Plugging in PulseCombiner and PulseNoise, and implementing PulseDigi

Minho Kim
Argonne National Laboratory

BIC Simulation Meeting
September 23, 2025

Argonne NATIONAL LABORATORY

FalphaNoise

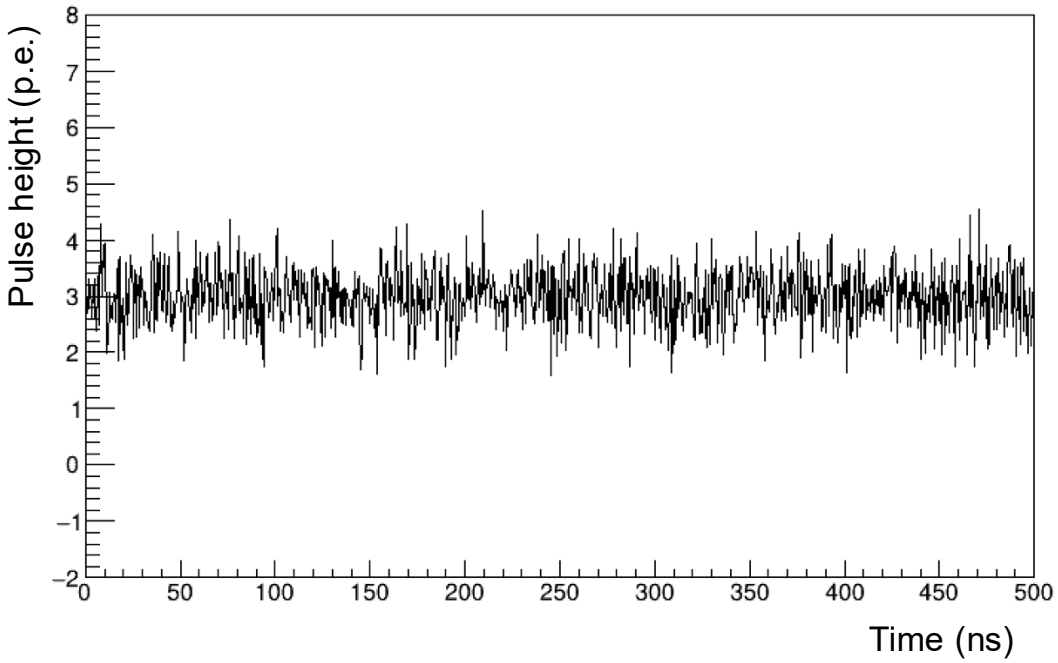
```
auto seed = m_uid.getUniqueID(*headers, name());
std::default_random_engine generator(seed);
dd4hep::detail::FalphaNoise falpha(m_cfg.poles, m_cfg.variance, m_cfg.alpha);
```

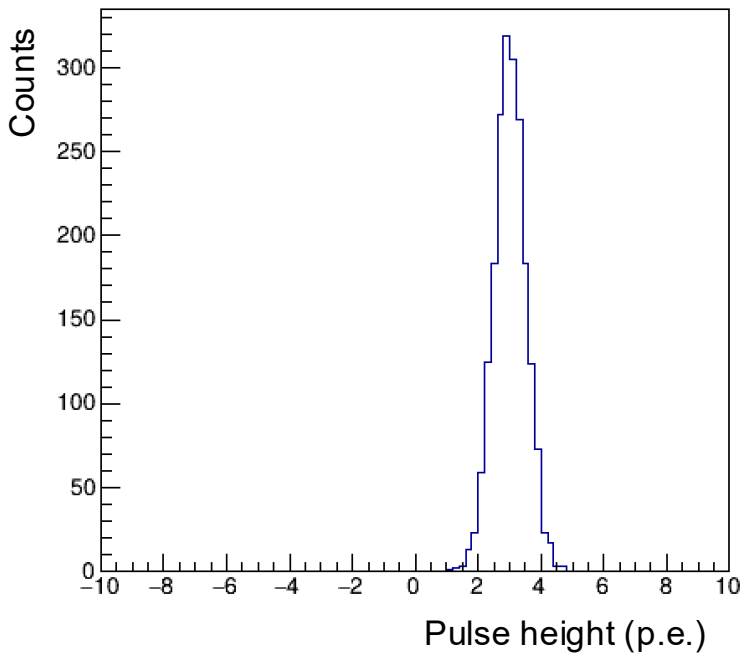
m_cfg.alpha m_cfg.variance

```
for (std::size_t i = 0; i < pulse.getAmplitude().size(); i++) {
  double noise     = falpha(generator) * m_cfg.scale;
  double amplitude = pulse.getAmplitude()[i] + noise;
  out_pulse.addToAmplitude(amplitude);
  integral += amplitude;
}
```

- PulseNoise algorithm uses FalphaNoise to generate the noise.
- If there are the following past noise samples, 0.8, -0.3, 0.1, 0.7, -0.2, 0.5, the FalphaNoise generates the current one using a random generator and
  - pole: The number of past samples that influence the current one.
  - alpha: How strongly the past samples influence the current one.
- However, the FalphaNoise provides a number, not a pulse shape.

FalphaNoise → Pedestal distribution





- Parameters were set so that we have a specific pedestal distribution (m~3p.e., σ~0.5p.e.) when the FalphaNoise is measured every 25 ns.
- An offset parameter was added to reproduce the pedestal mean.
- A scale was multiplied to convert p.e. to pulse height.

