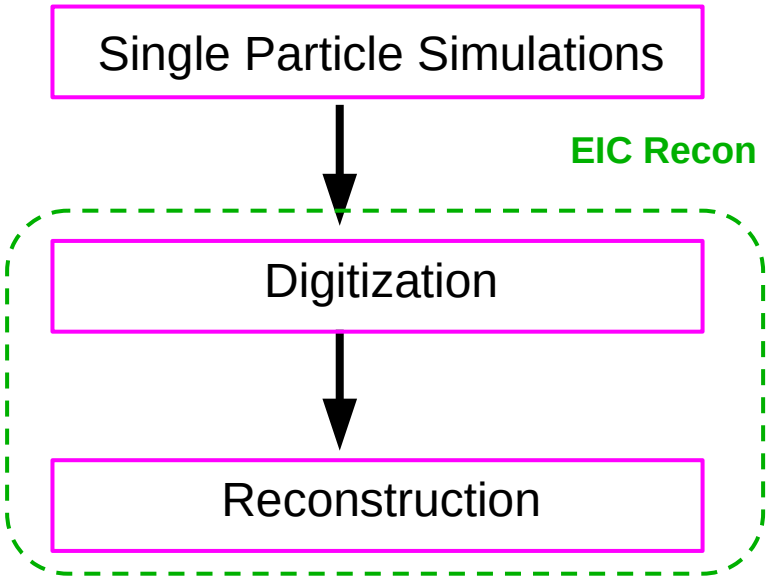
Summer 2024 Joint EICUG/ePIC Collaboration Meeting

Single-particle tracking benchmarks

Shyam Kumar*, Annalisa Mastroserio, Domenico Elia
INFN Bari, Italy

Simulation and Reconstruction

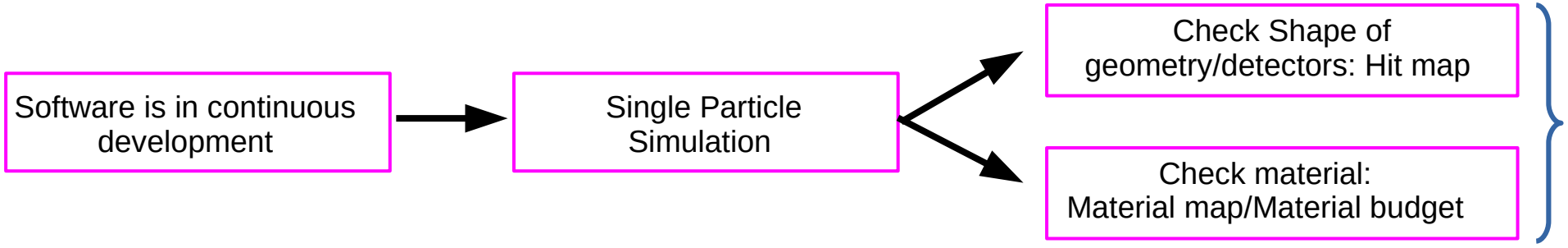


DD4HEP and GEANT4 (Particle Gun): simulation and propagation using Geant4: Hits using geometry and material information

Realistic detector response: starting from energy loss, charge creation, distribution of charge, cluster position reconstruction, etc.

Tracking in EIC Recon using ACTS (A Common Tracking Software): Momentum, distance of closest approach (DCA) resolutions

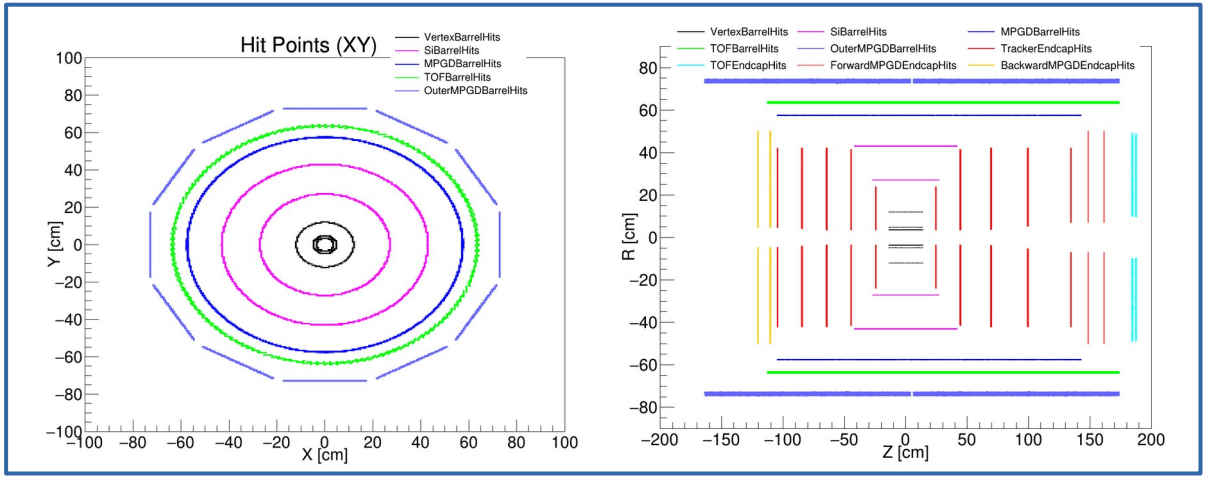
Reconstruction uses the information from simulation



Single Particle Simulations

Check geometry/detector surfaces: Hit map (XY, RZ) (**Generated level**)

Not included yet



Spotted missing hits in MPGD disks: reported in tracking meeting

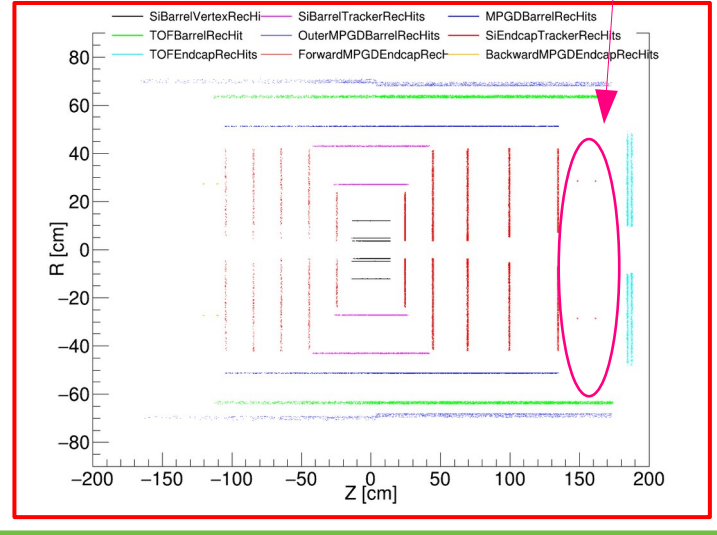
The plot can be produced on the fly for one momentum while running ddsim command

