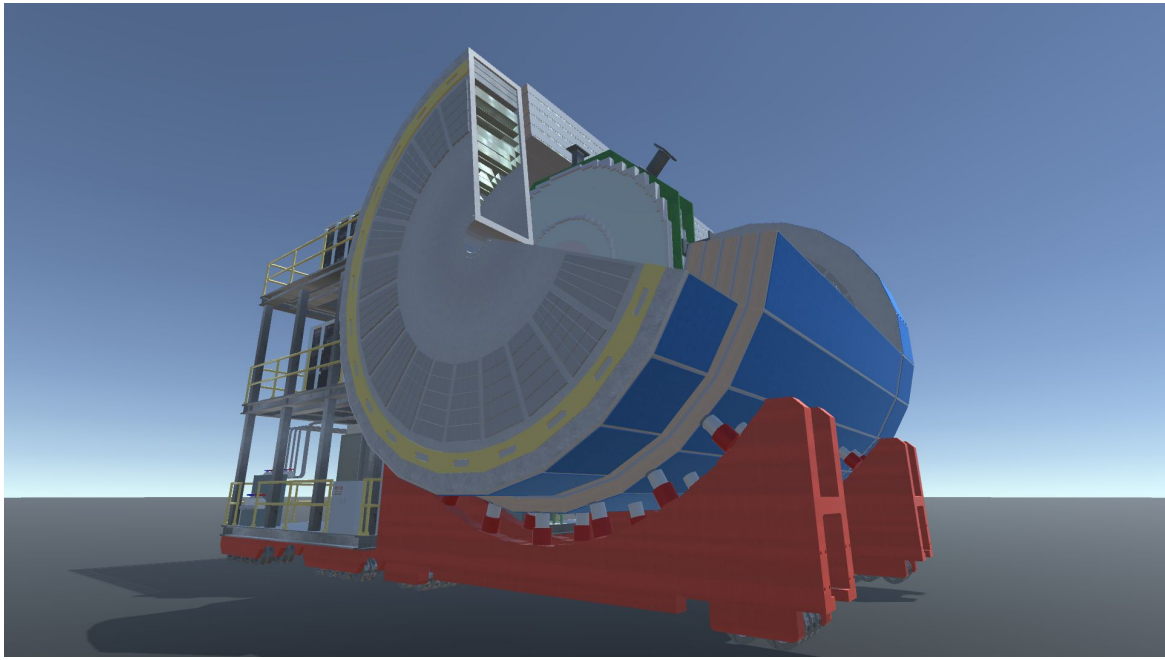
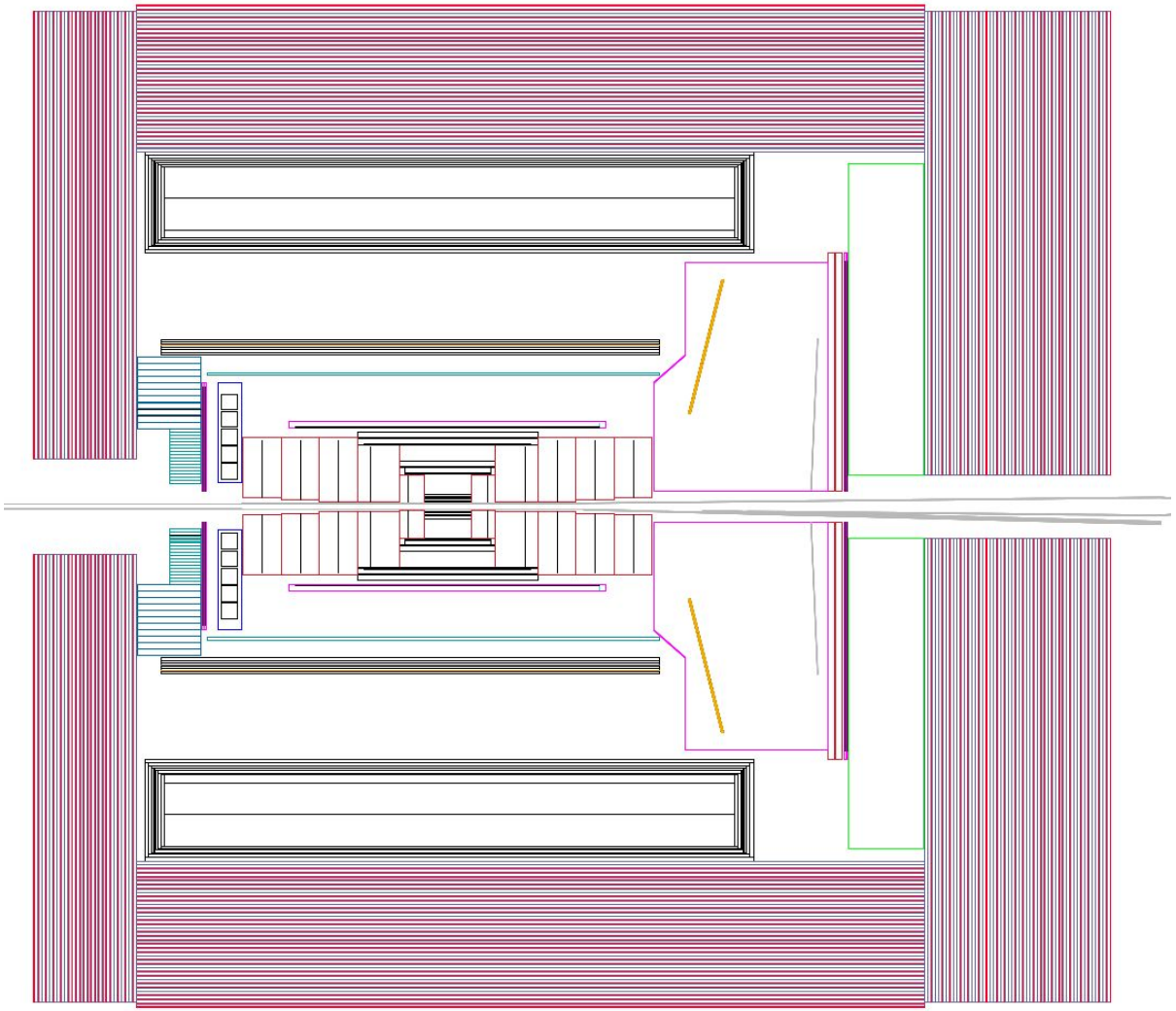
Hadronic reconstruction with full ATHENA sims

