Weekly Meeting

Matt Wilson – December 6 2017

Purpose of noisePSD Decider

- Check the noise PSD traces for each channel – determine whether the noise PSD crosses one or more maximum or minimum threshold.
- Can be used to check for peaks at specific frequencies (e.g. 60 kHz) and/or large range of frequencies.
- Thresholds can be made as detailed or as simple as desired.

The Decider Code

- Can be found: http://titus.stanford.edu:8080/git/tree/?f=python/dqm/deciders/noisePSD_decider&r=DAQ/dqm.git&h=feature/mysql-dev

- noisePSD_decider.py is the main code that updates settings, makes decisions, etc
- noisePSD_read_settings.py is used to read in the settings from the text files
- file_path.py is where the paths necessary for this decider are held
- plot_noisePSD.py is used to plot the noisePSD traces
- The remaining text files are what is used to set the settings for this decider

Using the noisePSD Decider

- Currently, this decider is only able to make decisions on old Soudan data (mainly because that is what I have readily available, and I don't know exactly what the SNOLAB root data will look like)
- If you would like to test the code, use Soudan data that uses iZIP4 type detectors
  - The root data I am using is from the FermiLab cdmsz3.fnal.gov node in /data1/cdmsmini/data3/cdmsbatsProcessedData/
- Eventually, this decider will need to be adapted for SNOLAB root data.

Using the noisePSD Decider

- To try the decider, make sure to update the paths in file_paths.py

```
8  raw_data_dir = '/home/mwilson/DQM/cdmsbatsProcessedData/'
9  noisePSD_threshold_settings_file = '/home/mwilson/DQM/dqm/python/dqm/deciders/noisePSD_decider/noisePSD_threshold_settings.txt'
10 last_noisePSD_threshold_settings_file = '/home/mwilson/DQM/dqm/python/dqm/deciders/noisePSD_decider/last_noisePSD_threshold_settings.txt'
11 noisePSD_decision_settings_file = '/home/mwilson/DQM/dqm/python/dqm/deciders/noisePSD_decider/noisePSD_decision_settings.txt'
12 last_noisePSD_decision_settings_file = '/home/mwilson/DQM/dqm/python/dqm/deciders/noisePSD_decider/last_noisePSD_decision_settings.txt'
13 plot_path = '/home/mwilson/DQM/test_folder/'
14 series_prefix = '01'
15 raw_data_format = 'Soudan'
16
17
18 def get_raw_data_dir():
19    return raw_data_dir
20 def get_series_prefix():
21    return series_prefix
22 def get_noisePSD_threshold_settings_file():
23    return noisePSD_threshold_settings_file
24 def get_last_noisePSD_threshold_settings_file():
25    return last_noisePSD_threshold_settings_file
26 def get_noisePSD_decision_settings_file():
27    return noisePSD_decision_settings_file
28 def get_last_noisePSD_decision_settings_file():
29    return last_noisePSD_decision_settings_file
30 def get_raw_data_format():
31    return raw_data_format
32 def get_plot_path():
33    return plot_path
```

update

noisePSD Settings

There are two categories of settings for this decider:

- Threshold settings – the actual thresholds that will be compared against the noisePSD traces
- Decision settings – the parameters to determine how decisions get made. For example, how many bins need to cross a threshold to be considered a “bad” channel? How many bad channels make a detector “bad”.

All settings can be applied to any or all data types and any or all channels, detectors, or towers

The text files beginning with last_*.txt are to not be altered…they are used to determine if updates are required.

Threshold Settings

- Can include as many thresholds as desired, labelled as t1, t2, …
- For each threshold, can have as many entries as desired for certain data types or certain elements (elements mean channels, detectors, towers, etc..)
- For each entry, you can put in as many ‘Max’ or ‘Min’ entries as desired, corresponding to max and min threshold, respectively.
- ‘Min’ and ‘Max’ entries required a specific format to add a threshold. The format is as follows:
  - <min frequency> <max frequency> <threshold value>
- This piecewise method is what allows the thresholds to be implemented in very customizable ways – from very simple to very detailed
- ‘Min’ and ‘Max’ entries that overlap in frequencies is fine – the decider always takes the most constraining threshold entry at any given frequency.

