Hit finding and pile-up tagging:
Establishing a benchmark

James Vincent Mead

Ongoing tasks

light	L. Mora-Lepin, J. Yang, L. Calivers, M. van Nuland	Light deconvolution / hit finder
charge	B. Russell, russell3@mit.edu	data volume assessment
charge		microphonics
charge		bit flips
charge		whole- and partial-tile triggers
charge	E. Hinkle, ehinkle@uchicago.edu	ADC saturation in beam events
charge	Z. Wu, zhongyw8@uci.edu	hot pixels
light	A. White, ajwhite@uchicago.edu	ground bounce study
light	J. Mead, jmead@nikhef.nl	light saturation study
light		baseline stability study
charge		electronics response
light		dark count study
charge	R. Mandujano, rcmanduj@uci.edu	E-board triggers
charge		Diffuse single hit events
light	J. Mead, jmead@nikhef.nl	cross talk study

Tasks

- Debugging Deconv inputs (L. Caliviers)
- New algorithms and filters (J. Yang)
- Assessing hit-finder without Deconv (I tried but poor results)

Implement own hit finder

- Time over threshold (ToT)
  - Identifies hits
  - Pile-up suppression
  - More needed to tag interactions
- Scipy's peak finder
  - ndlar-flow currently uses on deconv wvfms
  - 'height' gives ToT windows
  - 'threshold' compares to adjacent bins
- Manual adaptive threshold (asymmetric)
  - Explored diff wvfm+sqrt(N)
  - Explored explored rolling average+sqrt(N)

Hit-finder for single channels

- ***File being used on subsequent slides:***

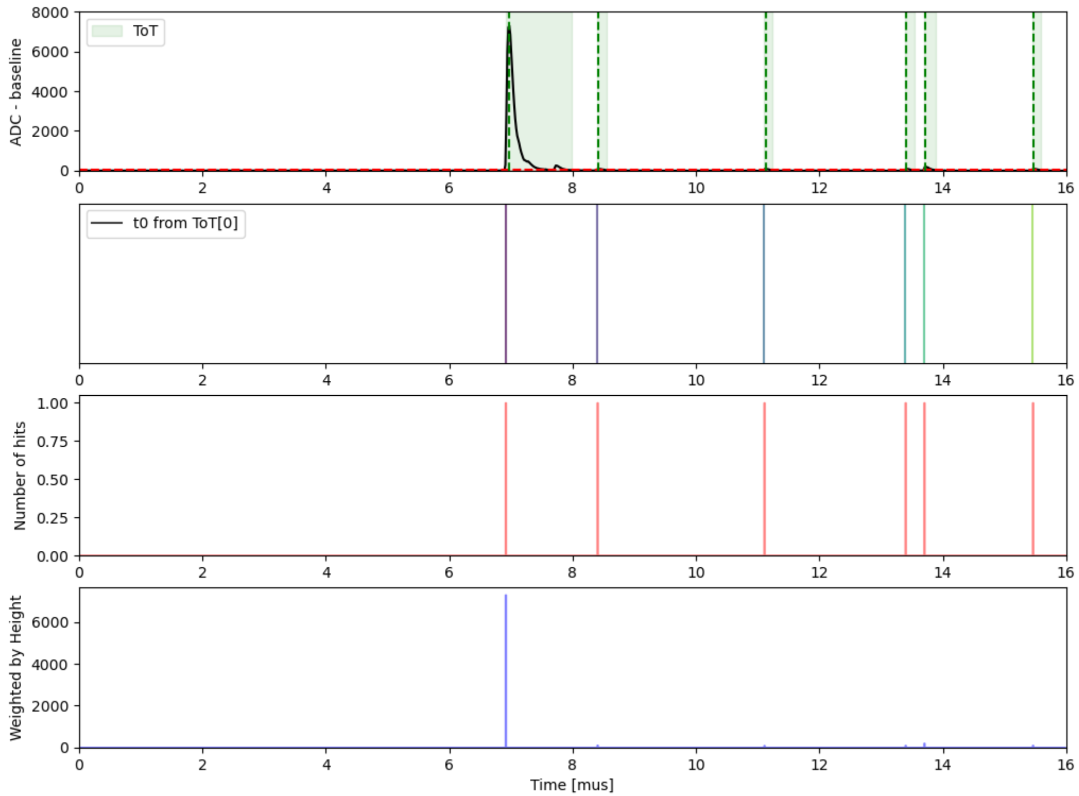
2x2 data:

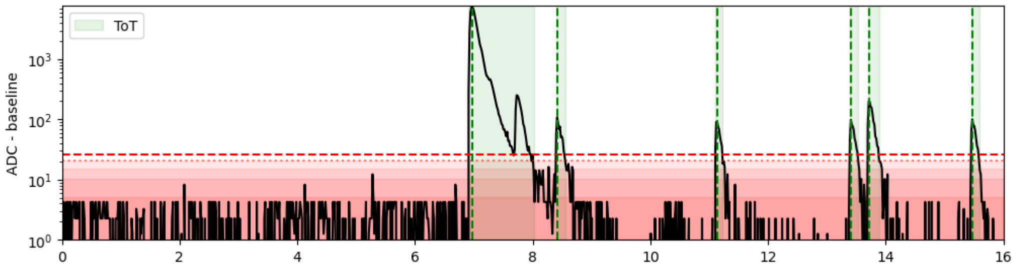
- File: mpd_run_hvramp_rctl_105_p350.FLOW.hdf5
- Geom: light_module_desc-5.0.0.yaml

2x2 MC:

- File: MiniRun5_1e19_RHC.flow.0000000.FLOW.hdf5
- Geom: light_module_desc-4.0.0.yaml

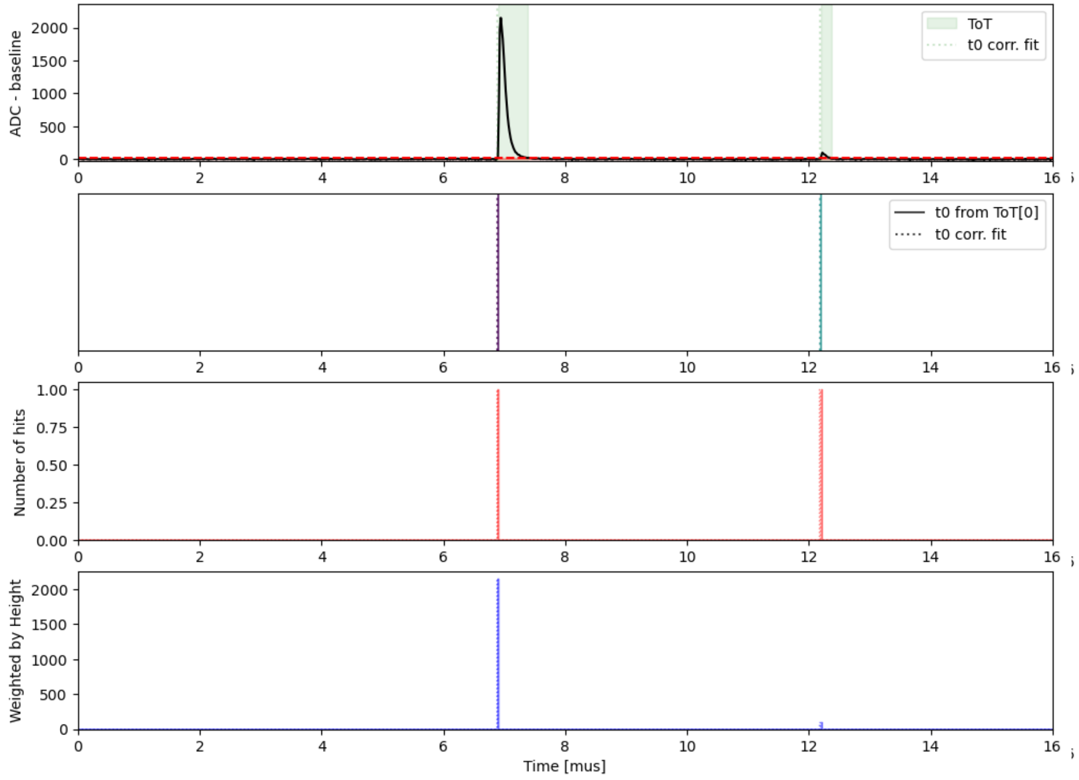
- ***Steps***
  - Threshold = {> **5** * **noise-width**}
  - Under-threshold = {< Threshold – sqrt(Threshold * **w**^2}
  - Finds max value in ToT
  - For each hit, saves:
    - Default: index_0, index_f, index_max, height_max, integral_tot
    - Optionally: t_0, t_f, t_0_corr

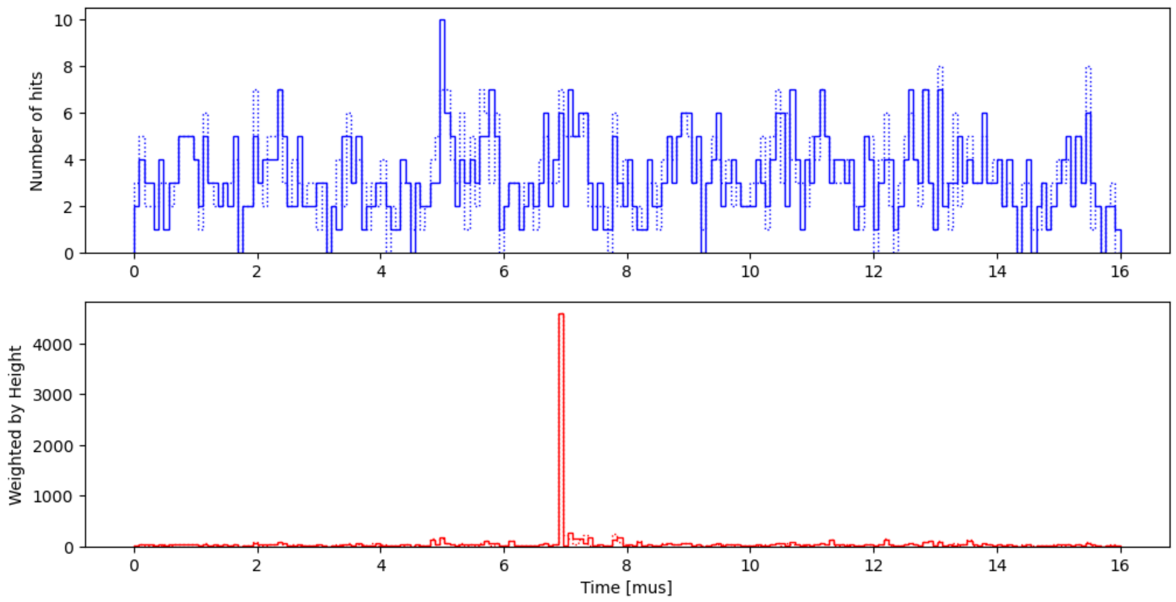




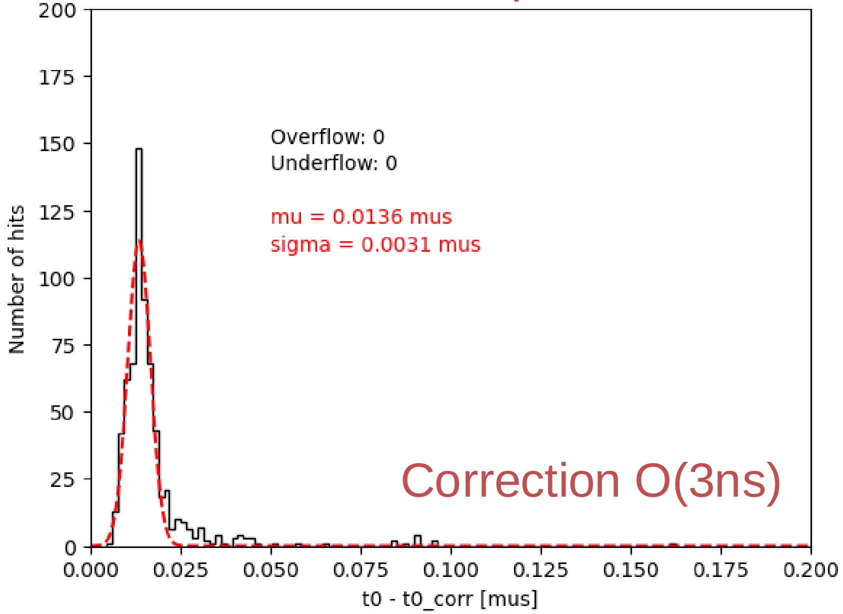
- ***Steps***
  - Threshold = {> 5 * noise-width}
  - Under-threshold = {< Threshold – sqrt(Threshold * w^2}
  - Finds max value in ToT
  - For each hit, saves:
    - Default: index_0, index_f, index_max, height_max, integral_tot
    - **Optionally:** t_0, t_f, **t_0_corr**

