

Experience with Data Analysis Programs: DAWN, DPDAK and Mantid

Gero Flucke

FS-EC



PETRA III Commissioning Meeting
April 2nd, 2014

Outline

- Introduction
- Overview of Analysis Programs:
 - DAWN
 - DPDAK
 - Mantid
- Summary

Introduction

- Data rates in modern X-ray experiments are rising.
 - ⇒ Raw data might not be “exportable” to users’ institutes.
- Groups with little X-ray physics background less experienced with “standard” data analysis.
- Standardisation of data format envisaged: NeXus.
- How to react at DESY?
 - ⇒ Provide central data processing?
 - ⇒ Provide “standard” data analysis tools?

⇒ Investigate existing analysis tools!

Note: Within the framework of the
Photon and Neutron data infrastructure initiative.

<http://pan-data.eu>

Investigation of Analysis Programs

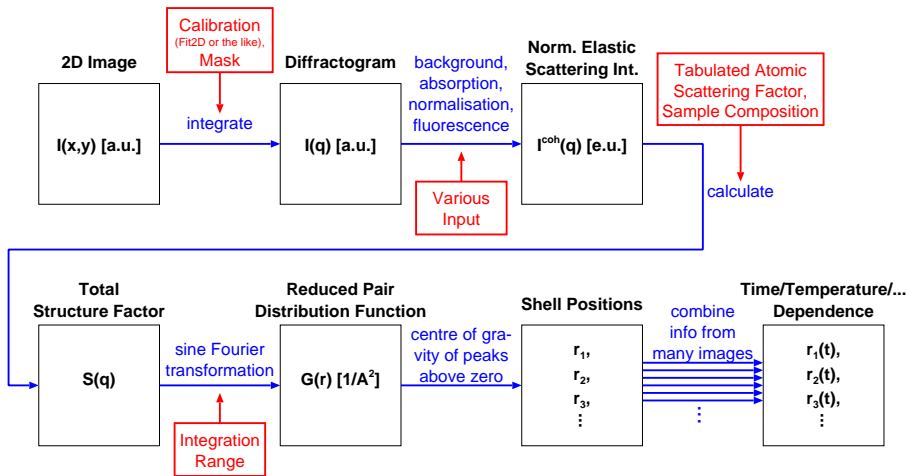
- Candidates: DAWN, DPDAK, Mantid.
- See what the programs provide,
 - but need analysis guidance,

⇒ example scenario:
Real space analysis of amorphous material,

 - thanks to J. Bednarcik (P02.1) for data and beam time!
- All programs share Python as analysis scripting language:
 - ⇒ write/(re)use tool independent code base in all programs!
- Notes:
 - My physics background is outside X-ray analysis, being familiar with ROOT (<http://root.cern.ch>) for analysis,
 - I spent more time on DAWN than on the other programs.

Real Space Analysis of Amorphous Material

- Changing conditions: temperature, pressure,...
- Capable of online monitoring.



DAWN: Data Analysis WorkbeNch

- Eclipse based program \Rightarrow written in Java (requires Java 1.7).
- “Implements sophisticated support for:
 - Visualization of data in 1D, 2D and 3D,
 - Python script development, debugging and execution,
 - Workflows for analyzing scientific data calling Python and binary codes.”
- Main developers from Diamond and ESRF.
- Open source: EPL or Apache license (getting rid of (L)GPL).
- Open to external collaborating developers.
- “By and for the synchrotron community - overlap with other communities like neutron scattering, photon science, etc.”
- Supported platforms:
Windows, Mac OS (beta version only), Linux.

<http://www.dawnsci.org>

<http://hasyweb.desy.de/services/computing/dawnDocu>

Concept

- GUI is based on “Perspectives” and “Views”.
- View: tab in a frame of the DAWN window, e.g.
 - a plot, contents of a file/directory, a tool...
- Perspective: groups views suitable for some task, e.g.
 - data browsing, python scripting, . . .
 - user can add or remove views.
- User’s data organised in “Projects”.
- Projects and status stored in “Workspace”
(default: `$HOME/workspace`).
- Many “Cheat sheets”: built-in tutorials.
- Feedback button – tell the developers
 - what does not work,
 - what would be nice to have.

Note: mostly tested with version 1.4.0 - there is 1.4.1 and soon 1.5.0.

Many Interactive Features

- Browsing many data formats: Nexus/HDF5, images, .edf, ascii,fio (for DESY version)
- Investigate multi-D datasets:
 - 1D curves: value vs index or vs value of other dataset
 - 'value' can be expression of datasets.
 - stack of lines, stack in 3D
 - slices of multi-D datasets,
 - 2D: as surface, as image (featureful colour mapping [for non-RGB images], pixel 'averaging')
 - various nice zoom/panning features,
- Masking: creation, storage, automatic application. . .

Many Interactive Features (ctd.)

- Distribution fitting
 - finding and fitting peaks,
 - fitting polynomials,
 - arbitrary function fitting not working (but will be in 1.5),
 - no fitting of 2D.
- Region of interest and profiles: line, box, sector, . . .
- Specific scientific methods:
 - calibration for 2D \rightarrow 1D integration (Fit2D) will be in 1.5,
 - many methods go as perspectives into Diamond version (ESRF?).
- Time needed to find out how things work (as to be expected).
- Things do not always work as they should,
 - but feedback highly appreciated by development team!

Guided Tour through DAWN: First Screen

Data Browser - DAWN Science - /tmp/dawnWS3

File Edit Window Help

Welcome

Welcome to DAWN

- First Steps**
Take your first steps
- Data**
Create data samples
- What's New**
Find out what is new
- Overview**
Get an overview of the features
- Perspectives**
List of available perspectives
- Tutorials**
Go through tutorials
- Web Resources**
Read more on the Web

DAWN
SCIENCE

DAWN: Image Browsing

The screenshot displays the DAWN software interface. The main window shows a 2D plot of 'image-01' with a grayscale color bar on the right. The plot has x and y axes ranging from 0 to 2000. The data table on the right lists the image's shape as [2048, 2]. Below the table, there is a section for creating a slice of the image, with a table showing slice variables and axis data.

Image Info

Name	Shape
<input checked="" type="checkbox"/> image-01	[2048, 2]

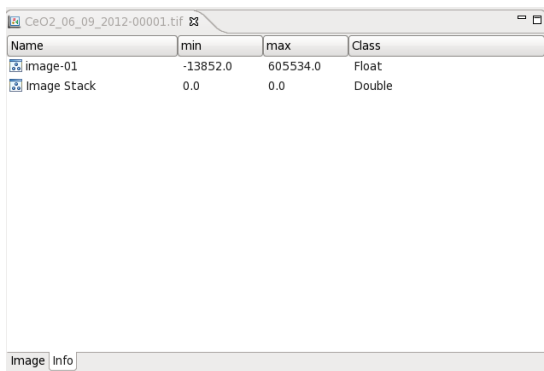
Create a slice of image-01.
It has the shape [2048, 2048]

