*Cite "Quantum Beam Science **4**, 41 (2020)" if you publish results obtained with the help of the software*

Phonon Explorer & Auxiliary Programs
Dmitry Reznik, Irada Ahmadova, and Tyler Sterling
Version 2.0

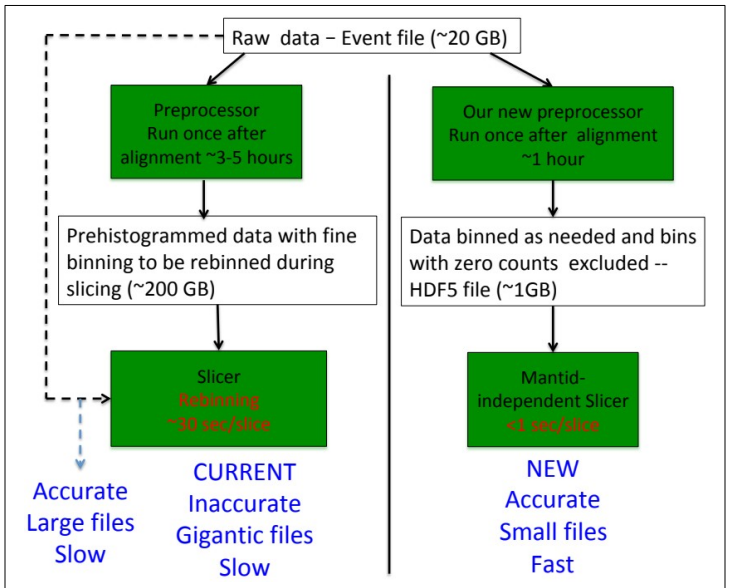
Release notes: This file contains documentation for the Phonon Explorer software (P. 1-8) and auxiliary programs (P. 9).

What's new in V2.0:

This is the new generation of Phonon Explorer that is based on the new paradigm

New paradigm for TOF data analysis:

Left side of the figure on the right illustrates the current workflow that is typical in the neutron scattering community. The entire dataset is initially prehistogrammed with fine binning, which results in files of about 100Gb. Next step is to slice this histogram, typically using larger bins, which results in rebinning of data that had already been binned.



Current vs. new workflow implemented here. Dashed line represents the workflow proposed in Savici et al.

Major flaws in this workflow are widely recognized:

1. A lot of disk storage is required, so that prehistogrammed data to be sliced often needs to be kept on external drives. It is also hard to send or download these files via the Internet.
2. Slicing these files is slow because they typically cannot be loaded into RAM and custom (not commercial) software written at the physics labs is used for slicing..
3. Rebinning data that have already been binned introduces extra noise and possibly false features (A, T. Savici, et al., Efficient data reduction for time-of-flight neutron scattering experiments on single crystals, J. Appl. Cryst. 55, 1514 (2022), https://doi.org/10.1107/S1600576722009645) Dashed arrow in the figure indicates an alternative workflow currently implemented only in Mantid proposed in Savici et al. Unfortunately this one-step generation of slices is too slow if one wants to generate hundreds of them

Phonon Explorer 2.0 implements a far superior alternative workflow (illustrated on the right side of the above figure that solves all three problems. Currently it has been

tested only for the SNS data and for the JPARC data. Subsequent versions will work also with data from other facilities.

Preprocessors

The new version includes two preprocessors – 1 for JPARC data and 1 for the SNS data that bin the data with the user-defined binning scheme. We found that it takes about 10-15mins to run these preprocessors on a typical dataset on the SNS analysis cluster.

1. JPARC data should be stored in a folder containing all NXSPE files for the experiment. These are prehistogrammed in energy but not in Q. This means that the energy bins should be the final energy bins desired by the user. If different energy binning scheme is needed, the NXSPE files need to be remade. Ask your JPARC local contact to do this for you.

2. SNS data should be stored in a "multidimentional events" (MDE) "nexus" (.nxs) file.

-- Preprocessors save the data in a universal format in an HDF5 file. Only [h, k, l, intensity, error squared, magnitude of q, azimuthal and polar angles away from the 0,0,1 direction] are saved. So there are 8 columns in this file.

-- Only the first 5 columns are used in the current 2.0 version. The last three columns are there to allow future planned upgrades.

-- The final HDF5 file is sparse: ONLY bins where detectors are active are saved. I.E. bins that are outside the physical measurement space are not saved. This results in file sizes of under 1Gb.

