Benchmark Studies with Perfect PFA

Standard Perfect PFA for SiD Detectors with :

- DigiSim – realistic hits with timing, threshold requirements
- Tracks – MC 4-vector for tracked (3 hit min) charged particles
- Perfect Calorimeter clusters for photons, neutral hadrons (perfect hit clusters w/ 3 hit min, real calorimeter energy)
- *Reconstructed Particle list - LCIO output or in analysis code*

Comparison to Fast MC Benchmarking :

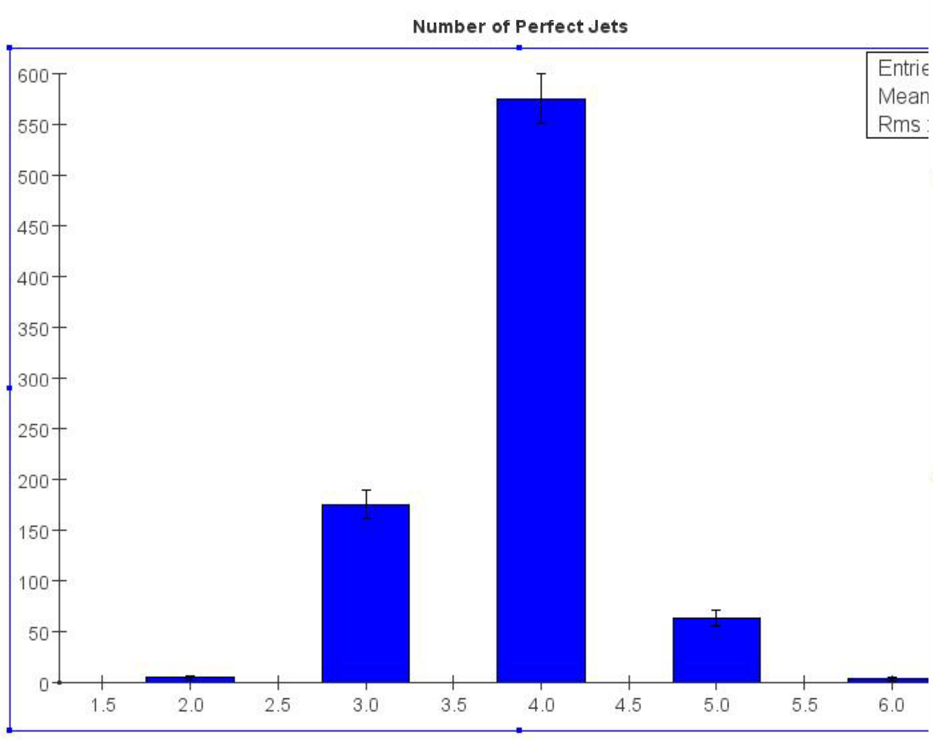
- Based on perfect patt. rec. FS particles, not generator particles
- Reconstructible tracks, perfect 4 vectors now, but realistic track 4-vectors coming (Rob K)
- Photon 4-vector formed from simulated calorimeter hits, not smeared energy (non-linear resolution effects)
- Neutral hadron 4-vector formed from simulated cal hits, including both ECAL and HCAL