Miguel Arratia (UCR),
June 28th 2021
(T-156 days)

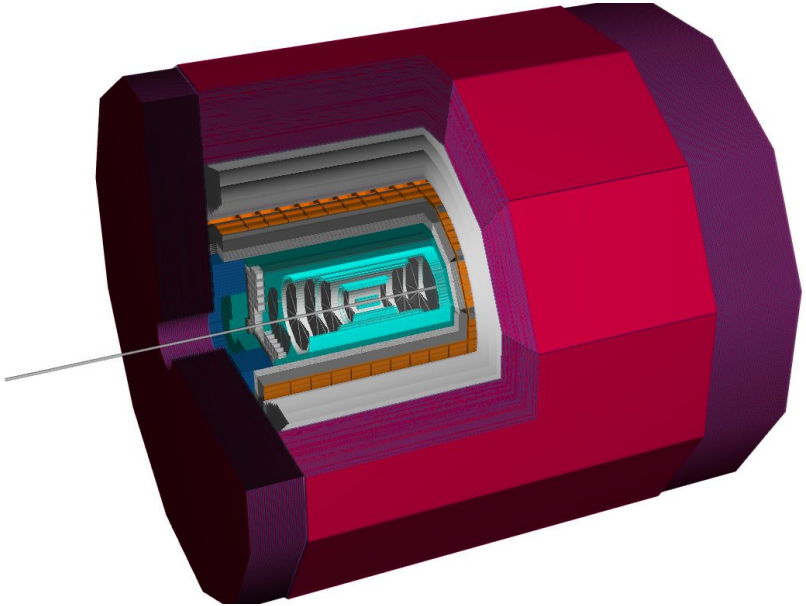


Status of ATHENA full sims



https://eicweb.phy.anl.gov/EIC/detectors/athena

Geometry, materials and reconstruction code is already in place to do calorimeter-based hadronic reco



