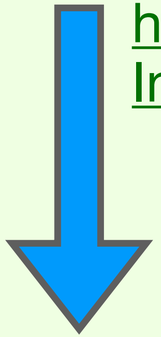
Simulation code structure

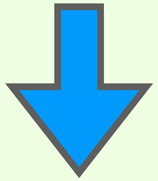
Particle Generation in Fun4All (Fun4All_physiTuto.C):
https://github.com/sPHENIX-Collaboration/INTT/blob/main/general_codes/tkumaoka/InttSeedingTrackDev/ParticleGen/macro/Fun4All_physiTuto.C



root file including DST clusters

Tracking Performance:
https://github.com/sPHENIX-Collaboration/INTT/blob/main/general_codes/tkumaoka/InttSeedingTrackDev/InttSeedTrackPerformance/src/InttSeedTrackPerformance.cxx

INTT Seeding Tracking :
https://github.com/sPHENIX-Collaboration/INTT/blob/main/general_codes/tkumaoka/InttSeedingTrackDev/InttSeedTrackPerformance/src/InttSeedTracking.cxx



root file: Performance histograms

How to Run the Particle Generation Simulation

Particle simulation
https://github.com/sPHENIX-Collaboration/INTT/tree/main/general_codes/tkumaoka/InttSeedingTrackDev/ParticleGen

```
mv physiTuto/build
source /opt/sphenix/core/bin/sphenix_setup.sh
make clean
 source /opt/sphenix/core/bin/sphenix_setup.sh
 ../autogen.sh --prefix=$PWD/../install
make install
make
source /opt/sphenix/core/bin/setup_local.sh ${PWD}/../install
export ROOT_INCLUDE_PATH=/sphenix/tg/tg01/commissioning/INTT/repositories/
macros/common:${ROOT_INCLUDE_PATH}
cd ../../macro
root -q -b 'Fun4All_physiTuto.C(10, 1, 0.2, "Electron", “<outputDir>”, "output.root")''
```

You can find my single electron output examples (but these are heavy, so please take care, if you copy them).
/sphenix/tg/tg01/commissioning/INTT/work/tkumaoka/InttSeedingTrackDev/ParticleGen/output/singleE*MeVEta0.root

How to Run the Tracking Performance Estimation Code

Tracking Performance Code
https://github.com/sPHENIX-Collaboration/INTT/tree/main/general_codes/tkumaoka/InttSeedingTrackDev/InttSeedTrackPerformance

Only type one line

```
source main.sh 0
```

However, you need to setup the input root file and output directory in main.C

This code does not require the Fun4All environment, so you can run it in your laptop.