Can include linear fit used to extrapolate to baseline intersect





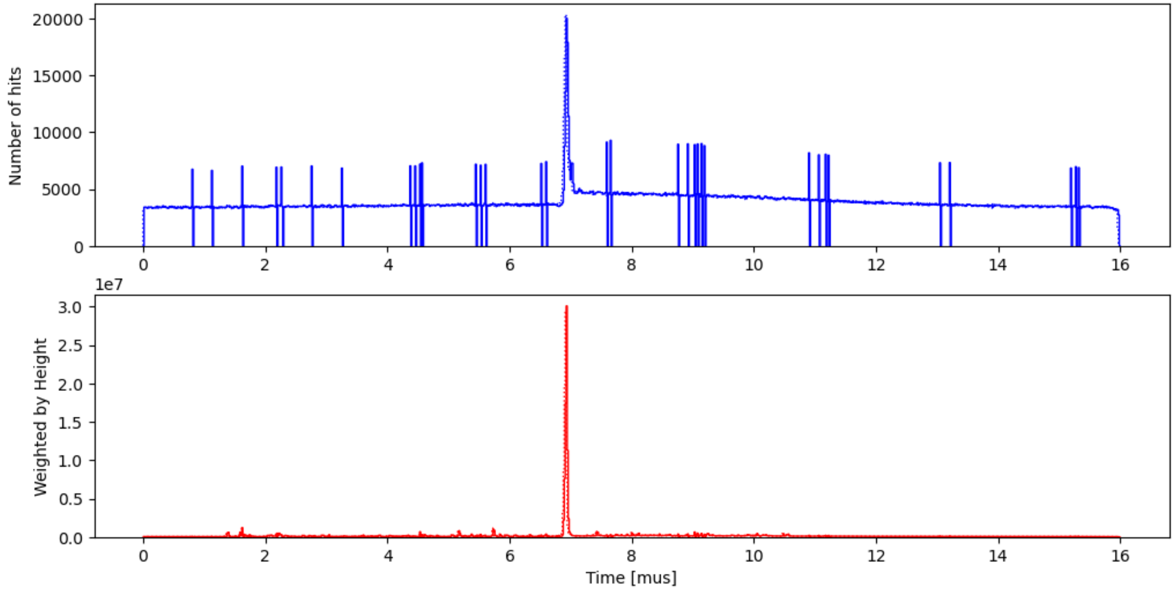
WIP: compare to true t0



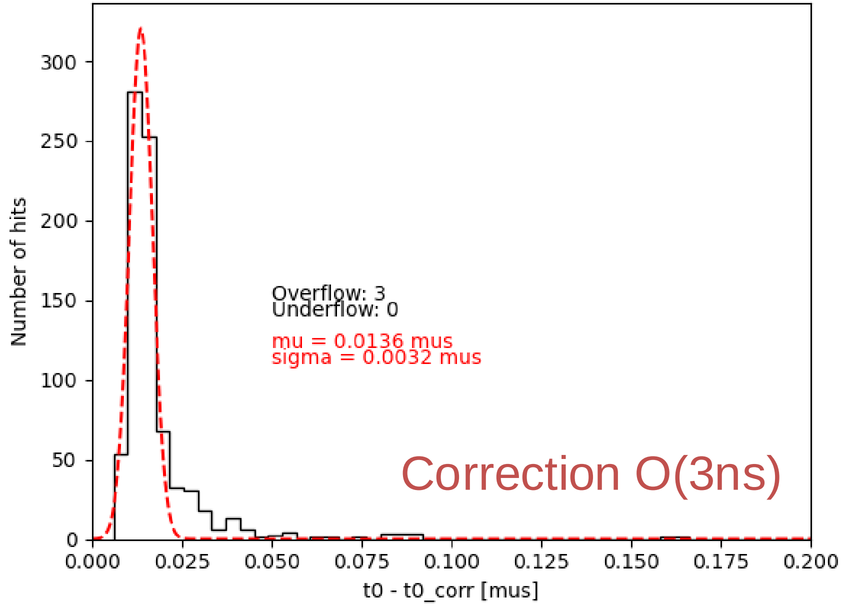
Correction O(3ns)

All channels in one event

Linear fit baseline intersect



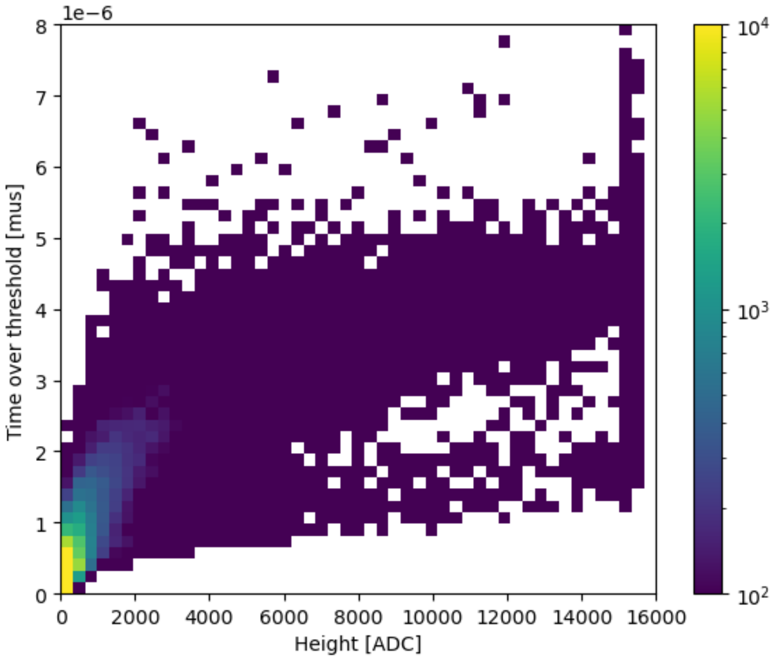
WIP: compare to true t0

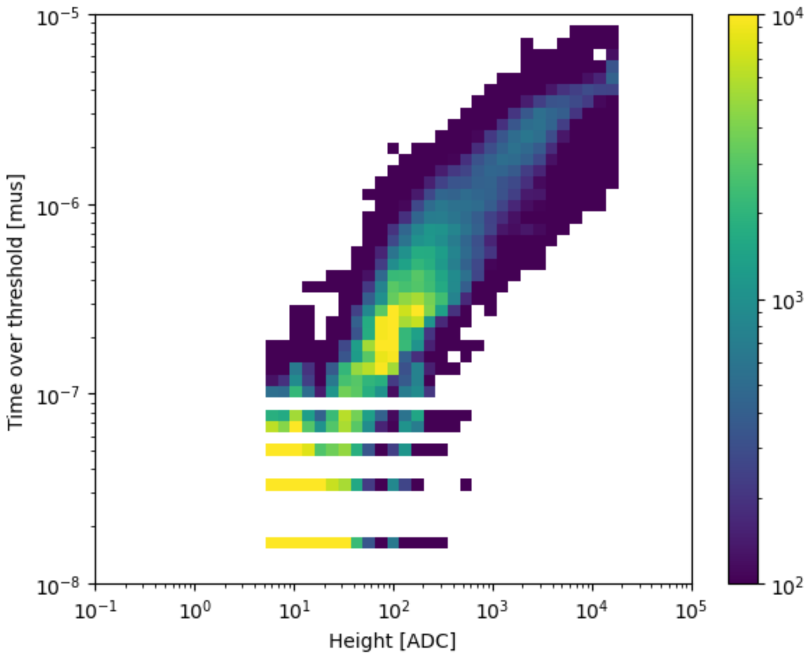


Correction O(3ns)

All events in one file

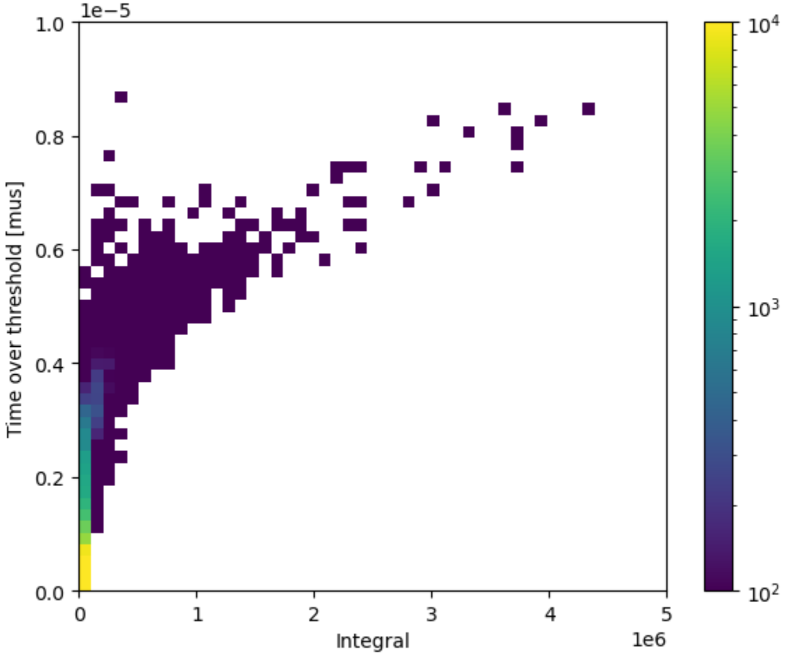
Fprompt proxy from ToT and Height

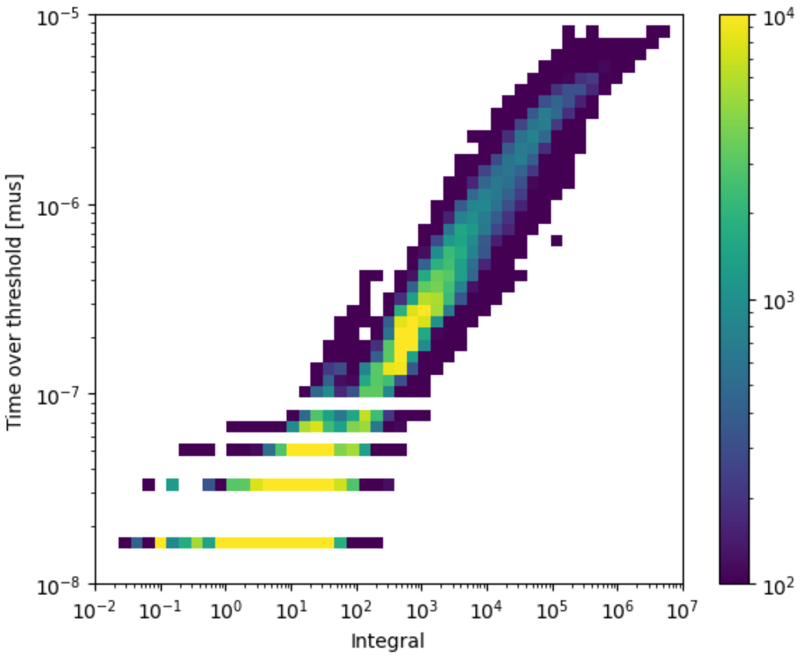




All events in one file

Integral as proxy for energy

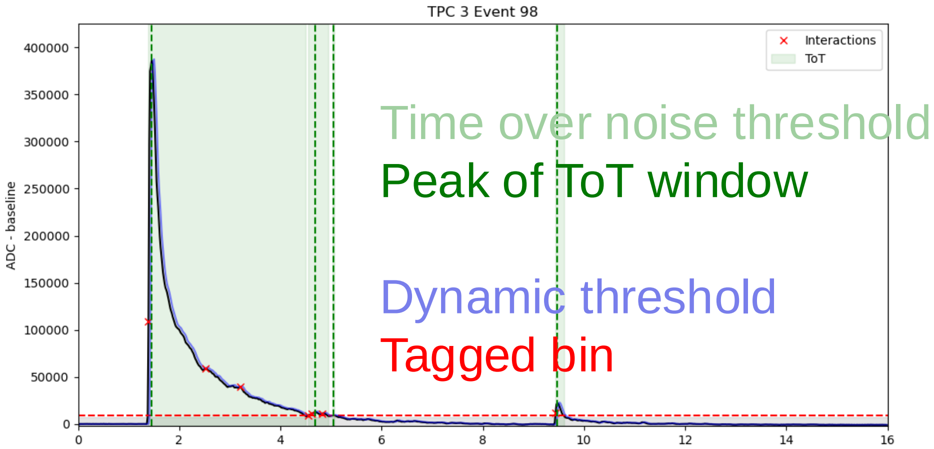


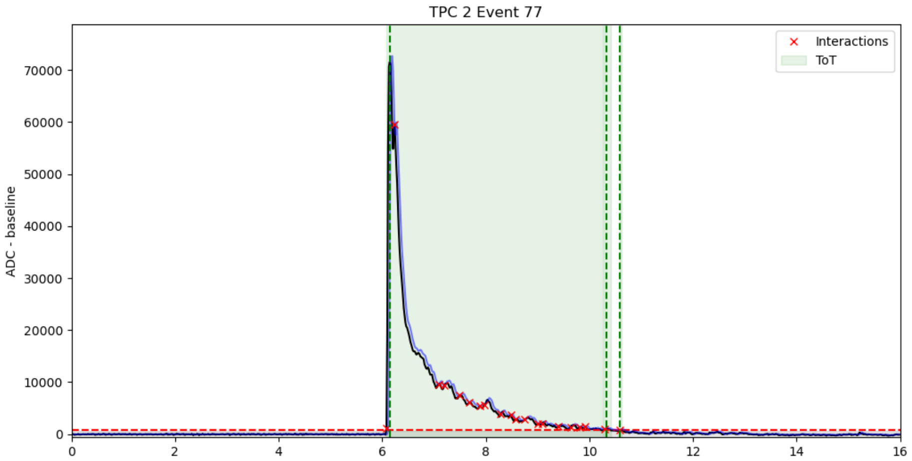


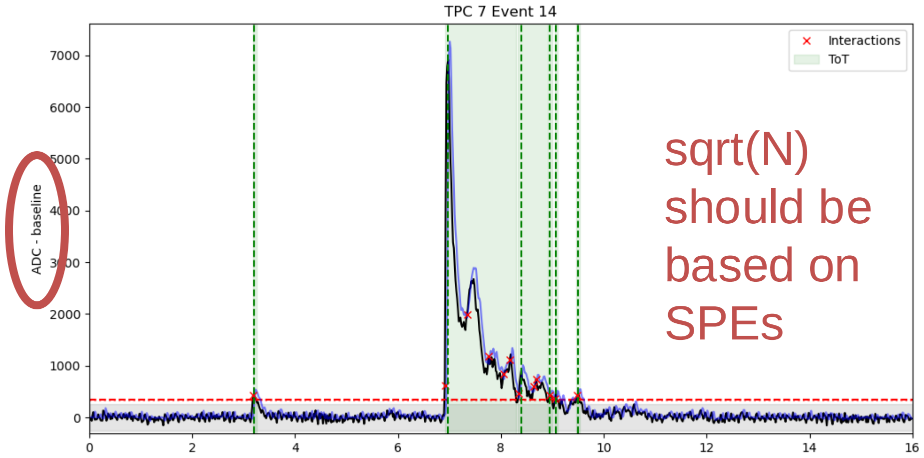
All events in one file

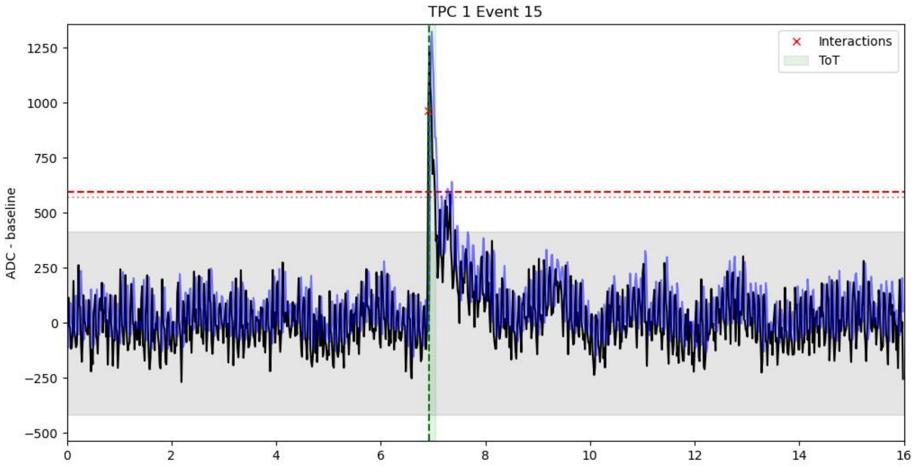
- ***Several options being explored***
  - Running \`scipy.find_peaks\`, parameters below:
    - 'Height' set to noise threshold – various methods:
      - First 50 ticks mean *– previous standard approach*
      - Gaussian fit *– presented previously*
      - MAD-masked StdDev *– discussed previously*
    - 'Threshold' set to dynamic threshold calculated from waveform + sqrt(N) – various methods:
      - Rolling average of previous n-bins + sqrt(N)
      - Where n=1, equivalent to np.diff() approach
    - 'Distance' minimum number of bins between tagged peaks (maximum is 6 ticks for < 100ns)

