Tutorial - Processing serial Laue crystallographic data with CrystFEL

Needed software

CrystFEL versions

- CrystFEL version 0.9.1
  - Original 0.9.1 version had to be modified - lines 865-876 removed from integration.c source code (see Nass et al., 2021 reference below).
  - Modified 0.9.1 is provided for installation as Apptainer by BioCARS
  - This version is used for image conversion to h5 format, indexing with pinkIndexer and integration
- CrystFEL version 0.10.2
  - provided for installation as Apptainer by BioCARS)
  - This version is used for resolving indexing ambiguity (if needed) and for merging the data

Apptainer (formerly known as Singularity) is a free and open-source container platform that allows to create and run applications in isolated environments (also called “containers”) in a simple and portable, manner.

To run CrystFEL and related *py scripts (see below) using Apptainer, specify working directory name (can omit this directory if running in the working directory), data directory and full directory path for the CrystFEL sif file (crystfel-0.9.1.sif or crystfel_0.10.2.sif).

For example:

apptainer shell -B /data/laue-data-processing -B /data-directory /data/sif-files/crystfel-0.9.1.sif

Directories above are just examples, change them to match your directory set-up for data processing.

To exit Apptainer: ctrl D

All CrystFEL commands need to be run from the specified Apptainer.

Useful CrystFEL and pinkIndexer links:

- https://journals.iucr.org/d/issues/2019/02/00/ba5291/
- https://www.desy.de/~twhite/crystfel/manual.html
- https://www.desy.de/~twhite/crystfel/tutorial.html
- Relevant literature:
  - Gevorkov et al. (2020) pinkIndexer – a universal indexer for pink-beam X-ray and electron diffraction snapshots. Acta Cryst A 76, 121–131. http://scripts.iucr.org/cgi-bin/paper?S2053273319015559
  - Nass et al (2021) Pink-beam serial femtosecond crystallography for accurate structure-factor determination at an X-ray free-electron laser. *IUCrJ 8*, 905–920. https://journals.iucr.org/m/issues/2021/06/00/zf5018/
  - Tolstikova A., Development of diffraction analysis methods for serial crystallography, PhD Thesis (2021). https://ediss.sub.uni-hamburg.de/handle/ediss/8388

These Python scripts are also necessary and provided by BioCARS:

- Python scripts (provided by BioCARS for download)
  - mccd2h5.py (converts BioCARS images to h5 format)

    - h5Viewer.py (to view h5 files)
    - maskMakerGUI.py (to make a detector mask file)
    - streamViewer.py (to check indexing and predictions)

- Scripts below were provided to BioCARS by Alexandra Tolstikova (CFEL, Germany) based on her PhD Thesis (https://bib-pubdb1.desy.de/record/441104/). They were modified by BioCARS staff as it was needed for processing of BioCARS 14ID Laue data.
    - peaks_hist.py
    - rdco.py
    - scale.py
    - remove_overlaps.py
    - lorentz.py

The scripts are run from the Apptainer (see above).

- Conversion from output hkl format to mtz format (provided by BioCARS for download). Need to edit to enter proper sample name and cell parameters.
    - create-mtz-modified-noF

  Can use any other available suitable conversion script.

Needed input files (files used for the example data set are provided):

- images-to-index.txt: ist of images to process (full path)
- mask.h5: detector mask file, see below and Appendix A
- crystal.cell: cell parameters file (see example below)
- mccd.geom: geometry file (see example below)
- xray-spectrum: BioCARS energy spectrum file that works with CrystFEL (provided by BioCARS staff based on the spectrum measured during the beamtime). This is a text file with the following format:
  # energy/keV current/A
  1.0000000000000000e+01 -1.524415940065128056e-03
  1.0009999999999979e+01 -9.492029257979610899e-04
  1.0019999999999957e+01 -1.588834576064361917e-03
  …

==========

Example data set provided on request

==========

Tutorial is based on a BioCARS Laue serial crystallography data set. If you want to use this example data set to practice data processing, please contact Robert Henning (henning@cars.uchicago.edu)

- Data set is generously provided by Seong Ok Kim, Department of Chemistry, KAIST, Center for Advanced Reactions Dynamics, Institute of Basic Science, Republic of Korea
- Serial crystallography Laue data collected in February 2022
- Sample: lysozyme
- BioCARS undulators setting: 12 keV, ~3% bandwidth

- Input parameters and other information for the example data set

- Input cell parameters, room T: 78.4 78.4 37.9 90 90 90, P43212, space group number 96
- Input crystal-to-detector distance: 362 mm
- Input beam center: 1987.3, 1974.9
- Number of images collected: 13,432
- Hits selected for CrystFEL processing: 990
    - Selection done by BioCARS pyPrecognition.py hit finding script
    - Selection criteria: more than 100 diffraction spots; minimum intensity above the background to be considered as a diffraction spot: 50

==========

0. Getting started

==========

You will need:

- mask.h5 (detector mask file, input to geometry file; beam stop, beam stop support and bad sections of the detector are masked out). You can get file for your data from BioCARS staff or see appendix A on creating such file.
- mccd.geom (geometry file). Modify the example file to include correct mask.h5 file, distance and beam center.
- crystal.cell file. Modify the example file for your crystal.
- images-to-index.txt file for your data (list of images to process images-to-index.txt, full path should be included).

