WAND^2
Single Crystal Diffraction Scripts Manual

July 2026

Xiao Hu
Neutron Scattering Scientist
Lillian Blanton
Next Generation STEM Intern
Matthias Frontzek
Neutron Scattering Scientist
Oak Ridge National Laboratory

Index

I. Prologue

This manual provides an overview of the general workflow of **WAND² single-crystal data analysis** using Mantid-based scripts.

The scripts can be found at

/HFIR/HB2C/shared/WANDscripts /Single Crystal/

on the analysis cluster:

- **WAND2_SCD_Reduction.py**
- **peakintegration_2D.py**

Two methods of peak integration are included:

- **Method 1 (Conventional Mantid algorithm)**
  Utilize Mantid's built-in functions (e.g., ***IntegratePeaksMD***). This method is well suited for rapid integration and quick peak inspection. However, its error propagation can be problematic. The method is challenged by modulated background for instance from aluminum powder rings.

- **Method 2 (Detector image-based algorithm)**
  A newly developed algorithm that performs peak integration directly from detector images. Although more sophisticated than Method 1, this approach resolves several limitations, such as handling adjacent peaks, irregular peak profiles, and aluminum powder ring contamination.

It should be noted that all the Mantid algorithms used in the script can be executed from the GUI. The algorithms are mentioned in this manual in ***bolditalic*** as for instance ***IntegratePeaksMD*** on this page.

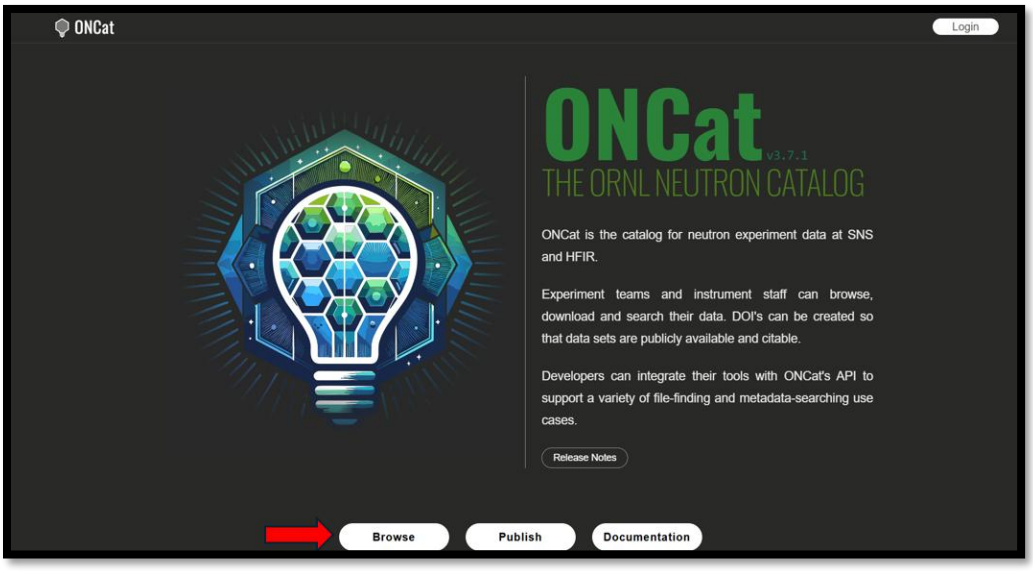
If you have any questions or feedback about this manual, please contact the instrument team, hux6@ornl.gov, mnf@ornl.gov or chens1@ornl.gov .

Back to Index

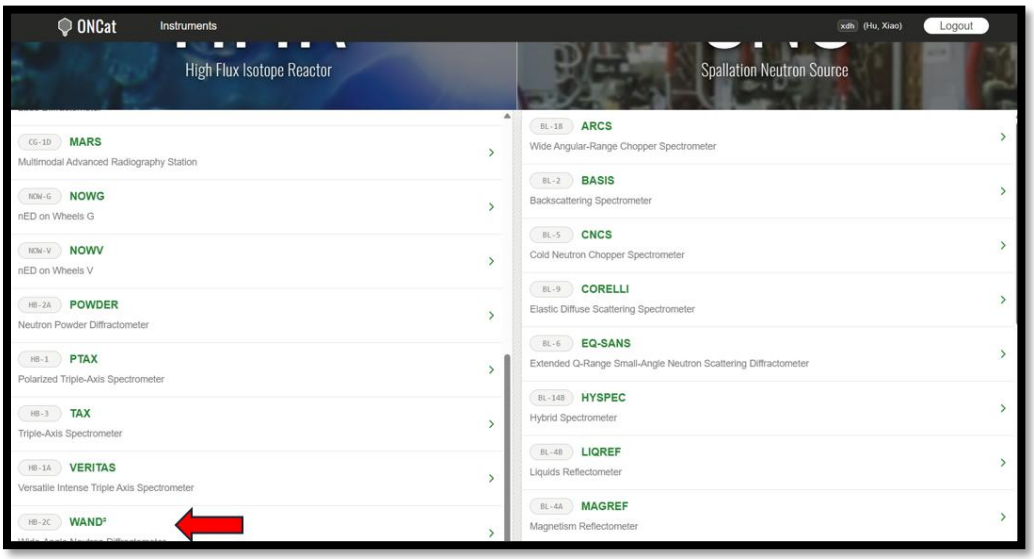
II. Locate IPTS and Run numbers & Open Mantid Workbench

(a) Find experimental data under proposal number (IPTS)

(1) Go to https://oncat.ornl.gov/ to view your experiment information. Click “Browse” and log in to ONCat catalog with UCAMS.



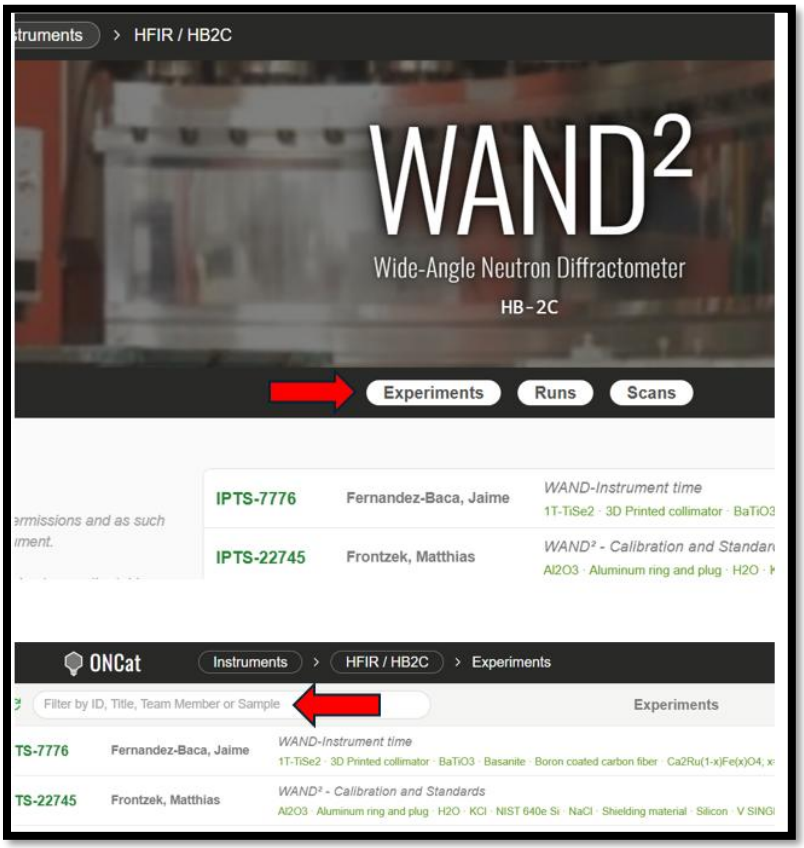
(2) Find the instrument in the HFIR/SNS instrument list and click on WAND^2 .

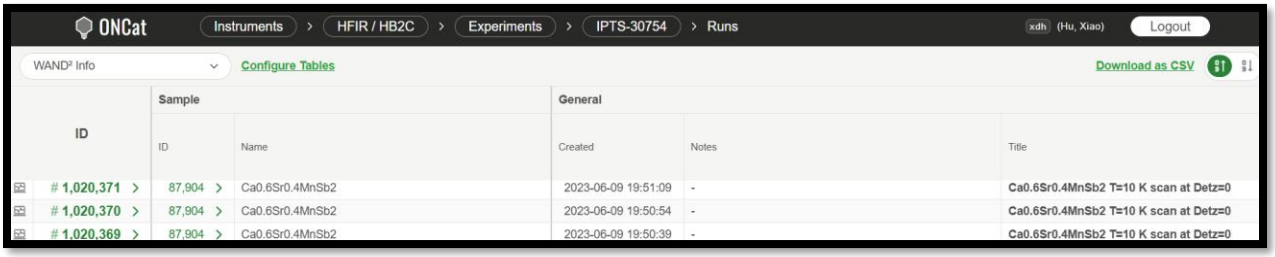


Back to Index

(3) Search for experiment data through: Experiments → type in proposal number (IPTS).

Scans will be displayed along with run number (ID), sample information, and scan descriptions. Record run numbers you need for analysis.





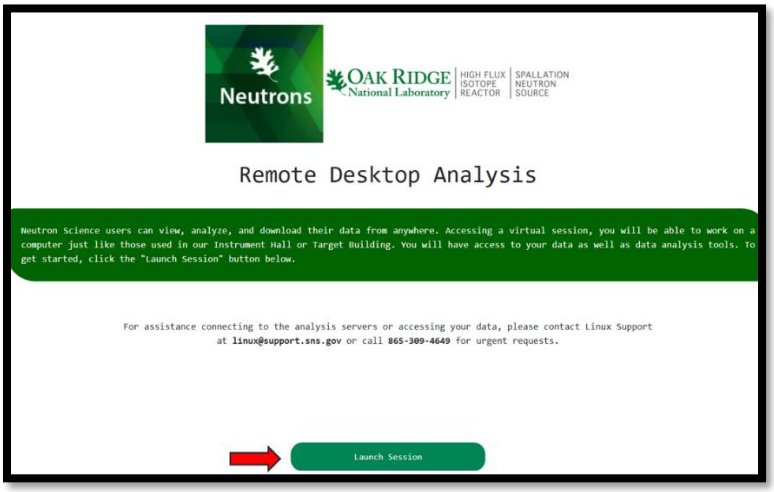
Note:

Vanadium calibration was usually performed before or at the beginning of each cycle. Users have access to the vanadium IPTS 23858. Use the vanadium run number that is closet (in timeline) to your beamtime.

