ParticleID Data Product Update

Larsoft Coordination Meeting
Dec. 14, 2021

H. Greenlee

Outline

- ParticleID update history and review
- Experiment-specific code
- Results from tests using larsoft v09_36_00_02
- Merge requests

History

- Original presentation for proposed anab::ParticleID update (breaking change) was made by Adam Lister and Kirsty Duffy in July 17, 2018 larsoft coordination meeting (link to agenda).
- Second presentation on anab::ParticleID update was made in Jan. 15, 2019 larsoft coordination meeting (link to agenda).
  - Agreed to in principle, pending implementation of ioread rule.
  - Authors provided merge branches for larsoft and experiment code.
- MicroBooNE forked MCC9 off of larsoft v08_05_00 (Jan. 16, 2019).
  - ParticleID update was eventually merged into MCC9 fork.
- Updated larsoft merge branches and I/O rule presented in Nov. 2, 2021 larsoft coordination meeting (link to agenda).
  - Test build larsoft v09_36_00_02.
- Additional tests and merge requests (today's meeting).

Old ParticleID Class

- The old (integration release) anab::ParticleID class is hard-wired to store values from a fixed set of particle id algorithms.

```
class ParticleID{
public:

  ParticleID();

  int    fPdg;                ///< determined particle ID
  int    fNdf;                ///< ndf for chi2 test
  double fMinChi2;            ///< Minimum reduced chi2
  double fDeltaChi2;          ///< difference between two lowest reduced chi2's
  double fChi2Proton;         ///< reduced chi2 using proton template
  double fChi2Kaon;           ///< reduced chi2 using kaon template
  double fChi2Pion;           ///< reduced chi2 using pion template
  double fChi2Muon;           ///< reduced chi2 using muon template
  double fMissingE;           ///< missing energy from dead wires for contained particle
  double fMissingEavg;        ///< missing energy from dead wires using average dEdx
  double fPIDA;               ///< PID developed by Bruce Baller
  geo::PlaneID fPlaneID;
```

New ParticleID Class

- The proposed new (and MCC9) anab::ParticleID class holds an arbitrary size collection of algorithm structs.

```
class ParticleID{
public:

  ParticleID();

  std::vector<sParticleIDAlgScores> fParticleIDAlgScores;
                  ///< Vector of structs to hold outputs from generic PID algorithms
```

Particle ID Algorithm Struct

```
struct sParticleIDAlgScores { ///< determined particle ID
  std::string fAlgName;
     ///< Algorithm name (to be defined by experiment). Set to "AlgNameNotSet" by default.
  kVariableType fVariableType;
     ///< Variable type enum: defined in ParticleID_VariableTypeEnums.h. Set to kNotSet by default.
  kTrackDir fTrackDir;
     ///< Track direction enum: defined in ParticleID_VariableTypeEnums.h. Set to kNoDirection by
          default.
  int fNdf;
     ///< Number of degrees of freedom used by algorithm, if applicable. Set to -9999 by default.
  int fAssumedPdg;
     ///< PDG of particle hypothesis assumed by algorithm, if applicable. Set to 0 by default.
  float fValue; ///< Result of Particle ID algorithm/test
  std::bitset<8> fPlaneMask;
     ///< Bitset for PlaneID used by algorithm, allowing for multiple planes and up to 8 total
          planes. Set to all 0s by default. Convention for bitset is that fPlaneMask[0] (i.e. bit 0)
          represents the collection plane, and then other planes work outwards from there.

  sParticleIDAlgScores(){
  fAlgName = "AlgNameNotSet";
  fVariableType = kNotSet;
  fTrackDir = kNoDirection;
  fAssumedPdg = 0;
  fNdf = -9999;
  fValue = -9999.;
  // fPlaneMask will use default constructor: sets all values to 0
  }
};
```

Advantages of New vs. Old anab::ParticleID

- Extensible.
  - New particle id algorithms can be added without modifying the anab::ParticleID class.
- Possibility of having particle id algorithms based on multiple planes.

