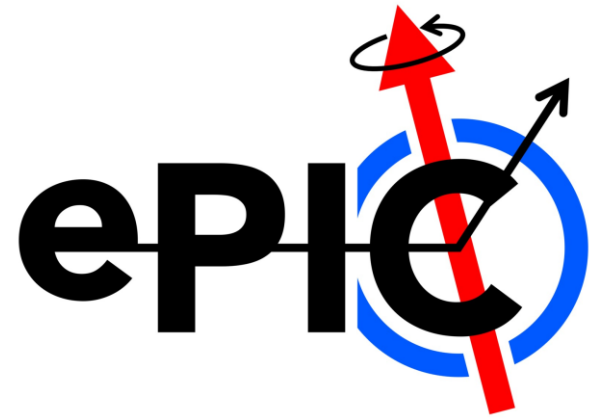




ePIC BHCal Simulation Status

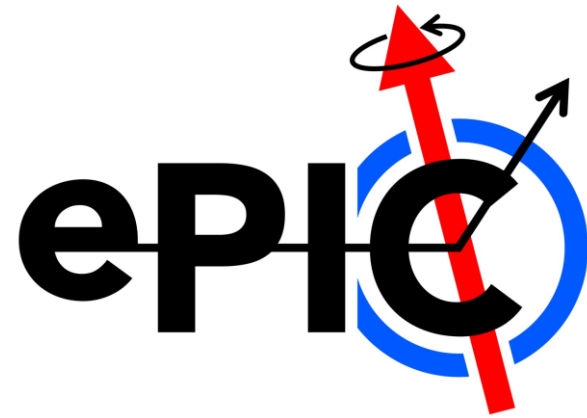
Derek Anderson

10.04.2024

pTDR Sim Status | Development Done/To-Do



- **Reminder:**
 - **Red** = more critical for pTDR
 - **blue** = less critical (but useful) for pTDR
- **Single Particle (Calibration)**
 - ✓ **[Done]** Rework NTuple filling for simulation campaign
 - ✓ **[Done]** Clean up TMVA training/application macro
 - 🖥️ **[Done]** Centralize/simplify calibration output plotting [last couple tasks will be done today]
- **Single Particle (Muons)**
 - 👉 **[Open]** Create macro to look at muon energy vs. cluster/hit layer/position
- **Event reconstruction**
 - 🖥️ **[ETA: early next week (FR this time)]** Create macro to calculate rough EM/HCal scale factors
 - 👉 **[Open]** Create macro to calculate true vs. reco JB variables, E_T Miss
- **Jet reconstruction**
 - 🖥️ **[ETA: late next week (FR this time)]** Create macro to generate tree of particles, tracks, EMCal, HCal clusters
 - 👉 **[Open]** Create macro to run jet reconstruction on particles, tracks + EMCal + HCal



ePIC BHCal

Analysis Crash Course

09.05.2024 Slides



Tasks | Single Particle (1/2)

For pions, neutrons, and protons

1) Rerun last year's studies on current geometry

- a) Run macro over output of September campaign
- b) Run TMVA macro over September campaign
- c) Generate the following plots:
 - › Uncalibrated E_{clust} vs. E_{par}
 - › Calibrated E_{clust} vs. E_{par}
 - › Energy resolution
 - › Linearity

For pions, neutrons, and protons (cont.)

2) Fine-tuning

- a) Rerun macro with additional variables for training (e.g. cluster shapes)
- b) Rerun TMVA macro, testing:
 - a) Different combinations of training variables
 - b) Different ML models
- c) Each time, regenerate the following plots:
 - › Uncalibrated E_{clust} vs. E_{par}
 - › Calibrated E_{clust} vs. E_{par}
 - › Energy resolution
 - › Linearity



Tasks | Single Particle (2/2)

For Muons

- 1) Generate sample of single muons w/ 2024.09.0 geometry
- 2) Generate the following plots:
 - a) **[Needs Dev.]** E_{clust} vs. calorimeter layer (12 ScFi layers in the BIC + the 1 in the BHCAL)
 - b) **[Needs Dev.]** E_{hit} vs. position in BIC, BHCAL
- 3) Iterate from there...

Legwork

- **Assignee:** Derek
 - **Timeline:** Monday
- 1) Finish PODIO macro
 - 2) Clean up TMVA macro
 - 3) Clean up plotting macros



Tasks | Event and Jet Reconstruction (1/2)

JB Kinematics, ET Miss: technically calculated in EICrecon, but NOT using any of the HCals

- 1) **[Needs Dev.]** Write macro to calculate these for:
 - a) EMCal clusters + HCal clusters
 - b) Generated hadrons
- 2) **Process September campaign using macro:**
 - a) For NC DIS sample
 - b) For CC DIS sample

Jets: *Very few studies have been done at ePIC using the HCals so far!*

- 1) **[Needs Dev.] Estimate calibration factors using single particle samples:**
 - a) Sum energy in BIC, BHCals
 - b) Fit $A(E_{\text{BIC}} + BE_{\text{BHCals}})$ to particle energy
- 2) **[Needs Dev.] Generate BIC+BHCals jets**
 - a) In NC DIS events ($Q^2 > 100$), generate TTrees of
 - Tracks
 - BIC, BHCals clusters (w/ track energy subtracted)
 - Generated particles



Tasks | Event and Jet Reconstruction (2/2)

Jets (cont.)

