Online Python Analysis in S3DF

Lauren F. Heald

Yusong Liu

2025/01

## Introduction:

This file outlines the usage of the gued python package written for processing gas phase electron diffraction images collected at the MeV-UED facility at the Stanford Linear Accelerator. The package is available in its entirety at <https://github.com/lheald2/gued>. To get the most up-to-date version, clone the package from github into your S3DF account (directions below). First, we will go through how to access S3DF. Then, we will cover the different functionality of the data processing package and how to most effectively using it during and after a beam time.

## Prerequisites:

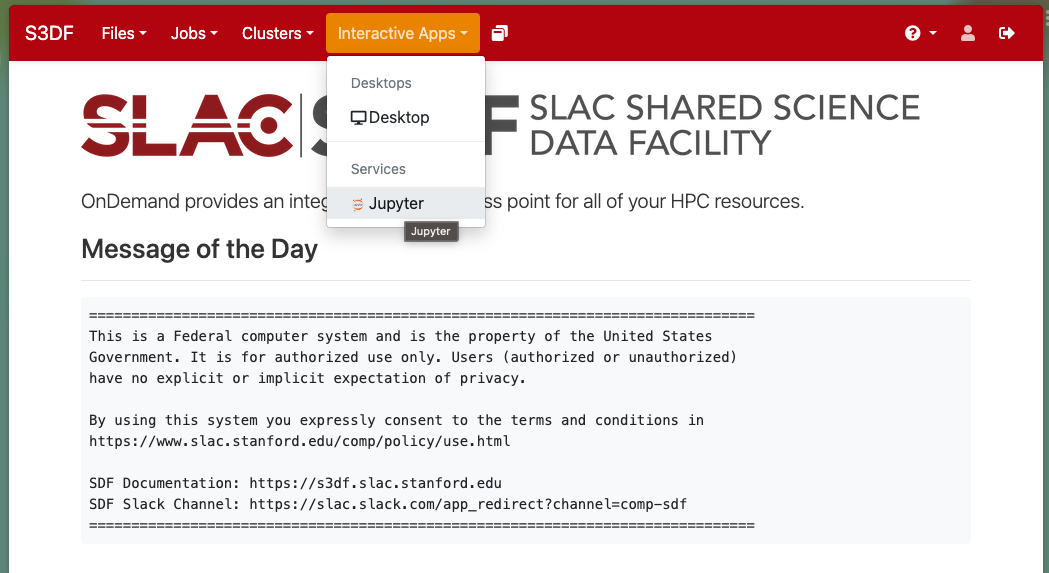
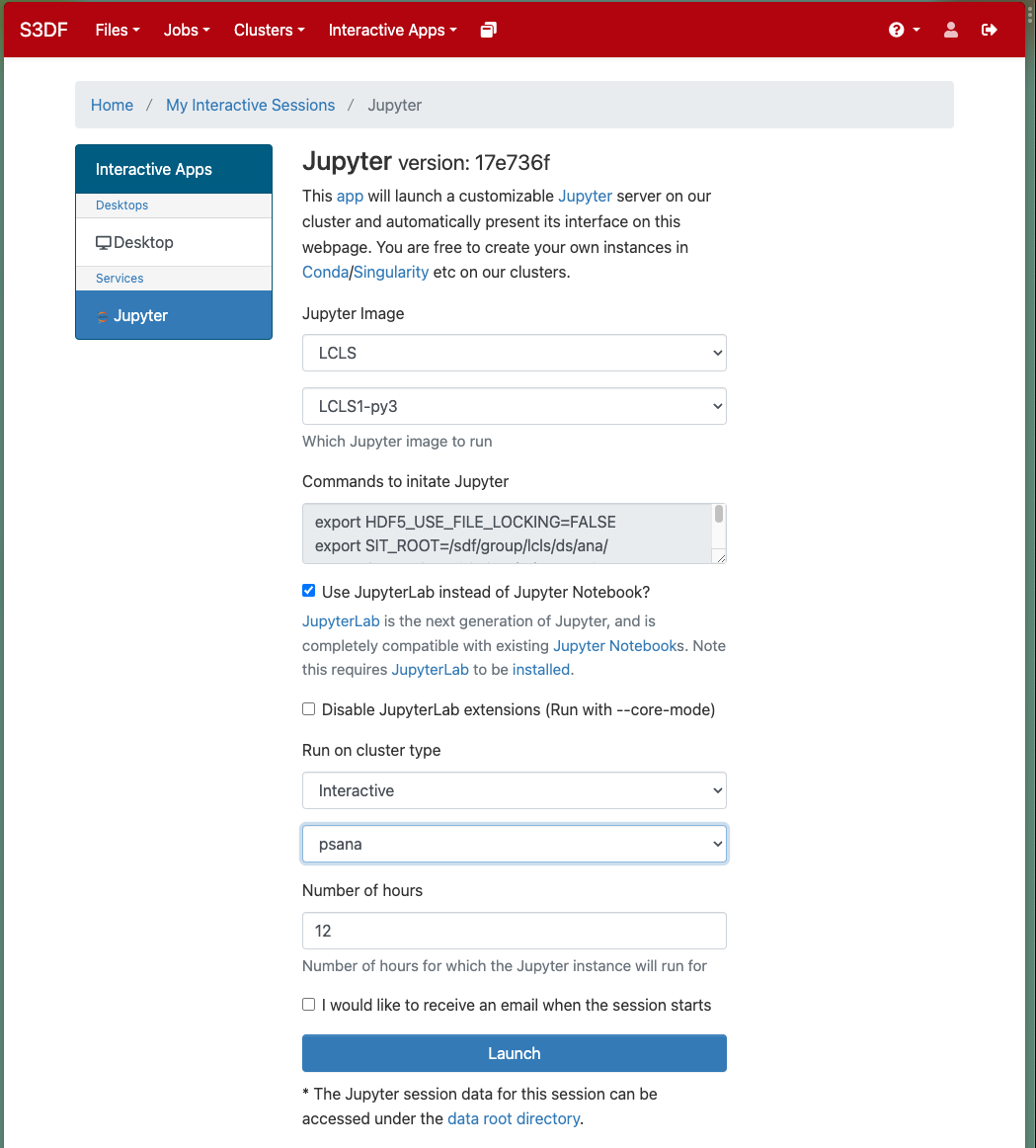
1. You need a valid SLAC **Unix** account
2. You need register an account in S3DF (SLAC Shared Scientific Data Facility)  
   If not, refer to confluence pages:

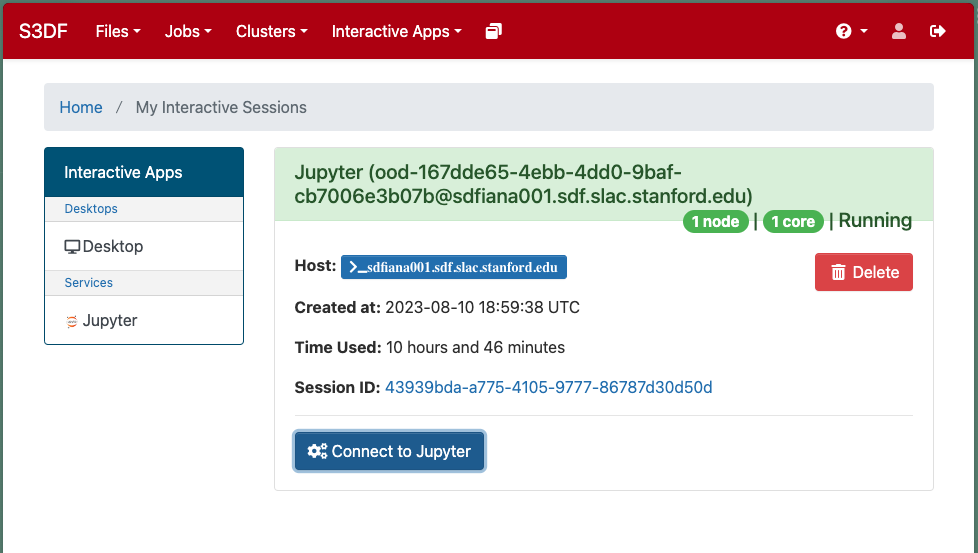
<https://s3df.slac.stanford.edu/public/doc/#/accounts-and-access?id=access>  
Below the confluence page introduces how to turn on a session in S3DF and open the Jupyter notebook and Jupyter lab:  
<https://confluence.slac.stanford.edu/display/PCDS/Running+at+S3DF#RunningatS3DF-Jupyter>

1. You have to be added in the **elog (**<https://pswww.slac.stanford.edu/uedlgbk/lgbk/experiments>**)** for your experiment in order to reach data.

