Changes to the Simulations Needed for Background Integration

Kolja Kauder (BNL NPPS)

TIC Meeting on Backgrounds

September, 7 2023

Background Embedding

Sources of Background:

- beam gas (FXT), 6+ events per 2 ms time slice (integration window)
- Synchrotron radiation (from the project), <photons/ts> ~ 5000
- Coming: Bethe Heitler Bremsstrahlung for the lumi detector and low-Q^2 tagger, <e+g pairs / ts> ~ 4000
  - (requires updates: faster detectors and tighter correlation to bunch crossing)

Method:

- Merge individual sources at the HepMC3 level, i.e. as the input into Geant4/npsim (could consider merging after Geant4)
- Started and continued in Python https://github.com/eic/HEPMC_Merger/tree/koljadev
  - Good for integration with dd4hep; problematic because of volatile memory footprint
- Time Slice model is naturally consistent with streaming readout

Usage for Signal

```
% python3 ./signal_background_merger.py --help

Merge signal events with up to three background sources.

options:
  -i SIGNALFILE, --signalFile SIGNALFILE
          Name of the HEPMC file with the signal events


  -sf SIGNALFREQ, --signalFreq SIGNALFREQ
          Poisson-mu of the signal frequency in ns.
          Default is 0 to have exactly one signal event per slice.
          Set to the estimated DIS rate to randomize.
```

Single particles, PYTHIA, …

Option:
DIS (or so) **freq.** from the Wiki

Default: Exactly **one event** in every time slice.
Could add: At least one event / slice (to exclude pure BG)

Poisson determines **how many** events in a slice. **"Position"** in the slice is **uniformly random**

- It's possible this should be 0, or the mid-point, depending on how the DAQ "triggers"

Usage for FXT BG

```
options:
  -bg1 BG1FILE, --bg1File BG1FILE
        Name of the first HEPMC file with background events
  -bf1 BG1FREQ, --bg1Freq BG1FREQ
        Poisson-mu of the first background frequency in ns. Default is
        the estimated hadron gas rate at 10x100. (Set to 0 to use the
        weights in the corresponding input file)
```

Ex.: h-gas

From the Wiki

See SR slide

- **Poisson** determines **how many** events in a slice. **"Position"** in the slice is **uniformly random**
- Same options, same meaning for `-bg2, -bf2` (Ex.: e-gas)
- Input files **"roll over"** when the end is reached. This could lead to artifacts. Randomizing (i.e. jumping around in the HepMC file) is very inefficient but possible. Better to generate a large enough background pool (though that's a lot of disk space).
  - Better yet to generate events on the fly

Usage for SR BG

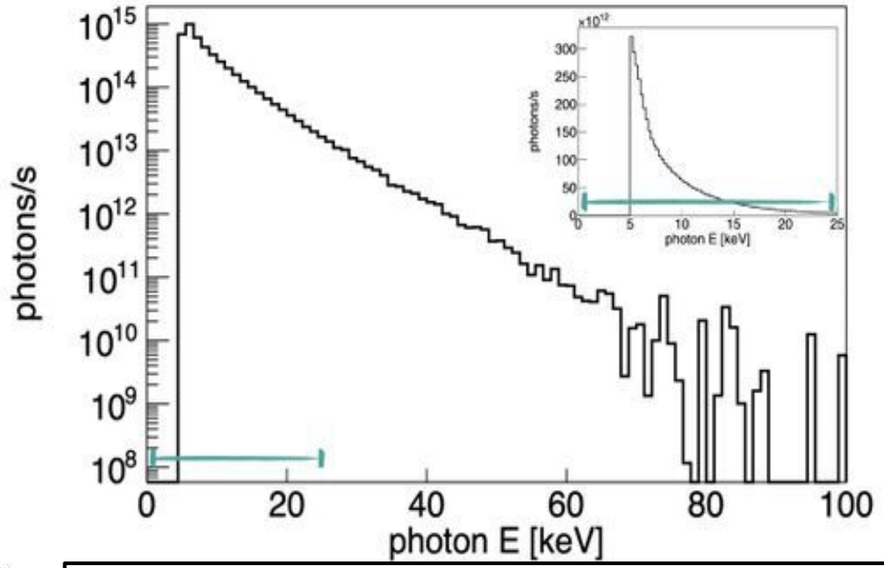
```
options:
  -bg3 BG3FILE, --bg3File BG3FILE
        Name of the third HEPMC file with background
        events
  -bf3 BG3FREQ, --bg3Freq BG3FREQ
        Poisson-mu of the third background frequency in
        ns. Default is 0 to use the weights in the
        corresponding input file. (Set to a value >0 to
        specify a poisson mu instead.)
```

