

Overview

NPPS

David Adams

BNL

October 11, 2019

Introduction (to me)

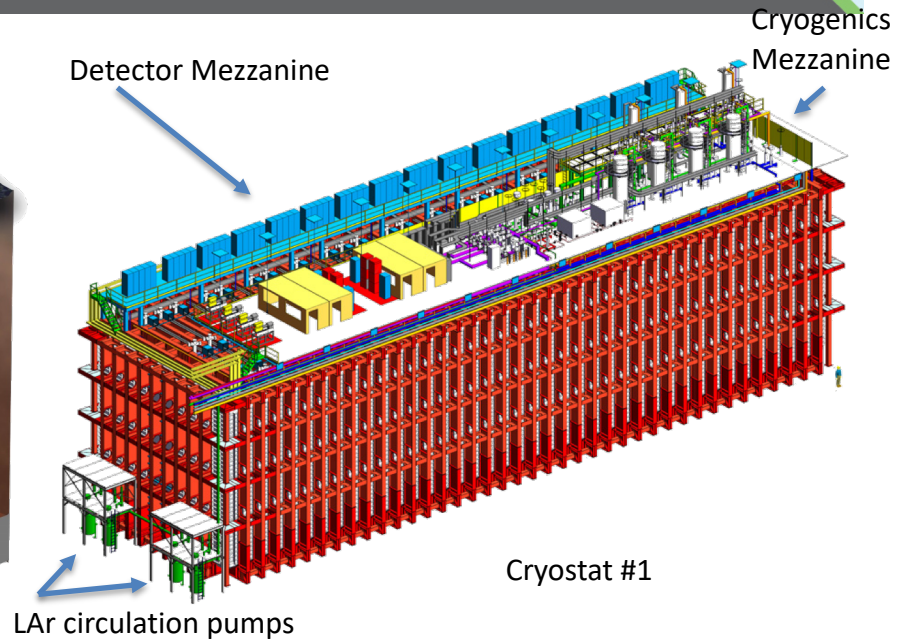
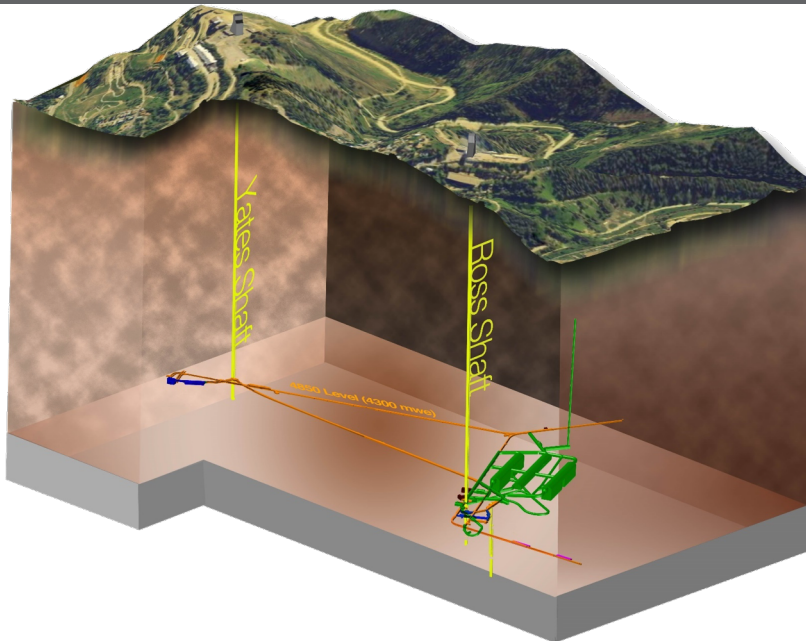
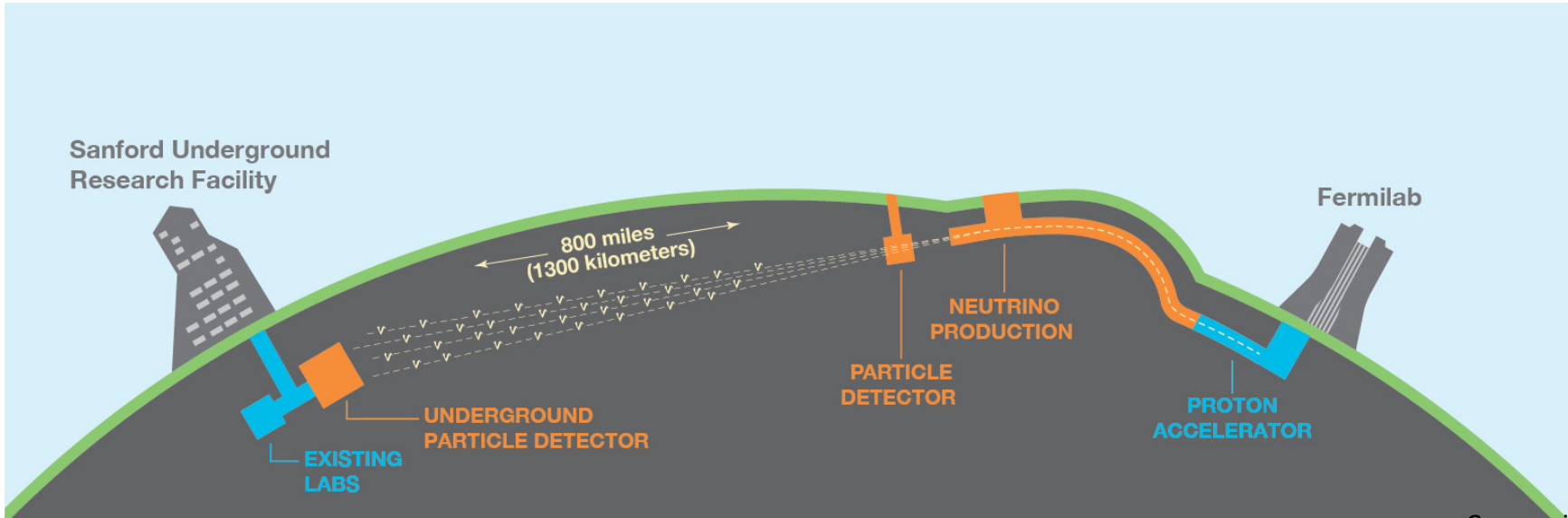
History

- D0: Track finding/reco, SW build infrastructure
- ATLAS: SW infrastructure, muon reconstruction, jet finding software, exotics searches: ($W' \rightarrow lv$, $X \rightarrow (jj)(jj)$)

Current: DUNE

- SW infrastructure
 - Algorithm encapsulation and frameworks
 - Offline (and beyond) SW packaging
 - Weekly release builds
- LAr TPC signal reconstruction
 - Sticky code mitigation
 - Noise removal
 - Performance metrics
 - Monitoring and event display

DUNE overview



Algorithm encapsulation

Where to put the code?