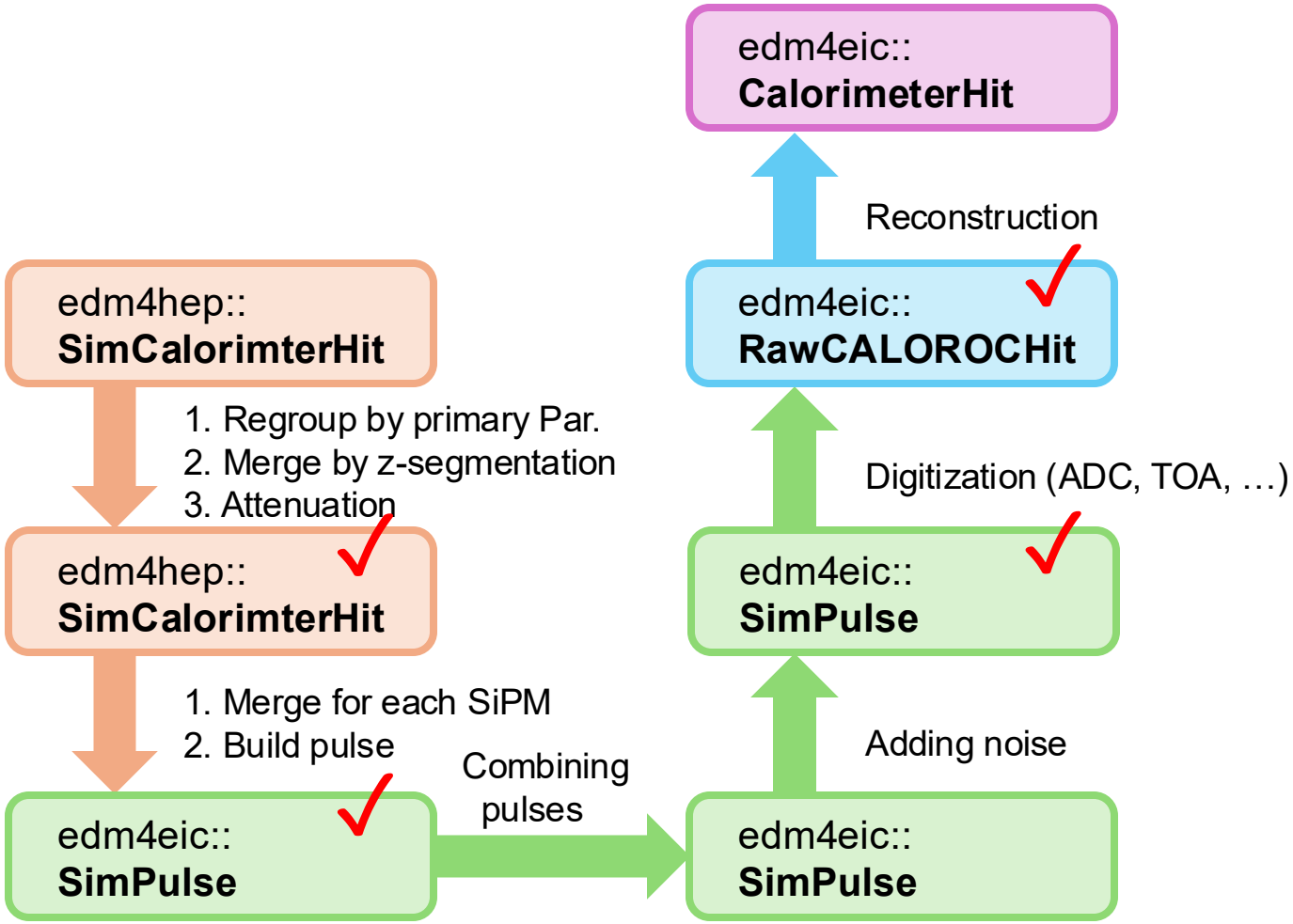
src/algorithms/digi/PulseNoiseConfig.h

```
20   20      double scale = 1000;
     21  +
     22  +   // Noise offset
     23  +   double offset = 0;
```

veprbl last week

Should we call this pedestal?

Data type to be saved for preTDR



BIC data volume (18x275_minQ2_1)

Status	Total (MB)	BIC (MB)
Without {P, N}Att.Hits and {P, N}Pulses	27	2
Add Attenuated{P, N}Hits	32	7
Zero-filling branches	30	5
Add TimeSeries {P, N}Pulses	31	6
Add TimeSeries {P, N}Comb.PulsesWithNoise	31	6
Remove RawHits and RawHitAsso.	30	5
Add {P, N}RawCALOROCHits	?	?

Zero-filling branches

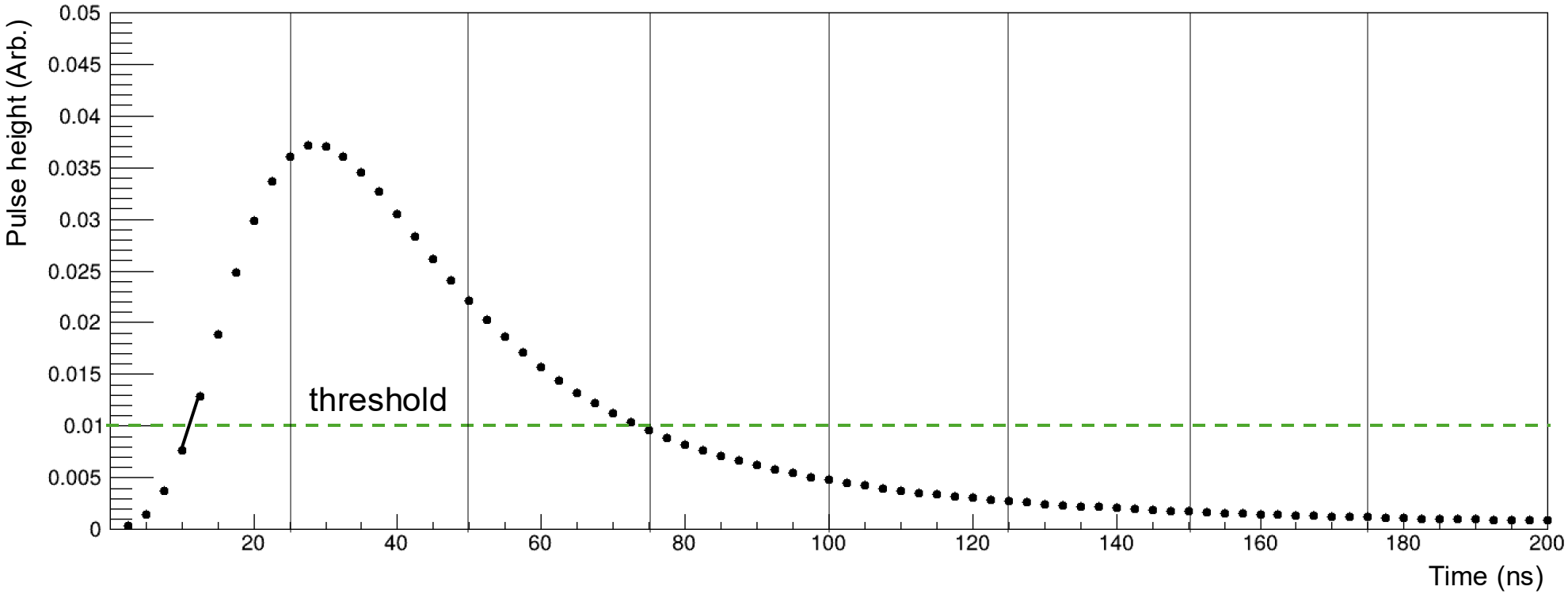
```
edm4hep::SimCalorimeterHit:
  Description: "Simulated calorimeter hit"
  Author: "EDM4hep authors"
  Members:
    - uint64_t cellID
    - float energy [GeV]
    - edm4hep::Vector3f position [mm]    (0, 0, z)
  OneToManyRelations:
    - edm4hep::CaloHitContribution contributions

edm4hep::CaloHitContribution:
  Description: "Monte Carlo contribution to SimCalorimeterHit"
  Author: "EDM4hep authors"
  Members:
    - int32_t   PDG                      0
    - float energy [GeV]                 0
    - float time [ns]
    - edm4hep::Vector3f stepPosition [mm] (0, 0, z)
    - float stepLength [mm]              0
  OneToOneRelations:
    - edm4hep::MCParticle particle
```

