Automated Calorimeter Calibration using the Energy Conservation Law

DESY Summer Student Program 2007

Viktor Hangartner
Supervisor: Oliver Wendt

September 2007

Abstract

The Large Detector Concept is a design for a future, high precision detector foreseen to investigate the nature of physics at centre-of-mass energies up to 1 TeV using electron - positron collisions at the International Linear Collider. An approach to an automated method to perform a global calorimeter energy calibration using the energy conservation law has been implemented in a Marlin processor. This report describes the method itself and presents some results as well.

Contents

1 Introduction

1.1 The International Linaear Collider

The International Linear Collider (ILC) [3] is a high precision electron positron collider reaching centre-of-mass energies up to 500 GeV. An extension up to 1 TeV is foreseen. The ILC is thought to be the next step investigating the nature of physics at high energies after the experiments at the Large Hadron Collider at CERN, Geneve have been performed. It is foreseen to install two detectors at the ILC, whereas only one of them will be residuing at the interaction point. Exchanging the Detectors will be possible due to a "push-and-pull" mechanism allowing easy exchange.

1.2 The Large Detector Concept

The Large Detector Concept (LDC) [2] is one of four proposals to build a high precision detector for the ILC. The LDC calorimeter consists of two parts: the electromagnetic calorimeter (Ecal) and the hadronic calorimeter (Hcal). The Ecal is a silicon-tungsten sampling calorimeter having in total 30 layers. One layer consists of a sensitve silicon pad and a thungsten absorber. The 30 layers are split in two sections having different sampling fractions. The transversal cell size in the Ecal is 1 x 1 cm^2. The Hcal is a steel - scintillator sampling calorimeter having a cell size of 3x3x3 cm^3

1.3 Marlin

Marlin [4] is a simple modular application framework for analysis and reconstruction code based on the LCIO [5] data model. The main purpose of Marlin is to provide a framework to the worldwide ILC community that is easy to handle an allows distributed modular software developement. Each Marlin Module is of the basic class `Marlin::Processor`. Marlin is steered by a file steerfile.xml (see Section A). Input data (slcio files) which will be processed by the selected Marlin modules and the parameters required by the processors are indicated in the steering file.

The task is to develop a Marlin module, i.e. a processor, that is able to perform an calorimeter energy calibration. Due the future "mass-production" of Monte Carlo events based on full detector simulation (Geant 4) [6] a completely automated and stable calibration procedure has to be developed. The basics of a method to perform a energy calibration will be described in Section 2.

2 Calorimeter Energy Calibration

In general, calibration "refers to the process of determining the relation between the output (or response) of a measuring instrument and the value

of the input quantity or attribute, a measurement standard“ [7]. Applying the above definiton to the LDC calorimetry we get the following overview:

- The measuring instrument are the calorimeters in the LDC detector.
- The input quantity for this particular method of calorimeter calibration is the center-of-mass energy E_{CM} of the e^+e^- collisions.
- Output of the calorimeters is the measured energy deposit of particles $E(\vec{x})$ as a function of space, i.e clorimeter cell.
- The relation in between input and output can basically be described as

$$E_{\mathrm{CM}} = f(E_{\mathrm{tot}}), \tag{1}$$

  where E_{tot} is the total energy deposited by paricles in the calorimeter. Assuming a relation of e.g.

$$f(c, E_{\mathrm{tot}}) = c * E_{\mathrm{tot}} \tag{2}$$

  calibrating the calorimeter means to determine the constant c.

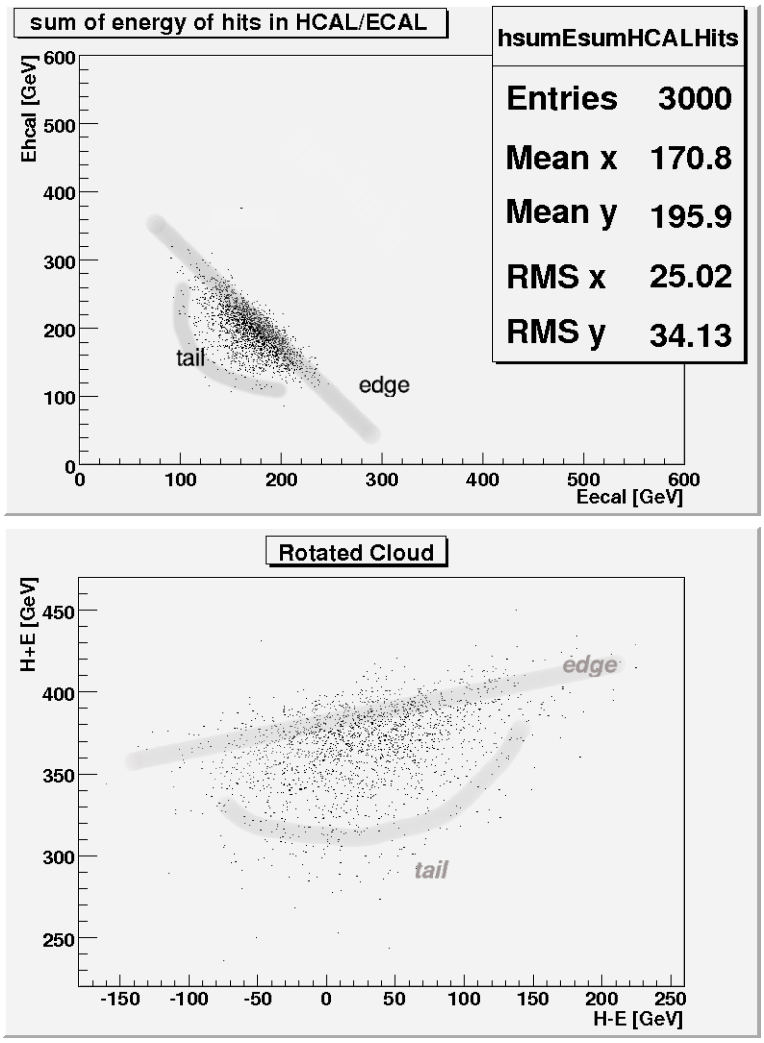
Generating output from a well known input quantity to perfor a calibration, one can use several methods, whereas some are mentioned in the following:

Using computer simulations one can shoot single particles to the calorimeter and study the response. Electrons, photons or photons are usually used to calibrate the electromagentic calorimeters and charged pions to to calibrate the hadronic calorimeter. However, this method is very complex. Especially to calibrate a hadronic calorimeter on this way is a highly sophisticated task.
Known radioactive sources can be set into the detector to calibrate it by measuring the energy deposit of the radiation.
In addition the Engegy conservation for entire events (see Equation 2) can be used to calibrate calorimeters. The big advantage of this approach is its applicability to both simulated events and data of real events. It is also good to perform a first, rather raw calibration of simulated data samples.