Check model out if you have not done so yet

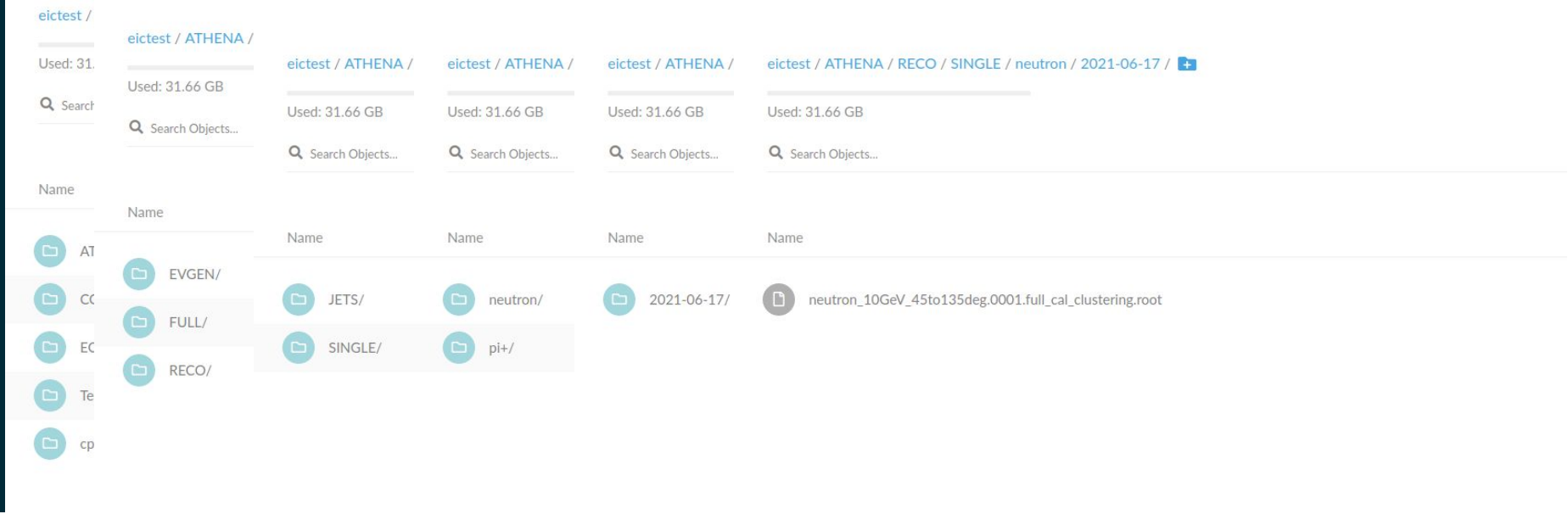
https://eic.phy.anl.gov/geoviewer/index.htm?nobrowser&file=https://eicweb.phy.anl.gov/api/v4/projects/473/jobs/artifacts/master/raw/geo/detector_geo.root?job=report&item=default;1&opt=clipxyz;transp30;zoom100;ROTY0;ROTZ0;trz100;trr0;ctrl;all&

Step #1: Get simulation samples

Slides from Wouter

Accessing Large Data Productions: From the Web (username: eicS3read , eicS3read pass)

https://dtn01.sdcc.bnl.gov:9000/minio/login



Accessing Large Data Productions: Command Line

Download the Minio client:

```
wget https://dl.min.io/client/mc/release/linux-amd64/mc
```

Register your S3 instance:

```
./mc config host add S3 https://dtn01.sdcc.bnl.gov:9000 $u $p
```

Copy files (recursively):

```
./mc cp -r S3/eictest/ATHENA/RECO/SINGLE/neutron/2021-06-17 .
```

Full docs: http://doc.athena-eic.org/en/latest/howto/s3_file_storage.html

Also works in one line in python

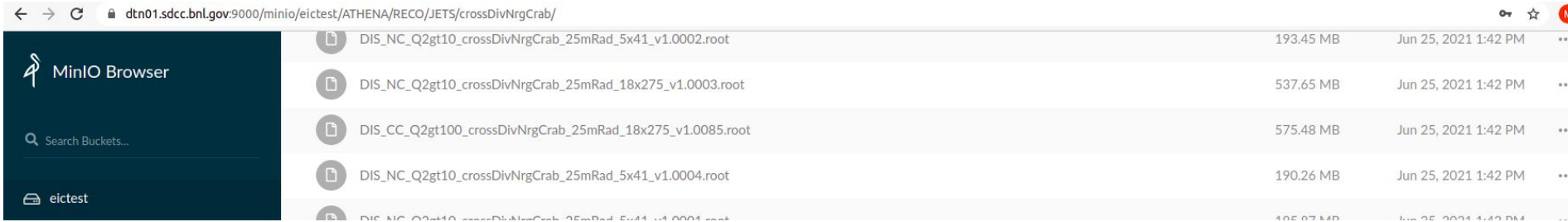
In [10]:
```
#%matplotlib notebook
import uproot as ur
import matplotlib.pyplot as plt
import numpy as np
import awkward as ak

import pandas as pd
```

