Kali: The framework for fine calibration of the LHCb Electromagnetic Calorimeter

Ivan Belyaev, Daria Savrina[†]

E-mail: [†]Daria.Savrina@cern.ch
Institute for Theoretical and Experimental Physics, ITEP, Moscow

Ricardo Graciani, Albert Puig[††]

E-mail: [††]Albert.Puig@cern.ch
Universitat de Barcelona

on behalf of the LHCb Collaboration

Abstract. The precise calibration (at a level of below 1%) of the electromagnetic calorimeter (ECAL) of the LHCb experiment is an essential task for the fulfilment of the LHCb physics program. The final step of this task is performed with two calibration methods using the real data from the experimental setup. It is a very CPU-consuming procedure as both methods require processing of $\mathcal{O}(10^8)$ events which must be selected, reconstructed and analyzed.
In this document we present the Kali framework developed within the LHCb software framework, which implements these two final calibration methods. It is integrated with Grid middleware and makes use of parallelism tools, such as python parallel processing modules, to provide an efficient way, both time and disk wise, for the final ECAL calibration.
The results of the fine calibration with the very first data collected by the LHCb experiment will also be presented. With the use of the Kali framework it took only two days of processing and allowed to achieve a calibration accuracy of 2-2.5% for the different ECAL areas.

LHCb-PROC-2011-027
03/05/2011

1. Introduction

The LHCb experiment is one of the particle physics detector experiments operating at the Large Hadron Collider accelerator at CERN. The main goal of the experiment is the precise measurement of the unitarity triangle angles and study of rare CP-violation effects in the decays of beauty particles. Studies of rare decays of charmed particles and exotic decays of the τ-lepton are also planned. A more detailed description of the LHCb physics motivation and detector structure may be found in the LHCb TDR [1], [2], [3].

The electromagnetic calorimeter (ECAL) is one of the four subsystems of the LHCb calorimeter system [4]. It is mainly designed to perform three tasks:

- Measure energies and positions of photons and electrons.
- Provide the level zero trigger with high-transverse momentum electron, photon and hadron candidates.
- Take part in particle identification algorithms performing electron/hadron separation.

The reconstruction of neutral pions and prompt photons with good accuracy is very important for the LHCb physics program.

The ECAL is formed by 6016 cells divided in three different areas, each with its own cell size, and is designed to provide a resolution of

$$\frac{\sigma_E}{E} = \frac{10\%}{\sqrt{E}} \oplus 1\%$$

where E is the particle energy in GeV and σ_E its error. The energy resolution of the ECAL modules has been determined at the test beam to be in agreement with the designed value.

The whole calorimeter as well as each individual cell needs to be calibrated in order to achieve the design resolution. The calibration procedures for the ECAL will be outlined in section 2, paying special attention to the fine calibration. The adopted software solution for this fine calibration, the Kali framework, will be introduced in section 3. With the help of the Kali framework, 2 methods for iterative calibration using π^0 have been developed. Calibration results and conclusions will be shown in section 4.

2. ECAL calibration procedure

Full calibration of the ECAL involves several steps. Initial calibration was set using the photoelectron signal detected from an LED system installed in the calorimeter, providing an intercalibration at the level of 10%. More precise calibration procedures [5] can be performed with the help of the real data from the experimental set. As a general rule, these methods consist in determining a multiplicative energy correction for each cell.

The ECAL is precalibrated with the energy flow method. This technique is based on the idea that the distribution of the transverse energy over the surface of the calorimeter should be a smooth function of the coordinates. Therefore, this calibration procedure consists in attempting to smoothen the distribution of energy integrated over time, allowing to reduce the initial 10% miscalibration to the $\sim 4\%$ level (and applied above any of the fine calibration methods, it may give even less than 2% calibration accuracy) [6].

For further improvement two different calibration procedures have been devised, both based on the measurement of a well known value, namely the mass of the resolved π^0 in its decay into two photons. These procedures, known as the fine calibration, are the *mass distribution method*, based on mass distribution fits, and the *minimization method*, based on minimization of a certain function running over all selected π^0 candidates. The former relies on adjusting the $\pi^0 \to \gamma\gamma$ mass distribution and calculating the calibration constants in order to have the mass peak at its nominal value, 134.9 MeV/c^2. In the latter, instead of working on invariant mass distributions the mass information of all π^0 candidates (all entries in the nTuple) is used. This allows to concentrate on event-by-event variables, instead of distributions.