I/O Read Rule

- The particle id update comes with an I/O read rule to convert the old version of ParticleID into the new version. Converted ParticleID objects contain nine algorithms with values corresponding to the following data members of the old class.

```
class ParticleID{
public:

  ParticleID();

  int    fPdg;                 ///< determined particle ID
  int    fNdf;                 ///< ndf for chi2 test
  double fMinChi2;             ///< Minimum reduced chi2
  double fDeltaChi2;           ///< difference between two lowest reduced chi2's
  double fChi2Proton;          ///< reduced chi2 using proton template
  double fChi2Kaon;            ///< reduced chi2 using kaon template
  double fChi2Pion;            ///< reduced chi2 using pion template
  double fChi2Muon;            ///< reduced chi2 using muon template
  double fMissingE;            ///< missing energy from dead wires for contained particle
  double fMissingEavg;         ///< missing energy from dead wires using average dEdx
  double fPIDA;                ///< PID developed by Bruce Baller
  geo::PlaneID fPlaneID;
```

Larsoft anab::ParticleID Dependencies

- lardataobj.
  - ParticleID class.
  - ParticleID enum header (added).
  - classes_def.xml (ioread rule).
- larana
  - Chi2PIDAlg algorithm class.
  - Chi2ParticleID_module.cc
- lardata
  - DumpParticleIDs_module.cc
- larreco
  - KalmanFilterFitTrackMaker_tool.cc (pdg accessor).
- lareventdisplay
  - AnalysisBaseDrawer.cxx (commented out).

Larsoft anab::ParticleID Dependencies

- lardataobj.
  - ParticleID class.
  - ParticleID enum header (added).
  - classes_def.xml (ioread rule).
- larana
  - Chi2PIDAlg algorithm class.
  - Chi2ParticleID_module.cc
- lardata
  - DumpParticleIDs_module.cc
- larreco
  - KalmanFilterFitTrackMaker_tool.cc (pdg accessor).
- lareventdisplay
  - AnalysisBaseDrawer.cxx (commented out).

Data product class and I/O rule

Larsoft anab::ParticleID Dependencies

- lardataobj.
  - ParticleID class.
  - ParticleID enum header (added).
  - classes_def.xml (ioread rule).
- larana
  - Chi2PIDAlg algorithm class.
  - Chi2ParticleID_module.cc
- lardata
  - DumpParticleIDs_module.cc
- larreco
  - KalmanFilterFitTrackMaker_tool.cc (pdg accessor).
- lareventdisplay
  - AnalysisBaseDrawer.cxx (commented out).

Producer module and algorithm class

Larsoft anab::ParticleID Dependencies

- lardataobj.
  - ParticleID class.
  - ParticleID enum header (added).
  - classes_def.xml (ioread rule).
- larana
  - Chi2PIDAlg algorithm class.
  - Chi2ParticleID_module.cc
- lardata
  - DumpParticleIDs_module.cc
- larreco
  - KalmanFilterFitTrackMaker_tool.cc (pdg accessor).
- lareventdisplay
  - AnalysisBaseDrawer.cxx (commented out).

Consumers

ParticleID Producer Module Update

- Larsoft has one producer module, and associated algorithm class, that produces the old version of anab::ParticleID.
  - Chi2ParticleID_module.cc in larana.
- No experiment-specific code produces the old version of anab::ParticleID.
- The original authors (Kirsty Duffy and Adam Lister) supplied a merge branch with an updated version of Chi2ParticleID_module.cc that natively produces the new version of anab::PartibleID.
  - The new version of Chi2ParticleID_module.cc only contains five algorithms, rather than the nine that are obtainable from the I/O rule.
  - This is the version that is included in larsoft v09_36_00_02.

ParticleID Producer Module Update (cont.)

- The five algorithms produced by the updated producer module are:
  - Four chi2 algoirthms (Chi2Muon, Chi2Pion, Chi2Kaon, Chi2Proton).
  - PIDA.
- The four missing algorithms are:
  - MinChi2 and DeltaChi2.
    - These algorithms are redundant (obtainable from other Chi2 algos).
  - MissingE and MissingEavg.
    - I think the reason why these were not included was that experiments didn't find them very useful (more on this later).