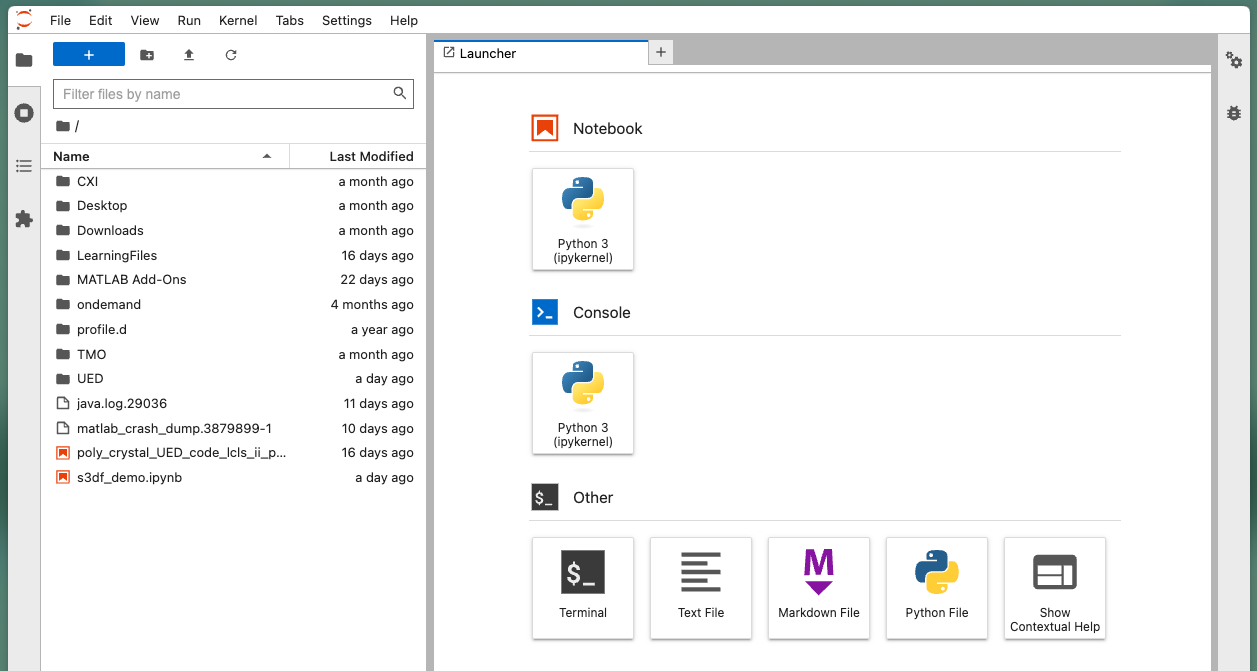
## Jupyter interface in S3DF

### Turn on the jupyter interface and access data and scripts

1. Go to OnDemand (<https://s3df.slac.stanford.edu/ondemand>) and log in with your Unix account (Unix ID) and credentials
2. Go to **Interactive Apps** and choose **Jupyter**
3. Set up jupyter conda and running clusters as below:
4. Choose the Jupyter Image of **LCLS1-py3 or LCLS2**
5. Either select **Jupyter Lab** or **Jupyter Notebook**
6. Choose **psana** cluster and then **Launch**
7. Click **Connect to Jupyter** to launch Jupyter interface,



And you will see the jupyter lab or jupyter notebook turned on. You will see the file structures at your home directory and the several options in Launcher.



