MDC_cut.py
User Manual
version 6.5.2 — 2025–07–10

Chih-Keng Hung

September 2, 2025

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Purpose of the Manual

This manual provides comprehensive instructions for the `MDC_cut.py` script, designed to quickly view experimental data and provide a more efficient data processing environment.

1.2 Target Audience

This manual is intended for researchers, scientists, and technicians who work with experimental data and use the **Spectrium** software (produced by **PREVAC sp. z o.o.** `https://prevac.eu/` for data acquisition and instrumentation control). Additionally, this manual is suitable for users who are interested in processing ARPES data obtained from other sources using the `MDC_cut.py` script.

1.3 Overview of Features

The `MDC_cut.py` script can handle data from the **Spectrium** software and process it intuitively. The script includes the following commonly used functionalities:

- **File Attribute Modifier:**
  - Modify Name, Excitation Energy, or Description when H5/JSON files are loaded.
  - Enables users to correct possible wrong attributes of the file.
- **Plotter:**
  - Visualize data in various ways.
  - Includes useful tools embedded in the plotting interface.
- **MDC/EDC Cutter:**
  - Cut MDC/EDC data and export the data to TXT files.
  - The exported data can be loaded by **CasaXPS** for further analysis.

- **MDC Fitter:**
  - Fit MDC data with Lorentzian functions with a linear baseline.
  - The fitting results can be visualized in real-time in the interface.
  - The fitting results can be exported to NPZ files.
- **Batch Master:**
  - Open all the data in **Spectrogram** interface.
  - Process the data to fit the experimental geometry and visualize the data in real / reciprocal space. Also, slice the data in the desired direction and export the data to H5 or NPZ files if needed. (In reciprocal space)
  - Batch export all the data stacked in one VMS file.
- **Graph Exporter:**
  - Export graphs to **OriginPro**, enabling users to perform further analysis or editing.

1.4 Installation Guide

The installation section provides instructions for installing the `MDC_cut.py` script in your Python environment.

1.5 Version Information

This manual applies to version 6.5.2 of the `MDC_cut.py` script, released on July 10, 2025. The script is compatible with Python 3.13.5.

Chapter 2

Installation

To use `MDC_cut.py`, ensure you have Python installed on your system. You can download Python from `https://www.python.org/`. The script is written in Python, and the operating system is suggested to be Windows OS. Remember to add Python to your system's PATH. There will be a check box for this option during the installation process. In addition to the standard Python packages, the following packages are required:

- numpy
- matplotlib
- h5py
- scipy
- xarray
- lmfit
- tqdm
- win32clipboard (pywin32)
- originpro
- PIL (Pillow)
- cv2 (opencv-python)
- cpuinfo (py-cpuinfo)
- psutil

You can get the `MDC_cut.py` script from the GitHub repository: `https://github.com/alex20000910/main.git` To install the required packages, just execute `MDC_cut.py`, and the script will ask for your permission (Y/N) to automatically install the required packages using `pip install` command in your terminal, or you can install the required packages manually by executing the following command in your terminal:

```
pip install matplotlib numpy h5py scipy xarray lmfit tqdm
   win32clipboard originpro Pillow opencv-python py-cpuinfo psutil
```

in your terminal.

You can run the script by two ways:

1. In your terminal, navigate to the script's directory (using `cd [PATH]`) and execute the following command: `python MDC_cut.py`
2. Execute the `MDC_cut.py` in the file explorer.

Chapter 3

Interfaces

3.1 Main Interface

3.1.1 Overview

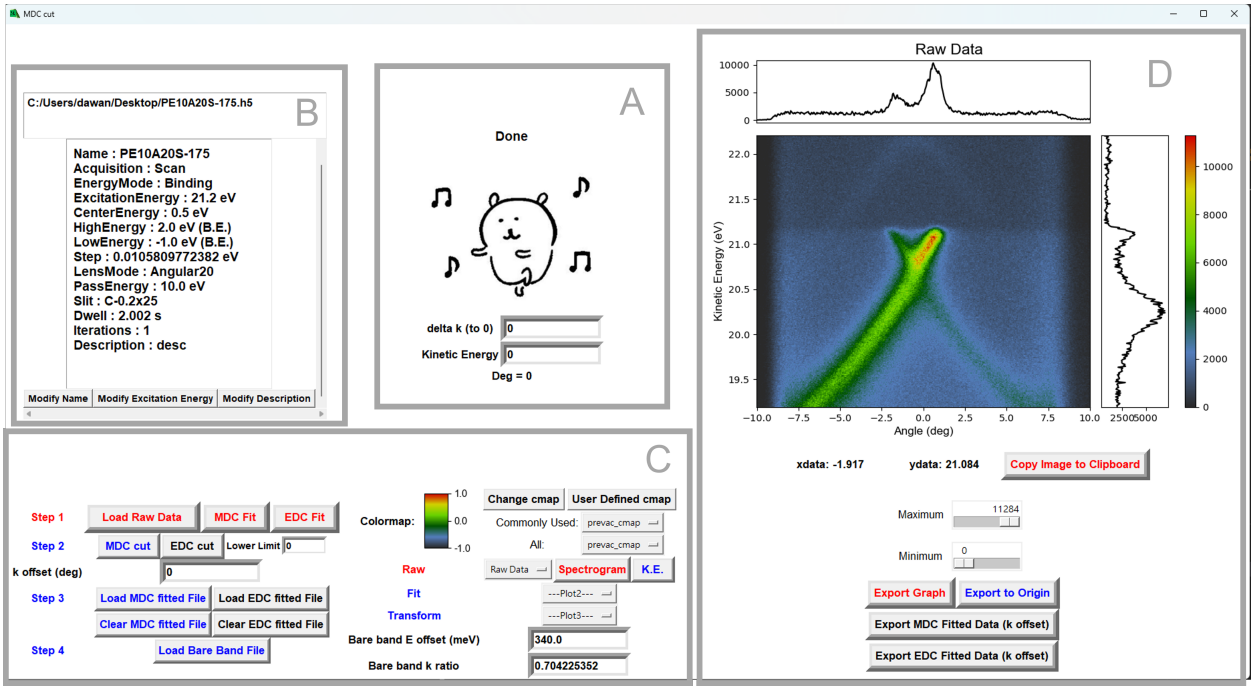


Figure 3.1: Main Interface of `MDC_cut.py`

The contents of the main interface are divided into four sections, as shown in Figure 3.1. The reference for each section is listed in Table 3.1. The whole interface requires a lot of Confirm buttons to ensure the user's intention. Notice that you can use the Enter key to confirm the action, which can save you a lot of time. Once you have chosen a certain plot, you can simply press the Enter key to plot it again with the new argument you have set or the new data you have loaded.

Section	Object	Ref
A	Spirited Companions	Section 3.1.2
	Status Indicator	Section 3.1.3
	E-k Angle Converter	Section 3.1.4
B	File Information Viewer	Section 3.1.5
	File Attribute Modifier	Section 3.1.6
	Batch Master	Section 3.1.7
C	Data Importer	Section 3.1.8
	MDC/EDC Cutter	Section 3.1.9
	MDC Fitter	Section 3.3
	Spectrogram	Section 3.2
	Plotter	Section 3.1.10
	Argument Setter	Section 3.1.11
D	Main Plotting Interface	Section 3.1.12
	Graph Exporter	Section 3.1.13
	TXT Exporter	Section 3.1.14

Table 3.1: Summary of Script Sections

3.1.2 Spirited Companions

The Spirited Companions section contains various delightful icons for each action, which play a crucial role in ensuring the user's experience is as enjoyable as possible. [Figure 3.2] The different icons randomly show up when the user does any action in the script.

Figure 3.2: Spirited Companions

3.1.3 Status Indicator

The Status Indicator is located upper of the Spirited Companions, which shows the current state of the script.

3.1.4 E-k Angle Converter

The E-k Angle Converter is a calculator for converting k-values to degrees, which is required for calibrating the E-k dispersion relation. [Figure 3.3] The raw data from the ARPES

experiment is usually in the form of degrees, and a slightly misaligned angle usually exists in the data. The E-k Angle Converter can help you determine the angle offset by converting the k-values to degrees.

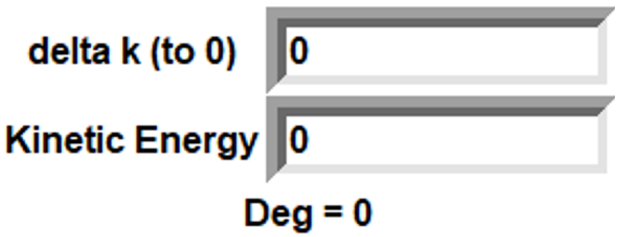


Figure 3.3: E-k Angle Converter

3.1.5 File Information Viewer

The File Information Viewer displays detailed information about the loaded data file. The information includes all the attributes of the data file, such as Name, Excitation Energy, and Description. [Figure 3.4]

3.1.6 File Attribute Modifier

The File Attribute Modifier allows you to adjust attributes like Name, Excitation Energy, or Description for H5/JSON files. This feature ensures that the incorrect file attributes can be corrected efficiently. [Figure 3.5]

3.1.7 Batch Master

The Batch Master components only show up when multiple data files are loaded. [Figure 3.6][Figure 3.7] The option list allows you to choose the data you want to process using main interface, and the Batch Master button provides three options for you to choose:

- **Spectrogram:** Open all the data in **Spectrogram** interface.3.2
- **k-Plane:** View the constant energy surface of the data in real or reciprocal space. The features of this utility will be described in detail in Section 3.4.
- **Export to Casa:** All data are exported and combined into a single VMS file, which includes the attributes of each dataset. The resulting VMS file can be loaded into **CasaXPS** for further analysis. Figure 3.8 illustrates how the VMS file appears when loaded in **CasaXPS**. The exported VMS file contains all datasets loaded in the main interface, together with their respective information.

