Deep Analysis of Hadronic Shower

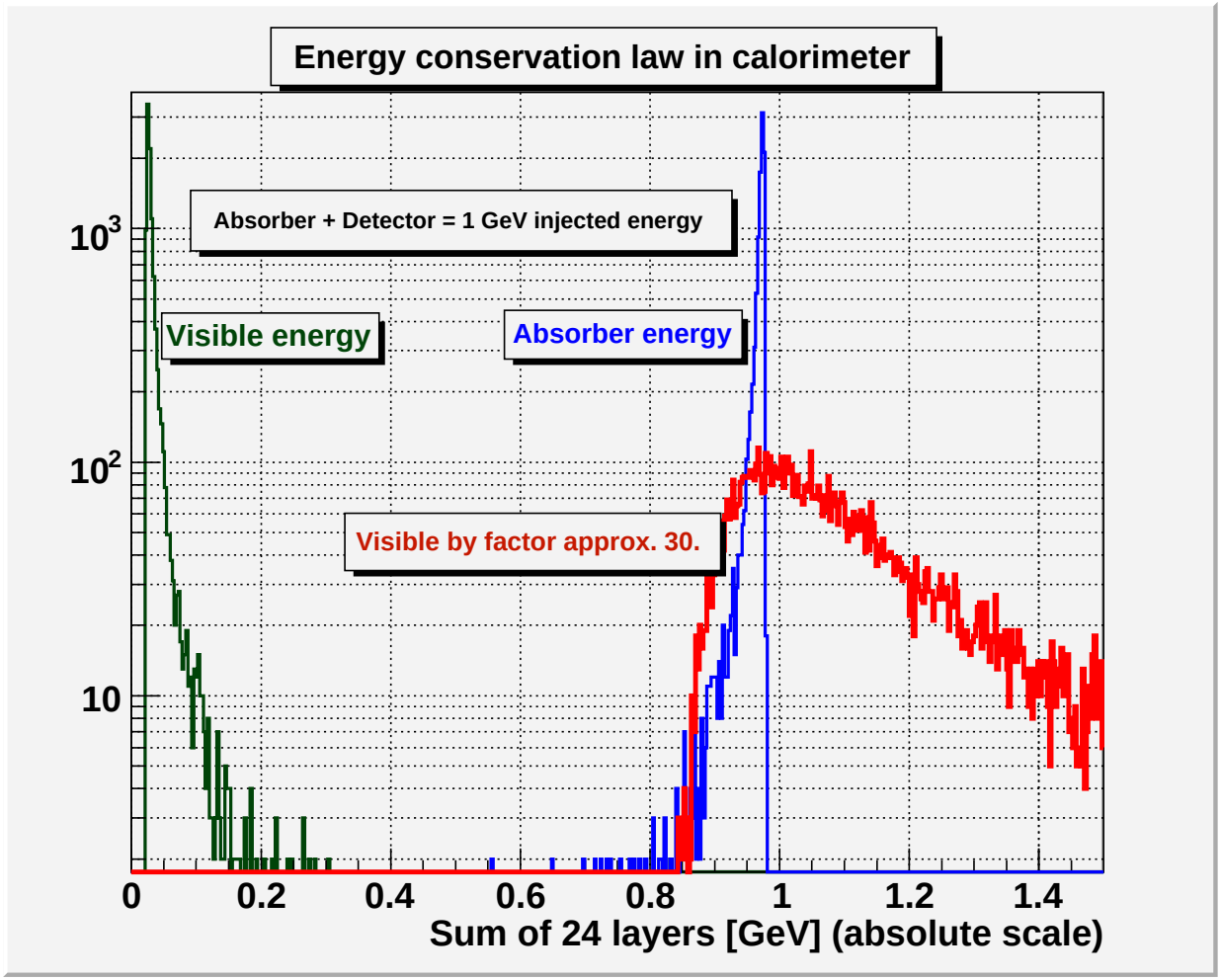
V.L. Morgunov and A. Zhelezov

DESY – ITEP

Paris, May 2007

The copy of this talk one can find at the http://www.desy.de/˜morgunov

Introduction: Muon track in toy MC model



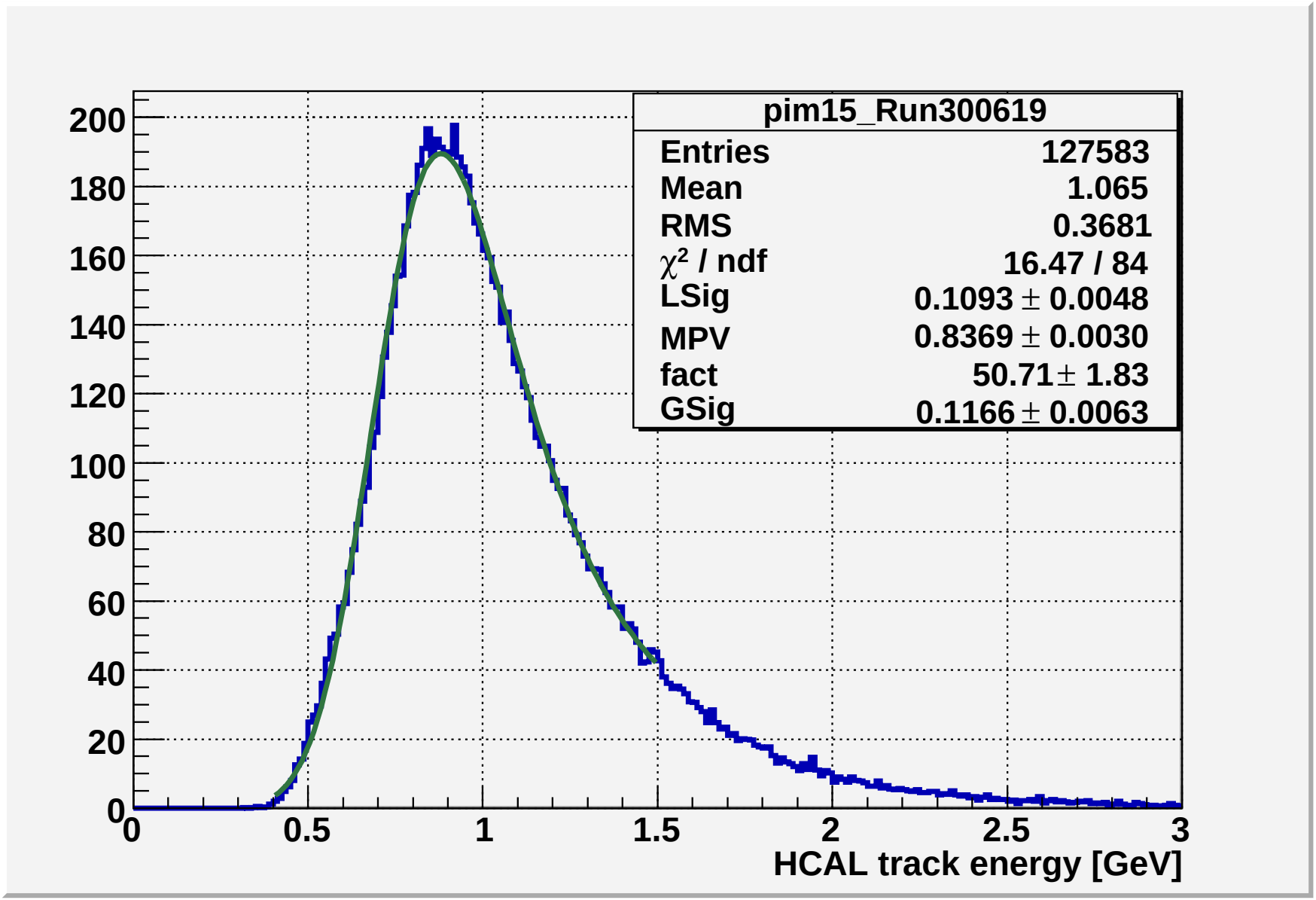
Whole injected energy distributes between absorber and scintillator (visible). To get a detector response in the physical scale one should multiply measured energy by so called sampling factor that is the ratio of whole deposited energy to visible one.