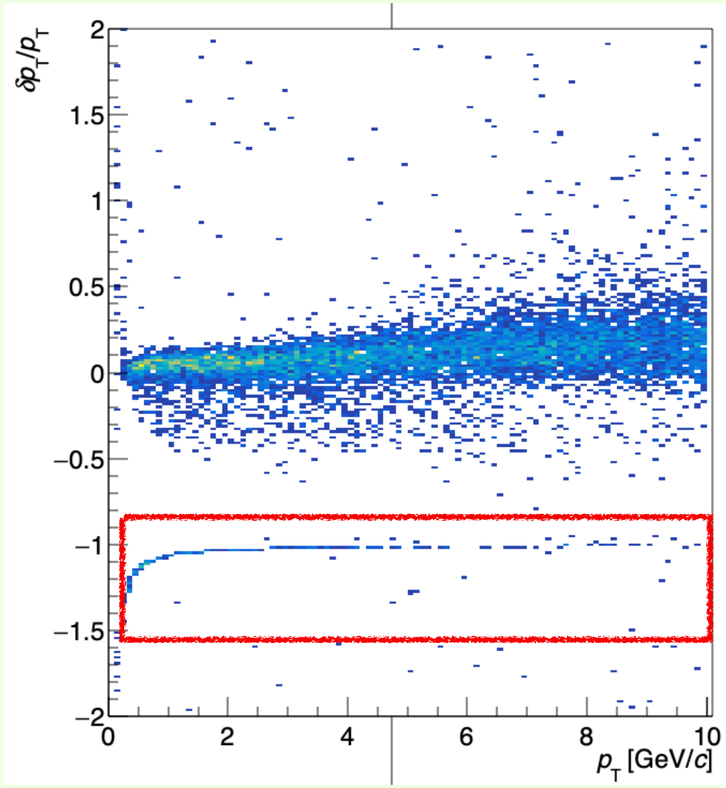
Super Technical Tips

Run Only Interesting Event

1. Scan All Events w/ EventJudge function:
   ex) If you interested in the events having -2 < dpT < -1, you can set following.
   https://github.com/sPHENIX-Collaboration/INTT/blob/main/general_codes/tkumaoka/InttSeedingTrackDev/InttSeedTrackPerformance/src/InttSeedTrackPerformance.cxx#L587

```
614     Double_t dPtOriFun = (OriFunTrackPt - truthPt)/truthPt;
615     m_HTruthPtVsFitFuncPt_IInttOInttEmcal->Fill(truthPt, dPtOriFun,
        truthEta);
616     EventJudge(pubEvNum, dPtOriFun ,-2, -1, true);
```

   The EventJudge function saves the event numbers in the m_vTargetEvents.
   And finally, the event numbers are shown in the last line.



2. Turn on the “bTargetEV = true;”
3. Set the event numbers in the vTargetEvents = {};
4. Set the number of event loop.
5. Check the event displays. The way will be shown the next slide.

Check Event Display (super manual stupid way)

1. Set ShowEventInfo() function and ShowTrackInfo()

```
61        std::cout << "== event No." << jentry << "
          ========================== "<< std::endl;
62        ShowEventInfo();
```

```
598       ShowTrackInfo(trk, dPt, centerX, centerY, sagittaR);
599     }
```

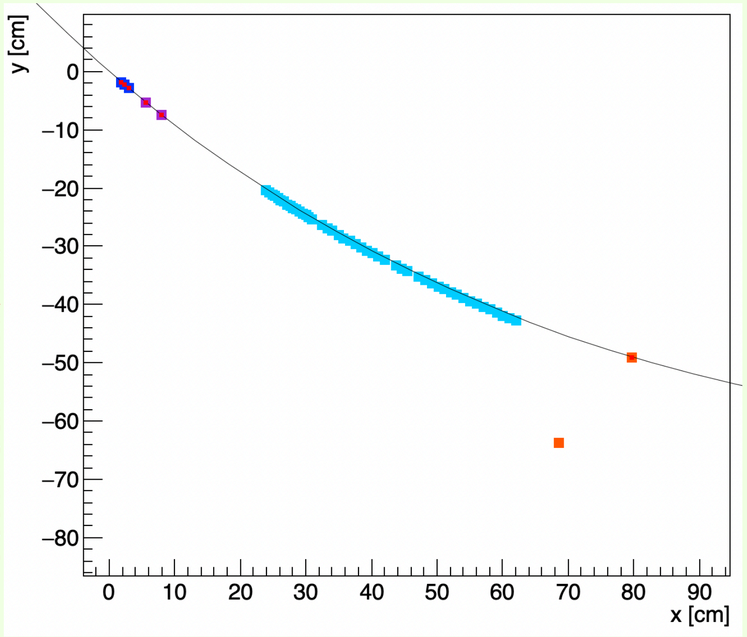
2. This function will show the command-lines to make event display.

```
== event No.35  ==========================
TH2D* hHitMapMvtx = new TH2D( "hHitMapMvtx", "hHitMap", 2000, -100, 1
00, 2000, -100, 100);
TH2D* hHitMapIntt = new TH2D( "hHitMapIntt", "hHitMap", 2000, -100, 1
00, 2000, -100, 100);
TH2D* hHitMapTpc = new TH2D( "hHitMapTpc", "hHitMap", 2000, -100, 100
, 2000, -100, 100);
TH2D* hHitMapEmcal = new TH2D( "hHitMapEmcal", "hHitMap", 2000, -100,
 100, 2000, -100, 100);
hHitMapMvtx->SetLineWidth(8);
hHitMapIntt->SetLineWidth(8);
hHitMapTpc->SetLineWidth(8);
hHitMapEmcal->SetLineWidth(8);
hHitMapMvtx->SetLineColor(860);
hHitMapIntt->SetLineColor(875);
hHitMapTpc->SetLineColor(870);
hHitMapEmcal->SetLineColor(807);
hHitMapMvtx->Fill(2.45271, -0.166589);
hHitMapMvtx->Fill(3.23022, -0.213374);
hHitMapMvtx->Fill(4.03776, -0.259878);
hHitMapIntt->Fill(8.15555, -0.455297);
```

3. Copy&Paste into the root.

> root

※ Don't paste every line once. Your terminal will turn down.

