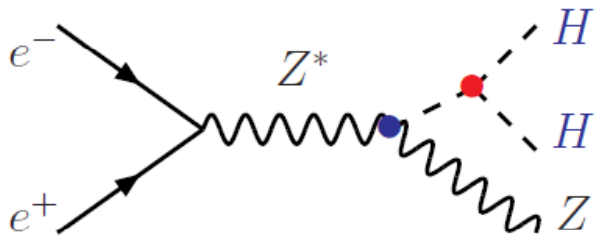
Higgs Self-Coupling Analysis – How To

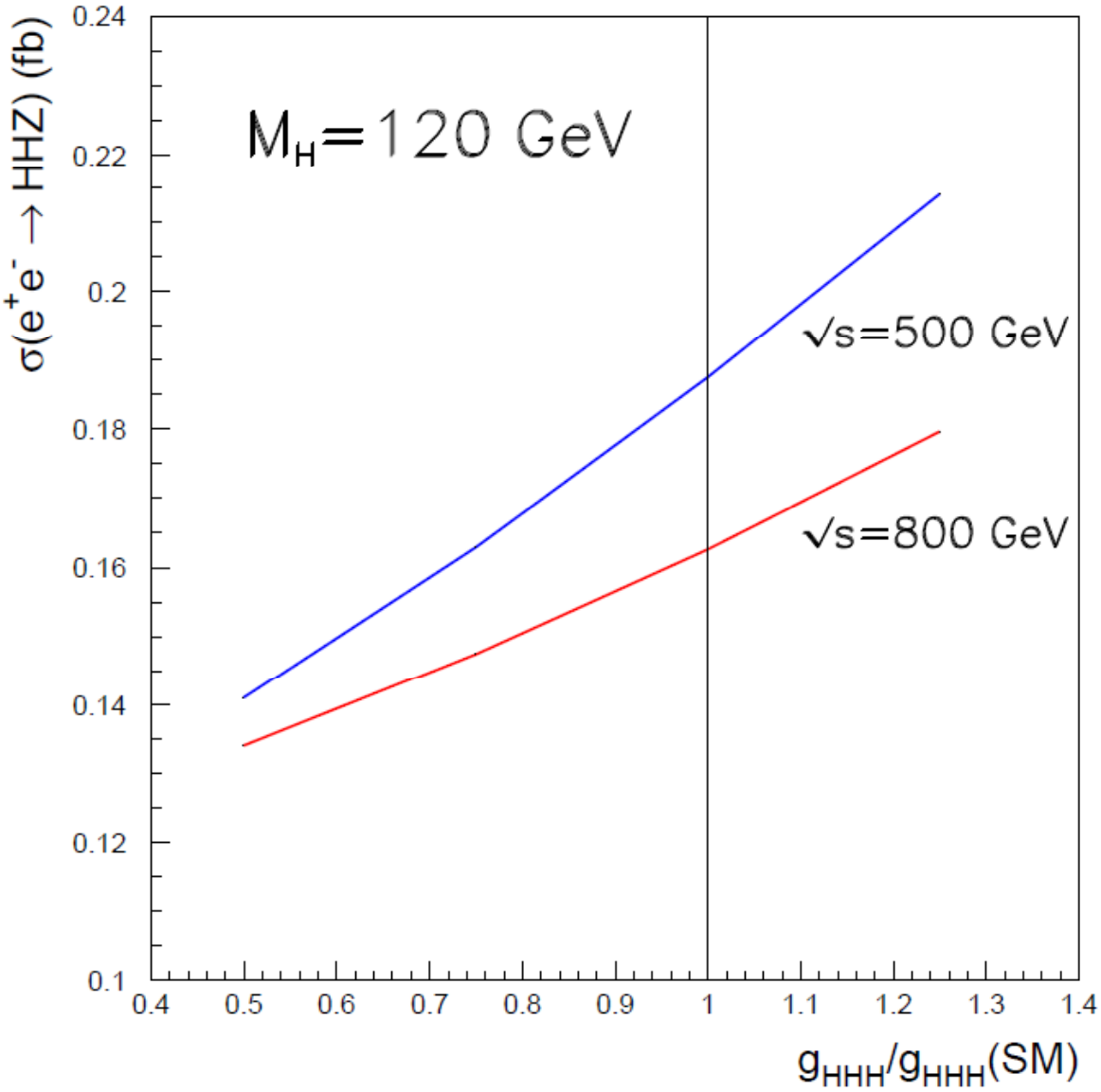
Tim Barklow

SLAC

October 25, 2007

$e^+e^- \to ZHH \to q\bar{q}b\bar{b}b\bar{b}$





Plan for Analysis of ZHH→qqbbbb at Ecm = 500 GeV

- Use org.lcsim Fast MC simulation of baseline SiD. This MC includes a reasonable algorithm for smearing charged track angles, curvature and impact parameters. Calorimeter simulation consists of simple single neutral particle smearing with EM resolution for photons and HAD res for n,$K0_{L.}$
- Add LCIO Reconstructed Particles to org.lcsim FASTMC
- Scale single particle calorimeter resolutions to get a particular ΔE_{jet} .
- Generate extra ZZ, ZH, tt, ZZZ, ZZH, ZHH datasets.
- Interface hep.lcd ZVTOP to org.lcsim
- Interface FORTRAN/SIMDET/PAW analysis system to LCIO
- Do analysis using FORTRAN/SIMDET/PAW system

Add LCIO Reconstructed Particles to org.lcsim FASTMC

