HB3A Data Processing Manual via Mantid

February 2026
Xiao Hu
Neutron Scattering Scientist
Oak Ridge National Laboratory

Index

I. Prologue

This manual provides an overview of the general workflow for single-crystal data analysis on the **DEMAND (HB-3A)** diffractometer. The workflow consists of:

- **Data access**
- **Data processing using Mantid Workbench**
- **Data processing using MATLAB**

Data processing on the monochromatic diffractometer DEMAND (HB-3A) generally includes data visualization, data conversion, and peak analysis. An important concept in neutron scattering data processing is called "data normalization", which is directly related to the number of neutron events collected by detector pixels.

For example, in a 10-second measurement on an isotropic sample, detector pixel 1 may collect a different number of neutron events than detector pixel 2 due to variations in detection efficiency. As another example, comparing the intensity of a peak in scan A with that of a peak in scan B is nontrivial if scan A has a longer measurement time than scan B. These realistic factors necessitate data normalization.

To perform data normalization, vanadium is used as the standard sample. This is why data processing for each experiment requires a corresponding vanadium IPTS.

Although rare, if scans contain overlapping regions of detection, please refer to the "Data Redundancy" section at the end of this manual.

Data processing using MATLAB (ReTIA) is not included in this manual. Relevant documents can be found on the ORNL analysis cluster (analysis.sns.gov) under /HFIR/HB3A/shared/Tutorials/.

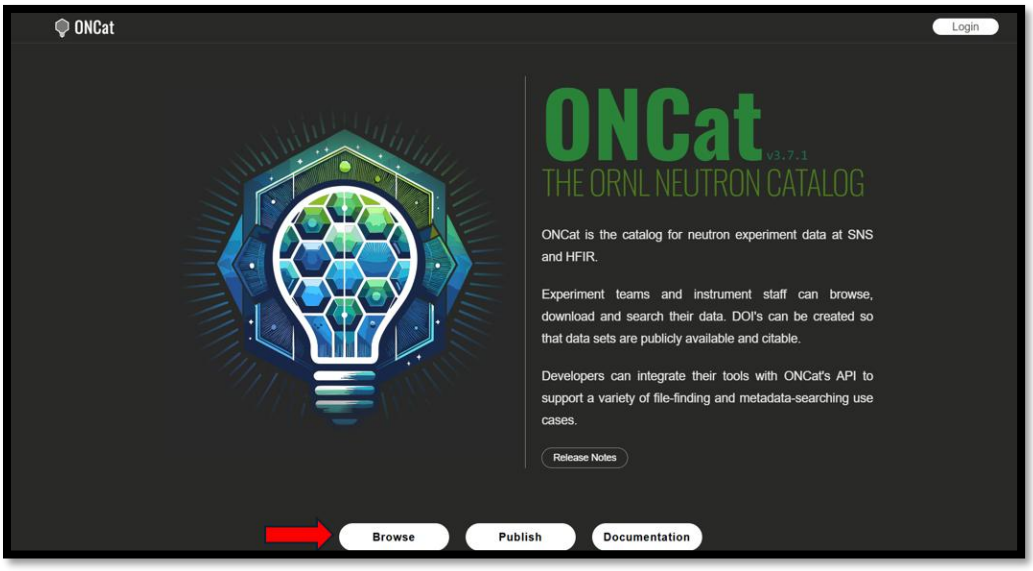
If you have any questions or feedback about this manual, please contact hux6@ornl.gov.

Back to Index

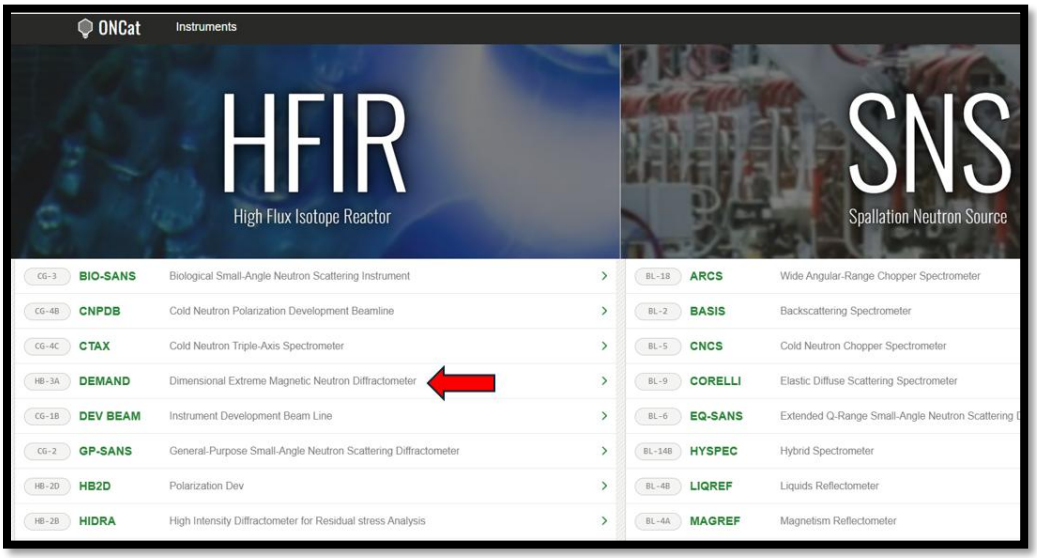
II. Locate IPTS and Run Numbers & Open Mantid Workbench

(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 the ONCat catalog using your UCAMS credentials.

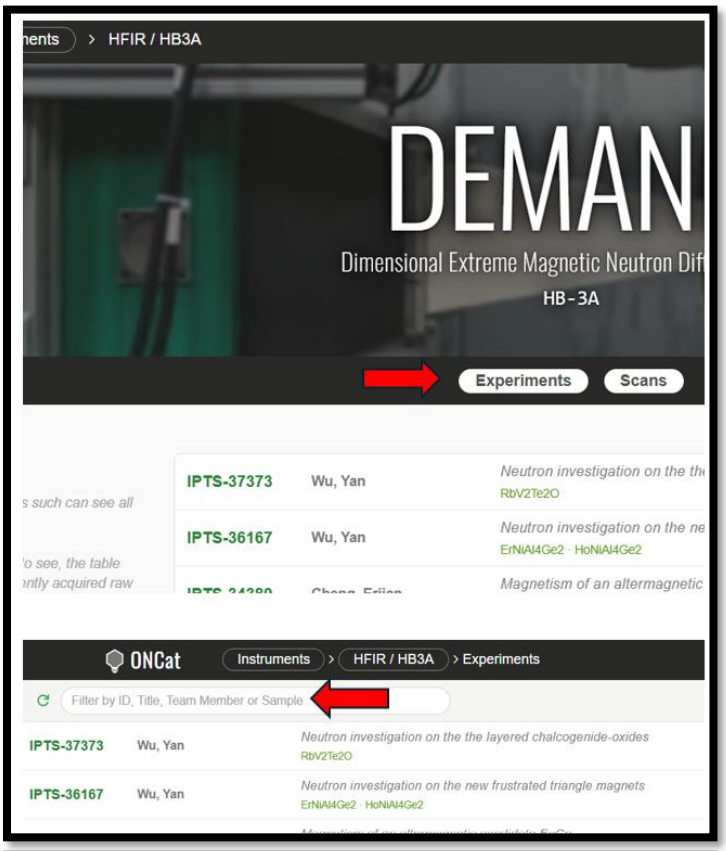


(2) In the HFIR/SNS instrument list, locate and select DEMAND.



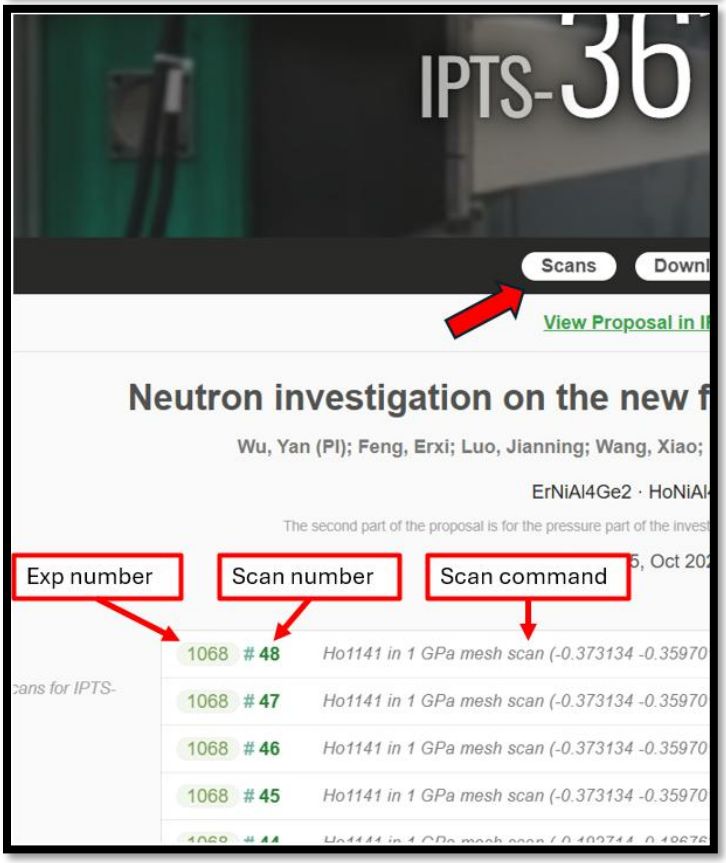
Back to Index

(3) Search for your experimental data under “Experiments” by entering the proposal number (IPTS). The scans will be displayed along with the experiment number, scan number, scan command, and scan description. Click the “Scan” button to view the full list of scans.



