Validation tools and benchmarks for calorimetry

Dmitry Kalinkin

Calorimetry WG meeting — August 19, 2026

Why validation? Running a future experiment

- ePIC software must **enable running of the experiment** once it is constructed — the software stack is being built and exercised now, years before data taking
- Continuous monitoring of performance of:
  - *individual subsystems* — e.g. e- energy resolution in EEEMCal
  - *joint operation* — e.g. π-rejection factor using calorimetry, tracking and PID detectors
- Validation also tracks the **evolution of reconstruction algorithms**
- Every change to geometry, digitization or reconstruction is quantified — not just "compiles & runs"

Why benchmarks? Beam tests & model improvement

- **Capturing beam test results** with adequate detector geometries and simulation models ensures the data *stays interpretable* for possible future re-analysis
  - geometry, materials, readout parameters are versioned alongside the analysis
- Geant4 physics models are trusted, but models for **sensor technology and the readout chain** can be improved via rigorous comparisons to beam-test and bench-test results
  - benchmarks literally serve as *benchmarks* here: fixed, reproducible reference points, challenges to the simulation

Where benchmarks sit in the ePIC stack

- **Geometry:** DD4hep (`eic/epic`), configurations built from modular compact files
- **Simulation:** Geant4 via `npsim` / DD4hep plugin
- **Digitization + Reconstruction:** EICrecon (JANA2, PODIO/EDM4hep/EDM4eic)
- **Benchmarks:** analysis scripts on top of the same PODIO output
  - `detector_benchmarks` — single subsystem performance
  - `physics_benchmarks` — whole-detector, physics-process driven
- **Orchestration:** Snakemake workflows, GitLab CI at ANL + GitHub Actions triggers
- Workflow per trigger:
  1. build container with the software versions under test (spack)
  2. run simulation + reconstruction stages
  3. run analysis stage, produce plots
  4. deploy artifacts to web-based plot browser

Benchmarks as an element of CI

- Triggered on **every pull request** to geometry (`eic/epic`), reconstruction (`EICrecon`), data model, and the benchmark repositories themselves
- Temporary container is built with the branch under test; benchmark pipelines run inside it
- **Pass/fail tests**: scripts return exit status 0 for success — any non-zero value fails the pipeline and blocks the change
- Plots attached to every pipeline: reviewers *see* the impact of a change (e.g. resolution shift after a geometry update) before merging
- Common simulation samples defined at the workflow root are shared between benchmarks — cost of adding a new benchmark is low
- Same workflows will be re-used for validation of monthly software releases and centralized simulation campaigns

Calorimetry benchmarks

A tour of the available benchmarks in `eic/detector_benchmarks`

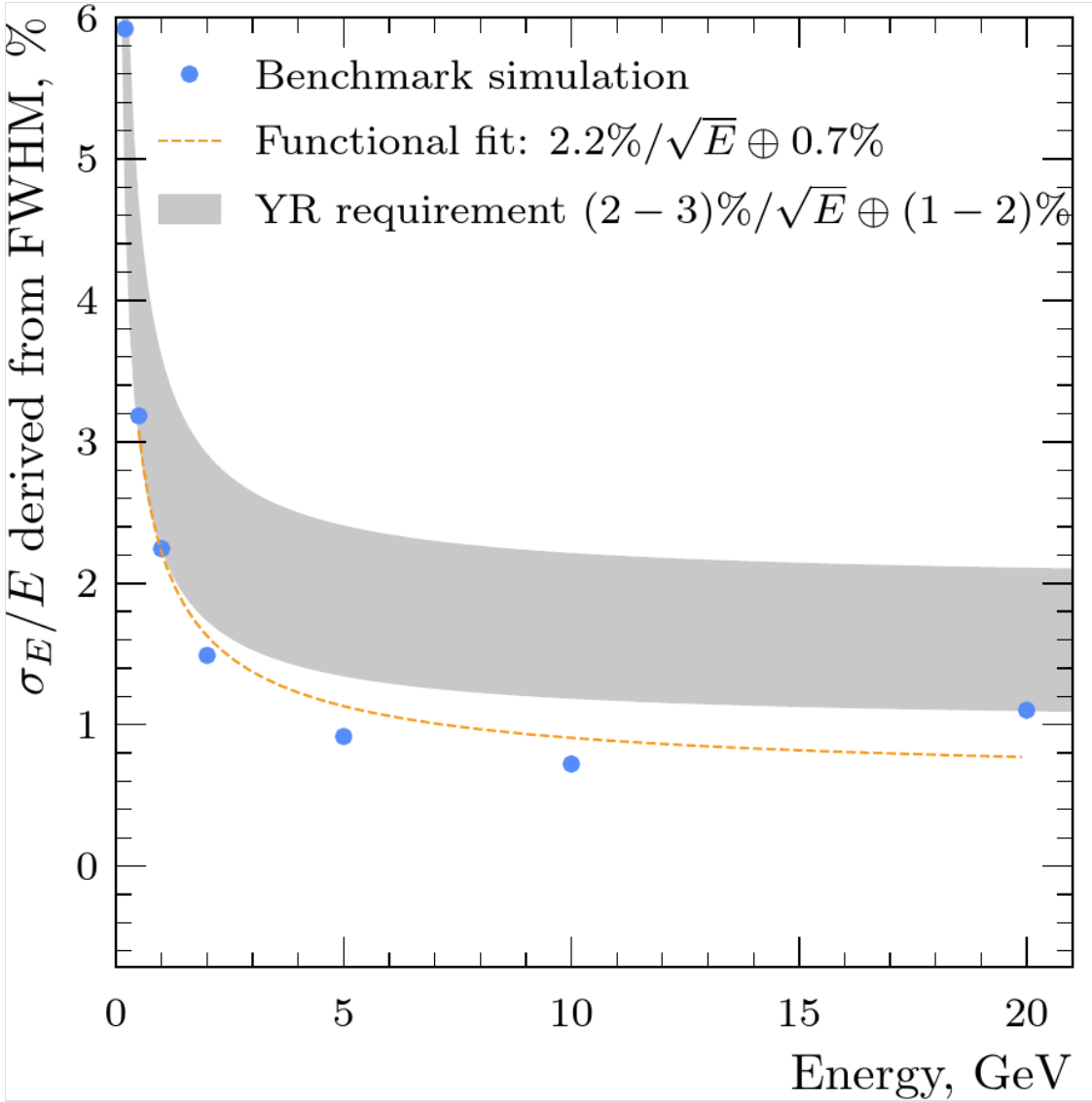
Overview of calorimetry benchmarks

Region	Benchmarks
Backward ECal (EEEMCal)	`backwards_ecal`, `calo_pid`
Barrel ECal (BIC)	`barrel_ecal`
Forward ECal (FEMC)	`femc_electron`, `femc_photon`, `femc_pi0`
Backward HCal (nHCal)	`nhcal_acceptance`, `nhcal_basic_distribution`, `nhcal_sampling_fraction`, `nhcal_pion_rejection`, `nhcal_dimuon_photoproduction`
Forward HCal (LFHCAL) + insert	`lfhcal`, `insert_muon`, `insert_neutron`, `insert_tau`
ZDC (far forward)	`zdc_lyso`, `zdc_neutron`, `zdc_photon`, `zdc_pi0`, `zdc_lambda`, `zdc_sigma`
Cross-detector	`ecal_gaps`