D	Type	Slice Value	Axis Data
1	Y		indices
2	X		indices

Image Orientation: Top left

2040.839, 1689.958

DAWN: Image Browsing

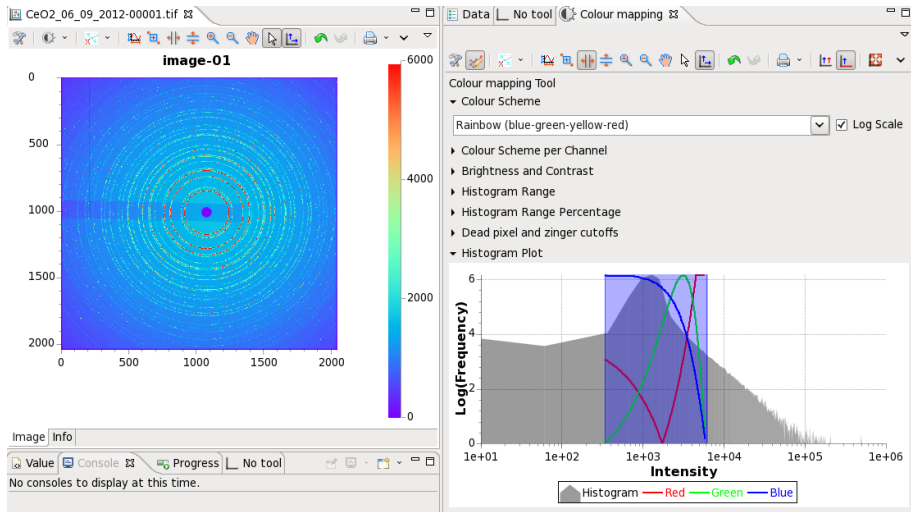


The screenshot shows a software window with a title bar containing the filename 'CeO2_06_09_2012-00001.tif'. Below the title bar is a table with four columns: 'Name', 'min', 'max', and 'Class'. The table contains two rows of data. The first row is 'image-01' with a minimum value of -13852.0, a maximum value of 605534.0, and a class of 'Float'. The second row is 'Image Stack' with a minimum value of 0.0, a maximum value of 0.0, and a class of 'Double'. At the bottom of the window, there are two tabs: 'Image' and 'Info', with 'Info' currently selected.

Name	min	max	Class
image-01	-13852.0	605534.0	Float
Image Stack	0.0	0.0	Double

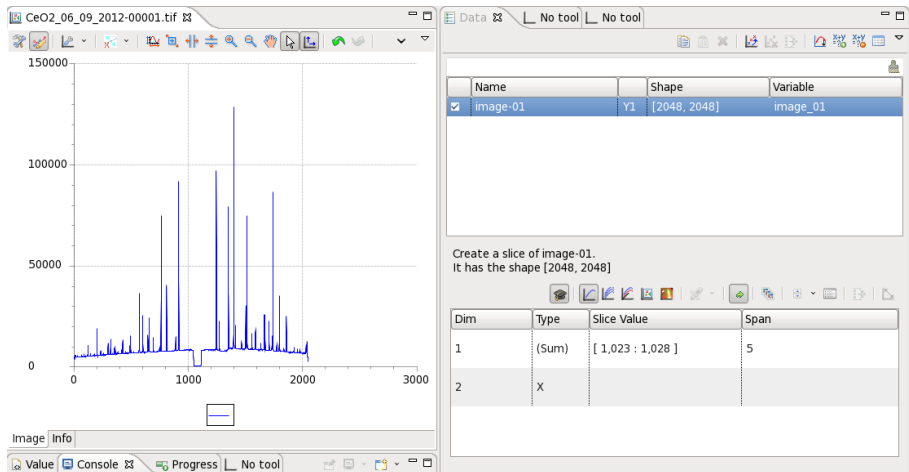
Easily check range of values in (here: image) dataset.

DAWN: Colour Scale



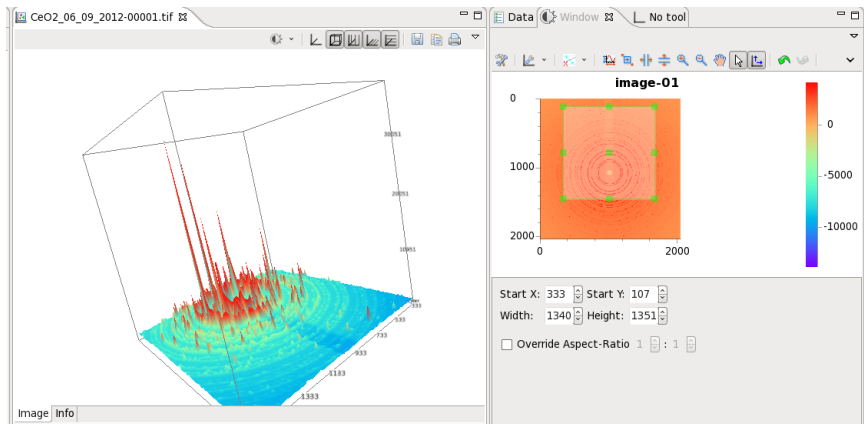
Full interactive control of colour mapping and outliers to ignore.

DAWN: Slice Datasets



Here: Sum of 5 consecutive slices.

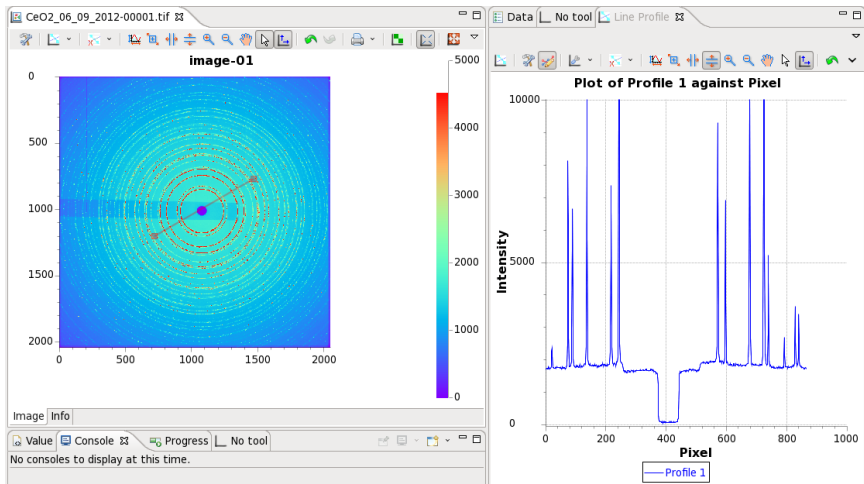
DAWN: Surface Plot



Live 3D view (instantly updated):

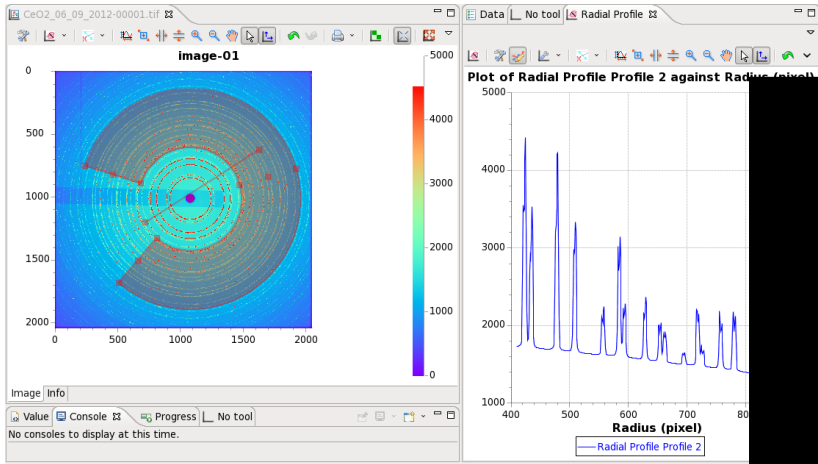
- Rotate as you want.
- Select a box

DAWN: Line Profile



1D plot updated instantly if mouse moves line.

DAWN: Radial Profile



Other profiles available as well, e.g. azimuthal profile.

DAWN: Masking

image-01

Masking 'image-01'

Create a mask, the mask can be saved and available in other tools.

