

dunetpc - Task #14082

Use 35t real data to simulate noise for MC

10/05/2016 06:29 AM - Matthew Thiesse

Status:	Closed	Start date:	10/05/2016
Priority:	Normal	Due date:	
Assignee:	Matthew Thiesse	% Done:	0%
Category:		Estimated time:	0.00 hour
Target version:			
Description			
The 35t simulated noise does not reflect real data. It should.			
Immediate goal: determine approximate parameters for current noise model			
Intermediate goal: determine and implement parameters for the non- time-correlated noise			
Ultimate goal: include correlated noise and non-correlated noise using data-driven methods			

History

#1 - 10/05/2016 07:06 AM - David Adams

Thanks Matt for identifying this as an issue. A starting point for the desired shapes might be the plots I showed in the Sep 28 sim/reco meeting. See the brown lines on the (many) individual channel plots. We have to decide if we want to add and then remove correlated noise or just put it in at the level after noise removal.

For the latter, we might be able to use the existing ExponentialChannelNoiseService in dunetpc/dune/DetSim/Service.

#2 - 10/05/2016 07:13 AM - Matthew Thiesse

Ideally, I'd say add the correlated noise anyway. Not all analyses will want to use the correlated noise subtraction (I know I don't).

But then again, for me, probably the overall noise level (power, rms, whatever it's called) is more important than being accurate about the correlated aspects. In this case, I would probably be happy enough with the "Intermediate goal" as I've stated above.

How do we go from the plots you've made to the fhicl parameters? Trial and error?

#3 - 10/05/2016 07:17 AM - Dominic Brailsford

David Adams wrote:

Thanks Matt for identifying this as an issue. A starting point for the desired shapes might be the plots I showed in the Sep 28 sim/reco meeting. See the brown lines on the (many) individual channel plots. We have to decide if we want to add and then remove correlated noise or just put it in at the level after noise removal.

For the latter, we might be able to use the existing ExponentialChannelNoiseService in dunetpc/dune/DetSim/Service.

Ideally, we should aim to be able to run the correlated noise removal routine in the MC as the removal routine also effects the amount signal that makes it into the reconstructed hits.

There are some of us who need the routine in analysis of 35t data and so, by not having the removal routine in MC, we are introducing a potentially large systematic difference between data and MC.

So, to whatever degree its possible, we should aim to mimic the raw noise as best we can.

#4 - 10/05/2016 07:24 AM - Dominic Brailsford

Matthew Thiesse wrote:

Ideally, I'd say add the correlated noise anyway. Not all analyses will want to use the correlated noise subtraction (I know I don't).

But then again, for me, probably the overall noise level (power, rms, whatever it's called) is more important than being accurate about the correlated aspects. In this case, I would probably be happy enough with the "Intermediate goal" as I've stated above.

How do we go from the plots you've made to the fhicl parameters? Trial and error?

One option might be to inject the frequency spectrum into the MC frequency space and inverse FFT that.

That will at least produce time correlated noise; channel correlated noise will require more thought.

Something Tom Junk also suggested to me is to overlay data noise in the MC. That would obviously include both, but we would need a lot of noise data to get decent event-to-event variation.

Something else that **might** work (not given it a lot of thought yet) is to produce frequency space plots and phase plots for every channel for many events. You could possibly take a count average of the frequencies and phases as well as producing a covariance matrix. You could then throw values based on those averages and the covariance matrix.

Though, thinking about it, such a covariance matrix would be very very big...

#5 - 10/05/2016 07:33 AM - Matthew Thiesse

Something Tom Junk also suggested to me is to overlay data noise in the MC. That would obviously include both, but we would need a lot of noise data to get decent event-to-event variation.

This is something I was thinking about too. Large quantities of noise data is no problem for us (right?). Even if we use the triggered data (with real signal) to overlay on top of the MC, it should be even more accurate as we would need to still separate the triggered track from the "background" tracks. Just don't overlay it with the trigger t0 equal to the simulated t0.

Randomly choose a tick window in a randomly chosen event in a randomly chosen run (with some assumption about whether the run is "good" or not), and just add the signals?

Can't get more "data-driven" than this...

#6 - 10/05/2016 08:30 AM - David Adams

I think there are a lot of steps between the intermediate and ultimate goals in the initial message. In particular, I think it is important to have roughly the right time correlation (i.e. frequency spectrum) in addition to having the correct overall RMS. Also note there are large channel-to-channel variations in the latter.