3.1.8 Data Importer

There are three types of data importer in the Data Importer section: Experimental Data, Fitted Data, and Bare Band Data. [Figure 3.9]

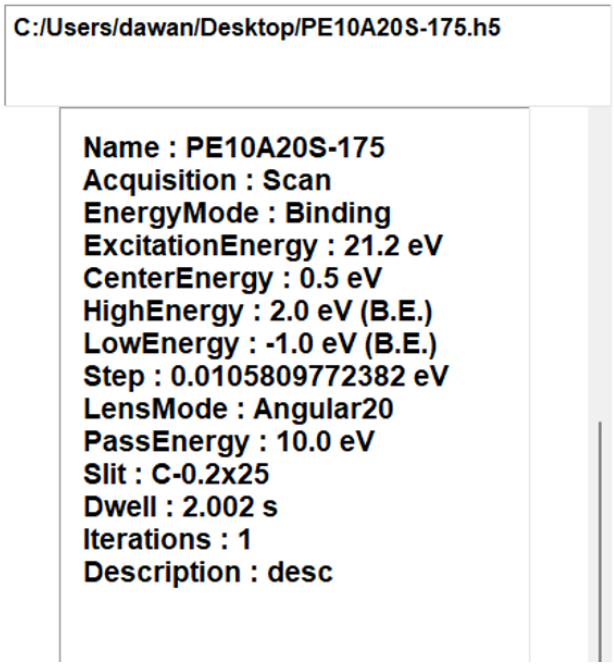


Figure 3.4: File Information Viewer

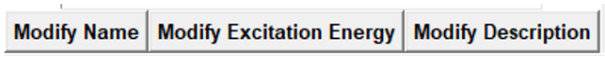


Figure 3.5: File Attribute Modifier

Experimental Data

The Experimental Data Importer allows you to load H5/JSON files generated by the **Spectrium** software. Moreover, you can load the data from TXT files generated elsewhere. The data structure is described in Section 4.1.

Fitted Data

The Fitted Data Importer allows you to load the fitted data from the NPZ/VMS files. The data structure is described in Section 4.2.

Bare Band Data

The Bare Band Data Importer allows you to load the bare band data from TXT files. The data structure is described in Section 4.3.

3.1.9 MDC/EDC Cutter

The MDC/EDC Cutter allows you to cut MDC/EDC data and export the data to TXT files. [Figure 3.10] There is an entry for you to set the lower bound of the intensity to cut the data if needed. All the TXT files are saved in a folder named **FileName_MDC_BaseCount**

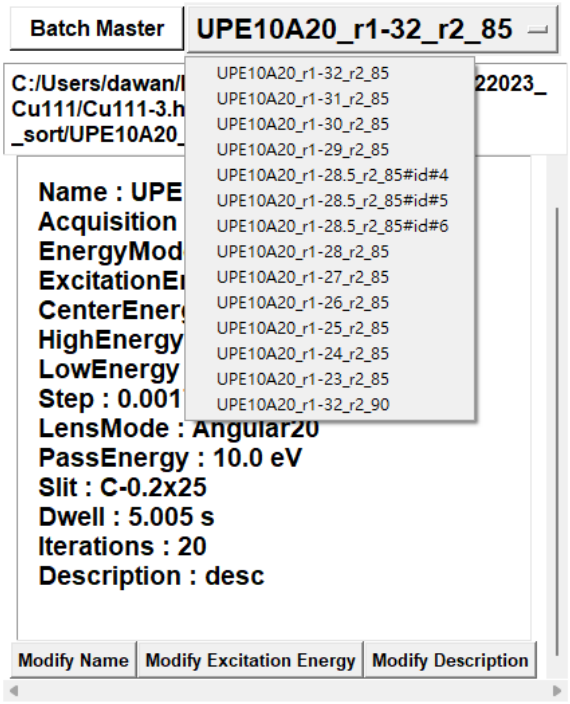


Figure 3.6: B section of the Main Interface when multiple data files are loaded

Figure 3.7: Batch Master

or **FileName_EDC_BaseCount** in the same directory as the experimental data. The exported data can be loaded by **CasaXPS** for further analysis.

3.1.10 Plotter

The Plotter contains a set of option list for you to choose the desired type of plot and a color map list for you to choose the color map of the plot. [Figure 3.11] The available plot types are listed in Table 3.2, and the color map can be customized by the user using the User Defined cmap button. This function allows you to define your own colormap and save it in one NPZ file, and this file can be loaded next time you need it. Figure 3.12 shows the interface of the custom colormap editor, color bar preview, and the color selector. Users can click on the color blocks in the window to open the color picker, set color positions using multiple input fields, and freely add or remove color blocks. The Register & Save button is used to save custom colormaps, the Load Colormap button is for loading previously saved custom colormaps, and the Show Colormap button previews the currently configured color bar. For instructions on embedding multiple custom colormaps in the script, please refer to Section 4.4.

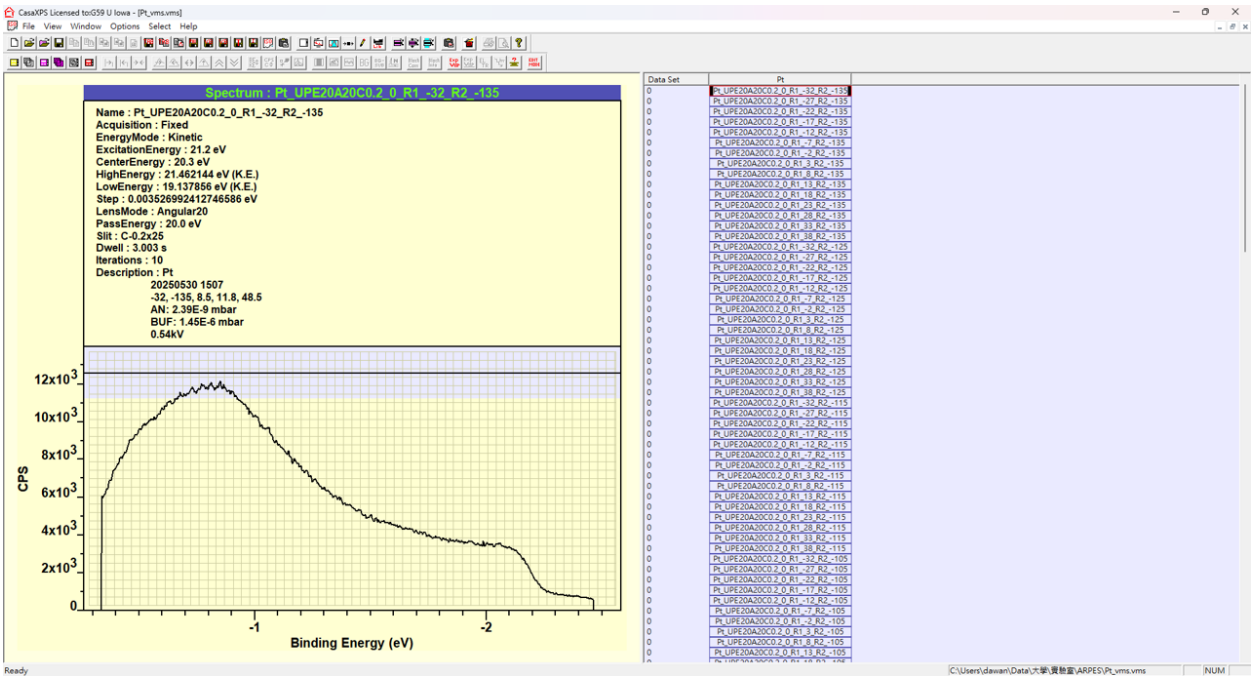


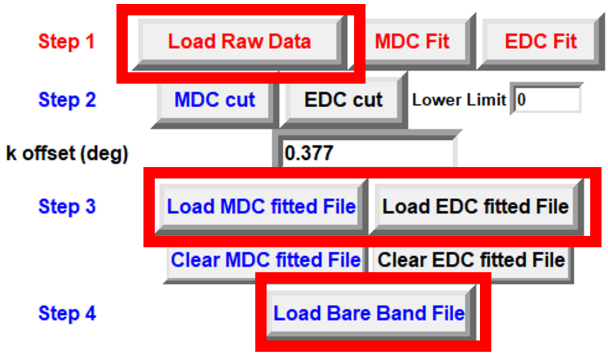
Figure 3.8: View of the VMS file loaded by CasaXPS

3.1.11 Argument Setter

The Argument Setter allows you to set the arguments for the desired experimental variables.

Angle Setter

The Angle Setter allows you to set the angle offset of the data. [Figure 3.13] You can set the angle offset by finding the center of data using raw data plot. The data indictator will show the current position of your mouse. Alternatively, if you could determine the center of the data in k-space, by using the k-value to degree converter, you can set the angle offset to the value you have determined.



Red box indicates the three types of data importers.

Figure 3.9: Data Importer

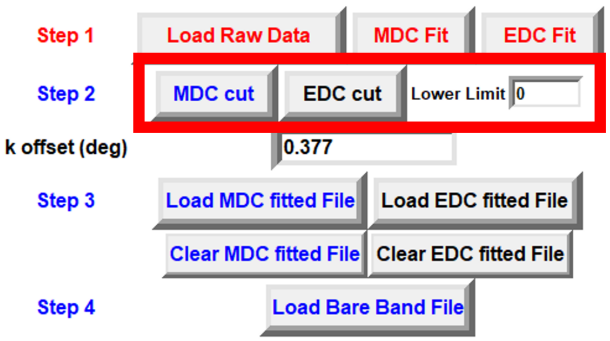


Figure 3.10: MDC/EDC Cutter

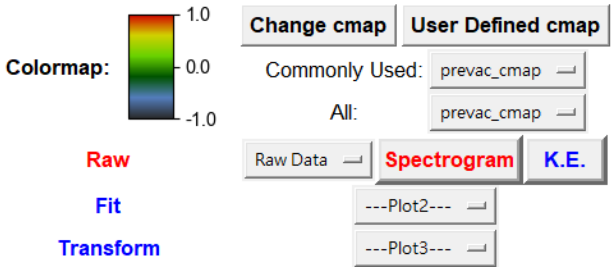


Figure 3.11: Plotter

Energy Setter

This section allows you to set the energy offset of the bare band data. [Figure 3.14] Take the case of Graphene as an example, the energy offset is usually set to the Fermi level (Binding Energy = 0). But if you have a doping sample, you can set the energy offset to the Fermi level + the doping energy to align the bare band data (calculated without doping) to the experimental data.

Ratio Setter

The Ratio Setter allows you to set the slope factor of the bare band data. [Figure 3.15] The slope factor is used to calibrate the bare band data to the experimental data. As usual, the slope factor is set to 1. However, if you have misaligned bare band data, you can set the slope factor to a value you have determined for a specific reason.

