

New noise model for DUNE simulations

B. Abi
P. Lasorak

- New noise model includes
 - Wire noise
 - Coherent noise
 - Digitisation noise
- All of which are consistent with ProtoDUNE data
 - We use 3 types of ProtoDUNE data:
 - Very low purity runs: run 9055 and 9056, more info here: <https://docs.google.com/spreadsheets/d/1gPqQbPFoOZjDLPfPVHBfCrZz9g0IQoqWUUhFnKVx6aQ/edit#gid=0>
 - No signal (can't see any tracks)
 - Long readout runs with HV off
 - Normal runs
- Wire and coherent noise are in `dune/Detsim/Service/ShapedCohProtoDUNENoiseService*`
- New digitiser is in `dune/Detsim/Tools/NoisyAdcSimulator*`

The noise modelling concepts review

The approach is **hardware-aware parametric Time series signal generation**:

- (simplified) Noise sources regard to hardware components, noise model and shape of charge-sensitive amplifier, CR-RC pulse-shaper and digitisers are well known.
 - There components, white noise, sine wave, gaussian used to generate the noise
- It has several advantages:
 - Signal is generated for each channel, it is statistically **noncorrelated** noise/signal, statistic and randomness event by event.
 - One can add noise in analysis on-the-fly. Parameters make the simulation versatile and could be updated regards to new noise environments or different noise hypotheses. No need to change the code for any noise model.
 - Phase information could be added to simulation (unlike InvFFT based methods).
 - It seems FEMB is the unit of coherent noise.

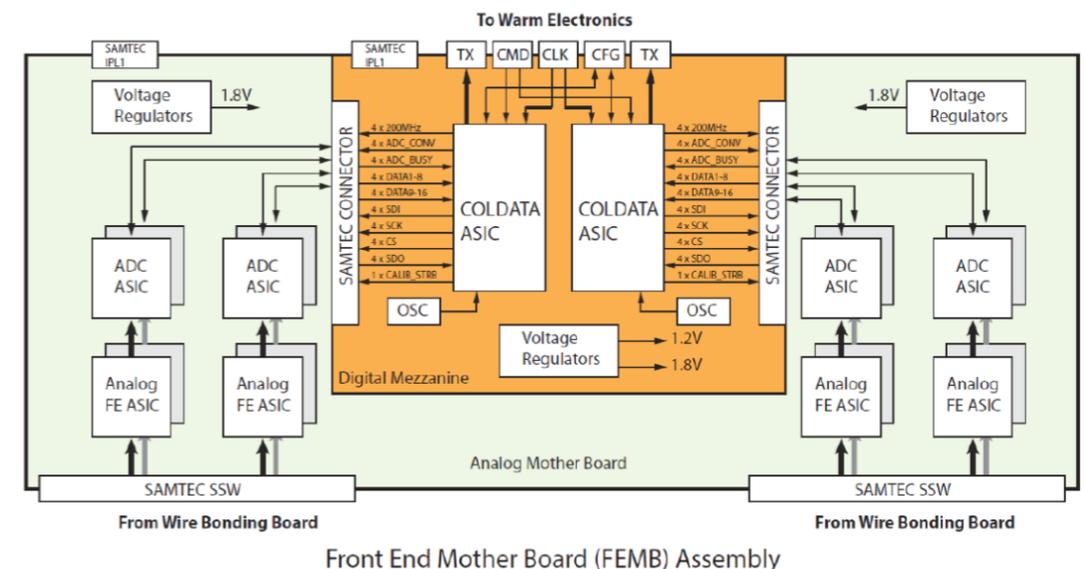
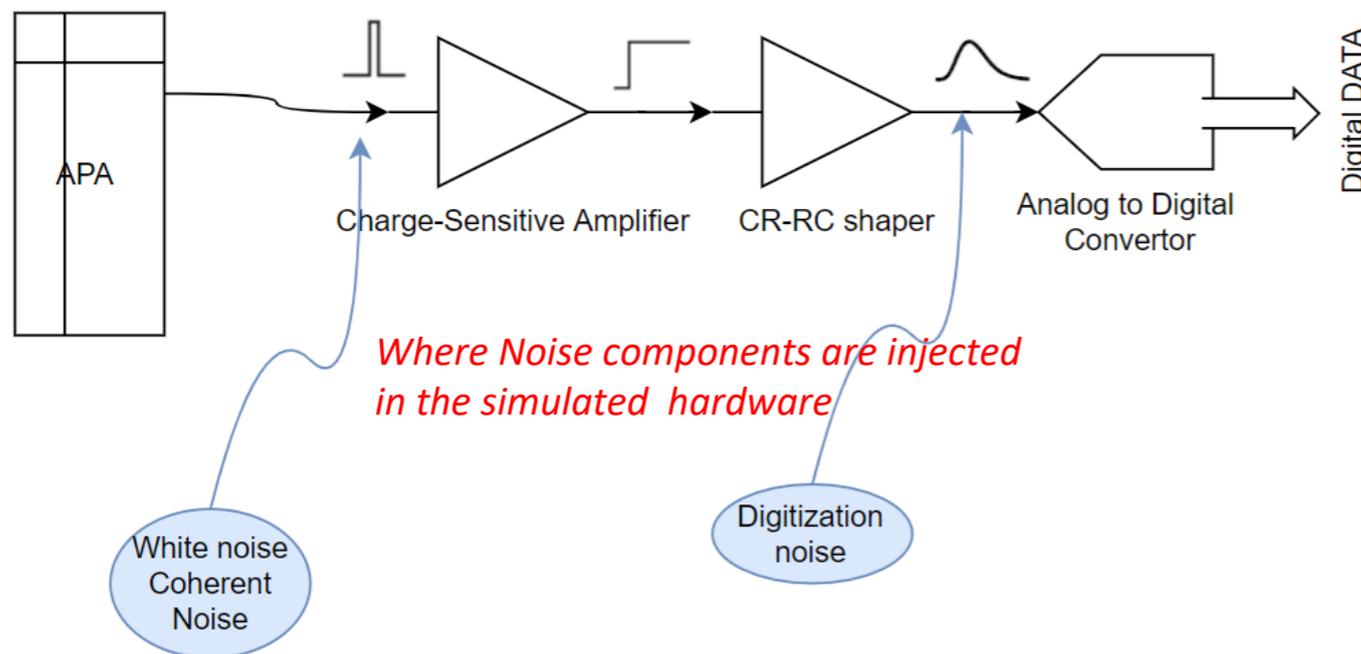


Figure 1. Block Diagram of 128 Channel Card

Noise (signal) extractions (and implementation)

- White noise and coherent noise extraction steps:
 1. Remaps the channels to physical/FEMB order, correlated bundles of 128 seems optimal choice (*Next slide Fig a*).
 2. Remove signal, if there is a signal, we change the waveform value to 0
 3. Removing coherent noise to separately measure the noise components specifications
 1. Create Mean of Sum of all coherent channels (128 channels FEMB), 20 Coherent waveform.
 2. Remove coherent waveform from waveform of bundle channel bundle, 2560 waveforms. (*Next slide Fig b*)
 3. FFT and analysis sinewave components of 20 Coherent waveform and create vectors of frequencies and amplitudes for each coherent bundle.
 4. Analysis FFT of 2560 waveform to extract white noise level and HV noise and other non-coherent components.
 5. Extract pedestal to generate a pedestal list also for a realistic pedestal variation channel by channels (*Next slide Fig c*)
 6. Frequency spectrums and its components are **dynamic** through ;
 - a) **The time**, Fig d and e show two frequency spectrum of the same channel but in two different time setting, one can see dynamic sinewave components through the time.
 - b) **Channel by channels**, Channel's 1 to 100 frequency spectrum has been showed in the backup. The plots are low frequency resolution and raw signal waveform.

