

# DigiSim Tutorial

Guilherme Lima

for the ILC-software group at NIU



NORTHERN ILLINOIS  
UNIVERSITY

ILC Simulations Workshop  
Boulder, January 09-11, 2006

# DigiSim usage instructions

- Download/install/build java 1.5, Maven 1.0.2, org.lcsim  
(see <http://www.lcsim.org> for details)
- Drivers needed: (all available from org.lcsim.digisim)  
CalHitMapDriver, DigiSimDriver and CalorimeterHitsDriver,  
**plus** LCIODriver (for standalone run, saving output file)  
**or** YourAnalysisDrivers (as an on-the-fly preprocessor)
- DigiSim configuration file stored on LCDetectors: digisim/digisim.steer
- Run it:
  - From command line: after setting the CLASSPATH (see docs for details)  
**java org.lcsim.digisim.DigiSimMain <inputfile>**  
an output file ./digisim.slcio will be produced, to be used for analysis or reconstruction
  - From inside JAS, as an on-the-fly preprocessor:  
DigiSimExample is available from JAS3 --> Examples --> org.lcsim examples

# Running DigiSim inside JAS3

From starting page, click on examples

--> org.lcsim examples

--> DigiSimExample.java

The image displays three overlapping screenshots of the JAS3 application interface. The top-left window shows the 'Welcome' page with a 'Welcome to JAS3!' message and links for 'release notes' and 'examples'. The middle window shows the 'Examples' page with a list of example sets: Python, Java, Pnuts, and org.lcsim. The bottom-right window shows the 'org.lcsim examples' page with a table of examples and their descriptions.

Example Name	Description
<a href="#">SimpleGenerator.java</a>	Simple diagnostic event generator.
<a href="#">Simple.java</a>	Simply prints the event header of each event analyzed.
<a href="#">Analysis101.java</a>	Intro to analysis with AIDA.
<a href="#">SimpleFastMC.java</a>	Running the Fast MC.
<a href="#">SimpleOutput.java</a>	Example of writing LCIO output
<a href="#">JetFinding.java</a>	Using the Jet Finder
<a href="#">ClusterFinding.java</a>	Cluster Finding example
<a href="#">DigiSimExample.java</a>	Digitization example

# Loading driver(s) and data

The screenshot shows the JAS3 software interface. The main window displays a code editor with the following Java code:

```
1 import org.lcsim.digisim.CalHitMapDriver;  
2 import org.lcsim.digisim.DigiSimDriver;  
3 import org.lcsim.util.Driver;  
4  
5 /**  
6  * A simple FastMC example  
7  */  
8 public class DigiSimExample extends Driver  
9 {  
10 public Digi  
11 {  
12 // Instan  
13 Driver hi  
14 // Add a  
15 add(hitm  
16  
17 // Instantiate DigiSim  
18 Driver digi = new DigiSimDriver();  
19 // Turn on diagnostic histograms  
20 // digi.setHistogramLevel(1);
```

A dialog box titled "Select Plugin" is open, showing a dropdown menu with "org.lcsim Plugin" selected. The dialog box also contains "OK" and "Cancel" buttons.

The status bar at the bottom of the window shows "11:37:39 PM ----- compile successful" and "classpath:/org/lcsim/plugin/web/examples/DigiSimExample.java 3.80/6.75MB".

Select the code window, compile it (F9) and load it (F2).

Repeat for your favourite drivers, loaded after DigiSim.

Then open an LCIO file. Make sure you select the org.lcsim plugin.

# Looking at raw hits...

The screenshot shows the JAS3 software interface. The main window displays a tree view on the left with folders for Programs, DataSets, and aida40080aida. The right pane shows a table of raw hits for 'Run:0 Event: 0'. The table has columns for CellID, Amplitude, and TimeStamp. The status bar at the bottom indicates 'Analyzed 1 records in 5812ms' and '6.56/8.27MB'.

Collection: HcalBarrRawHits size:267 flags:28000000

CellID	Amplitude	TimeStamp
52918747320549765	43559	-90
285052684468619	129328	5
-720547249997738...	43239	262
-844317555949137	42692	136
27303132870476174	55524	64
-653014515675295...	63300	77
17735801160860075	-47417	53
-402483747540168...	57165	-62
19987330391605664	49677	-114
-678354006677909...	-42588	199
-363089019011068...	-51179	33
12950498923512232	48461	89
284764921659778	56520	5
284885180744070	277069	5
46163584102695307	48640	94

KB/131.0 kB\_segmn is a NonprojectiveCylinder  
HADBarrel: Total # cells: 28851848

Compiler x Record Loop x

Analyzed 1 records in 5812ms 6.56/8.27MB

# ...and raw -> sim links

The screenshot shows the JAS3 software interface. The main window displays a file tree on the left and a central event viewer. The event viewer shows 'Run:0 Event: 3' and a table of simulation links. The table has columns for 'From', 'To', and 'Weight'. The status bar at the bottom indicates 'Analyzed 1 records in 327ms' and '8.15/12.8MB'.

