Towards a 2D Drift Simulation in LArSoft

Step 1: Proof of Principle

Tracy Usher / Leon Rochester

LArSoft Coordination Meeting

February 12, 2019

Basic Concept

- As just described by Leon…
- Nutshell summary:
    - Use look up tables to describe the response to a point charge on the nearest wire (“central wire”) and its 9 nearest neighbors on each side
        - Look up tables generated by Leon Rochester
            - A set of tables (central wire plus 9 nearest wires each side) given per plane (3 planes) and per “impact parameter” (distance of closest approach to the central wire in the wire plane - in 1/10 wire pitch increments)
            - Starting position for the tables is a plane 10 cm above the wire plane
                - Each look up table consists of a vector with (for ICARUS) 175 “ticks” from the starting position to describe the effect on a given wire
                - Note that if the drift starts inside the 10 cm region then the starting index in the table is adjusted to take into account the start position
    - Build up the response to a track by simply adding the individual responses of the deposited point charges

LArSoft Implementation

- For the “new” LArG4 refactored code the electron drift to the wire planes has been factored into a separate module called “SimDriftElectrons” which we can use as the starting point
    - Input are the new “SimEnergyDeposit” objects which record position, energy deposited and number of electrons (post recombination) at each step in the Geant4 tracking procedure
- Refactor this module to use “art tools” to do the actual electron drift and wire plane response
    - Even further, factor out the actual drift part from the response part
        - The drifting of electrons (breaking into “clusters”, applying lifetime/diffusion effects) is its own art tool which should allow straightforward switching
- The interface for the response tools (and their implementation) are such that they define their output data products and are responsible for allocating, filling and outputting them
    - For example, this allows the 2D drift simulation to output a new data product
- Believe it is now possible to define a pixel simulated data product and simply switch in a response tool at this point

Current Structure

- Producer module: SimDriftElectrons
- Response Tools:
  - ElectronDriftStandard - reproduces the default 1D drift responses
    - Primary output data product: SimChannels
    - Also optionally output SimDriftedElectronCluster
  - ElectronDrift2D - new tool to do the 2D drift simulation
    - Primary output data product: “Waveform”
    - Also output SimChannels (for truth information)
    - Optionally output SimDriftedElectronCluster
- Drift Tools:
  - DiffusionStandard - reproduces current electron drift model

Why a New Data Product?

- The drift simulation breaks a given SimEnergyDeposit contribution in “clusters” of electrons and drifts each independently. This allows modeling of diffusions
    - Each “cluster” is around 20 electrons, MinI deposit is ~70-100 clusters
- The standard drift simulation populates 1-2 “ticks” with truth information
    - Each tick a vector of a struct containing: MC “TrackID”, fraction of energy contributed by this entry, energy deposited, number of electrons
    - Filling requires “finding” the tick to add the info too and then do some number of calculations
- Filling a SimChannel entry is “acceptable” in the current simulation but one could see that it could be a time consumer if one was filling a larger number of ticks
    - There is also the question of what “truth” means in this case….

Why a New Data Product?

- The goal of the 2D drift simulation is to output an object which will combine the number of electrons surviving to a plane with the field response
    - This will look like a waveform in the number of electrons per tick (where we note that the number of electrons can be negative)
- The look up tables will have 175 ticks worth of information for each wire. We will want to add the contribution of each tick to not only the closest wire after the drift to the wire plane, but to the nearest 9 neighbors on each side of the wire.
- Using the SimChannels turns out to be prohibitively expensive
    - My first runs were taking 8-9 minutes for a single particle simulation

