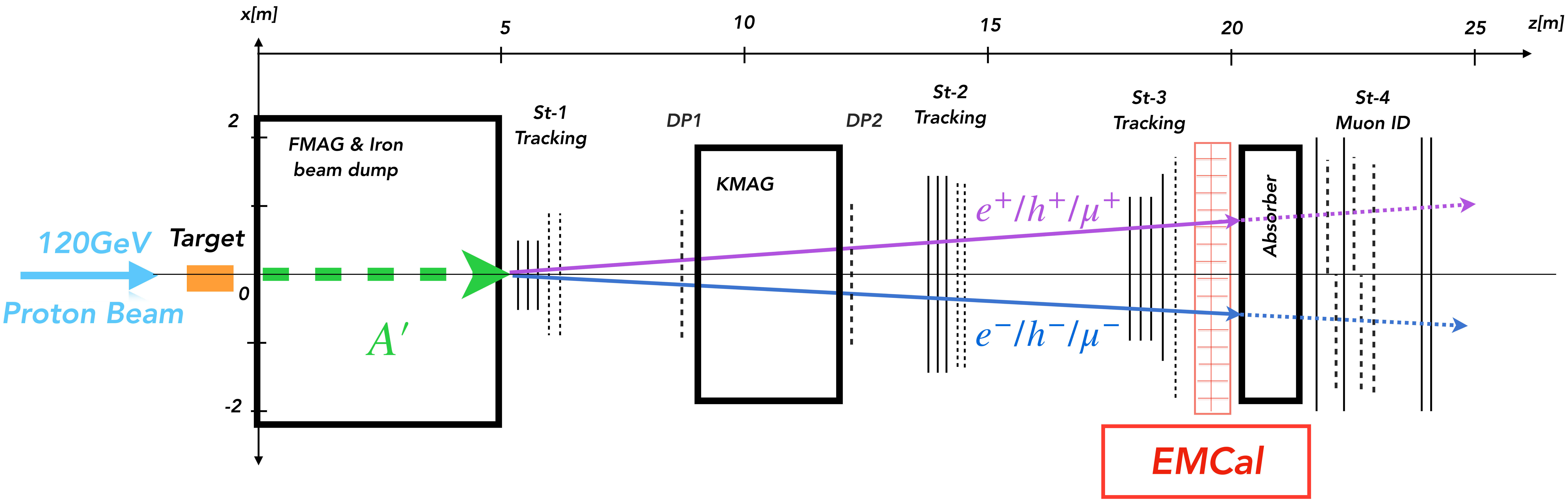
(EMCal) Simulations

Yongbin Feng (Fermilab)