Then one get a response function of the detector. (in red)

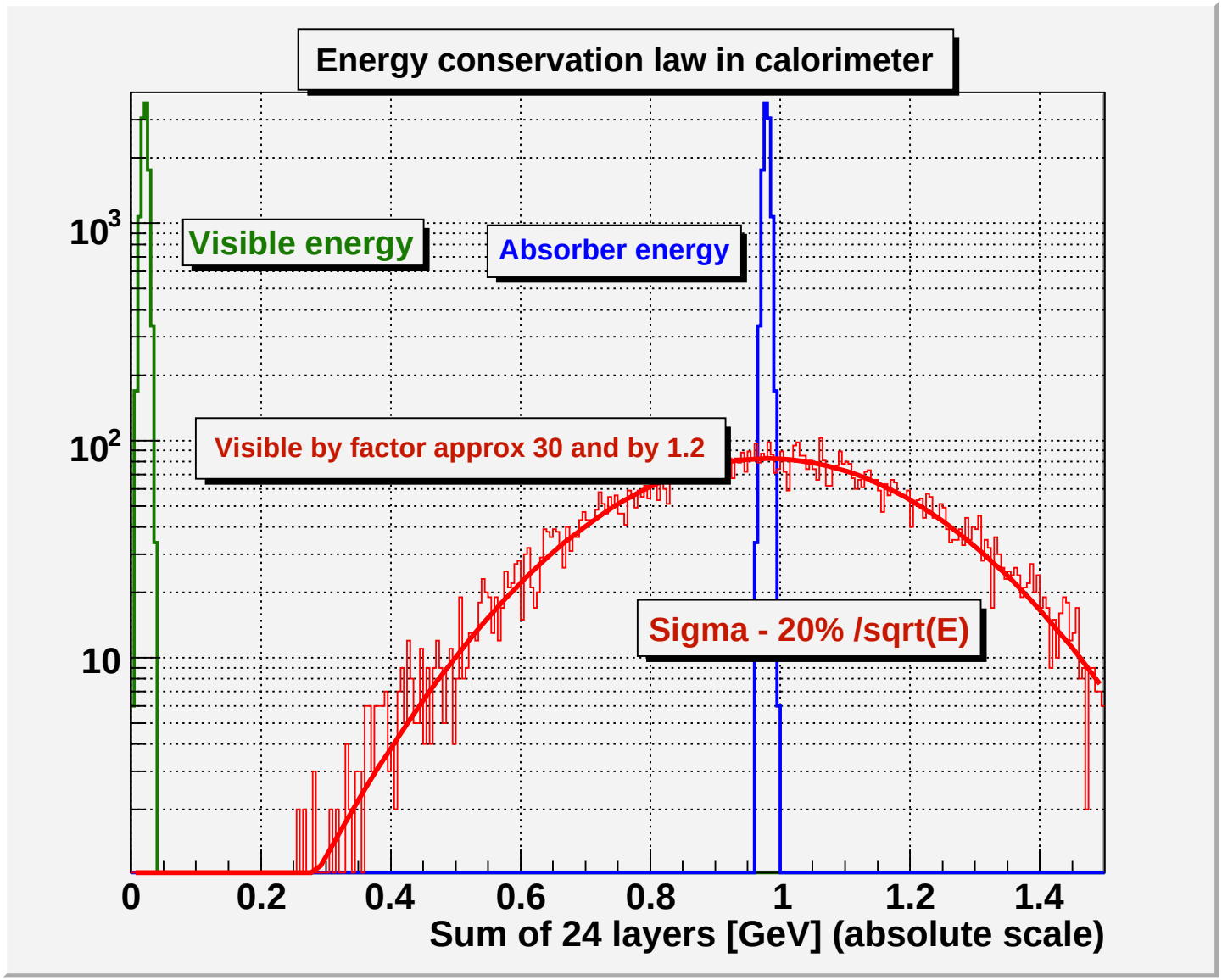
Remark: If one carefully look at the both energy curves, one can see than the visible energy multiplied by some factor does not reflect of correct shape of main energy deposition.

This particular response function is asymmetrical, with incorrect position of tail, and this is due to Landau fluctuations in the thin detector layers; and as well as due to the energy conservation law!