Geometry file mccd.geom for the example data

```
; Manually optimized with hdfsee
; Manually optimized with hdfsee
photon_energy = 11300                #11400 is fine as well
photon_energy_bandwidth = 0.04
adu_per_eV = 0.0001
clen = 0.362
res = 11287.47795414462

mask_file = mask.h5
mask = /data/data
mask_good = 0x01
mask_bad = 0x00

data = /data/data
dim0 = %
dim1 = ss
dim2 = fs

rigid_group_d0 = 0
rigid_group_collection_quadrants = d0
rigid_group_collection_asics = d0
rigid_group_collection_det = d0

0/min_fs = 0
```

0/max_fs = 3839
0/min_ss = 0
0/max_ss = 3839
0/corner_x = -1987.3
0/corner_y = -1974.9
0/fs = +1.000000x +0.000000y
0/ss = +0.000000x +1.000000y

Shown in red are parameters to be potentially adjusted:

- Values for photon_energy and photon_energy_bandwidth listed above work well for processing of Laue data, given the typical BioCARS X-ray spectrum used for serial crystallography
- photon_energy: mean energy (eV) of the part of the spectrum with sufficient intensity (do not list peak energy which is 12000 eV in this case)
- photon_energy_bandwidth: standard deviation of a Gaussian centered at photon_energy as a fraction of photon_energy. Peaks in this energy range will be predicted by pinkIndexer:
  photon_energy +/- 2.5 * photon_energy * photon_energy_bandwidth.
- clen is crystal-to-detector distance in meters
- mask_file defines full path to detector mask file (mask.h5)
- 0/corner_x and 0/corner_y are beam center values in pixels (except minus sign, they correspond to x,y beam center values provided by BioCARS staff)

Cell file crystal.cell for the example data

CrystFEL unit cell file version 1.0
lattice_type = tetragonal
unique_axis = c
centering = P
a = 78.4 A
b = 78.4 A
c = 37.9 A
al = 90.00 deg
be = 90.00 deg
ga = 90.00 deg

==========

Converting Rayonix diffraction images to h5 format

==========

Use mccd2h5.py script
(-h for help with definitions of input parameters and default values)

apptainer shell -B /data/laue-data-processing -B /data-directory /data/sif-files/crystfel-0.9.1.sif

mccd2h5.py images-to-index.txt . -g mccd.geom -m mask.h5 -n 16 --max-pix-count=100 --min-peaks=20

Based on *mccd2h5.py -h*, peakfinder8 is used for peak-finding just for this conversion command. Peak-finding is later repeated for actual data processing.
--max-pix-count is max number of pixels per diffraction spots. 100 recommended for the BioCARS Rayonix detector

--min-pix-count is the default value of 2
--min-peaks is the minimum number of peaks per image to be considered a hit
-n number of images processed in parallel and combined in one *h5 file

Input files
- images-to-index.txt (list of images to process, complete directory path)
- mccd.geom (geometry file, see above)
- mask.h5 (mask file, get from BioCARS staff or see Appendix A; not critical if incorrect mask file is used for this step)

Output files (16 files of each in this case, given -n 16 above)
- images-to-index*.h5
- images-to-index*.lst

Create a list of *h5 files.

find . -name "images.h5" > files.lst*

Output file: files.lst (it will be used for indexing)

==========

If room temperature unit cell is not known, use CrystFEL mono-indexing to get approximate cell parameters

==========

- Modify mccd.geom to create mccd-nopink.geom by replacing:

photon_energy = 11300
by
photon_energy = 12000

Run standard CrystFEL monochromatic indexing command, for example (run apptainer with crystfel_0.10.2.sif; see page 1):

apptainer shell -B /data/laue-data-processing -B /data-directory /data/sif-files/crystfel-0.9.1.sif

indexamajig -i files.lst -o nopink.stream -g mccd-nopink.geom --peaks=peakfinder9 --threshold=5 --max-res=1400 --min-pix-count=3 --min-snr=4.5 --max-pix-count=150 --min-peaks=25 --indexing=mosflm --no-refine --no-check-peaks -j 32

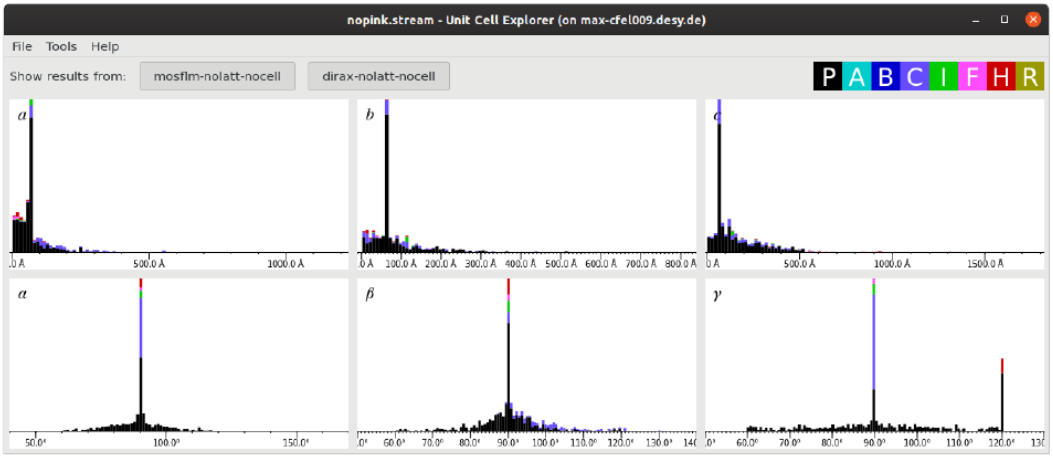
Output file: nopink.stream

Run cell_explorer (CrystFEL command) to determine the cell.