2) **Generating BIC+BHCal jets (cont.)**

- b) Run fastjet over Track + BIC + BHCal clusters
 - Initially w/ est. calibration factors
- c) Then run over only track + BIC clusters
- d) Finally run over generated particles

3) **[Needs Dev.] Generate plots**

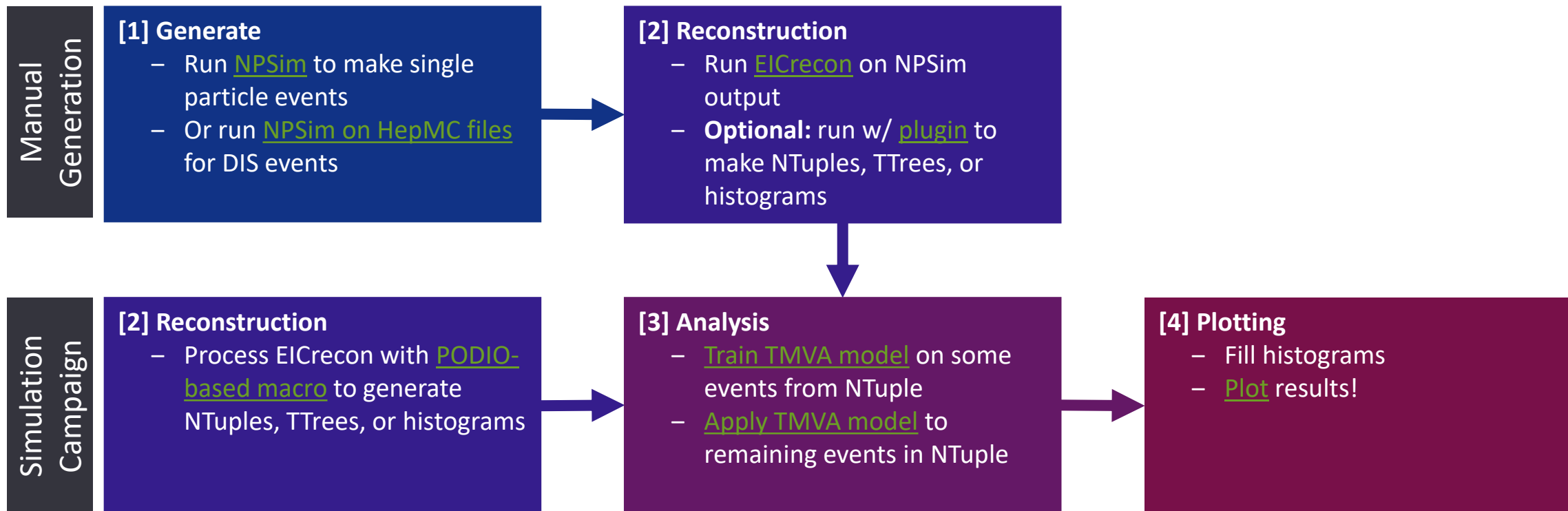
- a) Calculate JES, JER for Track+BIC+BHCal case and only Track+BIC case

Jets (cont.)

4) **[Needs Dev.] Calibrating jets**

- a) Would start with track + BIC + BHCal jets...
But then needs some more thought...

Tasks | Calibration Workflow





Tutorial | Building & Running NPSim, EICrecon

- **Running npsim:** runs ddsim + various afterburners
 - Added some QoL scripts I use to the pTDR repo:
 - › For [running npsim with a steering file](#)
`./RunNPSim.rb`
 - › And for [running npsim over a HepMC file](#)
`./RunNPSimOnHepMC.rb`
 - Also added a [small start-up script](#) to point your environment to a particular geometry
 - › Will update to 2024.09.0 geometry once released
- **Building EICrecon:** also added a [QoL script](#) to make compiling a one-liner
 - So when you're in your EICrecon directory
`./eic-build`
 - Also can be used in other contexts!
- **Running EICrecon:** like RunNPSim, there is a [QoL script](#) for EICrecon in the pTDR repo
 - Lets you set all of your options, output collections, and plugins and then run:
`./RunEICrecon.rb`

Tutorial | Running EICrecon with Plugins

- **EICrecon plugins:** run in parallel with all of the other algorithms
 - Handy way to generate histograms, etc. on the fly!
 - › You can find a couple in the pTDR repo [here](#)
 - **BUT you can't use it on simulation campaign output...**

- **Making a plugin:** in the EIC shell, after compiling EICrecon and sourcing eicrecon-this.sh, do:

eicmkplugin.py MyPlugin

cmake -S MyPlugin -B MyPlugin/build

cmake --build MyPlugin/build --target install -- -j8

- Also can do it with the [eic-build](#) script:

eicmkplugin.py MyPlugin

./eic-build MyPlugin

- But make sure \$EICrecon_MY is set first!