BIC data volume (18x275_minQ2_1000)

Status	Total (MB)	BIC (MB)
Without {P, N}Att.Hits and {P, N}Pulses	35	14
Add Attenuated{P, N}Hits	58	36
Zero-filling branches	46	25
Add TimeSeries {P, N}Pulses	53	32
Add TimeSeries {P, N}Comb.PulsesWithNoise	57	36
Remove RawHits and RawHitAsso.	48	26
Add {P, N}RawCALOROCHits	?	?

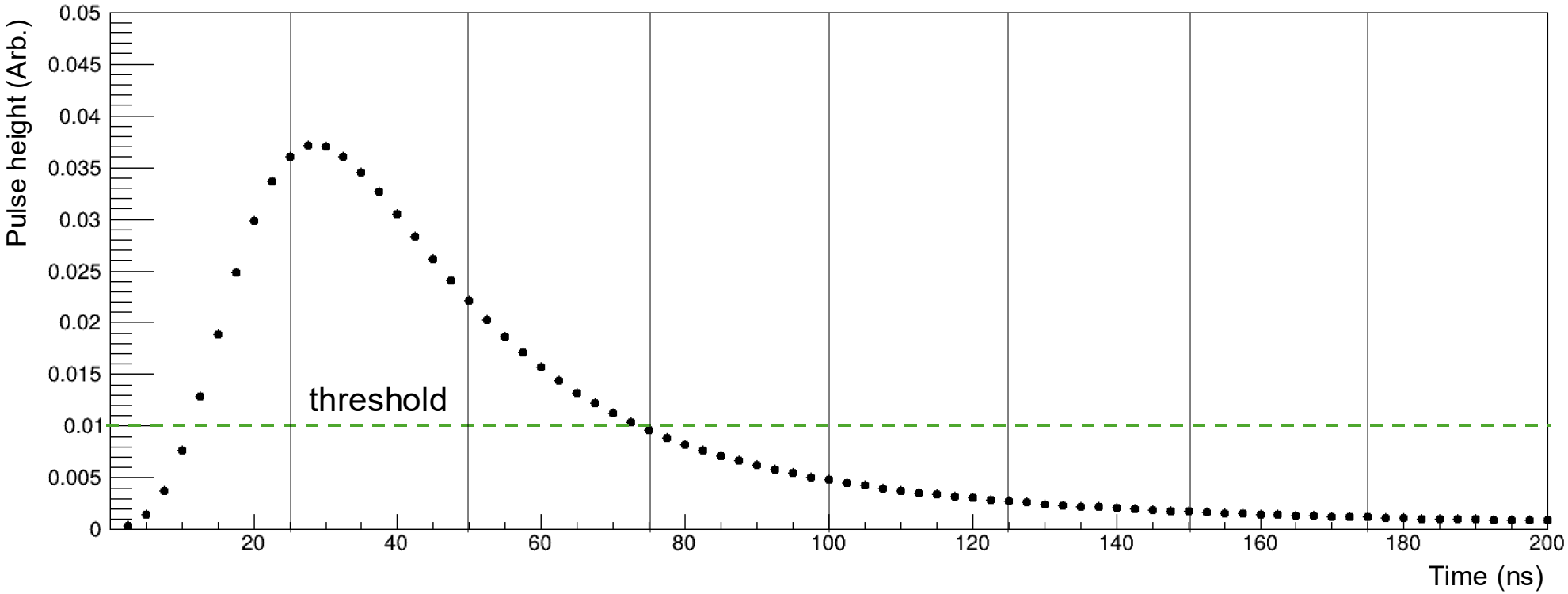
- Is it okay if the zero-filling branches is applied for BIC?

Digitization algorithm



- ADC, TOA, and TOT values are measured every 25 ns.
- Each 25 ns sample has a HGCROCSample and each pulse has a RawHGCROCHit.
- ADC is the maximum amplitude during a sample. 0 is filled to it when the pulse crosses the threshold.
- To measure the TOA and TOT, the crossing points between the pulse and threshold were calculated by linear interpolation.

Digitization algorithm




```
edm4eic::RawHGCROCHit:
  Description: "Raw hit from an HGCROC chip"

  Members:
    - uint64_t cellID
    - int32_t  samplePhase
    - int32_t  timeStamp
  VectorMembers:
    - edm4eic::HGCROCSample sample
```

```
edm4eic::HGCROCSample:
  Members:
    - uint16_t ADC
    - uint16_t timeOfArrival
    - uint16_t timeOverThreshold
    - bool     TOTInProgress
    - bool     TOTComplete
```