Experiment anab::ParticleID Dependencies

- In MCC9, MicroBooNE uses its own producer module that natively produces the new version of ParticleID (in ubana).
- argoneutcode
  - AnalysisTree/AnalysisTreeT962_module.cc
- dunetpc
  - dune/AnaTree/AnalysisTree_module.cc
  - dune/Protodune/dualphase/AnaRootParser_module.cc
- sbndcode
  - sbndcode/AnalysisTree/AnalysisTree_module.cc
- icaruscode
  - icaruscode/Analysis/AnalysisTree_module.cc

Experiment anab::ParticleID Dependencies (cont.)

- sbncode
  - sbncode/CAFMaker/FillReco.cxx
  - sbncode/PID/Dazzle_module.cc
  - sbncode/TPCReco/CalorimetryAnalysis/CalorimetryAnalysis_module.cc
- lariatsoft
  - No longer being maintained.

Experiment anab::ParticleID Dependencies (cont.)

- sbncode
  - sbncode/CAFMaker/FillReco.cxx
  - sbncode/PID/Dazzle_module.cc
  - sbncode/TPCReco/CalorimetryAnalysis/CalorimetryAnalysis_module.cc

  CAFMaker
- lariatsoft
  - No longer being maintained.

Experiment anab::Dependencies Summary

- In experiment-specific repositories (other than lariatsoft and MicroBooNE), there are eight modules or other files that depend on the anab::ParticleID data product.
  - Five of the analyzers are based on or variations of AnalysisTree.
    - All analyzer modules listed on page 13 or of this type.
  - The remaining three files and modules are owned by sbnutil.
    - New since original merge request in Jan., 2019.
      - No merge branch from original authors.
    - Two are are used in common SBN CAFMaker.
    - The remaining sbncode analyzer (CalorimetryAnalysis_module.cc) is its own ntuple-maker, distinct from AnalysisTree and CAFMaker.

Updating Experiment-Specific Code

- In addition to supplying merge branches for larsoft repos, the original authors supplied merge branches for five experiment-specific repositories (argoneutcode, dunetpc, sbndcode, icaruscode, and lariatsoft).
  - Original branches named feature/kduffy_updatePIDdataprod.
- Since the original merge request in 2019:
  - Sbndcode and icaruscode migrated to github.
  - New repository sbnutil added for code common to SBND and Icarus.
    - Including CAFMaker.
  - Lariatsoft has stopped being maintained.
- I generated an updated set of merge branches in an appropriate repo (github or redmine).
  - New branch name greenlee_pid_update.
  - All updated experiment branches tested (at least) to merge and build clean.

Updating Experiment-Specific Code (cont.)

- Each of the eight updated files have been updated to access only the five algorithms that are available from the version of anab::ParticleID produced by the updated producer module.
  - In cases where MinChi2 or DeltaChi2 was being accessed previously, the updated code calculates these quantities by looping over the other Chi2 algorithms.
  - The only larsoft or experiment-specific code that was accessing the MissingE data members from the old version of anab::ParticleID was argoneut's version of AnalysisTree (AnalysisTreeT962_module.cc).
    - In the updated version argoneutcode, the MissingE algorithms are simply removed (filled with zero in the resulting AnalysisTree ntuple).
    - This is the only backward-incompatible change that I know of in the existing consumers of anab::ParticleID.
- If someone strongly objects to the above way of doing things (removing four algorithms from new producer), the producer module could be revisited.

Testing Larsoft v09_36_00_02

- Following the Nov. 2, 2021 larsoft coordination meeting, I requested a test build of larsoft that included merge branches pertaining to updating ParticleID (plus a few other updates).
  - Larsoft v09_36_00_02.
- I did a number of tests using this release involving several of the experiment-specific ntuple-makers.
  - Ran old ntuple-maker on data produced using old version of larsoft.
    - Based on larsoft v09_36_00.
  - Ran new ntuple-maker on data produced using old version of larsoft.
    - Tests I/O rule.
  - Ran updated ntuple-maker on data produced using new version of larsoft.
    - Tests updated ParticleID producer module.
  - Compared above three ntuples.

Testing Larsoft v09_36_00_02 (cont.)