From	To	Weight
EcalBarrRawHits[0]	EcalBarrHits[31]	1.0000
EcalBarrRawHits[1]	EcalBarrHits[17]	1.0000
EcalBarrRawHits[2]	EcalBarrHits[104]	1.0000
EcalBarrRawHits[3]	EcalBarrHits[109]	1.0000
EcalBarrRawHits[4]	EcalBarrHits[8]	1.0000
EcalBarrRawHits[5]	EcalBarrHits[32]	1.0000
EcalBarrRawHits[6]	EcalBarrHits[56]	1.0000
EcalBarrRawHits[7]	EcalBarrHits[11]	1.0000
EcalBarrRawHits[8]	EcalBarrHits[10]	1.0000
EcalBarrRawHits[9]	EcalBarrHits[92]	1.0000
EcalBarrRawHits[10]	EcalBarrHits[20]	1.0000
EcalBarrRawHits[11]	EcalBarrHits[3]	1.0000
EcalBarrRawHits[12]	EcalBarrHits[38]	1.0000
EcalBarrRawHits[13]	EcalBarrHits[66]	1.0000

# Live demo 1: DigiSim as a JAS3 preprocessor

- Launch JAS3 and load any physics sidaug05 data file
- Download <http://nicadd.niu.edu/~lima/ilc-simws/plot4.py> and open it from JAS3
- From the org.lcsim examples tab, open/compile/load DigiSimExample and ClusterFinding (in this order)
- At ClusterFinding.java loading, a new folder appears, like aidaXXXXaida. Update “path” variable in plot4.py accordingly
- Fire the event processing and run plot4.py right away. Observe the effect of identity transformations on clustering
- Look at some raw and digitized hit collections

# Live demo 2: updating the steering file

- Checkout LCDetectors from CVS repository  
see for instance: detectors/sidaug05/digisim/digisim.steer
- Download <http://nicadd.niu.edu/~lima/ilc-simws/digisim.steer>  
and compare it to the existing one in the detector conditions area (LCDetectors/detectors/sidaug05/digisim/)
- Replace old with new steering file and rebuild your detector .zip file (use script LCDetectors/detectors/makeZips.sh)
- Copy the relevant .zip file to ~/.lcsim/cache
- Restart JAS3 and run again, observing the effect of energy thresholds on clustering



# Live Demo 3: running DigiSim standalone

- Type this sequence of commands (on Linux)
  - > cd <some/convenient/place>
  - > wget <http://nicadd.niu.edu/~lima/ilc-simws/digi.tgz> # download a tarball
  - > tar xzvf digi.tgz # open the tarball
  - > cd digi # move to the directory with standalone java code
  - > source ./addjars.sh ~/.maven/repository # define your CLASSPATH. This  
# requires the org.lcsim libs in the repository. Use “maven jar:install”
  - > ln -sf /path/to/your/data/file inputfile # point to the input data file
  - > make # compile the java standalone code
  - > java MainLoop # run over data in inputfile
- Two files produced: digisim.slcio (digitized data) and digisim.aida
  - > jas3 digisim.aida &