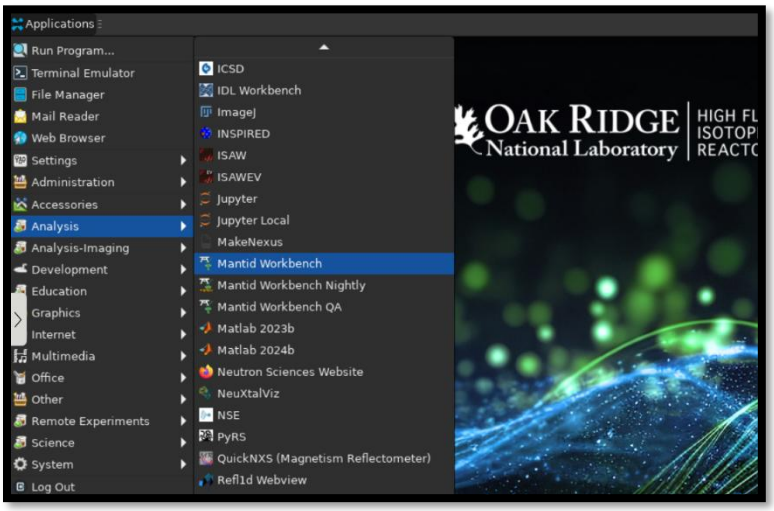
Back to Index

(b) Open Mantid Workbench in analysis cluster

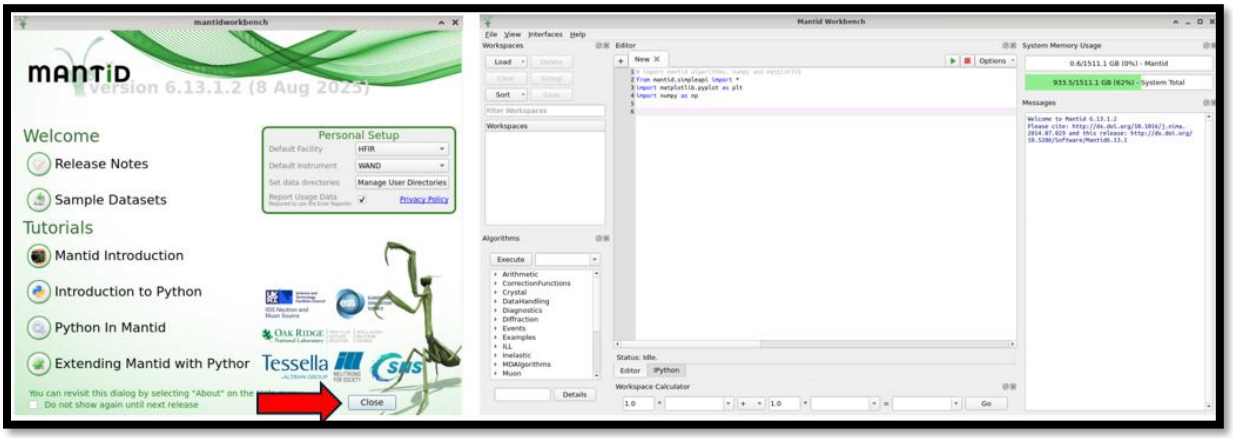
(1) Go to Remote Desktop Analysis at https://analysis.sns.gov/ and log in with UCAMS username and password.



(2) Once logged in, on the left upper corner, go to Application → Analysis → Mantid Workbench.



(3) When Mantid Workbench opens, there will be an option to personalize the setup and tutorial available if needed. If a personalized setup is not necessary, close the Welcome window. You should see the Mantid graphical user interface on your screen.

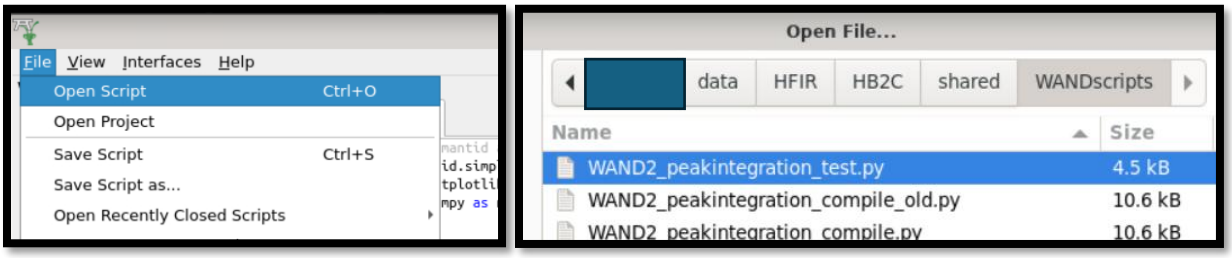


Back to Index

III. Script execution in Mantid

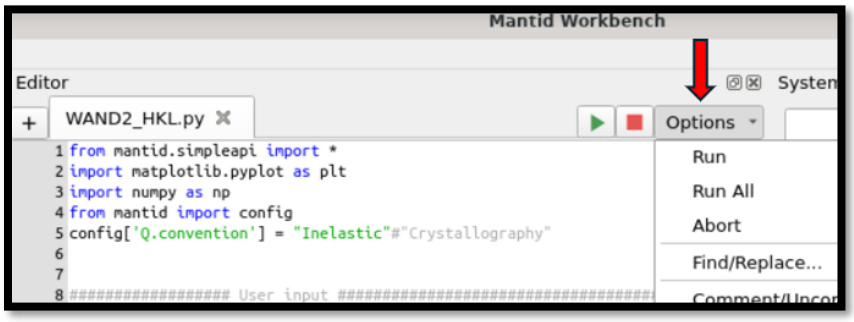
(a) Open a script in Mantid

Go to **File** on the left upper corner of the Mantid interface and hit "**Open Script**" or **Ctrl + O**. Locate and open the target script.

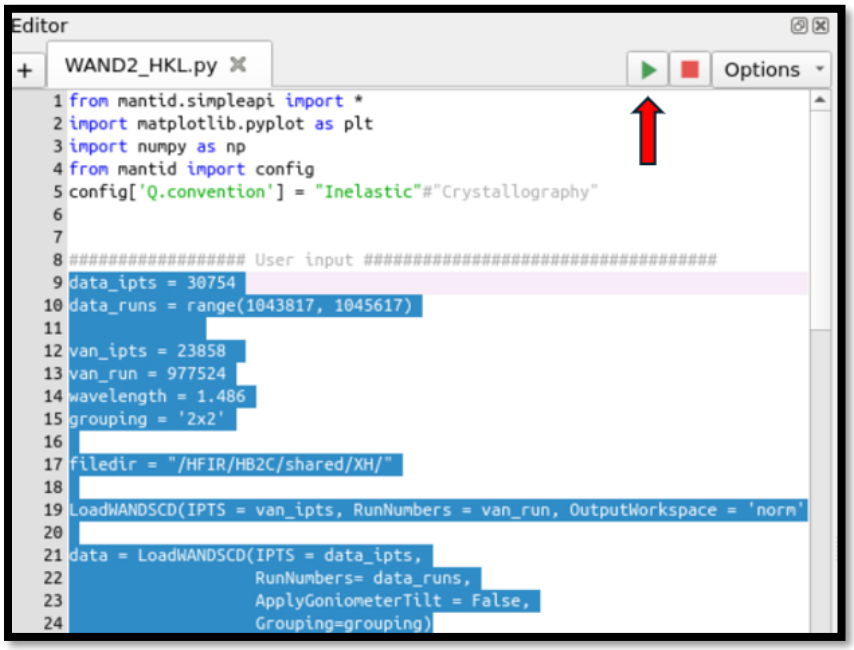


(b) Execute (entire or part of) the script

(1) To execute the entire script, click **Options** on the upper right corner and select **Run All.**



(2) To execute part of the script, highlight the target part and click the **Run** button (**green triangle**).



Back to Index

IV. Data Analysis

The common data analysis workflow for WAND^2 data involves two parts:

- Data visualization in reciprocal HKL maps
- Peak analysis for refinement

The Mantid script including the two parts can be found in the analysis cluster by following the path "data/HFIR/HB2C/shared/WANDscripts/SingleCrystal" and locate the script:

WAND_SCD_Reduction.py

Copy both the **WAND_SCD_Reduction.py** and **peakintegration_2D.py** into the shared folder under your IPTS number. Open the script as described in III.

The script contains following blocks:

- Block 1 --- Load data
- Block 2 --- Create Q (Å^{-1}) workspace for peak analysis
- Block 3A --- Predict parent (nuclear) peaks
- Block 3B --- Index, center, filter parent peaks
- Block 4A --- Integrate parent peaks – method 1: 3D ellipsoid integration
- Block 4B --- Filter and save parent peaks – method 1
- Block 4C --- Integrate parent peaks – method 2: detector image integration
- Block 4D --- Filter and save parent peaks – method 2
- Block 5 --- Create normalized HKL map
- Block 6A --- Predict satellite peaks
- Block 6B --- Filter satellite peaks
- Block 7A --- Integrate satellite peaks – method 1: 3D ellipsoid integration
- Block 7B --- Filter and save satellite peaks – method 1
- Block 7C --- Integrate satellite peaks – method 2: detector image integration
- Block 7D --- Filter and save satellite peaks – method 2

In the following sections, we will go through each block and explain some details regarding algorithms and practical workflow. Each block begins with parameters, which need to be optimized based on the data. It may be necessary to re-run single blocks at a time to optimize the reduction.
To run the script with default parameters, the minimum parameters to be edited are the ones in Block 1, Block 2 and Block 5.

Back to Index

(1) Block 1 --- Load Data

```
from mantid.simpleapi import *
import matplotlib.pyplot as plt
import numpy as np
from pathlib import Path
from mantid import config
import h5py
import re

# Default is intelastic and predict satellite peaks doesn't work with crystallography
config['Q.convention'] = "Inelastic"
#config['Q.convention'] = "Crystallography"

# ================================================================
# BLOCK 1 — LOAD DATA
# ================================================================
#User input starts here
data_ipts = 22745 # experiment IPTS number
data_runs = list(range(474136, 475936))   # scan IDs, python list
van_ipts = 23858
van_run = 1708902
wavelength = 1.486
normalizeby = "Monitor" # choices: "Monitor", "Time", "None"
grouping = "4x4" # choices: "2x2", "4x4", "None"
#User input for this block ends here

# Sort scans in s1 array
filename = '/HFIR/HB2C/IPTS-{}/nexus/HB2C_{}.nxs.h5'
filenames = [filename.format(data_ipts, run) for run in data_runs]
s1_array = np.array([])
scanidx = np.array([])
for n, run in enumerate(filenames):
    num = re.findall(r"\d+", run)
    #print("Loading scan_NO.{}".format(num[-2]))#, run)
    scanidx = np.append(scanidx, int(num[-2]))
    with h5py.File(run, "r") as f:
        s1_array = np.append(s1_array, f["/entry/DASlogs/HB2C:Mot:s1.RBV/average_value"][0])
    res = np.c_[scanidx, s1_array]
data_runs = res[res[:,1].argsort(),0]


# Load the detector images, normalized by the vanadium data
data = LoadWANDSCD(IPTS = data_ipts,
                   RunNumbers = data_runs,
                   VanadiumIPTS = van_ipts,
                   VanadiumRunNumber = van_run,
                   NormalizedBy = normalizeby.capitalize(),
                   Grouping = grouping,
                   ApplyGoniometerTilt = True)

SetGoniometer(Workspace = 'data',
          Axis0 = 'HB2C:Mot:s1.RBV,0,1,0,1',
          Average = False)

# ouput directories ------------------------------------------------------------
directory = '/HFIR/HB2C/IPTS-{}/shared/method1/'.format(data_ipts)
peaklist_outdir = '/HFIR/HB2C/IPTS-{}/shared/method2/'.format(data_ipts)
Path(peaklist_outdir).mkdir(parents = True, exist_ok = True)
#-------------------------------------------------------------------------------
```

Input parameters:

- **Experiment IPTS number**
- **Experiment scan IDs**

  It should be in a Python list format, such as
  - [scan1, scan2, scan3, …]
  - range(scan_start, scan_end)
  - list(…) + list(…) (see above example)
- **Vanadium IPTS number and run number**

Back to Index

- **Wavelength**
- **Normalizeby**

  The concept "Normalization" refers to that neutron intensity collected by detector pixels needs to be normalized by the detector pixel efficiency and rescaled by scan-related parameters, such as monitor counts, measurement time, or none. This is a process of intensity correction to avoid instrumental bias.
- **Grouping**

  The raw detector image has 3840 x 512 pixels. Grouping helps reduce the data size. "2x2" means every 2x2 pixels are grouped into 1 larger pixel.