- Enable lower mask 286
- Enable upper mask 4511
- Keep pixels already masked

Mask Color

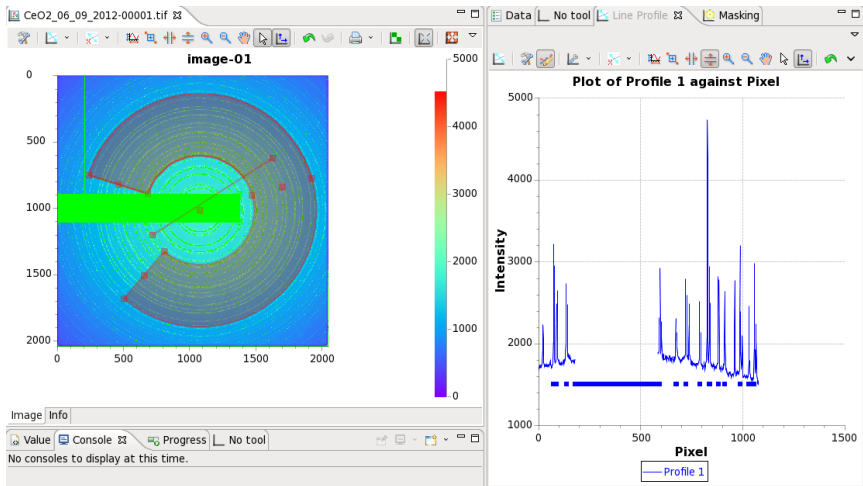
Mask using: Direct draw Regions

Mask	Name	Type
<input type="checkbox"/>	Profile 1	Line
<input type="checkbox"/>	Profile 2	Sector
<input checked="" type="checkbox"/>	Box 1	Box

Automatically apply mask when something changes.

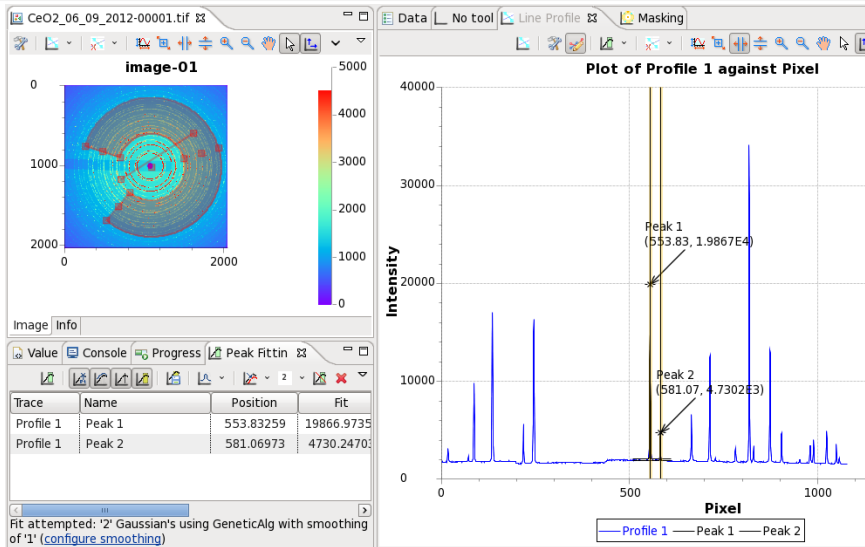
Choose regions, upper/lower masks, 'by hand',...

DAWN: Line Profile Masked



Mask taken into account in profiles.

DAWN: Peak Fitting



Interactively define range and maximum number of peaks to be fitted

(can be many!)



DAWN: HDF5/NeXus Browsing

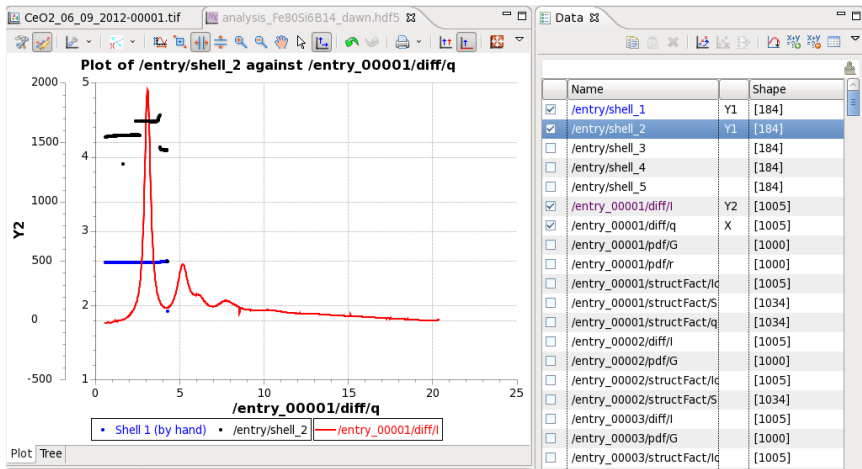
The screenshot shows the DAWN software interface with two main panes. The left pane, titled 'analysis_Fe80Si6B14_dawn.hdf5', displays a tree view of the file structure. The right pane, titled 'Data', displays a list of selected entries with their names and shapes.

Name	Class	Dims	Type	Data
entry	Group			
shell_1	SDS	184	FLOAT64	double-click to view
shell_2	SDS	184	FLOAT64	double-click to view
shell_3	SDS	184	FLOAT64	double-click to view
shell_4	SDS	184	FLOAT64	double-click to view
shell_5	SDS	184	FLOAT64	double-click to view
entry_00001	Group			
diff	Group			
l	SDS	1005	FLOAT32	double-click to view
q	SDS	1005	FLOAT64	double-click to view
pdf	Group			
G	SDS	1000	FLOAT64	double-click to view
r	SDS	1000	FLOAT64	double-click to view
structFact	Group			
lcoherent	SDS	1005	FLOAT64	double-click to view
S	SDS	1034	FLOAT64	double-click to view
norm	SDS	1		0.40231265
q	SDS	1034	FLOAT64	double-click to view
entry_00002	Group			
entry_00003	Group			

Name	Shape
<input type="checkbox"/> /entry/shell_1	[184]
<input type="checkbox"/> /entry/shell_2	[184]
<input type="checkbox"/> /entry/shell_3	[184]
<input type="checkbox"/> /entry/shell_4	[184]
<input type="checkbox"/> /entry/shell_5	[184]
<input type="checkbox"/> /entry_00001/diff/l	[1005]
<input type="checkbox"/> /entry_00001/diff/q	[1005]
<input type="checkbox"/> /entry_00001/pdf/G	[1000]
<input type="checkbox"/> /entry_00001/pdf/r	[1000]
<input type="checkbox"/> /entry_00001/structFact/lcoherent	[1005]
<input type="checkbox"/> /entry_00001/structFact/S	[1034]
<input type="checkbox"/> /entry_00001/structFact/q	[1034]
<input type="checkbox"/> /entry_00002/diff/l	[1005]
<input type="checkbox"/> /entry_00002/pdf/G	[1000]
<input type="checkbox"/> /entry_00002/structFact/lcoherent	[1005]
<input type="checkbox"/> /entry_00002/structFact/S	[1034]
<input type="checkbox"/> /entry_00003/diff/l	[1005]

Easy browsing of HDF5/NeXus file content.
(But it does not tell you what is a link...)

DAWN: HDF5/NeXus Browsing



Simple selection what to draw as X, Y1 and Y2.
(Non-default line styles a bit more complicated to achieve.)

DAWN: Python - Scripting Interface for Users

- Interactive prompt (using IPython).
- Nice script development environment (Eclipse and PyDev)!
- (C)Python: usual Python (“reference implementation”), implemented in C.
- Jython: Same language, implemented in Java.

(C)Python in DAWN

- Access to plotting, file access for various formats.
- Use standard python tools - e.g. for function fitting.
- Access to DAWN Java code possible via `py4j` (expert action).

Jython in DAWN

- Access to plotting, file access for various formats, fitting.
- No standard (C)Python tools available!
- Simple to use any DAWN Java code?