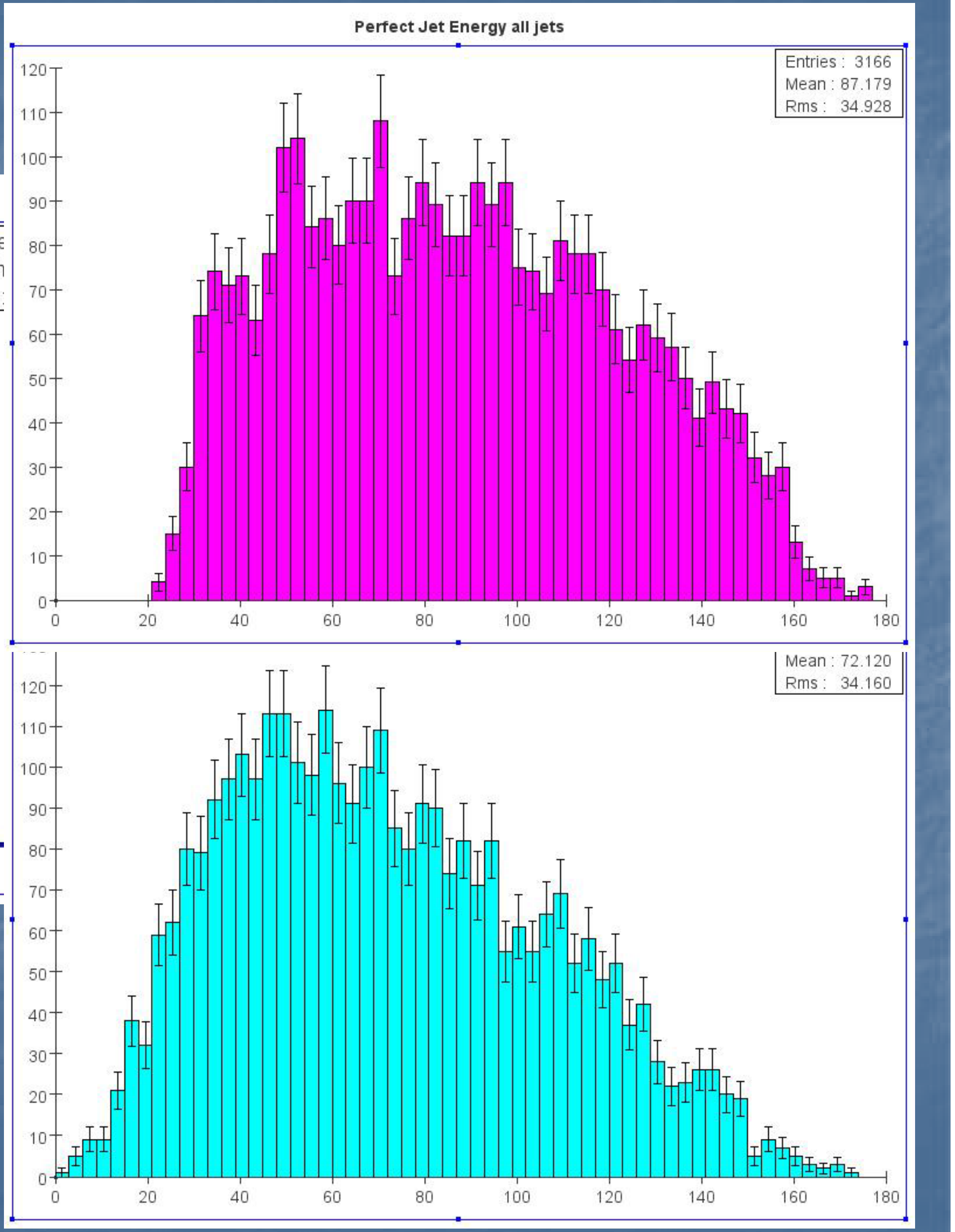
JAVA Code exists (org.lcsim) and is being used in PFA development

Can be written out in LCIO format

Perfect PFA is the PFA Target – more realistic measure of SiD LOI Benchmark performance

