Phonon Explorer & Auxiliary Programs

Dmitry Reznik and Irada Ahmadova

Preliminary trial version

Release notes: This file contains documentation for the Phonon Explorer software and auxiliary programs

The program allows to systematically, efficiently, and rapidly explore and fit data 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 raw data. This version has the capability to read .sqw and .nxs files. Users can write plugins to read other formats. It does not crash if it fails to get the slice for a particular Q. Instead it continues to process other Qs. In addition, only slices with a 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!

Installation Instructions:

1. Install the following packages are needed:
   - Python 3.6 or 3.7 if data is stored in an SQW file
   - Python 3.8 if data are stored in an NXS file
   - Libraries:
     - Math
     - Numpy
     - Matplotlib
     - Scipy
     - PyPdf

2. If data are stored in SQW file, install Matlab, HORACE, and matlab (python) engine. Make sure that the Matlab paths are configured so that the Matlab engine can be opened from Python 3.6. (Python 3.8 does not work with Matlab yet)

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

4. In an editor open file Phonon Explorer/Python Code/RSE_CONSTANTS.py. Enter the folder of the InputParameters.txt into INPUTS_PATH_MAIN

KEY POINTS:

UV matrix Standard algorithm for subtracting background works only if uv matrix vectors are at 90 degrees to each other. Keep this in mind if you decide to use it.

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.

HOW TO RUN THE PACKAGE:
The software contains three programs written in Python that users would run: **GenerateConstQCuts.py, SubtractBackground.py,** and **Multifit.py**.

These programs read a file **InputParametes.txt**, which contains input parameters together with detailed instructions on 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, but many of the parameters are not used by all of them. Comments in the file indicate which parameter is used by which program. You can ignore or delete parameters that you don't need.

There are also plugins that could be written by advanced users to customize some of the algorithms as described below.

Depending on your scientific purpose you need to run one or more of these programs.

If you just want **a quick look at the results** without fitting or background subtraction you can just run **GenerateConstQCuts.py** and set parameter **BkgMode=0**.

If you want to **subtract background automatically-calculated based on the data**, you need to run **GenerateConstQCuts.py** with **BkgMode=1** and then run **SubtractBackground.py.**

If you want to **subtract linear background specified by you** first run **GenerateConstQCuts.py** and set parameter **BkgMode=0**. Then run an auxiliary program **BackgroundAdjustmentLine.py (see instructions in the .**

If you also need to do a **multizone fit**, you need to run Multifit.py afterwards. How to run each program is explained below in detail.

PLOTTING

The code generates lots of plots to make it easier to see the data and to understand what the code is doing. All plots have an automatically set x-axis and the maxium of the y-axis is specified by the input parameter ymax. (e.g. ymax=1000) will set the maximum of every plot at 1000. You will need to change it often depending which energy range you want to see.

In addition all datasets generated by the code as well as all fits including individual fitted peaks are saved in text files so that users can make their own publication quality plots using their favorite software.

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 as 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.

IMPORTANT: The program will crash if you use the wrong version of Python.
If your data are in an NXS file you need to set
rawDataClassFile=NXSAccessMantid
*and run the program under **Python 3.8 using mantidpython**. Otherwise it will crash.*