DAWN: Python Development

The screenshot displays the PyDev IDE interface with the following components:

- Left Panel (Project Explorer):** Shows the project structure for 'realspace', including files like 'ASF.DAT', 'compareHDF5.py', 'imageanalysis.py', 'integrationTests.py', 'monitor_base.py', 'monitor_Cu5Zr35', 'monitor_down.py', 'monitor_Fe80Si6B', 'realspace_monitor', 'realspace.py', 'status.txt', 'python', 'realSpaceWorkflow', 'testRingData', and 'workflows'.
- Editor (structureFactorTests.py):** Contains Python code for testing pyFAI integration. Key lines include:

```
print 'diffDiff:', diff.dot[diff]
allSame = numpy.array_equal(resultPyFAInoCor[1], result)
print 'arrays are identical:', allSame
print 'Done testPyFAI'

def testMyPyFAI(fileNameIn = '/home/flucke/fsec/P02/structureFactorTest/ASF.DAT',
                fileNameRes0 = '/home/flucke/fsec/P02/structureFactorTest/structureFactorTest001.res',
                fileNameRes1 = '/home/flucke/fsec/P02/structureFactorTest/structureFactorTest002.res',
                centre = (1030.91, 1022.444, 'fit2d'),
                tiltDeg = 0.477222, rotDeg = 74.30787,
                dist = 0.387752, waveLen = 0.123984, pixelSize = (30., 990.),
                maskFile = '/home/flucke/fsec/P02/structureFactorTest/maskFile.txt'):
    """
    Compare my integration using pyFAI with my own
    """
    fit2DqsPerInvtm = numpy.loadtxt(fileNameRes0)
    fit2DIntens = numpy.loadtxt(fileNameRes1)
    image = dnp.io.load(fileNameIn)[0]
    mask = dnp.io.load(maskFile)[0]
    calc = rs.ImageToID(centre, tiltDeg*math.pi/180., rotDeg*math.pi/180., dist, waveLen, pixelSize)
```
- Outline Panel:** Lists variables defined in the script, such as 'rs = realspace', 'fit2DqsPerInvtm', 'fit2DIntens', 'image', 'mask', and 'calc'.
- Dataset Plot:** A line plot titled 'are diffractogram vs q [1/nm], Jozef, my pyfai'. The Y-axis ranges from 0 to 800,000, and the X-axis ranges from 0 to 30. The plot shows a sharp peak at approximately q=2. A legend at the bottom identifies 'Plot 0' (blue), 'Plot 1' (red), and 'Plot 2' (magenta).
- Console:** Shows the execution output, including warnings and errors:

```
<terminated> /home/flucke/workspace/realspace/src/structureFactorTests.py
WARNING:root:Exception No module named fftw3: FFTw3 not available. Falling back on Scipy
WARNING:pyFAI.opencil:Unable to import pyOpenCL. Please install it from: http://pypi.python.org/pypi/pyopencl
WARNING:pyFAI.azimuthalIntegrator:Unable to import pyFAI.ocl_azim
ERROR:pyFAI.azimuthalIntegrator:Unable to import pyFAI.ocl_azim_lut for Look-up table based azimuthal integration on GPU
No dark image subtraction.
No flat field correction.
Before _pyFAI_a1.integrateId
After _pyFAI_a1.integrateId
Done integrate2ThetaPyFAI with 1035 2theta bins
Done testMyPyFAI
```


- Idea:
 - graphically compose standard components (“actors”),
 - to form a new analysis/data reduction procedure.
- Actors for
 - request user input,
 - open files or monitor directories,
 - flow control,
 - mathematics operations or execute python code,
 - plot or review plots,
 - export to files,.. .
- Can be run in batch mode, parallelised using threads.
- Interesting to use for monitoring procedure,
 - but I failed: little docu, not (yet) flexible enough, bugs,.. .

DAWN: Workflow Example

The screenshot displays the DAWN workflow editor interface. The main workspace contains a workflow diagram with the following components and connections:

- Inputs:** Input_structFact, Input_bg, Input_calib, Input_filedir.
- Processing:** A Combiner actor receives inputs from Input_structFact and Input_bg. Its output goes to a Folder Monitor, which then feeds into a File Import actor.
- Image Processing:** The File Import actor outputs to an ImageTo1D actor.
- Mathematical Operations:** The ImageTo1D actor outputs to a StructureFactor actor (represented by a circle with $x+y$), which then feeds into an rPDF actor (also a circle with $x+y$).
- Control and Output:** The rPDF actor outputs to a Message actor (circle with a warning icon), which then feeds into a Plot Review actor. The StructureFactor actor also has a direct output path to a Plot Review actor.
- Final Output:** The Plot Review actor outputs to a final Plot actor.

The interface includes a Palette on the left with categories like Select, Marquee, Connection, Favorites, and various actor types (Input, Scalar, Combiner, If, Python PyDev, Message, Plot Image, File Import, Folder Monitor). The right side features a Monitor and Images Monitor panel with a type filter and a hierarchical tree of actor categories. The bottom of the window shows the Actor Attributes panel for the selected ImageTo1D actor.

Property	Value
Name	ImageTo1D
Create Separate Interpreter	<input type="checkbox"/>
Dataset Outputs	q, l
Interpreter	realSpaceWorkflow's PyDev interpreter settings - /usr/bin/python
Pass Inputs On	<input checked="" type="checkbox"/>

DAWN (1.5.0 beta): Diffraction Calibration

File Edit Window Help

Diffraction Calibration View

Drag/drop a file/data to the table below, choose the calibrant, select the rings to use and finally run the calibration.

Image

CeO2_06_09_2012-00001.tif

* Click to change value

Select calibrant:

CeO2

Show Calibrant and Beam Centre

Select rings to use for calibration:

Rings to use (from inner): 26

Select specific ring numbers:

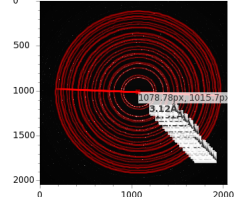
Auto Manual Settings

Finish with point calibration optimisation

Residual: 3.5687110769283E-6

Diffraction Plotting

CeO2_06_09_2012-00001.tif



1078.78px, 1015.7px
3.12Å

Powder Diffraction

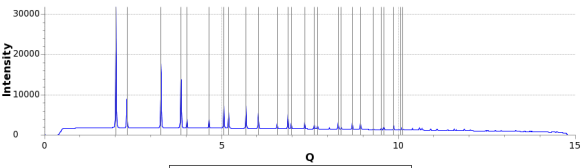
type filter text

Name	Value	Unit
Experimental Information		
Wavelength	0.2907 *	Å
Distance	353.31 *	mm
Detector		
Type	Perkin Elmer 1	
Exposure Time	1	s
Size		
Pixel		
Beam Centre		
X	1078.78 *	pixel
Y	1015.7 *	pixel
Normal (Orientation)		
Yaw	-0.52 *	°
Pitch	0 *	°
Roll	-177.6 *	°

* Click to change value

Powder Calibration Check

Plot of CeO2_06_09_2012-00001.tif:image-01_integrated against q



Intensity

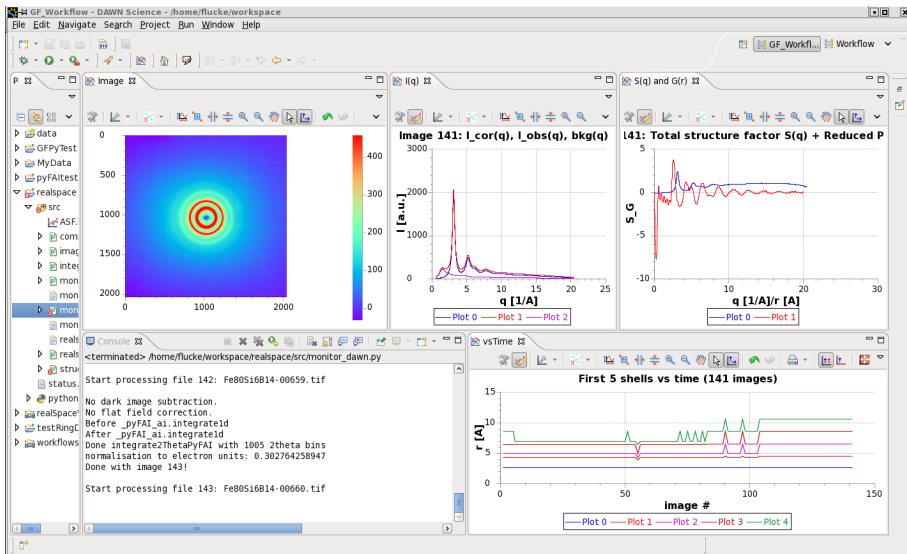
Q

CeO2_06_09_2012-00001.tif:image-01_integrated

DAWN: Real Space Analysis/Monitoring

- Use DAWN basically only for
 - Python code development,
 - graphics output.
- Configuration using configuration file.
- Monitoring directory for new images: “borrow” DPDAK’s concept.
- Storing results on HDF5 file using `h5py`.
- Arrangement of “Views” stored as new “Perspective”.
- Miss mechanism to “softly” stop procedure,
 - needed to close HDF5 file properly.
- Control of plot appearance via Python very limited:
 - cannot change colour/style of lines, titles, legend, . . .
 - no way (yet) to set proper legend entries
- Plotting acceptably fast (required fix from DAWN developers).

