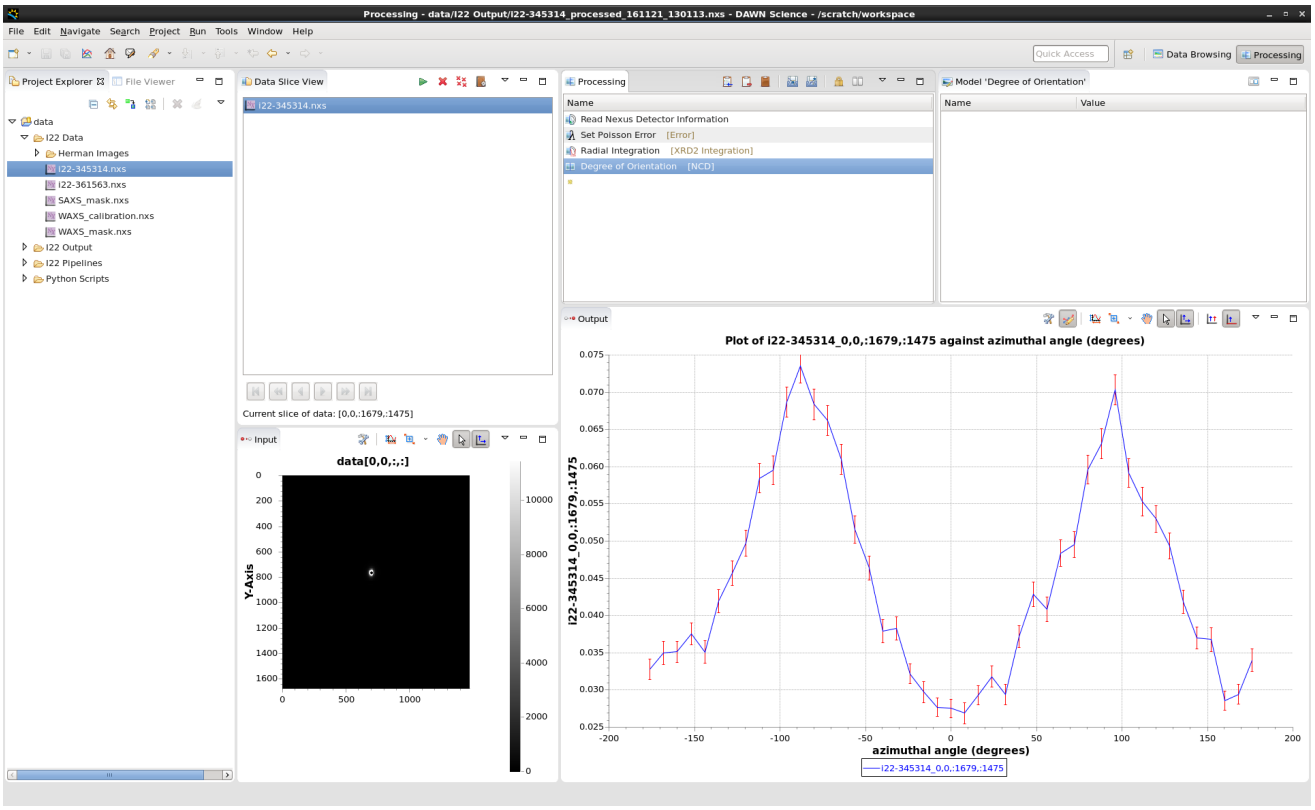
The Cinader & Burghardt equation provides an orientation parameter, alongside an orientation angle and an orientation vector, as a $\cos$ and $\sin$ angle pair of the orientation parameter.

$$CaB = \sqrt{\langle \mathbf{uu} \rangle} = \begin{bmatrix} \left\langle \cos^2 \beta \right\rangle & \langle \sin\beta \cos\beta \rangle \\ \langle \sin\beta \cos\beta \rangle & \left\langle \sin^2 \beta \right\rangle \end{bmatrix} \quad \text{where} \quad \langle f\beta \rangle = \frac{\int_0^{2\pi} f\, I(\beta)\, d(\beta)}{\int_0^{2\pi} I(\beta)\, d(\beta)}$$

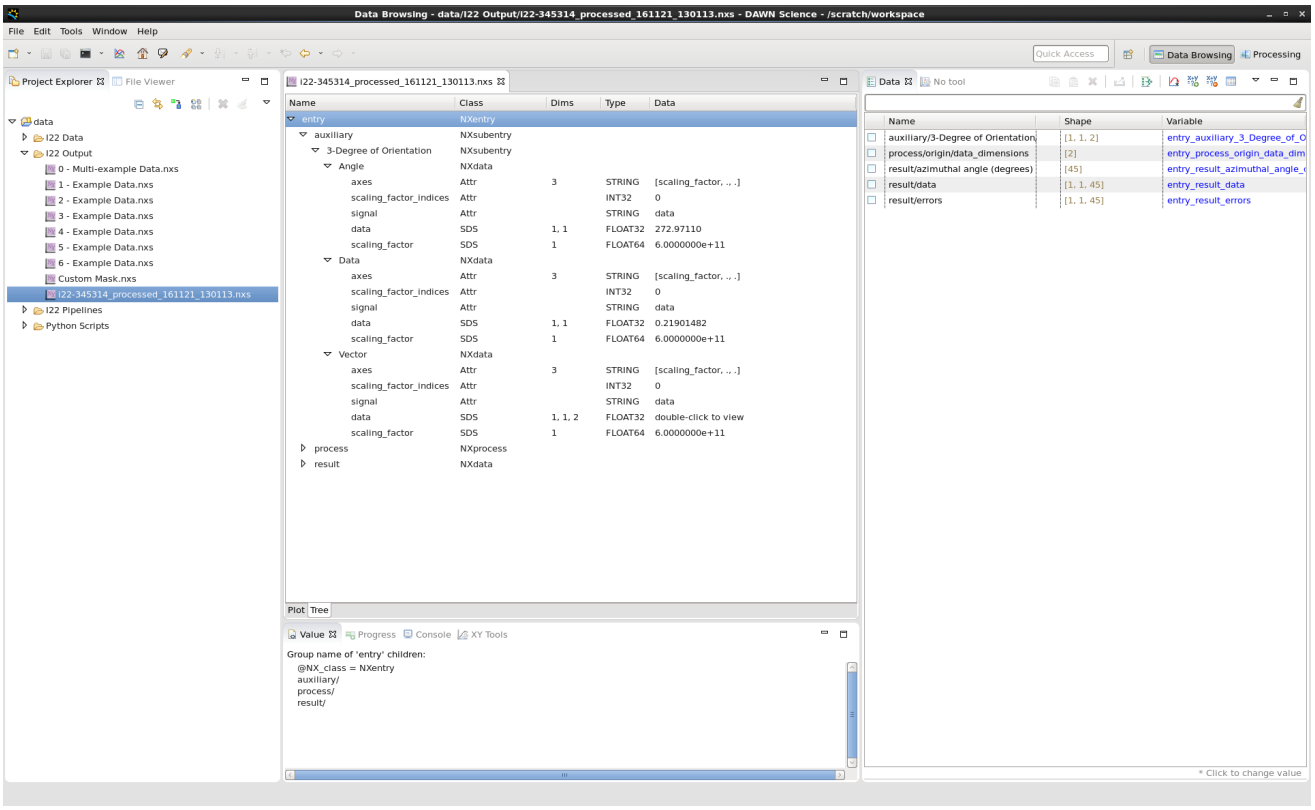
To use this plug-in, load the desired detector file, the *Read Nexus Detector Information, Set Poisson Error* and *Radial Integration [XRD2 Integration]* processing plug-ins into the processing perspective, as highlighted below.



In the radial integration plug-in's *Model* panel insert the radial range of interest, in q, and the number of data bins, if desired. When the radial plot reflects the area of interest, load the Degree of Orientation [NCD] plug-in and click *Process all files* in the *Data Slice View* panel.



Once the file, or files, have been processed, switch to the *Data Browsing* perspective, open the file from the *Project Explorer* panel and switch from the *Plot* to *Tree* view on the middle panel, as discussed on the Viewing results in DAWN.



In the auxiliary information section can be found the solution to the equation above, entitled *Data*, the angle of orientation (where zero degrees is at 3 o'clock on the detector image, increasing angle is clockwise) and result of the above equation as a $\cos$ and $\sin$ angle vector pair.

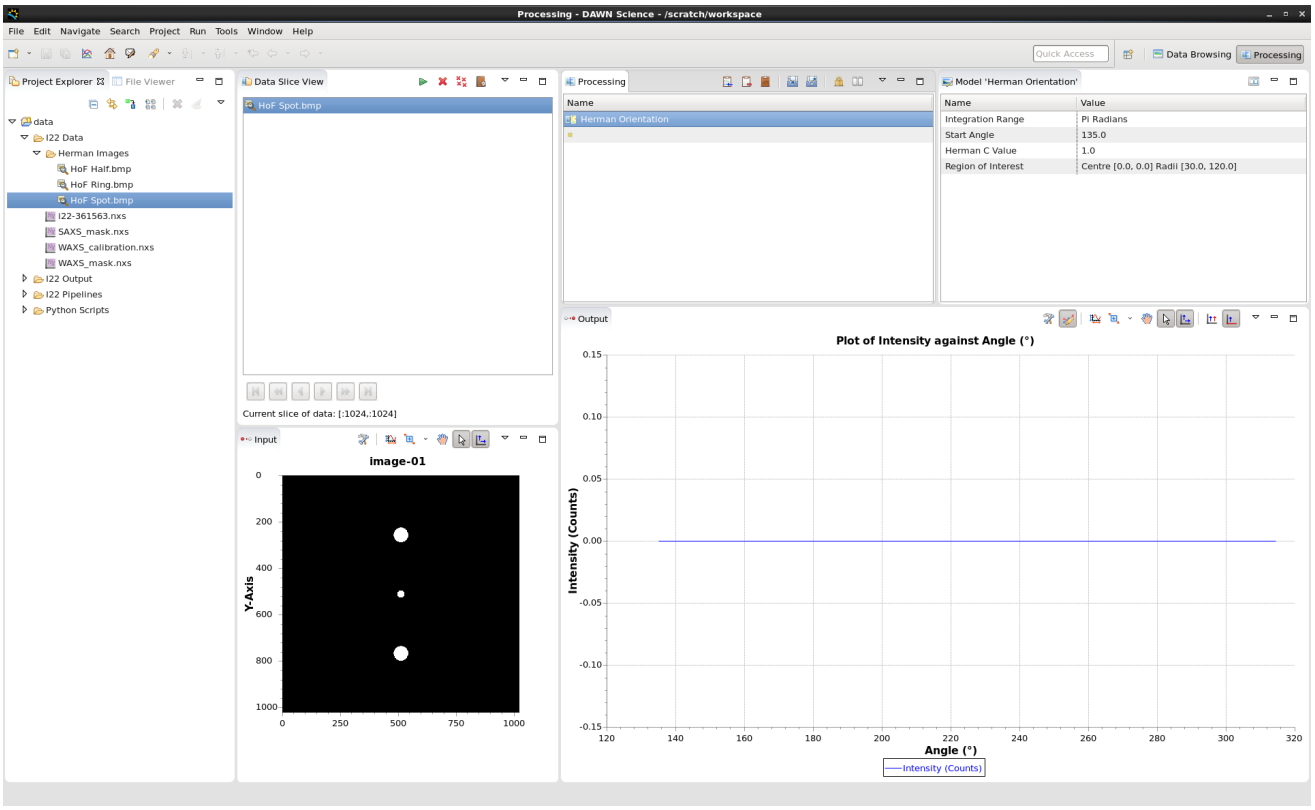
Hermans Orientation Factor

The calculations provided through the use of this plug-in come from: P.H. Hermans, *Recueil des Travaux Chimiques des Pays-Bas*, 1944, **63**, 13-24, DOI: 10.1002/recl.19440630103.