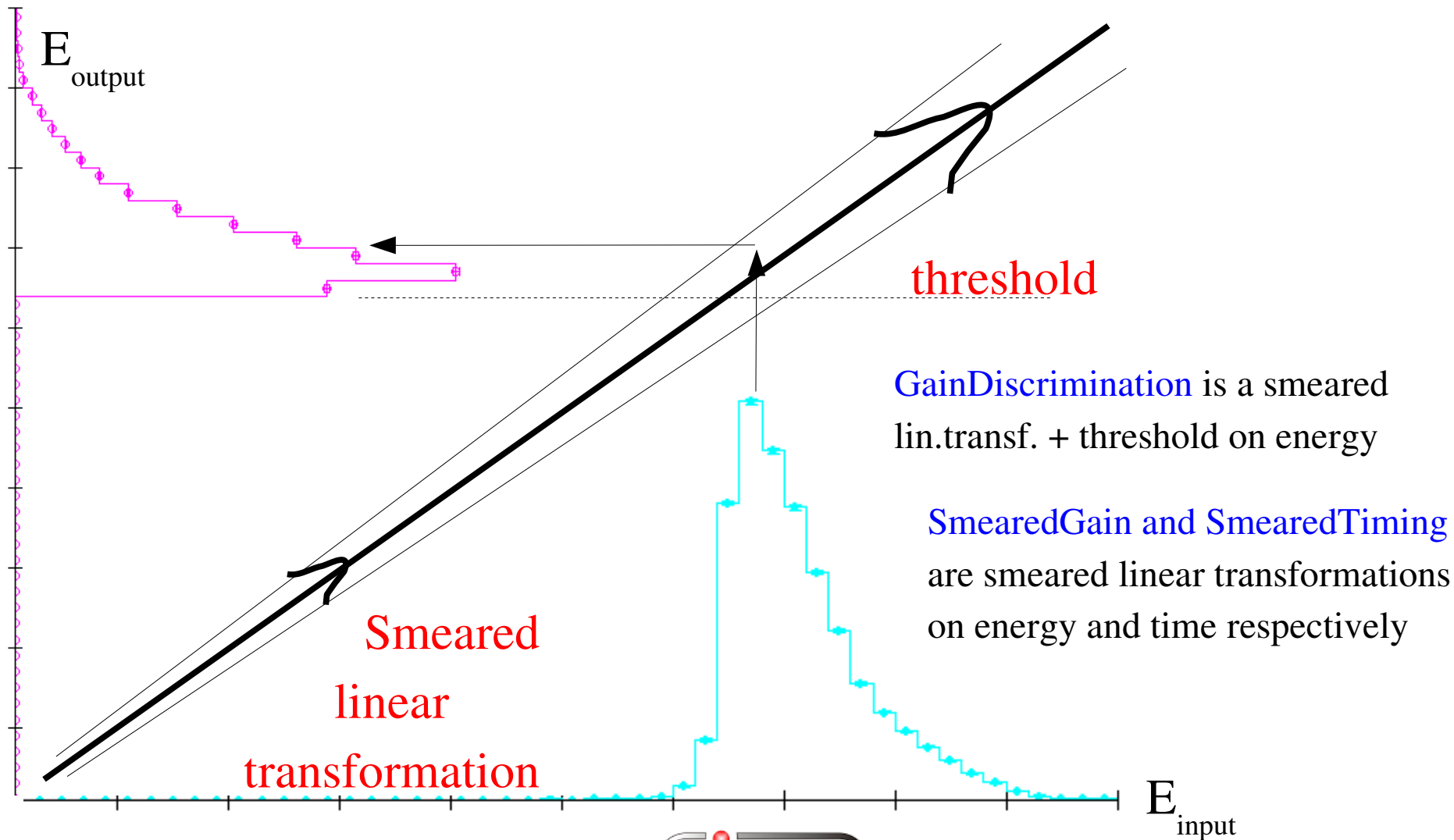
# SimCalorimeterHits or CalorimeterHits?

- Consider moving your reconstruction algorithms to use CalorimeterHits instead of SimCalorimeterHits
- How to do this:
  - All non-MC calls to SimCalorimeterHits (energy, position, time) can be transparently replaced with equivalent calls to CalorimeterHit objects.  
For MC-related methods, use (same) cellid as a key to access SimCalorimeterHits.
  - Detector conditions system is used to select the correct DigiSim configuration file.
  - Configuration files exist for most Snowmass detectors
    - All RPC-based have identity configurations
    - All scintillator-based: SDJan03, sidaug05\_scint, cdcaug05\* have non-identity configurations (see later)
  - Identity DigiSim config files are available for helping people to get started with DigiSim output

# Setting up DigiSim modifiers

- Modifiers' role is to *tweak* hit energy and timing
- Important: It is helpful to interpret the “energy” field according to the process to be modeled:  
    Energy (GeV) --> (light yield) --> # produced photons --> (photon collection) --> # photons collected --> (Quantum effic) --> # photoelectrons --> (Gain) --> uAmp --> (signal integration) --> charge collected --> (Digitization) --> ADC counts
- DigiSim modifiers are just factors (or more generally, functions) which represent each step along the digitization process

# A common transformation



# Setting up a DigiSim configuration file

```
#####  
# Example steering file for DigiSim  
#####  
.begin Global -----  
# specify one or more input files (in one or more lines)  
LCIOInputFiles inputfile  
  
# the active processors that are called in the given order  
ActiveProcessors CalHitMapProcessor  
ActiveProcessors EcalBarrelDigitizer  
ActiveProcessors EcalEndcapDigitizer  
ActiveProcessors HcalBarrelDigitizer  
ActiveProcessors HcalEndcapDigitizer  
ActiveProcessors OutputProcessor  
  
# limit the number of processed records (run+evt):  
MaxRecordNumber 500  
.end Global -----  
#####  
.begin EcalBarrDigitizer  
ProcessorType DigiSimProcessor  
  
InputCollection          EcalBarrHits  
OutputCollection         EcalBarrRawHits  
Raw2SimLinksCollection  EcalBarrRaw2sim  
  
ModifierNames           EMBDigiIdentity  
# modifierName          Type                Parameters (floats)  
EMBDigiIdentity         SmearedGain          100000000          0  
.end -----
```

One digitizer per subdetector

“Identity” factor  $10^8$  avoids  
precision loss in the conversion  
double -> int -> double