Continue: One track in HCAL

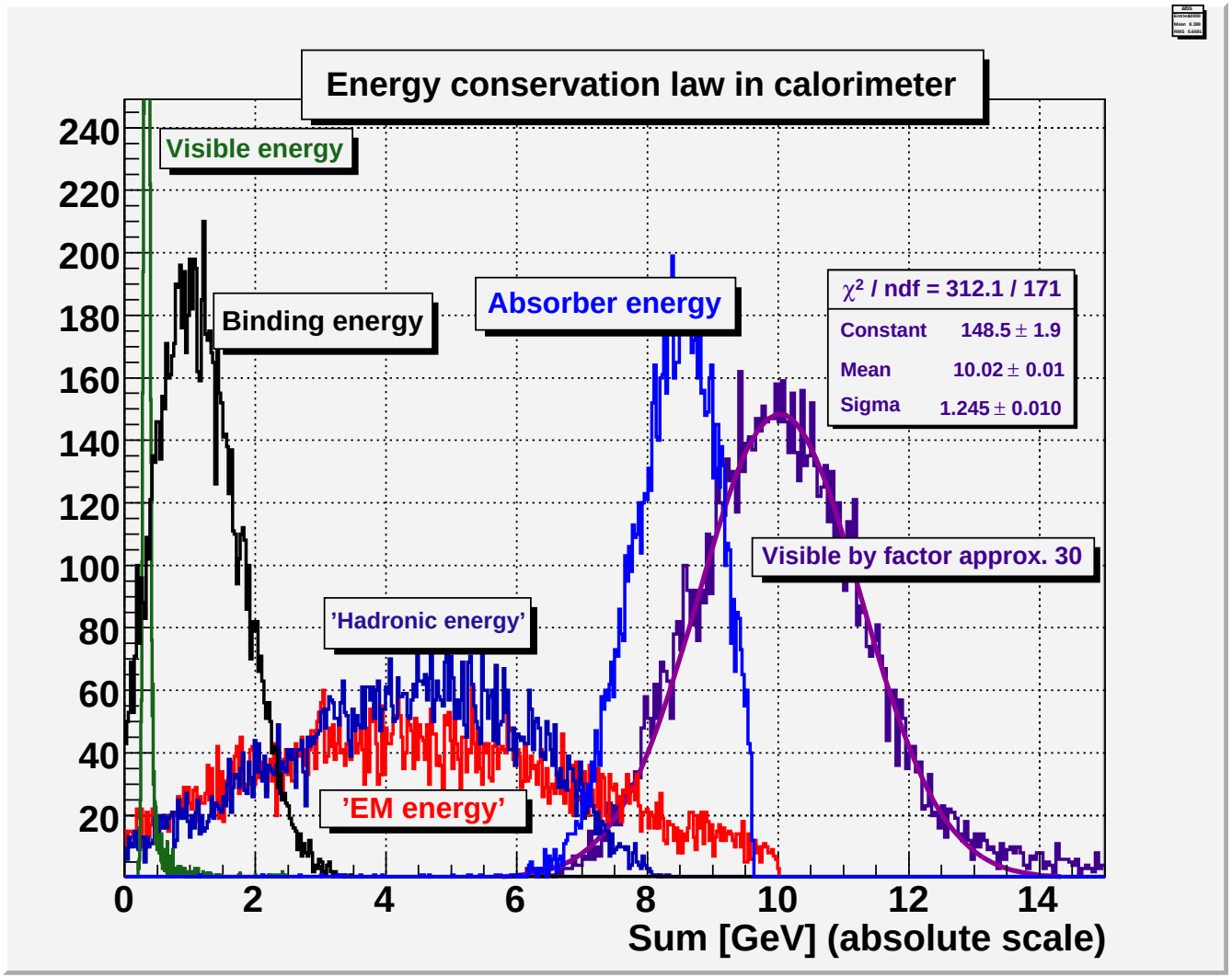


Introduction: EM shower in toy MC model



Sampling factor leads to sampling fluctuations. When a small portion of deposited energy is multiplied by rather big factor (≈ 30 in the case of HCAL prototype), then the fluctuation become to be not negligible.

Introduction: Hadronic shower in toy MC model



The case of hadronic cascade is much more complex.

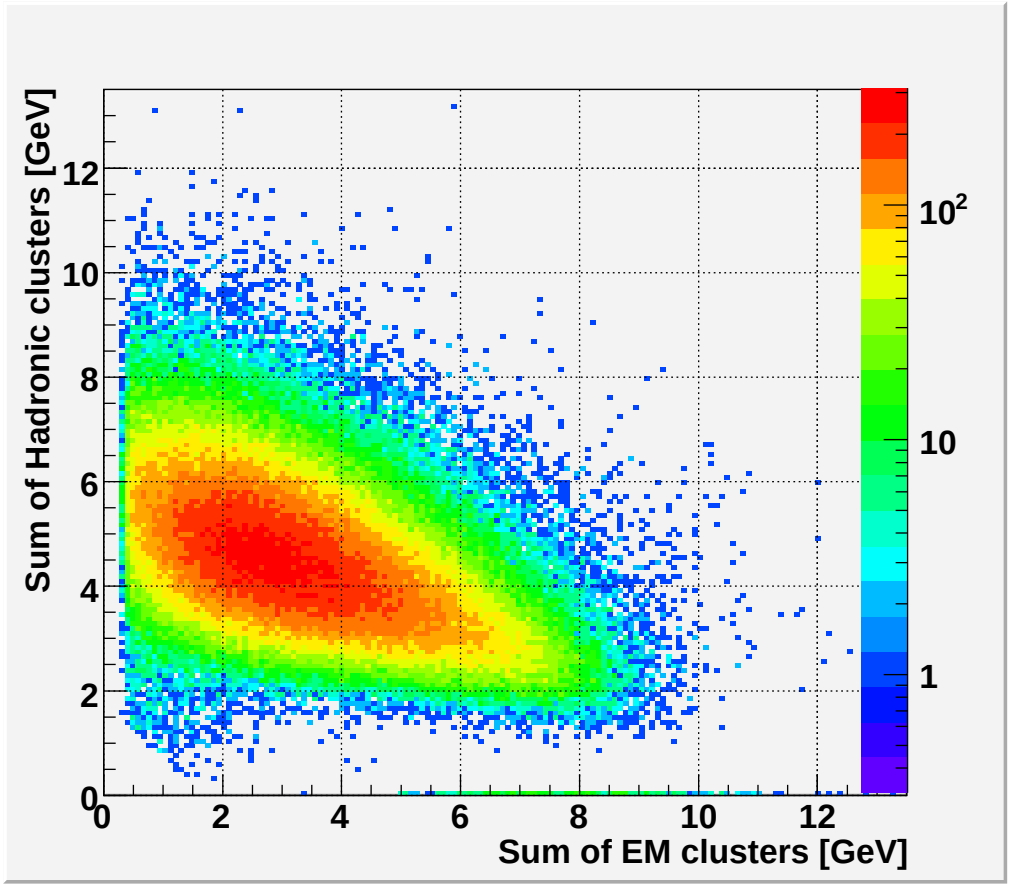
In addition to sampling fluctuation, injected energy is distributed between three different types:

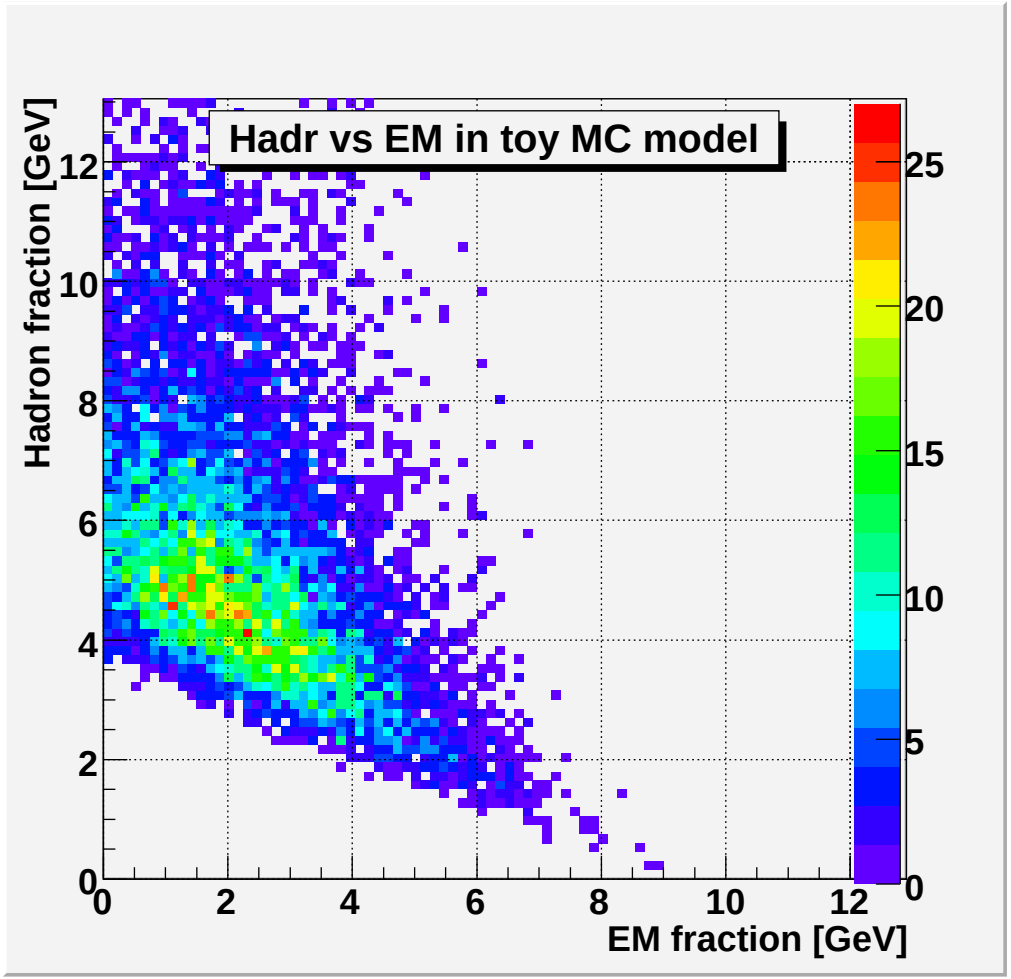
1. **Track–like or hadronic energy**
2. **Electromagnetic-like energy**
3. **So called binding energy, that is fully invisible portion of shower energy.**

And on top of all of these, we have electron to hadron detector response ratio.

Shower decomposition into three types of energy deposition might leads to better understanding of the physics of hadron calorimeters.

Continue: Pions in HCAL





High granularity calorimeter may allow to split hadronic shower signals into three types using the topology and hit amplitude spectrum.

History

This multi-purposes algorithm make a decomposition of hadronic shower (set of calorimeter hits) into number of clusters that are differ with it's physical properties.

FORTRAN code was written at the beginning of 2003 with aim of design of hadron calorimeter prototype.

- **It was first applied to define** *"Test-Beam Prototype Volume Coverage and Clusters in It"* **(2003)**
- **Then it was applied to** *"Tile Size Issues for HCAL Prototype"* **and** *"Prototype Geometry influence on Reconstruction Quality"* **and as well as for the estimation of effect of so–called software compensation. (2003)**
- **Latter on, it was applied for** *"Attempt to estimate the prediction power of the calorimeter properties following from different hadron models in the simulation programs; and the systematic errors of such predictions"* **(2003)**
- **And at last it was applied for** *"Two Particle Separation with Tile HCAL"* **by A. Raspereza. (2004)**

One can see: www.desy.de/˜morgunov talks about all of these applications.

- **Recently algorithm was rewritten into C++ by A. Zhelezov.**

That was done by two steps: code was rewritten first into C to reproduce the exact output numbers for each event with computer accuracy, and then into C++ to get a compact C++ Class, easy to use.

Utilities and applications

- **Eigenvalue solver for $N \times N$ symmetric matrix. (C++ version of CERNLIB code)**
- **2–D histograming with multi–peak search.**
- **2–D clustering algorithm on sphere or spherical shell.**
- **3–D tree search algorithm (uses all above)**
- **Relation database for C++ objects with any number of branches, levels and possible cross–links from any level to any level from any branch to any branch. (by A. Zhelezov)**

Possible applications are:

- **Hadronic shower analysis of CALICE HCAL data and software compensation of shower energy.**
- **As a sensitive tool for tuning of simulation programs.**
- **Hadronic shower splitting in particle flow analysis of LDC calorimeter.**

Class Deep Analysis

Technically it is written as one Class DeepAnalysis

#include “DeepAnalysis.hh” **#include “cernlib_utils.hh”**

What does it doing? **It splits of one hadronic shower into 4 different types of clusters.**

The types are:

Electromagnetic–like; Track–like; “Hadron–like” “Neutron–like”

All these types are represented as set of clusters. Each cluster has many parameters.