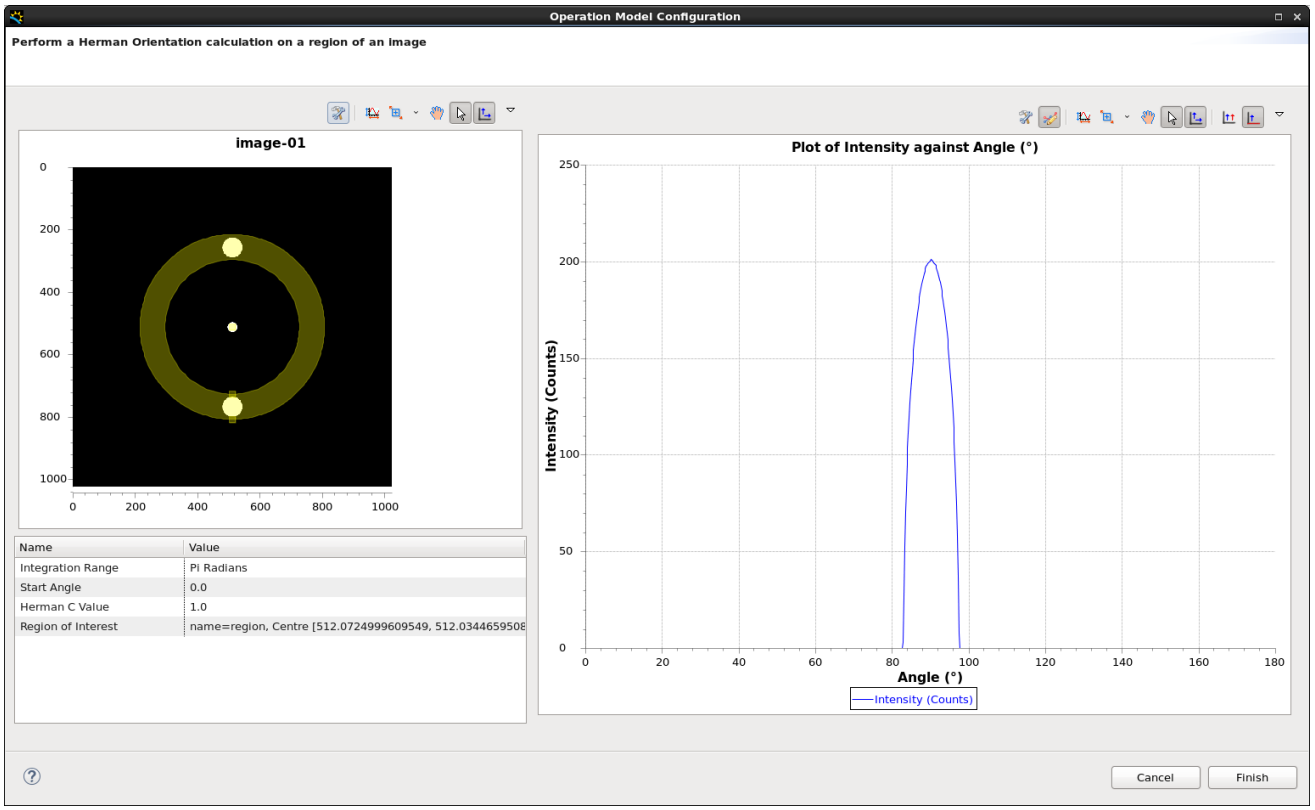
Herman's orientation factor gives a single number to indicate the degree of orientation along a given azimuth. A value of one indicates that the sample is perfectly aligned with respect to the given starting angle, *e.g.* a diffraction spot or Bragg peak. A value of zero indicates no sample alignment, *e.g.* a uniform diffraction ring. Through to minus a half, perfect alignment 90 degrees from the starting angle. It is calculated from:

$$HoF = \frac{3\left\langle \cos^2\phi \right\rangle - 1}{2} \quad \text{where} \quad \left\langle cos^2\phi \right\rangle = \frac{\sum_{i=0}^{i=x} I_i \cos^2\phi_i \sin\phi_i}{\sum_{i=0}^{i=x} I_i \sin\phi_i}$$

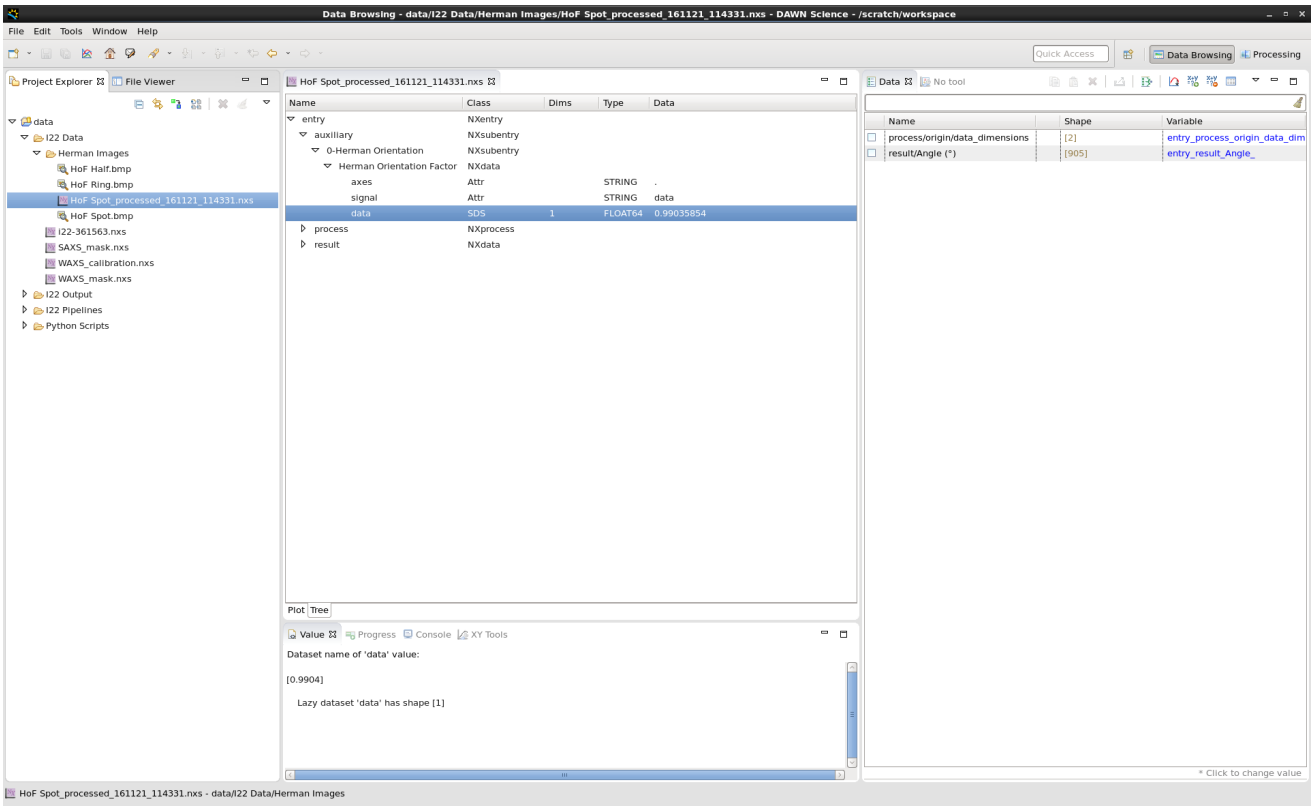
Where, depending on the user's input, x is either equal to 180 or 90 degrees, or π / $\frac{\pi}{2}$ radians, of interest. To use this plug-in, load the desired detector file, any masking or detector information with the *Read Nexus Detector Information* plug-in and the *Herman Orientation* processing plug-in into the processing perspective, as highlighted below.



When these have been loaded, clicking the *Live setup* button in the plug-in's *Model* panel allows for customisation of the region of interest to investigate, as discussed in the region of interest section on the processing perspective page, and as highlighted below.



After selecting the desired region of interest and clicking *Finish* the user will be returned to the processing perspective window with the *Output* plot showing the intensity as a function of intensity around the azimuth selected in the ROI tool. Running the processing pipeline on a file, or files, will then create result NeXus files from which the Herman orientation factor can be found. For the above example when the starting angle is set to 90 degrees, not zero as above, running the Herman orientation calculation should equate to one, or thereabouts.



