PMT analysis – Update

Cygno reco and analysis meeting - 10/03/2022

David Marques

Gianluca Cavoto

PMT analysis - Data

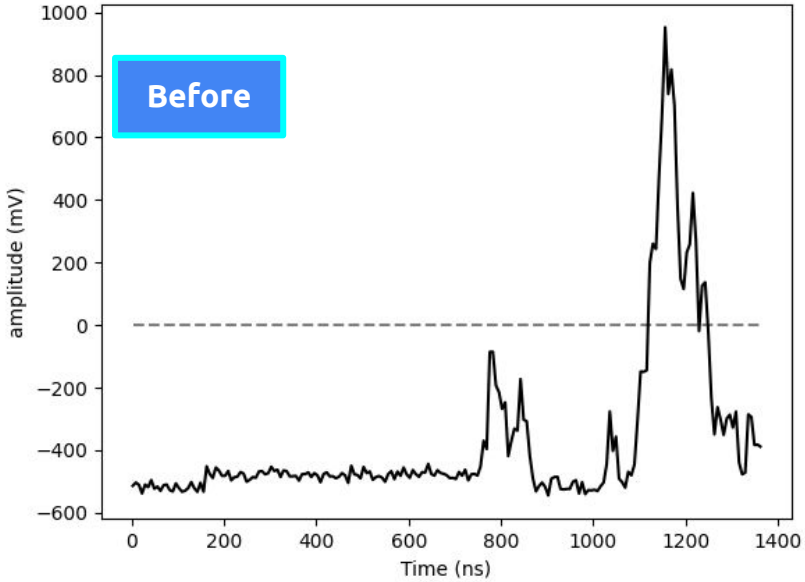
1. Runs to be analysed: 4432 - 4469
   a. Runs 4416-4431 can be discarded as they were taken using a different trigger logic.
2. Emanuele had already a base code for the analysis of the PMT waveforms:
   a. It's already correctly integrated with reco_code
   b. Already has the tree filling functions, and some parameters calculation.
   c. *I decided to use it as a basis instead of writing a new code from scratch.*
3. Find here more details about the data taken (from Davide):
   https://github.com/CYGNUS-RD/WIKI-documentation/wiki/Detector-General#overground-data-tackings
4. Initial work:
   a. Looking around for the meaning of the input parameters:

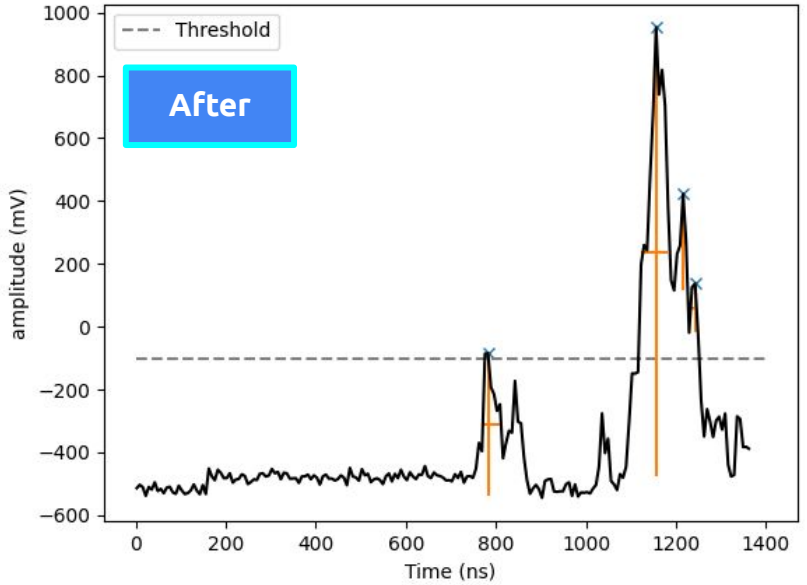
```
if self.options.pmt_mode:
    if obj.InheritsFrom('TGraph'):
        # PMT waveform reconstruction
        from waveform import PeakFinder,PeaksProducer
        wform = tf.Get('wfm_'+'_'.join(name.split('_')[1:]))
        # sampling was 5 GHz (5/ns). Rebin by 5 (1/ns)
        pkprod_inputs = {'waveform': wform}
        pkprod_params = {'threshold': options.threshold, # min threshold for a signal (baseline is -20 mV)
                         'minPeakDistance': options.minPeakDistance, # number of samples (1 sample = 1ns )
                         'prominence': options.prominence, # noise after resampling very small
                         'width': options.width, # minimal width of the signal
                         'resample': options.resample,  # to sample waveform at 1 GHz only
                         'rangex': (self.options.time_range[0],self.options.time_range[1]),
                         'plotpy': options.pmt_plotpy
        }
        pkprod = PeaksProducer(pkprod_inputs,pkprod_params,self.options)

        peaksfinder = pkprod.run()
        self.autotree.fillPMTVariables(peaksfinder,0.2*pkprod_params['resample'])
```