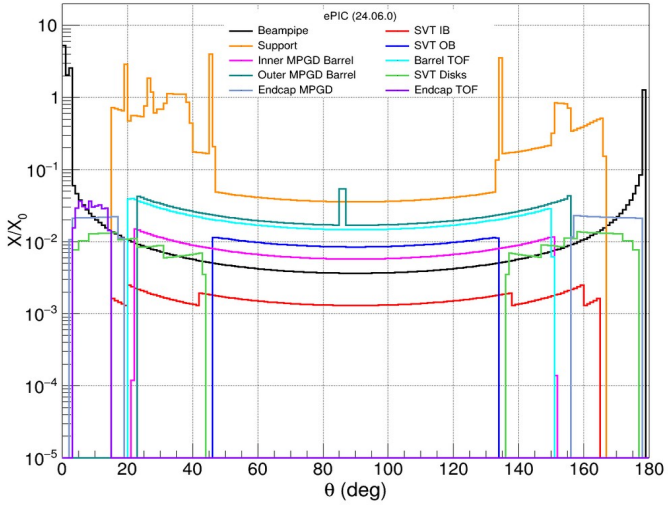
Code in EICRecon

Missing hits

September simulation campaign

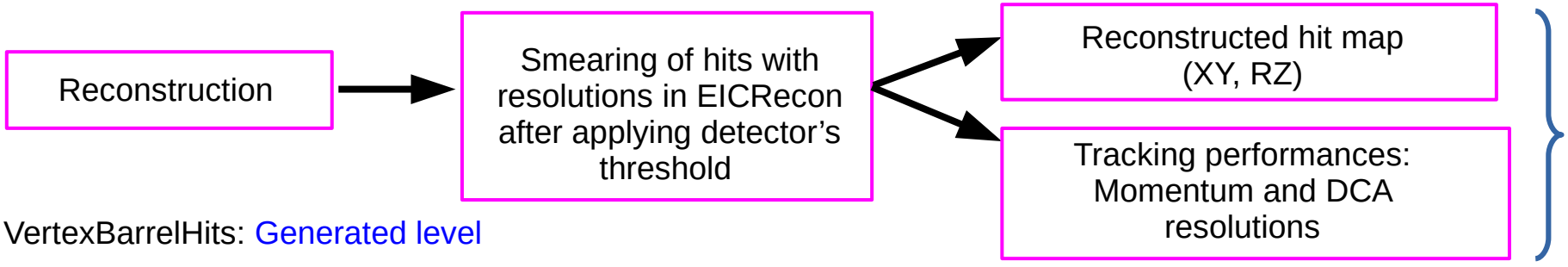
Reco Hits





Check material:
Material budget (absolute values)

Reconstruction



VertexBarrelHits: Generated level

VertexBarrel**Rec**Hits: Reconstructed level

Reconstructed hits are used in tracking

Tracking performances: Wrote a few scripts to perform automatic single particle simulation and reconstruction

Three steps:

- Local script (bash script) as an initial step
- Less events (10k) on git (validation)
- Simulation Campaigns

Simulation (Local script for testing before commit):

mom_array=(0.5 1.0 2.0 5.0 10.0 20.0) We can further add/remove values

nevents =10k

Latest geometry

Uniform in eta (range taken from thetaMin to thetaMax) Particle name

```
for ((i=0; i<${#mom_array[@]}; i++)); do
npsim --compactFile epic_craterlake_tracking_only.xml --outputFile sim${mom_array[i]}.edm4hep.root --numberOfEvents $nevents
--enableGun --gun.thetaMin 3*deg --gun.thetaMax 177*deg --gun.distribution eta --gun.particle pi- --gun.momentumMin $
{mom_array[i]}*GeV --gun.momentumMax ${mom_array[i]}*GeV --gun.multiplicity 1 --random.seed 100000
done
```

Momentum

Seed (keeping same seed)

Reconstruction

Reconstruction (EICRecon):

```
for ((i=0; i<${#mom_array[@]}; i++)); do
eicrecon \
 -Pnthreads=1 \
 -Pjana:debug_plugin_loading=1 \
 -Pjana:nevents=$nevents \
 -Pacts:MaterialMap=calibrations/materials-map.cbor \
 -Ppodio:output_file="${filename}"_${mom_array[i]}.edm4eic.root \
 -Pdd4hep:xml_files= epic_craterlake_tracking_only.xml \
 -Ppodio:output_collections="MCParticles,CentralCKFTrajectories,CentralCKFTrackParameters,CentralCKFSeededTrackParameters,CentralTrackVertices" \
 sim${mom_array[i]}.edm4hep.root
done
```

The simulation and reconstruction run with an automatic script (bash) to extract performances

Selected Output collection for the tree

Simulation and Reconstruction for benchmarks (Less events on git and simulation campaign):

Benchmark

Similar idea has been implemented to the benchmarks: only modification is simulation and reconstruction commands are taken from official simulation campaign (After a discussion with Dmitry Kalinkin and Torri)

GNU: Makefile (compile) → Python: Snakemake Concept is similar: target-rule (command)

Snakefile setting: Snakefile

MOMENTUM=[0.5, 1.0, 2.0, 5.0, 10.0, 20.0]

SEEDING=["real", "truth"],

PHASE_SPACE="(**3to50|45to135|130to177**)deg",

N_EVENTS=10000

If there are some changes in the software: Run local on git with 10k events and check performances to make sure everything is fine

Tracking Performances

Performances:

Tracking_Performances.C

It automatically reads the reconstruction output and stores the plots for momentum resolutions $(p_{rec}-p_{true})/p_{true}$

```
// MC and Reco information
TTreeReaderArray<Float_t> charge(myReader, "MCParticles.charge");
TTreeReaderArray<Double_t> vx_mc(myReader, "MCParticles.vertex.x");
TTreeReaderArray<Double_t> vy_mc(myReader, "MCParticles.vertex.y");
TTreeReaderArray<Double_t> vz_mc(myReader, "MCParticles.vertex.z");
TTreeReaderArray<Float_t> px_mc(myReader, "MCParticles.momentum.x");
TTreeReaderArray<Float_t> py_mc(myReader, "MCParticles.momentum.y");
TTreeReaderArray<Float_t> pz_mc(myReader, "MCParticles.momentum.z");
TTreeReaderArray<Int_t> status(myReader, "MCParticles.generatorStatus");
TTreeReaderArray<Int_t> pdg(myReader, "MCParticles.PDG");
TTreeReaderArray<Int_t> match_flag(myReader, Form("CentralCKF%sTrackParameters.type",name.Data()));
TTreeReaderArray<Float_t> d0xy(myReader, Form("CentralCKF%sTrackParameters.loc.a",name.Data()));
TTreeReaderArray<Float_t> d0z(myReader, Form("CentralCKF%sTrackParameters.loc.b",name.Data()));
TTreeReaderArray<Float_t> theta(myReader, Form("CentralCKF%sTrackParameters.theta",name.Data()));
TTreeReaderArray<Float_t> phi(myReader, Form("CentralCKF%sTrackParameters.phi",name.Data()));
TTreeReaderArray<Float_t> qoverp(myReader, Form("CentralCKF%sTrackParameters.qOverP",name.Data()));
```

MC Information

Reco Information (truth/real seed)

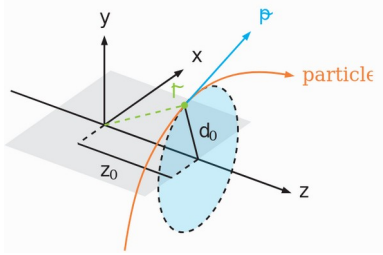
Track Parameters at given surface (real or perigee) $\{l_0, l_1, \theta, \phi, q/p\}$

p = std::abs(1.0 /(q/p));

p_T = p sin θ;

η = std::atanh(cos(θ));

ACTS example



1d histogram $((p_{rec}-p_{true})/p_{true})$ for each eta bin and momentum see backup

We want DCA as a function of p_T (Storing 3D histogram)

```
h_d0xy_3d->Fill(d0xy[j]*0.1, etamc, ptmc); // cm
h_d0z_3d->Fill(d0z[j]*0.1, etamc, ptmc); // cm
```

Tracking Performances

doCompare_truth_real_widebins_mom.C

Momentum Resolutions:

- Fitting 1D Gaussian distributions for each momentum and η range and storing **debug_plots/**
- Fitting done in two steps to select the core region and avoiding long tails specially at low momentum

```
double mu_truth = hist_truth->GetMean();
double sigma_truth = hist_truth->GetStdDev();
func_truth->SetRange(mu_truth-2.0*sigma_truth,mu_truth+2.0*sigma_truth); // fit with in 2 sigma range
hist_truth->Fit(func_truth,"NR+");
mu_truth = func_truth->GetParameter(1);
sigma_truth = func_truth->GetParameter(2);
func_truth->SetRange(mu_truth-2.0*sigma_truth,mu_truth+2.0*sigma_truth);
```