- FASTMC reconstructed particle code was written by 2005 SLAC summer student Daniel Furse
- After Daniel left I was forced to learn Java in order to maintain the FASTMC code and add new features . This was my first OO programming experience. I found the process to be relatively easy and painless. It was no problem to go back and forth between writing MC code in Java and analysis code in Fortran.

Scale single particle calorimeter resolutions to get a particular ΔE_{jet} .

JETParameterization: true
JETResolution: 0.306
JETHadDegradeFraction: 1.0
JETEMEnergyFraction: 0.278
JETHadEnergyFraction: 0.1

← ClusterParameters.properties parameters controlling α in

$$\frac{\Delta E_{jet}}{E_{jet}} = \frac{\alpha}{\sqrt{E_{jet}}}$$

IDEfficiency.properties parameter controlling fraction of time charged particles are kept by the PFA:

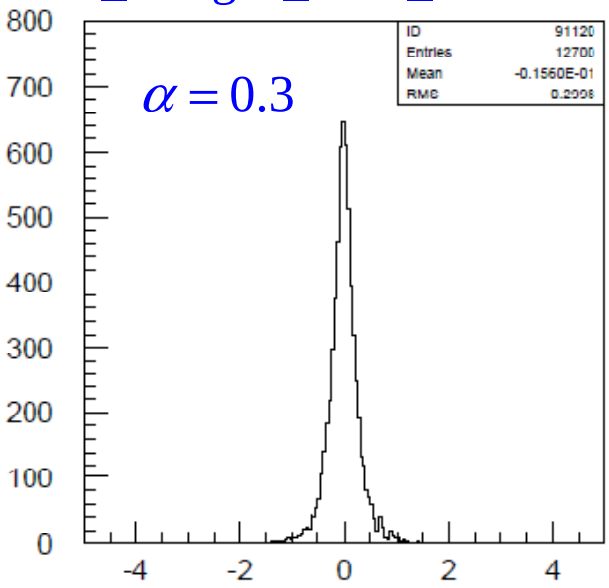
wt_charged_track_calorimeter_energy: 1.0