-- The exact binning sizes and bin centers that will be needed for an analysis have to be used, so there is no rehistogramming involved. If several different **binning** schemes are required for the project, several files can be created (1 file for each binning scheme) and handled easily. Identical binnings with different Q-point grids can be included in the same file.

Advantages of Running Phonon Explorer on the HBF5 files

--Phonon Explorer uses these HDF5 files as input, so there is **no dependency on Mantid** after pre-processing.

--Our testing has shown that it takes about 0.01 sec. for a 1D cut saved in this format, whereas the procedure of Ref (J. Appl. Cryst. 55, 1514 (2022)) (dashed line in Fig. 12) takes about 10 sec., and the traditional procedure (e.g. with Horace) takes about 30 sec.

This scheme results in:

1. 2 orders of magnitude reduction in time to do one slice, i.e. running Phonon Explorer once for, say 1000 slices **takes a few seconds instead of hours**.
2. Histogrammed data **files are now on the order of 1 Gb vs. 100Gb**.
3. The data are **histogrammed (binned) once**, so correct intensities and error bars are obtained.

This workflow only works for the event-mode data (SNS) or NXPSE data (JPARC) that could be processed with Mantid. Preprocessors for this are provided as described above.

The code from Phonon Explorer 1.2 is used to work with HORACE (sqw) files and none of the above applies.

Instructions on how to use the program

The program allows to systematically and efficiently explore datasets obtained on TOF chopper spectrometers such as ARCS.

The main principle: generate and display many constant Q slices at once, with the option to calculate and subtract background and perform multizone fitting.

Main features of the program: The program generates multiple constant-Q slices from either .sqw (HORACE) or .hdf5 (derived from Mantid) file at once based on user input. It does not crash if Horace or Mantid fails to get the slice for a particular Q. Instead it continues to process other Qs. In addition, only slices with reasonable number of points as defined by the user are kept. Cuts containing no data or only a few points are not saved. The program has 3 phases of data analysis, which can be done all at once or separately. 1. Generate constant-Q slices 2. Compute background, 3. Perform multizone fit.

DO NOT RENAME FILES GENERATED BY THE PROGRAMS!
1. SNS or JPARC DATA

Installation Instructions:

1. Install the following packages are needed:
   Python 3.8
   Libraries (Python 3.8 version):
   Math
   Numpy
   Matplotlib
   Scipy
   pyPDF2
   H5py (not needed for data in SQW file)

2. If data are stored in SQW file, install Matlab, HORACE, and matlab engine. Make sure that the Matlab paths are configured so that the Matlab engine can be opened from Python 3.8 and the Horace routines can be called from the matlab engine in python.

3. Copy the Phonon Explorer folder onto your computer. Do not move files into different subdirectories.

4. In an editor open file RSE_CONSTANTS.py. Enter the folder of the InputParameters.txt into INPUTS_PATH_MAIN

5. Open the file Phonon Explorer/Input_Files/InputParameters.txt. This file controls what the software does as well as how it interacts with the directory structure of your computer. Comments that start with "#" explain everything you need to know about what each parameter does and when it is needed depending on what you are trying to do. If a parameter is not needed, the program will not try to read it, so it can be either omitted or define as anything. Read through the file to familiarize yourself and keep it handy as you are reading these release notes.

Following steps are needed **ONLY** If you are working with the SNS or JPARC data:

6. **SNS** -- Make sure you have the raw data stored in a reduced MDE nxs file. These can can made be using e.g. the "reduce_data_to_MDE.py" provided on the SNS server. Ask your local-contact/instrument-scientist for help if you need it.

JPARC -- Make sure you have the raw data stored in a folder with NXSPE files. These files are already histogrammed in energy, but not in momentum. Make sure your energy bins are the same as you would like in the constant Q cuts that you plan to make. If they are not the same, we recommend asking your local contact to remake them with the right binning. If you are not sure what binning is better, prepare several folders – each with different binning.

7. Go to the SNS server (ssh your_user_name@analysis.sns.gov) or another computer where Mantid is installed and runs well, called "server" below. Copy the raw data file(s) described above there if they are not there already.

