MCTrack Data Product Modification: Adding Calorimetry

Joseph Zennamo
for MicroBooNE

LArSoft Coordination Meeting
Jan. 19th, 2016

THE UNIVERSITY OF CHICAGO

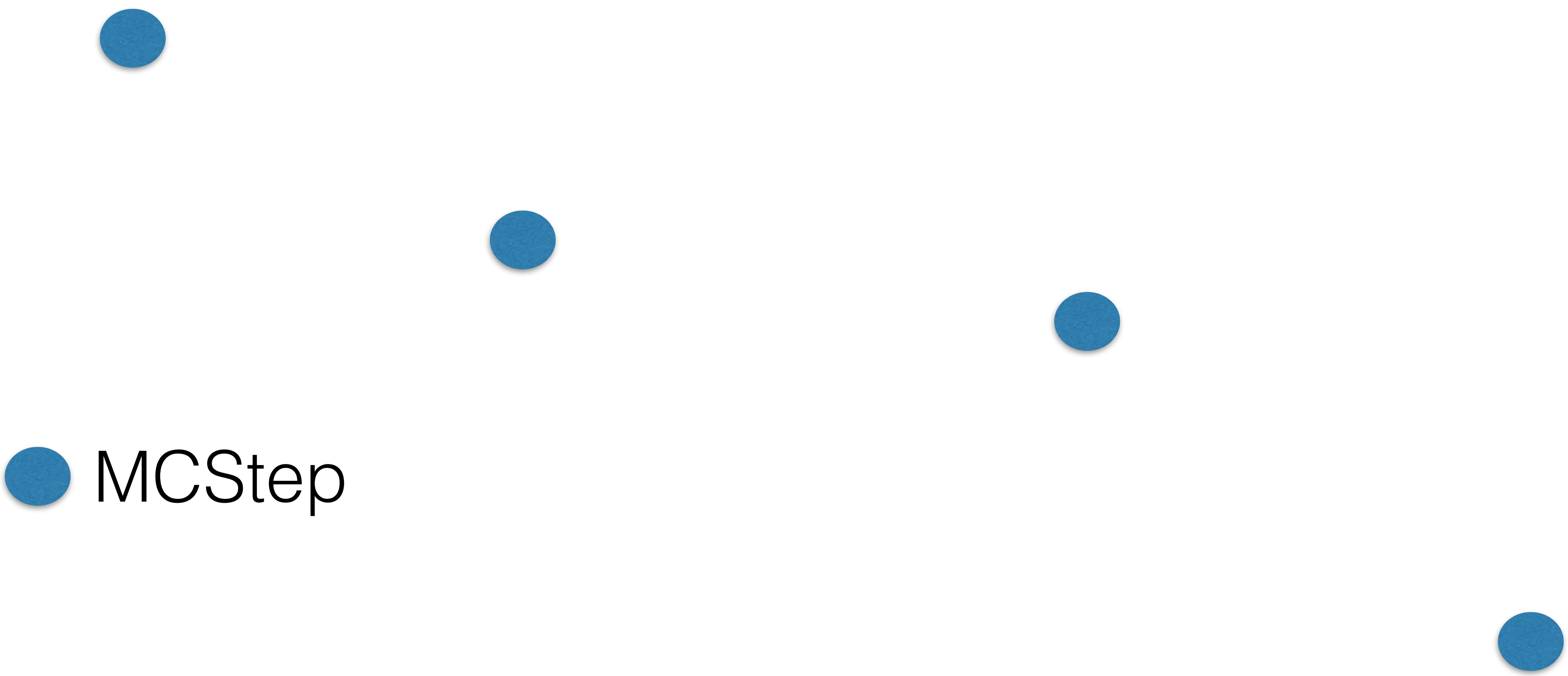
The Want

- Currently MCTrack contains no information about the ionization energy loss that one would use for doing particle identification

- The purpose of this work was to compute and store a step-by-step ionization energy loss

- This could have been done more efficiently if MCTracks were built at LArG4 level (a story for another time)