  - Asymmetric equivalent to \`scipy.find_peaks\`:
    - In scipy, 'Threshold' is applied symmetrically to adjacent bins, alternate only applied to previous

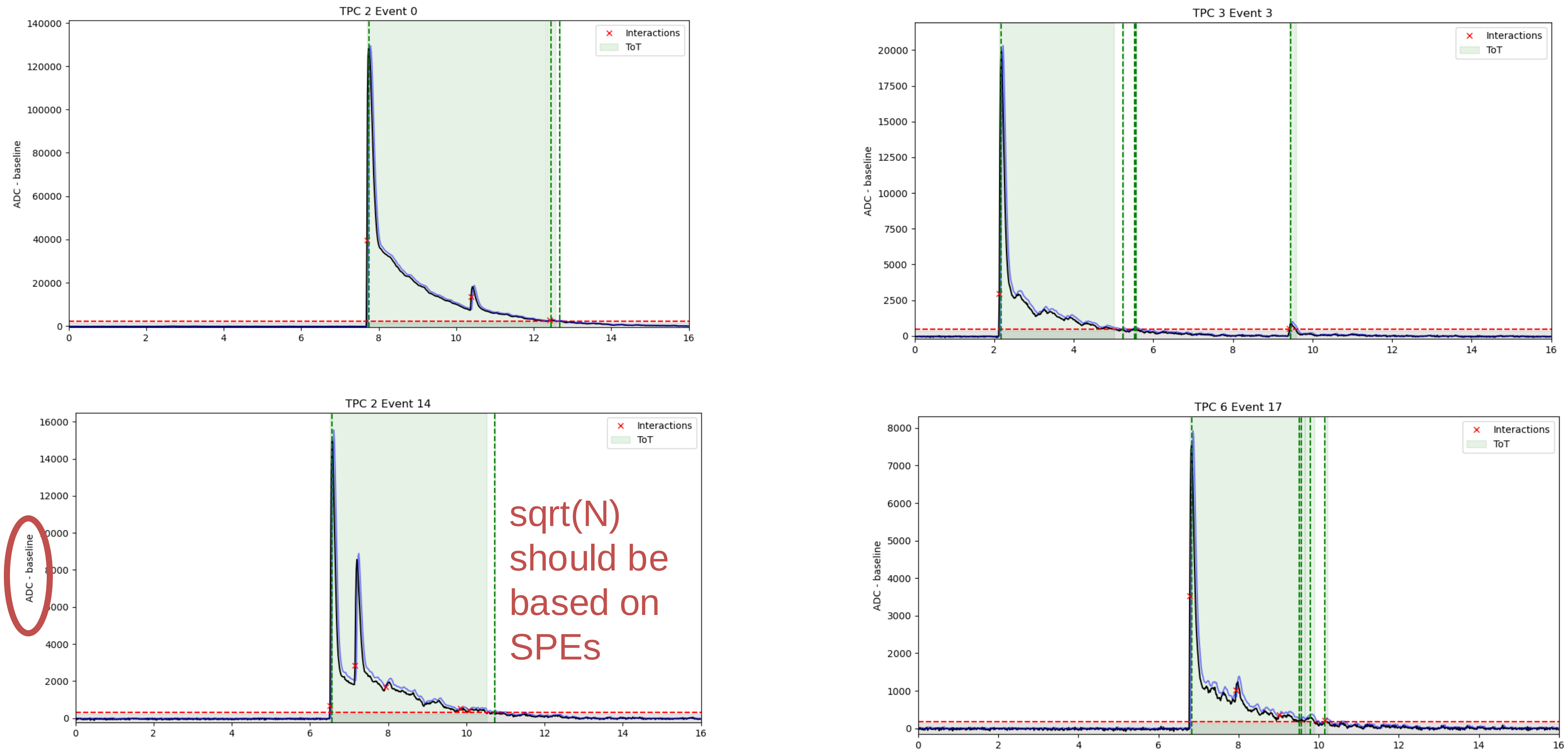

TPC 3 Event 98
Interactions
ToT
ADC - baseline
Time over noise threshold
Peak of ToT window
Dynamic threshold
Tagged bin


TPC 2 Event 77
Interactions
ToT
ADC - baseline


TPC 7 Event 14
Interactions
ToT
ADC - baseline
sqrt(N)
should be
based on
SPEs


TPC 1 Event 15
Interactions
ToT
ADC - baseline

Pulses in MC: summed per TPC



sqrt(N) should be based on SPEs

Fast interaction finder

- ***Assessing asymmetric equivalent to `scipy.find_peaks`***
  - Removed hit-finder for expediency
  - Running over TPC+TrapType waveforms
  - Baselining and noise floor calculation per summed waveform for expediency

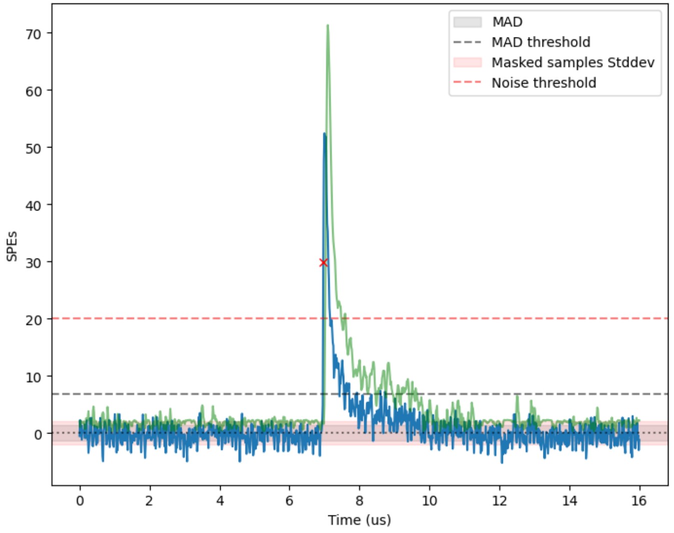
- ***Parameters:***
  - `n_noise_factor` - coefficient for noise floor used to set `height`
  - `n_bins_rolled` - number of bins used in rolling average for dynamic threshold
  - `n_sqrt_rt_factor` - coefficient for statistical uncertainty on rolling average threshold
  - `pe_weight` - weight in stat. uncertainty sum for correcting SPEs (degenerate with `n_sqrt_rt_factor`)

~`threshold`

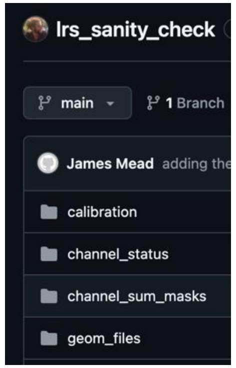
  - `n_bins_skipped` - equivalent to `distance` removed (effectively set to 1)
    - hits in subsequent consecutive bins removed
    - should this be added back in?

LRS sanity checking script

```python
def get_data(filename, calib_filename, geom_filename, channel_status_filename, maskfile, max_evts, n_mad_factor=5.0):

    # load file
    with h5py.File(filename, 'r') as f:

        # load data
        wvfms = f['light/wvfm/data']['samples']
        data_shape = wvfms.shape
        n_evts = data_shape[0]
        n_adcs = data_shape[1]
        n_channels = data_shape[2]
        n_samples = data_shape[3]
        print("Raw wvfms loaded, shape: ", data_shape)

        # get calibration csv file
        calib_csv = pd.read_csv(calib_filename, header=None)
        calib_npy = calib_csv.to_numpy()
        wvfms_calib = wvfms * calib_npy[np.newaxis, :, :, np.newaxis]
        print("Calibrated wvfms loaded, shape: ", wvfms_calib.shape)

        # summing channels by TPC, detector, or trap type
        if maskfile != None:
            masks_file = np.load(maskfile)
            masks = np.array(masks_file['masks'])
            print("Summed channels masks loaded, shape: ", masks.shape)

            # sum channels
            wvfms_summed = np.zeros((n_evts, masks.shape[0], n_samples))
            #noise_summed = np.zeros((n_evts, masks.shape[0]))
            for i in range(masks.shape[0]):
                wvfms_summed[:, i, :] = np.sum(wvfms_calib[:, masks[i]==1, :], axis=(1))
                #noise_summed[:, i] = np.sqrt(np.sum(noise_thresholds[:, masks[i] == 1]**2, axis=1))
        else:
            wvfms_summed = wvfms_calib
            #noise_summed = noise_thresholds
        print("Channels summed, shape: ", wvfms_summed.shape)

        # get baseline and noise threshold per waveform per channel
        baselines, noise_thresholds = get_baseline_and_noise_threshold(wvfms_summed, n_mad_factor=n_mad_factor)
        print("Baselines and noise thresholds calculated, shapes: ", baselines.shape, noise_thresholds.shape)
        wvfms_blsub = wvfms_summed - baselines[..., np.newaxis]
        print("Baseline subtracted wvfms loaded, shape: ", wvfms_blsub.shape)

    return wvfms_blsub, noise_thresholds
```



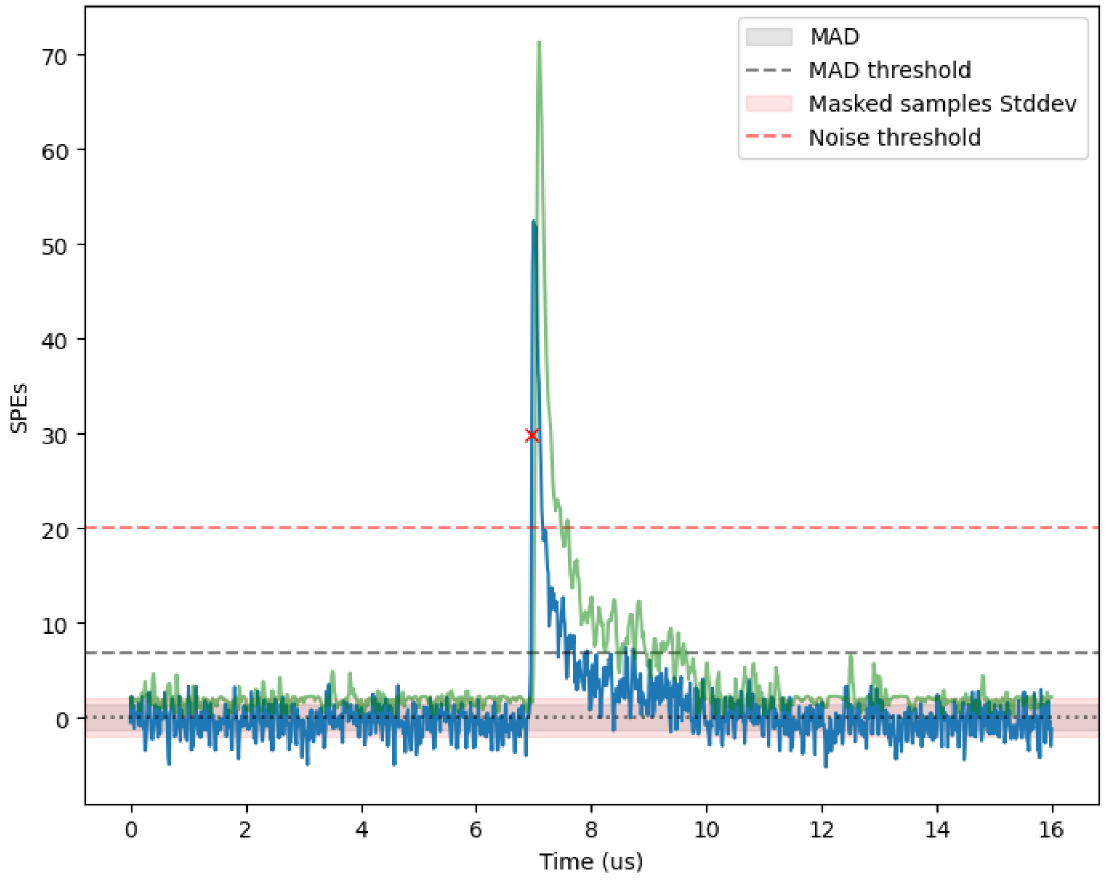
csv files




```python
# function for getting baseline and noise threshold per waveform per channel
def get_baseline_and_noise_threshold(wvfms, n_mad_factor=5.0):
    # Initialize median and MAD
    median = np.median(wvfms, axis=-1)
    mad = np.median(np.abs(wvfms - median[..., np.newaxis]), axis=-1)
    # identify outliers in the waveform
    mad_factor = n_mad_factor * mad
    noise_mask = np.abs(wvfms - median[..., np.newaxis]) < mad_factor[..., np.newaxis]
    print("Noise mask calculated, shape: ", noise_mask.shape)
    # set non mask values to nan
    noise_samples = np.where(noise_mask, wvfms, np.nan)
    print("Noise samples calculated, shape: ", noise_samples.shape)
    # calculate noise as stddev of noise_samples
    noise = np.nanstd(noise_samples, axis=-1)
    print("Noise calculated, shape: ", noise.shape)
    # calculate baseline as mean of noise_samples
    baseline = np.nanmean(noise_samples, axis=-1)
    print("Baseline calculated, shape: ", baseline.shape)

    return baseline, noise
```