Benchmark plots

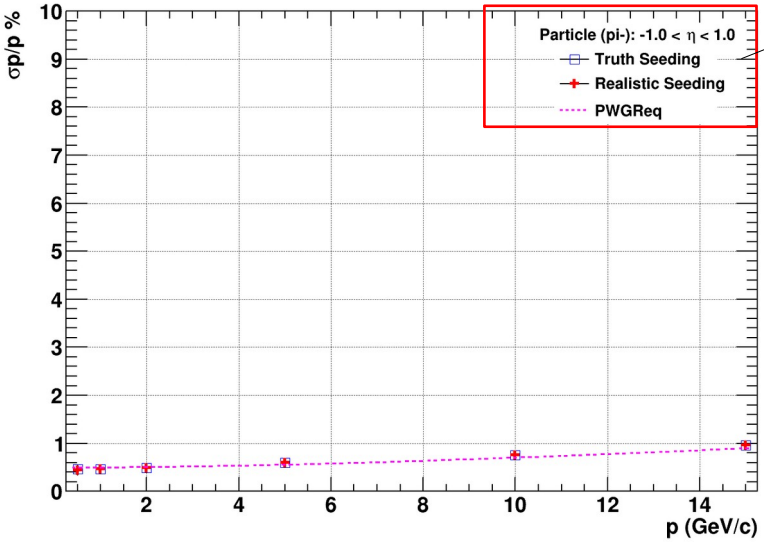
- Storing final plots in root files (also in png) with multigraph: can be modified to include in TDR
- The plots are produced automatically with the latest software
- If something looks strange: check 1D Gaussian plots to understand more

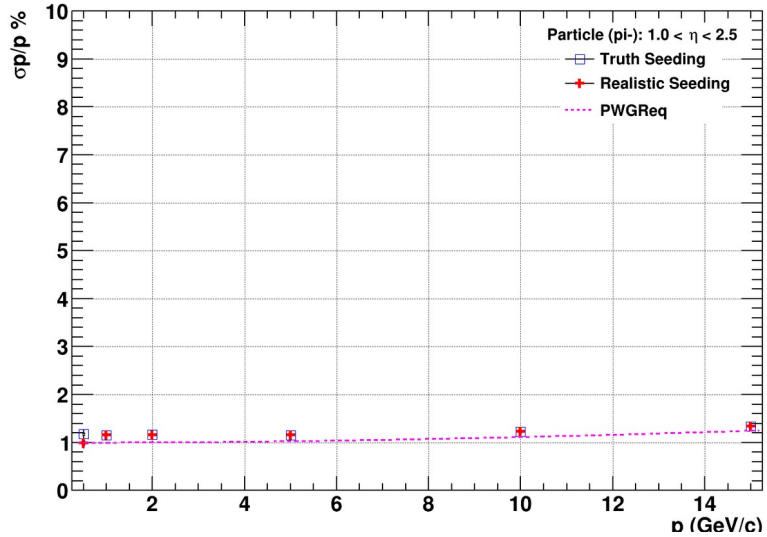
Momentum Resolutions

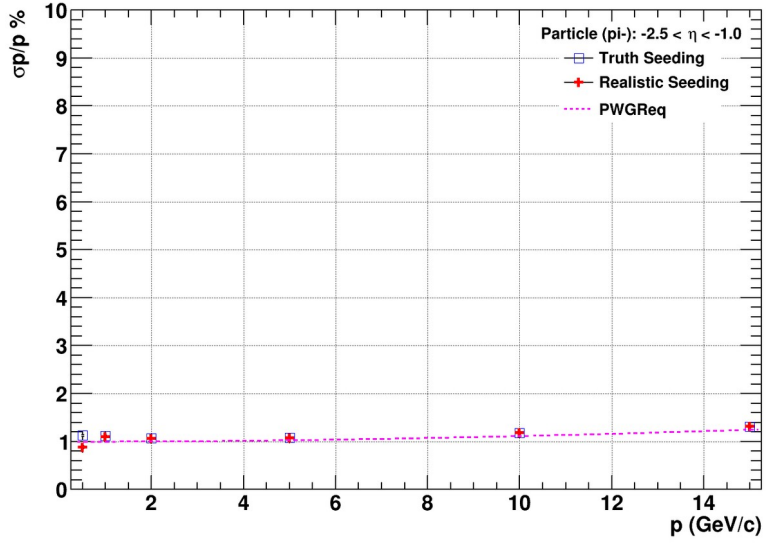
Local on git with 10k events (validation)

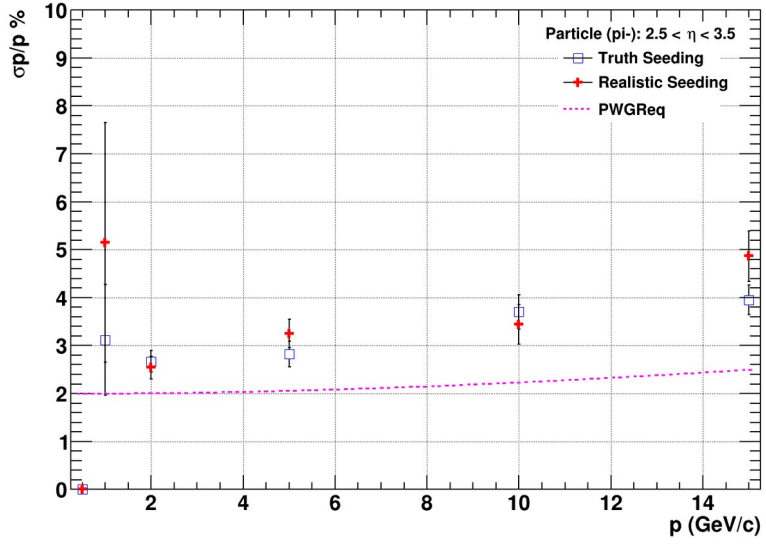
Particle (pi-)--->π^- ePIC (24.06.0/v1.14.0) (software version)