- **Rebin/resample:** Kept at 5 - related to the oscilloscope sampling.
- **Prominence:** Related to noise, but not sure what it does
- **Width:** ~minimum distance between peaks.
- **Threshold:** ~minimum amplitude to be considered a peak

PMT analysis - Data

1. Example:
   a. Threshold changed and plotted
   b. Peak finder optimized





PMT analysis - Parameters

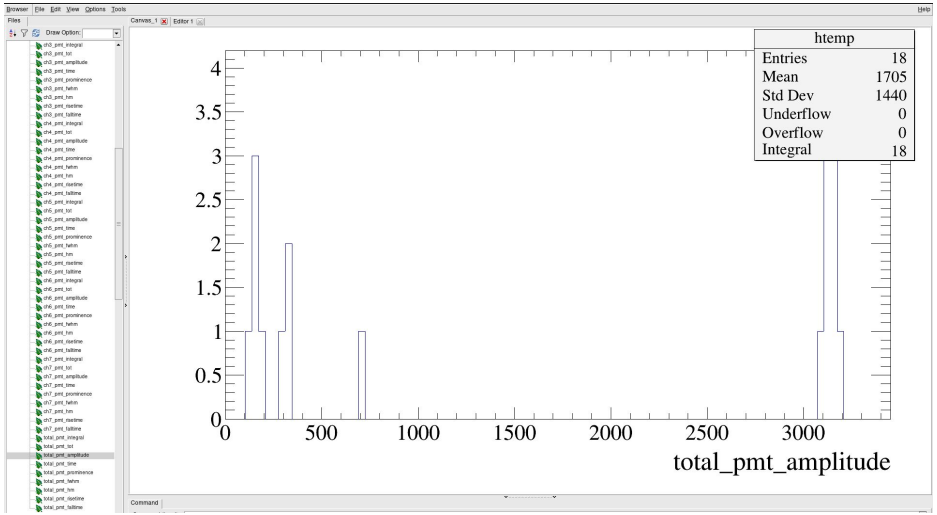
1. There were already quite a few parameters accessible for all the peaks:

```
def plotpy(self,pdir='./',xlabel='Time (ns)',ylabel='amplitude (mV)'):···
def getPeakBoundaries(self,side):···
def getFWHMs(self):···
def getFullWidths(self):···
def getTimes(self,side='rise'):···
def getHMs(self):···
def getPeakTimes(self):···
def getProminences(self):···
def getAmplitudes(self):···
def setTot(self,threshold=0):···
def getTot(self):···
def getIntegral(self):···
```

```
Peak information:
Amplitude: [262.84179688]
Peak left: [749.146]; Peak right: [775.806]
Full width at half maximum: [3.753062]
Rise time: [102.641]; Fall time: [1249.021]
Total: 13.329999999999927
Integral: 254.296875
Channel_check:  ch2
Trigger_check:  2
```