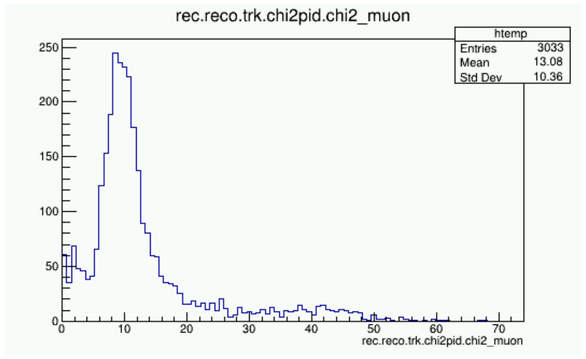
- I did not test all eight ntuple makers.
  - In some cases I could not get certain ntuple makers to run.
- I was able to test three ntuple makers.
  - SBND version of AnalysisTree.
  - CAFMaker (build in sbndcode release, should be same as icaruscode).
    - Including standard ParticleID and Dazzle-produced objects.
- Criteria for a successful test was that identical results for all affected variables in all three ntuples.
- Updates are basically the ones supplied by Kirsty and Adam.
  - In some cases I did a few tweaks (mainly involving handling of invalid values), to make the results identical to the old version.
- I supplied the updates and merge branches for sbnutil.

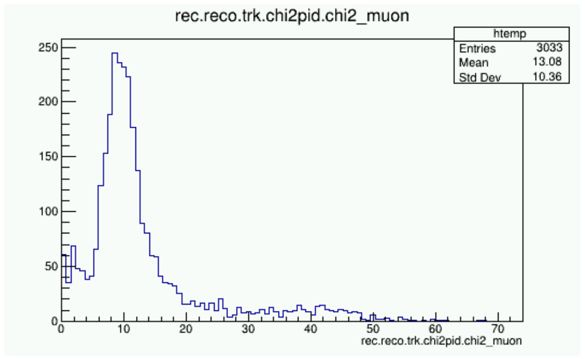
Example Plots (CAFMaker)

