NEC Update | Reminder & Caveats

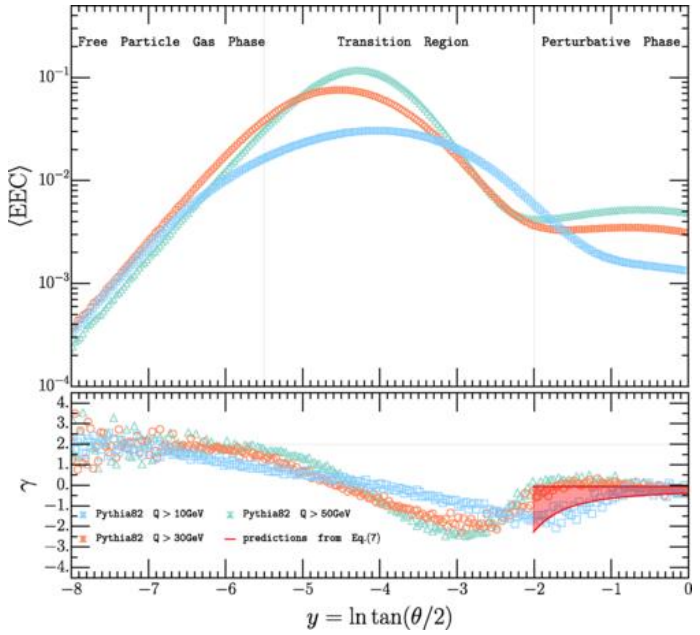
Nucleon-Energy Correlators (NEC)

$$\mathrm{NEC} = \sum_i \int d\sigma(x_B, Q^2, p_i) x_B^{N-1} \frac{E_i}{E_p} \delta(\theta - \theta_i)$$

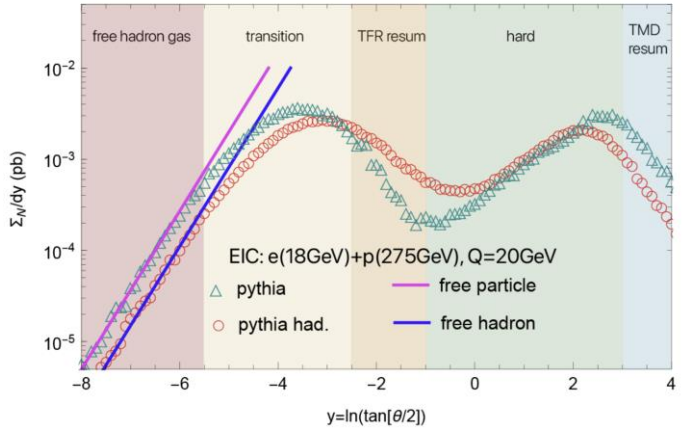
- E_i, θ_i = energy, Breit frame angle of i[th] particle
- E_p = energy of scattered proton

- **Code Development Ongoing:** see the repo here
  - Testing on 25.06.1 NC DIS 10x100 sample
  - Using 1 file (just under 700ish events)

- **Inputs:**
  - Reconstructed/Generated Inclusive Kinematics
    - 👉 Using electron method for now
  - Reconstructed/Generated Breit Frame particles
    - 👉 **Removing DIS electron still to-do!!**
    - 👉 Only full (tracks + clusters) available…
    - › Probably should open a PR for charged-only Breit Frame particles + Centauro jets…