Threshold Settings

```
1  Threshold = t1
2  Data types = all
3  Applies to = all charge
4  Max entries = 100 700000 1e-6 , 100 100000 7e-7
5  Min entries = 100 200000 1e-9
6  Applies to = all phonon
7  Max entries = 100 200000 4e-10 , 1000 300000 3e-10 , 100000 120000 1e-10, 120000 140000 9e-11, 140000 160000 8e-11, 160000 180000 7e-11, 180000 200000 6e-11, 200000 2
8  Min entries = 100 200000 1e-12
9  Applies to = T1Z5, P_TOP_4 T2Z2
10 Max entries = 100 300000 2e-9 , 1000 200000 6e-10 , 200000 700000 1e-10, 2000 2100 1e-10
11 Min entries = 100 200000 2e-12, 2000 5000 3e-12
12 Applies to = P_TOP_4 T3Z2
13 Max entries = 100 700000 8e-10 , 1000 700000 5e-10 , 200000 700000 4e-10, 2000 2100 1e-10
14 Min entries = 100 200000 2e-12, 2000 5000 3e-12
15 Threshold = t2
16 Data types = all
17 Applies to = all charge
18 Max entries = 100 700000 5e-6 , 100 100000 1e-6
19 Min entries = 100 200000 8e-10
20 Applies to = all phonon
21 Max entries = 100 100000 1e-9 , 1000 100000 8e-10 , 90000 700000 1e-8
22 Min entries = 100 200000 8e-13
23 end
```