MCTrack

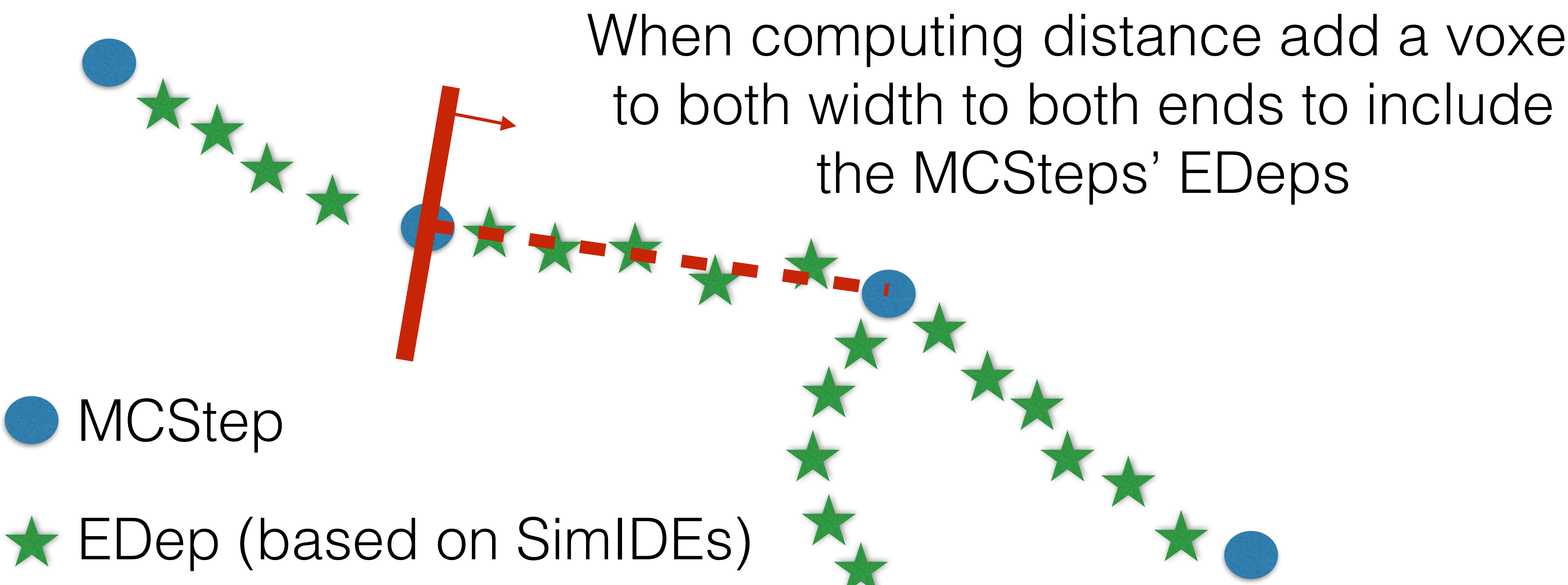


MCTrack + EDeps

The Solution

- For each pair of MCSteps we want to calculate the ionization energy loss

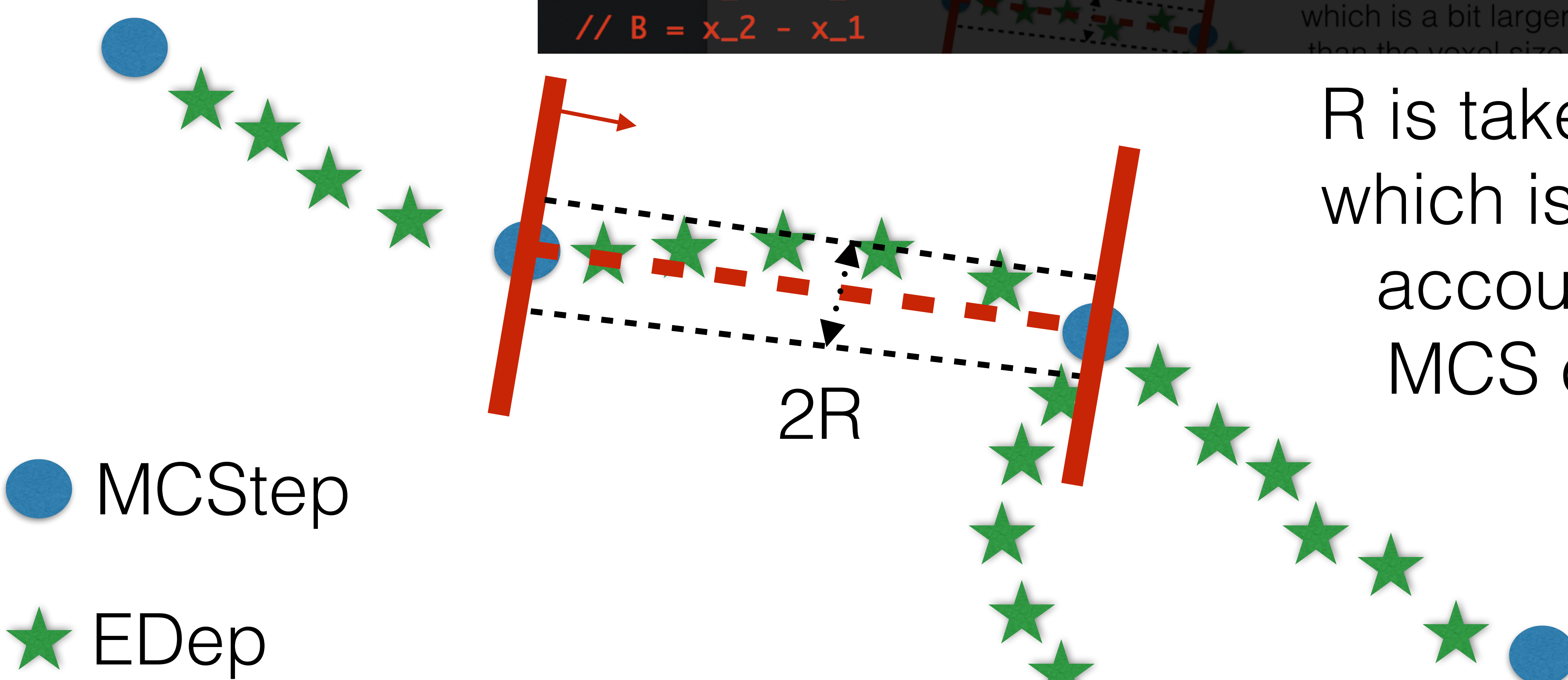
```
//Make a plane at the step pointed at the next step

//Need to define a plane through the first MCStep with a normal along the momentum vector of the step
//The plane will be defined in the typical way:
// a*x + b*y + c*z + d = 0
// where, a = dir_x, b = dir_y, c = dir_z, d = - (a*x_0+b*y_0+c*z_0)
// then the *signed* distance of any point (x_1, y_1, z_1) from this plane is:
// D = (a*x_1 + b*y_1 + c*z_1 + d )/sqrt( pow(a,2) + pow(b,2) + pow(c,2))
```



