Iguana

Algorithm Details:

- **Momentum Corrections**
- **Inclusive Kinematics**

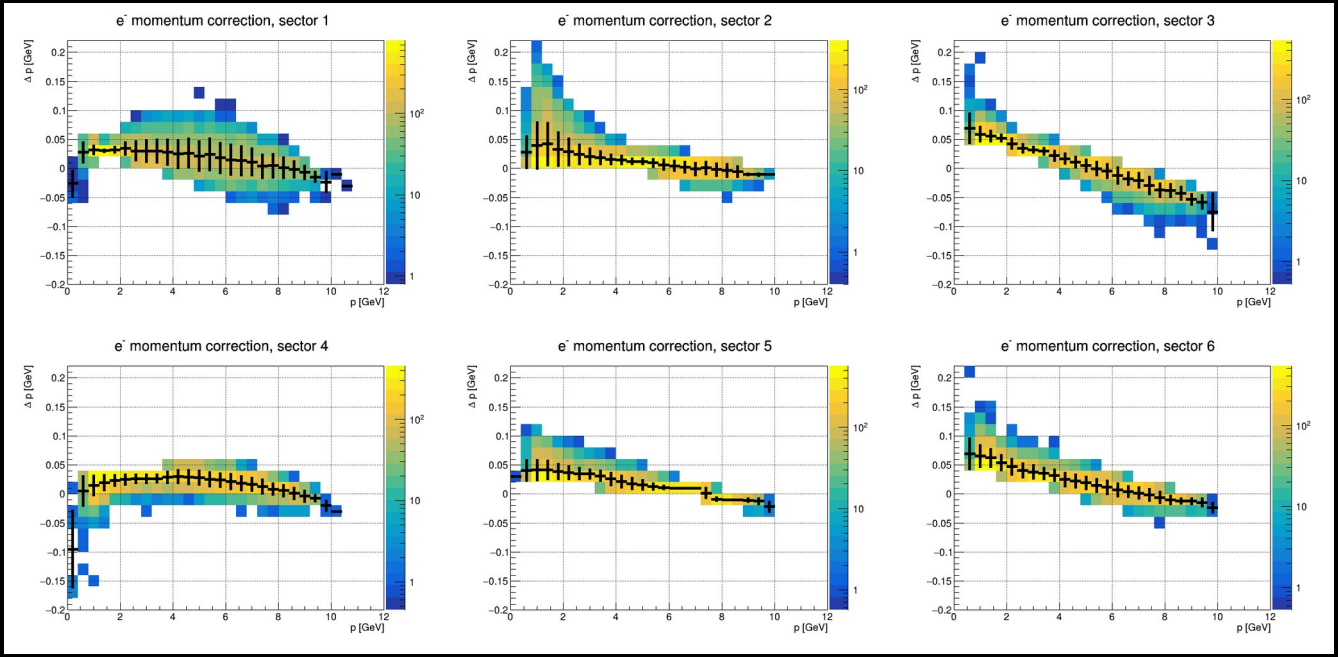
Jefferson Lab

clas

Christopher Dilks

Momentum Corrections

Implementation of Richard Capobianco's code from
https://clasweb.jlab.org/wiki/index.php/CLAS12_Momentum_Corrections#tab=Correction_Code



- Applies to electron, π^+, π^-, and proton
  - Proton includes energy-loss correction
  - For both inbending and outbending configurations
- The momentum correction validator just makes plots (left figure), so its up to a human to check them
- Richard C. is working on cross-checking this implementation

Momentum Corrections

This is a Transformer-type algorithm

Input Banks

- `RUN::config`
- `REC::Particle`
- `REC::Particle::Sector`
  `(created by SectorFinder algorithm)`