DAWN: Real Space Analysis/Monitoring



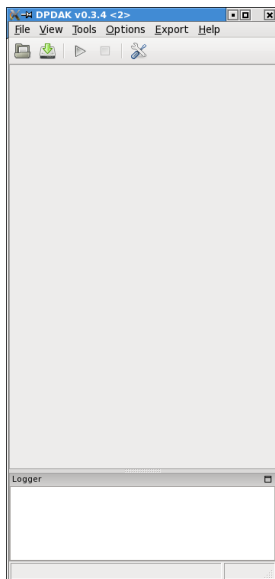
DAWN: Pros and Cons

- Flexible multi-dimensional data display and interactive analysis,
 - slicing, masking, profiling, fast 3D plots.
 - Browsing file content, e.g. HDF5.
 - Python as scripting language
(built-in development environment).
 - Large support from Diamond and ESRF.
 - OSGI plugin-based design is modular and thus flexible.
-
- Java: difficult to import “external” algorithms.
 - Fitting arbitrary functions not working yet.
 - Still bugs hitting the user.
 - Limited interface from Python to core functionalities
(besides expert way via `py4j`).
 - OSGI difficult to pick up until you understand it.

- Pure Python program using ‘standard’ Python packages:
 - NumPy, SciPy, matplotlib, fabio, wxPython.
- Developed in cooperation between DESY and MPIKG,
 - for “(online) analysis of 2D scattering data”,
 - standard tool at P03 (S. Roth) and for its users,
 - author G. Benecke left end of January.
- Supported platforms: Windows, Linux.
- Open Source, GNU GPL.

<https://dpdak.desy.de>

DPDAK: Concept

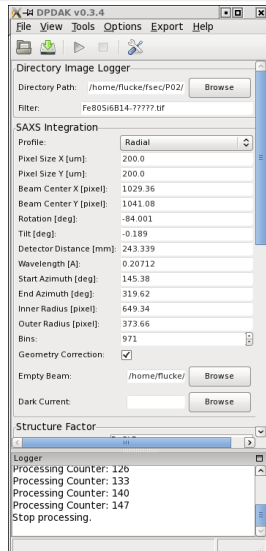
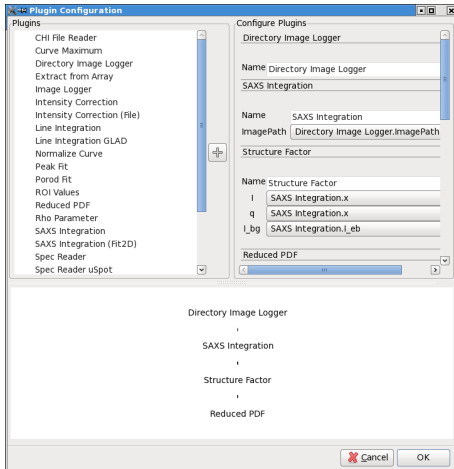


- Minimalistic start-up GUI.
- Select and configure 'plugins':
 - data processing steps,
 - tools (data display etc.),
 - data export.
- Interfaces of processing plugins:
 - input and output types, parameters,
 - types: scalars, 1D arrays, strings, file/directory paths.
- Processed data stored in 'database':
 - no check if data always identical,
 - for images just their paths,
 - interactive analysis via tools.
- General image options:
 - rotation, axis flipping, background.

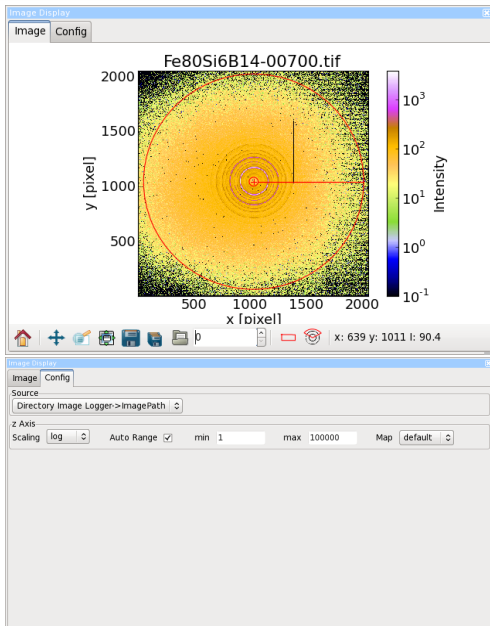
DPDAK: Configure Data Processing Plugins

- Select plugins from list.
- Select (matching) input.

- Set parameters.

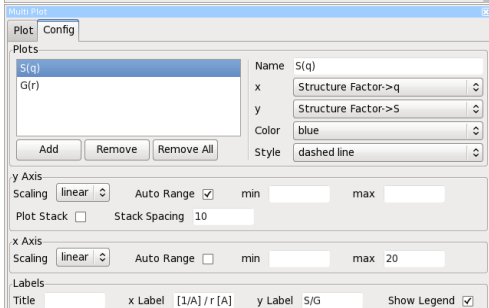
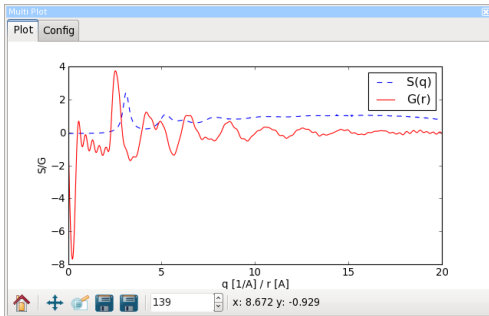


DPDAK: Image Display



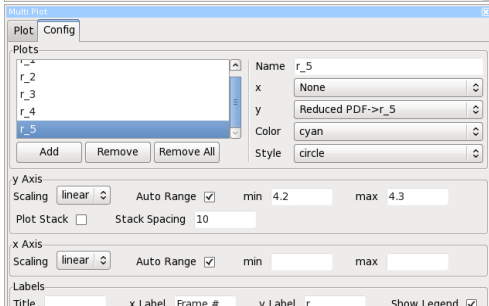
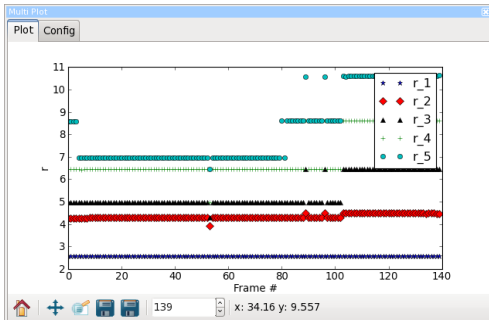
- Choose from database of processed images (broken in v0.3.4, showing always the last?).
- Directly select file.
- Range of 2D \rightarrow 1D integration shown - and editable!

DPDAK: 1D Plots



- Select x and y from output of plugins.
- Many lines can be overlaid.
- Choose result of which processed image to be shown.
- Stack of all lines also possible (broken in recent version 0.3.4?).