As the LDC detector is still under construction, i.e. design studies are performed, only simulation of both collisions (events) and the calorimeters itself are available. So there is no real data yet. The LDC calorimeter consist of three different sampling structures as there are two different sampling structures in the Ecal and one in the Hcal. Therefore the relation to determine is

$$E_{CM} = \underbrace{c_1 * E_{1,\mathrm{vis}} + c_2 * E_{2,\mathrm{vis}}}_{E} + \underbrace{c_3 * H_{\mathrm{vis}}}_{H}, \tag{3}$$

where the c_i are the calibration coefficents that are to be determined. E and H are the measured total energies of an event that were desposited in the electromagnetic and hadronic calorimeter respectively. A visusalisation of the calibration of a set of events is achieved by plotting the energies H vs E as it is shown in Figure 1. The scatter plot shows a cloud of entries having an upper edge and a tail towards lower energies. This is due to energy losses caused by neutrinos, muons which deposit only a little percentage of their energy in the calorimeters and the finite acceptance of the calorimeter.



(a)

(b)

Figure 1: (a) Each point in the scatterplot represents a e^+e^- to six jet event. The total measured energy in the hadronic calorimeter H (Ehcal in the plot) is plotted versus the total measured energy in the electromagnetic calorimeter (Eecal in the plot). Theoretically, the center-of-mass energy is conserved and $E_{\mathrm{CM}} = H + E + E_{\mathrm{miss}}$, therefore the cloud shows an upper edge. As neutrinos, muons and the finite acceptance of the calorimeter cause missing energy in the measurement, E_{miss}, the cloud has a tail towards low energies. (b) A rotated cloud (ot exactly the same as in (a)) after the coordinate transformation $x = E \rightarrow x' = H - E, \quad y = H \rightarrow y' = H + E$.

Given a uncalibrated data set the new calibration cefficients are calculated as shown in Formulae 4 to 11. The equations follow the method described by V. L Morgunov in "Calorimeter energy calibration using the energy conservation law" [1]. Basically, Equation 5 describing the uncalibrated data sample is multtiplied by the factor $f = E_{\mathrm{CM}}/E_0$ which contains

the requirement of energy conservation in an event. This multiplication is actually a simple rescaling of the total energy E_0 of the event measured with both calorimeters to get the centre-of-mass energy E_{CM}. The parameters a_0 and E_0 of the function $g(E)$ in Equation 4 are determined by fitting $g(E)$ to datapoints describing the orientation of the cloud. How to calculate the parameters from a given data set will be described in Section 2.2.

$$g(E) = a_0 \cdot E + E_0 \tag{4}$$
$$|a_0| \cdot (c_1 \cdot E_{1,\mathrm{vis}} + c_2 \cdot E_{2,\mathrm{vis}}) + c_3 \cdot H_{\mathrm{vis}} = E_0 \tag{5}$$
$$|a_0| \cdot f \cdot c_1 \cdot E_{1,\mathrm{vis}} + |a_0| \cdot f \cdot c_2 \cdot E_{2,\mathrm{vis}} + f \cdot c_3 \cdot H_{\mathrm{vis}} = f \cdot E_0 \tag{6}$$
$$f = E_{\mathrm{CM}}/E_0 \tag{7}$$
$$c_{1,calib} = |a_0| \cdot f \cdot c_1 \tag{8}$$
$$c_{2,calib} = |a_0| \cdot f \cdot c_2 \tag{9}$$
$$c_{3,calib} = f \cdot c_3 \tag{10}$$
$$c_{1,calib} \cdot E_{1,\mathrm{vis}} + c_{2,calib} \cdot E_{2,\mathrm{vis}}) + c_{3,calib} \cdot H_{\mathrm{vis}} = E_{\mathrm{CM}} \tag{11}$$

In the following Section will be described how which adaptions to the above method have been made to develop the Marlin processor CalibProcessor that is able to perform a recalibration of an existing, not well calibrated set of data. An initial calibration is the same as recalibrating an data sample where $c_i = 1$. Therefore, the processor will also be applicable for this task which has to be fulfilled may times as a large number of Monte Carlo data samples need to be calibrated during a "mass-production" of Monte Carlo events based on a full detector simulation.

2.1 Calorimeter Calibration with CalibProcessor

Basically the method to perform an energy calibration using the energy conservation law is implemented in the Marlin processor CalibProcessor. The goal is to calculate the new calibration coefficients c_i automatically. Crucial point of the procedure is to determine the upper edge as its position indicates the measured energy E_0 of an event. The edge is desctribed by the function $g(E)$. The parameters a_0 and E_0 have to be determined in order to calculate the new calibration coefficients according to Formulae 8 to 10. Fitting the $g(E)$ to the data in the H vs. E representation would cause large errors on the parameters due to binning effects. Therefore the cloud is in a first step rotated to get a horizontal position of the edge. The coordinate transformation is introduced to do the rotation:

$$x = E \quad \rightarrow \quad x' = H - E \tag{12}$$
$$y = H \quad \rightarrow \quad y' = H + E \tag{13}$$

Bin limits in H-E [GeV]								
-250	-100	-50	0	25	50	100	150	250

Table 1: The rotated cloud, see Fig. 1(b) is divided in eight binsas shown.

The second step is to slice the rotated cloud in eight H-E bins indicated in table 2.1.