e^+e^- -> ZH @ 500 GeV (4 jets) in SiD01 Detector Model

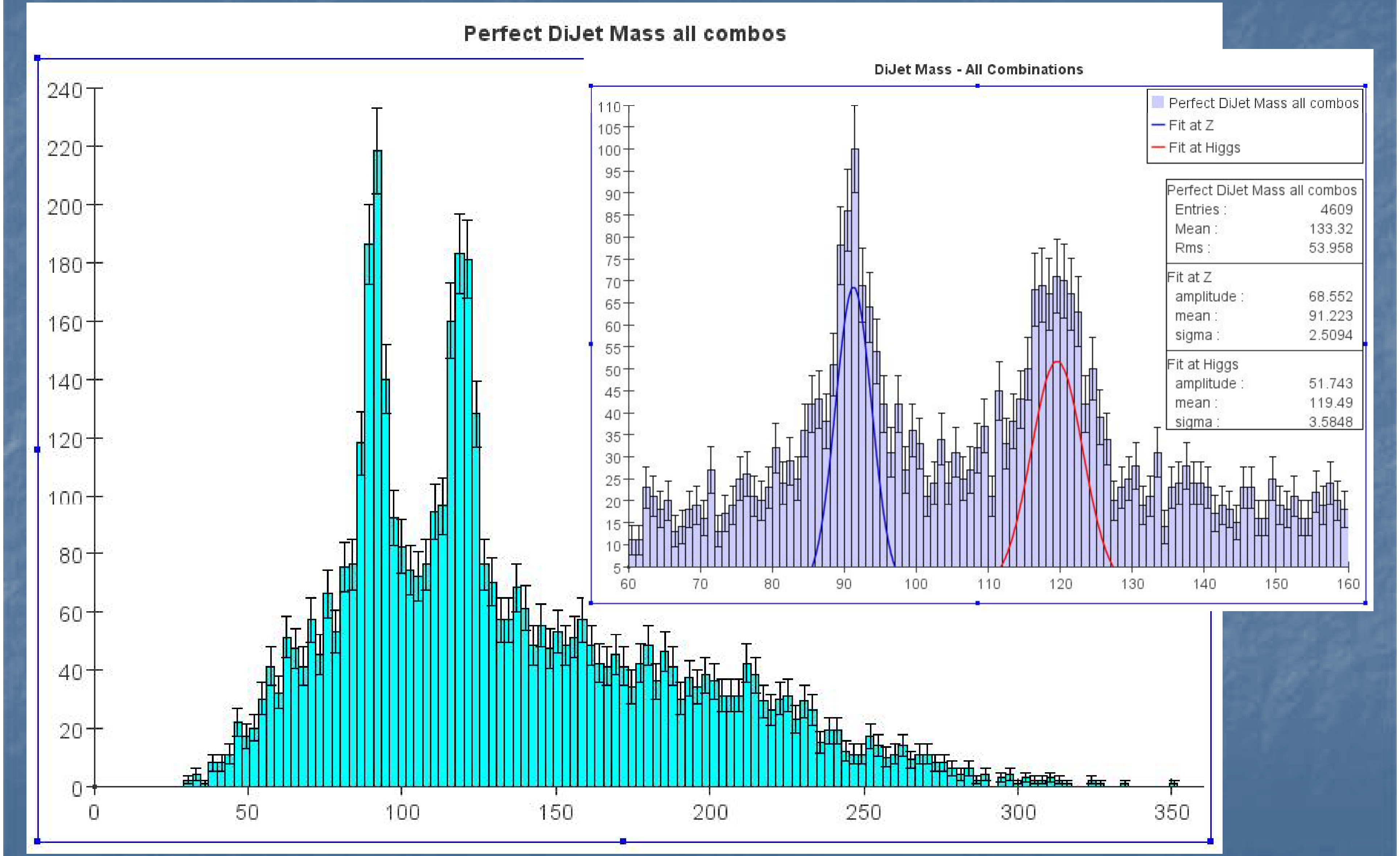




kT Algorithm with ycut = 0.008

Plot all dijet mass combinations, fit with comb. bkgrd. function

- Perfect PFA dijet mass resolution



Org.lcsim Example

```
package org.lcsim.contrib.Cassell.recon.Cheat;
import java.io.File;
import org.lcsim.event.EventHeader;
import org.lcsim.event.ReconstructedParticle;
import org.lcsim.event.MCParticle;
import org.lcsim.util.Driver;
import org.lcsim.util.loop.LCIODriver;
import org.lcsim.contrib.Cassell.recon.Cheat.CheatReconDriver;
import org.lcsim.contrib.Cassell.recon.Cheat.PPRDriver;
import hep.physics.vec.Hep3Vector;
import org.lcsim.util.aida.AIDA;
import java.util.*;

/**
 * A example of writing LCIO output.
 *
 * @see org.lcsim.util.loop.LCIODriver
 *
 * @author Tony Johnson
 * @version $Id: CheatReconOutputExample.java,v 1.1 2007/11/08 23:34:46 cassell Exp $
 */
public class CheatReconOutputExample extends Driver
{
  private AIDA aida = AIDA.defaultInstance();
  String CheatReconRname = "ReconPerfectReconParticles";
  String PPRPflowRname = "PPRReconParticles";
  String CheatReconFSname = "ReconFSParticles";
  public CheatReconOutputExample()
  {
//
// Do the cheating reconstruction (includes Digisim)
//
    CheatReconDriver crd = new CheatReconDriver();
    crd.setCheatReconstructedParticleOutputName(CheatReconRname);
    crd.setCheatFSParticleOutputName(CheatReconFSname);
    add(crd);
//
// Make the perfect pattern recognition pflow reconstructed particles from the
// Cheat Recon particles
//
    add(new PPRDriver(CheatReconRname,PPRPflowRname));
//
// Write the events to disk
//
    File output = new File("E:","CheatReconOutputsid01.slcio");
    add(new LCIODriver(output));
  }
```

```
  protected void process(EventHeader event)
  {
    super.process(event);
