PEARL Procedures

Igor procedures for the analysis of PEARL data

Version rev-distro-2.0.2-0-g3235d52

Sun Feb 10 2019 14:39:17

Contents

Chapter 1

Introduction

1.1 Introduction

PEARL Procedures is a suite of Igor Pro procedures developed for data acquisition and data processing at the PEARL beamline at the Swiss Light Source.

1.2 Installation

PEARL Procedures should be installed according to the regular Igor Pro guidelines. Please read the Igor help `About Igor Pro User Files` for details.

- Make a `pearl-procs` directory in your private or shared `User Procedures` folder, and copy the PEARL Procedures distribution there.
- Create shortcuts of the `pearl-arpes.ipf` and `pearl-menu.ipf` files, and move them to the `Igor Procedures` folder next to your `User Procedures` folder.
- Find the `HDF5.XOP` extension in the `Igor Pro Folder` under `More Extensions/File Loaders`, create a shortcut, and move the shortcut to the `Igor Extensions` folder next to your `User Procedures` folder.
- Find the `HDF5 Help.ihf` next to `HDF5.XOP`, create a shortcut, and move the shortcut to the `Igor Help Files` folder next to your `User Procedures` folder.

1.3 License Information

Author

Matthias Muntwiler, `matthias.muntwiler@psi.ch`

Version

This documentation is compiled from version rev-distro-2.0.2-0-g3235d52.

Copyright

Chapter 2

Angle-scan processing

2.1 Introduction

This page describes the data processing steps of angle-scans using the PEARL Procedures. The description relies on using the command line regardless of available GUIs.

2.2 Data reduction

The goal of this step is to import raw data and at the same time eliminate the energy dimension. We want a two-dimensional wave where the first dimension is the angle axis of the detector and the second dimension is the sequence of measurements, scanning one or multiple manipulator angles. The second dimension requires additional one-dimensional waves that describe the polar, tilt and azimuthal angle setting of the manipulator for each dimension index.

The processing steps depend on the complexity of the measured spectrum. The user may have to adopt one of the predefined or a custom procedure accordingly. Here, we describe two procedures that may cover many generic cases or that can serve as a starting point for a refined, customized procedure. However, any procedure that produces the datasets mentioned above is, of course, a valid approach. For instance, you could load the complete three-dimensional ScientaImage dataset, and generate the two-dimensional dataset using your own procedures.

2.2.1 Basic steps

The central import functions are psh5_load_reduced and psh5_load_dataset_reduced. The first form is sufficient if the file contains just one scan and region. Further regions/scans need to be loaded using the second form. The first form is also exposed in the PEARL data explorer window.

The functions require a data reduction function and processing parameters as arguments. Some particular reduction functions are described further below. More can be found in the source code (or obtained from other users). A list of functions that look like reduction functions can be got from adh5_list_reduction_funcs.

The basic call sequence looks as follows. Substitute the arguments in angle brackets as necessary. You may have to analyse a reference spectrum or the complete ScientaImage to figure out the processing parameters beforehand.

First form:

```
setdatafolder root:
string sparam
sparam = "<param1=1.5;param2=test;>"
psh5_load_reduced("<igor-datafolder>", "<igor-filepath>", "<filename>", <reduction_function>, sparam)
```

Second form:

```
// open the file
```

```
setdatafolder root:   // or other parent folder
variable fid
string sparam
fid = psh5_open_file("<igor-datafolder>", "<igor-filepath>", "<filename>")

// load metadata for scaling
psh5_load_scan_meta(fileID, "<scan 1>")
newdatafolder /s /o attr
psh5_load_scan_attrs(fileID, "<scan 1>")
setdatafolder ::

// load and reduce dataset
sparam = "<param1=1.5;param2=test;>"
psh5_load_dataset_reduced(fid, "<scan 1/region1>", "<ScientaImage>", <reduction_function>, sparam)

// close the file
psh5_close_file(fid)
fid = 0
```

2.2.2 Peak integration over linear background

The int_linbg_reduction function converts a two-dimensional Scienta image I(angle, energy) into a one-dimensional angle distribution I(angle). For each angle slice, it calculates a linear background. Then, it integrates the difference between the original data and the background over a specified interval.