Ex.: Synchrotron Radiation

From SynRad

- **Poisson** determines **how many** events in a slice. **"Position"** in the slice is **uniformly random**
- Details deserve their own slide

Synchrotron Radiation details



Spectrum from SynRad.
Important: Internally, this "histogram" is a lookup table for 180k individual SR photons

- Each photon in SynRad's output comes with its own **rate R_p**
- Use the **average rate $\langle R_p \rangle$** as **μ** in a Poisson distribution to determine the **number N of SR photons** in a given slice
- Using **rate as weight**, draw N individual photons from the spectrum and place them uniformly

- Weighted draw means the entire list needs to be in memory

Connection to thresholds

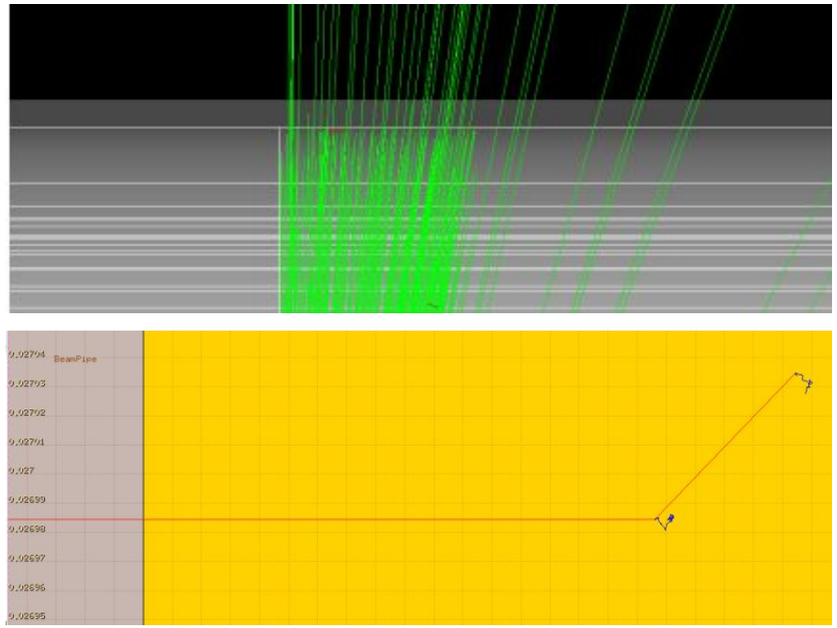
- First tests in track reconstruction showed remarkably few particles reaching the tracking detectors
- Ultimately tracked down to an arbitrarily set default cutoff at 5 keV in digitization from earlier tracking tests
- The fact that this cutoff coincides with the low end of the SR spectrum we have was a confusing red herring

```
    struct SiliconTrackerDigiConfig {
        double threshold  = 5 * dd4hep::keV;
        double timeResolution = 8;    /// TODO 8
in juggler. Probably [ns]
    };
```

```
+   struct SiliconTrackerDigiConfig {
+     // sub-systems should overwrite their own
+     // NB: be aware of thresholds in npsim! E.g.
  https://github.com/eic/npsim/pull/9/files
+     double threshold  = 0 * dd4hep::keV;
+     double timeResolution = 8;    /// TODO 8 of wha
  juggler. Probably [ns]
+   };
```

Important but not in this talk:

- It is also connected, before digitization, to cutoffs inside Geant4
- Controlled by a single penetration depth parameter for each species in the physics list, from there calculated internally for al materials
- Shyam has been doing a huge amount of work checking and tuning the Physics list from eAST and validating it against fluka