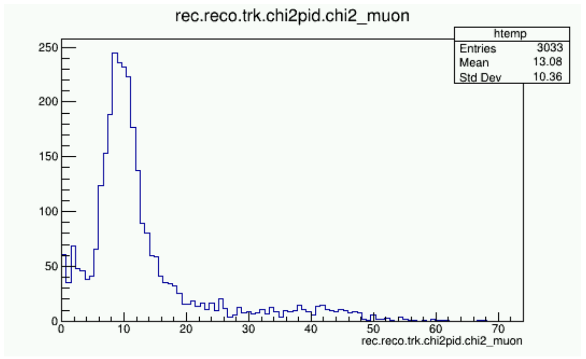
Ntuple 1
old/ old

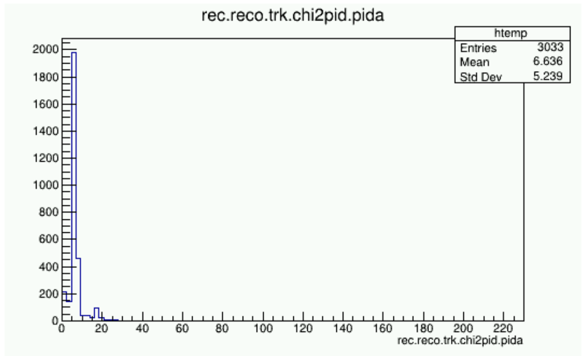
Ntuple 2
old/ new

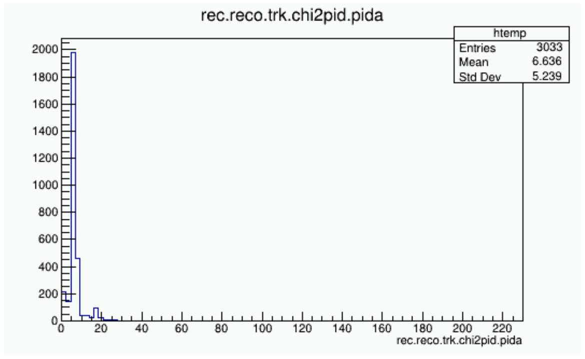
Ntuple 3
new/ new

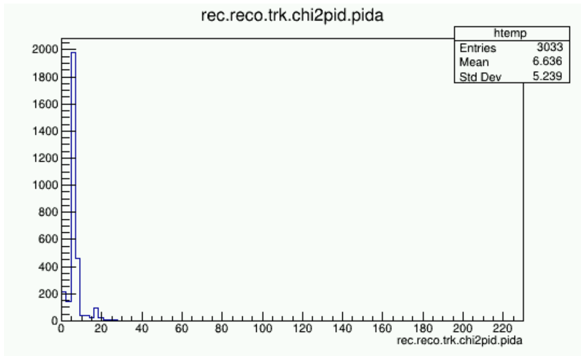










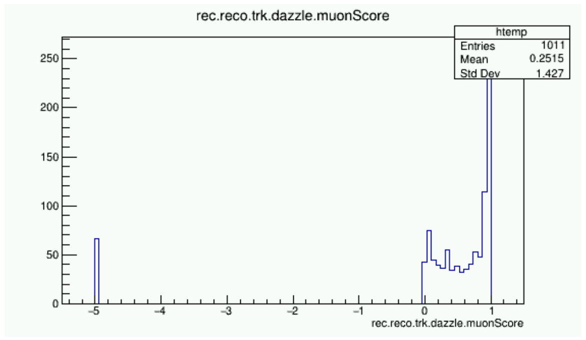


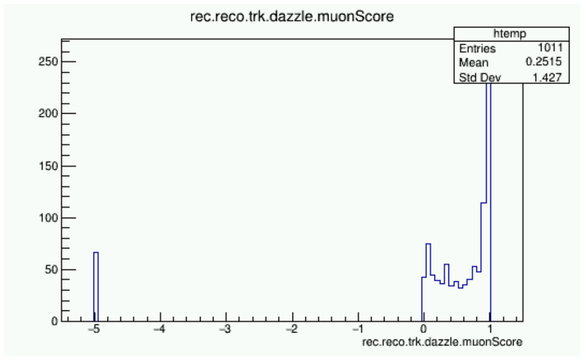
Example Plots (CAFMaker)

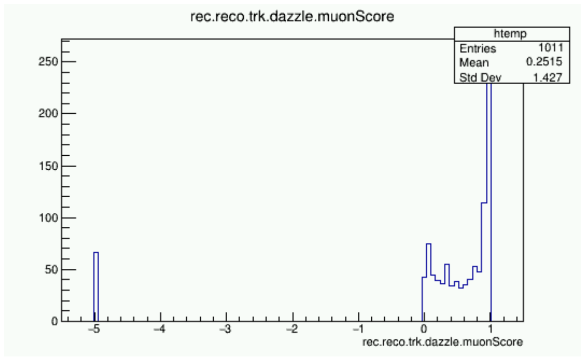
Ntuple 1
old/ old

Ntuple 2
old/ new

Ntuple 3
new/ new







Larsoft Merge Branches

- Here are the merge branches included in larsoft v09_36_00_02.
  - lardataobj – greenlee_mcc9_pid
  - lardata – greenlee_mcc9_pid
  - larreco – greenlee_mcc9_pid, greenlee_mcc9_mcs
  - larana – greenlee_mcc9_pid
  - lareventdisplay – greenlee_mcc9_pid
  - larpandora – greenlee_mcc9_event_building
- Above branches are updated to latest version of develop branch, and test built against larsoft v09_39_01.
  - Did not get any merge conflicts or build errors.
  - Did not add any other updates.
- No pull requests as yet.

Experiment Merge Branches

- Argoneutcode, dunetpc, sbndcode, icaruscode, sbncode.
  - Branch greenlee_pid_update, either in uboone github fork (for repos hosted on github), or in redmine.
- Uboone suite branches.
  - Branch greenlee_mcc9_develop.

Requests

- We request to merge the larsoft and experiment-specific branches mentioned in previous two pages in the next integration release of larsoft.
  - ParticleID data product update.
  - MCS fitter update (see backup).
  - Pandora event building update (backup).