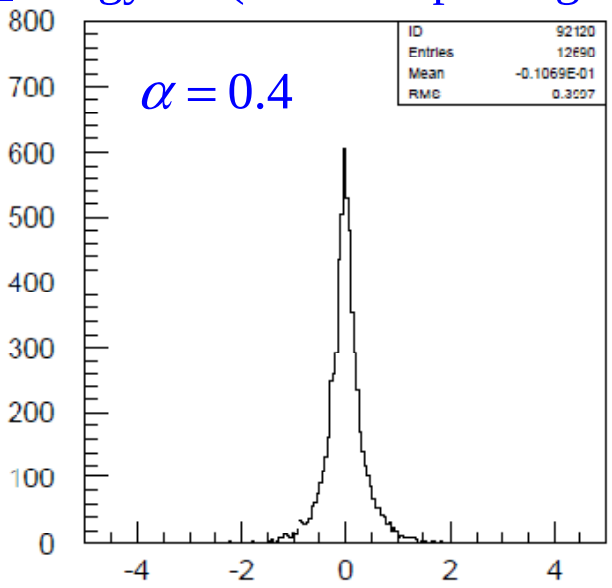
$e^+e^- \to u\bar{u}$

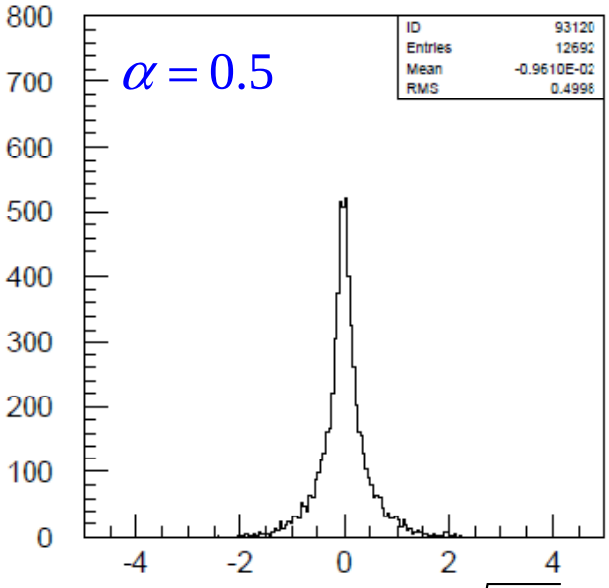
$\sqrt{s} = 500$ GeV

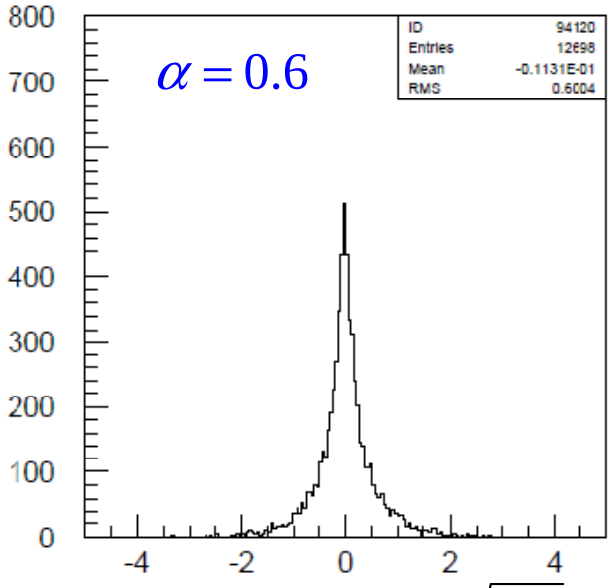
wt_charged_track_calorimeter_energy = 1 (never drop charged)

