Validation of single-particle test samples with SDHCal and comparison with AHCaL

ILD software & analysis meeting

C. Carrillo for the SDHCAL collaboration

GOBIERNO DE ESPAÑA
MINISTERIO DE CIENCIA E INNOVACIÓN

Ciemat
Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas

cfp
CIEMAT
física de partículas

First look at the datasets for the SDHCAL validation and AHCAL comparison

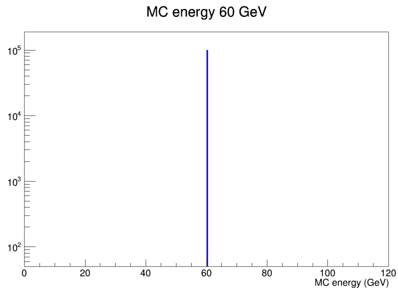
- Details about the ILD confluence production for the test production with the latest ilcsoft v02-01. `https://confluence.desy.de/display/ILD/Production+with+v02-01`
- The data (mostly single particles) are reconstructed with the AHCAL (scintillator) option ILD-l5-o1-v02 and few with SDHCal option ILD-l5-o2-v02.
- For the moment We have access to the high level objects in this dataset.

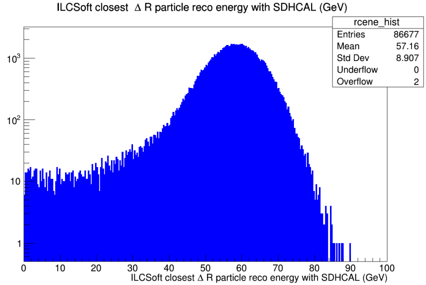
First look at the datasets for the SDHCAL validation and AHCal comparison, K_L^0

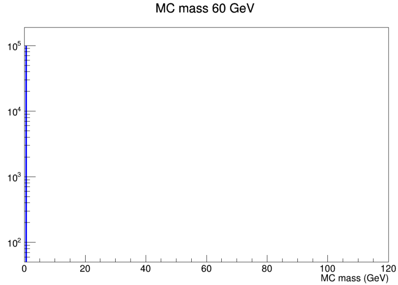
- Energy range: (1,2,5,10,20,30,40,50,60,70,80,90,100,110) GeV. **o1**
- Energy range: (1,2,5,10,20,30,40,50,70) GeV. **o2**
- We made a full copy of both datasets to our local cluster in CIEMAT dedicated to CALICE/ILD analysis by accessing the dataset via DIRAC[1]
- Using the same ilcsoft version (v02-01 $\rightarrow$ `/cvmfs/ilc.desy.de/sw/x86_64_gcc82_sl6/v02-01/init_ilcsoft.sh`) as for the central production we have produced the corresponding LCTuples.
- The resolution/linearity studies are now done with the total sum of the reconstructed enerygy in the event. Previous studies were done with the closest reco match in ΔR to the generated K_L^0.

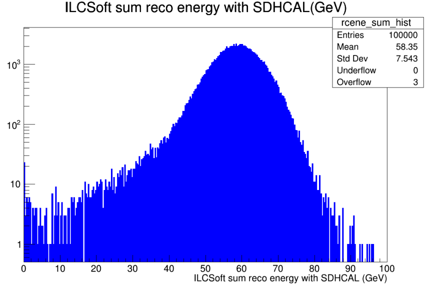
[1] We have solved certificate issues

SDHCAL validation, K_L^0 p=60 GeV









Crystalball fit

$$f(x;\alpha,n,\bar{x},\sigma) = N \cdot \begin{cases} \exp(-\frac{(x-\bar{x})^2}{2\sigma^2}), & \text{for } \frac{x-\bar{x}}{\sigma} > -\alpha \\ A \cdot (B - \frac{x-\bar{x}}{\sigma})^{-n}, & \text{for } \frac{x-\bar{x}}{\sigma} \leqslant -\alpha \end{cases}$$

$$A = \left(\frac{n}{|\alpha|}\right)^n \cdot \exp\left(-\frac{|\alpha|^2}{2}\right),$$