In each bin the distribution of the total measured energy H+E is of interest as the position of the maximum $\text{H+E}|_{\max}$ indicates the position of the edge, see Figure 4. Details about finding $\text{H+E}|_{\max}$ are described in Section 2.2. Except for the outermost bins the bin centre $\text{H-E}|_i$ and $\text{H+E}_{\text{i, max}}$ are determined. Fitting a stight line $h(H-E) = a \cdot (H-E) + b$ to the data set ($\text{H-E}|_i$, $\text{H+E}_{\text{i, max}}$), $i = 1, \ldots, 6$ determines the position of the edge. The parameters a_0 and E_0 are calculated according to the following formulae and the new calibration coefficients are given in Equations 18 to refc3n.

$$h(H-E) = a \cdot (H-E) + b \tag{14}$$

$$a_0 = -\frac{1+a}{1-a} \tag{15}$$

$$E_0 = \frac{b}{1-a} \tag{16}$$

$$f = \frac{E_{\text{CM}} \cdot (1-a)}{b} \tag{17}$$

$$c_{1,calib} = \frac{1+a}{1-a} \cdot f \cdot c_1 \tag{18}$$

$$c_{2,calib} = \frac{b}{1-a} \cdot f \cdot c_2 \tag{19}$$

$$c_{3,calib} = f \cdot c_3 \tag{20}$$

$$\tag{21}$$

2.2 Determination of the Edge

In this section the determination of the edge is described. This step is crucial for the calibration methode. The edge of the cloud is determined using the coordinates H+E vs H-E as described in Section 2.1. The upper edge is assumed to be a stright line. Therefore it is sufficient to determine the position $(H-E, H+E)$ of the edge in several areas of the cloud and fit a stright line to the selected set of data points, see Figure 7. In the current imlementation of the CalibProcessor the cloud is divided in eigth bins, see Table 2.1. Exept for the outermost bins, the position of the edge is determined in each bin by examining the distribution of the total Energy H+E. The location of the upper edge is currently considered to be the position of

the maximum H+E$|_{\max}$ of the distribution, see Figures 2 and 4.

In this way six points $(H-E|_i, H+E|_{\max,\,\mathrm{i}})$, where $H-E|_i$ is the bin centre, give the position of the edge. A line $f(H-E) = a_0 * (H-E) + b_0$ is fitted to the six points. The maximum can be determined by various methods. I concentrated on two approches. The first one, which is looking for the position of the bin having most entries, is a very simple one that was basically used to check out wheter the calibration method is working. The second approach is to fit a function the histogram describing the shape of edge best and to determine the position of the maximum of the function.

Method 1 — Selecting the maximal bin:
Methode 1 is very easy: the bin having the maximal number of entries is selected and its position H+E$|_{\max}$ on the axis is considered as the position of the edge. However, this methode has obvious disatvantages as an appropriate binning has to be chosen to find a unique maximum and will not be likely to fullfill the requests of an precices and automated calibration procedure. This metod is implemented in CalibProcessor.

Method 2 — finding the edge by a fit:
The basic idea used in methode 2 is to describe the distribution apearing in the bins of the cloud with a function and determine the position of the functions maximum. To check out wheter a function can be fitted to the histogram and how to achieve an appropriate result, studies have been done using the ROOT [8] analysis framework.

As the distribution, see Figure 2, has a tail on the lefthand side[1] and looks like a gaussian on the righthand side, one could try to convolute a gaussian with other functions to describe the whole distribution. However, only the position of the edge of the cloud is of importance to perform the calibration. Therefore, it is sufficient to fit a gaussian only to the righthand side (i. e. the edge) of the distribution. As one can see in Figure 2, iteration step 6, a gaussian is suitable if only the rightand part of the data is considered and the lefthand tail is ignored.

To find the suitable range $[a, b]$ in which the binned maximum likelyhood fit of a gaussian is performed, an iterative procedure has been implemented. The C++ program and its application on processor output is described in Section 3. The iterative fit procedure contains the following steps:

1. Fitting a gaussian to the whole distribution. The initial fit parameters p_1, p_2, p_3 of the gaussian

$$g(x) = p_0 \cdot e^{-0.5\cdot(\frac{x-p_1}{p_2})^2} \tag{22}$$

[1] due to not measurable energy (neutrinos, finite detector acceptance,...)

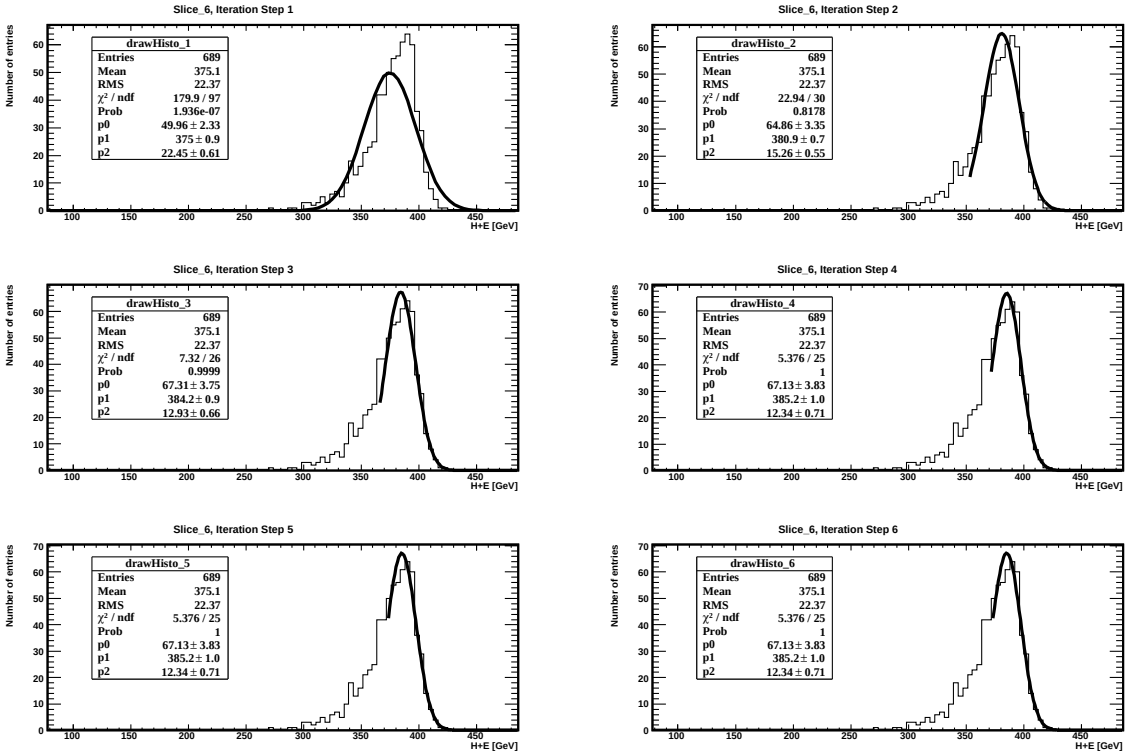


Figure 2: Each of the histograms shows the same distribution of the H+E values in the bin containing all events having H-E values in $[50, 100[$ GeV. Six iteration steps were performed to determine the range in H+E where the assumed gaussian fits the shape of the righthandside the best.

and the initial range $[a, b]$ the fit is performed on have to be chosen suitable to the given situation in the bin. Compare with Figure 2, iteration step 1.