## Importing the Data Analysis Package

From here, you can open new notebooks, consoles, and terminals etc. To import the most recent version of the data analysis code, open a terminal, navigate to the desired folder you wish to save the package in, and run the following:

$ git clone https://github.com/lheald2/gued.git

Once completed, you should see the following contents available in the newly created gued folder like so:

A screenshot of a computer

Description automatically generatedWithin the gued package you’ll find the base of the code is contained in the `.py` files while the interactive `.ipynb` files use the `.py` files for plotting and processing. Below, we will go through each Jupyter Notebook in the package and how it can and should be used during your beamtime.

While the package was designed such that edits to the .py files shouldn’t be required, I encourage you to look through them to get an idea of the complete functionality of the package. The functions related to processing images are contained in the `gued.py` file and functions related to simulating diffraction patterns are contained in the `gued\_theory.py` file. Other files like `process\_all.py` and `Process\_Trajectories.py` are meant for assistance with post-processing following the experiment. These will be briefly discussed below.

**Note that this package is user-developed and changes are made with each experiment as lessons are learned. If you feel you’ve relied heavily on this package, please cite it in future manuscripts as outlined in the README file. If you have issues, questions, suggestions, etc. feel free to email them to** [**lheald2@unl.edu**](mailto:lheald2@unl.edu) **with GUED in the subject line.**

## Summary of Notebooks and Functions

There are several notebooks (.ipynb files) and scripts (.py files) within the package that serve different purposes which will be outlined below.

### s\_Calibration.ipynb

This notebook is used when taking scattering patterns of Si single crystals for calculating the detector pixel to s calibration. The images are read in and an average image is calculated. First, the theoretical and experimental scattering patterns are plotted

A comparison of a graph

Description automatically generated

Then, an interactive plot is made of the experimental data and you are asked to click on the (2-20) peaks shown in the theory figure. Clicking on the image save the x and y information for the peaks which are then used to calibrate the detector.

A blue background with white dots and red dots

Description automatically generated 

### Set\_up\_Globals.ipynb

This notebook should be run before any image processing is done as the package relies on specific global variables that are stored within the package. Global variables are denoted with all caps and include the following:

A screenshot of a computer

Description automatically generated

This notebook loads in images from a specified directory and allows you to preprocess a small subset of the images to find the best values for the global variables.

* Each time you change a value in the global variables block of code, you need to restart the kernel and rerun the notebook.
* For most experiments, the main adjustments will relate to the masking and the center finding.
* MASK\_CENTER defines the mask location and MASK\_RADIUS defines the size of the circular mask being applied. Additional masks can be added using the ADDED\_MASK variable as a list of x, y, and radius for each added mask. The notebook allows you to visualize the mask location and should look something like this:

A graph with a purple square

Description automatically generated

* The THRESHOLD variable also may need to be adjusted. This value corresponds to how the center is found for each image. If THRESHOLD is set to 0, the algorithm will find it’s own threshold that it thinks is good. Often this results in inaccurate center finding. The image below shows an untrusty center finding threshold

A screenshot of a computer generated image

Description automatically generated

* In the above image, the threshold found a circle that isn’t much bigger than our mask, making it arguably unreliable. Instead, if I change the threshold to 2000, the radius of the circle it’s finding is much larger as shown below (maybe too big)

A yellow circle on a purple background

Description automatically generated

* For this particular example, the found center isn’t very different between the two thresholds. I recommend playing around with different values to find a sweet spot. Ideally, you can minimize the amount of shot to shot center drift by looking at a plot like this