`backwards_ecal` — EEEMCal performance

detector

- Electron-endcap EM calorimeter ($PbWO_4$ crystals) — key detector for scattered-electron kinematics in DIS
- Single-particle samples (e^-, π^-) across the backward acceptance and a range of momenta
- Monitors:
  - energy response and **resolution vs energy** (fit to stochastic + constant terms)
  - **e/π separation**: pion rejection factor at fixed electron efficiency — a direct physics requirement from the Yellow Report
- Implemented as a literate `org`-mode / Jupyter analysis, scaled out with Dask; Snakemake provides simulation inputs
- Example of a *requirement-tracking* benchmark: current design geometry is continuously checked against the π-rejection specification

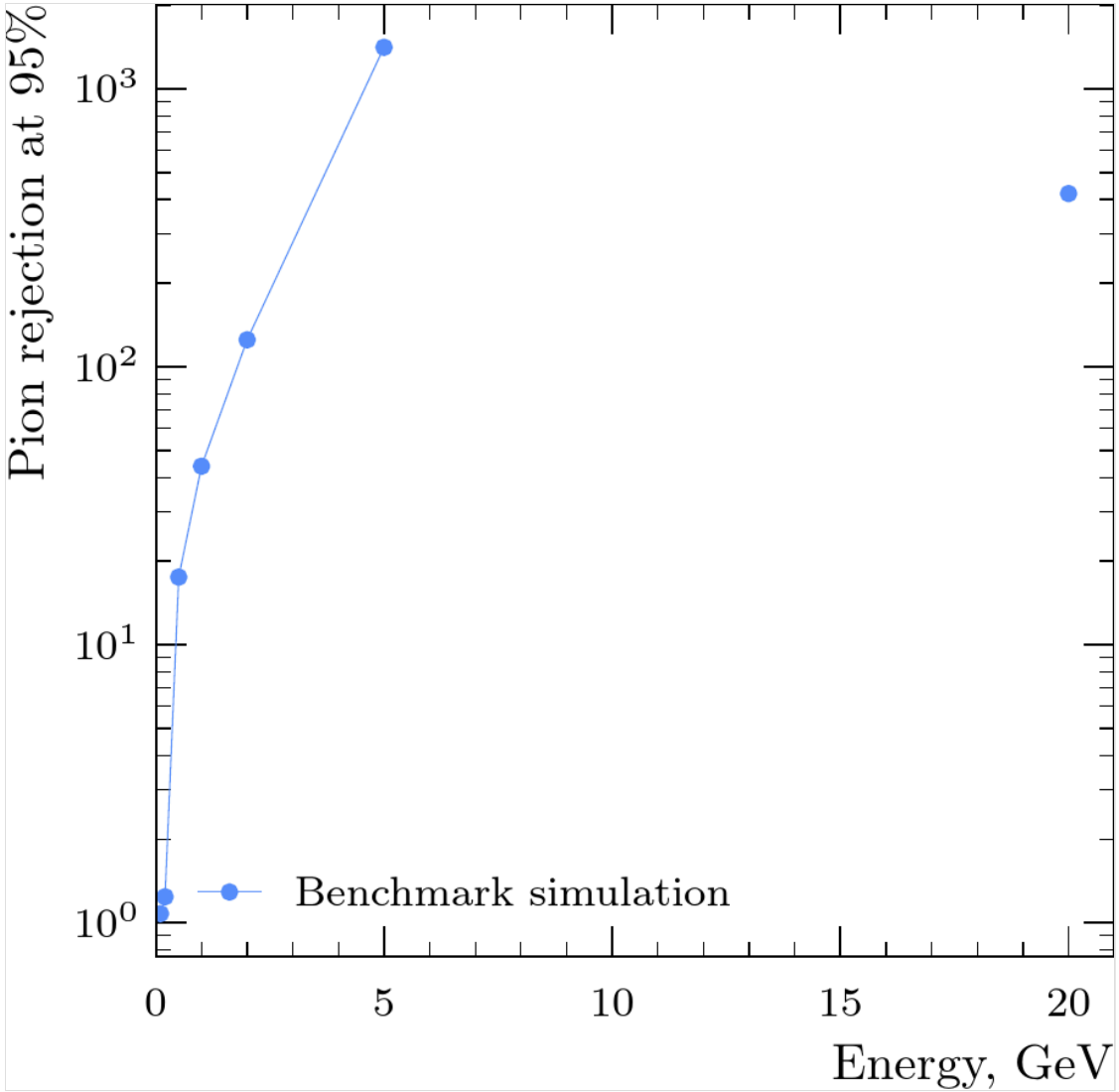


Energy resolution vs momentum

`calo_pid` — EEEMCal calorimetric PID

detector

- Dedicated study of **particle identification using calorimeter information** in the backward endcap
- Inputs: separate electron and pion single-particle samples over the EEEMCal acceptance
- Trains/evaluates classifiers on shower observables (e.g. *E/p*, shower shape features) and produces **ROC curves** (via `scikit-learn`)
- Quantifies achievable pion rejection vs electron efficiency as a function of momentum and pseudorapidity
- Demonstrates joint use of **tracking + calorimetry**: the achievable PID depends on the momentum resolution delivered by the current tracker design



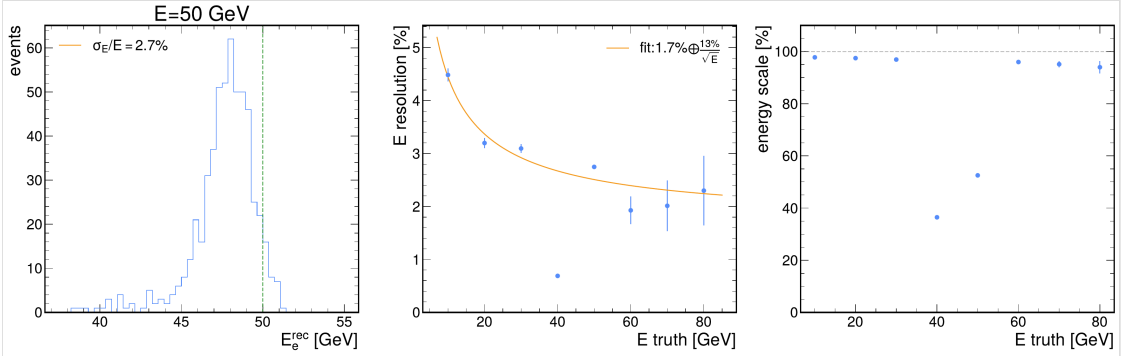
π rejection vs electron efficiency (ROC)

`barrel_ecal` — Barrel imaging calorimeter detector

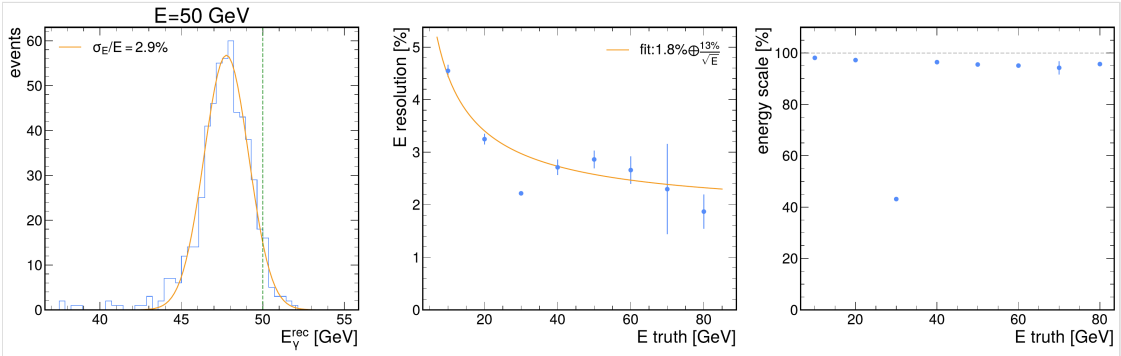
- A legacy benchmark from Athena
- Barrel EM calorimeter (imaging AstroPix layers + Pb/ScFi): rich set of ROOT-macro analyses
- Single-particle and energy-scan productions:
  - `emcal_barrel_particles` — response to e^-, γ, π, … sampling fraction, linearity, resolution
  - `emcal_barrel_energy_scan` — resolution parametrization vs energy
  - `emcal_barrel_pi0` — π^0 reconstruction, di-photon separation
  - `emcal_barrel_pion_rejection` — e/π discrimination with the imaging layers
- Includes automated **pass/fail checks** against expected sampling fraction and resolution — a geometry change that silently alters the response fails CI