DarkQuest Workshop, Boston, MA

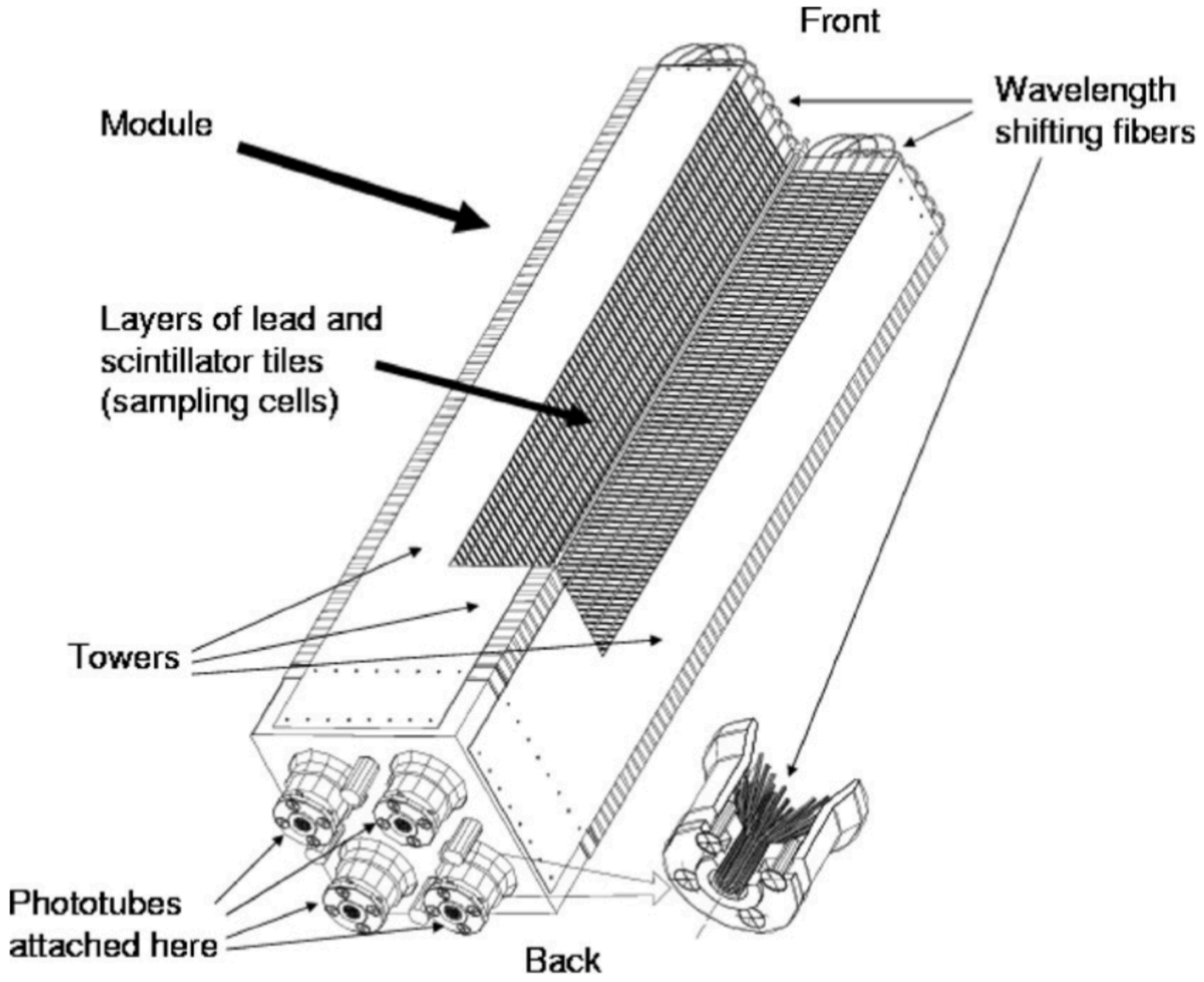
10/20/2023

Schematic



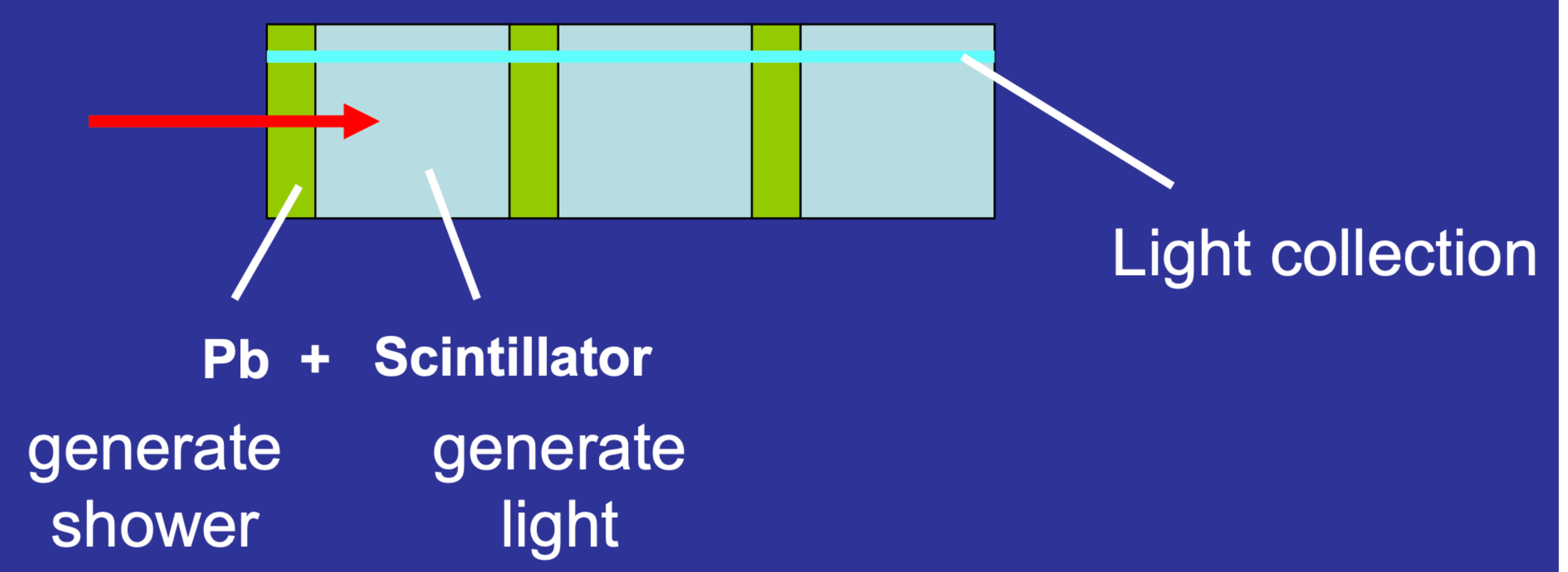
- Simulation studies for adding the Electromagnetic calorimeter (EMCal) (from PHENIX Experiment, 2mx4m)
  - Enable us access to electron final states, broaden the coverage to lower masses: $m_{A'} < 2m_\mu$
  - Used for trigger and also offline analysis
  - Provide more sensitivity by rejecting backgrounds

EMCal

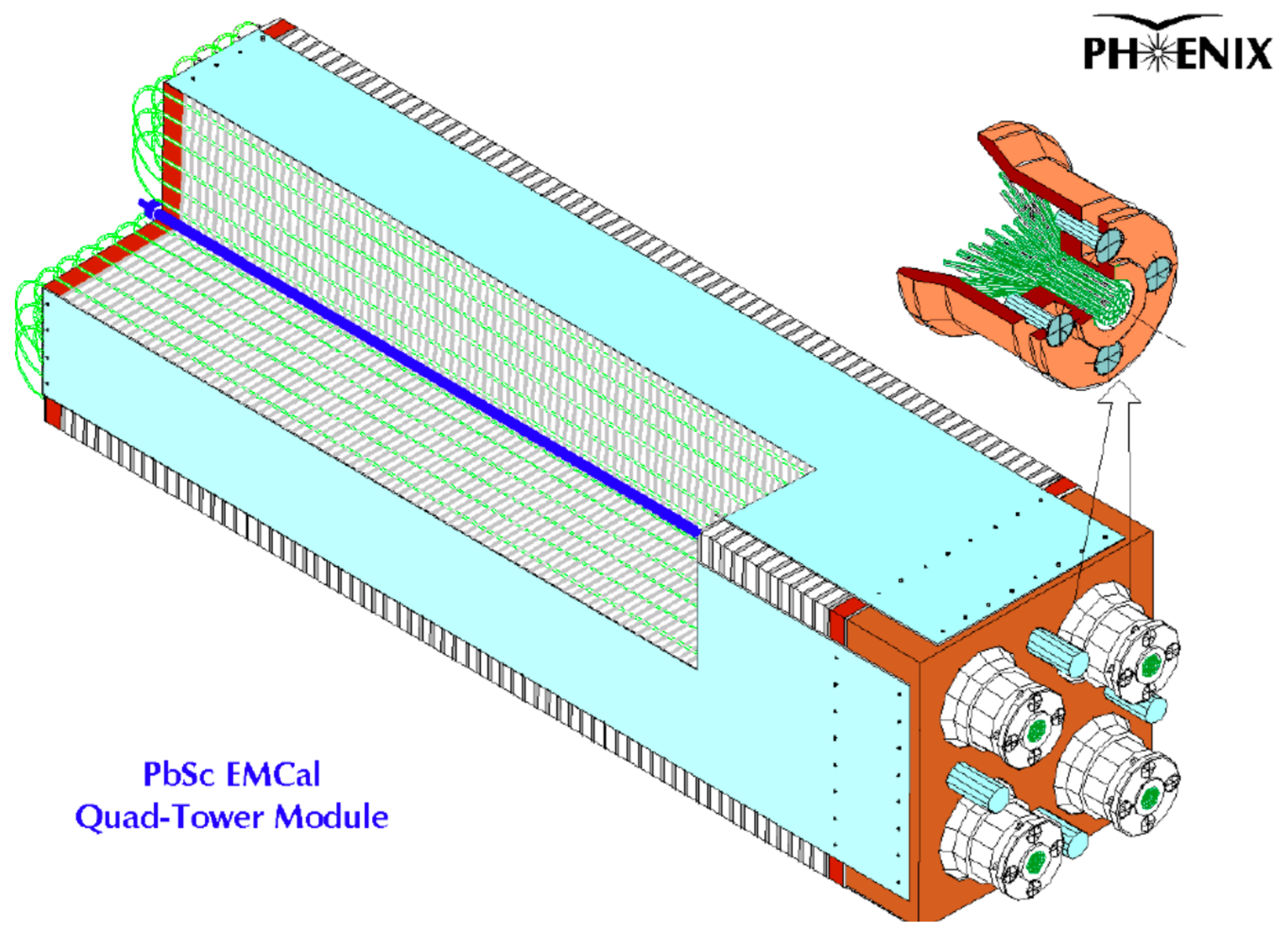


- EMCal from the PHENIX Experiment available at BNL (a 2m x 4m Pb-scintillator calorimeter).
- References on some documents from PHENIX (here)
- One EMCal Cell size is about 5.535cm x 5.535cm x 37.5cm. 36x72 towers. Moliere radius 3cm. Nuclear interaction length is 0.85cm
  - PHENIX EMCall also has Lead + Glass Cherenkov Radiator (4cm x 4 cm x 40cm). Moliere radius 3.7cm
- Sampling calorimeter, with a sampling fraction of around 10%

