# Reconstruction Cheater

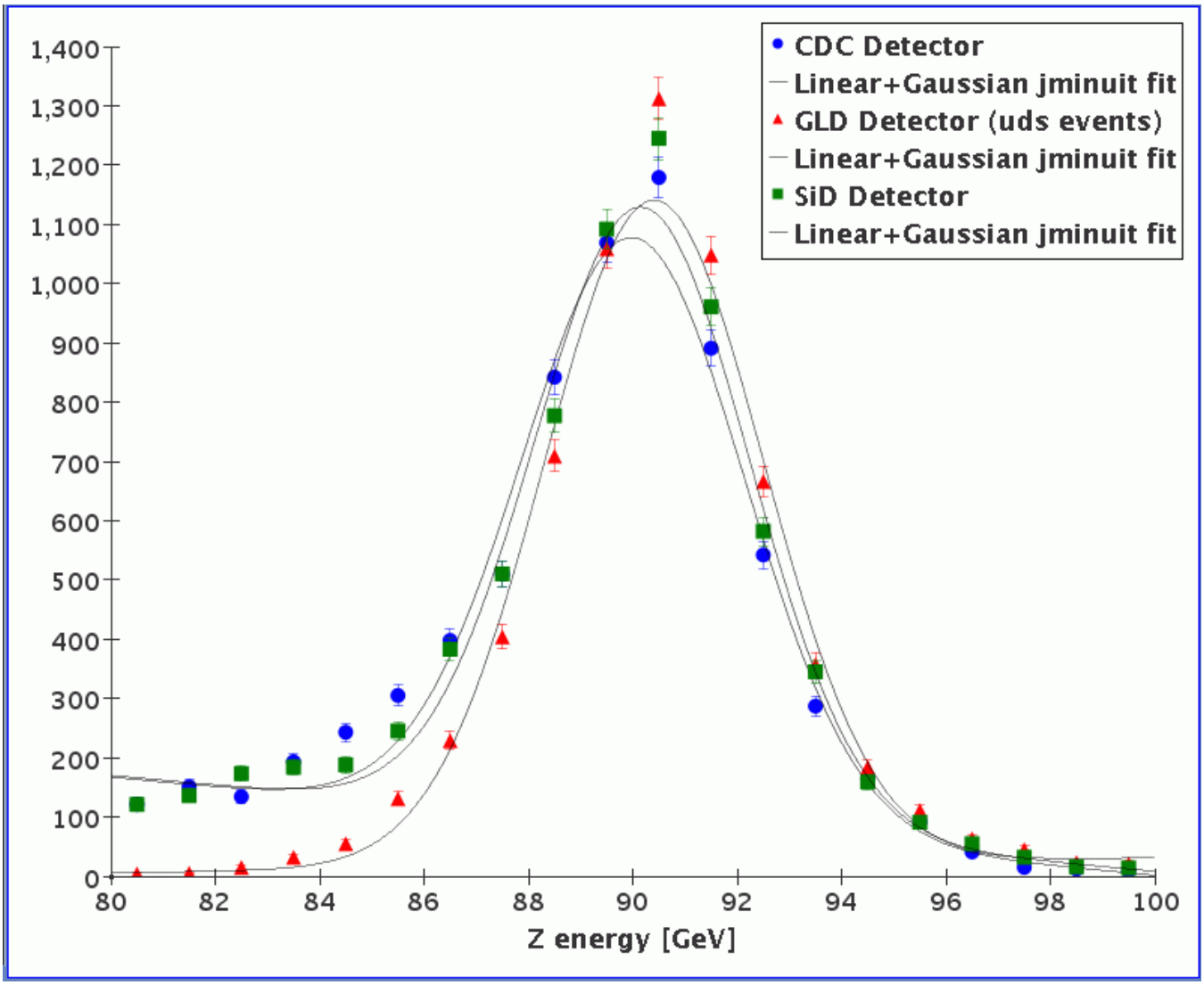
Simulated events can be fully reconstructed by a **ReconCheater** that uses Monte Carlo truth to reconstruct individual particles. Charged tracks found by the **TrackingCheater** and calorimeter clusters found by the **ClusterCheater** are used to provide the basis for full event reconstruction. Various options and parameter settings in a **Cheating** properties file or Cheating Table for each detector are used to mix reconstructed tracks and clusters with Monte Carlo particles. Options for obtaining various measures of the neutral energy deposition, for controlling decays and nuclear interactions, and simulating loses of neutral hadrons are included.

```
To add the ReconCheater to a process, use

    add(new org.lcsim.recon.cheater.ReconCheater());

The TrackingCheater and ClusterCheater drivers will be added by the ReconCheater.
An AIDA histogram package is included for optional use, see below.

To access reconstructed particles, use

    event.get(ReconstructedParticle.class);
```