- Tradition: art framework calls *modules* that pull input from event data store and write output back
- Many lines of code and so difficult to understand, maintain and configure (run time specification of parameters)
- New ideas → add more code and configuration
- Module interface is art specific and very difficult to use or test outside the art framework
- In practice many copies of code: new collaboration or detector within a collaboration started by copying earlier modules

Better approach

- Divide and conquer with *tools*
- Divide module code into steps implemented in tools that can be swapped around

Tools

Art tool

- C++ class with dynamic lookup and fcl configuration
 - Fcl is fairly simple configuration language (param: value)
- Typically inherit from interface so client has no compile or link time dependency on the tool class
- Minimal intrusion/dependency on framework
 - Class needs on-line macro to declare name for dynamic lookup
 - And ctor with FCL as argument
 - Easy to imagine adding another ctor for a new configuration language
 - No inheritance from generic tool base class (type safety issue)
 - Easy to use outside the art framework (standalone exe, Root script, ...)

Tool manager (DUNE extension)

- Map tool configurations to names
 - Configuration = tool class name plus values for parameters
- Referencing tools by name instead of full configuration simplifies the configuration of their clients
- And allows multiple clients to share (or not) a tool instance

Dataprep

Primary example of tool use is DUNE data preparation

- Input is (decoded) raw data
 - Array of ADC samples for many channels
 - ProtoDUNE is typically 6k samples for 15k channels
- Output is processed ROI waveforms
 - E.g. arrays of ~ 100 float samples
- Processing done in a series of steps that act on the samples
 - Identifying and mitigating problems
 - Removing noise
 - Deconvoluting with response function
 - Finding ROIs
- Each step is a tool instance with interface that takes one or a collection of channels
 - Multiple channels are useful for removing correlated noise

Example configuration of the dataprep service

```
services.RawDigitPrepService: {  
  AdcChannelToolNames: [  
    "digitReader",  
    "pd_adcPedestalFit",  
    "adcSampleCalibration",  
    "pdsp_sticky_codes_ped",  
    "pdsp_adcMitigate",  
    "adcTailRemovalKe",  
    "pdsp_noiseRemovalKe",  
    "pedmetSignalFinderKe"]  
  DoWires: "true"  
  LogLevel: 3  
  service_provider: "ToolBasedRawDigitPrepService"  
}
```

Processing tool names.

Display and monitoring tools can be inserted between steps to see progress in cleaning signal

Class name.

Top level of dataprep is implemented as *service*.

Could/should be a tool?

Example tool configuration

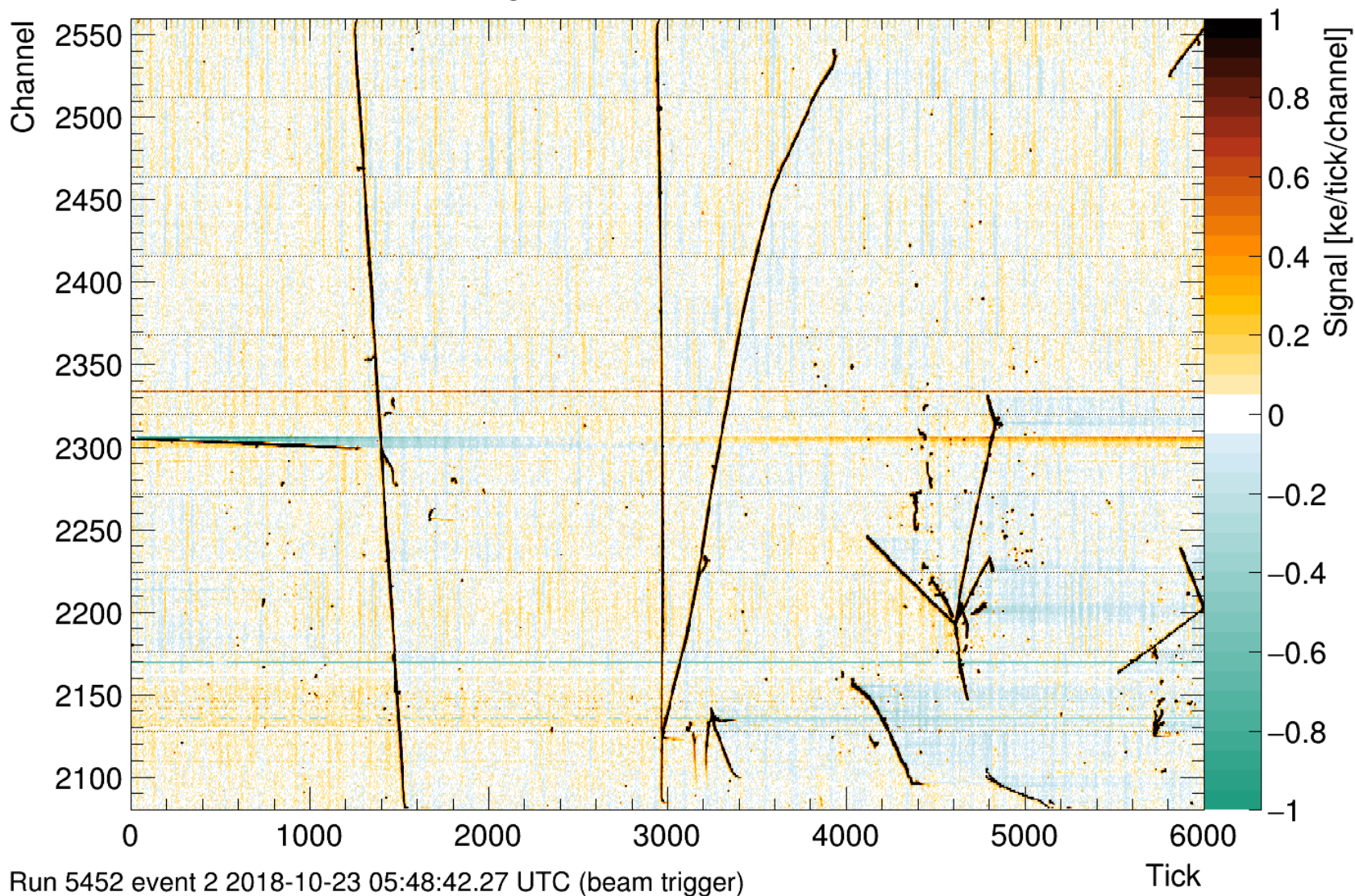
```
tools.pedmetSignalFinderKe: {  
  BinsAfter: 20  
  BinsBefore: 10  
  FlagNegative: "true"  
  FlagPositive: "true"  
  LogLevel: 1  
  MaxLoop: 20  
  SigFracMax: 0.8  
  ThresholdMin: 0.5  
  ThresholdRatio: 5  
  ThresholdRatioTol: 0.1  
  tool_type: "AdcNoiseSignalFinder"  
}
```