Updated code



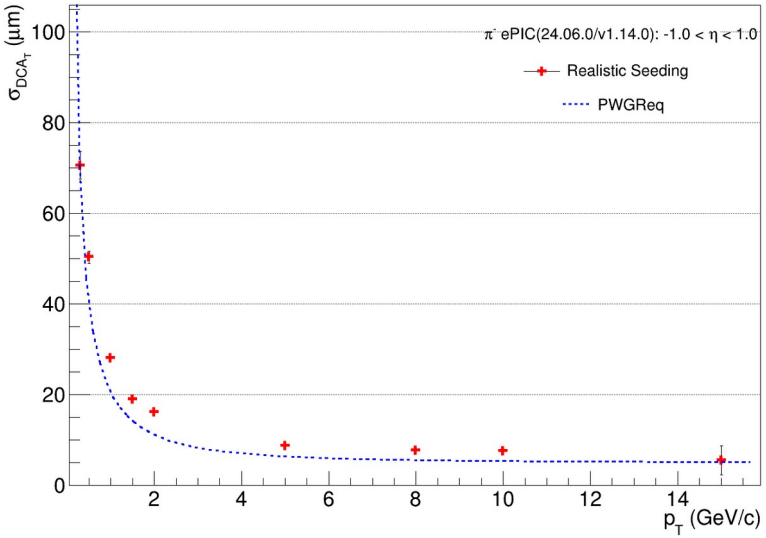


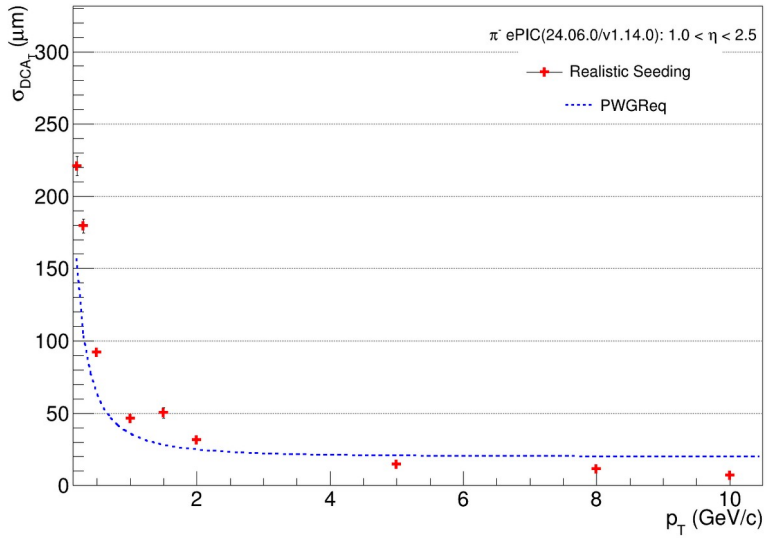


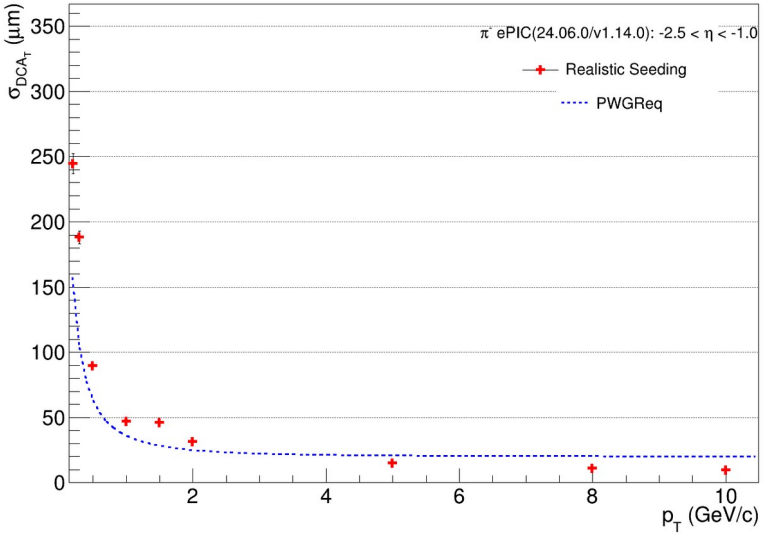


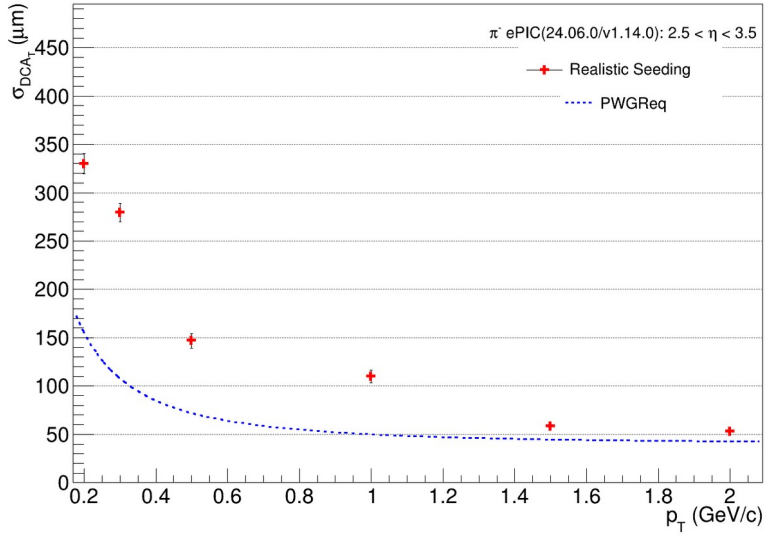
DCA_T Resolutions

Local desktop (testing phase will be merged on git soon)









DCA_T Resolutions (Text file)

ePIC	24.06.0	EICRecon	v1.14.0
Etamin	Etamax	Pt (GeV/c)	Resol #mum (Real)
-1	1	0.3	73.1379
-1	1	0.5	51.8567
-1	1	1	27.9246
-1	1	1.5	19.2027
-1	1	2	15.945
-1	1	5	8.94601
-1	1	8	7.6873
-1	1	10	7.71712
-1	1	15	5.51936
ePIC	24.06.0	EICRecon	v1.14.0
Etamin	Etamax	Pt (GeV/c)	Resol #mum (Real)
-2.5	-1	0.2	255.467
-2.5	-1	0.3	185.541
-2.5	-1	0.5	91.4802
-2.5	-1	1	47.3861
-2.5	-1	1.5	44.4547
-2.5	-1	2	30.9392
-2.5	-1	5	15.2223
-2.5	-1	8	10.1541
-2.5	-1	10	9.51886
ePIC	24.06.0	EICRecon	v1.14.0
Etamin	Etamax	Pt (GeV/c)	Resol #mum (Real)
1	2.5	0.2	229.771
1	2.5	0.3	186.687
1	2.5	0.5	92.6851
1	2.5	1	44.8082
1	2.5	1.5	47.1238
1	2.5	2	30.3747
1	2.5	5	14.5067
1	2.5	8	11.8879
1	2.5	10	8.74126
ePIC	24.06.0	EICRecon	v1.14.0
Etamin	Etamax	Pt (GeV/c)	Resol #mum (Real)
-3.5	-2.5	0.2	388.891
-3.5	-2.5	0.3	307.228
-3.5	-2.5	0.5	160.978
-3.5	-2.5	1	115.733
-3.5	-2.5	1.5	65.7672
-3.5	-2.5	2	57.4064
ePIC	24.06.0	EICRecon	v1.14.0

The current version of code also produces similar files for momentum, and DCAz resolutions

We can even fit the distributions using fit function already in the code

Simulation Campaigns

Running the code on all simulation campaigns using Snakefile

Snakefile Campaigns

```
136     rule tracking_performance_campaigns:
137         input:
138             expand(
139                 "results/tracking_performances/{CAMPAIGN}",
140                 CAMPAIGN=[
141                     "23.10.0",
142                     "23.11.0",
143                     "23.12.0",
144                     "24.03.1",
145                     "24.04.0",
146                 ],
147             )
```

At the moment, only momentum resolutions soon will also be DCA resolutions

Download

passed Job #3541344 in pipeline #98229 for c46707ed from pr/tracking_performance_campaigns by Wouter Deconinck 1 day ago

Artifacts / results / tracking_performances

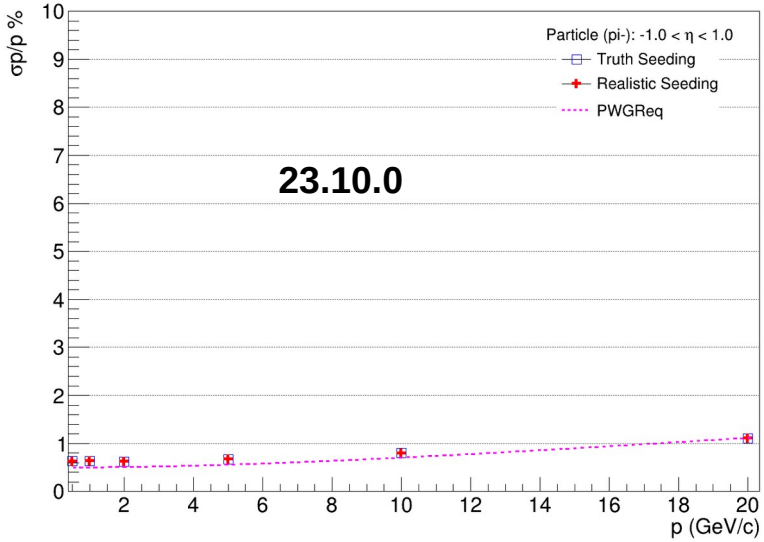
Download artifacts archive

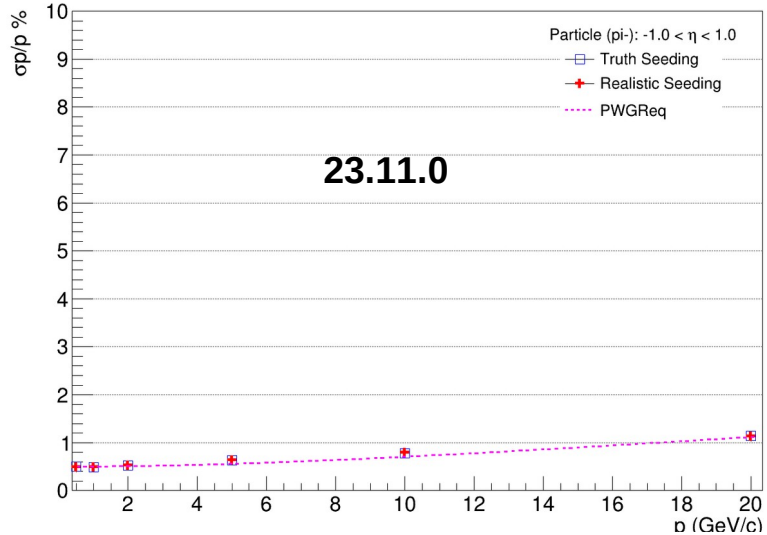
Name	Size
..	
23.10.0	
23.11.0	
23.12.0	
24.03.1	
24.04.0	