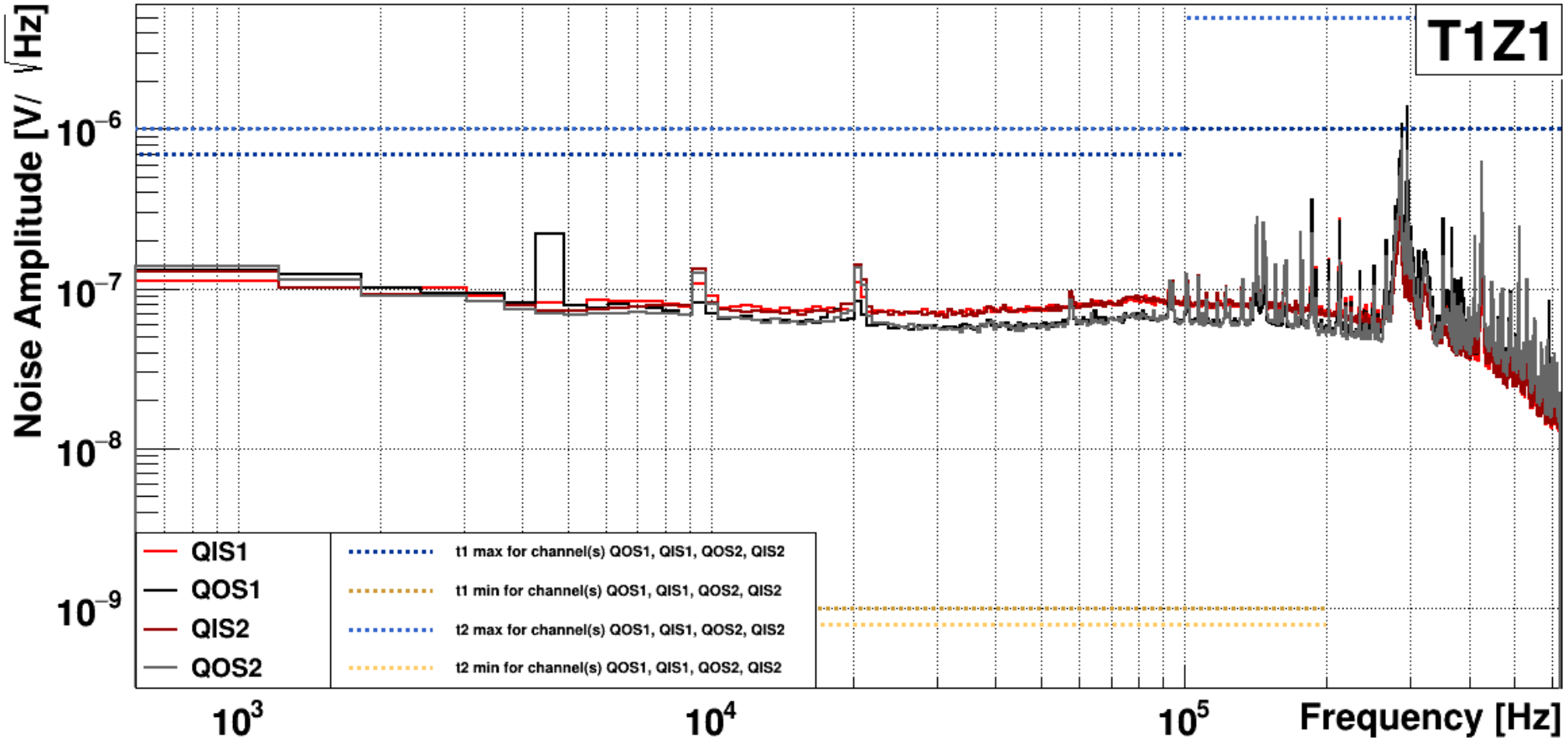
← Indicate threshold

← Indicate data types, comma separated. E.g. “Cf, Cs”

← Indicate which elements this entry applies to, comma separated. E.g. “T1Z2, T2Z3, T4 charge”

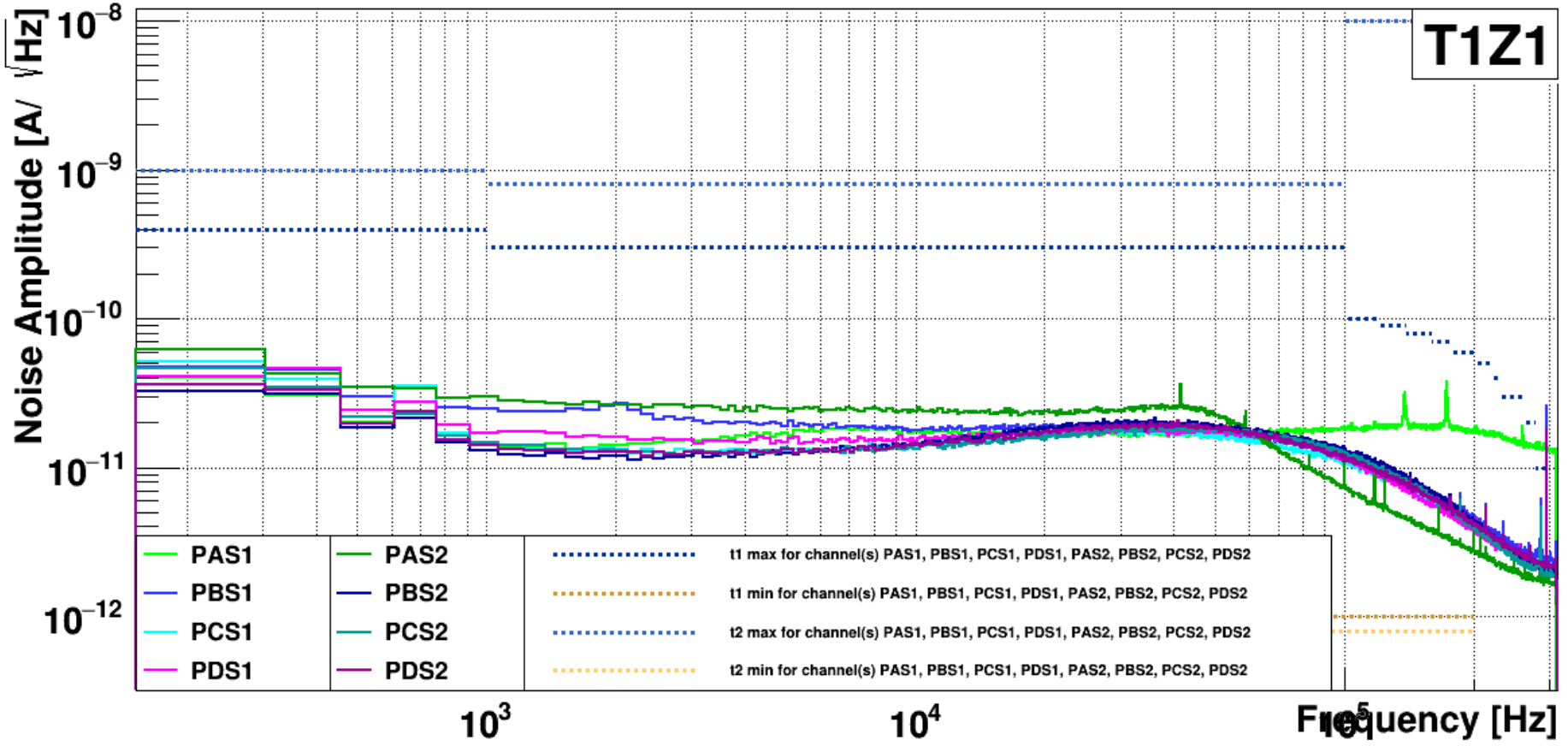
Min and max entries. Each min or max entry has a format of <min frequency> <max frequency> <threshold value>. Min or max entries are separated by comma