Get data, transform ROOT tree into array

In [11]:
```
events = ur.open('root://sci-xrootd.jlab.org//osgpool/eic/ATHENA/RECO/JETS/crossDivNrgCrab/DIS_NC_Q2gt10_crossDivNrg
```

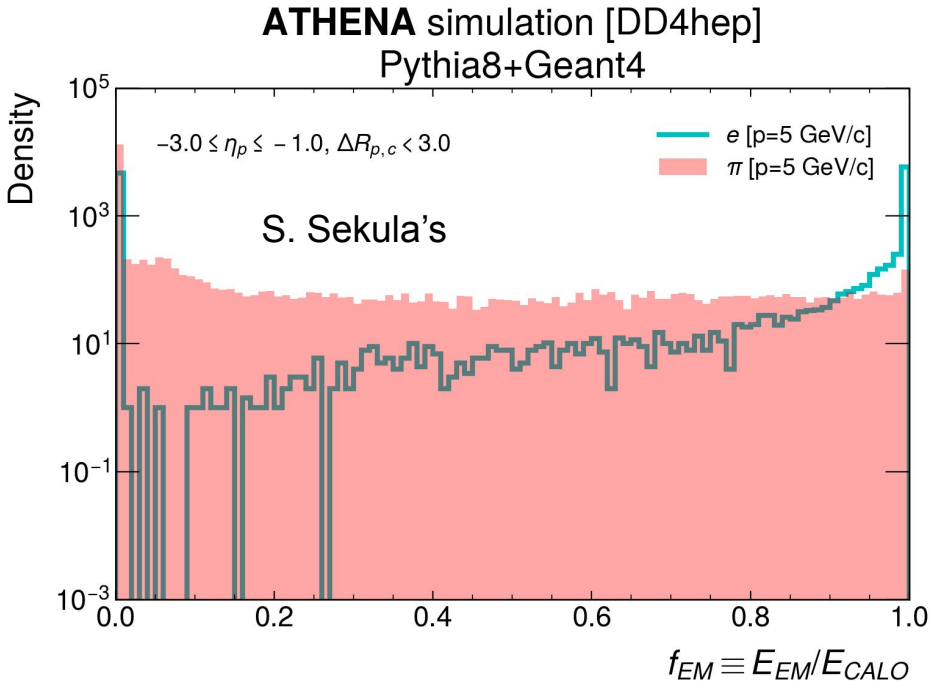
Both NC and CC samples available

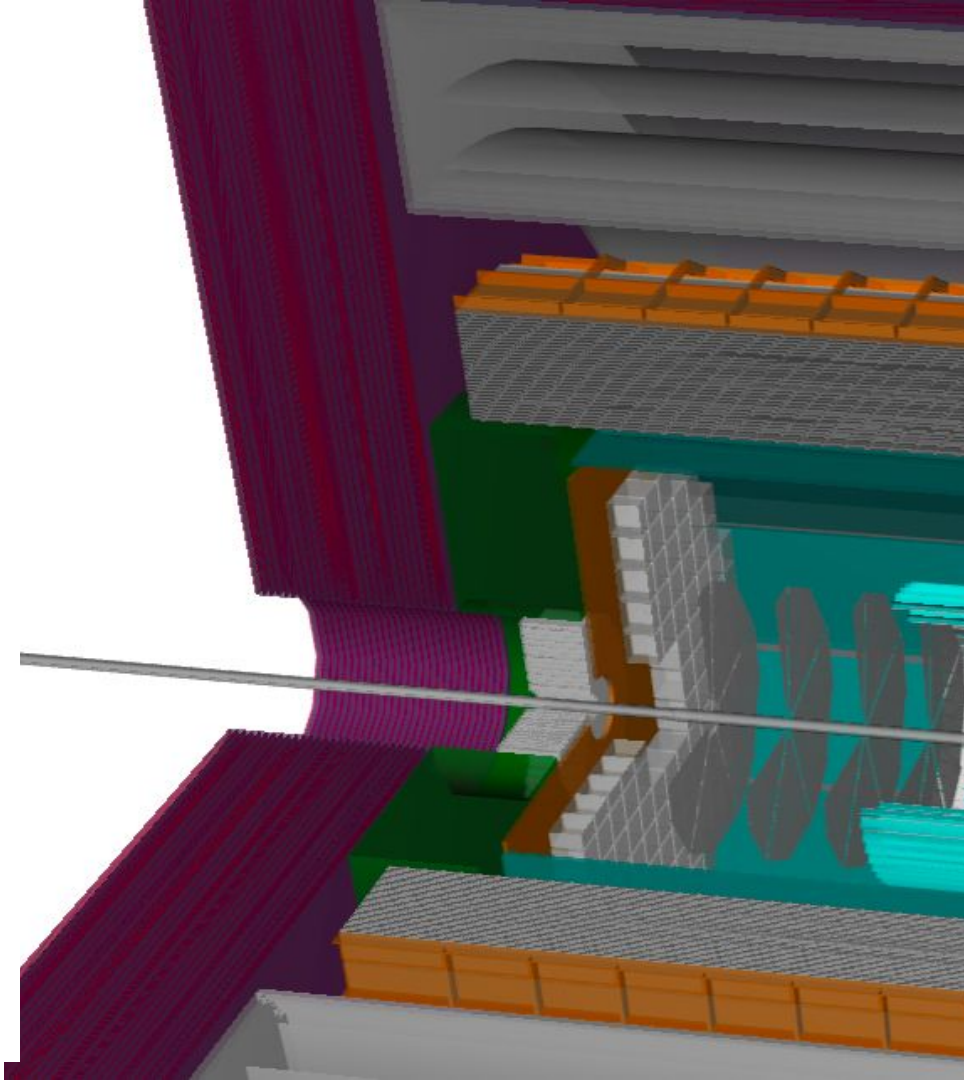
Min Q2 of 10 GeV2 or 100 GeV2 for now (Pythia8 samples)



