Data Processing and pre-analysis

C.Caputo
UCLouvain

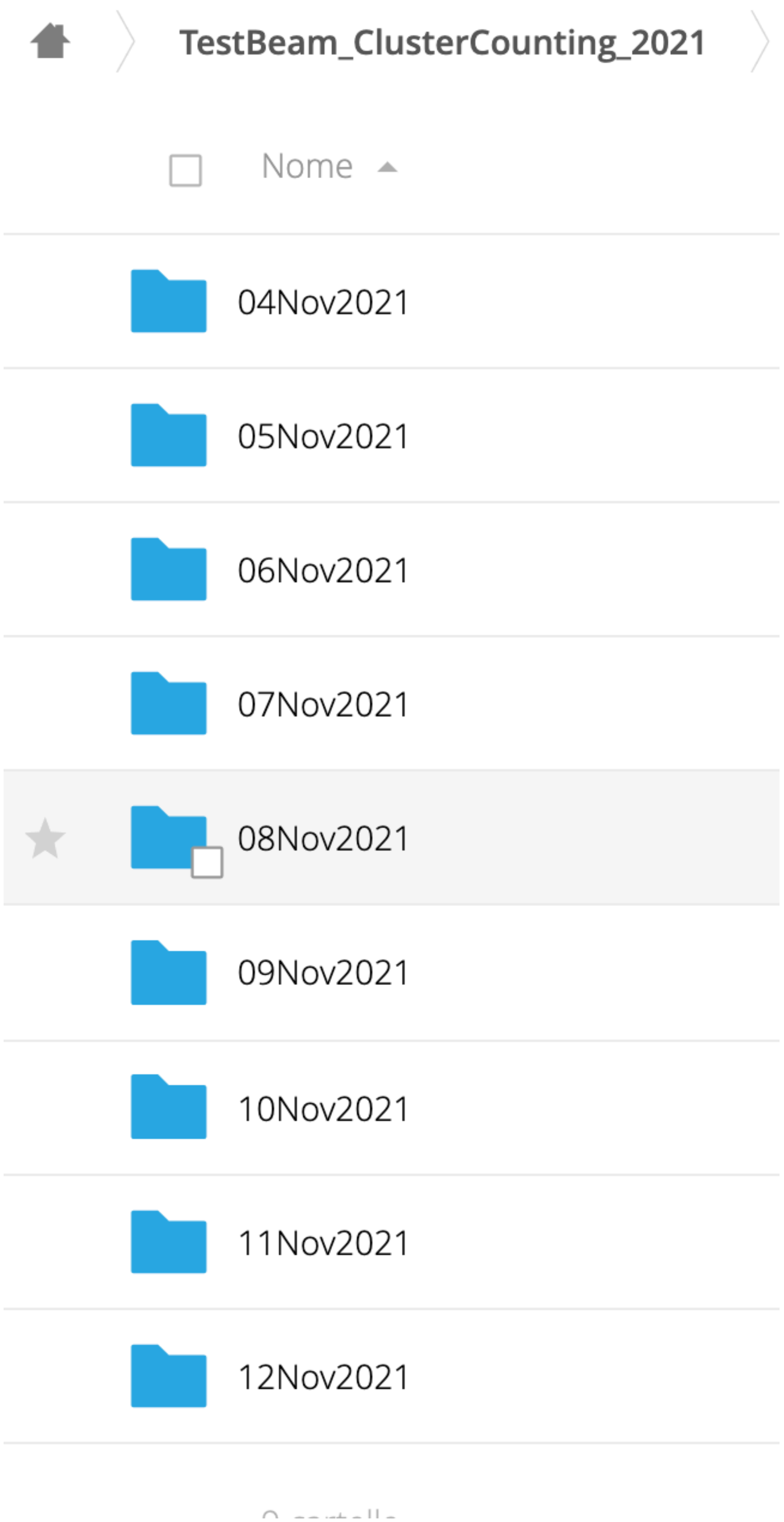
Meeting on cluster counting in drift chambers
25 Nov 2021

Useful links

- Raw data stored on CERNBox: https://cernbox.cern.ch/index.php/s/lzI6PygC4tx1DCE
  - Public accessible; you can download it
  - Only binary files provided
- Github with online analysis code and preliminary offline code: https://github.com/clacaputo/drifttubes_analysis
  - Code for converting RAW data to ROOT and PKL files is provided in the repository
  - Basic Runs database, based on YAML files, is provided
- Log Book (To be Updated): https://codimd.web.cern.ch/9UX0zxEwRK6vsJ4ilia9BA

CERNBox

- https://cernbox.cern.ch/index.php/s/lzI6PygC4tx1DCE
- One folder for each day of data taking, each folder should contain:
  - txt file with informations about the run (To be moved in the log book)
  - RAW files (binary),
  - ROOT and pkl files can be easily created with the code provided in my repo (more in next slides)