After loading the data, you can save the resulting workspace using ***SaveMD*** command line so that if later you need to redo the data analysis, you don't need to reload everything from scratch but use ***LoadMD*** to reload the workspace directly.

- **Output directories**

  The two directories, **directory** and **peaklist_outdir**, are directories to save any output files.

(2) Block 2 --- Create Q Workspace for Peak Analysis

```
# ====================================================================
# BLOCK 2 – CREATE Q (angstrom^-1) WORKSPACE FOR PEAK ANALYSIS
# ====================================================================
#User input starts here
#Normal Position for Detector Q_y [-0.6, 0.6]; 60 mm position for detector Q_y [-0.95,0.25]; 100mm position Q_y [-1.2, 0]
q_min = [-7.5, -0.95, -7.5] # reciprocal range (Qx, Qy, Qz), unit: angstrom^-1
q_max = [+7.5, +0.25, +7.5]
wavelength = 1.486
#User input for this block ends here

data_in_Q = ConvertHFIRSCDtoMDE(InputWorkspace = "data",
                                Wavelength = wavelength,
                                LorentzCorrection = True,
                                ObliquityParallaxCoefficient = 1.022, # no change
                                MinValues='{},{},{}'.format(*q_min),
                                MaxValues='{},{},{}'.format(*q_max))
```

Input parameters:

- **q_min, q_max**

  These two Python lists determine the range of the reciprocal space (in the unit of Å^{-1}). In most cases, $[\pm 10, \pm 1.2, \pm 10]$ will be enough.

The Mantid algorithm ***ConvertHFIRSCDtoMDE*** converts the detector images into reciprocal Q space. **LorentzCorrection** is a volume rescaling process between real space and reciprocal space, which is necessary for proper intensity histogram.

Back to Index

(3) Block 3A --- Predict Parent (nuclear) Peaks

```
# ======================================================================
# BLOCK 3A — PREDICT NUCLEAR PEAKS
# ======================================================================
#User input starts here
reflection_condition = "Primitive"
#  Choices:
# 'Primitive'
# 'C-face centred', 'A-face centred', 'B-face centred',
# 'Body centred', 'All-face centred',
# 'Rhombohedrally centred, obverse',
# 'Rhombohedrally centred, reverse',
# 'Hexagonally centred, reverse'
d_min = 0.8
#User input for this block ends here

predict = PredictPeaks(InputWorkspace = "data_in_Q",
                       ReflectionCondition = reflection_condition,
                       CalculateGoniometerForCW = True,
                       MinDSpacing = d_min,
                       Wavelength = wavelength,
                       OutputType='LeanElasticPeak')

predict = IntegratePeaksMD(InputWorkspace = "data_in_Q",
                           Peakradius = 0.15,
                           Backgroundinnerradius = 0.20,
                           Backgroundouterradius = 0.25,
                           PeaksWorkspace = "predict")

predict = FilterPeaks(InputWorkspace = "predict", FilterVariable = "Intensity", FilterValue = 0, Operator = ">")
```

Input parameters:

- **reflection_condition**
  Symmetry dependent
- **d_min**
  Minimum d-spacing for peak predictions

The Mantid algorithm ***PredictPeaks*** will generate a peak workspace, containing all possible Bragg peaks based on the input reflection conditions. Then a simple integration at each predicted peak position is performed using ***IntegratePeaksMD*** with a default 3D sphere model in the reciprocal space. Finally, ***FilterPeaks*** with filtering criterion of Intensity > 0 will remove peaks that have intensities <= 0. These removed peaks usually sit outside the detection range.

Back to Index

(4) Block 3B --- Index, Center, Filter Parent Peaks

```
# ======================================================================
# BLOCK 3B — INDEX, CENTER, FILTER PARENT PEAKS
# Inspect peak workspace "nuclear" in SliceViewer
# Rerun command lines ACCORDINGLY in this block after manually adding/removing peaks
# ======================================================================
#User input starts here
# Centroid peaks
centroid_radius = 0.1
# Index peaks
tolerance = 0.05
# Filter peaks with zero intensity
filtervariable = "Intensity"
filtervalue = 0
operation = ">"
#User input for this block ends here

CloneWorkspace(InputWorkspace = "predict", OutputWorkspace = "nuclear")

wavelength = 1.486
HFIRCalculateGoniometer(Workspace = "nuclear", Wavelength = wavelength)

IndexPeaks(PeaksWorkspace = "nuclear", Tolerance = tolerance, RoundHKLs=True)


CentroidPeaksMD(InputWorkspace = "data_in_Q",
                PeaksWorkspace = "nuclear",
                PeakRadius = centroid_radius,
                OutputWorkspace = "nuclear")

IntegratePeaksMD(InputWorkspace = "data_in_Q",
                 Peakradius = '0.08,0.12,0.08',
                 Backgroundinnerradius = '0.08,0.12,0.08',
                 Backgroundouterradius = '0.1,0.15,0.1',
                 PeaksWorkspace = "nuclear",
                 OutputWorkspace = "nuclear",
                 Ellipsoid=True,
                 IntegrateIfOnEdge=False)


FilterPeaks(InputWorkspace = "nuclear",
            FilterVariable = filtervariable,
            FilterValue = filtervalue,
            Operator = operation,
            OutputWorkspace = "nuclear")

# ======================================================================
# PAUSE — INSPECT 'nuclear' IN SLICEVIEWER
# ======================================================================
```

Input parameters:

- **centroid_radius**

  CentroidPeaksMD generates a 3D sphere with a radius of **centroid_radius** centered at each predicted peak position. Then it calculates the intensity-weighted center within the sphere for each peak and shifts the peak position to this new center.

- **Tolerance**

  HKL-indexing strictness

- **filtervariable, filtervalue, operation**

  These 3 inputs help ***FilterPeaks*** modify the peak workspace. “Intensity > 0” means it will remove those peaks with intensity <= 0.

Preset values work well in most cases. Parameters are free to change by users.

Back to Index

As indicated by the comment **“Pause — Inspect ‘nuclear’ in SliceViewer,”** users should pause at this stage and manually verify that the peak workspace **nuclear** contains the expected peaks and that their positions are correctly identified. Peaks may be added to or removed from the workspace as needed.

Newly added peaks are initially assigned an HKL index of (0, 0, 0) and may also have an arbitrary wavelength. Therefore, after adding peaks, users should rerun the relevant ***HFIRCalculateGoniometer*** and ***IndexPeaks*** commands, followed by ***CentroidPeaksMD*** if further refinement of the peak positions is needed.

These commands are intentionally placed at this point in the script so that users can execute them iteratively before peak integration. The goal is to ensure that all expected peaks are included, no relevant peaks are missing, and the peak positions are accurately captured.

Detailed instructions for adding, removing, and inspecting peaks in the Mantid SliceViewer are provided in the Appendix.

Back to Index

(5) Block 4A --- Integrate Parent Peaks – Method 1: 3D Ellipsoid Integration

```
# ======================================================================
# BLOCK 4A — INTEGRATE PARENT PEAKS ---> Method 1: 3D ellipsoid integration in reciprocal space
# ======================================================================
#User input starts here
ellipsoid_radius = [0.08,0.12,0.08]
innerbkg_radius = [0.08,0.12,0.08]
outerbkg_radius = [0.1,0.15,0.1]
#User input for this block ends here

IntegratePeaksMD(InputWorkspace = "data_in_Q",
                 PeakRadius = ellipsoid_radius,
                 PeaksWorkspace = 'nuclear',
                 BackgroundInnerRadius = innerbkg_radius,
                 BackgroundOuterRadius = outerbkg_radius,
                 Ellipsoid = True,
                 IntegrateIfOnEdge = False,
                 OutputWorkspace = 'nuclear')

monitor_cnt = mtd['data_in_Q'].getExperimentInfo(0).run().getProperty('monitor_count').value
time_array  = mtd['data_in_Q'].getExperimentInfo(0).run().getProperty('duration').value
flux = monitor_cnt / time_array
flux_err = np.std(flux)*6/np.mean(flux) # +/- 3 std fluctuation with respect to average flux

pws = mtd['nuclear']
for no in range(pws.getNumberPeaks()):
    p = pws.getPeak(no)
    pI, perr = p.getIntensity(), p.getSigmaIntensity()
    perr_new = np.sqrt( (pI*flux_err)**2 + perr**2 )
    p.setSigmaIntensity( perr_new )

# ======================================================================
# PAUSE — INSPECT INTEGRATION IN SLICEVIEWER, open 'nuclear' in SliceViewer
# ======================================================================
```

Input parameters:

- **ellipsoid_radius**
- **innerbkg_radius / outerbkg_radius**

Here we use the 3D ellipsoid model so 3 values (3 semi-axis lengths) are needed for each input.

The inner and outer background radius define a 3D ellipsoidal shell outside and co-centered with the peak integration ellipsoid. The background intensity within the peak integration ellipsoid is calculated as:

$$I_{bkg} = \frac{I_{shell}}{V_{shell}} \times V_{peak}$$

The latter part ($\mathrm{monitor\_cnt}$, $\mathrm{time\_array}$, flux, ……) adjusts the intensity error bar with the neutron beam fluctuations.

Again, please remember to inspect peak integration in Mantid SliceViewer. Details in Appendix.