mkdir ~/EICrecon_MY

export EICrecon_MY=~/EICrecon_MY

```
// root includes
#include <TH1D.h>
#include <TFile.h>
// jana includes
#include <JANA/JEventProcessorSequentialRoot.h>
// edm definitions
#include <edm4eic/CalorimeterHit.h>
#include <edm4hep/SimCalorimeterHit.h>
#include <edm4hep/RawCalorimeterHit.h>

// global constants
static const size_t NEtaRanges(4);
static const size_t NRange(2);

class GetRawEnergiesProcessor : public JEventProcessorSequentialRoot {

private:

    // data objects we need from jana
    PrefetchT<edm4hep::SimCalorimeterHit> simHits = {this, "HcalBarrelHits"};
    PrefetchT<edm4hep::RawCalorimeterHit> rawHits = {this, "HcalBarrelRawHits"};
    PrefetchT<edm4eic::CalorimeterHit> rechHits = {this, "HcalBarrelRechHits"};

    // sim hit histograms
    TH1D* hEneHitSim[NEtaRanges] = {nullptr, nullptr, nullptr, nullptr};
    TH1D* hPhiHitSim[NEtaRanges] = {nullptr, nullptr, nullptr, nullptr};
    TH1D* hEtaHitSim[NEtaRanges] = {nullptr, nullptr, nullptr, nullptr};

    // reco hit histograms
    TH1D* hEneHitRec[NEtaRanges] = {nullptr, nullptr, nullptr, nullptr};
    TH1D* hPhiHitRec[NEtaRanges] = {nullptr, nullptr, nullptr, nullptr};
    TH1D* hEtaHitRec[NEtaRanges] = {nullptr, nullptr, nullptr, nullptr};

    // raw hit histograms
    TH1D* hAdcHitRaw = nullptr;

public:

    // ctor
    GetRawEnergiesProcessor() { SetTypeName(NAME_OF_THIS); }

    // required jana methods
    void InitWithGlobalRootLock() override;
    void ProcessSequential(const std::shared_ptr<const JEvent>& event) override;
    void FinishWithGlobalRootLock() override;

}; // end 'GetRawEnergies' definition
```



Tutorial | Reading EICrecon Output

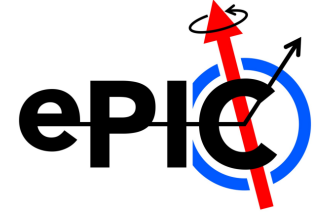
- Most tutorials focus on using plain ROOT when working w/ EICrecon output
 - Easy to get started with
 - But syntax can get tricky ([esp. with associations](#))
- But PODIO can be used in your analysis code too!
 - Makes life way easier!
 - Examples:
 - › [Calibration NTuple filler](#) in pTDR repo
 - › [Example Track-cluster matcher](#) in snippets repo
 - › [Example truth-cluster association reader](#) in snippets repo
- **One Drawback:** you either
 - a) Have to have PODIO, EDM4hep, and EDM4eic compiled
 - b) Or run macro in the EIC shell

```
// Analyze Reconstructed Jets
numRecoJetsEventHist->Fill(recoType.GetSize());
for(unsigned int i=0; i<recoType.GetSize(); i++)
{
    TVector3 jetMom(recoMomX[i],recoMomY[i],recoMomZ[i]);

    recoJetEvsEtaHist->Fill(jetMom.PseudoRapidity(),recoNRG[i]);
    recoJetPhiVsEtaHist->Fill(jetMom.PseudoRapidity(),jetMom.Phi());

    double esum = 0.0;
    for(unsigned int j=partsBegin[i]; j<partsEnd[i]; j++)
    {
        // partsbegin and partsEnd specify the entries from _Reconstr
        // _ReconstructedChargedJets_particles.index stores the Recon
        double mX = recoPartMomX[recoPartIndex[j]];
        double mY = recoPartMomY[recoPartIndex[j]];
        double mZ = recoPartMomZ[recoPartIndex[j]];
        double mM = recoPartM[recoPartIndex[j]];
    }
}
```

Tutorial | Reading ElCrecon Output w/ PODIO (1/4)

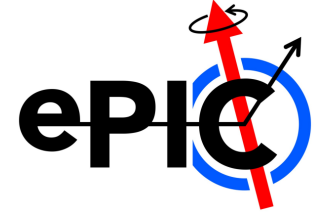


Step 1: make sure you include the relevant headers

Technically, these are redundant

```
// podio libraries
#include <podio/Frame.h>
#include <podio/CollectionBase.h>
#include <podio/ROOTFrameReader.h>
// edm4hep types
#include <edm4hep/Vector3f.h>
#include <edm4hep/Utils/vector_utils.h>
// edm4eic types
#include <edm4eic/Cluster.h>
#include <edm4eic/ClusterCollection.h>
#include <edm4eic/TrackPoint.h>
#include <edm4eic/TrackSegment.h>
#include <edm4eic/TrackSegmentCollection.h>
```

Tutorial | Reading ElCrecon Output w/ PODIO (2/4)

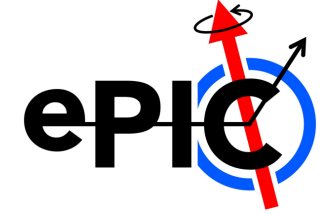


Step 2: open file w/ reader and (if needed) get no. frames (i.e. events)

```
// open file w/ frame reader
podio::ROOTFrameReader reader = podio::ROOTFrameReader();
reader.openFile( opt.in_file );
std::cout << "    Opened ROOT-based frame reader." << std::endl;

// get no. of frames and announce
const uint64_t nFrames = reader.getEntries(podio::Category::Event);
std::cout << "    Starting frame loop: " << reader.getEntries(podio::Category::Event) << " frames to process." << std::endl;
```

Tutorial | Reading ElCrecon Output w/ PODIO (3/4)



Here you could also do something like:
while (reader.readNextEntry(...))