8. Copy the files PreprocessEvents.py and/or PreprocessNXSPE.py and the file_tools folder from the /Phonon Explorer/Python Code folder on your computer to the folder where raw data are stored. PreprocessEvents.py is for raw data stored in an MDE nxs file from the SNS. PreprocessNXSPE.py is for raw data stored in NXSPE files from JPARC.

9. Edit the PreprocessEvents.py and/or PreprocessNXSPE.py to enter correct relevant file names and orientation matrix. Also enter desired binning. These short files should be self-explanatory.

10. Then run PreprocessEvents.py or PreprocessNXSPE.py on the analysis server for each desired binning scheme too generate hdf5 files. These should be on the order of 1Gb each. In the case of MDE nxs data, loading the nxs file is the most time consuming part. We recommend to run from an interactive mantidpython shell so that the loaded data are retained in memory. To do this, issue

```
mantidpython
```

and then

```
%run PreprocessEvents.py
```

For JPARC data, everything must be reloaded everytime so there is no advantage to using the interactive shell. You can ofcourse still do it with

```
mantidpython
%run PreprocessNXSPE.py
```

or run once with

```
mantidpython PreprocessNXSPE.py
```

IMPORTANT -- Make sure you run from mantidpython.

9. Copy the HDF5 output files to your laptop and run Phonon Explorer as usual.

KEY POINTS:

Data at requested Q-points are in hdf5 files. If the desired Q-point does not lie on the Q-grid of the prebinned data stored in a hdf5 files, the nearest Q-point on the grid will be used. Also, binning in energy or Q cannot be changed and is automatically read out of the data file. User inputs of bin sized are ignored. You need to remake the hdf5 file if you would like a different grid/binning scheme. See instructions just above.

UV matrix Current version of the program assumes that the angles between the uv matrix vectors are at 90 degrees to each other.

Naming of Files. DO NOT rename files or directories generated by the software unless you really know how the software works. It relies on file names for information such as wavevector, etc.

Different Operating Systems. This version is supposed to run on Windows, Mac, and Linux. The input text files should be saved in UTF-8 encoding except on Windows where the encoding should be ANSI. The code was tested with **Python 3.8 but later versions should also work**..

HOW TO RUN THE PHONON EXPLORER:

The software contains four programs written in Python: **GenerateConstQCuts.py, SubtractPolyBackgr.py, SubtractBackground.py,** and **Multifit.py**. (If the data are stored in an SQW file, there is also a short Matlab function that is called by one of the programs, but this is seamless from the point of view of the user. See below.)

These programs read a file **InputParametes.txt**, which contains input parameters together with detailed instructions of the functionality the parameters control. This

file needs to be edited in the text editor to provide information specific to the task at hand. There is one file that is used by all three programs.

Python classes for reading Data: The program can read data from and SQW file generated by HORACE or an HDF5 file generated by MANTID via the procedure explained above. Reading data by Phonon Explorer is seamless in the sense that the program decides how to handle raw data file based on extension. Python classes that interface with Raw data are in the "Tools to access raw data" folder. Users can edit this code if there are problems.

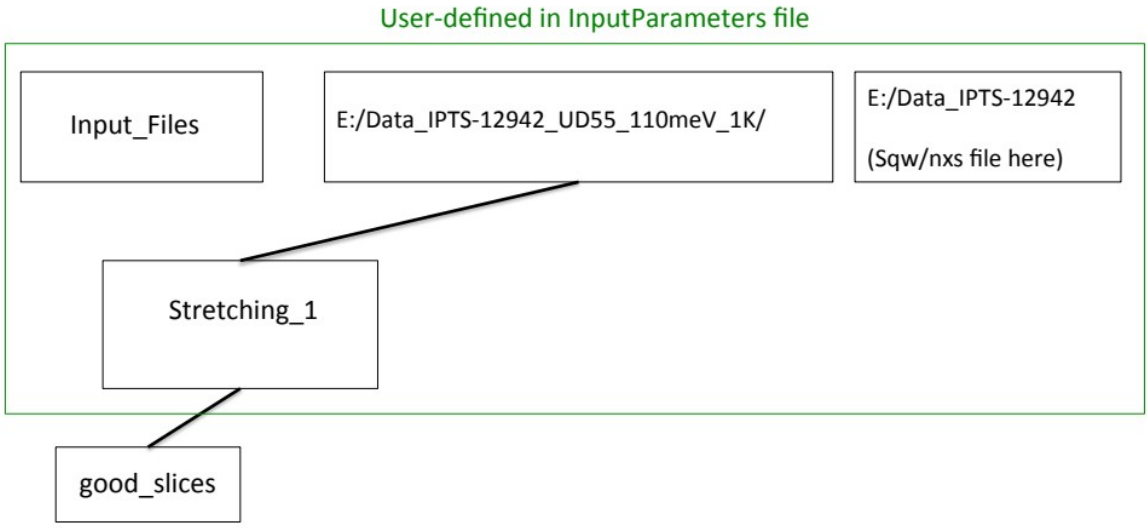
Generating a collection of Constant Q Cuts

