User Guide for the Focal Plane Detector Tilt Correction Scripts

Khang Pham

This collection of python scripts is aimed to find the tilt angle (relative to the wires), α, of the focal plane detector as well as the distance between the focal plane and any predetermined origin, H, to the method described in https://doi.org/10.1016/0029-554X(75)90121-4. The overall scheme of the scripts is to find the optimal H values (while the tilt angle is set parallel to the wires, i.e. $\alpha = 0.0$) of several peaks of the position spectrum such that the FWHM of each peak is smallest at its corresponding optimal H value. A line can then be drawn such that it instersects all optimal values of H, and the slope of this line will correspond to the correct value of α. The rest of this document is meant to describe the functionality of each script and the overall scheme of how the scripts are meant to be used together. At any point, you can use the "-h" option to see the arguments that each script takes.

In general, the order to use the scripts in are as follows:

1. Get x1-x2 coordinates (positions of incident particles when they cross the front and back wire, respectively) from the SPS-SABRE eventbuilder. This can be done by either printing out the coordinates, then grabbing them from the terminal, or adding in a way to save these values into a text file. Note that the coordinates should be extracted AFTER cuts are made. For now, let's call this text file "x1x2_withcuts.txt".

2. Use the calculate_xtilt.py script together x1x2_withcuts.txt as its argument to produce the focal plane positions (i.e. the points where the incident particles cross the focal plane).

   - This script takes multiple arguments: the text file of x1 and x2 coordinates, the start of H, the end of H, spacing between each H, and tilt angle.
   - For example, I used this script in this way: `python3 calculate_xtilt.py x1x2_after_cuts.txt 0.0 1.0 0.1 0.0`. This tells the script to start scanning H values from the back wire (0.0) to the front wire (1.0), using 0.1 increments of H, and setting α to 0.
   - Use plot_tilt_hists.py to view spectrum in the output files of xtilt.py.
     - This script takes an argument that controls the number of bins in the histogram. For example, I used this script like this: `python3 plot_tilt_hists.py tilt_files/* --bin 6000`
     - Every particle is incident upon the focal plane at an angle, and thus, we cannot expect the peak to be in the same place for each value of H. We must pick a region at each value of H in which we expect to see our peak of interest. For this we use createWindows.py.

3. Use the createWindows.py together with the output files of calculate_xtilt.py to calculate the boundaries of a specific region of the xavg spectrum.

   - Here, we will pick ranges for two different values of H in which we expect to find our peak of interest. You will first need to plot two specific H values using plot_tilt_hists.py. For example, if I want to plot the 0.2 file, I would use this: `python3 plot_tilt_hists.py tilt_files/0.2000H_0.0_degrees_xavg_tilt.txt --bin 6000`
   - You then zoom in on the peak of interest in your specific H value plot and determine a window in which you can see your peak. In my example, I picked the region 180 to 195, with a window size of 15.
   - Next, determine the window in your second specific H value plot, using the same window size, but not necessarily the same range of values. My boundaries for 0.7 value of H were 165 to 180.
   - For my example, I would use the script like this: `python3 createWindows.py 0.2 180 0.7 165 15`.
     - This calculates the window for values of H from 0.2 to 0.7. To calculate from 0.1 to 1.0, I would use: `python3 createWindows.py 0.2 180 0.7 165 15 0.1 --bs 0.0 1.0`.
   - Running this script gives a text file containing the boundaries for each H value.

If running these scripts for the first time, do the following two steps:

3a. Use the plot_specific_region.py script together with the output file of createWindows.py to confirm that the boundaries are correct.

- For my example, I use the script: python3 plot_specific_region .py boundariesFile_0.00_to_1.00_w_0.10spacing.txt −p tilt_files −s where "-p tilt_files" is an "optional" argument which is required to tell the script where your tilt files are located unless you are currently working in the same location and "-s" saves the output of the script.
- In this plot, your peak should be aligned vertically. If it is not, choose values of H that are further apart when using createWindows.py. Alternatively, you can use the same values of H but make sure that the center of your peak aligns with the center of the window that you have picked.

3b. Use the get_guess_pars.py script together with the output file of plot_specific_region.py to confirm that peaks in specific regions are always recognized. Most likely, one would need to adjust parameters in the script in order for the peaks to be reliably recognized every time.

- There should be a red mark on each of your H value peaks. If some are missing the red mark, change the prominence, height, and width parameters of the peak in the get_guess_pars.py script. You may have to change one or all of these parameters. You do not have to change all three of them.

If everything is optimized in plot_specific_region.py and get_guess_pars.py, skip the above two steps and go straight to step 4:

4. Use the fit_peaks.py script for the final results.

- This script requires the output from createWindows.py and the number of peaks expected in this region. Again, you must include the path to your tilt files as well.
- For my example, I used: python3 fit_peaks.py boundariesFile_0.00_to_1.00_w_0.10_spacing.txt 1 −p tilt_files
- This script creates a .csv output file containing the values of H, the amplitude of each peak, the centroid, the sigma of each peak, and the reduced chi squared value for the fit.

Here is the link to a video that demonstrates the usage of these scripts: `https://youtu.be/nnAzjDd-1k8`