New service for CRP gain retrieval for simulation

Vyacheslav Galymov
IP2I Lyon

Why?

- Currently simulation does not include effects of LEM dead areas
  - Difficult to study tracking performance without MC being able to reproduce effects of these dead spaces
    - track break-up due to gaps / stitching
  - Eventually check impact on EM shower resolution (reconstruction of ν_e energy spectrum)
- Possibility to include some variation in gains of each LEM
- Single interface for CRP effective charge gain retrieval for both charge and light (needed for S2 yield) simulations

Current simulation of CRP effective gain

```
int DPhaseSimChannelExtractService::
extract(const sim::SimChannel* psc, AdcSignalVector& sigs) const {

  // clear and resize temporary ADC buffer
  sigs.clear();
  sigs.resize(m_ntick, 0.0);


  std::vector<double> sigs_original;
  sigs_original.resize(m_ntick, 0.0);

  if ( psc == nullptr ) return 0;

  // get the channel number
  unsigned int chan = psc->Channel();

  //CLHEP::RandGaussQ rGauss(*m_pran, 0.0, fRedENC);

  for ( size_t itck=0; itck<sigs.size(); ++itck )
    {
      sigs[itck] = fDPGainPerView * psc->Charge(itck);
    }

  // perform convolution

  m_psss->Convolute(chan, sigs);

  return 0;
}
```

DP SimChannel extractor service: Creates waveforms on each channel from the simulated charge depositions on the wires

Fixed gain per view right now for all CRPs

➔ Should introduce variable gain factor here

However, need 2D information of the projected charge (not a 1D single channel)

“Imperfect” solution for LEM effects

- The “drift” of charge is done in larsoft SimDriftElectrons__module
  - The charge is assigned to channel/tdc from XYZ of deposit and taking into account LAr purity, diffusion, drift velocity, quenching effects …
- The position of the projected charge on the readout planes are not stored
- However, XYZ the energy deposit in the world coordinates is currently available via *SimChannel::TrackIDEs(TDC_t startTDC, TDC_t endTDC)*
- Can do 2D mapping needed for LEM gain / dead area effects
  - But this would ignore the diffusion effects as well as space-charge effects on the drifted charges → not the best solution

```
struct IDE{

  typedef int TrackID_t;

  IDE();


  IDE(IDE const& ide, int offset);

  IDE(TrackID_t tid,
      float nel,
      float e,
      float xpos,
      float ypos,
      float zpos)
  : trackID      (tid)
  , numElectrons (nel)
  , energy       (e)
  , x            (xpos)
  , y            (ypos)
  , z            (zpos)
  {}


  TrackID_t trackID;
  float numElectrons;
  float energy;
  float x;
  float y;
  float z;
}; // struct IDE
```

- Expand IDE structure in sim::simChannel to include a minimum doublet float[2] of projected position of the cluster on the readout plane

- Add transverse part of the projected position in SimDriftElectrons_module

- This would take care of any diffusion (and space-charge) effects when mapping to CRP LEMs

CrpGainService

- Can include:
  - Effect of LEM dead areas
  - Variation in LEM gains across CRPs (to do)
- Three methods available

Called by the DP SimChannel extractor service

```
// get charge collected on a view after amplification in CRP
double viewCharge( const sim::SimChannel* psc, unsigned itck ) const;

// calculate the gain based on position information
double crpGain( geo::Point_t const &pos ) const;

// default value of the effective gain
double crpDefaultGain() const { return m_CrpDefGain; }
```

Retrieves the effective gain value:
e.g., can be used for optical simulations to calculate number of S2 photons

CrpGainService

- Loops over IDEs and gets effective gain factor based on 2D information
- Uses the position information at the point of charge deposition