So to create similar effect a random (gaussian) smearing can be added in top of frequency and amplitude vectors for each bundle,

Noise (signal) extractions

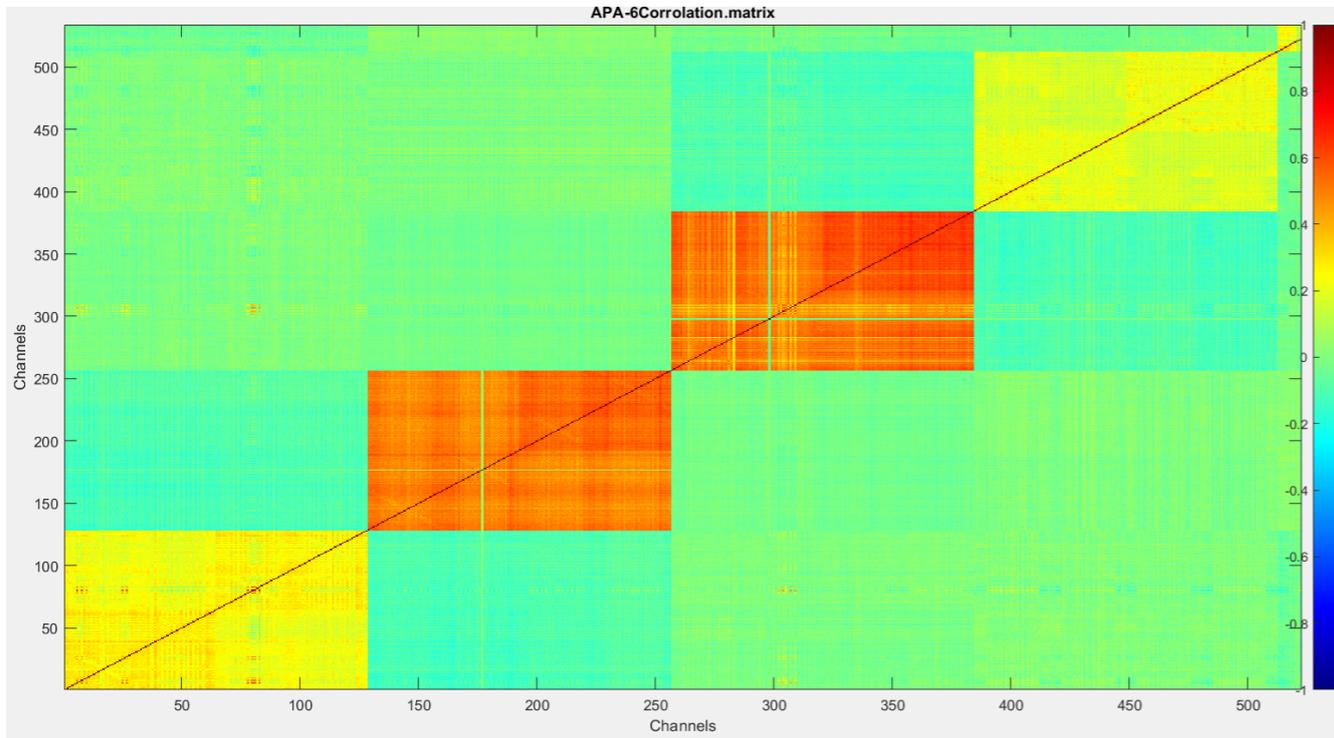
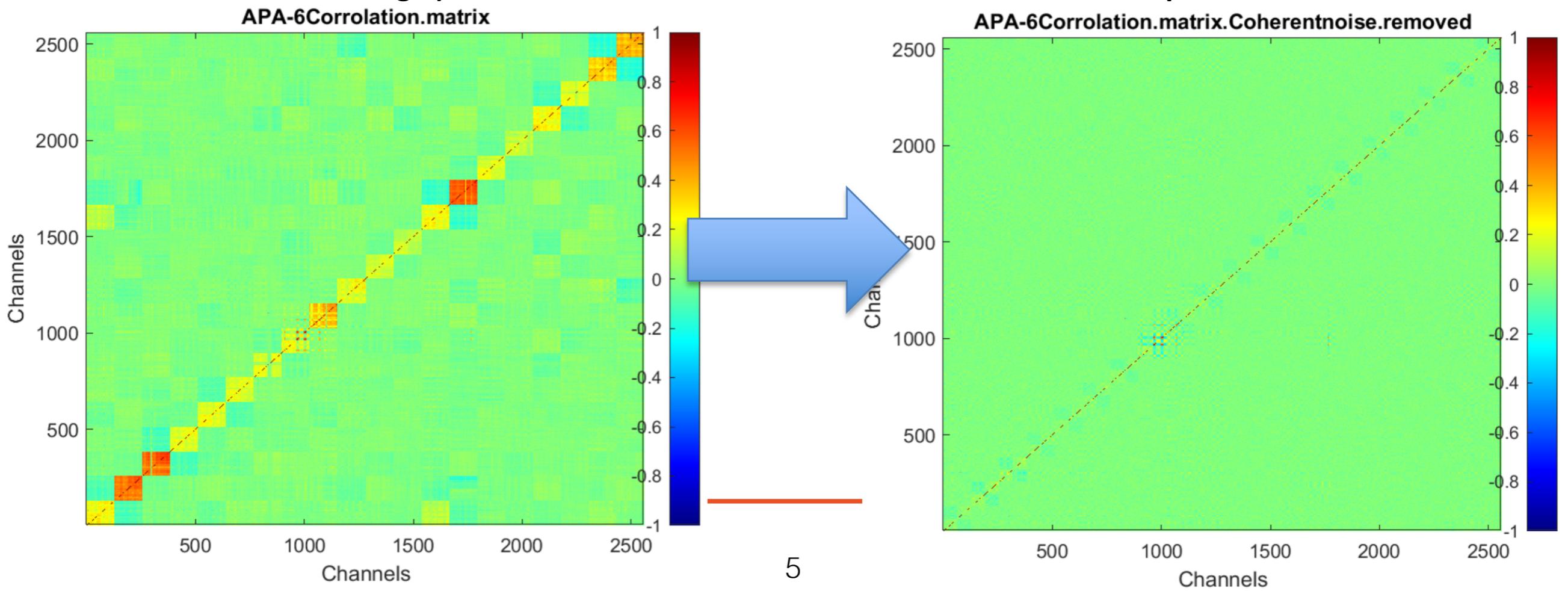
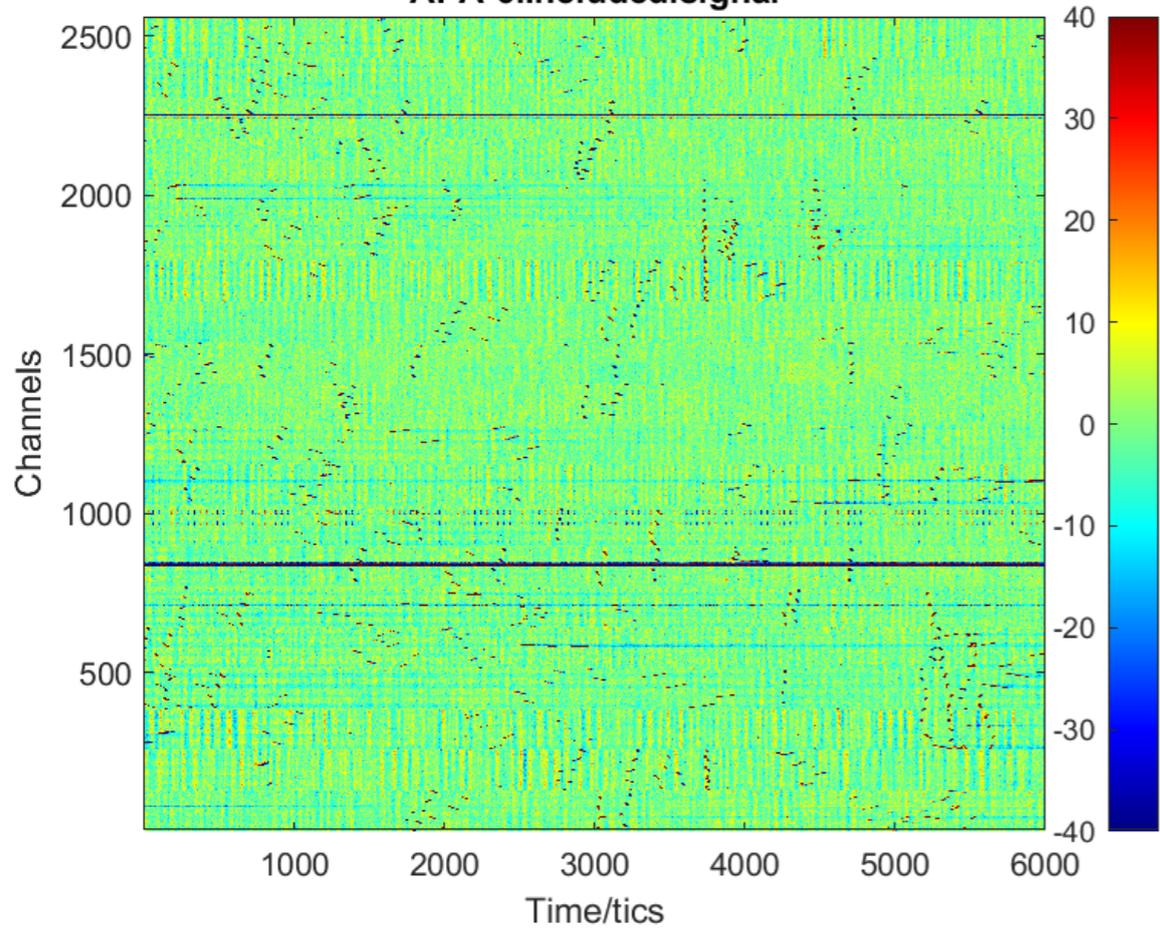


Fig a) correlation matrix shows a good 128 channels correlation. Correlation bundle can be chased down to smaller bundles too (16 channels ADC) but it does not give much different results. 128 channels (1 FEMB) might be the optimum.