Performance from PHENIX

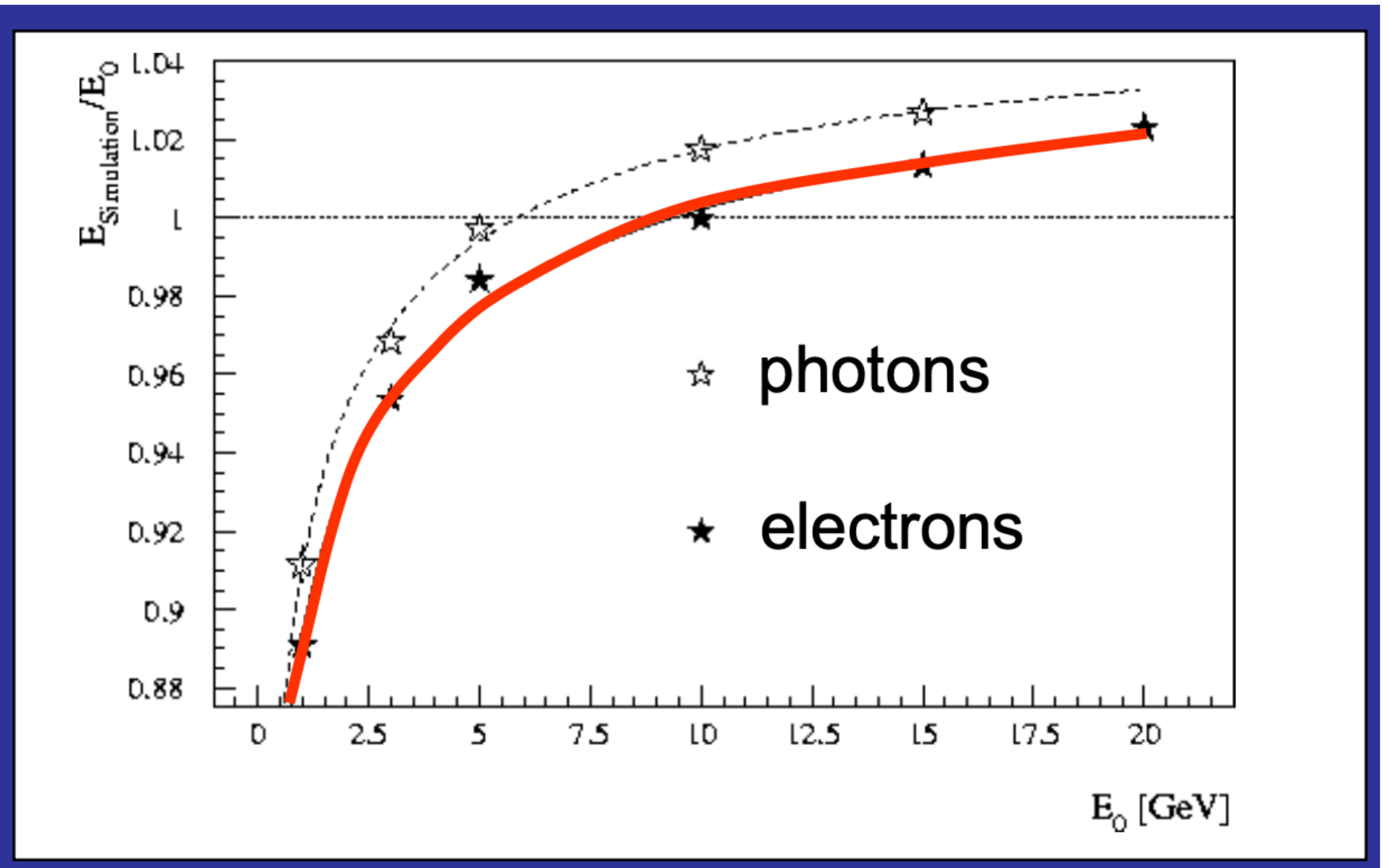


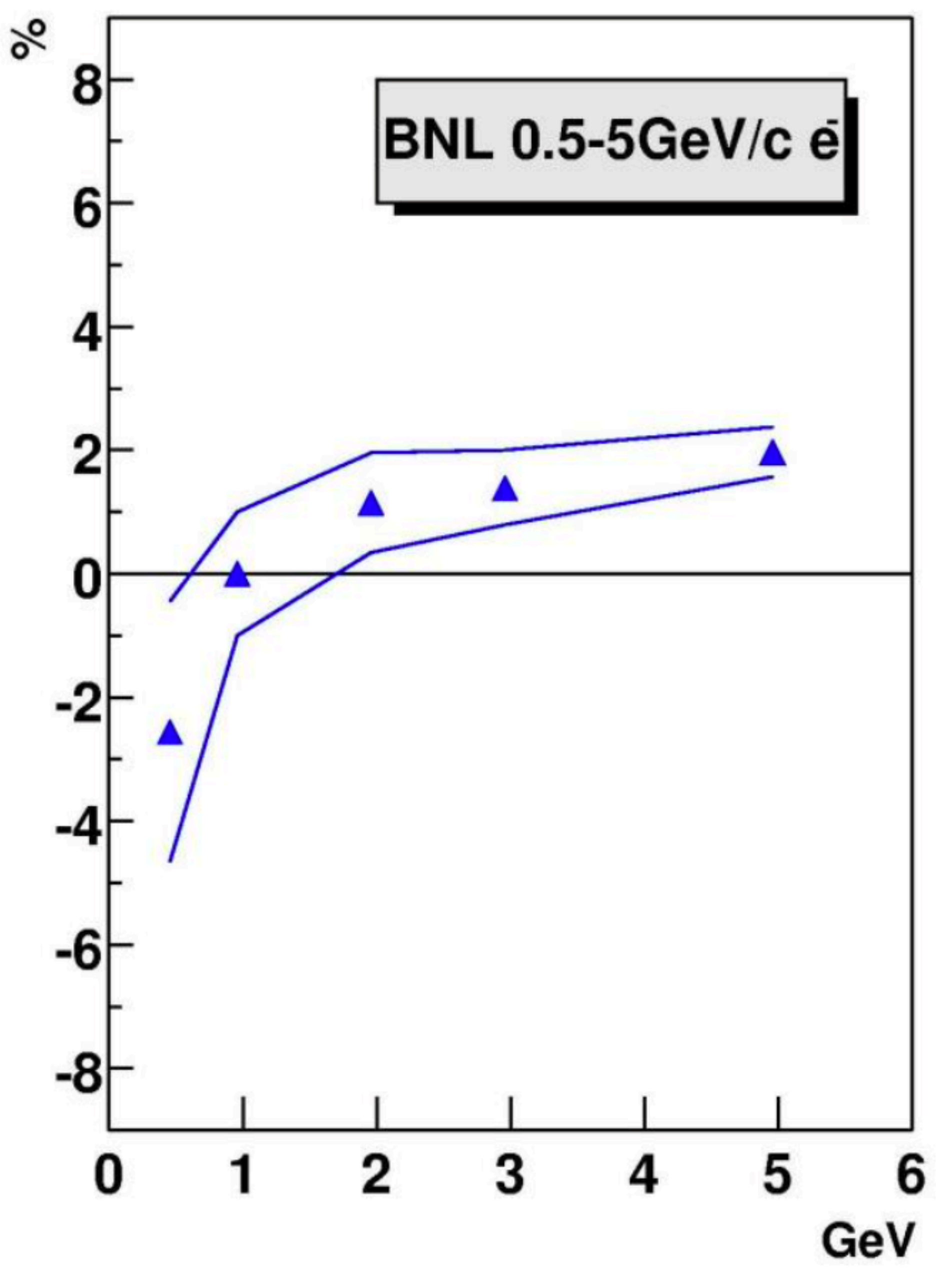
•Absorber : Pb
•Scintillator:
1.5 % PTP / 0.01 % POPOP