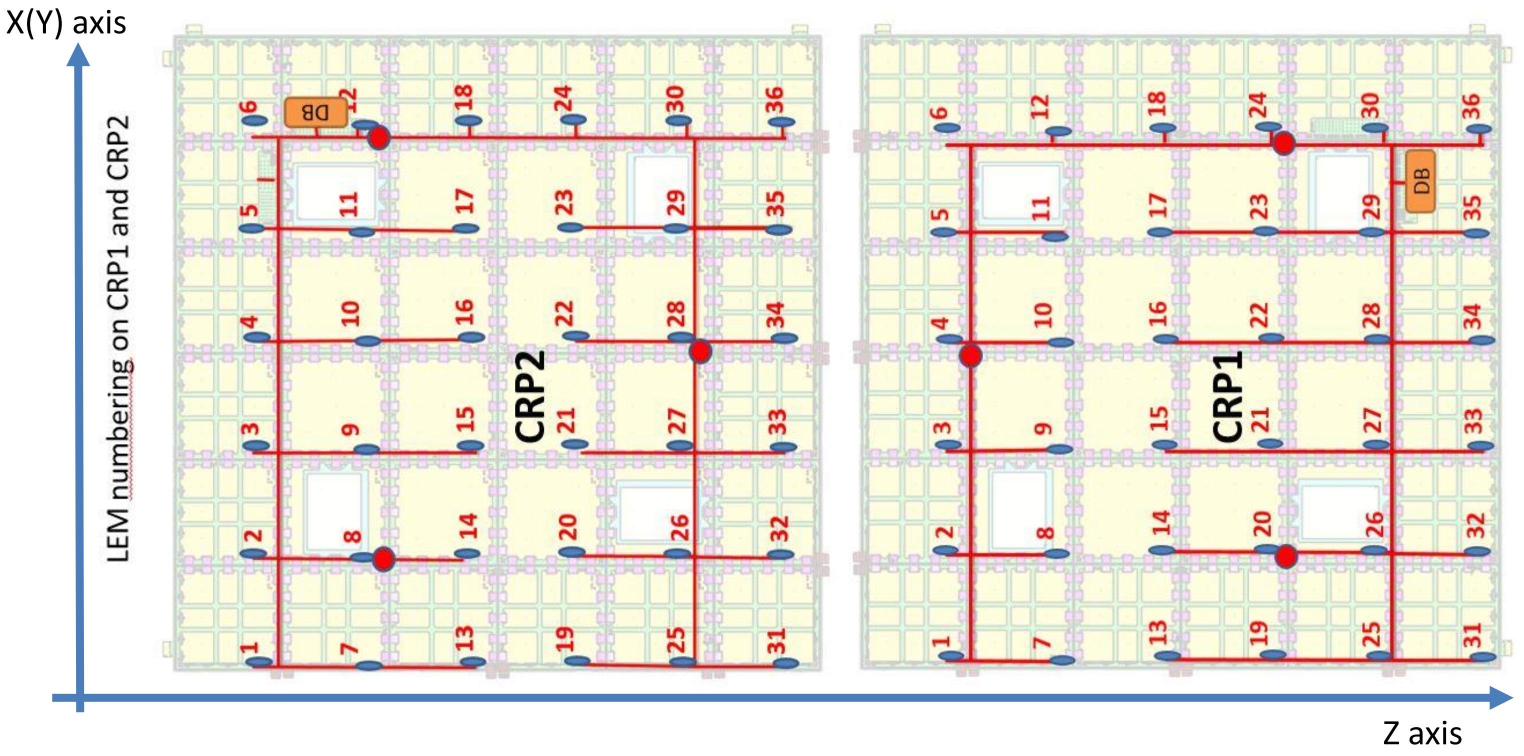
```
for(auto &ide: IDEs)
  {

    // get the wire number in the other view for this position
    int wother = pother.WireCoordinate( geo::Point_t{ide.x, ide.y, ide.z} );
    if( wother < 0 || wother >= (int)pother.Nwires() )
      {
        cout<<myname<<"WARNING the wire number appeares to be incorrect "<<wother<<"\n";
        continue;
      }

    double G = 0;
    if( tcoord < 2 ) // we are in view kZ
      {
        G = getCrpGain( wire, wother );
      }
    else // we are in view kX or kY
      {
        G = getCrpGain( wother, wire );
      }
    // the charge is divided equially between collectiong views
    // so the effective gain per view is 1/2 of the total effective CRP gain
    qsum += (0.5 * G) * ide.numElectrons;
  }
```