Backup

Additional MCC9 Updates - MCS Fitter

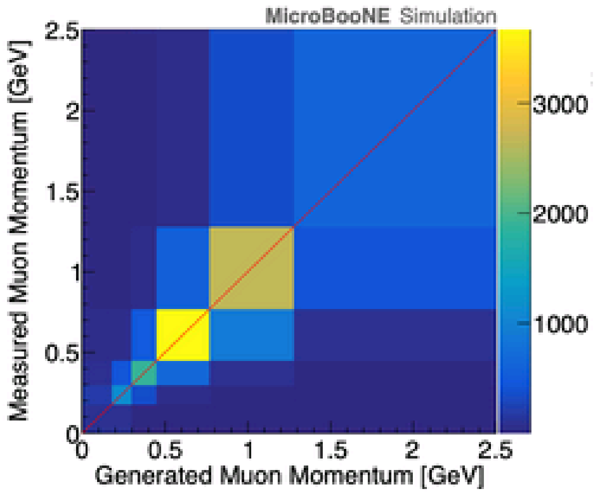
- Author Giuseppe Cerati.
- Larreco updates:
  - TrajectoryMCSFitter algorithm class.
  - mcsfitproducer.fcl
  - MCSFitProducer_module.cc not updated.
- Larreco merge branch: greenlee_mcc9_mcs

MCS momentum track fit in LArSoft

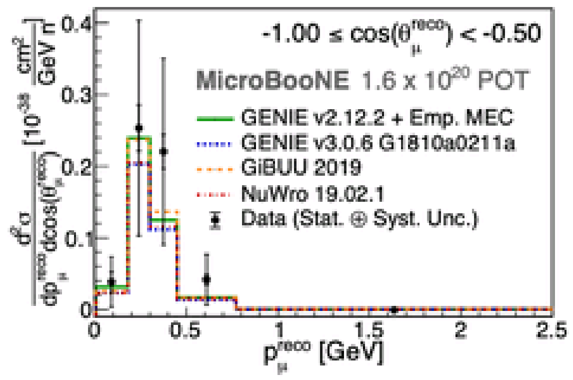
- MCS momentum track fit split the track into segments and performs a likelihood scan over scatterings angles defined between segments thus fitting for the initial track momentum
  - only method available to estimate energy of exiting tracks

- Description in MicroBooNE paper JINST 12 (2017) 10, P10010
  - performance in data studied and presented in MICROBOONE-NOTE-1049-PUB
  - used in several physics analyses, e.g. CC inclusive paper

- Fitter code in LArSoft is in larreco/RecoAlg/TrajectoryMCSFitter
  - called from larreco/TrackFinder/MCSFitProducer_module
  - output stored in lardataobj/RecoBase/MCSFitResult.h



Phys. Rev. Lett. 123, 131801 (2019)



Updates to MCS algorithm

- Update to MCS momentum fit code from MicroBooNE MCC9 development
  - larreco branch greenlee_mcc9_mcs
- Several improvements to the algorithm, all configurable so previous behaviour is retained:
  - Perform a double raster scan, first a coarse one over full range and then a fine grained one around the minimum
  - Parametrization of Highland formula defined at fhicl file level
    - previously hardcoded, from uB paper
  - Add configurable parametrization for detector angular resolution, dependent on u_z
    - previously a constant value, defined in fhicl file
  - Optionally correct trajectory point positions for space charge effect based on available SpaceChargeService
  - Introduce a tolerance in segment length definition

Additional MCC9 Updates - Pandora Event Building

- Authors Andy Smith and Wouter van de PontSeele.
- Larpandora updates:
  - Updates limited to directory larpandora/LArPandoreEventBuiding.
  - CollectionMerging_module.cc removed.
  - CollectionSplitting_module.cc modified.
  - LArPandoraExternalEventBuilding_module.cc modified.
  - Associated tools and fcl files added and modified.
- Larpandora merge branch: greenlee_mcc9_event_building
- Event building updates were designed to be non-breaking for non-MicroBooNE experiments at the time they were originally developed (around Feb. 2019).