# Configuring processors and modifiers

```
#####  
# A subdetector digitizer. It instantiates one or more calorimeter hit  
# "modifiers", which together represent the full digitization process  
.begin HcalBarrDigitizer  
  
ProcessorType DigiSimProcessor  
  
InputCollection HcalBarrHits  
OutputCollection HcalBarrRawHits  
Raw2SimLinksCollection HcalBarrRaw2sim  
  
ModifierNames HBlightYield HBcrosstalk HBlightColleff HBPDQuEffic HBExpoNoise HBGaussNoise  
HBdiscrim HBGain  
  
# Parameters:  
# modifierName      Type          gainNom  gainSig  thresh  thrSig  
HBlightYield        GainDiscrimination 10000000 0        1        0  
  
# Crosstalk  
# modifierName      Type          mean     sigma  
HBcrosstalk         Crosstalk     0.020    0.005  
  
# Smeared gain parameters:  
# modifierName      Type          gain     gainSigma  thresh  thrSig  
HBlightColleff      GainDiscrimination 0.0111   0.0029    1        0  
HBPDQuEffic         GainDiscrimination 0.15     0         1        0  
  
(...Truncated... see file cdcaug05_np.steer in DigiSim area)  
.end -----
```

(As many as needed)

# Simple example: configuration for the tile HCal

- **Scintillation:** 100 eV / photon , or  $10^{+4}$  photons/MeV

Ex: a MIP at normal incidence on 0.5cm-thick scintillator deposits ~ 0.9MeV, or 9000 photons

==> use GainDiscrimination modifier with  $10^{+7}$  photons/GeV and a threshold at 1 photon

#modifierName	type	gainNom	gainSig	thresh	thrSig
HBlightYield	GainDiscrimination	10000000	0	1	0

- **Light crosstalk:**

first neighbors: 1.5% to 2% --> (2.0 +/- 0.5) %

HBCrosstalk	Crosstalk	0.020	0.005		
-------------	-----------	-------	-------	--	--

- **Photon collection efficiency (QE~15%):**

9000 scint.photons/MIP --> 15 PE/MIP detected (cosmics measurements at NICADD)

$15 / 0.15 = 100$  incident photons =>  $\text{Eff}_{\text{coll}} = 100 \text{ inc.} / 9000 \text{ tot.scint.} = 0.0111$

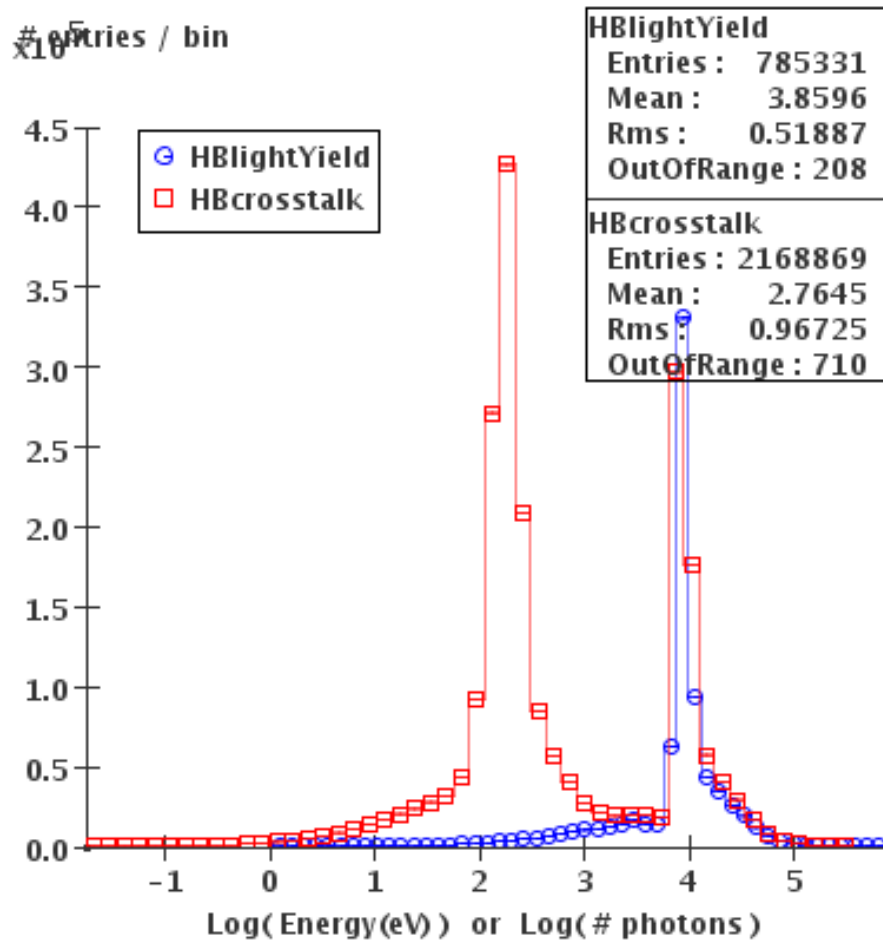
Variance (Poisson):  $\sigma_N^2 = \langle N \rangle \rightarrow$  for  $\langle N_{\text{PE}} \rangle = 15$ ,  $(\sigma_N / N) \sim 26\%$