1 **GenerateConstQCuts.py**: Generates all constant Q slices necessary for data analysis. One can specify BkgMode=0, if only raw cuts (without background subtraction) and their plots are needed. BkgMode=1 means that background subtraction will be performed. In the latter case, the program also generates cuts to be used for background determination, and no plots are generated.
There are two ways to run **GenerateConstQCuts.py**:
1. Generate all slices at the specific reduced wavevector q. I.e. it makes one slice per Brillouin zone for the q specified by the user and save the results as text files in a folder. (See Input File) To do this set **QMode=0.** The current version works for cubic, tetragonal and orthorhombic crystals to generate all wavevectors that correspond to a specific reduced wavevector. You will need to set the input parameter in the "InputParameters.txt" file to SmallQAlgorithm=Cubic, if the crystal symmetry is cubic. SmallQAlgorithm=Orthorhombic if it is Orthorhombic, SmallQAlgorithm=Tetragonal if it is Tetragonal.

2. Alternatively, the user can enter a list of wavevectors Q of interest (total wavevectors, not reduced), into a text file that should be placed in the folder assigned to the parameter InputFilesDIR in the InputParameters file. The program will read this text file and generate a folder containing the slices that correspond to each wavevector. To do this set **QMode=1** and assign the name of the text file to parameter "textfile_for_selectedQs" in the InputParameters.txt file. An example of such a file for Q=(5.6 0 1), (5.7 0 1), (5.8 0 1) is in the box on the right. The wavevectors should be in reciprocal lattice units.

```
5.6 0 1
5.7 0 1
5.8 0 1
```

Directory structure after running **GenerateConstQCuts.py** with **BkgMode=0**. **GenerateConstQCuts.py** was run once with input parameter ProcessedDataName=Stretching_1



Input Parameters:

sqw_path=E:/IPTS-12942/UD55_50meV_10K.sqw
projectRootDir=E:/Data_IPTS-12942_UD55_55meV_10K/
ProcessedDataName=Stretching_1

- **good_slices**: All slices that have a reasonable number of points (more than specified in the MinPointsInDataFile field in the input file) with reasonable (<ErrorToIntensityMaxRation field in the input file, default:30%) error bars are saved in the subdirectory "good slices" under the dataset directory. The PDF file that starts with "x" contains all plots in the folder in a single file for easy browsing.

Generating a collection of Constant Q Cuts with background subtraction

Phonon Explorer 2.0 has a new way to subtract background added.

If you simply want to subtract a quadratic background keep **BkgMode=0** and simply run **SubtractPolyBackgr.py. You do not need to provide any input to run this.** Everything, including determination of the coefficients of the parabola is done automatically. The program will create a folder **substr_background** where the background-subtracted files will be placed. It will also replot the data in the good_data folder with background plotted together with the data. If you are satisfied with this, you can skip to the end of this section.

If you are not satisfied, you can try a more complicated procedure that is retained from v1.2. This may work better if you have good reasons to belive that the background cannot be approximated by a quadratic function.