An example of
using DD4hep for hadronic reconstruction

Electron-pion separation With HCAL





Scattered electrons

Defined as highest pT
Isolated cluster in event

```
def find_electron(cluster_container,hcal_container, E_threshold=3.0):
    #Returns cluster with the highest pT in the event
    #print(cluster_container.keys())
    ptmax = 0.0
    electron_iso = 999
    index_max = -999
    nclusters= len(cluster_container['E'])

    for i in range(nclusters):
        clus_E = cluster_container['E'][i]/1000.0
        if(clus_E<E_threshold):
            continue
        if(cluster_container['theta'][i]*180.0/np.pi<2):
            continue
        clus_theta = cluster_container['theta'][i]
        clus_phi = cluster_container['phi'][i]

        clus_eta = -np.log(np.tan(clus_theta/2.0))
        clus_pt  = clus_E*np.sin(clus_theta)

        #print('Ecal isolation')
        clus_ecal_iso = isolation(clus_theta,clus_phi, cluster_container)
        #print('Hcal isolation')
        clus_hcal_iso = isolation(clus_theta,clus_phi, hcal_container)
        clus_iso = clus_ecal_iso + clus_hcal_iso - clus_E
        #remove non-isolated clusters
        if(clus_iso>clus_E*0.1): continue

        if(clus_pt>ptmax):
            ptmax = clus_pt
            index_max = i
            electron_iso= clus_iso
    if(index_max>0):

        print('This is the electron candidate:')
        print('ptmax %2.2f GeV'%ptmax, 'energy %2.2f GeV'%(cluster_container['E'][index_max]/1000.0))
        print('polar angle %2.2f, azimthal angle =%2.2f, degrees '%(cluster_container['theta'][index_max]*180.0/np.
        print('ISOLATION (R=1.0) isolation = %2.2f GeV'%(electron_iso))
        #print('clus ECAL iso =%2.2f, HCAL iso =%2.2f'%(clus_ecal_iso,clus_hcal_iso))
    #print(index_max)
```

```
]: def isolation(cone_theta,cone_phi, cluster_container, E_threshold=0.1):
    nclusters= len(cluster_container['E'])
    #if(cone_theta<0.05):
    #    return -999
    cone_eta = -np.log(np.tan(cone_theta/2.0))
    cone_iso = 0.0
    for i in range(nclusters):
        clus_E = cluster_container['E'][i]/1000.0
        if(clus_E<E_threshold):
            continue
        #print('Cluster E =%2.2f'%(clus_E))
        clus_E = cluster_container['E'][i]/1000.0
        clus_phi = cluster_container['phi'][i]
        clus_theta = cluster_container['theta'][i]
        clus_eta = -np.log(np.tan(clus_theta/2.0))
        clus_pt  = clus_E*np.sin(clus_theta)
        #print('Cluster E =%2.5f, cluster phi =%2.2f, cluster theta=%2.5f, cluster eta=%2.2f'%(clus_E,clus_phi,clus_
        dphi = clus_phi - cone_phi
        dphi = (dphi + np.pi) % (2 * np.pi) - np.pi
        #print(' Delta- phi = %2.2f'%(dphi))
        dr = np.sqrt((dphi)**2 + (clus_eta-cone_eta)**2)
        #print('dr=%2.2f (cone eta %2.2f, cone azimuth=%2.2f), so dphi =%2.2f'%(dr,cone_eta,cone_phi,dphi))
        if(dr<1.0):
            cone_iso += clus_E

    #print('Cone phi %2.2f, cone eta %2.2f, cone theta %2.2f, cone E= %2.2f GeV'%(cone_phi,cone_eta,cone_theta,cone_
    return cone_iso
```