On data files

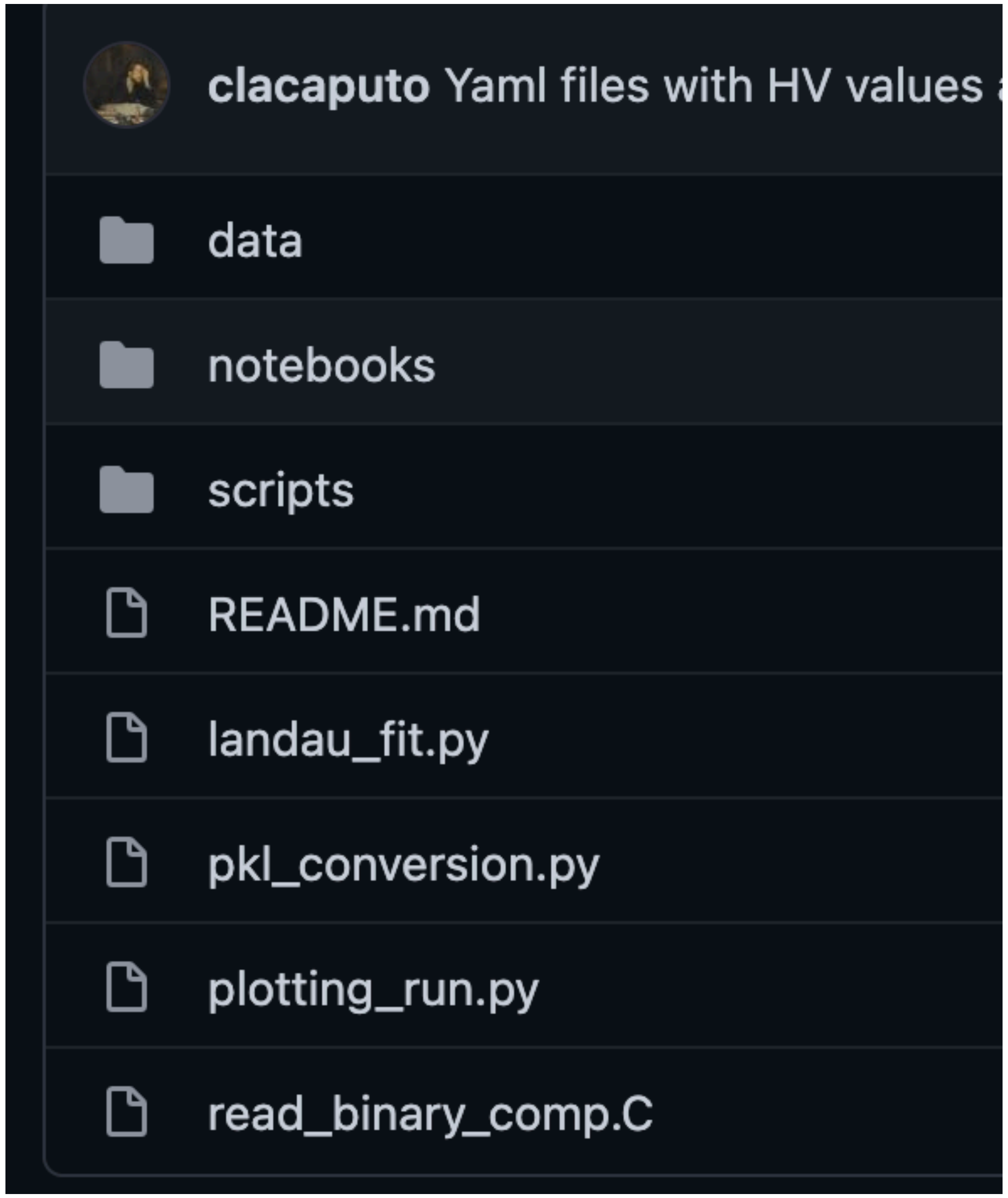
- RAW files (binary) converted to ROOT thanks to a code provided by Gianluigi

- ROOT file contains one TTree per event
  - Each TTree contains a Branch for each channels (15 = 4 Scintillators + 11 tubes). Voltage stored in the Branch
- PKL files (converted by me) stores Pandas DF, easier to manipulate in python, less memory using wrt ROOT
  - Each ROW is an event
  - Each COLUMN is a channel
  - Each Entry is a Array containing the voltage signal

Word	Byte 0	Byte 1	Byte 2	Byte 3	Contents
0	‘D’	‘R’	‘S’	‘8’	File header, Byte 3 = version
1	‘T’	‘I’	‘M’	‘E’	Time Header
2	‘B’	‘#’	Board number		Board serial number
3	‘C’	‘0’	‘0’	‘0’	Channel 0 header
4	Time Bin Width #0				Effective time bin width in ns for channel 0 encoded in 4-Byte floating point format
5	Time Bin Width #1				
…	…				
1027	Time Bin Width #1023				
1028	‘C’	‘0’	‘0’	‘1’	Channel 1 header
1029	Time Bin Width #0				Effective time bin width in ns for channel 1 encoded in 4-Byte floating point format
1030	Time Bin Width #1				
…	…				
2052	Time Bin Width #1023				
2053	‘E’	‘H’	‘D’	‘R’	Event Header
2054	Event Serial Number				Serial number starting with 1
2055	Year		Month		Event date/time 16-bit values
2056	Day		Hour		
2057	Minute		Second		
2058	Millisecond		Range		Range center (RC) in mV
2059	‘B’	‘#’	Board number		Board serial number
2060	‘C’	‘0’	‘0’	‘0’	Channel 0 header
2061	Scaler #1				Scaler for channel 0 in Hz
2062	‘T’	‘#’	Trigger cell		Channel 0 first readout cell
2063	Voltage Bin #0		Voltage Bin #1		Channel 0 waveform data encoded in 2-Byte integers. 0=RC-0.5V and 65535=RC+0.5V. RC see header.
2064	Voltage Bin #2		Voltage Bin #3		
…	…		…		
2574	Voltage Bin #1022		Voltage Bin #1023		
2575	‘C’	‘0’	‘0’	‘1’	Channel 1 header
2576	Scaler #2				Scaler for channel 1 in Hz
2077	‘T’	‘#’	Trigger cell		Channel 1 first readout cell
2578	Voltage Bin #0		Voltage Bin #1		Channel 1 waveform data encoded in 2-Byte integers. 0=RC-0.5V and 65535=RC+0.5V. RC see header.
2579	Voltage Bin #2		Voltage Bin #3		
…	…		…		
3089	Voltage Bin #1022		Voltage Bin #1023		
3090	‘E’	‘H’	‘D’	‘R’	Next Event Header
…					

