Track Fitting and Vertexing EDM

Joe Osborn
BNL
October 11, 2023

Overview

- Some discussion last week about the output collection from the fitter
- Issue - currently 3 outputs from the fitter, all of which contain (effectively) the same information
- Has downstream implications for all other reconstruction algorithms using tracks (electron finder, PID, vertexing…)
- Will propose a solution for the fitting output, and then a corresponding design for the vertexing EDM

CKF Output

- Current output:
    1. edm4eic::TrajectoryCollection - contains all track states
    2. edm4eic::TrackParametersCollection - contains track parameters at the target surface
    3. std::vector<ActsExamples::Trajectories> - An Acts EDM object which packages parameters and states for use by other Acts algorithms
        1. Note: Acts development team moving away from this object
- Proposal:
    - Single output of edm4eic::TrackCollection
        - This will contain the track parameters at the target surface, all track states and associated hits (eventually, measurements), and a vertex association
        - Missing a field for track position at target surface, which should be added
        - Additional fields from edm4eic::Trajectory can be added, e.g. nStates, nOutliers etc. (these are inspired from the Acts trajectory object)

```
edm4eic::Track:
  Description: "Track information at the vertex"
  Author: "S. Joosten"
  Members:
    - int32_t           type            // Flag that defines the type of track
    - float             chi2            // Total chi2 (sum) of the track fit
    - int32_t           ndf             // Numbers of degrees of freedom of the track fit
    - edm4hep::Vector3f momentum        // Track 3-momentum at the vertex [GeV]
    - edm4eic::Cov3f    momentumError   // Covariance matrix on the momentum
    - float             time            // Track time at the vertex [ns]
    - float             timeError       // Error on the track vertex time
    - float             charge          // Particle charge
  OneToOneRelations:
    - edm4eic::Trajectory trajectory    // Trajectory of this track
    - edm4eic::Vertex   vertex          // Track vertex of this track
  OneToManyRelations:
    - edm4eic::TrackerHit trackerHits   // Hits that were used for this track
    - edm4eic::Track    tracks          // Tracks (segments) that have been combined to create
```

Advantages/Disadvantages

- Advantages
  - Single output container that is defined within our EDM, so we maintain control and are not affected by external changes (e.g. Acts updates)
  - Contains all track information that will realistically be needed by any downstream algorithm or analysis
- Disadvantages
  - Have to use additional CPU time to swap in between edm4eic and Acts::EDM for other tracking algorithms (e.g. vertexing, track projections, whatever else comes along)
    - Ultimately a small price to pay to insulate ourselves from external changes

Vertex EDM

```
## ============================================================================
## Vertexing
## ============================================================================

edm4eic::Vertex:
  Description: "EIC vertex"
  Author: "W. Armstrong, S. Joosten, based off EDM4hep"
  Members:
    - int32_t             primary       // Boolean flag, if vertex is the primary vertex of the event
    - float               chi2          // Chi-squared of the vertex fit
    - float               probability   // Probability of the vertex fit
    - edm4hep::Vector3f   position      // [mm] position of the vertex.
    ## this is named "covMatrix" in EDM4hep, renamed for consistency with the rest of edm4eic
    - edm4eic::Cov3f      positionError // Covariance matrix of the position
    - int32_t             algorithmType // Type code for the algorithm that has been used to create the vertex - check/set the colle
    ## Additional parameter not in EDM4hep: vertex time
    - float               time          // Vertex time
  VectorMembers:
    - float               parameters    // Additional parameters related to this vertex - check/set the collection parameter "Vertex
  OneToOneRelations:
    ## @TODO: why one and not multiple particles?
    - edm4eic::ReconstructedParticle associatedParticle // reconstructed particle associated to this vertex.
```

- Current vertex object is missing several notable fields
    - Missing time covariance, primary is not clear in streaming context (what is “the” PV?), should have relations to many tracks (not a single particle), missing NDF of vertex fit

Vertex EDM

- There is also currently no truth vertex object, or any robust way to do vertex evaluation
  - This will become a critical issue when doing studies with backgrounds, where (e.g.) one may have many truth vertices in a single time frame that could be evaluated
- Robust vertex evaluation will be a longer term task

Proposal for Reco Vertex Object

```
edm4eic::Vertex:
  Description: "EIC vertex"
  Author: "J. Osborn"
  Members:
    - uint32_t            type       // Type flag, to identify what kind of vertex is identified
    - float               chi2          // Chi-squared of the vertex fit
    - float               ndf   // NDF of the vertex fit
    - edm4hep::Vector4f   fullPosition      // [mm] position + time t0 of the vertex.
    - edm4eic::Cov4f      fullPositionError // Covariance matrix of the position
  OneToManyRelations:
    - edm4eic::Track associatedTracks // reconstructed tracks associated to this vertex.
```

- Contains a complete 4D position and covariance
- Contains relations to reconstructed tracks, which is what the CKF will point to (from slide 3)
- Contains chi2/ndf to evaluate goodness of vertex fit

Proposal for Truth Vertex Object

```
edm4eic::GeneratedVertex:
  Description: "Generated EIC vertex"
  Author: "J. Osborn"
  Members:
    - uint32_t            type       // Type flag, to identify what kind of MC vertex is identified (e.g. background vs. primary)
    - edm4hep::Vector4f   position   // [mm] position + time t0 of the MC vertex
  OneToManyRelations:
    - edm4eic::ReconstructedParticle sim        // reference to the Generated particles associated to the GeneratedVertex
```

- Generated vertices are the vertices from final state particles, not MCVertex which is going to be overkill for our purposes
- Only require a 3+1 position+time and a type to identify the vertex
- Note - relation to ReconstructedParticle because currently in our reconstruction a ReconstructedParticleCollection is filled with the actual final state generated particles. See this algorithm
  - Some discussion about whether or not an edm4eic::Vertex object could be used as a GeneratedVertex object. However, we need a truth vertex to point back to truth particles, not edm4eic::tracks

To-Do List

- Change CKF output collection to edm4eic::TrackCollection. This will have downstream implications for all other algorithms that use tracks
- Introduce new Vertex and GeneratedVertex objects
- Alter vertexing algorithm to use this new object
- Write a MC algorithm to fill GeneratedVertexCollection and output with PODIO
- Need an algorithm to relate reconstructed to generated vertices