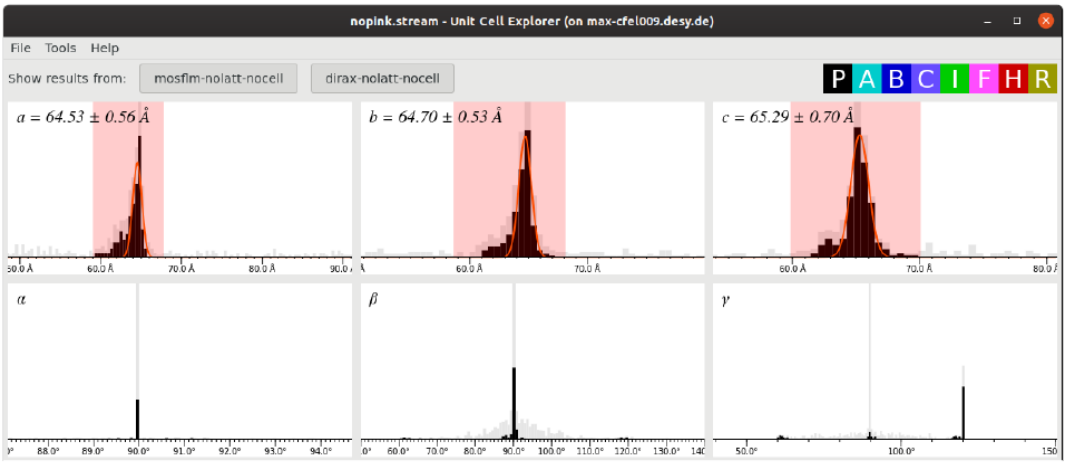
cell_explorer nopink.stream

You will need to zoom in on the displayed distributions of each cell values and select the region around the sharp peaks to get the estimated the cell values (see step 7 onhttps://www.desy.de/~twhite/crystfel/tutorial-0.9.1.html).

Plot before zoom in:



After zoom in to the sharp histogram peaks:



Enter cell values estimated this way to crystal.cell file and use this file for indexing of Laue data with pinkIndexer.

==========

1. **Indexing with pinkIndexer**

==========

Use Apptainer for the CrystFEL 0.9.1 version

This command finds spots with peakfinder9, indexes with pinkIndexer and integrates spots to the edge of the detector (no resolution limit for integration). Resolution can be limited (as below) for indexing, to speed up the indexing which is typically a slow step (*--pinkIndexer-max-resolution-for-indexing=0.40*, max resolution is in 1/A).

apptainer shell -B /data/laue-data-processing -B /data-directory /data/sif-files/crystfel-0.9.1.sif

indexamajig -i files.lst -o 0.stream -g mccd.geom -p crystal.cell --peaks=peakfinder9 --threshold=5 --min-snr=4.5 --local-bg-radius=5 --min-snr-biggest-pix=4 --min-snr-peak-pix=3 --min-peaks=25 --indexing=pinkIndexer-latt --pinkIndexer-considered-peaks-count=4 --pinkIndexer-angle-resolution=4 --pinkIndexer-refinement-type=5 --pinkIndexer-tolerance=0.03 --pinkIndexer-max-resolution-for-indexing=0.40 --no-retry --no-refine --no-check-peaks --no-check-cell --fix-profile-radius=0.001e8 --int-radius=5,7,9 --no-non-hits-in-stream --pinkIndexer-thread-count=20 -j 6

Inputs:

- *mccd.geom* (make sure it includes correct mask.h5 file)
- *crystal.cell*

Output:
0.stream file (text file)

For description of various parameters and options in indexamajig command above, check CrystFEL manual/tutorial and Gevorkov et al., 2020 paper (see p.1). pinkIndexer parameters, values and options listed above were used successfully for processing of the BioCARS example Laue data set, although perhaps they could be further optimized.

--int-radius=5,7,9 are integration radii for three-rings integration method described in indexamajig manual. Need to be adjusted depending on spot sizes in your case.

Make sure to change output stream file name (0.stream) if want to save previous version as it will be overwritten on repeated run of the command above.

Unit cell information in the crystal.cell serves only as a starting point. Indexing will produce unit cell values (a distribution of cell values in fact), while making sure there is a consistency with the input cell in crystal.cell.

Run indexing command several times partially (only until 20-30 images processed) while varying these parameters (related to peakfinder9):

--min-snr-biggest-pix=5
--min-snr-peak-pix=4

Check how many peaks have been found for each setting.

head -100000 0.stream | grep num_peaks

Target numbers are several hundred (100-500).

Running *indexamajig* command above is very slow and it may take days to process the entire data set of several thousands of images on a single computer. This is due to the indexing step when using pinkIndexer, which is necessary for indexing of Laue data.

--*pinkIndexer-thread-count=20 -j 6* specifies number of threads pinkIndexer is using and number of images processed in parallel. When multiplied, this should be the number of cores on your machine.

For faster processing, you may want to run the command on a cluster of computers if available at your home institution. Please contact Robert Henning (henning@cars.uchicago.edu) for some advice based on our BioCARS experience.

==========

2. **Assessing indexing and geometry refinement**

==========

Run all withing the Apptainer, same as for indexamajig.

How many hits:

grep "filename" 0.stream | wc -l

How many hits did not indexed:

grep "indexed_by = none" 0.stream | wc -l

How many unit cells found:

grep "Cell" 0.stream | wc -l

This is equal to the number of indexed images if only one indexing attempt was done for the given diffraction image, as in the command above. Some images in serial crystallography can have several overlapping diffraction patterns on the same image. They result from several crystals that were exposed at the same time. With pinkIndexer, you can actually specify if you want to index these additional diffraction patterns on the image. But this will make indexing process significantly longer.