GitHub REPO

- Github: https://github.com/clacaputo/drifttubes_analysis
  - YAML config files for the different runs
  - Decoding code (Thanks to Gianluigi!)
  - Conversion code to ROOT, pkl files, parallellized on HTCondor
  - script for submitting on HPC facilities, easily customisable
  - code for online Landau fit, plus plot productions
  - Peak finding (BETA version)



GitHub REPO - YAML

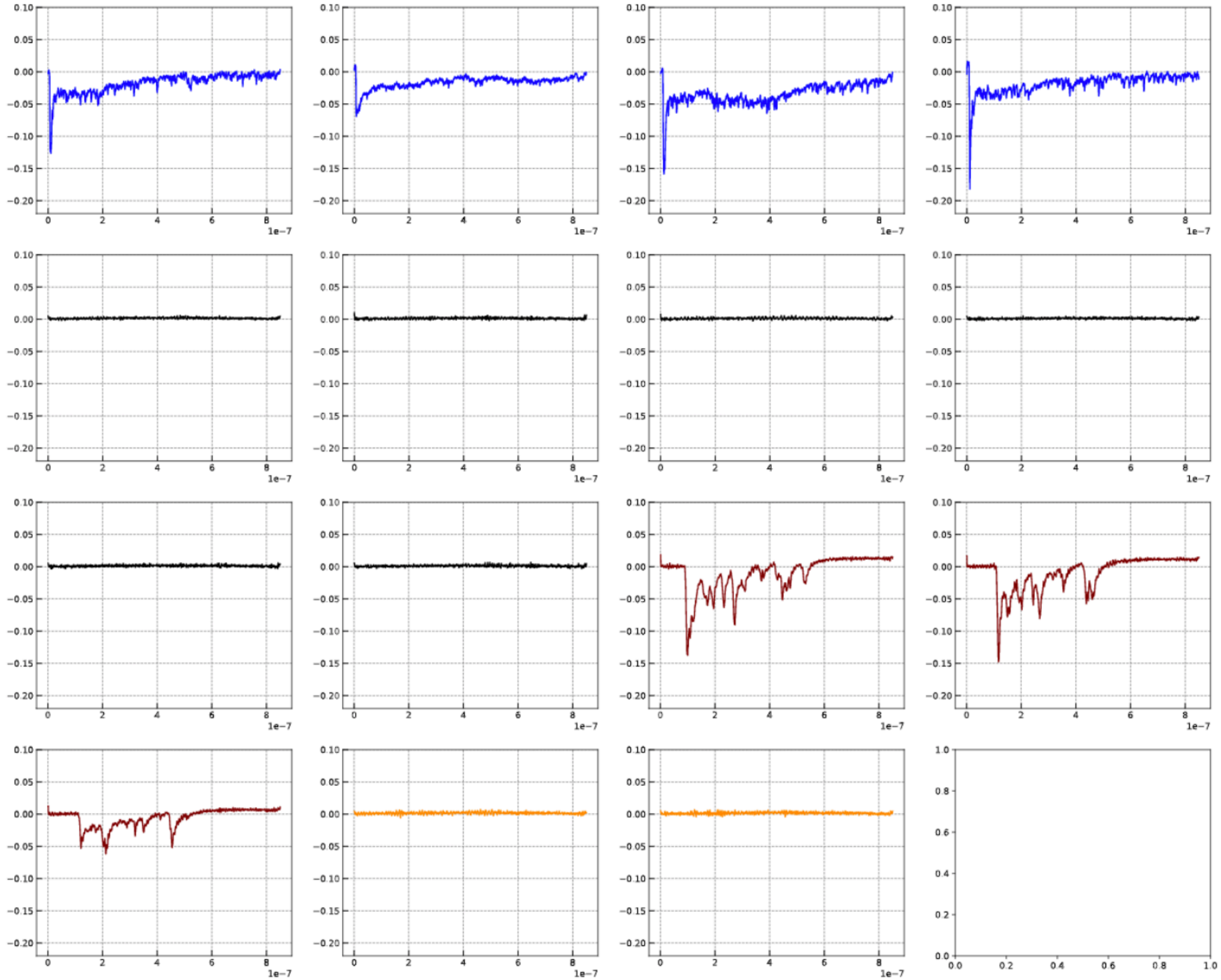
- YAML config files for the different runs, example at this link