femc_electron / femc_photon detector

- Forward EM calorimeter (W/SciGlass-style homogeneous or ScFi configurations)
- Single e^- (or γ) generated with $2° < \theta < 28°$ over a momentum scan; event counts scaled with energy to equalize statistics
- Produces per-energy and summary plots:
  - cluster-finding efficiency across the acceptance
  - energy scale (linearity) and resolution fits
  - position reconstruction performance



Electron energy resolution

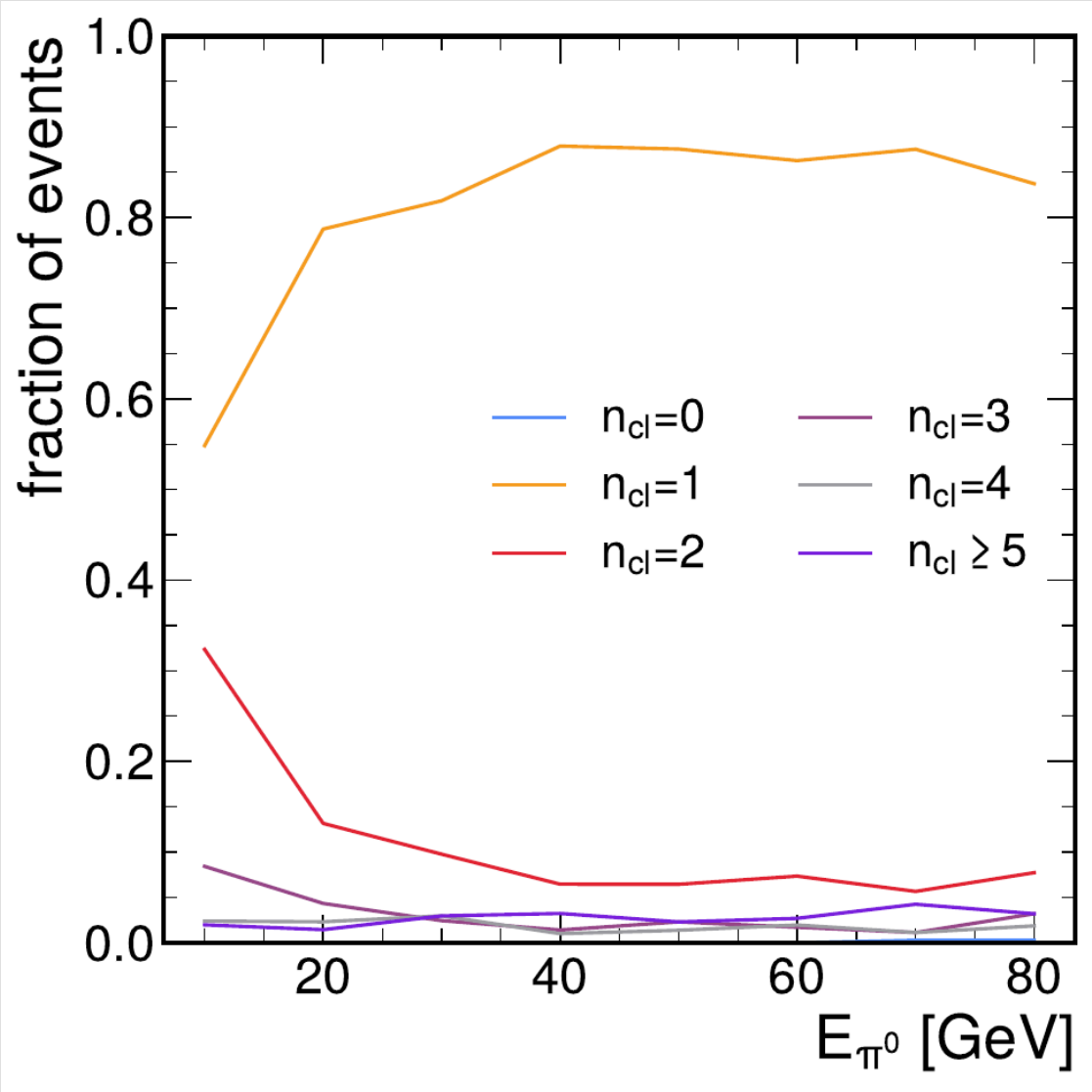


Photon energy resolution

`femc_pi0` — neutral pions in the FEMC

detector

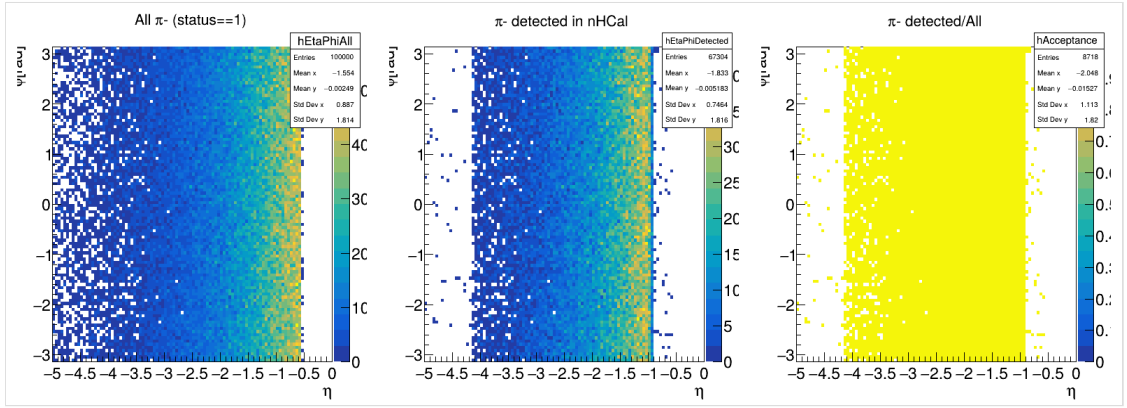
- Single π^0 samples in the forward endcap acceptance
- Tests the ability to reconstruct **merged / close-by photon showers**:
  - di-photon invariant mass peak position and width
  - reconstruction efficiency vs π^0 energy as showers begin to overlap
- Sensitive probe of **clustering algorithm evolution** — improvements in cluster splitting show up directly in this benchmark