Back to Index

(6) Block 4B --- Filter and Save Parent Peaks – Method 1

```
# ======================================================================
# PAUSE – INSPECT INTEGRATION IN SLICEVIEWER, open peak_to_integrate workspace in SliceViewer
# ======================================================================
# ======================================================================
# BLOCK 4B – Filter and Save PARENT peaks - Method1
# ======================================================================
#User input starts here
# Filter non-existent peaks
filtervariable = "Intensity"
filtervalue = 100
operation = ">"
#FileFormat for integrated intensities "Fullprof", "GSAS", 'Jana', 'SHELX'
format="Fullprof"
#User input for this block ends here
nuclear_temp=FilterPeaks(InputWorkspace = "nuclear",
            FilterVariable = filtervariable,
            FilterValue = filtervalue,
            Operator = operation,
            )
SaveReflections(mtd['nuclear_temp'], Format=format, Filename = directory + 'nuclear.int')
DeleteWorkspace(nuclear_temp)
```

Input parameters:

- **filtervariable**
  Common options are “Intensity”, “Signal/Noise”
- **filtervalue**
  Needs to be adjusted after checking the output to filter out non-existent peaks
- **operation**
  “<”, “>”, “!=”, “<=”,…
- **format**
  Fullprof is default. Available choices: **GSAS, Jana, SHELX.**

Back to Index

(7) Block 4C -- Integrate Parent Peaks – Method 2: Detector Image Integration

```
# ======================================================================
# BLOCK 4C – INTEGRATE PEAKS ---> Method 2: Detector image integration
# ======================================================================
#User input starts here
optimize_pos = False  # peak position optimization control
ROI_size = [15,15,30,30] # ---- peak pixel size, [x_left, x_right, y_up, y_down]
#User input for this block ends here

import importlib
import peakintegration_2D
importlib.reload(peakintegration_2D)

data_workspace = mtd["data"] # --- detector image workspace
peak_workspace = mtd["nuclear"] # --- peak workspace
output_folder = peaklist_outdir + '/integer/' # --- file output directory path


for no in range(0, peak_workspace.getNumberPeaks()):
    test = peakintegration_2D.peak2D(data_workspace, peak_workspace, no, ROI_size, output_folder, optimize_xy = optimize_pos)

    plt.close()
plt.close('all')

# ======================================================================
# PAUSE – INSPECT INTEGRATION IN OUTPUT DIRECTORY, peak-related png files are stored in the output directory folder
# ======================================================================
```

Input parameters:

- **optimize_pos**

  It decides whether further peak position optimization is performed or not.

- **ROI_size**

  It specifies the box size (ROI size) for peak integration on the detector images. The unit is pixels.

The fluctuation consideration is included in the algorithm. Please inspect the integration in output directory. Peak-related PNG files are stored in the output directory folder.

(8) Block 4D --- Filter and Save Parent Peaks – Method 2

```
# ======================================================================
# BLOCK 4D – Filter and Save PARENT peaks - Method2
# ======================================================================
#User input starts here
# Filter non-existent peaks
filtervariable = "Intensity"
filtervalue = 100
operation = ">"
#FileFormat for integrated intensities "Fullprof", "GSAS", 'Jana', 'SHELX'
format="Fullprof"
#User input for this block ends here
FilterPeaks(InputWorkspace = "nuclear",
            FilterVariable = filtervariable,
            FilterValue = filtervalue,
            Operator = operation,
            OutputWorkspace= 'nuclear'
            )
SaveReflections(mtd['nuclear'], Format=format, Filename = peaklist_outdir + 'nuclear.int')
```

Same workflow as Block 4B.

Back to Index

(9) Block 5 --- Create Normalized HKL Map

```
# ======================================================================
# BLOCK 5 — CREATE NORMALIZED HKL MAP
# ======================================================================
#User input starts here
#HKL Transformation for the projection
proj_axis1 = "1,0,0"
proj_axis2 = "0,1,0"
proj_axis3 = "0,0,1"
#BinningSize and Dimension of the hkl transformation, for u,v Dimension is lattice parameter times 1.2
#for wproj Dimension is lattice parameter times 0.09 (0mm detector), times 0.142 (60mm detector), times 0.177 (100mm detector)
bin_axis1 = "-5.3,5.3,900"
bin_axis2 = "-5.3,5.3,900"
bin_axis3 = "-1.42,1.42,64"
#rescale HKL by factor -------------------------
scale = 1.0
#Symmetry Operations ('x, y, z') for no symmetry operation
symops='x, y, z; -x,-y,z'
# UB optimization ------------------------------------------------------------
#['Cubic', 'Tetragonal', 'Orthorhombic', 'Hexagonal', 'Rhombohedral', 'Monoclinic', 'Triclinic']
cell_type = 'Tetragonal'
#User input for this block ends here

#6 Optimize Lattice for Cell Types----------------------
OptimizeLatticeForCellType(PeaksWorkspace='nuclear',
                           CellType=cell_type.title(),
                           Apply=True)

UB_file = None # If you have UB saved somewhere, replace None with the path: "xxx/xxx/.../xxx.mat"
if UB_file: LoadIsawUB("data", Filename = UB_file)

HKL = ConvertWANDSCDtoQ(InputWorkspace = "data",
                        UBWorkspace='nuclear',
                        NormaliseBy = "None", # no need for normalization since "data" is already normalized
                        Frame = "HKL", # choices: "HKL", "Q_sample"
                        Uproj = proj_axis1,
                        Vproj = proj_axis2,
                        Wproj = proj_axis3,
                        BinningDim0 = bin_axis1,
                        BinningDim1 = bin_axis2,
                        BinningDim2 = bin_axis3,
                        SymmetryOperations = symops,
                        )

CreateSingleValuedWorkspace(OutputWorkspace = 'singlevalue', DataValue=scale, ErrorValue=0)
MultiplyMD(LHSWorkspace='HKL', RHSWorkspace='singlevalue', OutputWorkspace='HKL')
```

Input parameters:

- **Projection axes (proj_axis1, proj_axis2, proj_axis3)**
  HKL maps also refer to reciprocal lattice unit (r. l. u.) maps. You need to decide how you want to visualize the data. The default axes are along canonical [1,0,0], [0,1,0], and [0,0,1] directions in the reciprocal space. In some special situations, users may also want to plot the data along for example [1,1,0], [1,-1,0], and [0,0,1].
- **Data binning dimensionality (bin_axis1, bin_axis2, bin_axis3)**
  As the name suggests, it determines how you want to histogram the data. Each bin_axis string follows the format "start, end, number-of-points-in-between".
- **Scale (optional)**
  A scalar factor to rescale the intensity and error bar in workspace "HKL".
- **Symops (optional)**
  Symmetry operations are performed for data folding. If you don't need it, just put **symops = "x, y, z"**. For a full reciprocal map in an orthogonal system, the line is

Back to Index

“symops=’x, y, z: -x, -y, z”. For hexagonal cases, the transformation is (y, x, -z) for H0L and (-y, -x, -z) for HHL.

- **cell_type**
  Lattice symmetry for pre-optimization of the UB matrix.

In practice, during the experiment setup, sample alignment will be done, and the UB matrix will be determined and recorded into the data files. Thus, in general UB file is not explicitly needed. But if you have a specific UB matrix file (e.g., obtained from UB matrix optimization discussed in Appendix), you can load it and set it up for the data using ***LoadIsawUB***.

If you want to save/load the HKL map, you can use ***SaveMD***/***LoadMD***. If you want to output the HKL map, please refer to Appendix. To inspect HKL data, please refer to details in Appendix.

Back to Index

(10) Block 6A --- Predict Satellite Peaks

```
# ======================================================================
# BLOCK 6A — Predict SATELLITE PEAKS
# ======================================================================
#User input starts here
mod_vector1 = [0.2,0,0]
mod_vector2 = [0,0,0]
mod_vector3 = [0,0,0]
cross_term = False # Whether including combinations of mod_vectors as mod_vectors
max_order = 1 # maximum order of satellite peaks
includeinteger = False #include integer peaks in magnetc peak prediction
wavelength = 1.486
#User input starts here

satellite = PredictSatellitePeaks(Peaks = "nuclear",
                                  ModVector1 = mod_vector1,
                                  ModVector2 = mod_vector2,
                                  ModVector3 = mod_vector3,
                                  IncludeIntegerHKL = includeinteger,
                                  MaxOrder = max_order,
                                  CrossTerms = cross_term)
```

Input parameters:

- **mod_vectors1, 2, 3**
- **cross_term**
- **max_order**
- **includeinteger**
- **wavelength**

The modulation vectors are specified in **reciprocal lattice units (r.l.u.)**, i.e., as fractional values of H, K, and L. They define the propagation vectors required to describe satellite reflections relative to the parent Bragg reflections (integer HKL).

You do not need to use all three modulation vector inputs — only those required by your crystal structure. Setting **IncludeIntegerHKL = False** excludes the parent Bragg reflections, i.e., reflections corresponding to the zero-modulation vector (mod_vector = [0, 0, 0]), from the predicted satellite peak list.

The **CrossTerms** option in ***PredictSatellitePeaks*** determines whether linear combinations of the input modulation vectors should also be considered when predicting satellite reflections. For example, if mod_vector1 = [0.5,0,0] and mod_vector2 = [0,0.5,0], then **CrossTerms = True** will additionally predict satellite peaks corresponding to combined modulation vector mod_vector = [0.5,0.5,0]. This option should be enabled only when such mixed-order satellite peaks are expected from the modulation symmetry of the crystal.

Back to Index

(11) Block 6B, 7A, 7B, 7C, 7D --- Satellite Peak Analysis

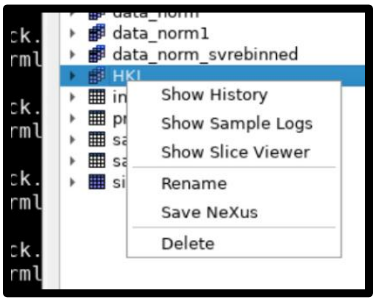
These blocks are essentially the same as those for parent peaks, except that the filtering parameters in ***FilterPeaks*** may need modifications, depending on the relative statistics of satellite peaks.