github

```python
# interaction finder function
def interaction_finder(wvfm, noise,
                       n_noise_factor = 10.0,
                       n_bins_rolled = 5,
                       n_sqrt_rt_factor = 3.0,
                       pe_weight = 1.0):

  # save hitfinder settings to config
  hit_config = {'n_noise_factor': n_noise_factor,
                'n_bins_rolled': n_bins_rolled,
                'n_sqrt_rt_factor': n_sqrt_rt_factor,
                'pe_weight': pe_weight}

  # height = flat threshold over noise (n*sigma)
  height = n_noise_factor * noise[..., np.newaxis] * np.ones(wvfm.shape[-1])
  print("Height calculated, shapes: ", height.shape)

  # dynamic_threshold = rolling threshold of previous 5 bins + n*sqrt(rolling threshold)
  wvfm_rolled = np.roll(wvfm, n_bins_rolled)
  rolling_average = uniform_filter1d(wvfm_rolled, size=n_bins_rolled)
  sqrt_rolling_average = np.sqrt(np.abs(rolling_average) * pe_weight**2)
  sqrt_rolling_average[sqrt_rolling_average == 0] = 1
  dynamic_threshold = rolling_average + n_sqrt_rt_factor*sqrt_rolling_average
  print("Dynamic threshold, shapes: ", dynamic_threshold.shape)

  # find rising edges
  bins_over_dynamic_threshold = (wvfm > dynamic_threshold) & (wvfm > height)
  # remove consecutive bins, keep only the first
  bins_over_dynamic_threshold[..., 1:] = bins_over_dynamic_threshold[..., 1:] & ~bins_over_dynamic_threshold[..., :-1]

  return bins_over_dynamic_threshold, hit_config
```



github

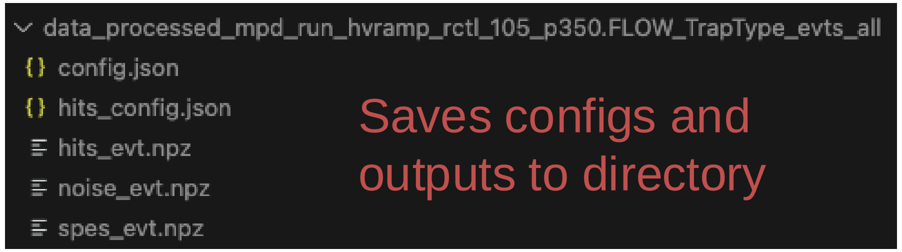
Running settings

Pre-processing config:

```
"timestamp":                "2024-12-17 21:13:21.311933",
"filename":                 "../hdf5s/mpd_run_hvramp_rctl_105_p350.FLOW.hdf5”,
"is_data":                  true,
"summed":                   "TrapType",
"max_evts":                 null,
"calib_filename":           "calibration/data_calib.csv",
"geom_filename":            "geom_files/light_module_desc-5.0.0.csv",
"channel_status_filename":  "channel_status/channel_status.csv",
"maskfile":                 "channel_sum_masks/TrapType_masks_data.npz”
```

Hit-finder config:

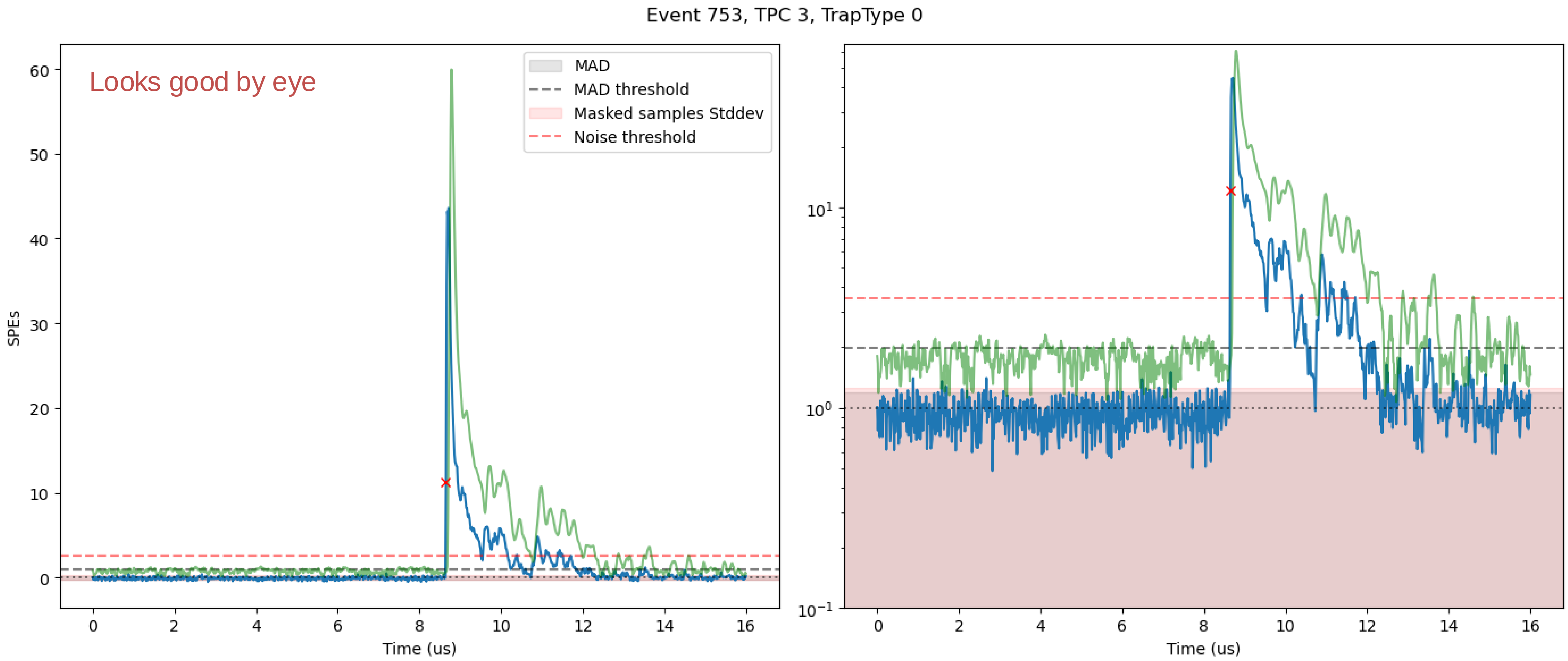
```
"n_noise_factor":   10.0,
"n_bins_rolled":    2,
"n_sqrt_rt_factor": 3.0,
"pe_weight":        1.0
```



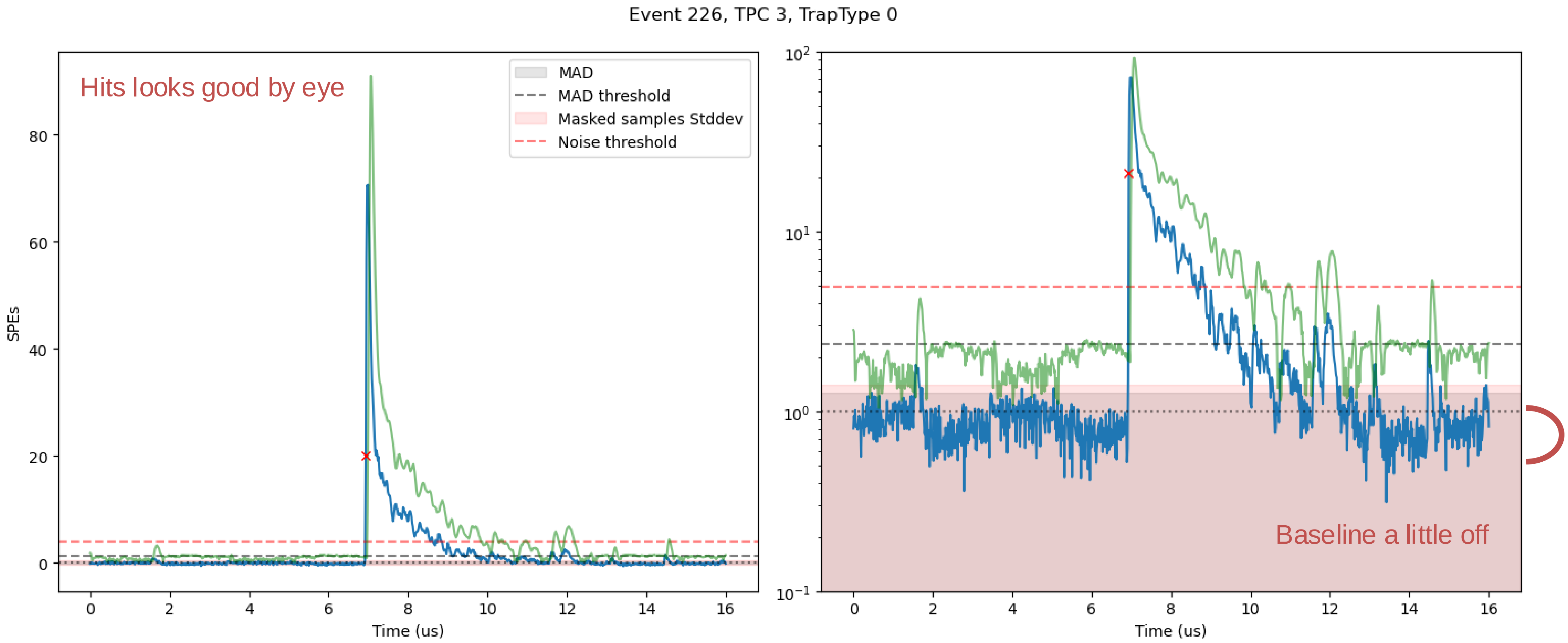
Saves configs and outputs to directory

Tuning possible but need assessment metrics

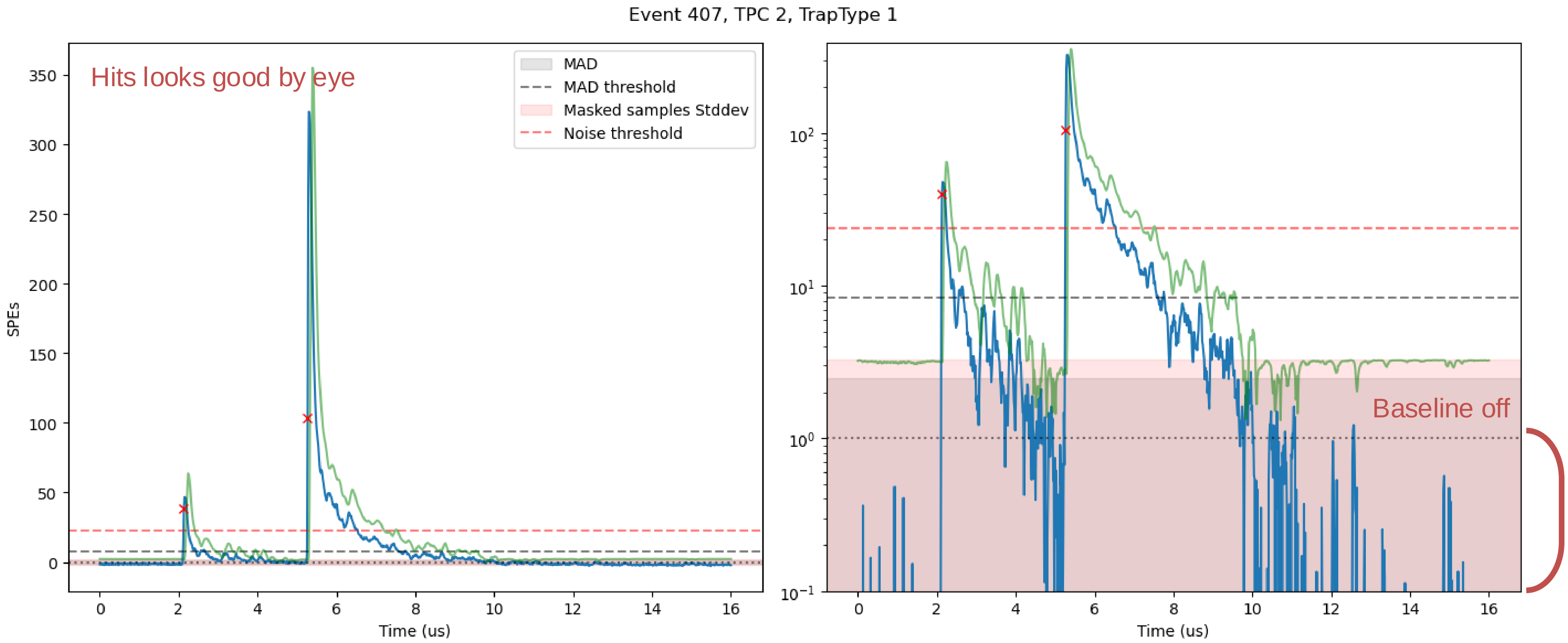
[github](https://github.com)


