BACoN analysis RUN 4

Carmen Romo Luque

June 2, 2025

Contents

1 Before data analysis

To analyze the data it is better to use the NERSC cluster. I always copy each BACoN file as the data is collected. I created time ago a folder in NERSC to host the BACoN root files and for the new data taking period we can use the directory: `/pscratch/sd/r/romo/bacon_data/run4/`. To copy the files you can do in the UNM machine:

```
scp bacon_files user@perlmutter.nersc.gov:/pscratch/sd/r/romo/bacon_data/run4/
```

It usually takes around 30 seconds to copy each file.

2 How to run my scripts

Important!! My scripts are made to analyze one file at a time, but with this script (*BACoN_signal_processing_all_files.py*), you can call all the files with same initial name and execute any desired script from my repo. For example:

```
$ python3 BACoN_signal_processing_all_files.py
 Run4_code/BACoN_signal_processing_hits_and_times_run4.py
 /pscratch/sd/r/romo/bacon_data/run3/ run-11_11_2024
  /pscratch/sd/r/romo/bacon_data_npz/run3/
```

Here we would been executing the script `BACoN_signal_processing_hits_and_times_run4.py` located in the directory `Run4_code`, we would be pointing to the folder where the files of run3 are, would be calling all the files starting with `run-11_11_2024`, so

```
run-11_11_2024-file.root
run-11_11_2024-file_1.root
run-11_11_2024-file_2.root ...
```

and the last path corresponds to the directory where we want the output files.

With these scripts you could send jobs to the NERSC queue or execute the scripts directly (what I usually do is to open some `screen` terminals and run the script for all files of October, or November, or whenever).

3 First look at the data and initial checks

I recommend to first take a look at the data and do some initial checks before analyzing all files. For this I like jupyter notebooks (you can use the tools in NERSC or copy a couple of files to your laptop and use jupyter there, this is the option I prefer because it is faster than NERSC and I don't need connection every time). The example notebook I sent some months ago can be found here.

Some initial checks I use to do:

1. Check that all channels (trigger and non-trigger SiPMs and PMT) have waveforms (in the past some channels eventually stopped working).
2. Plot some waveforms for each channel (SiPMs and PMT).
3. Try to identify baseline events, events with light, background events, events that saturate...
4. Plot sum of waveforms for each individual channel.

4 Fast analysis using my code

The main script is `BACoN_signal_processing_hits_and_times_run4.py`. There I extract the indices, heights and integrals of the peaks from the photons detected by the SiPMs (trigger and non-trigger).

Then, an *.npz* file is created with this information from each file of data and I analyze all the files, concatenate theis results and make the plots with the notebook `Analyze_BACoN_hits.ipynb`. There I plot the hit maps using the height or the integral vs the timestamp of each peak, and from the sum of the amplitudes (generally I use the heights of the peaks) for each individual timestamp, I obtain the time distribution for each channel. From a linear fit of the time distribution selecting the range of the triplet, I get the decay constant (everything is in that jupyter notebook). There there is also the fit for the PMT data (although some cuts need to be performed, right now the time distribution is very ugly). The script to do the analysis of the PMT data is $BACoN\_pmt\_analysis\_peaks.py$, which can be run in the same way the script `BACoN_signal_processing_hits_and_times_run4.py`.

5 Deeper analysis steps

Once you have made sure that the data looks ok, you should go deeper and continue with:

1. Now that we have LEDs installed inside BACoN it would be great to calibrate the channels using them.
2. Check the standard deviation of the waveforms for each channel (it gives an idea of the baseline stability). Check the standard deviation threshold per channel (it allows to reject baseline waveforms, which is important to make code faster because we don't iterate over the waveforms we don't need). Be careful because the values depend on the bias voltage applied to the channels. Typical values I used for the non-trigger channels are: 13-14 ADC, for trigger channels: 30-40 ADC and for the PMT: 4 ADC. But then to reject baseline events I multiplied the standard deviation by 3.
3. Check trigger time in the trigger SiPMs and confirm pretrigger region for baseline computation (0-650 timestamps).
4. Compare baseline in the whole waveform and at the beginning and end of waveform for a few files (you can use this script $BACoN\_data\_baseline.py$). I'd do it for one file at the beginning of the run and one at the end maybe, the script takes too long.
5. Check that the subtracted waveforms are centered at 0.
6. Check parameters of Savitzki-Golay filter ($window\_len = 30$ and $polyorder = 3$).
7. Check height, integral and shape of the single photoelectron (and 2PE and 3PE) for each channel of the SiPMs and the PMT.
8. Check the $min\_dist = 50$ ns (25 time samples) of the peak finder algorithm and threshold (80 ADC for normal chs and 200 ADC for trigger chs) for the zero zuppression.
9. Compute heights (from baseline, no deconvolution) and integral of peaks. Get the times of the maximum, not the threshold (it affects the singlet and the region afterwards). ($BACoN\_signal\_processing\_hits\_and\_times\_run4.py$).
10. Understand required cuts: rejection of events before the trigger region? rejection of high light events? Selection of 1PE events as in LLAMA?

11. Calibration of all channels (SiPMs and PMT) using the heights and the integrals to transform from ADC to PE.

12. Compute time distributions and make fit.

13. Compute the sum of detected PE for the 3 trigger SiPMs. We should be able to see the 60 keV gamma peak.

14. Check pile-up events.

15. Check PMT data analysis.