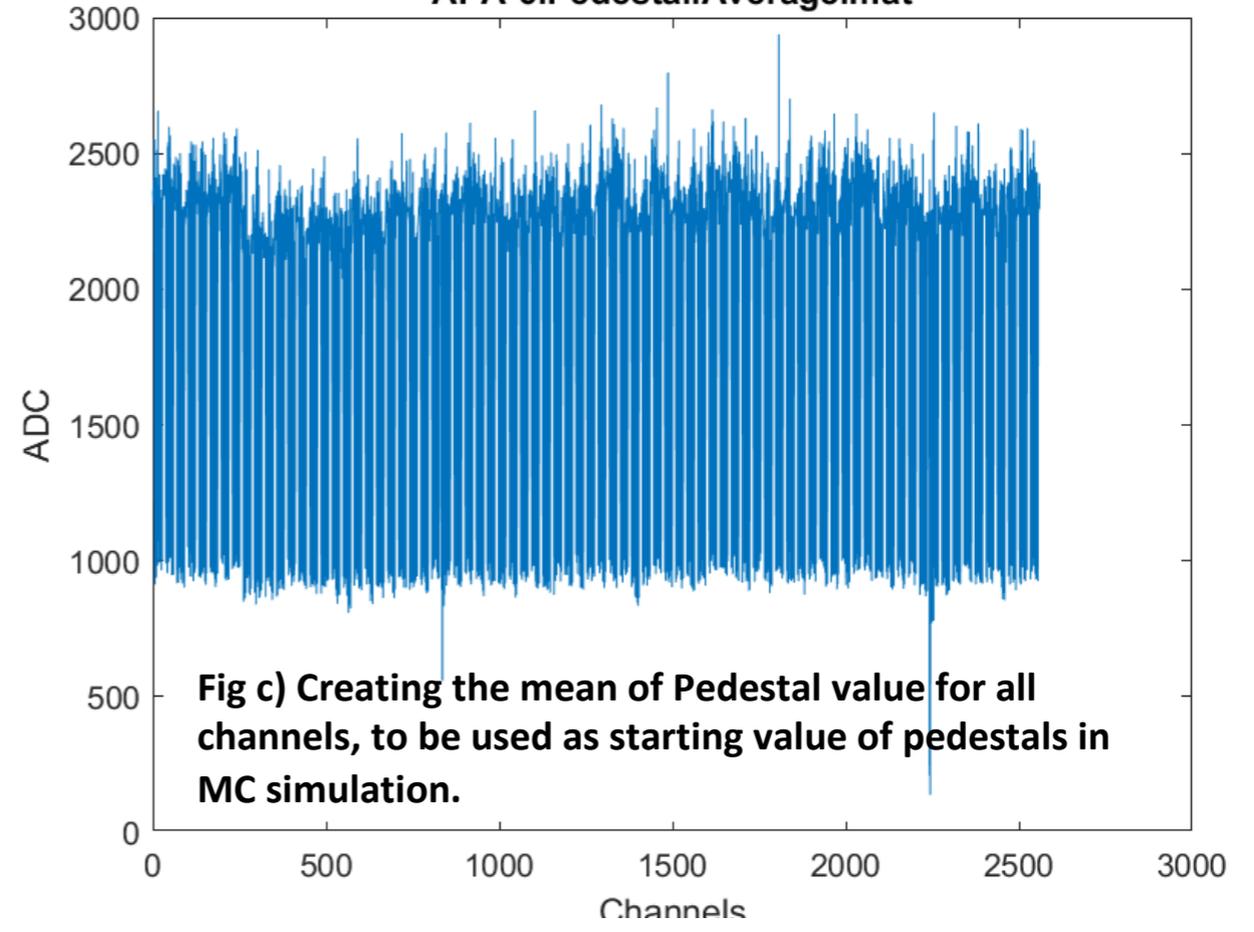
Fig b) correlation matrix before and after coherent removal procedure