Event 753, TPC 3, TrapType 0
Looks good by eye
MAD
MAD threshold
Masked samples Stddev
Noise threshold
SPEs
Time (us)
Time (us)

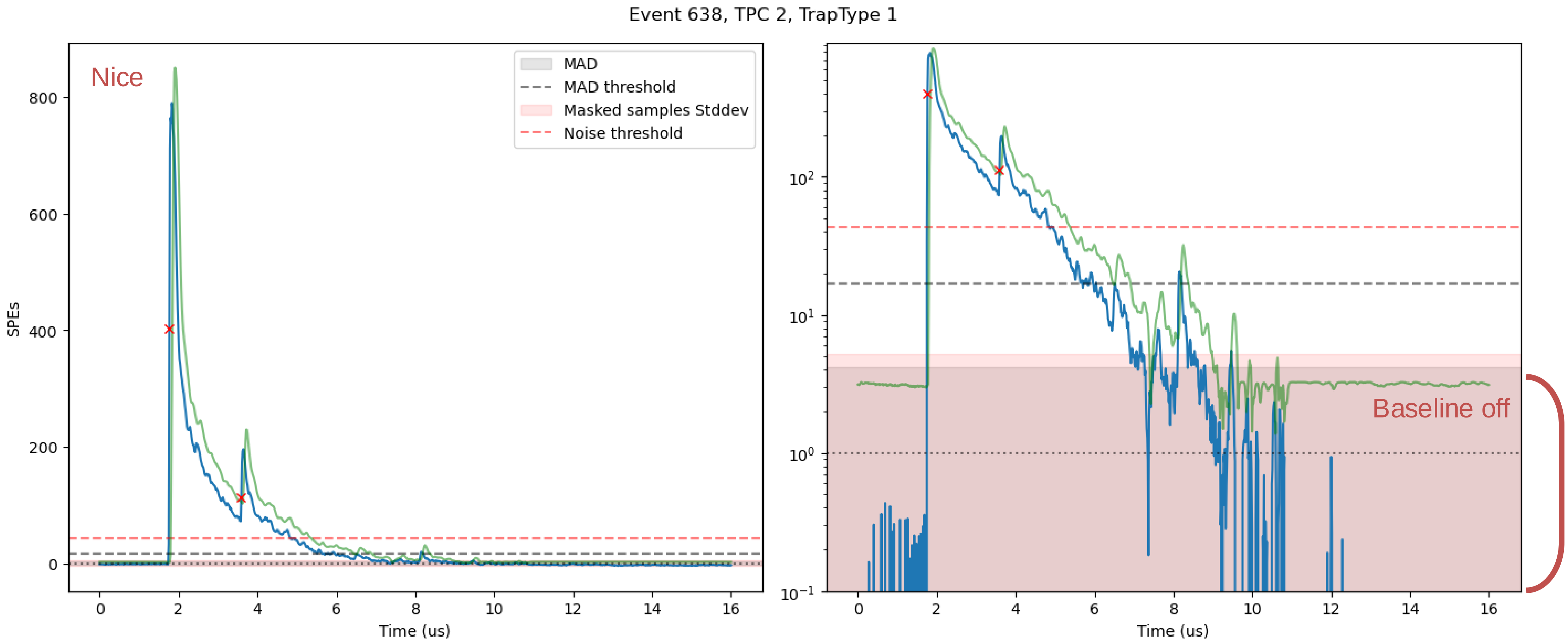
Working as intended?

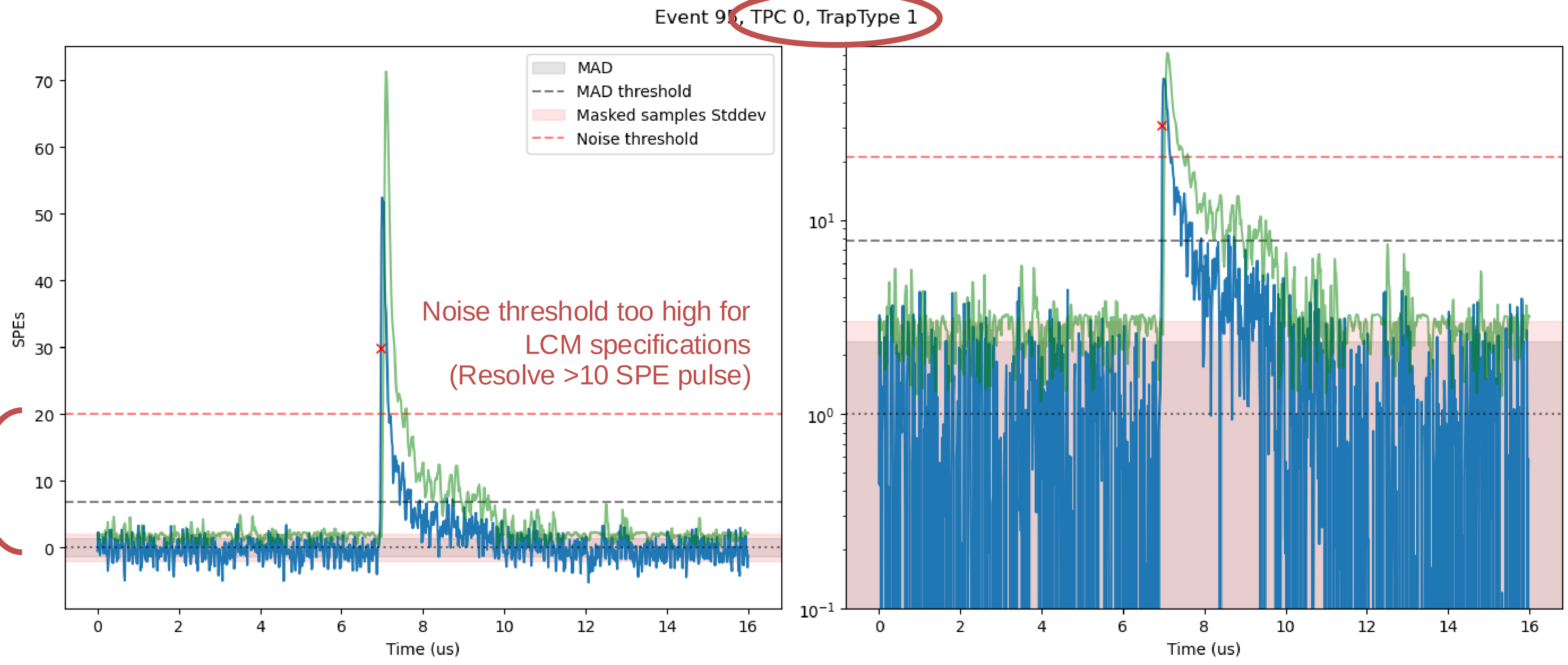


Working as intended?