Class name



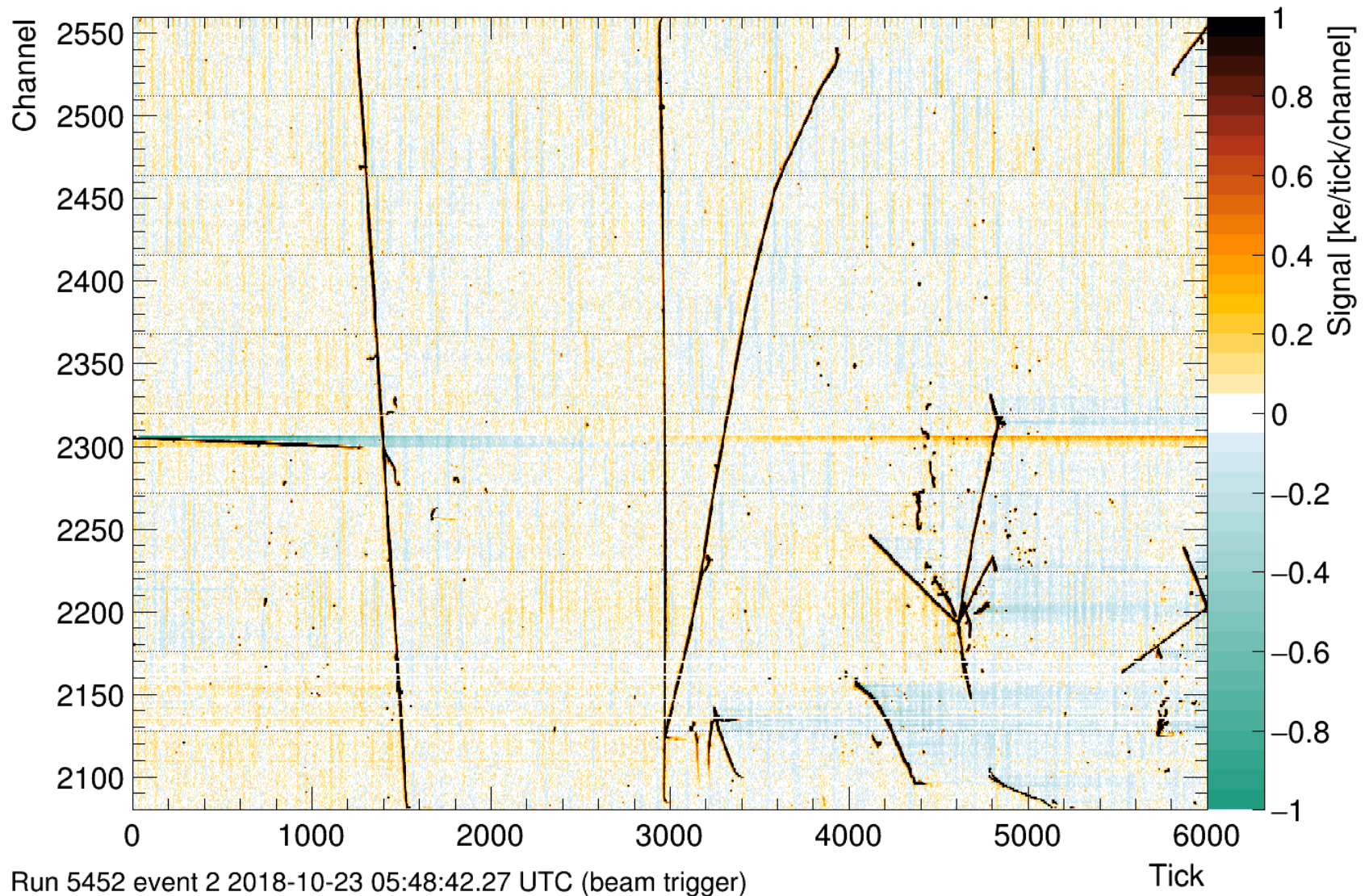
Example data: calibrated raw - pedestal

Charge after calibration for APA 3z



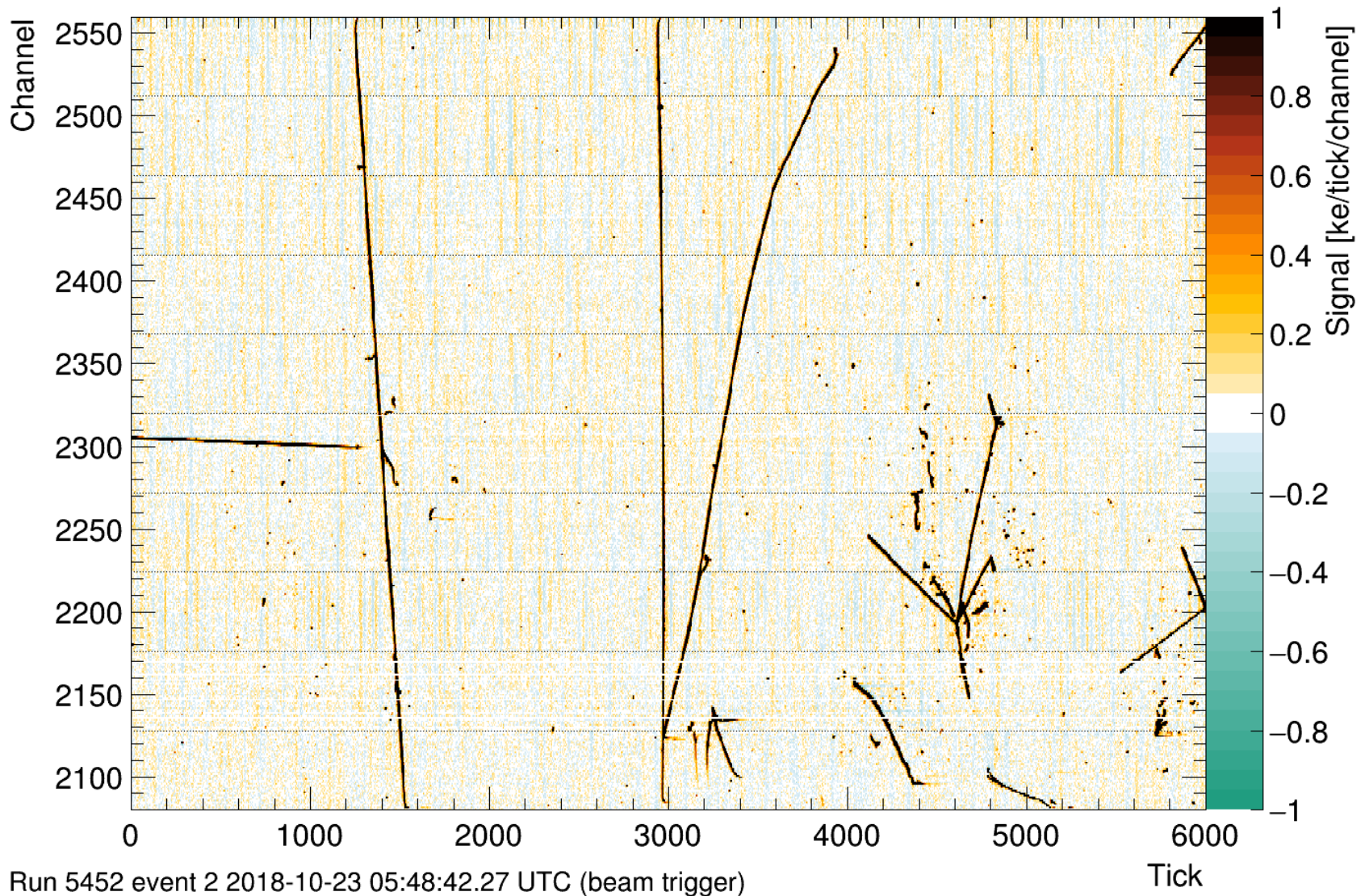
Plus sticky code mitigation, bad channel removal

