IDEA Drift Chamber simulation - Readout implementation
(Cluster Counting Algorithm)

Walaa Elmetenawee
for the working group

IDEA Physics and Software Meeting
Apr. 17, 2023

The cluster counting algorithm

The goal is to implement the cluster counting algorithm to the simulation of the drift chamber in the Geant 4 IDEA Full SIM framework.

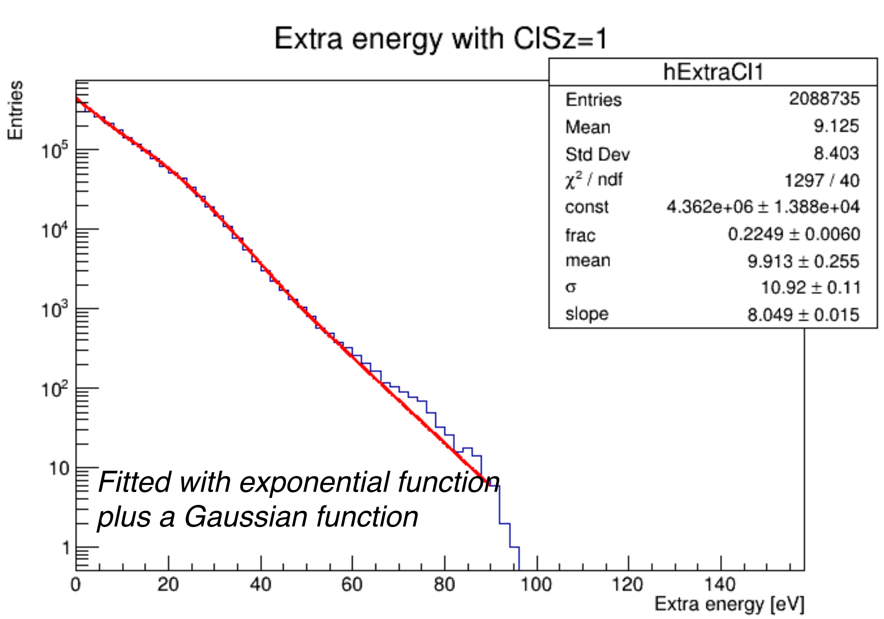
Drift Chamber simulation - Cluster Counting

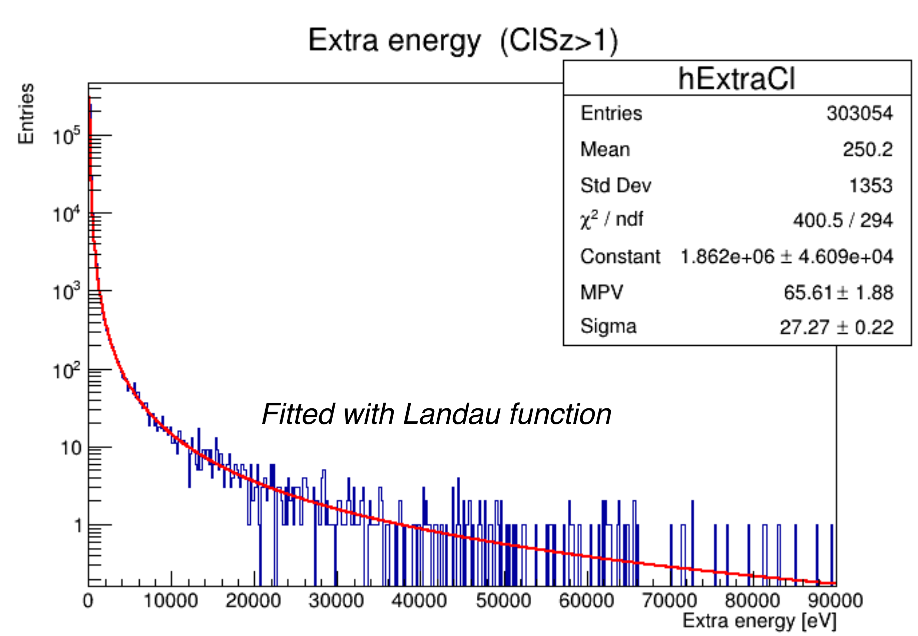
F.Cuna , G.F.Tassielli, F.Grancagnolo, N.De Filippis

