DUNE IN2P3 workshop

Ionization

- Charged particles traversing the detector ionize argon
- "Work" or average energy required to dissociate electron to form argon ion – electron pair in liquid:

$$W_{ion} = 23.6^{+0.5}_{-0.3}\ \text{eV}$$ (PhysRevA.9.1438)

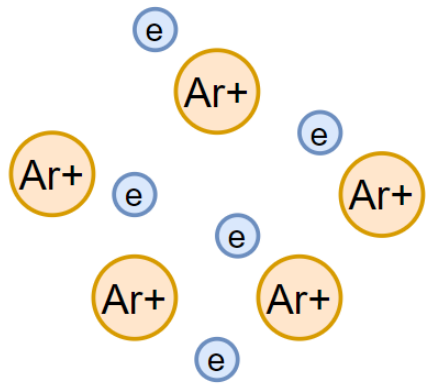
- The ionization electrons are the charge that is detected in LAr TPC by drifting it to sensing electrodes (e.g., wires, strips)
- Trivially one can find a number of electrons produced per 1 MeV of deposited energy: ~42 ke / MeV
- The fluctuation on this number can be estimated from Fano factor F = 0.107 [*NIM.* 134 (1976)]

$$\frac{\sigma_n}{n} = \sqrt{\frac{F}{n}} \sim 0.2\%$$

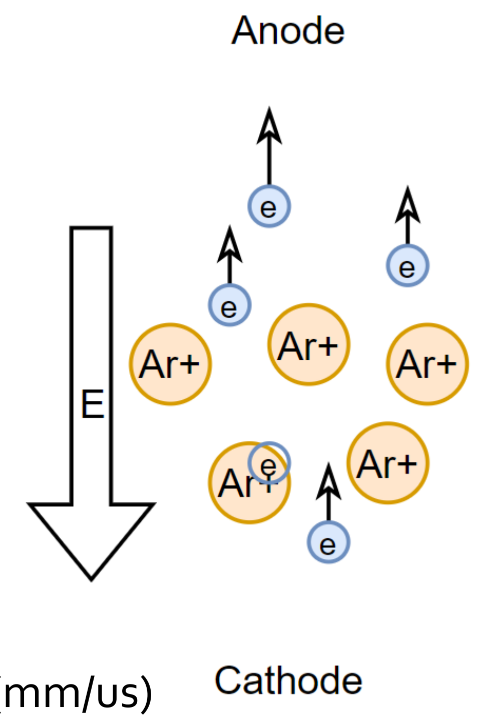
Negligible with respect to fluctuation in dE/dx

Recombination

- Liquid argon is a dense medium
- Some electrons recombine with ions leading to an intrinsic “loss” of ionization signal
- The fraction of recombined electrons depends on
  - The density of charge depositions (typically taken to be proportional as dE/dx)
    - The higher the local density of ion-electrons the more likely they will recombine
  - Strength of the electric field in the detector
    - With no field all electrons would recombine
    - With infinitely strong field the recombination would be minimal
- In practice the recombination (model) is fitted as function of dE/dx and electric field



Jaffe (Ann. Phys. 42 (1913) 303) columnar model



Electron drift velocity O(mm/us)
Ion drift velocity O(cm/s)

Recombination pumps light production

- The passage of ionizing particles also leads to formation of excimer states of Ar
- These decay via emission of VUV photons (128 nm)

- Two ways to produce excimer states
  1. Self-trapping: $Ar^* + Ar \rightarrow Ar_2^* \rightarrow Ar + Ar + \gamma$
  2. Recombination: $Ar^+ + Ar + e^-_{therm} \rightarrow Ar_2^* \rightarrow Ar + Ar + \gamma$

- Conversion of recombined electrons to light
- Max number of photons are produced in zero electric field (maximum recombination)

PhysRev B17 2762 (1978)

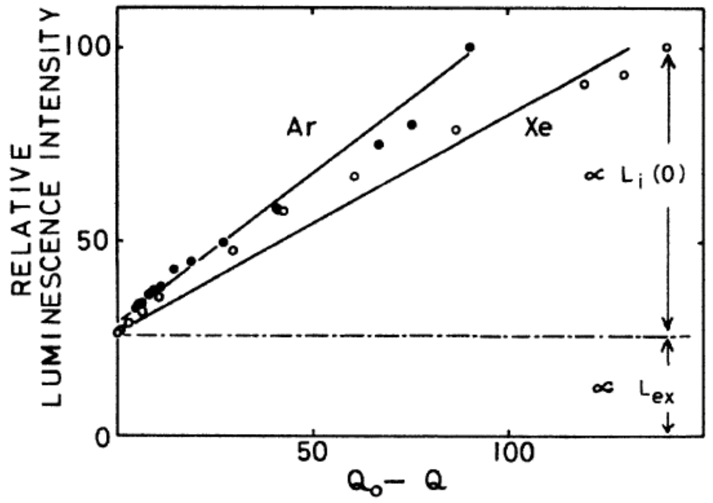


FIG. 4. Relative luminescence intensity against uncollected charge Q_0-Q.

Q_0-Q = 0: very large E-field, so no recombination contribution

Recombination parametrization: Birks form

- Birks form (ICARUS, NIMA 523 (2004) 275):

$$R = \frac{A}{1 + k/\varepsilon \times dE/dx}$$

ε – electric field x LAr density, dE/dx expected energy loss and A, k are constants

- The fitted values (muons) of A and k parameters (NIMA 523) :

$k = 0.0486$ (kV/cm)(g/MeV cm^2)

$A = 0.800$

Recombination parametrization: modified Box model

- ArgoNeuT [*JINST* **8** P08005 (2013)]:

$$R = \frac{\ln(A + \xi)}{\xi}$$

$$\xi = B/\varepsilon \times dE/dx$$

The fit parameters A & B; ε – electric field x density

- The fitted parameters (stopping protons) in the paper:

$$B = 0.212 \text{ (kV/cm)(g/MeV cm}^2\text{)}$$

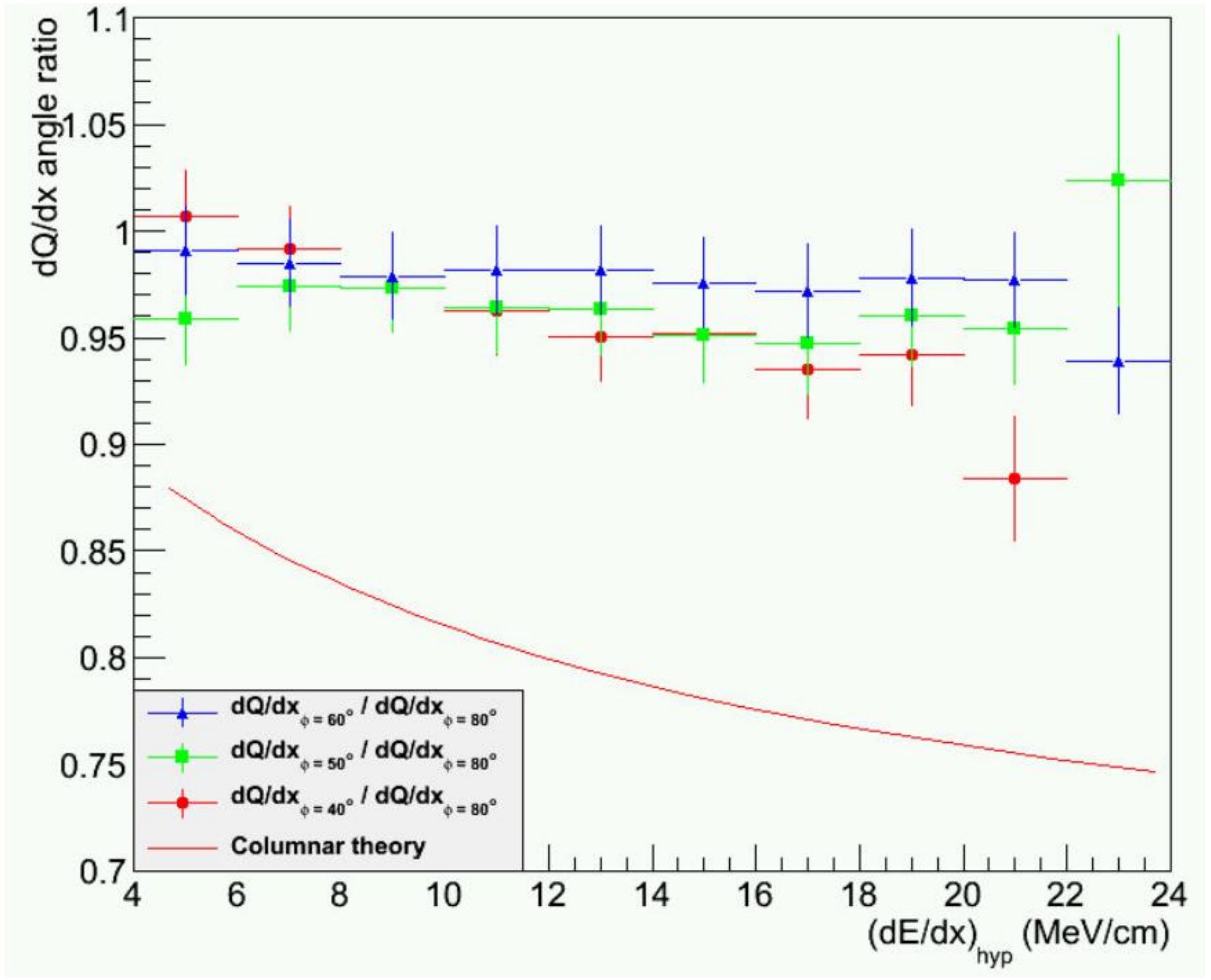
$$A = 0.930$$

A = 1 in canonical Box model in [Phys. Rev. A 36 (1987) 614] (hence "modified")

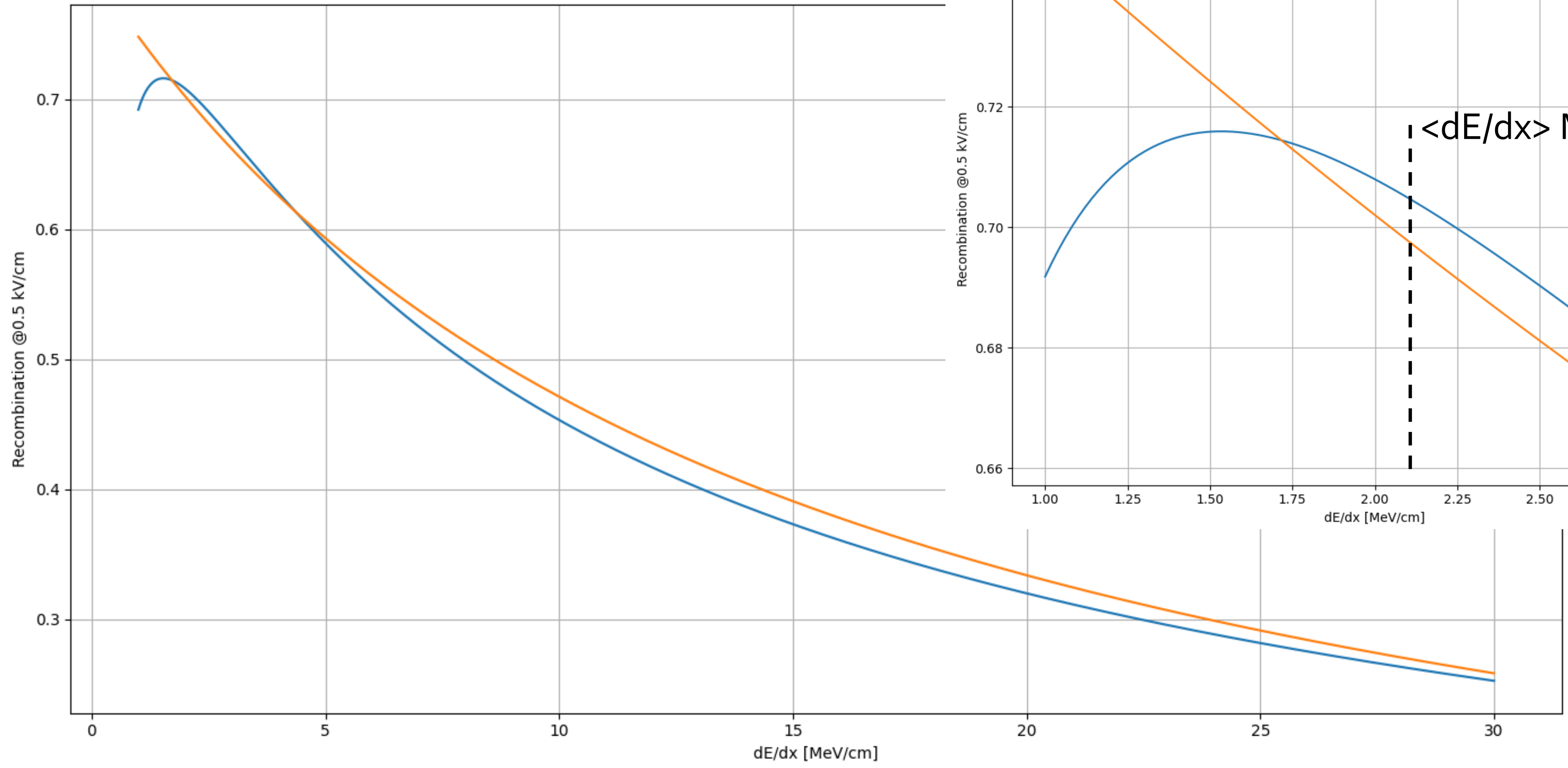
Angular dependence

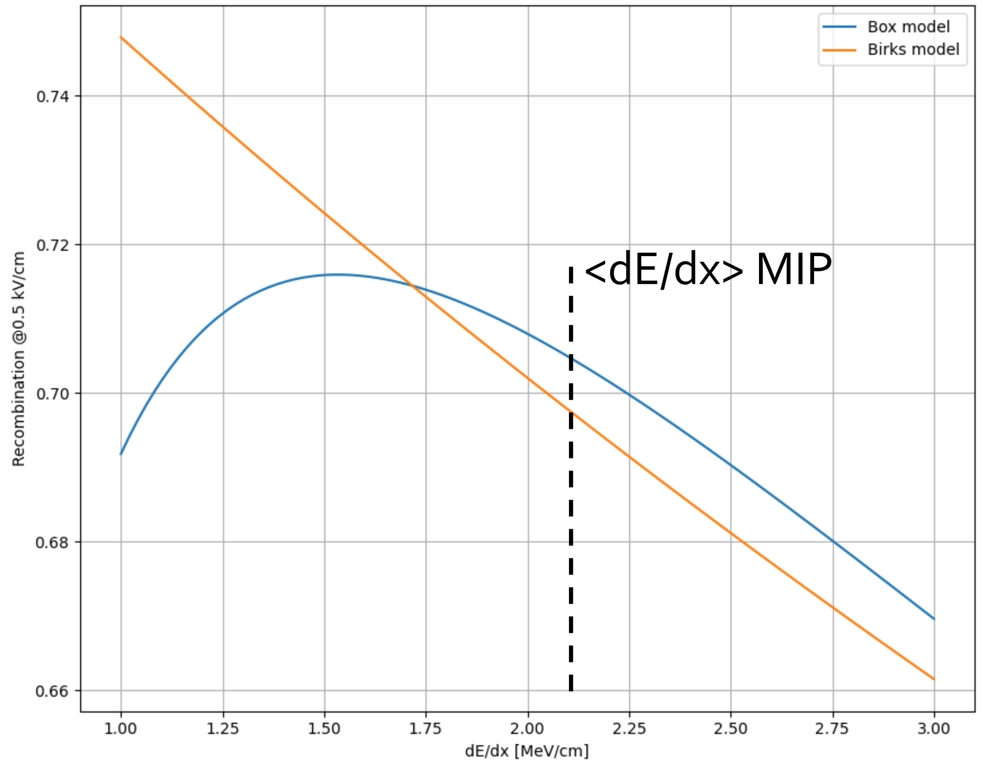
- Columnar theory predicts a fairly strong dependence on the angle of “columns” around particle direction relative to the field direction

