Checking of ILCSoft v01-17-08: update on PID

Hale Sert

DESY
Hamburg University

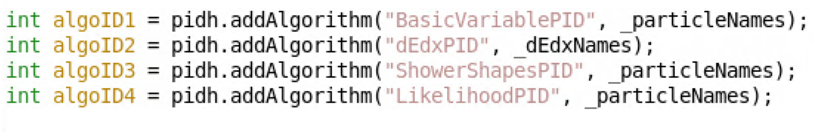
ILD Software and Analysis Meeting

Collaborative Research Centre 676
Particles, Strings, and the Early Universe
The Structure of Matter and Space-Time
DESY
UHH
Deutsche Forschungsgemeinschaft
DFG

Testing Low Momentum μ-π identification part of PIDTools

- PIDTools for Particle ID was released with ILCSoft v01-17-07.
  (Talk by M. Kurata on June 24, 2015)
- It was updated in several aspects in ILCSoft v01-17-08.
  (Talk by M. Kurata on July 10, 2015 - during High Level Reconstruction Week)
- Includes low momentum μ - π identification as well. ($p < 2\,\text{GeV}$)

- Four algorithms were introduced:



```
int algoID1 = pidh.addAlgorithm("BasicVariablePID", _particleNames);
int algoID2 = pidh.addAlgorithm("dEdxPID", _dEdxNames);
int algoID3 = pidh.addAlgorithm("ShowerShapesPID", _particleNames);
int algoID4 = pidh.addAlgorithm("LikelihoodPID", _particleNames);
```

- Basic variables : E/p, $E_{\mathrm{Ecal}}/(E_{\mathrm{Ecal}} + E_{\mathrm{Hcal}})$
- dEdx
- Shower shape variables

getShape()[0]: fitting χ 2

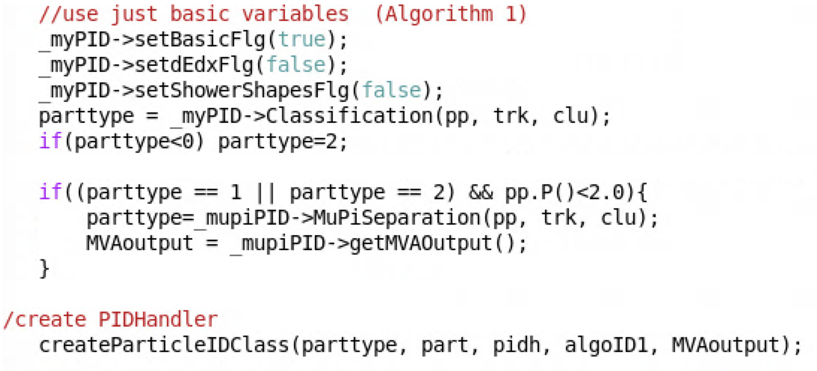
getShape()[1]: maximum energy deposit(GeV)

getShape()[2]: showerMax(mm)

- Likeliood considers combination of these

Testing Low Momentum μ-π identification part of PIDTools

- Low momentum μ -π separation was embedded into these algorithms



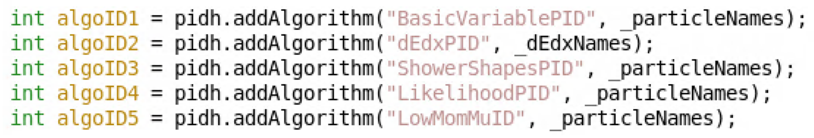
```
   //use just basic variables  (Algorithm 1)
   _myPID->setBasicFlg(true);
   _myPID->setdEdxFlg(false);
   _myPID->setShowerShapesFlg(false);
   parttype = _myPID->Classification(pp, trk, clu);
   if(parttype<0) parttype=2;

   if((parttype == 1 || parttype == 2) && pp.P()<2.0){
       parttype=_mupiPID->MuPiSeparation(pp, trk, clu);
       MVAoutput = _mupiPID->getMVAOutput();
   }

/create PIDHandler
   createParticleIDClass(parttype, part, pidh, algoID1, MVAoutput);
```

For example in the first algorithm (same for all algorithms);