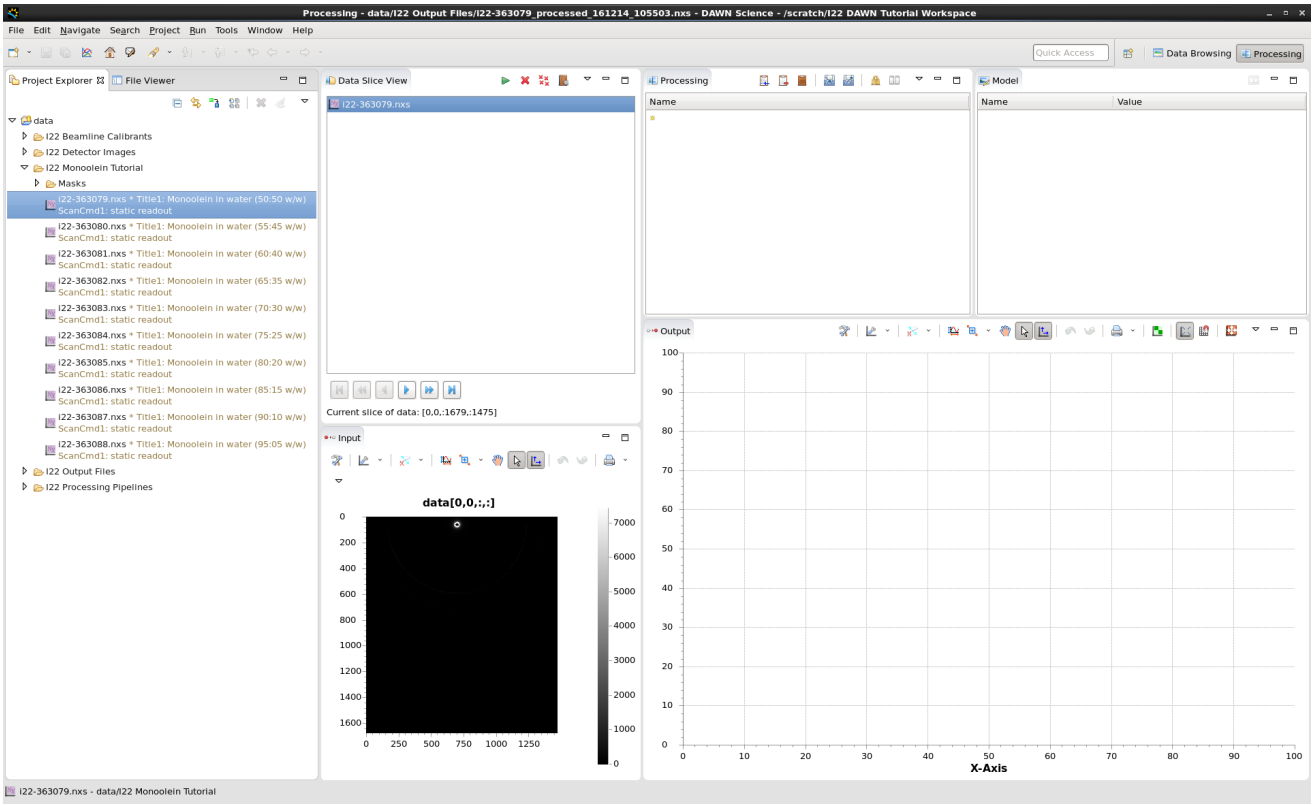
After running the orientation calculation, switching to the Data Browsing perspective and opening the results file in the *Tree* view, as discussed on the Viewing results in DAWN page, it is possible to see that the degree of orientation for this image is 0.99, as highlighted above. For the more curious reader, running the Herman orientation calculation on the image above with a start angle of 0° gives a value of -0.49 as the alignment is perpendicular to the start angle.

Previous: Crystallite parameters | **Next:** Viewing multiple images

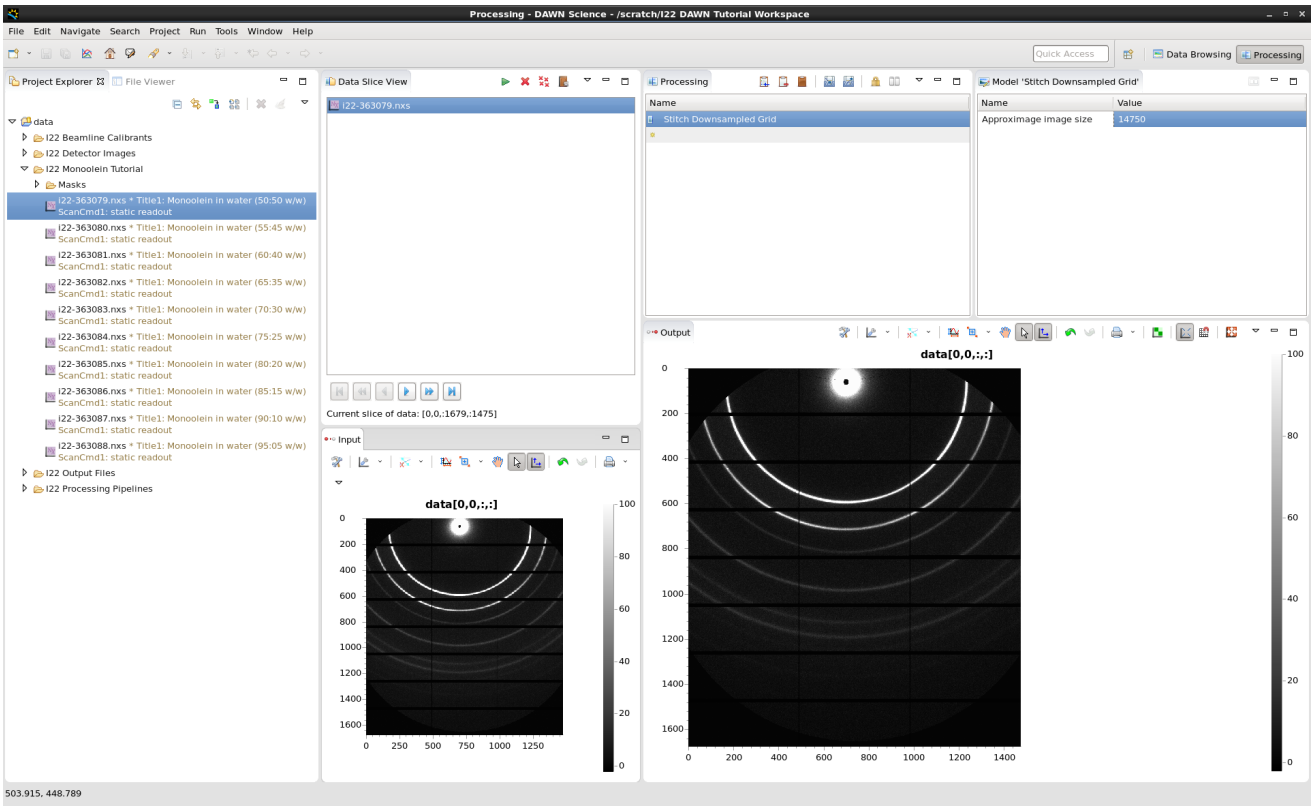
Viewing multiple images

This page provides information on how to view multiple images in one view using DAWN All instructions provided were performed on a Linux version of DAWN 2.4.x. Although there are no specific alterations to the way the program works between the operating systems, certain parts of DAWN might look different on your computer to the illustrations in this guide.

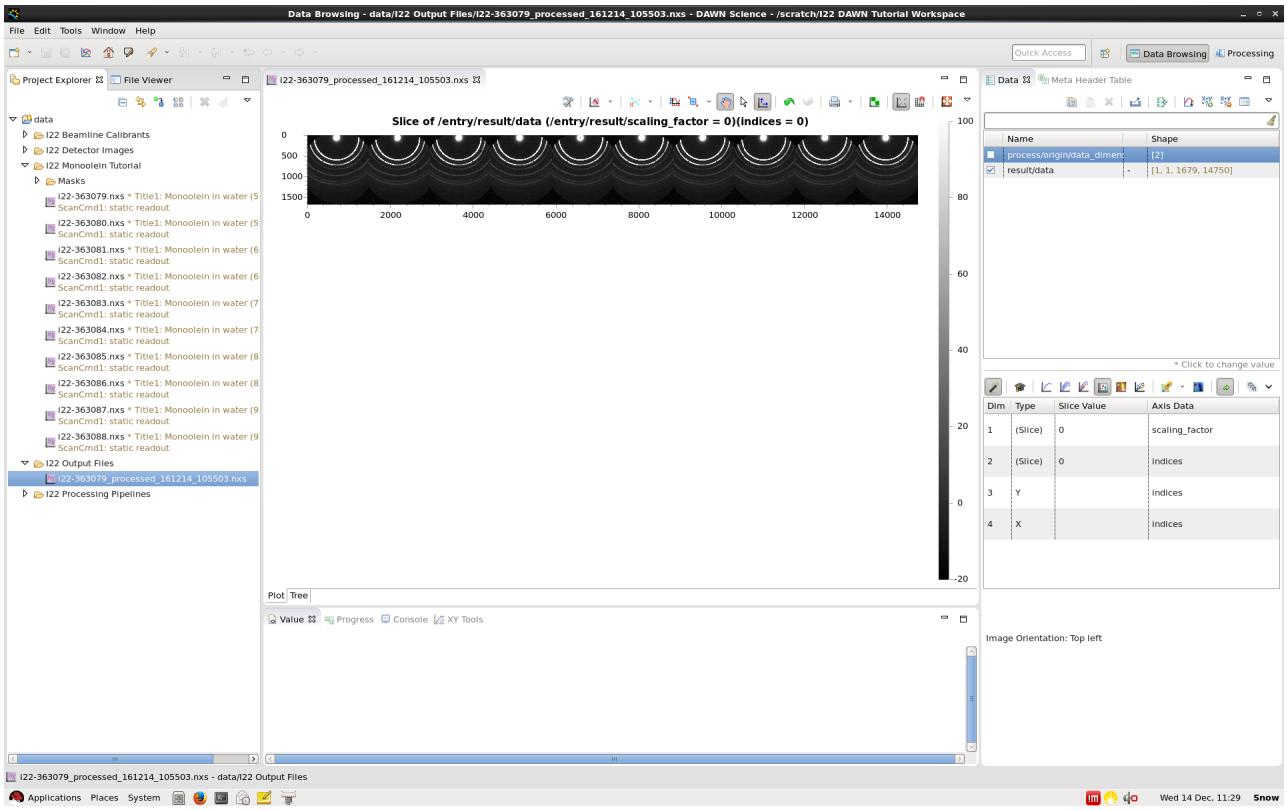
The first step is to open the Processing perspective within DAWN and load in the dataset you wish to be stitched together. If required, instructions how to do this can be found on the The Processing perspective page of this guide. After loading the desired file a view similar to the one below should be presented.