- No such dependence has been observed in [*JINST* **8** P08005 (2013)]



Recombination models





Number of electrons per MIP 2.1MeV/cm @0.5kV/cm: ~62 ke/cm or ~10 fC/cm

From dQ/dx to dE/dx

$$\frac{dQ}{dx} = \frac{1}{W_{ion}} R\left(\frac{dE}{dx}, \varepsilon\right) \frac{dE}{dx}$$

- To obtain dE/dx from dQ/dx need to invert recombination model

Birks: $$dE/dx = \frac{dQ/dx}{A_B/W_{ion} - k_B \cdot (dQ/dx)/\mathscr{E}}.$$

More stable at large dQ/dx as in Birks model the denominator → 0

Box: $$dE/dx = (exp(\beta W_{ion} \cdot (dQ/dx)) - \alpha)/\beta.$$

$$\alpha = A$$
$$\beta = B/\varepsilon$$

larsoft code

- Look at "ionization and scintillation" part of larsim for electron/light prod
- Normally the fastest way to find set parameters is to run fhicl-dump of <your_job>.fcl:

```
fhicl-dump my_job.fcl
```

- ***Exception***: if parameters are not specified by default in fcl! (i.e., fhicl-dump will only show what is declared in various fcl files)

DetectorPropertiesStandard:

The default values are used if no fcl parameters are specified

```
fhicl::Atom<double> ModBoxAlpha{
  Name("ModBoxAlpha"),
  Comment("alpha parameter in the Modified Box recombination model."),
  util::kModBoxA};

fhicl::Atom<double> ModBoxBeta{
  Name("ModBoxBeta"),
  Comment("beta parameter in the Modified Box recombination model."),
  util::kModBoxB};
```

util::kModBoxA(B) come from PhysicalConstants.h

```
constexpr double kModBoxA = 0.930;
constexpr double kModBoxB = 0.212;
```

larsoft code

DetectorPropertiesStandard   Birks function can also be found there

```
double DetectorPropertiesStandard::ModBoxCorrection(double dQdx, double E_field) const
{
  // Modified Box model correction has better behavior than the Birks
  // correction at high values of dQ/dx.
  double const rho = Density();                          // LAr density in g/cm^3
  constexpr double Wion = 1000. / util::kGeVToElectrons; // 23.6 eV = 1e, Wion in MeV/e
  double const Beta = fModBoxB / (rho * E_field);
  double const Alpha = fModBoxA;
  double const dEdx = (exp(Beta * Wion * dQdx) - Alpha) / Beta;

  return dEdx;
}
```

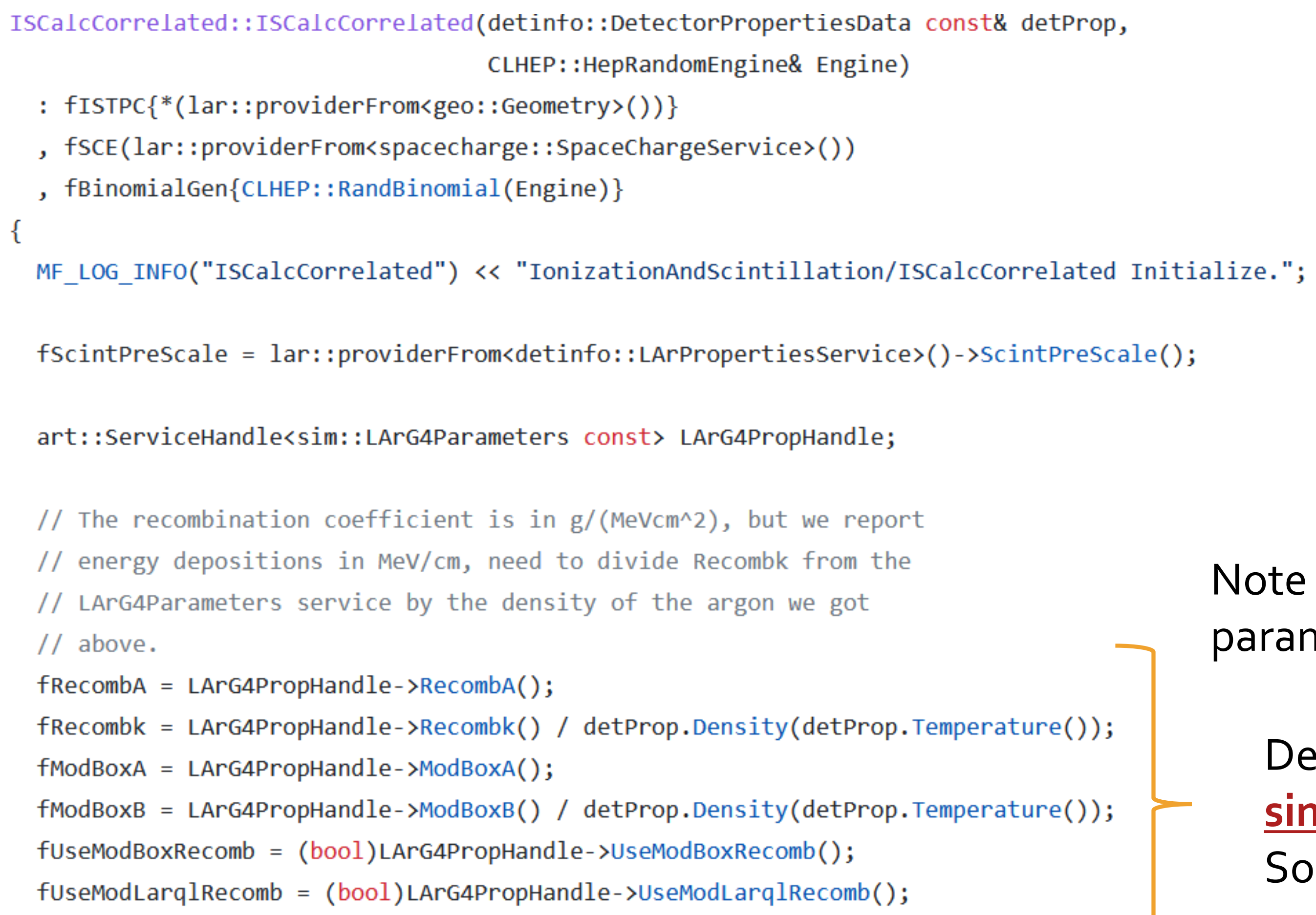
$$dE/dx = (exp(\beta W_{ion} \cdot (dQ/dx)) - \alpha)/\beta.$$

This is what is called when one asks to calculate dE/dx from dQ/dx in CalorimetryAlg

```
double CalorimetryAlg::dEdx_from_dQdx_e(detinfo::DetectorClocksData const& clock_data,
                                        detinfo::DetectorPropertiesData const& det_prop,
                                        double dQdx_e,
                                        double const time,
                                        double const T0,
                                        double const EField) const
{
  if (fDoLifeTimeCorrection) {
    dQdx_e *= LifetimeCorrection(clock_data, det_prop, time, T0); // (dQdx_e in e/cm)
  }

  if (fUseModBox) { return det_prop.ModBoxCorrection(dQdx_e, EField); }

  return det_prop.BirksCorrection(dQdx_e, EField);
}
```