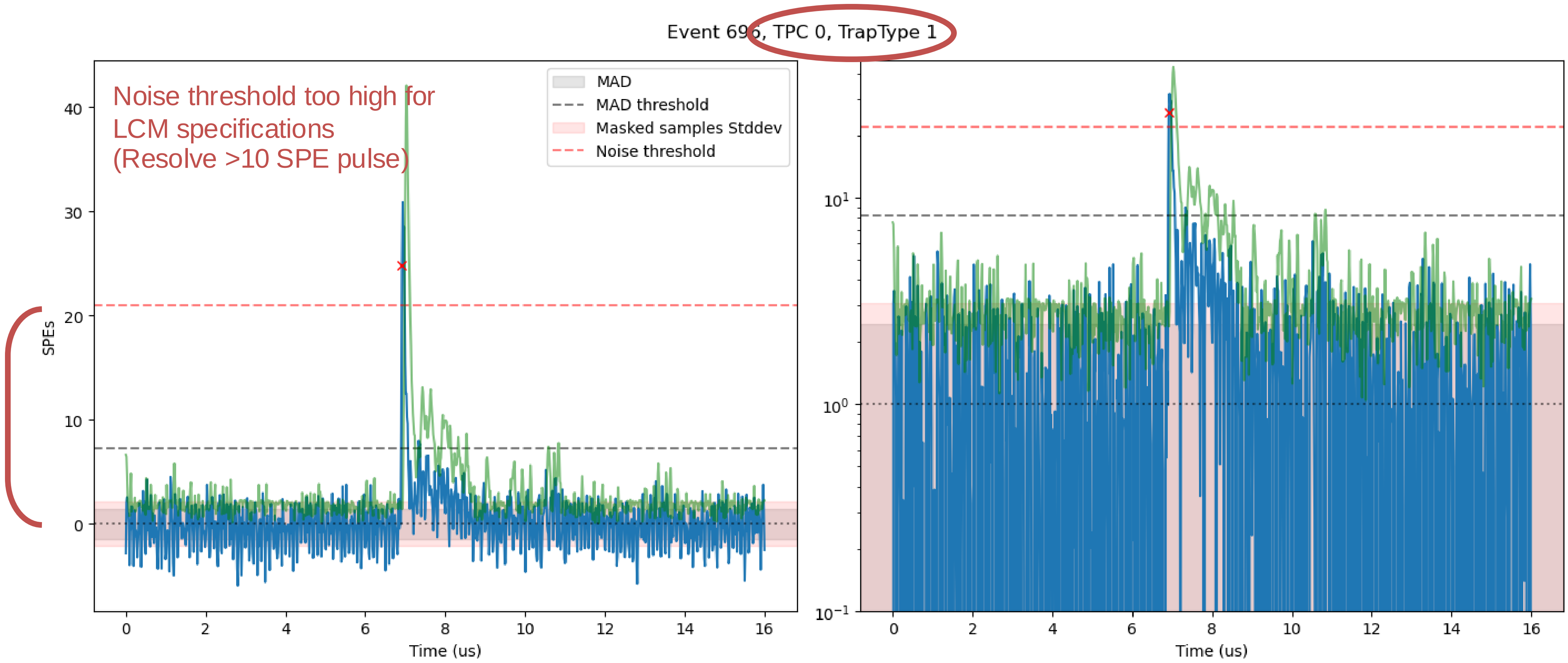


Summed by TPC+TrapType

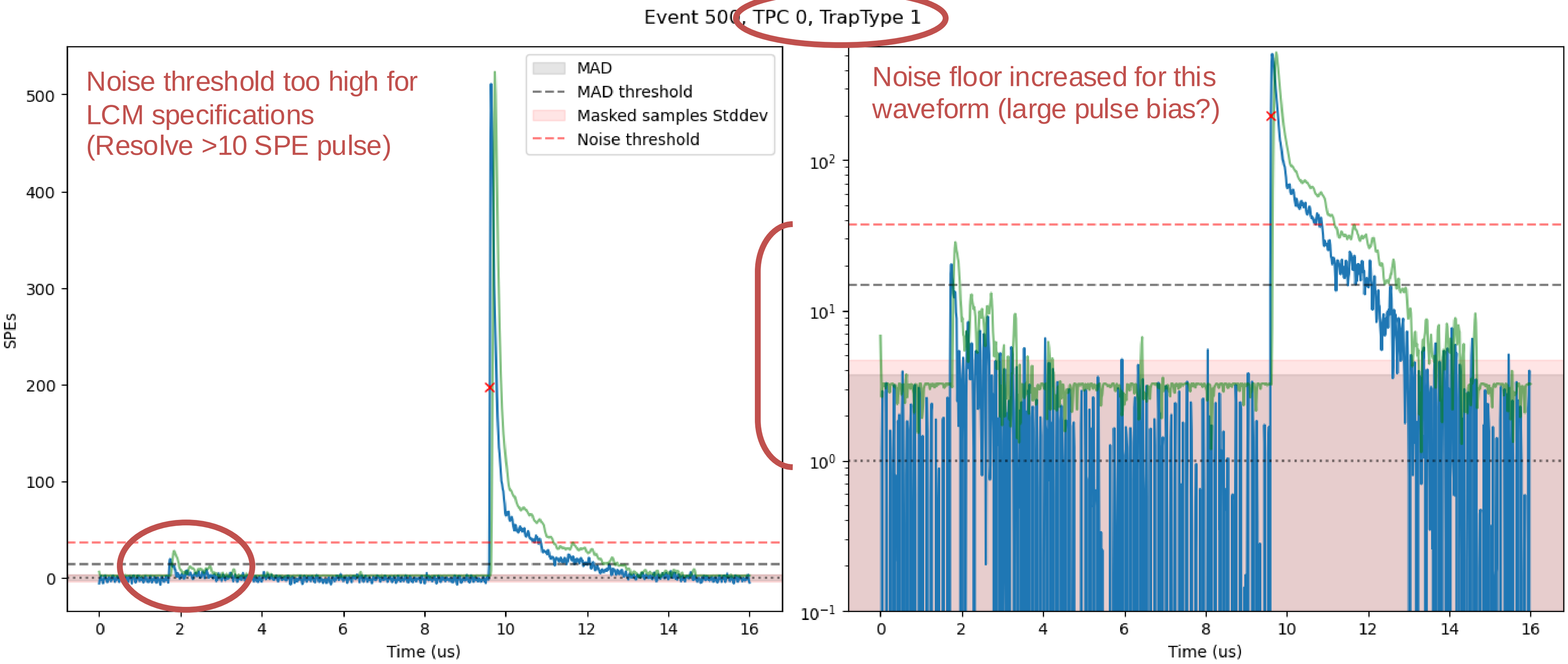




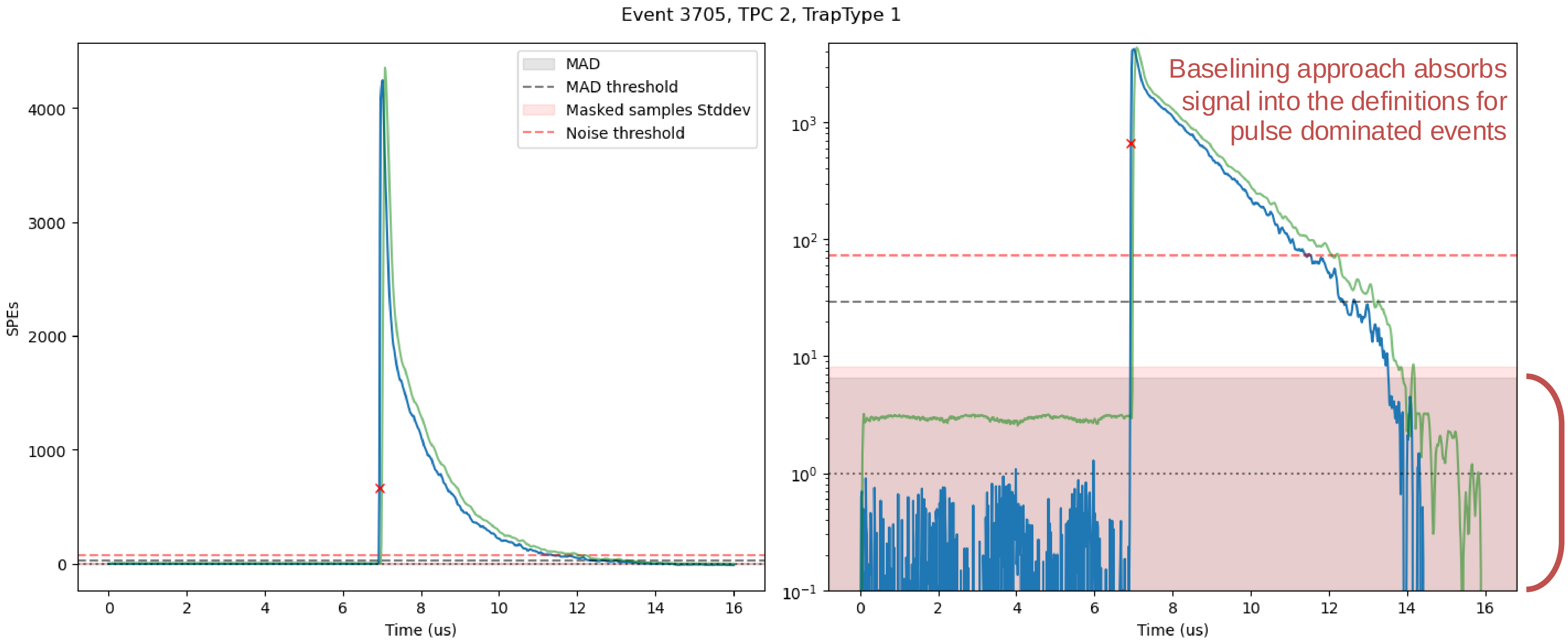
Noise threshold too high for LCM specifications (Resolve >10 SPE pulse)


Event 696, TPC 0, TrapType 1
Noise threshold too high for
LCM specifications
(Resolve >10 SPE pulse)
MAD
MAD threshold
Masked samples Stddev
Noise threshold
SPEs
Time (us)
Time (us)

Large pulse bias

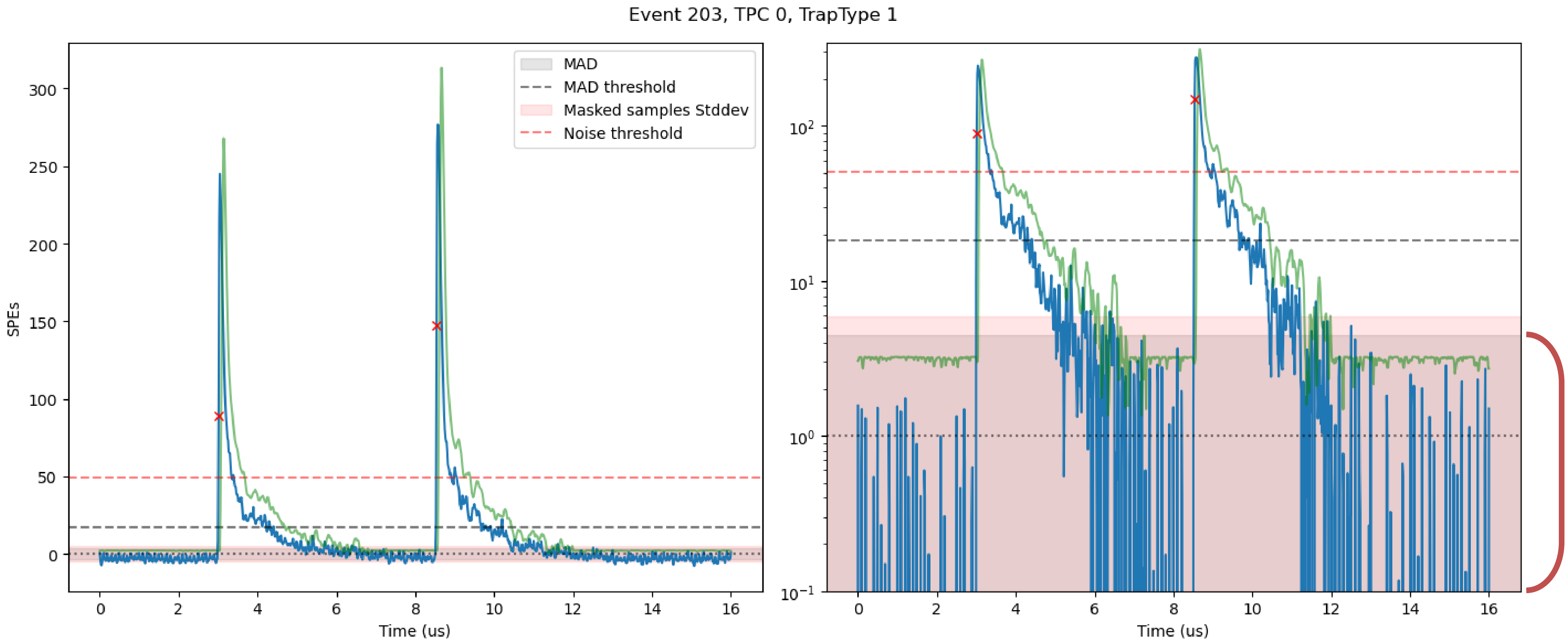


Large pulse bias

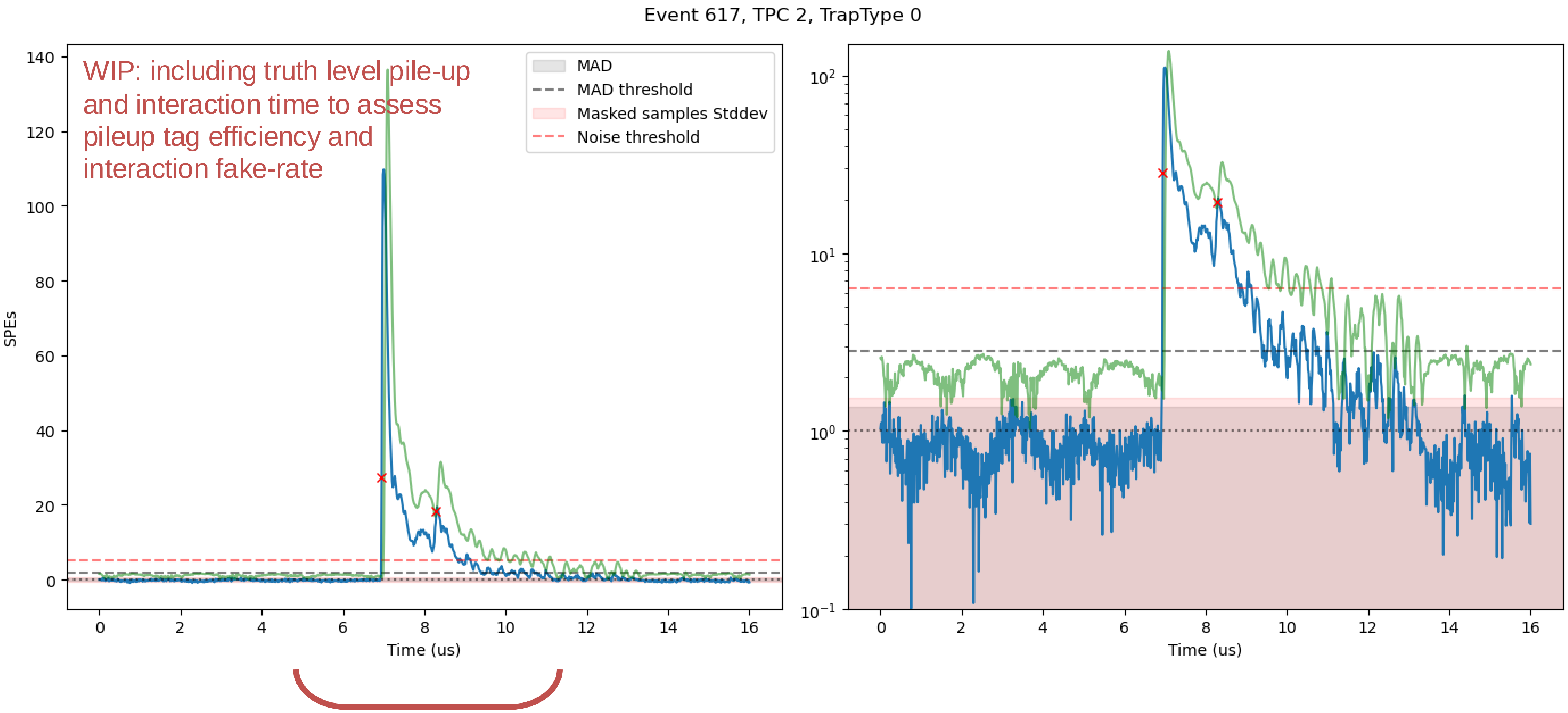


Baselining approach absorbs signal into the definitions for pulse dominated events

