Status of Digitization Code with Waveform Simulation

Sidney Leggett & Blair Jamieson

What our code does (1/2)

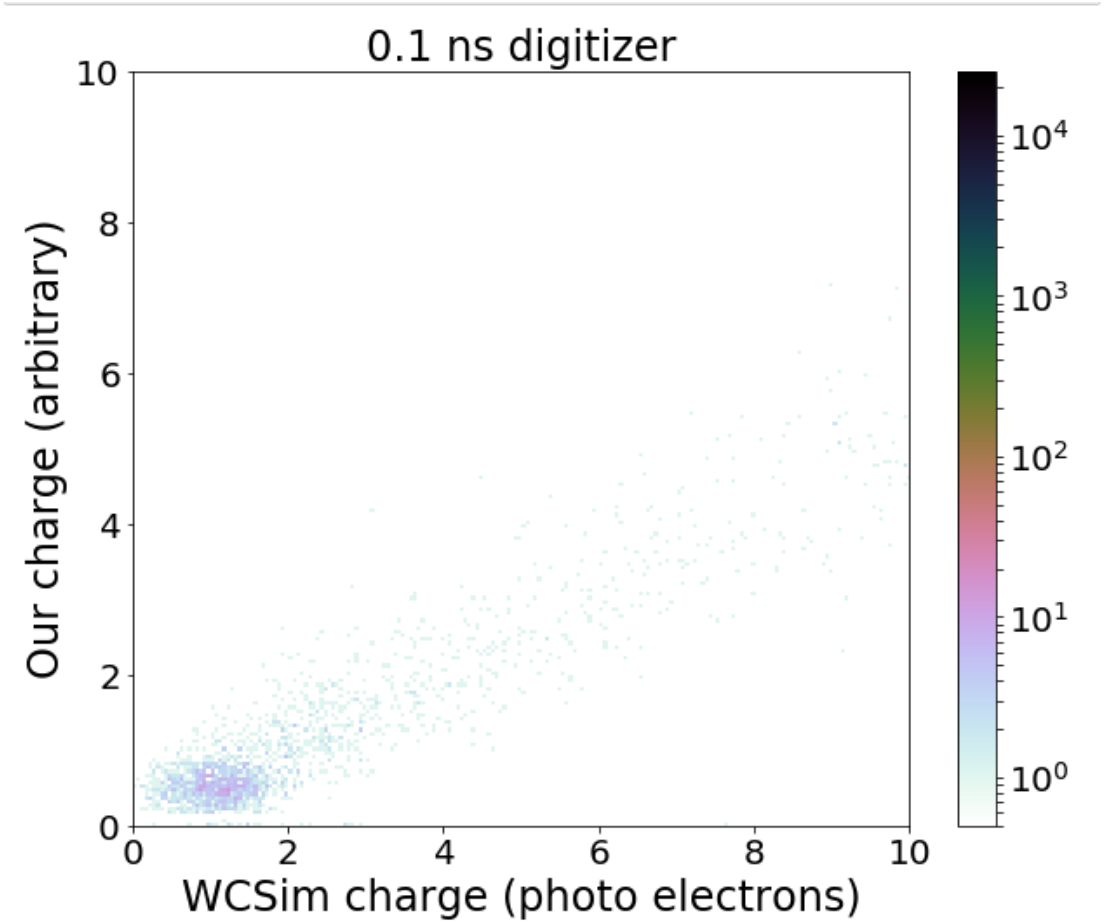
A module written in python

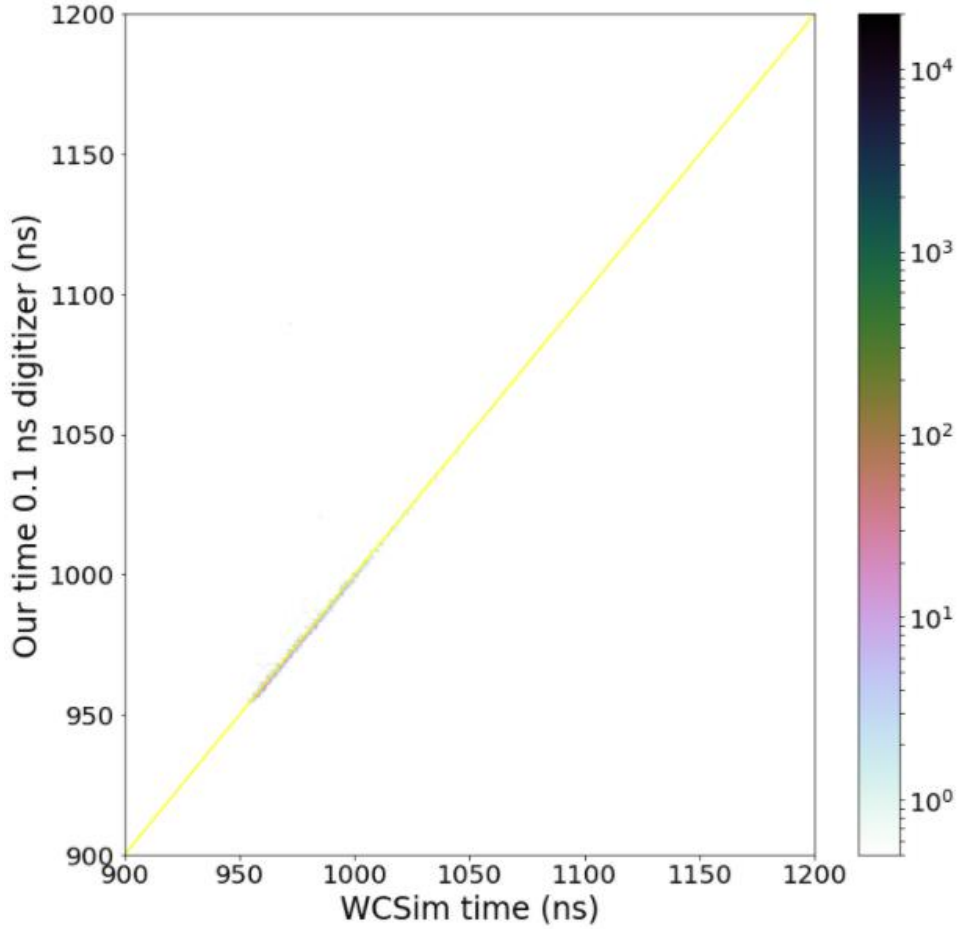
- Input truth hits data set (in an h5 file format)
  1. For each true time generate number of pe's following the PDF described by the f_of_q function
  2. Builds a wavetrain of pulses following model of arXiv:1801.08690 using the function buildwavetrain
  - each pulse shape given by the function f_waveform, and
  - the noise by the function f_noise
  3. Process the wavetrains to find the times and charges using the function WaveformsToTQ
  - an initial search for peaks is done to find the baseline, time and peak-height of each pulse using the function get_wf_peak_guesses

What our code does (2/2)

- -the output of that function is used to fit each peak to a gaussian in the function get_times_charges_from_wf
- which returns the 10% of the gaussian time and area of gaussian as charge for each pulse
- After calling digitize_event, you can access the results of intermediate steps from specific class members

Our digitization compared to WCSim





Old slides next for context of project

Overview of project

- Visualizing the new tank geo_mPMTshort
- Displaying events with truth information
- Methodology behind waveform sim for truth information
  - Started following the Charge reconstruction in large-area photomultipliers paper (here: grassi2018.pdf (infn.it)
  - Waveform simulation
  - Take waveforms and build digitization of true hits
- The Digitizer.py module

Digitizer_Testing notebook

GetTrueEvent and EventDisplay

- GetTrueEvent is a function that takes our loaded truth h5 dataset, and a desired entry of the event you want
  - It then outputs the true pmts that is an np.array of the PMT numbers hit and the true times of the times where the PMT was hit, then final the true parents of the parent number of the photon hitting PMT (which is = -1 for the dark noise)
- EventDisplay is a function that takes in an np.array of PMTs that were hit (called tubes), then an np array of either times or charges (quantities), a title for the display, and a cut range for the min and max for values to be shown on plot
  - Returns plot