Use streamViewer.py to check indexing and diffraction spot prediction quality:

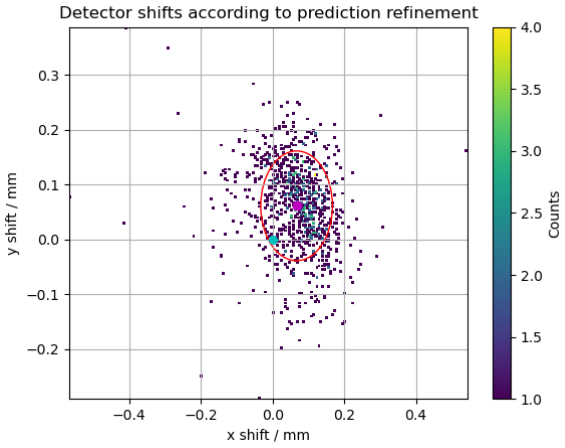
streamViewer.py 0.stream

Complete these steps to assess adjustments to beam center, cell parameters and distance.

1) Check beam center:

detector-shift 0.stream mccd.geom

(see https://www.desy.de/~twhite/crystfel/tutorial-0.9.1.html, step 9)



This command will save the corrected beam center (corresponding to the red spot on the plot) to mccd-predrefine.geom file (can be renamed of course). If beam center is significantly off, you will need to repeat the indexing. Getting beam center shift as close to 0 as possible is important. If necessary, vary also detector distance slightly to get the beam center shift closer to 0. For the example data set, only when beam center shift was brought to 0 (with necessary small distance adjustment, clen = 0.3625 rather than clen = 0.362), the broad cell parameter distribution shown below got significantly sharper and separated into two distributions (see below).

2) Check cell parameters and detector distance

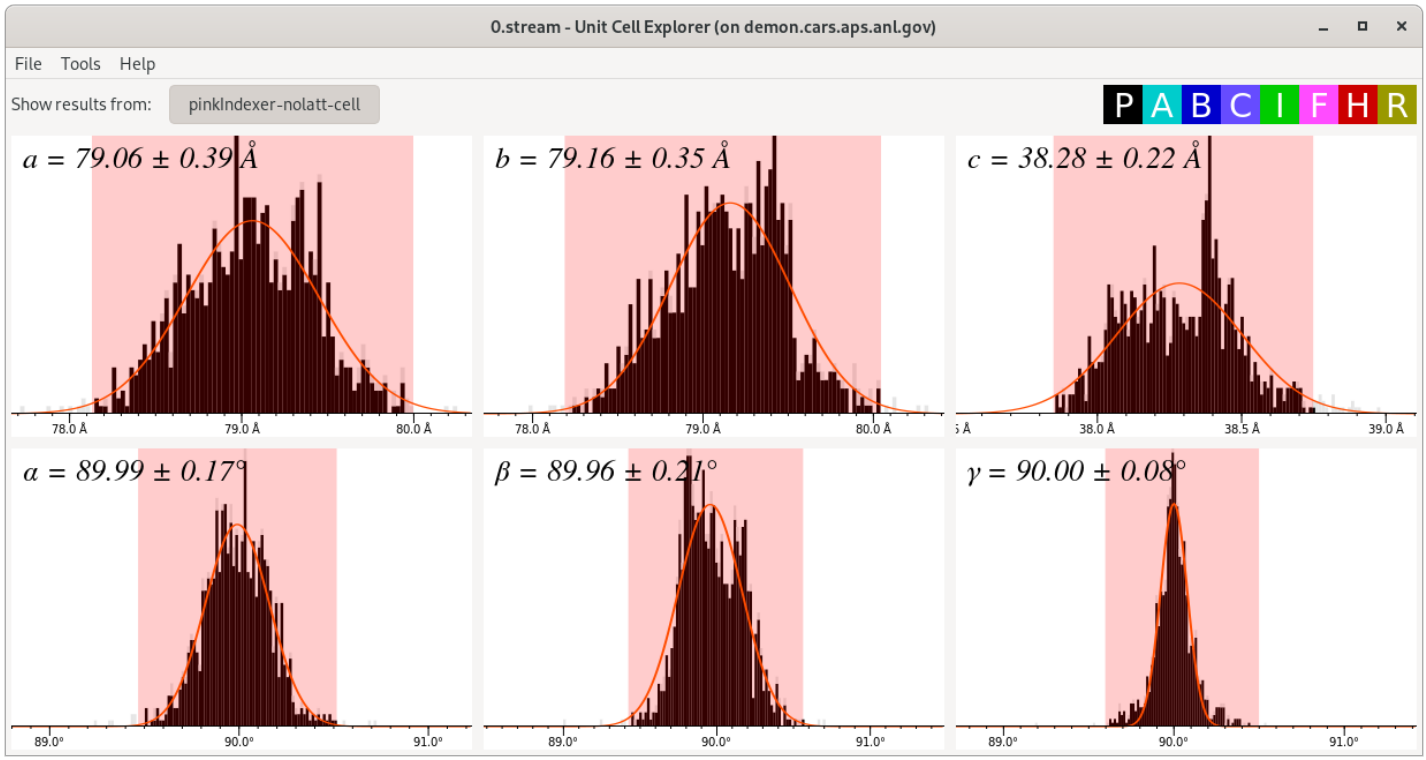
Detector distance and cell parameters are related and difficult to refine separately for Laue data. Best strategy is to use well known room temperature cell parameters is possible (from monochromatic room T data, for example) and to use the best detector distance provided by beamline staff.

a) To assess cell parameters:

cell_explorer 0.stream

You will need to zoom in on the distribution/histogram of each cell value and select the region around the sharp peaks to get the estimated cell values (see step 7 on https://www.desy.de/~twhite/crystfel/tutorial-0.9.1.html).