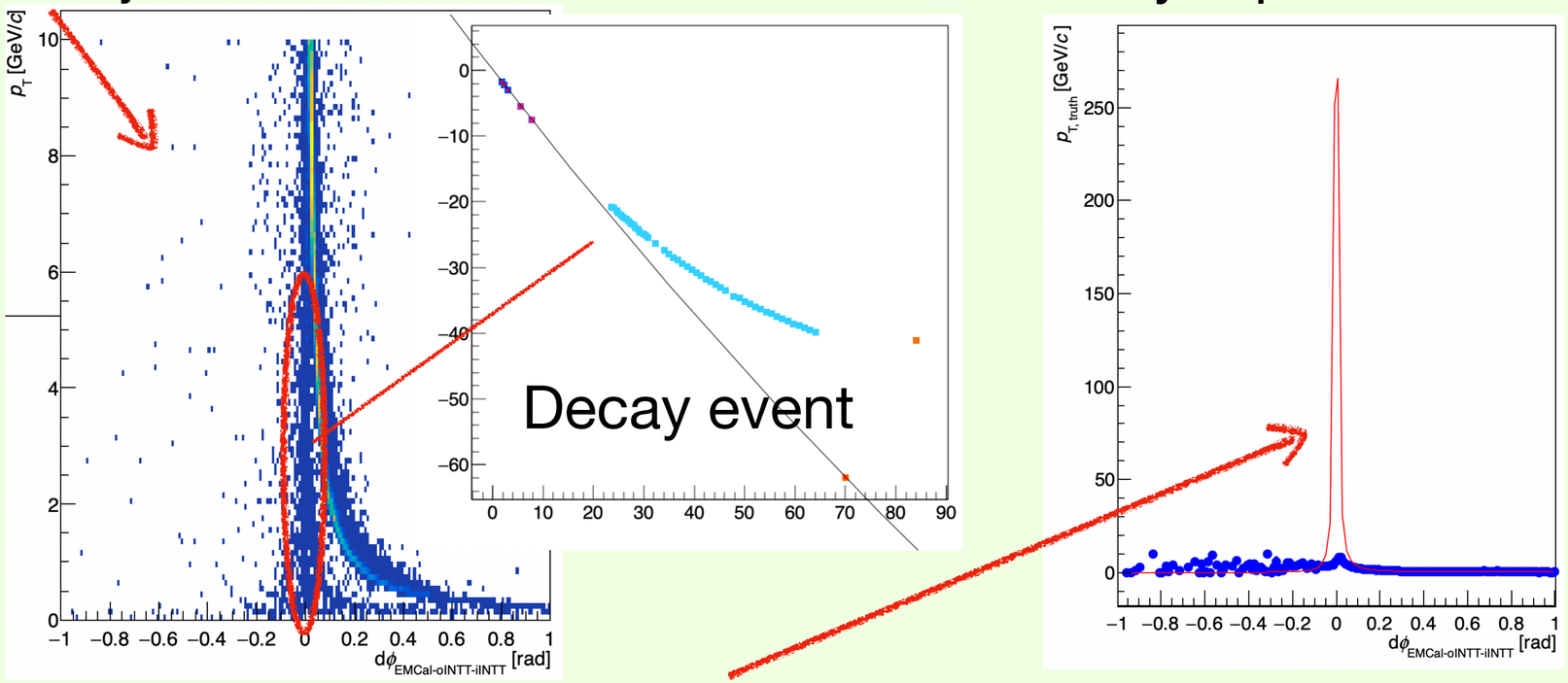


Pros: You can see event-by-event.
You can zoom event because you can check it as root not picture.

Cons: It is super manual & it is not good for multi-pariticles events.