PMT analysis - Small upgrades

1. Changed the tree to save the peak parameters for **each channel** and **all together**:

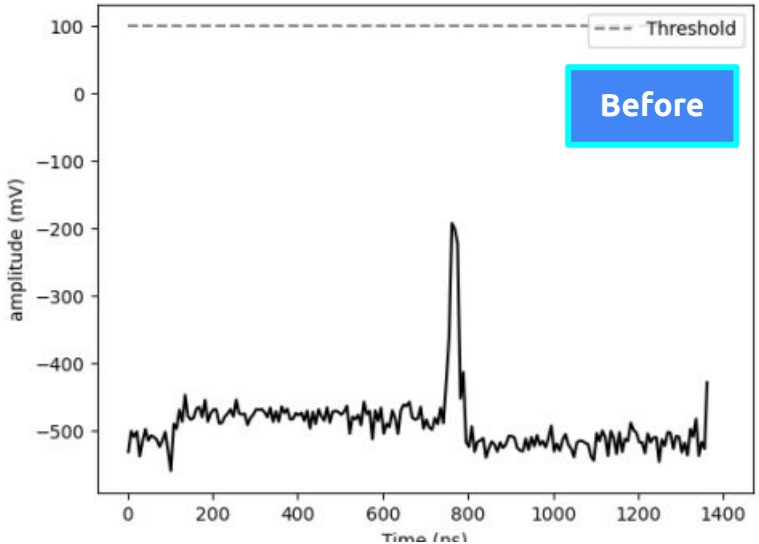


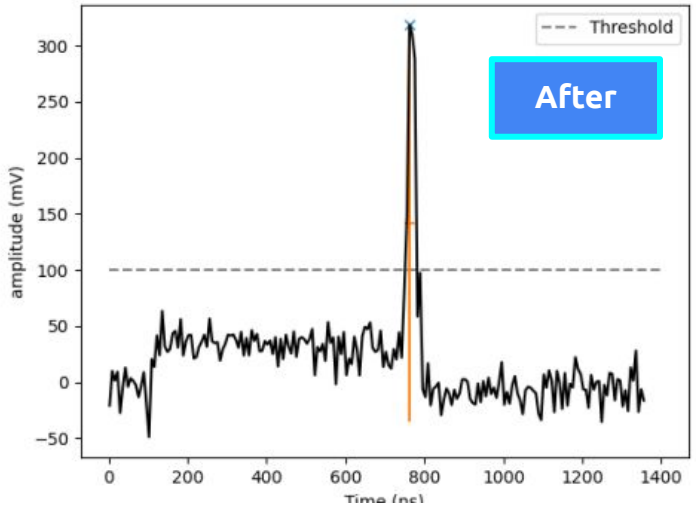

```python
def createPMTVariables(self,channels):
    for i in channels:
        self.outTree.branch('{ch}_pmt_integral'.format(ch=i), 'F')
        self.outTree.branch('{ch}_pmt_tot'.format(ch=i), 'F')
        self.outTree.branch('{ch}_pmt_amplitude'.format(ch=i), 'F', lenVar='nPeak')
        self.outTree.branch('{ch}_pmt_time'.format(ch=i), 'F', lenVar='nPeak')
        self.outTree.branch('{ch}_pmt_prominence'.format(ch=i), 'F', lenVar='nPeak')
        self.outTree.branch('{ch}_pmt_fwhm'.format(ch=i), 'F', lenVar='nPeak')
        self.outTree.branch('{ch}_pmt_hm'.format(ch=i), 'F', lenVar='nPeak')
        self.outTree.branch('{ch}_pmt_risetime'.format(ch=i), 'F', lenVar='nPeak')
        self.outTree.branch('{ch}_pmt_falltime'.format(ch=i), 'F', lenVar='nPeak')

def fillPMTVariables(self,peakFinder,sampleSize,channels):
    self.outTree.fillBranch('{ch}_pmt_integral'.format(ch=channels),peakFinder.getIntegral()*sampleSize)
    # print('I just filled branch named: ' + '{ch}_pmt_integral'.format(ch=channels))
    self.outTree.fillBranch('{ch}_pmt_tot'.format(ch=channels),peakFinder.getTot())
    self.outTree.fillBranch('{ch}_pmt_amplitude'.format(ch=channels),peakFinder.getAmplitudes())
    self.outTree.fillBranch('{ch}_pmt_time'.format(ch=channels),peakFinder.getPeakTimes())
    self.outTree.fillBranch('{ch}_pmt_prominence'.format(ch=channels),peakFinder.getProminences())
    self.outTree.fillBranch('{ch}_pmt_fwhm'.format(ch=channels),peakFinder.getFWHMs())
    self.outTree.fillBranch('{ch}_pmt_hm'.format(ch=channels),peakFinder.getHMs())
    self.outTree.fillBranch('{ch}_pmt_risetime'.format(ch=channels),peakFinder.getTimes('rise'))
    self.outTree.fillBranch('{ch}_pmt_falltime'.format(ch=channels),peakFinder.getTimes('fall'))
```

PMT analysis - Baseline removal

1. Found baseline in the first 1/10th of points.
2. Remove it in all points.
3. Also removed the last point that always goes up (issue discussed later)