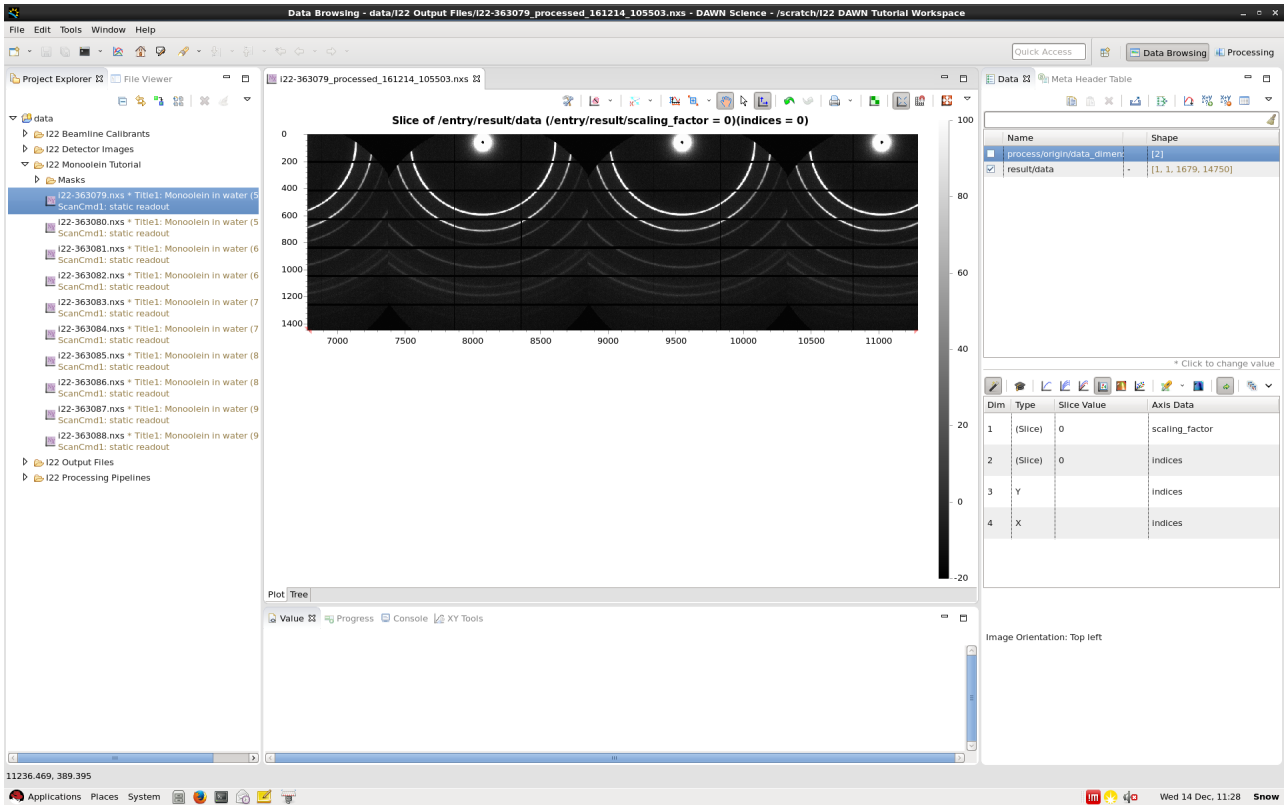
With the data loaded into DAWN the next step is to load in the *Stitch Downsampled Grid* plug-in into the processing pipeline, as highlighted below. In the *Model* panel for the plug-in there is one option *Approximage image size*. This plug-in will take all the frames within a single NeXus file and stitch them together along the x axis forming a series of images, the *Approximage image size* option defines how long this chain of images should be, they will be rescaled in y accordingly to fit. In the example below there are 10 images in the multi-frame NeXus file, each with an x dimension of 1475 pixels, therefore the *Approximage image size* has been set to 14750 in order to preserve the full size of the images. **Please note** making this value larger than the default value 10,000 can cause memory problems as the aggregate image is stored in memory whilst processing, change with caution.



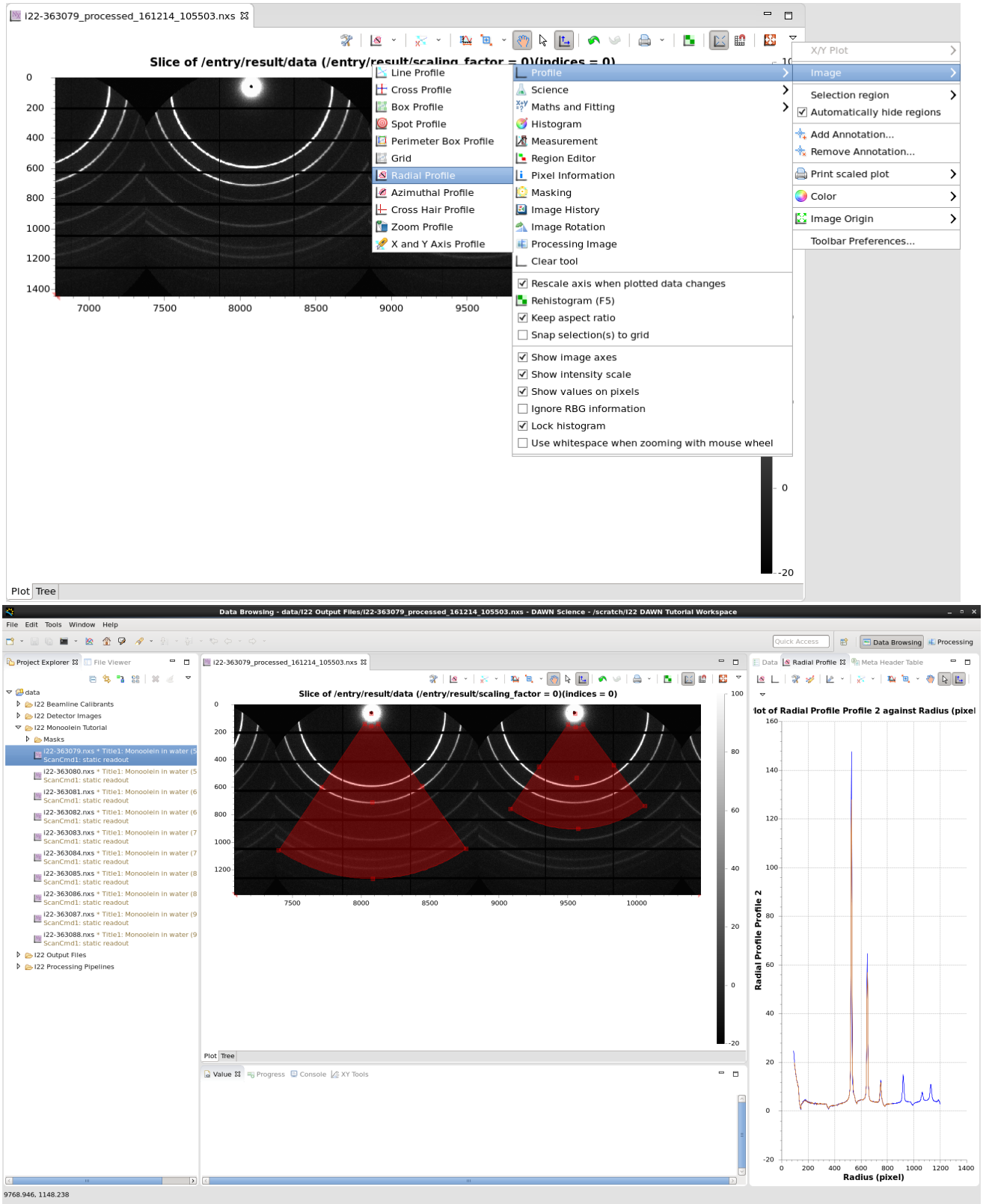
After the plug-in has been loaded and the Model parameters set, if not using the default, clicking the *Process all files* button will stitch the images in each multi-frame NeXus file together. It is worth noting that this will only stitch images together from each NeXus file, if there are multiple files in the *Data Slice View* panel each one will have its own output file containing the multi-frame data from that, individual, NeXus file. This processing step will not generate a single monolithic file containing all of the images stitched together.



After running the pipeline, switching to the *Data Browsing* perspective, to afford more screen space to view the result, loading the result file and opting to view the result data, by checking the tickbox next to *result/data* in the right hand *Data* panel, will display the stitched data.



At this point it is possible to zoom in on the data, or by using *a Profile* tool, display a region or multiple regions of interest plotted from this stitched image, highlighted below is an example using the *Radial Profile* tool.



In the above example, as the frames are almost identical, one region of interest has been made smaller to highlight in the plot on the right hand side of the screen that both regions are being plotted as the blue curve (larger region) and the orange curve (smaller region).

Previous: Orientation calculations | **Next:** Joining NeXus files

Joining NeXus files

This page provides information on joining NeXus files together. All instructions provided were performed on a Linux version of DAWN 2.7.x. Although there are no specific alterations to the way the program works between the operating systems, certain parts of DAWN might look different on your computer to the illustrations in this guide.

Linking datasets

There are a number of scenarios where, after acquiring data, it might be advantageous or desirable to join together several files into one dataset for analysis. If all of the source files are of the **same** *dimensionality*, *i.e.* all the files to be joined are all single frame or all line scan or all grid scan files of the same dimension then then can be achieved by creating a *.dawn* linker text file. When loaded within DAWN this file will act as a normal NeXus file, however, in reality it is acting as a virtual file linking a dataset, or several datasets, together from a number of different files. If the files you wish to join are of **different** *dimensionalities* then get in touch with your experimental local contact who will assist you further.

DAWN expects a .dawn file to be a standard ASCII text file and so almost any text editor can be used to create it. However, it is worth noting that more complex text editors, *.e.g.* Microsoft Word, although capable of generating such files will try to 'be helpful' which can hinder this process. For Windows based machines NotePad, bundled with Windows, is perfectly suited to creating .dawn files. For Linux/Mac/UNIX machines a number of simple text editors are available, for example gedit / vi / emacs, some of which have GUI interfaces and others which are only available *via* the command line.

An example, or starter, file can be downloaded Multiloader_Example.dawn.zip, this zip file contains a .dawn text file that reads:

This example will link the files i22-000001.nxs to i22-000010.nxs together; adding more entries (or removing entries) in the lines under the FILE_NAME line will join a greater, or lesser, number of files together. The *DIR_NAME* entry should point to the folder containing the files to be joined toegther *i.e.* /dls /i22/data/2017/cm16776-1 and the *DATASET_NAME* entry should be set to the dataset within the NeXus file that is desired to join together. Typically this will be a dataset such as */entry1/detector/data* (SAXS detector data) or */entry1/Pilatus2M_WAXS/data* (WAXS detector data). It is worth noting that if you add several *DATASET_NAME* entries on several lines, DAWN will join together multiple datasets from multiple files into one virtual file. A typical example of this, I22_Multiloader_Example.dawn.zip, for I22 might be:

Finally, should it be desired, it is also possible to define the size of the physical x and y dimensions which can be useful for mapping data, however, this feature has yet to be fully implemented for I22. For the more curious reader further instructions on this functionality can be found Linking Datasets from Multiple Files [DAWN 2.2], however, at this moment in time for experiments conducted on I22 this should be considered **experimental**.

Again, it is advised to get in touch with your local contact if more esoteric file joining is required.

Previous: Viewing multiple images | **Next:** Data Back-up

Q Errors

This page is a rough repository of errors in q that are present during a scan to work out whether it's required to take into account q errors in data processing.