Threshold Settings



*Note that the plots only show the most constraining thresholds (overlaps are removed)

Threshold Settings



*Note that the plots only show the most constraining thresholds (overlaps are removed)

Decision Settings

- Similar formatting to the threshold settings, but instead give parameters needed to make decisions. Includes the following:

- “bad channel t#” – how many bins make a bad channel for threshold #
- “bad detector charge” – how many bad charge channels make a bad detector
- “bad detector phonon” – how many bad phonon channels make a bad detector
- “bad tower” – how many bad detectors make a bad tower
- “bad decision channel” – how many bad channels make a global bad decision
- “bad decision detector” – how many bad detectors make a global bad decision

Decision Settings

```
1  Parameter = bad channel t1
2  Data types = all
3  Applies to = all
4  Max = 20 ← max threshold
5  Min = 1  ← min threshold
6  Parameter = bad channel t2
7  Data types = all
8  Applies to = all
9  Max = 2
10 Min = 1
11 Parameter = bad detector charge
12 Data types = all
13 Applies to = all
14 Max = 3
15 Min = 0
16 Parameter = bad detector phonon
17 Data types = all
18 Applies to = all
19 Max = 7
20 Min = 0
21 Parameter = bad tower
22 Data types = all
23 Applies to = all
24 Max = 4
25 Min = 0
26 Parameter = bad decision channel
27 Data types = all
28 Applies to = all
29 Max = 24
30 Min = 0
31 Parameter = bad decision detector
32 Data types = all
33 Applies to = all
34 Max = 5
35 Min = 0
36 end
```

Settings

- Seemingly overlapping entries in the “Applied to” line are fine in this code.
- For example, if one entry exists for “all phonon” channels, and another entry is added for “P_TOP_3 T1Z1” channel, there is an overlap in entries for this one channel. However, the code always assigns entries to the more specific element. In this case, the first entry will have instead all phonon less P_TOP_3 T1Z1. The user does not need to worry about this sort of overlap.

noisePSD Plots

- Currently, the decider is set up to plot, save, and display the noisePSD plot for a detector whenever a channel is deemed to be “bad”.
- This is arbitrary, and plots may not even need to be saved or displayed.

Issues with keeping entries unique

- The main issue I had when working on this decider has to do with ensuring entries are unique to data types or elements.
- For instance, there should only ever be one entry for Cf data type at a time, and there should only ever be one entry for P_TOP_1 T1Z1 channel at a time.
- The way that the functions in the settings code search for entries, the wrong entry might be returned for situations that are probably rare but possible.
- Examples on next slide:

Issues with keeping entries unique

- Example 1:
  - Initial have entry (1) that applies to both Cf and Cs calibration
  - Add entry to database – data type entries are ((Cf, Cs))
  - Later, decide that Cs should have its own entry (2). Add separate entry to database – data type entries are now ((Cf, Cs), (Cs))
  - I want to call upon the entry for Cs. The code returns the first entry in the list that includes Cs. In this case, it is entry (1). It should be entry (2).
  - The code does not ensure that there is only one entry per data type at a given time.

Issues with keeping entries unique

- Example 2:
  - Initial have entry (1) that applies to both P_TOP_1 T1Z1 and P_TOP_2 T1Z1 Add entry to database – element entries are ((P1, P2))
  - Later, decide that P_TOP_2 T1Z1 should have its own entry (2). Add separate entry to database – element entries are now ((P1, P2), (P2))
  - I want to call upon the entry for channel P2. The code returns the first entry in the list that includes P2. In this case, it is entry (1). It should be entry (2).
  - The code does not ensure that there is only one entry per channel at a given time.