78 lines (74 sloc) | 3.27 KB

Raw Blame


```yaml
GasMixture: 90.10
MuonEnergy: 165 #GeV
GSPS: 1.2
delay: 725 #ns
NumberEvents: 5000
HV:
  nominal: {"ch4": 1200, "ch5": 1245, "ch6": 1300, "ch7": 1300, "ch8": 1340, "ch9": 1340, "ch10": 1495, "ch11": 1550, "ch12": 1720, "ch13": 1670, "ch14": 1810}
main_path: /eos/user/c/ccaputo/TestBeam_ClusterCounting/11Nov/
Measurements:
  Voltage_m20:
    voltage: "m20"
    angle_scan:
      angle_15:
        file_bin: "11Nov_15angle_HVnominalMinus20_1p2GSPS_5k"
        file_root: "11Nov_15angle_HVnominalMinus20_1p2GSPS_5k.root"
        file_pkl: "11Nov_15angle_HVnominalMinus20_1p2GSPS_5k.pkl"
      angle_30:
        file_bin: "11Nov_30angle_HVnominalMinus20_1p2GSPS_5k"
        file_root: "11Nov_30angle_HVnominalMinus20_1p2GSPS_5k.root"
        file_pkl: "11Nov_30angle_HVnominalMinus20_1p2GSPS_5k.pkl"
      angle_45:
        file_bin: "11Nov_45angle_HVnominalMinus20_1p2GSPS_5k"
        file_root: "11Nov_45angle_HVnominalMinus20_1p2GSPS_5k.root"
        file_pkl: "11Nov_45angle_HVnominalMinus20_1p2GSPS_5k.pkl"
      angle_60:
        file_bin: "11Nov_60angle_HVnominalMinus20_1p2GSPS_10k"
        file_root: "11Nov_60angle_HVnominalMinus20_1p2GSPS_10k.root"
        file_pkl: "11Nov_60angle_HVnominalMinus20_1p2GSPS_10k.pkl"
  Voltage_nominal:
    voltage: "nominal"
    angle_scan:
      angle_0:
```