The primary source for equations and information relating to this is from: http://www.lookingatnothing.com/ however further sources of information can be found from a variety of different sources, for example:

- Q Resolution on LOQ at ISIS
- SANS at Pulsed Neutron Sources: Present and Future Prospects
- Chapter 15 - THE SANS INSTRUMENTAL RESOLUTION

Sample thickness error contribution

Any sample in the beam has a finite thickness which gives rise to an uncertainty in q this is derived from:

$$\Delta q = \frac{4\pi}{\lambda}[\sin\theta_1 - \sin\theta_2]$$

where

$$\theta_1 = \frac{1}{2}\arctan\left(\frac{l_d}{l_{sd} - r_s}\right)$$

and

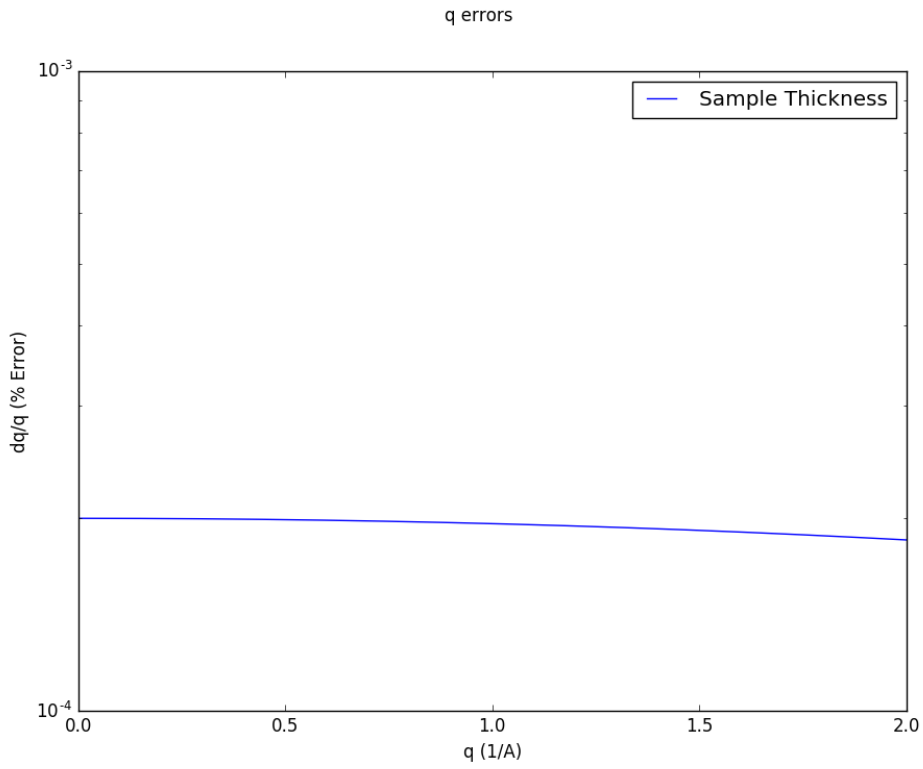
$$\theta_2 = \frac{1}{2}\arctan\left(\frac{l_d}{l_{sd} + r_s}\right)$$

and

$$l_d = l_{sd}\tan\left(2\arcsin\left(\frac{q\lambda}{4\pi}\right)\right) \quad .$$

In these equations l_{sd} is the sample-to-detector distance and r_s is the sample radius, for the purposes of this example we shall assume a wavelength of 1 $\mathring{\text{A}}$, a sample radius of 0.5 mm and a sample-to-detector distance of 5, expressed as a Python program, so that we can see this error over the normal q range:

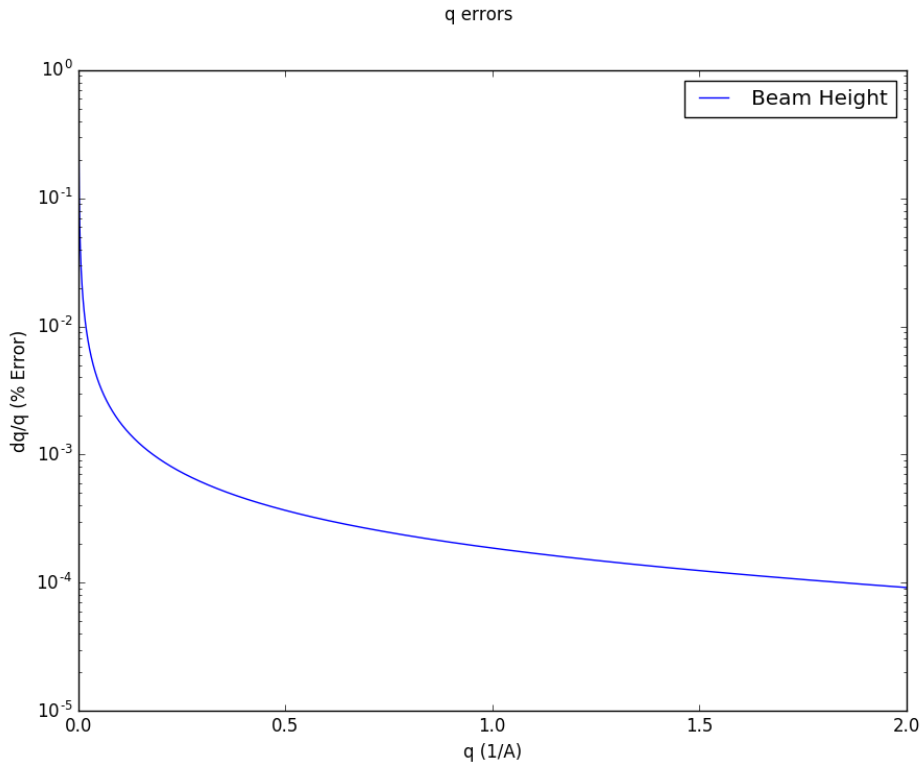
```
# Imports
import numpy
import matplotlib.pyplot as plt

# Assumptions - both qRange and wavelength are in A, so not converting to m
qRange = numpy.linspace(0.0002, 2, 2000)
l_sd = 5
r_s = 0.001
wavelength = 1

# Equations
l_d = l_sd * numpy.tan(2 * numpy.arcsin((qRange * wavelength) / (4 * numpy.pi)))
theta_1 = 0.5 * (numpy.arctan(l_d / (l_sd - r_s)))
theta_2 = 0.5 * (numpy.arctan(l_d / (l_sd + r_s)))
deltaQ = ((4 * numpy.pi) / wavelength) * (numpy.sin(theta_1) - numpy.sin(theta_2))

# Convert deltaQ into a percentage error
percentageError = deltaQ / qRange

# Plot the result
fig = plt.figure()
fig.suptitle('Sample thickness')
plt.plot(qRange, percentageError)
plt.xlabel('q (1/A)')
#plt.xscale('log')
plt.ylabel('dq/q (% Error)')
plt.yscale('log')
plt.show()
```



Beam height error contribution

Here the equations for l_d and Δq remain the same as for the sample thickness error calculation however r_s is removed in favour of the introduction of a new factor r_b for the radius of the beam. For this example we shall use a beam radius of 0.1 mm. Expressed as a Python program, so that we can see this error over the normal q range:

```
# Imports
import numpy
import matplotlib.pyplot as plt

# Assumptions - both qRange and wavelength are in A, so not converting to m
qRange = numpy.linspace(0.002, 2, 2000)
l_sd = 5
r_b = 0.001
wavelength = 1

# Equations
l_d = l_sd * numpy.tan(2 * numpy.arcsin((qRange * wavelength) / (4 * numpy.pi)))
theta_1 = 0.5 * (numpy.arctan(l_sd / (l_d - r_b)))
theta_2 = 0.5 * (numpy.arctan(l_sd / (l_d + r_b)))
deltaQ = ((4 * numpy.pi) / wavelength) * (numpy.sin(theta_1) - numpy.sin(theta_2))

# Convert deltaQ into a percentage error
percentageError = deltaQ / qRange

# Plot the result
fig = plt.figure()
fig.suptitle('Beam height induced errors')
plt.plot(qRange, percentageError)
plt.xlabel('q (1/A)')
#plt.xscale('log')
plt.ylabel('dq/q (% Error)')
plt.yscale('log')
plt.show()
```



Beam divergence error contribution

Here the equations for l_d and Δq remain the same as for the sample thickness error calculation, however, the equations for θ_1 and θ_1 change and a new term θ_d is introduced to calculate the beam's divergence.

$$\theta_1 = \frac{1}{2}\arctan\left(\frac{l_d}{l_{sd}}\right) + \theta_d$$

and

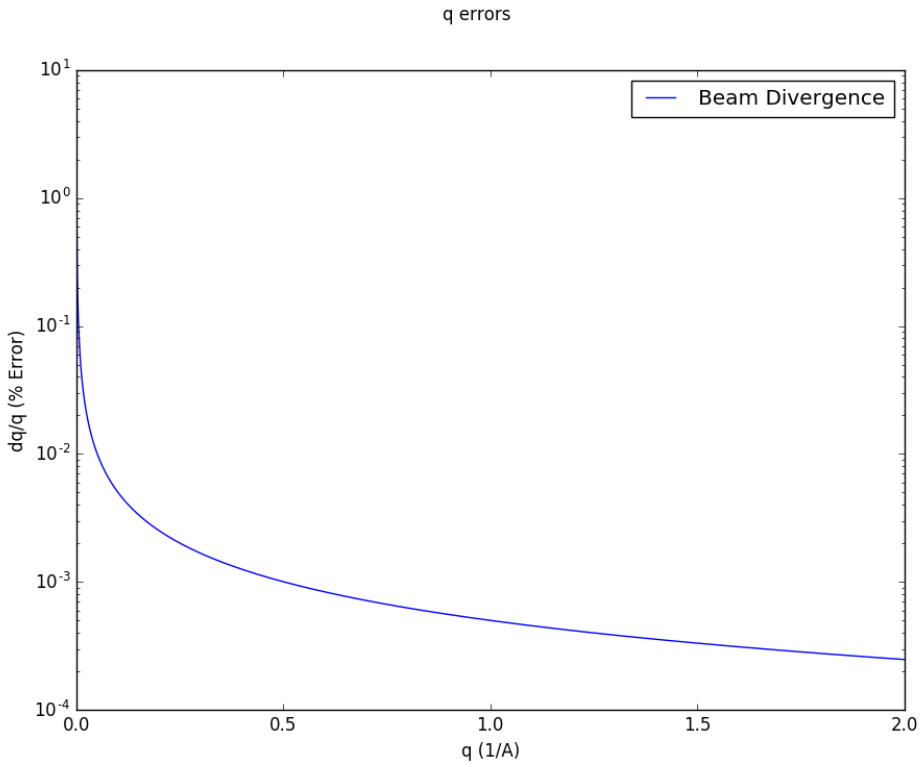
$$\theta_1 = \frac{1}{2}\arctan\left(\frac{l_d}{l_{sd}}\right) - \theta_d$$

where

$$\theta_d = \arctan\left(\frac{r_b}{l_{sd}}\right)$$

In these equations l_{sd} is the sample-to-detector distance and r_b is the beam radius, for the purposes of this example we shall assume a wavelength of 1 $\mathring{\text{A}}$, a beam radius of 1 mm and a sample-to-detector distance of 5 meters for when q = 0.2. Expressed as a Python program, so that we can see this error over the normal q range:

```
# Imports
import numpy
import matplotlib.pyplot as plt

# Assumptions - both qRange and wavelength are in A, so not converting to m
qRange = numpy.linspace(0.002, 2, 2000)
l_sd = 5
r_b = 0.001
wavelength = 1

# Equations
l_d = l_sd * numpy.tan(2 * numpy.arcsin((qRange * wavelength) / (4 * numpy.pi)))
theta_d = numpy.arctan(r_b / l_sd)
theta_1 = 0.5 * (numpy.arctan(l_d / l_sd)) + theta_d
theta_2 = 0.5 * (numpy.arctan(l_d / l_sd)) - theta_d
deltaQ = ((4 * numpy.pi) / wavelength) * (numpy.sin(theta_1) - numpy.sin(theta_2))

# Convert deltaQ into a percentage error
percentageError = deltaQ / qRange

# Plot the result
fig = plt.figure()
fig.suptitle('Beam divergence induced errors')
plt.plot(qRange, percentageError)
plt.xlabel('q (1/A)')
#plt.xscale('log')
plt.ylabel('dq/q (% Error)')
plt.yscale('log')
plt.show()
```



Polychromaticity error contribution

Here the equation for Δq relies upon knowing what, if any, polychromaticity there is in the beam that passes through the sample.