Waveform Simulation

- Following the Grassi paper
- Figure 1 from the paper (to the right) shows the building blocks of waveform simulation

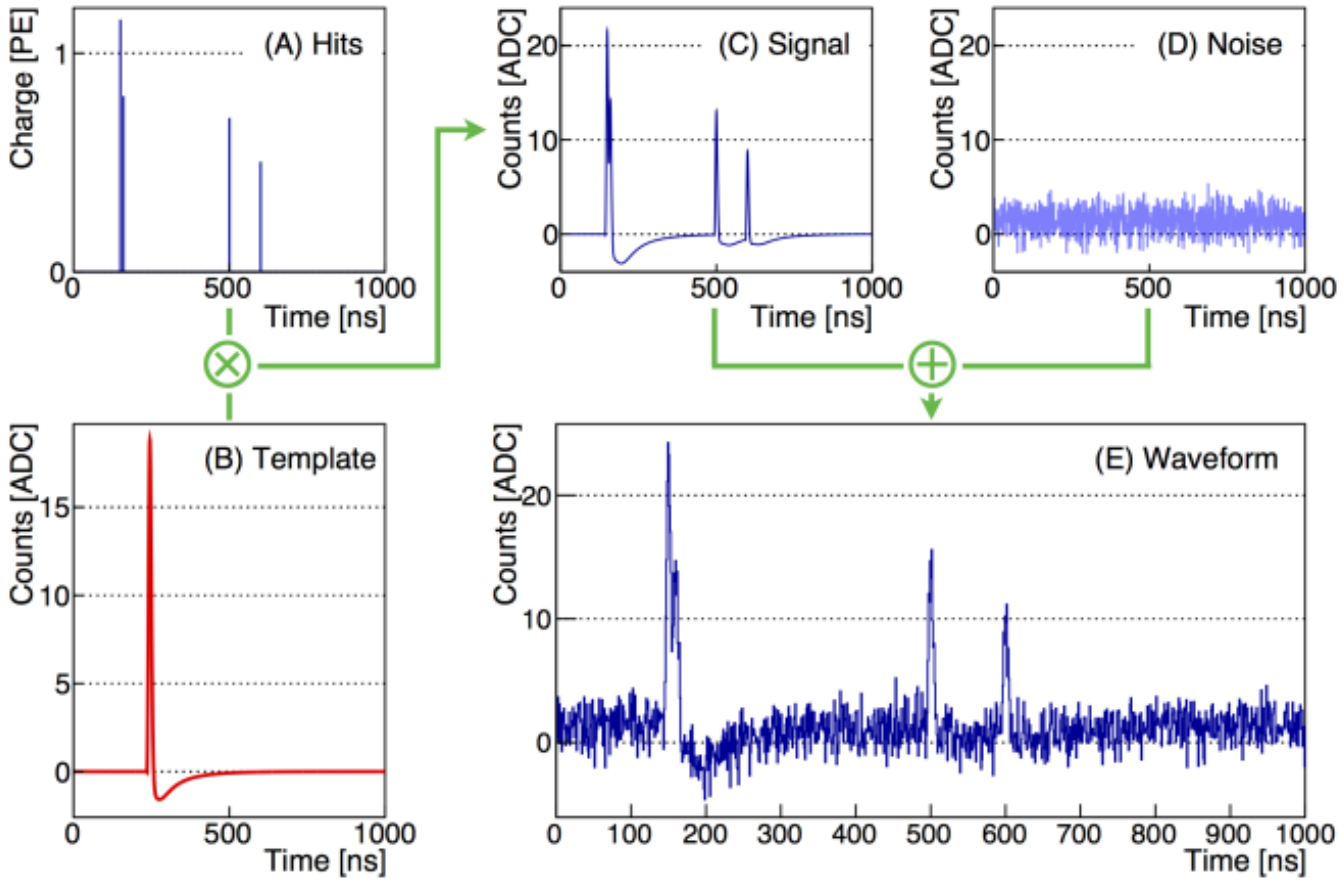
A. Shows hit generation, where for each detected photoelectrons a random charge value is generated