Therefore:  $\text{Eff}_{\text{coll}} = 0.0111 \pm 0.0029$  => use GainDiscrimination with smearing

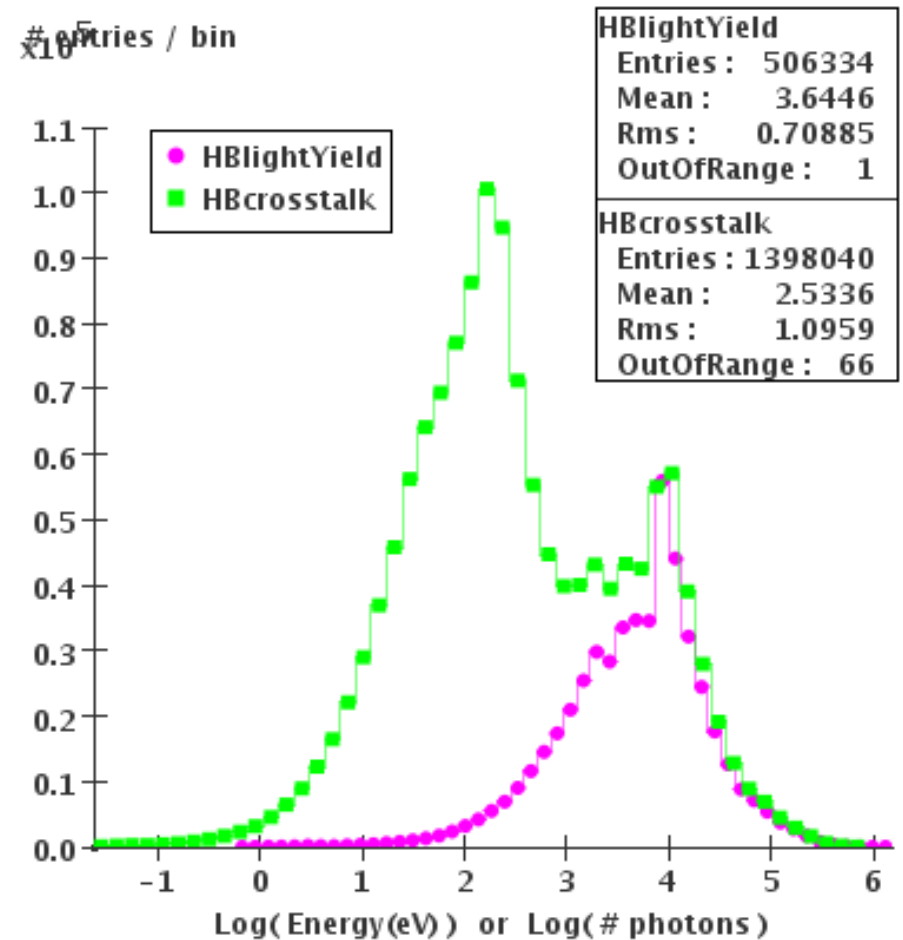
HBlightColleff	GainDiscrimination	0.0111	0.0029	1	0
----------------	--------------------	--------	--------	---	---

# Light cross-talk

100GeV muons - DigiSim for cdcaug05\_np HCal



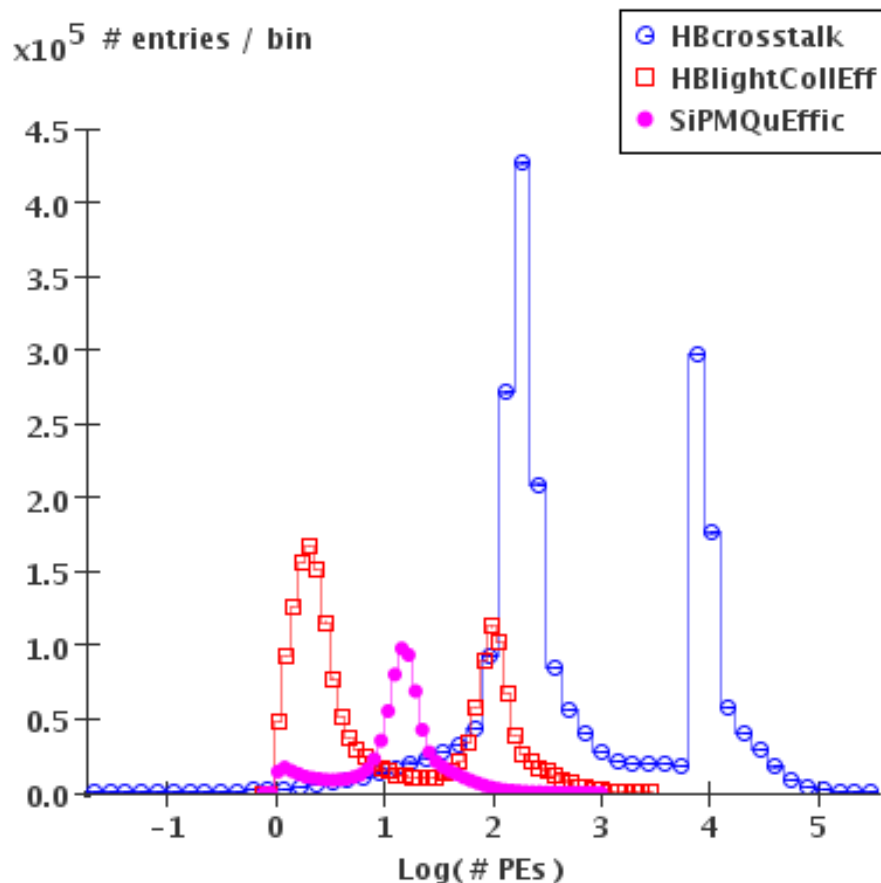
20GeV pions - DigiSim for cdcaug05\_np HCal