After zoom in, the plot will look something like this:



If cell parameters are off from the expected/known values (which were used as an input), this could be due to the wrong beam center and/or wrong distance. Next step will tell us more.

b) Check energy histogram:

In order to assess if we need to adjust the detector distance or unit cell parameters from a), we need to compare:

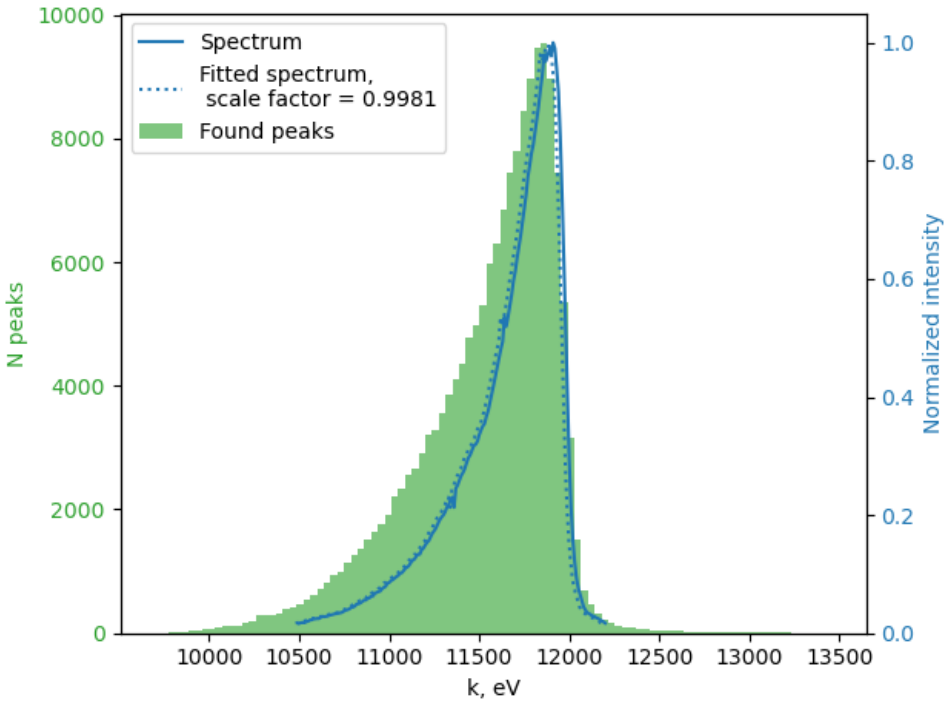
- the measured BioCARS X-ray spectrum (xray-spectrum)
- with distribution of the central X-ray energies contributing to found peaks

peaks_hist.py 0.stream mccd.geom xray-spectrum -n 32

Inputs:

- *0.stream* (output of indexamajog command
- *mccd.geom*
- *xray-spectrum* is the X-ray spectrum (provided by BioCARS staff from the measured spectrum during the beamtime)

This command produces a plot as the one shown below: measured spectrum (solid line), energy histogram from the found peaks in the data set, and measured spectrum shifted to fit the high-energy edge to the energy histogram (dashed line). The scale factor listed on the plot is unit cell scaling factor: to match the histogram to the spectrum, the unit cell from indexing (in 2a) should be multiplied by this scale factor.



At this point, one has to decide what is more realistic to adjust (based on expected/measured values for unit cell and distance):

- adjust cell parameters: multiply cell from 2a) by the scale factor from the plot (in this case, there is no need to repeat indexing, this scale factor will be applied in subsequent steps)
- adjust distance: multiply the input distance by (expected or known cell value) / (cell value from (2a) multiplied by the scale factor from the plot)

Command above also produces **k.dat** file as output. It contains X-ray energy and resolution for each found reflection/peak.

==========

3. **Repeat indexing and geometry refinement if needed**

==========

- If it was necessary to adjust beam center or distance after the first indexing attempt, repeat:

1. Indexing with pinkIndexer

For the example data set, for best results we reindexed the data with the new beam center and distance, listed in updated geometry file *mccd.geom*:
clen = 0.3625
0/corner_x = -1986.77
0/corner_y = -1974.49

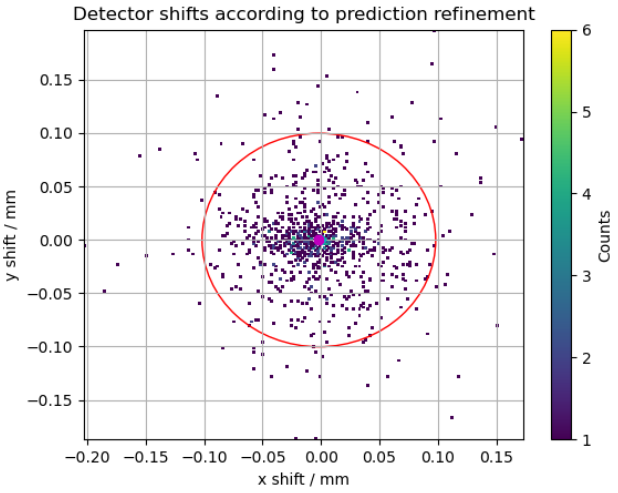
We also changed photon_energy = 11300 to photon_energy = 11400 in the updated *mccd..geom* file.