I like the idea of directly using the data to obtain the background. We get both the electronic and cosmic background and the correct time and channel correlations and variations. We can do many studies with single muons (or other particles) instead of the full cosmic simulation. Ideally this BG would be obtained from unbiased (or minimally biased) data, i.e. not triggered with the scintillators. Do we have such data? I suspect not.

The trigger times are in channels 4000-5000 in some events and 5000-6000 in others (see my slides from the March 16 35-ton sim/reco meeting). We can get minimally biased BG from scintillator-triggered data by using channels 7000-15000. This implies using a window of only 8000 channels in the simulation but I think this will be fine for most or all studies.

I would like to have some discussion of this in the 35-ton sim/reco meeting today and I will put up some slides. If anyone else wants to do the same or send anything to me, please do so. The meeting is in 90 minute and so there isn't much time.

da

#7 - 10/07/2016 02:53 AM - Matthew Thiesse

Thanks David for bringing this to the attention of the whole group on Wednesday.

If I am correct in remembering, there were no serious objections to using the data-driven method to simulate the noise. It should accurately portray the real noise situation, and all the features, including correlations, coherences, cosmic background, etc. To mitigate the possible bias of *triggered* cosmic particles, we can take the last 5200 ticks from each split event (which is the default window length for MC with the half-field, anyway).

Can we decide if this is the "preferred" option? Is it worth messing with the ExponentialChannelNoiseService if we can address all the possible noise sources at once with the data-driven option?

I've been thinking about the implementation of such a thing, and there are a couple of issues/tasks to address.

1. Decide which real data events we would like to sample from
 1. i.e. Do we want high-purity, low-purity?
 2. Do we care whether all components (e.g. SSPs) exist in the event?
 3. Does the trigger configuration matter (i.e. TSU PTB, SSP, telescope muons, etc.)?
2. Set up a database for efficient sampling of event-by-event noise
 1. Such a database would be huge, and probably slow. Any ideas?
3. Inject the noise into the simulation with appropriate normalizations
 1. Is it as simple as adding ADCs?
 2. Sample an event randomly from the database? This is my assumption, maybe this isn't the best.
4. Set up an alternate particle generator for simulations
 1. I've talked with Karl about this and we think this can be done with CRY: by adjusting the parameters, we can ensure only 1 particle is produced in the event, and at time = 0. It's a bit of a fudge, but it seems to work on first inspection. This is desired because the "real" data we are injecting as "noise" will already have cosmic backgrounds and we don't want double cosmic flux.
5. Improve the simulation of signals on wires -- David?

Anything I've forgotten?

Any comments to add?

#8 - 10/07/2016 04:53 AM - Dominic Brailsford

Matthew Thiesse wrote:

Thanks David for bringing this to the attention of the whole group on Wednesday.

If I am correct in remembering, there were no serious objections to using the data-driven method to simulate the noise. It should accurately portray the real noise situation, and all the features, including correlations, coherences, cosmic background, etc. To mitigate the possible bias of *triggered* cosmic particles, we can take the last 5200 ticks from each split event (which is the default window length for MC with the half-field, anyway).

Can we decide if this is the "preferred" option? Is it worth messing with the ExponentialChannelNoiseService if we can address all the possible noise sources at once with the data-driven option?

I've been thinking about the implementation of such a thing, and there are a couple of issues/tasks to address.

1. Decide which real data events we would like to sample from
 1. i.e. Do we want high-purity, low-purity?
 2. Do we care whether all components (e.g. SSPs) exist in the event?
 3. Does the trigger configuration matter (i.e. TSU PTB, SSP, telescope muons, etc.)?
2. Set up a database for efficient sampling of event-by-event noise
 1. Such a database would be huge, and probably slow. Any ideas?
3. Inject the noise into the simulation with appropriate normalizations
 1. Is it as simple as adding ADCs?
 2. Sample an event randomly from the database? This is my assumption, maybe this isn't the best.
4. Set up an alternate particle generator for simulations
 1. I've talked with Karl about this and we think this can be done with CRY: by adjusting the parameters, we can ensure only 1 particle is produced in the event, and at time = 0. It's a bit of a fudge, but it seems to work on first inspection. This is desired because the "real" data we are injecting as "noise" will already have cosmic backgrounds and we don't want double cosmic flux.
5. Improve the simulation of signals on wires -- David?

Anything I've forgotten?