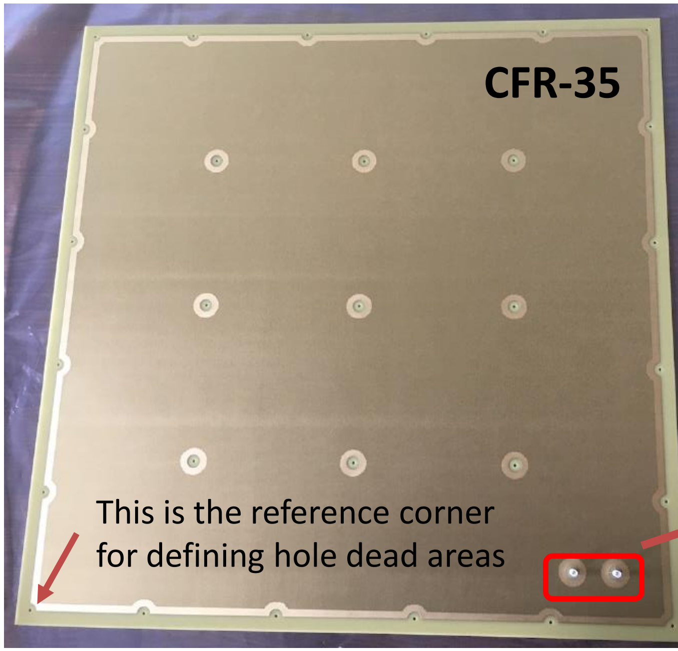
LEM numbering convention adopted in simulation

Convention to for LEM numbering that will be followed, when the LEM gain is specified for each unit: CRP# LEM# <gain value>



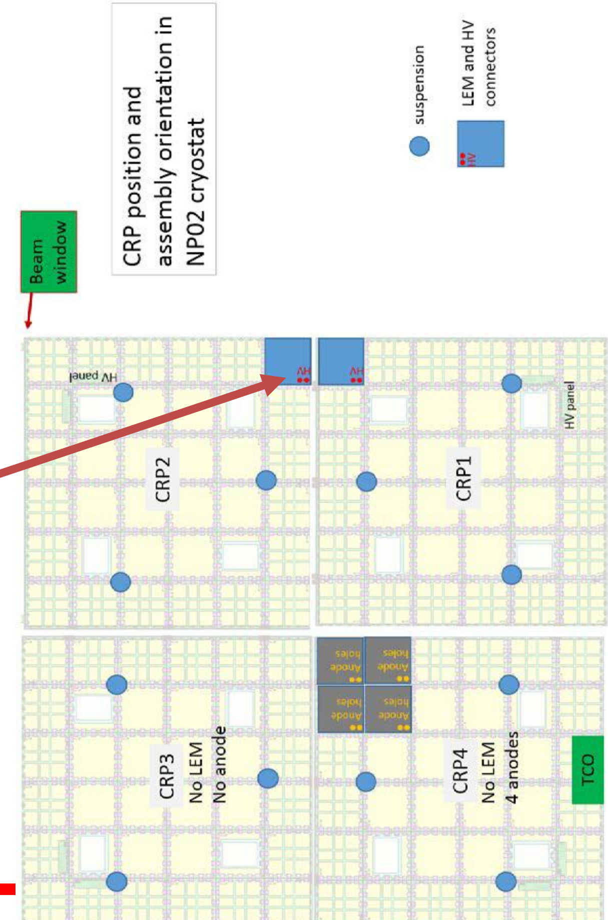
Need to take care of the actual position of the HV connections in these coordinates to correctly specify the dead area due to these utility holes

LEM dead areas



This is the reference corner for defining hole dead areas

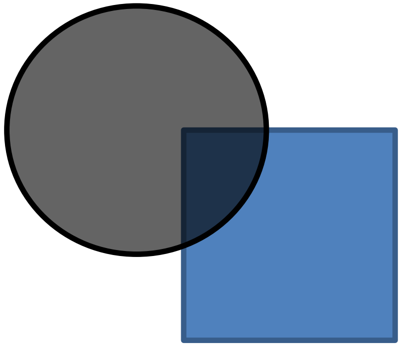
Need to know to mask the correct sections of the anode



		LEM borders		Screw holes			
LEM design	% Active area	FR4	copper guard ring	FR4 ring Φ	copper guard ring Φ	FR4 ring Φ	copper guard ring Φ
CFR-34	96.2	2 mm	2 mm	4.2 mm	6 mm	10 mm	12 mm
CFR-35	85.8	10 mm	5 mm	10 mm	20 mm	10 mm	20 mm
CFR-36	92.1	2 mm	5 mm	10 mm	20 mm	10 mm	20 mm

