PID Code in LCD/JAS: Update

John Cairns, Sky Rolnick, Bob Wilson
Colorado State University

ACLPG – SLAC
January 9th, 2004
Project Goals

- Provide a simple, flexible, fast tool to explore Particle ID issues
- Update current Particle ID fast simulation/reconstruction in JAS2/LCD
  - Usability, maintainability, veracity.
- Where to store track-level PID info dE/dx info
  - Currently saved in “separate” Hashmap
  - Extend track definition -> PIDTrack? (Include other track methods like path length)
- Prototype `ReconstructedParticle` class use
  - Natural place to add Particle ID information (`PidInfo` class?)
- What should be in the `PidInfo`?
  - e.g. combine systems “best ID”, all particle likelihoods, subsystem likelihoods…?
- How should `LCDEvent` be used?
  - Currently, a “catch-all” object to be passed along the event loop
  - Useful for prototyping, but skirts any overall design
- New OO design for entire package
  - Basic design done. Implementation on hold.
Example PIDSimpleDriver

- Standard MCFast generates smeared track list – PID package picks this up and adds PID specific information to our version of ReconstructedParticle.
**PIDInfo**

**goodness**

*e.g. lnLikelihood difference to next best ID*

```java
import java.lang.
import Object;

class PidInfo {
    public PidInfo();
    public void setInfo();

    // Member variables
    public ParticleType bestID;
    public double bestIDgoodness;
    public TrackTypeLnLikelihoodMap bestLnLikelihood;
    public double contributionSystems;
    public HashSet isaKaon;
    public double isaKion;
    public double isaElectron;
    public double isaPion;
    public double isaProton;
    public double notaPion;

    // Constructor
    PidInfo() {
        // Initializations
    }

    void setInfo() {
        // Set information
    }
}
```

**isaXXX**

“Expert” determined default criteria with well understood efficiency/purity tables
Restructuring dEdxFastRecon

- Original dEdxFastRecon was an inflexible amalgam of tasks for simulation/reconstruction/analysis in FORTRAN-style
- Didn’t allow natural reuse of code in different parts of the package:
  - e.g. the dEdx models are used in both fast simulation and reconstruction (compare “measured” dEdx with expected to produce a likelihood)
- Has been restructured following “OO” methodology
DEdxFastRecon

Overview
Performs fast reconstruction of particles energy loss in a gas chamber according to various DEdx models. Allows for Particle ID based on momentum and energy loss of particles.

Methods
- setAcceptance() - sets acceptance cut values.
- setGeometry() - sets geometry of detector.
- setDebug(boolean) - flag to create histograms.

Usage
```java
DEdxFastRecon = new DEdxFastRecon();
DEdxFastRecon.setRes(0.05) // sets resolution to 5%
DEdxFastRecon.setPtCut(0.5) // sets PtCut500 MeV/c
DEdxFastRecon.setCosthCut(30) // sets CosthCut to 30 degrees
DEdxFastRecon.setDebug(true) // will now create histograms
```

Changes
- separate modules for DEdxModels, DEdxResolution, and PIDTracking.

January 9th, 2004
DEdx Model Class

Overview
Allows the calculation of expected energy loss for several different models including Sternheimer model, Dzierba model, and the Yamamoto model. Currently all models are based off the Sternheimer parameters.

Methods
- getName() - returns the name of the model used.
- getExpected() - returns expected energy loss.

Usage
```java
dedxModel = new DEdxModelSternheimer(currDetector);
dedxModel.getExpected(pMass, ptot, Tcut);
dedxModel.getName();
```

Changes
- Can now be constructed with single parameter (Detector) instead of long list of parameters.

January 9th., 2004

R.J Wilson, Colorado State University
DEdxResolution

Overview
Calculates the energy loss resolution for the gas chamber based on the Walenta model. Currently depends on geometry of the detector, the number of samples, but should also depend on the pressure of the gas.

Methods
- getResModel() - returns the name of the model used.
- getResolution() - returns calculated resolution.
- getSamplesMax() - returns the max number of samples.
- getSamplingThickness() - returns samplingThickness.