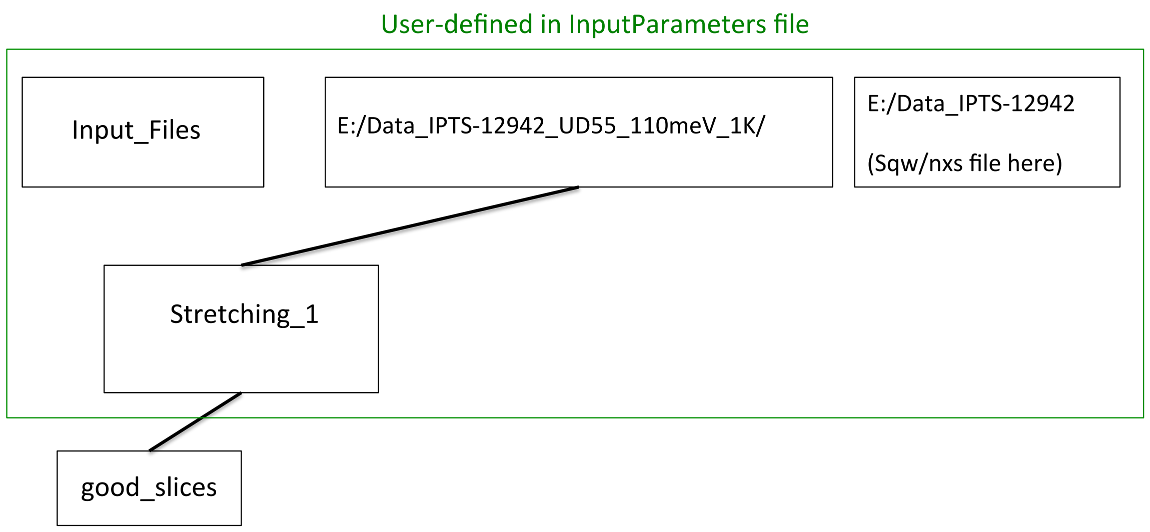
If your data are in an SQW file you need to set
rawDataClassFile=SQWAccess
*and run the program with **Python 3.6 or 3.7**. Otherwise it will crash. the Python code seamlessly calls the Getslice Matlab routine, which connects to **HORACE**, so make sure Horace and **matlab.engine** are installed.*