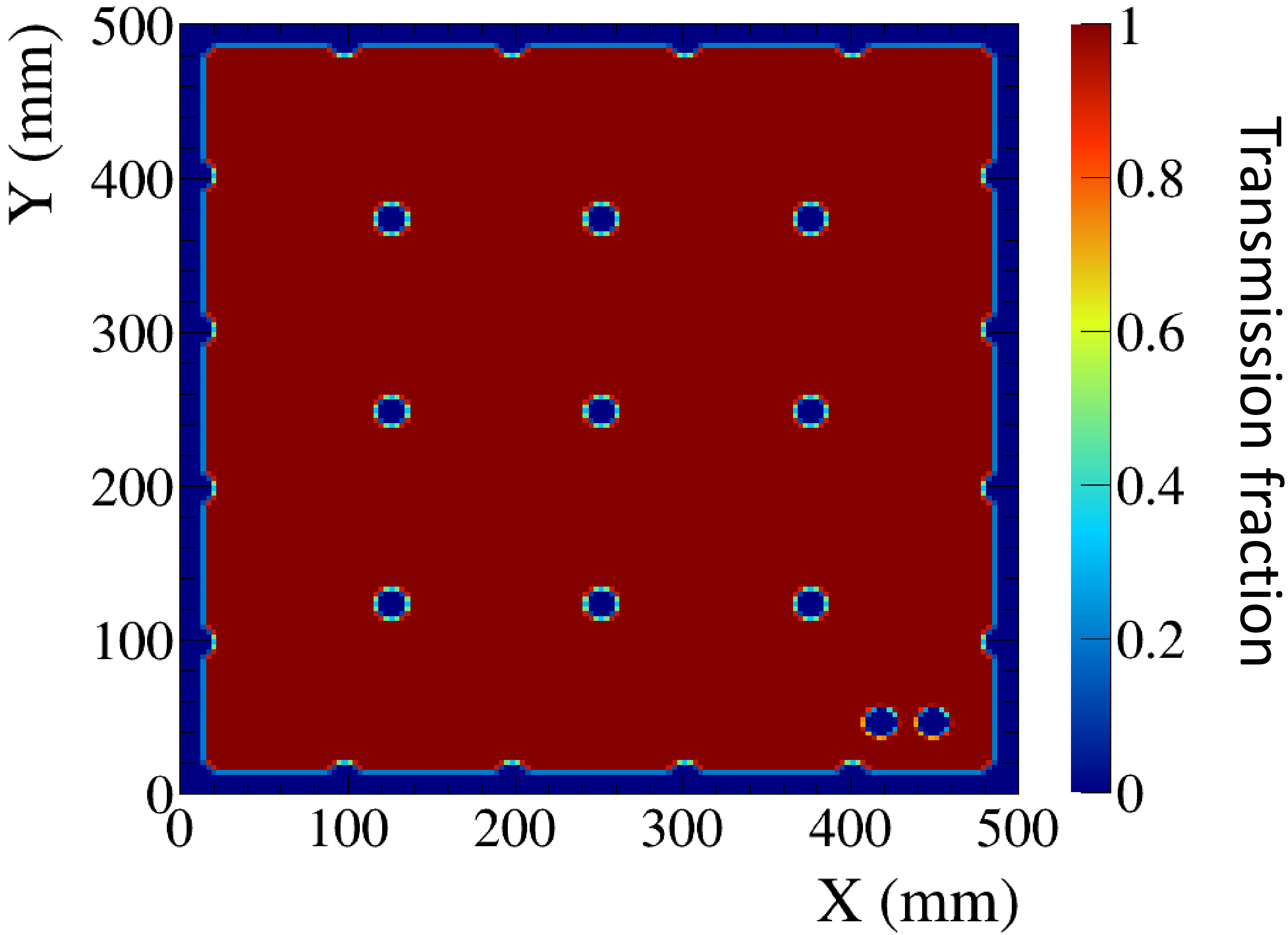
Transmission map calculation

- The transmission coefficient for each channel is calculated simply as a fraction of an area overlap between LEM dead region and a square pixel of 3.125 mm x 3.125 mm



- The area overlap is calculated using Monte Carlo (fall-in hits/total throws)

Calculated transmission map



Configuration

In dune/Utilities/crp_gain.fcl

```
dunefddphase_crpgain: {
   service_provider: CrpGainService
   LogLevel:         1
   CrpDefaultGain:   6
   CrpNumLem:        36
   LemViewChans:     160
   LemEffTool:       ""
}

# protodune dp LEM efficiency map for CFR-35 design
protodunedphase_crpgain:            @local::dunefddphase_crpgain
protodunedphase_crpgain.LemEffTool: "lemEfficiency"
```