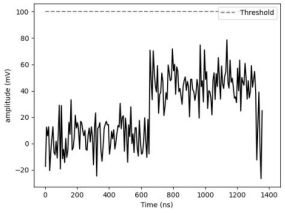
```
def substract_baseline(self,y):
    baseline = 0;
    first_tens = round(len(y)/10)
    for i in range(first_tens):
        baseline += (y[i]/first_tens)

    for j in range(len(y)):
        y[j] = y[j] - baseline
```

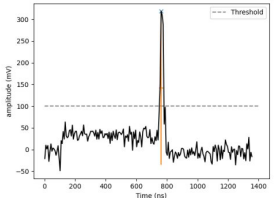


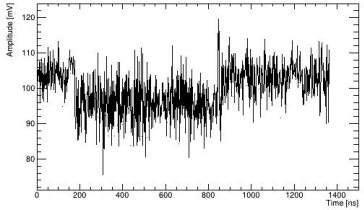


PMT analysis - Some data issue

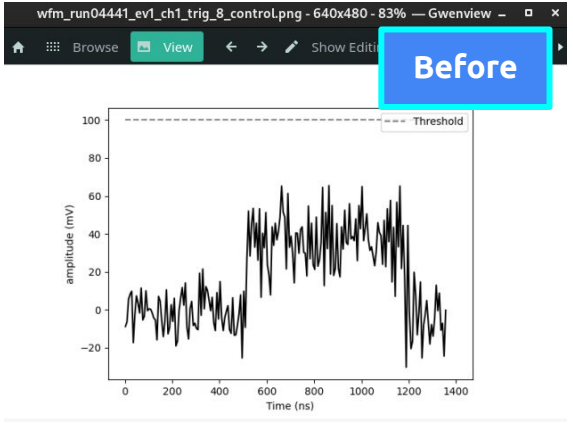
1. There is an additional baseline:
   a. This is sort of a **step function**, present in all and only PMT waveforms.

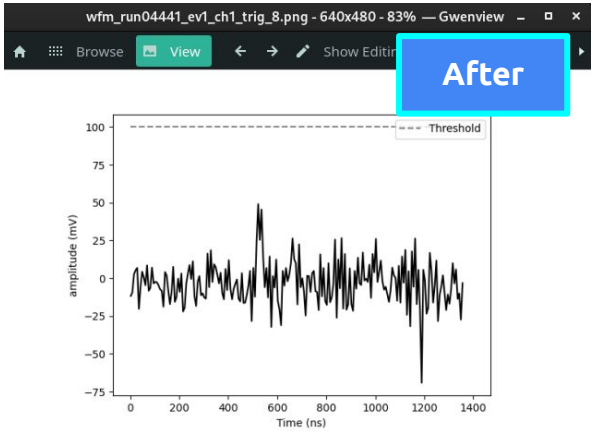






   b. Giorgio tells me that there should be a table incorporated in the DAQ to solve this issue, but maybe it's not working.
2. I'm working on a moving average function to correct this.
   a. It's on a good way, but there's still some work to do…



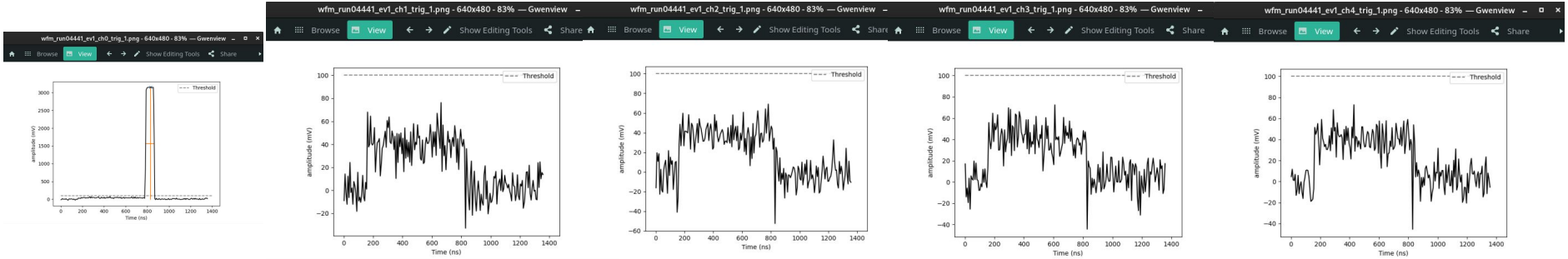


PMT analysis - Some data issue

1. The trigger system might not be working correctly.
   a. Let's look into run 4441

30/07 LIME Test (He/CF4) Global - Reset SCAN Vgem1 - Pos 46 cm (dark in lab)(air cooled) -Logic level 2 - TRG with 1 us Veto				
4441	200	200	Source Fe - Pos 46 cm - Vg1 440V	60/40
4442	200	200	Source Fe - Pos 46 cm - Vg1 431V	60/40