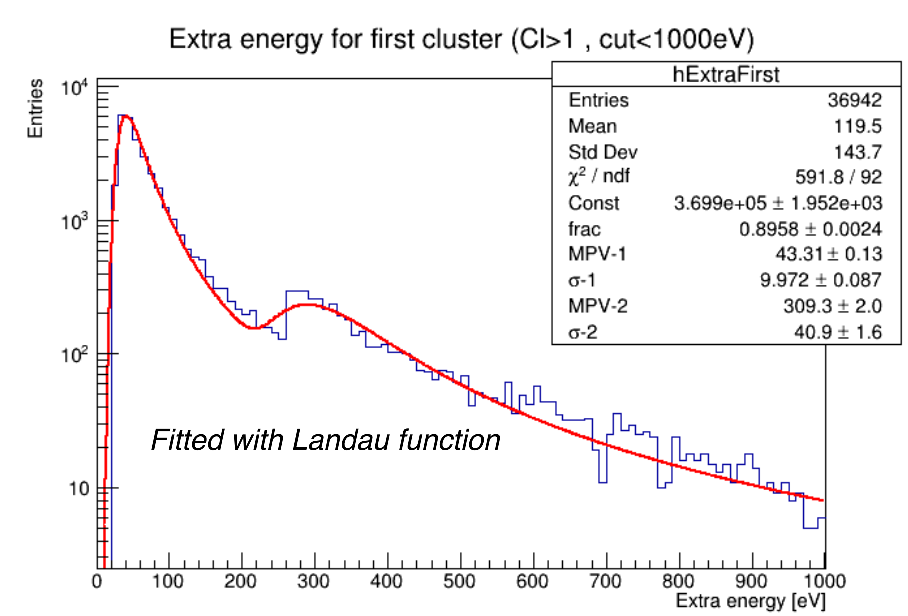
https://doi.org/10.48550/arXiv.2105.07064

The basic idea is to develop an algorithm which can use the energy deposit information provided by Geant4 to reproduce, in a fast and convenient way, the clusters number distribution and the cluster size distribution.

- **The algorithm implementation starts from Garfield++ simulations.**
- Firstly, we analyse the distribution of the kinetic energy for clusters that have a cluster size equal to 1(left), and clusters that have cluster size higher than 1 (middle) and the distribution of clusters with a cluster size higher than 1 up to a 1 keV, which is a cut equivalent to the single interactions range cut set by default in Geant4.







- Then we focused on the evaluation of the maximum kinetic energy spent to create clusters with cluster size higher than one. (maxExEcl).

Drift Chamber simulation - Cluster Counting

F.Cuna , G.F.Tassielli, F.Grancagnolo, N.De Filippis

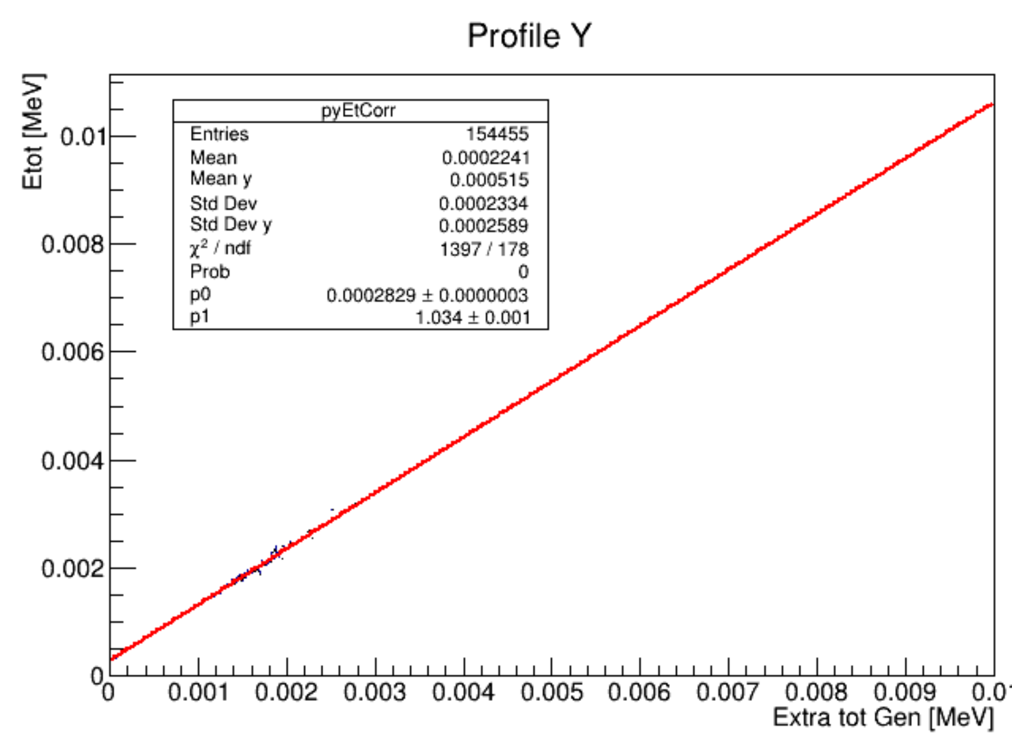
https://doi.org/10.48550/arXiv.2105.07064

The basic idea is to develop an algorithm which can use the energy deposit information provided by Geant4 to reproduce, in a fast and convenient way, the clusters number distribution and the cluster size distribution.