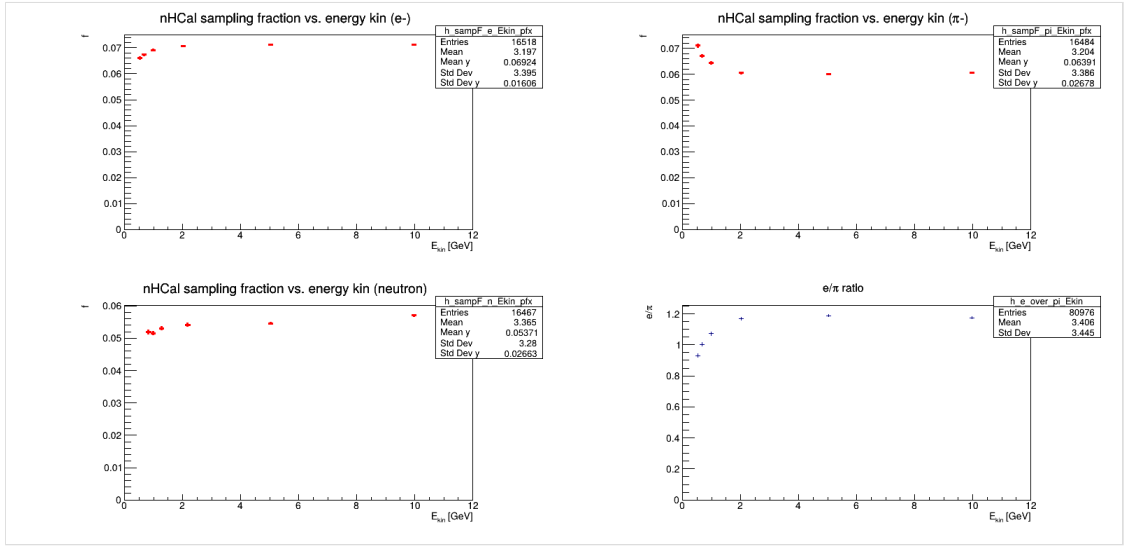
Number of clusters per π^0

nHCal I: `acceptance`, `basic_distribution`, `sampling_fraction` detector

- Backward hadronic calorimeter (nHCal) suite of foundational checks:
- `nhcal_acceptance` — maps geometric acceptance in η–φ with single particles; guards against dead regions introduced by geometry changes
- `nhcal_basic_distribution` — hit-level sanity distributions (hit positions, energy deposits, multiplicities)
- `nhcal_sampling_fraction` — measures the sampling fraction with dedicated `backward_hcal_only` geometry configurations; feeds the calibration constants used by reconstruction



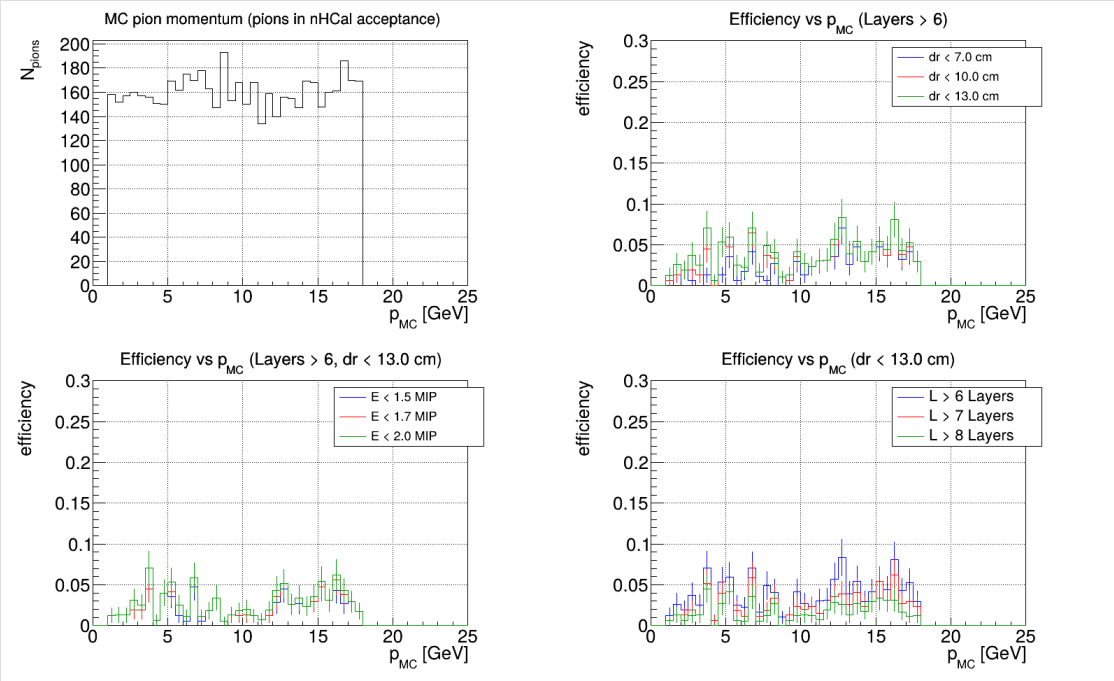
nHCal acceptance (10 GeV)



Sampling fraction vs kinetic energy

nHCal II: `pion_rejection`, `dimuon_photoproduction` physics

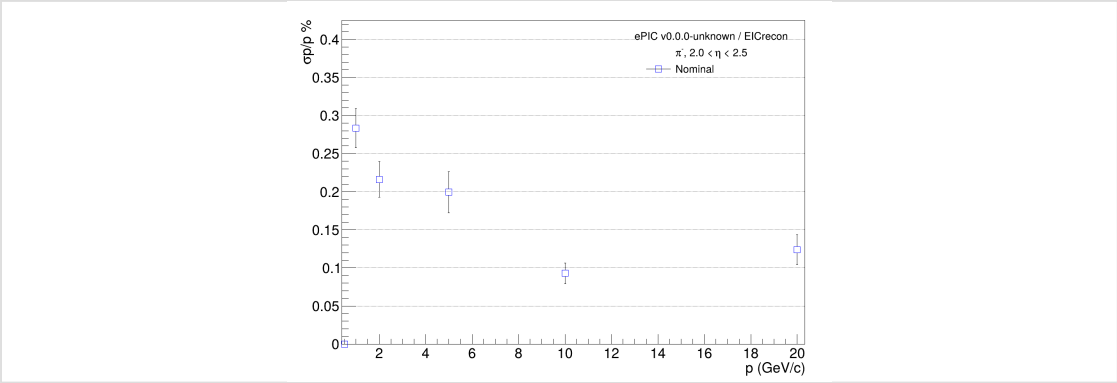
- `nhcal_pion_rejection`
  - μ/π discrimination using nHCal information for tracks projected into the acceptance
  - track projections matched to reconstructed hits along *z* (longitudinal profile)
  - quantifies muon identification for physics with muons in the backward region
- `nhcal_dimuon_photoproduction`
  - full physics process: Pythia di-muon photoproduction (`ep_DiMuon.cmnd`)
  - tests nHCal in **joint operation with tracking**: J/ψ → μμ-style signatures require both momentum measurement and muon ID
- Example of subsystem benchmarks graduating to a physics-signature benchmark
  - IMO despite using Pythia MC, this belongs in `detector_benchmarks`