$$B = \frac{n}{|\alpha|} - |\alpha|,$$

$$N = \frac{1}{\sigma(C+D)},$$

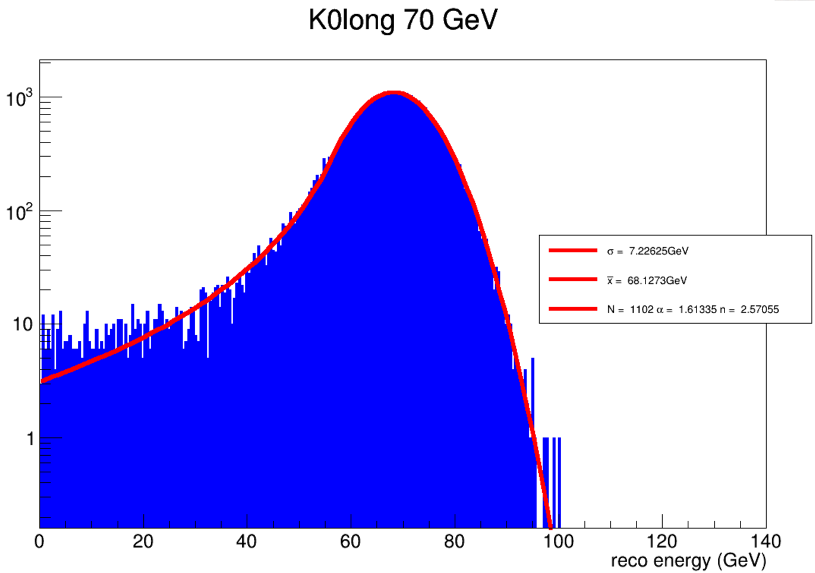
$$C = \frac{n}{|\alpha|} \cdot \frac{1}{n-1} \cdot \exp\left(-\frac{|\alpha|^2}{2}\right),$$

$$D = \sqrt{\frac{\pi}{2}} \left(1 + \operatorname{erf}\left(\frac{|\alpha|}{\sqrt{2}}\right)\right).$$

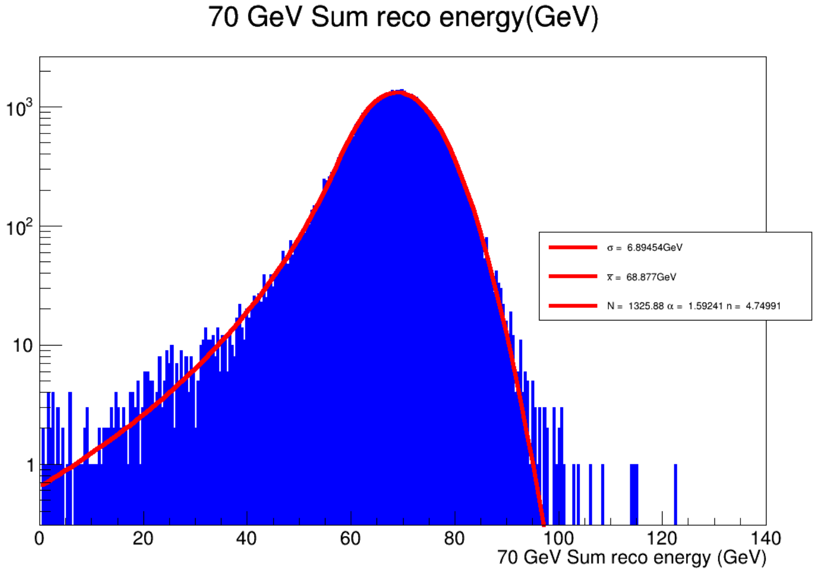
```
 FCN=342.074 FROM MIGRAD    STATUS=CONVERGED     184 CALLS          185 TOTAL
                    EDM=2.61519e-08    STRATEGY= 1  ERROR MATRIX UNCERTAINTY   0.3 per cent
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  N           2.00731e+03   9.14867e+00  -2.32131e-02   2.47481e-05
   2  mean        5.83022e+01   2.70121e-02   2.65898e-05  -4.92050e-03
   3  sigma       6.59899e+00   2.21181e-02   1.22279e-04   1.43070e-02
   4  alpha       1.80238e+00   2.83231e-02   7.01543e-05  -6.05402e-03
   5  n           1.97606e+00   1.01879e-01  -9.94635e-05   1.27298e-03
60 GeV thismax2.1e+03 mean=58 sigma=6.6 error=11%
```

https://en.wikipedia.org/wiki/Crystal_Ball_function

Crystalball fit, K_L^0 p=70 GeV, closest ΔR match, o2



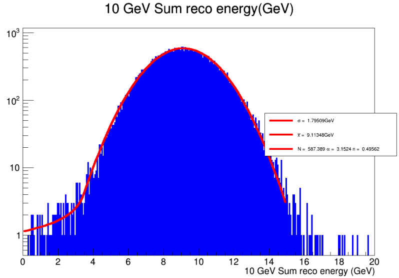
Crystalball fit, K_L^0 p=70 GeV, sum energy, o2