Deep Analysis does not change amplitude of any hit.

Now Deep Analysis code is tuned to the 1 cubic meter hadron calorimeter prototype.

It can not work for ECAL shower analysis without re–tuning.

DeepAnalysis code

```
void reconstruction(){  // Main routine

  cluster_finder_3d(*(kind[EM]),false);
  cluster_join(true);
  em_shower_find();
  reassignment_small_clusters(TRK);

  cluster_finder_3d(*(kind[TRK]),true);
  cluster_join(false);
  reassignment_small_clusters(HAD);

  detector.ext_cell_size = 5; // for HCAL prototype  only
  cluster_finder_3d(*(kind[HAD]),true);
  cluster_join(false);
  low_e_clean();

 for(RSIterator<Cluster> ci(clusters);ci.next();)
    ci->finalize(*this);
```

Using of Deep Analysis class

```
  DeepAnalysis deep_analysis; // Create object
  deep_analysis.detector.normal_thresh = threshold_1;
  deep_analysis.detector.neut_thresh   = threshold_1 + 0.1;
//  Fill an object with hits and predefine its type
  for(int i=ehit_count;i<hhit_count;i++){
    DeepAnalysis::KIND type = DeepAnalysis::KIND_COUNT;
    if(ampl_m[i] < threshold_0) // Cut at 0.5 MIP
      continue;
    else if(ampl_m[i] < threshold_2) // Cut at about 2 MIP
      type = DeepAnalysis::TRK;
    else if(ampl_m[i] < threshold_3)  // Cut at about 4 MIP
      type = DeepAnalysis::HAD;
    else
      type = DeepAnalysis::EM;
//  Occasionaly: it is helpfull to convert hits  HAD->TRK, for CALICE only
    if(type == DeepAnalysis::HAD)
      type = DeepAnalysis::TRK;
    deep_analysis.add_hit(xx0[i]/10.,yy0[i]/10.,zz0[i], // in centimeters
       ampl_g[i],ampl_m[i],lay0[i],type); // both  in GeV and in MIP units
  }  // end of hits loop
//    Call of main analysis function
  deep_analysis.reconstruction();
```

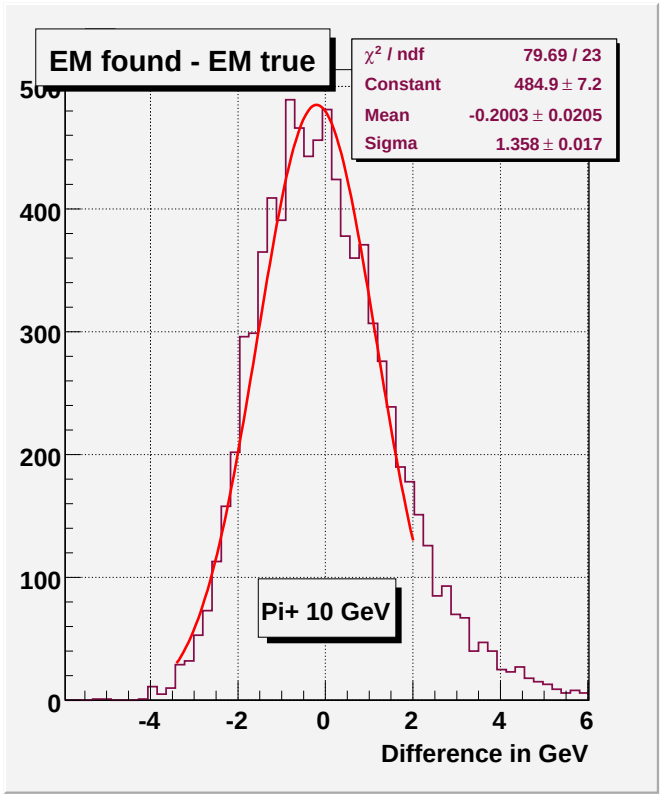
Extracting results

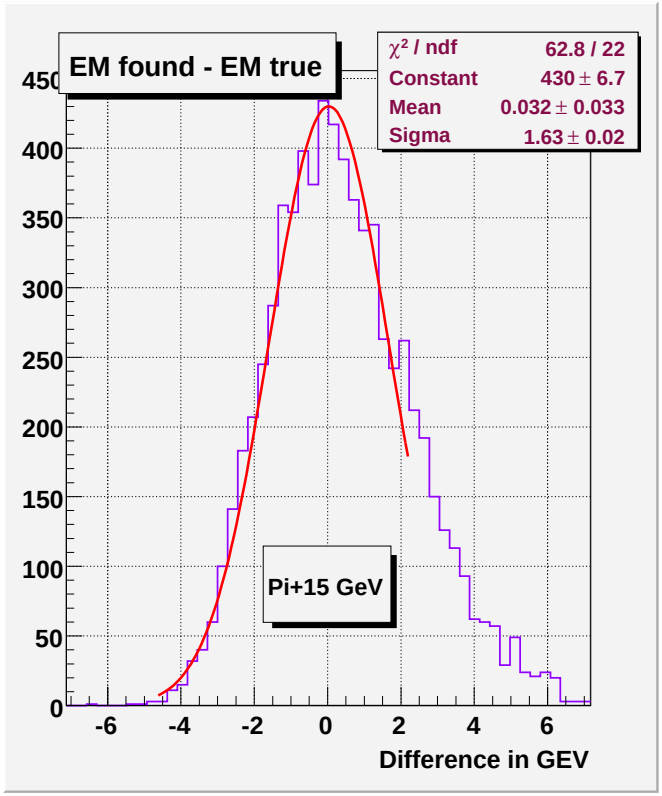
```
//    Get results of Deep Analysis
  unsigned ec_n_cl=ec_hits=0;
  double ec_sum=tc_sum=hc_sum=0.;
  deep_analysis.color_clusters_stat(DeepAnalysis::EM,ec_n_cl,ec_hits,ec_sum);

  unsigned tc_n_cl=tc_hits=0;
  deep_analysis.color_clusters_stat(DeepAnalysis::TRK,tc_n_cl,tc_hits,tc_sum);

  unsigned hc_n_cl=hc_hits=0;
  deep_analysis.color_clusters_stat(DeepAnalysis::HAD,hc_n_cl,hc_hits,hc_sum);

  unsigned nc_hits=0;
  double nc_sum=0.;
  for(RSIterator<DeepAnalysis::Hit> hi(deep_analysis.neutrons);hi.next();)
    nc_sum += hi->am;
  if(deep_analysis.neutrons->getNumberOf<DeepAnalysis::Hit>())
    nc_hits=deep_analysis.neutrons->getNumberOf<DeepAnalysis::Hit>();

  double ae_sum=0.;
  for(RSIterator<DeepAnalysis::Cluster> ci(deep_analysis.clusters);ci.next();)
    ae_sum += ci->ws;
```