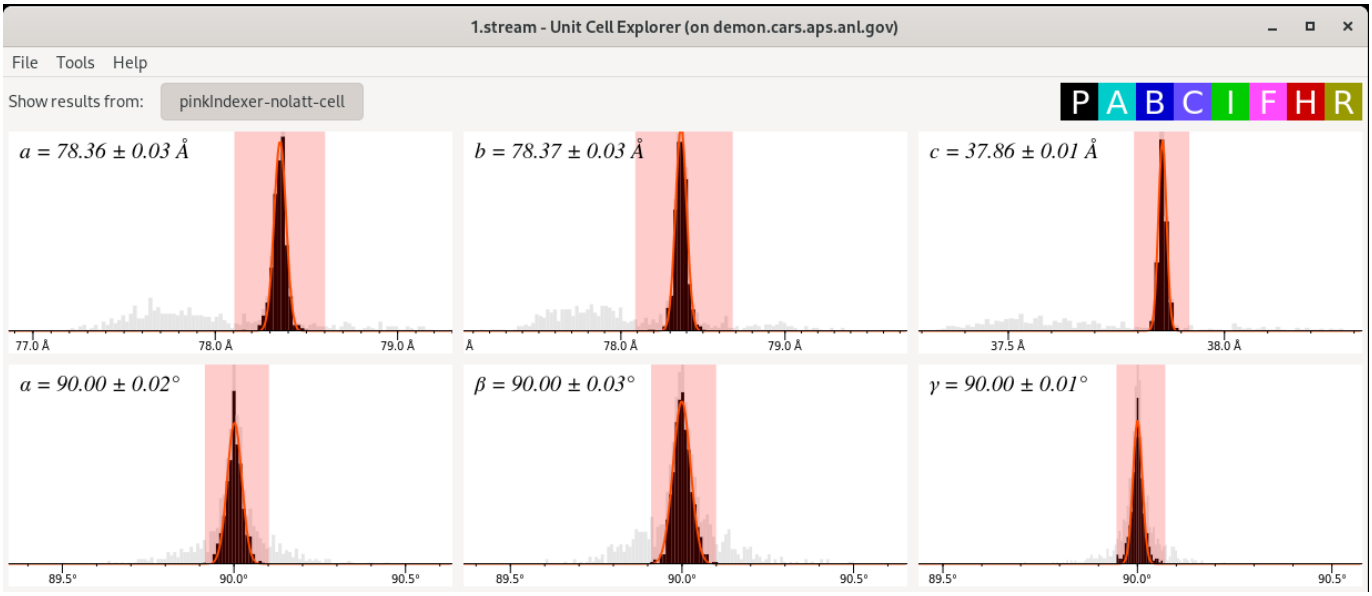
apptainer shell -B /data/laue-data-processing -B /data-directory /data/sif-files/crystfel-0.9.1.sif

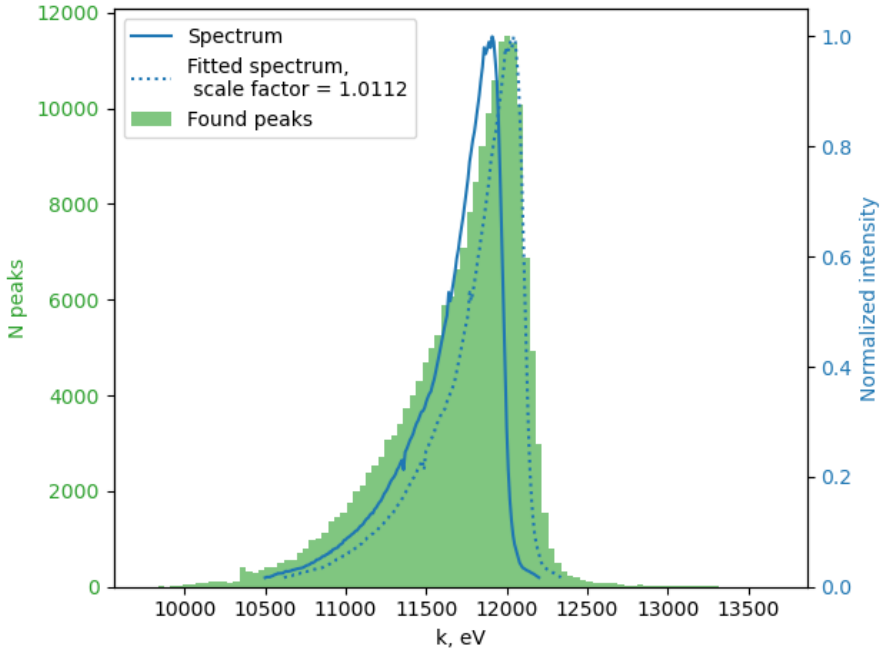
indexamajig -i files.lst -o 0.stream -g mccd.geom -p crystal.cell --peaks=peakfinder9 --threshold=5 --min-snr=4.5 --local-bg-radius=5 --min-snr-biggest-pix=4 --min-snr-peak-pix=3 --min-peaks=25 --indexing=pinkIndexer-latt --pinkIndexer-considered-peaks-count=4 --pinkIndexer-angle-resolution=4 --pinkIndexer-refinement-type=5 --pinkIndexer-tolerance=0.03 --pinkIndexer-max-resolution-for-indexing=0.40 --no-retry --no-refine --no-check-peaks --no-check-cell --fix-profile-radius=0.001e8 --int-radius=5,7,9 --no-non-hits-in-stream --pinkIndexer-thread-count=20 -j 6

Output file: *0.stream* (overwritten after second indexing)

The new beam center, unit cell and energy histograms:







Cell parameters from the histogram above, needed to be scaled by 1.0112.