The function requires the following, fixed parameters:

Parameter	Description	Typical value
Lcrop	size of the low-energy cropping region	0.11 (fixed mode)
Lsize	size of the low-energy background region	0.2
Hcrop	size of the high-energy cropping region	0.11
Hsize	size of the high-energy background region	0.2
Cpos	position of the peak center	0.5
Csize	size of the center region	0.3

All parameters are relative to the size of the image (length of the energy interval) and must be in the range from 0 to 1.

The cropping region is cut away from the image for the rest of the processing. This is necessary to remove the dark corners in fixed mode but can be neglected in swept mode (cropping size = 0).

The low and high background regions are adjacent to the cropping regions on either side. The function calculates two fix points of the linear background in the center of each background region. The intensity value of each fix point is the average intensity in the background region.

The peak region is integrated over the integral given by the Csize parameter centered at Cpos.

The background-subtracted peak integral is returned in ReducedData1. ReducedData2 receives the error estimate of the peak integral (assuming Poisson statistics).

2.2.3 Peak fitting

The gauss4_reduction function converts a two-dimensional Scienta image I(angle, energy) into a one-dimensional angle distribution I(angle). For each angle slice, it performs a Gaussian curve fit with up to four components on a linear background.

To improve the stability of the fit, the peak positions and widths are kept fixed while the amplitudes of the peaks and the background parameters are variable but constrained to reasonable values (positive amplitude). Furthermore, the function can optionally do a box averaging over three slices.

The function requires the following, fixed parameters:

Parameter	Description
rngl	lower limit of the fit interval

Parameter	Description
rngh	upper limit of the fit interval
npeaks	number of components
pos1	center energy of peak 1
wid1	width of peak 1
pos2	center energy of peak 2
wid2	width of peak 2
pos3	center energy of peak 3
wid3	width of peak 3
pos4	center energy of peak 3
wid4	width of peak 3
ybox	box size of slice averaging (1 or 3)

The peak parameters should be determined beforehand from fitting a reference spectrum, or the angle-scan integrated over all angles. Peak positions and widths have to be specified only up to the given number of peaks.

The data reduction procedure returns the peak integrals (amplitude times width times square root of 2) in waves named ReducedDataN where N is a numeric index from 1 to npeaks. The waves starting with an index of npeaks+1 contain the corresponding error estimate of the peak integral.

2.2.4 Custom reduction functions

See the documentation and source code of int_linbg_reduction, gauss4_reduction and adh5_default_reduction for help on writing custom reduction functions. To integrate your function with the PEARL data explorer, you have to provide an additional function that prompts for reduction parameters such as prompt_int_linbg_reduction, for example. Since reduction functions cannot be called from the command line, it is redommended to also write an adapter function for testing.

2.3 Normalization

The goal of the data normalization is to get a (still two-dimensional) dataset that ideally contains intensity variations due to diffraction features and statistical fluctuations only. In particular, instrumental variations should be removed. In some cases, it may be necessary to preserve the overall polar dependence of the intensity. Note that this latter case is not properly treated with the methods described here.

Depending on the quality of the measured data, only some of the following processing steps are necessary. Use your own judgement.

2.3.1 Preparations

Start by creating a new copy of the data and inspecting it:

```
duplicate ReducedData1, NormData1
ad_display_profiles(NormData1)
```

To update the display after changes to NormData1:

```
ad_update_profiles(NormData1)
```

2.3.2 Detector angle range

Crop the detector angle axis to a useful range (usually about -25 to +25 degrees):

```
crop_strip(NormData1, -25, 25)
```

2.3.3 Normalize detector angle

Remove inhomogeneity of the detector in the detector angle axis. This component may also include a contribution from the sample. If your raw data shows a flat distribution, this step is not necessary.

```
normalize_strip_x(NormData1, smooth_method=4, smooth_factor=0.15, check=2)
```

Note that the argument `check=2` causes the function to generate two check waves but not to modify the original data. To inspect the check waves:

```
display check_dist, check_smoo
ModifyGraph rgb(check_dist)=(0,0,0)
```

Vary the `smooth_factor` (between 0.1 and 1.0) until it follows the instrumental curve but does not affect diffraction features. Then set `check=1` to apply the normalization to `NormData1`.