$$\frac{\Delta E_{jet}}{E_{jet}} = \frac{\alpha}{\sqrt{E_{jet}}}$$







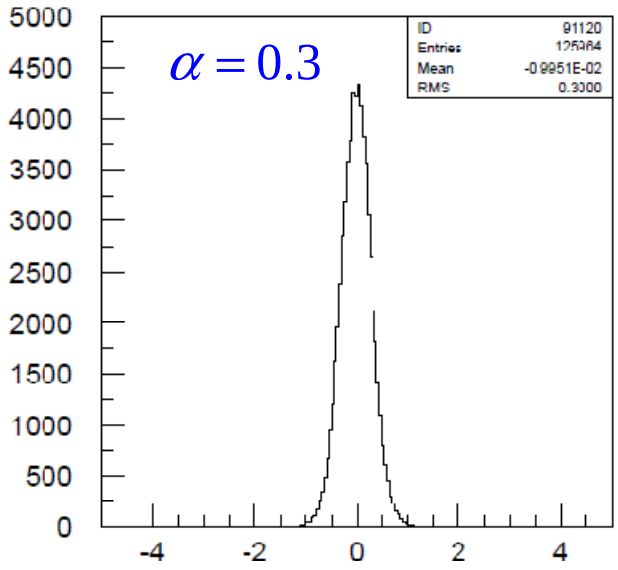


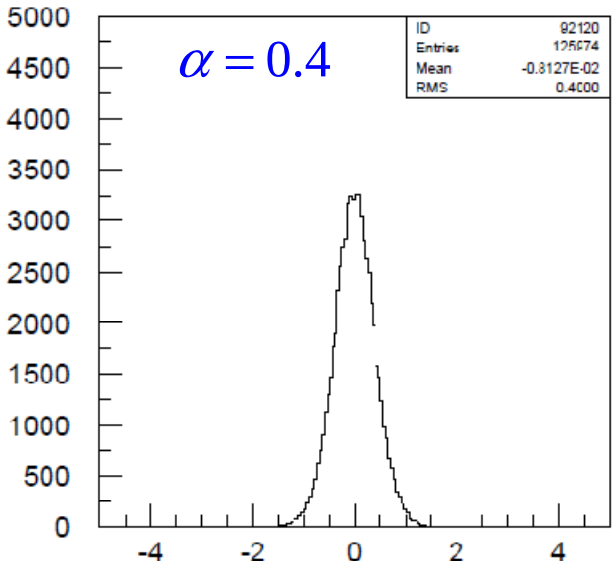
$e^+e^- \to u\bar{u}$

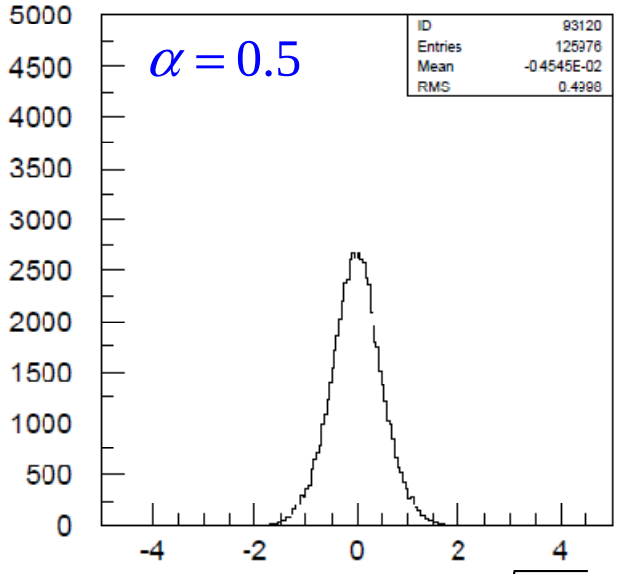
$\sqrt{s} = 500$ GeV

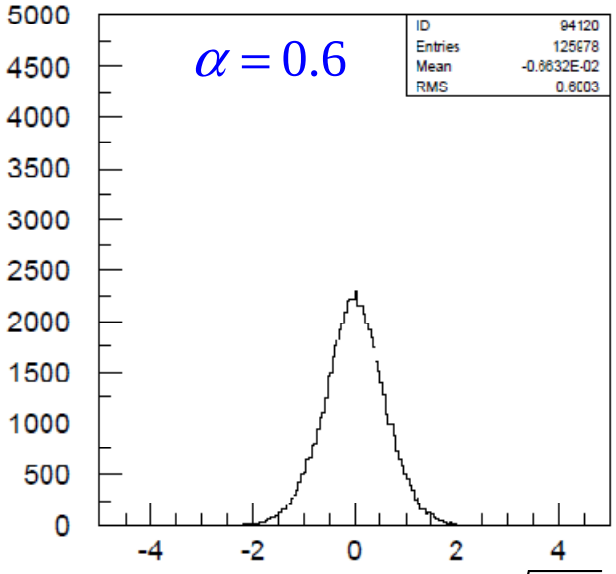
wt_charged_track_calorimeter_energy = 0 (always drop charged)