Note:

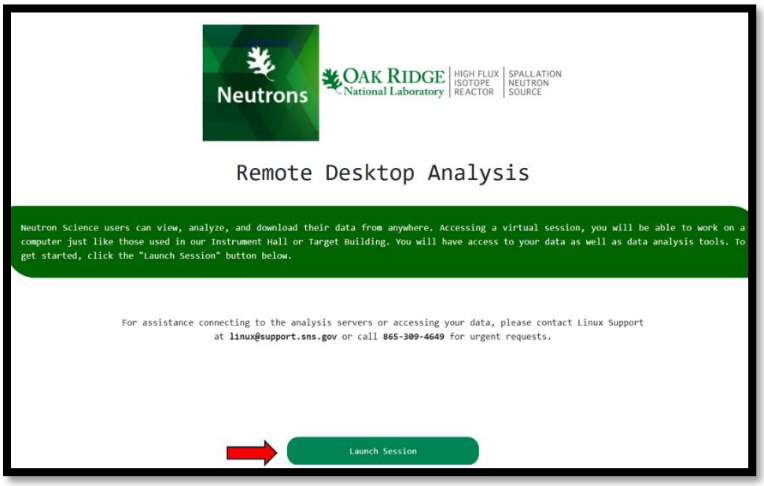
Vanadium calibration is usually performed before or at the beginning of each cycle. The vanadium measurement information will be provided by the instrument team.



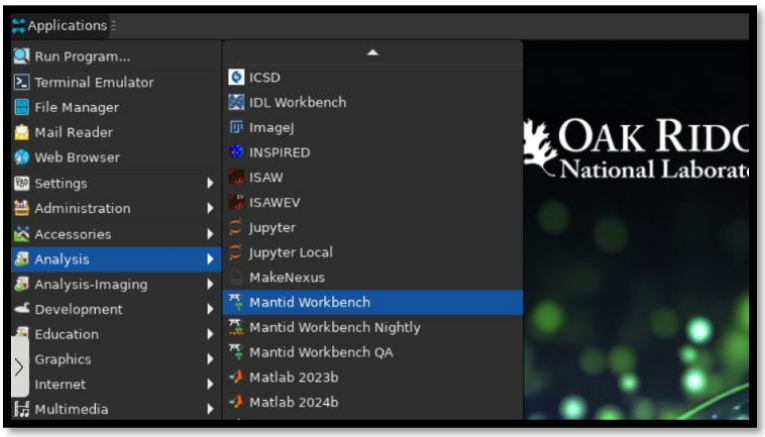
Back to Index

(b) Open Mantid Workbench on the analysis cluster

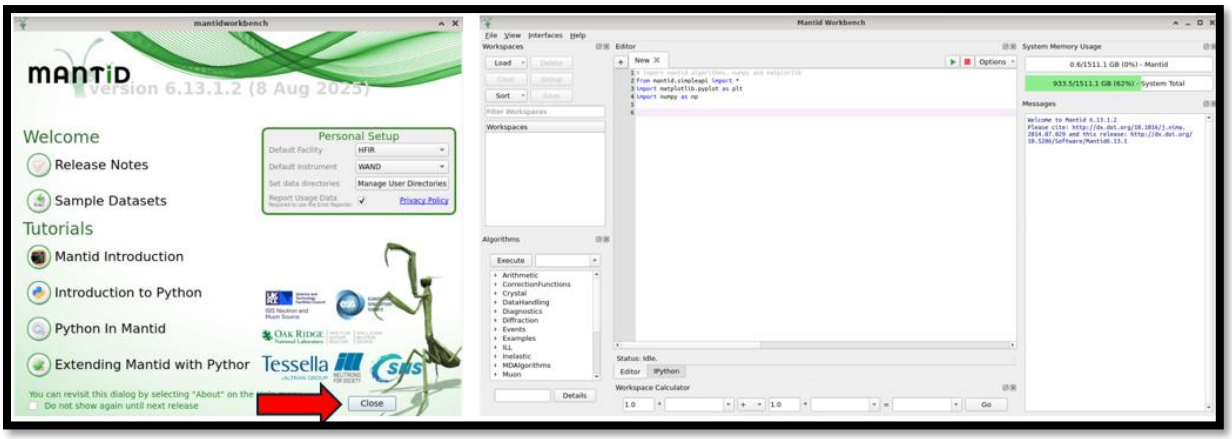
(1) Go to Remote Desktop Analysis at https://analysis.sns.gov/ and log in using your UCAMS username and password.



(2) Once logged in, navigate to the upper-left corner and select: Applications → Analysis → Mantid Workbench.



(3) When Mantid Workbench opens, you may see an option to personalize the setup and view tutorials. If personalization is not needed, close the Welcome window. You should now see the Mantid graphical user interface.

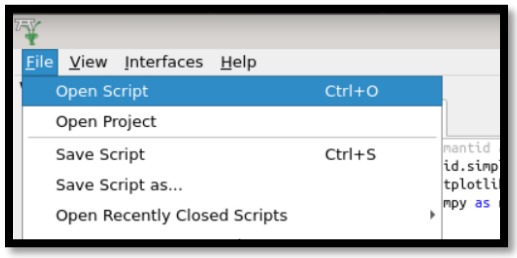


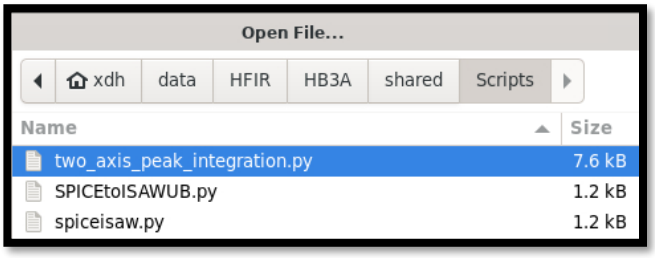
Back to Index

III. Script Execution in Mantid

(a) Open a script in Mantid

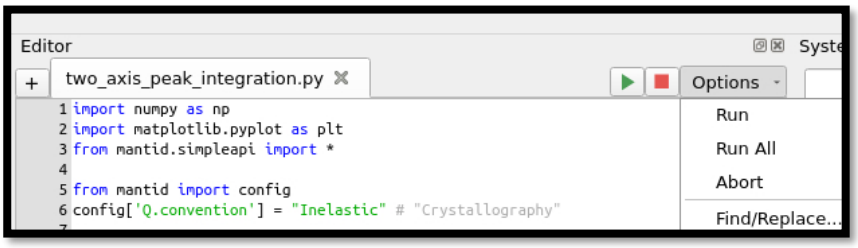
In the Mantid interface, click **File** in the upper-left corner and select "**Open Script**", or press **Ctrl + O**. Navigate to the target directory and open the desired script.



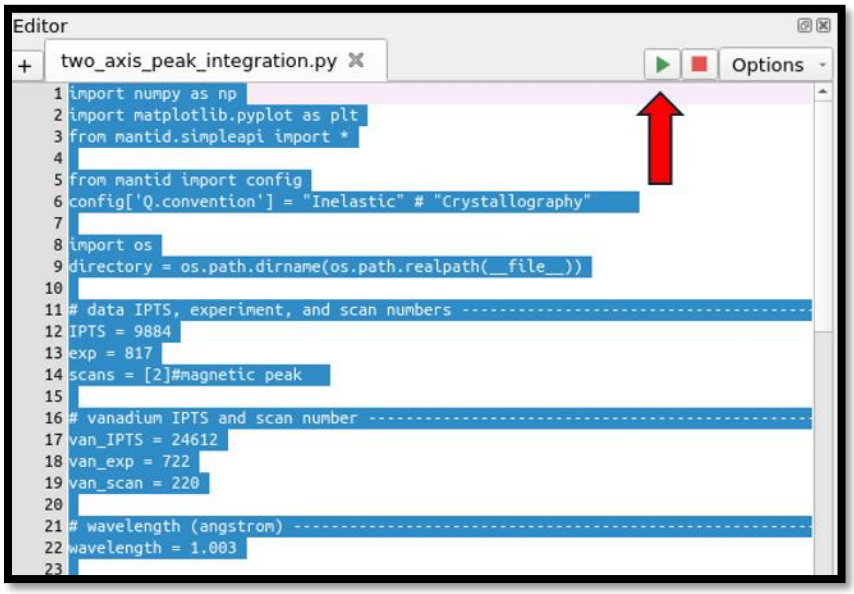


(b) Execute the script (entirely or partially)

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



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

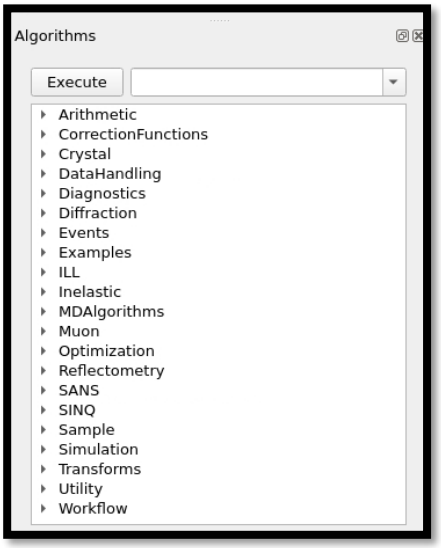


Back to Index

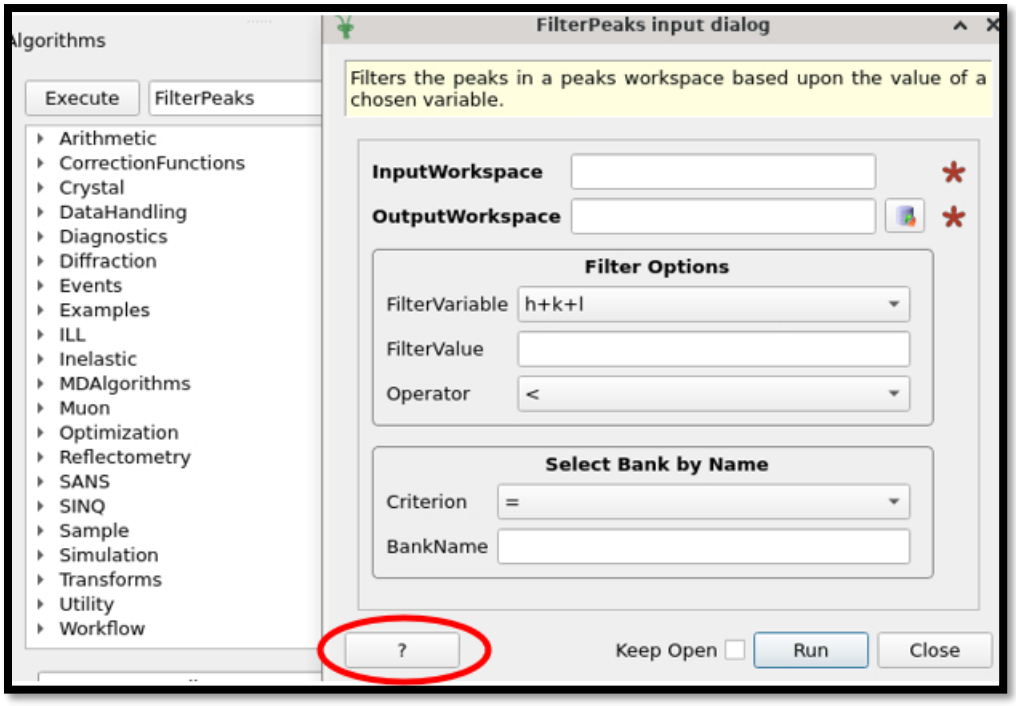
IV. Looking up Algorithm Details

This manual makes extensive use of built-in Mantid algorithms. The provided scripts convert algorithm interfaces into fast, executable code for efficient data processing. To learn more about a specific algorithm:

(1) Open the Mantid interface
(2) Locate the "Execute" panel on the left-hand side.
(3) Type the algorithm name into the search bar or locate it in the menu list below.
(4) Click "Execute" to open the algorithm interface
(5) Click the "?" button to open the documentation webpage for detailed information.