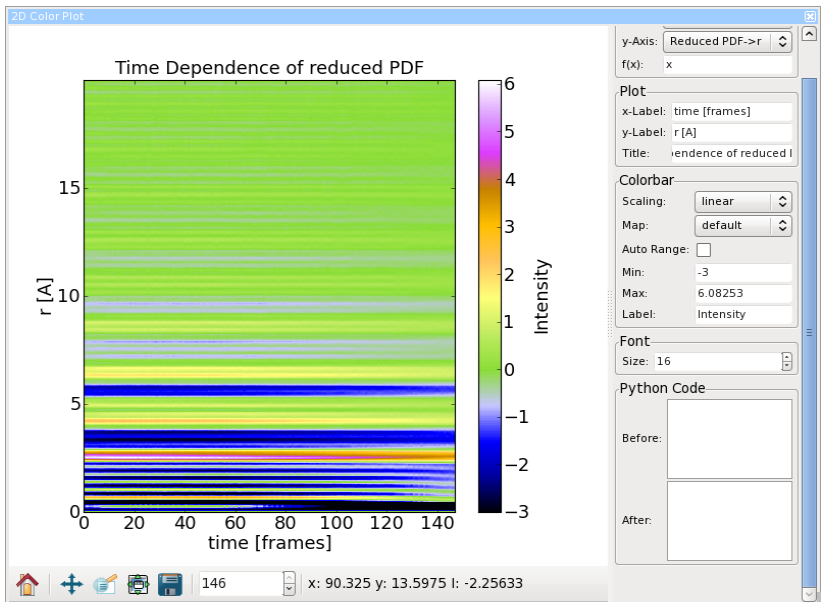
DPDAK: 1D Plots (ctd.)



No x axis chosen:

- y data is 1D array: array plotted vs its index.
- y data is scalar: scalar plotted vs frame number (can choose up to which frame number).
- Legend position static (but can be switched off).

DPDAK: 2D Plot of Distribution vs Time

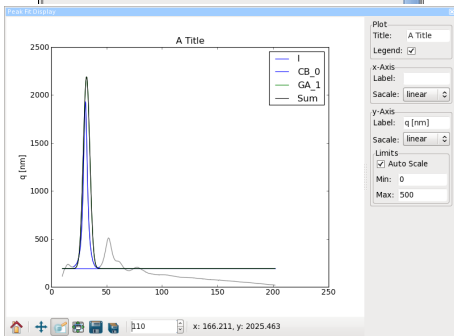
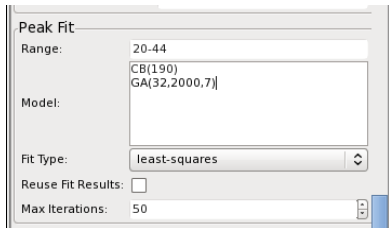


Processing Plugin to Fit

- (sum of) predef. functions, backgr.: constant, linear, peaks: Gauss, Lorentz, Pseudo-Voigt,
- restrict fit range,
- (possibly fixed) start values.

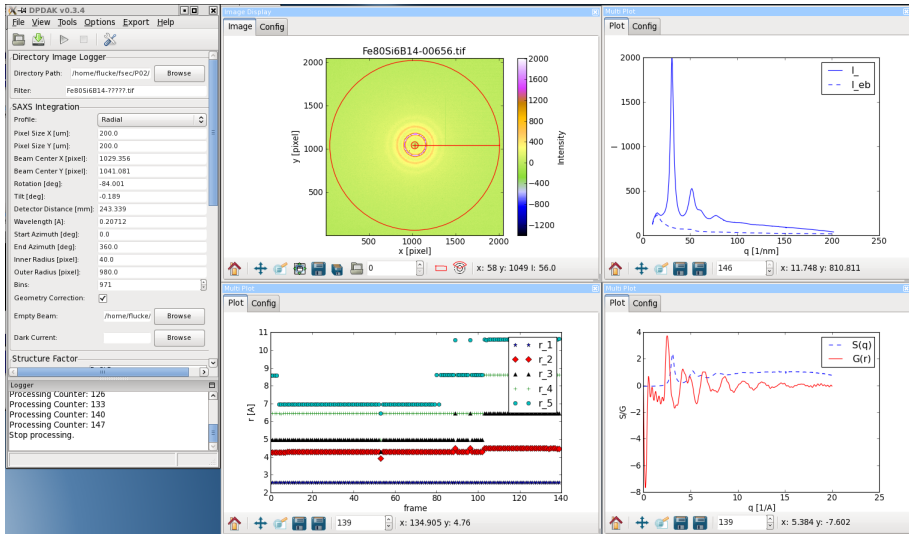
Peak Fit Display Plugin

- Original distribution (highlight fit region).
- Fitted curve and its components.



- DPDAK designed for that purpose:
 - add data processing plugins,
 - configure (including display plugins),
 - run!
- Already there:
 - directory monitoring,
 - 2D \rightarrow 1D integration,
 - storage of results as DPDAK 'database'
(can be exported to ascii - completely or per plugin).

DPDAK: Real Space Analysis/Monitoring



DPDAK: Pros and Cons

- Built to be extendible by user,
 - and fulfils that promise (within boundaries...)!
 - Rather simple interface for new plugins.
-
- How to handle multi-dimensional datasets?
 - Does internal database scale with 10 thousands of images?
 - Some intransparent shortcuts.
 - Display plugins do not store configuration (could be added).
 - Limited core support now that G. Benecke left.

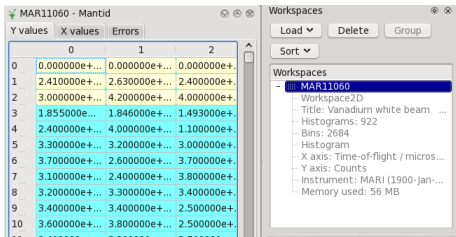
- Framework for “high-performance computing and visualisation of scientific data”.
- To “manipulate and analyse Neutron and Muon scattering data, but could be applied to many other techniques.”
- Main developers from ISIS and SNS.
- Written in C, C++ and Python (using scientific visualisation software “QtPlot”).
- Open source: GNU General Public License.
- Each release can be cited with a digital object identifier.
- Supported platforms: Windows, Linux, Mac OS.

<http://www.mantidproject.org>

Mantid: Concepts

- Data processing via **algorithms** working on **workspaces**.
- Once data loaded into a workspace
 - investigate interactively: 1D, 2D, 3D plots, 1D fits, slices
 - or call algorithms.
- Store status (workspace, plots, . . .) as 'project'.
- Smooth integration of Python as scripting language:
 - access algorithms,
 - graphics control,
 - extend Mantid by Python algorithms and fit functions,
 - workspaces keep track of their history - as Python script,
 - all but graphics available also outside MantidPlot GUI.
- Visualisation of instruments.
- Good documentation for beginners (GUI plus Python).

Mantid: Workspaces



The screenshot shows the Mantid software interface. On the left, a table displays data for workspace 'MAR11060'. The table has columns for 'Y values', 'X values', and 'Errors'. The rows are indexed from 0 to 11. On the right, a 'Workspaces' panel shows a list of workspaces, with 'MAR11060' selected. The panel includes buttons for 'Load', 'Delete', and 'Group', and a 'Sort' dropdown menu.

	0	1	2
0	0.000000e+...	0.000000e+...	0.000000e+...
1	2.410000e+...	2.630000e+...	2.400000e+...
2	3.000000e+...	4.200000e+...	4.000000e+...
3	1.855000e+...	1.846000e+...	1.493000e+...
4	2.400000e+...	4.000000e+...	1.100000e+...
5	3.300000e+...	3.200000e+...	3.000000e+...
6	3.700000e+...	2.600000e+...	3.700000e+...
7	3.100000e+...	2.400000e+...	3.800000e+...
8	3.200000e+...	3.300000e+...	3.400000e+...
9	3.400000e+...	3.400000e+...	2.500000e+...
10	3.600000e+...	3.800000e+...	2.500000e+...
11	3.400000e+...	3.100000e+...	3.500000e+...