There are two alternative 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**
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
```

GenerateConstQCuts.py generates the following directory structure.
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

are stored in the good_slices folder.

If you run with with BkgMode=0, the good_slices directory will contain constant Q slices of the raw data at all wavevectors specified by the user and their plots.

If you run with with BkgMode=0, the good_slices directory will contain constant Q slices of the raw data at all wavevectors specified by the user and their plots as well a subdirectory of files from which to generate the background for EACH Q. The name of each subdirectrory starts with "randomGoodFilesf)orBackground_".

---

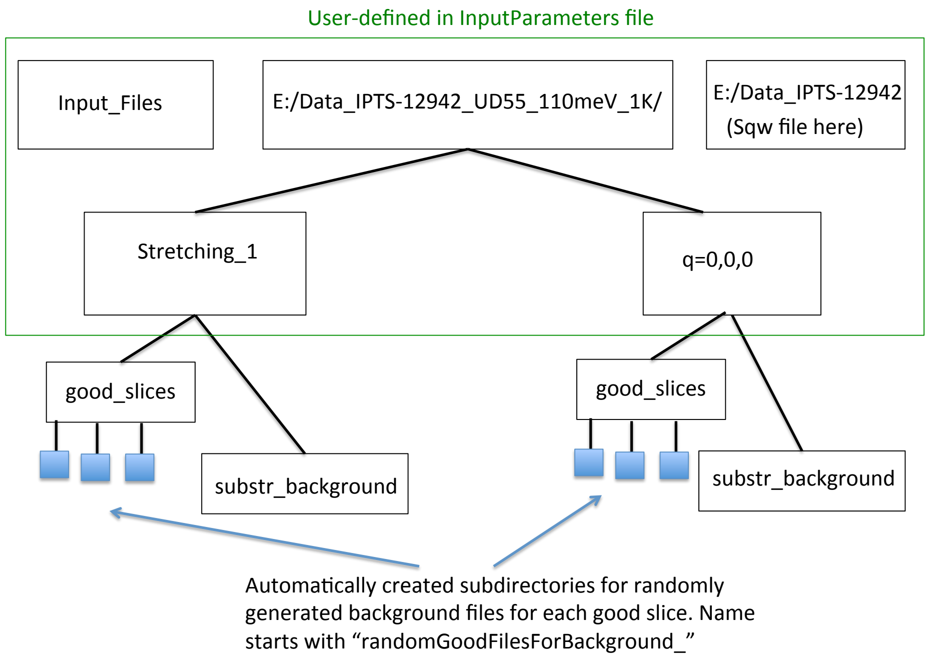
2. **SubtractBackground.py**. Make sure you already ran **GenerateConstQCuts.py** with the BkgMode parameter set to 1 (**BkgMode=1**). Then run **SubtractBackground.py**. This program calculates and subtracts background and also plots all results. It will plot all data (including in Files_for_Background folders) and save background-subtracted files in subdirectory: subtr_background. (see below)

You will have to tweak the following input parameters by trial and error to make sure you obtain good fits to background files:
Resolution, WidthLowerBoundBackground, maxfiles.

(See the description of the algorithm below and input parameters in InputParameters.txt.) Note that if maxfiles=0, then a constant will be subtracted from the data to make the lowest intensity points at each Q near zero. Otherwise a background algorithm specified in the inputparameters.txt file will be executed. (see below)

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


User-defined in InputParameters file
Input_Files
E:/Data_IPTS-12942_UD55_110meV_1K/
E:/Data_IPTS-12942 (Sqw file here)
Stretching_1
q=0,0,0
good_slices
substr_background
good_slices
substr_background
Automatically created subdirectories for randomly generated background files for each good slice. Name starts with "randomGoodFilesForBackground_"

- **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.
Files with background (names start with "B_") are stored in the same directory. Files used for background determination are saved in 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. Results of multizone fitting will be stored here as well.

---

3. **Multifit.py**. This program performs **multizone fit** of the files in the subtr_background folder as described in (Phys. Rev. B **89**, 064310 (2014)).

It knows how many peaks to fit based on how many guesses for peak positions were entered. **The current version constrains the fitted positions to be within 10% of the position guesses.** Keep adjusting your guesses until the fits give you results within 10%.

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.

In addition to other input parameters it requires a file where guesses for the peak positions are entered. The name of the file is assigned to parameter fileWithGuesses in InputParameters.txt. For example the entry could be:

fileWithGuesses=myguesses.txt

There are 2 ways to run this program depending on the format of this file.

The first option assumes that all files in the subfolder "**substr_background**" correspond to data at a single reduced wavevector q. In this case peak positions and widths should be the same in all datasets. To run the program first enter guesses for the peak positions into fileWithGuesses (in our case myguesses.txt).

Example of such a file is the box on the right, where 31, 35 are the initial guesses of peak positions.

```
31 35
```

Output is 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 for two phonons observed in 5 Brillouin zones is shown on the right.

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**

The second option assumes that all files in the subfolder "**substr_background**" correspond to data at different reduced wavevectors q.

Here you need to enter reduced qs as well as peak position guesses for each q. See example in the right.