For detailed questions about algorithms used later in this manual, this is the recommended approach.



Back to Index

V. Data Visualization

By “visualization”, we refer to plotting data in reciprocal space (Q or HKL) to inspect peak distributions, symmetry features, and reciprocal-space coverage. While many visualization approaches are available in Mantid, this manual introduces a commonly used method based on the script:

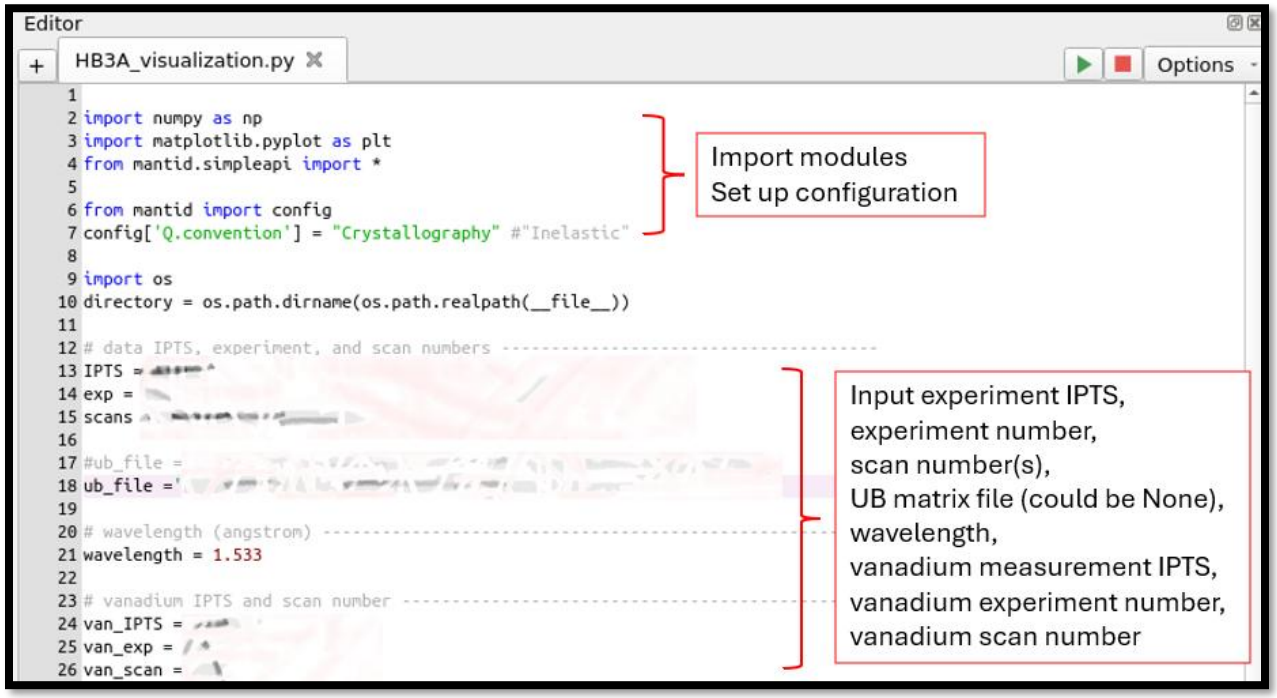
HB3A_visualization.py

(1) The script is located on the analysis cluster in: /HFIR/HB3A/shared/Scripts/

(2) Copy the script to your working directory.

(3) Open the script in Mantid via **File → Open Script** (or **Ctrl + O**).

(4) The script is structured as follows:
Lines 1 – 26: Input experiment information. The experiment information can be obtained from the ONCat website. Vanadium measurement details are provided by the instrument team.

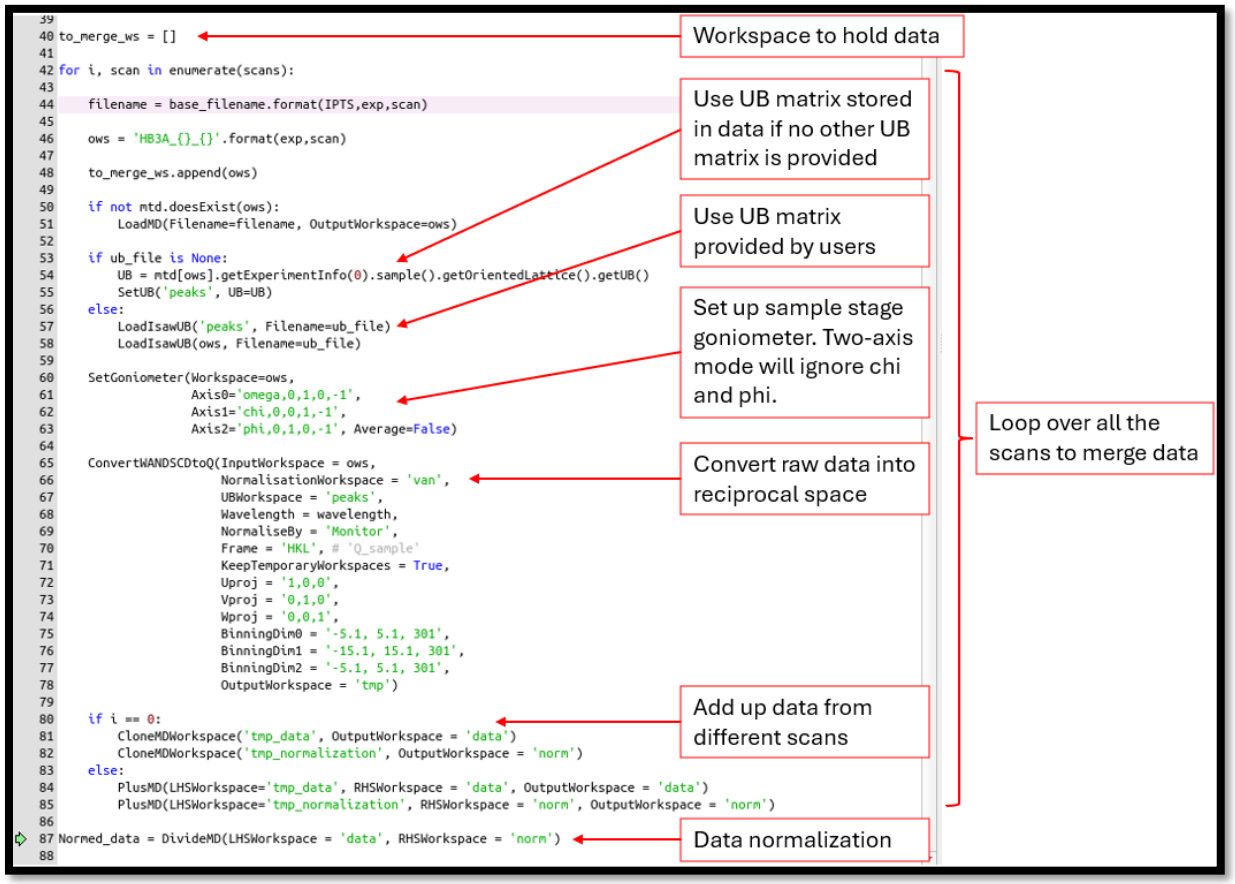


Back to Index

Lines 29 – 37: Definition of input file formats and UB matrix handling. Auto-reduced .nxs data files are stored under: /HFIR/HB3A/IPTS-{your exp IPTS number}/shared/autoreduce/. An empty workspace **"peaks"** is created to store UB matrix information if available. The vanadium file is loaded as a multi-dimensional (MD) workspace **"van"** for later use.

```
28
29 base_filename = '/HFIR/HB3A/IPTS-{}/shared/autoreduce/HB3A_exp{:04}_scan{:04}.nxs'
30 van_filename = '/HFIR/HB3A/IPTS-{}/shared/autoreduce/HB3A_exp{:04}_scan{:04}.nxs'.format(van_IPTS,van_exp,van_scan)
31
32 CreatePeaksWorkspace(NumberOfPeaks=0,
33                      OutputType='LeanElasticPeak',
34                      OutputWorkspace='peaks')
35
36 if not mtd.doesExist('van'):
37     LoadMD(Filename=van_filename, OutputWorkspace='van')
38
```