Charge after mitigation for APA 3z



Plus tail removal (tails from AC coupling)

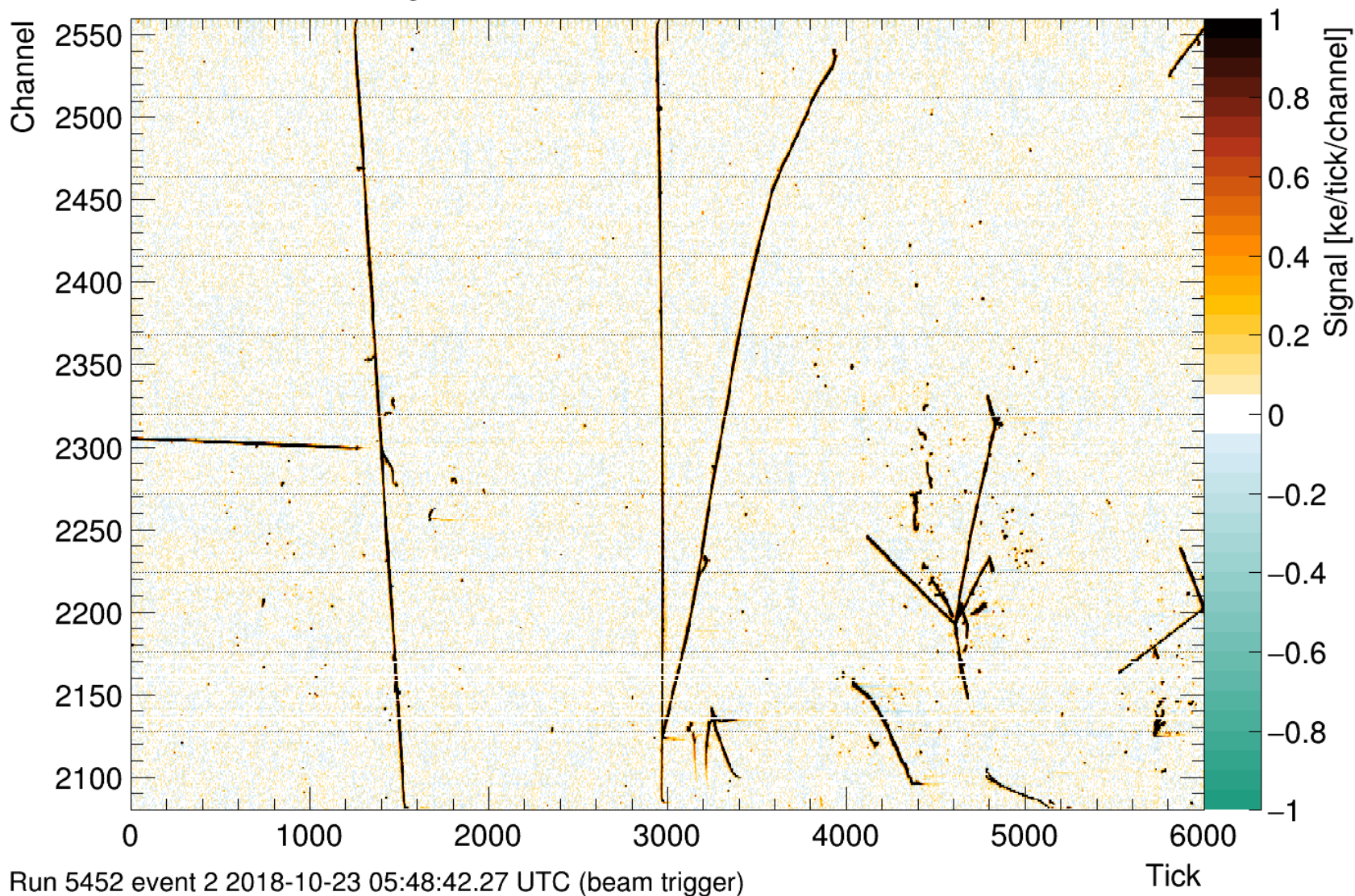
Charge after tail removal for APA 3z



Run 5452 event 2 2018-10-23 05:48:42.27 UTC (beam trigger)