Way to get the fitting function $d\phi - dp_{\mathrm{T}}$ (super manual stupid way)

1. Project “m_HDPhiVsTruthPtVsEta_*” on a y-x plane.



2. Profiling the 2D hist w/ BKG makes bad dPhi distribution.
And the automatic fitting also does not work well.

```
TF1* fDPhiDPt = new TF1("dPhiDPt", “[0]+[1]/x+[2]/(x*x)"
m_HDPhiVsTruthPtVsEta_IInttOInttEmcal_yx_pfx->Fit(fDPhiDPt)
```

3. In the next slide, some my solutions will be shown. (I think there is more sophisticated one…)

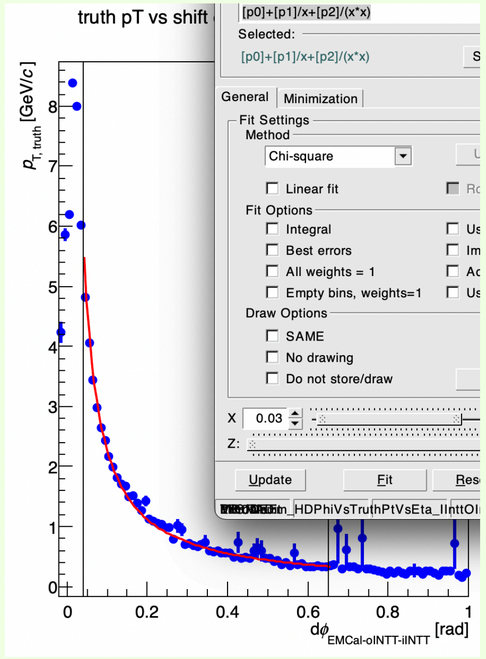
Way to get the fitting function $d\phi - dp_{\mathrm{T}}$ (super manual stupid way)

3. There are three my solutions. (I think there is more sophisticated one…)
   3.1 Select fit region manually to remove background region.
   3.2 Use cut function or TCutG. (The projection seems more clean, but automatic fit also does not work well.)

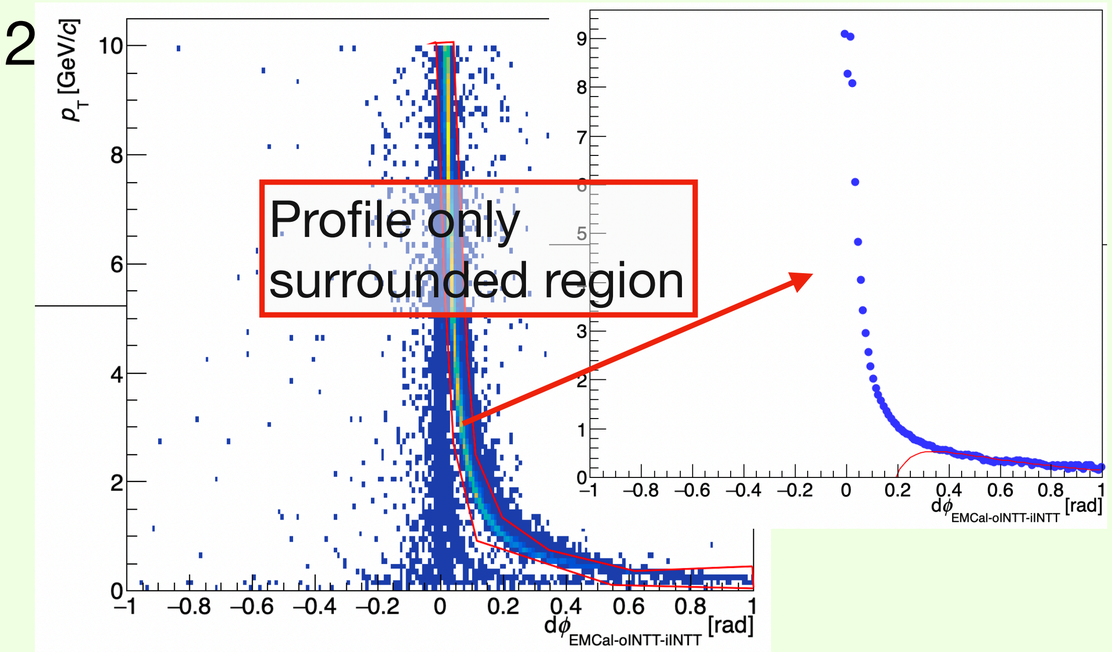
```
((TCutG*)gROOT->GetListOfSpecials()->FindObject("CUTG"))->SetName("cut1");
m_HDPhiVsTruthPtVsEta_IInttOInttEmcal_yx->ProfileX("prof1", 1, -1, "[cut1]")
```

3.3 Use event cut for decay events. However, my cut dramatically reduced statistics…
→ But the event cut solution has no human bias. So I suggest to optimize the event cut.