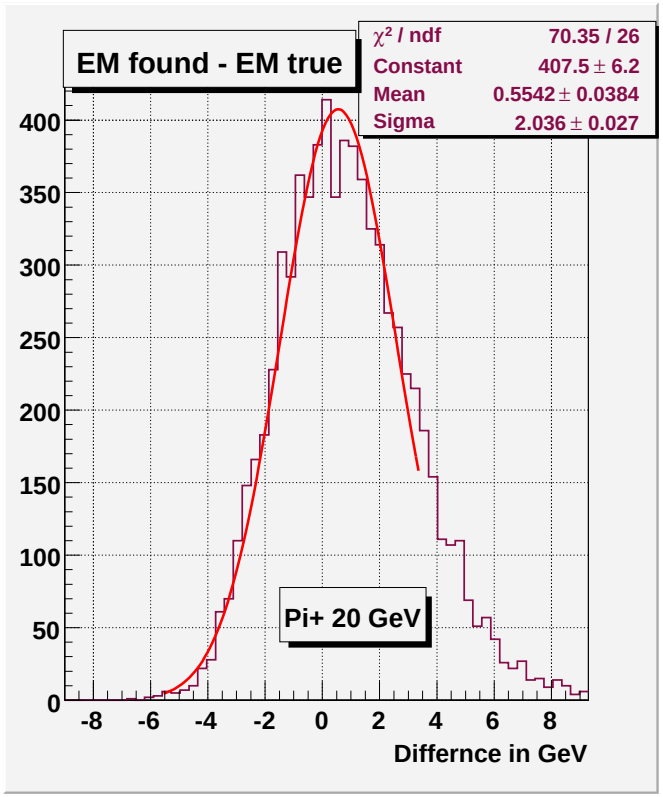
Quality control by Monte-Carlo, thanks Marius Groll

True electromagnetic (EM) energy for cascade is a sum of energies of π^0s and γ^s, that were created in hadronic interactions during cascade development.

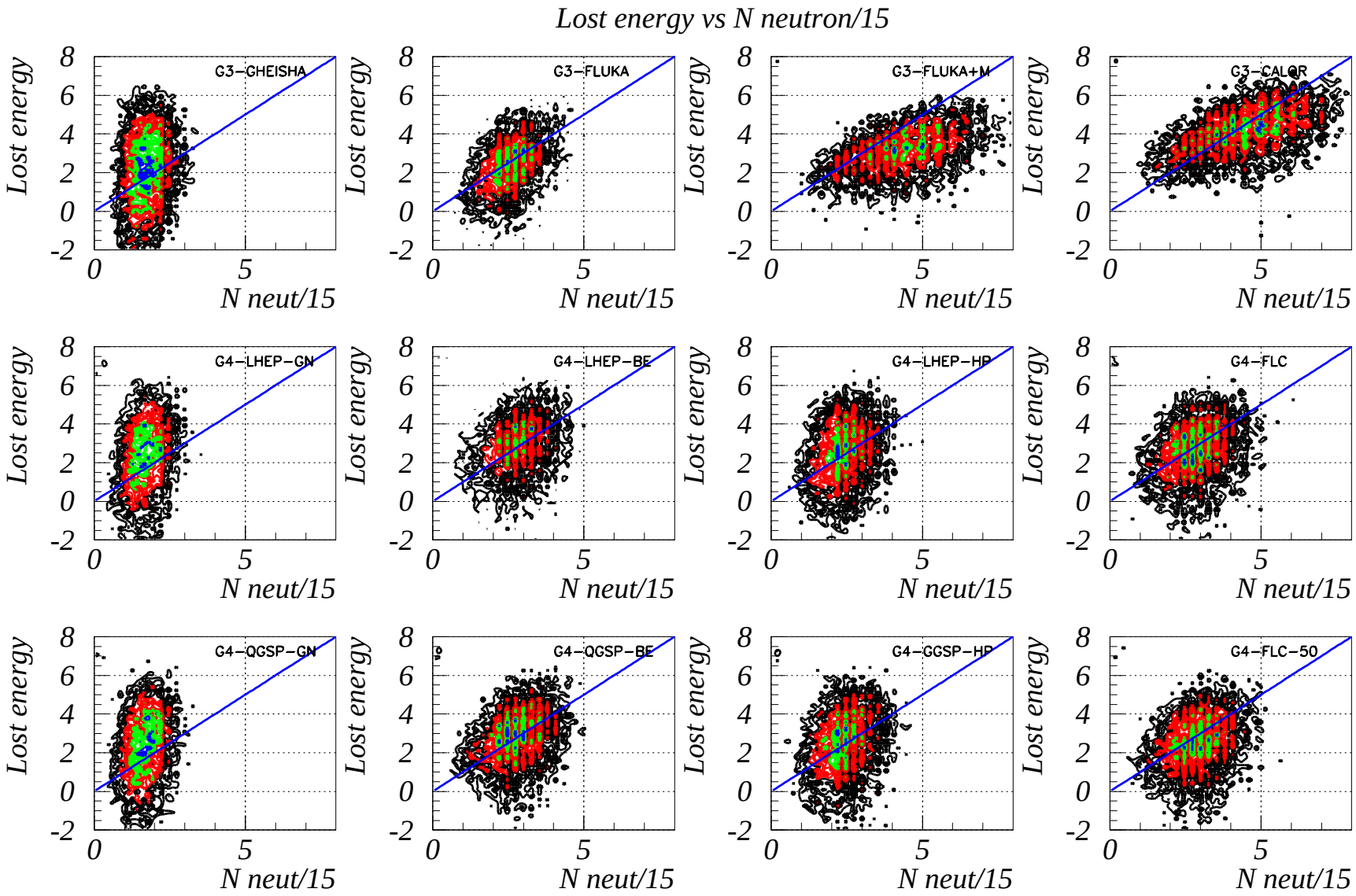
True EM energy compares with energy found as electromagnetic energy by DeepAnalysis algorithm.