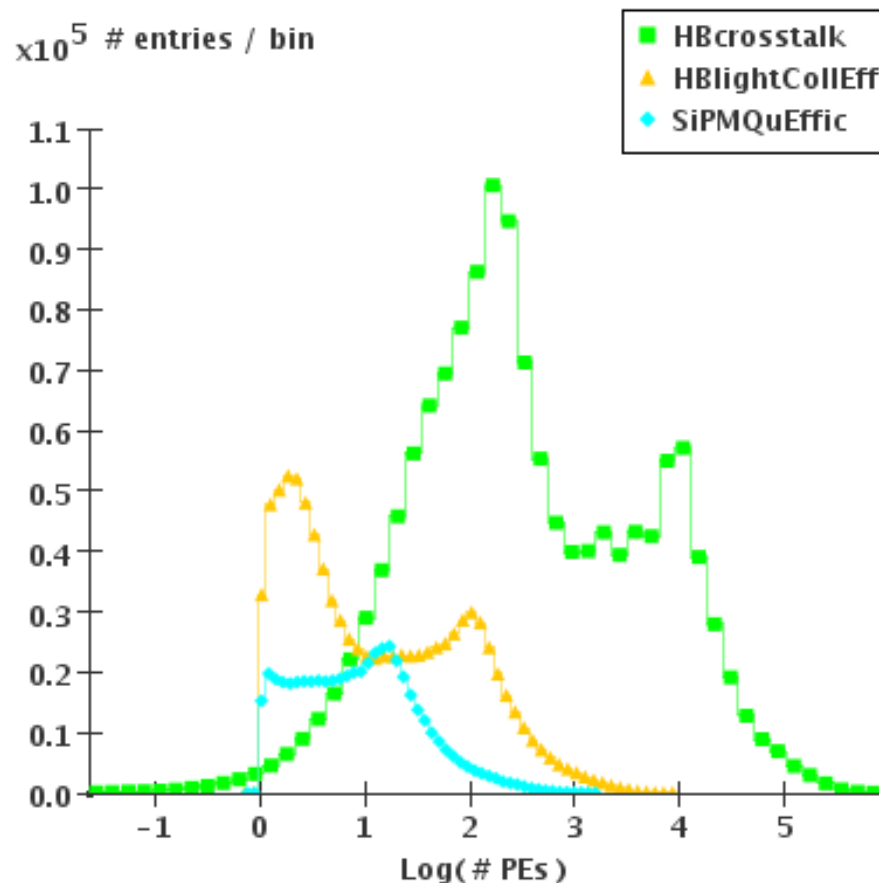


# Photodetection

100 GeV muons - DigiSim for cdcaug05\_np HCal



20 GeV pions - DigiSim for cdcaug05\_np HCal



# Simple example: parameters from the tile HCal

- **Photosensor detection efficiency:** QE ~ 15 %

```
HBPDQuEffic      GainDiscrimination      0.15      0      1      0
```

- **Noise simulation:**

- Photosensor noise: exponential distribution (guess: mean 0.6)
- Electronics noise: gaussian distribution (guess: mean 0, sigma 1.6, keep +/- tails)