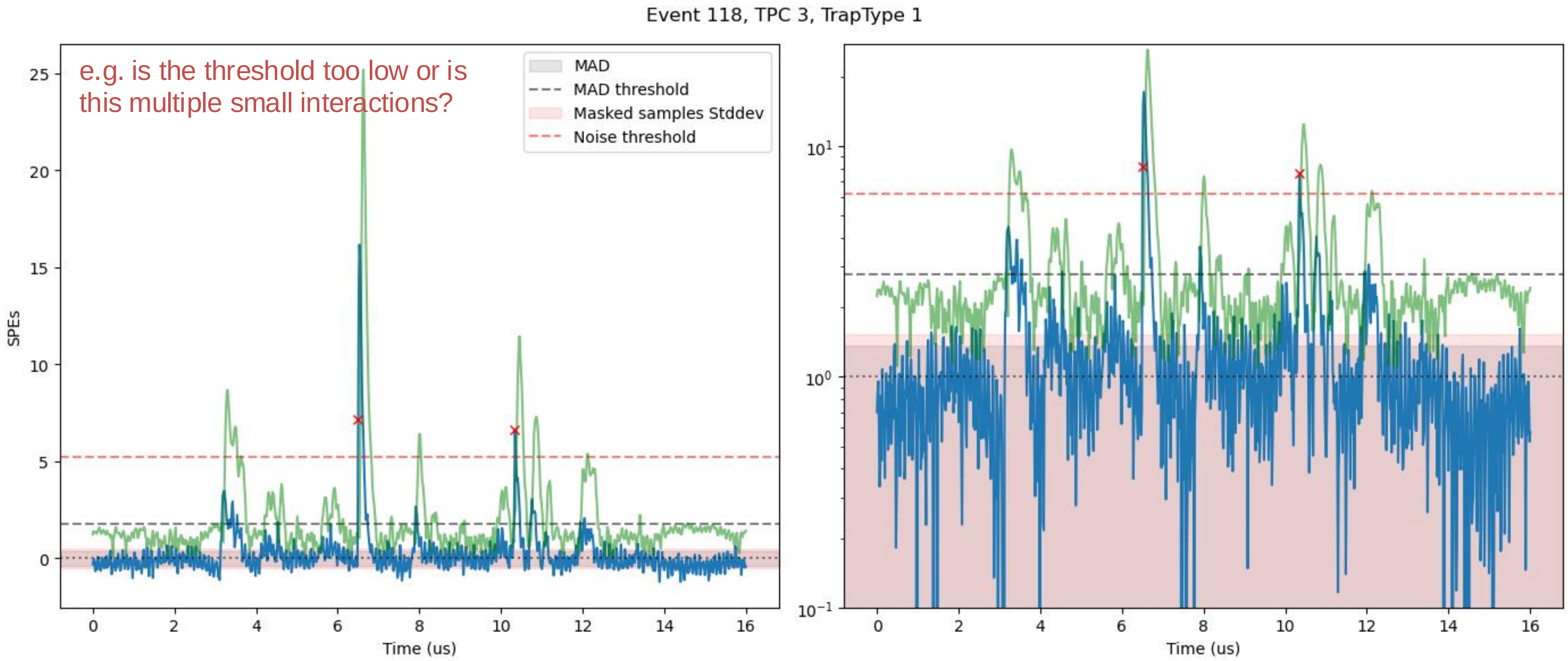
Large pulse bias



More information required

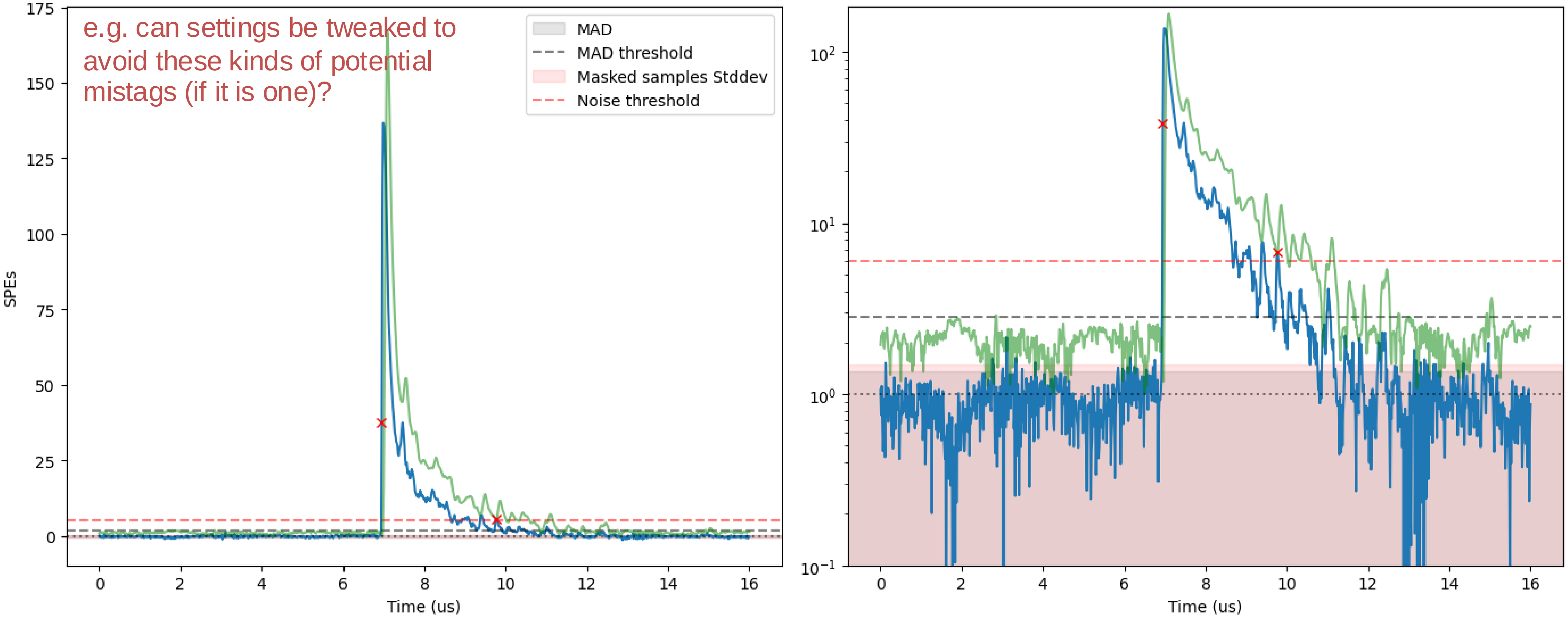


More information required

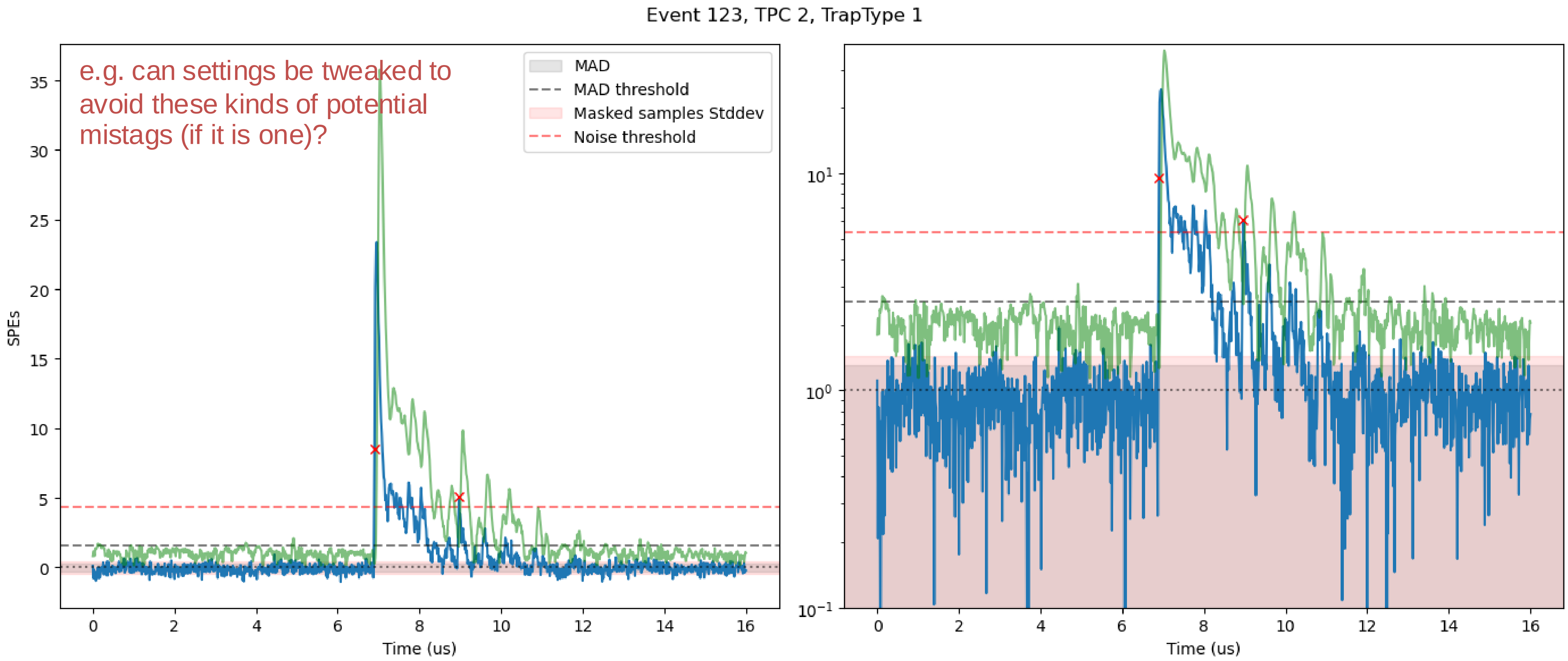


More information required

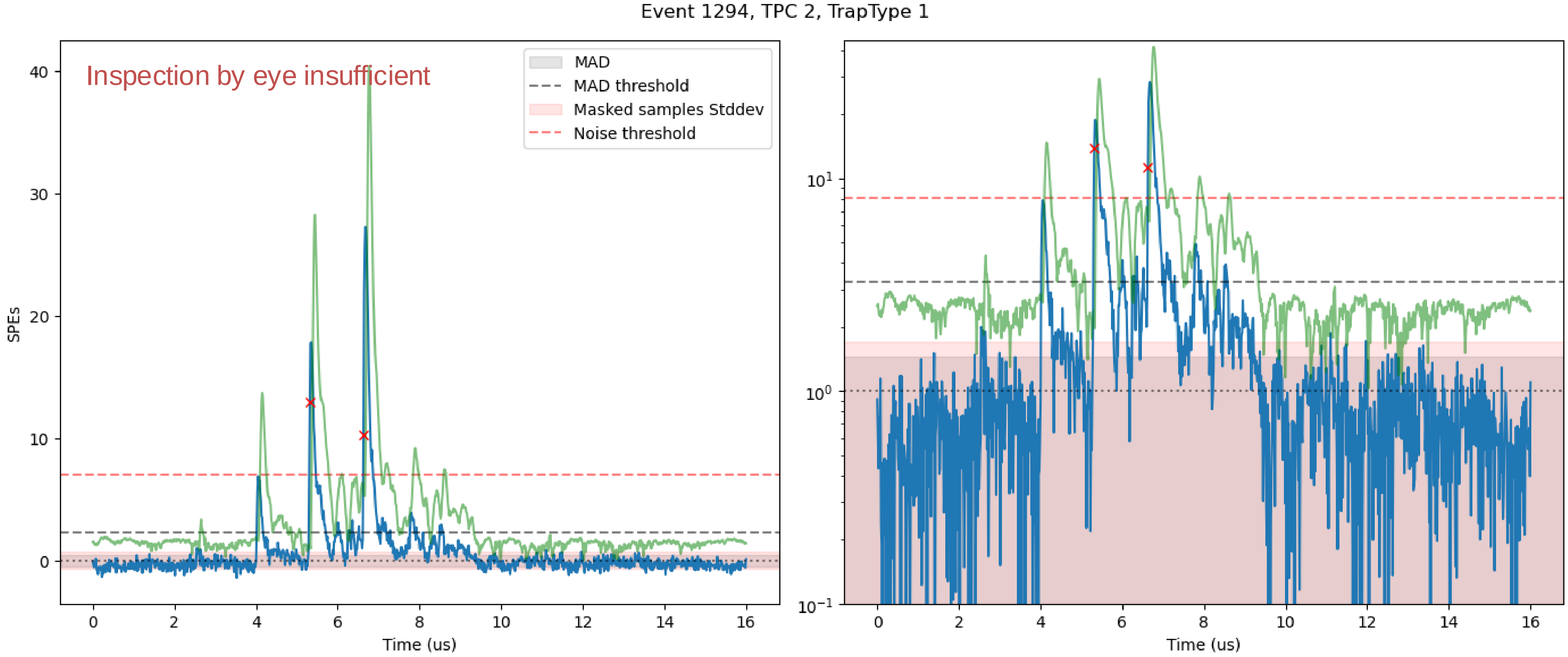
Event 284, TPC 2, TrapType 1



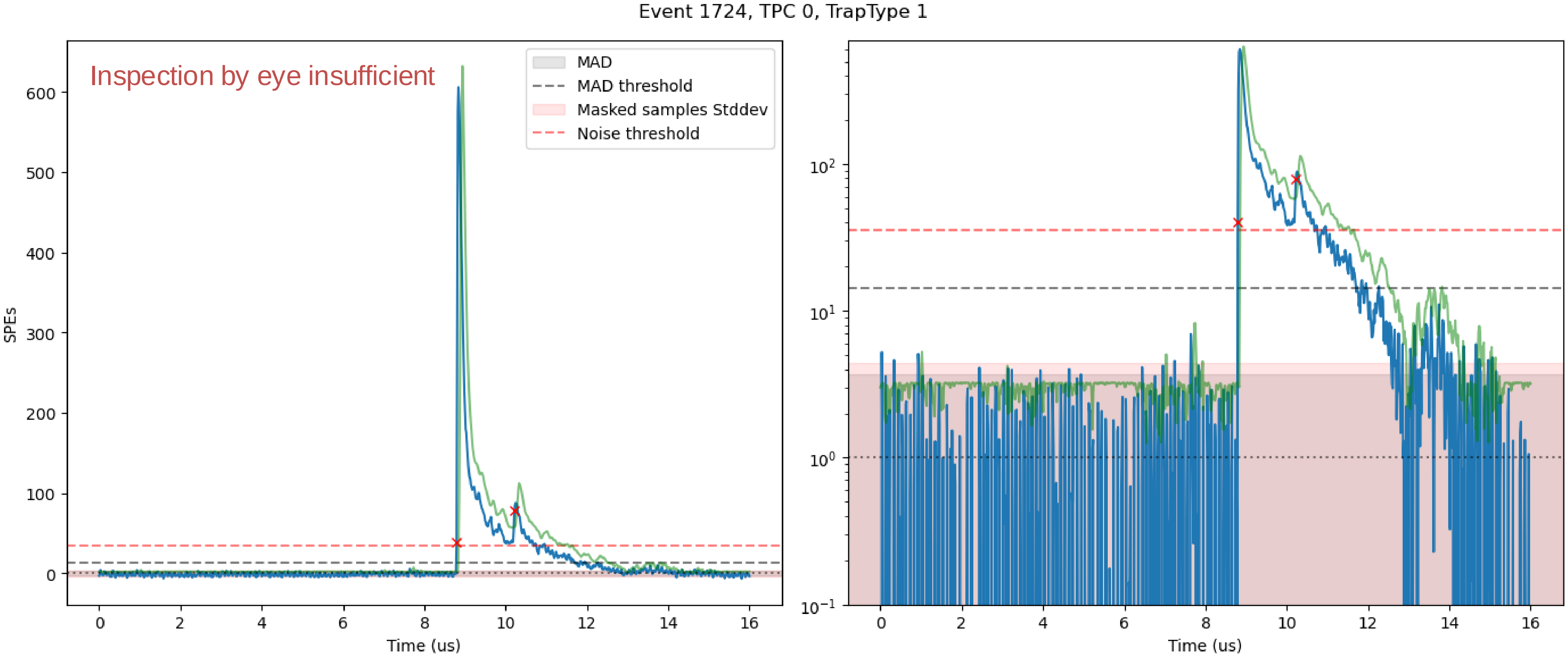
More information required

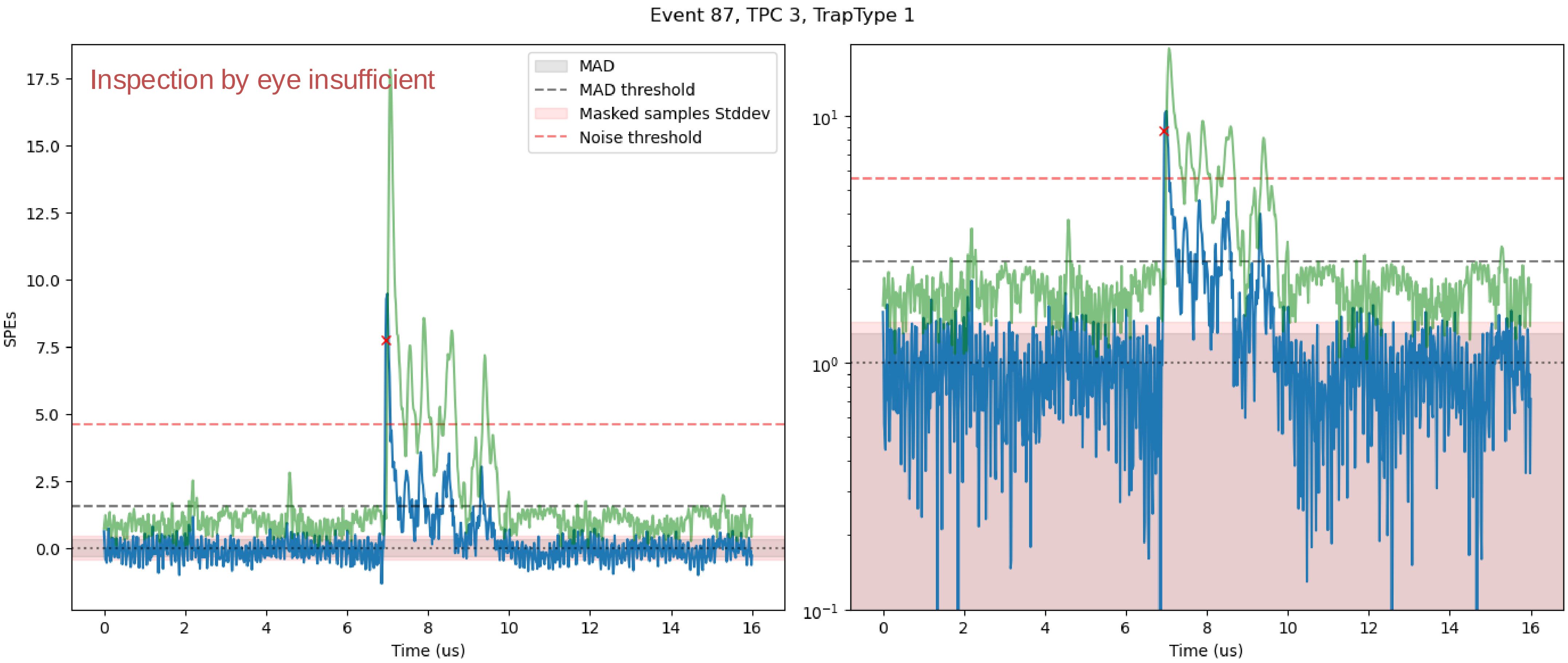


More information required



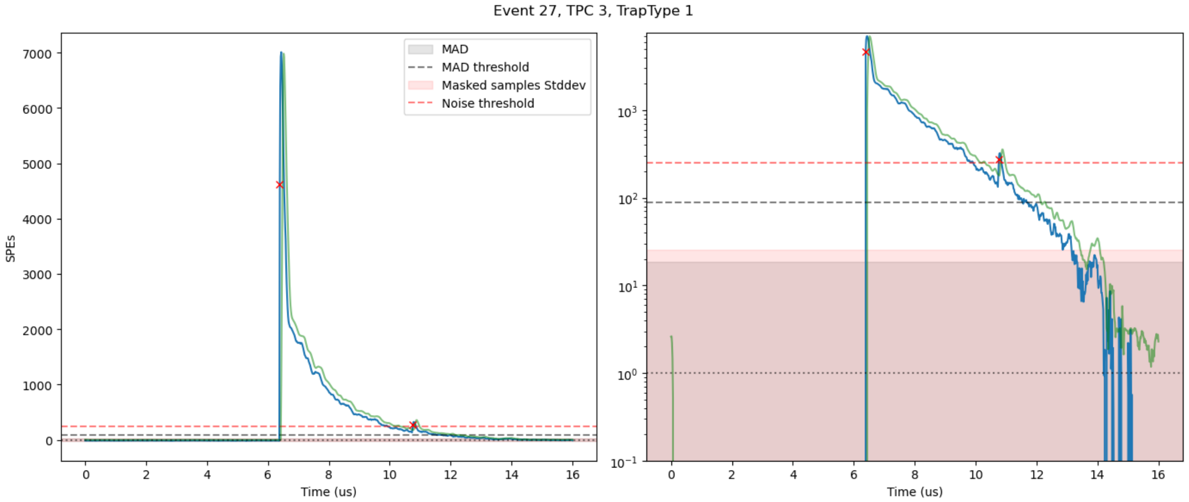
More information required

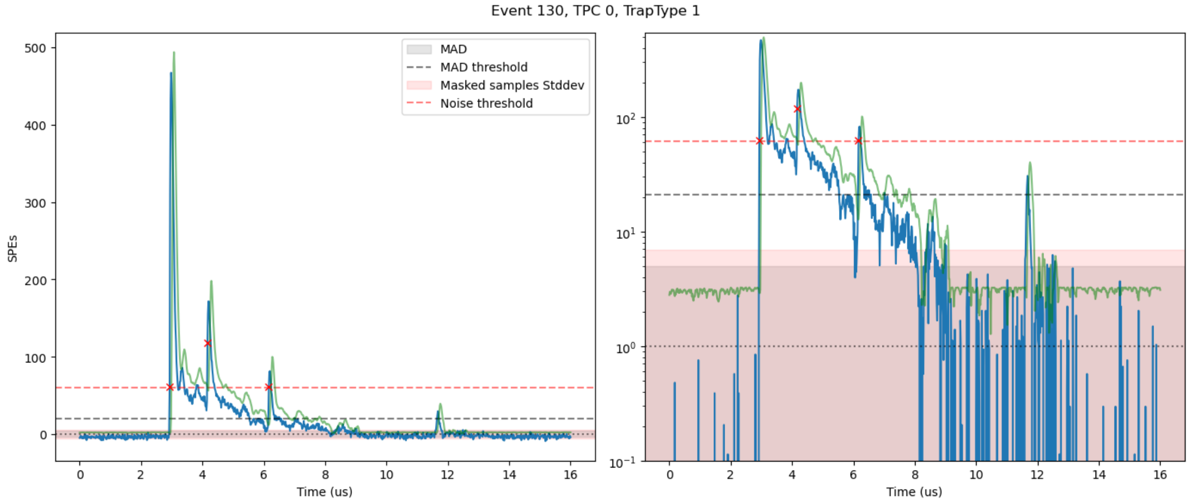




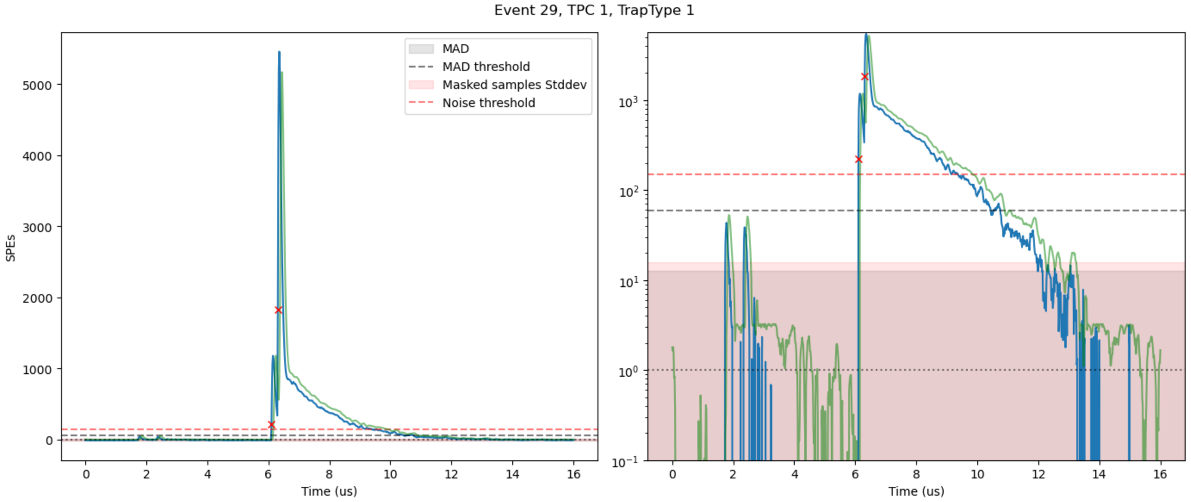
Event 87, TPC 3, TrapType 1
Inspection by eye insufficient
MAD
MAD threshold
Masked samples Stddev
Noise threshold
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
SPEs
0
2
4
6
8
10
12
14
16
Time (us)
10^1
10^0
10^{-1}
Time (us)

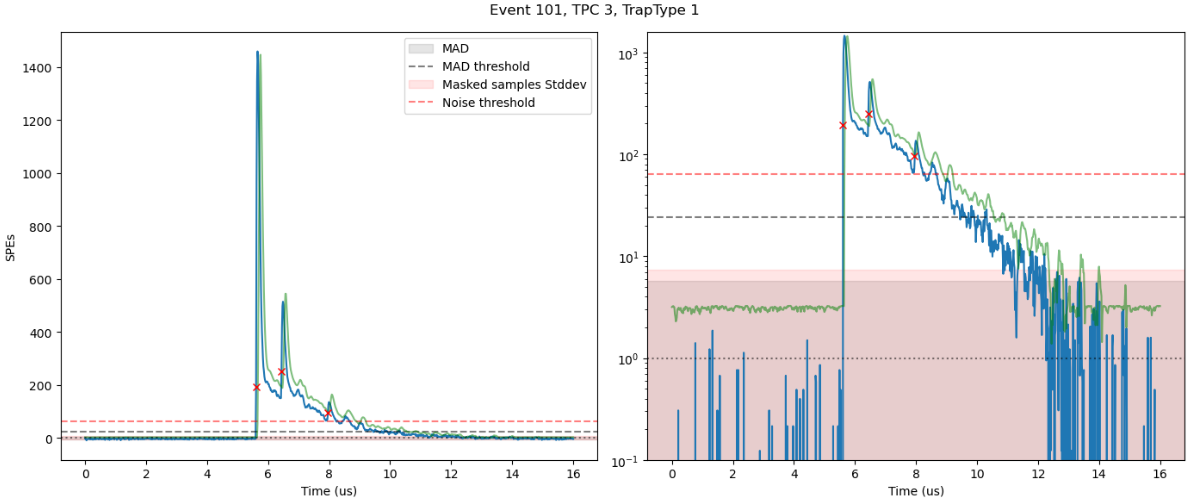
Examples from MC: same settings





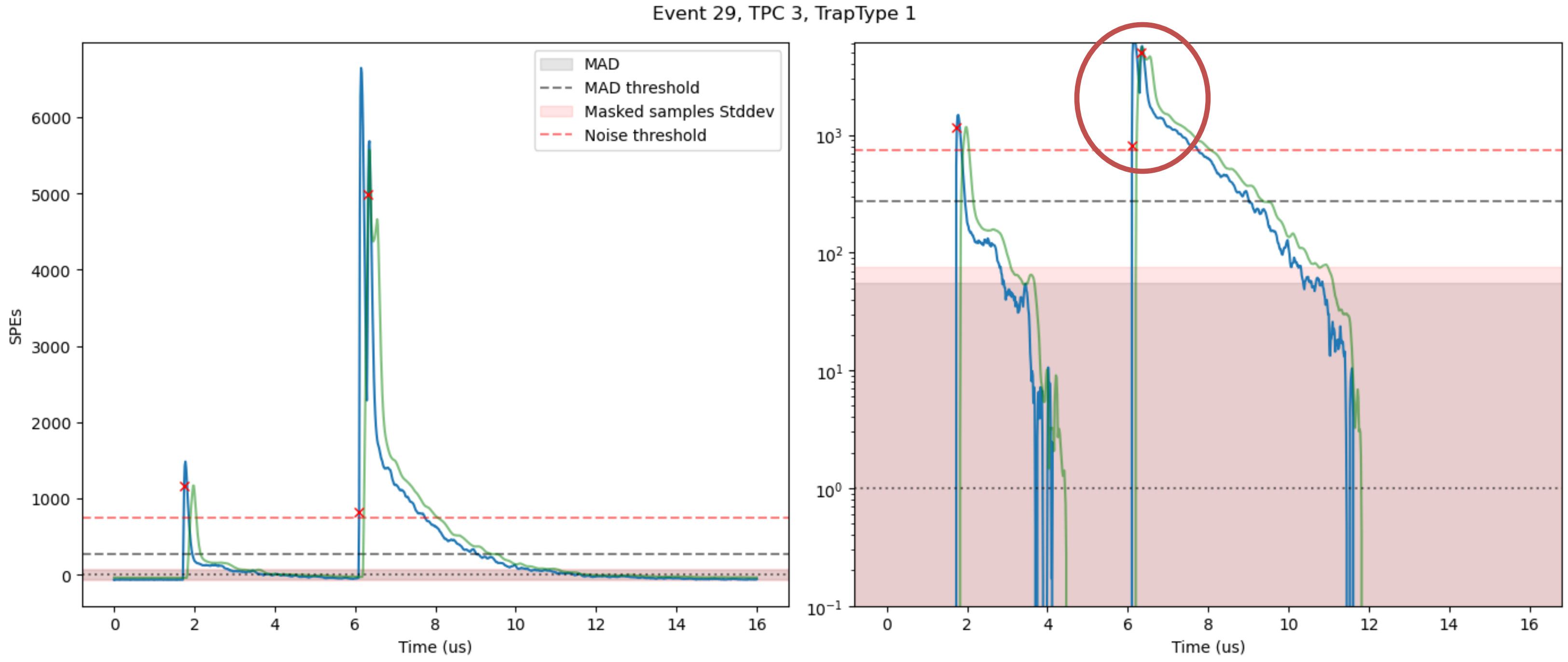
Is the bad baselining saving us from stat. fluctuations in the tail by inflating the noise threshold?





Examples from MC: changed settings

Δt dependant performance study underyway by Tom



Increased n_bins_rolled to 10 to smooth the dynamic threshold…
… should also increase n_sqrt_rt_factor to compensate for the resulting reduction in threshold level?

Conclusion

- Standardising some tools for diagnostics / analysis
  - csv/numpy channel maps – status, TPC, light-trap – done. WIP: (x,y,z)
  - Including calibrated gains for bits -> SPEs scale
  - Swap out baseline and noise estimation method?
- MAD-mask baselining
  - Able to be performed on every channel for every event in a file
  - Still one of the slowest parts of the preprocessing chain
  - Current method is susceptible to noisy channels and pulse-dominated waveforms
- Including truth level information (for time resolution and pileup efficiency/fake-rate)
  - MC: trajectories → t_start OR segments → t0_start
  - Data: Charge-light matching through-going muon OR Δt between adjacent channels?

Assess impact of:
Clipping/saturation
Cross-talk
Baselining biases
Hit finding algorithms
…on these metrics

Assess impact of:
Sampling rate
Bit depth
WLS decay-time
Trap geometry
Instrumentation density