Old Comparison of Monte-Carlo programs

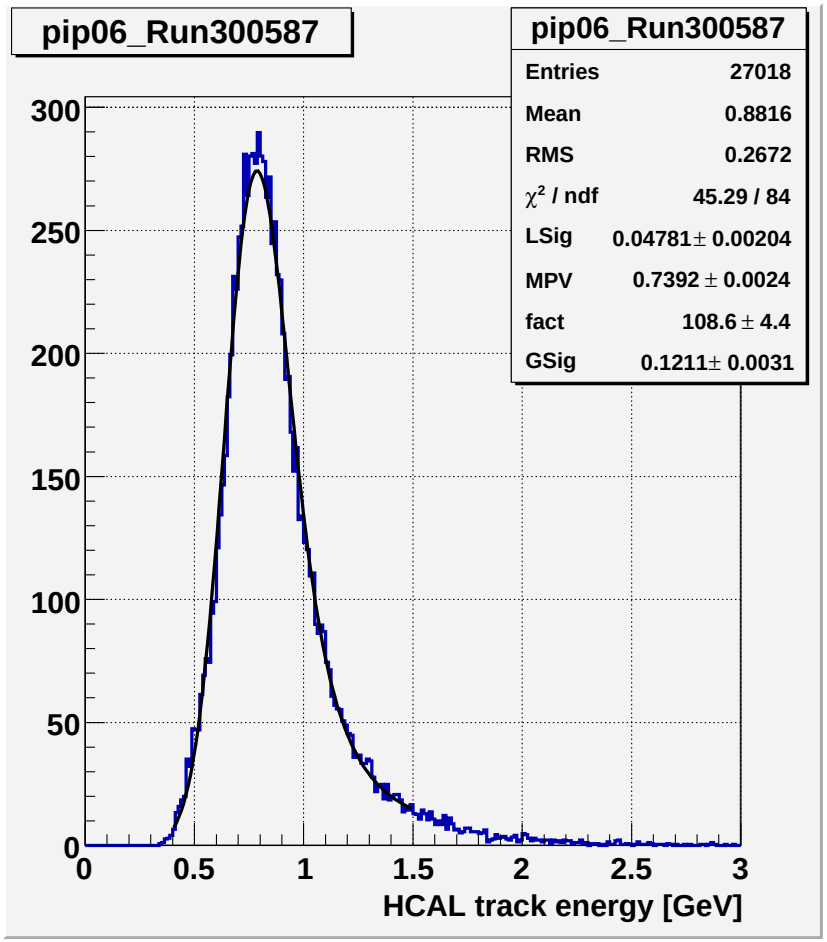


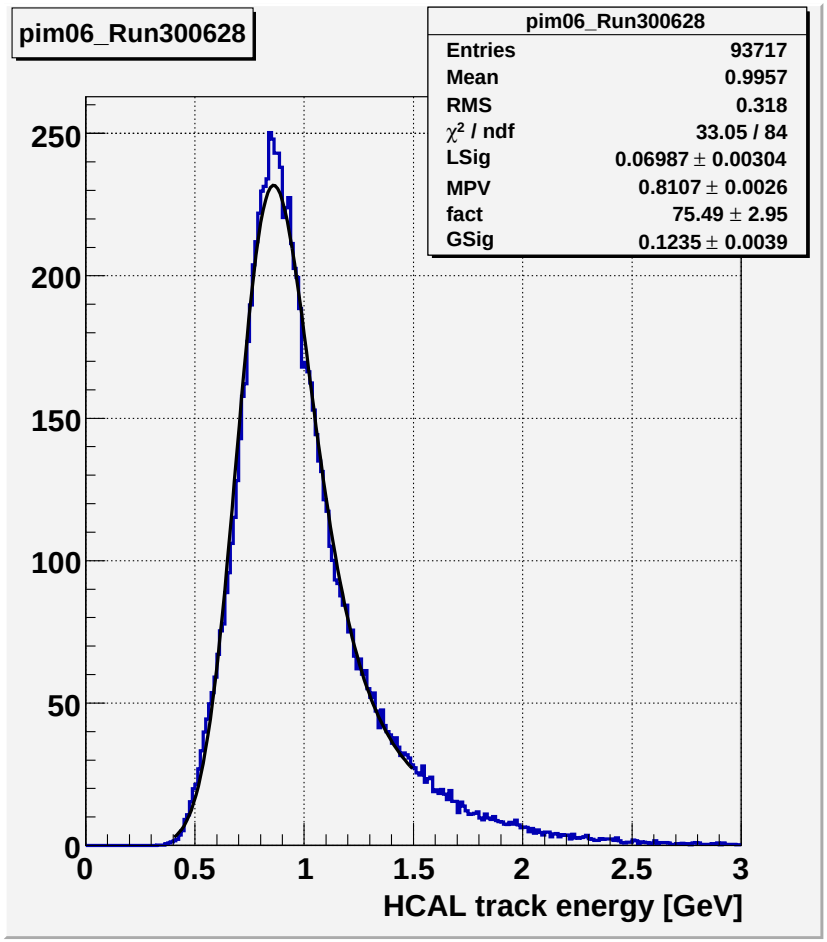
Application for CALICE

It is easy to get a track after DeepAnalysis for each event, if exists.