Usage
```java
dedxRes = new DEdxResolution(currDetector);
dedxRes.calculateDEdxRes(trackPathLength);
```

Changes
- Added several constructors so that resolution can be set fixed or calculated depending on the parameters passed to DEdxResolution.
Using HashMaps for adding info...

```java
// Get the track list map.
HashMap m_PIDmap = new HashMap(event.getTrackList().getNTracks());

// Instantiate the map to carry the track/energy loss map
TrackdEdxMap map = new TrackdEdxMap();

// Store totalEnergyLoss and corrected resolution in the map.
map.setdEdx(totalEnergyLoss);
map.setdEdxRes(totalDEdxResCorr);

// Insert into the track map.
m_PIDmap.put(t,map);

// Get the PID map for the event
HashMap EventPIDmaps = (HashMap) event.get("PIDmapsTable");

// Get the PID map.
HashMap m_PIDTracksmap = new HashMap(event.getTrackList().getNTracks());

// Get the Reconstructed Particles and the PIDmapsTable.
PartVec = (ReconstructedParticleVector) event.get("ReconstructedParticles");
pidMaps = (HashMap) event.get("PIDmapsTable");

// Get the list of PID systems available
HashMap hSysmap = (HashMap) pidMaps.get(m_systemName);

HashMap SysdEdxmap = (HashMap) hSysmap.get(m_systemType);
TrackdEdxMap map = (TrackdEdxMap) SysdEdxmap.get(t);

// Get the stored dEdx and dEdxRes from TrackdEdxMap
double DEdxmeas = map.getdEdx();
double DEdxResCorr = map.getdEdxRes();

// Create LnLikelihoodMap for particle ID.
TrackPTypeLnLikelihoodMap lnLmap = new TrackPTypeLnLikelihoodMap();
lnLmap.setLnLikelihood(idType[i], Math.max(lnL[i], lnLcap));

// Insert in the track map.
m_PIDTracksmap.put(t,lnLmap);

// Save the dEdx map into the event
event.put("PIDmapsTable", EventPIDmaps);

// Save the dEdx map in the PID map
hSysmap.put("lnL", m_PIDTracksmap);
```
To Do List

- Implement cut values in DEdxFastRecon to give users more flexibility
- Include other parameters for other DEdxModel’s
  - currently only Sternheimer parameters used for all models.
- Create new class capable of calculating gas parameters for various gas mixtures, either as a lookup table or from a model.
- Enable PIDTrack so that dEdx information can be stored using a flag.
- Extend ReconTrack with PIDTrack so that TrackPathLength and dEdx information can be stored in track.
- Modify the concept of storing dEdx information in TrackdEdxMap.
  - use PID objects to store information that can be passed from object to object without the use of Hashmaps.
- Produce a module capable of looping through many detector configurations to test various resolution and threshold values and the effects on efficiency and purity.
- Possibly write a counter class that can be used instead of Histograms since this would reduce the complexity and allow several systems to be analyzed at once.
July 2003 ALCPG Meeting: Summary Slide

• Cross subsystem Particle ID implementation at an impasse
  – E.g. ReconstructedParticle and PID code existing “on the fringe” for long time
• To take best advantage of outside contributors …
  – Guidance/consultation on the s/w design/architecture
  – Clear mechanism for review and subsequent inclusion in LCD code releases (CVS a good step in that direction)
January 2004 ALCPG Meeting: Summary Slide

- Cross subsystem Particle ID implementation at an impasse
  - E.g. ReconstructedParticle and PID code existing “on the fringe” for a long time
- To take best advantage of outside contributors …
  - Guidance/consultation on the software design/architecture
  - Clear mechanism for review and subsequent inclusion in LCD code releases (CVS a good step in that direction)
Summary

• Did not submit a continuation LCRD PID proposal because…
  – Not enough time to accomplish goals of the previous proposal – funds arrived mid-summer
  – Funding request limitation inadequate to do an adequate job… especially in light of inadequate support infrastructure (at SLAC)

• Complete current effort by summer, but will not have the resources to improve further or convert to JAS3 etc.

• Wait to see if funding situation improves.

• Use existing code for PID studies.