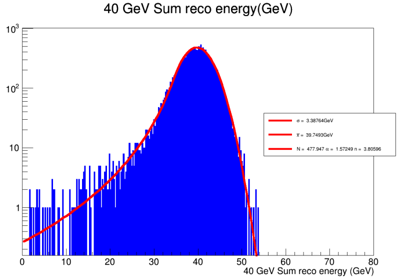


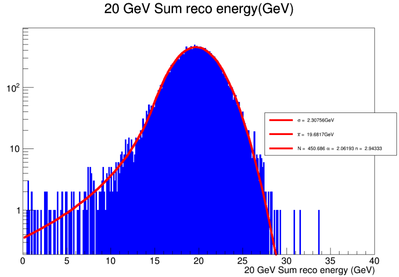
links with all results, please explore yourself:

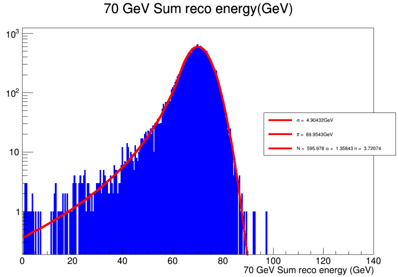
http:
//wwwae.ciemat.es/~carrillo/calice/indexk0o1.html
http:
//wwwae.ciemat.es/~carrillo/calice/indexk0o2.html