2. The fit range $[a, b]$ is reset to $[p_1 - p_2, b]$ and the fit is reperformed.

3. Repeat step 2 as long as necessary. The number of iterations should be limited if certain criteria is fulfilled, i.e the quality of the fit using χ^2/ndf or a certain number of iterations.

As in a gaussian the the position of the maximum $x|_{\max}$ is the same as the mean, $x|_{\max} = p_1$ The edge of the cloud therefore is determined by the mean p_1 of the determined rightsided gaussian. See Figure 2, iteration step 6. However, to consider the edge at the position of the maximum may not be the most appropriate definition. Considering a different position of the edge is easy once the gaussian is determined. One could take $p_1 + 1 * p_2$ as edge position and its implementation would be straightforward as only minor changes in the code are necessary.

As Figure 2 shows this metod of determining the shape of the distribution in a bin by a righthandside gaussian works quite well. The next step in improving the performance of CalibProcessor would therefore be to implement this metod in the processor source code. To be consitent with the guidelines for Marlin applications this implementation should be done using the GNU Scientific Library GSL [9] providing similar tools as ROOT does. The implementation using GSL should be easily possible as only standard tools like fitting a gaussian to a histogram are used.

2.3 Statistical Limit on the Number of Bin Entries

The data sample of simulated $t\bar{t}$ to six jets events contains not more than 3000 events. As these events are distributed on a total of eight bins only about 500 events are to expect in a central bin of the cloud. To study the error of the mean of the fitted gaussian more statistics are needed. Therefore, a data sample for this process with 50000 events has been generated, see figure3. Based on this large data set the estimated errors on the mean of the guassian has been studied.

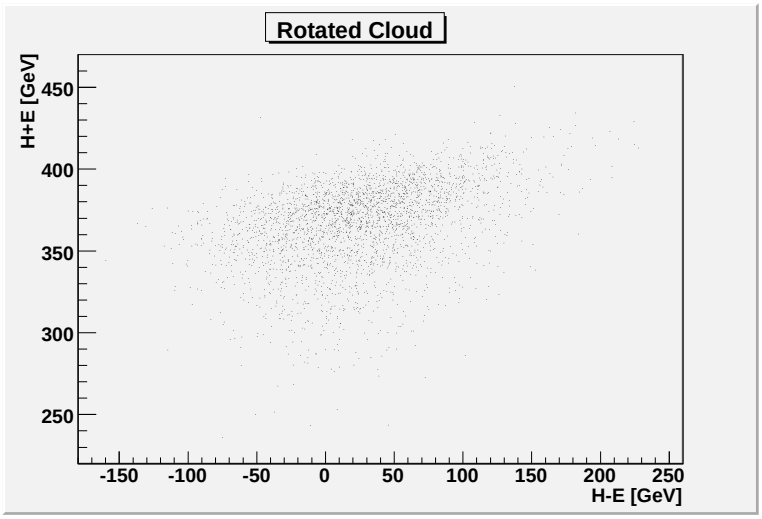


Figure 3: This cloud has about a factor ten times more entries than the first data sample of $t\bar{t}$ to six jets events contained. The cloud has 50000 entries and provides enough statistics to study statistical effects in the fit procedure used to determine the edge of the cloud.

Only one bin, containing the entries having H-E $\in [-10, 10]$ GeV, is selected (see Figure 4). This bin contains contains 3115 events of the 50000 events in the cloud. To study the error on the mean as a function of the entries in the bin (see Figure 5), only a fraction of $\frac{n}{10} * N_{\text{bin}}, \quad n = 1, \ldots, 10,$ has been selected and the fit procedure has been performed on the ten

subsamples. Important is that the error is likely to become stable above 2500 entries in a bin as it gives an lower limit of the number of events a slice should become. The first data point at 500 entries corresponds to a total number of 5000 entries and the last data point at 3100 entries corresponds to a total number of 50000 entries in the cloud.

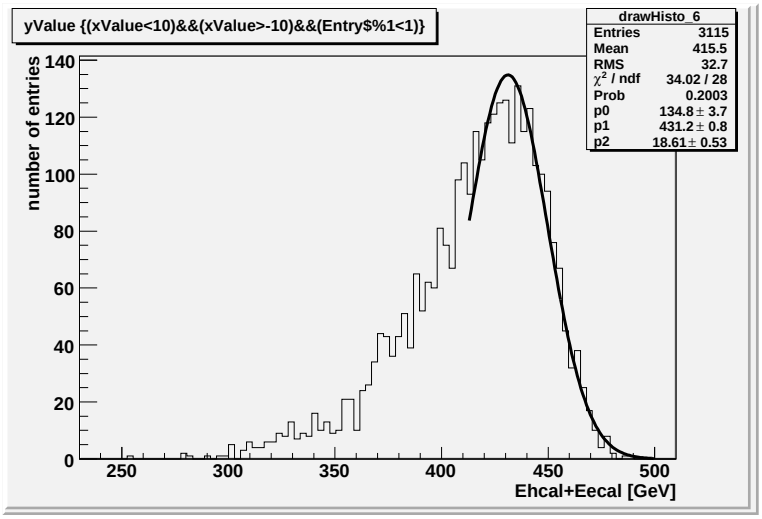


Figure 4: This Figure shows the distribution in the bin $(H - E) \in [-10, 10]$ GeV. The histogram contains 3115 events that are selected of the central region of the cloud. The full data sample contains 50000 events.