Issues with keeping entries unique

- Example 3:
  - Initial have entry (1) that applies to channel P_TOP_1 T1Z1
  - Add entry to database – element entries are ((P1 T1Z1))
  - Later, decide that T1Z1 should have its own entry (2). Add separate entry to database – element entries are now ((P1 T1Z1), (T1Z1))
  - I want to call upon the entry for channel P1. The code returns the first entry in the list that includes P1. In this case, it is entry (1). It should be entry (2).
  - The code does not ensure that there is only one entry per channel at a given time.

Issues with keeping entries unique

- In noisePSD_decider.py lines 450 – 587 and 649 - 785, I have included a lot of code to try to work around these type of scenarios
- However, it may not be complete, as these scenarios are involve very specific sequences of events.
- It also seems to be very costly computationally – it take several minutes to read in settings and upload them to the database

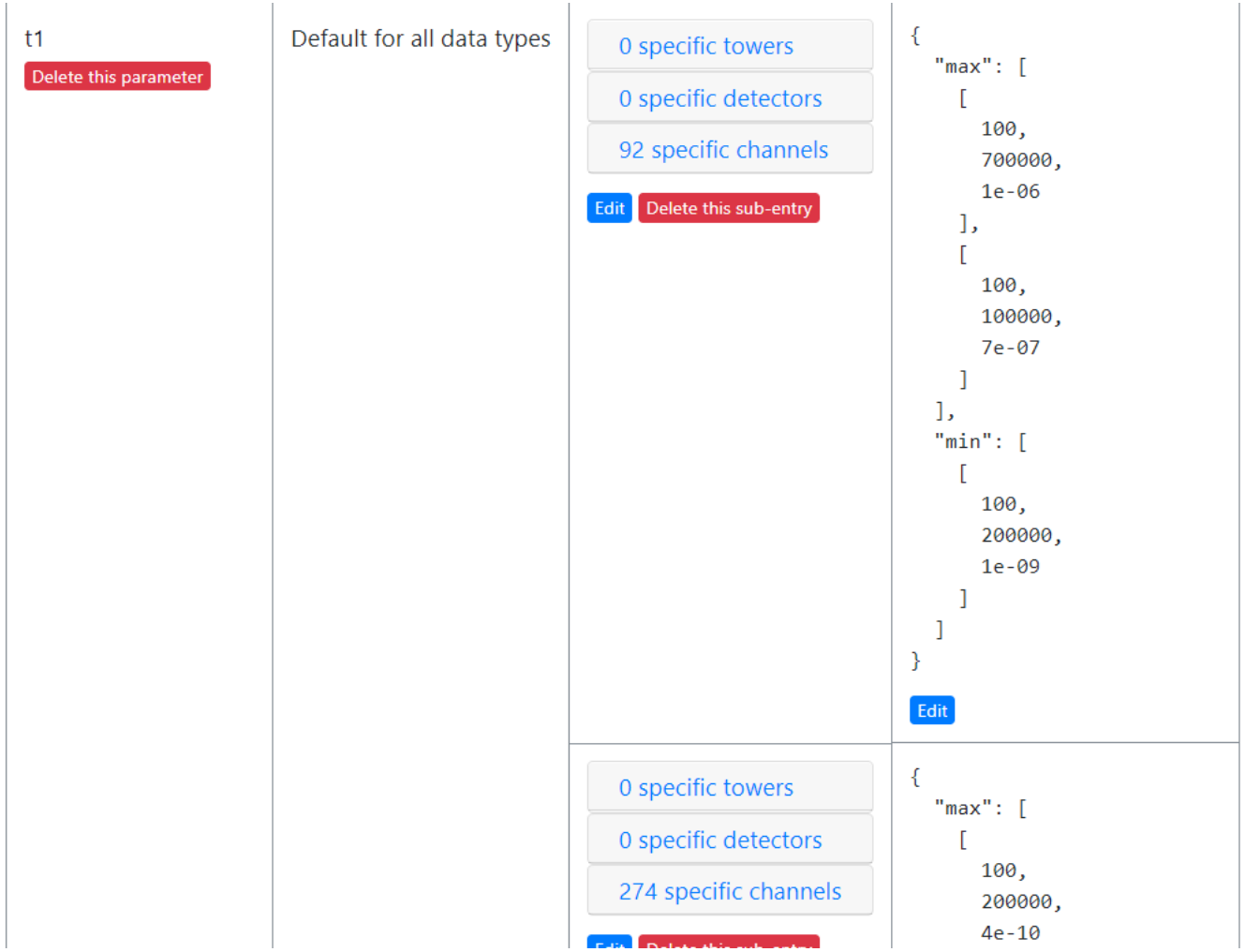
Tracking Settings

- The issues I previously discussed may be part of a broader concern of tracking settings over time.
- Ideally, the database should hold a “current” version of the settings, so when I call upon the settings to make a decision, I get the most recent version of the settings.
- Furthermore, the settings themselves should be logged over time (I know that there is a log of when parameters are added for a channel/detector/tower, but the actual setting values themselves should be logged as well)
- It would be good to know what version of the settings were used at any given time when looking back.
- My apologies if this is already done! In this case, I just haven’t been able to see where settings are logged.