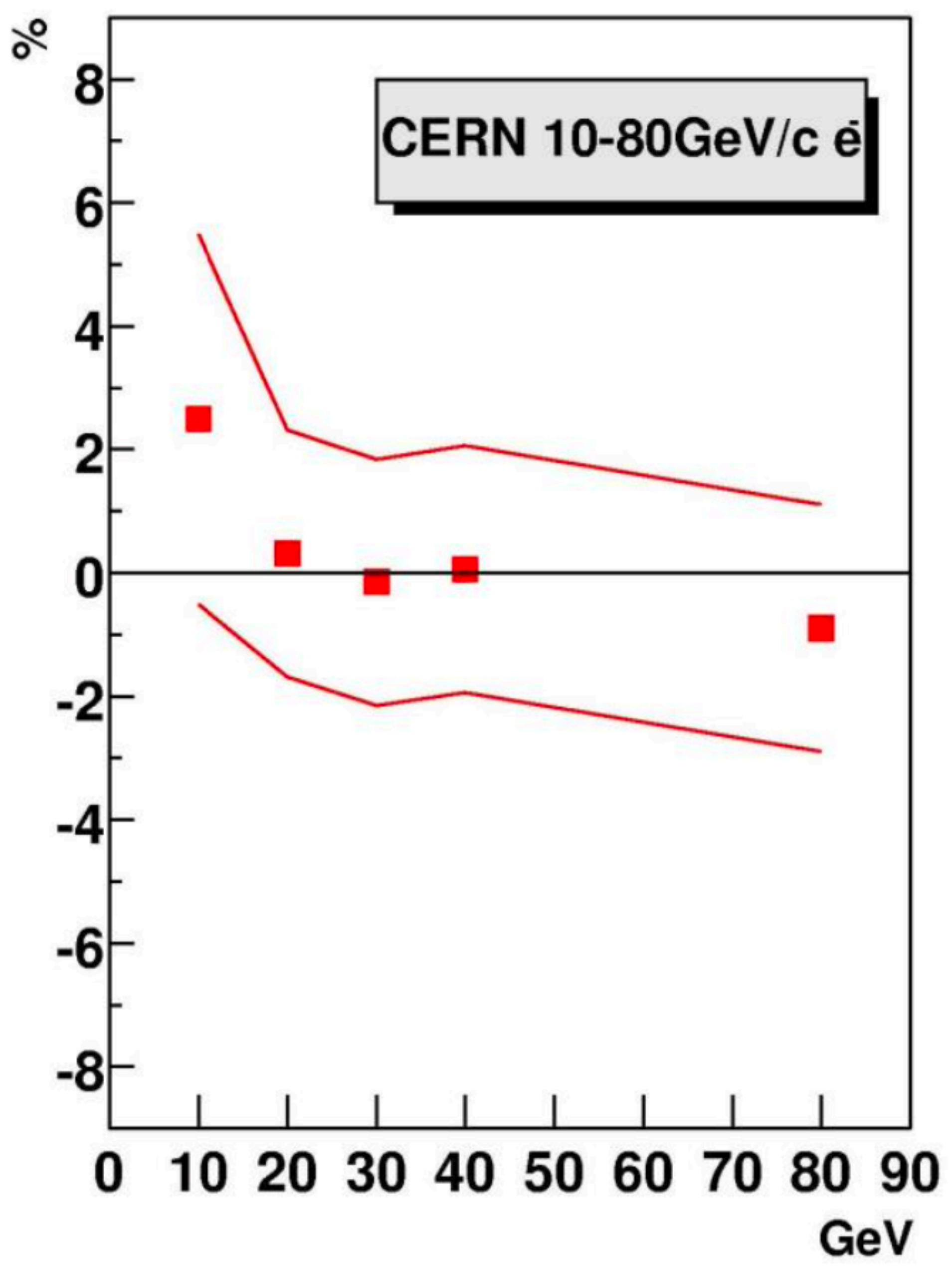


1 PbSc tower:
• 66 sampling cells
• 1.5 mm Pb, 4 mm Sc
• Ganged together by penetrating wavelength shifting fibers for light collection
• Readout: FEU115M phototubes