Hadronic reco

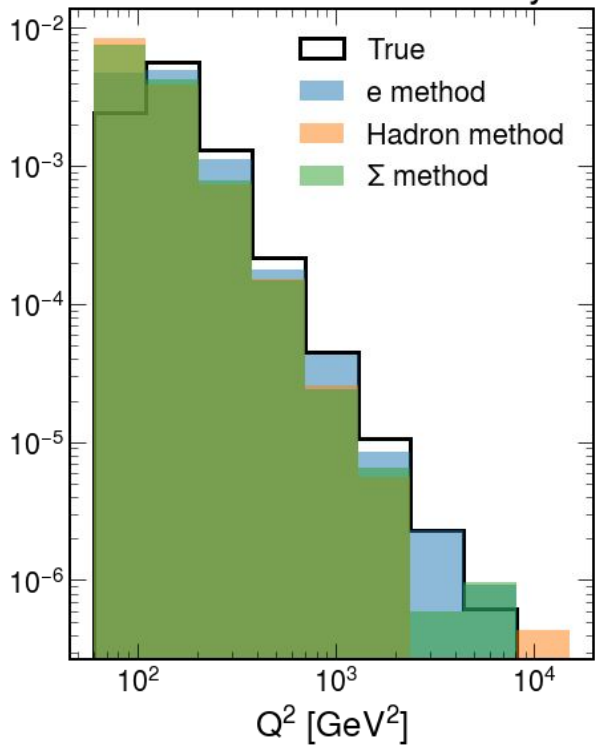
```
def get_Empz(cluster_container,skip=None):
    nclusters= len(cluster_container['E'])
    Empz = 0.0
    #print('nclusters ', nclusters)
    for i in range(nclusters):
        if( skip is not None):
            if(skip==i):
                print('Skipping electron %i'%(skip))
                continue
        #print(cluster_container['E'][i])
        Empz += cluster_container['E'][i]*(1-np.cos(cluster_container['theta'][i]))
    #print(Empz)
    return Empz/1000.0

def get_total_pxpy(cluster_container,E_threshold=0.1,skip=None):
    nclusters= len(cluster_container['E'])
    sum_px = 0.0
    sum_py = 0.0
    sum_pt = 0.0
    for i in range(nclusters):

        if( skip is not None):
            if(skip==i):
                print('Skipping electron %i'%(skip))
                continue

        clus_E = cluster_container['E'][i]/1000.0
        if(clus_E<E_threshold):
            continue
        clus_phi =   cluster_container['phi'][i]
        clus_theta = cluster_container['theta'][i]
        clus_eta = -np.log(np.tan(clus_theta/2.0))
        clus_pt  = clus_E*np.sin(clus_theta)
        clus_px  = clus_pt*np.cos(clus_phi)
        clus_py  = clus_pt*np.sin(clus_phi)
        sum_px += clus_px
        sum_py += clus_py
    sum_pt = np.sqrt(sum_px*sum_px + sum_py*sum_py)
    #print('sum px', sum_px)
    #print('sum py', sum_py)
    #print('sum pt ', sum_pt)
    return sum_px, sum_py
```