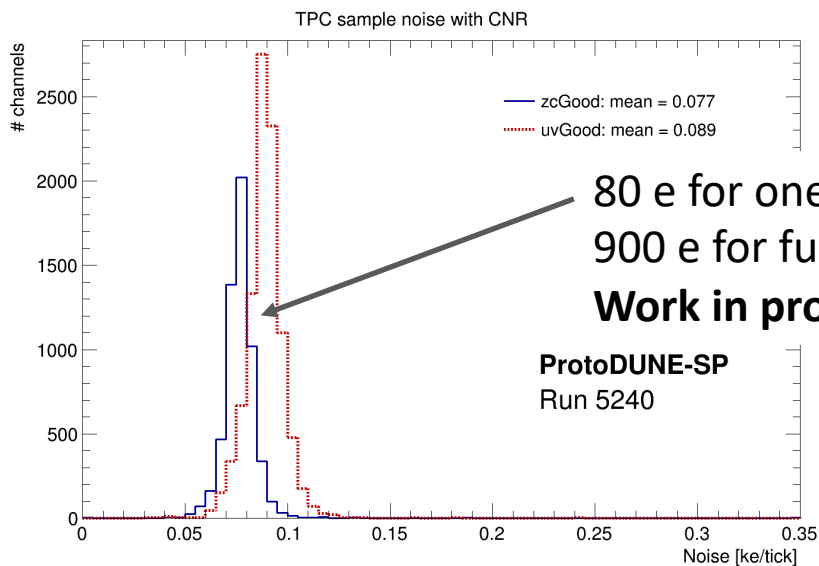
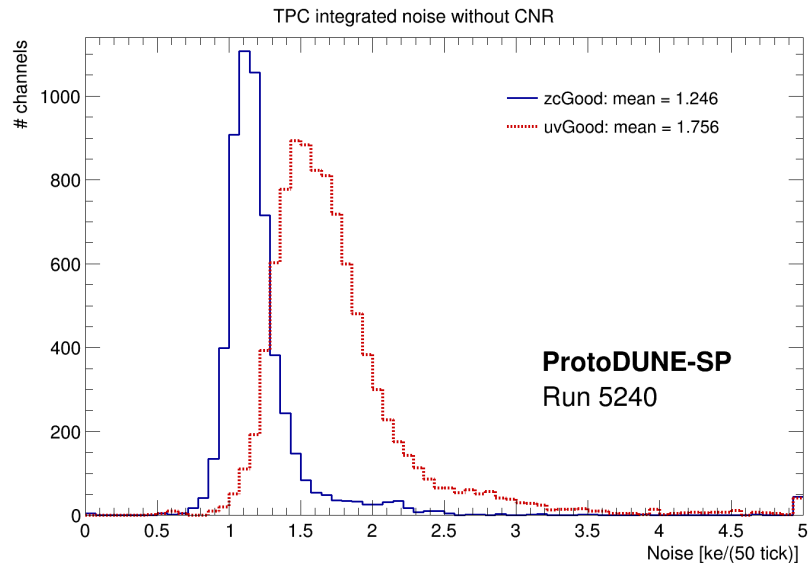
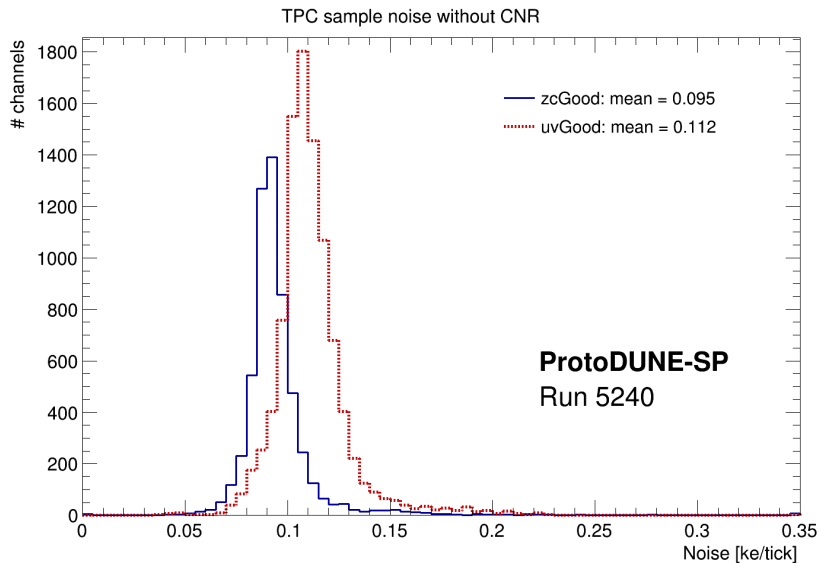
Pus channel-correlated noise removal

Charge after correlated noise removal for APA 3z

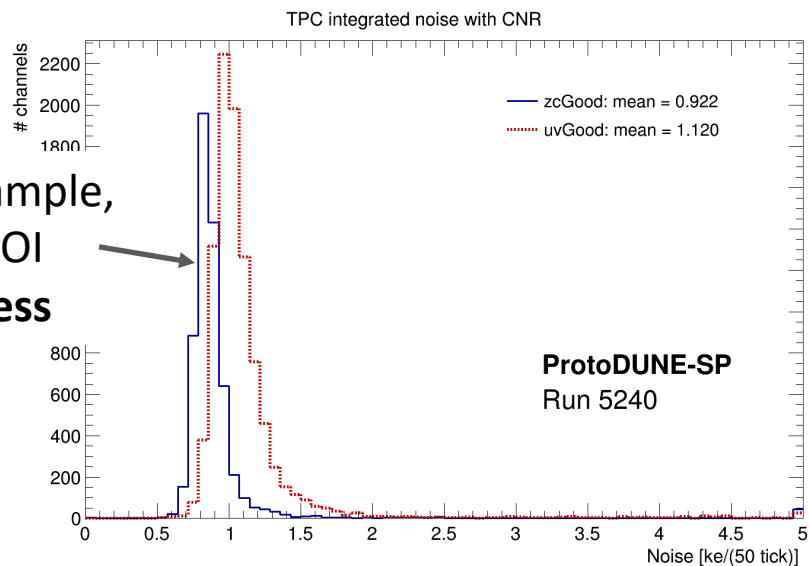


Run 5452 event 2 2018-10-23 05:48:42.27 UTC (beam trigger)