Digitization algorithm

```
for (const auto& pulse : *in_pulses) {
  bool sample_boundary = false;
  double amplitude_max = -std::numeric_limits<float>::infinity();
  std::vector<float> amplitudes_neighboring(2);
  double toa         = 0;
  double tot         = 0;
  bool totInProgress = false;
  bool totComplete   = false;

  auto out_digi_hit = out_digi_hits->create();

  for (std::size_t i = 0; i < pulse.getAmplitude().size(); i++) {
    double t                      = pulse.getTime() + i * pulse.getInterval();
    sample_boundary               = is_sample_boundary(t, m_cfg.sample_period);
    amplitude                     = pulse.getAmplitude()[i];
    amplitudes_neighboring[i % 2] = amplitude;
    amplitude_max                 = std::max(amplitude_max, amplitude);

    if (!totInProgress && amplitude > m_cfg.threshold) {
      toa           = get_crossing_time(m_cfg.threshold, t, t - pulse.getInterval(),
                                        amplitudes_neiboring[i % 2], amplitudes_neighboring[1 - i % 2]);
      totInProgress = true;
    }

    if (totInProgress && !totComplete && amplitude < m_cfg.threshold) {
      totComplete   = true;
      totInProgress = false;
      tot = get_crossing_time(((m_cfg.threshold - amps_neighbor[i % 2]) * pulse.getInterval()) /
                              (amps_neighbor[i % 2] - amps_neighbor[1 - (i % 2)])) +
            t - toa;
    }
```

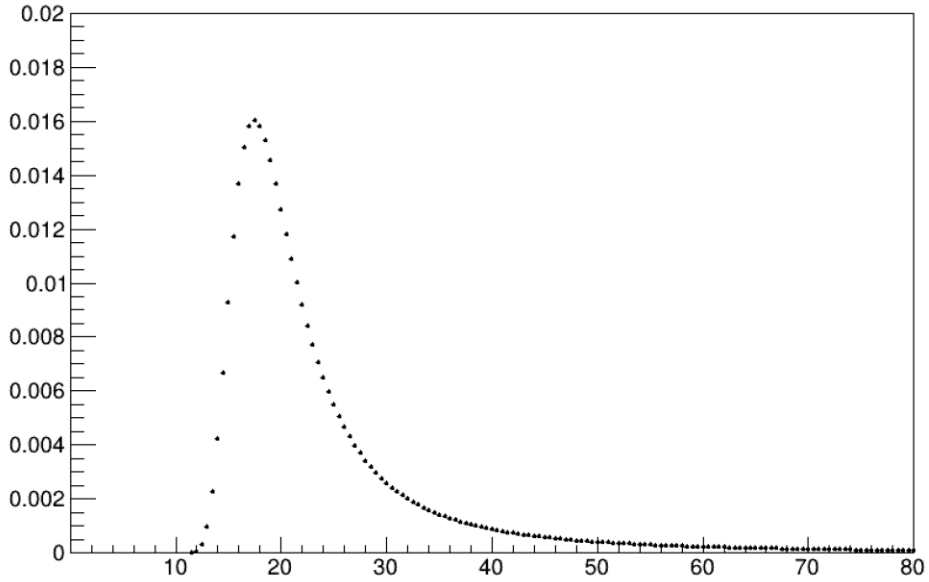
Digitization algorithm

```
      if (sample_boundary) {
        auto out_digi_sample = out_digi_samples->create();

        out_digi_sample.setADC(amplitude_max);
        out_digi_sample.setTimeOfArrival(toa);
        out_digi_sample.setTimeOverThreshold(tot);
        out_digi_sample.setTOTInProgress(totInProgress);
        out_digi_sample.setTOTComplete(totComplete);
        out_digi_hit.addToSamples(out_digi_hit);

        amplitude_max = -std::numeric_limits<float>::infinity();
        toa           = 0;
        tot           = 0;
      }
    }

    out_digi_hit.setCellID(pulse.getCellID());
    out_digi_hit.setSamplePahse();
    out_digi_hit.setTimeStap();
  }
} // PulseDigi:process
```

Discussion

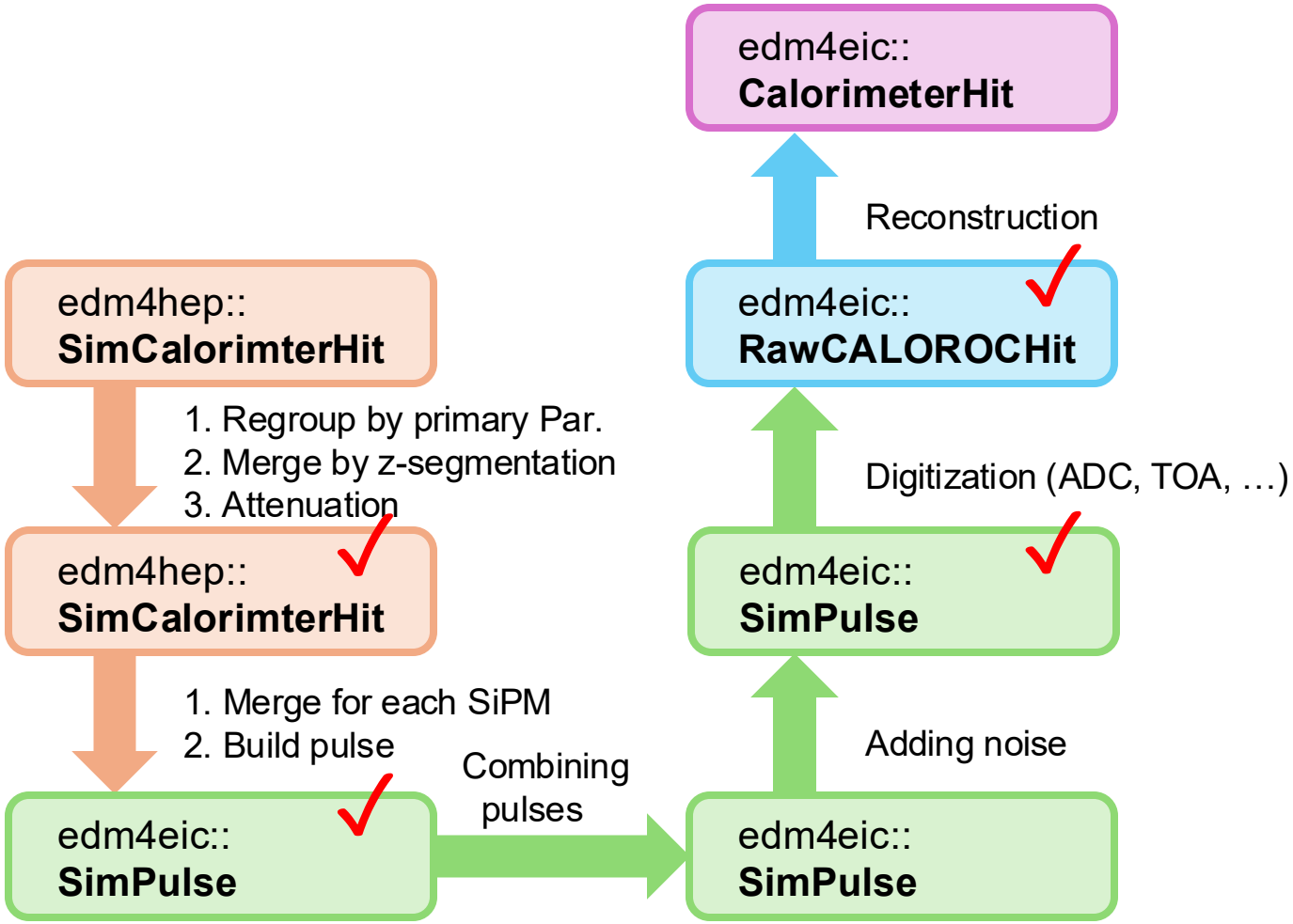
```
// get reference position for attenuating hits and contributions
if (!m_cfg.attenuationReferencePositionName.empty()) {
  m_attenuationReferencePosition =
      m_geo.detector()->constant<double>(m_cfg.attenuationReferencePositionName) *
      edm4eic::unit::mm / dd4hep::mm;
}
```