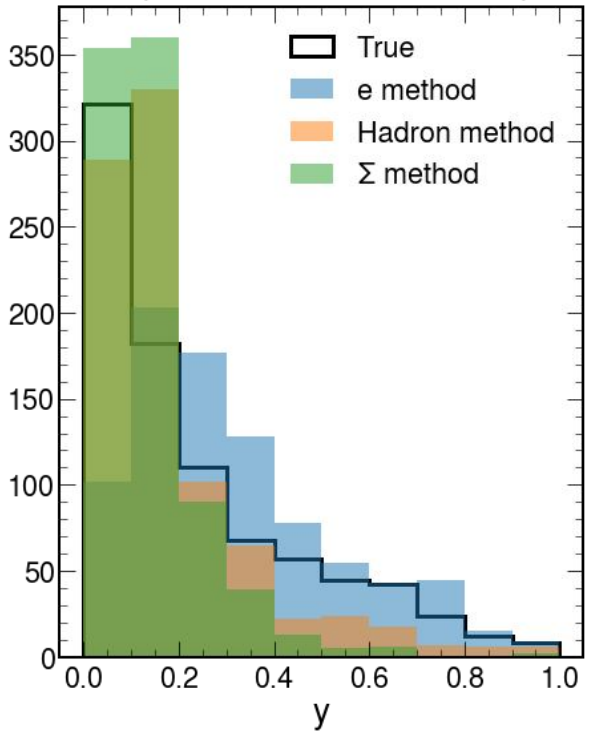
Hadronic reco

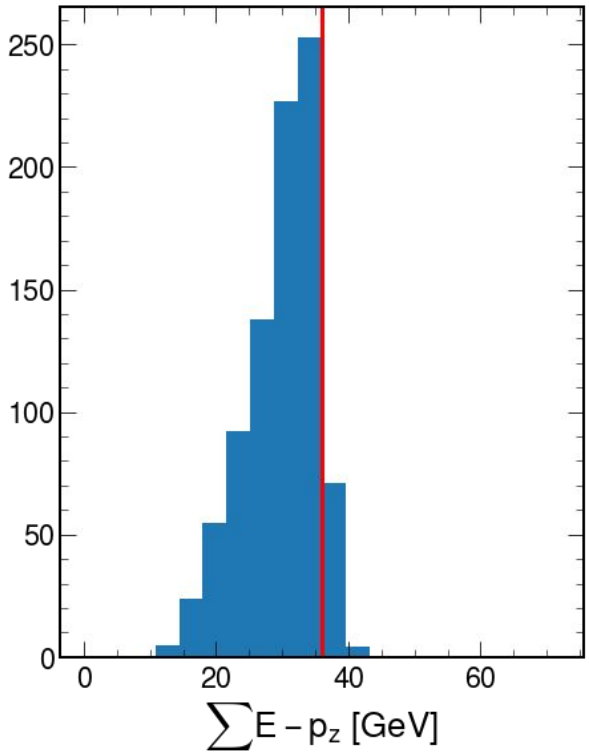
```
Empz = get_Empz(hcal_clusters) + get_Empz(ecal_clusters,skip=e_index)
evt_Empz = np.append(evt_Empz, get_Empz(hcal_clusters) + get_Empz(ecal_clusters))


ecal_px, ecal_py = get_total_pxpy(ecal_clusters,skip=e_index)
hcal_px , hcal_py = get_total_pxpy(hcal_clusters)
total_px = ecal_px + hcal_px
total_py = hcal_py + hcal_py
#print('TOTAL ', total_px, total_py)
TOT = np.sqrt(total_px*total_px + total_py*total_py)
print('MET = %2.2f'%MET)
evt_TOT = np.append(evt_TOT, TOT)

##Hadronic reconstruction (Jacquet Blondel)
y_h = Empz/(2.0*18.0)
Q2_h = MET*MET/(1.0-y_h)
evt_yh = np.append(evt_yh, y_h)
evt_Q2h = np.append(evt_Q2h, Q2_h)

print('electron theta %2.2f' %(electron_theta), ' electron energy =%2.2f'%(electron_E))
## Sigma method
print('Empz = %2.2f '%(Empz))
print('electron_E*(1-np.cos(electron_theta)) =%2.2f'%(electron_E*(1-np.cos(electron_theta))))
y_sigma = Empz/(Empz + electron_E*(1-np.cos(electron_theta)))
Q2_sigma= (electron_E*electron_E)*(np.sin(electron_theta)*np.sin(electron_theta))/(1-y_sigma)
evt_ysigma = np.append(evt_ysigma, y_sigma)
evt_Q2sigma = np.append(evt_Q2sigma, Q2_sigma)
```

ATHENA full simulation [DD4hep]