First attempt in Geant4 via npsim

```
62  +   # Set energy thresholds for all detectors
63  +   # Note: For most detectors we can apply this threshold either here, in
64  +   # digitization, or in reconstruction. To allow studies at each stage,
65  +   # only some of the threshold may be applied here, so the rest can be
66  +   # applied at the digitization. Ideally nothing should be applied at the
67  +   # reconstruction stage.
68  +   # Ref: https://docs.google.com/spreadsheets/d/1s8oXj36SqIh7TJeHFH89gQ_ayU1_SVEpWQNkx6sETKs/
69  +   energy_deposit_minimum_cuts = {
70  +       "VertexBarrel": "0.65*keV",
71  +       "SiBarrel": "0.65*keV",
72  +       "MPGDBarrel": "0.25*keV",
73  +       "TrackerEndcap": "0.65*keV",
```

```
87  +       "ZDC_PbSci": "100*MeV",
88  +   }
89  +   for detector, cut in energy_deposit_minimum_cuts.items():
90  +       name = f"EnergyDepositMinimumCut/{detector}/{cut}"
91  +       SIM.filter.filters[name] = dict(name=name, parameter={"Cut": cut})
92  +       SIM.filter.mapDetFilter[detector] = name
```

src/dd4pod/python/npsim.py

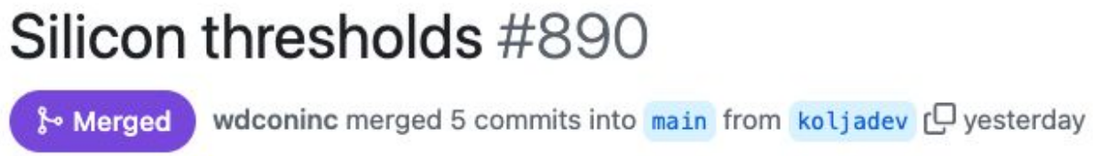
Not working: Attempt in Geant4 via npsim

- This is an attractive approach
  - Very simple
  - Keeps everything in one place, easy to check against the digitization spreadsheet
  - While not reducing simulation time, it cuts back the amount of "unneeded" data for digitization

- Problem: Energy deposition below threshold gets thrown away even though it could add up in a calorimeter
  - Could work for tracker-style detectors, but more consistent to move it all to digitization

Instead: Set thresholds in EicRecon

```
// Digitization
app->Add(new JChainMultifactoryGeneratorT<SiliconTrackerDigi_factoryT>(
    "TOFBarrelDigiHit",
    {"TOFBarrelHits"},
    {"TOFBarrelDigiHit"},
    {
        .threshold = 0.5 * dd4hep::keV,
        .timeResolution = 0.025,    // [ns]
    },
    app
));
```



Set for all individual detectors in EicRecon

- Done for the tracker style detectors
- Need feedback and details from DSCs for calo-style detectors, see call on the next slide

- Note: Time information gets propagated through Geant4 but despite the timeResolution field above is currently digitized away, working on "worst case" first

Call for Input

The following is needed from the different sub-detector collaborations:

Two major types of detectors with respect to applying thresholds in the MC

§ First type of detectors are the ones for which the energy is summed like in calorimeters.

If one particle is hitting for example a PbW04 tower the deposited energy of this one particle is summed and if two particles are hitting a tower, the energy deposited from both particles is summed. One applies the threshold on what is called a valid on the summed energy / ADC value.

§ The other type of detectors are these ones no energy is summed. In this case I have a large pixelation, such that the probability for two particles hitting the same pixel is close to zero.

In this case if the one particle does not deposit enough energy in the detector to trigger a response one gets no hit. Examples are the MAPS and also the MPGDs as they are implemented in dd4hep in a pixelized way. So here thresholds can be and should be placed is on the energy/momentum of the particle, the way it was implemented.

§ The latter also works for Cherenkov detectors, because if a particle of a certain type (pion, Kaon, proton, electron) is not above the Cherenkov threshold of the respective radiator no Cherenkov photons are produced.

Please fill in the updated digitization spreadsheet

Supplementary slides