The Solution

- This keeps us from picking up energy associated with other (non-ionization) forms of energy loss

```
//Make a line connecting the two points and find the distance from that line
//
//                                      [A dot B]^2     [A dot B]^2
// distance^2 = A^2 - 2* ____________ + ______________
//                                          B^2              B^2
// Test point == x_0
// First Step == x_1
// Next step == x_2
// A = x_1 - x_0
// B = x_2 - x_1
```

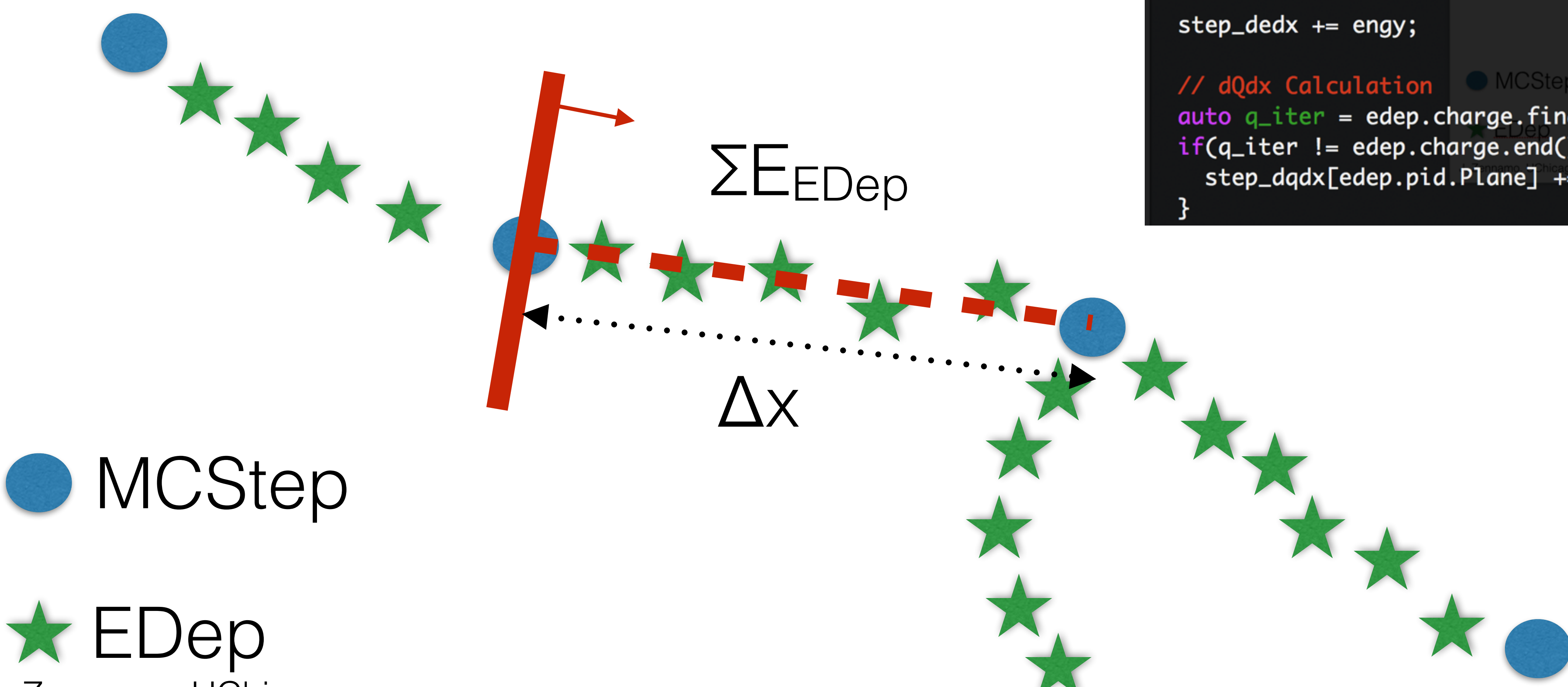


R is taken as 0.1cm which is to take into account GEANT MCS correction