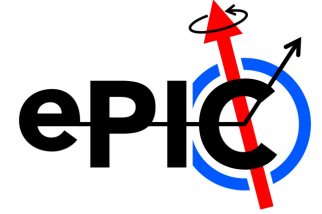
Step 3: iterate through frames

```
// iterate through frames (i.e. events in this case)
uint64_t nClustTotal    = 0;
uint64_t nClustMatched = 0;
for (uint64_t iFrame = 0; iFrame < nFrames; ++iFrame) {

    // announce progress
    std::cout << "      Processing frame " << iFrame + 1 << "/" << nFrames << "..." << std::endl;

    // grab frame
    auto frame = podio::Frame( reader.readNextEntry(podio::Category::Event) );
```

Tutorial | Reading ElCrecon Output w/ PODIO (4/4)



Step 4: grab collections and do stuff with them!

```
// grab collections
auto& clusters = frame.get<edm4eic::ClusterCollection>( opt.clusters );
auto& segments = frame.get<edm4eic::TrackSegmentCollection>( opt.projections );

// loop over clusters
for (size_t iClust = 0; edm4eic::Cluster cluster : clusters) {

    // grab eta/phi of cluster
    const double etaClust = edm4hep::utils::eta( cluster.getPosition() );
    const double phiClust = edm4hep::utils::angleAzimuthal( cluster.getPosition() );

    // match based on eta/phi dstiance
    double distMatch = std::numeric_limits<double>::max();

    // loop over projections to find matching one
    std::optional<edm4eic::TrackPoint> match;
    for (edm4eic::TrackSegment segment : segments) {
        for (edm4eic::TrackPoint projection : segment.getPoints()) {

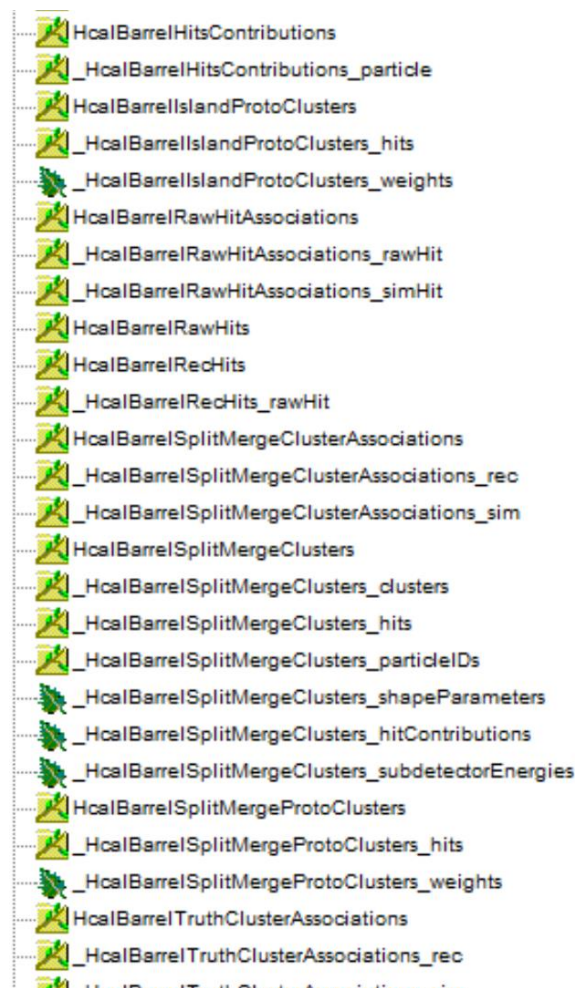
            // ignore if not pointing to calo or at face of calo
            const bool isInSystem = (projection.system == opt.system);
            const bool isAtFace = (projection.surface == 1);
            if (!isInSystem || !isAtFace) continue;
```

Tutorial | ElCrecon Collections

- ElCrecon generates a LOT of collections by default, so what's relevant?
 - **HcalBarrelHits**: sum of Geant4 hits for a tile (AKA a “Sim Hit”)
 - **HcalBarrelRawHits**: digitized BHCAL sim hit
 - **HcalBarrelRecHits**: reconstructed tile (energy + position + time info)
 - **HcalBarrelClusters**: clusters of tiles
 - **EcalBarrelScFiClusters**: cluster of scintillating fibers (only energy info)
 - **EcalBarrelImagingClusters**: cluster of AstroPix pixels (position info + some energy info)
 - **EcalBarrelImagingLayers**: weighted sum of all pixels in an imaging layer
 - **EcalBarrelClusters**: combined imaging + ScFi clusters
 - **MCParticles**: ALL simulated particles produced in “tracking region”
 - **GeneratedParticles**: all “final state” (status == 1) particles from MCParticles
 - **ReconstructedParticles**: tracks + ECal clusters*
 - **ReconstructedChargedParticles**: tracks + PID info
 - **ReconstructedJets**: anti-kt (R = 1) jets made from ReconstructedParticles**
 - **GeneratedJets**: anti-kt (R = 1) jets made from GeneratedParticles**

Legend

- | | |
|---|--|
| – Light Blue = edm4hep::SimCalorimeterHit | – Magenta = edm4eic::Cluster |
| – Dark Blue = edm4hep::RawCalorimeterHit | – Pink = edm4hep::MCParticle |
| – Violet = edm4eic::CalorimeterHit | – Red = edm4eic::ReconstructedParticle |