More FCL parameter complications

Ex. ISCalcCorrelated.cxx



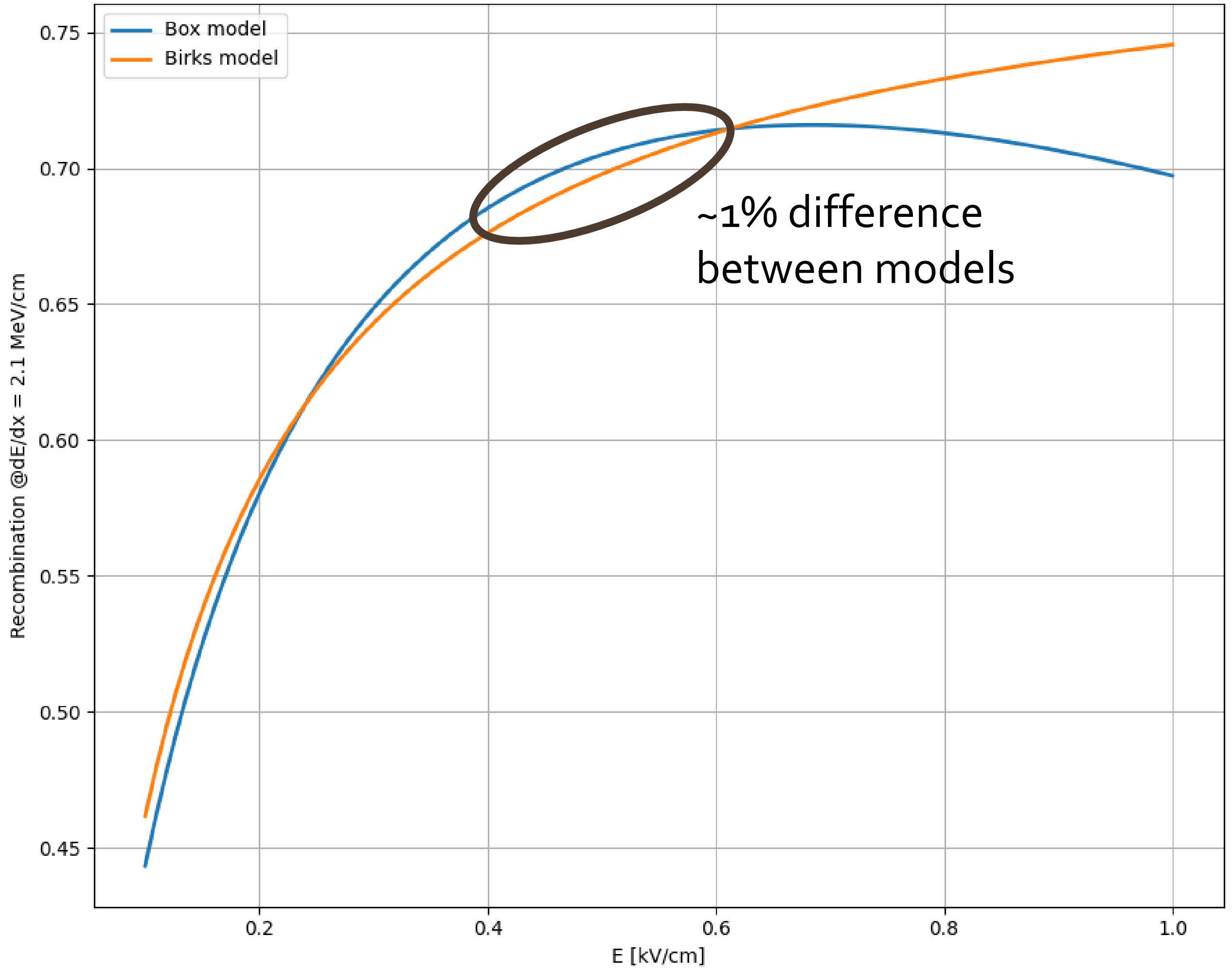
```
ISCalcCorrelated::ISCalcCorrelated(detinfo::DetectorPropertiesData const& detProp,
                                   CLHEP::HepRandomEngine& Engine)
  : fISTPC{*(lar::providerFrom<geo::Geometry>())}
  , fSCE(lar::providerFrom<spacecharge::SpaceChargeService>())
  , fBinomialGen{CLHEP::RandBinomial(Engine)}
{
  MF_LOG_INFO("ISCalcCorrelated") << "IonizationAndScintillation/ISCalcCorrelated Initialize.";

  fScintPreScale = lar::providerFrom<detinfo::LArPropertiesService>()->ScintPreScale();

  art::ServiceHandle<sim::LArG4Parameters const> LArG4PropHandle;

  // The recombination coefficient is in g/(MeVcm^2), but we report
  // energy depositions in MeV/cm, need to divide Recombk from the
  // LArG4Parameters service by the density of the argon we got
  // above.
  fRecombA = LArG4PropHandle->RecombA();
  fRecombk = LArG4PropHandle->Recombk() / detProp.Density(detProp.Temperature());
  fModBoxA = LArG4PropHandle->ModBoxA();
  fModBoxB = LArG4PropHandle->ModBoxB() / detProp.Density(detProp.Temperature());
  fUseModBoxRecomb = (bool)LArG4PropHandle->UseModBoxRecomb();
  fUseModLarqlRecomb = (bool)LArG4PropHandle->UseModLarqlRecomb();
```

Note that for simulation the recombination parameters are in a LArG4Parameters

Defined in standard_largeantparameters in **simulationservices.fcl**
So as far as I can tell one defines the same parameters in different places.

From dQ/dx to dE/dx

- Converting dQ/dx to dE/dx is not a just inversion of recombination
- The charge that is measured needs to be corrected for a number of effects:
  - E.g., attenuation due to attachment to electro-negative impurities
- The drift field non-uniformity needs to be assed and included, since recombination rate depends on the local field strength



Drifting charge

- Ionization electrons drift in applied electric field
- Some fraction of the charge is captured by electronegative impurities like O2
- The electrons bounce off the atoms as they traverse the medium
  - Diffusion of original cloud both longitudinally and transversely to the direction motion
- The average speed “drift velocity” is a function of field strength and temperature (density)

Electron attachment to impurities

- With capture rate to a given molecule species k_s (E-field dependence) and assuming a constant concentration of impurities $n_s \gg N_e$, the number of electrons captured per time interval dt:

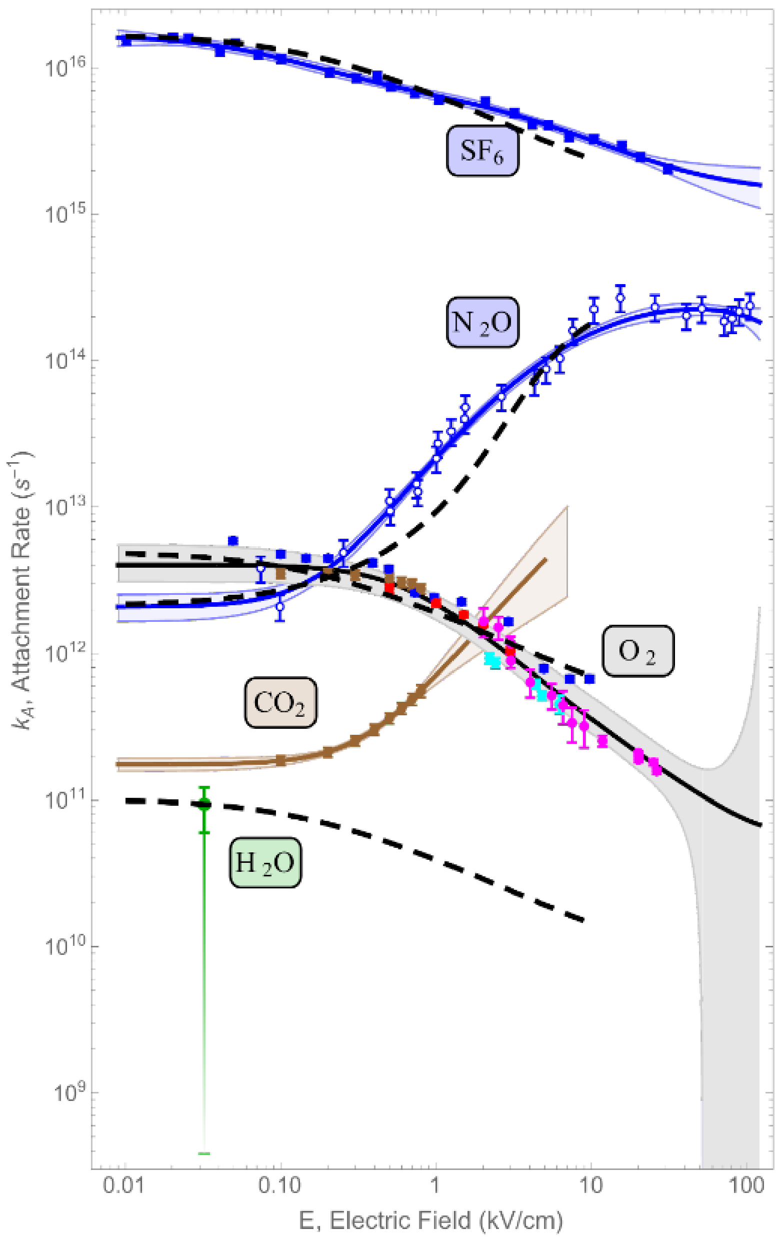
$$\frac{dN_e}{dt} = -k_s n_s N_e$$

With:
$$\tau_e = \frac{1}{k_s n_s}$$

Fitted from data

$$N_e(t) = N_e(0)e^{-t/\tau_e}$$

From 2022 *JINST* **17** T11007



Electron attachment to impurities

$$N_e(t) = N_e(0)e^{-t/\tau_e}$$

- The electron lifetime τ_e is measured in s (on a ms scale)
- One expresses concentration of impurities in oxygen equivalents (the dominant)
- If τ_e measured in ms, what equivalent concentration of O2 this corresponds to?

