**Program for XMLD and XMCD spectra analysis**

**Technical documentation**

# Libraries and modules.

## Numpy.

Used for conversion of .txt file containing data to approachable to work with format of data.

## Os.

Used for file handling in the process of uploading and saving.

## Matplotlib.

Used for graphical visualization of data in the form of various plots.

## Re.

Used for searching the file to find individual measurements.

## CSV.

Used for writing results to file in a form enabling further analysis.

## Tkinter.

Used for creating a graphical user interface for the program.

# Uploading.

## Uploading a file.

A standard data set is a .dat file that contains numbers of measurements. Every measurement consists of several columns proceded by several lines of comments describing measurement. Essential for analysis is the first line of comment which contain a number of measurement and its type and two columns of the data.

To load a file *select\_file()*  function is used. It opens File Explorer which allows for choosing a file. If the file is correct the program writes the file line by line to variable *Lines* and closes the file

To find all measurements existing in the whole data set the function *list of measurement()* is used.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**Figure 1.** *List of measurement()* function.

There are two regular expressions used:

* pattern\_all – finding all measurements
* pattern\_rscan – finding only measurements made as a function of energy (subject of analysis)

Variable *total* is the number of all measurements in the data set and is used in the program for creating a list of available measurements (measurements are saved in the data set sequentially from the first to the last).

Measurement numbers made as a function of energy are saved in list *rscans*.

## Opening specific measurement.

For opening specific measurements for the purpose of plotting, for example, the function *search in file(nb)* is used.

Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie

**Figure 2.** Regular expressions in *search in file* function.

There are two regular expressions used (Figure 2.):

* pattern\_first – the first line of commentary of a given measurement (*nb* – number of the measurement you are looking for)
* pattern\_last – the last line of data of a given measurement

To obtain the number of the first line of data of a given measurement, the number of comment lines is added to the number of the first line of commentary (Figure 3.).

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

**Figure 3.** Searching for the first line of data in the *search in file* function.

The function returns a tuple with a number of the first and the last line of data.

# Displaying plots.

## Plotting spectra.

For displaying spectra *Plotter* function is used. The function takes five parameters, including two optional ones:

* *measurement\_nb* – the number of spectra that you want to plot
* *ax* – matplotlib Axes object that you want to use to plot spectrum
* *mode* – the function has two modes of spectra displaying which are activated by passing to the function proper string:

- ‘virgin’: In this mode the raw data is used to draw the graph. This mode is used in the first stages of the program.

- ‘sub and normalized’: In this mode data after normalization and eventual background removal is plotted.

* *w* (optional) – first parameter of fitted background (strip background width)
* *iter\_nb* (optional) – second parameter of fitted background (strip background number of iterations)

The function writes data for specific measurement to a temporary file and extracts two columns from it and save to ndarray *data* (Figure 4.).

Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie

**Figure 4.** Preparing data for the plotting.

The creation of variable *plot\_obj* allows for getting settings used for plotting.

## Plotting results.

For displaying results the argumentless function *plot\_results* is used.

Obraz zawierający tekst, Czcionka, linia, zrzut ekranu

Opis wygenerowany automatycznie

**Figure 5.** Determination of x values.

If the user wants to specify values for the x-axis, the title, and the values will be passed to variables *x\_tittle* and *x\_tab*. They are both strings, so elements in *x\_tab* have to be converted to float numbers. After that, they are saved in the list *x\_int*.

If the user doesn’t want to specify values, *x\_title* is set as ‘x’ and *x\_int* as a list of integers from 0 to the number of the measurements.

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

**Figure 6.** Plotting all spectra.

The first plot displayed shows all analyzed spectra. For plotting the function *plotter* is used. Measurement numbers and background parameters are received from the *accepted\_p* dictionary (Figure 6.).

RL for both SROT and ΔRL coefficients are plotted based on results from appropriate lists: *RL3\_0\_tab*, *RL3\_0\_tab*, *dRL3\_tab*.

# Background removal.

## Function for fitting background - *background*.

For background fitting the program uses a strip background model. It is an iterative procedure based on two parameters:

* Strip background width (*Width*)
* Strip background number of iterations (*Iterations*)

At each iteration, if the contents of channel i, y(i), is above the average of the contents of the channels at w channels of distance, y(i-w) and y(i+w), y(i) is replaced by the average. At the end of the process, we are left with something that resembles a spectrum in which the peaks have been “stripped”.

## Function for fitting background - *background*.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**Figure 7.** Fitting the background – *background* function.

The function has three parameters:

* Number of measurement channels (*channel*)
* Strip background width (*w*)
* Strip background number of iterations (*iter\_nb*)

As a first step, two lists are created:

* *y\_mean* –list with intensity values for individual measurement channels (*y\_mean(i)* in the code is equivalent of y(i) in the description of the algorithm from the previous section)
* *temp* – temporary list for storing data for the duration of loop iteration

Using two for loops strip background procedure is implemented in the function.

## Accepting background removal settings – *acc\_all* function.

### No background removal or using the same parameters for all spectra.

If the user doesn’t want to subtract the background or use the same background removal parameters for all spectra, he ticks the proper check button which results in calling the appropriate function (*noback* or *oneback*).

In the *acc\_all function*, if one of the above options has been selected, all spectra are plotted using for loop and measurement numbers obtained from the list *accepted* (Figure 8.).