The Solution

- Next we sum the energy and charge associated with all of the EDeps and measure the 3D distance between the two MCSteps

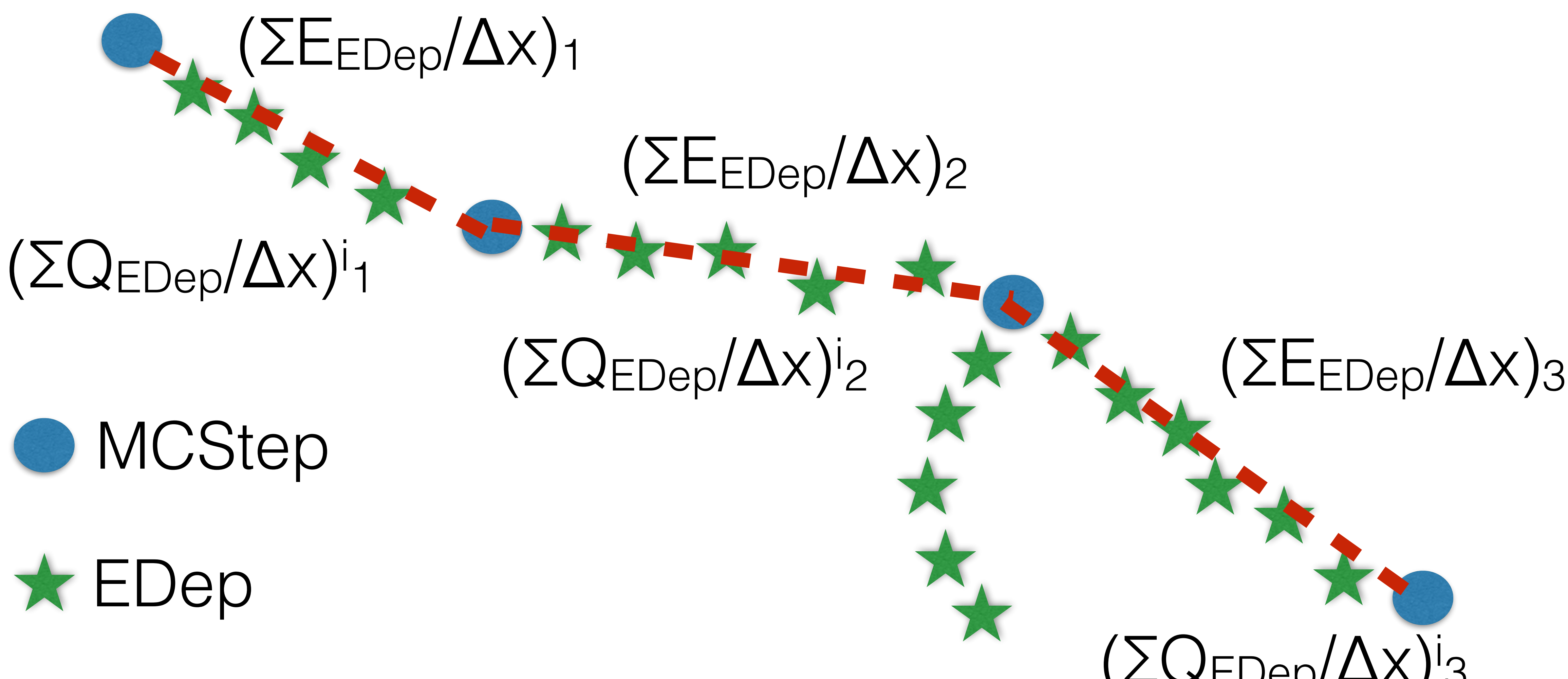
```
//dEdx Calculation
int npid = 0;
double engy = 0;

for(auto const& pid_energy : edep.energy){
  engy += pid_energy.second;
  npid++;
}

if(npid != 0){
  engy /= npid;}
else{engy = 0;}

step_dedx += engy;

// dQdx Calculation
auto q_iter = edep.charge.find(edep.pid);
if(q_iter != edep.charge.end()){
  step_dqdx[edep.pid.Plane] += (double)((*q_iter).second);
}
```