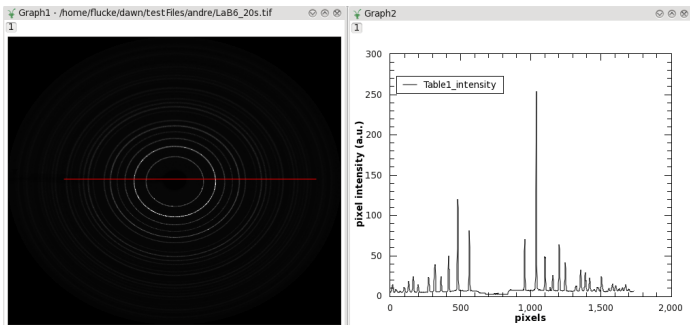
Most Common: Workspace2D

- Vertically: many “Spectra”.
- Horizontally: histogram/curve, tabs choosing
 - signal “Y” values,
 - their “Errors”,
 - “X” positions/bin borders.

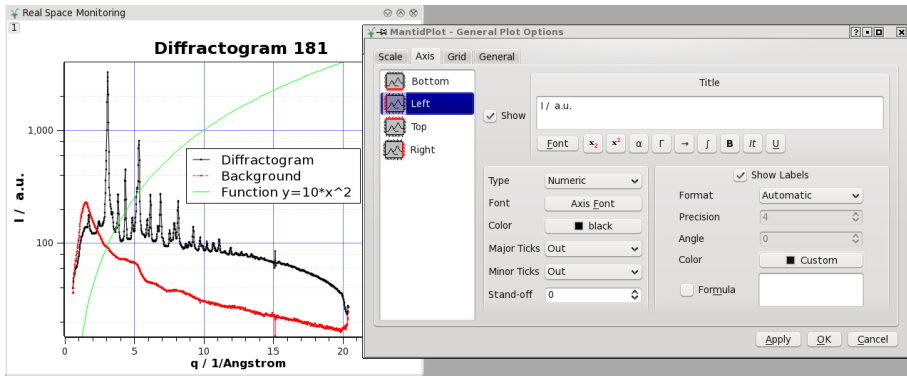
- EventWorkspace: as Workspace2D, keeping data unbinned.
- MDWorkspaces: Signal and error for coordinates of 1–9 dimensions.
- TableWorkspace: rows of columns of particular types (text, integer, ...).
- Several variants, e.g. file-backed, special purpose,

Mantid: Data Formats

- Generic “Load” algorithm chooses proper sub-algorithm.
- No generic loader to browse arbitrary HDF5 files.
- No loader for image files.
 - MantidPlot *can* open image files.
 - All I found: make a line profile.
 - Nothing to create a workspace from an image.



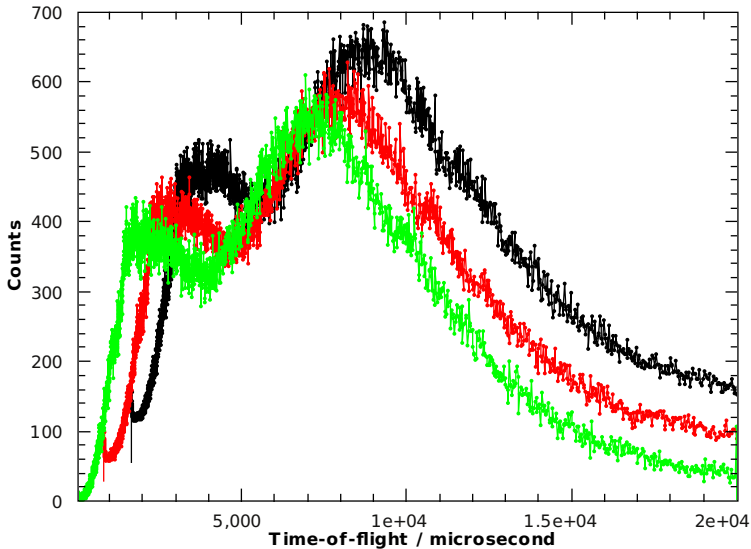
Mantid: 1D Plots



- Plotting needs input workspaces.
- Interactive control of plot appearance.
- Add (parametric) functions.
- May place plots next to each other in same window.

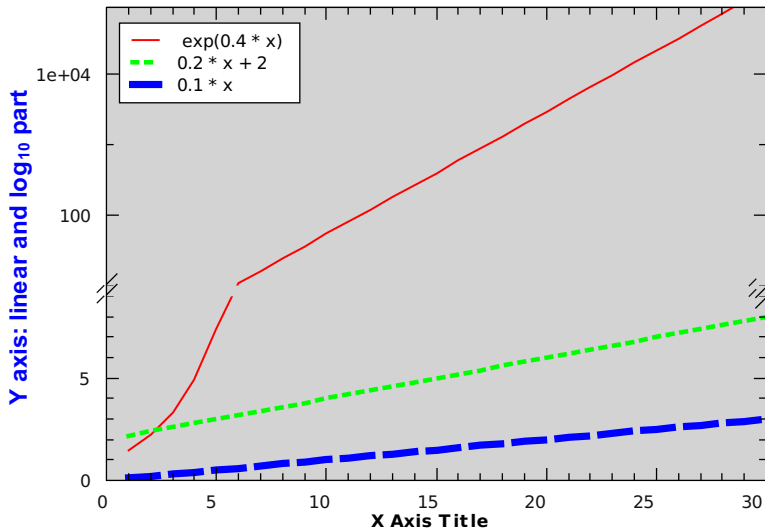
Mantid: 1D Waterfall

MAR11060

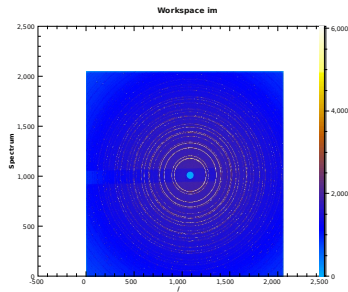
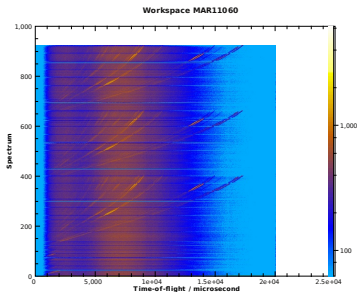


Mantid: Axis with Break

Plot with axis break

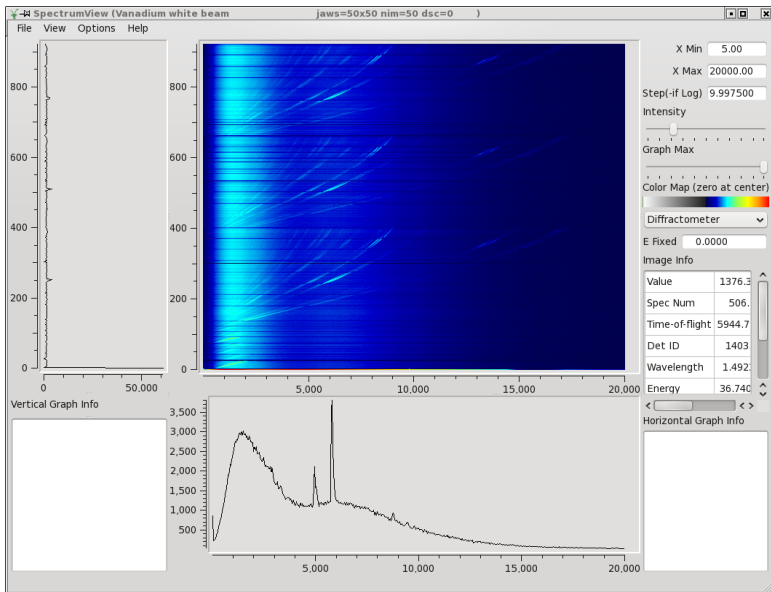


Mantid: “2D Colour Fill”



- Default colour scale does not exclude outliers.
- Also 3D views available (first display slow).