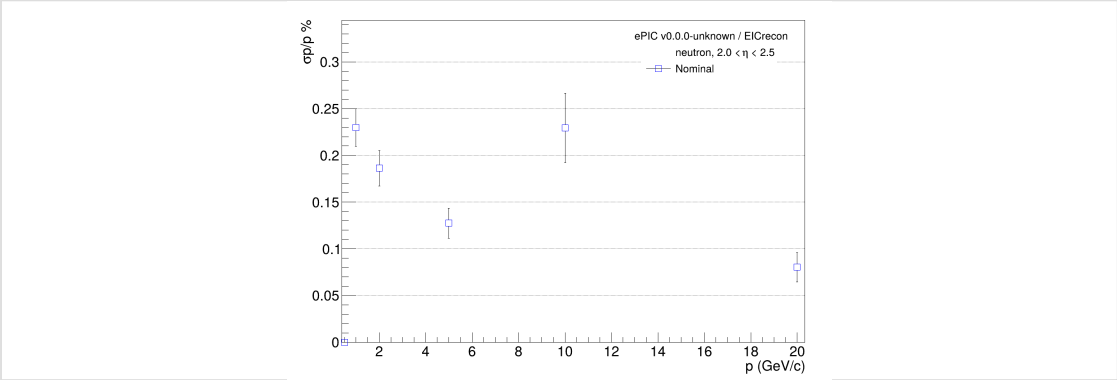
Track matching efficiency

`lfhcal` — forward hadronic calorimeter detector

- Longitudinally-separated forward HCal (LFHCAL, SiPM-on-tile)
- Events generated with a **range of particle species at fixed energies**, aimed at (or forward of) the LFHCAL
- Clustering performance evaluated:
  - in **isolation** (LFHCAL only)
  - in **combination with the FEMC** in front of it — realistic compensation and shower-matching conditions
- Momentum-binned resolution comparisons (`doCompare_widebins_mom.C`) track performance across software versions
- This is also where the **2024/2025 test-beam geometries** could connect (next section)



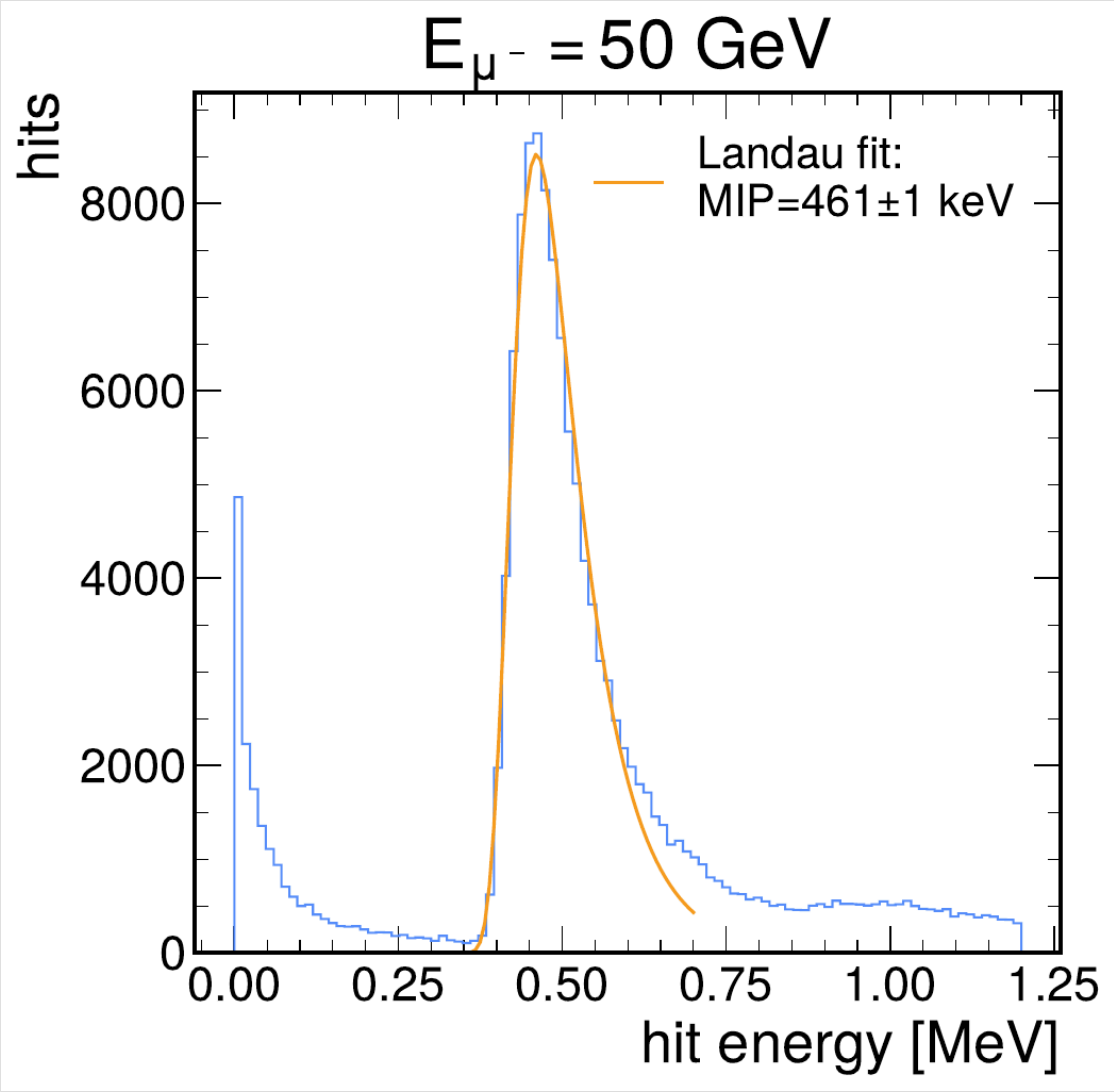
π^- momentum resolution, 2.0<η<2.5



Neutron momentum resolution, 2.0<η<2.5

HCal insert: `insert_muon`, `insert_neutron`, `insert_tau` detector

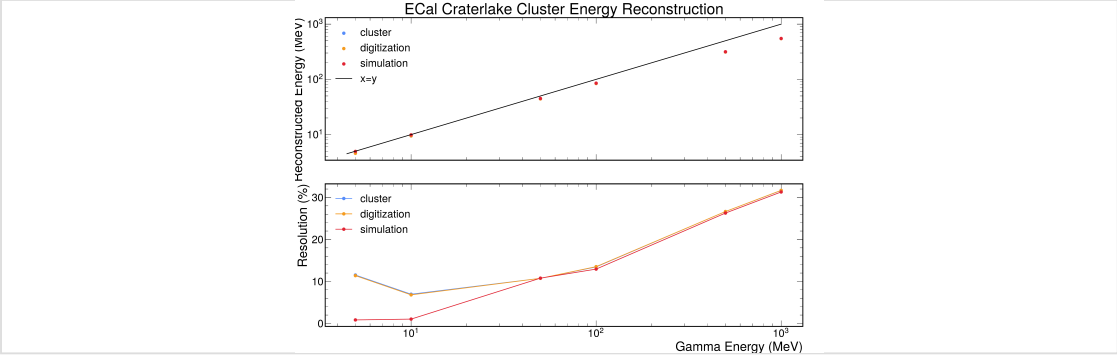
- High-η calorimeter insert around the beampipe — the most forward hadronic coverage
- `insert_muon` — single muons as MIP "candles":
  - MIP peak position validates the digitization/light-yield model of the readout chain
- `insert_neutron` — single neutrons:
  - hadronic energy response, resolution and acceptance at extreme rapidity
- `insert_tau` — τ decays in the insert region:
  - multi-prong hadronic showers, a physics-motivated stress test of clustering
- Uses common HepMC particle-gun generation macros shared across benchmarks