   b. This has a trigger ***logic level 2***
      i. This means that at **least two PMT** should **be over-threshold** to have a **<u>trigger</u>** … but … there seems to be no signals. The raw files show the same.

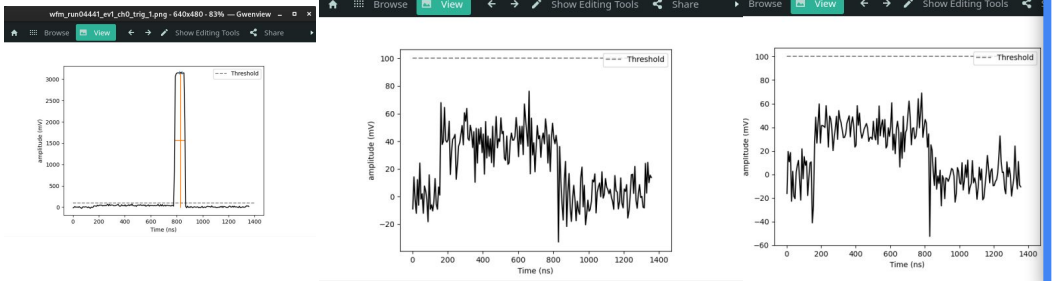


PMT analysis - Some data issue

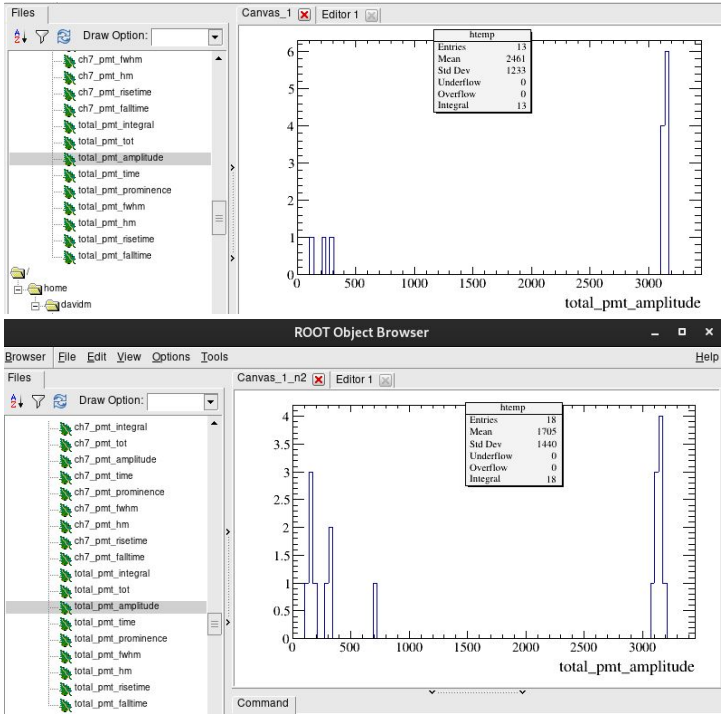
1. The trigger system might not be working correctly.
   a. Let's look into run 4441

30/07 LIME Test (He/CF4) Global - Reset SCAN Vgem1 - Pos 46 cm (dark in lab)(air cooled) -Logic level 2 - TRG with 1 us Veto				
4441	200	200	Source Fe - Pos 46 cm - Vg1 440V	60/40
4442	200	200	Source Fe - Pos 46 cm - Vg1 431V	60/40

   b. This has a trigger ***logic level 2***
      i. This means that at **least two PMT** should **be over-threshold** to have a **trigger** … but … there seems to be no signals. The raw files show the same.



   a. In fact, looking at the number of found peaks…



Most of them **are the trigger peaks**, at 3V.