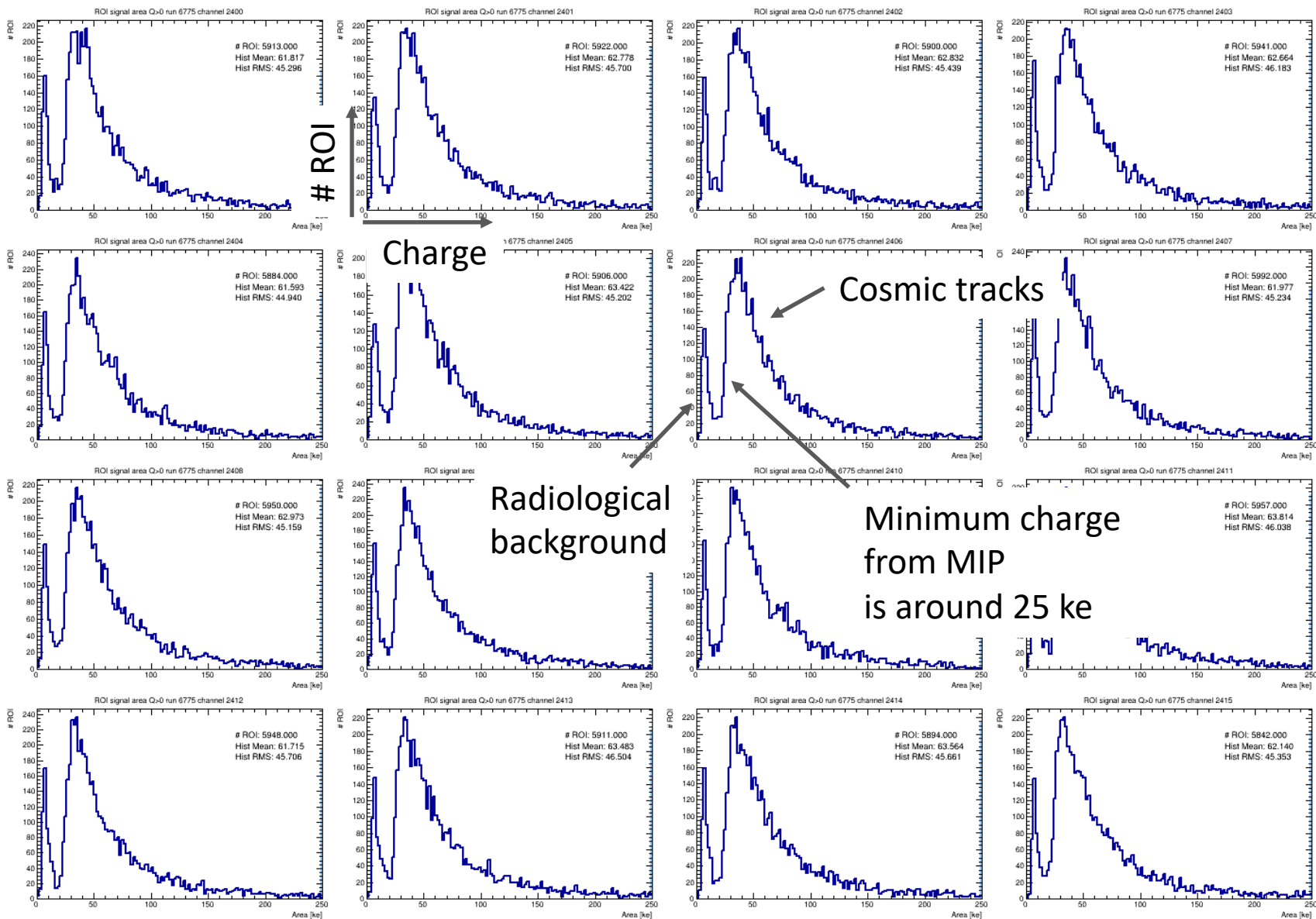
Example from resolution study



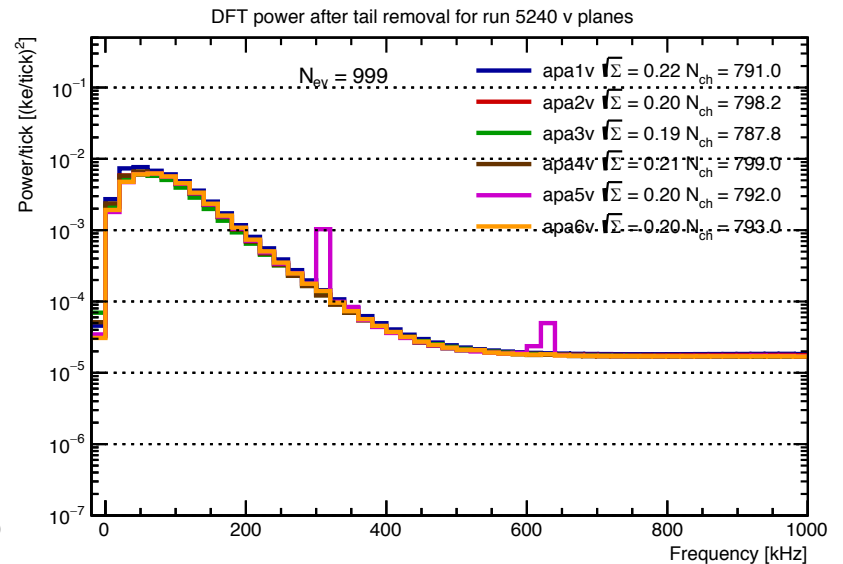
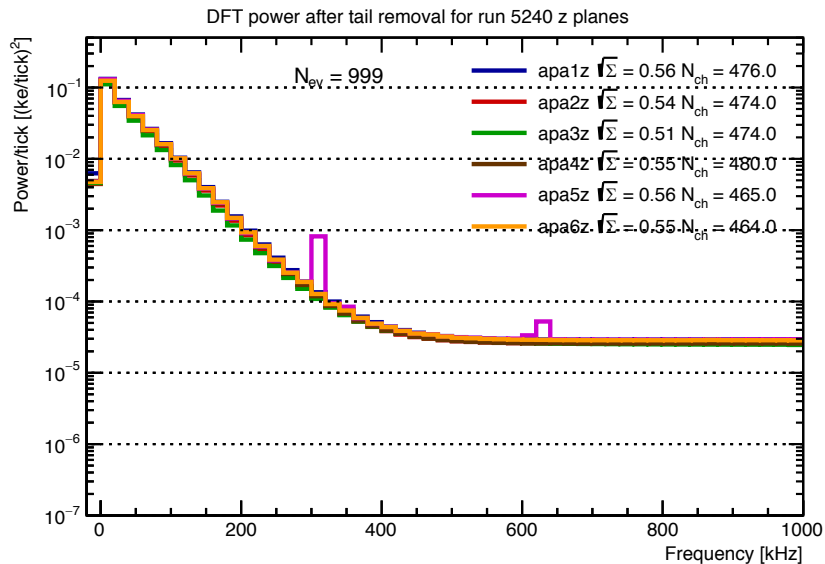