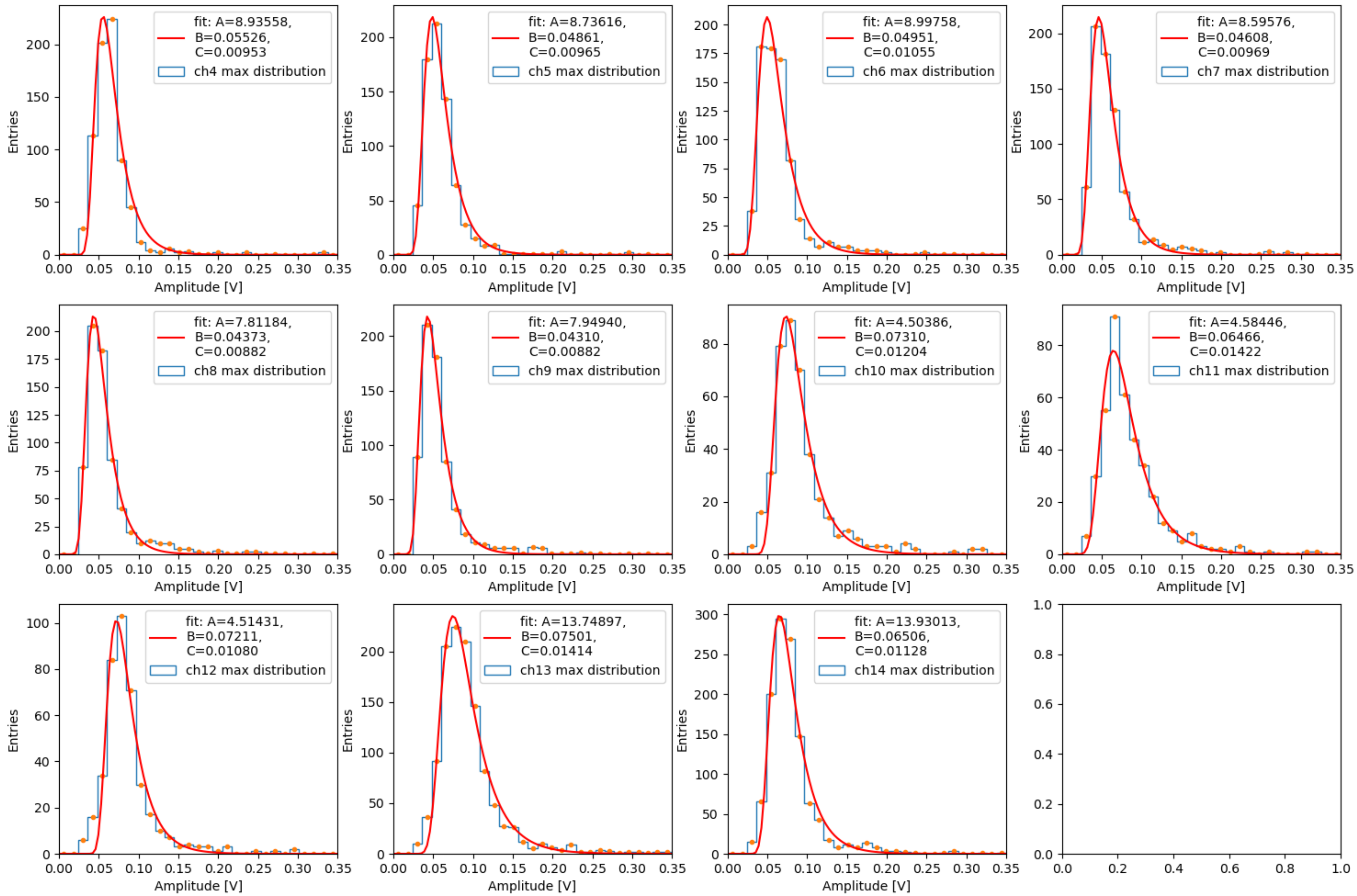
Signal shape, an example



- More of these plots: https://cernbox.cern.ch/index.php/s/yjoJLkgUbPC1ELG

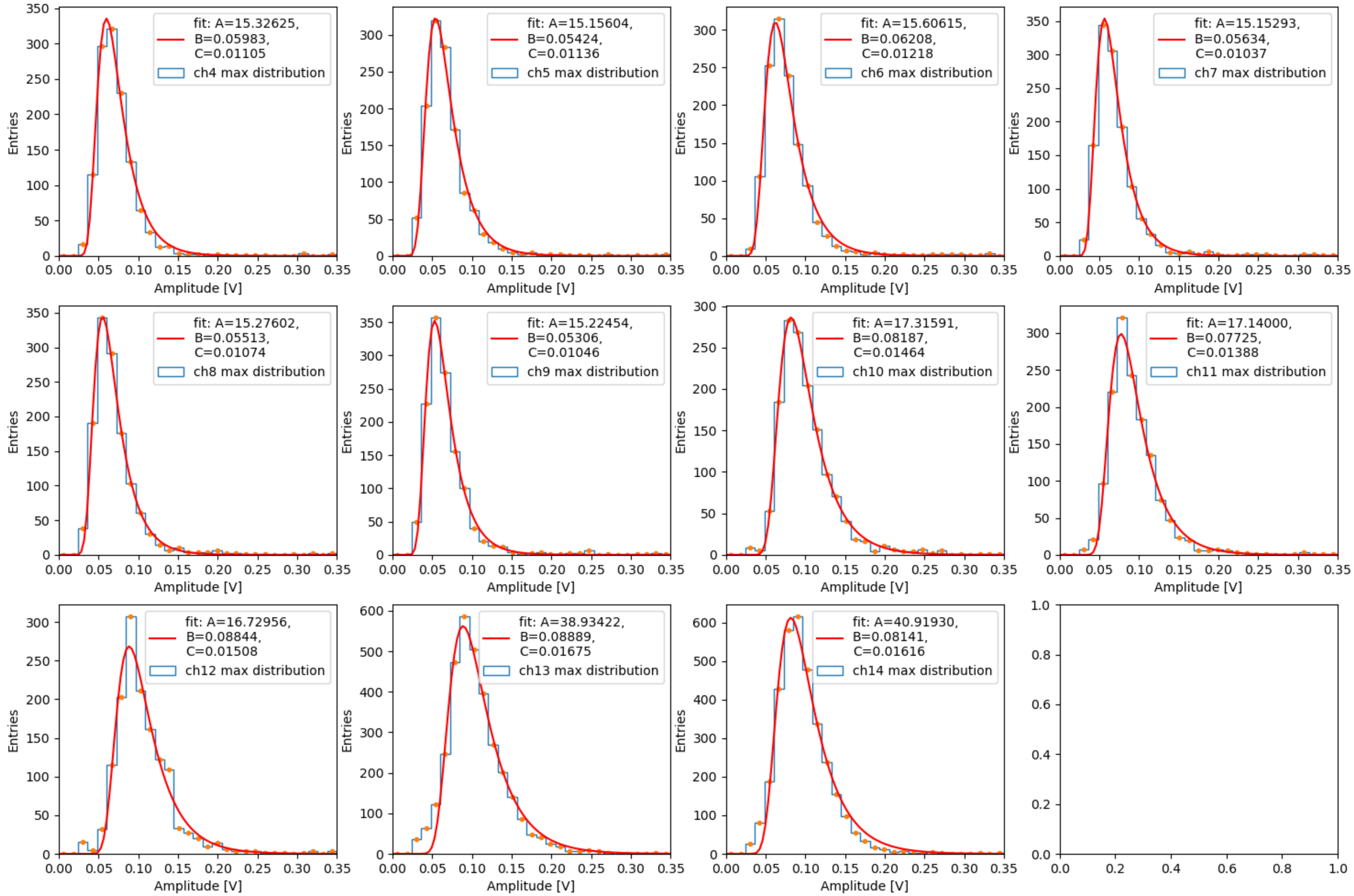
Online Gain measurement

- Max amplitude/ per tube channel distribution, fitted with a landau - **11Nov_0angle_HVnominal_1p2GSPS_5k_LANDAU**
  - https://github.com/clacaputo/drifttubes_analysis/blob/main/landau_fit.py
- More of these plots: https://cernbox.cern.ch/index.php/s/yjoJLkgUbPC1ELG



Online Gain measurement

- Max amplitude/ per channel distribution, fitted with a landau - **11Nov_45angle_HVnominal_1p2GSPS_10k_LANDAU**
  - https://github.com/clacaputo/drifttubes_analysis/blob/main/landau_fit.py
- More of these plots: https://cernbox.cern.ch/index.php/s/yjoJLkgUbPC1ELG



Peak Finding

Two methods checked VERY PRELIMINARY

- Using **Scipy** libraries for signal processing
  - **scipy.signal.find_peaks_cwt**: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks_cwt.html

This approach was designed for finding sharp peaks among noisy data, however with proper parameter selection it should function well for different peak shapes.