```
0.1 0.1 0
31 35

0.2 0.1 0
34 36 37

0.3 0.1 0
32 35
```

IMPORTANT: In order for this option to work, you also need to write a Python routine specifyting how to obtain reduced q from the total wavevector Q. This version comes with the algorithm for the simple orthorhombic structure. You will need to write your own following the example or use the first option only.

In this case the output will be written to different files, one file for each reduced q. The format of each file will be the same as **FittingParam.txt** described just above, but the name will contain reduced q value. (e.g. **FittingParam_0.1_0.2_0.txt, err_FittingParam_0.1_0.2_0.txt**)

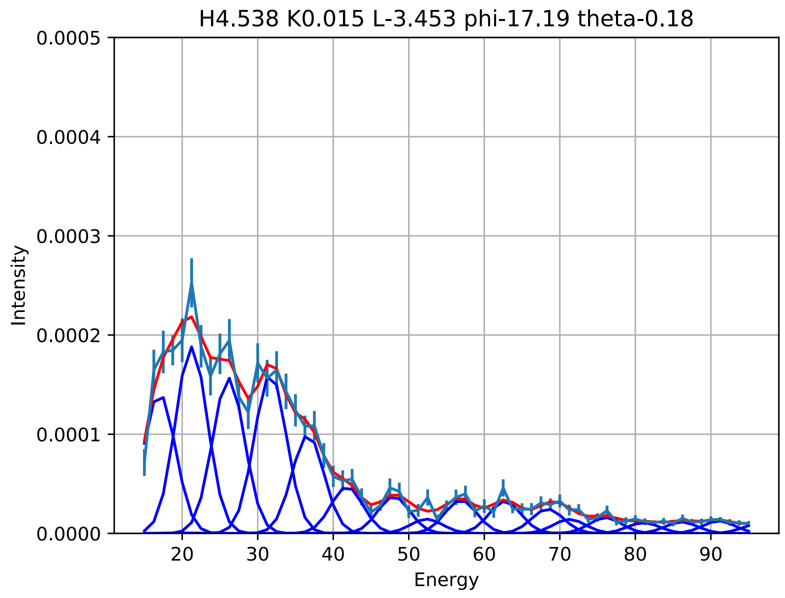
There will also be a file containing positions and position errors at each reduced q (positions.txt) as well as a file with widths and width errors (widths.txt) at each reduced q. All output goes to the subfolder **substr_background.**

HOW BACKGROUND SUBTRACTION WORKS:

The folder with the Python routines contains a subfolder: Background Tools. This folder contains routines that generate a wavevector (Q'), used in background determination. These are supposed to be written by the user and return a wavevector Q', where the background should be. These routines can be called more than once and can return different Q' every time they are called. (e.g. if Q' is generated based on a random number as is done in the Standard Algorithm supplied with this package). The program does a cut at each Q' as well as at the original Q. These cuts are saved to the "randomGoodFilesForBackground_..." folder until the user defined maximum number of files (maxFiles parameter). 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**).

SubtractBackground.py program then fits each file in these "randomGoodFilesForBackground_..." folders to multiple Gaussian peaks which produces a smooth curve through each Q' slice. *This minimal width and resolution are specified by the user in the input file and will require some trial and error.* (see below) Here is an example of such a plot, which makes the fitting procedure clear. The final curve that goes into the background calculation is the red line obtained as a sum of the individual blue peaks. **The purpose is to draw a smooth line through the data**



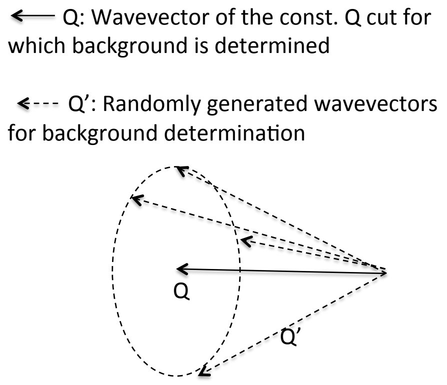
The results of the fit are saved in a text file, which contains fitting parameters. These text files go into the same folder "randomGoodFilesForBackground…" as the randomly generated cuts. Their names start with "_". The PDF files that start with "H" show these cuts together with the results of the fit. The smooth curves (red) themselves and individual blue peaks are saved in files whose names starts with "fitH..".

Background is calculated as the minimum of these smooth curves at every energy. An example of randomly generated files together with the background is shown 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".

Subtracting an automatically calculated constant background:
Just specify maxFiles=0 in the inputparameters.txt file.

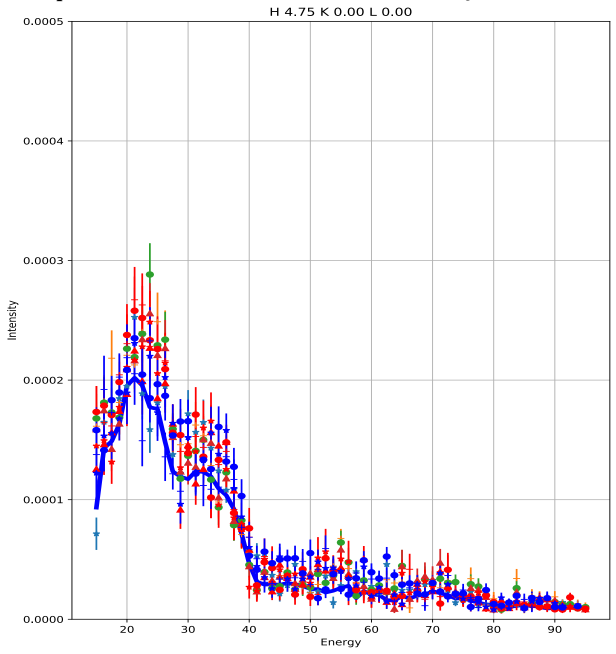
Using the Standard Algorithm for calculating Q's:
The Standard algorithm included with this package generates Q's at random, so that it needs to be called several times to have good sampling. (See below)


Q: Wavevector of the const. Q cut for which background is determined
Q': Randomly generated wavevectors for background determination
Q
Q'

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

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