1. Run **GenerateConstQCuts.py** exactly like in the previous section, except set **BkgMode=1**.

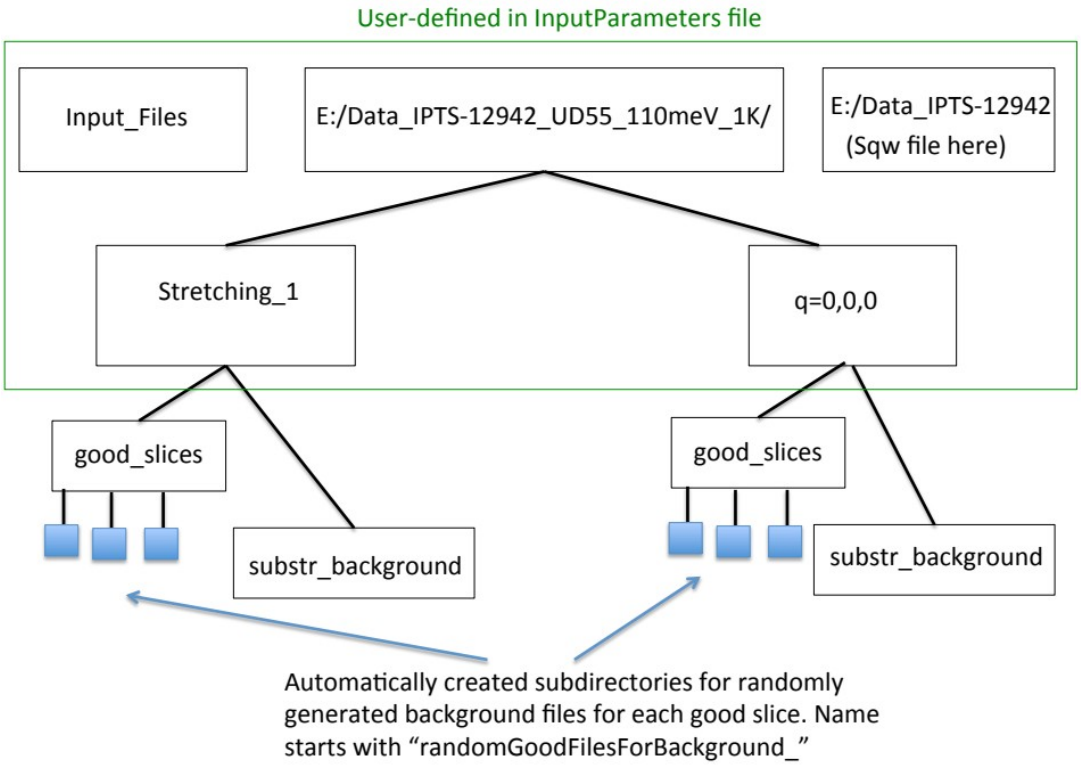
2. Then run **SubtractBackground.py** ALWAYS with Python3.8. This program calculates and subtracts background and also plots all results. It will plot all data (including in Files_for_Background folders) and save background-subtracted files in subdirectory: subtr_background. (see below)

You will have to tweak the parameter "*Resolution"* by trial and error to make sure you obtain good smoothing of background files:

(See the description of the algorithm below and input parameters in InputParameters.txt.)

Output after running SubtractBackground.py

The following directory structure is created. **GenerateConstQCuts.py+ SubtractBackground.py** were run once with input parameter ProcessedDataName=Stretching_1 and another time with ProcessedDataName=q=0,0,0



- **good_slices**: All slices that have a reasonable number of points (more than specified in the MinPointsInDataFile field in the input file) with reasonable

(<ErrorToIntensityMaxRatio field in the input file, default:30%) error bars are saved in the subdirectory "good slices" under the dataset directory.
Files with background (names start with "B_") are stored in the same directory. Files used for background determination are saved in separate subfolders for each good slice (see section How Background Subtraction works below for details). In addition plots are stored in good_slices as well.
- **substr_background**: Background-subtracted files stored here by the Python part. Results of multizone fitting will be stored here as well.