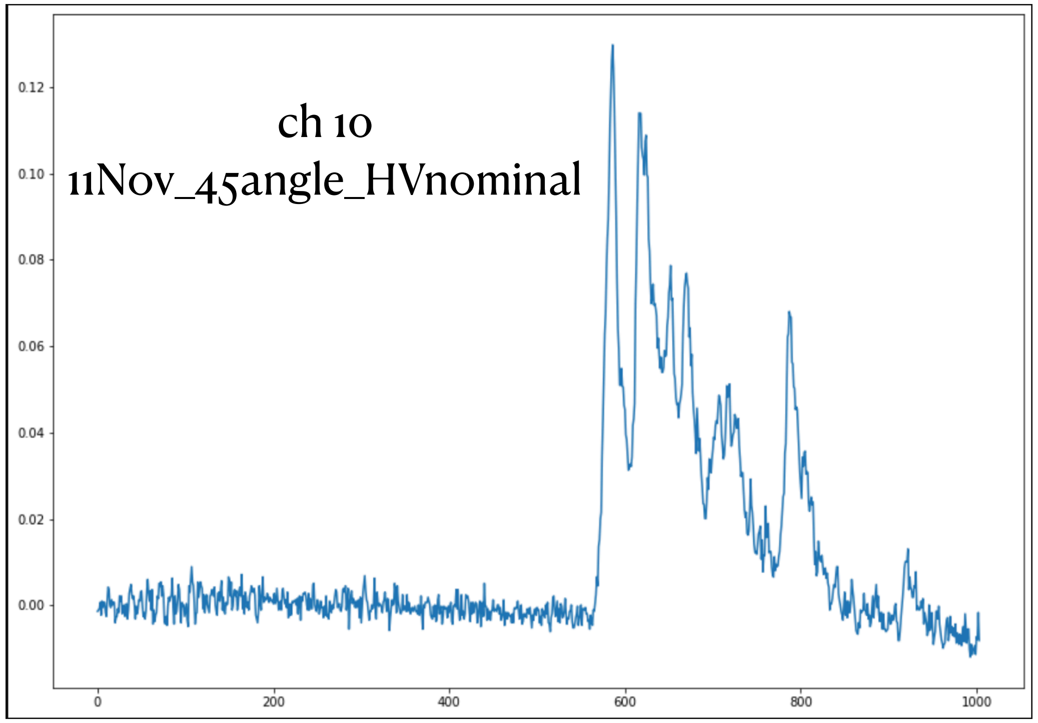
The algorithm is as follows:

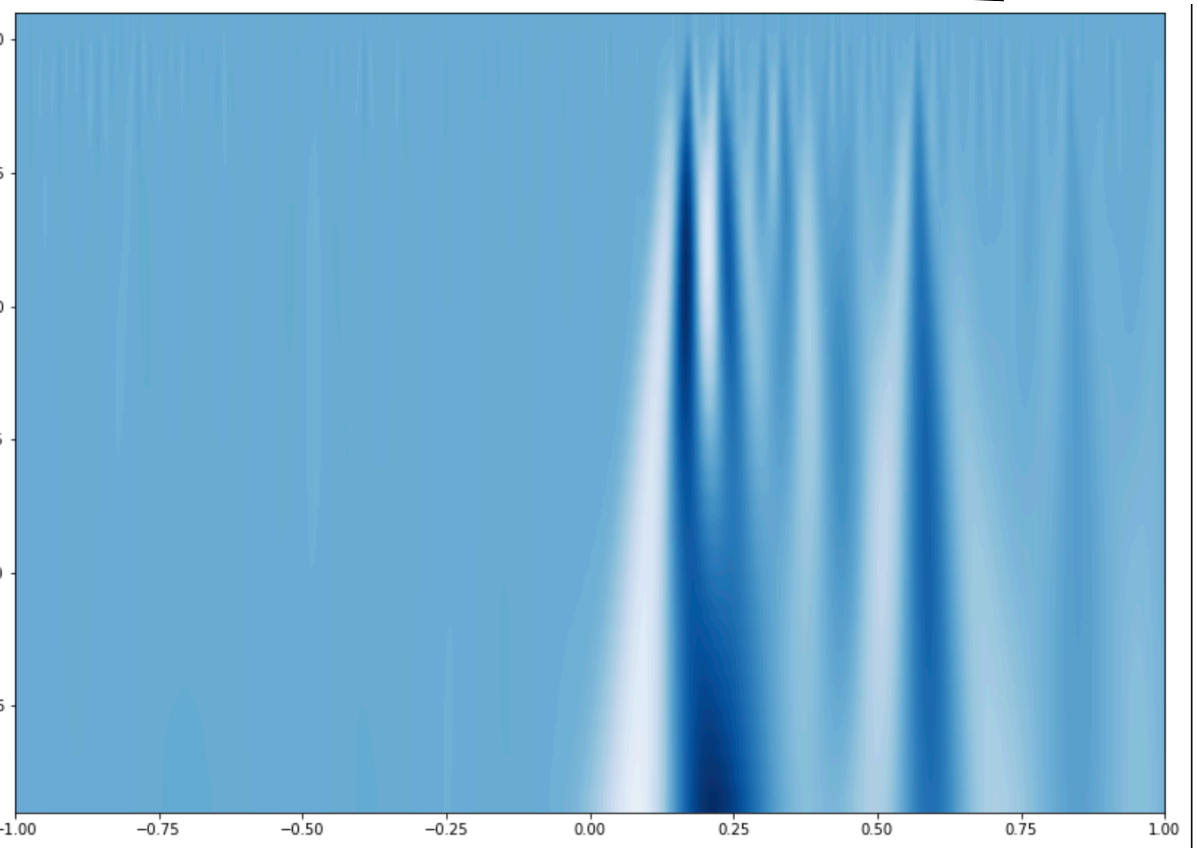
- Perform a continuous wavelet transform on vector, for the supplied widths. This is a convolution of vector with wavelet(width) for each width in widths. See cwt.
- Identify “ridge lines” in the cwt matrix. These are relative maxima at each row, connected across adjacent rows. See identify_ridge_lines
- Filter the ridge_lines using filter_ridge_lines.