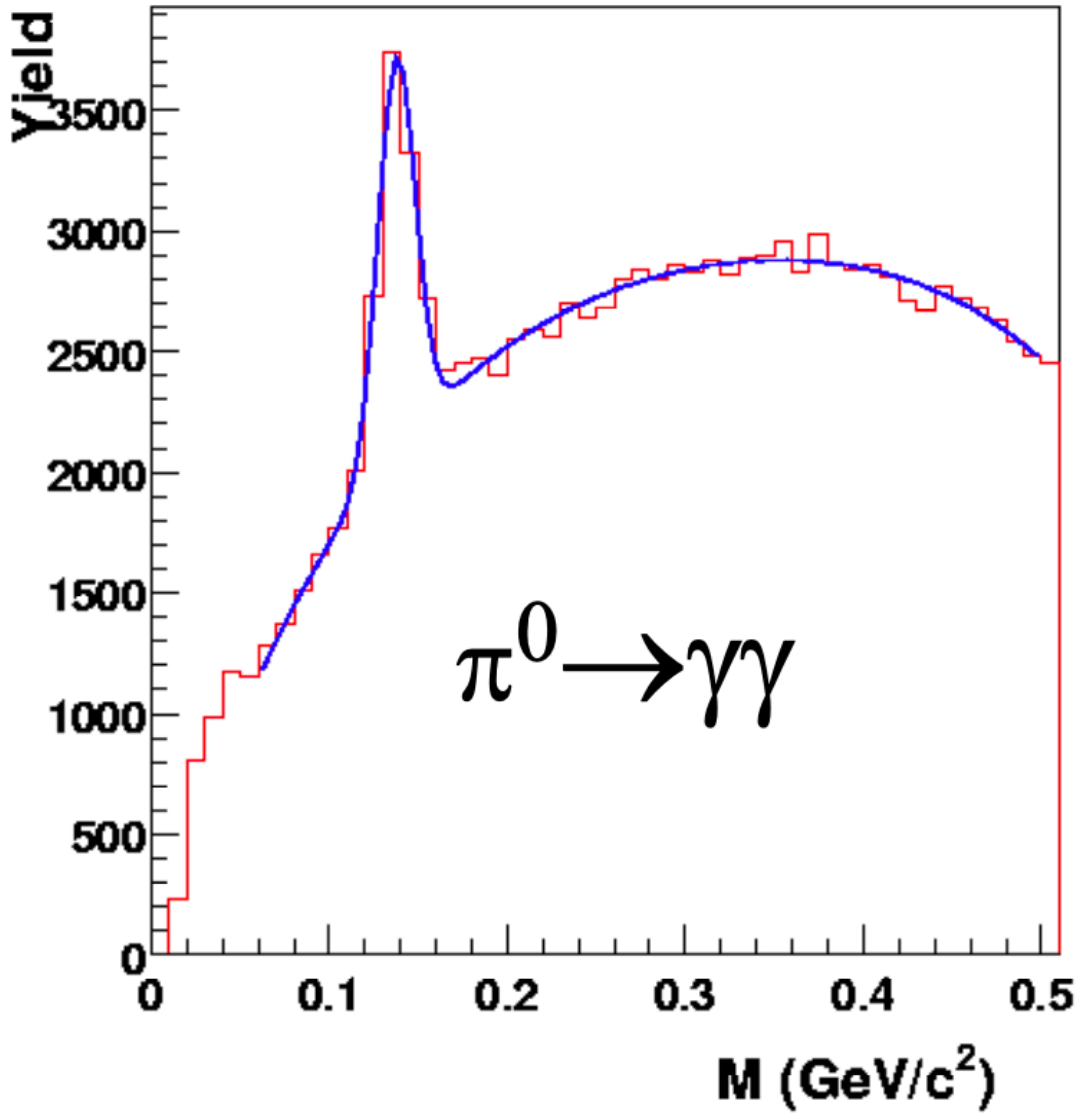
1 FEM reads out 1 Supermodule

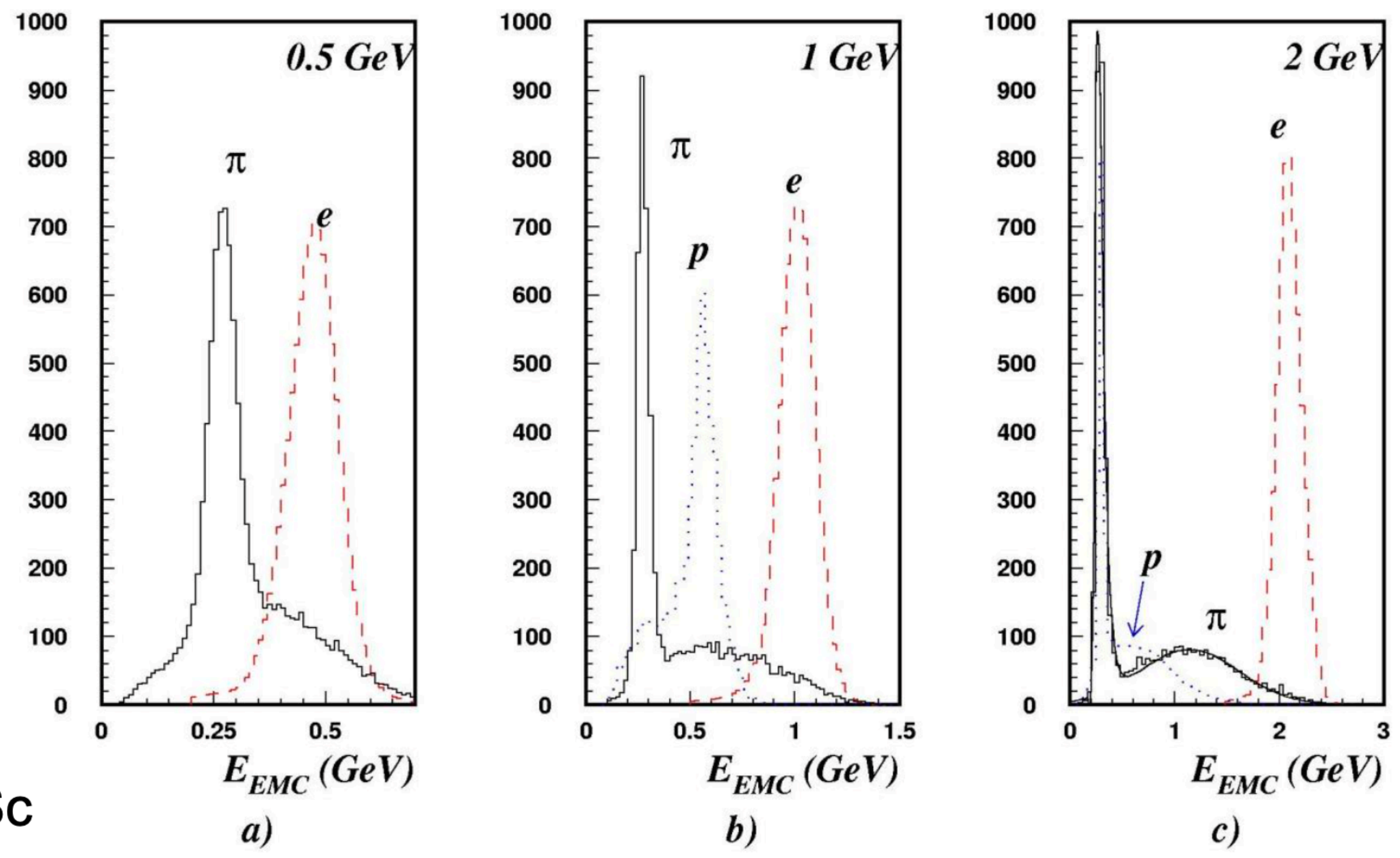


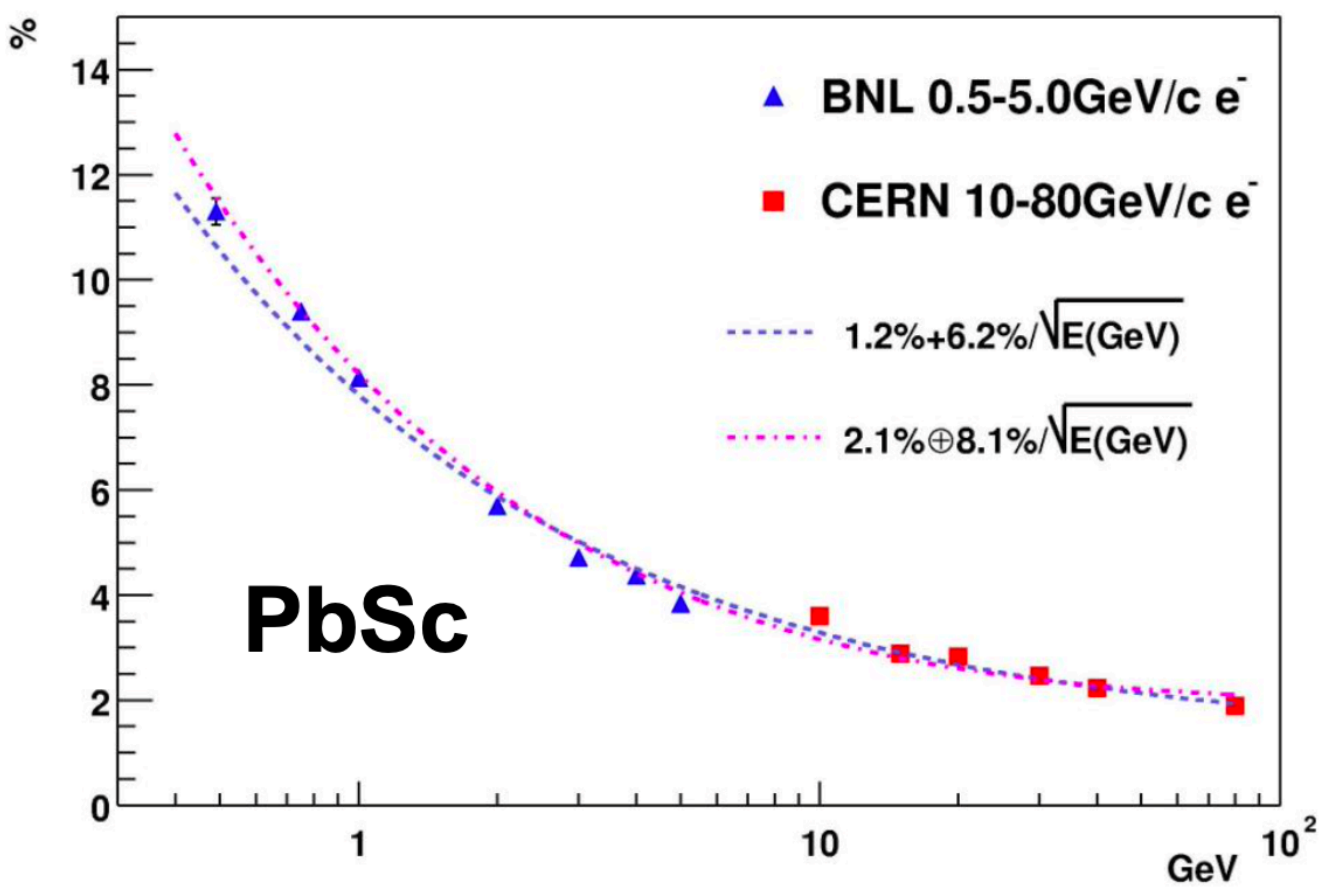




Performance from PHENIX







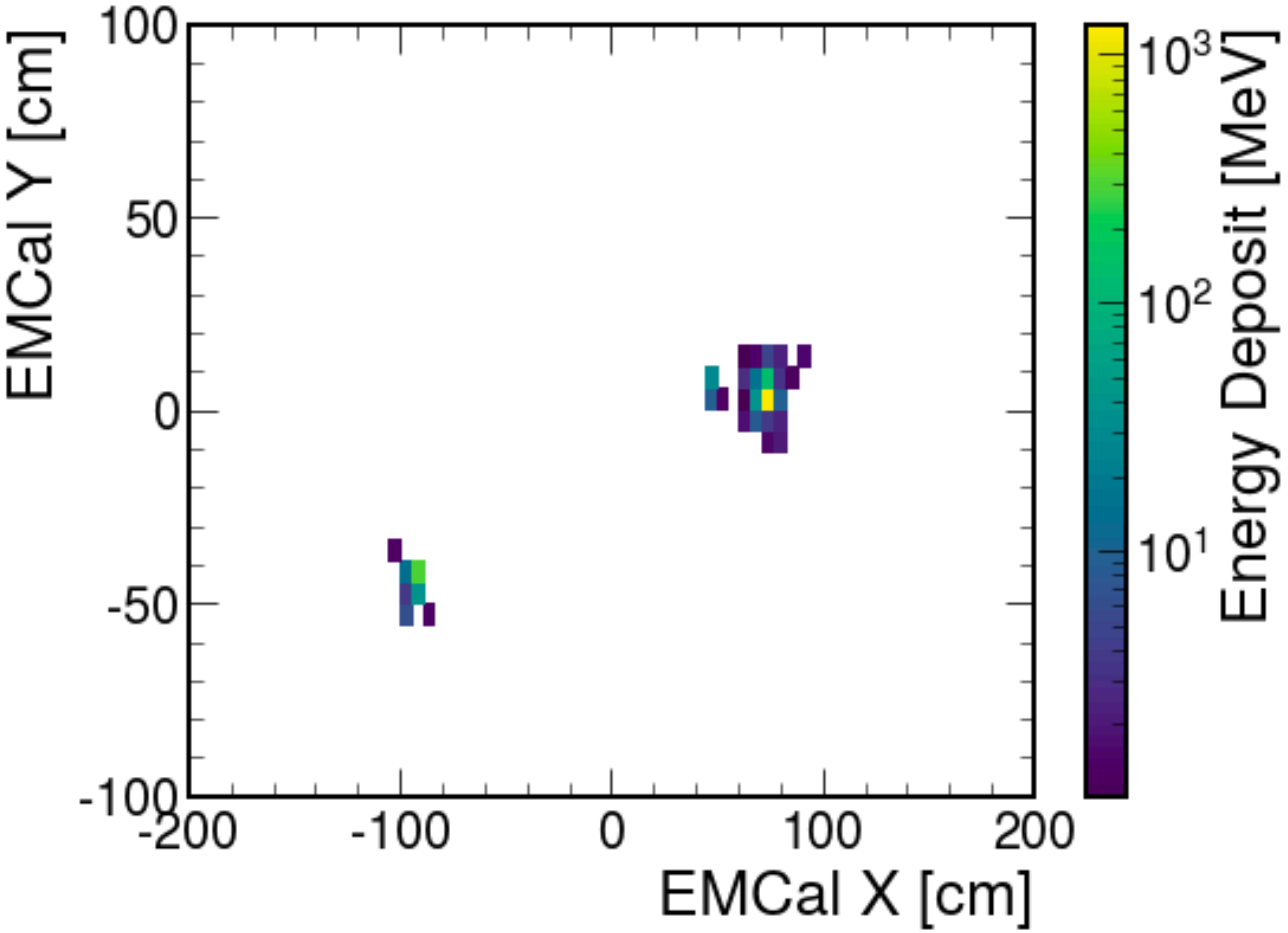
EMCal Simulation Setup

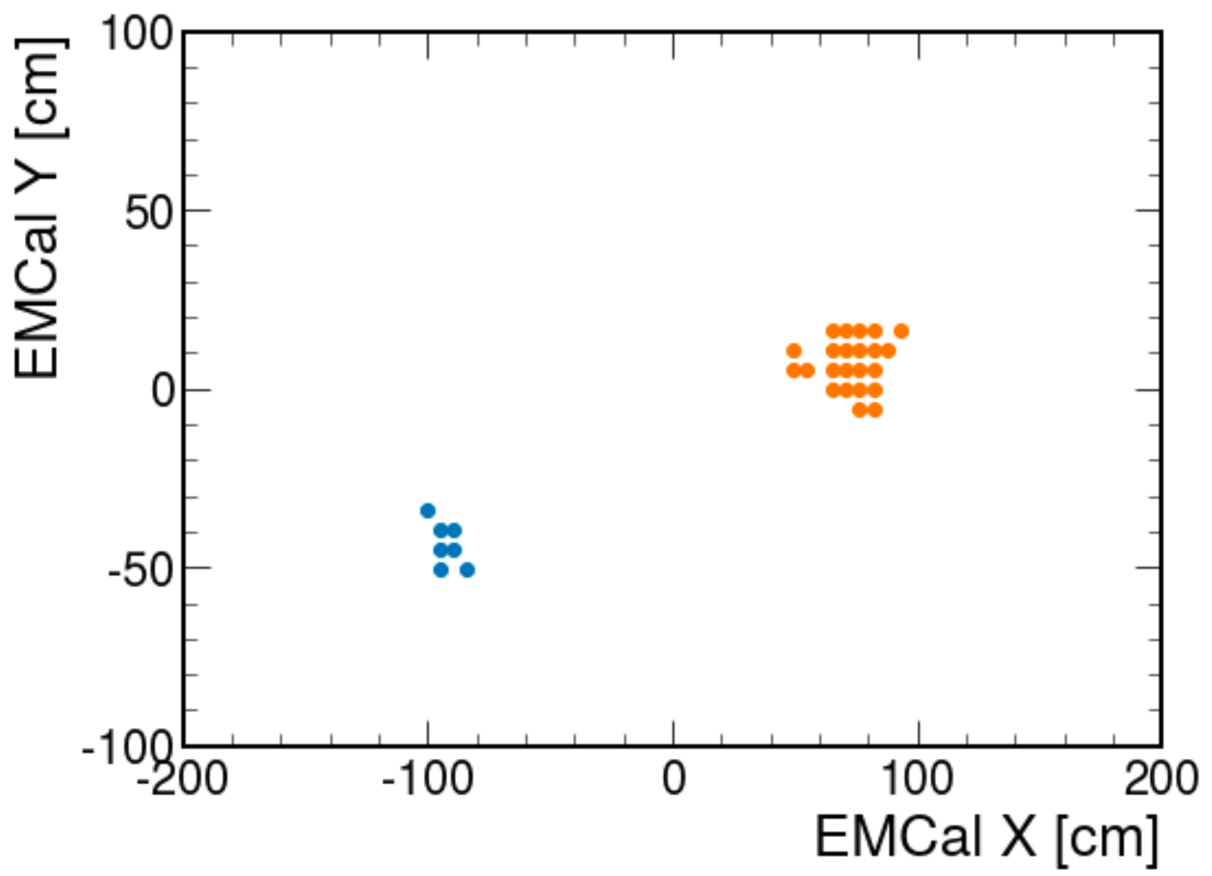
- PHG4EMCalDetector.cc
- PHG4EMCalDetector.h
- PHG4EMCalSteppingAction.cc
- PHG4EMCalSteppingAction.h
- PHG4EMCalSubsystem.cc
- PHG4EMCalSubsystem.h
- PHG4EMCalSubsystemLinkDef.h

```
// This is an approximation for the W saturated epoxy of the EMCal.
G4Material *W_Epoxy = new G4Material("W_Epoxy", density = 10.2 * g / cm3, ncomponents = 2);
W_Epoxy->AddMaterial(G4Material::GetMaterial("G4_W"), fractionmass = 0.5);
W_Epoxy->AddMaterial(G4Material::GetMaterial("G4_POLYSTYRENE"), fractionmass = 0.5);
```