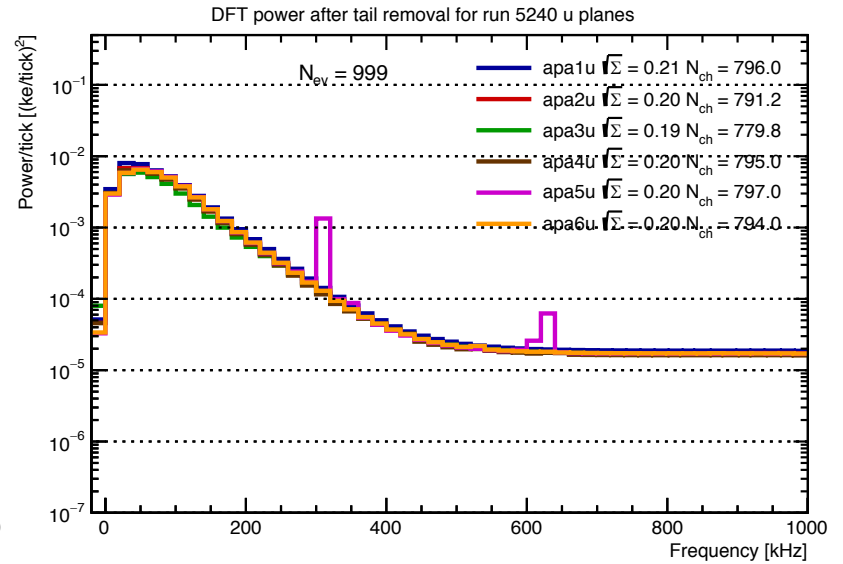
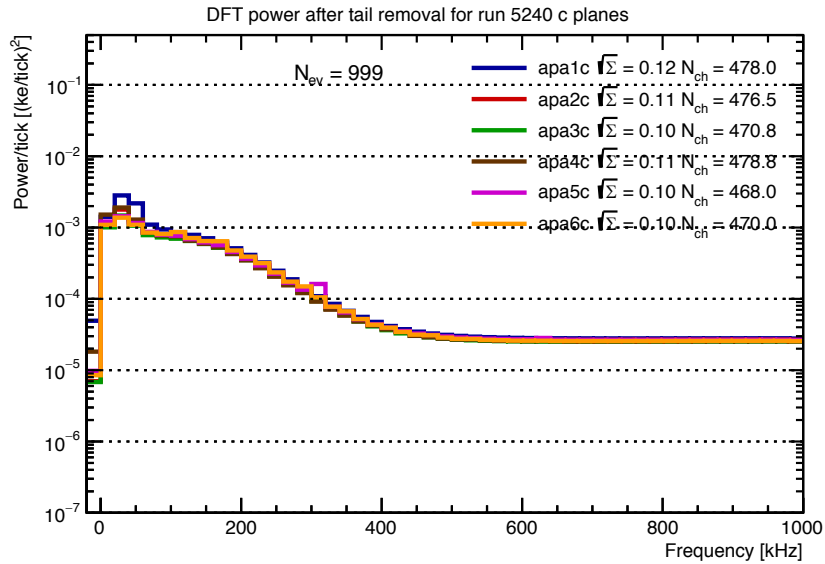
80 e for one sample,
900 e for full ROI
Work in progress