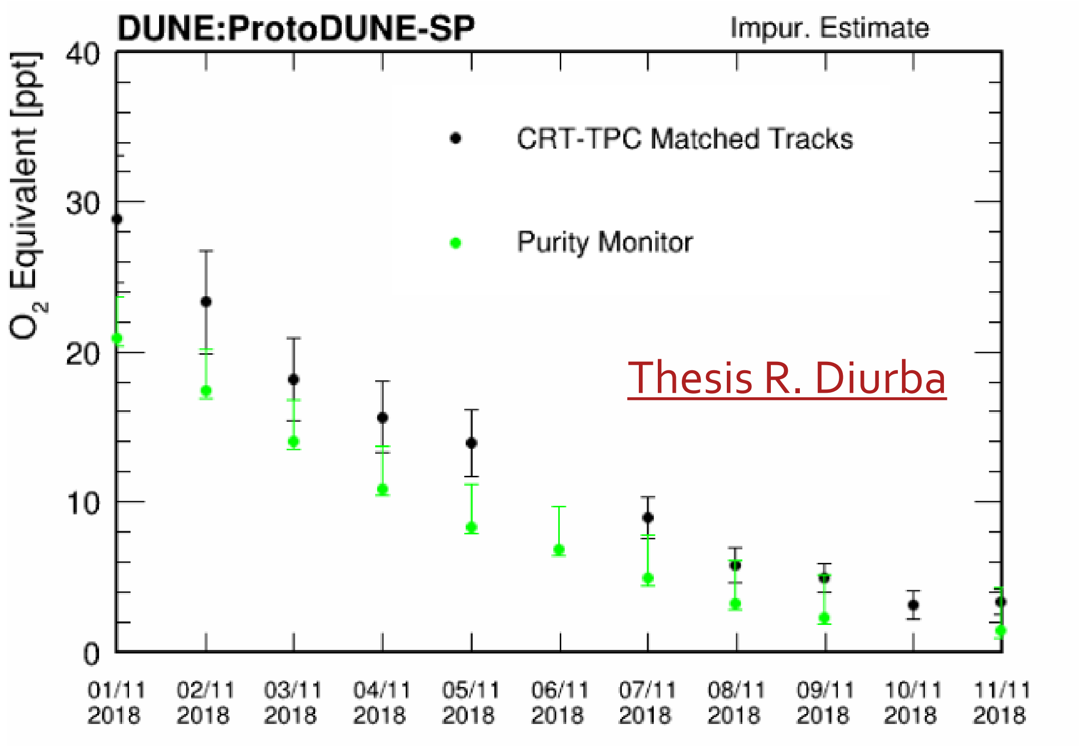
$$\rho_{O_2}[\mathrm{ppb}] = 10^9 \times n_s/n_{Ar}$$

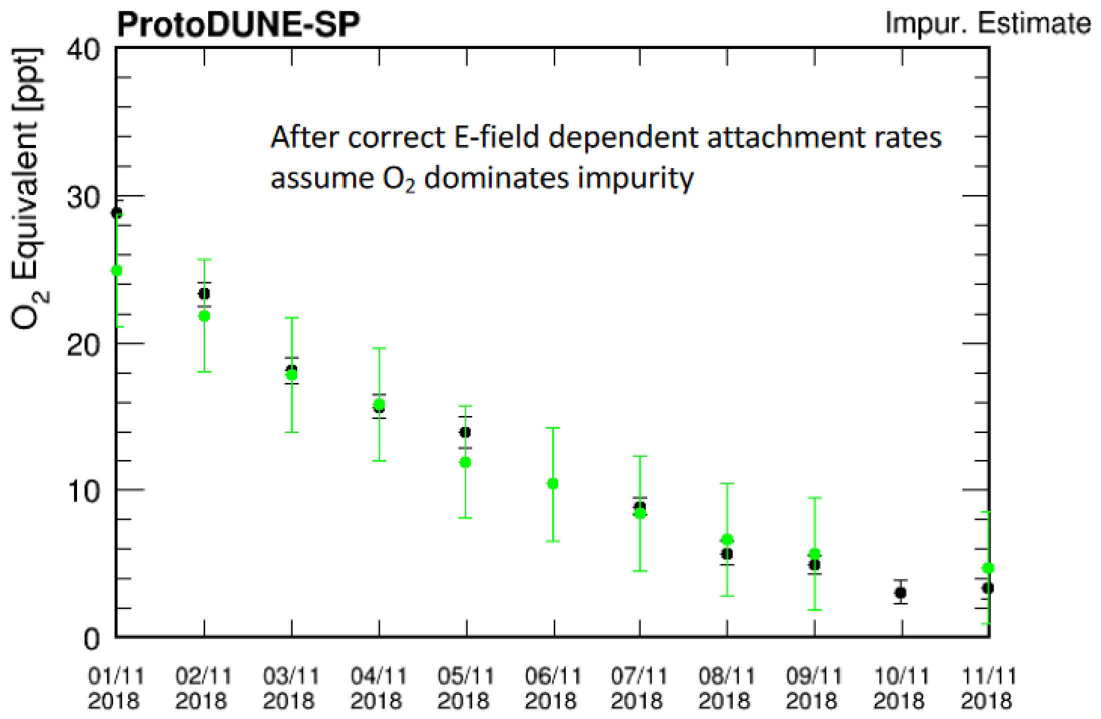
with n_{Ar} number of moles per L of LAr = 35

With $k_s \sim 10^{11}$ [L/mol/s] for O2

$$\tau_e[\mathrm{ms}] \sim \frac{0.3}{\rho_{O_2}[\mathrm{ppb}]}$$

Purity monitor and purity in TPC





- Purity monitors (10-100 V/cm) and TPC (500 V/cm) work at different fields: impact on the attachment rate constants
- From parametrization of O2 data $k_s(E_{PrM})/k_s(E_{TPC}) \sim 1.3$

Electron diffusion

- In presence of external electric field ("drift field") electrons are accelerated between collisions. In each collision:
  - They lose some part of energy
  - They lose the original direction
- On average the electron cloud n moves or drifts towards anode with "drift" velocity v_d:

Current conservation of moving and diffusing charge current $J = \boldsymbol{v_d} n - D\nabla n$

$$\frac{\partial n}{\partial t} = D\nabla^2 n - v_d \frac{\partial n}{\partial x}$$

Drift direction in X with velocity v_d & no losses due to impurities

- Electrons have position dependent energy within the cloud
  - Electrons diffused to the leading edge of the swarm tend to be more energetic and therefore less sensitive to drift field effects
  - These fall back to the middle/tail of the bunch when they loose enough energy
  - Semiquantitative treatment in [Phys. Rev. 181 (1969)] solving Boltzmann transport equations
- Leads anisotropic diffusion with two different diffusions coefficients D_L and D_T in the direction parallel and perpendicular to the electric field with $D_L \le D_T$:

$$\frac{\partial n}{\partial t} = D_L \frac{\partial^2 n}{\partial x^2} + D_T \left(\frac{\partial^2 n}{\partial y^2} + \frac{\partial^2 n}{\partial z^2} \right) - v_d \frac{\partial n}{\partial x}$$