3.1.12 Main Plotting Interface

This section displays the main plotting window, which is used to visualize the data. [Figure 3.16] There are several tools embedded in the plotting interface.

- **Value Indicator:** Show the current position of your mouse.
- **Zoom Control:** Zoom in: Left-click and drag, zoom out: Right-click.
- **Color Scaler:** Change the color scale of the plot for better visualization.
- **Plot Copy:** Copy the plot to the clipboard.
- **Data Slicer:** Show the data slice of the current position of your mouse.

Some tools might not be available for specific types of plot. Table 3.2 lists the availability of the tools for each type of plot.

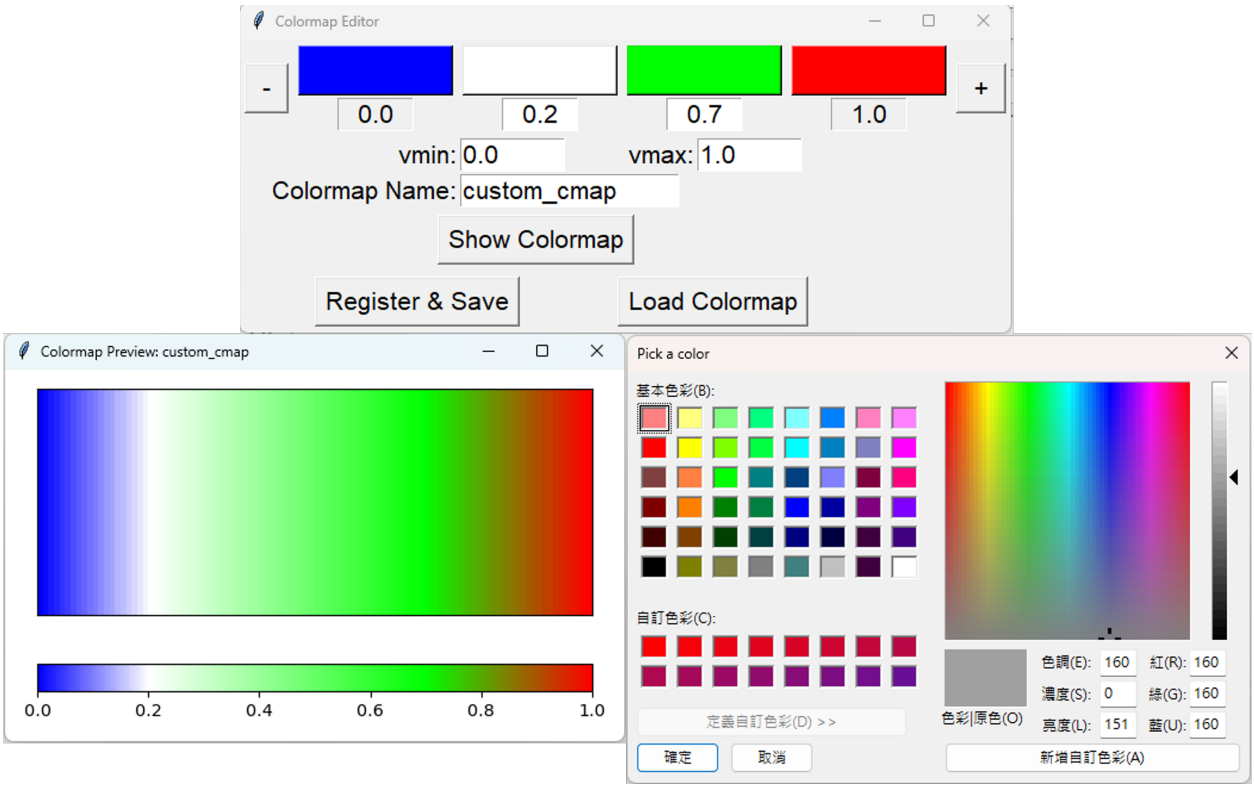


Figure 3.12: Custom Colormap Editor

Figure 3.13: Angle Setter

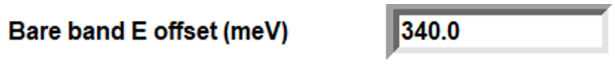


Figure 3.14: Energy Setter

3.1.13 Graph Exporter

The Graph Exporter is shown in Figure 3.17. It allows you to export the graph in two ways:

- **Export Graph:** Export the graph by **matplotlib**'s built-in function, enabling users to save the graph in various formats. [Figure 3.19] The Graph Exporter has the tools with a similar usage as the main plotting interface. The availability of the tools is shown in Table 3.3.

- **Export to Origin:** Export the graph to **OriginPro**, enabling users to do further analysis or editing. [Figure 3.18] The graph and the data will be exported to a .opj file, which can be opened by **OriginPro**. All the parameters of the experimental data and the variables set by **Argument Setter** will be recored in the Data Information Note. The reason why using .opj file is that the later version of **OriginPro** is still compatible with the older version while the older version of **OriginPro** is not compatible with the later format (.opju). However, export the data to a later project format is available by modifying the script.

  Please search for the keyword "`origin_temp_save =`" in the script to find the place to

Category	Plot Type
Raw	Raw Data
	E-k Diagram
	MDC Normalized
	First Derivative
	Second Derivative
	MDC Curves
	E-k with MDC Curves
Fit	MDC fitted Data
	EDC fitted Data
	Real Part
	Imaginary Part
Transform	Real & Imaginary
	KK Transform Real & Imaginary
	KK Transform Real Part
	KK Transform Imaginary Part
	KK Transform Real Part 2nd Derivative
	KK Transform Imaginary Part 1st Derivative
	Data Plot with Pos
	Data Plot with Pos and Bare Band

Table 3.2: Plot Types

Figure 3.15: Ratio Setter

Plot Type	Value Indicator	Zoom Control	Color Scaler	Plot Copy	Data Slicer
Raw Data	✓	✓	✓	✓	✓
E-k Diagram	✓	✓	✓	✓	✗
MDC Normalized	✓	✓	✗	✓	✗
First Derivative	✓	✓	✓	✓	✗
Second Derivative	✓	✓	✓	✓	✗
MDC Curves	✓/✗	✗	✗	✓	✗
E-k with MDC Curves	✓/✗	✗	✓/✗	✓	✗
MDC fitted Data	✓/✗	✗	✗	✓	✗
EDC fitted Data	✓/✗	✗	✗	✓	✗
Real Part	✓/✗	✗	✗	✓	✗
Imaginary Part	✓/✗	✗	✗	✓	✗
Real & Imaginary	✓/✗	✗	✗	✓	✗
KK Transform Real & Imaginary	✓/✗	✗	✗	✓	✗
KK Transform Real Part	✓/✗	✗	✗	✓	✗
KK Transform Imaginary Part	✓/✗	✗	✗	✓	✗
KK Transform Real Part 2nd Derivative	✓/✗	✗	✗	✓	✗
KK Transform Imaginary Part 1st Derivative	✓/✗	✗	✗	✓	✗
Data Plot with Pos	✓	✓	✓	✓	✗
Data Plot with Pos and Bare Band	✓	✓	✓	✓	✗

✓: Available, ✓/✗: Unavailable when using **Graph Exporter**, ✗: Unavailable.

Table 3.3: Tool Availability for Each Type of Plot

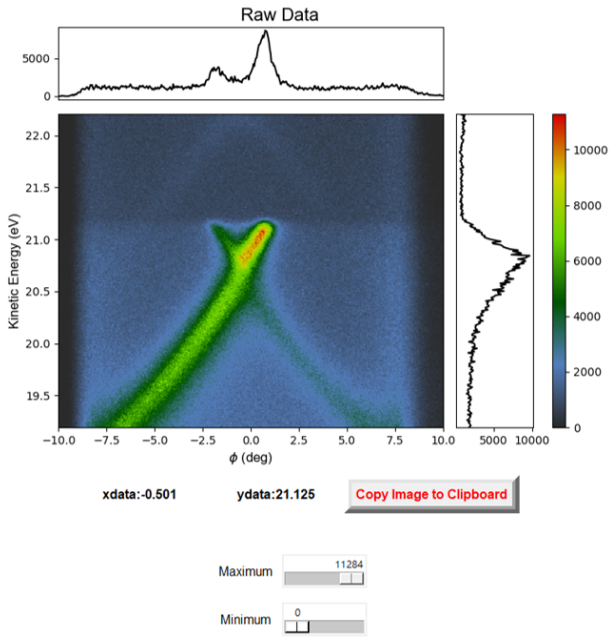


Figure 3.16: Main Plotting Interface

modify if you want to export the data in **.opju** format. You'll see the following code snippet:

```
    origin_temp_save = r'''
note()
save()
'''
```

Replace `save()` with `save('opju')` and save the script. Then run the script again.

3.1.14 TXT Exporter

The TXT Exporter allows you to export the MDC/EDC fitted data to TXT files. But the **Export to Origin** function is more recommended if you have the **OriginPro** software installed.