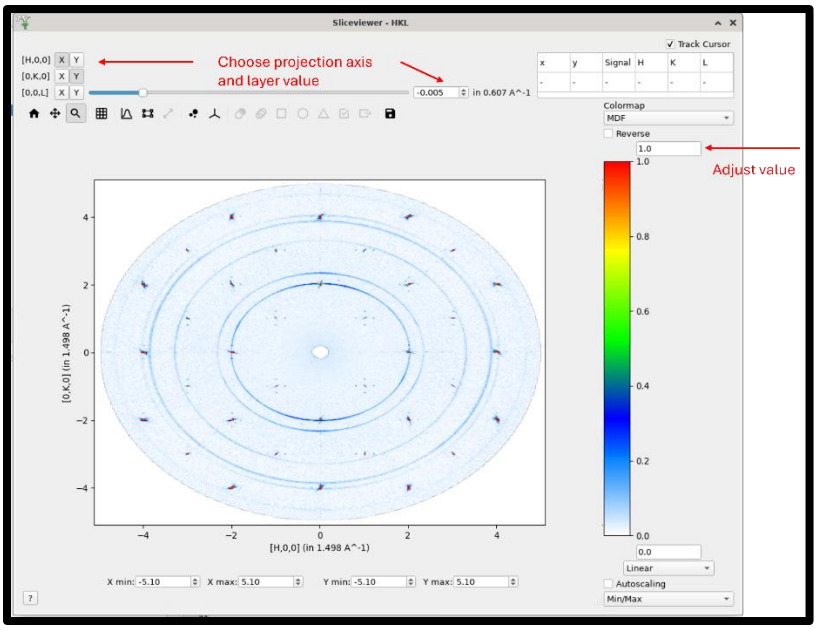
Back to Index

Appendix

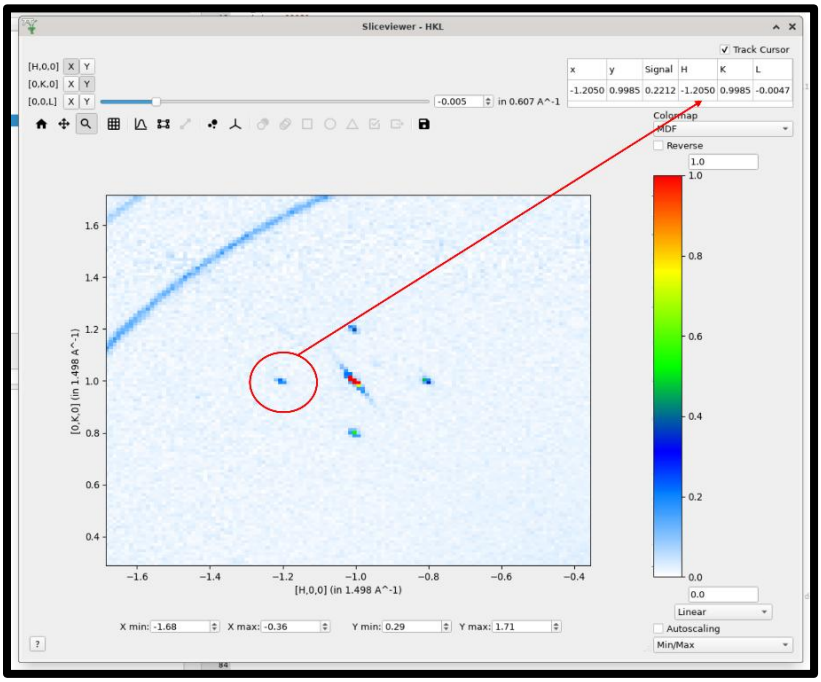
(a) Inspect HKL data

The reason to take this step is to check any additional peaks missing from previous analysis and to verify the correctness of the UB matrix. The UB matrix for the HKL conversion is the one which is used in peak integration. Right click on the HKL workspace in the left-hand side “Workspaces” panel. A dropdown menu allows to use the “Slice Viewer”. Sample logs contain sample environment values during the measurement.





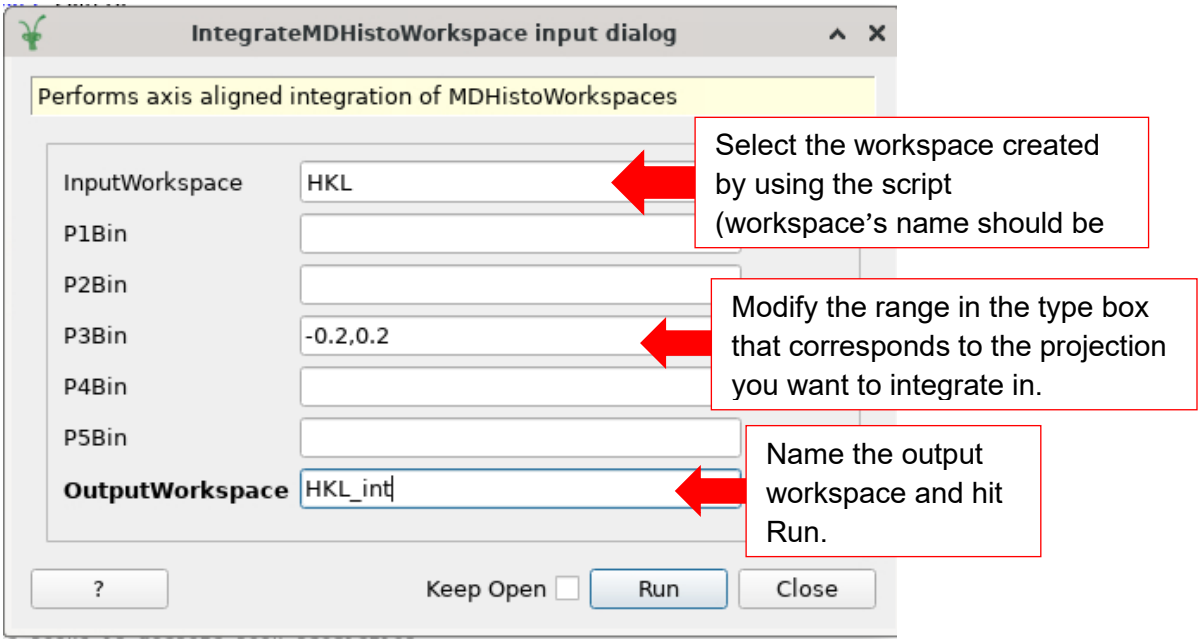
In this tutorial example, zooming in on one of the nuclear peaks, we find satellite peaks corresponding to a propagation vector (0.2, 0, 0).



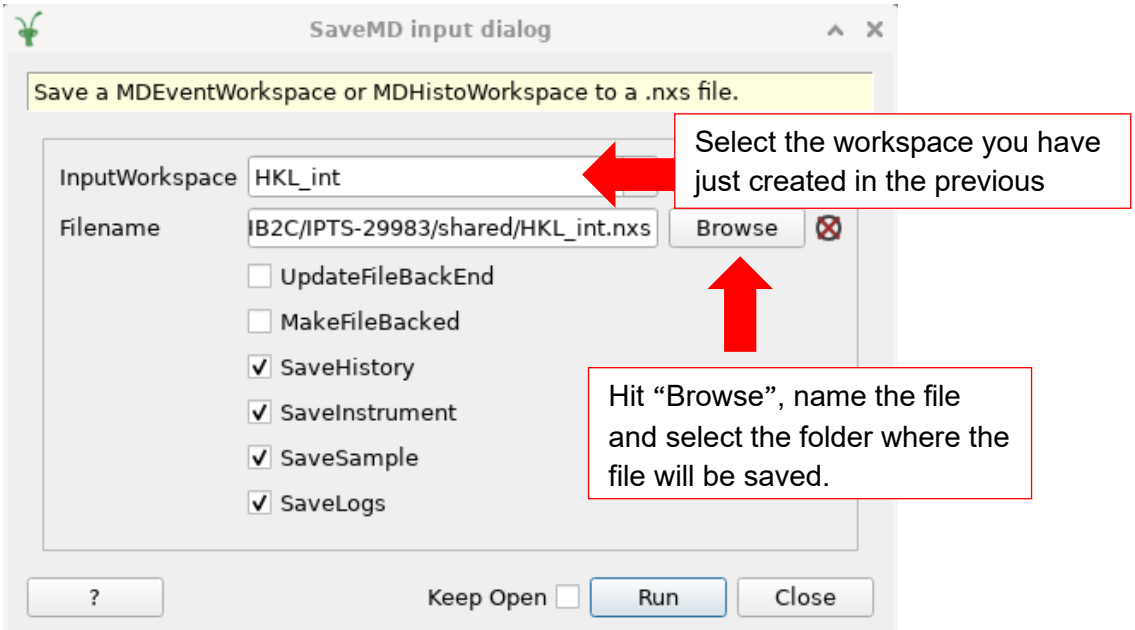
Back to Index

(b) Export HKL data for plotting

For plotting in a different program, a 2D slice can be exported as follows: Go to "Algorithms" panel on the left-hand side of Mantid interface, type in "***IntegrateMDHistoWorkspace***", hit Execute. This algorithm will perform axis aligned integration of MDHistoWorkspace.



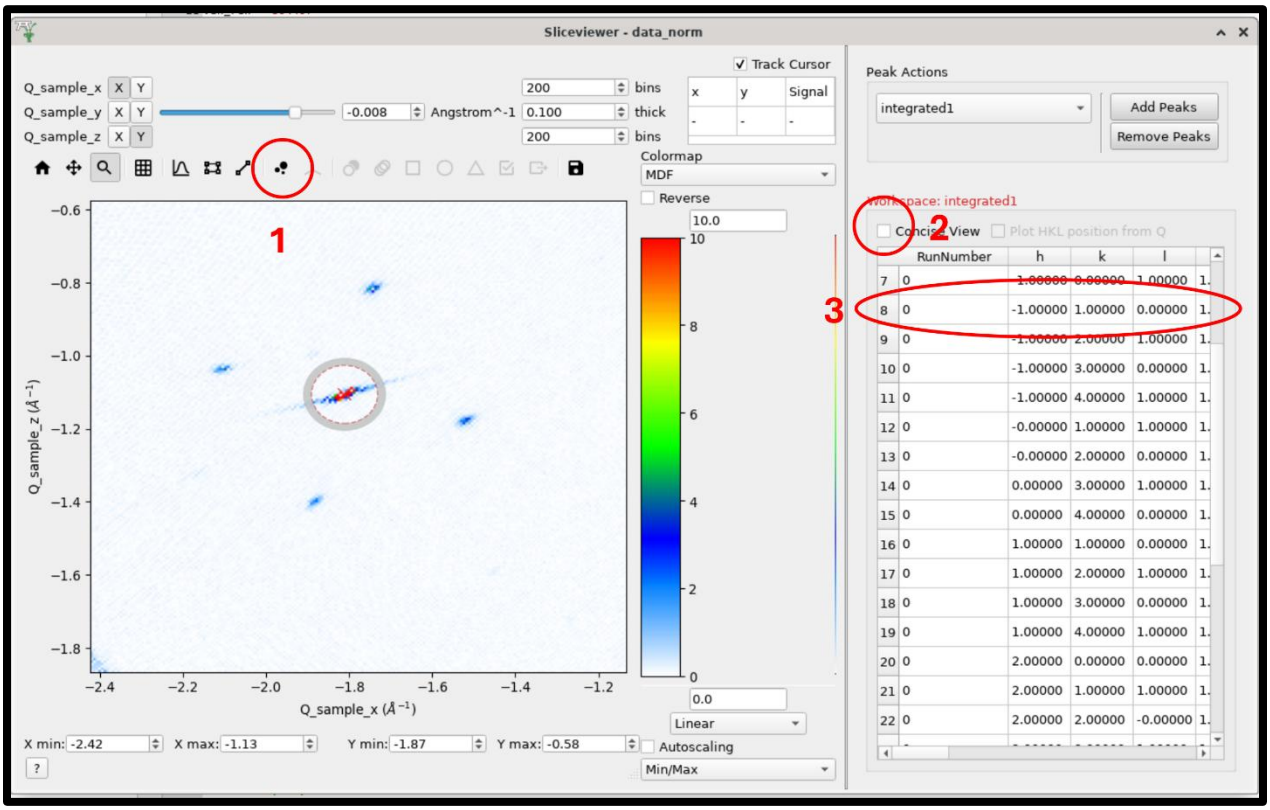
Go to "Algorithms" panel, type in "**SaveMD**", hit Execute. This will save an MDEventWorkspace or MDHistoWorkspace to a .nxs file that can be read by most programs (extension may be changed to .hdf5)



Back to Index

(c) Inspect peaks in Slice Viewer

(1) Right-click **data_in_Q** in the "Workspaces" window and select **Show Slice Viewer**.
(2) In the slice viewer window, click "Add peaks overlays" button (step **1**), shown below.
(3) Check the target peak workspace in the pop-up window and click OK. The workspace content will be listed on the right-hand side.
(4) Check "Concise View" (step **2**).
(5) Click on any peak in the right-hand side window (step **3**), for example (-1, 1, 0) here. The peak statistics will be highlighted in the left-hand side contour plot. The **red cross** in the plot is the peak center. Details will be covered in the later "Integrate peaks" section.