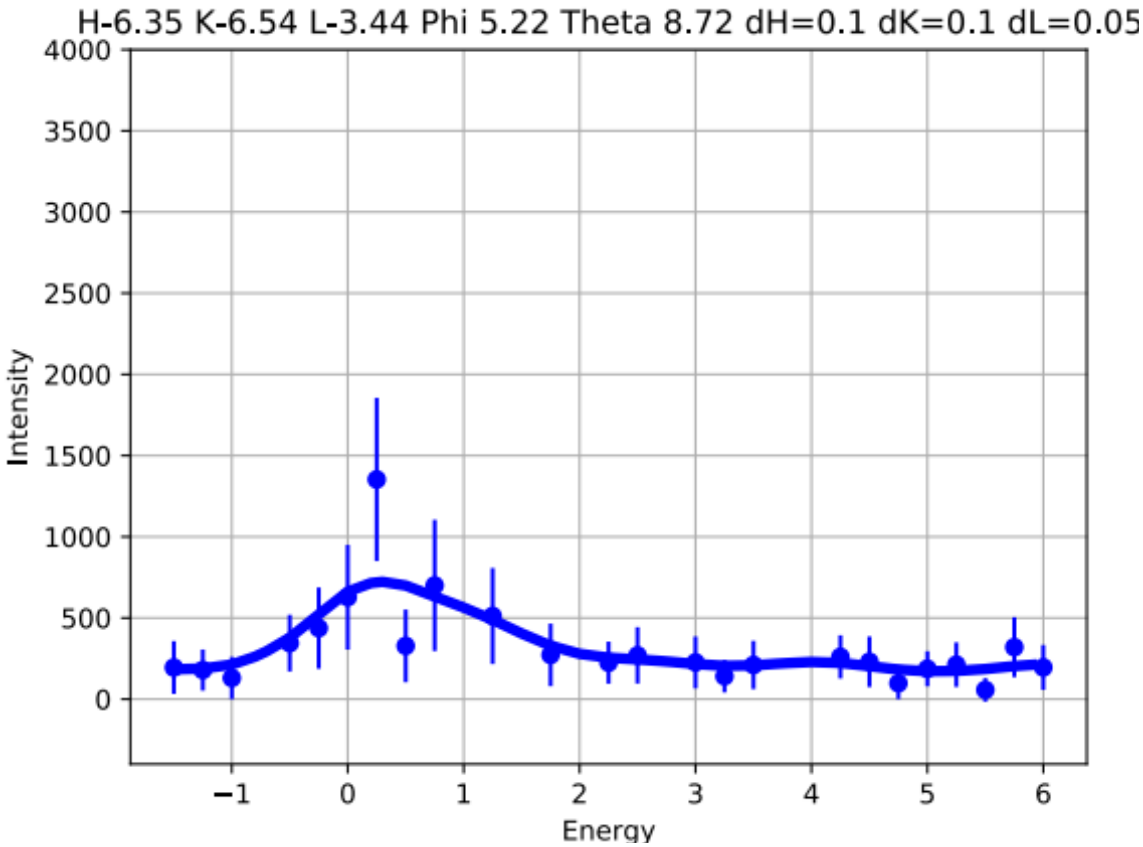
HOW BACKGROUND SUBTRACTION WORKS:

1. For each constant Q cut from which the background is subtracted the program generates cuts at wavevectors Q'. Background for that particular file is calculated from these cuts. The Q' wavevectors are generated via user-defined algorithms implemented as python plug-ins depending of the dataset. A generic "Standard" algorithm is included with this distribution (See below).

These cuts are generated one by one and saved to the "GoodFilesForBackground_..." folder until the user defined maximum number of files (maxFiles parameter) is reached. There is one such folder for each Q in good_slices.

These folders are generated by **GenerateConstQCuts.py** when the BkgMode parameter is set to 1 (**BkgMode=1**).

2. Next step is to smooth the curves. The current version of Phonon Explorer utilizes Gaussian smoothing, which is simpler and works better than the previous version documented in Quantum Beam Science 4, 41 (2020) . The result of smoothing the data in the figure below is shown as a solid curve. Be sure to set the input parameter "Resolution", which corresponds to the Gaussian width such that you are happy with the smooth curves.



Smooth curves are saved in the "GoodFilesForBackground_..." folders in text files whose name begins with "smooth". This plot also contain plots of the data in each background file together with the smoothed curve such as the one above.

Background is calculated as the point-by-point minimum of these smooth curves at every energy value. An example of randomly generated files together with the background is shown in the next figure below.

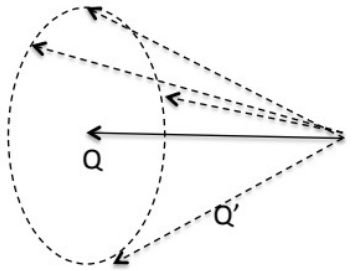
Background is subtracted from the original file in good_slices and the result together with the PDF plot is saved in a different folder called "subtr_background".