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

**Figure 8.** The first case in the *acc\_all* function.

### Individual selection of background removal parameters for each spectrum.

Obraz zawierający tekst, Czcionka, linia, zrzut ekranu

Opis wygenerowany automatycznie

**Figure 9.** The second case in the *acc\_all* function.

If the user wants to select background removal parameters for each spectrum individually procedures after *else* are executed (Figure 9.). The spectra are plotted based on the measurement numbers and background removal parameters obtained from the dictionary *accepted\_p*.

# Detection of the peaks.

## Function *peaks\_detection*.

The function has three parameters, including two optional:

* *nb* – measurement number
* *w* (optional) - strip background width
* *iter\_nb* (optional) - strip background number of iterations

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**Figure 10.** Preparing spectra for detection in the *peaks\_detection* function.

In the first step, data is converted to a two-column form as in the *plotter* function (3.1). Both columns are saved to the lists: *data­\_x* (x values) and *data\_y* (y values). In the next step, if background removal parameters were provided to the function, the background is subtracted. As a final step of the spectrum preparation for peak detection, spectra are normalized using the *normalization\_vec* function.

### Automatic peak detection.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**Figure 10.** Automatic detection is implemented in the *peaks\_detection* function.

To find peaks the function *find\_peaks* from the scipy module is used. Founded peaks in the form of channel numbers are saved in the *peak* variable. Next, x and y values for these channels are saved in *peak\_x* and *peak­\_y* lists.

If the automatic mode is on, the function saved RL results in the *RL3\_tab* list, unless the program failed to find the peaks. In this case, variable *no\_peaks* is set as *True*. This is information for the program that using manual mode is necessary.

### Manual mode.

If manual mode is on, the function *picking\_points*, which is described in the next paragraph, is used.

## Function *picking\_points*.

The function has three parameters, including two optional:

* *nb* – measurement number
* *w* (optional) - strip background width
* *iter\_nb* (optional) - strip background number of iterations

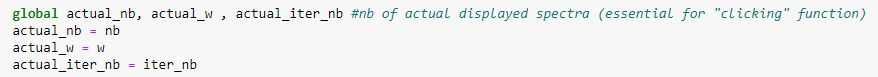
In the first step, like in *find\_peaks* (5.1.1.), the correctness of automatic peak detection is checked (Figure 11.).

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

**Figure 11.** Checking if automatic detection was successful.

In the next step, global variables, which will be used in the *clicking* function, are created (Figure 12.).



**Figure 12.** Creating variables necessary for the *clicking* function.

After that, the spectrum with the provided number is plotted together with automatically found peaks (if automatic detection was successful). After that, the function *clicking* connected with the right-clicking event is called (Figure 13.). This function is described in the next paragraph.

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

**Figure 13.** The last part of the *picking\_points* function.

## Function *clicking*.

The function allows you to select points on the chart and delete them.

### Selecting the points.

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

**Figure 14.** Selecting points in the *clicking* function.

If some point is right-clicked, the closest x value and its index in the actually displayed spectrum are saved to the *closest\_val*. That variable is created through the use of *enumerate*, *min,* and a *lambda* function as a key. The search method is as follows:

* enumerate(data\_x): This part of the code uses the enumerate function to iterate through the elements of the data\_x along with their corresponding indices. It creates pairs of (index, element) for each element in data\_x
* key = lambda x: abs(x[1] - event.xdata): This is a lambda function that defines a custom key for the min function. It takes each pair (index, element) generated by enumerate(data\_x) as x and calculates the absolute difference between x[1] and event.xdata (x coordinate of the selected point). This lambda function is used to determine the comparison value for each element in the iterable when finding the minimum
* min(...): The min function is used to find the minimum element from the iterable generated by enumerate(data\_x) based on the custom key defined by the lambda function. In other words, it finds the element in data\_x that has the smallest absolute difference from event.xdata

In the next step, the point defined through *closest\_val* is displayed on the spectra and its coordinates are saved in *x\_man* and *y\_man* lists. Additionally, coordinates of the point are displayed.

### Deleting the points.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**Figure 15.** Deleting the last point in the *clicking* function.

If the right mouse button is clicked, a spectrum with marked automatically found peaks (if automatic detection was successful) is plotted. Besides, the last chosen point is removed from the *x\_man* and *y\_man* lists. Finally, peaks from the *x\_man* and *y\_man* lists are displayed (without the deleted point).

## Function *pick\_manual*.

The function is used in the process of accepting manually chosen peaks for the purpose of saving calculations based on the choices. The function has one parameter:

* *prev\_nb* –measurement number for which peaks have been accepted

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**Figure 16.** Function *pick\_manual*.

The global variable *man\_nb* is created to provide information on the number of analyzed spectra. It allows the detection mode to end at the appropriate moment and repetition of analysis for the spectra in case of some errors in detection.

### Rejection of automatically found peaks by the user.

If the user chooses two peaks their heights are used for calculations and the results are saved in the *RL3\_tab* list. Moreover, the positions of the peaks are saved in the dictionary *man\_peaks*.

If the user chooses not exactly two peaks, the pop-up window is displayed and the *man\_nb* variable is decremented by one which allows for repetition of the analysis.

### Acceptance of automatically found peaks by the user.

To accept automatically found peaks user should not select any peak. In that case, heights of automatically searched peaks are used for calculations and the results are saved in the *RL3\_tab* list. The positions of the peaks are saved in the dictionary *man\_peaks*. If the automatic detection was failed, the pop-up window is displayed and *man\_nb* variable is decremented by one what allows for repetition of the analysis, instead of this.