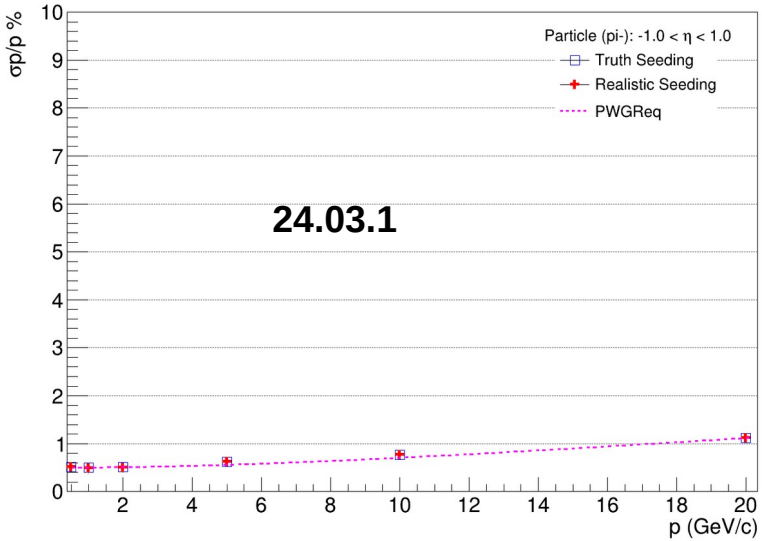
Results

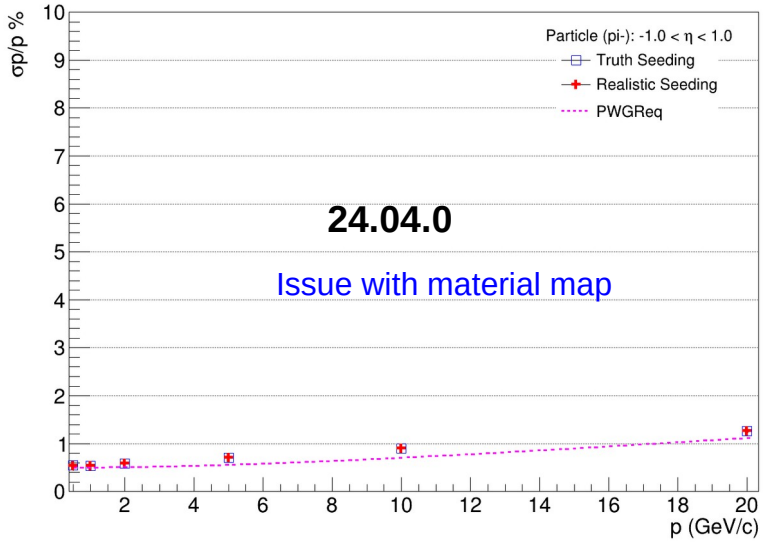
Simulation Campaigns (Results)

-1.0 < η < 1.0



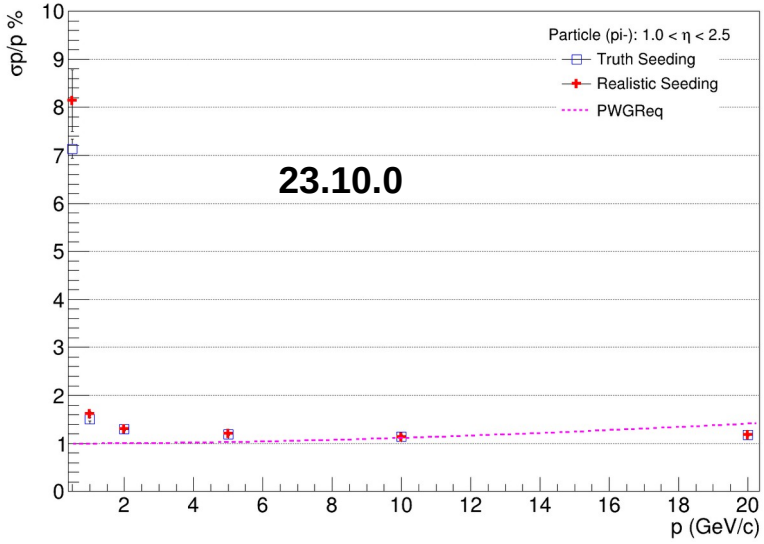


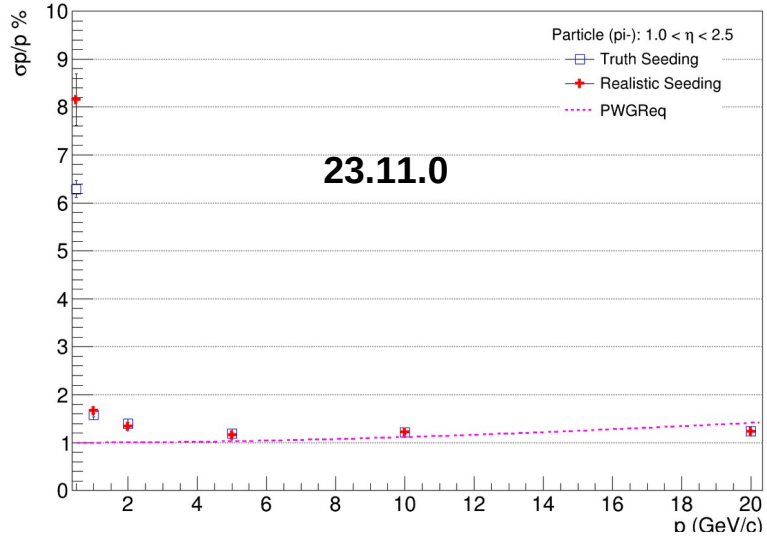


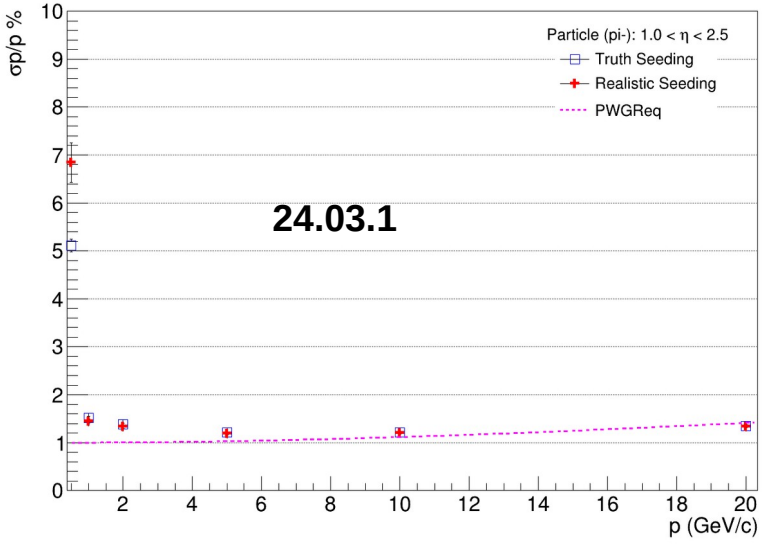


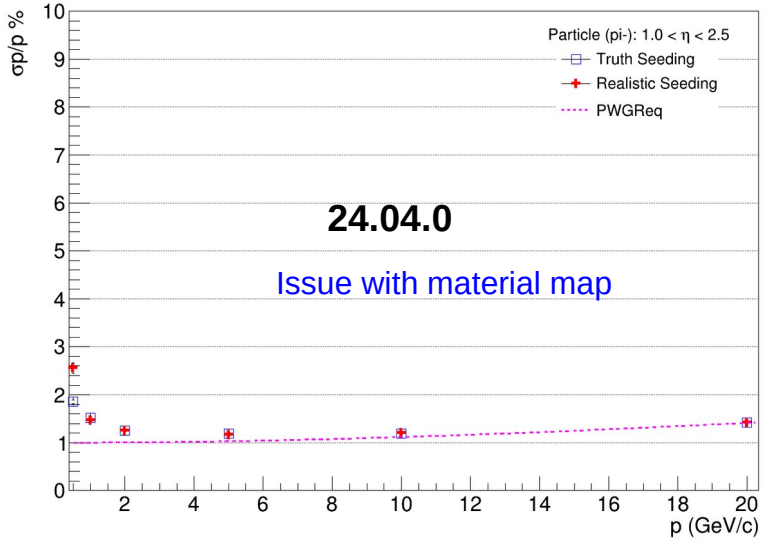
Simulation Campaigns (Results)