Lines 39 – 87: Data merging and normalization. Because the HB-3A detector operates over a wide range of take-off angle (2θ), data collected at different angles needs to be merged. This is handled using a for-loop.




```
39
40 to_merge_ws = []
41
42 for i, scan in enumerate(scans):
43
44     filename = base_filename.format(IPTS,exp,scan)
45
46     ows = 'HB3A_{}_{}'.format(exp,scan)
47
48     to_merge_ws.append(ows)
49
50     if not mtd.doesExist(ows):
51         LoadMD(Filename=filename, OutputWorkspace=ows)
52
53     if ub_file is None:
54         UB = mtd[ows].getExperimentInfo(0).sample().getOrientedLattice().getUB()
55         SetUB('peaks', UB=UB)
56     else:
57         LoadIsawUB('peaks', Filename=ub_file)
58         LoadIsawUB(ows, Filename=ub_file)
59
60     SetGoniometer(Workspace=ows,
61                   Axis0='omega,0,1,0,-1',
62                   Axis1='chi,0,0,1,-1',
63                   Axis2='phi,0,1,0,-1', Average=False)
64
65     ConvertWANDSCDtoQ(InputWorkspace = ows,
66                       NormalisationWorkspace = 'van',
67                       UBWorkspace = 'peaks',
68                       Wavelength = wavelength,
69                       NormaliseBy = 'Monitor',
70                       Frame = 'HKL', # 'Q_sample'
71                       KeepTemporaryWorkspaces = True,
72                       Uproj = '1,0,0',
73                       Vproj = '0,1,0',
74                       Wproj = '0,0,1',
75                       BinningDim0 = '-5.1, 5.1, 301',
76                       BinningDim1 = '-15.1, 15.1, 301',
77                       BinningDim2 = '-5.1, 5.1, 301',
78                       OutputWorkspace = 'tmp')
79
80     if i == 0:
81         CloneMDWorkspace('tmp_data', OutputWorkspace = 'data')
82         CloneMDWorkspace('tmp_normalization', OutputWorkspace = 'norm')
83     else:
84         PlusMD(LHSWorkspace='tmp_data', RHSWorkspace = 'data', OutputWorkspace = 'data')
85         PlusMD(LHSWorkspace='tmp_normalization', RHSWorkspace = 'norm', OutputWorkspace = 'norm')
86
87 Normed_data = DivideMD(LHSWorkspace = 'data', RHSWorkspace = 'norm')
88
```

Back to Index

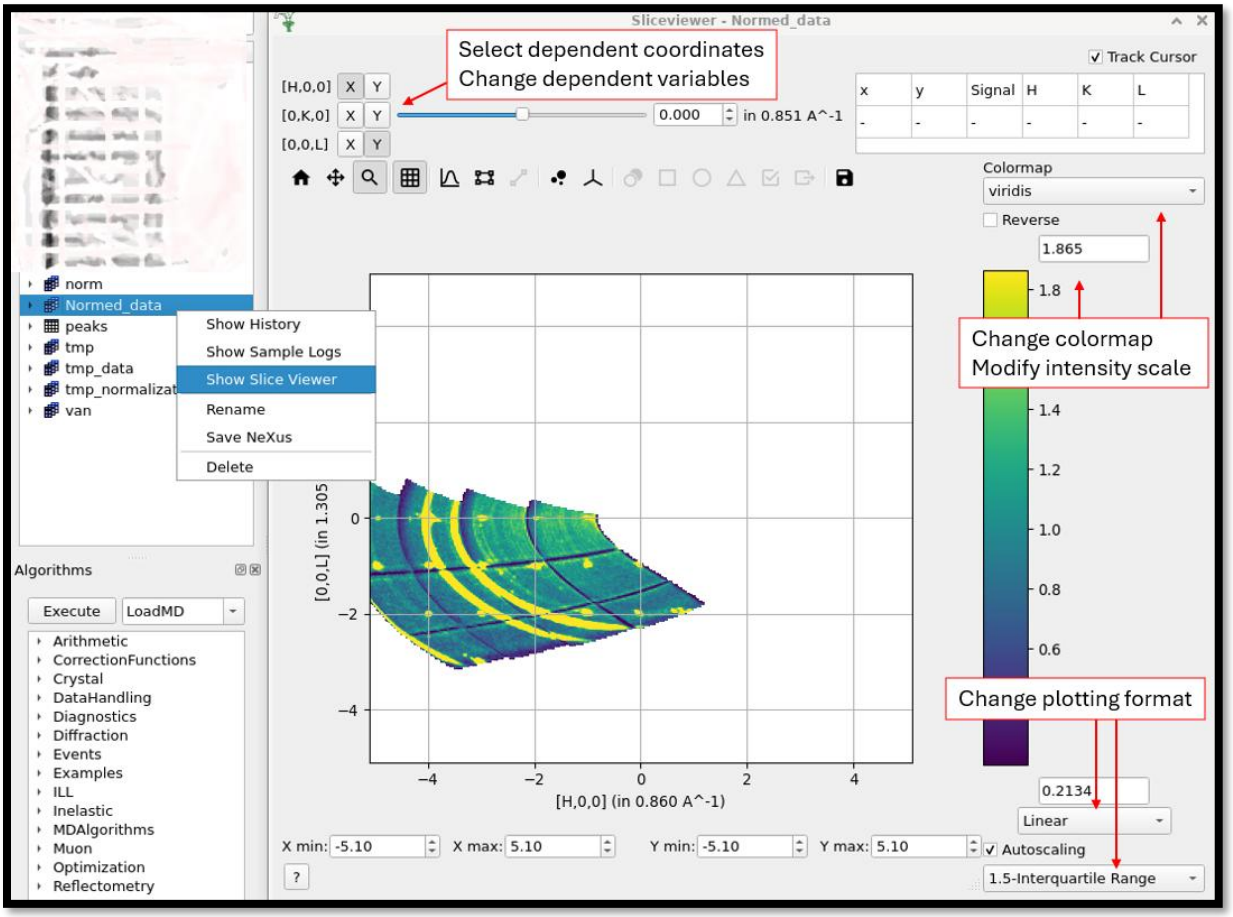
Lines 65 – 87 need emphasis.

```
65    ConvertWANDSCDtoQ(InputWorkspace = ows,
66                      NormalisationWorkspace = 'van',
67                      UBWorkspace = 'peaks',
68                      Wavelength = wavelength,
69                      NormaliseBy = 'Monitor', # Time
70                      Frame = 'HKL', # 'Q_sample'
71                      KeepTemporaryWorkspaces = True,
72                      Uproj = '1,0,0',
73                      Vproj = '0,1,0',
74                      Wproj = '0,0,1',
75                      BinningDim0 = '-5.1, 5.1, 301',
76                      BinningDim1 = '-15.1, 15.1, 301',
77                      BinningDim2 = '-5.1, 5.1, 301',
78                      OutputWorkspace = 'tmp')
79
80    if i == 0:
81        CloneMDWorkspace('tmp_data', OutputWorkspace = 'data')
82        CloneMDWorkspace('tmp_normalization', OutputWorkspace = 'norm')
83    else:
84        PlusMD(LHSWorkspace='tmp_data', RHSWorkspace = 'data', OutputWorkspace = 'data')
85        PlusMD(LHSWorkspace='tmp_normalization', RHSWorkspace = 'norm', OutputWorkspace = 'norm')
86
87 Normed_data = DivideMD(LHSWorkspace = 'data', RHSWorkspace = 'norm')
88
```

- **ows** is the input data workspace; **van** is the vanadium workspace for normalization. **peaks** is the workspace containing UB matrix information set by line 53 – 58.
- NormaliseBy = 'Monitor' means data normalization using monitor counts. It can be 'Monitor' or 'Time'.
- Frame = 'HKL' means the plotting frame is reciprocal space in HKL fractional coordinate. It can be 'Q_sample' or 'HKL'.
- In the for-loop, when each data workspace goes through the algorithm **ConvertWANDSCDtoQ**, the algorithm will generate two sub-workspaces for it: one is the converted (based on the chosen frame) data workspace, and another one is the converted normalization workspace (coming from vanadium). If KeepTemporaryWorkspaces = True, these two sub-workspaces will be outputted, which are necessary in later data normalization when we merge all the data.
- Uproj, Vproj, Wproj are projection basis vectors for data visualization/plotting. The default vectors are (1,0,0), (0,1,0), (0,0,1).
- BinningDim0, BinningDim1, BinningDim2 are data binning configurations along Uproj, Vproj, Wproj. The format is 'start, end, number of bins'. If you want to integrate the data in a certain direction within a certain interval, put number of bins = 1 for that direction.
- Here the output workspace from **ConvertWANDSCDtoQ** is **tmp**. When KeepTemporaryWorkspaces = True, there will be two sub-workspaces generated as **tmp_data** and **tmp_normalization**. Along the for-loop, we merge them individually using command lines 80 – 85: data workspaces merge into **data**, normalization workspaces merge into **norm**. Then in line 87, we divide them to obtain the normalized data workspace **Normed_data**.

Back to Index

To view the normalized data, right-click on the workspace **Normed_data** in the left-hand side of the Mantid interface and select **Show Slice Viewer**. A visualization window will pop up. Move the mouse to each button and icon to see the pop-up illustration of its functionality.



Back to Index

Line 91 – 128: Save output workspace into external ASCII files. To save the normalized data into an external file, a specific function **SaveMDtoAscii** is needed. Note: Do not save 3D data workspaces directly, as they can be very large and may cause Mantid to crash.

```
90
91 def SaveMDToAscii(workspace, filename, exclude_integrated = True, format = '%.6e'):
92     """
93     workspace: str -- Name of workspace as string
94     filename: str -- Path to output file
95     exclude_integrated: bool, optional -- Exclude integrated dimensions with bin sie of one
96     format: str -- Column format
97     """
98     ws = mtd[workspace]
99     if ws.id() != 'MDHistoWorkspace':
100         raise ValueError("The workspace is not an MDHistoToWorkspace")
101     #get dimensions
102     if exclude_integrated:
103         dims = ws.getNonIntegratedDimensions()
104     else:
105         dims = [ws.getDimension(i) for i in range(ws.getNumDims())]
106     # get rid of the edge points by step/2
107     dimarrays = [np.arange(d.getMinimum() + (d.getX(1) - d.getX(0)) * 0.5, \
108                            d.getMaximum(), d.getX(1) - d.getX(0)) for d in dims]
109
110     if len(dimarrays) > 1:
111         newdimarrays = np.meshgrid(*dimarrays, indexing = 'ij')
112     else:
113         newdimarrays = dimarrays
114
115     data = ws.getSignalArray()*1
116     err = np.sqrt(ws.getErrorSquaredArray())
117     header = 'Intensity Error ' + ' '.join([d.getName() for d in dims])
118     header += '\n shape: ' + 'x'.join([str(d.getNBins()) for d in dims])
119
120     to_save = np.c_[data.flatten(), err.flatten()]
121
122     for d in newdimarrays:
123         to_save = np.c_[to_save, d.flatten()]
124
125     np.savetxt(filename, to_save, fmt = format, header = header)
126
127
128 SaveMDToAscii('Normed_data', '/HFIR/HB3A/shar                    :', exclude_integrated = False)
```

Back to Index

VI. Peak Analysis

(a) Data loading, conversion, and merging

Two peak analysis approaches are available for DEMAND data: (1) Mantid-based analysis (described here); (2) MATLAB-based, documented under: /HFIR/HB3A/shared/Tutorials/.