Crystalball fit, K_L^0, o1 sum reco energy

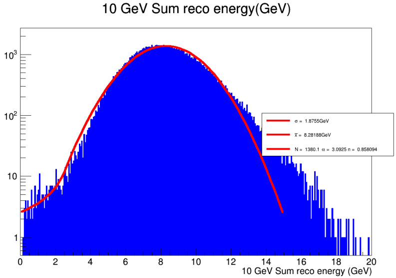


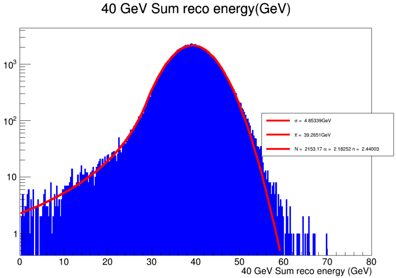


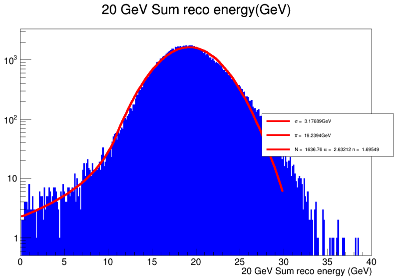


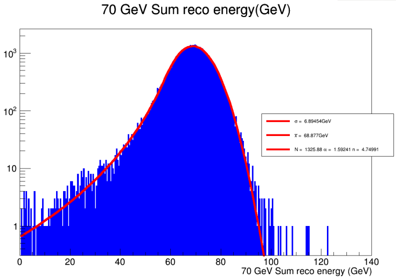


Crystalball fit, K_L^0, o2 sum reco energy







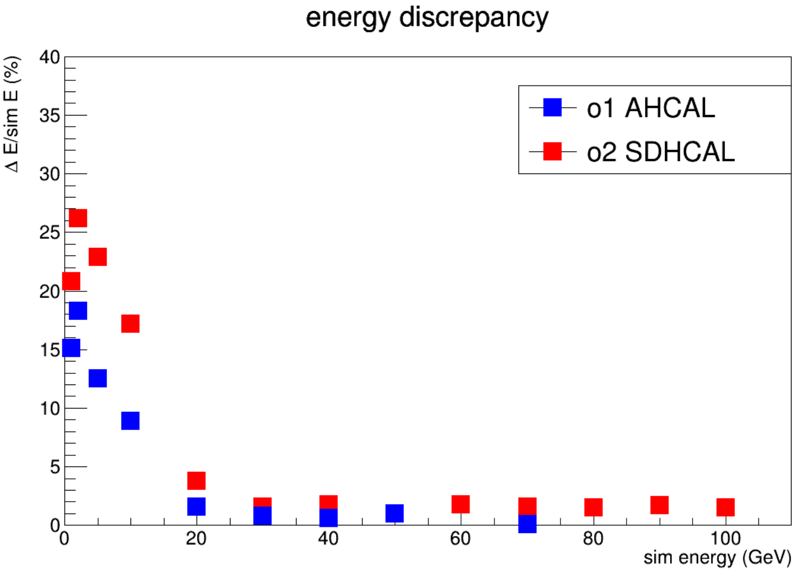


resolution and discrepancy for o1 and o2, fit results

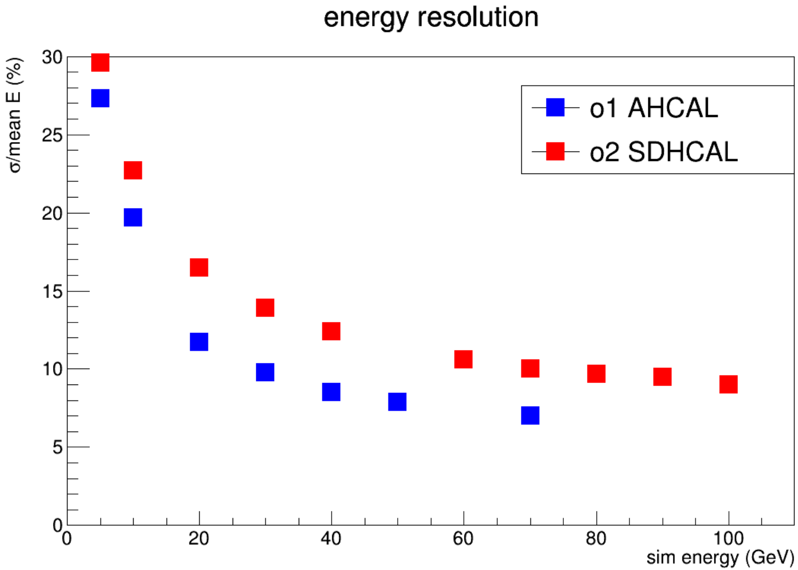
sim p (GeV)	mean (GeV)	sigma (GeV)	**resolution (%)**	**discrepancy(%)**
		o1		
1	0.85	0.34	**39.6%**	**15.1%**
2	1.64	0.61	**37.2%**	**18.3%**
5	4.37	1.19	**27.3%**	**12.5%**
10	9.11	1.80	**19.7%**	**8.9%**
20	19.68	2.31	**11.7%**	**1.6%**
30	29.75	2.91	**9.8%**	**0.8%**
40	39.75	3.39	**8.5%**	**0.6%**
50	49.50	3.94	**7.9%**	**1.0%**
70	69.95	4.90	**7.0%**	**0.1%**
		o2		
1	0.79	0.31	**38.6%**	**20.8%**
2	1.48	0.56	**38.2%**	**26.2%**
5	3.86	1.14	**29.6%**	**22.9%**
10	8.28	1.88	**22.7%**	**17.2%**
20	19.24	3.18	**16.5%**	**3.8%**
30	29.51	4.11	**13.9%**	**1.6%**
40	39.27	4.85	**12.4%**	**1.8%**
60	58.95	6.27	**10.6%**	**1.8%**
70	68.88	6.90	**10.0%**	**1.6%**
80	78.77	7.62	**9.7%**	**1.5%**
90	88.45	8.40	**9.5%**	**1.7%**
100	98.50	8.91	**9.0%**	**1.5%**

$$\text{resolution} = \frac{sigma}{mean}, \text{ discrepancy} = \frac{sim\,p - mean}{sim\,p}$$

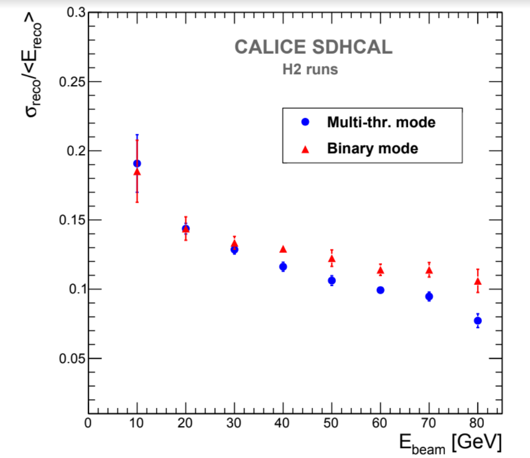
Comparison for the two scenarions, discrepancy.