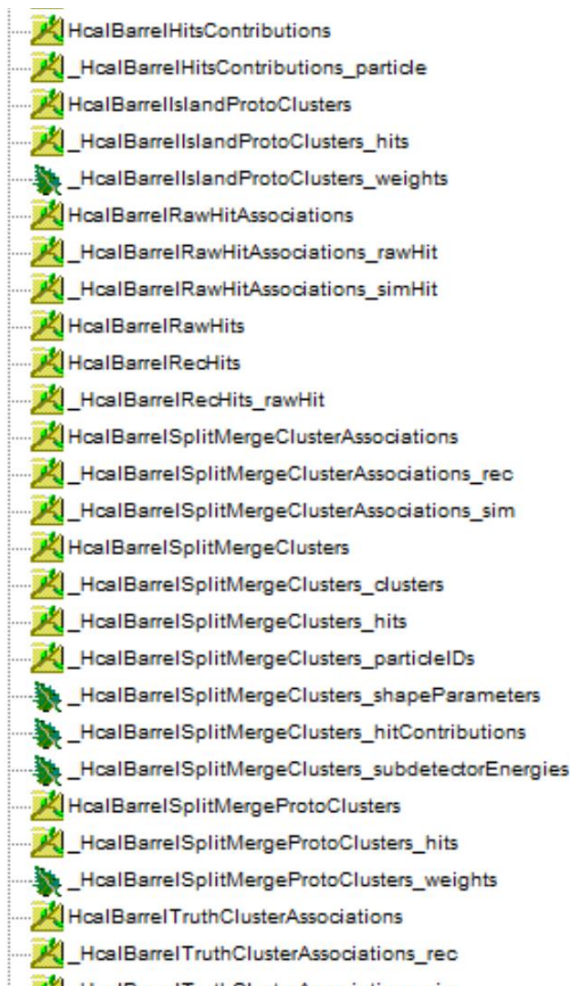
Tutorial | ElCrecon Collections

- ElCrecon generates a LOT of collections by default, so what's relevant?
 - **HcalBarrelHits**: sum of Geant4 hits for a tile (AKA a “Sim Hit”)
 - **HcalBarrelRawHits**: digitized BHCAL sim hit
 - **HcalBarrelRecHits**: reconstructed tile (energy + position + time info)
 - **HcalBarrelClusters**: clusters of tiles
 - **EcalBarrelScFiClusters**: cluster of scintillating fibers (only energy info)
 - **EcalBarrelImagingClusters**: cluster of AstroPix pixels (position info + some energy info)
 - **EcalBarrelImagingLayers**: weighted sum of all pixels in an imaging layer
 - **EcalBarrelClusters**: combined imaging + ScFi clusters
 - **MCParticles**: ALL simulated particles produced in “tracking region”
 - **GeneratedParticles**: all “final state” (status == 1) particles from MCParticles
 - **ReconstructedParticles**: tracks + ECal clusters*
 - **ReconstructedChargedParticles**: tracks + PID info
 - **ReconstructedJets**: anti-kt (R = 1) jets made from ReconstructedParticles**
 - **GeneratedJets**: anti-kt (R = 1) jets made from GeneratedParticles**

Notes

* **WARNING**: tracks+clusters combined in a hacky way using truth info

** Charged-only versions also exist, as do Centauro equivalents

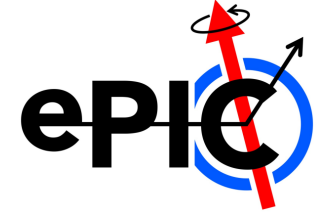




Tutorial | Other Tips

- **Running NPSim/DD4hep:**
 - The [eic-build](#) script can also build the DD4hep simulation via
`./eic-build epic`
 - You can always check what detector/version you're using with
`echo $DETECTOR_PATH/$DETECTOR_CONFIG`
- **Working with EICrecon:**
 - You can check what collections are available in the output (and what type they are) with
`eicrecon -L`
 - Option to control the number of events is
`-Pjana:nevents=X`
- **Working with EICrecon (cont.):**
 - And option to adjust the verbosity is:
`-Peicrecon:LogLevel={debug,trace,...}`
- **Working with EDM4eic/EDM4hep:**
 - If you need to look at what's actually in a type:
 - › The YAML files in the [EDM4eic](#) and [EDM4hep](#)
 - › But you can also look in your `/opt/local/include/edm4{eic,hep}` when in the EIC shell

Resources | Useful Links and More

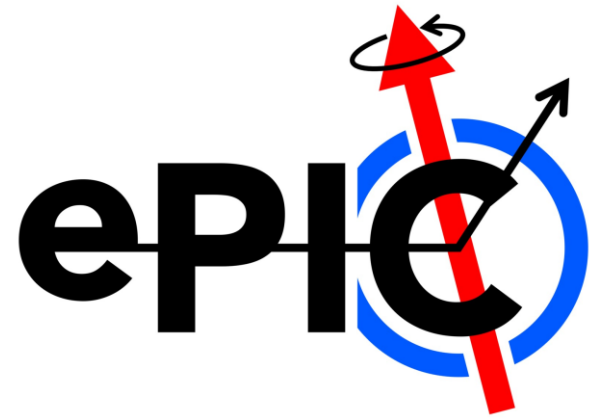


- **GitHub Links:** should be your go-to for most resources, esp. [the Landing Page](#)
 - [EIC GitHub Organization](#)
 - Data model: [EDM4eic](#) + [EDM4hep](#)
 - [DD4hep Simulation](#)
 - [EICrecon](#)
 - › [Documentation](#) (Changing fast! Some things might be out of date...)
 - › See esp. [flags](#) for adjusting parameters
 - [Snippets](#): Lots of useful scripts, macros, and more can be found here!
 - [BHCal pTDR Studies](#): We can use this to consolidate all of the code we need for the studies
 - [BHCal Calibration](#): Contains a lot of my WIP for calibration studies
- **Tutorials:** the [User Learning WG](#) and S&C team have hosted a lot of very thorough tutorials
 - [Series 2](#): had lots of good resources on working with simulation output
 - [JANA2 tutorial](#): this is very old, but still is a good one for learning about EICrecon plugins
 - [Carpentries Analysis Tutorial](#): has some very good reference code for using plain ROOT to work w/ EICrecon output
 - [2024 CERN S&C Workshop](#): had a good “end-to-end” tutorial in the Wednesday session
 - [Wouter’s PODIO Tutorial](#): is great intro to using it in your analysis!