- First the identification using basic variables was performed
- Then, μ-π identification was performed
- So, it was not possible to check the μ-π identification individually

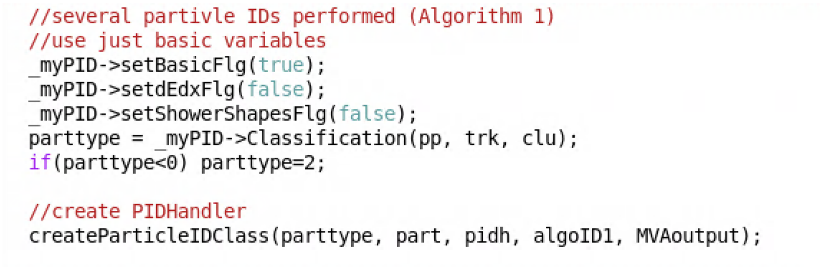
Testing Low Momentum μ-π identification part of PIDTools

- Define another algorithm for low momentum μ -π identification



```
int algoID1 = pidh.addAlgorithm("BasicVariablePID", _particleNames);
int algoID2 = pidh.addAlgorithm("dEdxPID", _dEdxNames);
int algoID3 = pidh.addAlgorithm("ShowerShapesPID", _particleNames);
int algoID4 = pidh.addAlgorithm("LikelihoodPID", _particleNames);
int algoID5 = pidh.addAlgorithm("LowMomMuID", _particleNames);
```

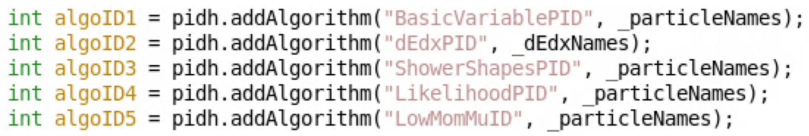
- Remove the low momentum μ -π ID part from the other algorithms



```
//several partivle IDs performed (Algorithm 1)
//use just basic variables
_myPID->setBasicFlg(true);
_myPID->setdEdxFlg(false);
_myPID->setShowerShapesFlg(false);
parttype = _myPID->Classification(pp, trk, clu);
if(parttype<0) parttype=2;

//create PIDHandler
createParticleIDClass(parttype, part, pidh, algoID1, MVAoutput);
```

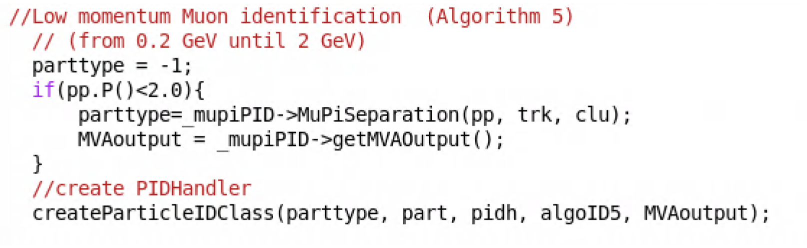
Testing Low Momentum μ-π identification part of PIDTools

- Define another algorithm for low momentum μ -π identification



```
int algoID1 = pidh.addAlgorithm("BasicVariablePID", _particleNames);
int algoID2 = pidh.addAlgorithm("dEdxPID", _dEdxNames);
int algoID3 = pidh.addAlgorithm("ShowerShapesPID", _particleNames);
int algoID4 = pidh.addAlgorithm("LikelihoodPID", _particleNames);
int algoID5 = pidh.addAlgorithm("LowMomMuID", _particleNames);
```

- Perform the low momentum μ -π ID part in the new algorithm



```
//Low momentum Muon identification  (Algorithm 5)
  // (from 0.2 GeV until 2 GeV)
  parttype = -1;
  if(pp.P()<2.0){
      parttype=_mupiPID->MuPiSeparation(pp, trk, clu);
      MVAoutput = _mupiPID->getMVAOutput();
  }
  //create PIDHandler
  createParticleIDClass(parttype, part, pidh, algoID5, MVAoutput);
```

Summary and Outlook

- Additional algorithm was introduced for low momentum μ-π separation
- The performance of this part of the PIDTools will be checked