- EMCalDetector and EMCalSubSystem defined in Geant Classes, with certain Geometry and material information
- Can add another material, geometry, etc into the simulation if necessary. Should not be hard.
- Can also add effects in the offline analysis code

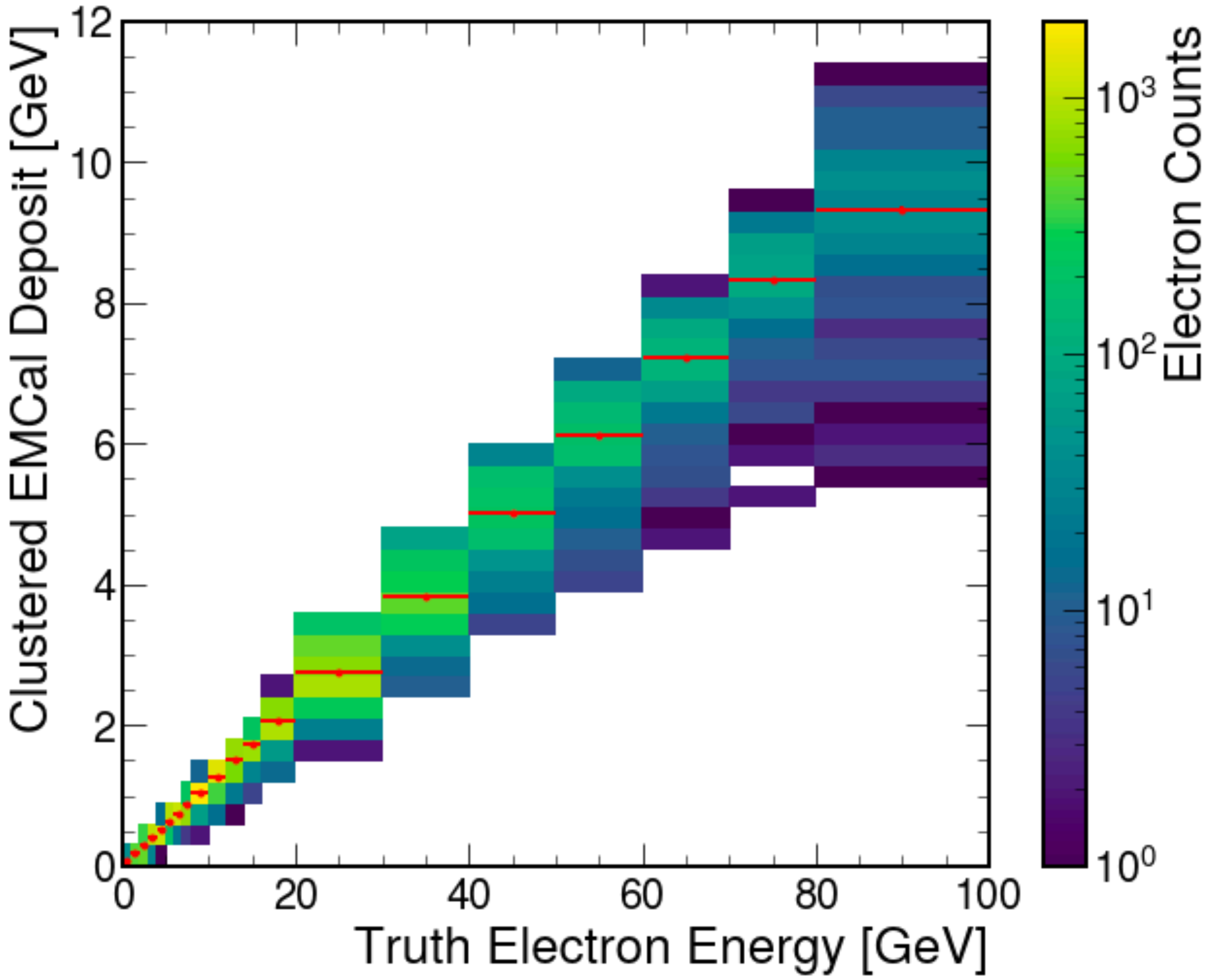
EMCal Energy Clustering

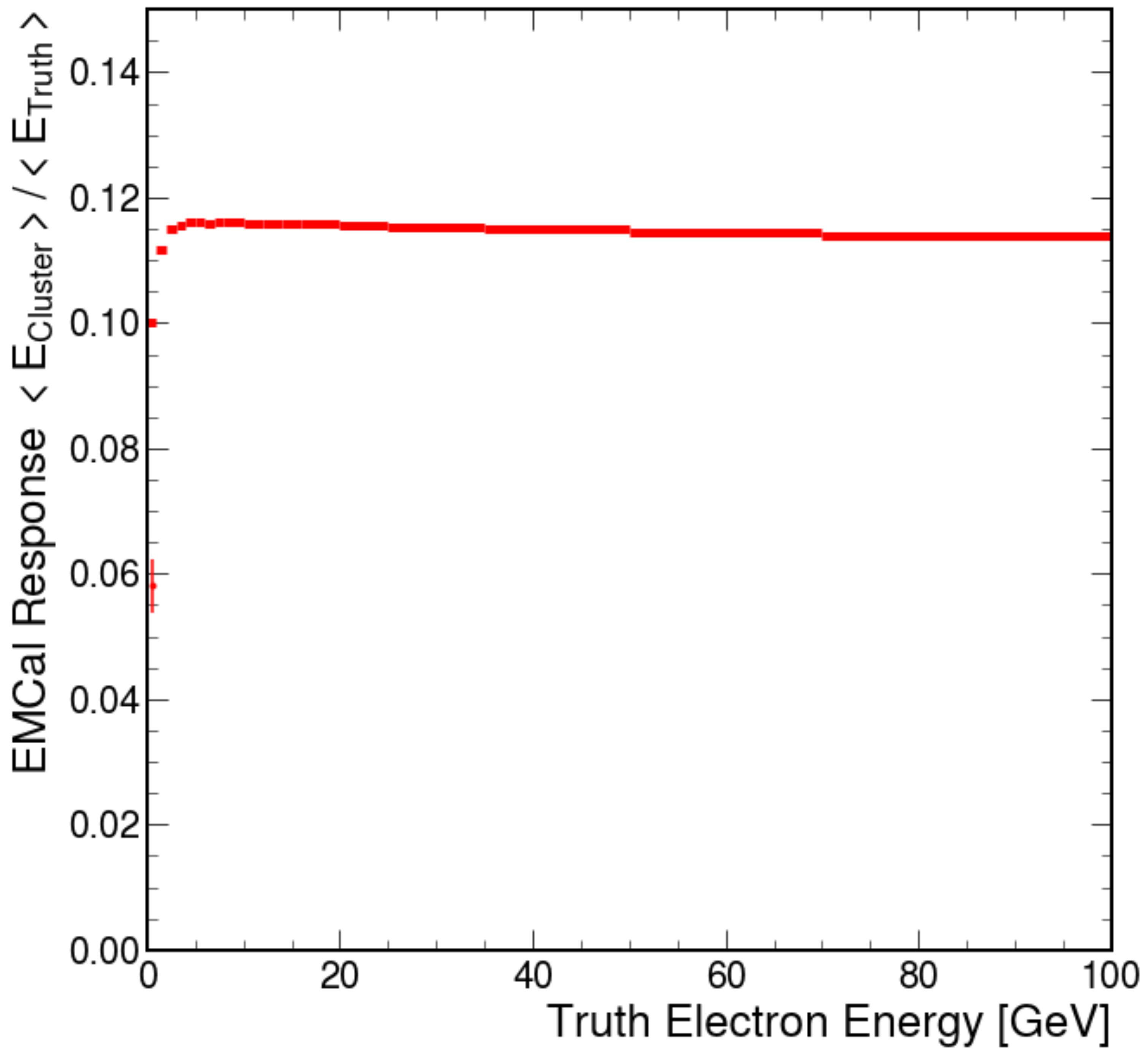


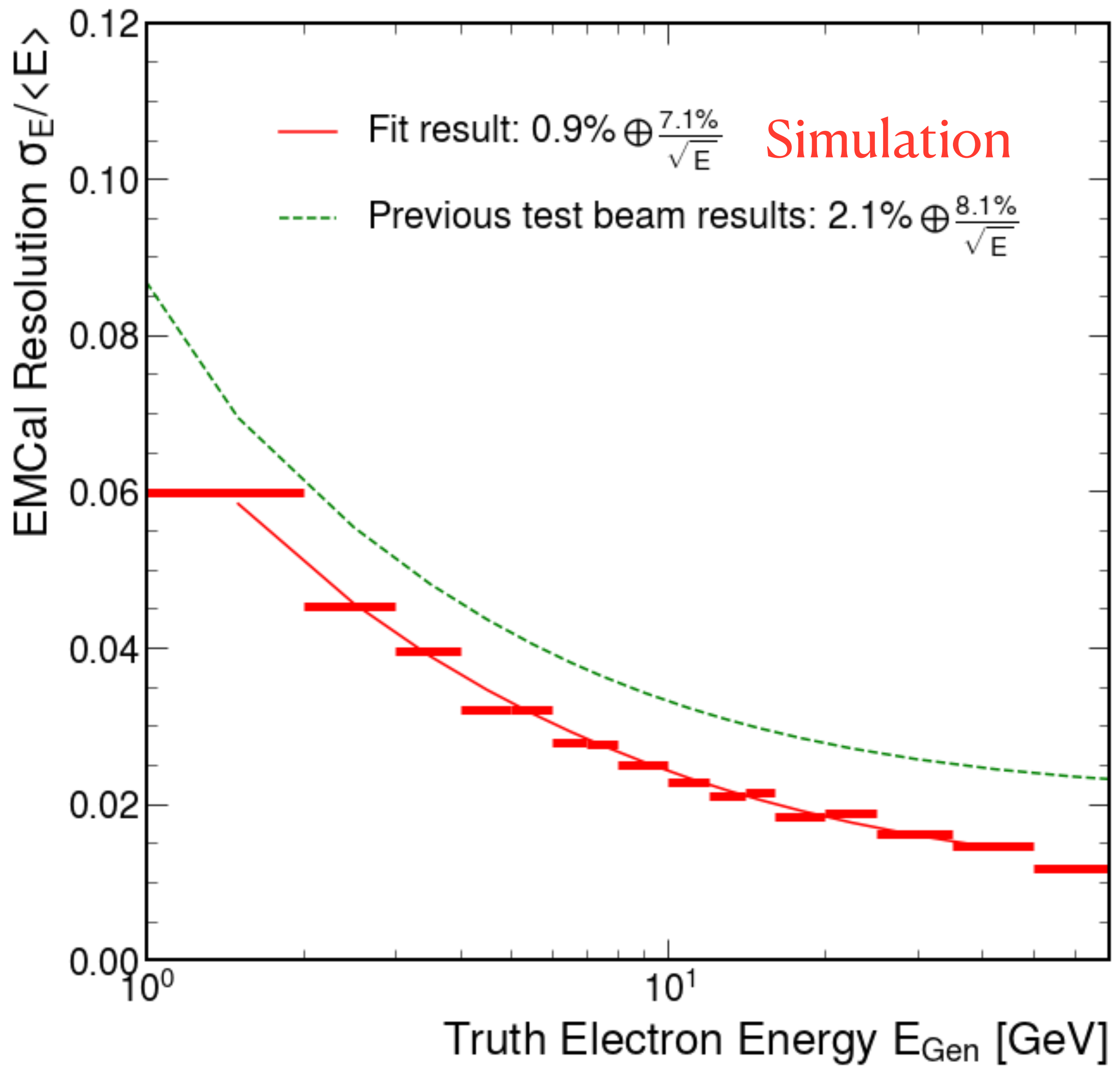


- One example of the energy deposit in the EMCal (left) and the result with BirchClustering (top), using one A' decaying to two electron signal event.
- Truth energy of the two electrons are 3.2 GeV and 13.8 GeV, with truth EMCal energy deposit of 0.37 GeV and 1.61 GeV;
- The energy deposits are centered around the 1-4 towers, with some small remaining energy deposits at its neighbors.
- Two well-separated clusters. The energy-weighted clustering performance is similar.
- Cluster code, if necessary, can be added to the offline reco code (cpp) or further downstream (python)

EMCal Performance







- Left plot is the clustered electron energy vs the generated electron energy. Aligns well, except at large E there are some electrons with smaller EMCal energy deposits.
- Middle plot is the response distribution - flat around 0.115 (sampling fraction)
- Right plot is the resolution distribution - follows the $a \oplus \frac{b}{\sqrt{E}}$, except at large E. (Here the energy resolution is defined as $q_{84} - q_{50}$ to avoid the long tail effects; q_{50} is the 50% quantile)

Backgrounds

- Proton simulations basically takes 0.1s - 1.0 s per proton, depending on the proton energy
- Can do separate particles: KL, Ks, etc. But what should be a reasonable kinematic distribution of these particles?
  - KL and pion0 distribution can be calculated to some extend?
  - If this is the case, can easily inject these particles via gun and study the performances

Framework Setup

- Signal simulations: HepMC file with predefined cross sections and kinematic distributions, for fixed couplings and mass points:
  - Can we have the degree of freedom to generate signals ourselves, calculate cross sections, acceptance, and finally the expected number of events (before folding experimental effects).
- Currently different modules are running separately: trigger efficiency, tracking reco, vertexing, etc
  - Would be nice to build a pipeline to connect everything together, so that different modules can be correlated to some extend.

- Implement some CI for the code developments and checks on github
- Docker container for the python analyzers?
- End2end analysis: from simulation, efficiency, to final sensitivity

Back Up