2.3.4 Azimuthal variation (wobble)

Reduce the effect of azimuthal wobble (misaligned surface) on intensity. A misaligned surface may cause a sinusoidal variation of the intensity as a function of azimuthal angle with a 360 period. A strong azimuthal variation may affect the polar normalization in the next step. The azimuthal normalization can be based on a restricted range of polar angles (theta range). You have to find out which value works best for your sample.

```
normalize_strip_phi(NormData1, :attr:ManipulatorTheta, :attr:ManipulatorPhi, theta_offset=-8.8, theta_range
      =10, check=2)
```

Note, however, that his function does not correct for angle shifts induced by the misalignment!

2.3.5 Polar dependence

Remove the polar angle dependence (matrix element and excitation/detection geometry).

```
normalize_strip_theta(NormData1, :attr:ManipulatorTheta, theta_offset=-8.8, smooth_method=4, smooth_factor=
      0.5, check=2)
```

Use the check waves and the `check` argument as described above.

2.4 Binning and plotting

2.4.1 Basic steps

You can bin and plot the data in one step:

```
pizza_service(NormData1, "Nickname1", -8.8, 0.5, 6)
```

or two steps:

```
pizza_service(NormData1, "Nickname2", -8.8, 0.5, 6, nograph=1)
display_hemi_scan("Nickname2")
```

The benefit of the latter is that you have more control over the graph through optional arguments. In particular, you can select the projection or hide the ticks and grids. See display_hemi_scan for details.

The pizza_service function requires the waves with manipulator positions in a specific place, namely `:attr:ManipulatorTheta` (for the polar angle), and the normal emission values as function arguments. If you have moved the waves, or if you have subtracted the offsets yourself, use the alternative pizza_service_2 function.

Additional parameters of the pizza_service function allow for rotational averaging, larger angle steps (default 1 degree), or the creation of metadata including a notebook for xpdPlot.

Note there is currently a bug in the nick name argument of some of the following functions. If the lines shown below do not work, try to switch to the data folder that contains the generated polar plot data, and call the function with an empty nickname `""`.

2.4.2 Refinements

To remove high polar angles above θ = 80 from the plot (and data):

```
trim_hemi_scan("Nickname1", 80)
```

Modify the pseudocolor scale by changing the `polarY0` trace:

```
ModifyGraph zColor(polarY0)={mod_values, *, *, BlueGreenOrange, 0}
ModifyGraph zColor(polarY0)={mod_values, -0.2, 0.2, BlueGreenOrange, 0}
```

To set the contrast to clip specified percentiles of the data points, use the

- set_contrast function:

```
set_contrast(2, 2, graphname="graph_Nickname1", colortable="BlueGreenOrange")
```

2.4.3 Interpolation

Polar plots can also be interpolated to a rectangular matrix, which may in some cases produce nicer images:

```
interpolate_hemi_scan("Nickname1")
display_hemi_scan("Nickname1", graphtype=3, graphname="intp")
matrix = sqrt(x^2 + y^2) <= calc_graph_radius(80) ? matrix : nan
ModifyImage matrix ctab= {*,*,BlueGreenOrange,0}
```

The `matrix =` line optionally removes artefacts at high polar angles. Replace the cut-off angle with your own.

2.4.4 Modulation function

To calculate the modulation function and substitute it in the graph:

```
setdatafolder Nickname1
calc_modulation(values, factor1=pol, factor2=az)
ModifyGraph zColor(polarY0)={mod_values,-0.2,0.2,BlueGreenOrange,0}
```

2.5 Data export

2.5.1 Export picture

The following line is an example of how to export a graph window. Click on the desired graph window, then issue the following command, substituting the file path and file name as appropriate.

```
SavePICT/P=home/E=-5/B=144/O as "some_filename.png"
```

2.5.2 Export processed data

The following line saves the dataset to an Igor text file. The file contains all data necessary to recreate a polar plot without further processing.

```
save_hemi_scan("Nickname1", "home", "some_filename")
```

For structural optimization using the PMSCO software, it is necessary to generate an ETPI file. There is currently no special function for this. Instead, you have to create and set an energy wave,

```
duplicate pol, en
en = 123.4   // kinetic energy of the photoelectron
```

and write the four waves `en, pol, az, values` to a general text file. Be careful about the ordering of the waves! You will also have to rename the file to the `.etpi` extension because Igor always saves with `.txt` extension. If you have a wave with statistical errors, add a fifth column and use the `.etpis` extension.

```
Save /G /M="\n" /O /P=home en, pol, az, values, sig as "Nickname1.etpis.txt"
```

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

File Documentation

Index