Standard Algorithm for calculating Q's:

The Standard algorithm included with this package generates Q's at random, so that it needs to be called several times to have good sampling. (See below)

← Q: Wavevector of the const. Q cut for which background is determined

<--- Q': Randomly generated wavevectors for background determination

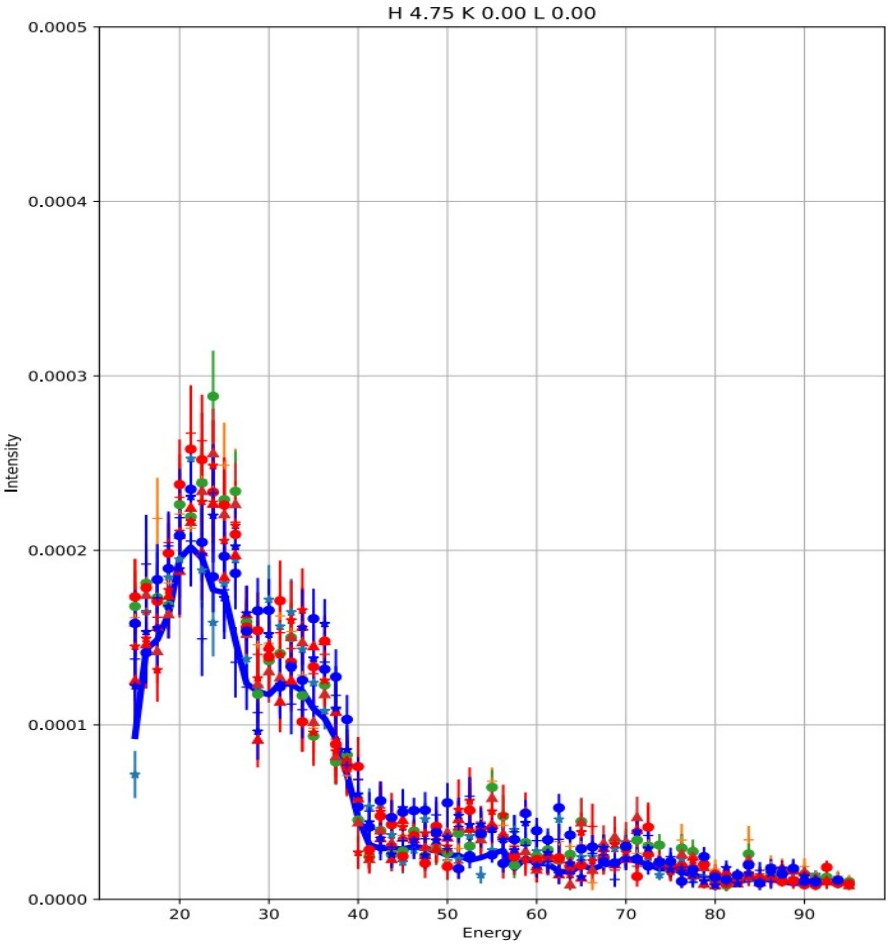


$$\left|\vec{Q}'\right| = \left|\vec{Q}\right|$$

$$\vec{Q}' \cdot \vec{Q} = \left|\vec{Q}\right| \cos(phiRange)$$

For each Q:

1. Generate slices at random wavevectors Q' such that |Q|=|Q'| and an angle, phirange, between Q and Q' is in the range determined by the program (This angle is determined automatically in the current version based of the lattice parameters in CalculatePhiTheta.m). The random slices are saved in "GoodFilesForBackground…" folders under "good_slices". The figure below shows the background (solid blue line) compared with the data at different Q's



Note that calculating the background this way significantly reduces but not completely eliminates the effect of statistics in the randomly generated cuts. In the end the background comes out slightly smaller than it is supposed to be.

User-defined generation of Q'

The folder with the Python routines contains a subfolder: Background Tools. This folder contains files with different implementations of one class called BackgroundQ that generate wavevectors (Q') and another parameter, used in background determination.

Its implementation should be edited by the user and saved under a new name. This name should be assigned to the BackgroundAlgorithm parameter in the InputParameters file. e.g. BackgroundAlgorithm=Standard

On creation, the BackgroundQ object is passes the h,k,l of the wavevector for which the background is calculated, the parameters object, and an integer index, which tells it how many times it has been called previously for this particular h,k,l. It allows the user to program it to generate a different value of Q' each time.

Only one function in the class, CalcQslash, needs to be edited by the user.

*Cite "Quantum Beam Science **4**, 41 (2020)" if you publish results obtained with the help of the software*

It is called several times, and the user can specify how many. (See example code in Python Code/Background Tools/BackgroundLSNO80meV.py

The user can also set the property mult for each Q' separately, which is the number by which to multiply the intensity of the cuts at Q'. Typically this number would be the $(|Q|/|Q'|)^2$, but it is totally up to the user to program how it is calculated. Again see Python Code/Background Tools/BackgroundLSNO80meV.py for an example.