//
// Get the final state particles
//
    List<MCParticle> fs = event.get(MCParticle.class,CheatReconFSname);
//
// Get the perfect pattern recognition reconstructed particles
//
    List<ReconstructedParticle> ppr = event.get(ReconstructedParticle.class,PPRPflowRname);
//
// Plot the energy sum and invariant mass for each event
//
    double evtE = 0.;
    double evtPx = 0.;
    double evtPy = 0.;
    double evtPz = 0.;
    for(MCParticle p:fs)
    {
      int pdg = Math.abs(p.getPDGID());
      if( (pdg == 12)||(pdg == 14)||(pdg == 16) )continue;
      evtE += p.getEnergy();
      Hep3Vector P = p.getMomentum();
      evtPx += P.x();
      evtPy += P.y();
      evtPz += P.z();
    }
    aida.cloud1D("Event non-neutrino final state energy").fill(evtE);
    double evtM = Math.sqrt(evtE*evtE - evtPx*evtPx - evtPy*evtPy - evtPz*evtPz);
    aida.cloud1D("Event non-neutrino final state mass").fill(evtM);
    evtE = 0.;
    evtPx = 0.;
    evtPy = 0.;
    evtPz = 0.;
    for(ReconstructedParticle p:ppr)
    {
      evtE += p.getEnergy();
      Hep3Vector P = p.getMomentum();
      evtPx += P.x();
      evtPy += P.y();
      evtPz += P.z();
    }
    aida.cloud1D("Event reconstructed energy").fill(evtE);
    evtM = Math.sqrt(evtE*evtE - evtPx*evtPx - evtPy*evtPy - evtPz*evtPz);
    aida.cloud1D("Event reconstructed mass").fill(evtM);
  }

  protected void endOfData()
  {
    super.endOfData();
  }
}
```

Perfect PFA Reconstructed Particle List

Available now (more realistic tracking parameters soon)

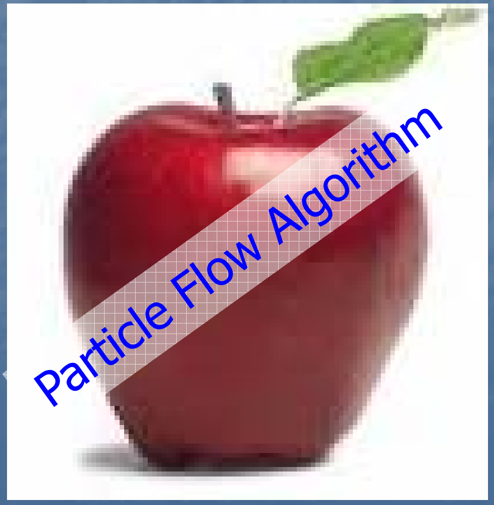
Replaced with PFA Reconstructed Particle List also ~soon

Can start now with PPFA RP List to setup analyses for benchmarks
Substitute PFA RP List when ready for like comparison

Format :

PFA RP List = Perfect PFA RP List = MCFast RP List

However, content :



≈

≠