- Is it okay if a sign variable (= ± 1) is added to the Config file?



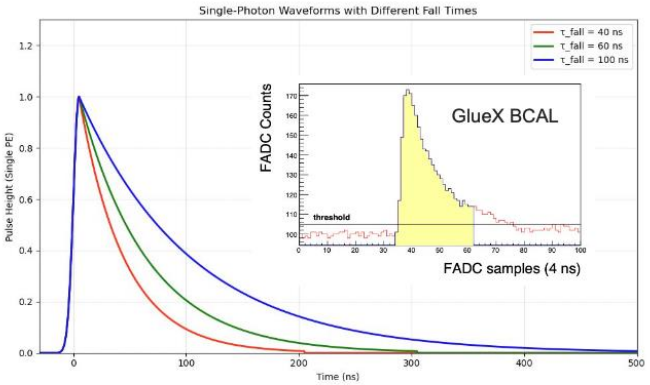
- Do we need to increase the pulse width?

Discussion


edm4eic::
CalorimeterHit
Reconstruction
edm4eic::
RawCALOROCHit
edm4hep::
SimCalorimterHit
1. Regroup by primary Par.
2. Merge by z-segmentation
3. Attenuation
Digitization (ADC, TOA, …)
edm4hep::
SimCalorimterHit
edm4eic::
SimPulse
1. Merge for each SiPM
2. Build pulse
Adding noise
Combining
pulses
edm4eic::
SimPulse
edm4eic::
SimPulse

Backup

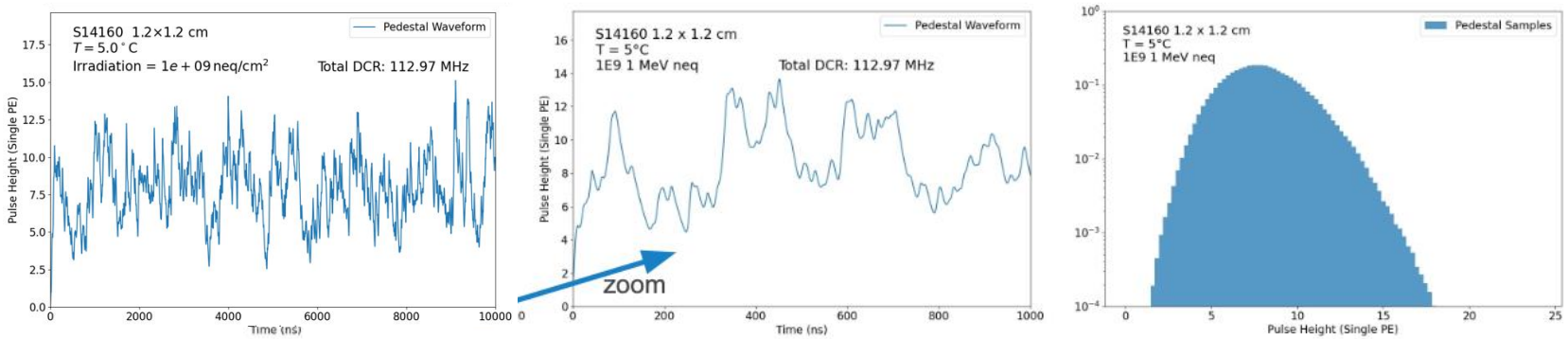
Optimizing parameters



Green template taken for the studies

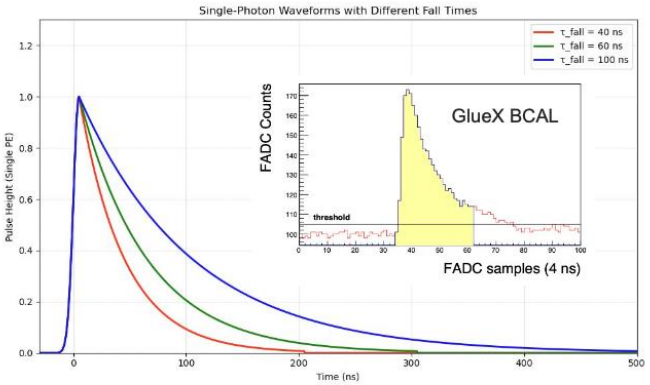
SiPM GlueX Template with 112.97 MHz DCR

S14160 SiPMs at 5 °C irradiated 1e+09 1-MeV neutron equivalent dose (Bolognia)



- However, the important thing is reproducing the pedestal mean and sigma if the HGCROC measures the amplitudes every 25 ns.
- In addition, it would be good if the PulseNoise could generate a pulse height trend similar to Henry's simulation as a function of time.