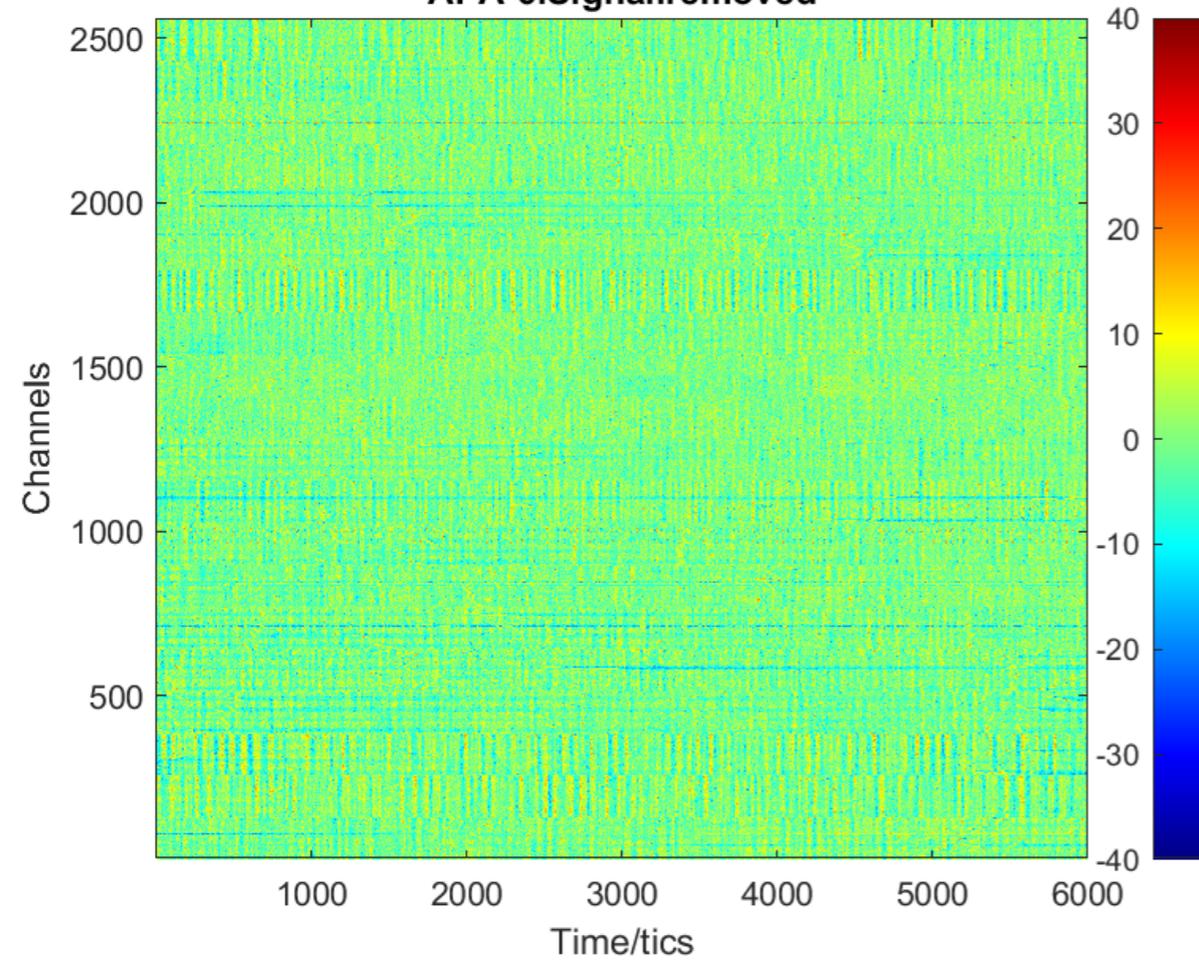
APA-6.Included.signal



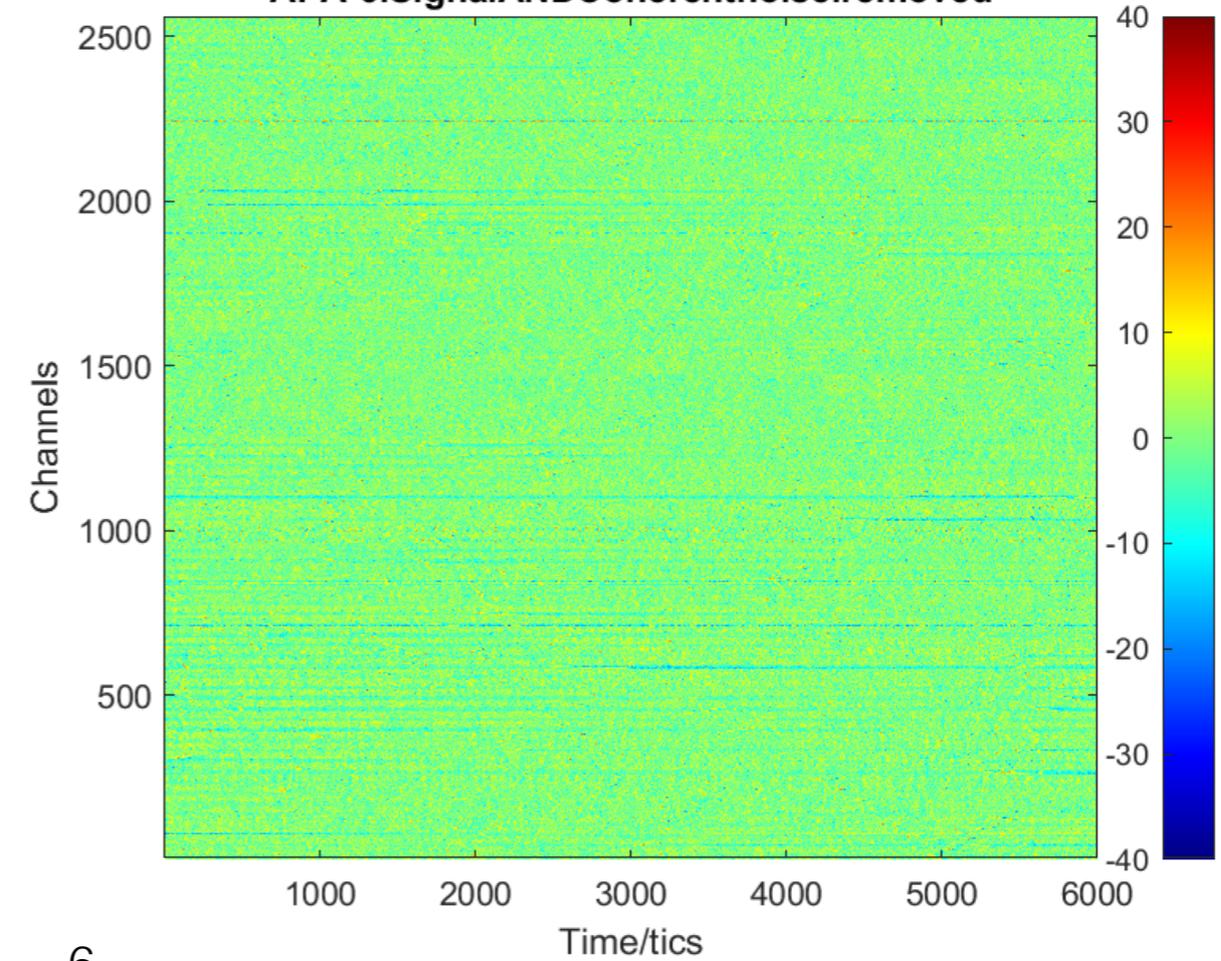
APA-6.Pedestal.Average.mat



APA-6.Signal.removed

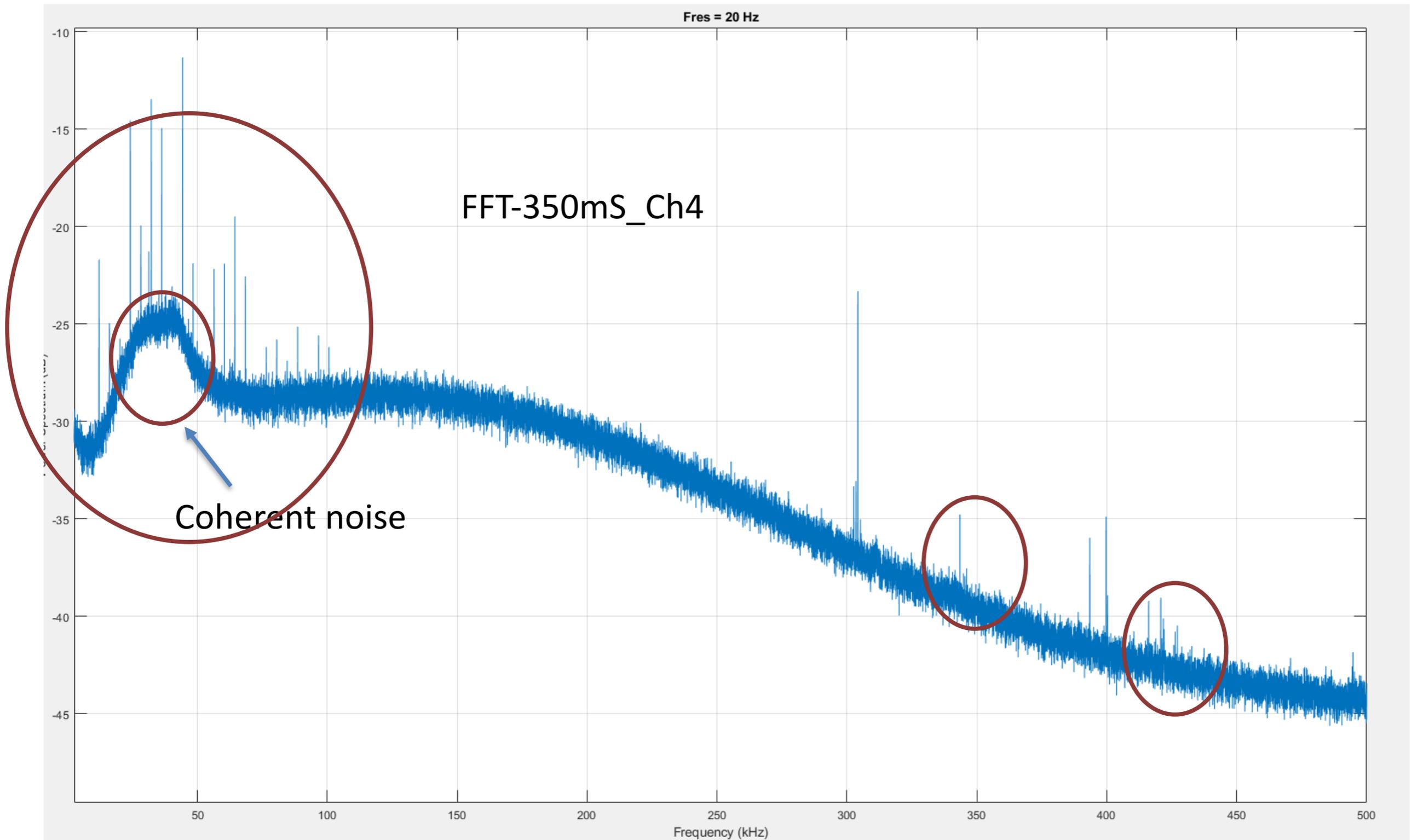


APA-6.SignalANDCoherentnoise.removed



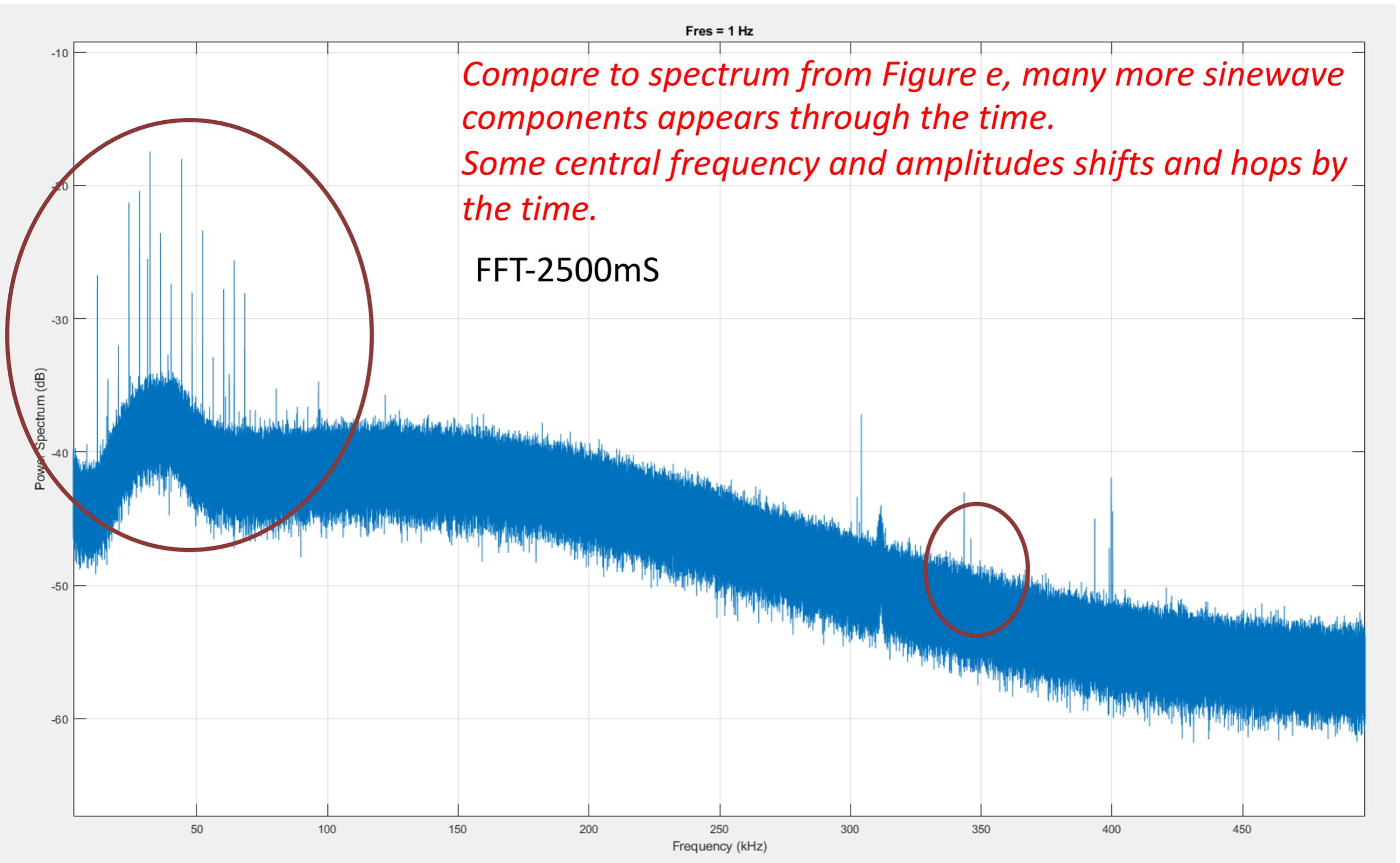
Frequency spectrum of one channel for 350ms

- *Fig d*

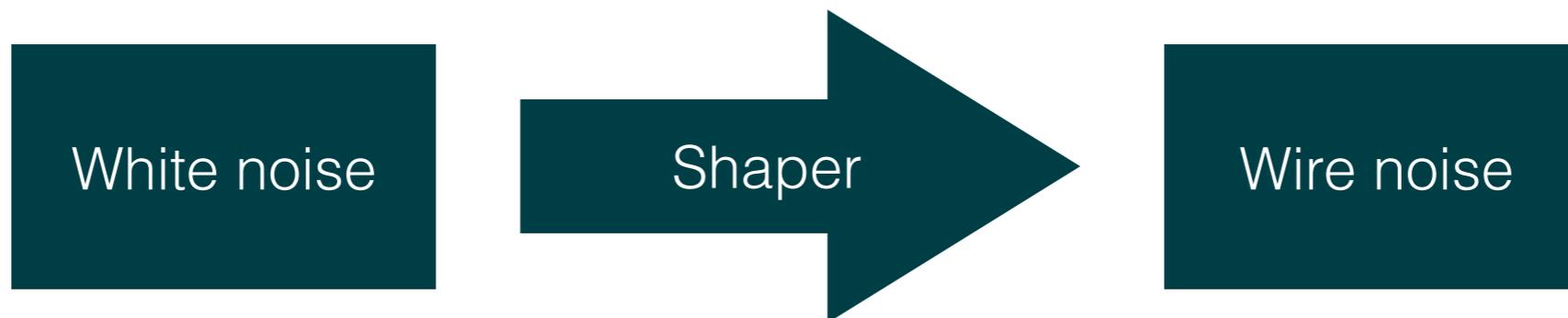


Frequency spectrum of one channel for 2500ms

- *Fig e*



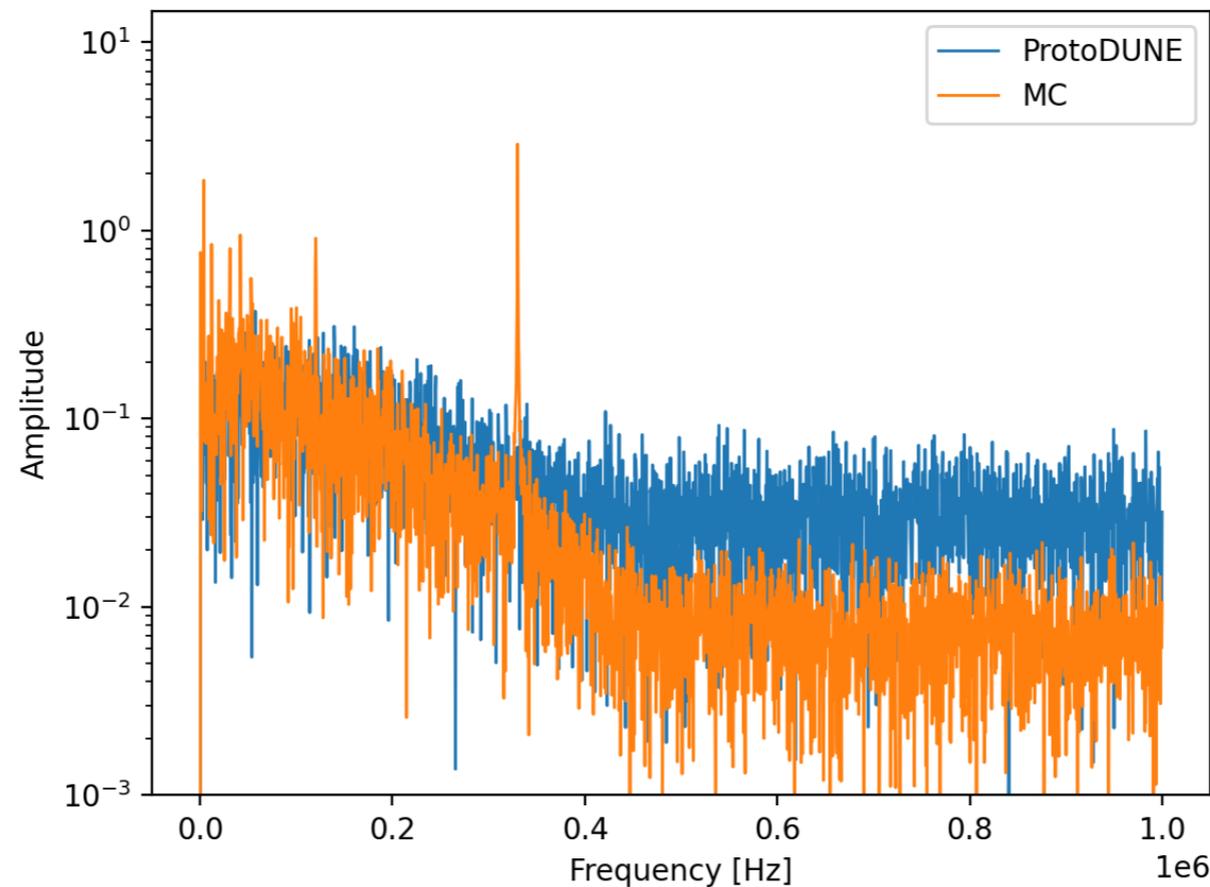
- Idea here is to use the signal shaper to get the correct shape of the noise.
- Shaper in `dunetpc` lives in `dune/Utilities/SignalShapingServiceDUNE_service.cc`
 - Includes field shaping, sampling and electronics shaping
 - Need to remove field shaping for the noise generation
 - Added other `SignalShaping` objects that only includes the electronics and sampling responses
 - Added `ConvoluteElectronicResponse` function (no deconvolution equivalent)
- Then, simply take a white noise and convolute it with the electronic response.