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 "randomGoodFilesForBackground…" 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.

PLUGINS THAT CAN BE WRITTEN BY THE USER

Python Code folder contains 3 subdirectories with plugins that can be custom-written by users. These are

Background Tools.
Reduced q Algorithms
Tools to access raw data

These have classes that have standard methods that are called by the code, but their implementation can be customized. Do NOT change the names of classes or methods that are called by the Phonon Explorer code. You can and should save your implementations in new files whose names you would enter into the inputparameters.txt file as described below.

Background Tools.

Assign file name to the input parameter BackgroundAlgorithm.

Contains implementations of different ways of calculating Q's from the data. (see section **HOW BACKGROUND SUBTRACTION WORKS**)

The class is called BackgroundQ. (see Standard.py for an example)

Only one method is called by the code:

__init__(h,k,l,params,index)

h,k,l is the wavevector. Params points to the parameters class instance. Index counts the number of times the method was called. This allows the user to obtain different Q's each time it is called.

This method must set the following properties:
Qslash, which is a vector where Qslash[0] is the h-component of Q', Qslash[1] is the k-component of Q' Qslash[2] is the l-component of Q'.

There are also optional properties that can be set:

1. flag, which stops the generation of additional Q's when it is set to 1 (flag=1)
2. mult, which is a factor by which to multiply the data at Q'. It should be set to 1 unless there is a good reason to make it something else.

Reduced q Algorithms.

The class is called Smallq.

Only one method is called by the code:

__init__(Q)

This method should set a property: q, which should be the reduced wavevector that corresponds to Q.

Tools to access raw data.

The class is called RawData.

Two methods are called:

__init__(dataFile)
dataFile is the string with the name of the file with data. This method is called only once and it should set a global variable (we recommend in the RSE_Constants class) that would store a handle to the loaded file and/or to the matlab engine or equivalent. See how the files already present in the folder do this.

GetSlice(bin_h, bin_k, bin_l, bin_e, Projection_u, Projection_v)

These parameters are the same as in the HORACE cut_sqw command.

The function should set

self. Energy
self. Intensity
self.Error

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 and temperatures. Note that the program divides the data by the Bose factor, so

Make sure that the temperatures at which the data were taken are entered correctly.

If you don't want to divide by the Bose factor just enter very low temperatures (e.g. 1K)

BackgroundAdjustmentLine.py and **BackgroundAdjustmentLine2.py**

These allow subtraction of the linear background using user-defined slope and offset. Just add an extra column to BackgroundAdjustment.txt, which contains the slope. So to use these files each row in BackgroundAdjustment.txt must contain 5 entries:
Qh Qk Ql offset slope.

BackgroundAdjustmentLine.py subtracts background from data files in the good_files folder.

BackgroundAdjustmentLine2.py subtracts background from data files in the subtr_background folder.

Output of both goes into the subtr_background folder. When running **BackgroundAdjustmentLine2.py** the files from which linear background is subtracted are replaced with new ones.