$$\frac{\Delta E_{jet}}{E_{jet}} = \frac{\alpha}{\sqrt{E_{jet}}}$$





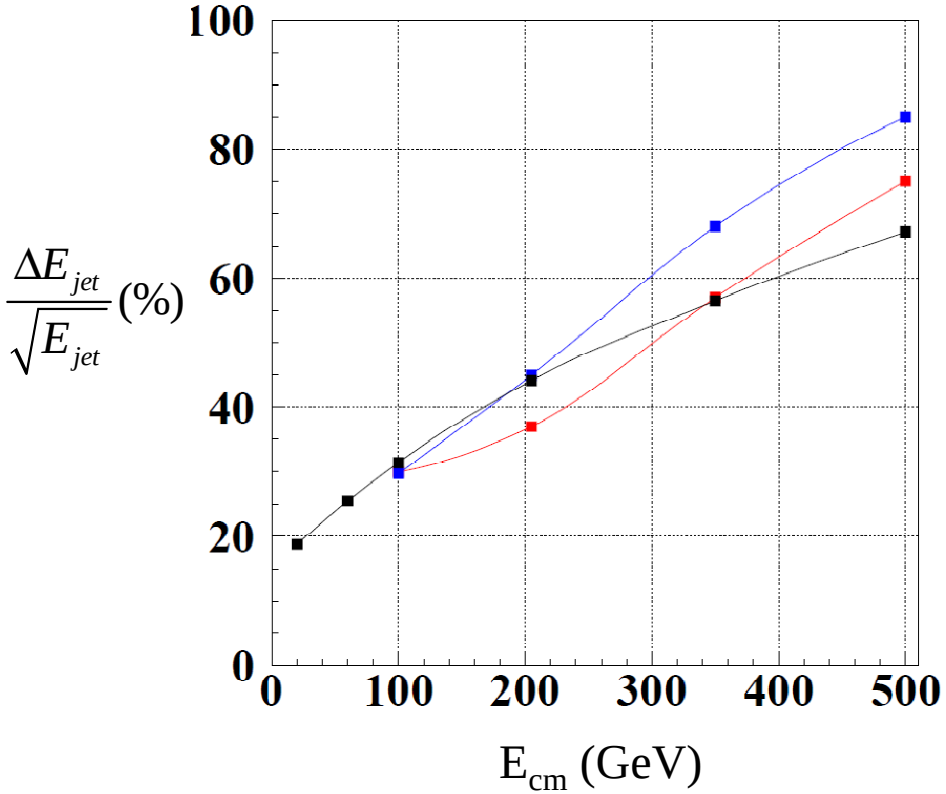


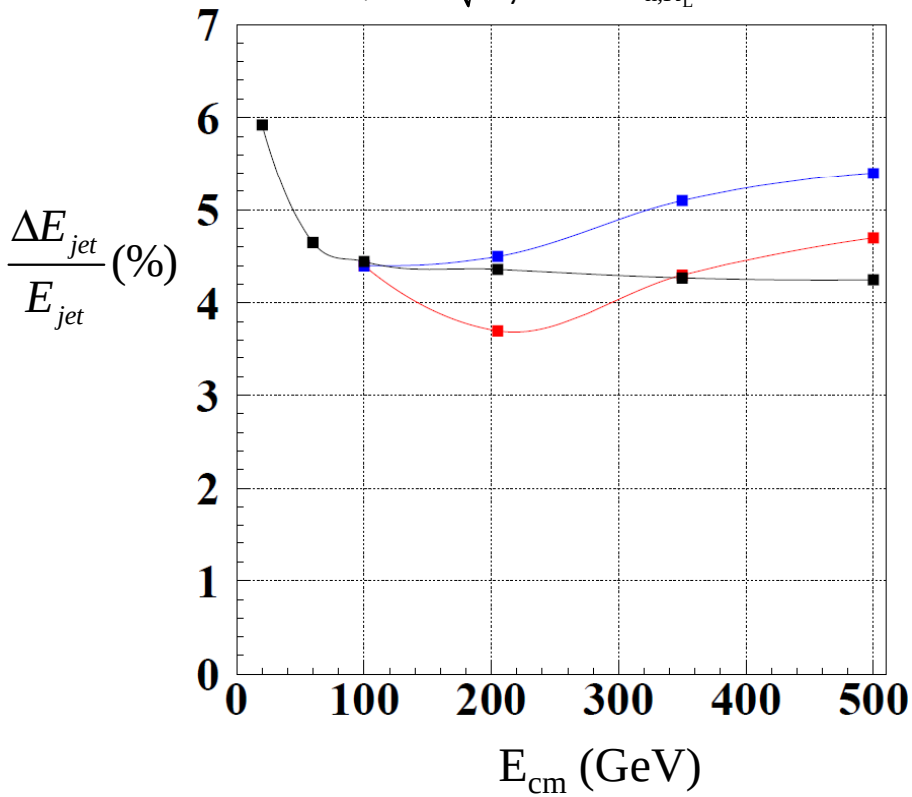


$$\frac{\Delta E_{jet}}{E_{jet}} \approx \text{ const.}$$

GLD PFA LDC PFA FASTMC with JETParameterization = false

$\frac{\Delta E_{\gamma}}{E_{\gamma}} = \frac{0.18}{\sqrt{E_{\gamma}}}$ $\frac{\Delta E_{n,K_L^0}}{E_{n,K_L^0}} = 0.28$





Generate extra ZZ, ZH, tt, ZZZ, ZZH, ZHH datasets

- The 1 ab^{-1} SM data set at Ecm=500 GeV with its general 2-fermion, 4-fermion and 6-fermion final states contains all the backgrounds and signal for a study of ZHH. It was used for example to identify a new background: ZZH.
- However, once the dominant backgrounds were identified, it was convenient to generate separate ZZ, ZH, tt, ZZZ, ZZH, and ZHH datasets for detailed studies. This was done with the same WHIZARD MC that was used to generate the 1 ab^{-1} SM data .

Interface hep.lcd ZVTOP to org.lcsim

- To identify b jets without c jet contamination, and to possibly distinguish b jets anti-b jets, a good vertex reconstruction algorithm is required.
- In late 2005 org.lcsim did not contain a working version of ZVTOP. It was relatively straightforward to interface the old org.lcsim version of ZVTOP.

Code snippets from hep.lcd ZVTOP Interface to org.lcsim:

```
import hep.lcd.vertexing.zvtop.ZvTrackList;
import hep.lcd.vertexing.zvtop.ZvTrack;
import hep.lcd.vertexing.zvtop.ZvTopVertexer;
import hep.lcd.vertexing.zvtop.ZvVertexList;
import hep.lcd.vertexing.zvtop.ZvVertex;
import hep.lcd.vertexing.zvtop.ZvBField;
```

```
private static final double av = 0.1;
private static final double ov = 0.1;
m_parm[2]=-m_parm[2];   // hep.lcd as opposite sign convention for omega
if(convert_mm_to_cm) {
    m_parm[0]*=av;
    m_parm[2]/=ov;
    m_parm[3]*=av;
    m_err[0][0]*=pow(av,2);