- To extract this parameter, named maxExEcl, we studied the correlation plot, between the total energy loss by particles traversing the gas mixture and the total kinetic energy of clusters with cluster size higher than 1; moreover we evaluated the parameter named ExSgm to take into account the smearing around the mean value of the total energy loss. The profile plot is fitted with a linear function and the formula for evaluating the maxExEcl is :

$$maxExEcl = \frac{E_{tot} - p0 + Random(Gaus(0, ExSgm))}{p1}$$

where p0 and p1 are the fit parameters of the linear fit and Etot is the total energy loss by the particles traversing the 200 cells of gas.



Drift Chamber simulation - Cluster Counting

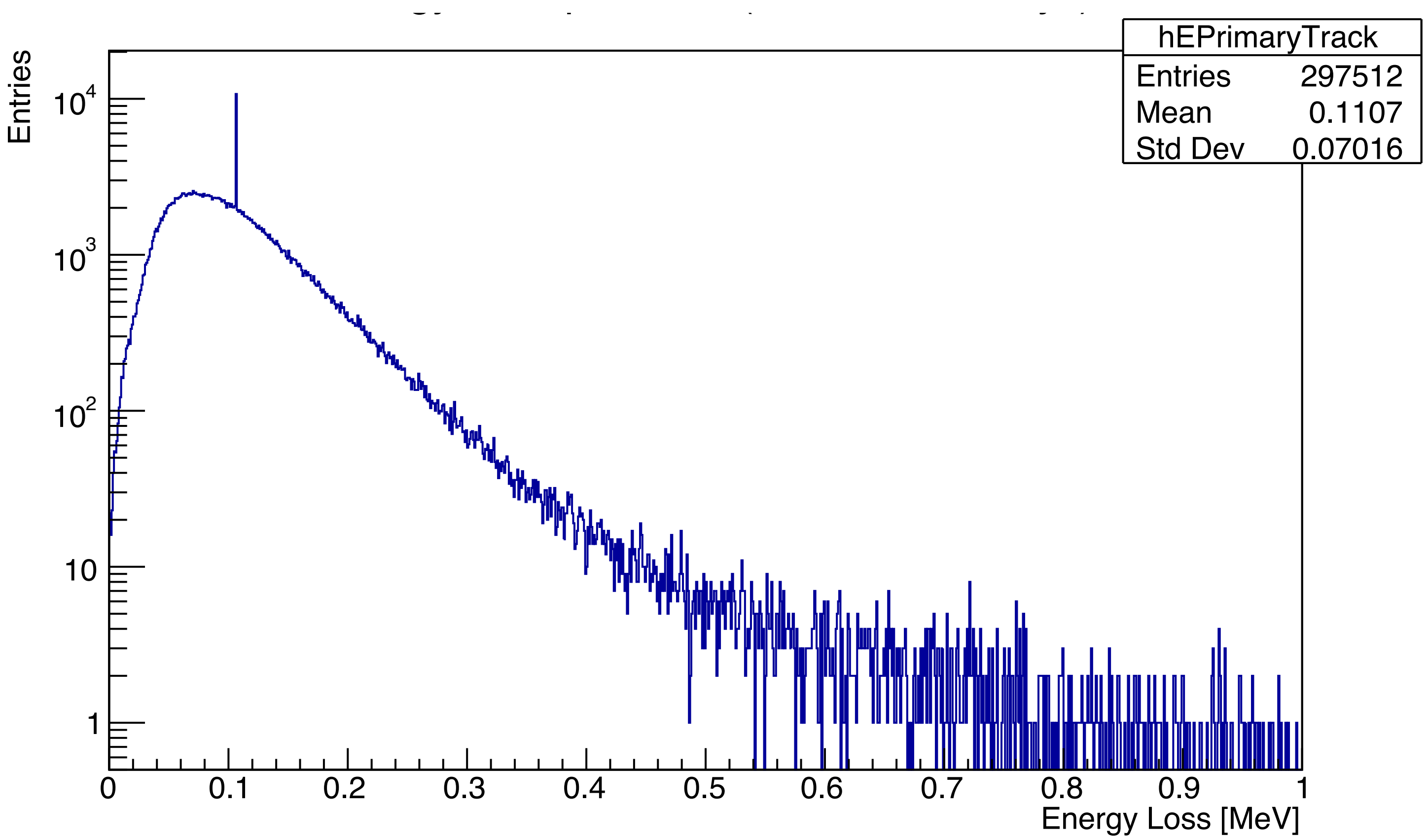
The Algorithm implementation:

- Using the results from the Garfield++ analysis described in the previous slides, we started to implement the algorithm in Geant 4 as following:
  - If maxExEcl is higher than zero, generates the kinetic energy for clusters with cluster size higher than one by using its distribution and evaluates the cluster size.
  - This procedure is repeated until the **sum of primary ionization energy and kinetic energy** per cluster **saturate the maxExEcl** of the event.
  - Then, using the remaining energy (Eloss-maxExEcl), the algorithm creates clusters with cluster size equal to one by assigning their kinetic energy according to the proper distribution.

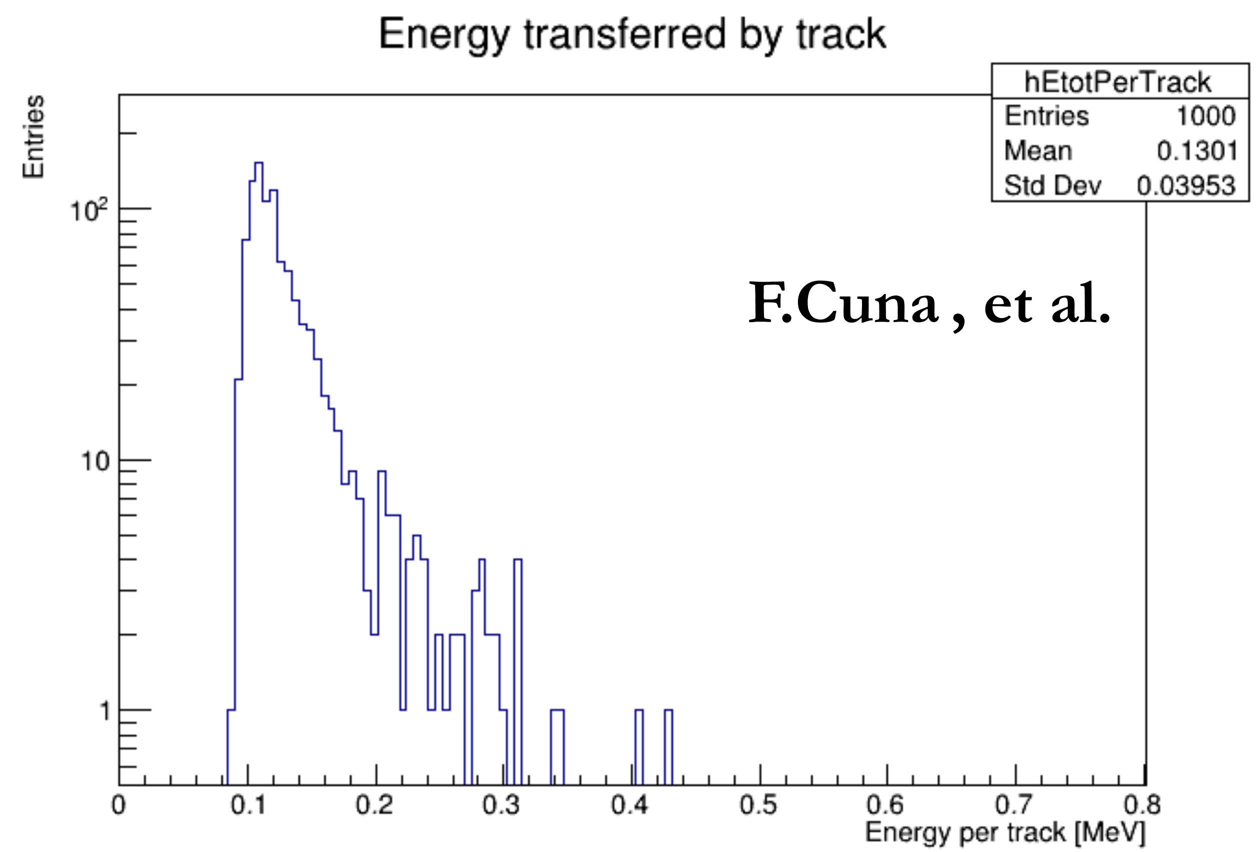
Drift Chamber simulation - Cluster Counting

- Energy loss distribution of a muon traversing 200 cells, 1 cm per side, filled with 10% He and 90 % iC_4H_{10},

- simulated by Geant4.