Default effective gain (divided by 2 per each collection view) same as specified in TDR requirements

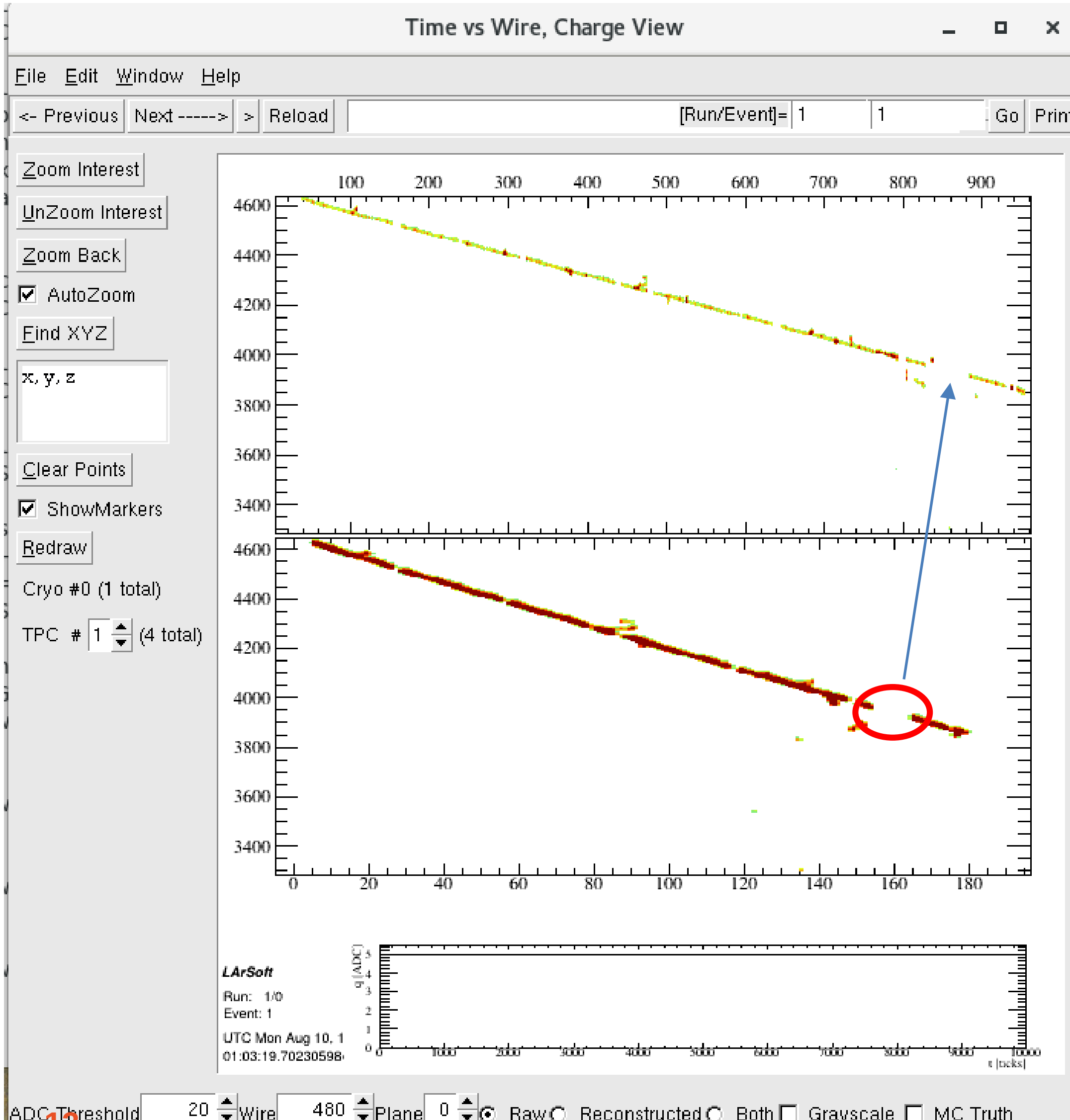
Configuration of CRP readout geometry (this is checked also with respect to the declared geometry)

Includes LEM dead area effects using the calculated transmission map
Default service configuration is without (issues with CI otherwise)

```
tools.lemEfficiency: {
  tool_type: FclFileFloatArray
  LogLevel: 1
  Label: "lem_eff_29102019"
  DefaultValue: 0.0
  FileNames: ["lem_efficiency_v29102019.fcl"]
}
```