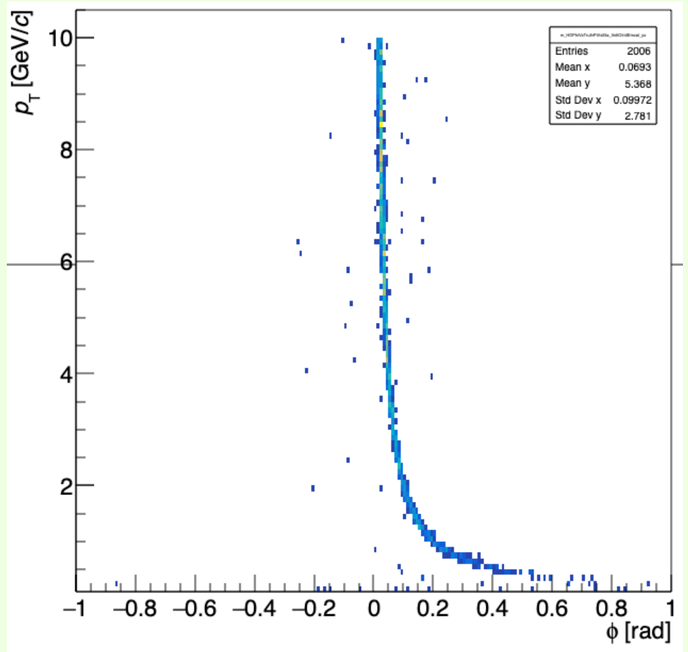
3.1



3.2



3.3

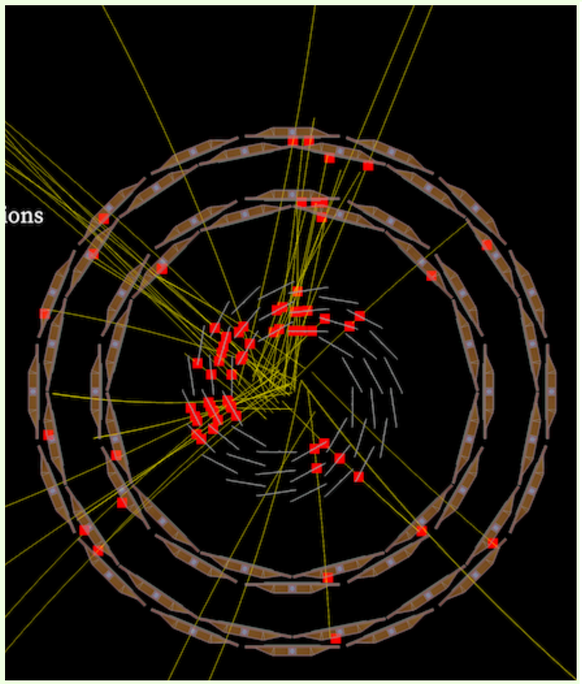


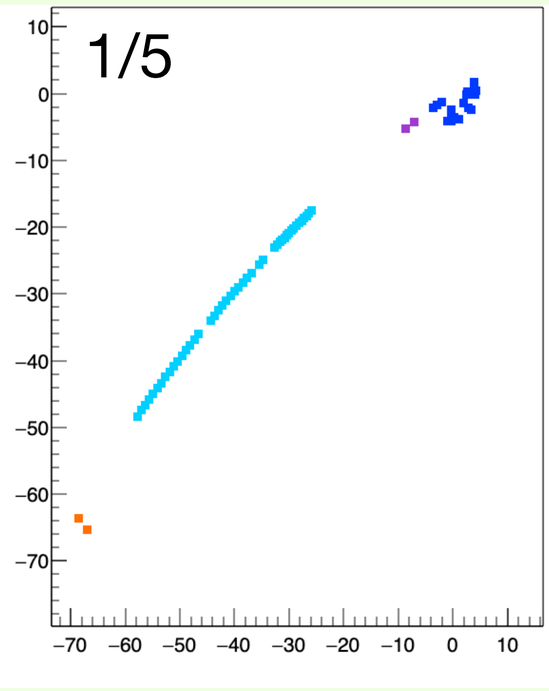
The Way to Estimate Performance in the Mixed Events for MVTX