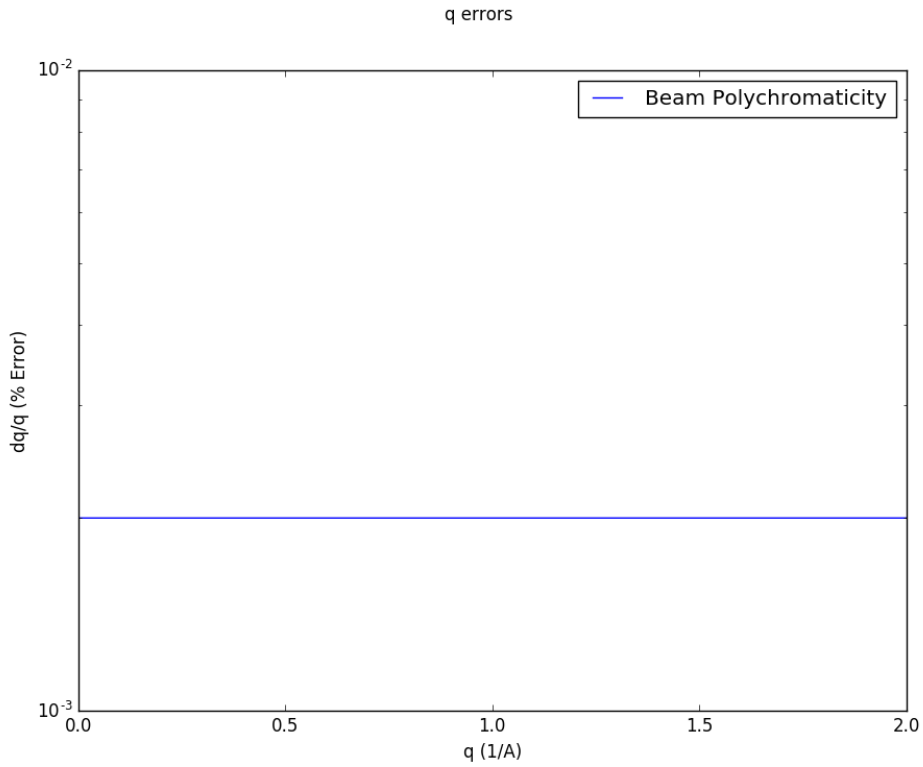
$$\Delta_q = \left(\frac{4\pi}{\lambda_1} - \frac{4\pi}{\lambda_2} \right) \sin\theta$$

where

$$\theta = \arctan\left(\frac{q\lambda}{4\pi} \right)$$

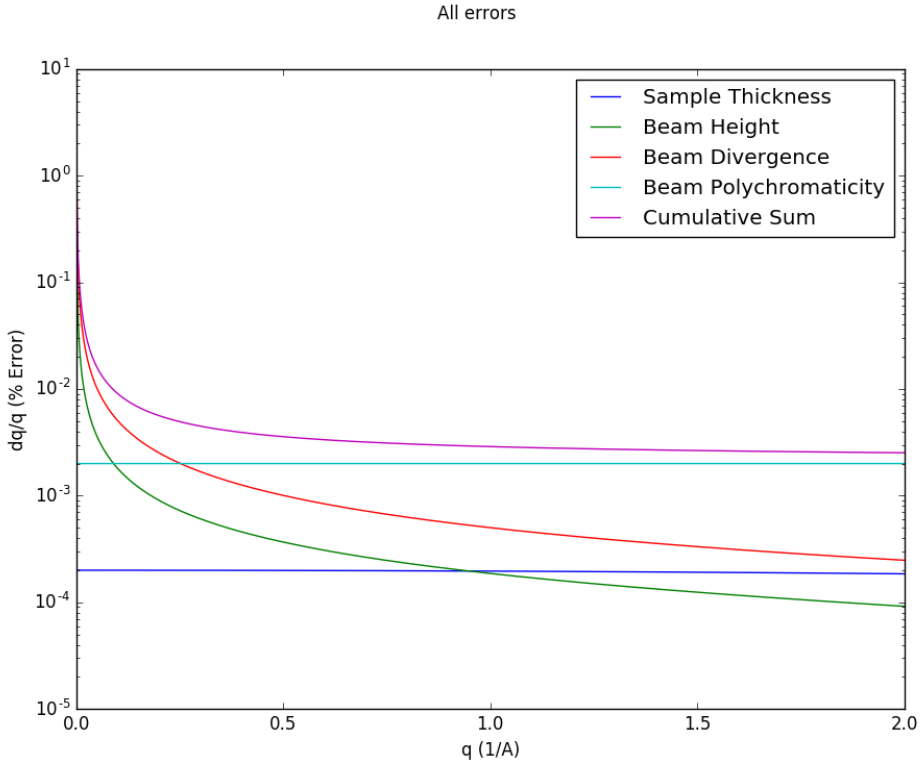
Because of this, primarily the variables are constants with the only variable being the polychromaticity of the beam. Expressed as a Python program, so that we can see this error over the normal q range:

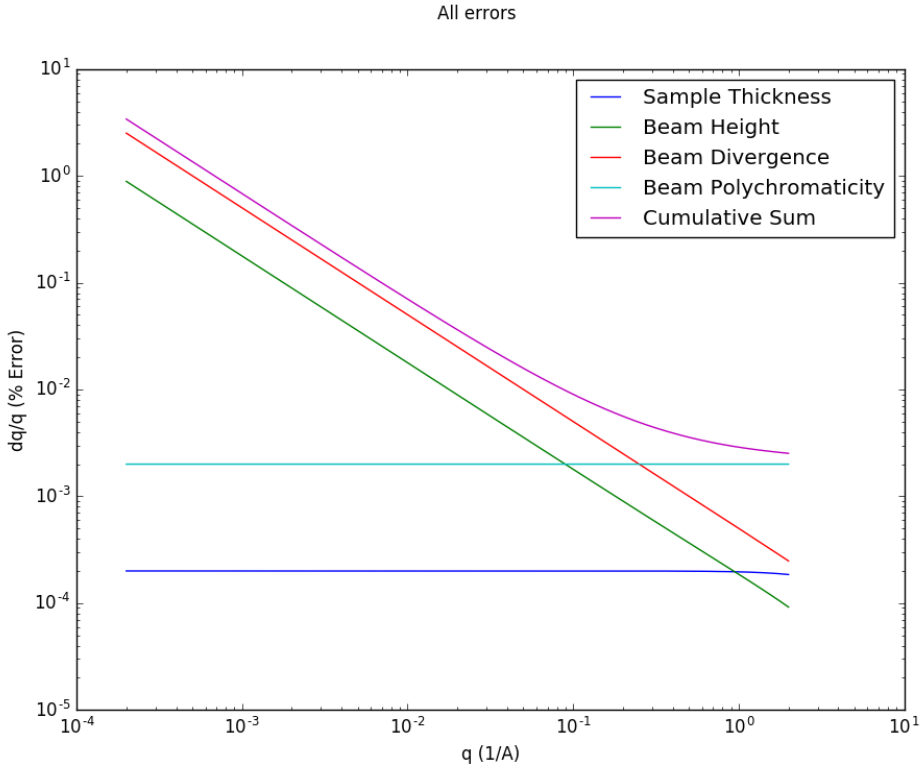
```
# Imports
import numpy
import matplotlib.pyplot as plt

# Assumptions - both qRange and wavelength are in A, so not converting to m
qRange = numpy.linspace(0.002, 2, 2000)
wavelength_1 = 0.999
wavelength_2 = 1.001

# Equations
theta = numpy.arcsin((qRange * wavelength) / (4 * numpy.pi))
deltaQ = (((4 * numpy.pi) / wavelength_1) - ((4 * numpy.pi) / wavelength_2)) * numpy.sin(theta)


 # Convert deltaQ into a percentage error
percentageError = deltaQ / qRange

# Plot the result
fig = plt.figure()
fig.suptitle('Polychromaticity induced errors')
plt.plot(qRange, percentageError)
plt.xlabel('q (1/A)')
#plt.xscale('log')
plt.ylabel('dq/q (% Error)')
plt.yscale('log')
plt.show()
```