The Solution

- The result is a dEdx calculations associated with each MCStep pair and a dQdx calculation per plane with each MCStep pair
  - dEdx is the true ionization energy loss of the particle
  - dQdx is the ionization energy seen on the wires

```
std::vector<std::vector<double> > fdQdx; //< the G4 electron yeild per plane between each step // [N Plane][MCSteps - 1]
std::vector<double> fdEdx;//< the G4 "ionization" energy loss between each step // [MCSteps - 1]
```



The Implementation

- The default implementation of MCTrack contains no iterations through EDeps, had to add an iteration through all the MCSteps with an imbedded loop through EDeps
    - Take advantage of the fact that MCSteps are ordered
    - EDeps are not ordered, so currently iterate through full list for (N-1) MCSteps

- This implementation on a feature-branch “zennamo_MCTrackdEdx” for LArData, LArSim, and UBooNECode, and “newMCTrack_calo” in LArLite

Slow Down?

- To gauge the effect of introducing these additional iterations I ran over same 5 Corsika “Art events” (6.4ms exposure) with a CMC model (**extra activity!**)

“Vanilla” LArSoft (v04_33_00)

Total Time: 740 seconds (Average time 148 seconds)

New Implementation

Total Time: 806 seconds (Average time 161 seconds)

Slows down by only ~9%

Validation

- Comparison of the dE/dx and dQ/dx vs. residual range can be found in back-up, things agree as expected

- When we look at the MIP average energy deposit for MCSteps “far from the end of the track” we find that it is ~1.955 MeV/cm which is lower than my expected 2.2 MeV/cm

Conclusions

- This implementation is a good first attempt to characterize the ionization energy loss and provides results that approach my expectations
- I request that these be merged into the next LArSoft release so that MicroBooNE can use this in its next large production run
- Used this opportunity to clean up some aspects of MCSTReco module
  - Errant cout's in MCShower (that I left…)
  - Killed off zero length MCTracks (they annoyed me…)
  - etc.