B. Shows the analytical shape of a single pe template

C. Result of the convolution between the SPE template and the hits

D. PMT simulated noise

E. Final PMT waveform resulting from the sum of the signal and noise component



Information on Figures

- Figure **A** is showing a sequence of hits resulting from several pes impinging on the same pmt that are shown as instantaneous pulses
- the analytical shape of a spe waveform is show in figure **B**, this is the shape we can call spe template
- The signal only waveform resulting from the hits shown in A is built by convolving each hit the the spe template and shown in **C**
- To make simulation more realistic, a noise waveform is added to the waveform built using only signal hits shown in D
- The ultimate waveform made up of signal hits, and all noise components is shown in **E**

$$U_{\text{peak}}(t) = U_0 \cdot \exp\left(-\frac{1}{2}\left(\frac{\ln(t/\tau_0)}{\sigma_0}\right)^2\right) \tag{2.2}$$

Table 2. Parameters involved in the signal (top rows) and noise (bottom rows) simulation of the PMT waveform. Parameters for which the "Range" column is filled are generated randomly according to a flat probability density function at the beginning of the simulation.

Parameter	Value	Range	Description	Function
U_0	20 ADC			
σ_0	0.15		SPE template	eq. 2.2
τ_0	30 ns			
U_1	-1.2 ADC			
σ_1	55 ns		Overshoot	eq. 2.3
t_1	-4 ns			
U_2	-2.8 ADC		Overshoot	eq. 2.4
τ_2	80 ns			
μ_N	1.5 ADC		Baseline	Gaussian
σ_N	1 ADC		White Noise	
n_{FF}	5		N(Components)	Fixed Frequency
f_{FF}		[1, 480] MHz	Frequency	
A_{FF}		[0.1, 0.3] ADC	Amplitude	

$$U_{OS1}(t) = U_1 \cdot \exp\left(-\frac{1}{2}\left(\frac{t-t_1}{\sigma_1}\right)^2\right) \tag{2.3}$$

$$U_{OS2}(t) = U_2 \cdot \frac{1}{\exp\left(\frac{50\,\text{ns}-t}{10\,\text{ns}}\right)+1} \cdot \exp\left(-\frac{t}{\tau_2}\right) \tag{2.4}$$

$$U(t) = U_{\text{peak}} + U_{OS1} + U_{OS2} \tag{2.5}$$

Why do it?

- Build a PMT waveform simulation with the goal of developing and validating the charge reconstruction algorithm with known input signal
- The charge value q is generated randomly according to the distribution f (q)
- The weight (omega) determines the relative contribution of a gaussian dist'n with respect to an exponential tail, modelling the faction of under amplified pes

Parameter	Description	Value
q_o	SPE calibrated gain	1 PE
q_p	Pedestal cutoff	0.3 PE
σ	SPE Gauss width	0.3 PE
ω	Under-amplified PE fraction	0.2
τ	Exponential decay constant	0.5 PE

$$f(q) = \begin{cases} (1-\omega)\frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(q-q_0)^2}{2\sigma^2}\right] + \frac{\omega}{\tau} \exp -\frac{q}{\tau} & q \geq q_p \\ 0 & q < q_p \end{cases}$$

Creating the Digitizer.py module

- Provides a class for doing the digitization of events
- The class holds all the settings for the digitization
- Function digitize_event which takes np.arrays of the true PMTs and true times (and there are multiple entries per PMT)

Digitizer.py

- The digitization follows these steps:

1. For each true time we generate the number of pe's following the probability density function F(q)
2. Build a wavetrain of pulses following the papers model using the function buildwavetrain
    1. Each pulse shape given by the function f_waveform
    2. And the noise by the function f_noise
3. Process the wavetrains to find times and charges using the function WaveformsToTQ
    1. Initial search for peaks is done to find the baseline, time and peak height of each pulse using the function get_wf_peak_guesses, the output of the function is used to fit each peak to a gaussian in the function get_times_charges_from_wf, which returns the 10% of the gaussian time and area of gaussian as charge for each pulse

Digitizer.py

- After calling digitize_event you can access the result of the intermediate steps in the class members

```
self.pmt_time_dict      pmt : list of truetimes of photon arrivals
self.pmt_pe_dict        pmt : list of number of pe for each photon arrival
self.pmt_wf_dict        pmt : ( times, charges) wavetrain for each pmt
self.digi_qt_dict       pmt : (digitized time, charge)
self.firstguesses       pmt : (baseline, peaktime, peakheight)
self.peakfits           pmt : (params, covariance)
```

Testing

- Testing the new digitizer class with the truth data h5 file: IWCD_mPMT_Short_emg_E0to1000MeV_truehits.h5
- Creating 3 classes to read initialize the h5 data
  - First to initialize the digihits dataset
  - Second to initialize the truth data, and use the digitization class with parameters set to match the WCSim digitization, following 0.1 ns binning
  - Third to initialize the truth data, and use the digitization class with parameters set to match the IWCD 8 ns binning

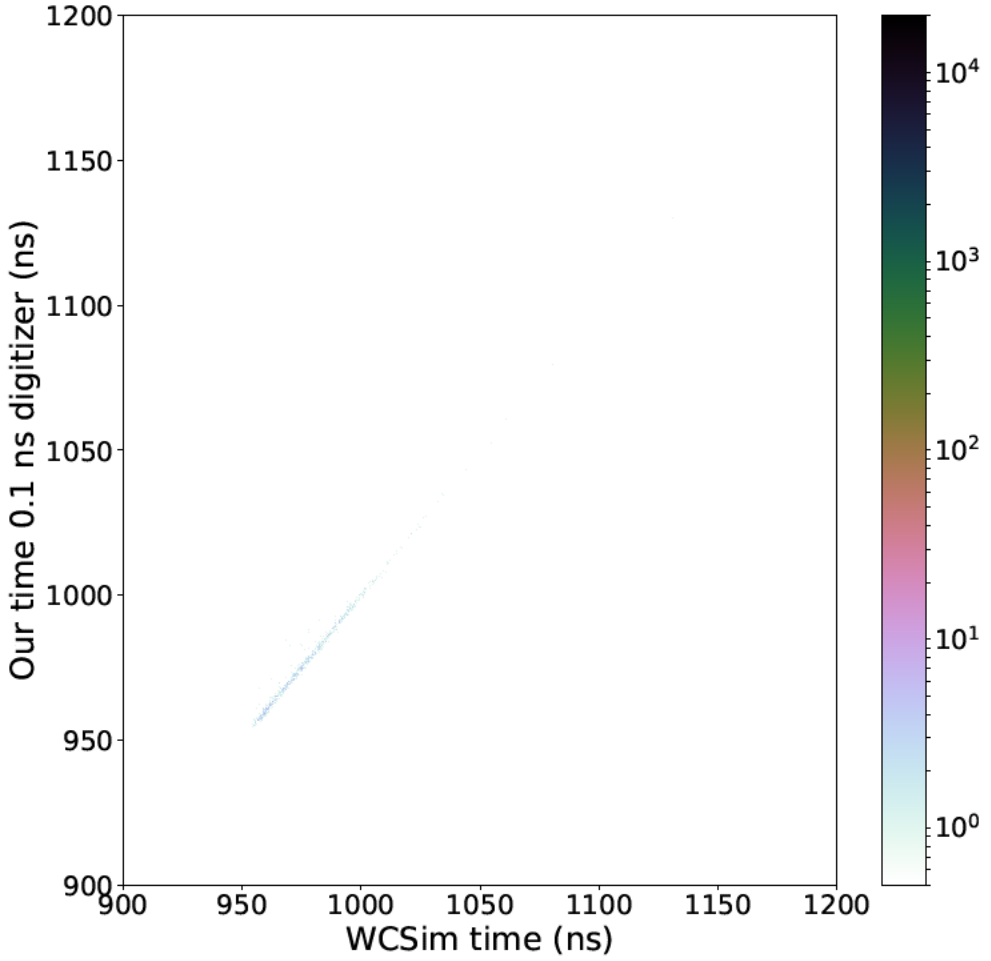
Test the intermediate steps in digitization for the 0.1 and 8 ns binning seperately