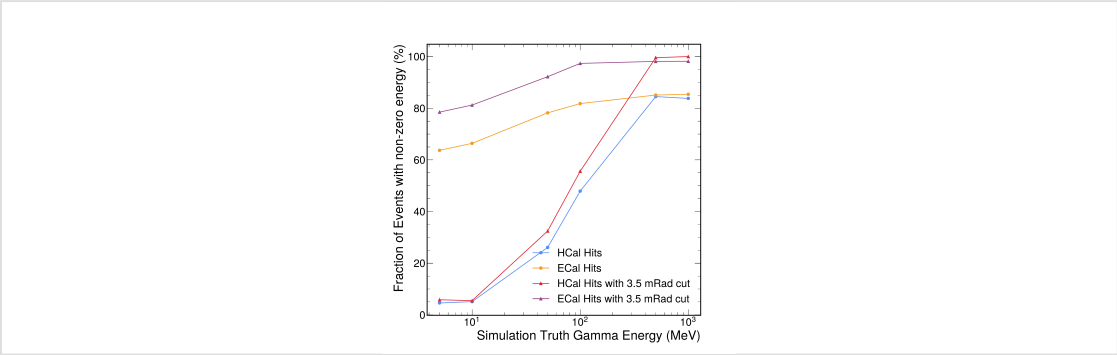
MIP peak — insert muon

`zdc_lyso` — LYSO crystal ZDC detector

- Zero-Degree Calorimeter EM section: LYSO crystals for **low-energy photons**
- Single photons, **5 MeV – 1 GeV**, generated at 0–5.24 mrad w.r.t. the proton/ion beam direction
- Produces acceptance and energy-resolution performance plots
- Directly relevant for nuclear de-excitation photon tagging in e+A physics



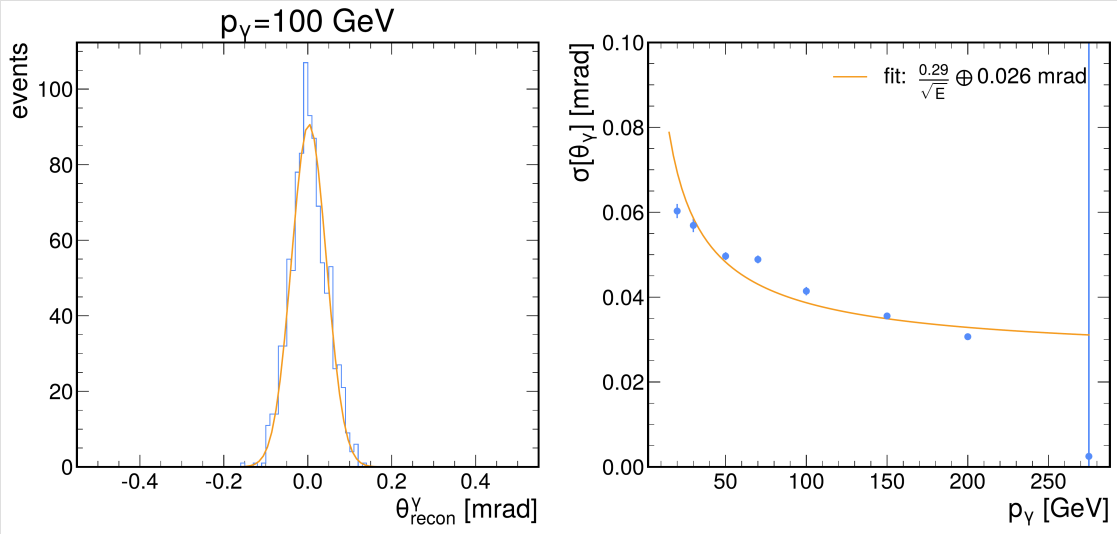
Energy linearity & resolution vs. E_γ



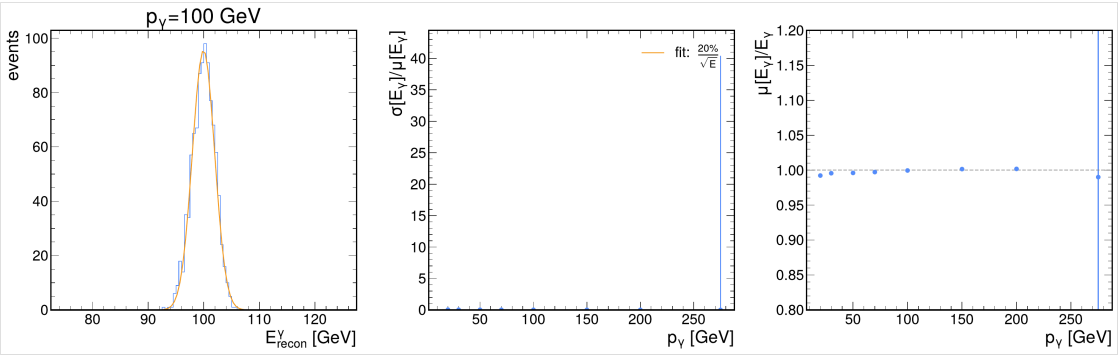
Detection acceptance vs. E_γ (with/without angular cut)

zdc_neutron / zdc_photon detector

- zdc_neutron
  - single neutrons at 100 GeV, polar angles 0–6 mrad (uniform in cosθ), uniform in azimuth w.r.t. the ion beam
  - acceptance maps and reconstruction performance (energy scale, resolution, position)
  - full far-forward beamline transport included — validates beampipe/magnet apertures too
- zdc_photon
  - high-energy photons into the ZDC EM + hadronic sections
  - γ acceptance and resolution needed for u-channel and exclusive processes



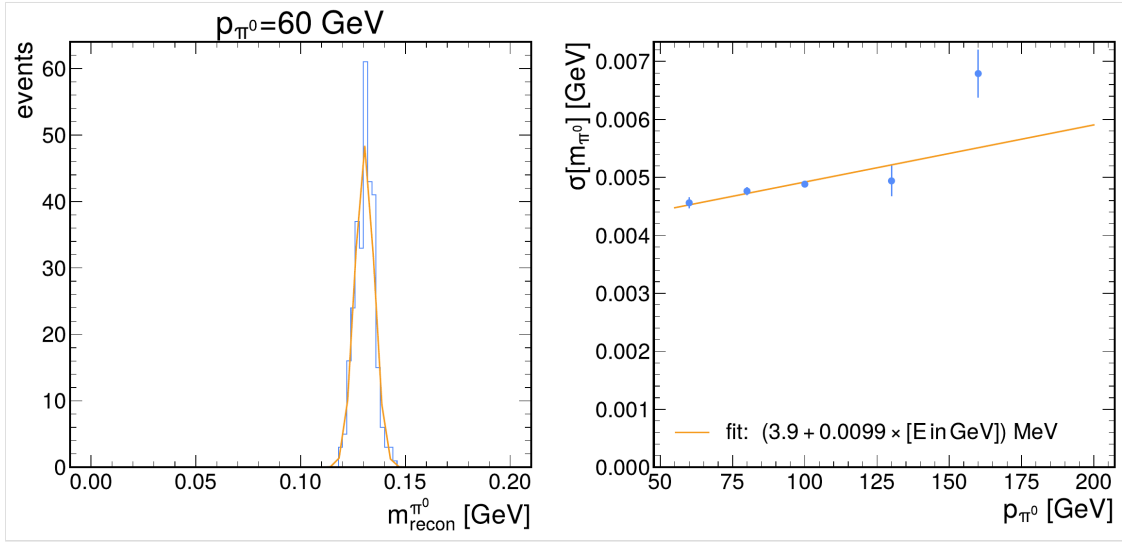
Photon θ reconstruction & resolution vs. p_γ