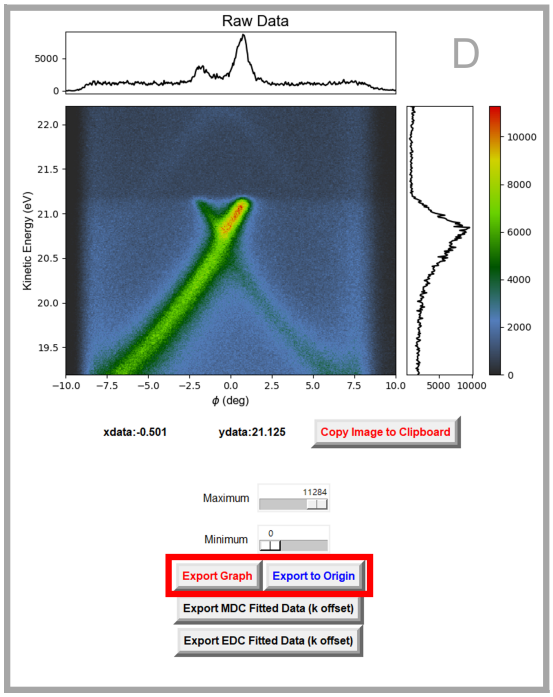


Figure 3.17: D section of the Main Interface

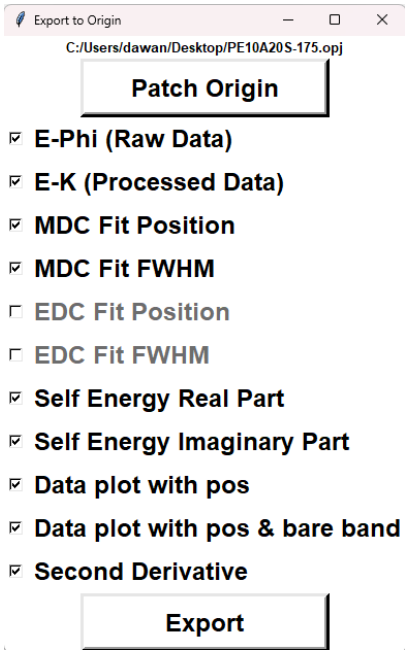


Figure 3.18: Graph Export Interface for **Export to Origin**

3.2 Spectrogram Interface

The spectrogram interface is used to visualize the spectrogram of the loaded data. It allows the user to open multiple spectrogram windows simultaneously to view different data. The same usage as the main plotting interface, zooming in/out, and copying the plot are all possible with the tools provided. [Figure 3.20] And there are three types of plots available: Raw, Smooth, and First Derivative. [Figure 3.21] The spectrogram interface also provides a data range adjustment bar, allowing the user to select the range of data to be summed and displayed as a line profile. The user can adjust the position of the red triangle symbol or drag the green bar to choose the interesting region of the data. To set the precise position of the region, right-click the adjustment bar and enter the value in the pop-up window (see Figure 3.22). Then, do the Fermi Level Fitting by clicking the Fermi Level Fitting button if needed. Or export the data to the format you want.

This interface is packed in a class; it can be used in other scripts by importing the class from `MDC_cut.py`.

```
spectrogram(g: Tk, data: DataArray, cmap: str)
```

- `g:Tk` - A graphical user interface object obtained by `tk.Tk()`.
- `data:DataArray` - The data to be visualized, which is packed in a `xr.DataArray` object. The details of the data structure are described in Section 4.1.
- `cmap:str` - The color map of the spectrogram.

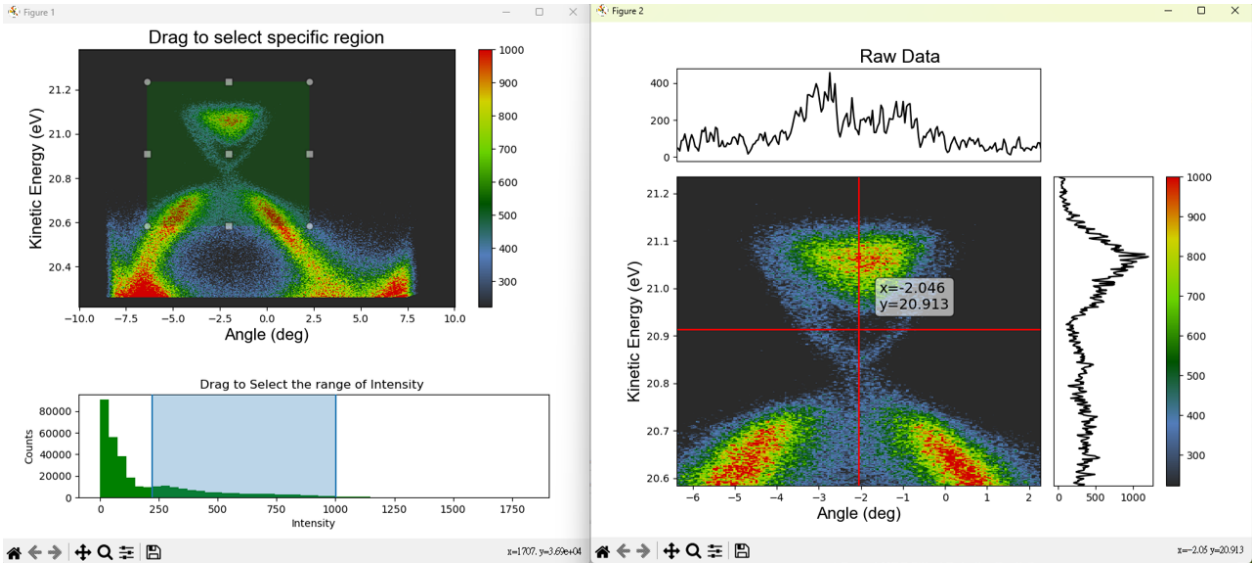


Figure 3.19: Graph Export Interface for **Export Graph**

Usage example:

```
from MDC_cut import *

root = tk.Tk()
path = r"\path\to\data.h5"
data = load_h5(path)
spec = spectrogram(root, data, 'viridis')
spec.plot()
root.mainloop()
```

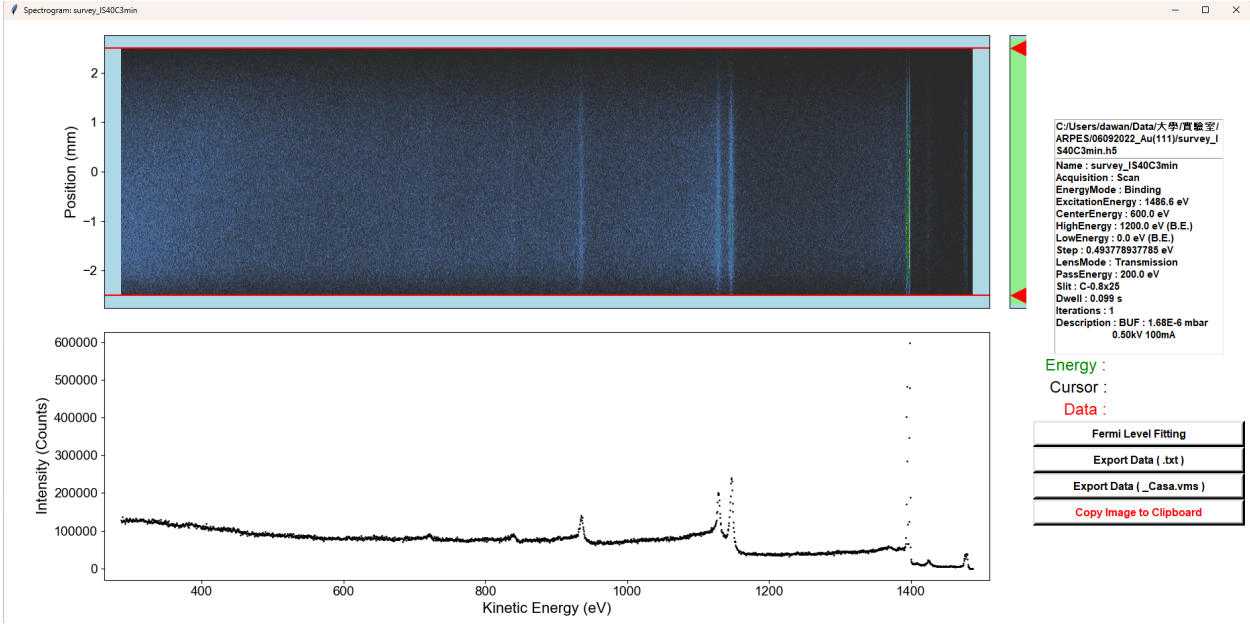


Figure 3.20: Spectrogram Interface

Figure 3.21: Spectrogram Plot Type

3.3 MDC Fitter Interface

The MDC Fitter interface contains the fitting results of the MDC data. The fitting procedure is designed somewhat similarly to **CasaXPS**, but the fitting results can be visualized in the interface real-time, which provides tremendous assistance to the user while dealing with the MDC data obtained from the ARPES experiment. The real-time display function can be turned on/off as needed, and the fitting results can be exported to NPZ files. [Figure 3.23]

3.3.1 Index Selector

The **Index Selector** shows the current index and the fitting status of the current curve. [Figure 3.24] By using the [<<] / [>>] buttons, the index can be quickly switched to the previous or next curve with a different fitting status. There are three types of fitting status:

- ■**Unfitted:** The curve has not been fitted yet.
- ■**Fitted:** The curve has been fitted successfully.
- ■**Failed:** The curve has failed to be fitted.

The selector will change the color of the buttons [<<] / [>>] according to the fitting status, which offers a clear view of the fitting status of each curve. The user can directly change

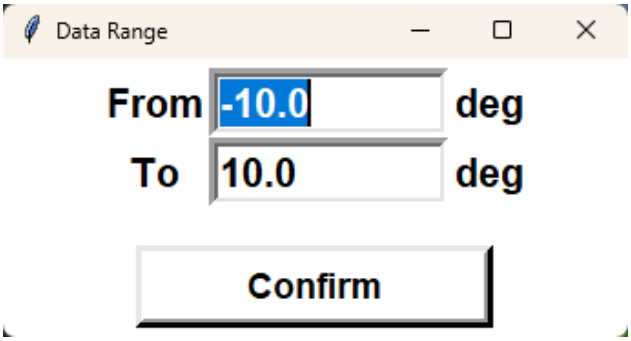


Figure 3.22: Data Range Adjustment

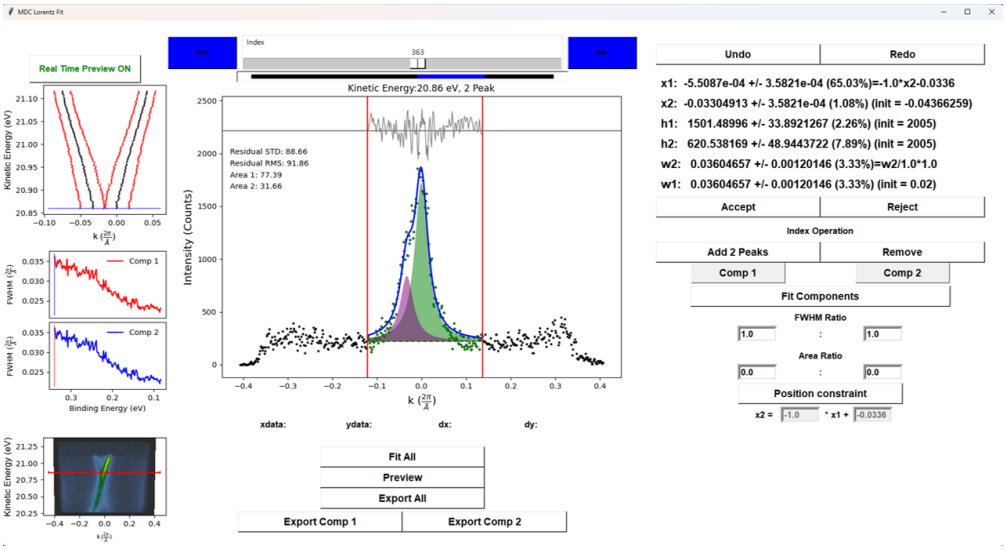


Figure 3.23: MDC Fitter Interface

the index by dragging or clicking the slider. Also, the user can change the index one by one using the < / > on the keyboard.