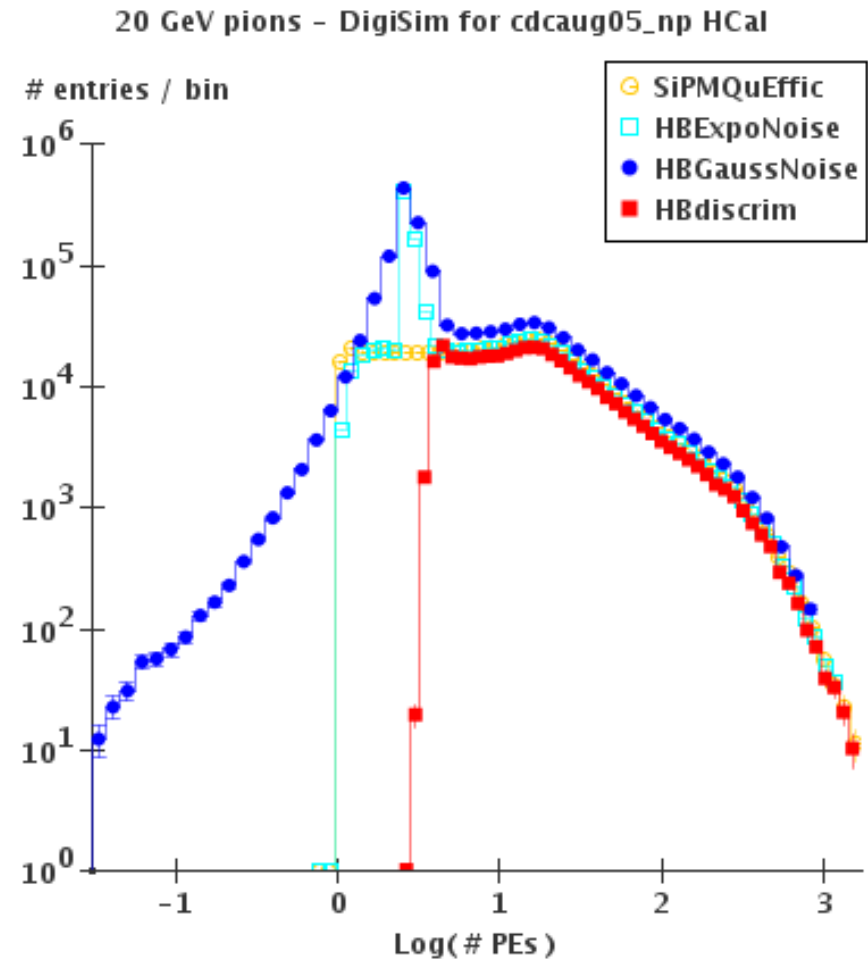
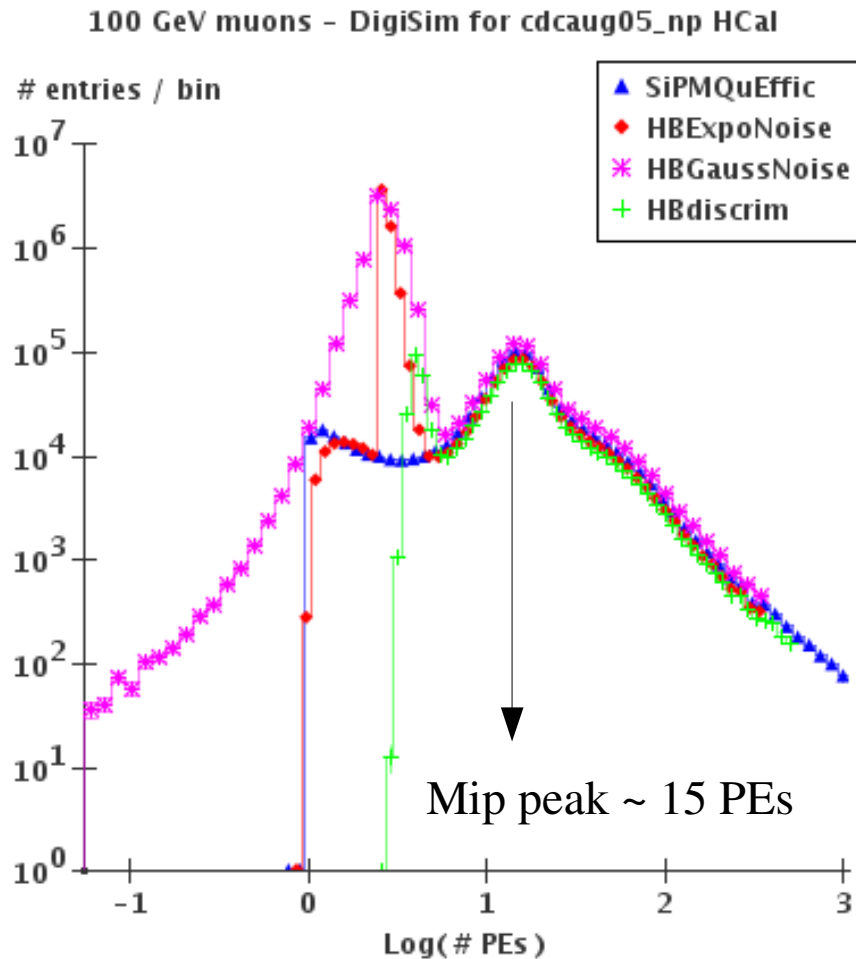
```
# GaussNoise parameters:      sys      be      Ecut      TimeNom      TSig      Mean      Sigma
# Note: sigma<0 means that threshold acts on absolute value only
HBGaussNoise      GaussianNoise      3      0      2.5      100      100      0.0      -0.58
```

```
# ExponentialNoise parameters:  sys      be      Ecut      TimeNom      TSig      Mean
HBExpoNoise      ExponentialNoise      3      0      2.5      100      100      0.23
```