Mantid: Spectrum Viewer



Mantid: 1D Fitting

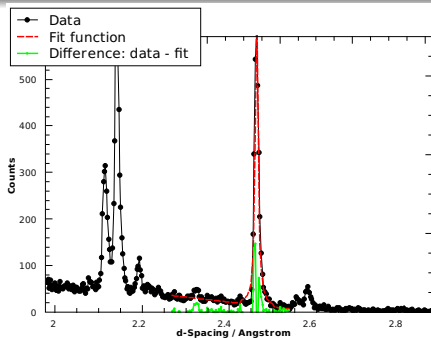


Table-1 - GEM38370_Focused_NormalisedCovarianceMatrix

Name[L]	f0.A0[Y]	f0.A1[Y]	f1.Height[Y]	f1.PeakCentre[f1.Sigma[
0 f0.A0	100	-99.9113	0.00387227	0.775206	4.36445
1 f0.A1	-99.9113	100	0.0112753	-0.744489	-4.85306
2 f1.Height	0.00387227	0.0112753	100	-9.86734	-60.0745
3 f1.PeakCentre	0.775206	-0.744489	-9.86734	100	13.4125
4 f1.Sigma	4.36445	-4.85306	-60.0745	13.4125	100

- Graphically choose fit range.
- Compose fit function: various background and peak types.
- Graphically choose start values of peaks.
- Graphical result includes difference “data minus fit” (and possibly individual components).
- Fit result and errors (incl. normalised covariance matrix) available as workspace.

Mantid: 1D Fitting (ctd.)

Fit Function (Chi-sq = 4.43327)

Fit Display Setup

Property	Value
- Functions	
Type	CompositeFunction
NumDeriv	<input type="checkbox"/> False
- f0-LinearBackground	
Type	LinearBackground
A0	275.711808
A1	-105.840393
- f1-Lorentzian	
Type	Lorentzian
Amplitude	7.885054
PeakCentre	2.462696
FWHM	0.007469
- Settings	
Workspace	GEM38370_Focussed
Workspace Index	4
StartX	0.013646
EndX	3.369098
Output	GEM38370_Focussed
Minimizer	Levenberg-Marquardt
Ignore invalid data	<input type="checkbox"/> False
Cost function	Least squares
Plot Difference	<input checked="" type="checkbox"/> True
Plot Composite Me...	<input type="checkbox"/> False
Convolve Composit...	<input type="checkbox"/> False

Many options

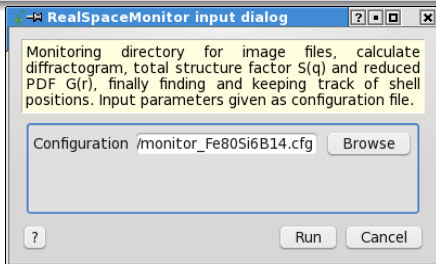
- Fit function composed of arbitrary amount of components.
- Rich set of predefined functions.
- Define your own function (e.g. implemented in Python).
- Tie parameter values to functions of other parameters.
- Fix parameters.
- ...

Mantid: Miscellaneous

- Masking? There is something I did not investigate.
- Poor Python editor (but can use external one).
- Python algorithms are not really object-oriented - problem:
 - CPU-intense initialisation,
 - two different configurations in one data processing chain.
- Cannot reset python terminal once I messed it up... (?)

Mantid: Real Space Analysis/Monitoring

- Make use of algorithm/workspace concept.
- Write algorithms in Python:
 - LoadImage, using `fabio`,
 - other using “common code base”,
 - Fourier transformation from $S(q)$ to $G(r)$ already in Mantid.
- (Ab)use Workspace2D even for images (waste of space: errors, “X” values).
- One master algorithm “RealSpaceMonitor”:
 - Configured via configuration file (not really Mantid style).



Master Algorithm RealSpaceMonitor

- Watching directory (“borrow” DPDAK’s logic again).
- Calling other algorithms.
- Write HDF5 output file (using `h5py`).
- Report progress (and catch cancellation to flush HDF5 file).
- Doing graphics output.
- Large control of line plots:
common window, axes, legend, ...
- 2D image plot in extra window:
 - no control of appearance from Python,
 - slow.

Mantid: Real Space Analysis/Monitoring

The screenshot displays the Mantid software interface with several panels:

- Results Log:** Shows the execution of various tasks: FindShells successful, AppendSpectra started and successful, Transpose started and successful, LoadImage started, and warnings about non-standard TIFF files.
- Real Space Monitoring:** Contains three plots:
 - Diffractogram 91:** A plot of intensity $I / \text{a.u.}$ versus $q / \text{1/Angstrom}$. It shows a sharp peak at $q \approx 4$ and a broader peak at $q \approx 8$. A legend indicates 'Diffractogram' (black line) and 'Background' (red line).
 - S(q) and G(r) 91:** A plot with two y-axes. The left y-axis is 'Structure Factor' (0 to 3) and the right y-axis is 'PDF' (-6 to 6). The x-axis is $q / \text{1/Angstrom}$ (0 to 25). It shows 'Struc. Fact: S(q)' (black line) and 'PDF: G(r)' (red line).
 - Shells (96 images):** A plot of $r / \text{Angstrom}$ versus 'Image number' (0 to 100). It shows multiple horizontal lines representing different shells, with some showing small peaks.
- Graph1:** A 2D heatmap titled 'Image 91' showing the 'Spectrum' (y-axis, 0 to 2,500) versus q (x-axis, -500 to 2,500). The plot shows a strong signal at $q \approx 4$.
- Workspaces:** A list of workspaces including 'allshellsWS', 'allshellsWS_trans', 'bgDiff', 'bgimageWS', 'dff', 'GofRWS', 'imageWS', 'shellsWS', and 'structFactWS'.
- Algorithms:** A list of algorithms including 'RealSpaceMonitor', 'DataProcessing', 'FindShells v.1', 'Integrate2Dto1D v.1', 'RealSpaceMonitor v.1', and 'StructureFactor v.1'.
- Algorithm progress:** A dialog box showing the progress of the 'RealSpaceMonitor' algorithm on file '96: fe8056b14-00613.tif' at 53% completion.

Mantid: Pros and Cons

- Deep integration of Python as scripting language:
 - can extend MantidPlot!
 - Lots of graphics control (interactively and via Python).
 - Flexible function fitting with error propagation.
 - Version control via doi.
-
- No built-in image handling.
 - Workspace concept may not be flexible enough,
 - but did not investigate too much the multi-dimension version.
 - Neutron and muon scattering driven:
 - set of available algorithms,
 - workspaces with signal (counts...) and error,
 - wording.

Summary

- All three programs have their own value.
- **DAWN** has the potential to cover standard needs of X-ray data analysis tasks, but suffers from limited extendability by users and (still) little bugs.
- **DPDAK** has proven itself in praxis and is extendible by design, but is not 'complete' and might fail for multi-dimensional data sets.
- **Mantid** provides great integration of user (Python) extensions, good control of plotting, and a professional fitting interface, but is currently driven by neutron/muon scattering needs.

Outlook

- Plan to exercise a SAXS analysis procedure as well.

What to Do to Test on Your Own?

DAWN, DESY Version (i.e. .fio Format Aware)

- Login on workgroup server `p3-wgs12`.
- Run

```
/scratch/DawnVanilla-1.4.1.v20131212-1003-linux64/startDawn.sh.
```

- Note that it redirects the workspace to
`/scratch/$USER/dawn/workspace`
(`$USER` is your login name).
- Or try a more recent, but probably less stable version that supports reading of `.fio` files even from CPython:

```
/scratch/DawnVanilla-1.5.0.v20140326-1353-linux64/startDawn.sh.
```

Mantid

- A bit more complicated - please contact me. . .

Please contact me (gero.flucke@desy.de) if you need help.

What to Do to Test on Your Own? (ctd.)

DPDAK - Windows

- Follow https://dpdak.desy.de/index.php/Install_dpdak_v0.3.2#Binary_Package.
- I did not test that. . .

DPDAK - Linux

- Login on workgroup server p3-wgs12.
- Download 'Source/Linux' from <https://dpdak.desy.de>.
- Put the *.tar.gz file into SOME/DIRECTORY.
- `cd SOME/DIRECTORY`
- `tar xzf dpdak-dpdak.tar.gz`
- `python dpdak/main.py`

Please contact me (gero.flucke@desy.de) if you need help.