Comparison for the two scenarions, resolution.



Only SDHCAL resolution observed in test-beams K_L^0



CALICE collaboration, First results of the CALICE SDHCAL technological prototype, JINST **11** (2016) P04001.

Rerunning the ILCSoft Reconstruction

we had two ideas:

- Get the SDHCAL hits out of the corresponding datasets in an ntuple for some of the masses. And forward the information to Imad's algorithm.
- Rerun the ILCSoft changing the parameters so we can improve the algorithm by changing the alpha beta gamma parameters. `https://github.com/iLCSoft/ILDConfig/blob/master/StandardConfig/production/Calibration/Calibration_ILD_l5_o2_v02.xml`

SDHCAL hits out of corresponding dataset

- Identify the correct dataset.
- To Run the standard LCTuple production with SDHCAL Calorimeter Hit collection only, to get this info: `https://github.com/iLCSoft/LCTuple/blob/master/src/CalorimeterHitBranches.cc`
- The number of hits will be the variable "ncah", the variable "caene" will contain the energy for each hit.
- For SDHCAL, there will be 3 values.
- It would be easy to compute N1, N2 and N3 from this ntuple.

SDHCAL hits out of corresponding dataset

- The Marlin parameters to get calorimeter hit in the standard ntuple are: https://github.com/iLCSoft/LCTuple/blob/master/src/LCTuple.cc
  - — WriteCalorimeterHitCollectionParameters to set to true
  - — CalorimeterHitCollection to set to the collection name containing the SDHCAL calorimeterHit.
- An example Marlin config to produce the LCTuple is here: https://github.com/iLCSoft/ILDConfig/blob/master/StandardConfig/production/MarlinStdRecoLCTuple.xml
- We have to add a MergeCollections processor to merge the SDHCAL calo Hit collections into one and then set correctly the 2 parameters above

Rerun the ILCSoft changing the parameters so we can improve the algorithm

- This option is not yet clear to me. This week we are planing a short meeting with Gerald.
- This second step will be needed as well according to Gerald(?).
- As a first step, it can be done by giving flags options to the Marlin executable.

Conclusions

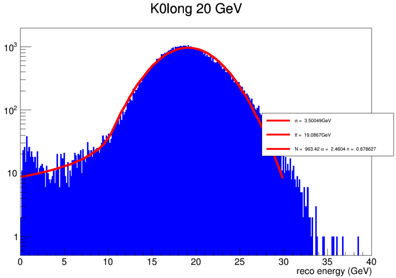
- Observed reconstructed p for the K_L^0 samples in **ilcsoft v02-01** test samples behave as expected with SDHCAL $\rightarrow$ ILD-l5-**o2**-v02.
- Studies with muon samples still to be done.
- Next steps:
  - extra variables to check the SDHCAL calibration are under scrutiny.
  - study the SDHCAL local reconstructed objects (cluster performance).
- key point about SDHCAL in ilcsoft[2]:
  - Geant4 physics model used in ilcsoft is QGSP-Bert which is not ideal to simulate SDHCAL.
  - FTF-BIC is the more appropriate for SDHCAL.