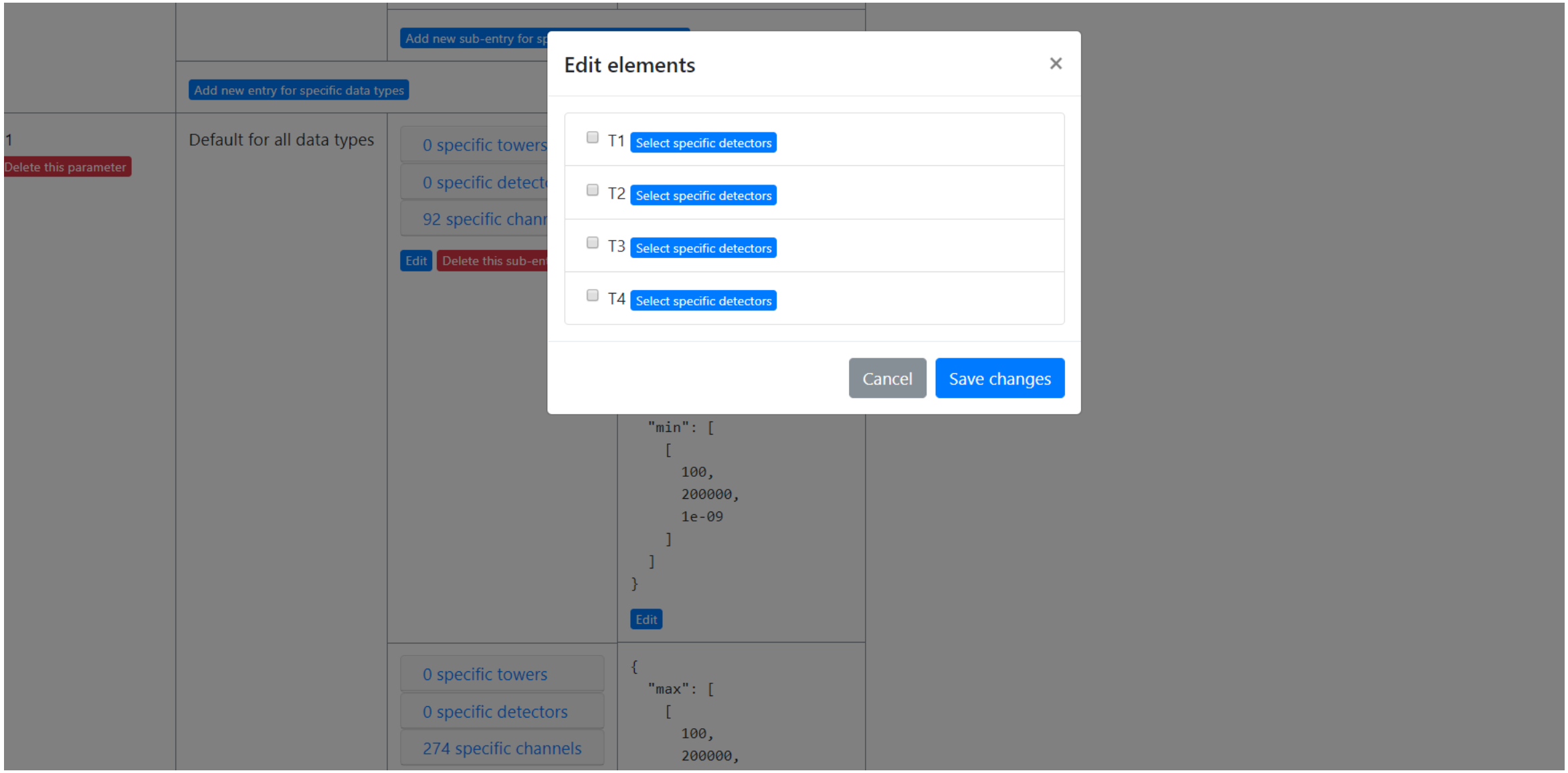
Ben’s Web interface

- It seems to connect directly to the database

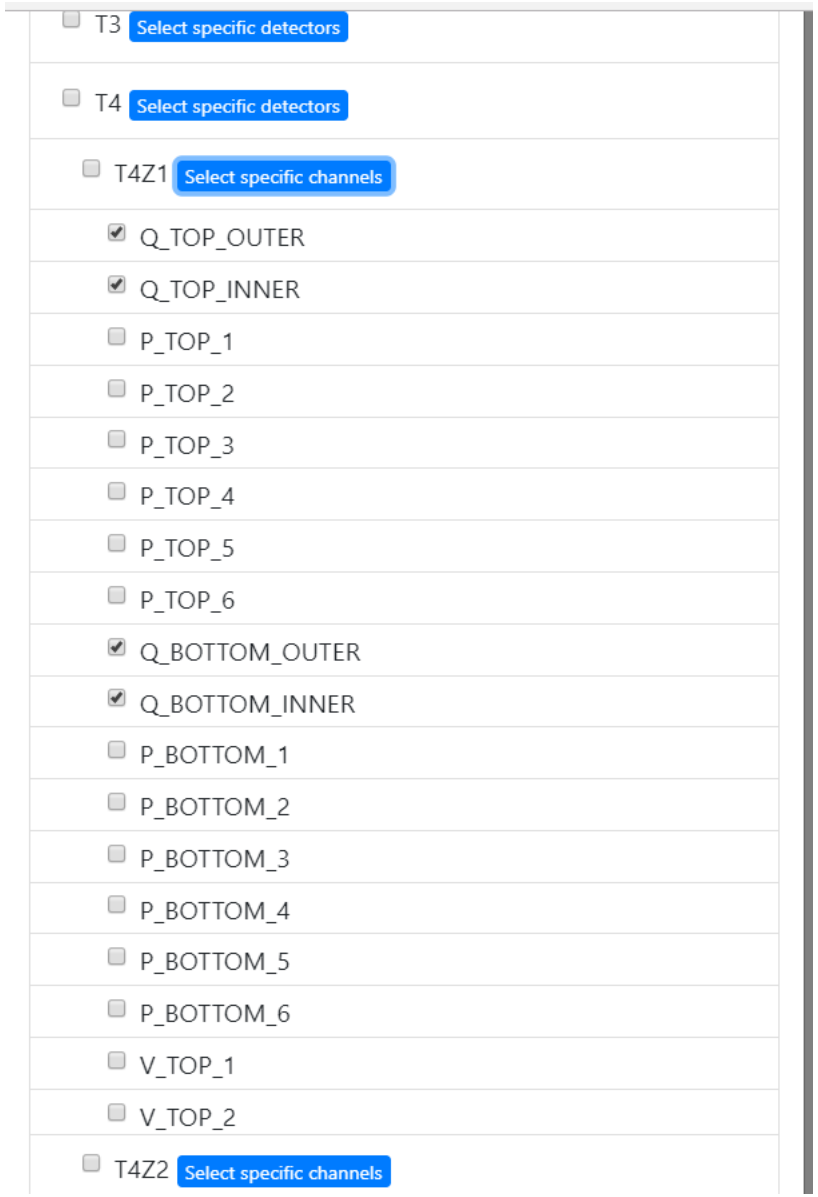
Ben’s Web interface



Ben’s Web interface



Ben’s Web interface



Ben's Web interface

- Kind of ugly for my json variable
- Difficult if you want to apply an entry to "all charge" or "all phonon"
- Just now – able to overlap element to two entries (Add two charge channels to the entry for just phonon channels).
- I think this would be approved if you weren't allowed to add an element (or child/parent element) if it overlaps with another entry
- Same with data type
- Still concerned about tracking settings versions

Dark Photon

- Start working on sensitivity code
- First steps include getting the WIMP sensitivity code to work – and understand the calculations involved

Meetings in the New Year

- Proposed meeting for tower/electronics/DAQ at SLAC in Feb
- Proposed meeting for Analysis (Dark photon) between Jan and Mar