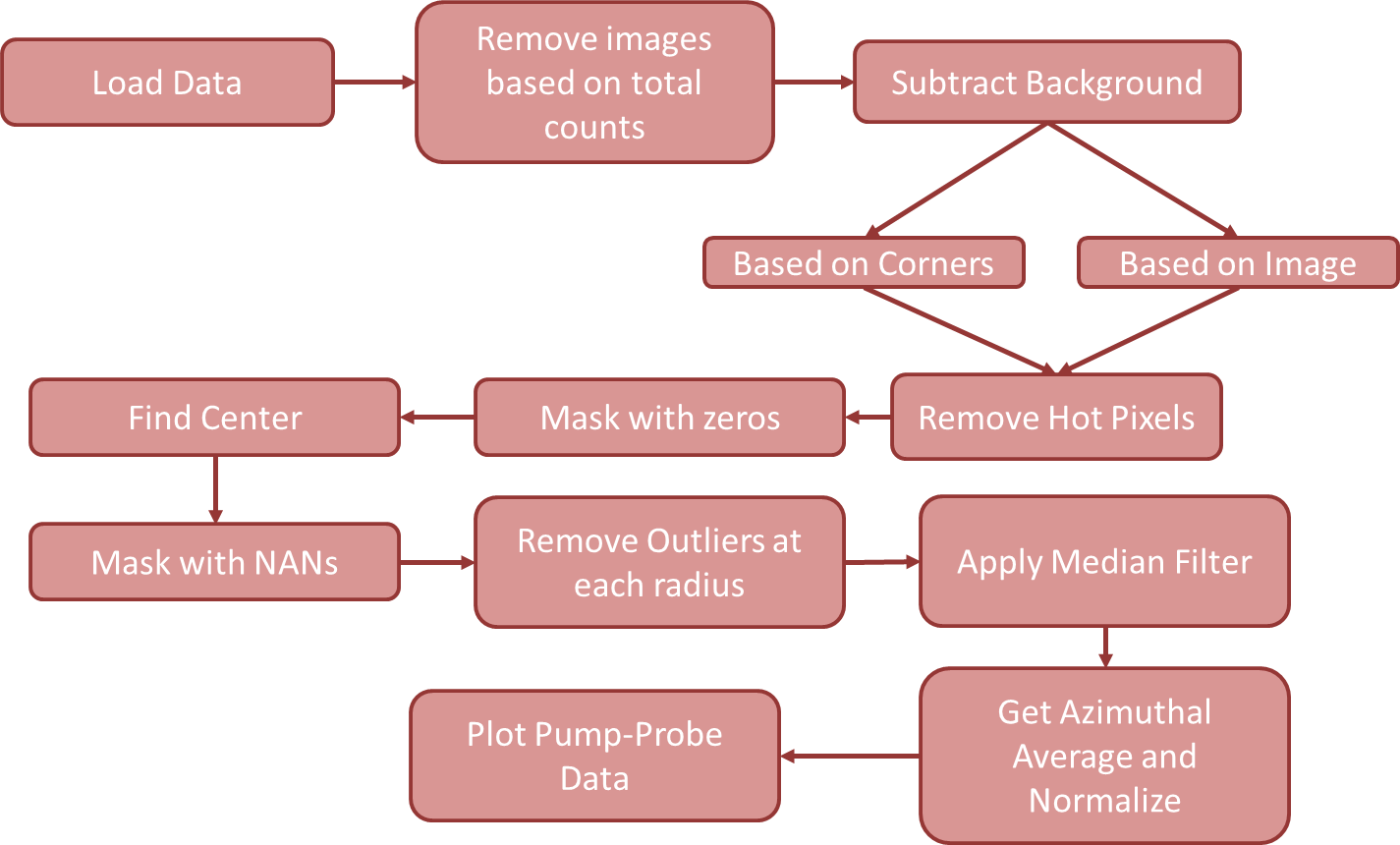
A graph of a graph

Description automatically generated with medium confidence

Once you have parameters set and can run the Set\_up\_Globals.ipynb notebook with a good mask and good center finding results, it’s time to move on to the Fast\_Analysis.ipynb notebook. **Remember to copy and paste the newly defined global variables into the fast analysis notebook.**

### Fast\_Analysis.ipynb

This notebook is the main data processing notebook to be used during beamtimes. The functionality of the code is outlined in the notebook itself, and a summary will be provided below. Ultimately, the goal of this notebook is to allow you to plot ΔI/I as a function of time. Steps are taken to remove noise within the data before the azimuthal average is applied. The basic functionality of the gued.py package being implemented in the Fast\_Analysis.ipynb notebook is outlined in the flow chart below



**A note about documentation and flexibility:**

A lot of work has gone into making the package well documented including docstrings which can be called using help(function\_name) as demonstrated in the notebooks. Additionally, most functions have an optional plot argument which allows you to display pre-written plots with each step (as is done in the Fast\_Analysis notebook). However, the plotting option can be set to False and you can craft your own plots as well.