1.0 < η < 2.5 If there is some issue we can check one dimensional Gaussian distributions (debug plots)

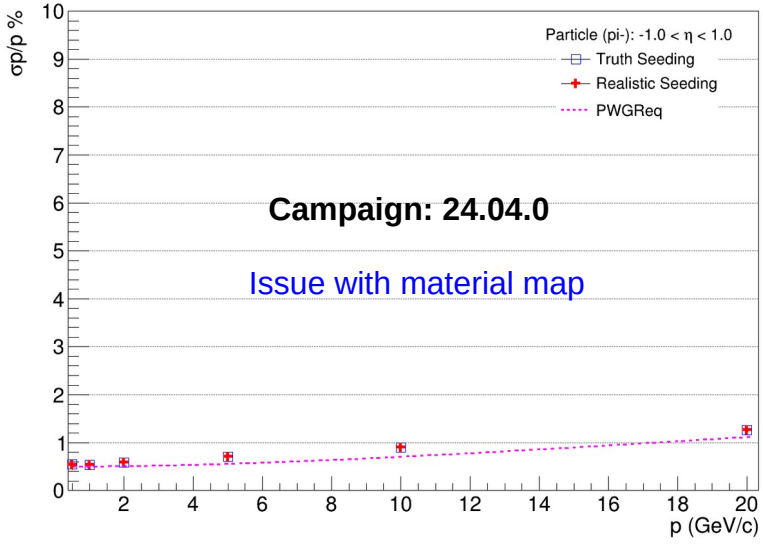




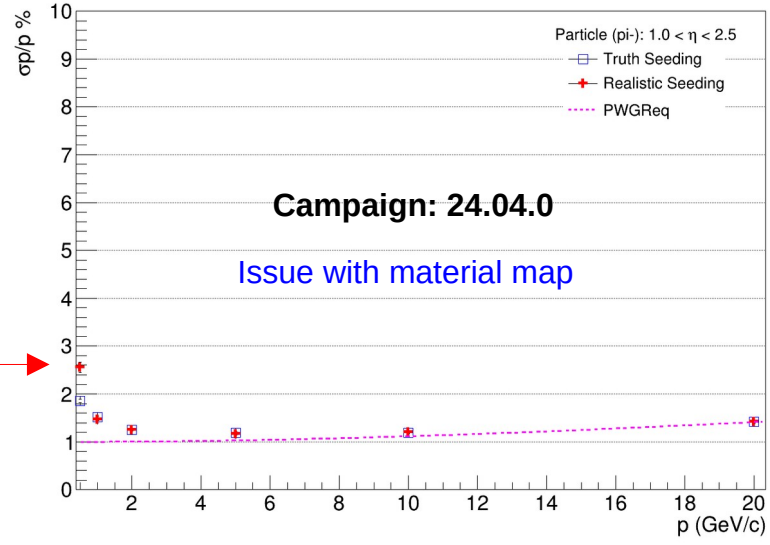


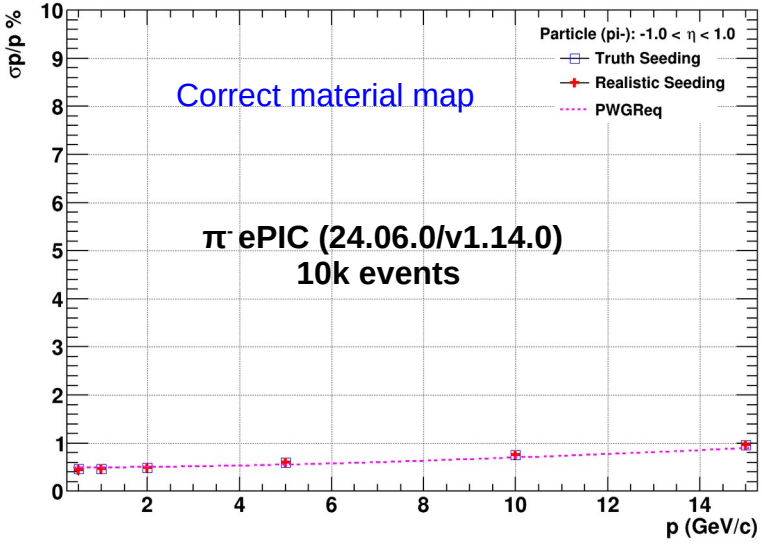


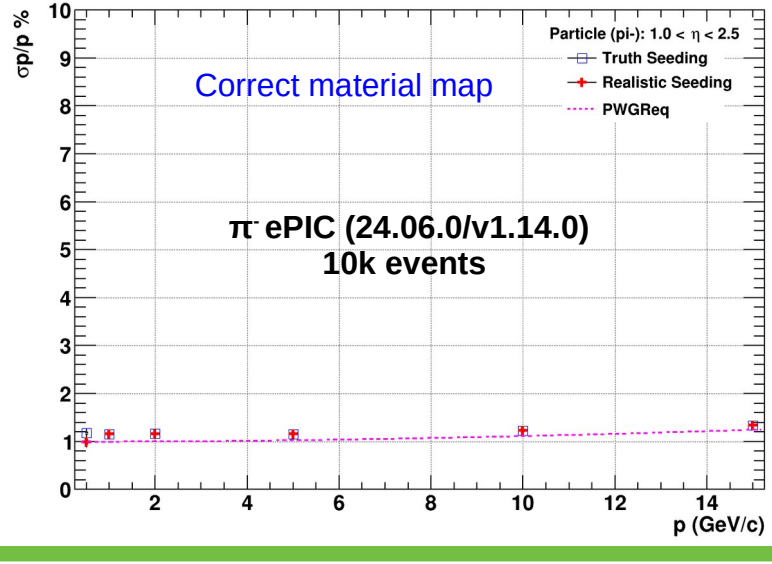
Momentum Resolutions



Higher →







Summary

- The automated tracking benchmarks are a significant step for the ePIC software
- Once there is a major update (geometry/software), the local simulation on git (10k events) with latest changes will produce the result
- Further will include the script for detector hit maps
- Working with Torri to further add these plots to the image browser
- In future will also add the comparison with some reference plots