Electron diffusion

- Effective solution for anisotropic diffusion:

$$n(r,t) = \frac{n_0}{4\pi D_T t\sqrt{4\pi \mathrm{D}_L t}} \exp\left[-\frac{(x - v_d t)^2}{4D_L t} - \frac{y^2 + z^2}{4D_t t}\right]$$

- The cloud dimensions increase with time:

$$\sigma_T^2(t) = 2D_T t + \boxed{\rho_0^2}$$
$$\sigma_L^2(t) = 2D_L t + \boxed{\sigma_0^2}$$

Some initial dimensions of the electron cloud

- For longitudinal diffusion it is more natural to express width in time units (the drift coordinate in TPC):

$$\sigma_{L,t}^2 = \frac{2D_L t}{v_d^2} + \sigma_{0,t}^2 = \frac{2D_L x}{v_d^3} + \sigma_{0,t}^2$$

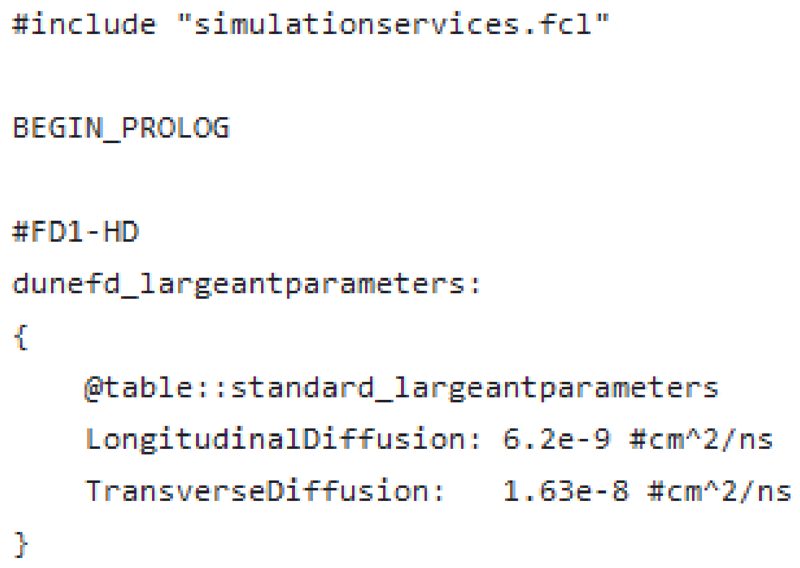
Electron diffusion DL/DT values

- The sim values for DL / DT:

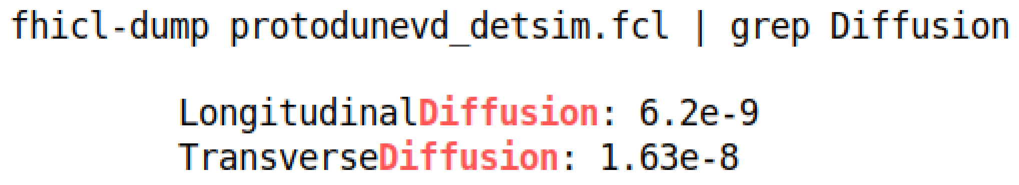
$$D_L = 6.2 \text{ cm}^2/\text{s}$$
$$D_T = 16.3 \text{ cm}^2/\text{s}$$

- These come from to theoretical calculations and higher than experimental data (docdb-14407)
- The docdb offers: (L) 5.3 cm²/s & (T) 12.8 cm²/s
- The values are higher than existing data
  - ICARUS: $D_L = 4.74$ cm²/s
  - In Protodune-SP data: $D_L = 3.91$ cm²/s
- Ideally should try to be coherent at least within DUNE …

simulationservices_dune.fcl



```
#include "simulationservices.fcl"

BEGIN_PROLOG

#FD1-HD
dunefd_largeantparameters:
{
    @table::standard_largeantparameters
    LongitudinalDiffusion: 6.2e-9 #cm^2/ns
    TransverseDiffusion:   1.63e-8 #cm^2/ns
}
```



```
fhicl-dump protodunevd_detsim.fcl | grep Diffusion

        LongitudinalDiffusion: 6.2e-9
        TransverseDiffusion: 1.63e-8
```

DL measured with gold photocathode

$$D_L = \frac{\mu \epsilon_L}{e} = \left(\frac{a_0 + a_1 E + a_2 E^{3/2} + a_3 E^{5/2}}{1 + (a_1/a_0)E + a_4 E^2 + a_5 E^3}\right)\left(\frac{b_0 + b_1 E + b_2 E^2}{1 + (b_1/b_0)E + b_3 E^2}\right)\left(\frac{T}{T_0}\right)^{-3/2}\left(\frac{T}{T_1}\right)$$

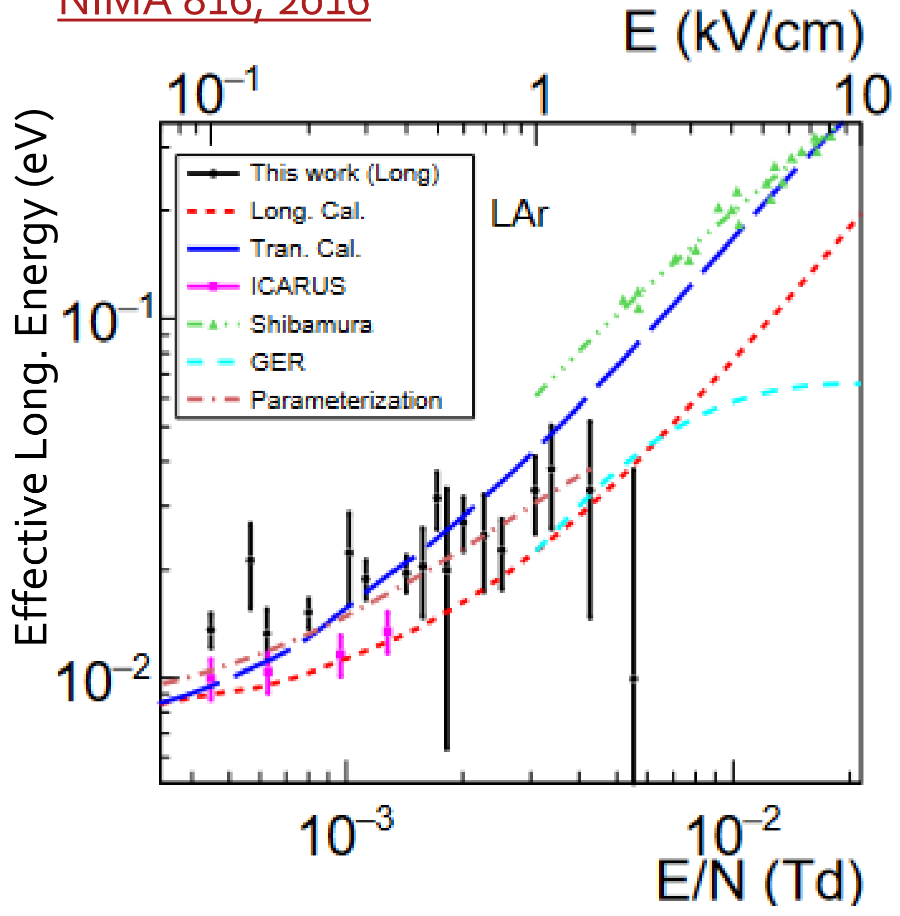
Table 1: Function paraemeters in Eq. (21)

a_0	=	551.6
a_1	=	7953.7
a_2	=	4440.43
a_3	=	4.29
a_4	=	43.63
a_5	=	0.2053

Table 2: Function parameters in Eq. (22)