Resources | Additional Resources

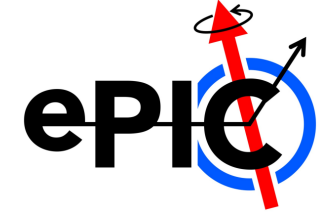
- **Other References:** curious about PODIO, ETC4hep, or our software stack? Here are some links!
 - [Description of PODIO](#)
 - [Description of the Key4hep stack](#)
 - [Description of our stack](#)
 - [JANA2 Website](#)
 - [DD4hep manual](#)
- **TMVA References:** here a few resources for learning about TMVA
 - [Users manual](#)
- **TMVA (Cont.):**
 - [Documentation website](#)
 - › Has links to example code, esp. note [TMVAREgression.C](#) and [TMVAREgressionApplication.C](#)
- **ML Power Week:** ePIC simulations were used in the 2023 HGS-HIRe ML Power Week
 - Challenge was to use ML to do calorimeter clustering
 - › Hannah Bossi's [summary presentation](#)
 - › All code available on [Kaggle](#)
 - Not really relevant to pTDR, but good food-for-thought!



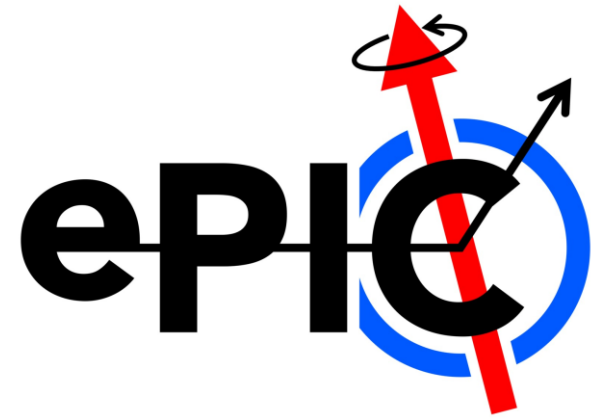
08.14.2024 Slide

BHCal DSC Meeting

TDR Sims | Needed Samples and Development



- **Reminder:** some ideas might be better suited for the physics paper rather
 - **Red** = plots critical for TDR
 - **blue** = maybe for physics paper
- **Single particle: energy spectra** (uncalibrated vs. calibrated), and **linearity/resolution**
 - Machinery in place
 - **Needed Samples:** several different single particles @ different energies
 - › Part of sim campaign output now!
 - **Needed Dev:** set-up tuple generator to run on sim campaign output [**assignee: Derek**]
 - › **Maybe:** switch to using a tree structure rather than tuple
- **Muons: reconstruction efficiency**
 - **To-Do:** ping Andrew Hurley again
- **Event reconstruction: JB variables, E_T^{miss}**
 - **Needed Samples:** NC/CC DIS
 - › Part of sim campaign!
 - › Also JB kinematics calculated as part of EICrecon
 - **Needed Dev:** minimally, cross-calorimeter topo-clusters [**assignee: Tristan**]
- **Jet reconstruction: JES/JER**
 - **Needed Samples:** High- Q^2 NC/CC DIS
 - › Part of sim campaign!
 - **Needed Dev:** need to think through a little more...