This Mantid-based approach converts detector images into reciprocal space (in the unit of $Å^{-1}$) and integrates peaks using a 3D ellipsoidal model. Users are welcomed to design their own scripts for peak integration. Here we introduce the commonly used method:

HB3A_peakintegration.py

(1) The script is located on the analysis cluster in /HFIR/HB3A/shared/Scripts/.

(2) Copy it to your working directory.

(3) Open the script in Mantid via **File** on the left upper corner of the Mantid interface and selecting "**Open Script**" or pressing **Ctrl + O**.

(4) The script is structured as follows:

Line 1 – 30: Input experiment information. Same as data visualization, input the experiment IPTS, scan numbers, vanadium IPT, vanadium scan number, UB matrix (if available), and wavelength.

Back to Index

Line 32 – 56: Data loading, conversion, and merging.

Lines 32 – 43 load and merge the raw detector data and vanadium data.

```
31
32 data_files = [base_filename.format(IPTS,exp,s) for s in scans]
33 data_ws = []
34
35 for data_file in data_files:
36     ws, _ = os.path.splitext(os.path.basename(data_file))
37     data_ws.append(ws)
38     if not mtd.doesExist(ws):
39         Load(Filename=data_file, OutputWorkspace=ws)
40
41 van_ws, _ = os.path.splitext(os.path.basename(van_filename))
42 if not mtd.doesExist(van_ws):
43     Load(Filename=van_filename, OutputWorkspace=van_ws)
44
45 if not mtd.doesExist('merged'):
46
47     merged = HB3AAdjustSampleNorm(InputWorkspaces= ','.join(data_ws),
48                                   VanadiumWorkspace = van_ws,
49                                   NormaliseBy = 'Time',
50                                   Wavelength = wavelength,
51                                   DetectorHeightOffset = 0,
52                                   DetectorDistanceOffset = 0,
53                                   OutputType = 'Q-sample events',
54                                   MergeInputs = True,
55                                   MinValues = '-10, -10, -10',
56                                   MaxValues = '+10, +10, +10')
57
```

Lines 45 – 56 convert the raw data into a multi-dimensional event workspace (MDE) **merged**, normalized by the vanadium workspace **van_ws**.

Details for lines 45 – 56:

- The algorithm **HB3AAdjustSampleNorm** combines all data files and normalizes them by the vanadium. NormaliseBy = 'Time' (or 'Monitor') rescales the intensity by the counting time or monitor counts.
- DetectorHeightOffset and DetectorDistanceOffset are 0 except specifically assigned by the instrument team.
- OutputType can be "Q-sample events", "Q-sample histogram", and "Detector".
  - "Q-sample events" means that data will be converted to reciprocal space in the unit of Å^{-1}
  - "Q-sample histogram" means that data will be converted to reciprocal space in HKL lattice units (r. l. u.).
  - "Detector" means that data will stay the same as they are collected.
- MinValues and MaxValues are the dimension ranges. For example, it means that **merged** will be dataset spanning +/-10 Å^{-1} in Qsample-x, Qsample-y, Qsample-z directions in the screenshot example.

Back to Index

(b) Predict, index, integrate, and filter peaks (primary peaks)

```
58
59 if ub_file is not None:
60     LoadIsawUB('merged', Filename = ub_file)
61
62 ################## Peak prediction & integration ####################
63 epeak_radius = [0.04,0.10,0.03]
64 # ellipsoidal peak integration background envelope semi-axes ----------
65 einner_radius = [0.05,0.11,0.04]
66 eouter_radius = [0.06,0.12,0.05]
67
68 ######## Integer-HKL peaks
69 HB3APredictPeaks(InputWorkspace = 'merged',
70                  ReflectionCondition = 'Primitive',
71                  Wavelength = wavelength,
72                  MinDSpacing = 0.5,
73                  MaxDSpacing = 10,
74                  SatellitePeaks = True,
75                  IncludeIntegerHKL = True,
76                  ModVector1 = '0,0,0',
77                  MaxOrder = 0,
78                  CrossTerms = False,
79                  OutputWorkspace = 'bragg')
80
81 bragg = IntegratePeaksMD(InputWorkspace = 'merged',
82                          PeakRadius = epeak_radius,
83                          BackgroundInnerRadius = einner_radius,
84                          BackgroundOuterRadius = eouter_radius,
85                          PeaksWorkspace = 'bragg',
86                          Ellipsoid = True,
87                          IntegrateIfOnEdge = True)
88
89 bragg = FilterPeaks(InputWorkspace = 'bragg',
90                     FilterVariable = 'Signal/Noise',
91                     FilterValue = 10,
92                     Operator = '>')
93
94 for iter in range(5):
95     CentroidPeaksMD(InputWorkspace = 'merged',
96                     PeakRadius = 0.1,
97                     PeaksWorkspace = 'bragg',
98                     OutputWorkspace = 'bragg')
99
100 bragg = IntegratePeaksMD(InputWorkspace = 'merged',
101                          PeakRadius = epeak_radius,
102                          BackgroundInnerRadius = einner_radius,
103                          BackgroundOuterRadius = eouter_radius,
104                          PeaksWorkspace = 'bragg',
105                          Ellipsoid = True,
106                          IntegrateIfOnEdge = True)
107
108 ### Lorentzian correction for peak intensity
109 peak = mtd['bragg']
110 for no in range(peak.getNumberPeaks()):
111     p = peak.getPeak(no)
112     lorentz = np.abs( np.sin(p.getScattering()) * np.cos(p.getAzimuthal()) )
113     peak.getPeak(no).setIntensity(p.getIntensity() * lorentz)
114     peak.getPeak(no).setSigmaIntensity(p.getSigmaIntensity() * lorentz)
115
```

Line 59 – 114: Primary peak prediction & integration (integer-HKL).
Line 59 – 60 sets the UB matrix of **merged** as the customized one if available.

A 3D ellipsoidal integration model is used. Line 63 – 66 are ellipsoidal radii used for integration.

Line 69 – 79: **HB3APredictPeaks** predicts Bragg peaks in the d-spacing (Å) range set by MinDSpacing and MaxDspacing based on the ReflectionCondition. "Primitive" is usually chosen but others can be chosen if confident.

Line 74 – 78 are specifically set for integer-HKL peaks. Do not change.

Line 81 – 87: **IntegratePeaksMD** uses a 3D ellipsoidal model to do peak integration with Ellipsoid = True. Sphere is the default shape if this is False and only 1 radius is needed. The 3 radii of the ellipsoid are determined by epeak_radius for peak ellipsoid, einner_radius and eouter_radius for background ellipsoid. Details about its calculation procedures can be found in the algorithm introduction webpage.

Line 89 – 92: **FilterPeaks** filters peaks based on FilterVariable, here it is 'Signal/Noise' ratio. FilterValue = 10 and Operator = '>' mean that this filtering algorithm will keep those peaks that have intensity/error > 10 and remove other peak predictions. Peak radii, filter variable, filter operator, and filter values are all free to change.

Back to Index

Line 94 – 98: **CentroidPeaksMD** refines the peak positions by calculating "center of intensity" for each peak and moving the predicted position to the "center". You can iterate more if needed.

Line 100 – 106: Integrate peaks again but based on new (optimized) peak positions.
Line 109 – 114: Rescale peak intensities by Lorentzian factors. Lorentzian factor is a numerical coefficient that rescales the ratio of unit volume in (Qx, Qy, Qz) cartesian coordinates and polar coordinates.

(c) Predict, index, integrate, and filter peaks (satellite peaks)

Line 120 – 172: Satellite peaks prediction & integration.

```
118
119 ######## Satellite peaks
120 mod_vector1 = [0.333,0.333,0]
121 mod_vector2 = [0,0,0]
122 mod_vector3 = [0,0,0]
123 max_order = 1
124 cross_terms = False
125
126
127 HB3APredictPeaks(InputWorkspace='merged',
128                  ReflectionCondition='Primitive',
129                  Wavelength=wavelength,
130                  SatellitePeaks=True,
131                  IncludeIntegerHKL=False,
132                  ModVector1='{},{},{}'.format(*mod_vector1),
133                  ModVector2='{},{},{}'.format(*mod_vector2),
134                  ModVector3='{},{},{}'.format(*mod_vector3),
135                  MaxOrder=max_order,
136                  CrossTerms=cross_terms,
137                  OutputWorkspace='satellites')
138
139 satellites = IntegratePeaksMD(InputWorkspace='merged',
140                               PeakRadius=epeak_radius,
141                               BackgroundInnerRadius=einner_radius,
142                               BackgroundOuterRadius=eouter_radius,
143                               PeaksWorkspace='satellites',
144                               Ellipsoid=True,
145                               IntegrateIfOnEdge=True)
146
147 satellites = FilterPeaks(InputWorkspace='satellites',
148             FilterVariable='Signal/Noise',
149             FilterValue=1,
150             Operator='>')
151
152 for iter in range(5):
153     CentroidPeaksMD(InputWorkspace = 'merged',
154                     PeakRadius = 0.1,
155                     PeaksWorkspace = 'satellites',
156                     OutputWorkspace = 'satellites')
157
158 satellites = IntegratePeaksMD(InputWorkspace='merged',
159                               PeakRadius=epeak_radius,
160                               BackgroundInnerRadius=einner_radius,
161                               BackgroundOuterRadius=eouter_radius,
162                               PeaksWorkspace='satellites',
163                               Ellipsoid=True,
164                               IntegrateIfOnEdge=True)
165
166 ### Lorentzian correction for peak intensity
167 peak = mtd['satellites']
168 for no in range(peak.getNumberPeaks()):
169     p = peak.getPeak(no)
170     lorentz = np.abs( np.sin(p.getScattering()) * np.cos(p.getAzimuthal()) )
171     peak.getPeak(no).setIntensity(p.getIntensity() * lorentz)
172     peak.getPeak(no).setSigmaIntensity(p.getSigmaIntensity() * lorentz)
173
```

Satellite peaks are treated similarly, except that non-zero modulation vectors are defined.