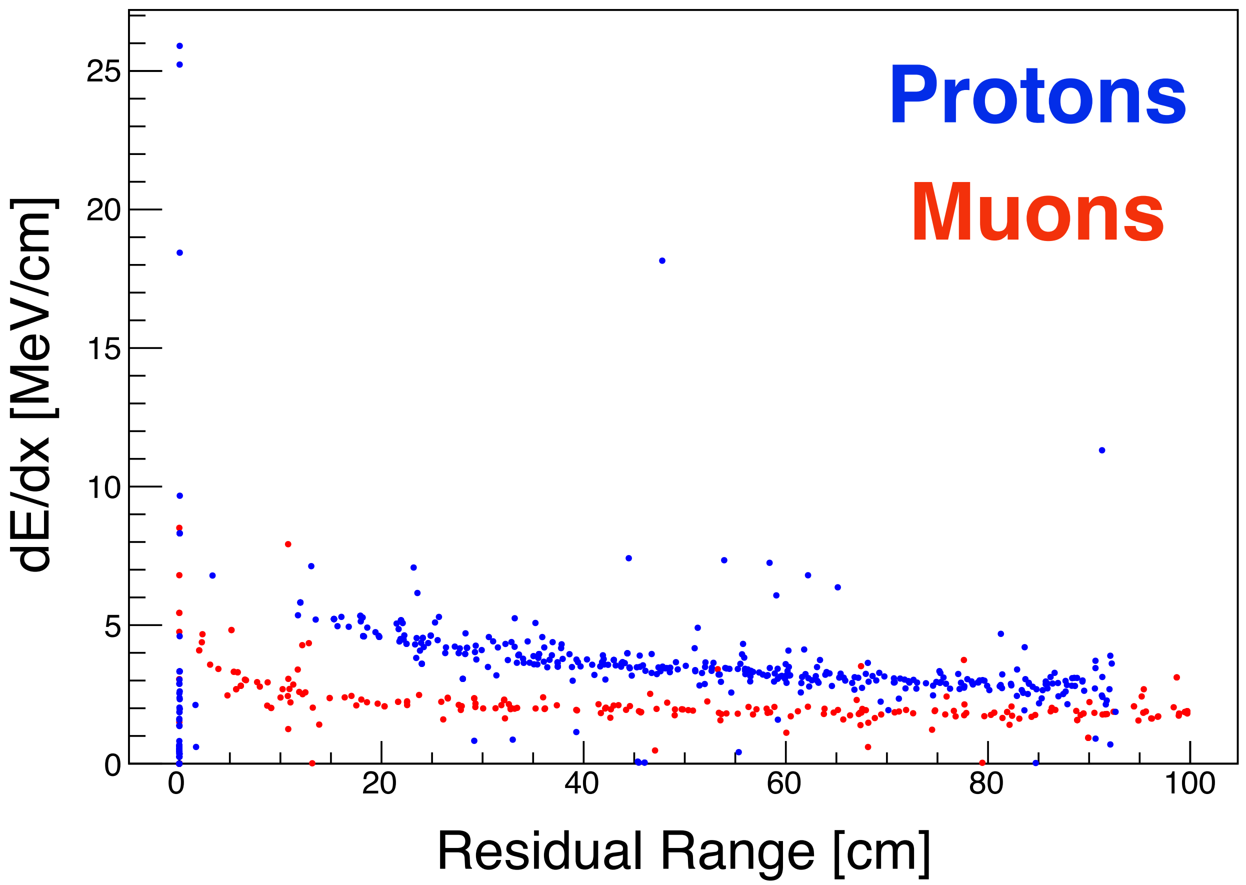
Future Studies

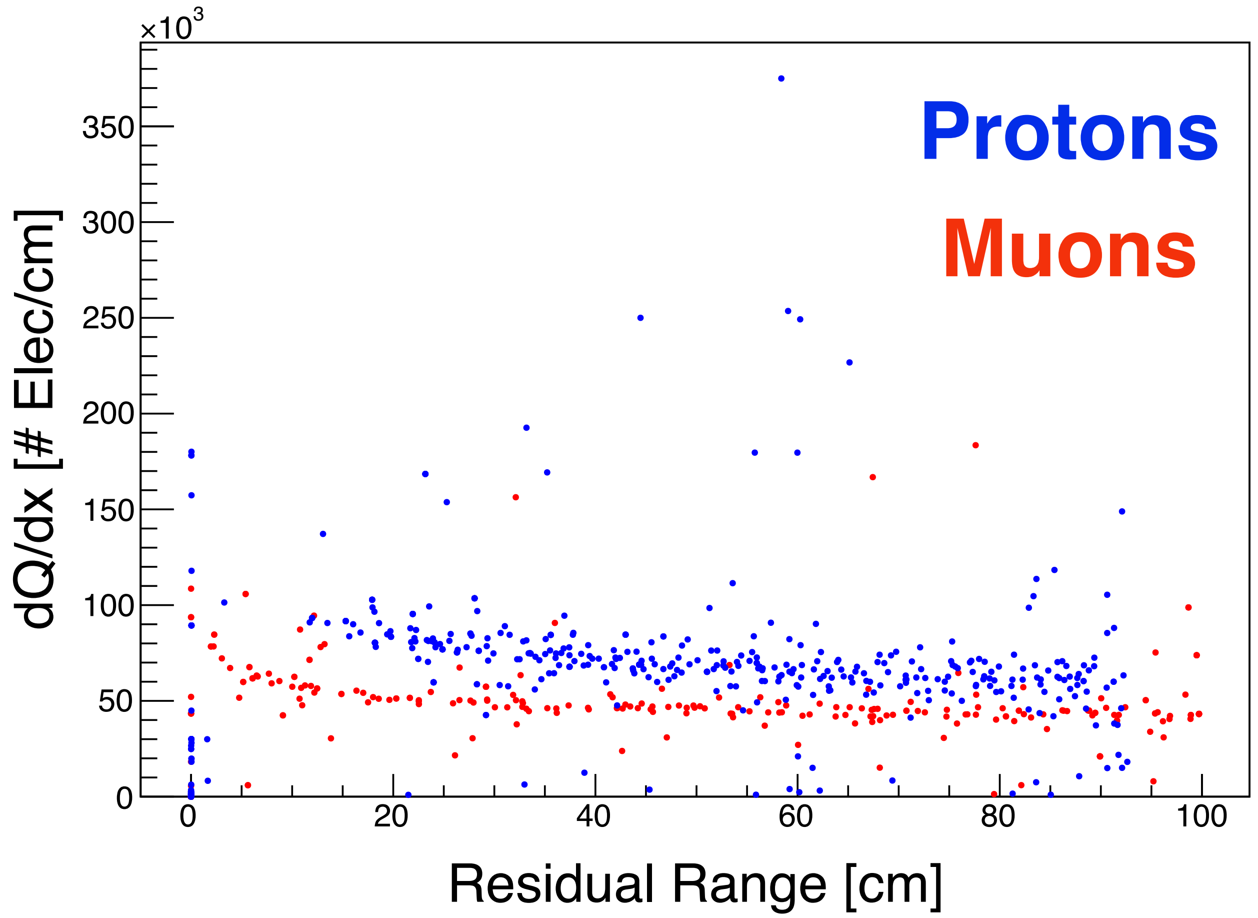
- Could improve this implementation:
  - Currently disregarding MCStep pairs that are closer than voxel size apart, can do better
  - Notice that the last pair of MCSteps is ALWAYS zero, not sure why
  - Results in a negligible decrease in MIP energy loss (1%)