- simulated by Garfield++..

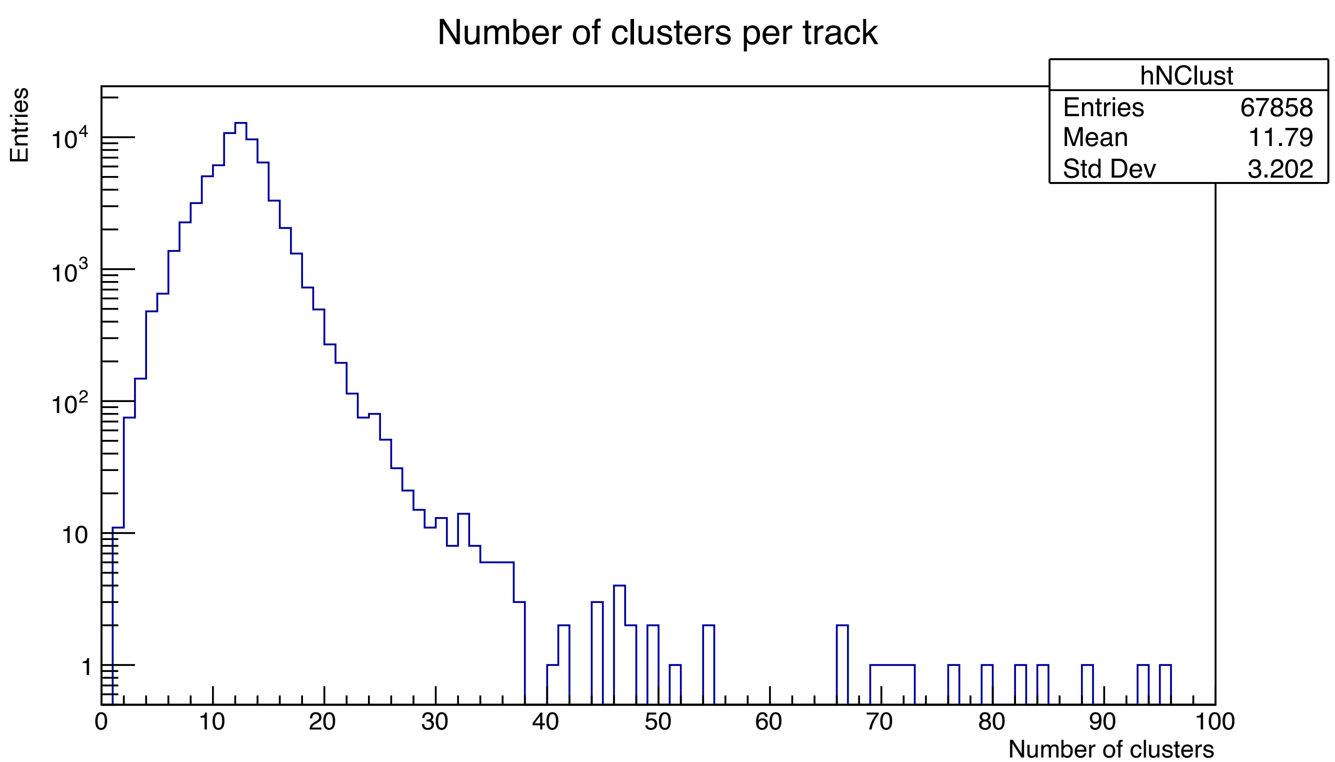


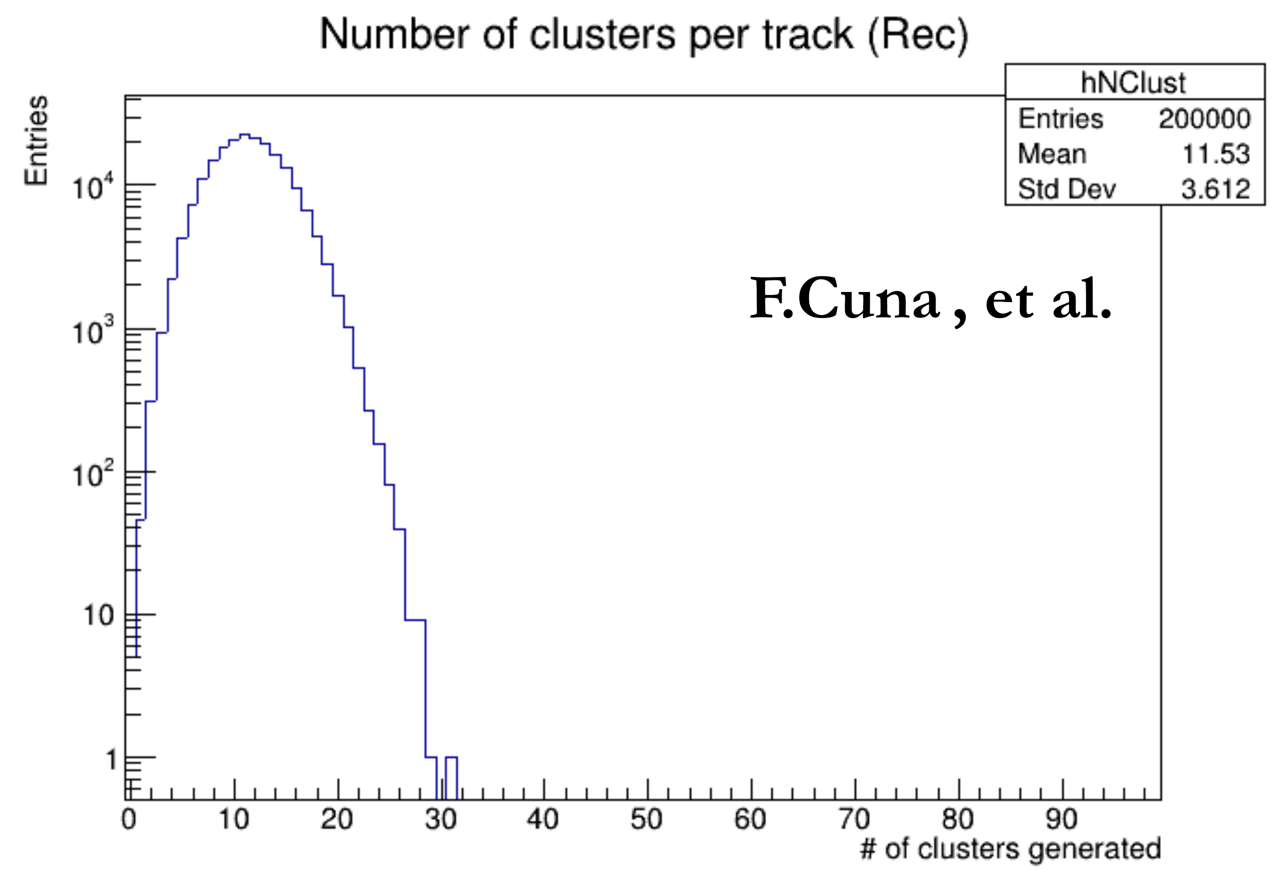
F.Cuna , et al.

https://doi.org/10.48550/arXiv.2105.07064

Drift Chamber simulation - Cluster Counting

- Clusters number distribution, for a muon at 300 MeV traversing 200 cells, 1 cm per side, filled with 10% He and 90 % iC_4H_{10},