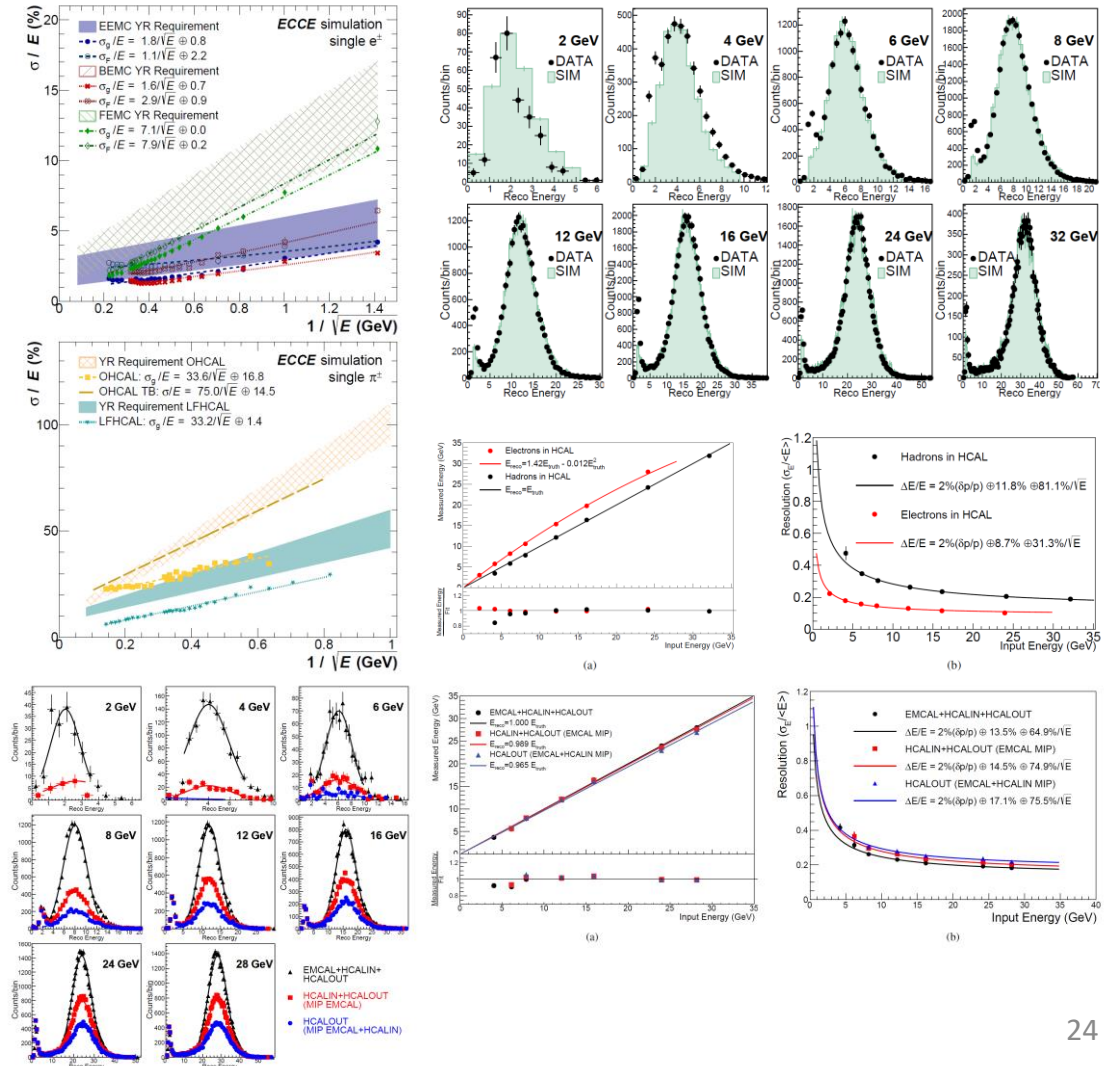


07.19.2024 Slides

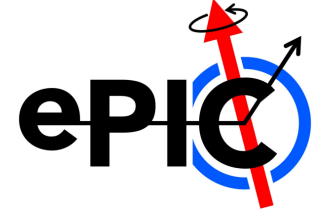
BHCal DSC Meeting

ePIC BHCAL Meeting | Possible TDR Plots (1/2)

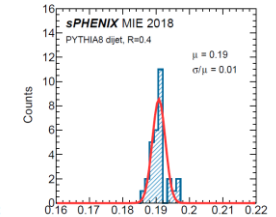
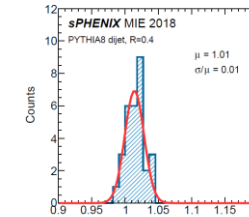
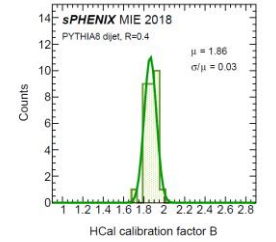
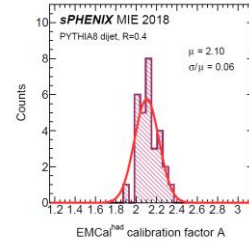
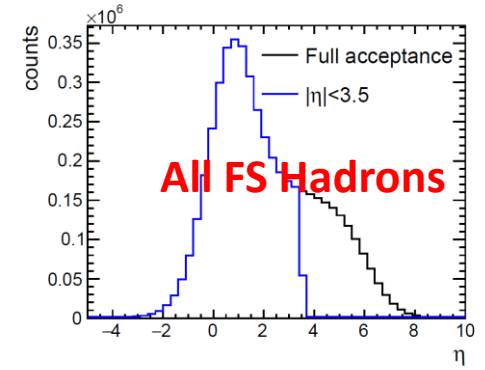
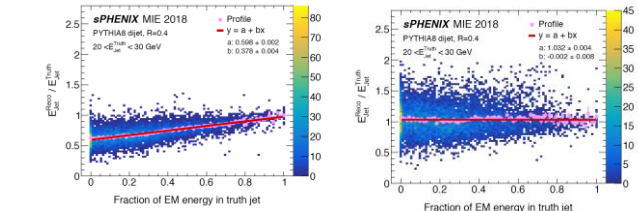
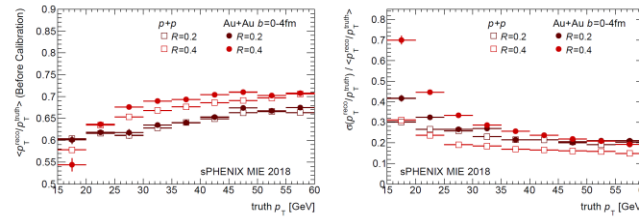
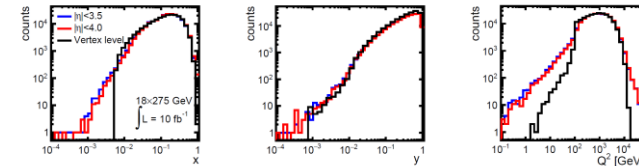
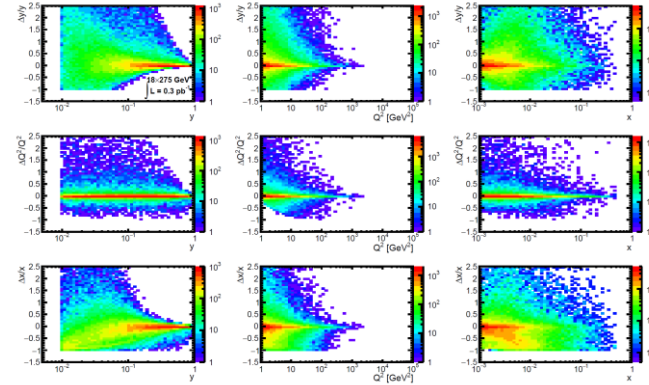
- **Single Particle:** do we meet YR requirements?
 - **Plots:** reconstructed particle energy; resolution + linearity
 - › $\pi^\pm, n^0 (p^+, k_L^0?)$
 - › Calibrated, uncalibrated
 - ☞ BHCAL + BIC, HCal only
 - ☞ Single tile vs. multi-tile? (1, 2, 3, 4, 5 tiles?)
- **Single – Few Particles:** do we help with $\mu^\pm$ ID?
 - **Plots:** $\mu^\pm$ energy; reconstruction efficiency; non- $\mu^\pm$ rejection factors
 - ☞ Andrew Hurley at UMass Amherst has started looking at $\mu^\pm$ ID in the Barrel
- **Right:** reference plots from ECCE proposal (upper left) and sPHENIX Test Beam Paper (all others)