Any comments to add?

A couple of extra things to consider. Do we literally overlay noise in time? If so, we need to be careful about the tick size of an event compared to the noise overlay. It might be easier to inject the frequencies and phases from a noise event into an MC event and then FFT said spectrum into time space.

As mentioned in the meeting, if you don't want to have a cosmic background in the noise event you want to overlay, try looking at the data collected after the bellows pipe burst and the Ar purity died; everything was still switched on but the signal should have been destroyed.

If we go with noise events where there is a cosmic background, I think we will need to generate noise 'models' for every PTB trigger.

#9 - 10/07/2016 05:07 AM - Matthew Thiesse

A couple of extra things to consider. Do we literally overlay noise in time? If so, we need to be careful about the tick size of an event compared to the noise overlay. It might be easier to inject the frequencies and phases from a noise event into an MC event and then FFT said spectrum into time space.

As mentioned in the meeting, if you don't want to have a cosmic background in the noise event you want to overlay, try looking at the data collected after the bellows pipe burst and the Ar purity died; everything was still switched on but the signal should have been destroyed.

If we go with noise events where there is a cosmic background, I think we will need to generate noise 'models' for every PTB trigger.

I see your point about the tick size of events. I was under the impression that we would, as you say, literally overlay noise in time. What issues would arise if we were to simply set $\text{simADC}[\text{tickN}] = \text{dataADC}[\text{tickN}]$? It's a 1:1 correspondence, isn't it? (as long as the simulation sets the tick length as 500ns, same as data, which it does)

I do want cosmic background, not sure if everyone does, maybe it's worth including the option for both?

This is getting to be a big project...

#10 - 10/07/2016 05:17 AM - Dominic Brailsford

Matthew Thiesse wrote:

I see your point about the tick size of events. I was under the impression that we would, as you say, literally overlay noise in time. What issues would arise if we were to simply set $\text{simADC}[\text{tickN}] = \text{dataADC}[\text{tickN}]$? It's a 1:1 correspondence, isn't it? (as long as the simulation sets the tick length as 500ns, same as data, which it does)

I think that would do it. We will get more mileage out of the noise events if we pick a random tick to start from in the overlay. So something like `simADC[tickN]+=dataADC[tickN+offset]`, where the offset is randomly chosen at the start of the event. We would just need to make sure that we don't run over the end of the tick vector.

I do want cosmic background, not sure if everyone does, maybe it's worth including the option for both?

Why do you need the cosmic background in the noise event?

#11 - 10/07/2016 05:27 AM - Matthew Thiesse

Why do you need the cosmic background in the noise event?

In general, I want cosmic background because it affects the track finding efficiency in the case where multiple tracks happen in the same window (which, looking at the data, happens a lot more often than I would have originally guessed). But specifically, I suppose I don't necessarily need cosmic background in the noise event. The cosmics could be simulated, but then we have to still worry about the signal shape simulation.

On a more practical note: there's a lot of data with cosmic background. Also, the noise characteristics we want to replicate occur during the data-taking run, not before or after, especially since the noise changes so much from run-to-run. So if we sample the noise from very early on (e.g. `run<10000`), or after the data run, we shouldn't assume we have the same noise characteristics as the "good" data runs.

I guess the point is that it's easier to put the cosmic background in the noise event, rather than cherry-pick events without cosmic background.

#12 - 10/07/2016 08:23 AM - David Adams

Here are my thoughts on the above.

To keep things simple, I am just thinking of directly overlaying data on MC:

```
sigMC[i] += sigData[i+offset]
```

If offset is chosen large enough to get us past any effect from the track that caused the trigger, then we have unbiased BG including both electronic noise and cosmics. I am inclined to think of the latter as a good thing. Of course we are then limited to a MC window size of 15k-offset ticks. I forget the drift length at half field but I **think** it is around 4000 ticks. If we take `offset=9000`, we get 6000 ticks for the MC. We can put the MC trigger at tick 1000 to give some protection from residual trigger signal.

I would just read the data files directly to get the BG. And use the same run and event numbers (or maybe with offsets) in the MC and data so we can easily use the same pedestals and bad channel lists for MC and data. Of course we should not put MC signal in the bad channels but it shouldn't matter as these are filtered out in reco. I would not include the bad channels in the MC. As for which runs to use, we can choose the same ones people are using in their analyses.