b_0	=	0.0075
b_1	=	742.9
b_2	=	3269.6
b_3	=	31678.2

NIMA 816, 2016



- This calculator from BNL group:
  - DL = 6.63 cm2/s
  - DT = 13.23 cm2/s

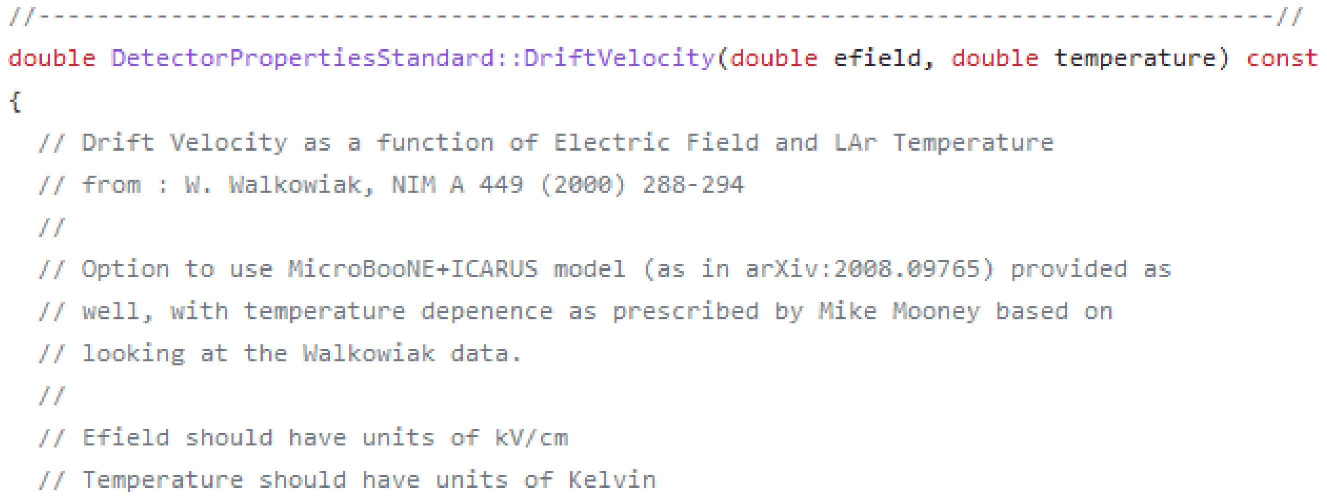
$$\frac{D_L}{D_T} = 1 + \frac{E}{\mu}\frac{\partial \mu}{\partial E}$$

See refs in NIMA 816 for more details

Drift velocity in liquid argon

- Drift velocity has dependence on not only field strength but also temperature
- A smooth interpolation of a so-called "Walkowiak" parameterization [NIM A 449 (2000) 288-294] and fitted ICARUS measurements below 0.7 kV/cm is a default option in larsoft
- Not sure about wirecell. In this paper [2018 *JINST* **13** P07006] it says "the electron drift velocity as function of electric field is taken from recent measurements..."

From DetectorPropertiesStandard

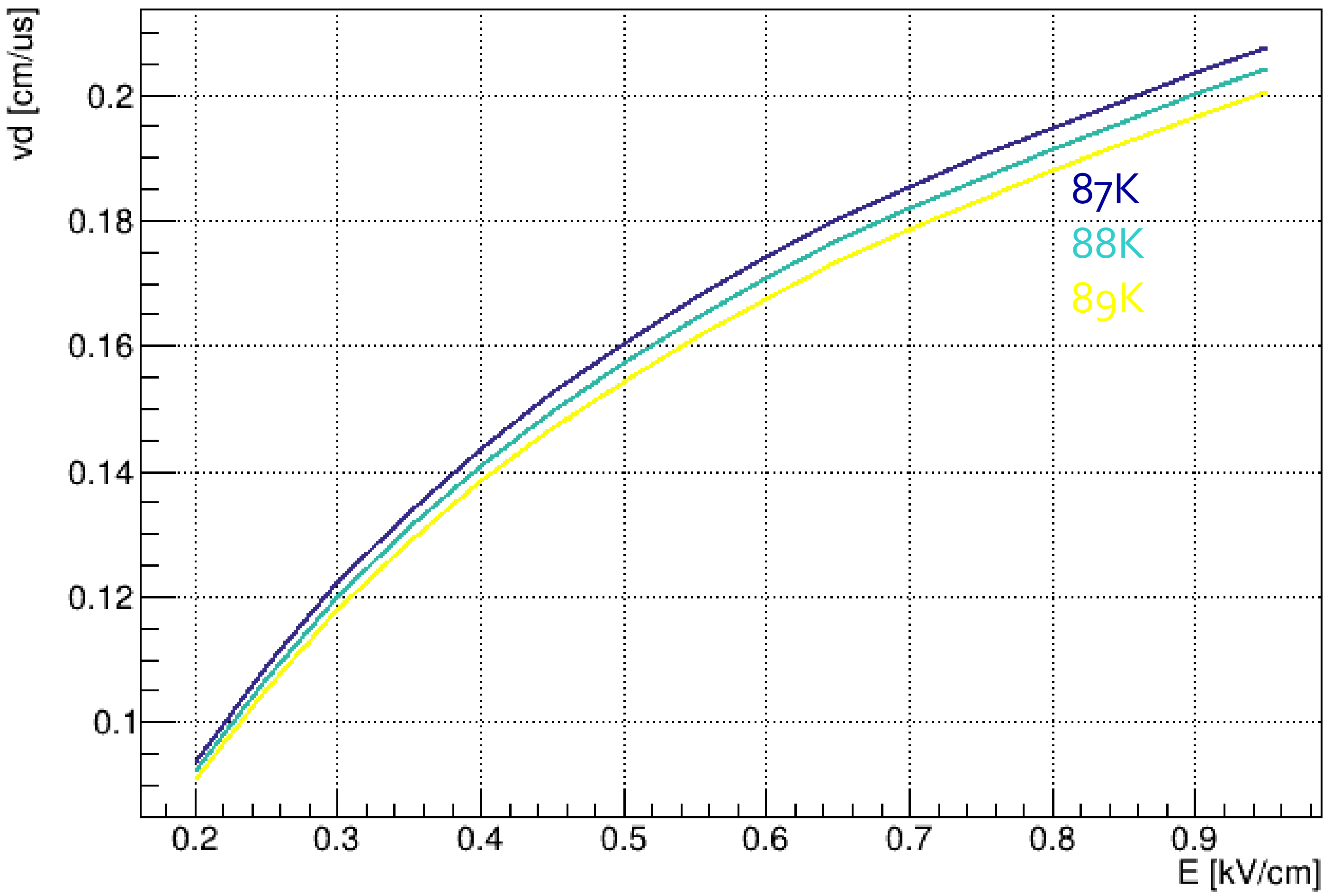


```
//--------------------------------------------------------------------------------------//
double DetectorPropertiesStandard::DriftVelocity(double efield, double temperature) const
{
  // Drift Velocity as a function of Electric Field and LAr Temperature
  // from : W. Walkowiak, NIM A 449 (2000) 288-294
  //
  // Option to use MicroBooNE+ICARUS model (as in arXiv:2008.09765) provided as
  // well, with temperature depenence as prescribed by Mike Mooney based on
  // looking at the Walkowiak data.
  //
  // Efield should have units of kV/cm
  // Temperature should have units of Kelvin
```

- At 0.5 kV/cm vd = 1.6 mm/us
- 3m drift distance in constant field: ~1.9 ms

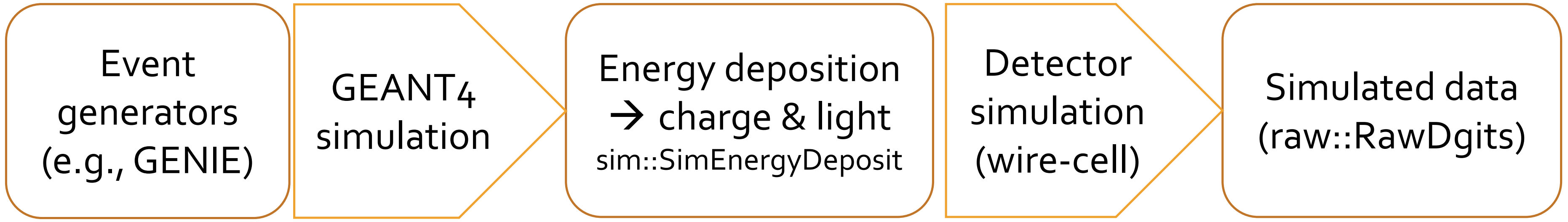
- Need a decent electron lifetime to see charge close to the cathode
- Example: $\tau_e = 5$ ms, max attenuation ~30%