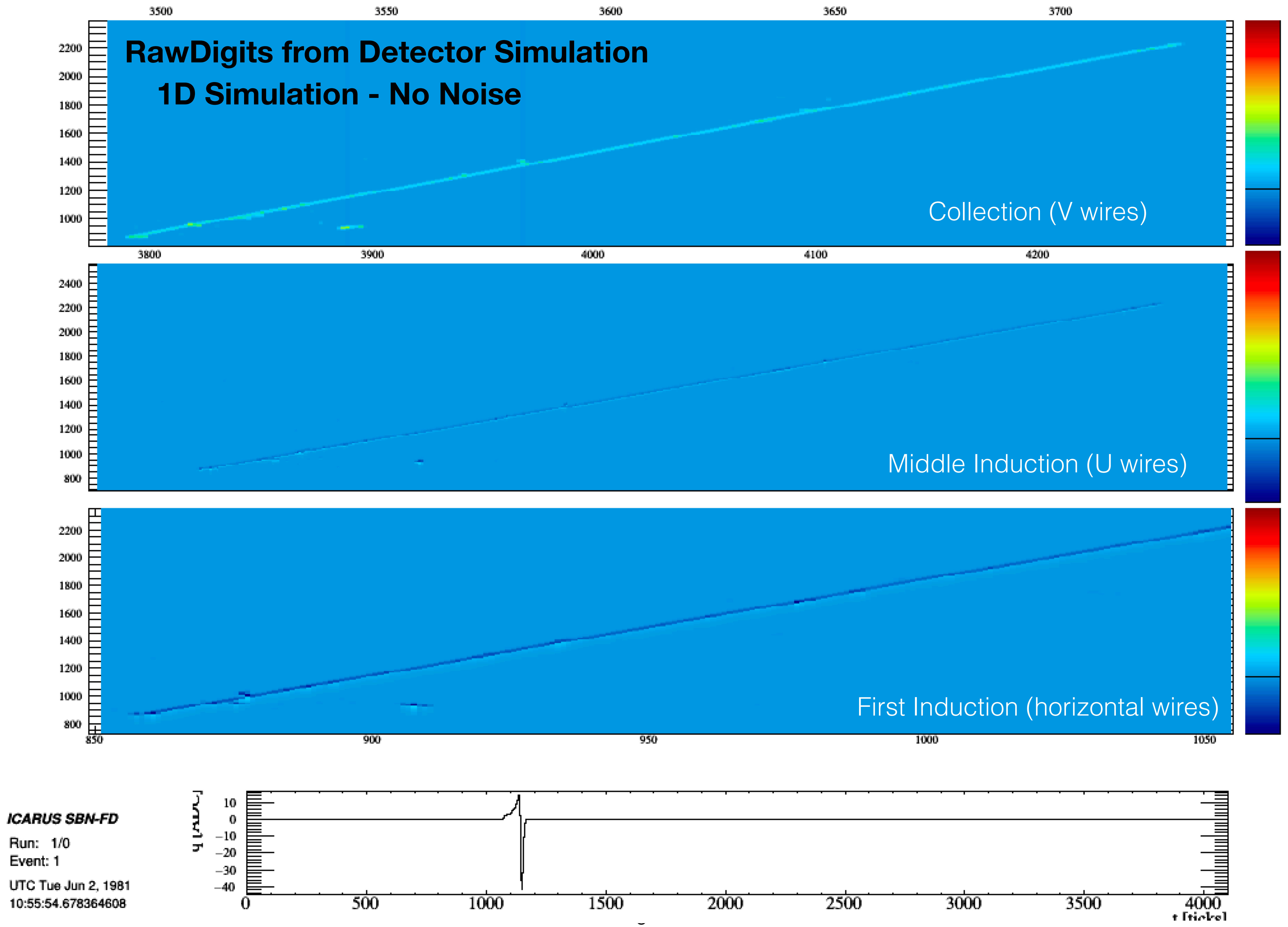
Currently Proposed Solution

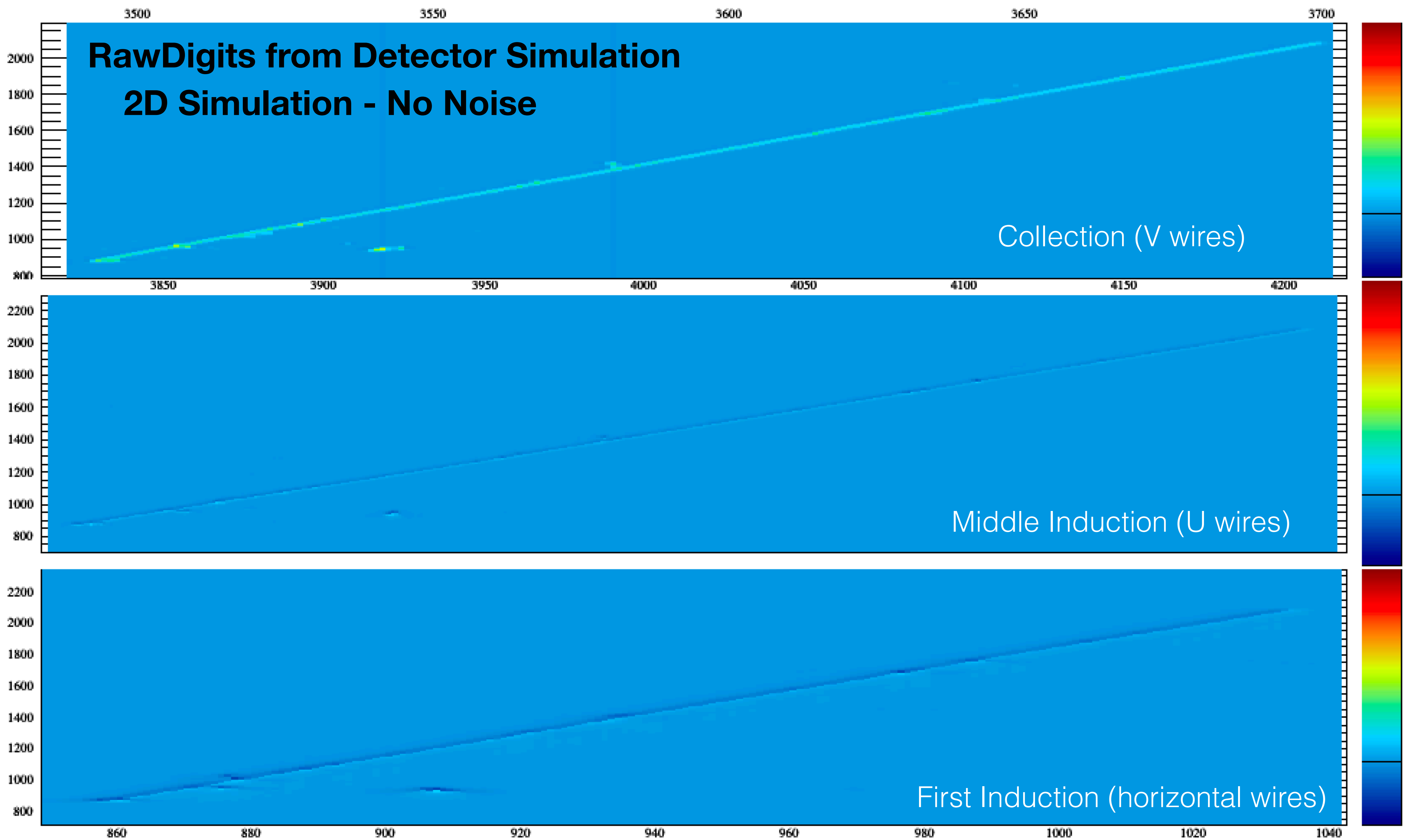
- Define a new output data object that is strictly a waveform containing the number of electrons per tick on the wire
    - Starting point is to simply clone the existing Wire.h object used in the reconstruction
        - In particular it is defined using sparse_vectors so will be minimally sized
    - These are filled for each tick of each of the lookup tables used for a given drifted electron cluster (19 wires x 175 ticks)
- Keep the SimChannel output (or switch to the SimDriftElectronCluster objects) to handle the truth matching
    - These are filled only once per drifted electron cluster
- With this scheme able to get time to run a single particle simulation down to ~30 seconds
    - This is still too slow, ~O(10) times slower than the 1D drift simulation

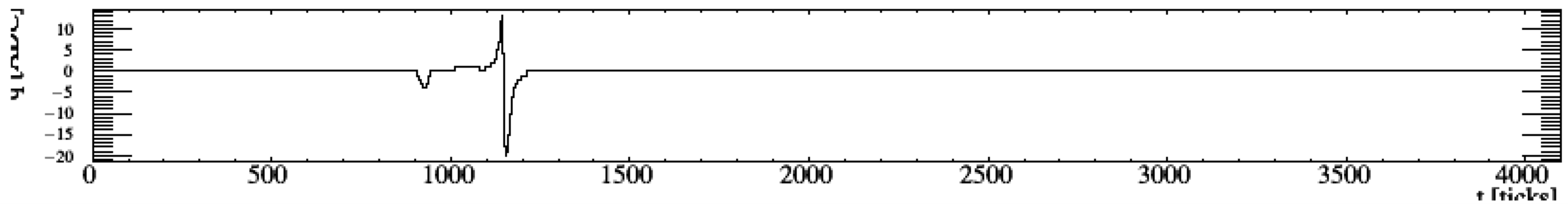
Detector Simulation

- The detector simulation step takes as input the simulated waveforms and then outputs “RawDigits”
  - Simpler than previous detector simulation - no longer convoluting with an approximate field response
  - The input simulated waveform represents number of electrons per tick, this is convoluted with the electronics response and then scaled to go from electrons to ADC units
- Note that in the LArSoft simulations to date the digitization of the electronics has always been effectively implemented in the simulation
  - In principle, one can now keep a finer grain simulated waveform and do the digitization during the detector simulation…

RawDigits from Detector Simulation
1D Simulation - No Noise







ICARUS SBN-FD
Run: 1/0
Event: 1
UTC Tue Jun 2, 1981
10:55:54.678364608

Where To From Here?

- This looks promising!
- Much work to do:
  - Need to consider the output data products
    - And we should keep an eye open to what might be useful for a pixel detector as well since we are close to wanting to integrate this into LArSoft too
  - Need to understand an efficient way to implement this scheme
    - Even at ~10x slower than current it is probably too slow to be usable?
  - Definitions for interface classes?
  - etc.
- Currently this work is aimed at ICARUS which is preparing for another major round of sim/recon to support the upcoming SBN workshop
  - Is it realistic to have a first implementation available in LArSoft by the beginning of March to have this available for ICARUS?