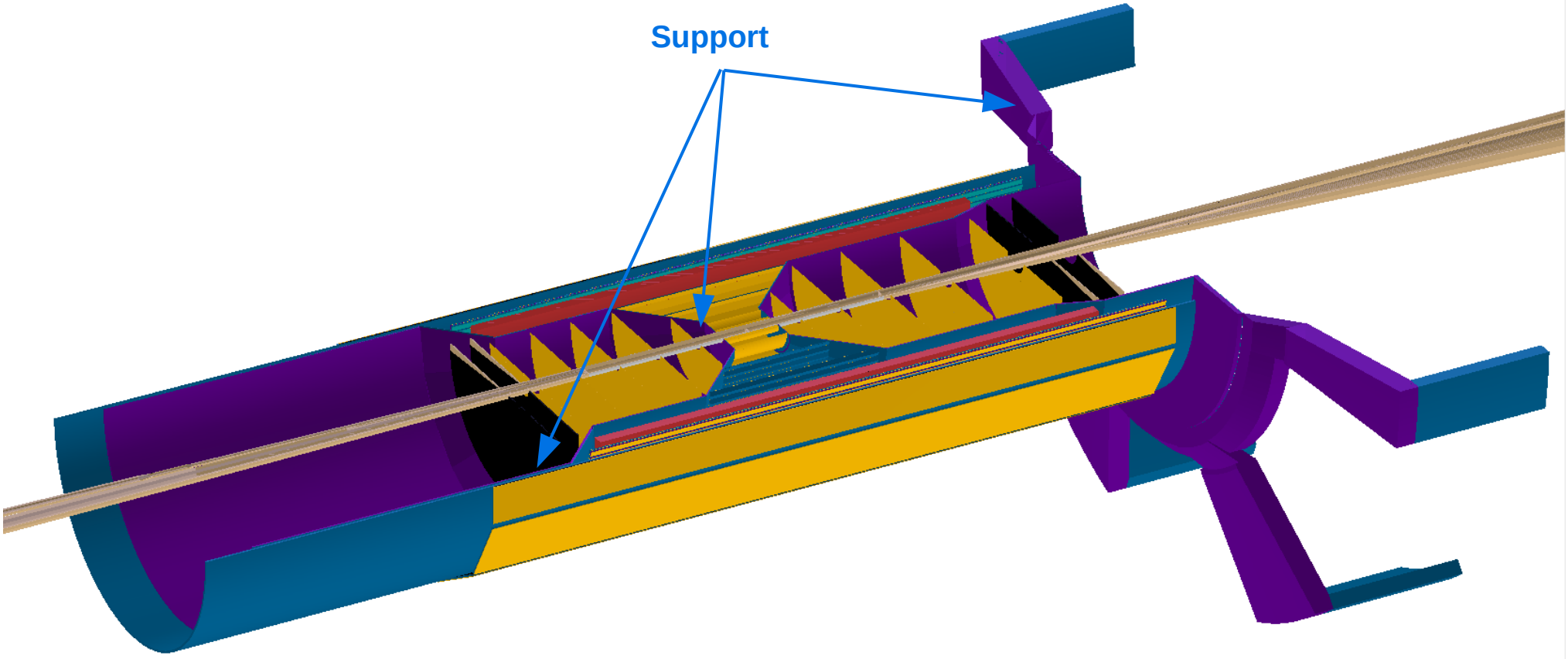
Image Browser

Thanks Dmitry Kalinkin and Torri for help and discussion

Thank You !!

ePIC Tracker (24.06.0)

epic_craterlake_tracking_only.xml



There is an extra support layer before OuterMPGD: Additional multiple scattering

Code to draw momentum resolution from root files

```
void draw_req_Mom(double etamin, double etamax, double xmin=0., double xmax=0.);
void draw_mom(TString particle = "pi-", double etamin=-1.0, double etamax=1.0){
   gStyle->SetPalette(1);
   gStyle->SetOptTitle(1);
   gStyle->SetTitleOffset(1.0,"XY");
   gStyle->SetTitleSize(.04,"XY");
   gStyle->SetLabelSize(.04,"XY");
   gStyle->SetHistLineWidth(2);
   gStyle->SetOptFit(1);
   gStyle->SetOptStat(1);

  TCanvas *c_mom = new TCanvas("cmom","cmom",1400,1000);
  c_mom->SetMargin(0.10, 0.05 ,0.1,0.05);
  c_mom->SetGrid();

  TFile *f = TFile::Open(Form("mom_resol_%1.1f_eta_%1.1f.root",etamin,etamax));
  TMultiGraph *mg = (TMultiGraph*)f->Get(Form("mom_resol_%1.1f_eta_
%1.1f",etamin,etamax));
  TGraphErrors *gr = (TGraphErrors*) mg->GetListOfGraphs()->At(1);
  c_mom->cd();
  gr->Draw("AP");
  gr->GetXaxis()->SetRangeUser(0.45,15.2);
  gr->GetYaxis()->SetRangeUser(0.,2.0*TMath::MaxElement(gr->GetN(),gr->GetY()));
  draw_req_Mom(etamin,etamax,0.,gr->GetXaxis()->GetXmax());

  TLegend *lmom = new TLegend(0.70,0.80,0.90,0.93);
  lmom->SetTextSize(0.03);
  lmom->SetBorderSize(0);
  lmom->SetHeader(Form("Particle (%s): %1.1f < #eta <
%1.1f",particle.Data(),etamin,etamax),"C");
  lmom->AddEntry(gr,"Real Seed");
  lmom->Draw("same");
}
```

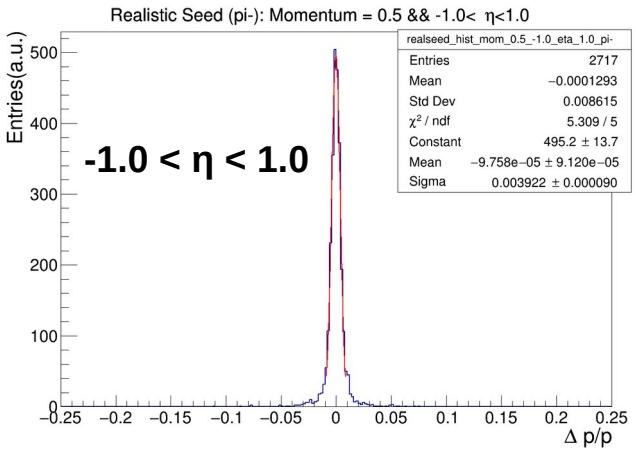
```
void draw_req_Mom(double etamin, double etamax, double xmin=0.,
double xmax=0.)
{

   TF1 *dd4hep_p;
   if (etamin >= -3.5 && etamax <= -2.5) dd4hep_p = new
TF1("dd4hep_p", "TMath::Sqrt((0.1*x)^2+2.0^2)",xmin,xmax);
   else if (etamin >= -2.5 && etamax <= -1.0) dd4hep_p = new
TF1("dd4hep_p", "TMath::Sqrt((0.05*x)^2+1.0^2)",xmin,xmax);
   else if (etamin >= -1.0 && etamax <= 1.0) dd4hep_p = new
TF1("dd4hep_p", "TMath::Sqrt((0.05*x)^2+0.5^2)",xmin,xmax);
   else if (etamin >= 1.0 && etamax <= 2.5) dd4hep_p = new
TF1("dd4hep_p", "TMath::Sqrt((0.05*x)^2+1.0^2)",xmin,xmax);
   else if (etamin >= 2.5 && etamax <= 3.5) dd4hep_p = new
TF1("dd4hep_p", "TMath::Sqrt((0.1*x)^2+2.0^2)",xmin,xmax);
   else return;
   dd4hep_p->SetLineStyle(7);
   dd4hep_p->SetLineColor(kMagenta);
   dd4hep_p->SetLineWidth(3.0);
   dd4hep_p->Draw("same");

  TLegend *l= new TLegend(0.70,0.75,0.90,0.80);
  l->SetTextSize(0.03);
  l->SetBorderSize(0);
  l->AddEntry(dd4hep_p,"PWGReq","l");
  l->Draw("same");
 }
```

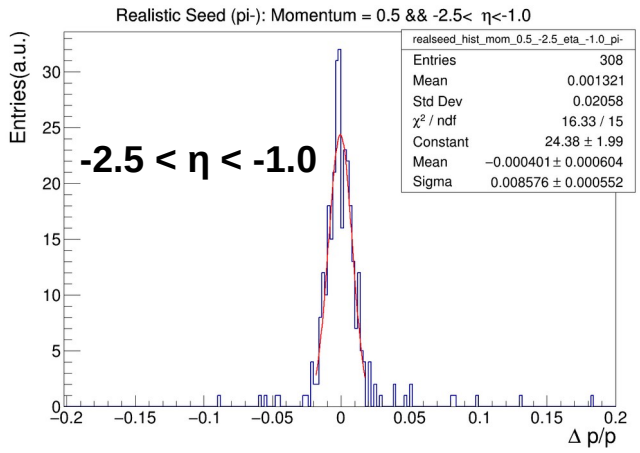
Momentum Resolutions (Debug Plots)