- Adjust for the amplitude.
 - 6 parameters to adjust the amplitude:
 - ADC amplitude for induction and collection wires
 - 3.8 ADC collection + RMS of 0.9
 - 5 ADC induction + RMS of 1
 - Factor to adjust for the amplitude change due to the shaper
 - ~50: this needs to be changed if the shaper changes

```
#####
## Wire noise (incoherent noise generated according to the FT of the shaper)
#####
collection_plane_noise: 3.80
induction_plane_noise: 5
collection_plane_noise_rms: 0.9
induction_plane_noise_rms: 1

## Change if you touch the electronic_shaper!!
amplitude_multiplier_induction: 50.51
amplitude_multiplier_collection: 46
```



- Makes use of the `PdspChannelMapService` to grab FEMB → Channels mapping
- Add sine waves for each FEMB for each new event
 - Need another function in `ChannelNoiseService` :
 - `void ChannelNoiseService::newEvent() {};`
 - I've added that in `DuneInterface/Service/ChannelNoiseService.h`
 - Doesn't require any change anywhere (by default, it doesn't do anything)
 - In the `ShapedCohProtoDUNENoiseService`, it calculates the total amplitude of the coherent noise for each FEMB (otherwise the noise generation is very slow, it takes a lot of time calculating sines).

- Configuration:
 - Amplitude, frequencies
 - Frequencies and amplitudes have an RMS
 - Phases are ignored for now

```
#####
## Coherent noise (FEMB-wise sines)
#####
random_phase_noise: true
frequency_rms: 21
amp_rms: 0.5
# Units are:
# Hz for frequencies
# ADC for the amplitude
# pi for the phase
FEMBCo_Frq: [
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB1
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB2
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB3
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB4
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB5
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB6
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB7
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB8
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB9
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB10
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB11
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB12
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB13
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB14
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB15
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB16
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB17
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB18
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ], # FEMB19
[ 200, 3000, 4000, 12000, 31000, 42000, 53200, 120000, 330000 ] # FEMB20
]
```

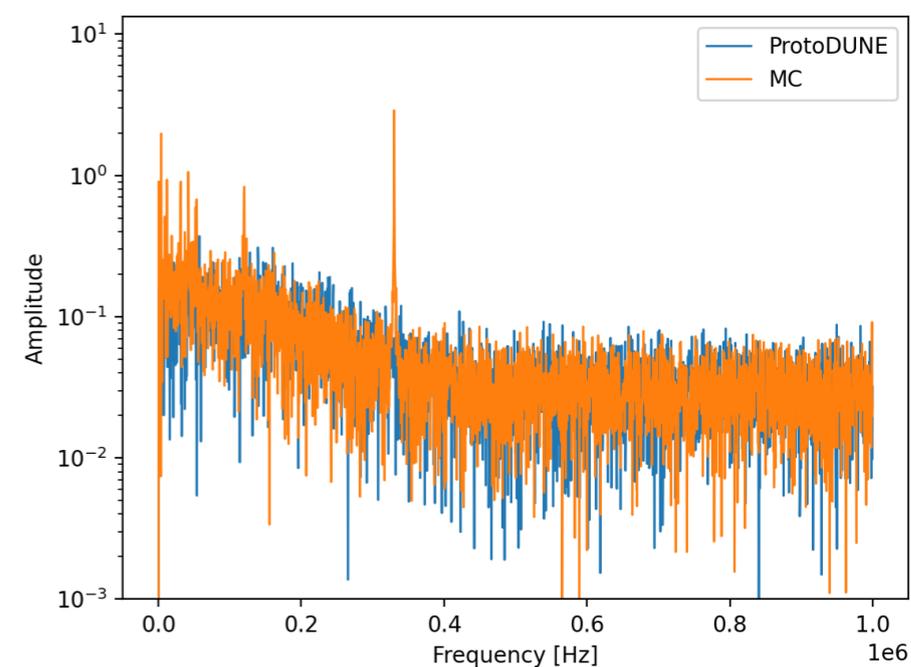
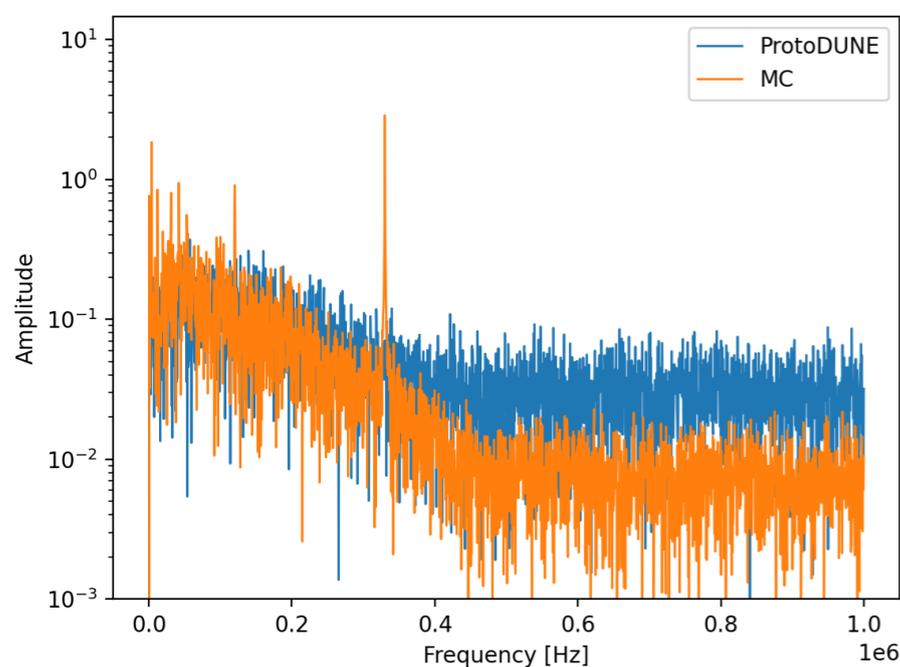
```
FEMBCo_Amp: [
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB1
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB2
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB3
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB4
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB5
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB6
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB7
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB8
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB9
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB10
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB11
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB12
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB13
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB14
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB15
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB16
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB17
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB18
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ], # FEMB19
[ 1, 0.2, 2, 1, 1.1, 1.2, .8, 1.2, 3 ] # FEMB20
]

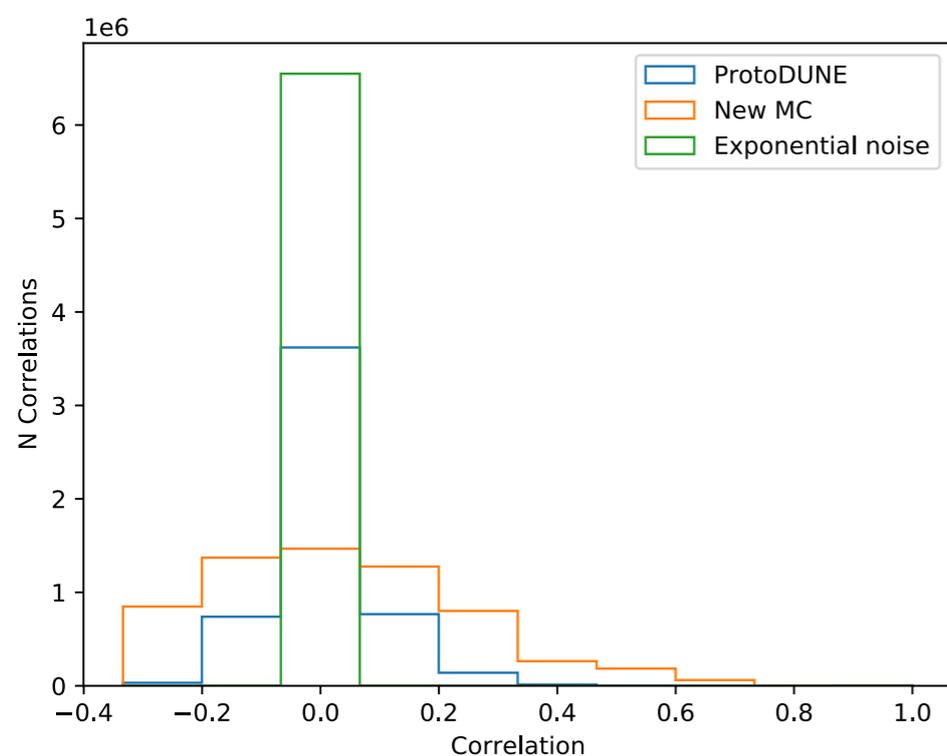
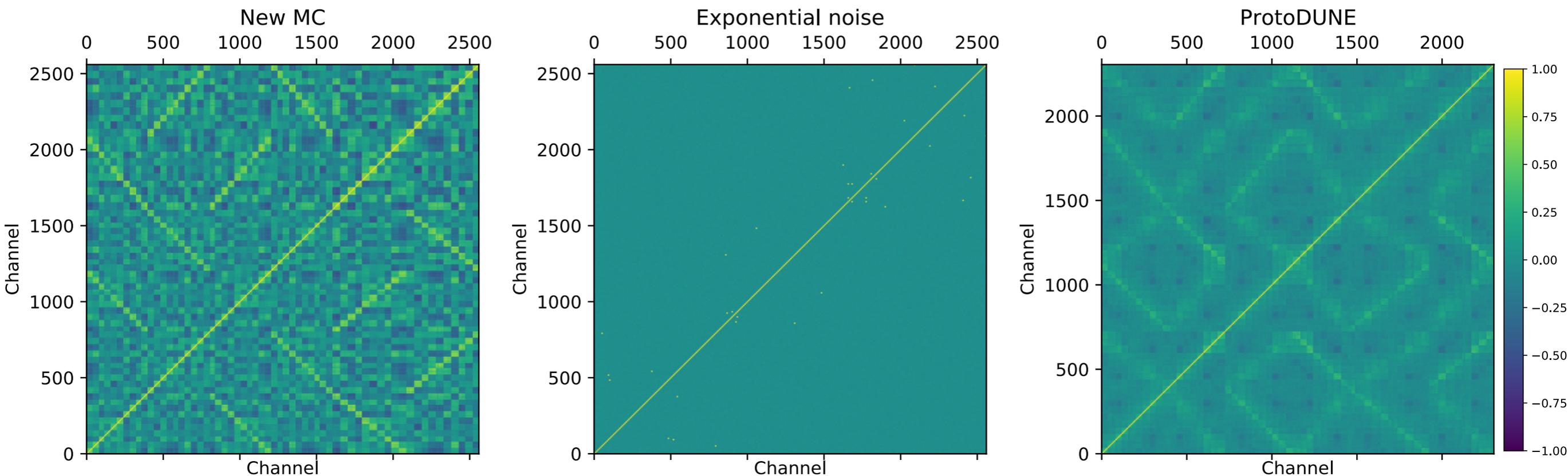
FEMBCo_Phs: [
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB1
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB2
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB3
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB4
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB5
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB6
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB7
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB8
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB9
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB10
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB11
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB12
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB13
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB14
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB15
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB16
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB17
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB18
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ], # FEMB19
[ 0, 0, 0, 0, 0, 0, 0, 0, 0 ] # FEMB20
]
```