#13 - 10/07/2016 08:34 AM - David Adams

Stuck bits are a difficult issue here (and everywhere else). Although it is not exactly what we want, I suggest to keep bits stuck when they are found so in the BG. Easiest to do this by simply using the BG value:

```
if ( stuck ) sigMC[i] = sigData[i+offset]
else sigMC[i] += sigData[i+offset]
```

The MC is (implicitly) zero where there is no signal and so this only affects the signal region. In the signal region, the MC stuck bit rate will be a bit worse than that in data and won't have the right correlation with corrected raw signal count but it is probably much better than the simulation we have now for stuck bits. Especially for the channel-to-channel variations.

If we ever have an accurate simulation of the stuck bits (i.e. with the correct rates and correlations for individual channels), we can switch to interpolated data for the BG followed by that stuck bit simulation. But I doubt we will have such a thing for 35-ton data.

#14 - 10/08/2016 08:43 AM - Matthew Thiesse

Going back to the list of runs from which to sample the noise: I'm going to start working on making the "database". Based on the run numbers in <https://cdcvns.fnal.gov/redmine/projects/35ton/wiki/35tCosmicGoodRunList>

Here is a list of my current criteria for determining a good run/event:

- PTB trigger (any type, they should all be equally unbiased for the last couple thousand ticks in the event, regardless)
- X Fraction of RCEs functioning and reading data (60%?)
- "Low noise state"
- exactly 10k ticks of data present following the trigger from all functioning RCEs
- No deflector voltage (if someone actually wants this in future, they can put it in)

The format of the database will be very, very crude. It would be ideal (to keep organization less complicated and to save space) to read the data runs

from sam during the detsim stage of simulation, but I don't know how to do that. So I'll probably make a binary file of the event waveforms and put it in /pnfs/dune/persistent. Then during detsim, the run/event number will be selected from random, then the corresponding file read and overlaid onto the event. If anyone has a better idea, I'm all ears.

#15 - 10/10/2016 07:40 AM - Matthew Thiesse

Quick update: I'm testing the module which will create the "database" of waveforms. I've decided to go with a ROOT file, which is so much easier than raw binary, go figure.

I'm working on feature/mthiesse_datadrivennoise.

David: Since you're quite intimate with DataPrep and DetSim services, could you please work on the adding of data waveforms to the MC generated waveforms? Assume for each event you will read in a ROOT file containing a TTree. This TTree will have five branches: run number (int), subrun number (int), event number (int), offline channel number (int), and signal waveform (std::vector<float>). Run number, subrun number, and event number will be decided by whatever random algorithm we decide, and will determine which file to read, which will associate the desired real data set to be added to the simulated event. It would make sense for this to be a brand new MC dataset (as opposed to milliblock and non-milliblock), so we'll create events of 5200 ticks.

I'll make a text file which contains two columns: run, event. One line of this file will be selected at random (seeded by LArSeedService), the file will be read, and so on. The filenames for reading have the format "runXXXXX.root" where XXXXX is the 5-digit run number.

If there are any objections, as always, let me know. But otherwise, this is the plan I'm working towards.

Thinking ahead, we also need to make a new CRY generator fcl file which ensures one particle simulated at T0. And since we are really only bothered about looking at counter-triggered muons, we could pipe this through the normal filters. Again, volunteers?

#16 - 10/10/2016 08:51 AM - Matthew Thiesse

An example file is at /dune/app/users/mthiesse/run13770.root

#17 - 10/10/2016 09:49 AM - David Adams

Matt:

A few comments:

1. I think we should keep the run number (and event number and time) from the data. This way we can use the pedestals, bad channels and struck bit services from the data processing.
2. I don't think we need to copy the data to root files but can just read the raw data files directly. We should check with the art people if it is easy to have two event input files. This way we avoid making copies that will be as large as the original files. And it is easy to make changes in the use of the input data, e.g. changing the offset.

#18 - 10/10/2016 10:31 AM - David Adams

Instructions for event mixing are here: https://cdcvns.fnal.gov/redmine/projects/art/wiki/Product_Mixing

#19 - 10/11/2016 04:12 AM - Matthew Thiesse

David:

The code to write the files is done, it works, and it's currently running. I agree that doing it the ART way is better, but unfortunately I don't have the time to spend implementing it, especially not when the stake is a mere couple of TB of /pnfs/dune/persistent, at most. If you (or someone else) would like to do this, then by all means, go ahead.