    m_err[0][1]*=av;
    m_err[0][2]=av*m_err[0][2]/ov;
```

Interface FORTRAN/SIMDET/PAW analysis system to LCIO

- Prior to Snowmass05 my ILC physics analyses had been performed using a Fortran90/Simdet/Paw system. People at DESY had written silciowrite.F to convert Simdet output to LCIO reconstructed particles. However, nothing existed that allowed you to go the other way. I therefore wrote silcioread.F to read LCIO reconstructed particles into Simdet common blocks. Data that was incompatible with Simdet common blocks was read into public data members of a Fortran90 module.

Interface FORTRAN/SIMDET/PAW analysis system to LCIO

- In addition to performing physics analyses, the existing Fortran90 code also selected which .stdhep files to read out and the number of events to read out given the specified initial state electron/positron polarization, final state partons, and other parameters. Java code was only used for FastMC simulation. How did Java and Fortran communicate?
- I first tried the Java Native Interface (JNI). This was an endless source of headaches, with memory leaks the number one problem, as I recall.
- I ended up abandoning JNI, and instead used a Fortran subroutine system which synchronously executes the linux shell command given in its character string argument. I therefore started the Java FastMC program using system . The LCIO file produced by the FastMC was then read by silcioread .

Future Plans

- In my case, future SiD physics analysis code will be written using C++/Root . Despite its Paw-like quirkiness and lousy documentation, Root will enjoy unequalled support for particle physics analyses in the coming years.
- Just as there was no problem going back and forth between Java simulation code and Fortran analysis code, I see no problem doing the same with Java and C++/Root.