The terminology "satellite" means that these peaks could have non-integer HKL indices, compared with the conventional integer-HKL Bragg peaks.

Up to 3 modulation vectors are supported. Line 120 – 124 set the modulation vectors. These vectors describe the offsets of the satellite peak indices from integer H, K, L.

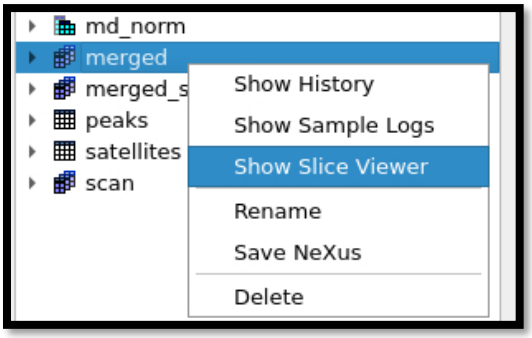
IncludeIntegerHKL = False will remove the integer-HKL peaks from the list.

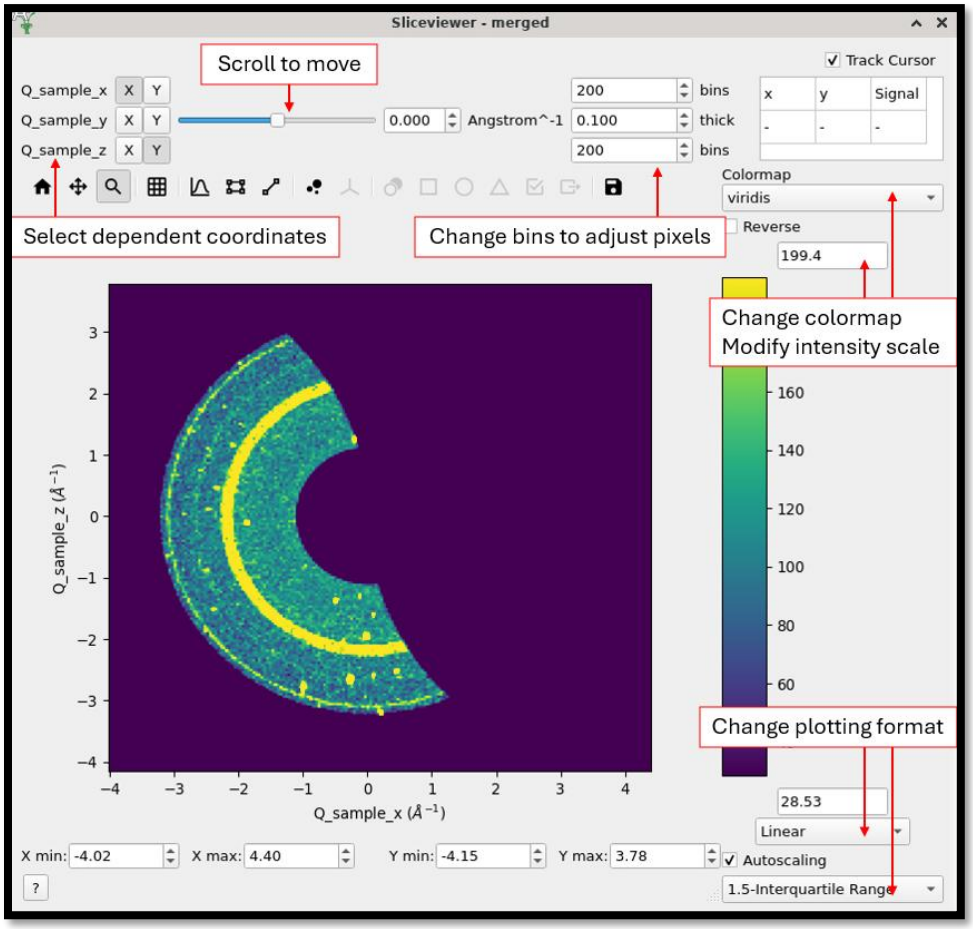
Back to Index

(d) Inspect peaks in Slice Viewer

After predicting and integrating peaks, it is necessary to check if the prediction and integration are good enough. We can do it through Slice Viewer in Mantid.

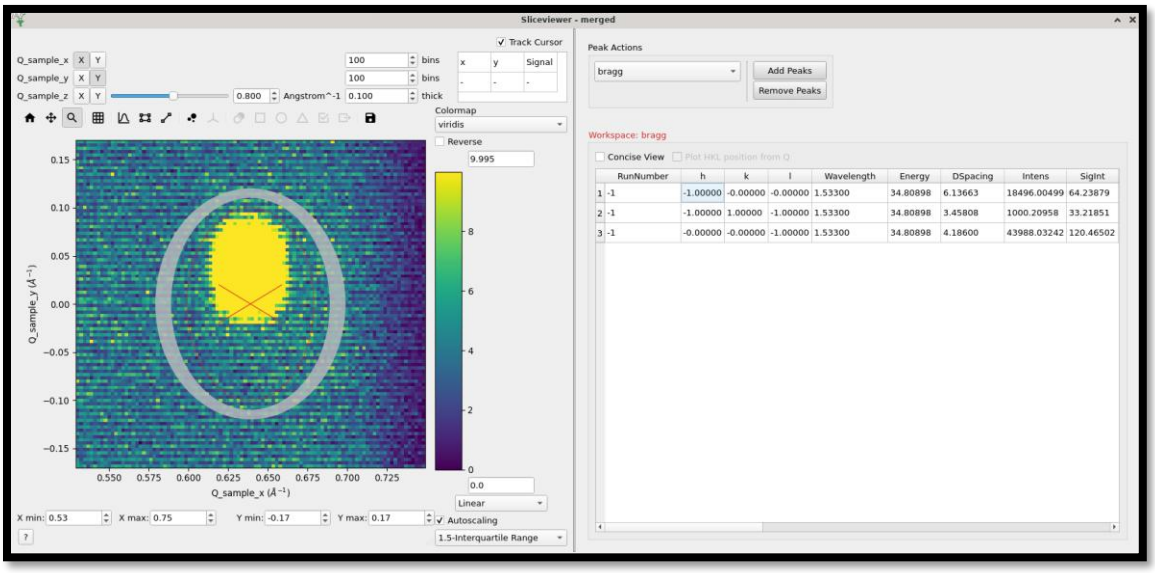
Right-click on the data workspace **merged** and select **Show Slice Viewer**. An interactive window will pop out and the contour map it shows is the merged data in the reciprocal space (Å^{-1}), as shown below.





Back to Index

Click the 3-dot icon  , and check the peak workspace you would like to examine. The peak workspace will be overlayed with the data contour. Click any peak in the right-hand side table, the contour will focus on the adjacent region around the peak, as shown below peak (-1,0,0) is chosen.



In the left-hand side contour, red cross is the predicted peak position, red dash line is the peak ellipsoid, grey ring is the background shell.

We notice that the predicted peak position of this chosen peak does not sit in the correct position. Running **CentroidPeaksMD** onto the peak workspace usually can solve this problem.

If still not satisfied, you can choose to enlarge the peak ellipsoid – make sure the red dash line covers the entire peak and not include other impurity signal except background. In this case, even though the peak position deviates from the correct one, the peak integration is still OK.

Alternatively, you can choose to refine the UB matrix (see section (e) later) for better peak prediction.

The ideal peak integration is that: the red dash line covers the entire peak; the grey background ring covers only the nearby background signal.

Back to Index

(e) UB matrix optimization

Sometimes, the UB matrix needs further optimization. One approach is to use strong Bragg peaks.

```
174
175 ############# UB optimization
176 epeak_radius = [0.04,0.10,0.03]
177 # ellipsoidal peak integration background envelope semi-axes --------
178 einner_radius = [0.05,0.11,0.04]
179 eouter_radius = [0.06,0.12,0.05]
180
181 ######## Integer-HKL peaks
182 HB3APredictPeaks(InputWorkspace = 'merged',
183                  ReflectionCondition = 'Primitive',
184                  Wavelength = wavelength,
185                  MinDSpacing = 0.5,
186                  MaxDSpacing = 10,
187                  SatellitePeaks = True,
188                  IncludeIntegerHKL = True,
189                  ModVector1 = '0,0,0',
190                  MaxOrder = 0,
191                  CrossTerms = False,
192                  OutputWorkspace = 'peakforUB')
193
194 peakforUB = IntegratePeaksMD(InputWorkspace = 'merged',
195                           PeakRadius = epeak_radius,
196                           BackgroundInnerRadius = einner_radius,
197                           BackgroundOuterRadius = eouter_radius,
198                           PeaksWorkspace = 'peakforUB',
199                           Ellipsoid = True,
200                           IntegrateIfOnEdge = True)
201
202 peakforUB = FilterPeaks(InputWorkspace = 'peakforUB',
203                         FilterVariable = 'Signal/Noise',
204                         FilterValue = 50,
205                         Operator = '>')
206
207 for iter in range(5):
208     CentroidPeaksMD(InputWorkspace = 'merged',
209                     PeakRadius = 0.1,
210                     PeaksWorkspace = 'peakforUB',
211                     OutputWorkspace = 'peakforUB')
212
```

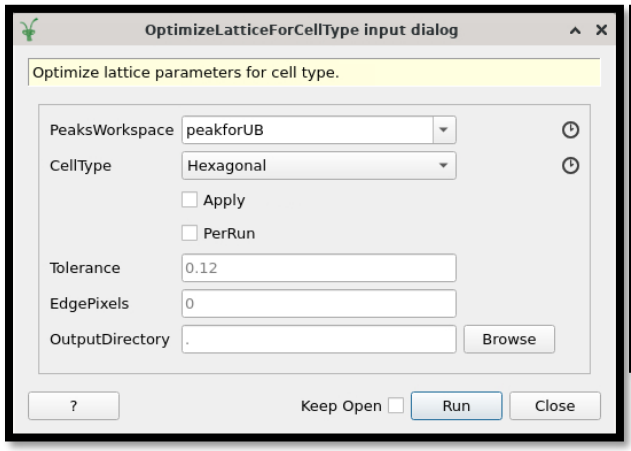
We use the same algorithms for peak integration, except that the filter value of Signal/Noise = 50 --- filter strong Bragg peaks.
(The value of 50 can be changed)

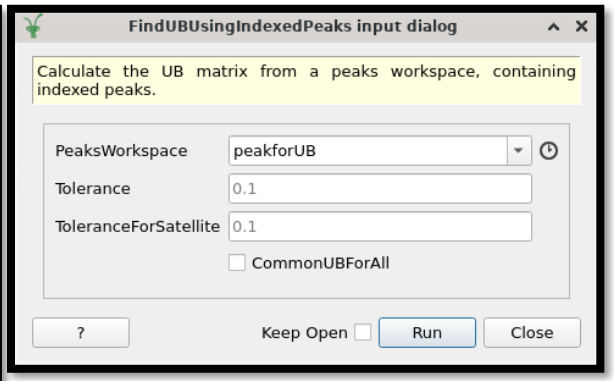
Inspect peaks **peakforUB** in Slice Viewer before moving on.