3.3.2 Interactive Plotter

The **Interactive Plotter** is used to visualize the fitting results of the MDC data. [Figure 3.25] The fitting process is done in this place, and the fitting results are displayed in real-time. This window allows the user to drag the peak position (Left-click and drag), adjust the baseline (Up / Down on the keyboard), and zoom in/out the plot (Left-click and drag then Left-click the box / Right-click). The cursor data and the box data are displayed at the bottom of the plot. The upper part of the plot shows the fitting residuals with gray line. Notice that the red line indicates the fitting region. The user can adjust the fitting region by dragging the red line. And the region will automatically move linearly between two separated indexes, enabling a more sufficient fitting process. It means that the user only needs to select the fitting region for the start and stop curve, and the fitting region will be automatically adjusted for the curves between the start and stop curve. If the region is not

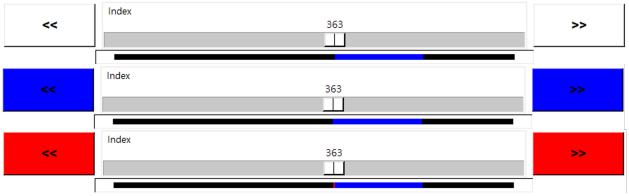


Figure 3.24: Index Selector

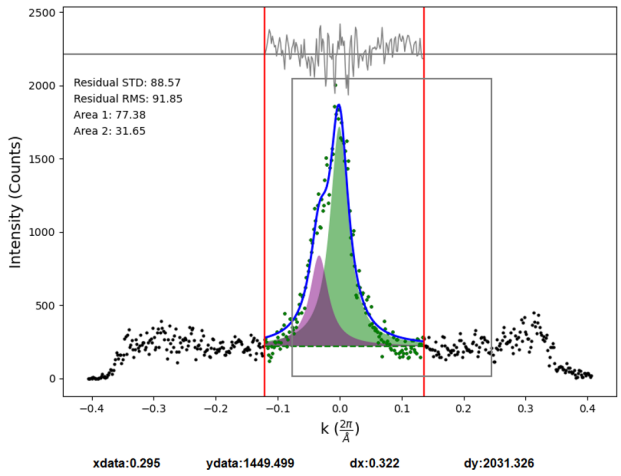


Figure 3.25: Interactive Plotter

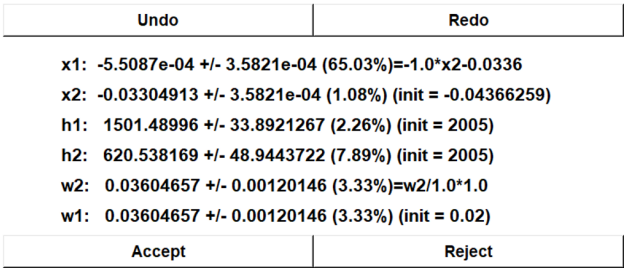


Figure 3.26: Fitting Result

suitable for the curve, the user can adjust the region manually by dragging the red line.

3.3.3 Fitting Result

This section shows the fitting parameters of the current curve. [Figure 3.26] The fitting parameters include the peak position, the full width at half maximum (FWHM), and the height of the peak, which are crucial for determining the Lorentzian peak. The Undo / Redo buttons allow the user to undo/redo the fitting process. And the Accept / Reject buttons allow the user to accept/reject the fitting results of the current curve.

3.3.4 Fitting Constraint

This section allows the user to set the fitting constraints for the peak position, FWHM, and area. [Figure 3.27] The Position constraint button should be **ON** state if the user wants to use the position constraint. The user can choose a certain peak position by clicking the Comp 1 / Comp 2 button, and dragging the peak position in the **Interactive Plotter** to the desired position. Also, the user can decide if the data should be deconvoluted to two peaks or not. And the user can remove the fitting region in batch by clicking the Remove button to start the remove mode, then change the index to the end of the range that needs to be removed. Finally, click the Remove button again to remove the region between these two indexes. The usage of Add 2 Peaks is similar to the Remove button. Remember to click

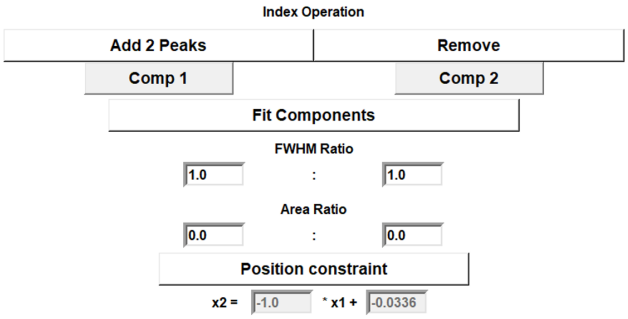


Figure 3.27: Fitting Constraint

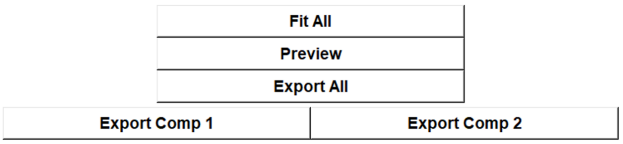


Figure 3.28: Export Fitting Results

the Fit Components button to start the fitting process after setting the fitting constraints or dragging the fitting peak position.

3.3.5 Export Fitting Results

This section allows the user to export the fitting results to NPZ files. [Figure 3.28] Fit All button will fit all the curves that have the fitting region defined. This operation will only apply on those curves that have not been fitted or failed to be fitted. The user can preview the fitting results here, and export the fitting results to NPZ files by clicking the Export All / Export Comp 1 / Export Comp 2 button. Please always remember to save the fitting results after finishing the fitting process no matter using the Export All / Export Comp 1 / Export Comp 2 button. They will save all the fitting results you have done in the current session.

3.3.6 Real-time Display

The real-time display function can be turned on/off by clicking the Real Time Preview ON/OFF button. There are some technical issues that the whole fitting process will be slower when the real-time display function is turned on. So it is recommended to turn off the real-time display function when fitting the data one by one. The real-time display function is more suitable for users who want to fit data by referring to nearby curves, requiring a more convenient guidance visualization for the fitting process.

3.4 k-Plane Interface

The k-Plane interface is used to visualize the constant energy surface of the data in real or reciprocal space. The real space [Figure 3.29], referring to the transmission mode in the interface, constant energy mapping could help users choose where the better position to probe and aquire the data. The reciprocal space [Figure 3.30], referring to the reciprocal mode in the interface, constant energy mapping shows the k-space information of the data. The user can choose a certain region in the reciprocal space to slice the whole data cube to an E-k 2-D data for further analysis. The figures below are just for demonstration purpose. The data presented in the figures were obtained from the ARPES experiment on the Cu (111) surface; however, the file name does not precisely correspond to the experimental conditions. Instead, some duplicate data files are used to check the function of the interface.

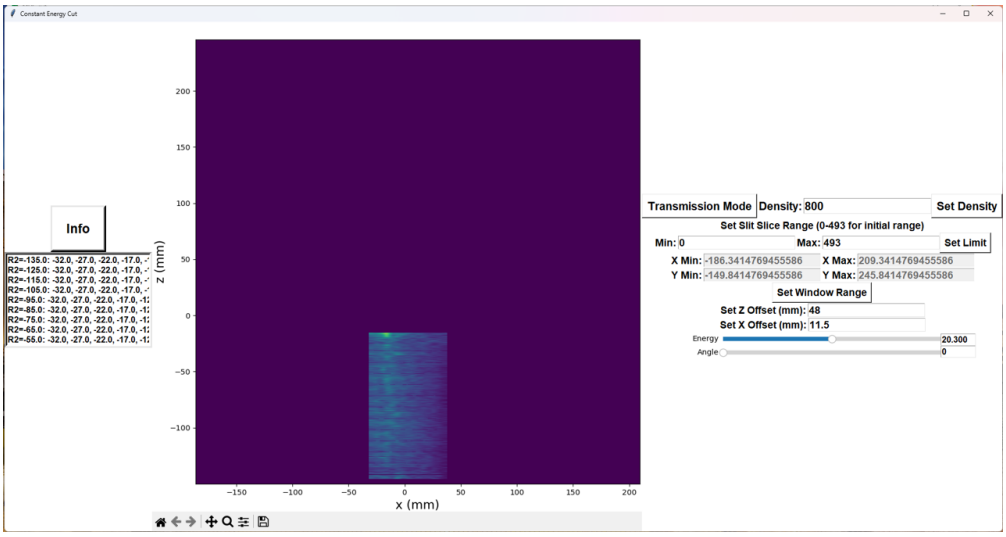


Figure 3.29: k-Plane Real Space

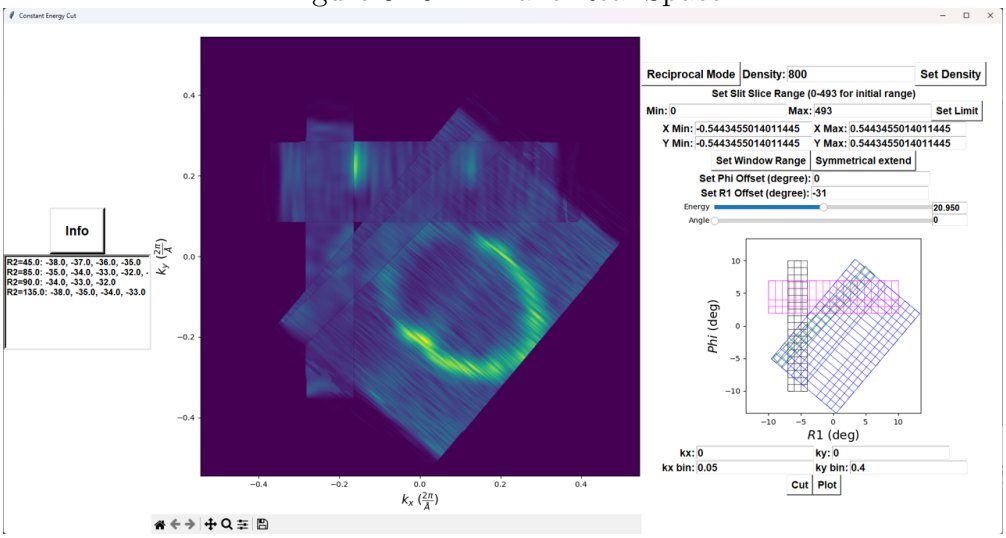


Figure 3.30: k-Plane Reciprocal Space

3.4.1 Data sorting

Please note that all the data files should be better acquired from the same experimental condition, such as the same `Slit, Energy Setup, Pass Energy, etc`. A data cube will be generated from the data files you loaded. By stacking them with the specified `R1, R2` values in the file name, the raw data series will be well-aligned in the k-space.