Analysis Steps (Images based on data found in the folder located in the header of the document):

1. Import the data using gued.get\_images\_details. Loads in all the images that start with “ANDOR1\_” and plots the following information about the images

A diagram of a blue circle

Description automatically generated with medium confidence

A graph of a curve

Description automatically generated with medium confidence

1. Remove Images with bad total counts using gued.remove\_counts

A blue line graph with black lines

Description automatically generated

1. Average images based on stage position. Code block calculates and average image per stage position
2. Subtract background based on corners using gued.remove\_background\_pool or subtract a background image using gued.subtract\_background. The first function interpolates a background image based on the corners of the experimental data.

A graph of different colored squares

Description automatically generated with medium confidence

1. Remove hot pixels using the gued.remove\_xray\_pool function which removes pixel instances which lie outside specified standard deviation of the average image. Shouldn’t be removing more than 5% of original data

A graph and chart with a purple square

Description automatically generated with medium confidence

1. Mask data with 0 values using gued.apply\_mask with the fill\_value = 0.0 (required for center finding)

A chart with a purple square

Description automatically generated with medium confidence

1. Find center for all images using gued.find\_centers\_pool. Note that the x and y positions don’t drift too much from image to image, indicating effective center finding (most likely)

A graph of a variety of blue lines

Description automatically generated with medium confidence

1. Reapply mask with np.nan. Need to swap the 0 values in the mask for nan values to do further processing steps.
2. Remove outliers based on radial average using gued.remove\_radial\_outliers\_pool. Note this function can be run using an average center value for all images or using an array of unique center positions for each image. Again, the percentage removed should remain small and is based on the specified standard deviation

A close-up of a chart

Description automatically generated

1. Apply median filter using gued.median\_filter\_pool. This function first replaces any nan values created in the prior processing steps with an interpolated value based on the radial average of the image. Then a median filter is applied and the data is returned. Note that this function is still a work in progress and seems to cause issues in the low s regime. You can choose to bypass this step and skip to the azimuthal averaging.
2. Get azimuthal average and normalize data to range using gued.get\_azimuthal\_average\_pool which calculated the azimuthal average of the scattering images then normalizes them based on a specified range within the code (can be adjusted). The log intensity is plotted to evaluate linearity. Non-linear instances may indicate a bad center value (image below is not great). Additionally, can be useful to look for any time dependent changes in the normalization factor.

A graph of a graph with numbers and lines

Description automatically generated with medium confidence

1. Plot ΔI/I and time lineouts based on ranges of interest

A close-up of a graph

Description automatically generatedA graph with blue and orange lines

Description automatically generated

1. Optional: Apply a polynomial fit to the data to eliminate noise at high s values

A graph of blue lines

Description automatically generated with medium confidence

Finally, the data can be saved to an h5 file using the gued.save\_data function.

### Solid\_State\_t0.ipynb

This notebook is used specifically for finding t0 with Bi samples. It performs similar steps as the Fast\_Analysis.ipynb notebook, but focuses less on noise reduction and is used for finding the stage position where the onset of structural changes occur. Data is fit to an error function to establish t0. In the below figures, t0 ~ 37.5

A diagram of a graph

Description automatically generated with medium confidence

A graph of a graph of a person's peak

Description automatically generated with medium confidence

### GUED\_Simulations.ipynb

This notebook allows you to simulate diffraction patterns based on an input structure or structures such as those output by ORCA. This notebook is built using the gued\_theory.py file and you will likely have to update the PATH\_DCS variable in the gued\_globals.py file. My PATH\_DCS = ‘sdf/home/l/lheald2/GUED/gued/packages/dcs\_repositiory/3.7MeV/’

Functionality:

1. Simulate Static Scattering: Simulates an sM and PDF for the specified molecule

A graph of a graph of a function

Description automatically generated with medium confidence

1. Simulate Time Resolved Data (for trajectory simulations)

A graph with red and blue lines

Description automatically generated

1. Simulate Vibrations (from ORCA generated .hess file) \* Work in progress

A blue and red wave pattern

Description automatically generated with medium confidence

1. Simulate Fragmentation

A graph of different products

Description automatically generated with medium confidence

### PDF\_Generation.ipynb

This notebook allows you to take your experimental ΔI/I from the Fast\_Analysis notebook and convert into a time resolved Pair Distribution Function. This notebook relies on both the gued and gued\_theory packages and allows you to read in scattering data from an h5 file saved either with the Fast\_Analysis notebook or the output of the process\_all.py file.