3 The CalibProcessor

The calorimeter calibration is implemented as described above using method 1. Some constants as E_{CM} of the events have to be set in the sourcecode. Therefore, the processor is not ready to any kind of input data without changes in the source code. All histograms are allocated twice: once as GSL histograms and for a second time as Aida histograms to provide data transfer to ROOT for the fiting studies that have been done. To use the processor first create a suitable steering file and execute it using Marlin.

Input — Steering Parameters:
A example of a steering file is shown in the Appendix. The CalibProcessor needs the follwoing parameters:

Name of the ECAL hit collection: Default setting = Ecal
Name of the HCAL hit collection: Default setting = Hcal
The existing calibration coefficients: c_1, c_2, c_3.

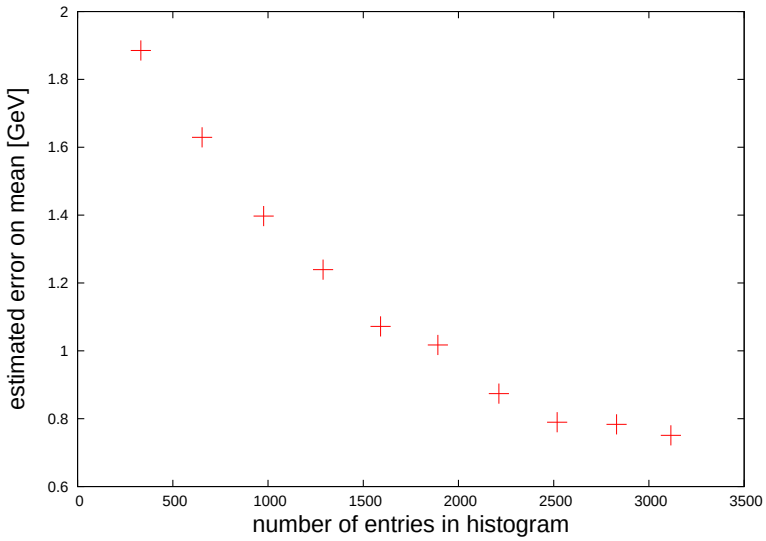


Figure 5: The plot shows the of the error on the mean as a function of the numer of events in the bin. The bin range in (H-E) is $[-10, 10]$ GeV and it contains $N_{\text{bin}} = 3115$ events. To get lower statistics only a fraction of $\frac{n}{10} * N_{\text{bin}}, n = 1, \ldots, 10$ has been selected. The error seems to become stable above 2500 entries in the. This gives an lower limit to the of the number of events a bin should contain of 2500 events. The first data point at 500 entries corresponds to a total number of 5000 entries in the cloud and the last one to 3100 entries.

Output:
CalibProcessor generates the output files listed in Table 2 and stores them in the directory Marlin runs. The current processor is still in a rather experimental implementation. All files except the file CalibProcessor_calibration_coeff.txt are only to provide the data transfer to ROOT or Gnuplot. The scripts to generate plots using Gnuplot and to perform the iterated fit procedure using a ROOT are also decribed.

Filename	Content
CalibProcessor_calibration_coeff.txt	Calibration coefficients $c_{1,\mathrm{calib}}$, $c1, \mathrm{calib}$, $c_{1,\mathrm{calib}}$.
Output.root	ROOT trees of all created histograms. To generate this output the Processor MyAidaProcessor has to be enabled.
CalibProcessor_data_fit.txt	Data points (H-E$\vert_{\mathrm{i}}$, H+E$\vert_{\mathrm{i,\ max}}$) to fit edge h(E) to.
CalibProcessor_data_histo.txt	Data of the 2d gsl histogram H vs E
CalibProcessor_data_histo_r.txt	Data of the 2d gsl histogram H+E vs H-E
CalibProcessorRot_lparam.txt	Parameters of the fitted funcion $g(E) = a_0 \cdot E + E_0$ and $h(H-E) = a \cdot (H-E) + b$ where a_0 = a_prime and E_0 = b_prime

Table 2: Output files produced by the CalibProcessor.

How to perform the fit of a gaussian to the H+E distribution in a bin of the cloud using executeScriptSlice.C:

Script:	executeScriptSlice.C
Task:	Fiting a gaussian to the histogram to. The mean of the gaussian is the edge position.
Comments:	The script compiles and executes all needed functions itself. Number of iterations that are performed has to be changed in the script.
Input:	DrawSliceHistogram.C FitRsGaussToSlice.C SetStyle.C Output.root
Output:	Slice_i.eps, $i = 1, \ldots, 8$ FitRsGaussToSlice_output.txt
Step 1:	Define the Number of iteration N_iter steps you want to perform in the file executeScriptSlice.C by setting the argument of `FitRsGaussToSlice(N_iter)`.
Step 2:	Start ROOT.
Step 3:	Load the file Output.root containing the trees called Slice_1, ..., Slice_8.
Step 4:	Type `.x executeScriptSlice.C` to execute the script.
Step 5:	The output is written into the current directory.

How to create plots like Figurefig:recal shows using the script gnuplot_CalibProcessor_contour.sh:

Script:	gnuplot_CalibProcessor_contour.sh
Task:	Creates plots like Fig. 7 from processor output.
Comments:	
Input:	CalibProcessor_data_fit.txt, CalibPro cessor_data_histo.txt, CalibPro cessor_data_histo_r.txt, CalibPro cessorRot_lparam.txt
Output:	CalibProcessorRot_data_histo_con tour_data.txt, CalibProcessorRot_data_histo_contour.eps, CalibProcessorRot_data_histo_r_contour_data.txt, CalibProcessorRot_data_histo_r_contour.eps
Step 1:	Run CalibProcessor on data. Several ASCII files are created as indicated in reftab:files.
Step 2:	Run the script gnuplot_CalibProcessor_contour.sh in the same directory. Commandline argument of the script is `CalibProcessor_data_histo`. The plots are saved as encapsulated postscript files do not appear on the screen. They are written directly into the current directory.

3.1 To do

- The Implementation of the iterated fit procedure (methode 2) in the processor using GSL has to be done. In addition to the described method the iteration should be performed until a certain criteria is fulfilled, i.e the quality of the fit using χ^2/ndf or a certain number of iterations.
- Automated, non aequidistant binning in $[H - E]$ of the cloud has to be implemented as it is already not suitable anymore if the processor is applied to a well calibrated data sample, compare Figure 7. The number of of bins has to be selected with respect to the number of entries in the full data sample and the entries in the bins in order to keep the error on the fit parameters minimal. See Section 2.3.
- The processor has to be tuned to be user friendly, e.g more steering parameters such as E_{CM} of the used file should be defined.
- The gnuplot script could be implemented in the Processor as well, if desired.