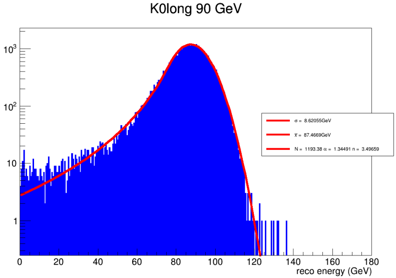
[2] https://geant4.web.cern.ch/node/155

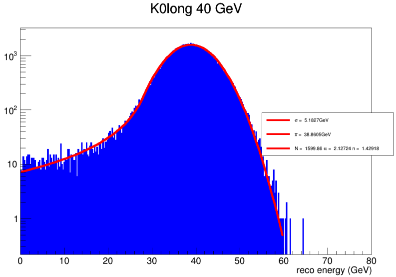
Backup

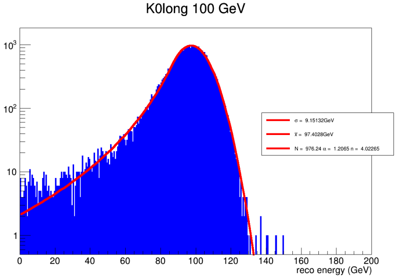
Backup

Crystalball fit, K_L^0, o2





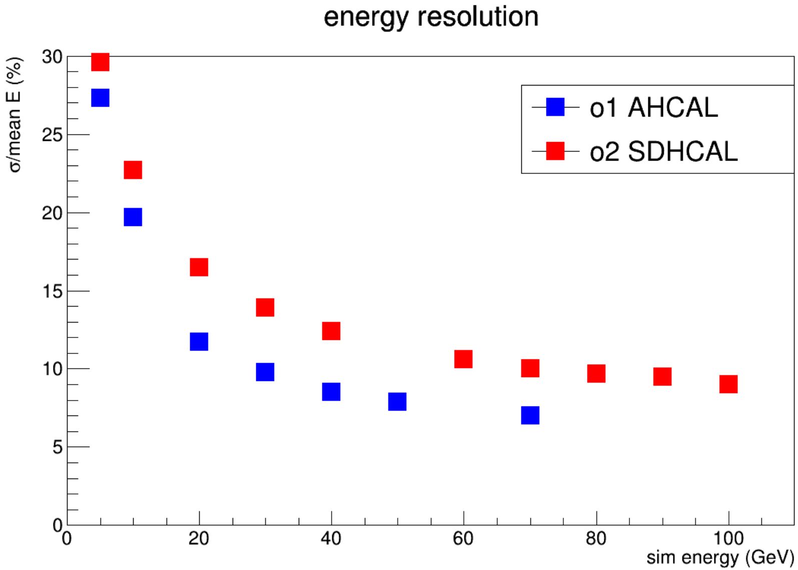




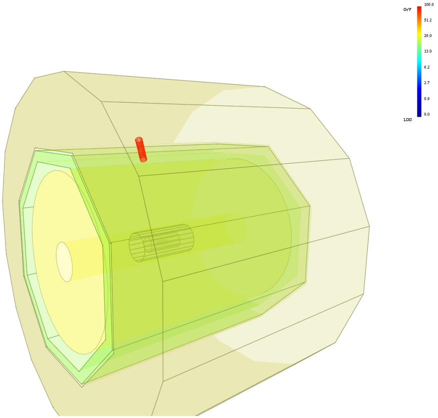
Summary, K_L^0, o2

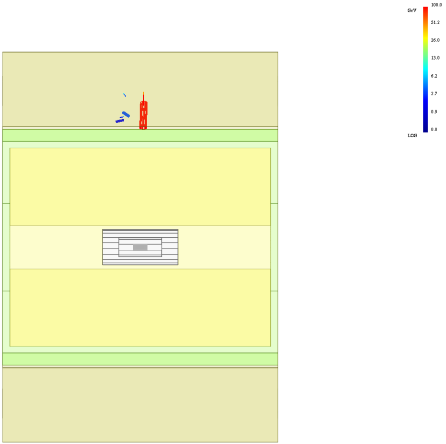
sim energy (GeV)	CB $\bar{x}$ (GeV)	CB σ (GeV)	$\frac{\sigma}{E}$ (%)
1	0.79	0.3	30
2	1.4	0.53	26
5	3.5	1.2	25
10	7.8	2.1	21
20	19	3.5	18
30	29	4.4	15
40	39	5.2	13
60	58	6.6	11
70	68	7.2	10
80	78	7.9	9.8
90	87	8.6	9.6
100	97	9.2	9.2

Summary Resolution, K_L^0



First look at the datasets for the SDHCAL validation, event display K_L^0 110 GeV, energy deposit in SDHCAL



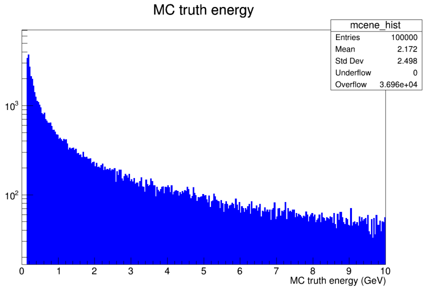


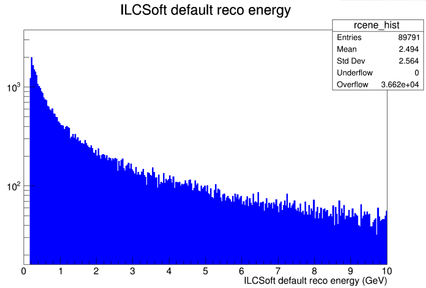
List of variables available in the standard LCTuple