As the correction constants depend on the quality of the previous calorimeter reconstruction, a number of the re-reconstructions is needed to achieve convergence. That makes both fine calibration methods to have iterative nature. With these two methods used together one may achieve the calibration accuracy of about 1%.

3. The Kali framework

Kali is the framework used for fine calorimeter calibration, and it is mainly useful for iterative calibration scenarios. It includes a C++ Gaudi-based algorithm for selecting and saving neutral pions, a base python module and two sets of python modules which implement the *mass distribution* and *minimization* calibration techniques. However, more calibration methods may be added to this package at any time by inserting the proper selection algorithm and a set of python modules into the framework.

The typical schema for an iterative calibration using Kali can be seen in figure 1. At the first step, common for both methods, data (in DST[1] format) is processed with the Kali selection algorithm, obtaining a fmDST (see section 3.1) and an nTuple. The latter can be processed with any of the two fine calibration methods to obtain calibration constants. The data is then re-reconstructed from the fmDSTs with the same Kali selection algorithm, but applying the calibration coefficients to correct the cell energies. It results in a set of new, corrected, nTuples and fmDSTs. These ones are then reused in an iterative fashion until the method has converged.

Both *mass distribution* and *minimization* calibration techniques need a knowledge of the background events underlying the π^0 mass peak. Combinatorial background – which accounts for almost all background – can be generated when running Kali using the opposite cell information, leaving the π^0 selection phase as follows:

(i) Build signal π^0 particles by building all the two-pair combinations of photons and applying suitable cuts.

(ii) For each photon, create a new photon set built by taking all the rest of the photons and changing their position in the ECAL plane from (x, y) to $(-x, -y)$ (the z-direction corresponds to the beam direction). Combine the photon with these *combinatoric* photons and apply the same cuts as before to create combinatoric π^0's.

(iii) Save signal π^0's and combinatoric π^0's in the nTuple, marking the latter with a background flag.

To make the calibration procedure faster both methods use a technique of parallel processing in python by using PyROOT scripts (based on the TPySelector python extension of the ROOT TSelector). It allows to divide input data between several machines (or several cores of a multi-core machine) and process it with one common algorithm, then merging the output from different workers so that a user receives it as a whole.

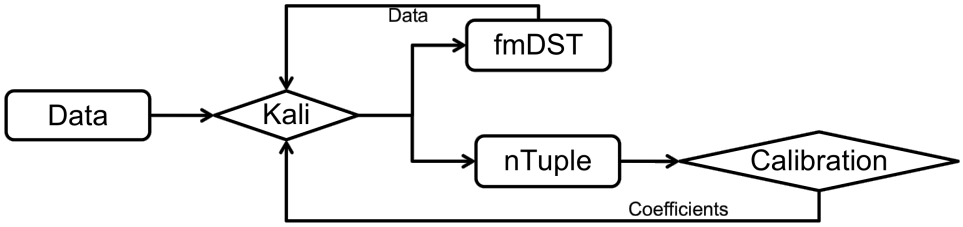

Figure 1. Schema of a typical iterative calibration process using Kali

3.1. The fmDST

The femto-Data Summary Tape (fmDST) is a special format for data storage, generated as output of the Kali selection algorithm. Its structure is the same as that of the typical DST used in LHCb for data storage (and hence it can be used by standard LHCb software), but each event contains only the data needed to perform calorimeter calibration tasks:

- Only relevant tracks, clusters, hits, etc.
- Only information from relevant subdetectors (calorimeter, tracking if needed...).

[1] The DST (Data Summary Tape) format is the ROOT-based data storage format used to to stored reconstructed LHCb data.

The result is a very compact file, ~ 300 bytes/event, ~ 100 times less than usual DSTs.

The fmDST is fully compatible with the standard LHCb calorimeter reconstruction tools, which implies that it can be reused as if it were a normal DST, allowing the application of the calibration constants. Given the simpler nature of the fmDST, this reprocessing is ~ 10 times faster than typical DSTs.

The fmDST was designed for calorimeter calibration scenarios since they involve processing large amounts of data ($\mathcal{O}(10^6)$ events, representing ~ 30 GB) in an iterative fashion. Using a stripped-down data storage produces huge improvements both in storage and processing time.

3.2. Mass distribution fit method

This method relies on fitting the $\pi^0 \to \gamma\gamma$ mass distribution and adjusting the calibration constants of each cell in order to have the mass peak at its nominal value, 134.9 MeV/c^2. Since this needs to be done for 6016 cells, this technique requires filling and fitting a large number of histograms.