- Doing this by

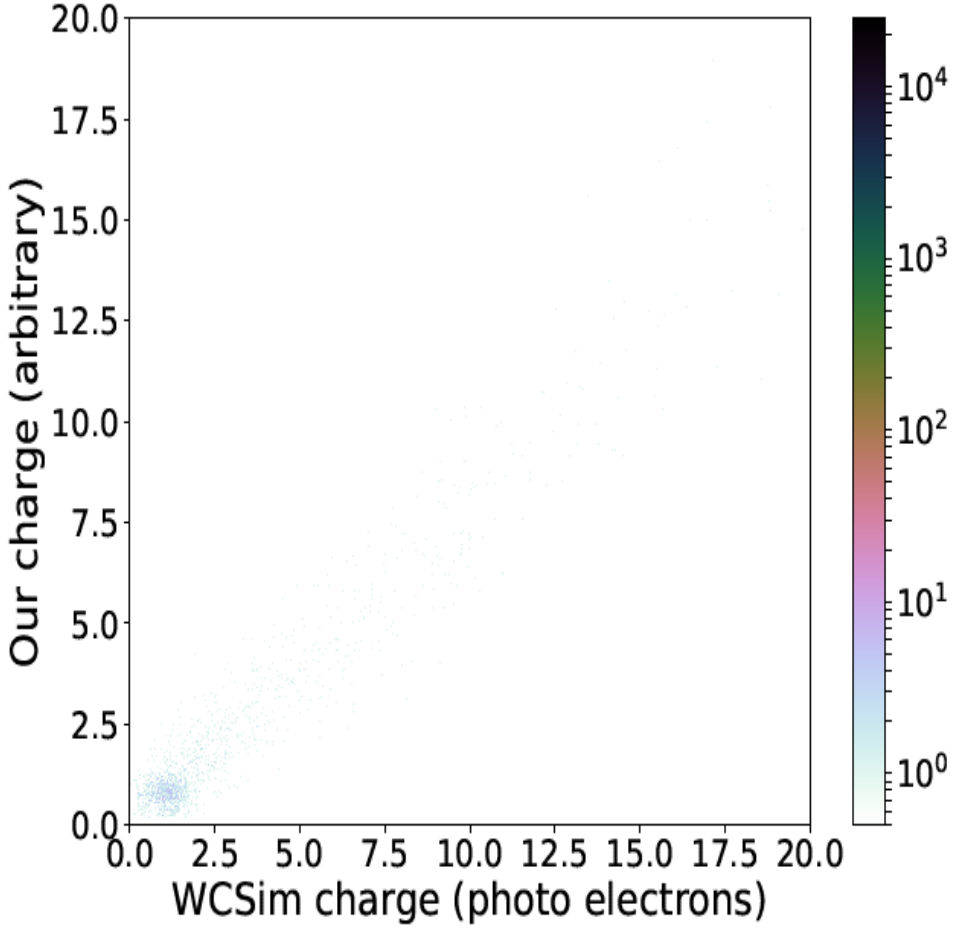
1. Building waveforms
2. Finding peaks (first guess)
3. Fitting peaks
4. Calculating final charge

Building plots of our digitized event vs WCSim

Plot 1: WCSim time vs Our min-time



Plot 2: WCSim charge vs Our charge



What we are working on now

- Parts of code run very slow, trying to optimize to speed up
- Finalize plotting of intermediate steps