Two options are available for UB matrix optimization:

- **OptimizeLatticeForCellType**
- **FindUBUsingIndexedPeaks**

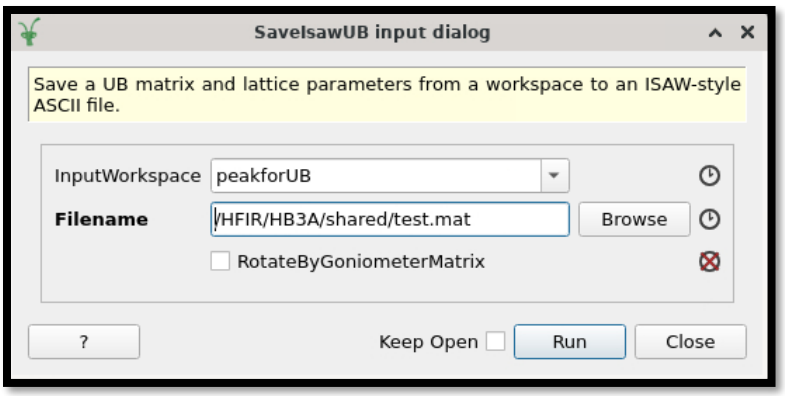
OptimizeLatticeForCellType refines the UB matrix based on the preset unit cell type within tolerance. **FindUBUsingIndexedPeaks** needs at least 3 linearly independent peaks to refine the UB matrix.





Back to Index

After running either one of the two algorithms, the new UB matrix is stored in the workspace **peakforUB**. We can use **SaveIsawUB** to export the optimized UB matrix into a .mat file. Then the optimized UB matrix can be used back at the beginning of the data analysis (Line 59 – 60).



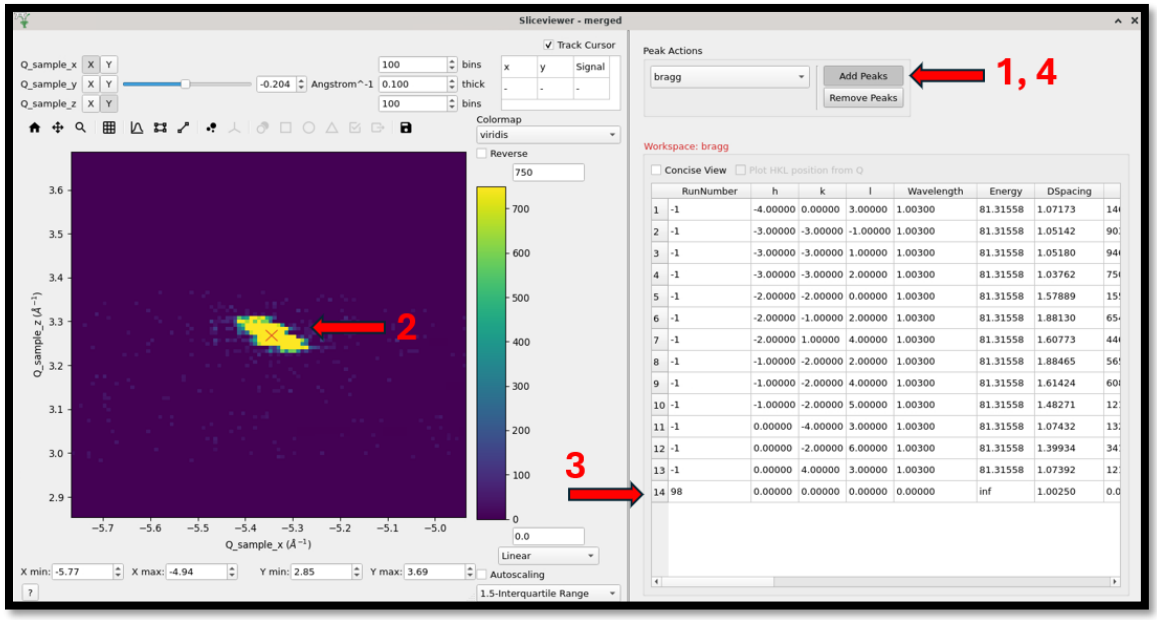
If the output UB file contains '-nan', replace it with 0.0000. The appearance of NaN is usually because the number of peaks is not enough to determine the error bars of the optimized lattice constants.

Back to Index

(f) Manually remove, add, and index peaks

In some rare situations, some peaks are not predicted by algorithms. In more common situations, some predicted positions deviate from the real peak positions. These peaks sometimes could be important to our data analysis and hence we need to adjust their characterization manually.

As informed in section (d), we open the data workspace in Slice Viewer and overlay the peak workspace. Find the target peak, as shown below.



(1) Click on **Add Peaks** button.
(2) Move the mouse to the target peak position in the contour plot. Click at the correct peak position. A red cross will appear.
(3) After the click, a new line will appear at the bottom of the peak list on the right-hand side. It is a handler for this newly added peak.
(4) Un-click **Add Peaks** button to avoid mis-clicking later.
(5) Repeat (1)-(4) if you need to add multiple peaks.

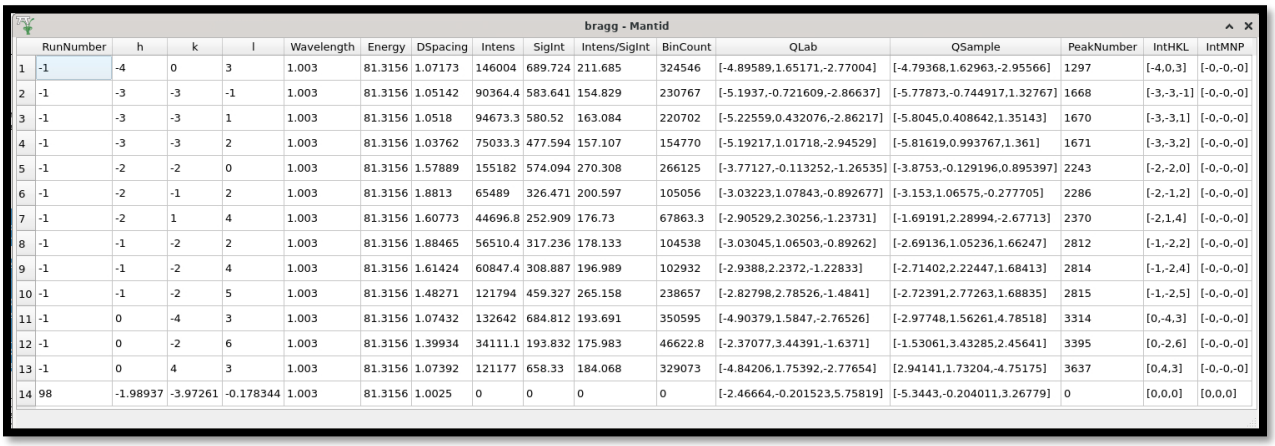
To characterize this new peak, run command lines 215 ~ 226, as shown on the right-hand side. (Continued next page)

```
213
214 ##### Manually add & re-index peaks on HB3A
215 peakws = mtd['bragg']
216 UB = peakws.sample().getOrientedLattice().getUB()
217
218 for no in range(peakws.getNumberPeaks()):
219     p = peakws.getPeak(no)
220     H0, K0, L0 = p.getHKL()
221     if (H0 == 0) & (K0 == 0) & (L0 == 0):
222         Qsample = p.getQSampleFrame()
223         UB_inv = np.linalg.inv(UB)
224         H, K, L = np.dot(UB_inv, Qsample)/2/np.pi
225         p.setHKL(H, K, L)
226         p.setWavelength(wavelength)
227
```

Back to Index

These command lines extract the UB matrix of the target peak workspace (here **bragg**) and locate the Q-sample position (red cross in the Slice Viewer contour plot) of the target peak ('QSample' column in the peak table). Then calculate the HKL index of the peak based on the analytical relationship $\boldsymbol{Q}_{sample} = 2\pi UB \begin{pmatrix} H \\ K \\ L \end{pmatrix}$ and set corresponding wavelength.