Cell from histogram (sharp peak only included):

a=78.36 A
b=78.37 A
c=37.86 A

Corrected cell will be:
a=79.24
b=79.25
c=38.28

For best processing in this case, images corresponding only to the sharp peak in the unit cell histogram above can be separated from the rest of the images from for 0.stream file (BioCARS can provide split.py script if

needed) and scaled/merged separately. In this case, 488 of 990 images corresponding to the sharp peak will be scaled and merged).

2. Assessing indexing and geometry refinement

- If unit cell needed to be scaled, like in this case, you will use the scale factor from the final "2b) Check energy histogram" plot described above in the scale step (see below).

==========

4. Normalizing integrated intensities according to measured spectrum and applying energy cut-off before data merging

==========

For Laue diffraction images, reflections are stimulated by various wavelengths, λ, from the incident X-ray spectrum, where integrated intensities of reflections, $I_{hkl}(\lambda)$, vary according to the spectrum. Intensities $I_{hkl}(\lambda)$ therefore have to be normalized according to the X-ray spectrum.

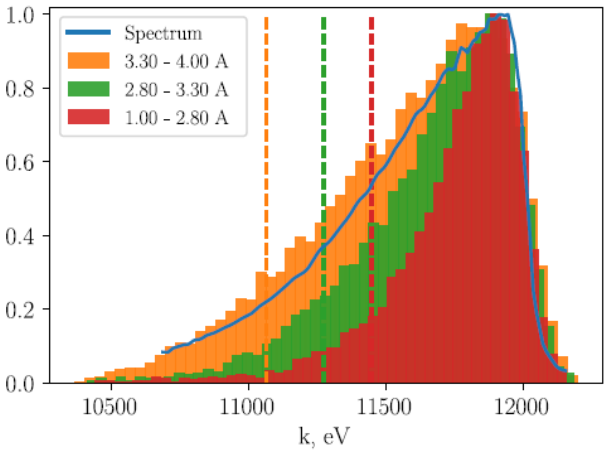
In addition, intensities of reflections stimulated by wavelengths at the low-intensity tail of the X-ray spectrum are poorly measured. It helps if they are excluded. One method that provides good merging statistics is a resolution-dependent spectral intensity cut-off determined by the X-ray energy (dashed lines in the plot below) where 10% of peaks are stimulated by energies lower than that energy for each resolution bin.

To get the resolution-dependent spectral cut off, run command:

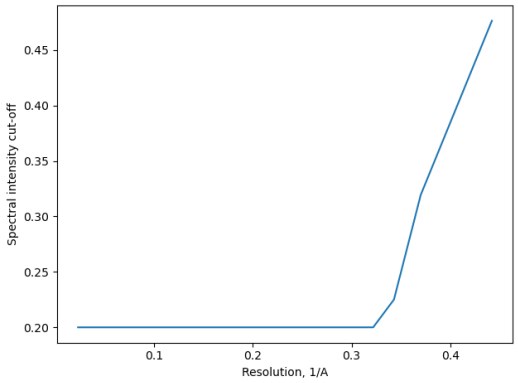
rdco.py -i k.dat -s xray-spectrum -o rdco.dat

Inputs:

- *k.dat* file is output of *peaks_hist.py* command (see above); make sure you use the latest one in case you repeated indexing in step 3.
- *xray-spectrum* is the X-ray spectrum used in *peaks_hist.py* (see above)



Output:
rdco.dat: first column is resolution (1/A), second column is spectral weight cut-off (see plot below). Spectral cut-off in this file corresponds to the intersection of the vertical dashed lines for each resolution bin and the spectrum in the plot above.



To apply spectral intensity cut-off and normalize integrated intensities of reflection according to the X-ray spectrum, run command:

scale.py 0.stream xray-spectrum -n 32 -u 1.0112 -r rdco.dat

Inputs:

- *0.stream* is output from indexamjig command (step 3, repeated indexing)
- *1.0112* is the unit cell scale factor from step 3 for the example data set; change for your data
- *rdco.dat* is output of rdco.py command (see above)

Output:

- *0-s.stream* is the new stream file after scaling.

CrystFEL does not resolve spatially overlapping reflections.
To remove overlaps, run:

remove_overlaps.py 0-s.stream 10 -n 32

Inputs:
0-s.stream is output of scale.py
10 is the minimum distance (in pixels) for keeping the reflections as non-overlapping. All reflections with a nearest neighbor within this distance are deleted. This distance is twice the inner integration radius from indexamajig command (-int-radius=5,7,9 in this case) and should be adjusted if different values were used.