4 Results

The following plots in Figures 7 and 9 show that implemented method to perform and automated calorimeter calibration is basically working.
In Figures 7 and 9 two examples of a recalibration of a data set are given. The used data is for both the same. The difference is that different initial calibration coefficients are used. In Figure 7 wrong initial calibration coefficients as indicated in 3 were chosen on purpose. In Figure **??** the calibration coefficients used in 2006 for detector optimisation and PFA studies were examined. It turned out that the used calibration was not perfect. Therefore a recalibration of those data samples could be considered.

However, the performance of the existing processor seems not to be very good as one takes into accout that in both Figures 7 and 9 the same data sample has been used and different calibrations result. Both the plots and some the coefficients show a clear deviation, i.e the difference of a factor of 2 between the new c_2 coefficients is not understood, whereas the new coefficients c_1 and c_3 seem to agree.

Before doing necessary processor performace studies to understand the error on the calibration coefficients, the processor must be improved with respect to tune the determination of the edge position as described in Section 3.1.

Calibration Coefficients:

Calibration Coefficients				
coeff	Processor test		Calibration used 2006	
	old	new	old	new
c_1	33.0235	49.8505	46.7703	53.7973
c_2	93.5682	141.27	86.3749	99.3523
c_3	21.19626	24.6161	22.01925	23.2179

Table 3: The calibration coefficients that have been used (old) and the coefficients calculated by the processor (new) are indicated in the table. To test the Processor, wrong old coefficients have been put into the processor. The new coefficient have been calculated and are listed above. It is not understood is why the two different inputs bring different new calibration coefficients, i.e. the deviation of 50 in between the new c_2 coefficients is remarkable.

Determination of the edge: maximal bin vs. gaussian
The two methodes are compared by visualising the determined edge on a plot to get a first hint on the improvement of the performace using the iterated fit procedure. As Figure 10 shows the edge determined with the iterated

fit procedure (black) describes the oritentation of the cloud better as the method implemented in the processor does (red). The 'red' fit is performed within the processor and its parameters a and b are loaded by the Gnuplot script to plot the graph. As the iterated fit procedure is not implemented in the processor, the position of the edge has been calculated using the ROOT script. The fit of the black line to the data is performed in gnuplot to get a visualisation of the performance of the iterated fit procedure.

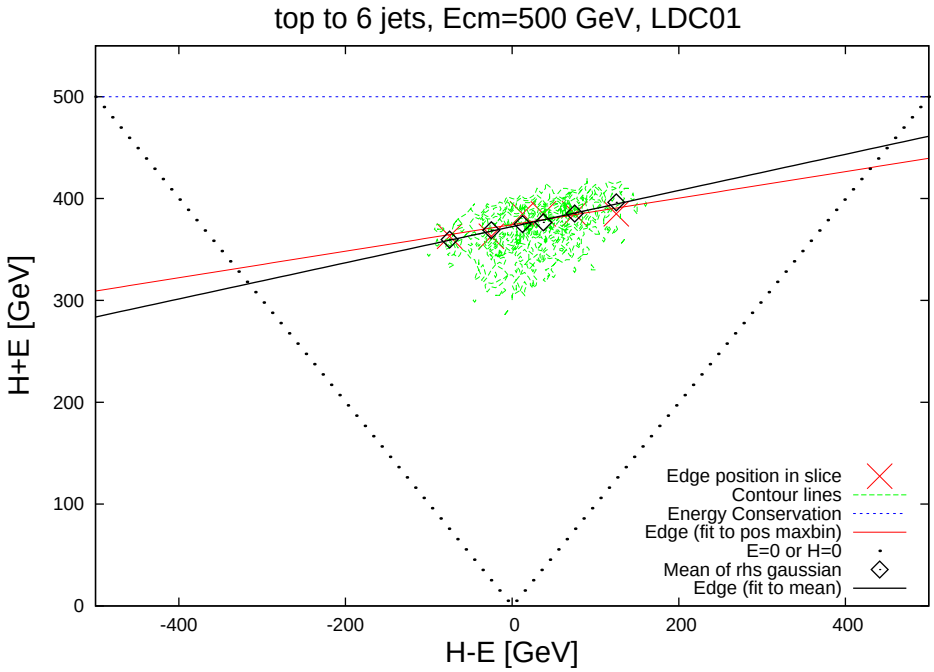


Figure 10: The red line shows the position of the edge determined by the processor. The black line shows the position of the edge determined using the iterated fit procedure.

5 Conclusion and Outlook

The Conclusion is that it is possible to automate the calorimeter calibration using the energy conservation law. A basic makability study has been sucessfully performed. However, lots of improvements and testing as listed below are still missing and have to be done.

- Fist of all, the processor has to be improoved. See section 3.1.
- Performace and stability studies have to be done to understand clearly the precision of the calculated calibration coefficients.
- Recalibration of existing Data samples has to be done.

References

[1] V. L. Morgunov, Calorimeter energy calibration using the energy conservation law, to be published in proceedings of LCWS 2006 or see programm of http://www.tifr.res.in/lcws06

[2] Detector outline document for the Large Detector Concept, http://www.ilcldc.org (2007)

[3] International Linear Collider, http://www.linearcollider.org

[4] Marlin: Modular Analysis and Reconstruction for the Linear Collider, http://ilcsoft.desy.de/marlin

[5] LCIO: Linear Collider Input Output, www.http://ilcsoft.desy.de/portal/software_packages/lcio/index_eng.html