Pythia8+Geant4, NC DIS 18x275 GeV, $Q^2 > 100$ GeV^2

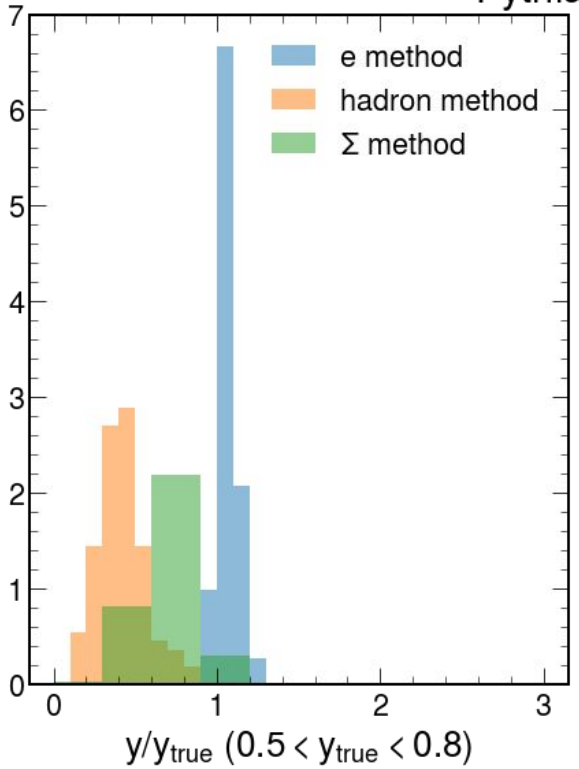


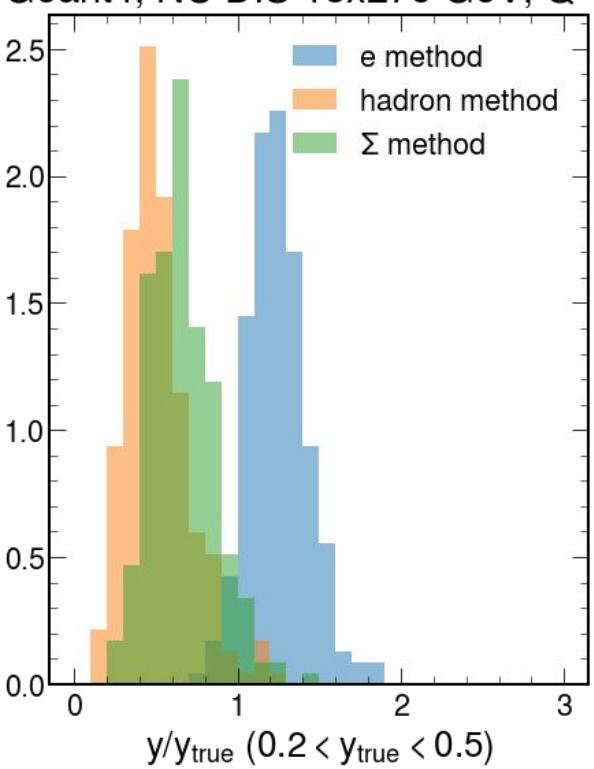


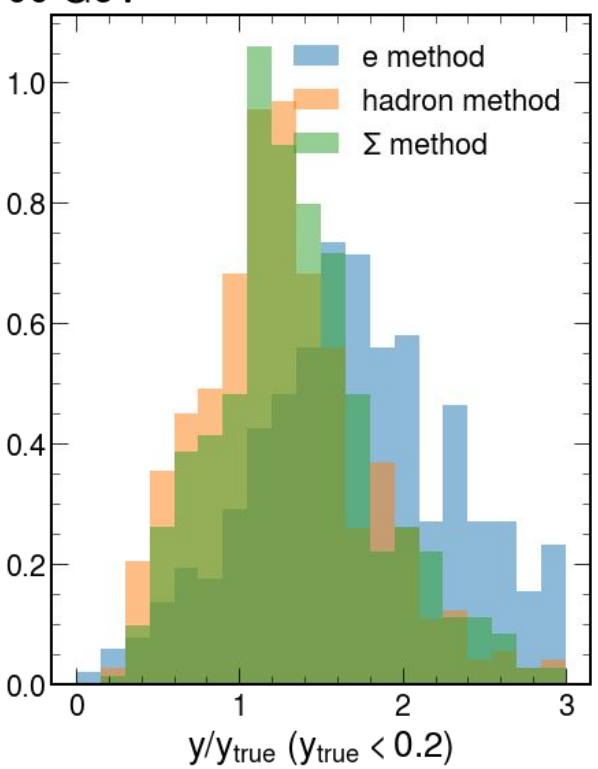


ATHENA full simulation [DD4hep]

Pythia8+Geant4, NC DIS 18x275 GeV, $Q^2 > 100$ GeV^2







Summary.

ATHENA full simulation chain is working, samples for NC and CC analysis are ready
(*calorimeter-based reco only for the moment)

Let’s work together on implementing energy-flow for hadronic reconstruction (needs tracking)