The photons are reconstructed as 3x3 neutral clusters in LHCb ECAL. The reconstruction procedure is built in such a way that the central cell of a cluster (called the *seed*-cell) contains most of the cluster energy. This leads to an almost linear dependence between the energy measured in the seed cell and the energy of the reconstructed photon provided that the energy deposition in the preshower detector (PS) is small.

For calibration of each cell from the reconstructed data sample all the $\gamma\gamma$ pairs are selected in which one of the photons has this cell as the seed one. With the event statistics high enough, the neutral pion peak is visible in the invariant mass distribution of these pairs and one is able to calculate the calibration coefficient, assuming that the peak is shifted only because of the miscalibration of the current cell.

In order to avoid interference from the PS miscalibration, one would like to use photons with low energy deposition on the PS. However, such requirement drastically reduces the statistics. Thus, the sample is divided into three subsamples according to the PS energy, and for each of these –and for each cell– a signal and a combinatoric background histogram is filled. If possible, the most favorable sample is used. Otherwise, a sample with more statistics is used.

The peak position is determined by fitting the di-photon invariant mass distribution with a sum of a second order polynomial and a gaussian function. To reduce the uncertainties in the fit parameters the fit is performed through several steps.

(i) First the background histograms are fitted to estimate the initial background fit parameters for the signal histograms.

(ii) Then reference histograms with big number of entries but similiar properties are used to estimate initial values of the parameters (for example a histogram containing the entries from the whole calorimeter zone to which the current cell belongs to).

(iii) The initial parameters are set up to the fit function and the fit of the signal histogram is performed.

(iv) If the fit did not converge, reference and background histograms with a different preshower energy cut, which contain more entries, are used for the estimation of the parameters. After that the previous three steps are repeated.

An average of six fits per cell are performed. Several iterations are needed to achieve convergency, defined by the stabilization of the position of the mass peak around the nominal value (four of five iterations are usually enough). In the end, for a full calibration procedure 2×10^5 histograms are filled and over 10^5 several-step fits are performed.

The calibration procedure has been performed in the LHCb-CAF[2], using seven out of eight cores in the worker node. Processing speed scales almost linearly with the number of cores used. Having a dedicated machine for this time-consuming calibration greatly simplifies the procedure, and in this sense the usage of the CAF has proven to be very useful. For one iteration on 10^6 of events, processing takes ~ 2.5 hours with seven cores, where ~ 1.5h are needed for filling the histograms and ~ 1h is used by the fitting procedure.

3.3. Minimization method

In this technique the mass information of individual π^0 candidates (all entries in the nTuple) is used. This allows to concentrate on event-by-event variables, instead of distributions. Clearly, to use this method a sample that reproduces the observed background is needed so it can be subtracted from the signal data, which includes mixed signal and background information.

From the nTuple containing both data and background candidates, correction constants are calculated minimizing the following function:

$$f = \sum_{data} w_i \delta m_i(c_{i,1}, c_{i,2}) - \frac{1}{2} \sum_{bkg} w_j \delta m_j(c_{j,1}, c_{j,2})$$

where

- m_i are the π^0 candidates masses, δm_i the difference of these candidates wrt to the nominal π^0 mass and the $c_{i,1-2}$ are the two coefficients corresponding to the i-th candidate.
- w_i are the weights used in order to control the influence of those candidates farther from the mass peak. A gaussian function with δm_i as the independent variable emphasizes the reduction of the mass peak's width.
- the first summation runs over all candidates in the ntuple, and the second sum only on those marked as background in order to subtract the combinatorics.

The minimization procedure is performed independently for each pair of opposite cells. Usage of opposite cell pairs is due to the fact that signal π^0 candidates are calculated avoiding double-counting[3] so both cells are needed for a complete description of the signal and the background, with the needed 1/2 factor in the function f. This implies that 3008 minimizations have to be performed for each calibration *step*. In order to do so, the nTuples which contain all candidates are cut in smaller nTuples containing only information about each pair of cells. The reason for the separation of cell pairs is three-fold:

- Convergence of a fit containing 6016 parameters is very difficult and slow. An equivalent result can be achieved by reconstructing the data with the full set of new constants and iterating the procedure.
- Having 3008 separate nTuple files allows for a faster computation of f because the loop over the candidates is much smaller.
- Parallelization is much easier since each pair of cells is completely independent of the others.