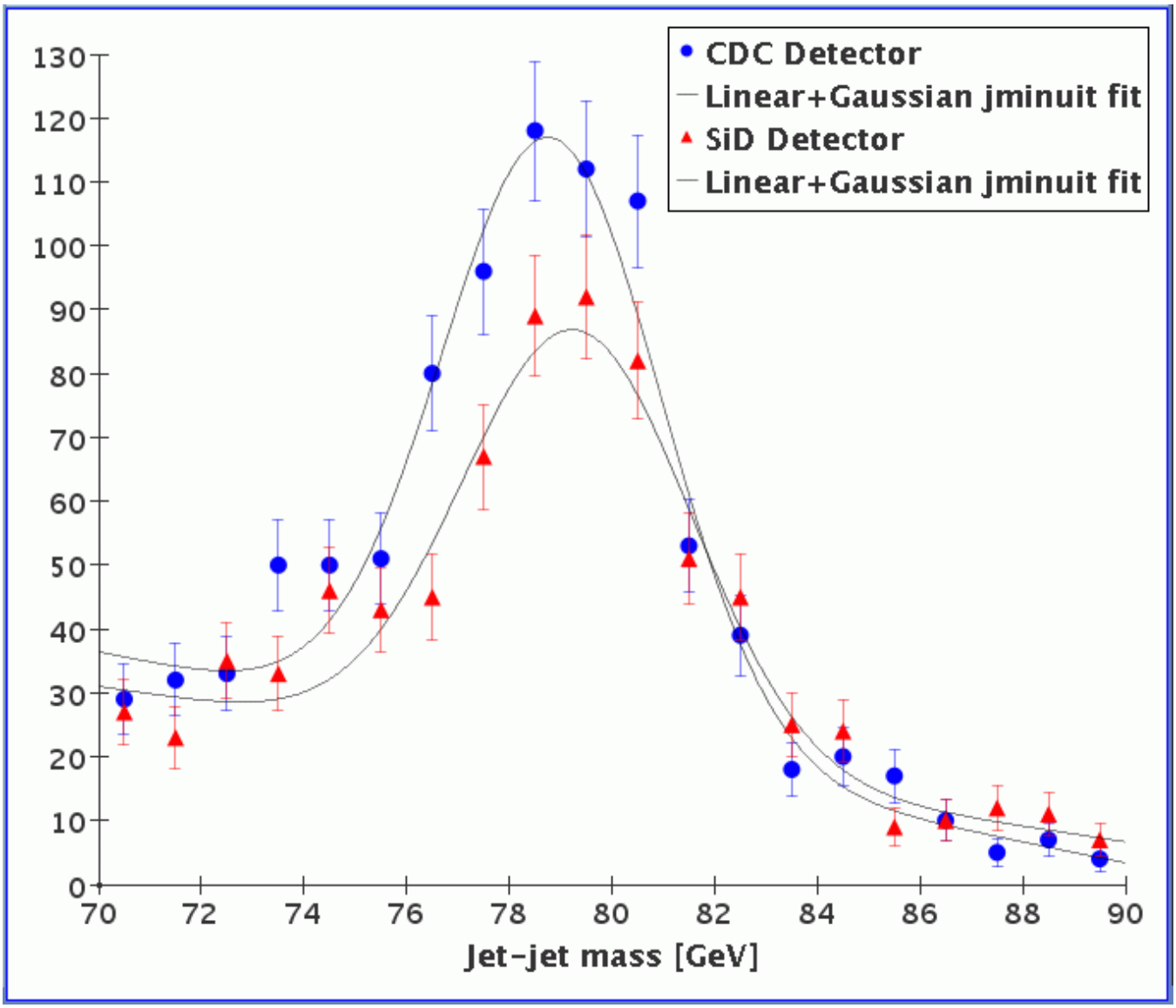
Fully simulated GEANT4 events are reconstructed by the **ReconCheater** in the following way:

- Only those tracks and clusters that were reconstructed, e.g. by the corresponding cheaters, are used, so particles do get lost.
- For charged particle decays, the track from the primary particle is used if it was found, otherwise the secondary track is used, allowing neutrinos to carry away energy.
- For converted photons, the found tracks are used.
- At this stage, perfect energy flow is used where a cluster's parentage is checked to see if there is a charged track which should be associated.
- Nuclear interactions are turned off by default since the reconstruction is not optimized for them yet: for example should low energy protons be ignored, can low energy neutrons be cut out, can the leading particle be used to get an estimate of the corresponding neutral energy in the calorimeter.
  The original particle (or reconstructed primary track if it's charged) is used and all secondary tracks and clusters are removed. Nuclear interactions can be turned on for comparisons.
- Radiation is not allowed at this time.
- Perfect energy flow reconstruction can be specified. Otherwise, ...
- Neutral hadrons are not considered reconstructable if the distance from a nearby cluster with 2X or 4X the neutral cluster energy is less than specified.

The total Z pole energy reconstructed for the CDC, GLD and SiD detectors is shown below. Only uds decays of the Z where simulated for the GLD detector. Here a parameterized HCal response of 60%/sqrt(E) was used to obtain resolutions of 2.1 GeV for all 3 detectors.




High energy W mass reconstruction for the CDC and SiD detectors is shown below. A parameterized HCal response of 60%/sqrt(E) and perfect energy flow were used to obtain resolutions of 2.2 GeV for both detectors.




The **ReconCheater** code is contained in the following classes:

| Packages: | Classes: |
|---|---|
| org.lcsim.recon.cheater | CheatingTable |
| | CheatParticleID |
| | CheatReconstructedParticle |
| | ReconCheater |

```
To use the ReconCheater histogram package, use

    cheater = new org.lcsim.recon.cheater.ReconCheater();

    cheater.setHist(true);
```

Example of analysis code:

```
public void process(EventHeader event)
{
    // Get reconstructed particle 3Vector's.
    List<Hep3Vector> particles = new ArrayList();
    List<List<ReconstructedParticle>> rpLists = event.get(ReconstructedParticle.class);
    for (List<ReconstructedParticle> collection : rpLists) {
        String name = event.getMetaData(collection).getName();
        if (!name.equals("ReconCheater")) continue;
        for (ReconstructedParticle rp : collection) {
            particles.add(rp.getMomentum());
        }
    }
    // Set up Jet finder.
    double ycut = 0.0005, dycut = 0.0005;
    finder.setYCut(ycut);
    finder.setEvent(particles);
    NJets = finder.njets();

    while (NMJets>MaxNumberJetsExpected) {
        ycut += dycut;
        finder.setYCut(ycut);
        NMJets = finder.njets();
    }
    aida.cloud1D("# jets").fill(NJets);
}
```

Example of JAS3 driver:

```
/* EventRecoDriver.java
 */

import java.util.List;

import org.lcsim.event.EventHeader;
import org.lcsim.event.MCParticle;
import org.lcsim.event.ReconstructedParticle;
import org.lcsim.recon.cheater.ReconCheater;
import org.lcsim.util.Driver;
import org.lcsim.util.aida.AIDA;

public class EventRecoDriver extends Driver
{
    private AIDA aida = AIDA.defaultInstance();

    /** Creates a new instance of EventRecoDriver */
    public EventRecoDriver()
    {
        add(new ReconCheater());
    }

    public void process(EventHeader event)
    {
        super.process(event); // this takes care that the child Drivers are loaded and processed.

        List<List<ReconstructedParticle>> rpLists = event.get(ReconstructedParticle.class);
        for (List<ReconstructedParticle> list : rpLists) {
            if (!event.getMetaData(list).getName().equals("ReconCheater")) continue;
            aida.cloud1D("RP list size").fill(list.size());
        }
    }
}
```