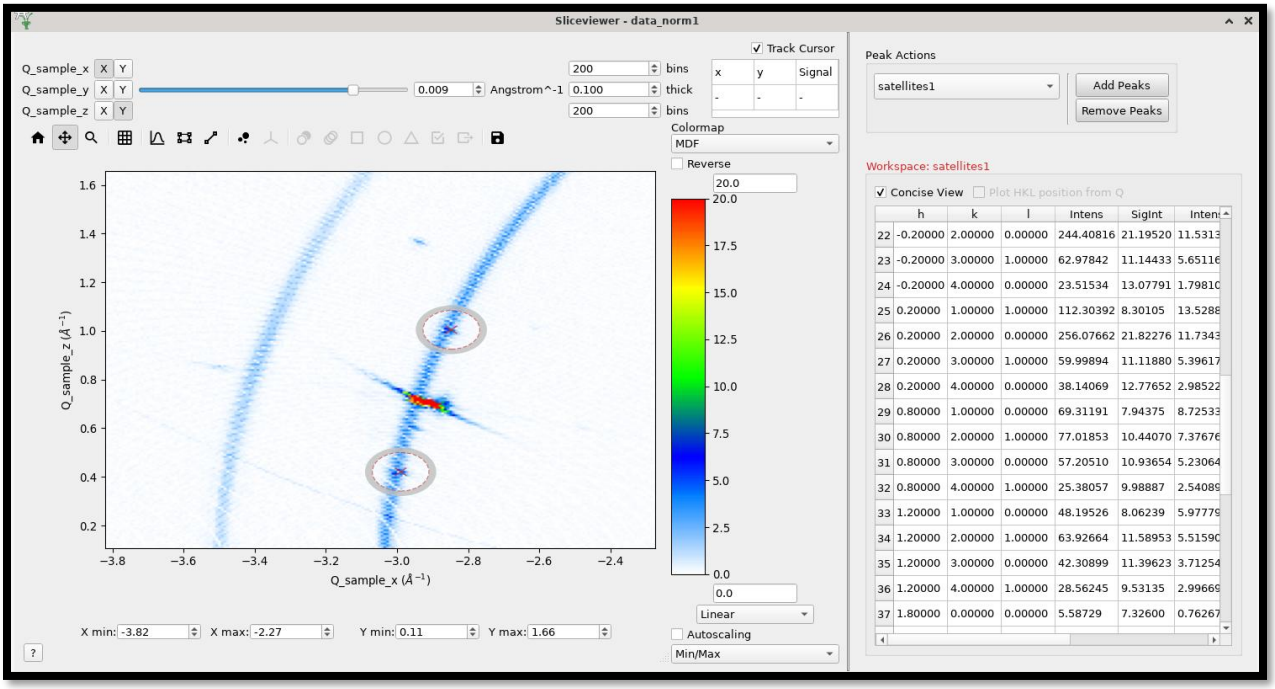
In this example we are checking the position and the integration range for the nuclear peak. In addition, we have a look at the background. The integration range is good; the main peak is covered. The "streak" is intrinsic to position sensitive gas detectors and is in the range of 0.3% to the peak intensity.

- Click on any peak to check integration statistics. The integration ellipsoid will be visualized as superimposed on the peak, shown above.
- The **red cross** is the peak center.
- The **red dash ellipse** is the 3D ellipsoid range.

Back to Index

- The **grey ring** is the region of background estimation, determined by **BackgroundInner/OuterRadius.**
- The error (SigInt shown in the table below) of the integrated peak intensity obtained from ***IntegratePeaksMD*** is highly dependent on **BackgroundInner/OuterRadius** (try different settings to see the dependence). The percentage uncertainty is additionally propagated to the intensity error to alleviate the problem.
- *Details of **IntegratePeaksMD** can be found on https://docs.mantidproject.org/nightly/algorithms/IntegratePeaksMD-v2.html.*

Example for the limitation of Method 1. The peaks shown in the picture below are the magnetic satellites (0, 2, 0) $\pm\tau$. The position is correct, but their intensity is comparable to the underlying powder line. A visual comparison shows that their intensity is comparable to magnetic peaks on uniform background (1, 1, 0) $\pm\tau$ However, the intensity from the integration is two times larger. This is due to the unsuccessful subtraction of the background from the powder line.

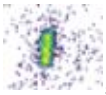


To inspect the peak integrated with Method 2, in a file browser open the output folder defined in script execution. In this folder a PNG for each peak has been generated. In this PNG file, the title shows the target peak (with index), and 3 sections are summarized:

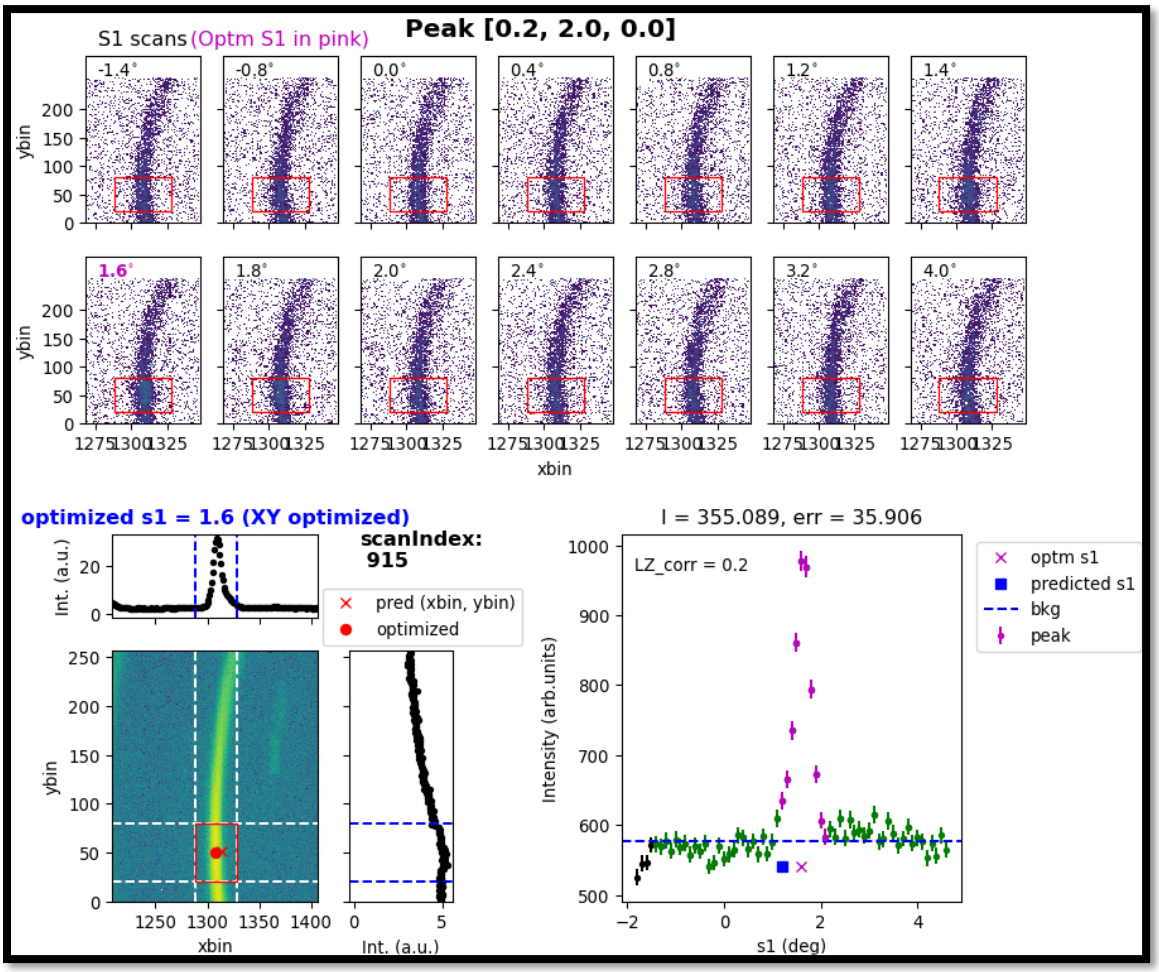
Back to Index

Detector image integration:

- In the upper half of the PNG file, a series of detector images are listed. Each image corresponds to an s1 motor value (the numerical value shown in each image).

- In each detector image, one colored spot, something like this  , corresponds to one peak in the logarithmic color scale. A peak will gradually appear in the detector image and then fade away as s1 motor rotates, as observed in the example below for peak (0.2, 2, 0). The image with **bold magenta** s1 value refers to where peak intensity collected in the detector image reaches its maximum as s1 motor rotates.

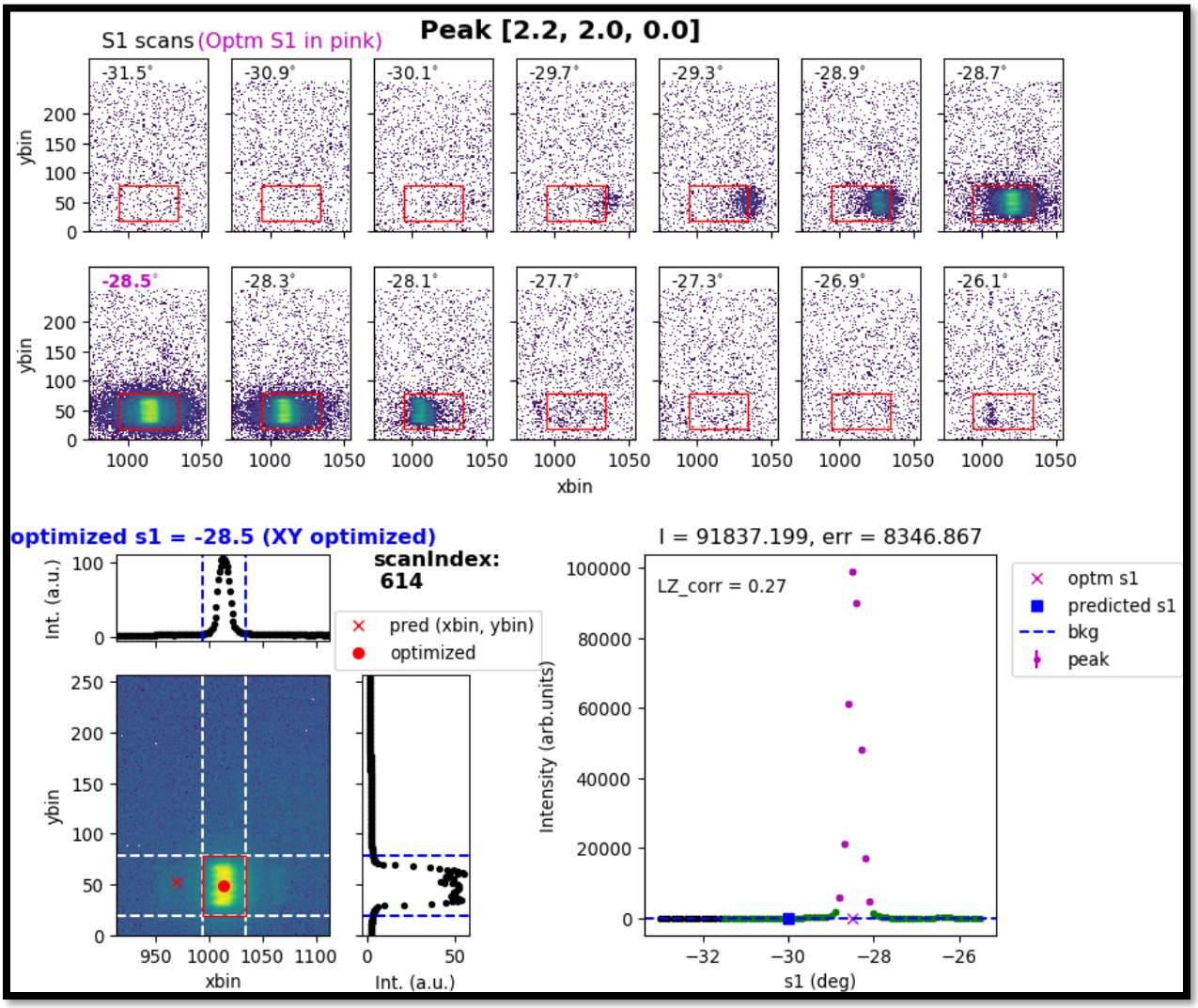
The **red box** in each image highlights the integration region. Make sure the peak intensity is covered by the **red box**. The size of the **red box** is determined by the input 4-number array (ROI_size in the input).



