Reco-to-Truth matching

Andy Furmanski, Tingjun Yang

LArSoft Coordination Meeting

15th August 2017

Matching?

- Most (all?) physics analyses need to determine selection purity and efficiency
- Most physics analyses need to determine resolutions for momentum, angle, etc
  - Needed for any unfolding techniques

- For these, we need particle-by-particle matching from reconstruction to truth

Backtracker

- LArSoft already provides the BackTracker service for matching
  - Match hits to true energy depositions
  - Calculate a matching metric (most energy contributed to hits in object usually)
- What this service does not do:
  - Tell you what metric to use
  - Calculate the metric
  - Function outside of LArSoft (e.g. Gallery)

Standardising

- It makes sense to standardise, within an experiment, how this matching is done
- The fewer lines of code reproduced in many places, the better
  - Changes only need to be made once
- The fewer lines of code written, the better
  - People make mistakes

One module to match them all

- larana/T0Finder/MCTruthMatching_module.cc
- This has existed for some time
  - Inserts an anab::T0 object into the event
  - **Association between recob::Track/Shower and anab::T0**
  - T0 contains the MCParticle ID
    - T0->TriggerBits()
- **Not perfect** though
  - TriggerBits? That's confusing
  - Requires an additional loop through all G4 particles
    - For every track and shower
  - No support for other objects (currently)

Direct associations

- Can directly associate the reco object (track, shower) to the MCParticle
- Can also add auxiliary information (match cleanliness, completeness)

- I have begun making these modifications
  - feature/afurmans_MCTruthMatching
  - larana and lardataobj

Outline of changes (1)

- add association between recob::Track and simb::MCParticle
- add association between recob::Track and simb::MCParticle
- add association between PFParticle and simb::MCParticle
  - NEW – previously did not touch PFParticles
  - new fcl parameter determines PFParticle producer to use. Default set so change is non-breaking

Outline of changes (2)

- Associations will contain auxiliary data
  - std::vector<double, std::string>
  - double contains value, string contains name
  - values are cleanliness and completeness of object
- Can be filled in either order, as desired
- Flexible enough to easily add other data if desired

- All non-breaking