The proper-named data files are required to be loaded in order to use this interface. The data file should be specified clearly showing the 'R1' or 'R2' as well as the 'X' or 'Z' in the name of the data file. Both the lowercase and uppercase letters are acceptable in ideal case. The seperator between the name and the 'R1' or 'R2' could be '_' or ' '. Notice that these are the keywords for the script to determine the geometry of the data using the same definition as the **Spectrium** software. The file name should be like '`Name1_R1_-31_R2_85.h5`', '`Name2_X_15_Z_48.h5`', '`Name3_X15Z48.h5`', etc. If the file name is not well defined, this interface will not be able to work properly, so the k-Plane button will not show up in the batch master.

That's say all the format of the data file names meet the requirements, the script will automatically determine the geometry of the data and load the data in the correct way. Also, the script will automatically determine if there is a repeat **R1, R2, X, Z** combination in the loaded dataset. If so, it'll be a selector window pops up for you to choose the data file you want to load. [Figure 3.31] Notice that this function is for convenience only, actually the user should gather the proper data in a folder to avoid the confusion of the repeat **R1, R2, X, Z**.

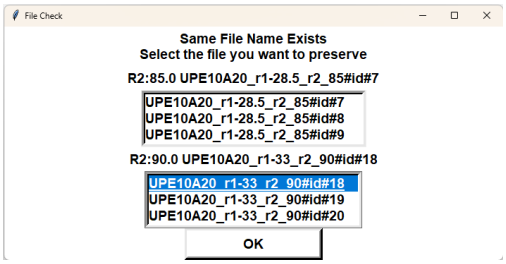


Figure 3.31: k-Plane File Selector

3.4.2 Calibration

The Phi and R1 calibration are crucial for k-space mapping for all the k-values should be calculated based on the correct zero point of Phi and R1 values. The Phi value can be set in the **Set Phi Offset** entry, and the R1 value can be set in the **Set R1 Offset** entry. After setting the value, the user can either click the button Set Limit or just press Enter on the keyboard to refresh the plot (automatically save the graph to the clipboard).

The corresponding geometry of Phi and R1 values is shown in [Figure 3.32]. These two values are used to tilt the data in degree. The R1 rotation axis is the same as the R1 motor, and the Phi rotation axis is perpendicular to the R1 and R2 axes, which might be the sixth axis that the motor not installed in our instrument. The angle slider in the interface can be recognized as the R2 motor, which is used to rotate the data in the R2 axis. The calibration

is done by keep adjusting the Phi and R1 values to make the dataset show a good rotational symmetry in the R2 axis. As long as the data is well aligned with the R2 axis, we can probably assert that the calibration is done.

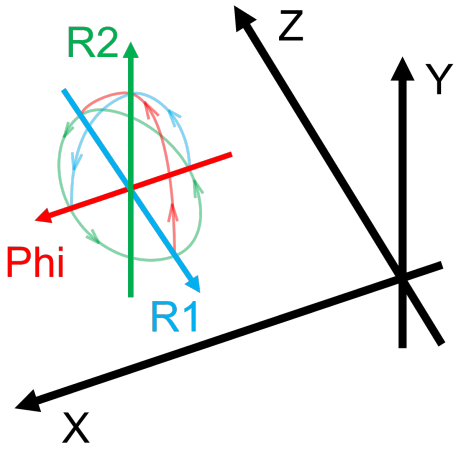


Figure 3.32: Spatial Orientation & Rotation Axes

3.4.3 Symmetry

The Symmetrical extend button is used to set the symmetry of the data. After clicking the button, a window appears allowing users to select the symmetry mapping form, offering 2-fold, 3-fold, and 4-fold options. This feature maps the current constant energy surface according to the selected symmetry to other angular positions, creating a complete ($0° \sim 360°$) constant energy surface (the image will be automatically copied to the clipboard). When this feature is enabled, you can also perform E-k diagram slicing, and the sliced data will include the symmetry-mapped data.

Figure 3.33 shows the constant energy map before setting the symmetry. The data shows a 120-degree covered region in the k-space. As it is known that the data is from a Pt (111) surface, the symmetry could be set as **3-fold** to cover the whole k-space.

After setting the symmetry, the data will be extended in the k-space as shown in Figure 3.34. The data is now extended to cover the whole k-space, which is a 360-degree region.

3.4.4 Data Cube Slicing

Sling is only available in reciprocal mode. [Figure 3.30] When processing the data, you might discover the region near the boundary not so ideal as expected. That's because the data is acqured through a 2D detector, and the photoelectrons are not collected to project onto full of the detector. Therefore, the data in the boundary region is not so reliable as the data in the center region. **Min, Max** entries could help user to solve this kind of problem by limiting the data range along the original angle-resolved axes. These two entries should be set as integer values, ranging from 0 to 493. This range represents the number of pixels in the

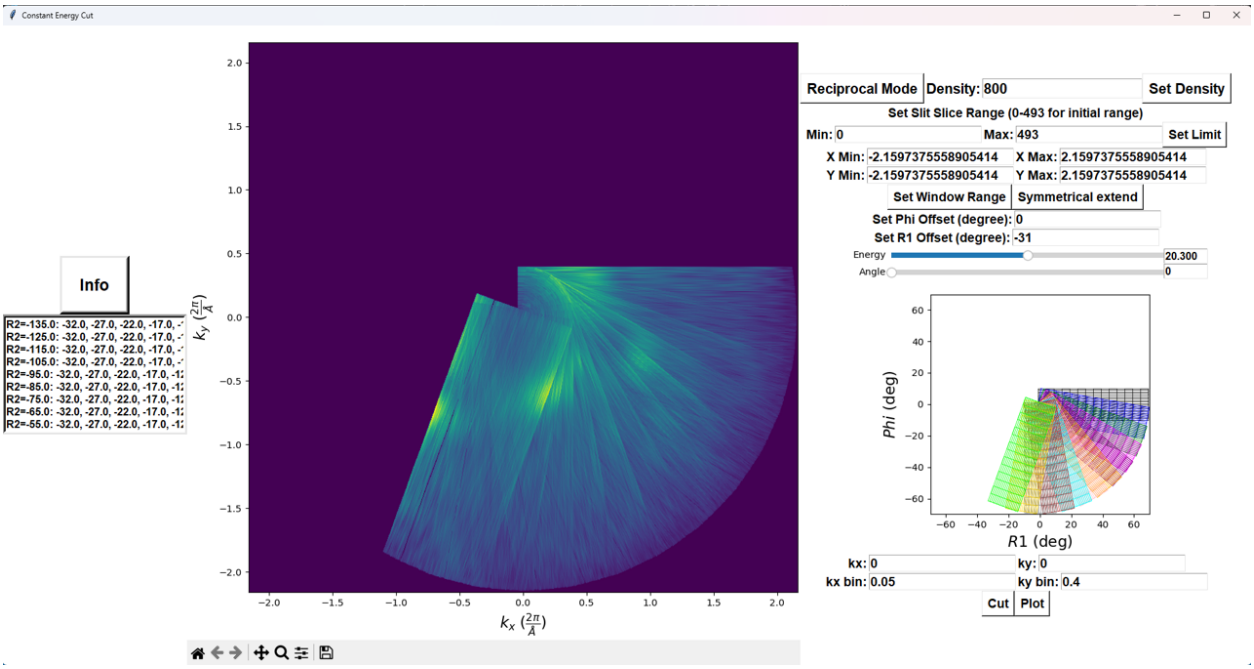


Figure 3.33: Constant energy map before symmetical extend

detector along the angle-resolved axes. (The energy axes of the detector has 659 pixels, which is not adjustable in this interface, while the angle-resolved axes has 494 pixels.) Finally, you can click the button Set Limit or simply press Enter on the keyboard to refresh the plot.

The user can use **kx, ky, kx bin, ky bin** entries and click Cut to specify the cutting region in the reciprocal space, then modify the proper **Angle** to slice the data cube in the desired direction. [Figure 3.35]

After checking the cutting region, the user can click the Plot button to start the slicing process. This operation will spend a little time to slice the data cube, so please be patient and wait for the plot to show up. The progress will be shown in the command window. [Figure 3.36]

Press Enter to show the result when the slicing process is done. The sliced data (E-k diagram) will be shown in a tk.Toplevel window [Figure 3.37].

Please click Save button or press Enter to save the sliced data to a H5/NPZ file. The H5/NPZ file will save a similar data structure as the raw data file obtained from **Spectrium** software, but some additional attributes like the data paths, the processing parameters, and calubration values will be added to the file. The next time you load the sliced data (H5/NPZ file) through the **Data Importer** in the main interface, the script will automatically retrieve the data series and display the previous processing status in the k-Plane interface, provided these data paths are accessible.