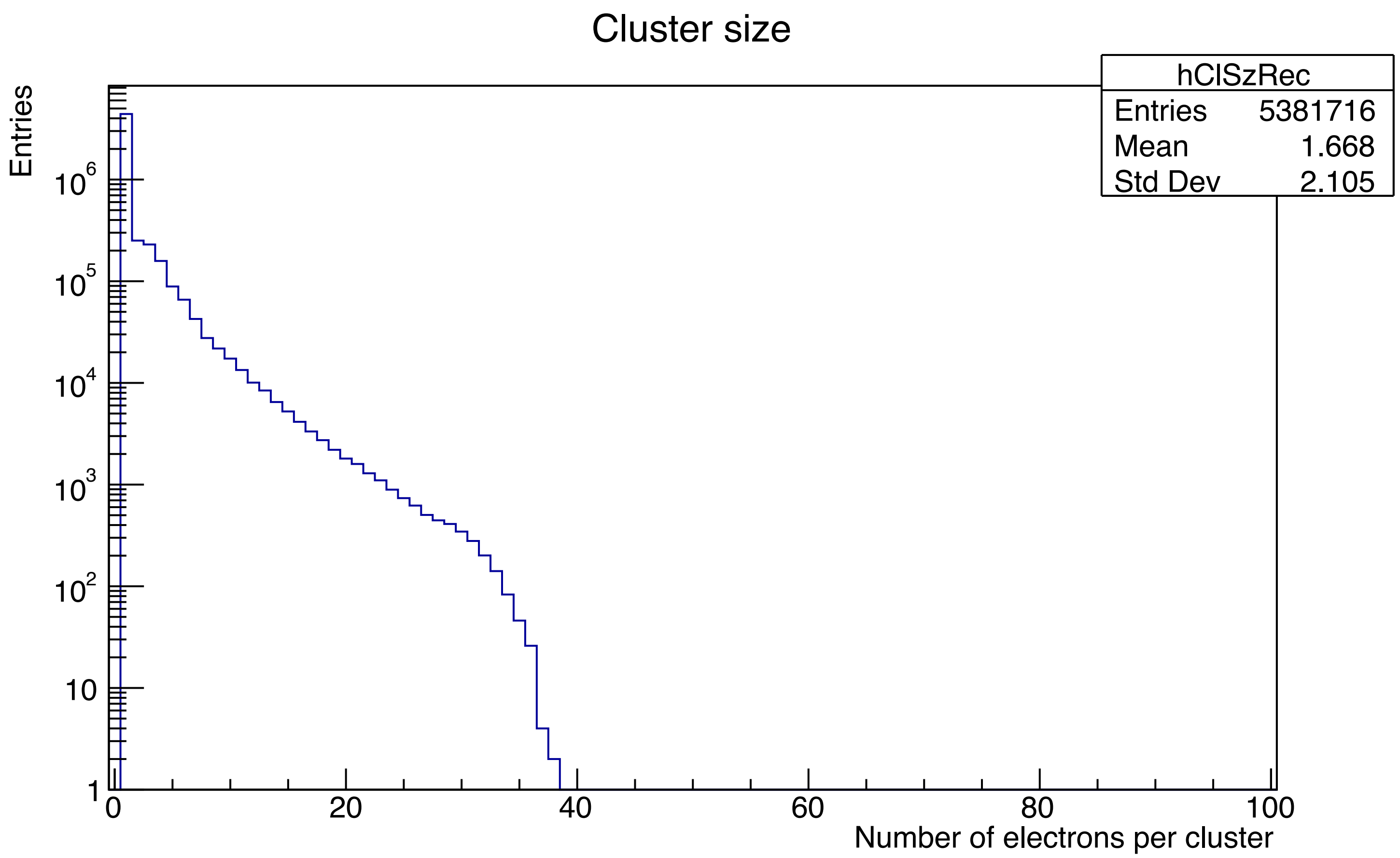


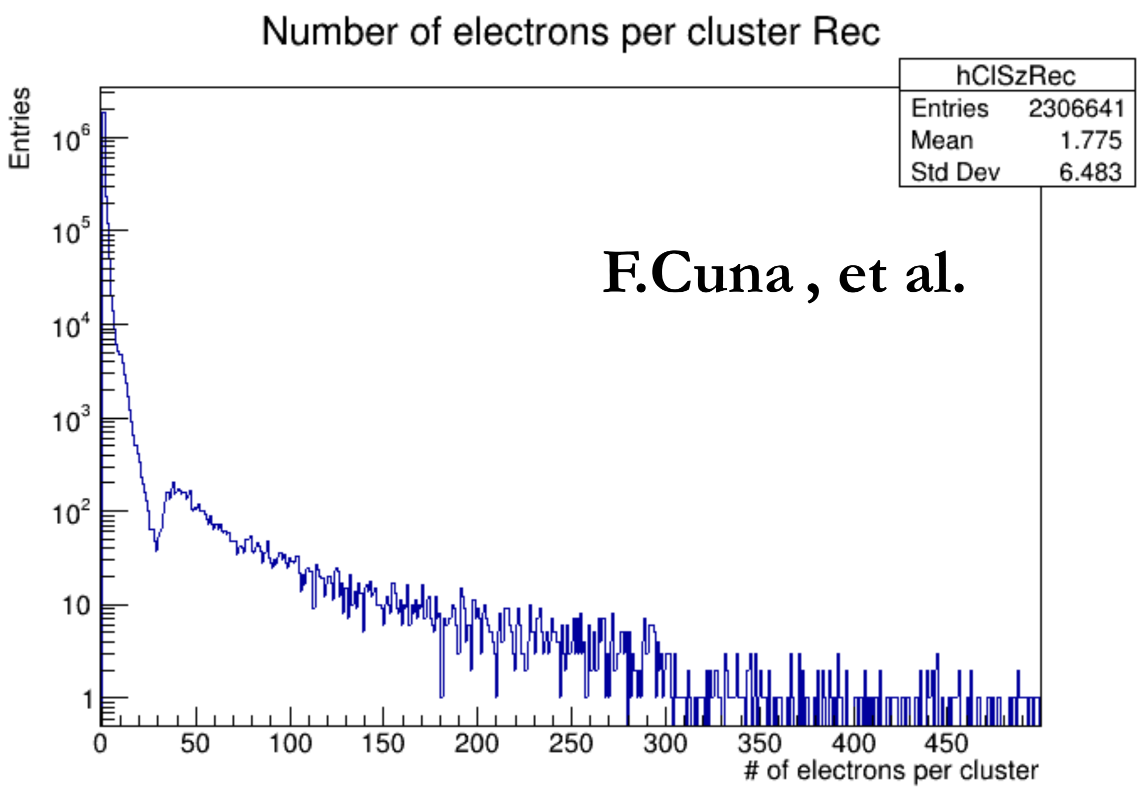
F.Cuna , et al.