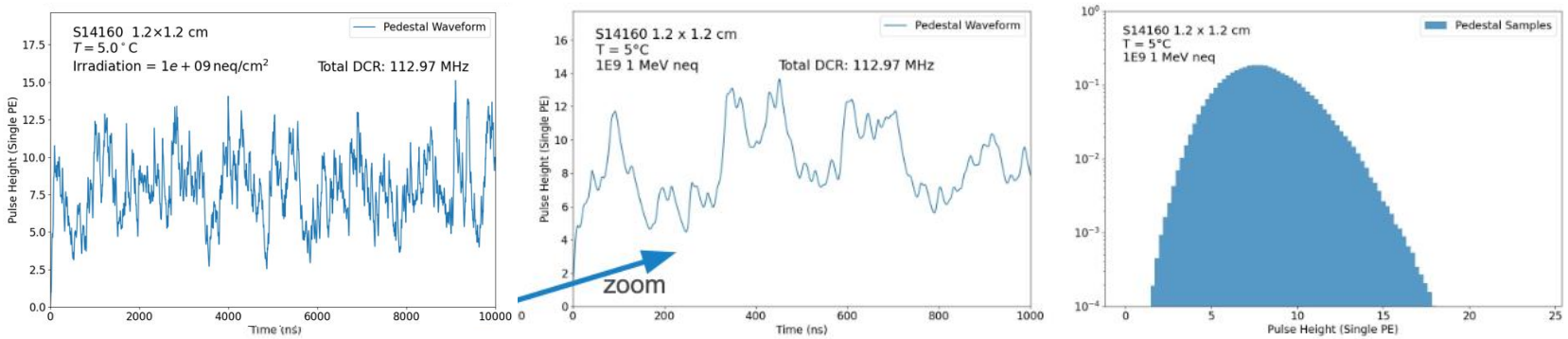
Optimizing parameters



Green template taken for the studies

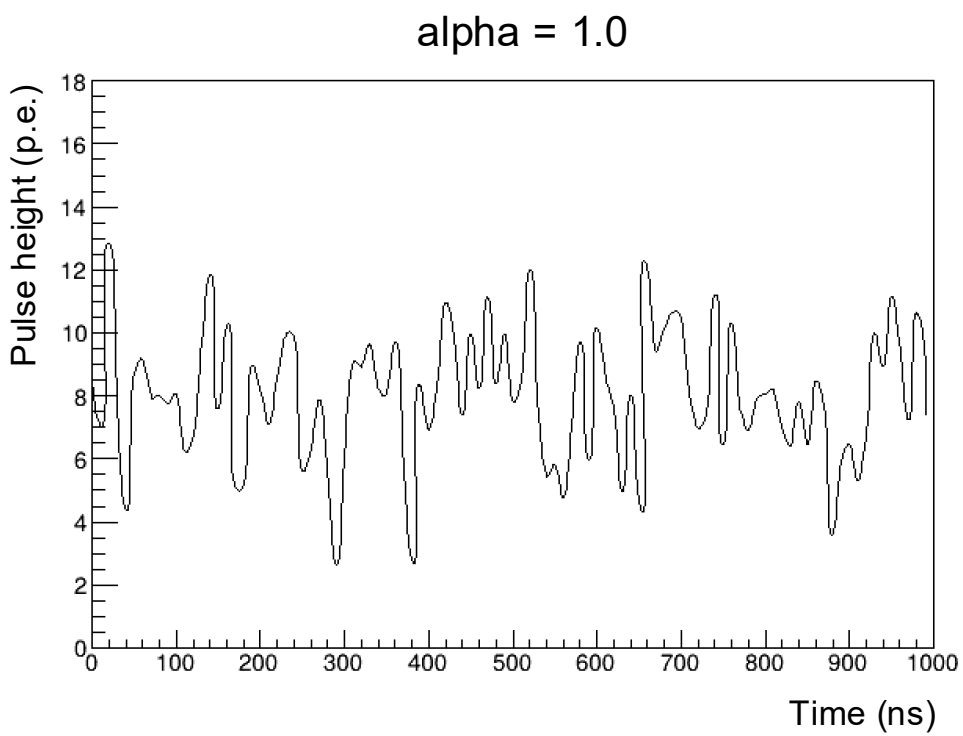
SiPM GlueX Template with 112.97 MHz DCR

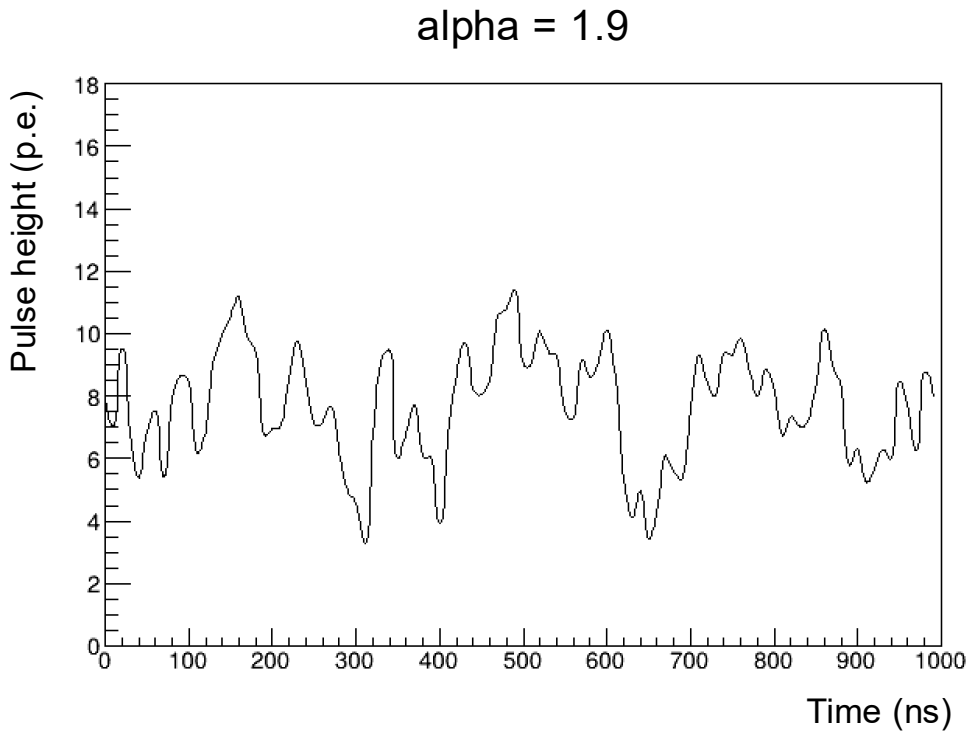
S14160 SiPMs at 5 °C irradiated 1e+09 1-MeV neutron equivalent dose (Bolognia)



- Ideally, the FalphaNoise should generate noises at least every ~10 ns and pole should be ~30.
- In the SimPulse data model, the amplitudes were filled every 0.5 ns. → We may need a pole larger than 30, but using a large value of pole will be quite inefficient.
- The pole was fixed as 2 for a fast operation and alpha & variance were optimized so that the corresponding distributions reproduce Henry's plots.