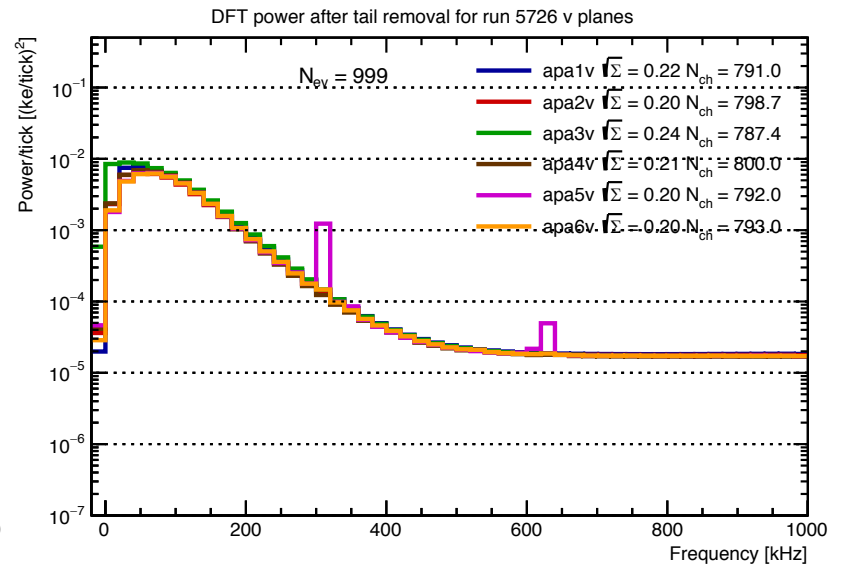
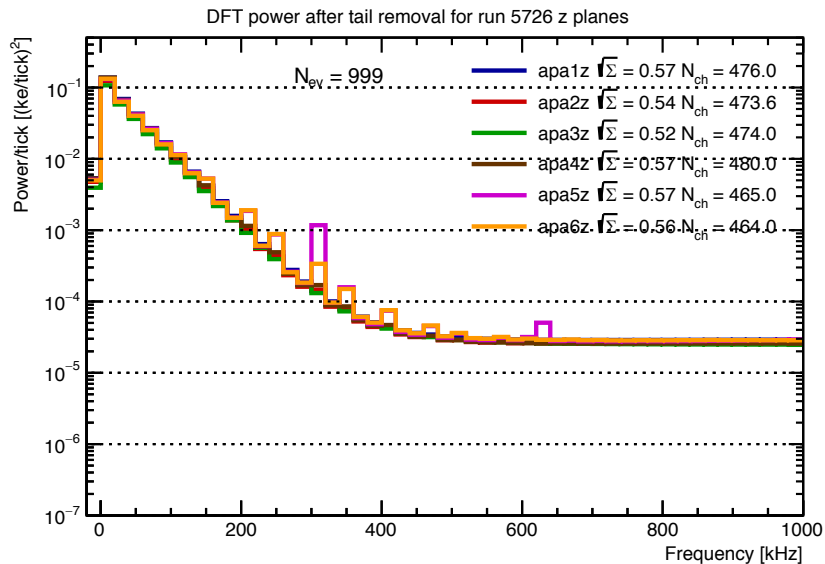
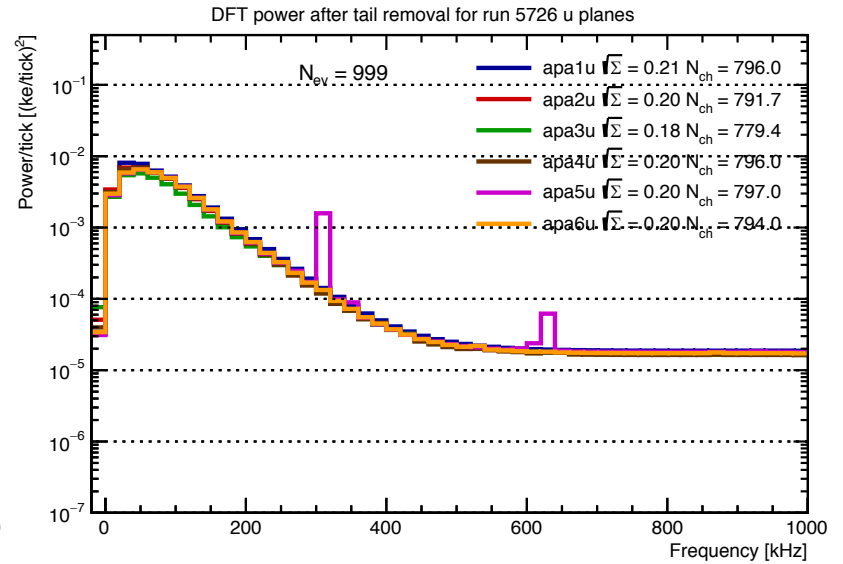
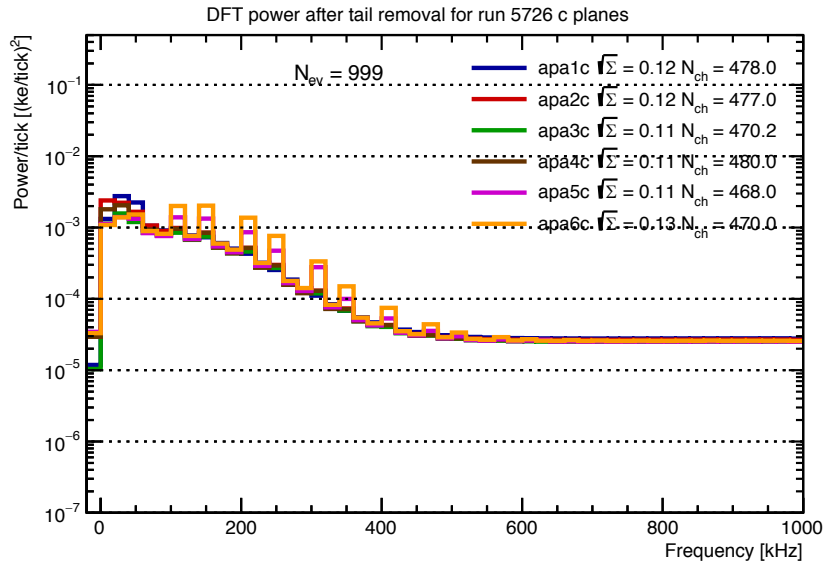
ROI collected charge (calibrated)



DFT for run 5240



DFT for run 5726



Packaging

At present, most DUNE SW in one package

- Offline sim, reco, analysis
- Discussion underway to split this up
 - Analysis SW has been moved into a higher level package
- Avoid very long time for developers to compile their contributions
- I will likely contribute significantly here

New dataprep module

TPC decoder

- Decoding TPC data uses much memory
- Copy large block of data from Root to event store to transient representation used in data processing
- Already have option to process one APA at a time in dataprep
 - ProtoDUNE has 6 APAs
 - DUNE will have many more
- New tool-based decoder tool (instead of module) provides the option to also decode one APA at a time
 - And not put data in event store
- Dataprep being modified (by me) to use this tool in this way
- Data be restructured to facilitate this
 - Data for each APA written to a separate container
 - Expect this in December

Tracking

Torre pointed out new tracking package ACTS

- I looked at one paper and the approach looks good
- I.e. very similar design to what we implanted for D0 run II
- DUNE already has couple solutions here
- But something I may look at later...

Performance

- Independent of the above, I am interested in looking at current track reco performance
- And then using track info to predict charge deposits
 - I.e. like truth info in MC
 - If track has known t -zero and end point in detector volume, we can know dE/dx along the track plus energy of Michel electron
 - Tracks from beam have known energy

The event

Traditional HEP data processing based on events

- Each event processed independently
- Art framework assumes this
- DUNE plane for event is typically full detector for 6k or 10k time samples (3-5 ms)
 - Call this big event
- Can also trigger on and record data around ROIs
 - Small events

But model is breaking down

- Supernova will emit neutrinos over 10s of seconds
- If events are big, we will want to split them to separate overlapping signals and merge if track crossed from one to the next
- Small events will have to be merged

Don't let offline processing model restrict DAQ

- Want maximum exposure time

Comments/conclusions

I am keeping busy with DUNE

- Software/computing design
- Understanding TPC data and helping to develop tools for signal processing