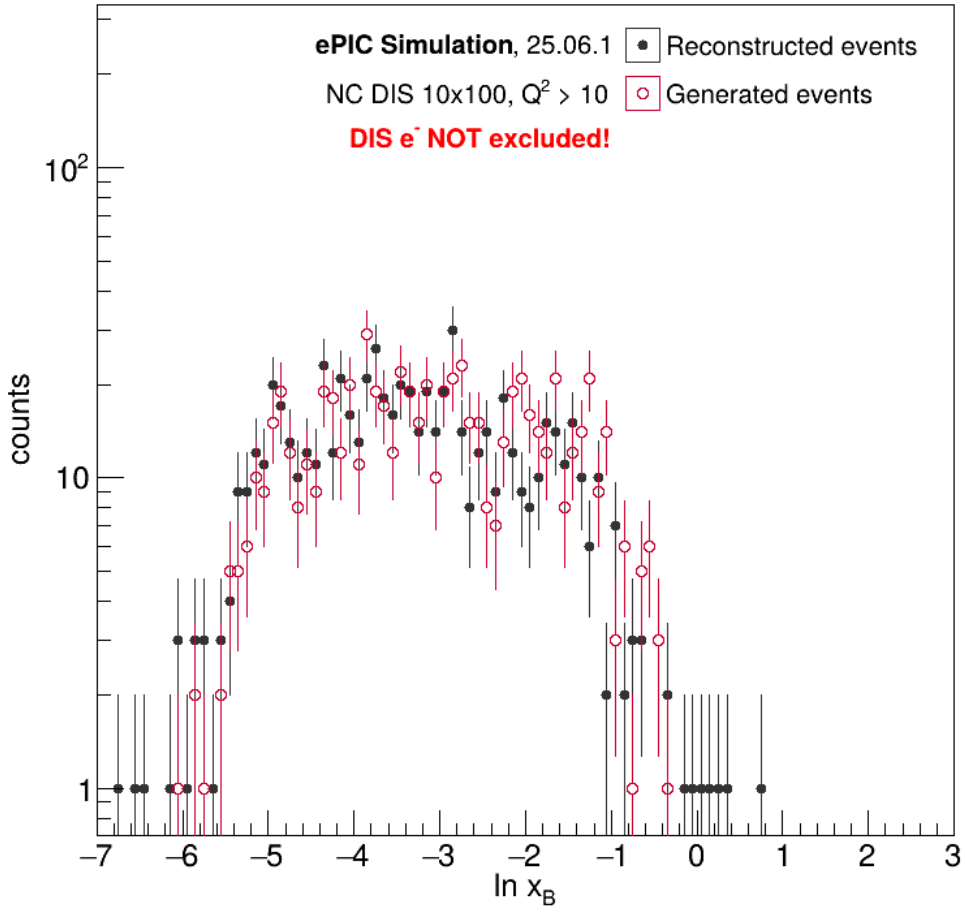


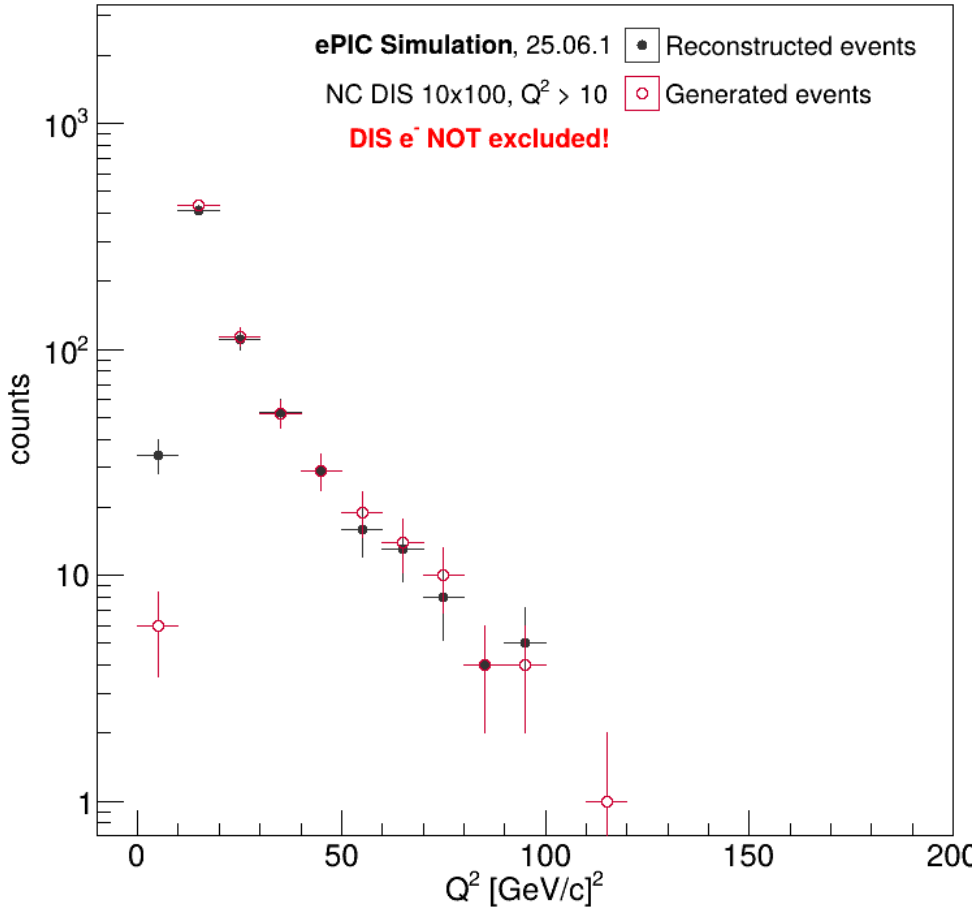
PRL 130, 091901



arXiv:2312.07655

NEC Update | Event Kinematics

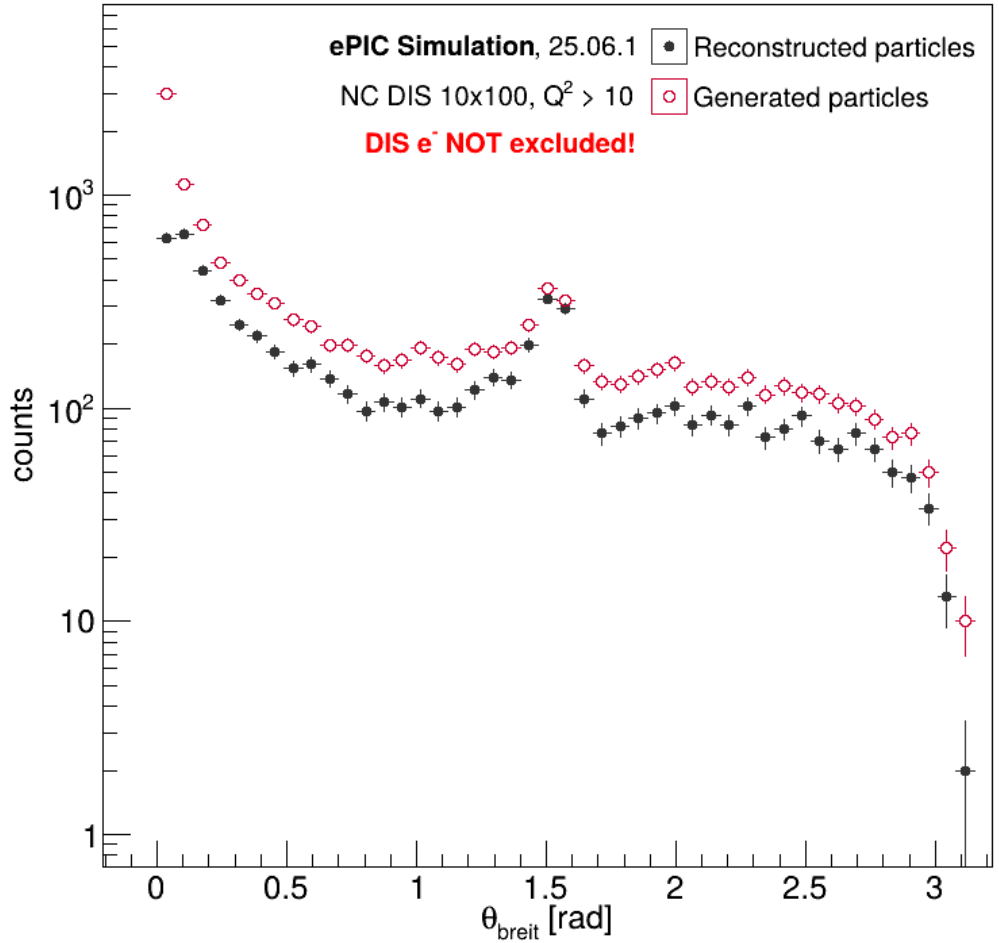




NEC Update | Particle Breit Frame Angles

- **Right:** calculated Breit Frame angle for reconstructed & generated particles
  - i.e. the polar angle in the Breit Frame, so polar angle wrt virtual photon
    - ☞ Virtual photon should be at z = 0 in frame, right?
  - **Weird feature at pi/2:** the scattered electron?

```
// extract particle breit angle
//   - n.b. by definition, the beam is at z = 0 in the breit frame
//   - TODO expand this to extract additional information
auto getBreitAngles = [](const std::vector<edm4eic::ReconstructedParticleData>& pars) {
  std::vector<float> angles;
  for (const auto& par : pars) {
    angles.push_back(
      edm4hep::utils::anglePolar(
        edm4hep::Vector3f(par.momentum.x, par.momentum.y, par.momentum.z)
      )
    );
  }
  return angles;
};
```



NEC Update | Particle Breit Frame “Rapidity”

- **Right:** calculated Breit Frame “rapidity” for reconstructed & generated particles
  - i.e. the polar angle in the Breit Frame, so polar angle wrt virtual photon
    - ☞ Virtual photon should be at z = 0 in frame, right?
  - **Weird feature at pi/2:** the scattered electron?

```
// calculate particle rapidity
//   - TODO collect this into a struct and merge with
//     getBreitAngle lambda
auto getRapidity = [](const std::vector<float>& angles) {
  std::vector<float> raps;
  for (const float angle : angles) {
    raps.push_back(
      std::log(std::tan(angle/2))
    );
  }
  return raps;
};
```