$$(T\_n\_hit > 15 \&\& T\_n\_hit < 30) \;\&\&\; T\_n\_cl < 3$$

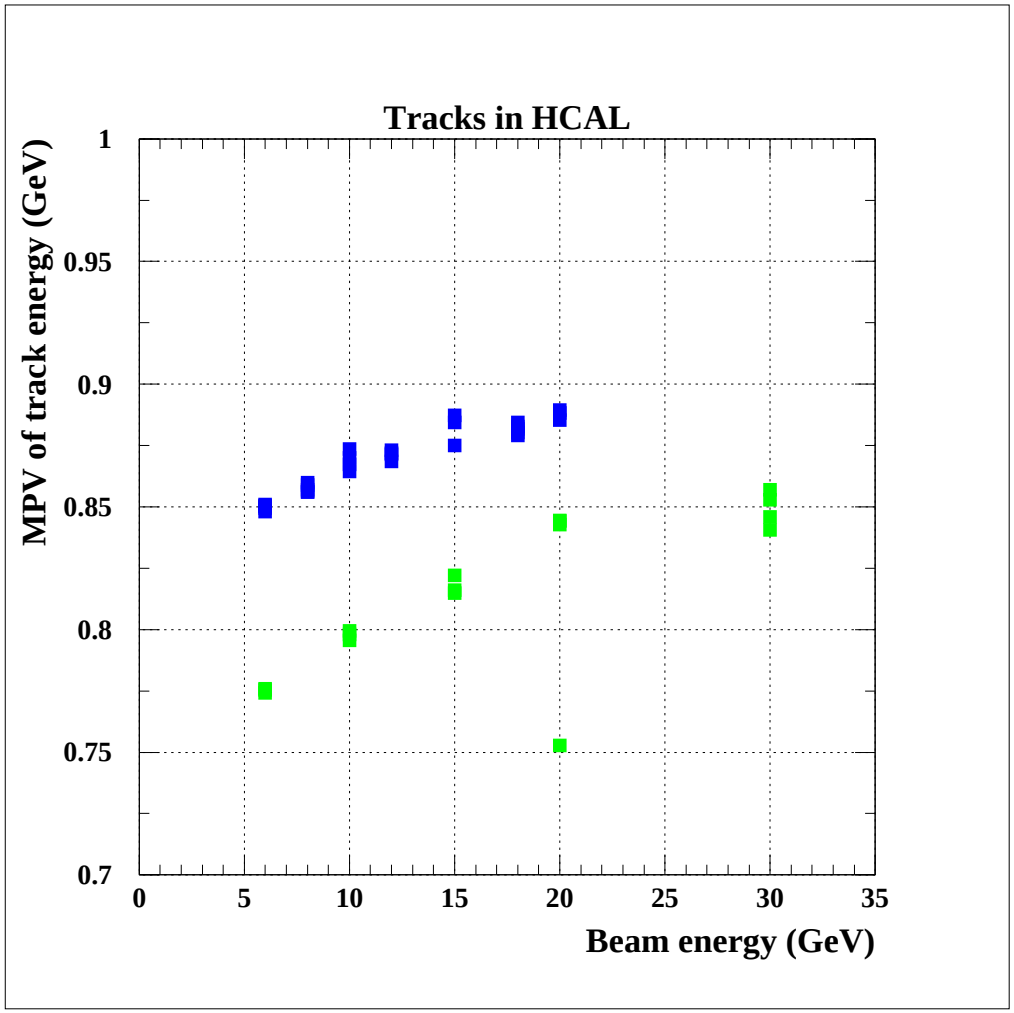
Sum up of energy along one track in HCAL; and fit it by Landau⊕Gaussian function

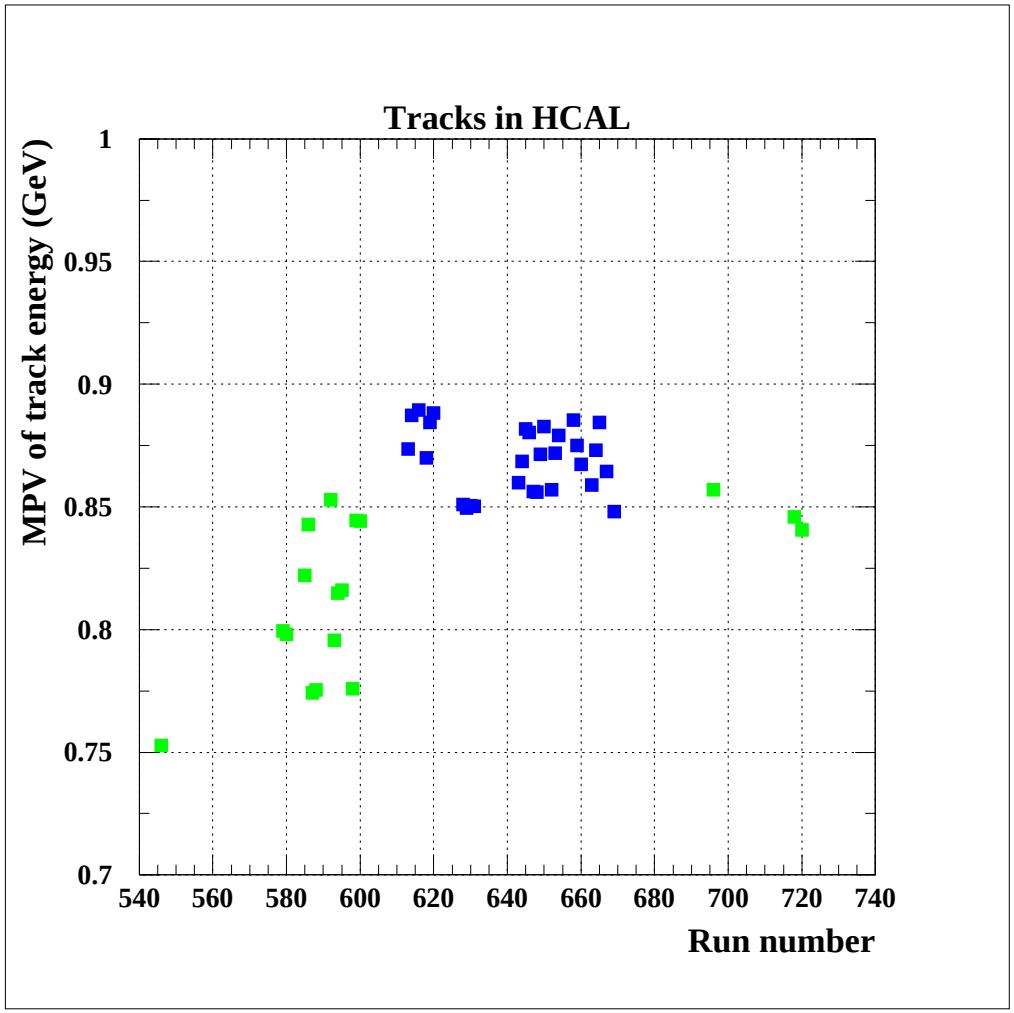




Application for CALICE

HCAL response stability control by tracks found in each run





green π^+, blue π^- beam

46 CERN October runs

Appendix

Hadronic cascade toy MC model code can be taken from:

http://www.desy.de/~morgunov/soft