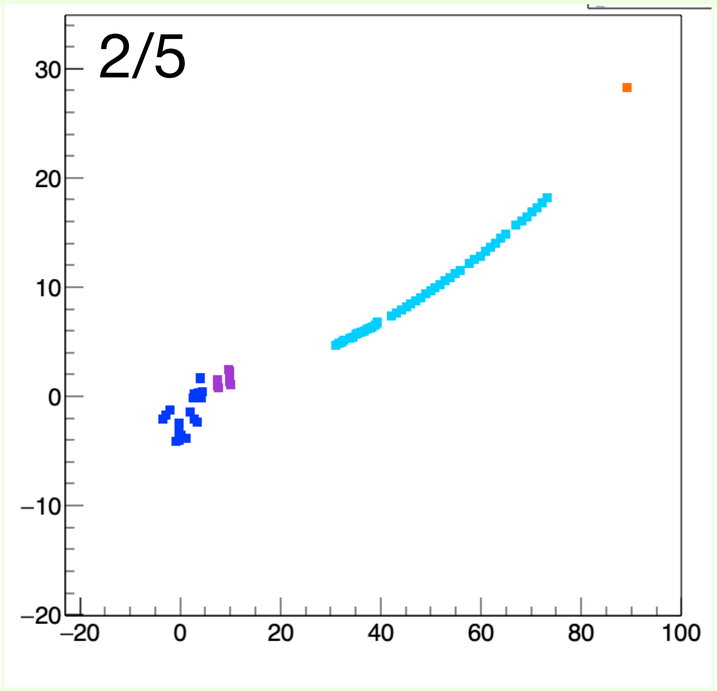
The time resolution of MVTX is not enough to distinguish each event. (left figure)
Therefore, within the sPHENIX concept, the INTT is used to select MVTX hits from certain events.

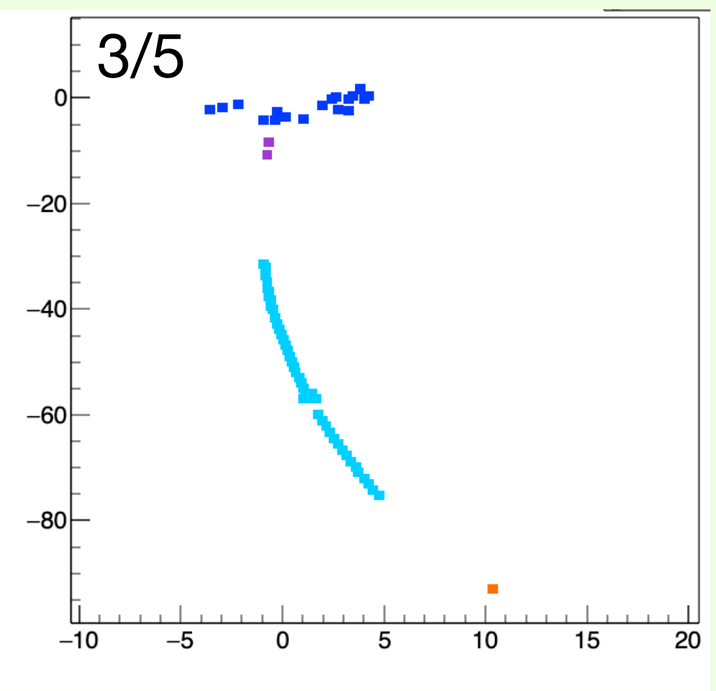
We need to check that the tracking performances are kept in the such mixed up situations.

If you set stockEvents to a high number (e.g., 5), you can test such situations.
The three event displays show that situation, but you need to test it or PYTHIA.