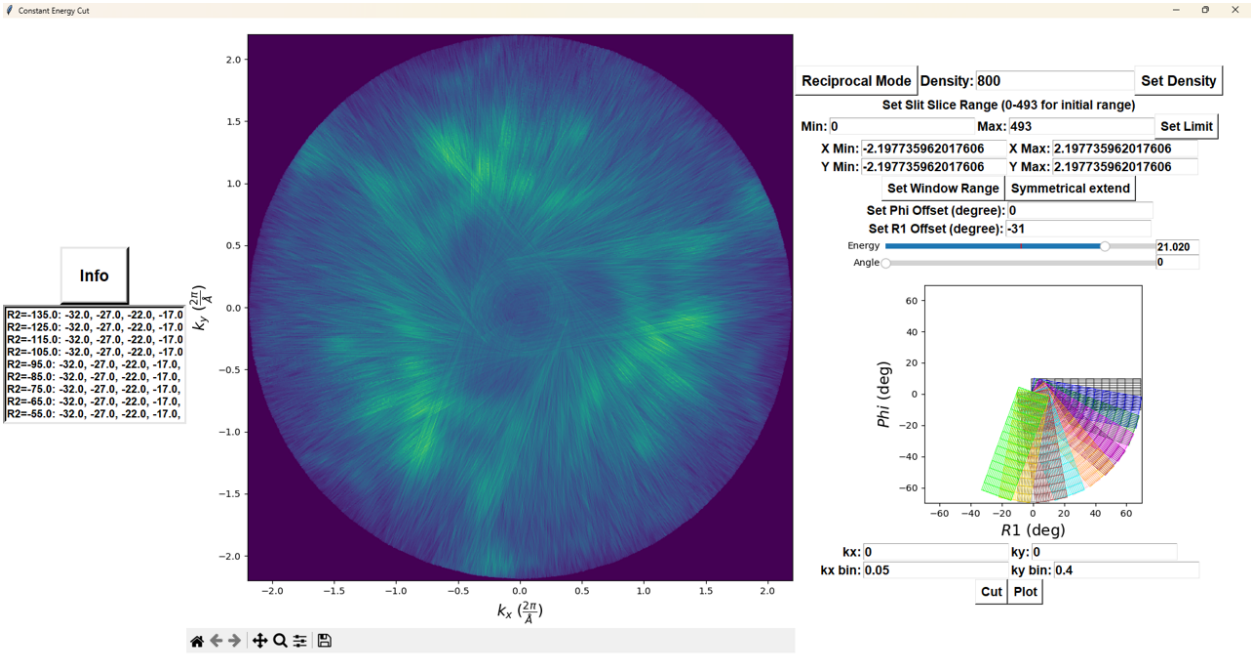


Figure 3.34: Constant energy map after symmetical extend

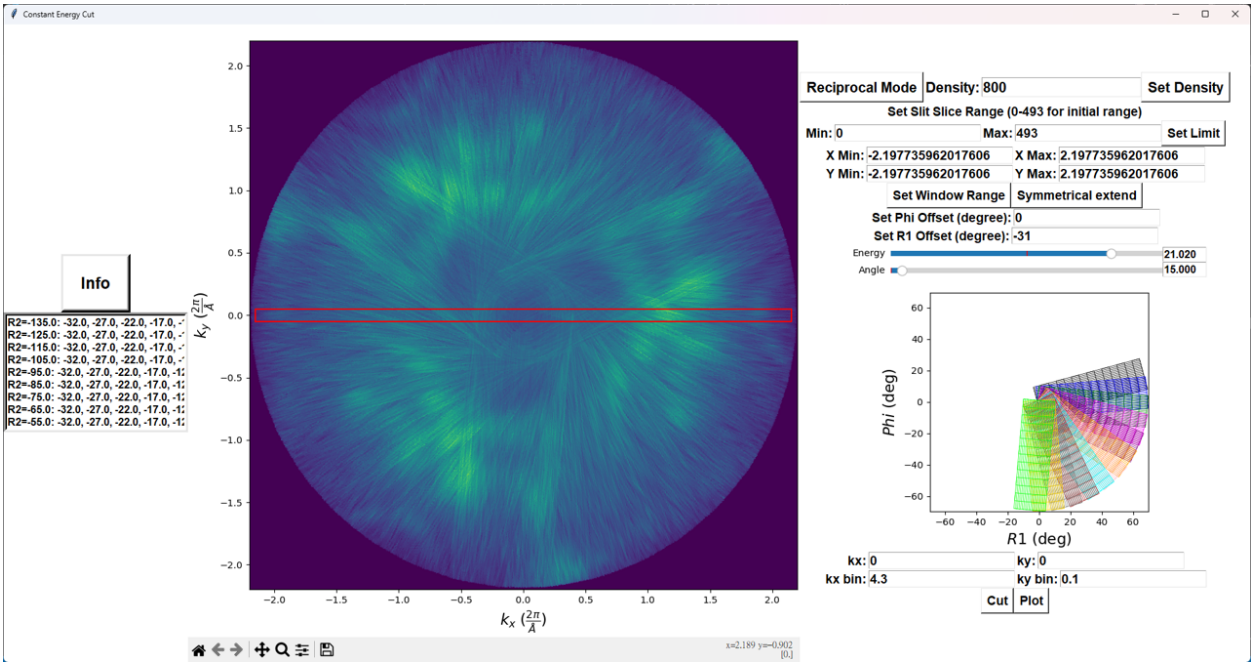


Figure 3.35: Example of Cutting Region

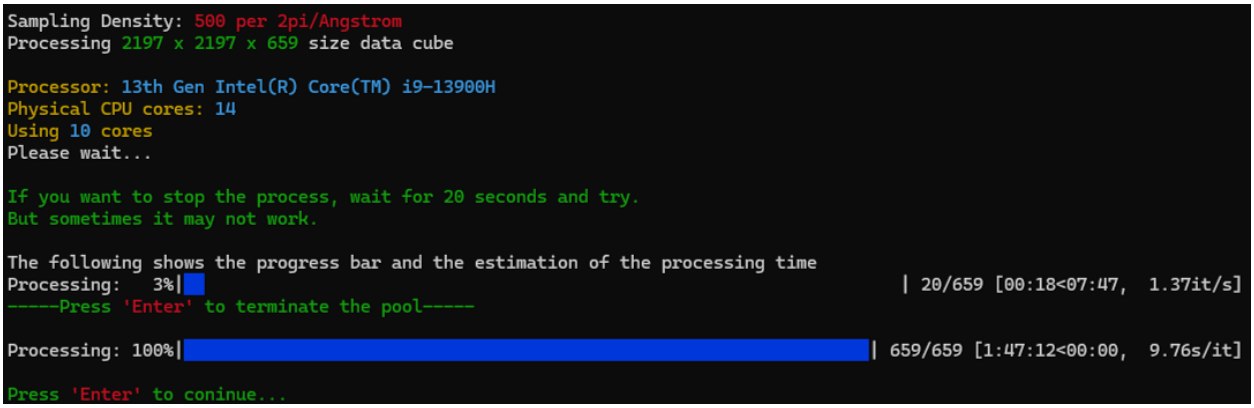


Figure 3.36: Progress shown in Command Window

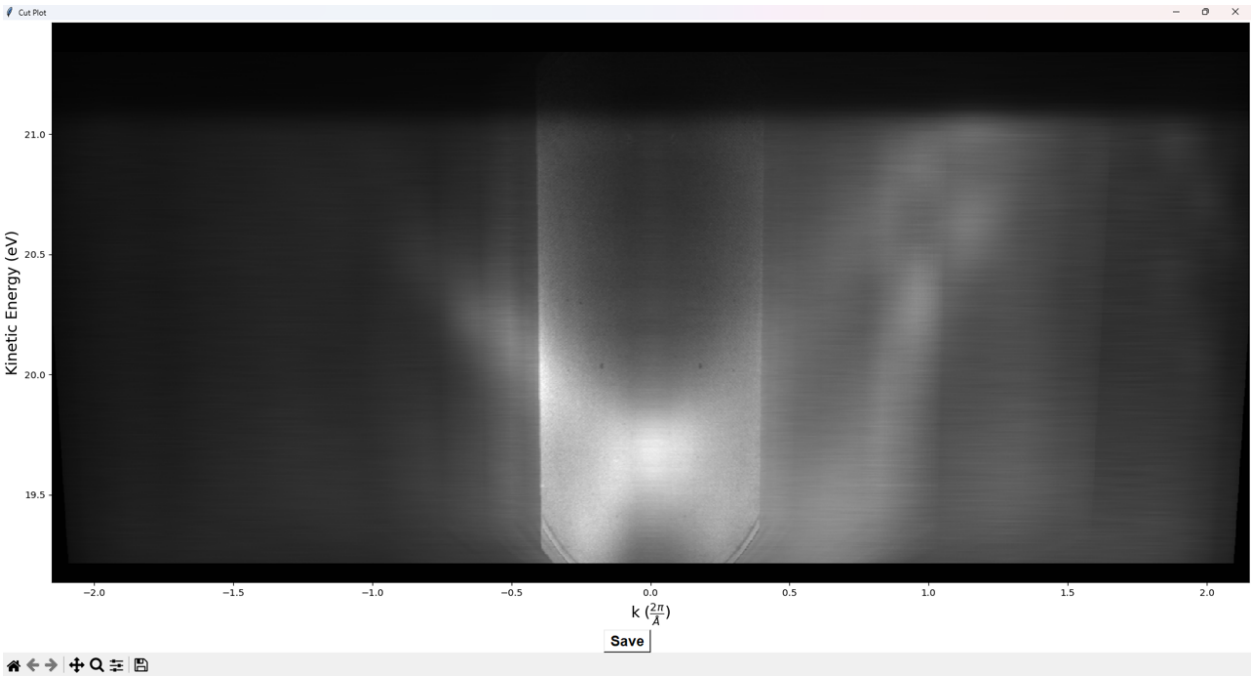


Figure 3.37: Sliced E-k Diagram

Chapter 4

Data Structure

The following sections describe the data structure required for the script to process the data correctly. The user can modify the script, referring to the following sections as needed.