- Could compute at LArG4 and wouldn't have to guess about ionization energy loss anymore with silly geometry functions…

Backups and Validation

The Checks

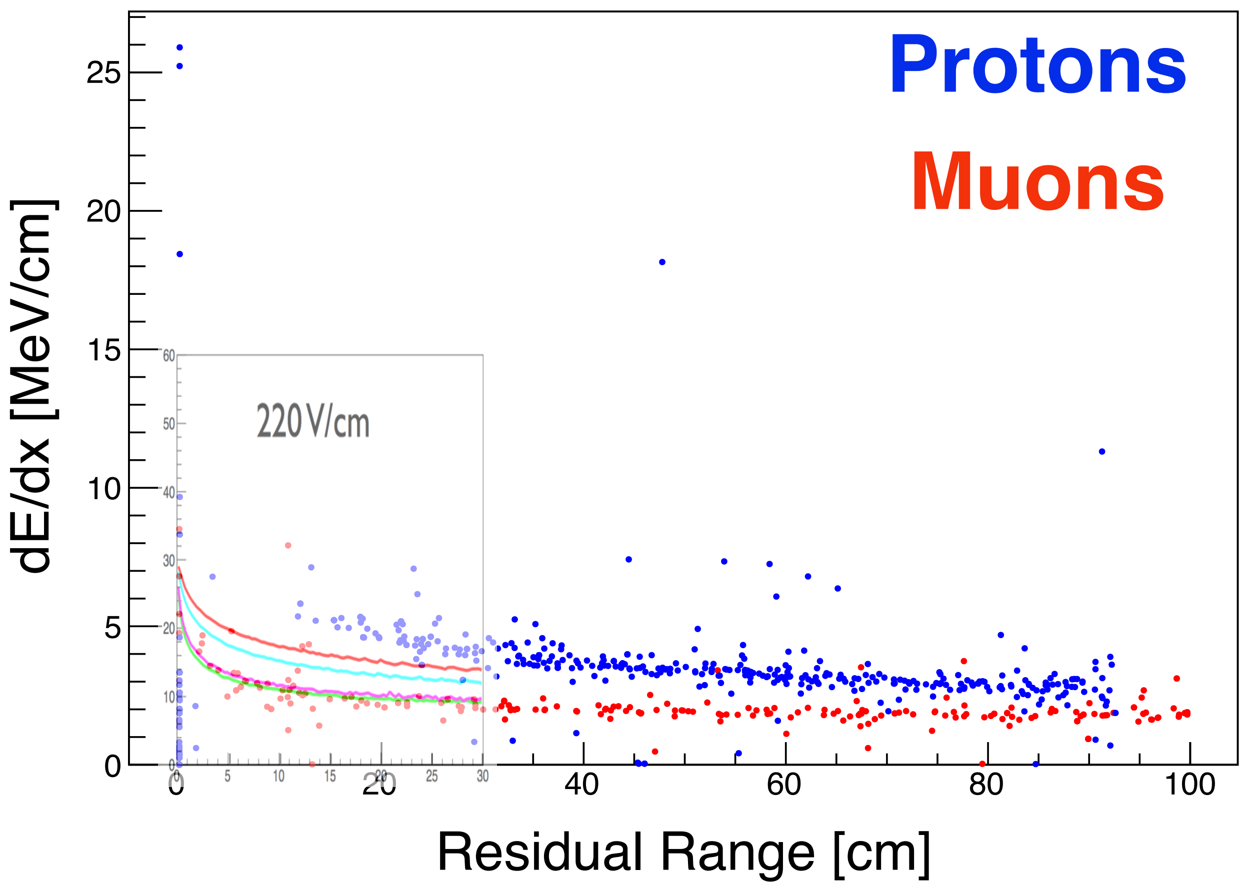


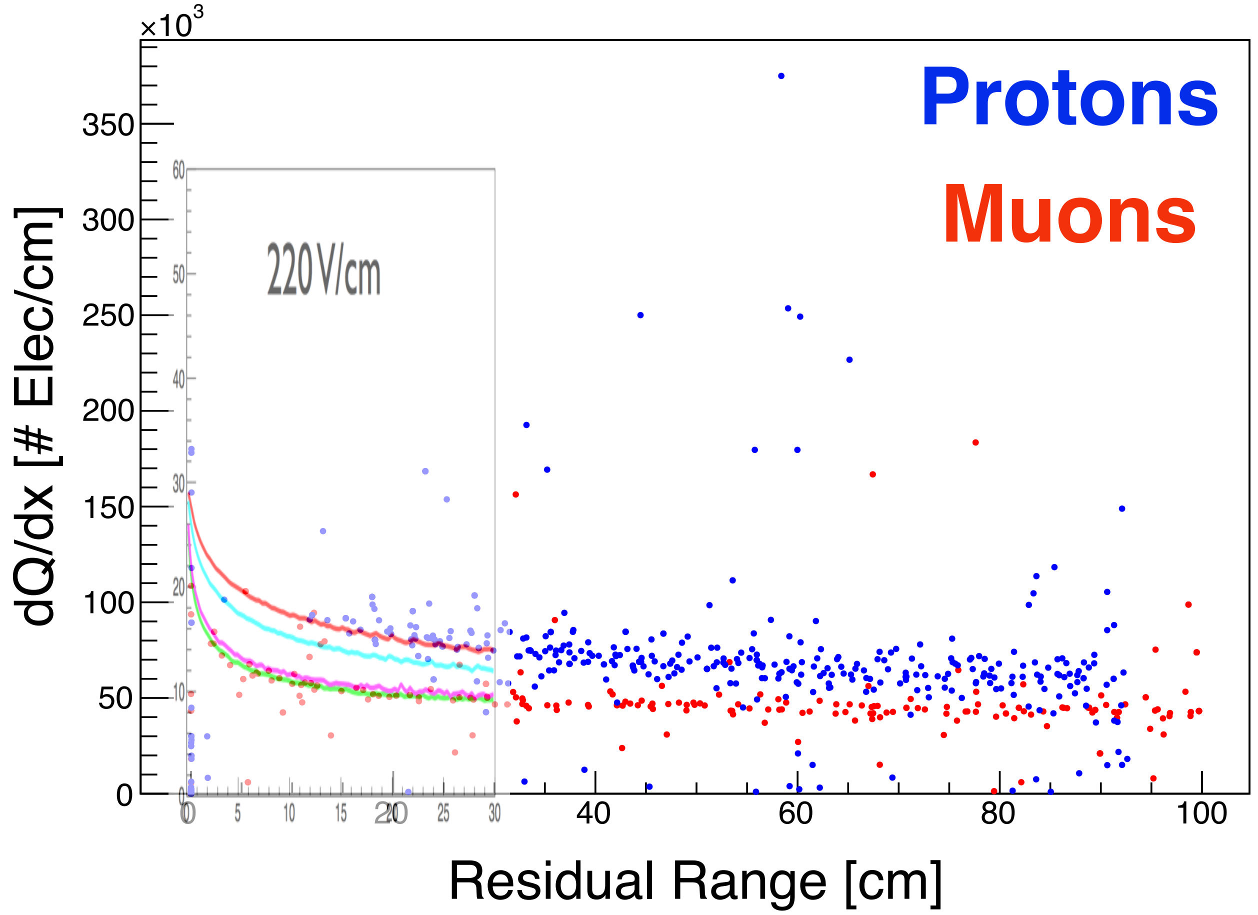


- We would expect to see a turn up and low values of residual range as we move out of the MIP regime
- We can compare this to our expectations

The Checks

From Ornella's DocDB 4672





- Looking at the expectation from GEANT we can see disagreement in dEdx (as expected, no recombination applied) but good agreement with dQdx

Less Pretty Plot!