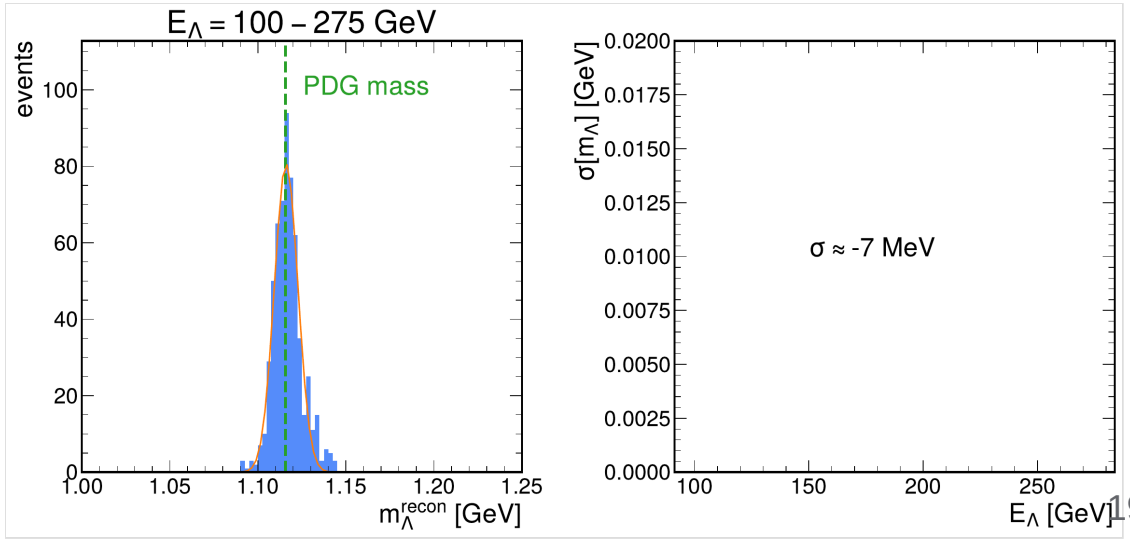
Photon energy resolution vs. p_γ

ZDC physics: `zdc_pi0`, `zdc_lambda`, `zdc_sigma` physics

- Benchmarks built around **decayed particles** with dedicated event generators (`gen_pi0_decay.cxx`, `gen_lambda_decay.cxx`, `gen_sigma_decay.cxx`)
- `zdc_pi0` — $\pi^0 \rightarrow \gamma\gamma$ at ZDC distances: two-photon separation at ~35 m from the IP, invariant-mass reconstruction
- `zdc_lambda` — $\Lambda \rightarrow n\pi^0$ reconstruction: combined neutron + photon measurement, decay-vertex sensitivity
- `zdc_sigma` — Σ hyperon reconstruction: cascaded decays, the most demanding far-forward combinatorics
- These validate the **joint operation** of the ZDC subsystems and the far-forward reconstruction chain end-to-end

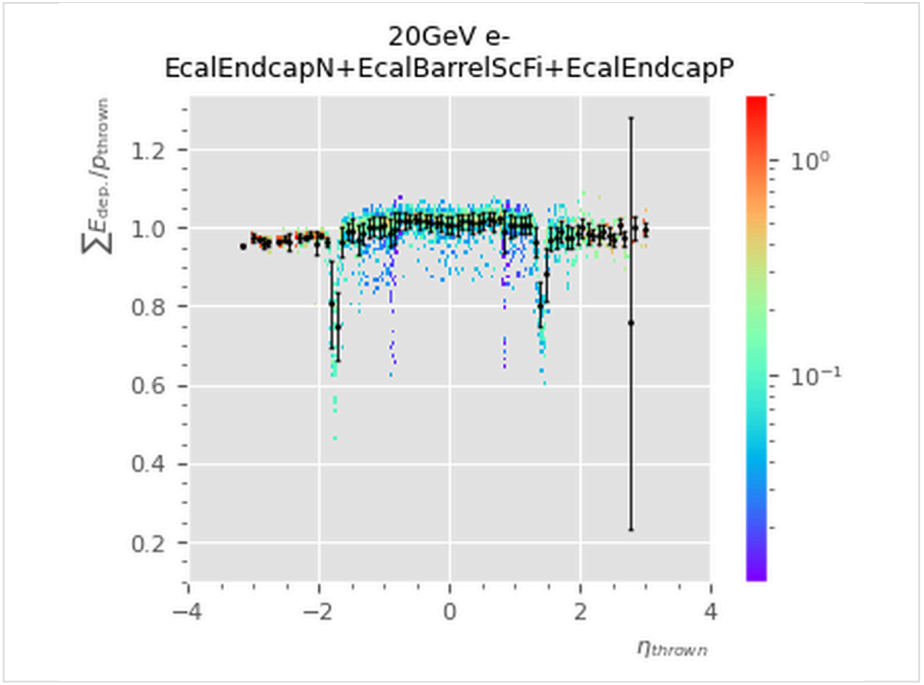


π^0 mass reconstruction



`ecal_gaps` — cross-detector uniformity detector

- Not a single-subsystem benchmark: scans electrons over the **full η range** covering backward endcap ↔ barrel ↔ forward endcap transitions
- Maps reconstructed energy response vs η and φ to expose:
  - acceptance **gaps and cracks** between calorimeter systems
  - response non-uniformity near module and sector boundaries
- Dask-based analysis over large scans; produces heatmaps used in detector-design discussions
- A prime example of validation guarding **global detector integration**, not just individual detector performance



Energy response across all ECal systems, 20 GeV e^-

Beam tests in the same software stack

Goal: preserving test-beam configurations as first-class geometries

Why beam-test geometries belong in `eic/epic`

- Beam tests are where **sensor and readout-chain models** are confronted with reality
  - Geant4 physics is trusted; SiPM saturation, light collection, thresholds, cross-talk, digitization are *ours* to model and tune
- Capturing the **as-tested geometry** in the same DD4hep repository:
  - the exact detector stack, materials and readout segmentation are versioned with the code
  - simulations remain **reproducible**; beam-test data stays interpretable for future re-analysis
  - the *same* digitization/reconstruction code runs on test-beam and full-detector setups
- Once in the repository, a test-beam configuration can be exercised by CI like any other benchmark — tuned parameters cannot silently drift away from the beam-test anchor
- Working example: eic/epic#881 — LFHCAL test-beam geometries (2024, 2025, 2026 options, 2028)

Case study: LFHCAL test beam (eic/epic#881)

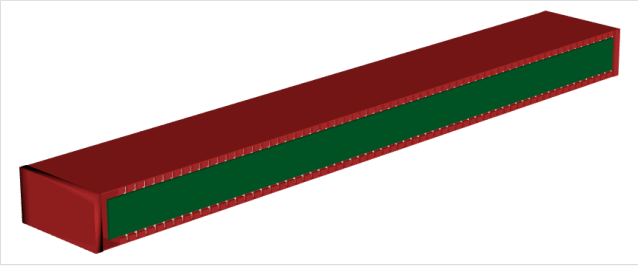
- Adds reduced, beam-test-like detector configurations reusing the **production LFHCAL geometry plugin** (`src/LFHCAL_geo.cpp`):

```
compact/hcal/lfhcal/module_definitions.xml   # shared module definitions
compact/hcal/lfhcal_2024_TB.xml              # 2024 test-beam stack
compact/hcal/lfhcal_2025_TB.xml              # 2025 test-beam stack
compact/hcal/lfhcal_2026_TB_opt1.xml         # planned options...
configurations/lfhcal_2024_TB.yml            # buildable detector configs
configurations/lfhcal_2025_TB.yml
```