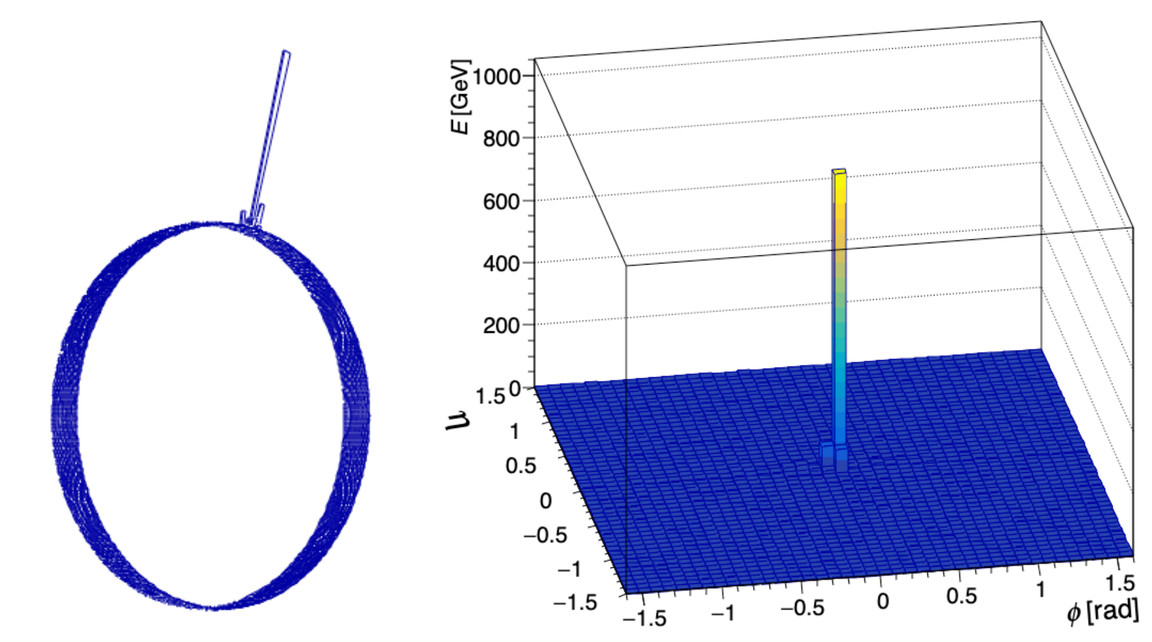


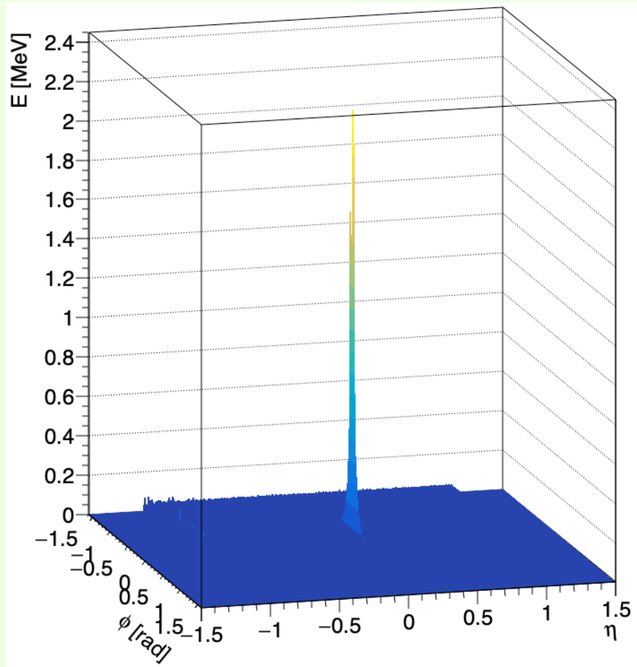
Calo Study

The bCaloClus variable can switch the calo information.
My input DST root file includes calo information both tower and cluster.
Then, on the following lines, we can get the calo information.

```
58          if(bCaloClu) ReadCalCluHitting(m_emcalHits, m_iHCalHits, m_oHCalHits);
59          else ReadCalHitting(m_emcalHits, m_iHCalHits, m_oHCalHits);
```

If you want to study clustering for tracking, I recommend using towers because the calo group clustering does not seem to be considering using tracking.





New Algorithm

By only comment-out the one algorithm lines, you can use.

InttSeedTracking.cxx

```
17    // reco way1
18    // HitMatching(tracks, vFMvtxHits, vSMvtxHits, vTMvtxHits, vIInttHits, vOInttHits,\
19    //    vEmcalHits, vIHCalHits, vOHCalHits);
20    // for(Int_t iTrk = 0; iTrk < tracks.size(); iTrk++){
21    //    TrackPropertiesEstimation(tracks.at(iTrk), vFMvtxHits, vSMvtxHits, vTMvtxHits);
22    // }
23
24    // reco way2
25    RecoTracksInttSeed2(tracks, vFMvtxHits, vSMvtxHits, vTMvtxHits,\
26       vIInttHits, vOInttHits, vEmcalHits, vIHCalHits, vOHCalHits);
27    for(Int_t iTrk = 0; iTrk < tracks.size(); iTrk++){
28       TrackPropertiesEstimation2(tracks.at(iTrk));
29    }
```

Previous

New