Reproducing Henry's plots

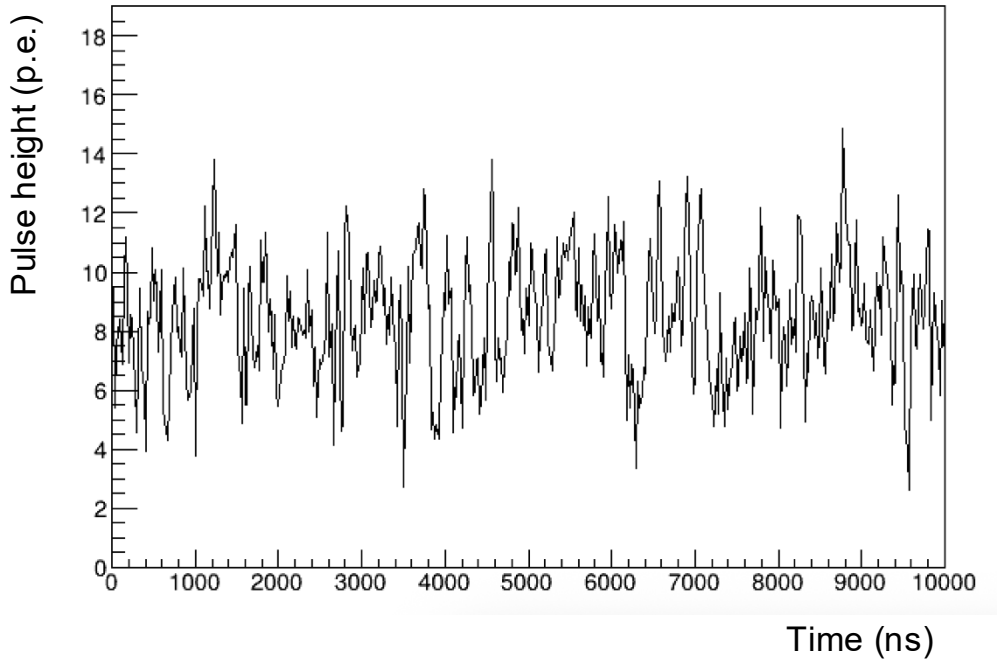


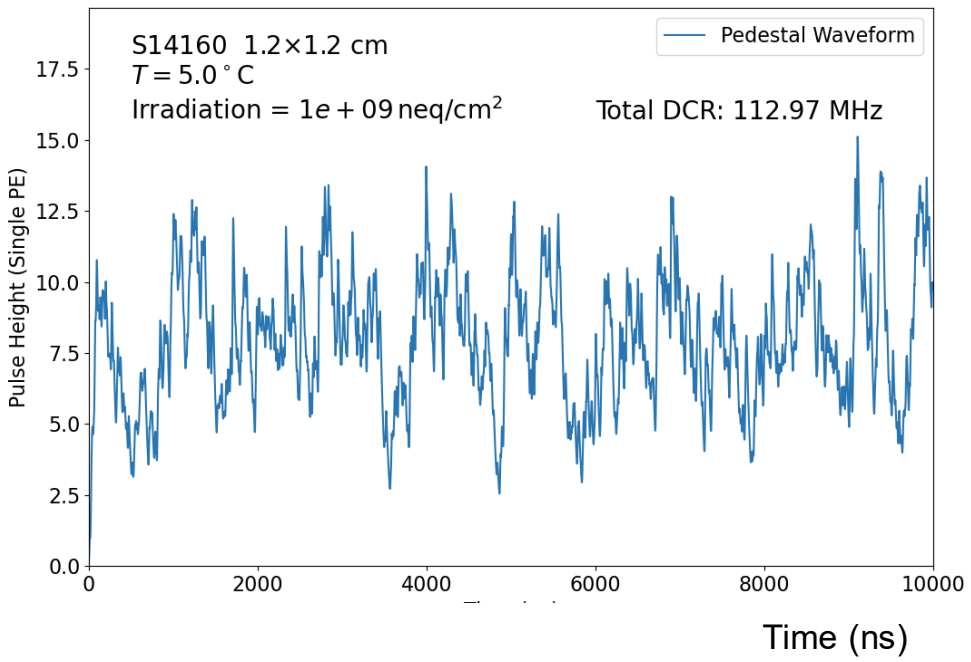


- Since the FalphaNoise provides a $\pm$ fluctuation, an offset of 8.0 was applied to the sample.
- For a given alpha, the variance was set so that the pedestal distribution provides comparable σ.
- If alpha is too low, the Pulse height vs. Time distribution looks different from Henry's plot.
- If alpha becomes about 1.9, the distribution gets similar.