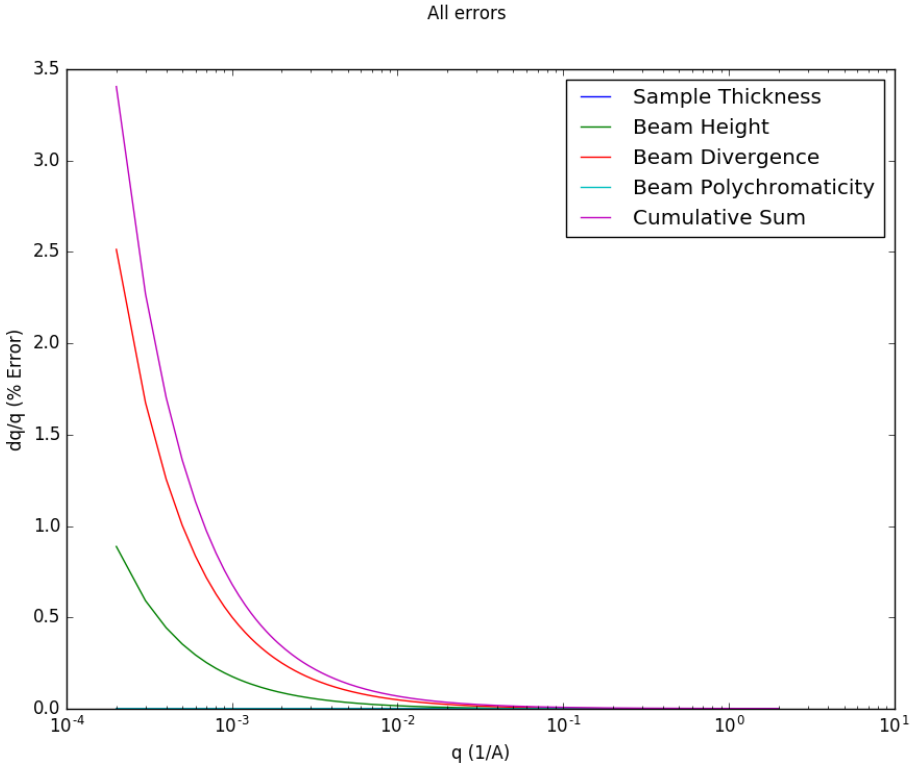


Cumulative plot & comparison to reference

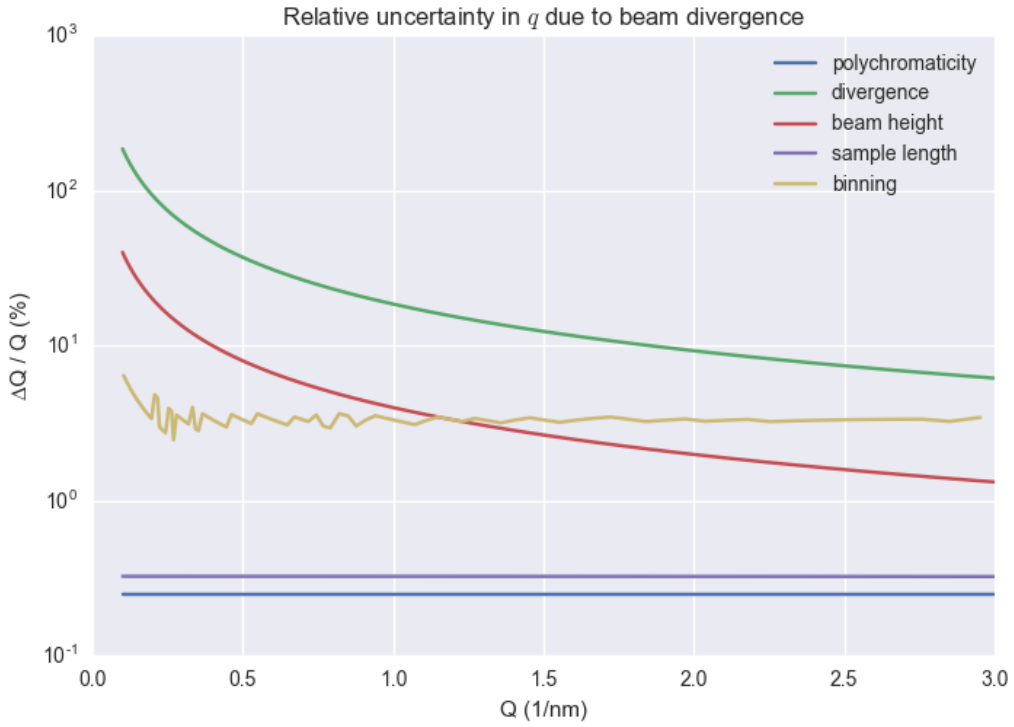
Firstly, if we are to overlay all the previous plots we can obtain the following overlaid plot and, if we sum all the contributions, a total "maximum" error (although this is not how errors work...).


All errors
Sample Thickness
Beam Height
Beam Divergence
Beam Polychromaticity
Cumulative Sum
dq/q (% Error)
q (1/A)


All errors
Sample Thickness
Beam Height
Beam Divergence
Beam Polychromaticity
Cumulative Sum
dq/q (% Error)
q (1/A)



We can see that overall we are running at 1% or less error at q values higher than 1×10^{-3}, however, these formulae might require more analysis as the assumptions taken into consideration may, or may not be accurate. Comparing these results to an study of x-ray scattering error contributions (http://www.lookingatnothing.com/index.php/archives/2258) we can visually see the following:



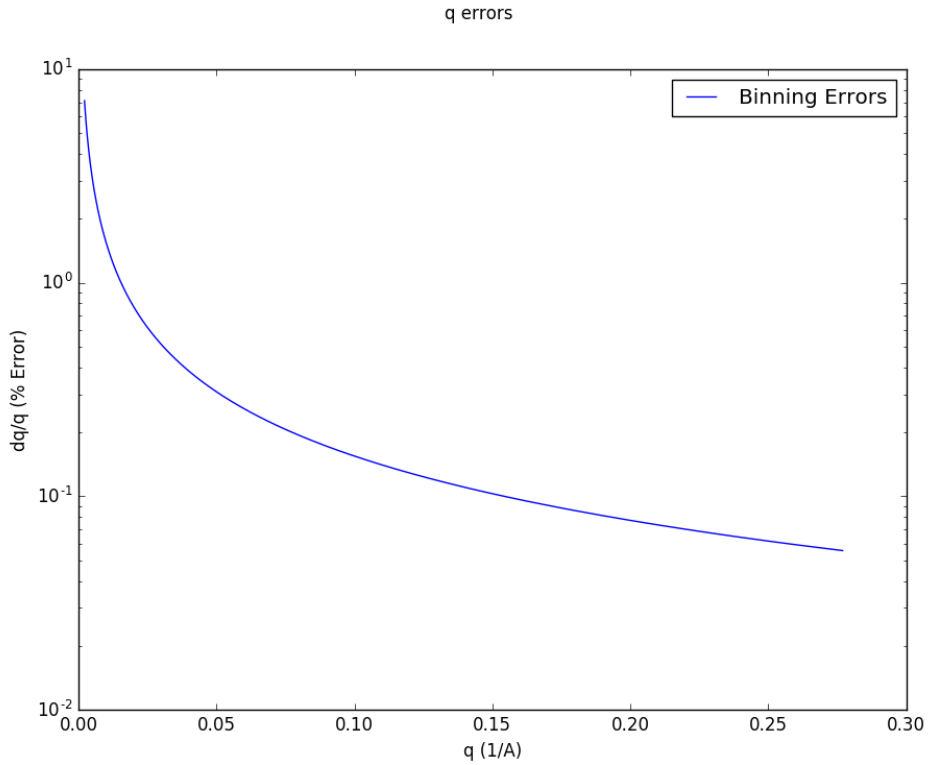
As one might expect the error values are much lower, typically an order of magnitude or better, however, especially at lower q a consideration of error may be valid.

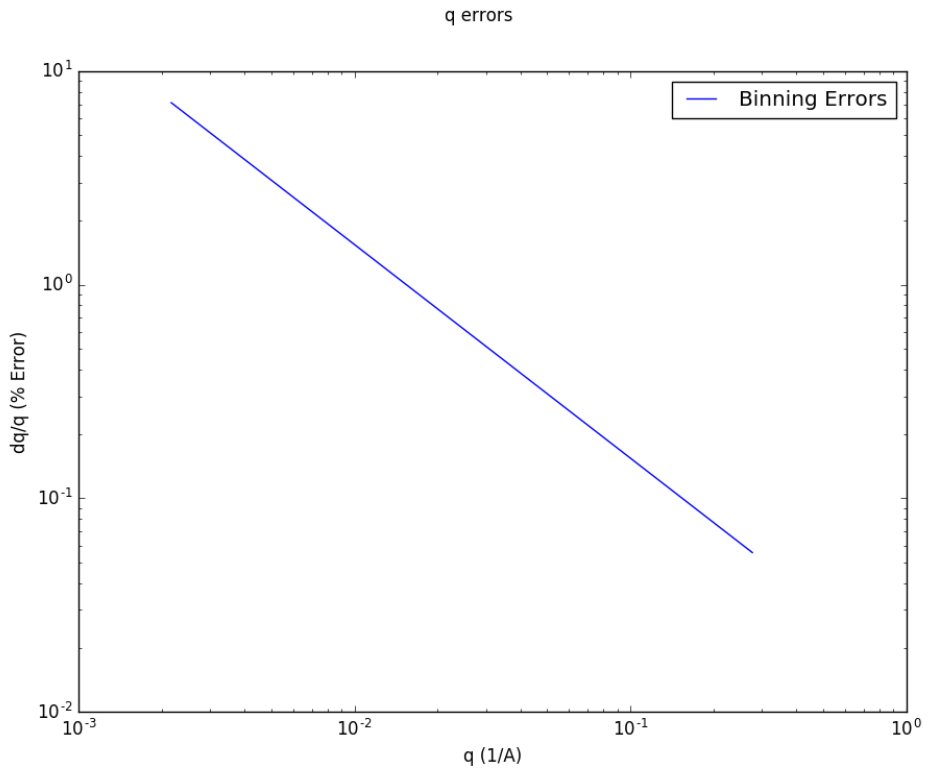
As for all the other examples, expressed as a Python program, so that we can see this error over the normal q range:

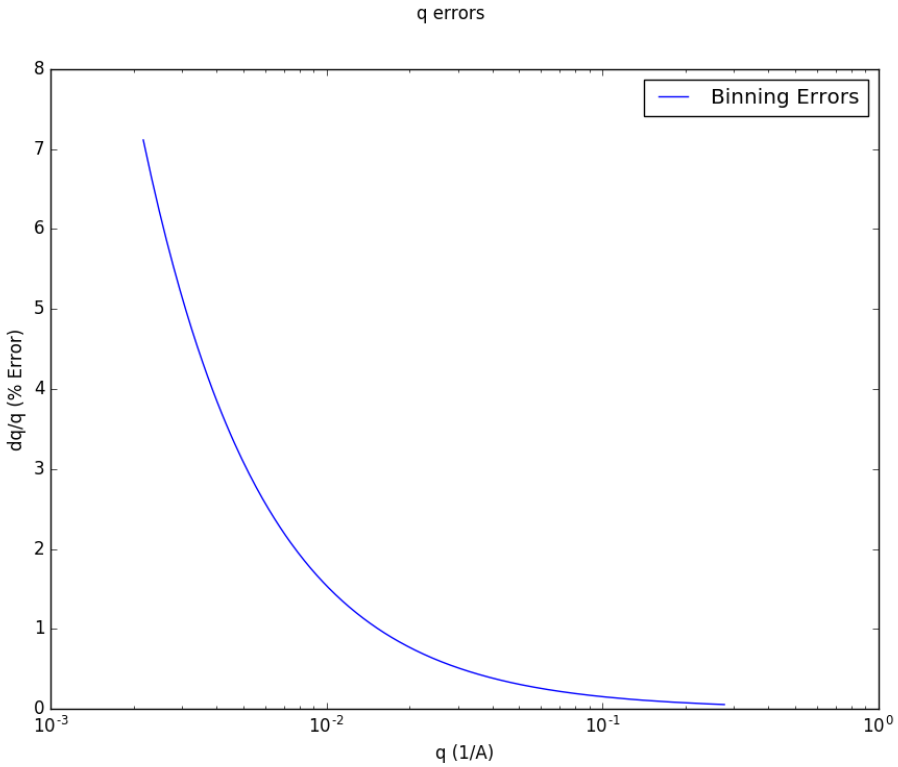
```
# Imports
import numpy
import matplotlib.pyplot as plt

# Assumptions - both qRange and wavelength are in A, so not converting to m
qRange = numpy.linspace(0.0002, 2, 20000)
l_sd = 5
r_s = 0.0005
r_b = 0.0001
wavelength = 1
wavelength_1 = 0.999
wavelength_2 = 1.001


# Sample thickness calculations
l_d = l_sd * numpy.tan(2 * numpy.arcsin((qRange * wavelength) / (4 * numpy.pi)))
theta_1 = 0.5 * (numpy.arctan(l_d / (l_sd - r_s)))
theta_2 = 0.5 * (numpy.arctan(l_d / (l_sd + r_s)))
deltaQ = ((4 * numpy.pi) / wavelength) * (numpy.sin(theta_1) - numpy.sin(theta_2))
thicknessError = deltaQ / qRange


# Beam height calculations
l_d = l_sd * numpy.tan(2 * numpy.arcsin((qRange * wavelength) / (4 * numpy.pi)))
theta_1 = 0.5 * (numpy.arctan(l_sd / (l_d - r_b)))
theta_2 = 0.5 * (numpy.arctan(l_sd / (l_d + r_b)))
deltaQ = ((4 * numpy.pi) / wavelength) * (numpy.sin(theta_1) - numpy.sin(theta_2))
beamHeightError = deltaQ / qRange


# Beam divergence calculations
l_d = l_sd * numpy.tan(2 * numpy.arcsin((qRange * wavelength) / (4 * numpy.pi)))
theta_d = numpy.arctan(r_b / l_sd)
theta_1 = 0.5 * (numpy.arctan(l_d / l_sd)) + theta_d
theta_2 = 0.5 * (numpy.arctan(l_d / l_sd)) - theta_d
deltaQ = ((4 * numpy.pi) / wavelength) * (numpy.sin(theta_1) - numpy.sin(theta_2))
beamDivergenceError = deltaQ / qRange


# Beam polychromaticity calculations
theta = numpy.arcsin((qRange * wavelength) / (4 * numpy.pi))
deltaQ = (((4 * numpy.pi) / wavelength_1) - ((4 * numpy.pi) / wavelength_2)) * numpy.sin(theta)
polychromaticityError = deltaQ / qRange


# For an overall look
cumulativeSumError = thicknessError + beamHeightError + beamDivergenceError + polychromaticityError


fig = plt.figure(figsize = (10.24, 7.68), dpi = 100)
fig.suptitle('q errors')
plt.plot(qRange, thicknessError, label = "Sample Thickness")
plt.plot(qRange, beamHeightError, label = "Beam Height")
plt.plot(qRange, beamDivergenceError, label = "Beam Divergence")
plt.plot(qRange, polychromaticityError, label = "Beam Polychromaticity")
plt.plot(qRange, cumulativeSumError, label = "Cumulative Sum")


plt.xlabel('q (1/A)')
plt.xscale('log')
plt.ylabel('dq/q (% Error)')
plt.yscale('log')
plt.legend()
plt.show()
```

Pixel binning

Calculation of pixel binning was performed from an experimentally obtained scan at 7 meters, the experimental file can be downloaded from here (a tutorial scan) and the processed file can be downloaded from here.







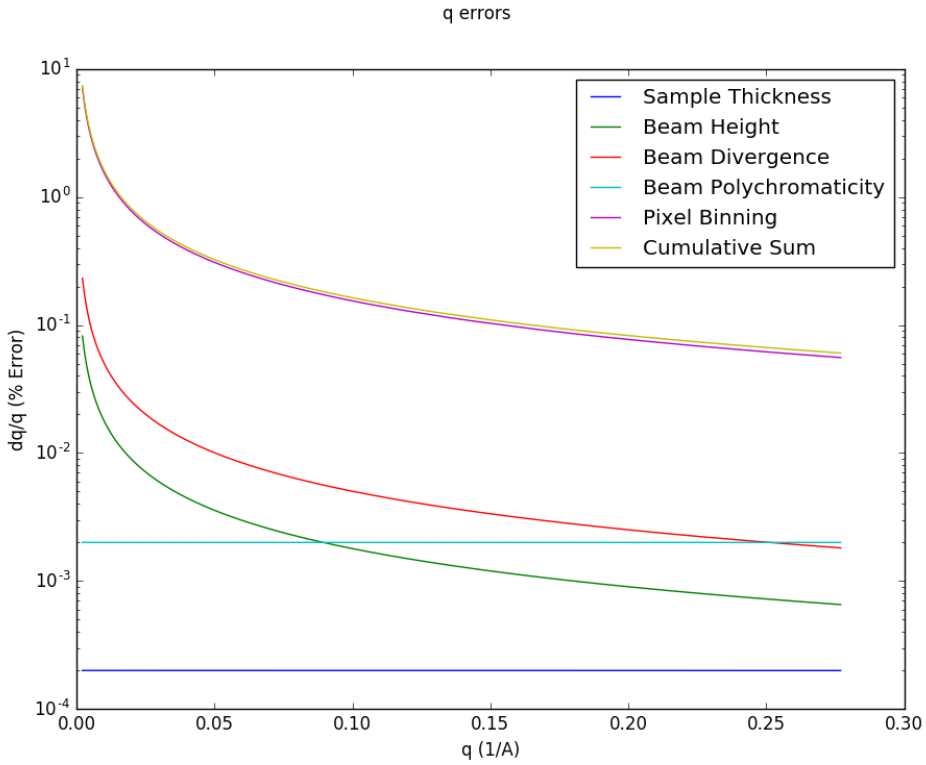
Mathematically the following was performed to generate the q_{err} values from the q dataset.

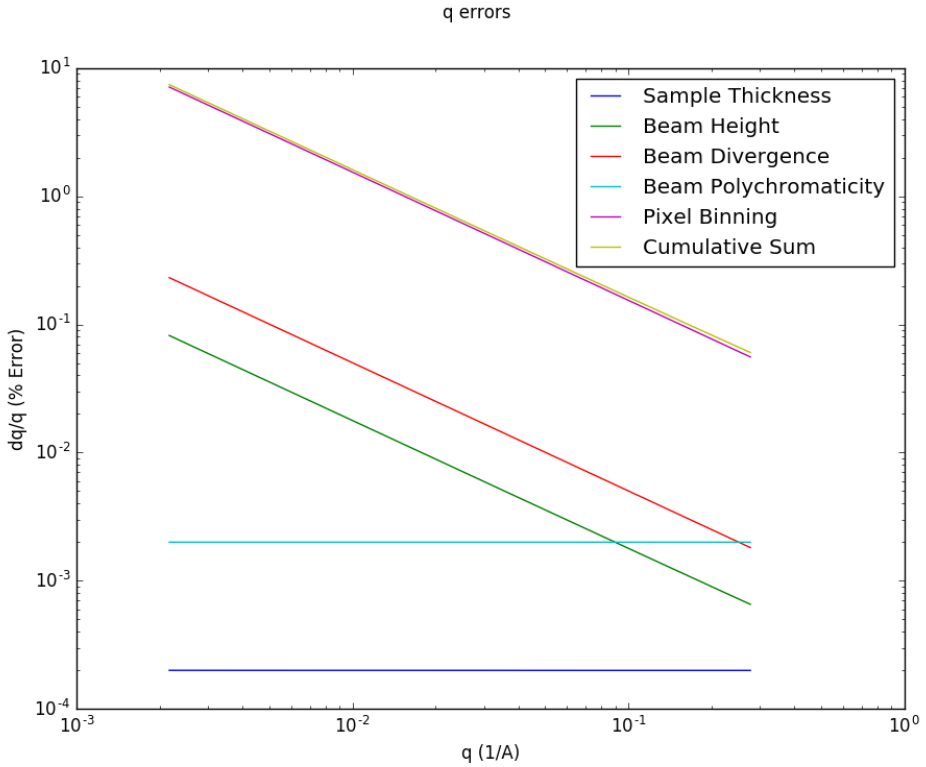
$$\Delta q_{x+1} = q_x - q_{x+1}$$

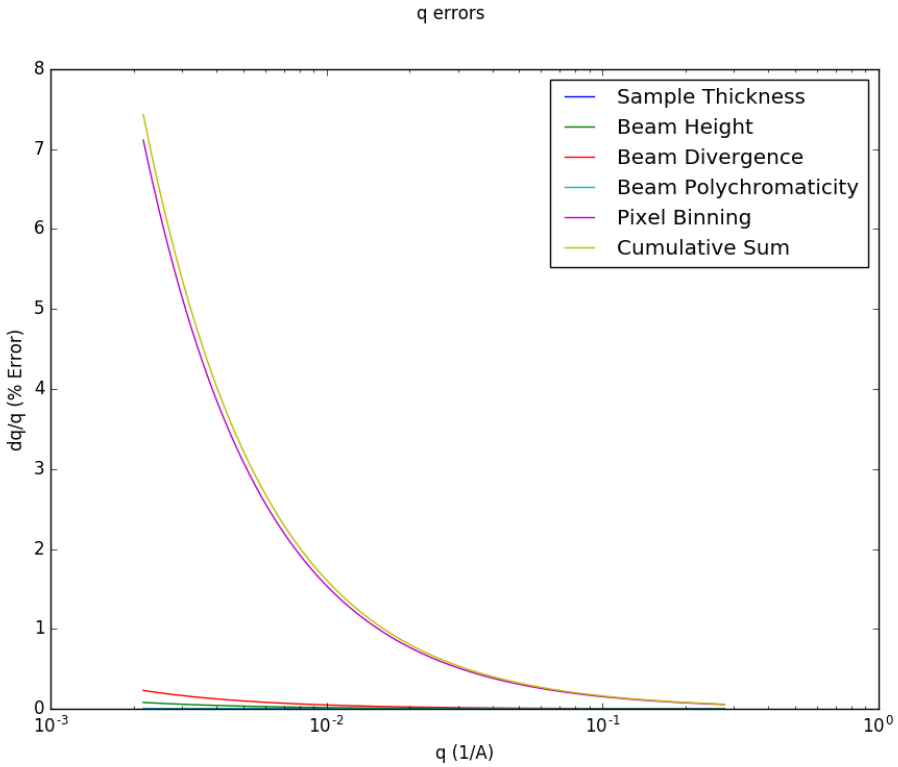
$$\Delta q_0 = \Delta q_1$$

$$q_{err} = \frac{\Delta q}{q} \times 100$$

Again, with all other traces plotted alongside the pixel binning


q errors
Sample Thickness
Beam Height
Beam Divergence
Beam Polychromaticity
Pixel Binning
Cumulative Sum
dq/q (% Error)
q (1/A)


q errors
Sample Thickness
Beam Height
Beam Divergence
Beam Polychromaticity
Pixel Binning
Cumulative Sum
dq/q (% Error)
q (1/A)


q errors
Sample Thickness
Beam Height
Beam Divergence
Beam Polychromaticity
Pixel Binning
Cumulative Sum
dq/q (% Error)
q (1/A)
0
1
2
3
4
5
6
7
8
10^{-3}
10^{-2}
10^{-1}
10^{0}