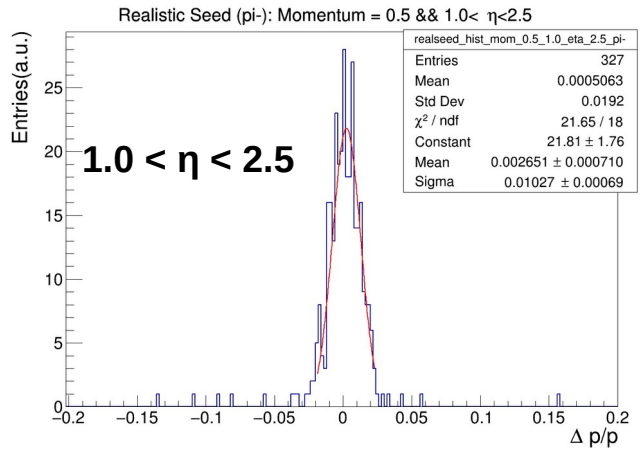
10k Events

Debug plots for each η and momentum are produced

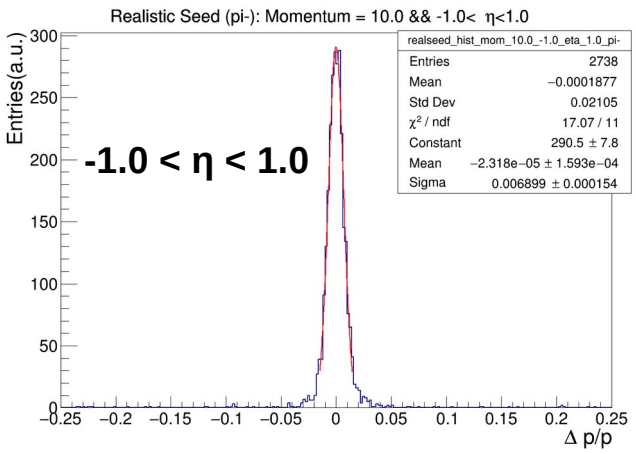
***p* = 0.5 GeV/c**

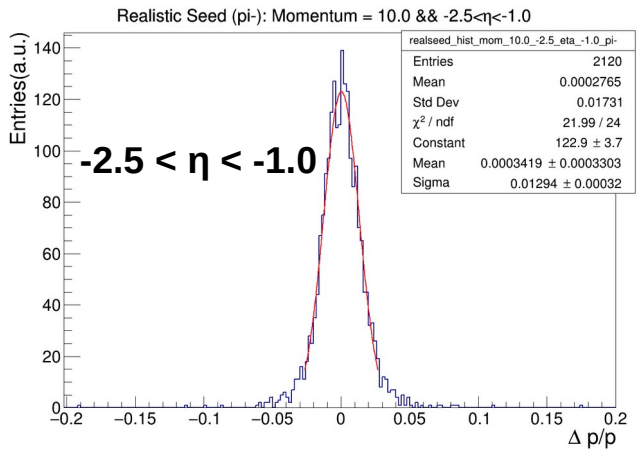


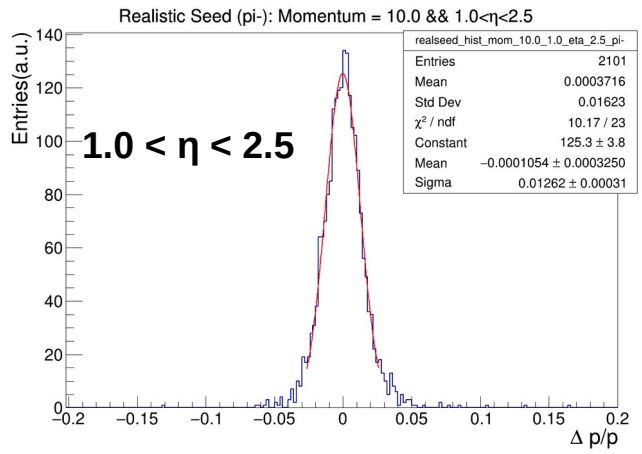




***p* = 10. GeV/c**







DCA_T Resolutions

Committed the code to the repository soon will be merged (again will produce debug plots/ final resolution plot)

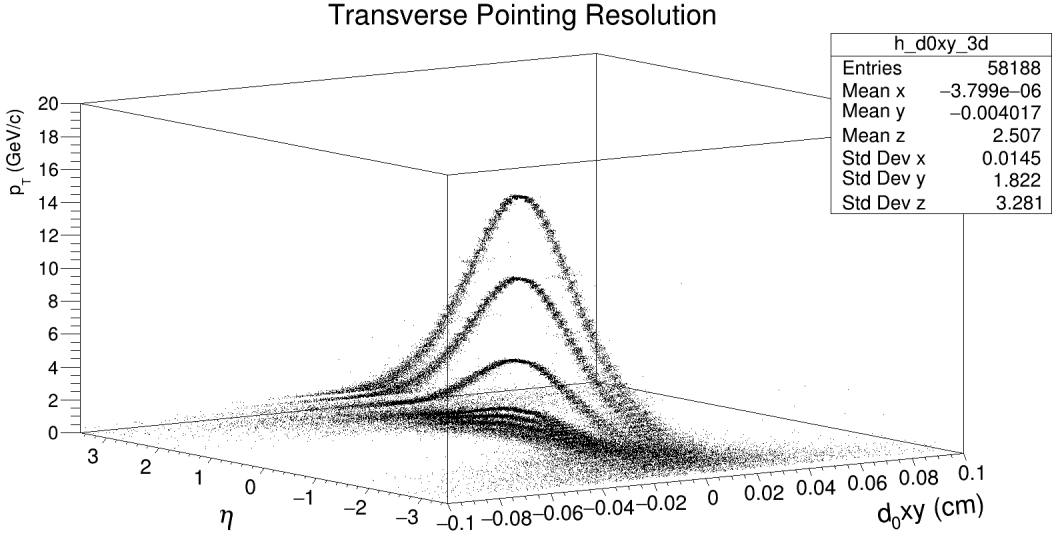
```
h_d0xy_3d->Fill(d0xy[j]*0.1, etamc, ptmc); // cm
h_d0z_3d->Fill(d0z[j]*0.1, etamc, ptmc); // cm
```

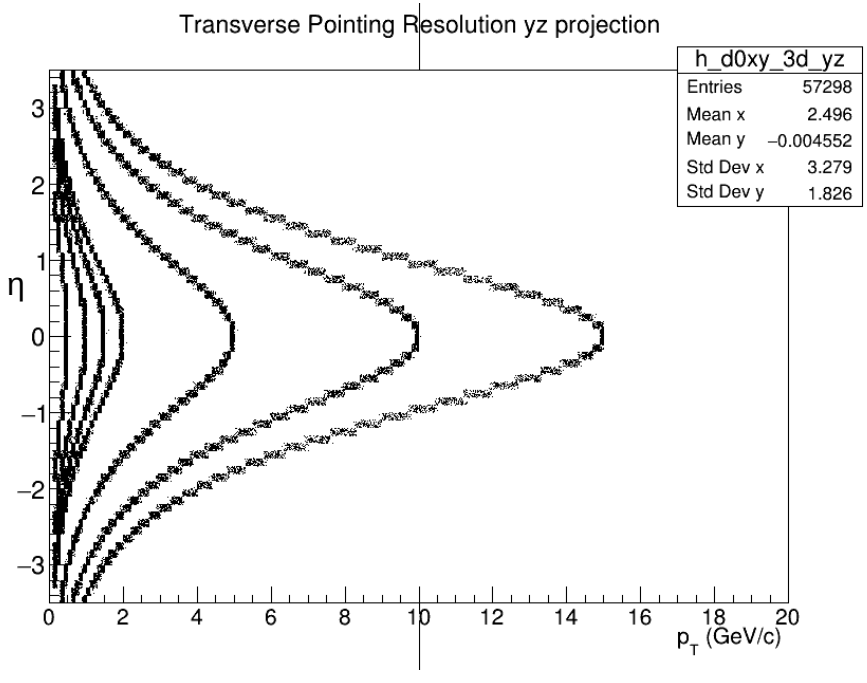
Tracking_Performances.C

mom_array=(0.5 1.0 2.0 5.0 10.0 15.0) Total expected entries = 60k (10k for each momentum)

Code committed after local test but not merged (Local desktop)

double pt[nptbins] ={0.2, 0.3, 0.5,1.0, 1.5, 2.0, 5.0, 8.0, 10., 15.0};





During projection: 10 % range is considered for each p_T bin, e.g. p_T = 0.5 range (0.45-0.55)