https://doi.org/10.48550/arXiv.2105.07064

Drift Chamber simulation - Cluster Counting

- Cluster size distribution, for a muon at 300 MeV traversing 200 cells, 1 cm per side, filled with 10% He and 90 % iC_4H_{10},





F.Cuna , et al.

https://doi.org/10.48550/arXiv.2105.07064

Backup

Detector Segmentation

- Detector Segmentation:

Code: CDCHGasLayerSD_Square.cc

- **The informations needed to be stored from this task:**

```
fHitsCollection->insert( new GMCG4TrackerHit(aStep->GetTrack()->GetTrackID(),
                                             det,
                                             edep,
                                             aStep->GetNonIonizingEnergyDeposit(),
                                             aStep->GetPreStepPoint()->GetGlobalTime(),
                                             aStep->GetPreStepPoint()->GetProperTime(),
                                             prePosTracker,
                                             postPosTracker,
                                             preMomWorld,
                                             step,
                                             aStep->GetPostStepPoint()->GetProcessDefinedStep()->GetProcessName()
                                             ));
```

Detector Hits

- Detector Hits:: Each G4Step has to be 'processed' to keep relevant information

Code: GMCTReadMCDataCDCH.cpp

```
    for( size_t i=0; i<ihits.size(); i++ ) {

      GMCDCGeantStep& hit = *(ihits.at(i));

      hitPerTrk_it = hitPerTrk.find(hit.GetfTrackID());
      hitEndPerTrk_it = hitEndPerTrk.find(hit.GetfTrackID());

      if ( hitPerTrk_it == hitPerTrk.end() ) {
        gAnalyzer->SetDCHHitSize(NrHits+1);
        GMCDCHHit *ahit = gAnalyzer->GetDCHHitAt(NrHits);
        ++NrHits;
        hitPerTrk.insert( std::pair<int,GMCDCHHit *>( hit.GetfTrackID(), ahit ) );
        hitEndPerTrk.insert( std::pair<int,TVector3 *>( hit.GetfTrackID(), hit.GetfPosEnding() ) );

        ahit->SetfCellId( dcell_id );
        ahit->SetfTrkIDofHit( hit.GetfTrackID() );
//        ahit->SetfxPCA(fCApoint.X());
//        ahit->SetfyPCA(fCApoint.Y());
//        ahit->SetfzPCA(fCApoint.Z());
        ahit->SetfEntranceX( hit.GetfPos()->X() );
        ahit->SetfEntranceY( hit.GetfPos()->Y() );
        ahit->SetfEntranceZ( hit.GetfPos()->Z() );
        ahit->SetfEntranceMomX( hit.GetfMomentum()->X() );
        ahit->SetfEntranceMomY( hit.GetfMomentum()->Y() );
        ahit->SetfEntranceMomZ( hit.GetfMomentum()->Z() );
//        ahit->SetfImpact(fDCA);
        ahit->SetfTotalEnergyLoss( hit.GetfEdep() );
            ahit->SetfGlobalTime( hit.GetfGlobalTime() );
            ahit->SetfToF( hit.GetfProperTime() );
        ahit->SetfLength( hit.GetfStepLen() );

      } else {
        GMCDCHHit *ahit = hitPerTrk_it->second;
        ahit->SetfTotalEnergyLoss( ahit->GetfTotalEnergyLoss() + hit.GetfEdep() );
        ahit->SetfLength( ahit->GetfLength() + hit.GetfStepLen() );
        hitEndPerTrk_it->second=hit.GetfPos();
      }

    }
```