Afterwards the peak workspace will be updated as follows:



bragg - Mantid

	RunNumber	h	k	l	Wavelength	Energy	DSpacing	Intens	SigInt	Intens/SigInt	BinCount	QLab	QSample	PeakNumber	IntHKL	IntMNP
1	-1	-4	0	3	1.003	81.3156	1.07173	146004	689.724	211.685	324546	[-4.89589,1.65171,-2.77004]	[-4.79368,1.62963,-2.95566]	1297	[-4,0,3]	[0,-0,-0]
2	-1	-3	-3	-1	1.003	81.3156	1.05142	90364.4	583.641	154.829	230767	[-5.1937,-0.721609,-2.86637]	[-5.77873,-0.744917,1.32767]	1668	[-3,-3,-1]	[0,-0,-0]
3	-1	-3	-3	1	1.003	81.3156	1.0518	94673.3	580.52	163.084	220702	[-5.22559,0.432076,-2.86217]	[-5.8045,0.408642,1.35143]	1670	[-3,-3,1]	[0,-0,-0]
4	-1	-3	-3	2	1.003	81.3156	1.03762	75033.3	477.594	157.107	154770	[-5.19217,1.01718,-2.94529]	[-5.81619,0.993767,1.361]	1671	[-3,-3,2]	[0,-0,-0]
5	-1	-2	-2	0	1.003	81.3156	1.57889	155182	574.094	270.308	266125	[-3.77127,-0.113252,-1.26535]	[-3.8753,-0.129196,0.895397]	2243	[-2,-2,0]	[0,-0,-0]
6	-1	-2	-1	2	1.003	81.3156	1.8813	65489	326.471	200.597	105056	[-3.03223,1.07843,-0.892677]	[-3.153,1.06575,-0.277705]	2286	[-2,-1,2]	[0,-0,-0]
7	-1	-2	1	4	1.003	81.3156	1.60773	44696.8	252.909	176.73	67863.3	[-2.90529,2.30256,-1.23731]	[-1.69191,2.28994,-2.67713]	2370	[-2,1,4]	[0,-0,-0]
8	-1	-1	-2	2	1.003	81.3156	1.88465	56510.4	317.236	178.133	104538	[-3.03045,1.06503,-0.89262]	[-2.69136,1.05236,1.66247]	2812	[-1,-2,2]	[0,-0,-0]
9	-1	-1	-2	4	1.003	81.3156	1.61424	60847.4	308.887	196.989	102932	[-2.9388,2.2372,-1.22833]	[-2.71402,2.22447,1.68413]	2814	[-1,-2,4]	[0,-0,-0]
10	-1	-1	-2	5	1.003	81.3156	1.48271	121794	459.327	265.158	238657	[-2.82798,2.78526,-1.4841]	[-2.72391,2.77263,1.68835]	2815	[-1,-2,5]	[0,-0,-0]
11	-1	0	-4	3	1.003	81.3156	1.07432	132642	684.812	193.691	350595	[-4.90379,1.5847,-2.76526]	[-2.97748,1.56261,4.78518]	3314	[0,-4,3]	[0,-0,-0]
12	-1	0	-2	6	1.003	81.3156	1.39934	34111.1	193.832	175.983	46622.8	[-2.37077,3.44391,-1.6371]	[-1.53061,3.43285,2.45641]	3395	[0,-2,6]	[0,-0,-0]
13	-1	0	4	3	1.003	81.3156	1.07392	121177	658.33	184.068	329073	[-4.84206,1.75392,-2.77654]	[2.94141,1.73204,-4.75175]	3637	[0,4,3]	[0,-0,-0]
14	98	-1.98937	-3.97261	-0.178344	1.003	81.3156	1.0025	0	0	0	0	[-2.46664,-0.201523,5.75819]	[-5.3443,-0.204011,3.26779]	0	[0,0,0]	[0,0,0]

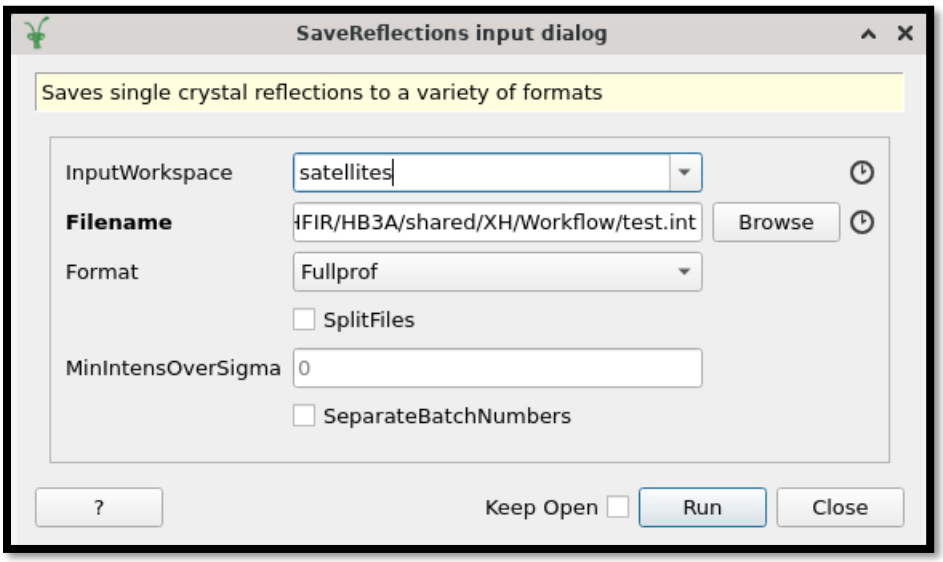
The final line is the newly added peak with its HKL indices, which are not rounded up to integers.

- This method helps us characterize the additionally added peaks.
- **IntegratePeaksMD**, **CentroidPeaksMD**, **FilterPeaks**, **SaveReflections** can still be applied to the updated peak workspace if needed.
- In this example, if you are sure this peak is an integer-HKL peak even though L = -0.178, you can **double-click** on the table block and change the number inside to integers.

Back to Index

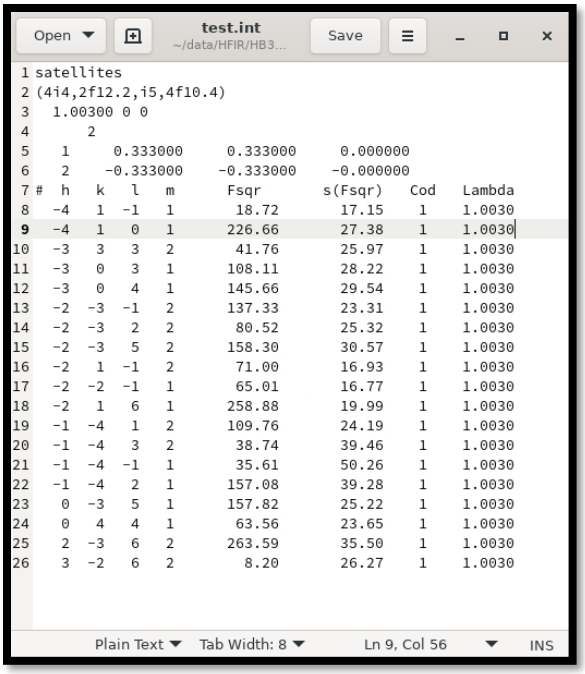
(g) Save peaks for refinement

The algorithm **SaveReflections** saves the peak workspace into an external file for refinement in software such as Fullprof, Jana, GSAS, or SHELX.



Particularly, when saving a satellite peak workspace, the output file generally looks like:

- 1st number on line 3 is wavelength
- Line 4 is the number of propagation vectors. Here is 2, listing on line 5 and 6 with code 1 and 2.
- Compared with integer-peak file, satellite peak file contains one additional column 'm' with code 1 or 2 here. This number means which propagation vector the peak is corresponding to. For example, line 8 peak (-4, 1, -1) with m = 1 corresponds to that this peak is (-4, 1, -1) + (0.333, 0.333, 0.000) = (-3.667, 1.333, -1). This is an equivalent format of representing satellite peaks.



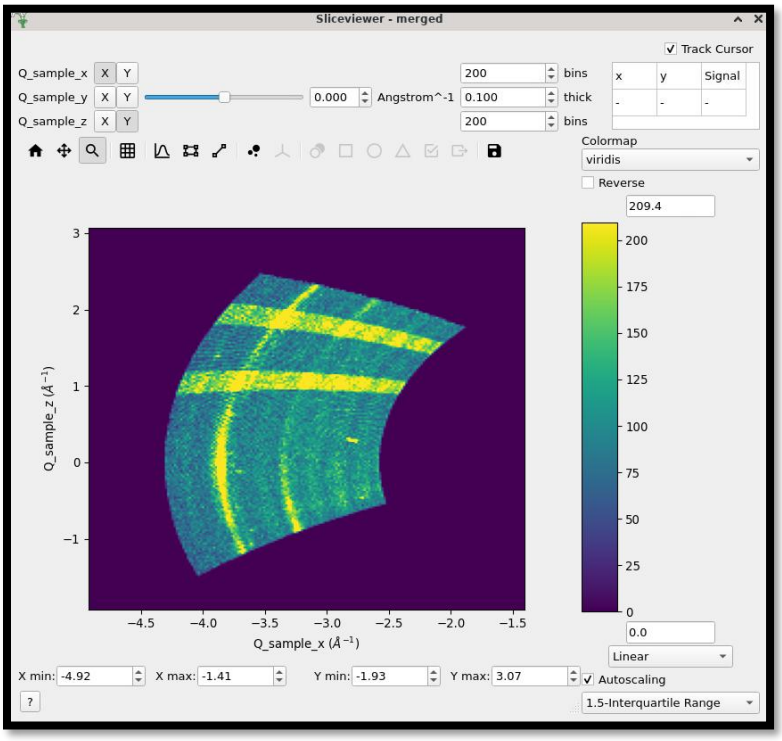
Be careful: Manually added satellite peaks (from section (f)) will have m = 0.

Back to Index

VII. Data Redundancy

Data redundancy refers to the situation that the same region of reciprocal space is measured multiple times, resulting in artificially enhanced intensities in overlapping regions.

For example, two HB-3A scans with the commands “scan omega -30 20 0.1” and “scan omega -10 40 0.1” contain overlapping regions in the range of omega = -10 ~ 20. As a result, the corresponding regions in reciprocal space appear noticeably “brighter” in the Slice Viewer. A specific example is shown as follows:



Two stripe-like regions exhibit significantly higher intensities than the surrounding areas. This occurs because the merged data workspace **merged** is compiled from 4 scans:

1. “Scan omega 0 40 0.15”
2. “Scan omega 40 80 0.15”
3. “Scan omega 32 36 0.15”
4. “Scan omega 45 49 0.15”

Scans 3 and 4 remeasured regions already covered by scans 1 and 2. Consequently, intensities in the overlapping regions are accumulated, making these regions explicitly brighter in the Slice Viewer.

In section V. Data visualization, this issue does not arise because the data are histogrammed and combined properly in the for-loop. In contrast, in section VI. Peak analysis, this effect can occur if scans are not carefully designed, as the data are compiled in event mode rather than histogrammed. For peak analysis, this redundancy is not a problem if no peaks fall within the overlapping regions. However, if peaks do lie within these regions, special care is needed. An effective approach is to analyze such peaks using data from a single scan only.

Back to Index