[6] Geant 4, http://geant4.web.cern.ch/geant4

[7] http://en.wikipedia.org/wiki/Calibration

[8] ROOT, http://root.cern.ch

[9] GNU Scientific Library, http://www.gnu.org/software/gsl

[10] gnuplot, http://www.gnuplot.info/documentation.html

A Steering file

```
<!--This is a comment and will not be executed-->
<!--Save this file as steerfile.xml-->
<!--Run Marlin using the command:
 $ ./bin/Marlin steerfile.xml-->
<marlin>

<execute>
<processor name="MyAIDAProcessor"/>
<processor name="MyCalibProcessor"/>
</execute>

<global>
...
</global>

<processor name="MyAIDAProcessor" type="AIDAProcessor">
  <!--Processor that handles AIDA files. Creates on directory per processor.
  Processors only need to create and fill the histograms, clouds
  and tuples. Needs to be the first ActiveProcessor-->
  <!-- compression of output file 0: false >0: true (default) -->
  <parameter name="Compress" type="int">1 </parameter>
  <!-- filename without extension-->
  <parameter name="FileName" type="string"> Output </parameter>
  <!-- type of output file xml (default) or root ( only OpenScientist)-->
  <parameter name="FileType" type="string">root </parameter>
</processor>

<processor name="MyCalibProcessor" type="CalibProcessor">
<!--calculates new calibration coefficients-->
<!--Name of the ECAL hit collection-->
<parameter name="colNameECALHits" type="string">ECAL </parameter>
<!--Name of the HCAL hit collection-->
<parameter name="colNameHCALHits" type="string">HCAL </parameter>

<!-- LDC01Sc -->
<parameter name="CalibrECAL" type="FloatVec"> c_1 c_2 </parameter>
<parameter name="CalibrHCAL" type="FloatVec"> c_3 </parameter>
</processor>

</marlin>
```

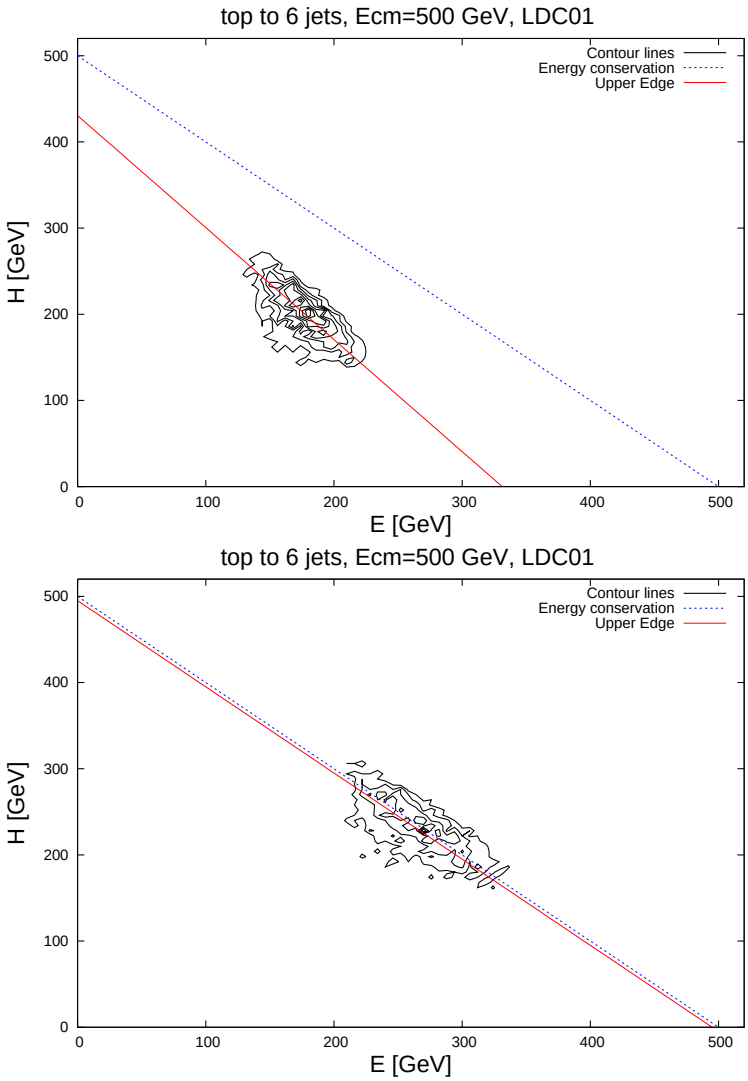


Figure 6: The H versus E representation for a performed recalibration of $t\bar{t}$ to six jet events.

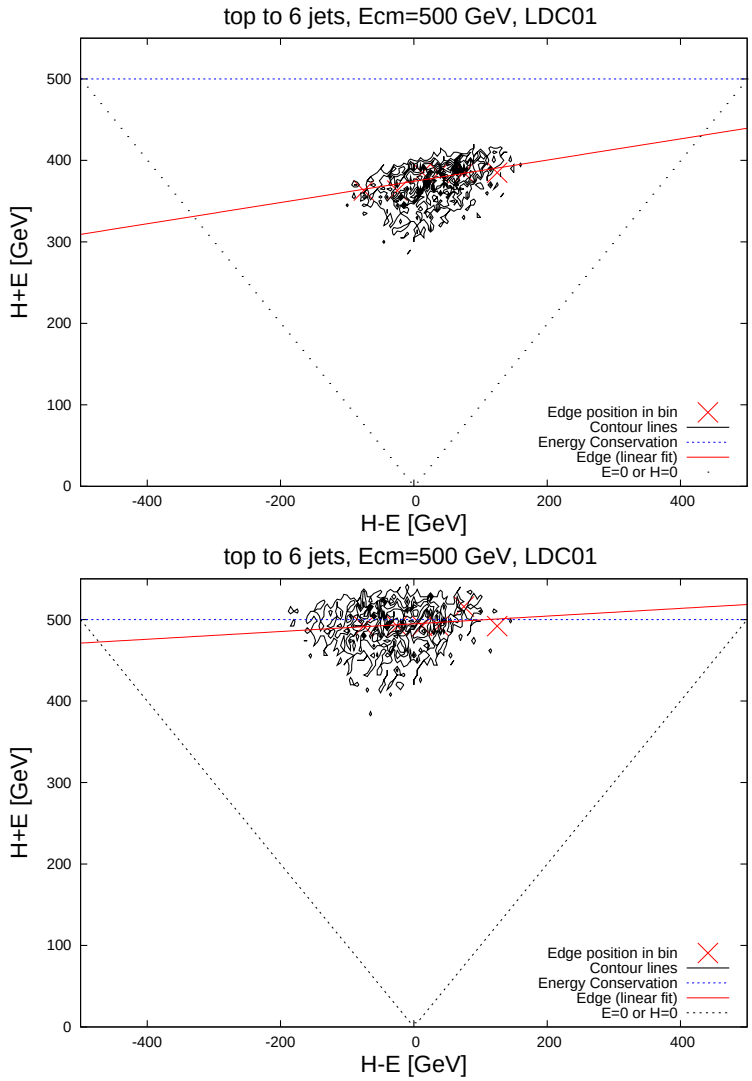


Figure 7: Example for a recalibration of a not perfectly calibrated data set of $t\bar{t}$ to six jets events. The H+E vs. H-E coordinates show the rotated cloud. These plots correspond to the ones in Figure 6as they show the recalibration of the same data set.