Output:
0-s-ol.stream is the new steam fine after resolving spatial overlap

==========

5. **Resolving indexing ambiguities (if present for your space group) and merging data**

==========

For these steps, use the apptainer for the 0.10.2 version of CrystFEL rather than for the 0.9.1 version (see p.1).

apptainer shell -B /data/laue-data-processing -B /data-directory /data/sif-files/crystfel_0.10.2.sif

- CrystFEL doesn't use space groups for merging data but point groups (see Symmetry in CrystFEL on https://www.desy.de/~twhite/crystfel/manual.html).
- CrystFEL point groups:

Triclinic: 1, -1
Monoclinic: 2/m, 2, m
Orthorhombic: mmm, 222, mm2
Tetragonal: 4/m, 4, -4, 4/mmm, 422, -42m, -4m2, 4mm
Trigonal (rhombohedral axes): 3_R, -3_R, 32_R, 3m_R, -3m_R
Trigonal (hexagonal axes): 3_H, -3_H, 321_H, 312_H, 3m1_H, 31m_H, -3m1_H, -31m_H
Hexagonal: 6/m, 6, -6, 6/mmm, 622, -62m, -6m2, 6mm
Cubic: 23, m-3, 432, -43m, m-3m

Resolving indexing ambiguity

There is no indexing ambiguity for the example data set (space group P43212, space group number 96).

If you do need to resolve indexing ambiguity for your space group (https://www.desy.de/~twhite/crystfel/twin-calculator.pdf), check *ambigator* command on https://www.desy.de/~twhite/crystfel/manual-ambigator.html.

ambigator 0-s-ol.stream -o 0-s-ol-a.stream -y XXX --operator=YYY -j 32 -n 20 --fg-graph=fg.dat

XXX - actual symmetry of the crystal
YYY - indexing ambiguity operator. Example: --operator=k,h,-l.
(see CrystFEL manual for other options)

Merging data

partialator -i 0-s-ol.stream -o 0-s-ol.hkl -y 422 --model=unity -j 1 --iterations=3

Inputs:
0-s-ol.stream is output of remove_overlaps.py in step 4.
422 is the point group for the example data set (change for your sample). One can also use the point group you arrive at if you add an inversion operation to 422, in this case 4/mmm, if not interested in preserving potential anomalous signal since this will treat Friedel pairs as equivalent (see https://www.desy.de/~twhite/crystfel/tutorial-0.9.1.html#merge). This will provide better redundancy.

Output:
0-s-ol.hkl is the overall hkl file after merging
0-s-ol.hkl1 and 0-s-ol.hkl2 are produced by two-way splitting of merged reflections for Rsplit and CC1/2 statistics calculations.

==========

6. Merging statistics

==========

To get merging statistics, use standard CrystFEL commands check_hkl and compare_hkl (see https://www.desy.de/~twhite/crystfel/manual-check_hkl.html and https://www.desy.de/~twhite/crystfel/manual-compare_hkl.html).

Examples (use corrected cell parameters from CrystFEL):

1. Completeness/SNR vs resolution

check_hkl 0-s-ol.hkl -p crystal-CrystFEL.cell --highres=2.0

2. R-split vs resolution

compare_hkl 0-s-ol.hkl1 0-s-ol.hkl2 -p crystal-CrystFEL.cell --fom=rsplit --highres=2.0

3. CC1/2 vs resolution

compare_hkl 0-s-ol.hkl1 0-s-ol.hkl2 -p crystal-CrystFEL.cell --fom=cc --highres=2.0

==========

7. Lorentz factor correction

==========

Accounts for the fraction of the spectrum that contributes to a reflection decreasing with resolution.
Run with apptainer for crystfel-0.9.1.sif.

apptainer shell -B /data/laue-data-processing -B /data-directory /data/sif-files/crystfel-0.9.1.sif

lorentz.py 0-s-ol.hkl crystal-CrystFEL.cell xray-spectrum 0.001e8

Inputs:
0-s-ol.hkl is output of partialator
Crystal-CrystFEL.cell is the cell parameters file with corrected cell
xray-spectrum is the X-ray spectrum file
0.001e8 is the reflection profile radius used with indexamajig (*--fix-profile-radius=0.001e8*)

Output:
0-s-ol-l.hkl is the new hkl file

==========

8. Convert final hkl file to mtz file

==========

You can use any suitable mtz conversion script but these also work:

1. Use CrystFEL get_hkl (https://www.desy.de/~twhite/crystfel/manual-get_hkl.html)

2. Use create-mtz-modified-noF script provided by BioCARS (it uses ccp4 f2mtz command).

Edit the file to list proper crystal name, space group and cell parameters.
Then:

awk '{print $1,$2,$3,$4,$6}' o-s-ol-l.hkl > 0-s-ol-l-5col.hkl

./create-mtz-modified-noF 0-s-ol-l-5col.hkl

Input: 0-s-ol-l-5col.hkl
Output: 0-s-ol-l-5col.mtz

mv 0-s-ol-l-5col.mtz 1-s-ol-l.mtz

Appendix A - Create a detector mask file

- Select one diffraction image
- Create 1_image.txt file with full path of that image
- Convert this image to an h5 format

apptainer shell -B /data/laue-data-processing -B /data-directory /data/sif-files/crystfel-0.9.1.sif

mccd2h5.py 1_image.txt . -g mccd.geom

Output:
1_image-0.h5
1_image-0.lst

(mccd,geom file requires mask file that we are trying to create but it will run even if listed file does not exist)

View the *.h5 file (can also use adxv image viewer):

h5Viewer.py 1_image-0.h5 /data/data /data/peaks

Make mask:

maskMakerGUI.py 1_image-0.h5 data/data -g mccd-lhee-1.geom

Notice that the image viewed here is rotated as compared with the h5Viewer.
Create a round mask to block the beam stop shadow and a rectangular mask to block the beam stop support shadow. Use rectangular mask for any “bad” sections of the detector.
When the "save mask" option is selected, a file "mask.h5" is automatically created. It will over-wright an existing mask.h5 file.

View the mask with adxv (outside of Apptainer):

adxv mark.h5