Drift velocity in liquid argon



1K diff ~2% effect on drift velocity @0.5kV/cm

Detector simulation workflow


Event generators (e.g., GENIE)
GEANT4 simulation
Energy deposition → charge & light
sim::SimEnergyDeposit
Detector simulation (wire-cell)
Simulated data (raw::RawDgits)

- Some care must be taken for WCL (Wire-Cell) module to pick up the "transport" parameters from larsoft services:

In wirecell_dune.fcl

For protodunespmc (set interpedently of larsoft):

```
// Make available parameters via Jsonnet's std.extVar()
params: {
}
structs: {
    // Longitudinal diffusion constant [cm2/s]
    DL: 4.0
    // Transverse diffusion constant [cm2/s]
    DT: 8.8
    // Electron lifetime [ms]
    lifetime: 35.0
    // Electron drift speed from SP measurement
    driftSpeed: 1.565
}
```

Example protodunevd (using values from larsoft services):

```
structs: {
   nticks: @local::protodunevd_services.DetectorPropertiesService.NumberTimeSamples
   lifetime: @local::protodunevd_services.DetectorPropertiesService.Electronlifetime
   DL: @local::dunefd_largeantparameters.LongitudinalDiffusion
   DT: @local::dunefd_largeantparameters.TransverseDiffusion
   efield: @local::protodunevd_services.DetectorPropertiesService.Efield[0] # kV/cm
   temperature: @local::protodunevd_services.DetectorPropertiesService.Temperature # K
}
```

The drift speed is computed by WCL then

Passing info to WCL

```
wirecell_protodunevd_mc:
{
    module_type : WireCellToolkit
    wcls_main: {
        tool_type: WCLS
        apps: ["Pgrapher"]
        plugins: ["WireCellPgraph", "WireCellGen","WireCellSio","WireCellRoot","WireCellLarsoft"]
        // needs to be found via your WIRECELL_PATH
        configs: ["pgrapher/experiment/protodunevd/wcls-sim-drift-simchannel.jsonnet"]
        inputers: ["wclsSimDepoSource:electron"]
        outputers: [
            "wclsSimChannelSink:postdrift",
            "wclsFrameSaver:simdigits"
        ]

        // Make available parameters via Jsonnet's std.extVar()
        params: {
        }
        structs: {
           nticks: @local::protodunevd_services.DetectorPropertiesService.NumberTimeSamples
           lifetime: @local::protodunevd_services.DetectorPropertiesService.Electronlifetime
           DL: @local::dunefd_largeantparameters.LongitudinalDiffusion
           DT: @local::dunefd_largeantparameters.TransverseDiffusion
           efield: @local::protodunevd_services.DetectorPropertiesService.Efield[0] # kV/cm
           temperature: @local::protodunevd_services.DetectorPropertiesService.Temperature # K
        }
    }
}
```

wcls-sim-drift-simchannel.jsonnet

```
local wcls_input = {
  // depos: wcls.input.depos(name="", art_tag="IonAndScint"),
  depos: wcls.input.depos(name='electron', art_tag='IonAndScint'),
};
```

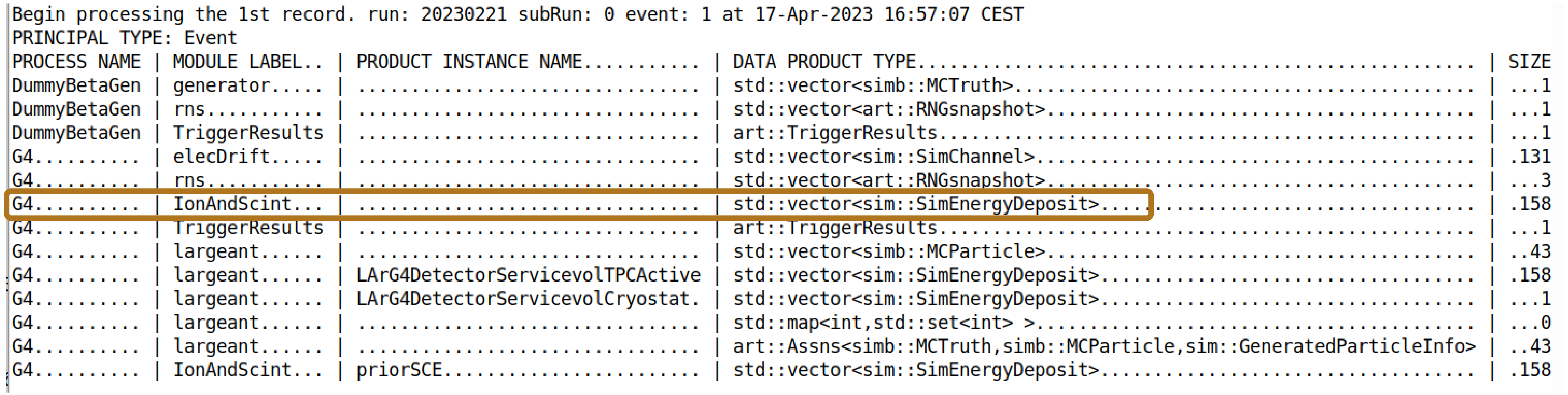
Need IonAndScint sim::Energy art product

```
local params = base {
  daq: super.daq {
    nticks: std.extVar('nticks'),
  },
  lar: super.lar {
    // Longitudinal diffusion constant
    DL: std.extVar('DL') * wc.cm2 / wc.s,
    // Transverse diffusion constant
    DT: std.extVar('DT') * wc.cm2 / wc.s,
    // Electron lifetime
    lifetime: std.extVar('lifetime') * wc.ms,
    // Electron drift speed
    // drift_speed: std.extVar('driftSpeed') * wc.mm / wc.us,
    drift_speed: util.drift_velocity(std.extVar('efield'), std.extVar('temperature')) * wc.mm / wc.us,
  },
};
```

drift_speed is computed here from field value and temperature
Q: How does one treat non-uniformities in drift field (e.g., due to SCE)?

Example eventdump.fcl

- To quickly examine art products in a file use eventdump.fcl
- Example: lar –c eventdump.fcl <my_file>.root –n 1



```
Begin processing the 1st record. run: 20230221 subRun: 0 event: 1 at 17-Apr-2023 16:57:07 CEST
PRINCIPAL TYPE: Event
PROCESS NAME | MODULE LABEL.. | PRODUCT INSTANCE NAME........... | DATA PRODUCT TYPE.................................................. | SIZE
DummyBetaGen | generator..... | ................................ | std::vector<simb::MCTruth>......................................... | ...1
DummyBetaGen | rns........... | ................................ | std::vector<art::RNGsnapshot>...................................... | ...1
DummyBetaGen | TriggerResults | ................................ | art::TriggerResults................................................ | ...1
G4.......... | elecDrift..... | ................................ | std::vector<sim::SimChannel>....................................... | .131
G4.......... | rns........... | ................................ | std::vector<art::RNGsnapshot>...................................... | ...3
G4.......... | IonAndScint... | ................................ | std::vector<sim::SimEnergyDeposit>................................. | .158
G4.......... | TriggerResults | ................................ | art::TriggerResults................................................ | ...1
G4.......... | largeant...... | ................................ | std::vector<simb::MCParticle>...................................... | ..43
G4.......... | largeant...... | LArG4DetectorServicevolTPCActive | std::vector<sim::SimEnergyDeposit>................................. | .158
G4.......... | largeant...... | LArG4DetectorServicevolCryostat. | std::vector<sim::SimEnergyDeposit>................................. | ...1
G4.......... | largeant...... | ................................ | std::map<int,std::set<int> >....................................... | ...0
G4.......... | largeant...... | ................................ | art::Assns<simb::MCTruth,simb::MCParticle,sim::GeneratedParticleInfo> | ..43
G4.......... | IonAndScint... | priorSCE........................ | std::vector<sim::SimEnergyDeposit>................................. | .158
```

According to WCL config it should look for IonAndScint product

On to part 2 …