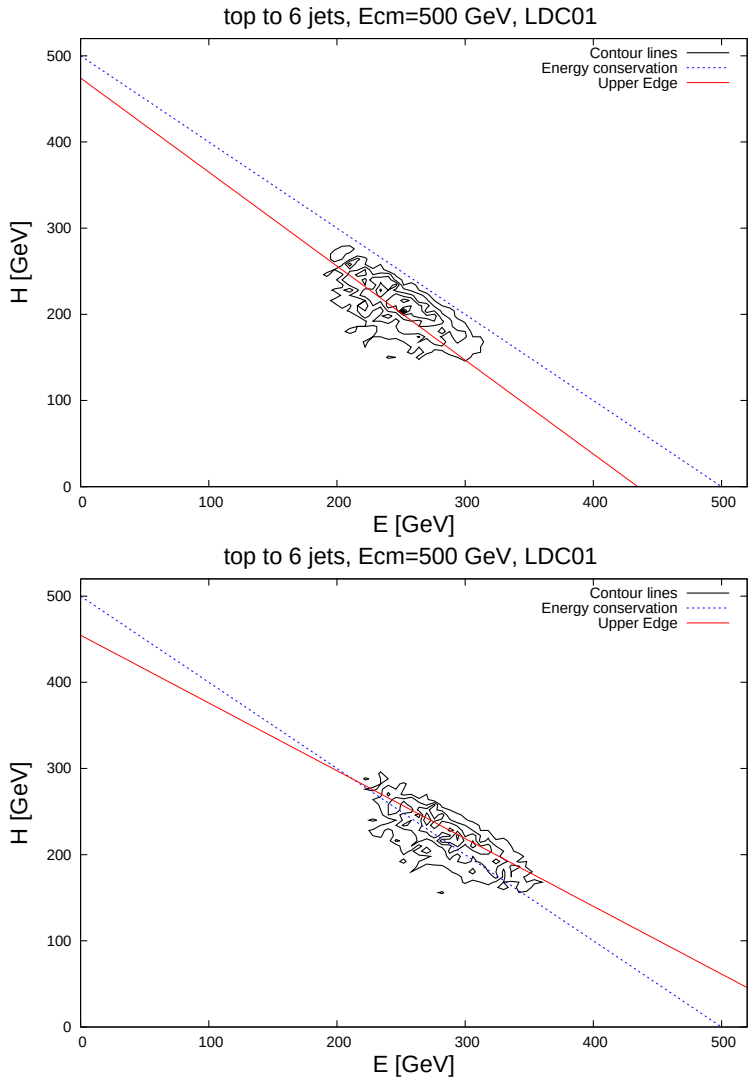


Figure 8: Example for a not well calibrated sample used in 2006 for detector optimisation and PFA studies. $t\bar{t}$ to six jets events are shown in a H versus E plot.

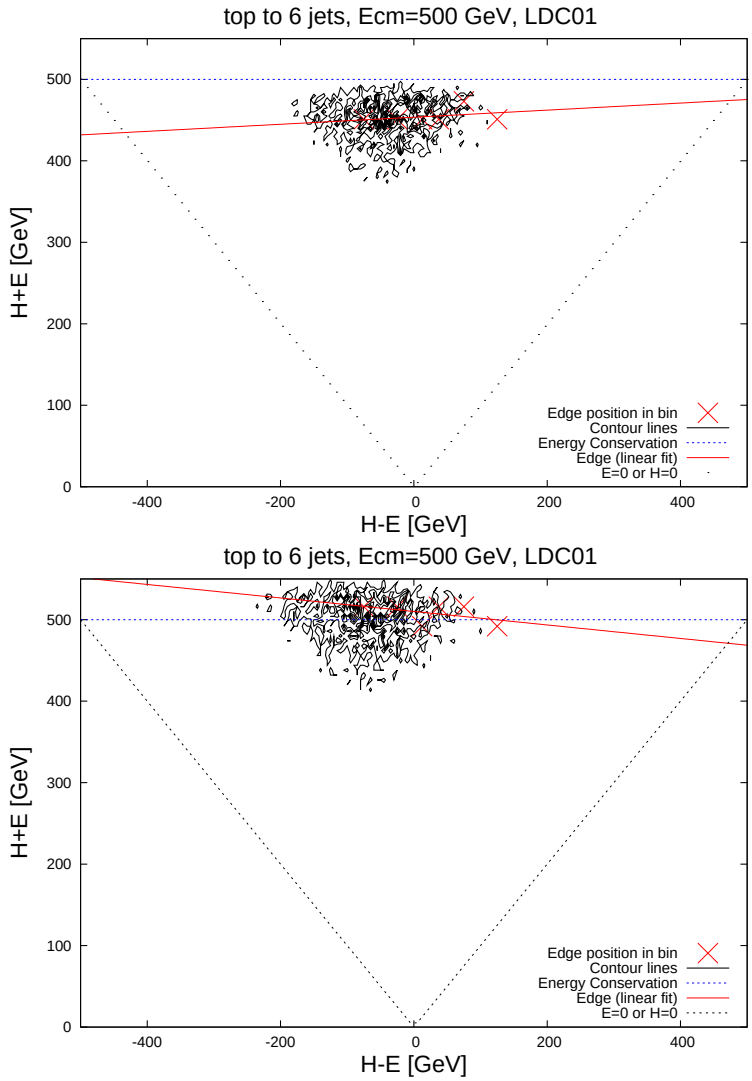


Figure 9: $t\bar{t}$ to six jets events are shown in a H+E versus H-E plot. This is an example for a not well calibrated sample used in 2006 for detector optimisation and PFA studies.These plots correspond to the ones in Figure **??**as they show the recalibration of the same data set.