Would you still be able to tell me where and how a vector of floats should be inserted into DetSim?

#20 - 10/11/2016 06:55 AM - David Adams

Matt:

You need only make ChannelNoiseService as you did for ExpoWhiteChannelNoiseService.

#21 - 10/18/2016 04:33 AM - Matthew Thiesse

I think that we should NOT use the method David has mentioned to deal with stuck bits (i.e. if (stuck) {outtick = stucktick} else {outtick += intick})

The stuck bit probability seems to depend on the measured analogue signal -- if within a small range of voltages around 0x3f, the ADC is more likely to get stuck there. But, if a signal exists in the region, and the voltage raises significantly, the ADC will not necessarily get stuck at the same place anymore. So, by imposing this restriction on the addition of data and MC, we would get unrealistic signals. For example, an MC event could have a waveform 132213478951223112, which clearly has a pulse in it. If the data event we happen to overlay has 241222221232222412222, and if we impose that for every stuck 2 in the real data, we claim that the mixing of events should have a 2 regardless of the MC signal, then we lose the pulse almost completely. Or worse, the pulse is split into many smaller pulses. This is not right.

If MC with no noise injected is added to real data, the stuck codes will still exist, but the signal is still kept, as in real data. I think just a simple addition

is more realistic, unless anyone has a better idea.

#22 - 10/18/2016 04:41 AM - Matthew Thiesse

Also, I've copied the product mixing code from microboone (didn't know it existed before Tom pointed it out). Yes, it is better than my previous method, but it isn't fully mature yet -- not production-ready. It's been stripped and DUNE-ified in feature/mthiesse_mcdatamix. Please take a look at it and help the development move forward.

The TODO list is as follows:

- Make some sort of run/subrun/event selection based on low-noise events, and choose the corresponding file in SAM, no need for randomization as long as "high-noise-state" runs are not selected
- Select ticks > ~6000 from the real data waveforms (I think probably in the RawDigitMixer.cxx/h files)
- Implement mixing for OpDetWaveforms in 35t (I have no idea how). What about AuxDet stuff???

#23 - 10/18/2016 07:03 AM - David Adams

Matt:

I agree that my proposal for handling the stuck bits is not perfect but I don't think we can do better without a good simulation of the bit sticking and I don't think we are going to get that. I agree we don't get the right correlation with the true signal bit pattern but we will get the right rate including channel-to-channel variations. I would at least like to have it as an option for the simulation. --da

#24 - 10/18/2016 09:20 AM - Matthew Thiesse

It's about 90% complete now. Check out feature/mthiesse_mcdatamix and you can try mixing some events. There is a fhicl file in dune/DataOverlay/DataOverlayMixer/run_DataOverlayMixer.fcl which you can run on a detsim stage. Look at "daq" in the event display on the detsim file, then look at "mixer:TPC" in the mixed root file. Pretty neat. (Make sure detsim stage has 5200 ticks and the split data file has 15000 ticks, or change RawDigitMixer.cxx to accommodate more general values)

David: your fhicl parameter exists in OverlayRawDataDUNE35t_module.cc for doing the stuff about stuck bits.

Still to do:

- Put together a list of input real data files which contain the noise runs/subruns/events we want to sample and edit OverlayRawDataDUNE35t to only sample those.
- Sort out a MC generator which creates a MC particle at time t=0 (or near enough) but also creates no additional cosmics. Can be done with CRY -- probably just needs messing with the fhicl parameters in cry_dune.fcl

I'm off for today, but please feel free to attack these TODO items and anything else I've forgotten.

#25 - 10/20/2016 09:31 AM - David Adams

- *Subject changed from Make simulation noise parameters reflect real data to Use 35t real data to simulate noise for MC*

I have change the title of this issue to better reflect the direction taken here. I will open a new ticket for the original idea of tuning the MC noise simulation to match the noise spectrum observed in data.

#26 - 10/20/2016 10:32 AM - David Adams

- *Status changed from New to Assigned*

- *Assignee set to Matthew Thiesse*

#27 - 11/02/2016 09:05 AM - Matthew Thiesse

As I will discuss in the meeting today, this project seems to be finished. The noise addition works (Dom is verifying), and the code works in a grid job. In other words, the grid job can access and read the production sliced data files from the 35ton run via SAM.

Once verified, I'll close this issue.

#28 - 11/11/2016 03:10 AM - Matthew Thiesse

- *Status changed from Assigned to Closed*