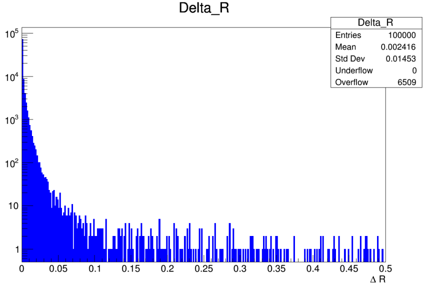
```
// Fixed size dimensions of array or collections stored in the TTree if any.

   // Declaration of leaf types
   Int_t           evevt;
   Int_t           evrun;
   Float_t         evwgt;
   Long64_t        evtim;
   Float_t         evsig;
   Float_t         evene;
   Float_t         evpoe;
   Float_t         evpop;
   Int_t           evnch;
   Char_t          evpro[1];   //[evnch]
   Int_t           nmcp;
   Int_t           mcori[807];   //[nmcp]
   Int_t           mcpdg[807];   //[nmcp]
   Int_t           mcgst[807];   //[nmcp]
   Int_t           mcsst[807];   //[nmcp]
   Float_t         mcvtx[807];   //[nmcp]
   Float_t         mcvty[807];   //[nmcp]
   Float_t         mcvtz[807];   //[nmcp]
   Float_t         mcepx[807];   //[nmcp]
   Float_t         mcepy[807];   //[nmcp]
   Float_t         mcepz[807];   //[nmcp]
   Float_t         mcmox[807];   //[nmcp]
   Float_t         mcmoy[807];   //[nmcp]
   Float_t         mcmoz[807];   //[nmcp]
   Float_t         mcmas[807];   //[nmcp]
   Float_t         mcene[807];   //[nmcp]
   Float_t         mccha[807];   //[nmcp]
   Float_t         mctim[807];   //[nmcp]
   Float_t         mcspx[807];   //[nmcp]
   Float_t         mcspy[807];   //[nmcp]
   Float_t         mcspz[807];   //[nmcp]
   Int_t           mccf0[807];   //[nmcp]
   Int_t           mccf1[807];   //[nmcp]
   Int_t           mcpa0[807];   //[nmcp]
   Int_t           mcpa1[807];   //[nmcp]
   Int_t           mcda0[807];   //[nmcp]
   Int_t           mcda1[807];   //[nmcp]
   Int_t           mcda2[807];   //[nmcp]
   Int_t           mcda3[807];   //[nmcp]
   Int_t           mcda4[807];   //[nmcp]
   Int_t           nrec;
   Int_t           rcori[41];   //[nrec]
   Int_t           rccid[41];   //[nrec]
   Int_t           rctyp[41];   //[nrec]
   Float_t         rccov[41][10];   //[nrec]
   Float_t         rcrpx[41];   //[nrec]
   Float_t         rcrpy[41];   //[nrec]
   Float_t         rcrpz[41];   //[nrec]
   Float_t         rcgpi[41];   //[nrec]
   Int_t           rcpiu[41];   //[nrec]
   Int_t           rcnpi[41];   //[nrec]
   Int_t           rcfpi[41];   //[nrec]
   Float_t         rcmox[41];   //[nrec]
   Float_t         rcmoy[41];   //[nrec]
   Float_t         rcmoz[41];   //[nrec]
   Float_t         rcmas[41];   //[nrec]
   Float_t         rcene[41];   //[nrec]
   Float_t         rccha[41];   //[nrec]
   Int_t           rcntr[41];   //[nrec]
   Int_t           rcncl[41];   //[nrec]
   Int_t           rcnrp[41];   //[nrec]
   Int_t           rcftr[41];   //[nrec]
```