- **Discrimination:** ¼ MIP cut ~ 4 PEs: threshold at 4 +/- 0.25 (on abs value)

```
# Discrimination      threshold      sigma
HBdiscrim      AbsValueDiscrimination      4      0.25
###HBdiscrim GainDiscrimination      1      0      4      0.25
```

# Noise and discrimination

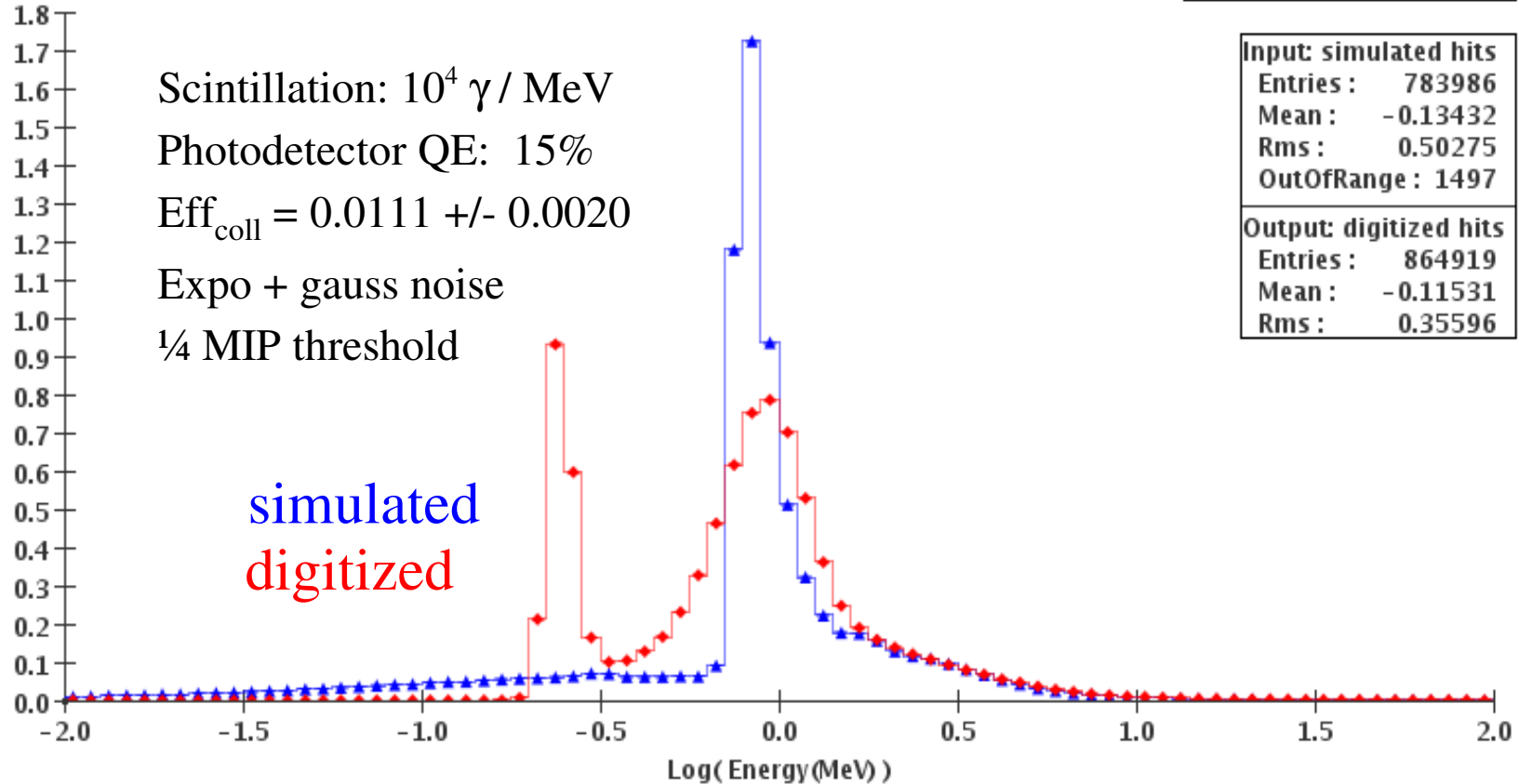


# HCal scintillator digitization (preliminary)

100 GeV muons - DigiSim for cdcaug05\_np HCal

# entries / bin  
 $\times 10^4$

▲ Input simulated hits  
◆ Output digitized hits



# HCal scintillator digitization (preliminary)

20 GeV pions - DigiSim for cdcaug95\_np HCal