ePIC BHCAL Meeting | Possible TDR Plots (2/2)

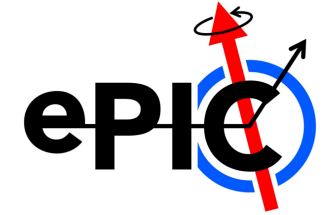


- **Event Reconstruction 1:** do we help with JB?
 - **Plots:** true vs. reco. x_{JB} , y_{JB} , Q_{JB}^2
 - › w/ vs. w/o BHCAL?
- **Event Reconstruction 2:** do we help with CC DIS tagging?
 - **Plots:** true vs. reco. E_T^{miss}
 - › w/ vs. w/o BHCAL?
 - › NC vs. CC DIS?
- **Jet Reconstruction:** do we improve the JES/JER?
 - **Plots:** JES/JER
 - › w/ vs. w/o BHCAL?
 - › Calibrated vs. uncalibrated?
- **Right:** reference plots from EIC YR (upper 3) and sPHENIX TDR (all others)



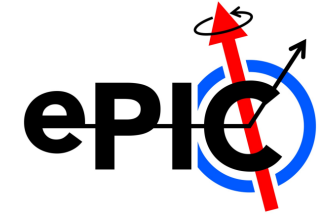
Note: “vertex level” = truth level

ePIC BHCAL Meeting | Thinking Through Plots (1/2)



- **Note:** some ideas might be better suited for the physics paper rather than the TDR
 - Also, several plots have synergy with other DSCs or PWGs
 - **Red** = plots critical for TDR, **blue** = maybe for physics paper
- **Single particle: energy spectra** (uncalibrated vs. calibrated), and **linearity/resolution**
 - Machinery in place
 - › Could stand a couple improvements...
 - › e.g. setting up macros to run on campaign output rather than as a plugin
 - ML part of calibration needs tuning (esp. for neutrons)
- **Single particle: (cont.)**
 - **Varying no. of tiles challenging:**
 - a) Need to rerun EICrecon for each combination of tile
 - b) Then would run calibration/plotting macros on output from each
- **Muons: reconstruction efficiency**
 - We should reach out to Andrew Hurley:
 - ☞ He's carried out fairly extensive studies of muon ID in the barrel

ePIC BHCAL Meeting | Thinking Through Plots (2/2)



○ Jet reconstruction: **JES/JER**

- Needs quite a bit of development, though
 - › Won't be able to use campaign output (HCal not used in jets yet)
 - › And we'll need EMCal-HCal calibration factors...
 - ☞ Could extend ML study: train on jets rather than clusters...
 - ☞ Good to have non-ML option available as well (e.g. ch. 8 of sPHENIX TDR)
- Possible intermediate plots:
 - 1) Jet energy vs. eta
 - 2) Fraction of EM vs. hadronic energy
 - ☞ Functionality is available to do basic track-matching
 - 3) Calibration factors

○ Jet reconstruction: (cont.)

- 4) EM energy fraction vs. jet energy
- 5) And finally, JES/JER
- Additional thoughts:
 - › I think the relevant scale to calibrate against would Q^2 ...
 - › Also would be good to explore asymmetric jet algorithm (e.g. Centauro)

○ Event reconstruction: **JB variables, E_T^{miss}**

- Algorithmically, very easy to calculate (sum over all hadron energies)
 - › But need to avoid double-counting...
 - › So need PF (or calibration factors?)

Backup | JB Variables & More Reference Plots

- **Jacquet-Blondel (JB) Kinematic Variables:** i.e. reconstructed event kinematics using only the hadronic final state

$$\begin{aligned}
 - \quad y_{JB} &= \frac{\sum_h (E_h - p_{z,h})}{E_{beam}^e} \\
 - \quad Q_{JB}^2 &= \frac{(\sum_h p_{x,h})^2 + (\sum_h p_{y,h})^2}{1 - y_{JB}} = \frac{(E_T^{miss})^2}{1 - y_{JB}} \\
 - \quad x_{JB} &= \frac{Q_{JB}^2}{s y_{JB}}
 \end{aligned}$$

- **Upper Right:** reference plot for generated vs. reconstructed E_T^{miss} (from [arXiv:2006.1520](https://arxiv.org/abs/2006.1520))
- **Lower Right:** reference plot for JES/JER with vs. without HCal's (from EIC YR)