Multizone fitting of background subtracted data

3. Run Multifit.py. Runs with Python 3.6. This program performs **multizone fit** of the files in the subtr_background folder as described in (Quantum Beam Science **4**, 41 (2020) and Phys. Rev. B **89**, 064310 (2014)).

The main idea is to fit constant Q cuts for phonons observed in different Brillouin zones but at the same wavevector keeping positions and widths the same in different zones while allowing the amplitudes to vary from zone to zone.

The program works in two modes, which are chosen automatically depending on the entries in position_guesses.txt.

Mode 1: All files in "subtr_background" folder correspond to the same reduced q:

In addition to other input parameters it requires a file where guesses for the peak positions are entered. Example of such a file is the box on the right, where 31, 35 are the initial guesses of peak positions. The name of the file is assigned to parameter **position_guesses** in InputParameters.txt. Make sure that this file has the right guesses for peak positions.

```
31 35
```

Output of this program stored in a file **_FittingParam.txt** goes to the same subdirectory (subtr_background folder). The file is in the following format:
1st row – Peak positions
2nd row – Peak linewidths
3rd row – Wavevector1
4th row Amplitudes at wavevector1
5th row—Wavevector2
6th row – Amplitudes at wavevector2
etc.

Example of this file called (_FittingParameters.txt) for two phonons observed in 5 Brillouin zones is shown at the bottom of previous page.

Example of _FittingParam.txt file. (See text)

```
33.5       52.56616861

4.78868355 8.96622224

H-2.50 K 0.00 L 0.00
0.00182147 0.00057426

H-3.50 K 0.00 L 0.00
0.0045662  0.00108261

H-4.50 K 0.00 L 0.00
0.00282977 0.00113734

H-5.50 K 0.00 L 0.00
0.00513573 0.00211125

H-6.50 K 0.00 L 0.00
0.0056896  0.00252221
```

Note that amplitudes are different at every wavevector, but peak positions and widths are the same, since the data files are at the same reduced wavevector: **q**=(0.5, 0, 0)

Errors are calculated as the sqrt of the diagonal components of the covariance matrix and are stored in the same format in the file named: **err_FittingParam.txt**

Mode 2: Files in "subtr_background" folder correspond to different reduced qs:

This mode is triggered automatically if the "guesses" file specifies also the reduced wavevectors for which multifit must be performed. The example of such a file is on the next page. Then the fit results for each reduced q are written into a separate _FittingParam/err_ FittingParam files, with the reduced q added to the end of the file name. **IMPORTANT: Remember that the position guesses file must be created by the user with the right number of peaks at each reduced q (see below for an example)!**

Example of the Position_guesses file where for specific reduced wavevectors are entered **after** corresponding guesses:

```
60 85          <- Position guesses
0 0 0          <- Reduced q
60 85          <- Position guesses
0.1 0.1 0      <- Reduced q
60 70 80
0.2 0.2 0
60 70 80
0.25 0.25 0
55 60 70 80
0.3 0.3 0
55 59 72 78
0.4 0.4 0
56 60 73 82
0.5 0.5 0
```

Auxiliary Programs

Background Adjustment functionality allows adjusting background to multiple files by adding a constant.

1. Run **MakeAdjustmentList.py**
   It creates a file BackgroundAdjustment.txt in the same folder as the background-subtracted data files. The file has 4 columns:
   First 3 are H K L , the 4th column is all zeros.

2. Replace zeros in 4th column by the constant amount you wish to subtract from the data, which would normally be different for every H K L.
3. Run **BackgroundAdjustment.py**
   The program will subtract these numbers from the data and replot everything.

IMPORTANT: If you rerun **SubtractBackground.py**, adjustments in BackgroundAdjustment.txt will be applied automatically, if you run **BackgroundAdjustment.py** again this will done twice!

If BackgroundAdjustment.txt is not there, **SubtractBackground.py** will not do any adjustments.

---

compare_data.py allows you to compare data at the same wavevectors coming from two different datasets. Normally it is used when you want to compare data at different temperatures at the same wavevectors. The program automatically makes plots for every wavevector in each directory. Edit the Python file to enter the correct directory names.