Several steps (usually ~ 10, depending on the convergence criteria) need to be performed per iteration, after which the sample needs to be re-reconstructed again. Around 4-5 iterations are needed for full convergence (stabilization of the mass peak).

As a result, ~ 150k MINUIT minimizations (running over all signal and background entries of the Kali nTuple) need to be performed. For a typical sample of 10^6 events, using a 2.33GHz

[2] The LHCb-CAF (Calibration and Alignment Facility) is a dedicated cluster providing computational resources for calibration: four worker nodes with eight cores each.

[3] The summation over the signal photons is of the type $\sum_{i=0}^{photons} \sum_{j>i}^{photons}$ and not $\sum_{i=0}^{photons} \sum_{j\neq i}^{photons}$, while obviously this cannot be done for the background.

Intel Xeon machine each of these MINUIT minimizations takes about ten seconds. Again, parallelization is called for, and in this case two methods have been used:

- *Parallelization via python subprocess.* Each step takes $\sim$ 1 hour, so full iteration needs $\sim$ 10 hours. Speed scales almost linearly with the number of spawned processes up to the number of cores in the machine.
- *Using local cluster.* Send up to 200 parallel jobs, but one needs to be careful that these jobs are not too short, otherwise initialization takes a substantial processing time.

4. Results and conclusions

With the arrival of the first data, and after the precalibration of the whole calorimeter system, the first fine calibration of the LHCb ECAL was performed at the end of May 2010. The sample was 80M events, and the whole process took up to two days. The two methods provided very good agreement, with an estimated intercalibration level of $\sim$ 2%. An example of the peak position improvement under calibration is shown in figure 2. The final result with the all data available by the end of June 2010 is shown in figure 2.

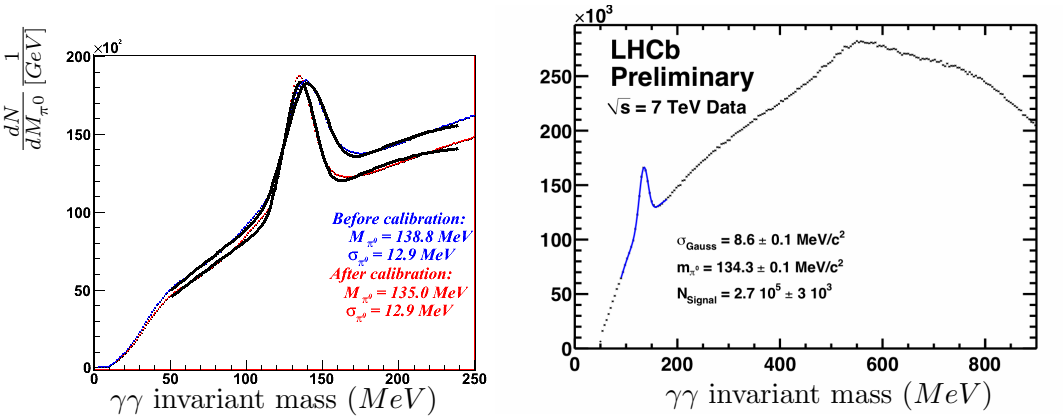

Figure 2. Left: view of the π^0 peak position before (blue line) and after (red line) the calibration, right: $\pi^0 \to \gamma\gamma$ mass distribution plot after the first fine calibration

For this procedure Kali has proven to be very useful, allowing fast reprocessing of data while keeping storage demands low. It has become the standard calorimeter fine calibration tool, used now for π^0 calibration but easily extendable to other types of fine calibration with similar needs.

Acknowledgments

The authors would like to say many thanks to the LHCb calorimeter group for fruitful discussions. A. Puig acknowledges financial support by the MICINN FPU grant.

References

[1] The LHCb Collaboration. *LHCb technical proposal.* CERN/LHCC/98-4
[2] The LHCb Collaboration. *LHCb TDR: reoptimized detector.* CERN-LHCC-2003-030
[3] The LHCb Collaboration. *The LHCb Detector at the LHC*, 2008 JINST 3 S08005
[4] The LHCb Collaboration. *LHCb calorimeters TDR.* CERN-LHCC-2000-036
[5] Korolko I., Obraztsov V., Popescu S. and Yushchenko O. *On the possibility of in situ calibration of LHCb calorimeters.* Preprint LHCb-2000-051
[6] K. Voronchev and I. Belyaev, *Energy flow calibration of LHCb ECAL*, CERNLHCb-2006-051
[7] Rene Brun et al. *The ROOT User's guide 5.27*