Back to Index

Looking at the integration, it becomes clear that Method 2 deals with the integration on the powder line much better and indeed the integrated intensity values are now similar to the (1,1,0) $\pm\tau$.

Method 2 is limited when multiple peaks appear in the **red box** of the same detector image, be careful. It could be due to twin domains, or impurity, or very close satellite peaks. In addition, very weak peaks are sometimes mis-recognized by the optimization routine when a stronger signal is present as seen in the PNG below for the (2.2, 2, 0) peak. In such situations, you can turn off the peak position optimization by using **optimize_pos = False**.



Back to Index

(d) Add/Remove/Adjust peaks in Slice Viewer

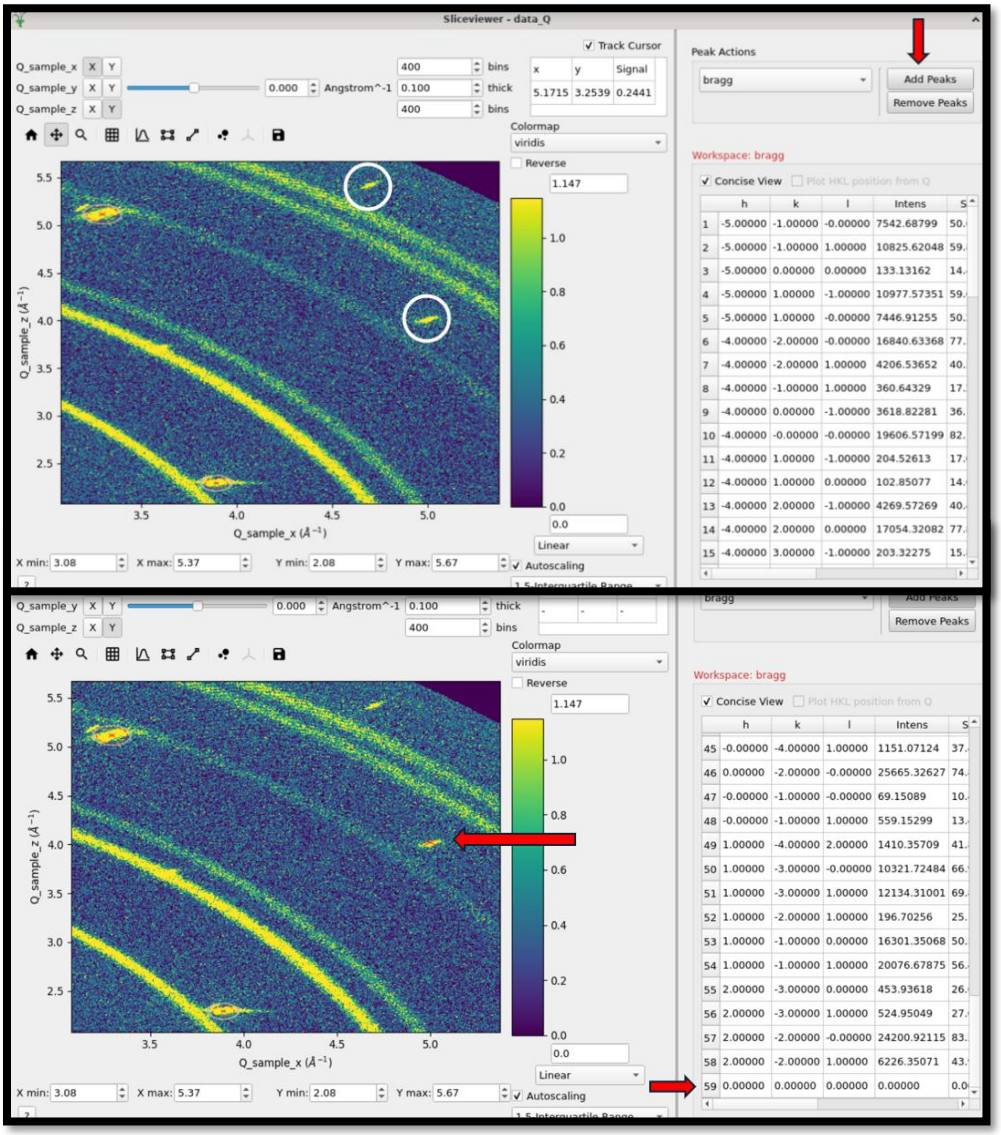
In some rare situations, some of the observed peaks are not included in the peak workspace. For example, in the slice viewer shown below, each peak in the **bragg** workspace will have a **red cross** assigned at its peak position. However, we see that there are two peaks not included in the **bragg** workspace (highlighted by white circles).

To add these two peaks to the peak workspace,

(1) Click on **Add Peaks** button on the upper right corner.

(2) Move the mouse to the peak position and click. A **red cross** will appear at the peak position, and the new peak will be included at the end of the workspace table (right-hand side) with a (0,0,0) index.

(3) Rerun corresponding block command lines of Index, Center, Filter, Integrate peaks.



.

Back to Index

In more common situations, some of the peaks are not well predicted, as shown below. For peak (-4, -1, 1), its predicted peak position (**red cross**) clearly deviates from the peak intensity spot. To adjust the prediction, follow the workflow:

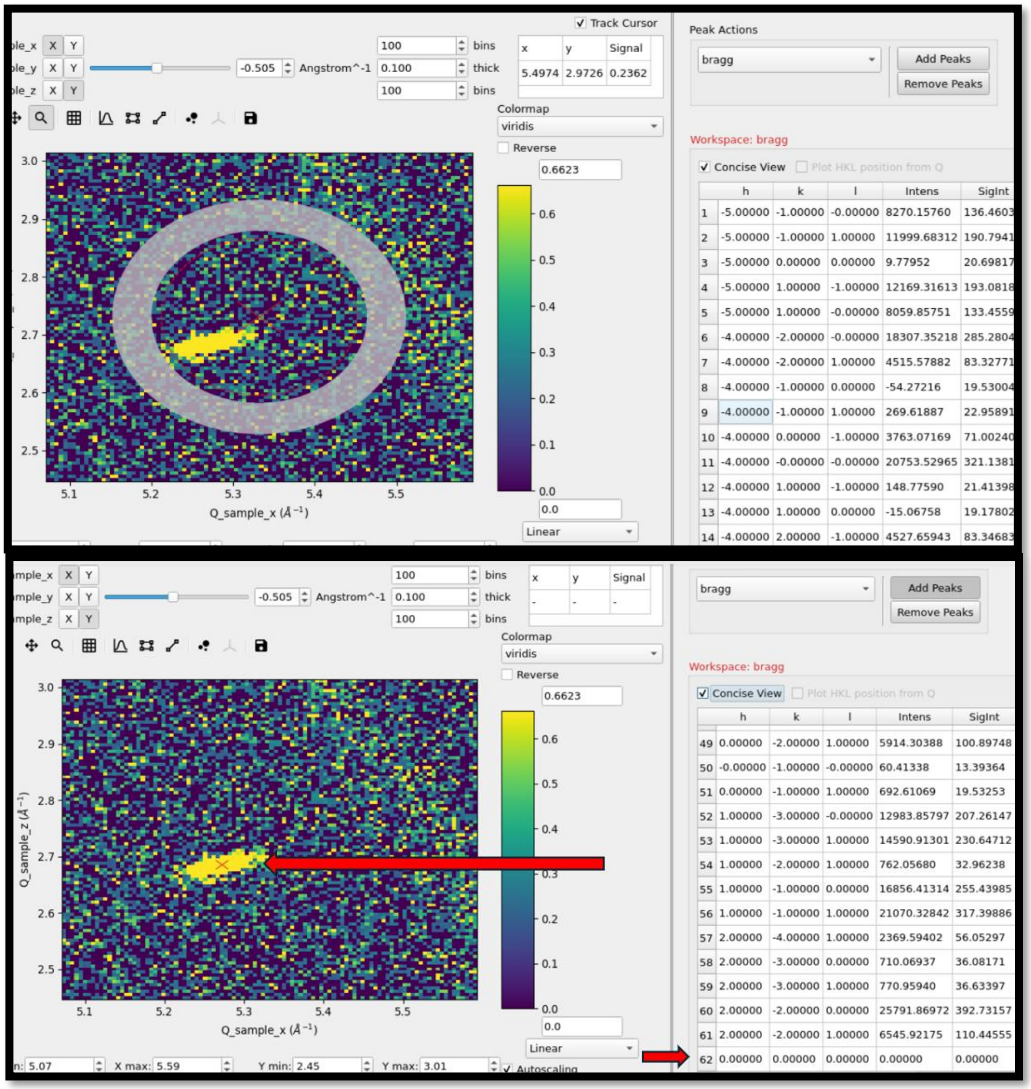
(1) Click on **Remove Peaks** on the upper right

(2) Click on the **red cross** in the color plot (this will remove the predicted peak from the peak workspace)

(3) Click on **Add Peaks** on the upper right

(4) Click at the expected peak intensity spot (a new **red cross** appears at the expected position, shown below). This operation will include the target peak in the peak workspace with a (0,0,0) index with the revised peak position, shown in the second snapshot.

(5) Rerun corresponding block command lines of Index, Center, Filter, Integrate peaks.