### process\_all.py

This python file allows for fast bulk processing of individual images. However, some care should be taken to avoid overwhelming the systems memory. On my personal computer, I process < 2000 images at a time in groups of 200 images per group. This helps balance the amount of temporary memory being used during the processing steps. This file is not meant to be interactive and is meant to quickly batch process data following the experiment. There are several adjustable parameters that need to be changed before you run this file including the paths to the data and the path to where an output h5 file should be saved, the number of files to be processed (max\_idx and min\_idx) and the group size (how many to process at once). If I wanted to do the first 2000 images of a data set I would set the min\_idx to 0, max\_idx to 2000 and group\_size to 200. Then the parameter save\_factor works similar to a “run number” for each of the 200 image groups. For the first 2000 images I’d set save\_factor to 1 and for the next 2000 images I’d set save\_factor to 10.

A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

Parameters like exp\_label and group\_name allow you to add details to the h5 file regarding what the sample is etc. Finally, the smooth\_centers option is recommended and applies a moving average smoothing algorithm to the centers once they’ve been found using the center finding algorithm. The output h5 files are still quite large and I would recommend saving them to either a scratch directory on a computing cluster or a external personal hard drive.

Once these values are set, you can run the code in a terminal following

$ python process\_all.py

The processing took 15 minutes for 2000 images on the SDF server.

### Inspect\_Data.ipynb

This notebook is used only after the process\_all.py script has been run. This notebook looks at different parameters retrieved from the data processing to identify any changes, trends, or abnormalities in a data set. Below, the first 2000 images from the demo data set were processed. As can be seen, the center smoothing algorithm didn’t work very well for this run so changing the window size is something I’d consider in the future.

A screenshot of a graph

Description automatically generated

### T0\_Analysis.ipynb

This notebook is also best used after running process\_all.py. It allows you to take subsets of the data set and fit the rise time to look for drifts in t0 over lab time. Once your h5 file has been created, you can select different sets of images based on the 200 image grouping applied in the process\_all algorithm and evaluate the t0 for smaller sets of images.

A graph with a red line and black dots

Description automatically generated

This fitting algorithm is also incorporated into the Fast\_Analysis notebook for evaluating the rise time of a particular feature across all images.

### Tracking\_LabTime.ipynb

This notebook is a faster way to check for changes in the data with respect to lab time. Instead of sorting images based on stage position, they are grouped based on time of collection then the centers are found to see if there are any changes in beam steering over time.

A screenshot of a computer program

Description automatically generated

### Energy\_Converter.ipynb

This is a notebook for quickly converting units that often come up during ultrafast experiments (such as cm-1 to fs or mm of delay to fs).

### Process\_Trajectories.py

This was written for me to batch process trajectory xyz files provided by theory collaborators into simulated UED scattering data. Likely not useful for any experiment unless you have trajectory simulations to work with.

### all\_functions\_backup.py

This file contains all the archived functions that were once used in data processing for MeV-UED. It is there to be explored and used if there is something that isn’t in the gued main package. However, this is more of an archive and functions have not been tested or documented like the ones in gued.py and gued\_theory.py

## Final Comments:

This package was written by Lauren F. Heald, PhD. ([lheald2@unl.edu](mailto:lheald2@unl.edu)) in coordination with Yusong Liu and members of SLAC and the University of Nebraska – Lincoln. This package is meant to assist in the data analysis process and streamline experimental analysis. However, in no way is this package perfect nor does it contain everything needed to get publishable results. If you have any questions, comments, suggestions, etc. please email me (Lauren) direction with GUED in the subject line and I will do my best to address any issues in a timely fashion.