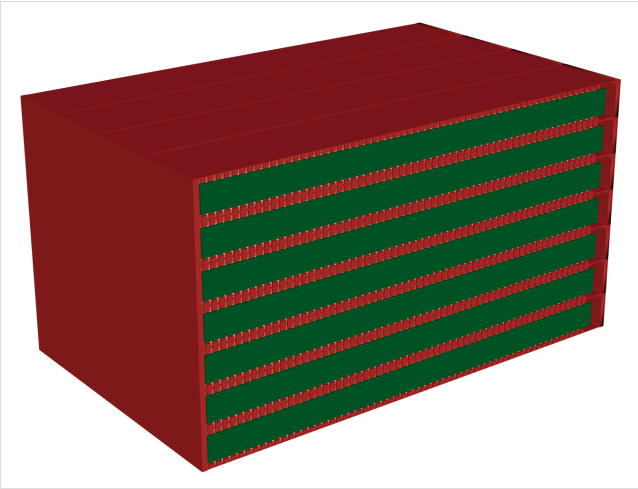
- Each `configurations/*.yml` selects features, e.g.:

```
features:
  hcal:
    lfhcal_2024_TB:
```

- The build then produces a standalone `epic_lfhcal_2024_TB.xml` detector description



LFHCAL reduced beam-test geometry (`lfhcal_2024_TB`)



LFHCAL reduced beam-test geometry (`lfhcal_2028_TB_opt2`)

How-to: a reduced beam-test geometry in DD4hep (1/2)

1. **Create a compact file** describing the test stand, reusing the production detector plugin and readout:

```xml
<!-- compact/hcal/lfhcal_2024_TB.xml -->
<lccdd>
  <include ref="lfhcal/module_definitions.xml"/>
  <detectors>
    <detector id="1" name="LFHCAL" type="epic_LFHCAL"
              readout="LFHCALHits">
      <!-- number of modules / stack layout as installed
           at the test beam; positions of the prototype -->
    </detector>
  </detectors>
  <readouts> <!-- same segmentation as production --> </readouts>
</lccdd>
```

2. **Declare a configuration** so it builds as its own detector:

```yaml
# configurations/lfhcal_2024_TB.yml
features:
  hcal:
    lfhcal_2024_TB:
```

How-to: a reduced beam-test geometry in DD4hep (2/2)

3. **Build** — every YAML configuration becomes an `epic_*.xml`:

```
cmake -B build -S . -DCMAKE_INSTALL_PREFIX=install
cmake --build build -- install
ls install/share/epic/epic_lfhcal_2024_TB.xml
```

4. **Simulate** with the standard tools (a particle gun stands in for the beam):

```
source install/bin/thisepic.sh
npsim --compactFile $DETECTOR_PATH/epic_lfhcal_2024_TB.xml \
      --enableGun --gun.particle pi- --gun.energy "5*GeV" \
      --numberOfEvents 1000 --outputFile tb_pi-_5GeV.edm4hep.root
```

5. **Reconstruct & compare** with the same EICrecon digitization used in production:

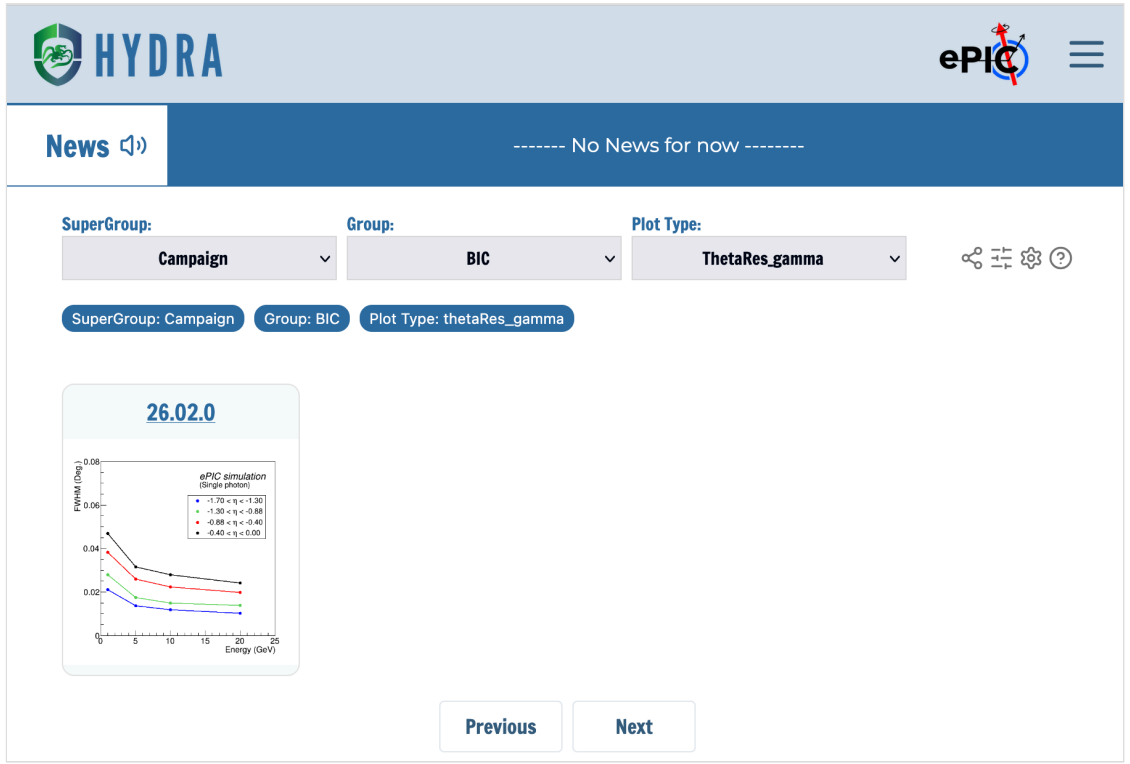
```
eicrecon -Pdd4hep:xml_files=epic_lfhcal_2024_TB.xml \
         -Ppodio:output_file=tb_reco.root tb_pi-_5GeV.edm4hep.root
```

→ overlay simulated response with beam-test data; tune SiPM/readout model parameters; commit the comparison as a benchmark

Summary

- ePIC maintains a **broad suite of calorimetry benchmarks**: backward/barrel/forward ECal, backward/forward HCal + insert, and the ZDC — from hit-level sanity checks to physics-signature reconstruction
- Benchmarks run as **CI on every change** to geometry, digitization and reconstruction: performance is continuously monitored, regressions are caught before merging
- **Beam-test geometries as code** (eic/epic#881): as-tested configurations live in the same DD4hep repository
- Adding a benchmark is straightforward: `Snakefile` + `config.yml` + analysis script and **it's very easy to ask AI to fit your script/macro** — contributions welcome, specifically:
  - Systems without coverage (B0, BHCal, need updated BIC)
  - Codes used for preTDR plots (a nice way to satisfy analysis preservation requirement)
  - **Benchmarks that target results of beam tests** (please, talk to us at Validation WG!)
- Many thanks to Authors of the nice benchmarks already in the suite!

Validation with Hydra



- **Hydra** is a machine-learning based data-quality monitoring system: it uses computer-vision models to automatically classify monitoring/validation plots as good or bad and flag anomalies, reducing reliance on manual inspection
- Developed at JLab (originally for detector monitoring, e.g. GlueX) and is now deployed for ePIC
- Integration of `epicprod` with Hydra is ongoing towards **automated campaign validation**

Thank you

github.com/eic/detector_benchmarks
eic.github.io/tutorial-developing-benchmarks