- Simple sine waves added coherently to all the channels
 - Amplitude, frequency
 - Phase ignored

```
#####  
## HV noise (sine wave moving everything altogether)  
#####  
# HV noise  
HV1_Frq: [ 50, 3000 ]  
HV1_Amp: [ .5, .3 ]  
HV1_Phase: [ 0, 0 ]
```

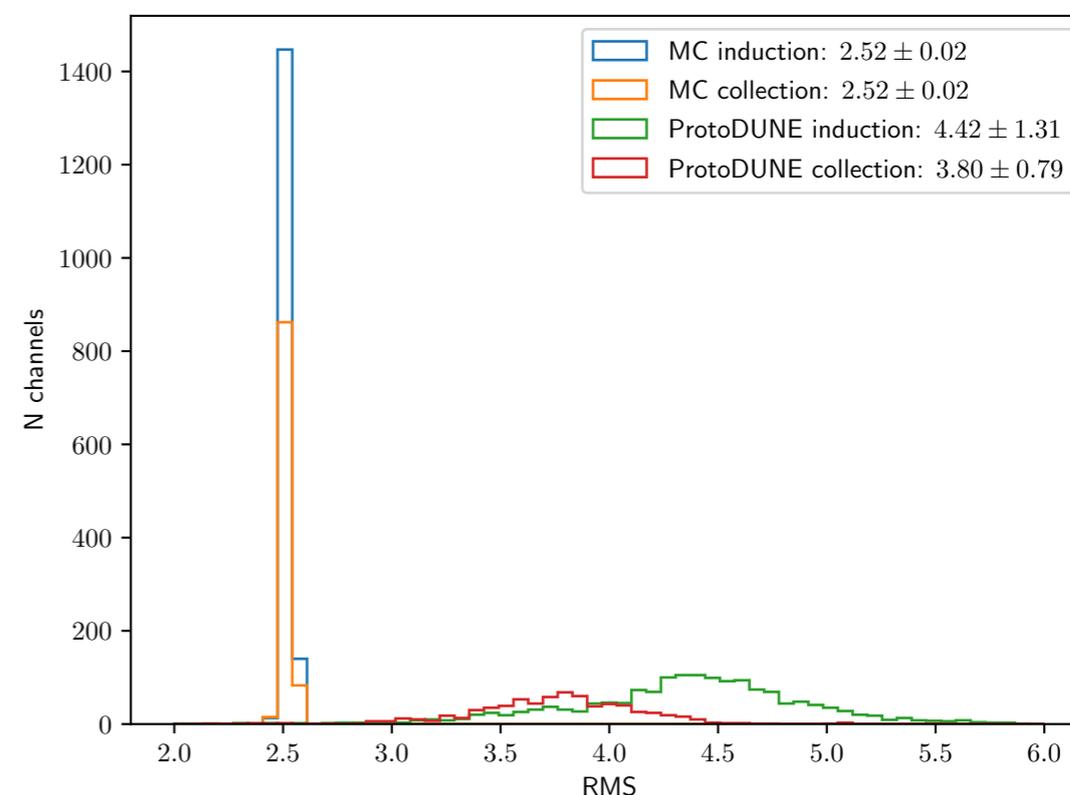
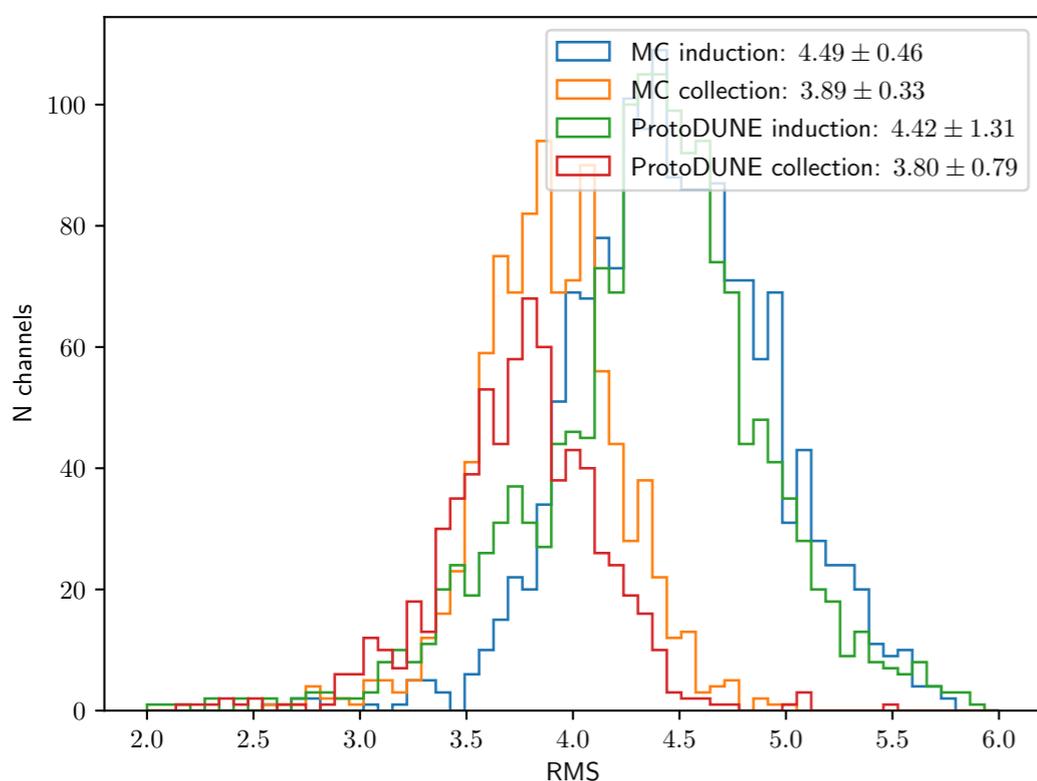
- Digitisation noise is needed to explain the remaining of the noise at high frequencies.
- We expect the digitiser to be imperfect and introduce a binomial noise
- Again, this is tuned to ProtoDUNE data
 - Amplitude of the digitiser noise is 5