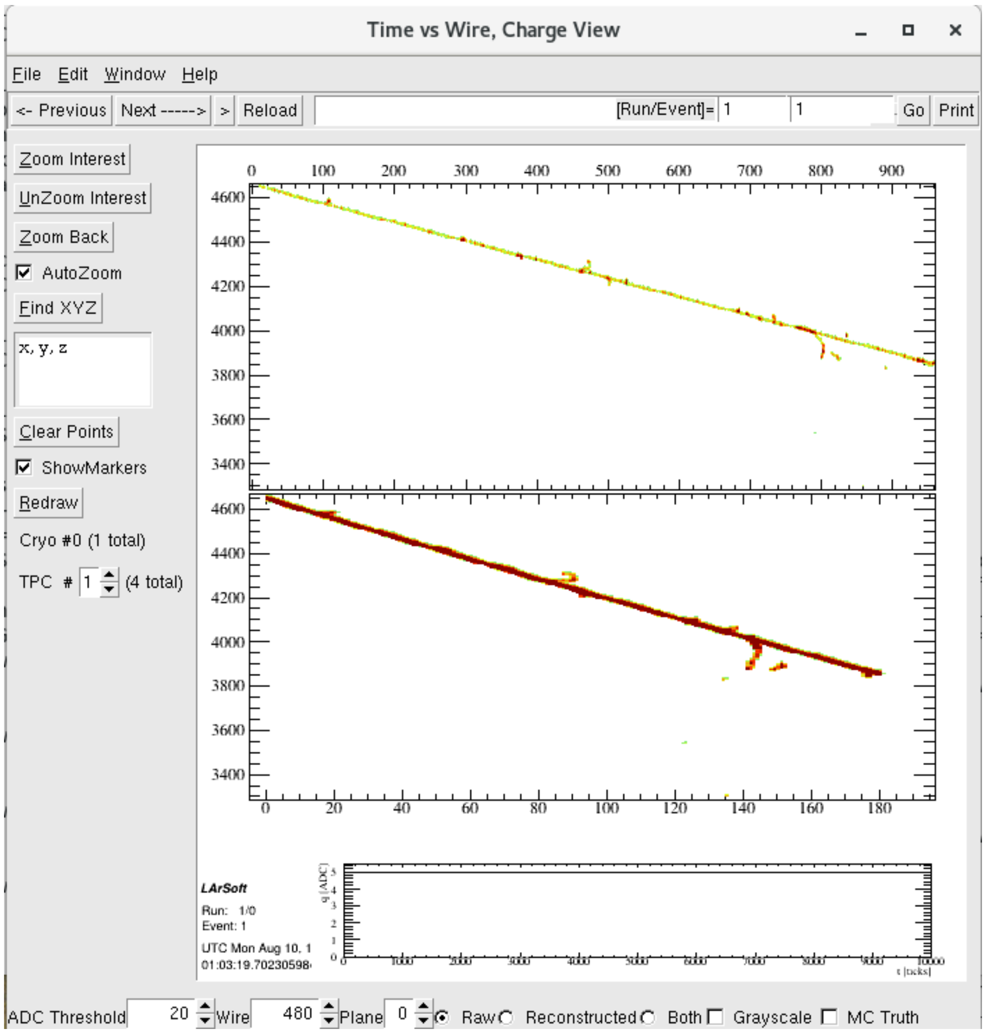
Tool to read in LEM transmission efficiency in dunetpc/fcl/protodunedp/common

LEM dead areas



Masked channels in this view due to LEM border
No charge is also collected in the other view for this region as it should be

Disabling LEM transmission efficiency




```
dunefddphase_crpgain: {
   service_provider: CrpGainService
   LogLevel:         1
   CrpDefaultGain:   6
   CrpNumLem:        36
   LemViewChans:     160
   LemEffTool:       ""  #lemEfficiency
}
```

Setting to empty string disables tool for LEM transmission efficiency retrieval

Conclusions

- New service for retrieving / simulating CRP effective gain
  - Incorporates effects of LEM dead areas
  - Provides unified interface for charge / light yield simulation

- Should allow to proceed with evaluation/tuning of tracking performance in ProtoDUNE-DP with more realistic Monte Carlo
  - Although the effects of drift field distortions due to malfunctioning HV feedthrough are not there

- Checking DP SignalShaping service to ensure correct normalization of the simulated collected charge when it is translated into ADCs