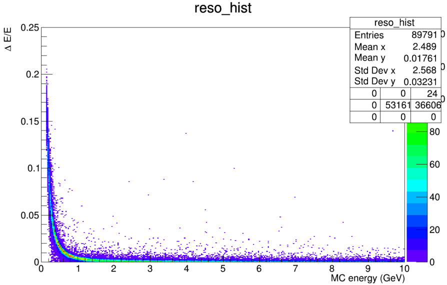
```
   Int_t           rcftr[41];   //[nrec]
   Int_t           rcvts[41];   //[nrec]
   Int_t           rcvte[41];   //[nrec]
   Int_t           rccom[41];   //[nrec]
   Int_t           npid;
   Int_t           pityp[212];   //[npid]
   Int_t           pipdg[212];   //[npid]
   Float_t         pillh[212];   //[npid]
   Int_t           pialg[212];   //[npid]
   Int_t           ntrk;
   Int_t           trori[36];   //[ntrk]
   Int_t           trtyp[36];   //[ntrk]
   Float_t         trch2[36];   //[ntrk]
   Int_t           trndf[36];   //[ntrk]
   Float_t         tredx[36];   //[ntrk]
   Float_t         trede[36];   //[ntrk]
   Float_t         trrih[36];   //[ntrk]
   Int_t           trshn[36][12];   //[ntrk]
   Int_t           trnts[36];   //[ntrk]
   Int_t           trfts[36];   //[ntrk]
   Int_t           trsip[36];   //[ntrk]
   Int_t           trsfh[36];   //[ntrk]
   Int_t           trslh[36];   //[ntrk]
   Int_t           trsca[36];   //[ntrk]
   Int_t           ntrst;
   Int_t           tsloc[144];   //[ntrst]
   Float_t         tsdze[144];   //[ntrst]
   Float_t         tsphi[144];   //[ntrst]
   Float_t         tsome[144];   //[ntrst]
   Float_t         tszze[144];   //[ntrst]
   Float_t         tstnl[144];   //[ntrst]
   Float_t         tscov[144][15];   //[ntrst]
   Float_t         tsrpx[144];   //[ntrst]
   Float_t         tsrpy[144];   //[ntrst]
   Float_t         tsrpz[144];   //[ntrst]
   Int_t           nsth;
   Int_t           stori[1];   //[nsth]
   Int_t           stci0[1];   //[nsth]
   Int_t           stci1[1];   //[nsth]
   Double_t        stpox[1];   //[nsth]
   Double_t        stpoy[1];   //[nsth]
   Double_t        stpoz[1];   //[nsth]
   Float_t         stedp[1];   //[nsth]
   Float_t         sttim[1];   //[nsth]
   Float_t         stmox[1];   //[nsth]
   Float_t         stmoy[1];   //[nsth]
   Float_t         stmoz[1];   //[nsth]
   Float_t         stptl[1];   //[nsth]
   Int_t           stmcp[1];   //[nsth]
   Int_t           nsch;
   Int_t           scori[1];   //[nsch]
   Int_t           scci0[1];   //[nsch]
   Int_t           scci1[1];   //[nsch]
   Float_t         scpox[1];   //[nsch]
   Float_t         scpoy[1];   //[nsch]
   Float_t         scpoz[1];   //[nsch]
   Float_t         scene[1];   //[nsch]
   Int_t           r2mnrel;
   Int_t           r2mf[169];   //[r2mnrel]
   Int_t           r2mt[169];   //[r2mnrel]
   Float_t         r2mw[169];   //[r2mnrel]
```

SDHCAL validation, μ sample

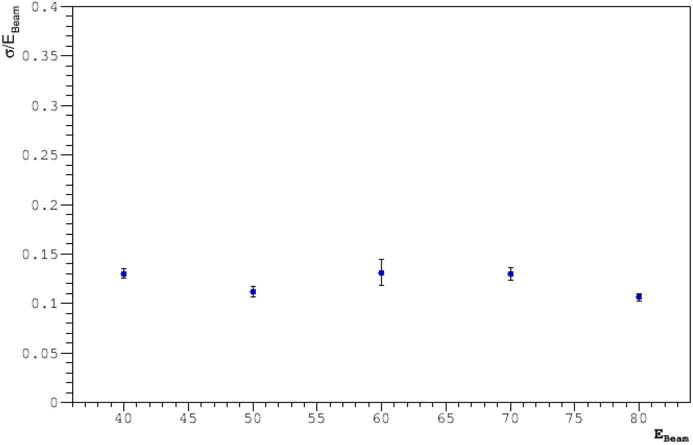








SDHCAL validation, TB2018



The tools we have learned.

In the framework of the SDHCAL test-beams data analysis we have learned:

- How to work in the ILCSoft analysis framework. (Installed in CIEMAT running in dedicated nodes)
- Run from scratch a simulation using the standard sequences in the framework and switching from one sceario to another (large $\rightarrow$ small), (AHCAL $\rightarrow$ SDHCAL), etc.
- Navigate and run over the centrally produced datasets (DIRAC)
- Produce ntuples out of the samples for detector/physics analysis. (AIDA,REC,SIM)
- Use reconstructed physics objects and produce event cut flows for analysis.
- Event display, etc.

The tools we have learned

Private CIEMAT-SDHCAL pion gun simulation for comparison with TB-2018.