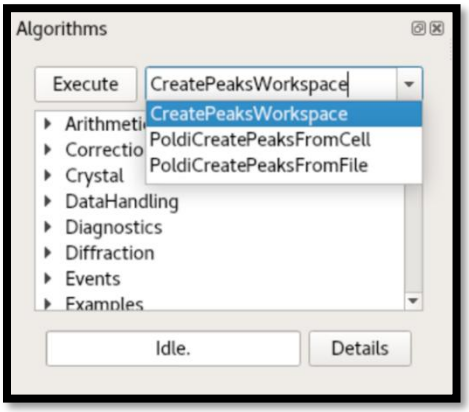


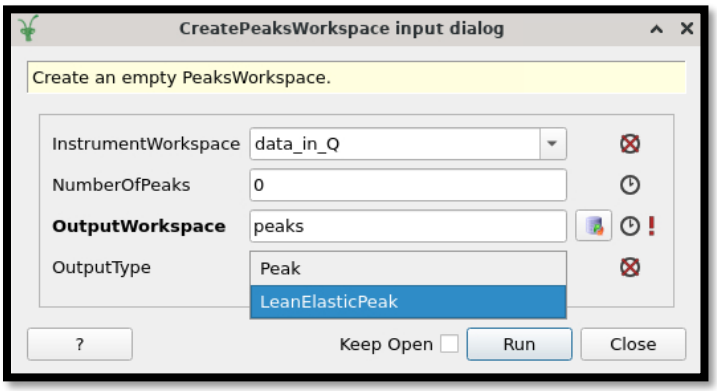
Back to Index

(e) UB matrix determination from scratch

In the case the UB-matrix has not been determined correctly in the experiment set-up or the experiment yielded a commensurate superstructure and a different cell choice is warranted use the following steps:

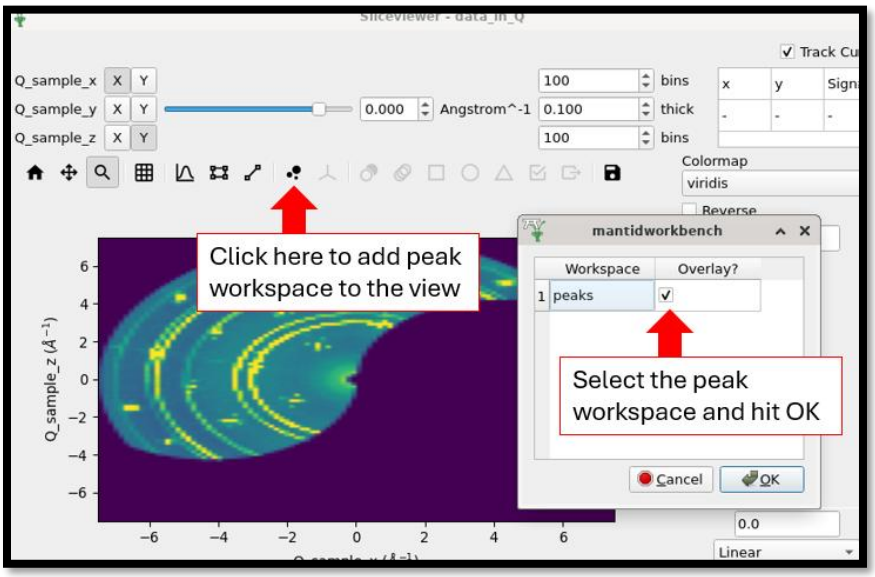
1) After loading all the data and creating the **data_in_Q** workspace, go to Algorithms panel, and type in "***CreatePeaksWorkspace***" and hit Execute. This will create an empty peaks workspace.





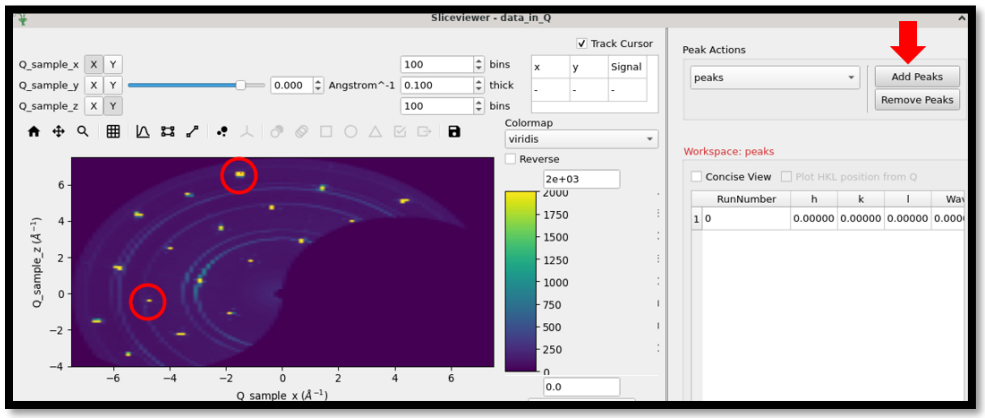
Select **data_in_Q** as the Instrument Workspace; type in 0 for the number of peaks; name the Output Workspace and select "LeanElasticPeak" for the output type. Then hit Run.

(2) Once the workspace **Peaks** is created, right click on **data_in_Q** workspace and select "Show Slice Viewer". Select the created peaks workspace.



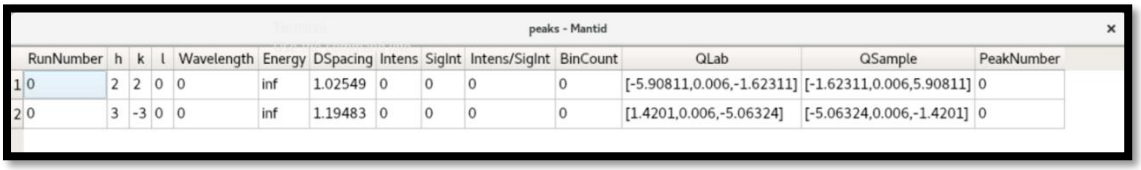
Back to Index

(3) In order to find the UB matrix, we need to find two independent peaks with known HKL values. Click on "**Add peaks**" and select two independent peaks from the slice view.



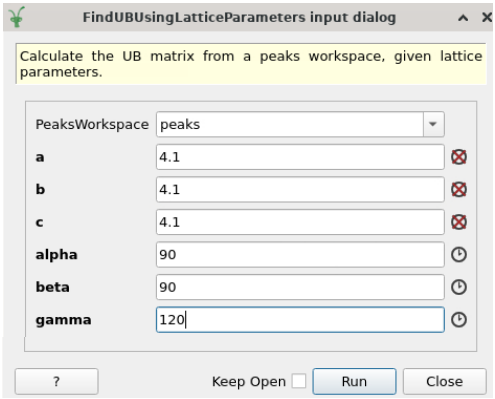
Then click on **Add peaks** again to stop adding. This will add two peaks, without its hkl values.

(4) In order to add HKL values to these peaks, right click on peaks workspace and select "Show Data". This will show the two selected peaks, which will be used to find the UB matrix. Double-click the HKL values of those peaks in the appropriate columns to modify them manually.

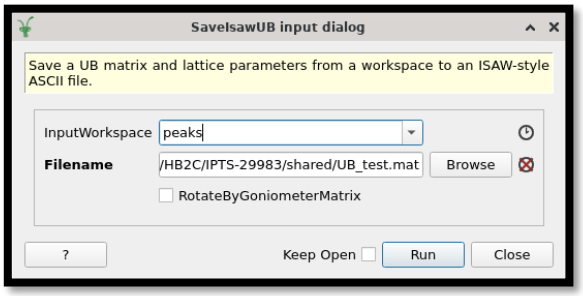


	RunNumber	h	k	l	Wavelength	Energy	DSpacing	Intens	SigInt	Intens/SigInt	BinCount	QLab	QSample	PeakNumber
1	0	2	2	0	0	inf	1.02549	0	0	0	0	[-5.90811,0.006,-1.62311]	[-1.62311,0.006,5.90811]	0
2	0	3	-3	0	0	inf	1.19483	0	0	0	0	[1.4201,0.006,-5.06324]	[-5.06324,0.006,-1.4201]	0

(5) After the peaks have their HKL values assigned, go to Algorithms panel and type in "***FindUBUsingLatticeParameters***", then hit Execute. This will calculate the U matrix from peaks workspace, given the lattice parameters.



(6) Use ***SaveIsawUB*** to save the UB matrix locally.
Up to this point, a reasonable UB matrix is obtained. Next stage could be UB matrix optimization.

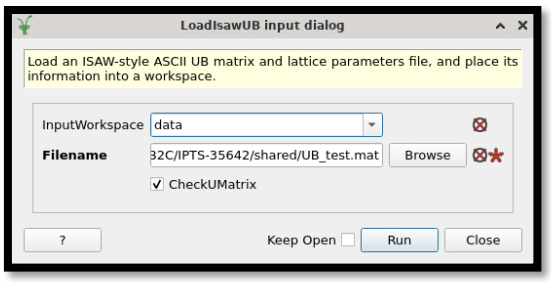


Back to Index

(f) UB matrix optimization

The UB matrix optimization workflow is straightforward. After we have a reasonable UB matrix (either obtained from scratch or in the experiment set-up),

- If you obtain the UB matrix from scratch, you must have a UB matrix file (.mat) saved. Then after you load the data (**Block 1**), use ***LoadIsawUB*** to assign the UB matrix to the workspace **data**.



- If you have the UB matrix determined in the experiment set-up, then you don't need to do anything else; simply load the data (**Block 1**), because the UB matrix is already recorded in the workspace sample logs.

Now we have a workspace **data** with a reasonable UB matrix. To further optimize the UB matrix, we follow steps:

(1) Run **Block 3** convert detector image data to reciprocal Q space

(2) Run **Block 4A** to predict peaks (only integer-HKL peaks, not satellite peaks)

(3) In **Block 4B**, change the filtering criteria to get strong Bragg peaks. For example,

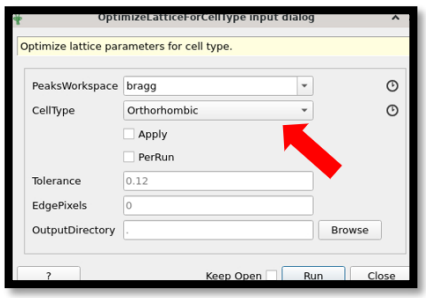
- **filtervariable = Signal/Noise**
- **filtervalue = 50**
- **operator = ">"**

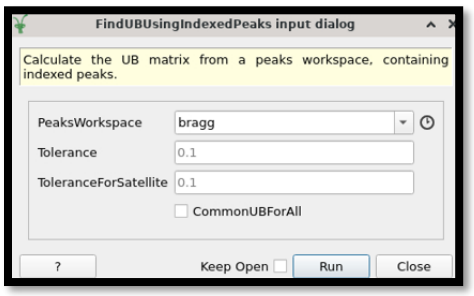
can usually give us very strong Bragg peaks (**filtervalue** is free to change accordingly). About 10 strong peaks are usually enough for UB optimization.

(4) Iteratively run this modified **Block 4B** until nothing changes anymore. DO NOT include **CloneWorkspace** line in iteration.

(5) At this point, we have a peak workspace containing very strong Bragg peaks with HKL indices. Go to Algorithms panel, type in ***OptimizeLatticeForCellType*** or ***FindUBUsingIndexedPeaks***. Both can help optimize the UB matrix based on peak information. Choose expected CellType in ***OptimizeLatticeForCellType***.

Note: ***FindUBUsingIndexedPeaks*** sometimes will fail if peaks do not include linearly independent HKL indices in (H, K, L), e.g., only (H, K, 0) peaks.





Back to Index