Reproducing Henry's plots

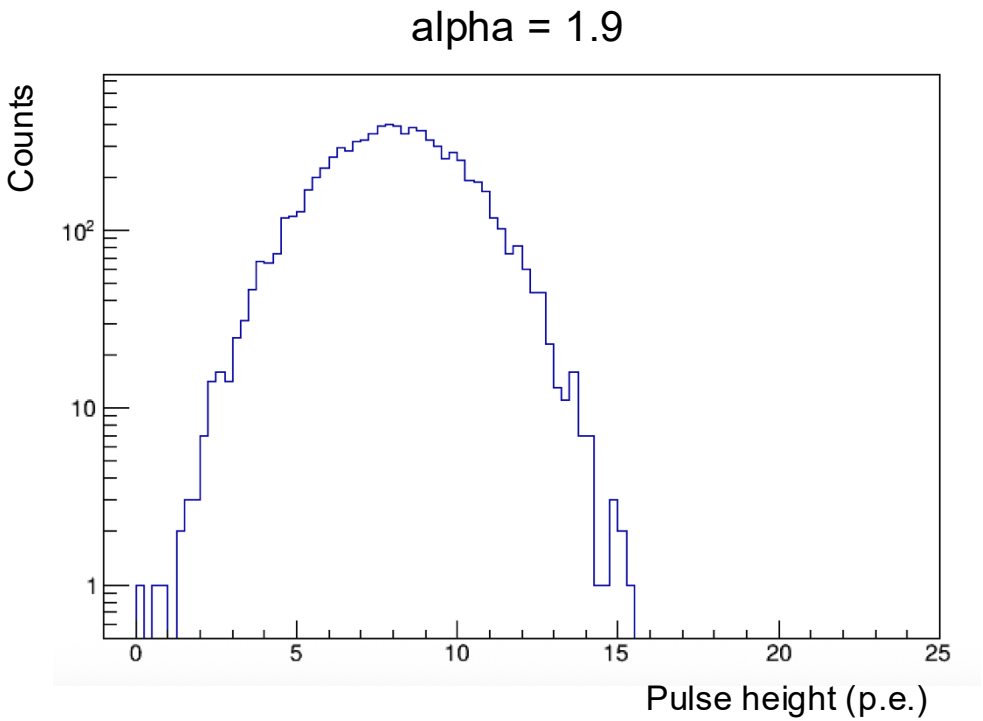
alpha = 1.9

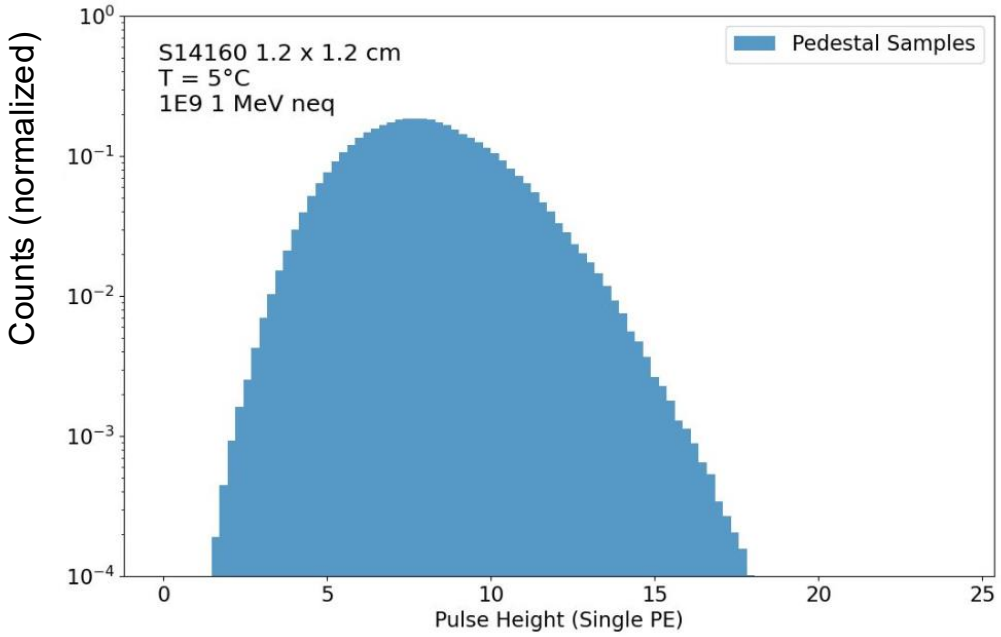




- For a given alpha, the variance was optimized so that the pedestal distribution provides the same. σ.
- The distribution in a large time scale is also comparable.

Reproducing Henry's plots





- The shape is a little different because FalphaNoise doesn't generate the pulse shape, but the difference affecting the HGCROC measurement should be negligible.
- The parameters for PulseNoise will be optimized as presented so far. If agreed, submit PR for PulseCombiner and PulseNoise. → Start implementing CALOROC algorithm.

BIC data volume (18x275_minQ2_1000)

Without Att.Hits and Pulses

Branch	Size (MB)
Total	35
EcalBarrel*	14

With Att.Hits

Branch	Size (MB)
Total	58
EcalBarrel*	36

With Att.Hits (Unnecessary values were set to 0.)

Branch	Size (MB)
Total	45
EcalBarrel*	23

With Pulses (timestep = 0.5 ns, thres = 1.0e-5)

Branch	Size (MB)
Total	64
EcalBarrel*	42

With Pulses (timestep = 1.0 ns, thres = 5.0e-5)

Branch	Size (MB)
Total	48
EcalBarrel*	27

Excluding RawHits

Branch	Size (MB)
Total	39
EcalBarrel*	17

20 edm4eic.yaml

```diff
@@ -191,6 +191,15 @@ components:
      - uint32_t boundValuesSize          // size of bound values
      - std::array<double,16> transform   // row-wise 4x4 affine transform [R T; 0 1] with 3x3 rotation matrix R and translation column 3-vector T

+  ## An individual sample output by an HGCROC chip
+  edm4eic::HGCROCSample:
+    Members:
+      - uint16_t ADC                // [ADC Counts], amplitude of signal during sample, valid IFF TOTInProgress is false
+      - uint16_t timeOfArrival      // Time of arrival (TOA) [TDC counts], nonzero IFF ADC crossed threshold upwards during sample
+      - uint16_t timeOverThreshold  // Time over threshold (TOT) [TDC counts], nonzero IFF ADC crossed threshold downwards during sample AND if TOA fired in a previous sample
+      - bool     TOTInProgress      // Flag which indicates if a TOT fired in a previous sample and calculation is ongoing, ADC value may be corrupted if this is true
+      - bool     TOTComplete        // Flag which indicates if a TOT calculation is complete and TOT value is valid
+
  datatypes:

    edm4eic::Tensor:
@@ -265,6 +274,17 @@ datatypes:
    ## ============================================================================
    ## Calorimetry
    ## ============================================================================
+
+  edm4eic::RawHGCROCHit:
+    Description: "Raw hit from an HGCROC chip"
+    Author: "D. Anderson, S. Joosten, T. Protzman, N. Novitzky, D. Kalinkin"
+    Members:
+      - uint64_t cellID               // Detector specific (geometrical) cell id
+      - int32_t  samplePhase          // Phase of samples in [# samples], for synchronizing across chips
+      - int32_t  timeStamp            // [TDC counts]
+    VectorMembers:
+      - edm4eic::HGCROCSample sample  // ADC, Time of Arrival (TOA), and Time over Threshold (TOT) values for each sample read out
+
    edm4eic::CalorimeterHit:
      Description: "Calorimeter hit"
      Author: "W. Armstrong, S. Joosten"
```