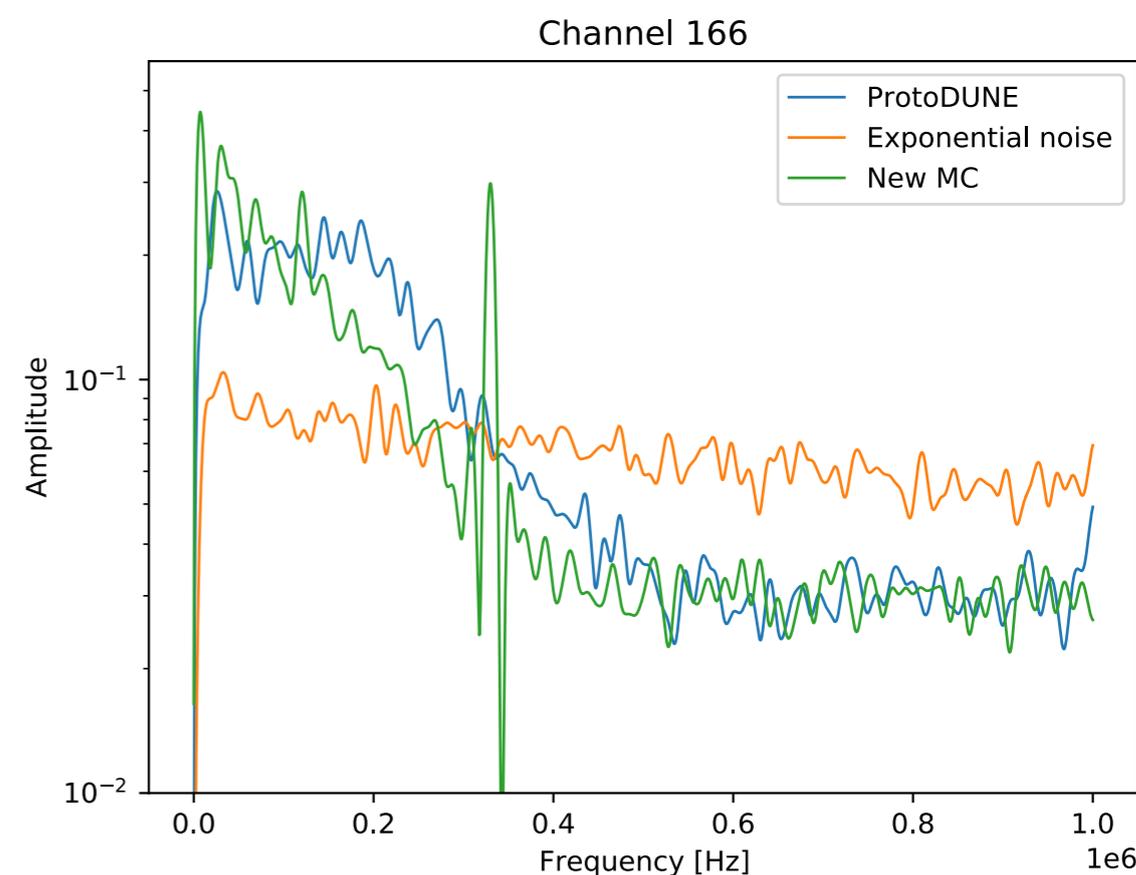
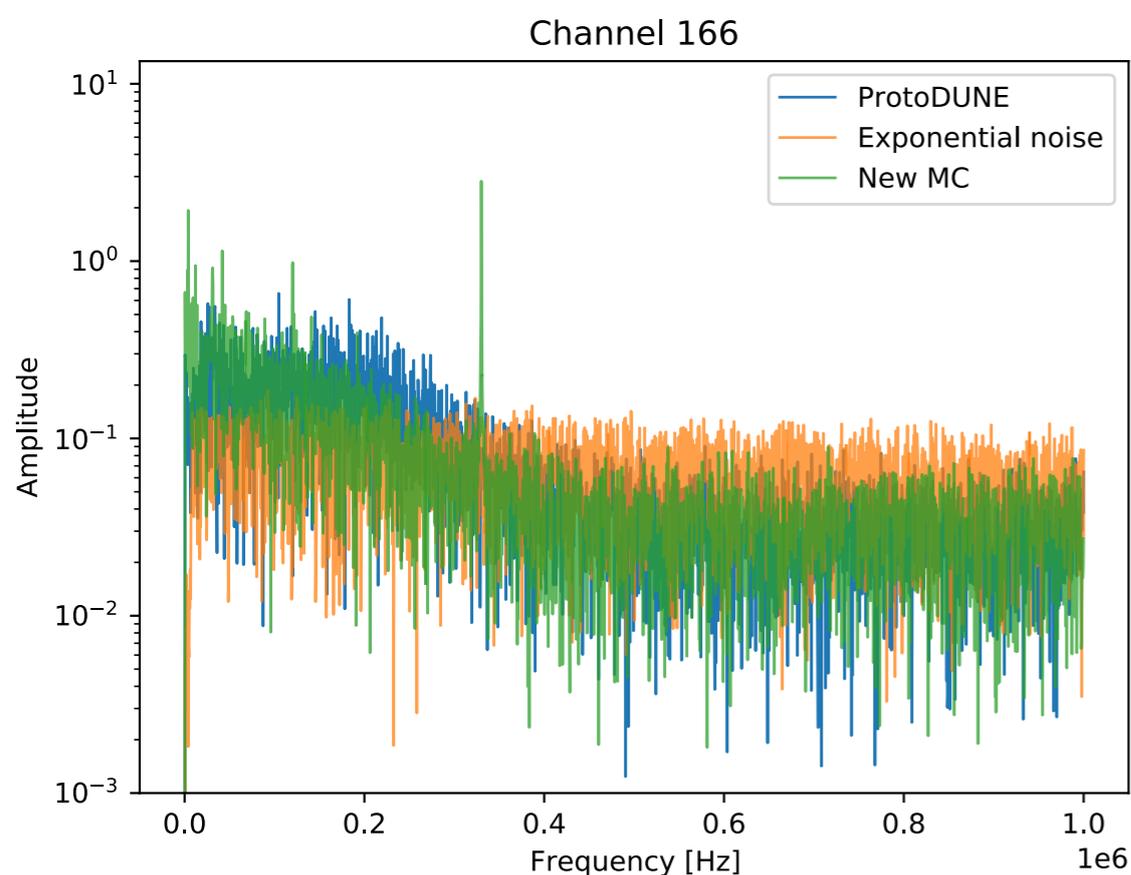


- Basic shapes of the correlations are reproduced.
 - Exponential noise has no correlation, except a couple fully of channels fully correlated randomly
- More correlations in new MC than in data...
 - More fine tuning of the amplitudes of the coherent noise is needed.

- Total RMS seems to agree
 - Note: no removal of stuck bits for real data, this might have an effect on the ProtoDUNE RMS...
 - Left: new noise model, right: exponential noise



- 50 other FTs are available here [docdb link]
- Amplitude is random, but generally agreement is good
 - Left without any filter, right with a filter (don't ask me what filter exactly)



```
=====
TimeTracker printout (sec)
=====
```

	Min	Avg	Max	Median	RMS	nEvts
Full event	73.2761	77.4455	80.8102	78.0561	2.6221	10
source:RootInput(read)	0.000831172	0.00306357	0.0111831	0.00163339	0.0034638	10
simulate:rns:RandomNumberSaver	9.44e-05	0.000329996	0.00113348	0.000255259	0.000282858	10
simulate:daq:SimWireDUNE	67.9605	71.752	75.1656	72.3519	2.41088	10
simulate:opdigi:OpDetDigitizerDUNE	0.000299063	0.00409999	0.0124957	0.00467527	0.00365887	10
[art]:TriggerResults:TriggerResultInserter	3.1914e-05	0.00011342	0.000213787	0.000136405	6.83044e-05	10
end_path:out1:RootOutput	9.492e-06	1.46927e-05	2.2491e-05	1.51325e-05	4.04538e-06	10
end_path:out1:RootOutput(write)	5.30503	5.68408	6.34892	5.67471	0.318179	10

```
=====
```

```
=====
MemoryTracker summary (base-10 MB units used)
```

```
Peak virtual memory usage (VmPeak) : 2224.82 MB
Peak resident set size usage (VmHWM): 1650.72 MB
```

```
=====
```

- Around 72 seconds per 1x2x6 event (~30k channels)
- 1.65 GB max resident RAM
- 2.22 GB max virtual RAM

- Added a new noise service `ShapedCohProtoDUNENoiseService` and `NoisyAdcDigitiserTool`
 - Statistically independent noise
 - Available on develop and next `dunetpc` releases
 - Reproduces the ProtoDUNE noise
 - Still some fine tuning in the amplitudes of the coherent noise and number of components to get the correlations exactly right
 - To use, see the following fhicl:


```
fc1/dunefd/detsim/protodunelike_detsim_dune10kt_1x2x6.fcl
```
- Plenty of FFTs: <https://docs.dunescience.org/cgi-bin/private/ShowDocument?docid=22027>