4.1 Experimental Data

The experimental data is usually stored in H5/JSON files, which are generated by the **Spectrium** software. `MDC_cut.py` can handle the data as long as the correct format is used. `MDC_cut.py` defines three kind of loading data methods:

- `load_h5`
- `load_json`
- `load_txt`

User can modify them in the script to fit the desired data format. The **data** for `xr.DataArray` should be a 2-dimensional array with the shape of (**energy**, **angle**), and the coords should be defined as **'eV'** for energy and **'phi'** for angle.

```
data = xr.DataArray(data, coords={'eV': e, 'phi': a}, attrs=......)
```

The attribute structure is listed in Table 4.1. Note that the blue colored attributes are required to be well defined instead of "Unknown". Here is an example to define `load_txt` method:

```
def load_txt(path_to_file: str) -> xr.DataArray:    #for sklearn txt files
    name=path_to_file.split('/')[-1].removesuffix('.txt')
    e = np.linspace(21.2-1,21.2+0.2,659)    #fix BE 1~-0.2
    # e = np.linspace(21.2-2,21.2+1,284)     #scan BE 2~-1
    a = np.linspace(-10,10,494)     #A20
    description='SKNET'
    e_low = str(np.min(np.float64(e)))+ ' eV (K.E.)'
    e_high = str(np.max(np.float64(e)))+ ' eV (K.E.)'
    e_photon = 21.2
    #   attrs
    e_mode = 'Kinetic'
```

```
    LensMode = 'Unknown'
    PassEnergy = 'Unknown'
    Dwell = 'Unknown'
    CenterEnergy = str(np.average(np.float64(e)))+ ' eV (K.E.)'
    Iterations = 'Unknown'
    Step = abs(e[1]-e[0])
    Slit = 'Unknown'
    aq = 'SRNET'
    data = np.loadtxt(path_to_file).transpose()*100
    data = xr.DataArray(data, coords={'eV': e, 'phi': a},
                        attrs={'Name': name,
                            'Acquisition': aq,
                            'EnergyMode': e_mode,
                            'ExcitationEnergy': str(e_photon)+' eV',
                            'CenterEnergy': CenterEnergy,
                            'HighEnergy': e_high,
                            'LowEnergy': e_low,
                            'Step': str(Step)+' eV',
                            'LensMode': LensMode,
                            'PassEnergy': str(PassEnergy)+' eV',
                            'Slit': Slit,
                            'Dwell': Dwell,
                            'Iterations': Iterations,
                            'Description': description,
                            'Path': path_to_file
                            })
    return data
```

All attributes are required to be defined in **String** type, and the data structure should be consistent with the Table 4.1. Unknown attribute can be set to "Unknown" if the information is not available. Ensuring a consistent data structure is crucial for data processing by the script.

Attribute	Description	Example
Name	Name of the data file	Name/Unknown
Acquisition	Acquisition method	Scan/Fixed/Unknown
EnergyMode	Energy mode	Kinetic/Binding
ExcitationEnergy	Excitation energy	21.2 eV/3000 eV
CenterEnergy	Center energy	21.1 eV/Unknown
HighEnergy	Highest energy value	21.3 eV (K.E.)/Unknown
LowEnergy	Lowest energy value	20.7 eV (B.E.)/Unknown
Step	Energy step size	0.01 eV/0.02 eV/Unknown
LensMode	Lens mode	Transmission/Angular10/Unknown
PassEnergy	Pass energy	5 eV/20 eV/Unknown
Slit	Slit name	C-0.1x25/C-0.2x25/Unknown
Dwell	Dwell time	5.005 s/0.1 s/Unknown
Iterations	Number of iterations	1/5/Unknown
Description	Description of the data	desc..../''
Path	File path	C:/Users/Admin/Experiment/data.txt

Table 4.1: Experimental Data Structure

4.2 Fitted Data

4.2.1 NPZ File

The fitted data generated by the **MDC Fitter** can only be saved in NPZ format. It is a dictionary-like file format that can be read by `numpy`. Table 4.2 shows the description of the parameters stored in the NPZ file.

Key	Description
path	The path to the **Experimental Data** file.
fev	The fitted energy values in Kinetic Energy.
fwhm	The fitted Full Width at Half Maximum values in k space.
pos	The fitted peak position values in k space.
skmin	The minimum value of the fitted region in k space.
skmax	The maximum value of the fitted region in k space.
smaa1	The one-peak fitting parameters.
smaa2	The two-peak fitting parameters.
smfp	The peak number for each energy cut in the fitted region.
smfi	The index for fitted energy values.
smresult	The fitting results.
smcst	The fitting constraints.

Table 4.2: Description of NPZ fitted data parameters

k	**E**
-0.24	-0.5
-0.22	-0.4
-0.2	-0.3
-0.18	-0.2
-0.16	-0.1
-0.14	0
-0.12	0.1

Table 4.3: Example of Bare Band Data

4.2.2 VMS File

The VMS file should be generated by a certain procedure using **CasaXPS**. The details will be described in Section 5.3.

4.3 Bare Band Data

The Bare Band data is usually stored in **TXT** files, which encoded by `utf-8` and seperated by tab (`'\t'`). Table 4.3 shows the example of the Bare Band data structure. **k** is the k-value required in units of $\mathbf{2\pi\text{Å}^{-1}}$, and **E** is the energy value in units of **eV** defined by $E = E_k - E_F$, where E_k is the kinetic energy and E_F is the Fermi energy. Note that the **E** value below the Fermi level should be negative. The following is an simple example of the Bare Band data loading method:

```
k_values = [-0.24, -0.22, -0.2, -0.18, -0.16, -0.14, -0.12]
e_values = [-0.5, -0.4, -0.3, -0.2, -0.1, 0, 0.1]
with open('band_structure_data.txt', 'w') as f:
    f.write('k\tE\n')
    for i in range(len(k_values)):
        f.write(f'{k_values[i]}\t{e_values[i]}\n')
```

4.4 Custom Color Map

The user can define the custom color map in the script. The color map is defined by class:

```
matplotlib.colors.LinearSegmentedColormap
```

`MDC_cut.py` define the color map by a list of (x, (R, G, B)) tuples, where x is the position in the color map ranging from 0 to 1, and R, G, B are the red, green, and blue values. Here is an example to define the custom color map:

```
# Define your custom colors (as RGB tuples)
# (value,(color))
custom_colors = [(0, (1, 1, 1)),
                 (0.4, (0.3, 0, 0.3)),
                 (0.5, (0.3, 0, 0.6)),
                 (0.6, (0, 1, 1)),
                 (0.7, (0, 1, 0)),
                 (0.8, (1, 1, 0)),
                 (1, (1, 0, 0))]
# Create a custom colormap
custom_cmap = LinearSegmentedColormap.from_list('custom_cmap',
                                        custom_colors, N=256)
mpl.colormaps.register(custom_cmap)
```

After defining the custom color map, you can also add it to the color map list (`optionList3`) in the script. So that you can see it in the "Commonly Used" color map in the Control Panel.

```
optionList3 = ['terrain', 'viridis', 'turbo', 'custom_cmap',........]
```

Chapter 5

Data Processing Workflow

All the features in the script are designed to be used in a intuitive way. However, there are still some processes that are not so straightforward. The following sections present some potentially more complex workflows for data processing. Hopefully these workflows can help you process the data more efficiently.

5.1 Transmission Mode

The data measured in transmission mode is usually storing XPS or UPS data. The **Plotters** will be unavailable for these kinds of data, for the Transmission Mode data are usually used in 1-dimensional form (intensity - energy) and not suitable for the ARPES related data processing functions. However, the 2-dimensional raw data (spatial-resolved) still can be loaded by the **Data Importer**. The **Spectrogram** interface is mainly designed to visualize these kinds of data. The user can use **Spectrogram** to quickly view the data and do further analysis by exporting the data to TXT files. [Section 3.2] Also, if multiple data files are loaded, the user can use the **Batch Master** to process the data as well.

5.2 Angular Mode

There are two types of workflows after loading the Angular Mode experimental data:

1. Fitter → Load Fitted Data → Calibrate → Export
2. Cutter → CasaXPS Fitting → Load Fitted Data → Calibrate → Export

Workflow 1 works exclusively in the **MDC_cut.py** interface, while **Workflow 2** requires **CasaXPS** software to fit the data. The data measured in angular mode is usually storing ARPES data. All the features in the script are available for the Angular Mode data. To process the data, you can load the data by the **Data Importer**, briefly check the data by the **Plotter**, and decide how to deal with the fitting process. If your data is not too complicated, you can use the **MDC Fitter** to fit the data directly for the **MDC Fitter** provides only up to two peaks fitting. [Section 3.3] If your data is quite more complicated, you can use the **MDC/EDC Cutter** to cut the data and export the data to TXT files. Then you can use the **CasaXPS** software to fit the data and export the fitting results to the

VMS file. [Section 5.3] After the fitting process,you can load the fitted data by the **Data Importer**. Here you should take care of the angle misalignment of the experimental data, and the **Angle Setter** can help you determine the angle offset. Also, if you should have the bare band data, you can load it by the **Data Importer** and set the energy offset and the slope factor by the **Argument Setter** if needed. Then you can visualize the data by the **Plotter** and export the graph or export data to **OriginPro** for further analysis.

5.3 CasaXPS Fitting

After cutting the data by the **MDC/EDC Cutter**, The folder named **FileName_MDC_BaseCount** or **FileName_EDC_BaseCount** will be created in the same directory as the experimental data.

The folder will look like this:

```
|
|-PE10A20f-170_MDC_0
    |
    |-ecut_19.200.txt
    |-ecut_19.211.txt
    |-ecut_19.221.txt
    |-ecut_19.232.txt
    |-ecut_19.242.txt
    |-ecut_19.253.txt
    |-ecut_19.264.txt
    |-ecut_19.274.txt
    .
    .
    .
    .
    .
    .
```

or this:

```
|
|-PE10A20f-170_EDC_0
    |
    |-angcut_-10000.txt
    |-angcut_10000.txt
    |-angcut_-09959.txt
    |-angcut_09959.txt
    |-angcut_-09918.txt
    |-angcut_09918.txt
    |-angcut_-09878.txt
    |-angcut_09878.txt
    .
    .
    .
    .
    .
    .
```

To load all these TXT files into the **CasaXPS** software, you must follow the instructions below:

1. *Open **CasaXPS** software.*
2. *Click the* File *button in the top left corner.*
3. *Click the* Convert and Merge ... *button.*
4. *Selected all the TXT files in the folder and click* Open(O).

Then the **CasaXPS** software will automatically load all the data into the software. The tiles will show up as a sequence of the energy cut or angle cut. During the fitting process, the name of the component that you want to load back to the **MDC_cut.py** should be

named as "Survey". There shouldn't be more than two components named as "Survey" for the **Data Importer** can only load up to two components. After the fitting process, you can export the fitting results to the VMS file by the following procedure:

1. *Click the* File *button in the top left corner.*
2. *Click the* Save As... *button.*
3. *Select the* ***VMS*** *file format and save the file.*

Then the VMS file will contains all the fitting results of the data, and you can load the VMS file by the **Data Importer** in the **MDC_cut.py** script.

Contact

If you have any questions or suggestions about the script, please feel free to contact the author by email: **alex1010512@gmail.com**