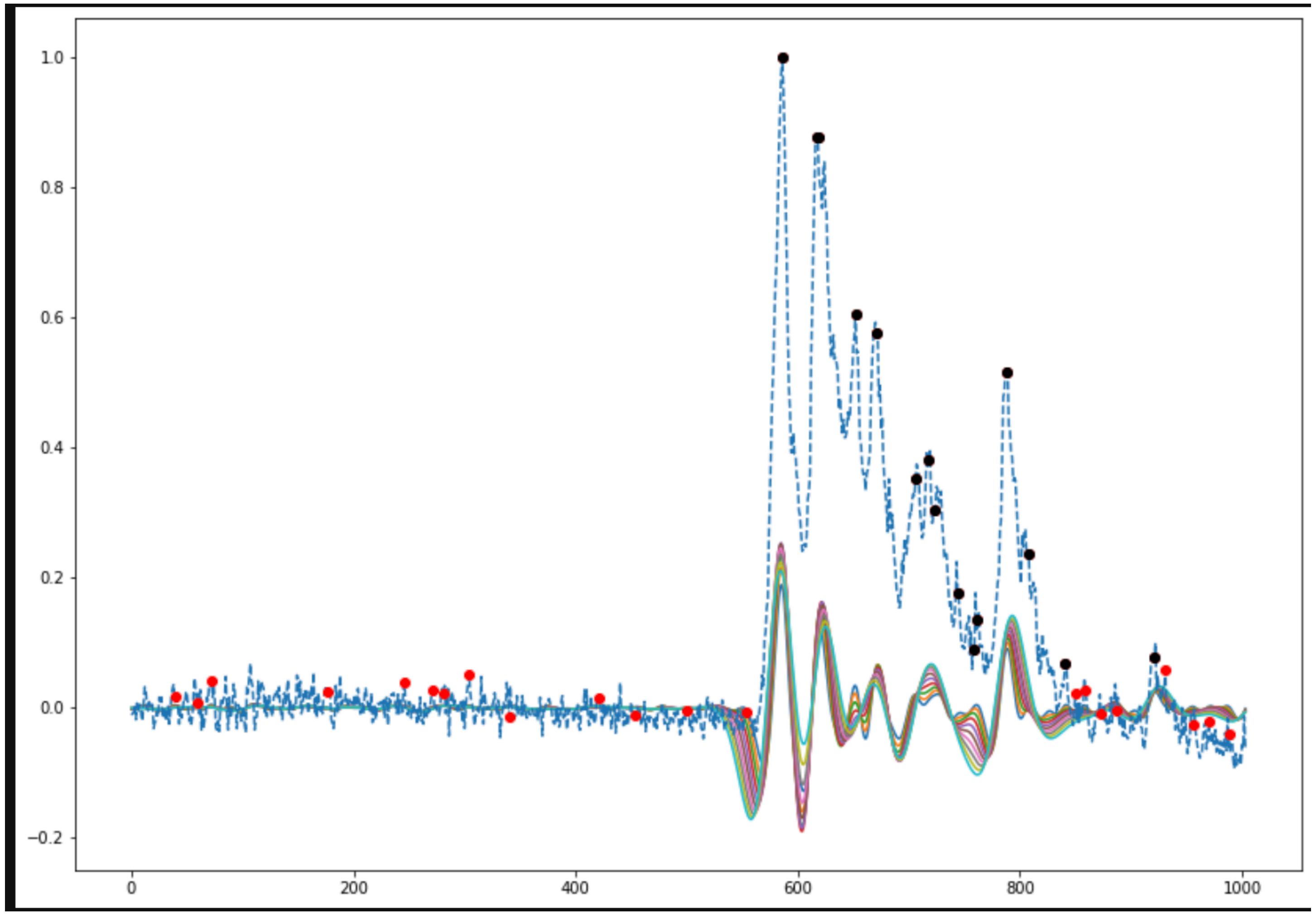
- Second derivative Algo implemented

Peak Finding CWT, an example

- scipy.signal.find_peaks_cwt

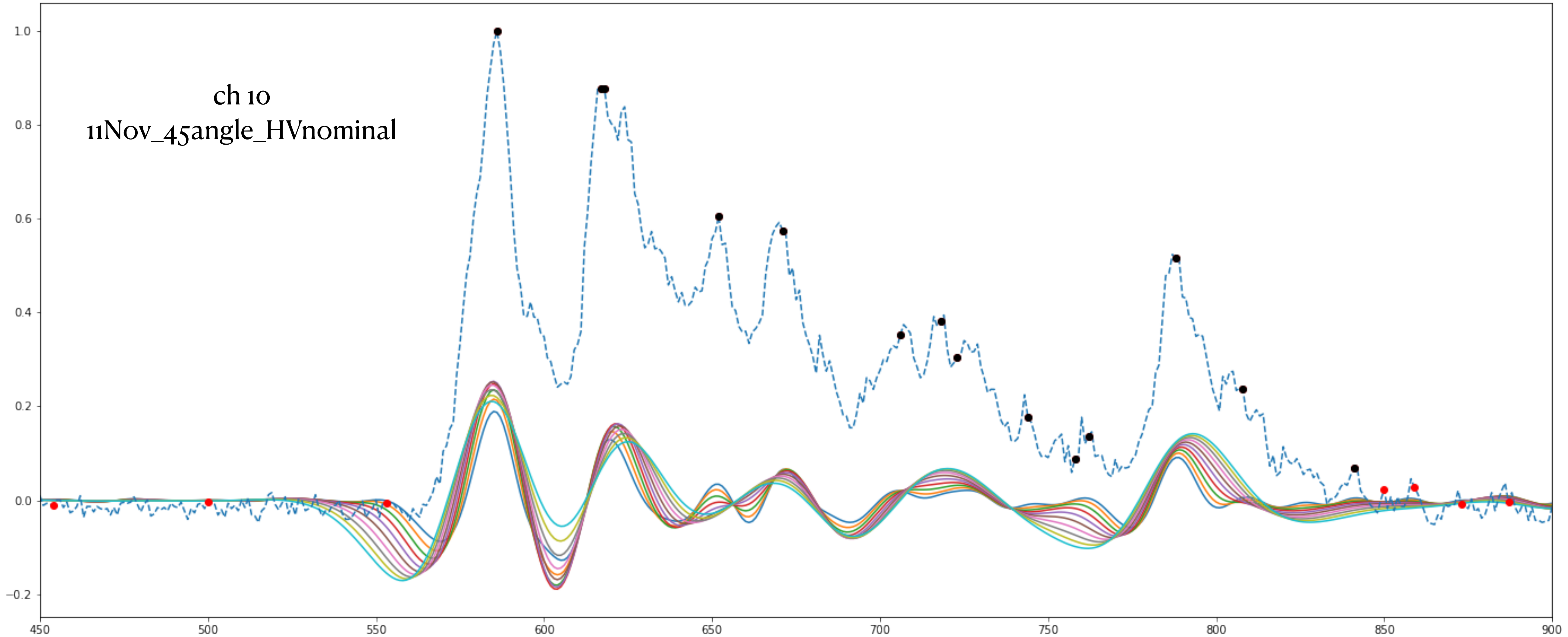






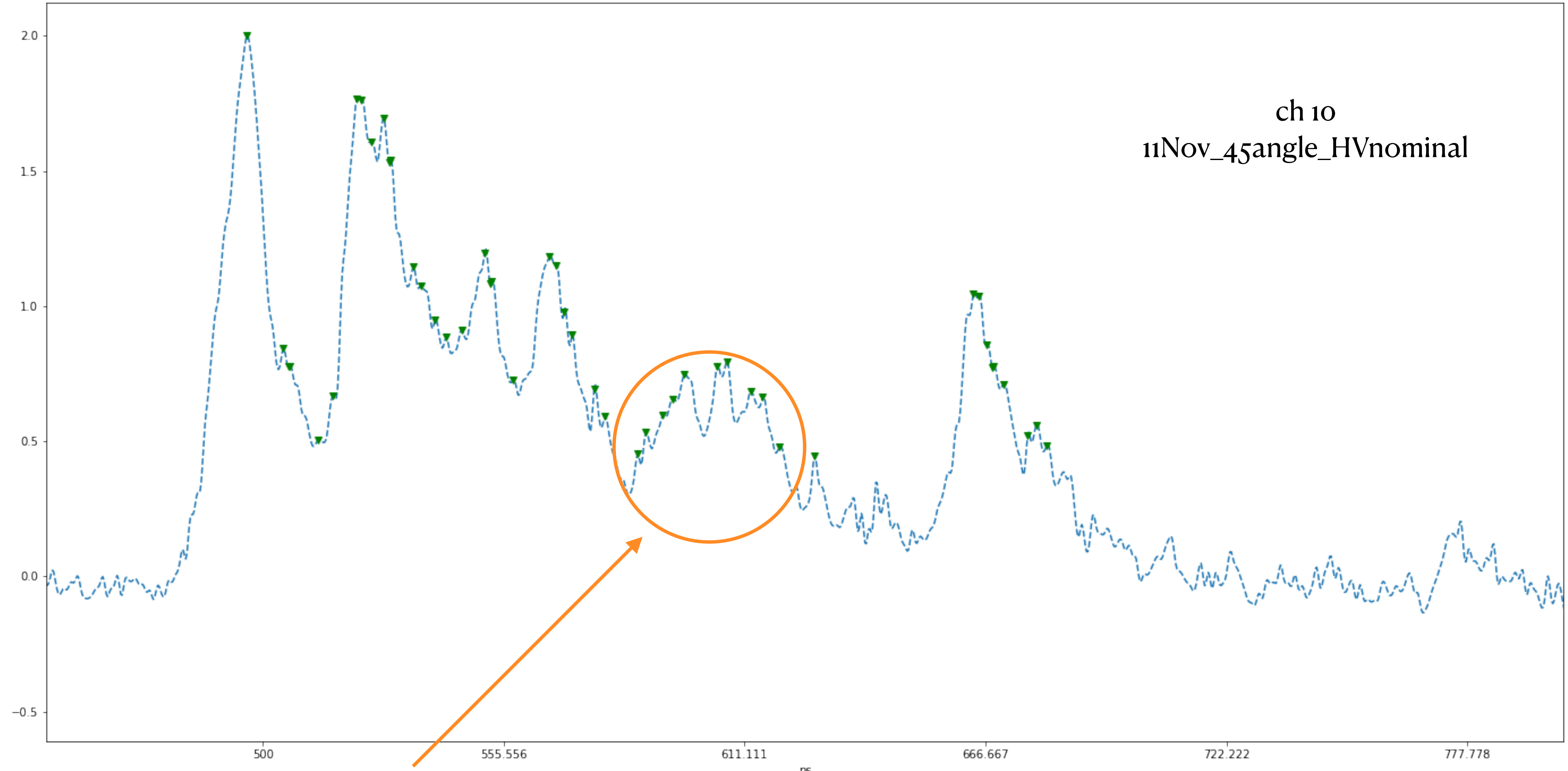
Peak Finding CWT, an example

- **scipy.signal.find_peaks_cwt**
- Zooming on the peaks



Peak Finding II Der, an example

- Implemented a simple algorithm, based on the Second derivative



- Is noise overlapping our signal?
- If so, could we filter it?

Conclusion and outlook

- Data available in RAW format on CERNBox
  - https://cernbox.cern.ch/index.php/s/lzI6PygC4tx1DCE
  - Can be transferred to local clusters
  - Code for conversion and a database has been provided and can be found here: https://github.com/clacaputo/drifttubes_analysis
- Online code for gain measurement
- Preliminary peak finding analysis in place
  - Still in beta
  - to be compared with other algorithms