Momentum Corrections

Modified Output Bank

- `REC::Particle`,

  with corrected momenta

Momentum Corrections

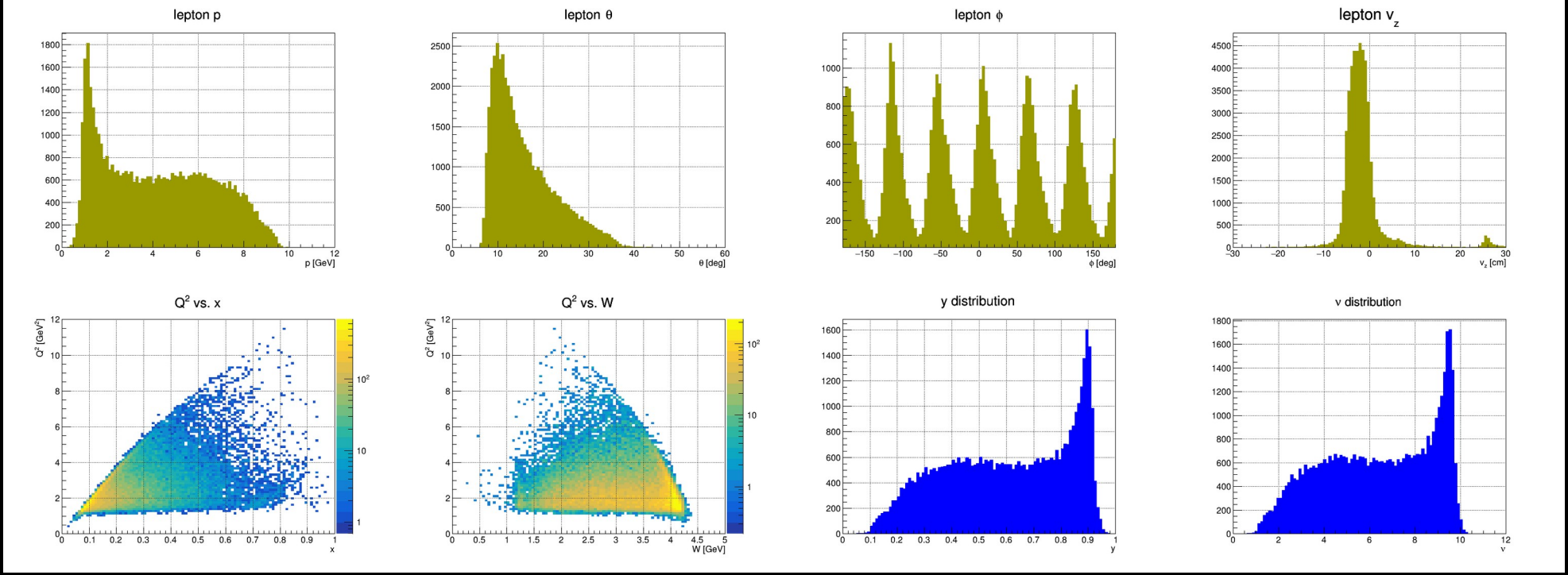
Action Functions:

- Transform() calls the other functions, returning the corrected 3-momentum
- Use the other functions if you just want the correction factor

```
/// @action_function{scalar transformer} Apply the momentum correction
/// @returns the transformed momentum
vector3_t Transform(vector_element_t px, vector_element_t py, vector_element_t pz, int sec, int pid, float torus) const;

/// @action_function{scalar creator} Calculate the correction factor for inbending data
/// @returns the correction factor
double CorrectionInbending(vector_element_t const Px, vector_element_t const Py, vector_element_t const Pz, int const sec, int const pid) const;

/// @action_function{scalar creator} Calculate the correction factor for outbending data
/// @returns the correction factor
double CorrectionOutbending(vector_element_t const Px, vector_element_t const Py, vector_element_t const Pz, int const sec, int const pid) const;

/// @action_function{scalar creator} Energy loss correction for inbending data
/// @returns the correction factor
double EnergyLossInbending(vector_element_t const Px, vector_element_t const Py, vector_element_t const Pz, int const pid) const;

/// @action_function{scalar creator} Energy loss correction for outbending data
/// @returns the correction factor
double EnergyLossOutbending(vector_element_t const Px, vector_element_t const Py, vector_element_t const Pz, int const pid) const;
```

Inclusive Kinematics

The inclusive kinematics algorithm calculates various quantities, such as x and Q^2



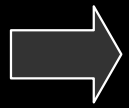
Inclusive Kinematics

This is a Creator-type algorithm

Input Banks

- `REC::Particle`

Inclusive Kinematics Reconstruction



New Output Bank

- `physics::InclusiveKinematics`, with x, Q^2, W, etc.

Action Functions:

- Returns a ‘struct’ with all the variables

```
/// @action_function{scalar creator} compute kinematics from the scattered lepton.
/// @param lepton_px scattered lepton momentum component @f$p_x@f$ (GeV)
/// @param lepton_py scattered lepton momentum component @f$p_y@f$ (GeV)
/// @param lepton_pz scattered lepton momentum component @f$p_z@f$ (GeV)
/// @returns the reconstructed inclusive kinematics in a `iguana::physics::InclusiveKinematicsVars` instance
InclusiveKinematicsVars ComputeFromLepton(
    vector_element_t lepton_px,
    vector_element_t lepton_py,
    vector_element_t lepton_pz) const;
```

Inclusive Kinematics

iguana::physics::InclusiveKinematics **creates** and fills a **new** bank (schema)

Example for 1 event:

```
BANK :: NAME physics::InclusiveKinematics , ROWS      1
          pindex :         0
              Q2 :   4.24874
               x :   0.52003
               y :   0.41074
               W :   2.19131
              nu :   4.35387
              qx :   0.47320
              qy :  -1.49650
              qz :   4.55428
              qE :   4.35387
```

index of the scattered electron

virtual photon momentum

Inclusive Kinematics

Configuration:

- **TODO:** Beam energy should NOT be a config parameter (this is a stopgap until we have a more automated way, e.g., RCDB)
- So far the only reconstruction method is the scattered-electron method
  - … and so far the only scattered-electron finder is one that looks for the highest-E trigger electron in the Forward Detector
  - Contributions for other reconstruction methods or electron finders are welcome!

```
physics::InclusiveKinematics:

  initial_state: # FIXME: add run ranges
    - default:
      beam_energy: 10.6
      beam_direction: [ 0.0, 0.0, 1.0 ]
      beam_particle: electron
      target_particle: proton

  method:
    reconstruction: scattered_lepton
    lepton_finder: highest_energy_FD_trigger
```