sPHENIX EMCal Tower-By-Tower Calibration using Pi0 method
+
Photon Optimal Cuts

Presenter: Shyam Chauhan

Advisor: Justin Frantz

Ohio University

First Part:

Overall steps of pi0-tbt-method:

analysis note is being released soon: https://www.overleaf.com/read/yxsdjvgjsbfp

1. Calculate the invariant mass of cluster pairs in each event. There are several sets of cuts implemented before calculating invariant mass in order to reduce background. Fill the histograms with the invariant mass values for each corresponding maximum tower from each cluster.

2. Fit the histograms of π^0 peaks of each tower with Gaussian + polynomial function to extract the peak mean and other parameter values.

3. Calculate the correction factor for each maximum tower from each cluster. The correction factor is the ratio of pi0 true invariant mass to the peak mean of the histogram from that cluster's maximum tower. The correction factor values varies tower-by-tower, and iteration-by-iteration. Obtain the correction factor from all the towers, and save it in the file as root's "NTuple".

4. Iterate by repeating all steps as above applying the derived correction factors from the last iteration. Recalculate the invariant mass for the towers and follow the steps as above, iterating until convergence of the correction values i.e. until the desired accuracy is met. It takes about 6-8 iterations (each iterations takes about 4 hours, approximately) to perform complete calibration (referring back to PHENIX). After certain number of iterations, the pi0 fit mean comes approximately close to the true invariant mass value of pi0, and doing further iterations also won't do any better, then we know convergence has been met.

Flow–Chart showing steps followed for Calibrations in this analysis

Code in : coresoftware/calibrations/calorimeter/calo_emc_pi0_tbt/ CaloCalibEmc_Pi0.cc/h class

Node trees

Fun4all….C macro
run/condor jobs

::Process event()
make TTree for each event
saving all cluster info.

(~30% of clusters
– 0.6 GeV
cut/peripheral)

::Loop() save
histograms using
cluster pairs
invariant mass for
EACH TOWER
(24K towers)

loop macro run

corrections applied

::GetHIstos()
Invariant Mass histograms
-1 for each tower

save_corr_val.C
macro run

::Fit() fit the histograms and save
TNtuple with correction values saved

loop macro: loops the saved clusters in pairs and calculates inv. mass and saves them to histogram
Applies corrections derived from any previous iteration

save_corr_val: fits the histogram and collects the correction values to be applied in next iteration

Motivations:

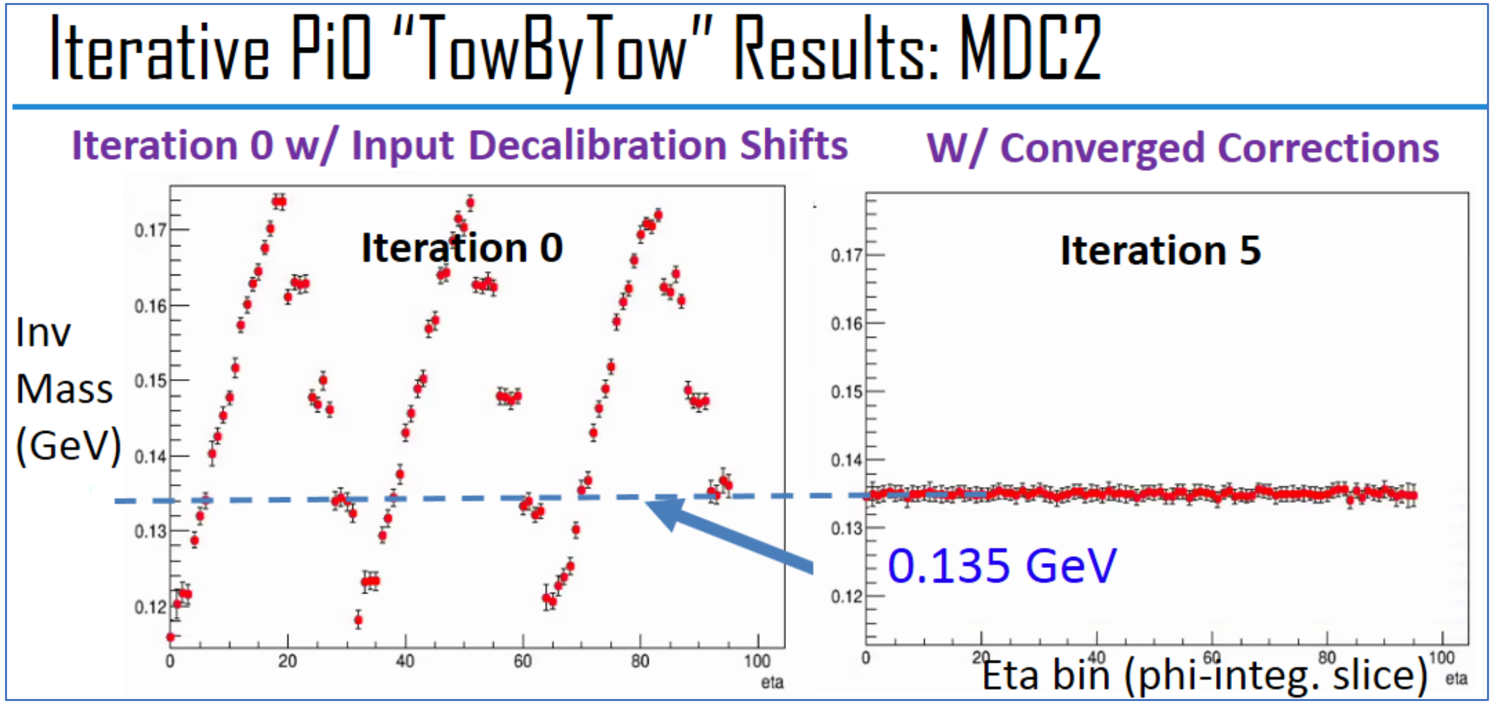
- Can our fit functions and pi0 calibration code machinery work for possible worst-case-scenario of decalibrations ?
- We need lots of statistics to perform the fits for each tower
- It will take lots of time to perform fitting for each of the EMCal towers

Ideas:

- Implement the worst-case pattern shifts of energy to the EMCal towers and see if our code can recover it
- To obtain lots of statistics for individual towers, after implementing shift patterns, add pieces of tower blocks at one locations (or 3 locations) and perform fits only on that added tower block locations
- Performing fits only in few towers' location helps to do tests quickly

Review previous MDC2 test on Eta Slices

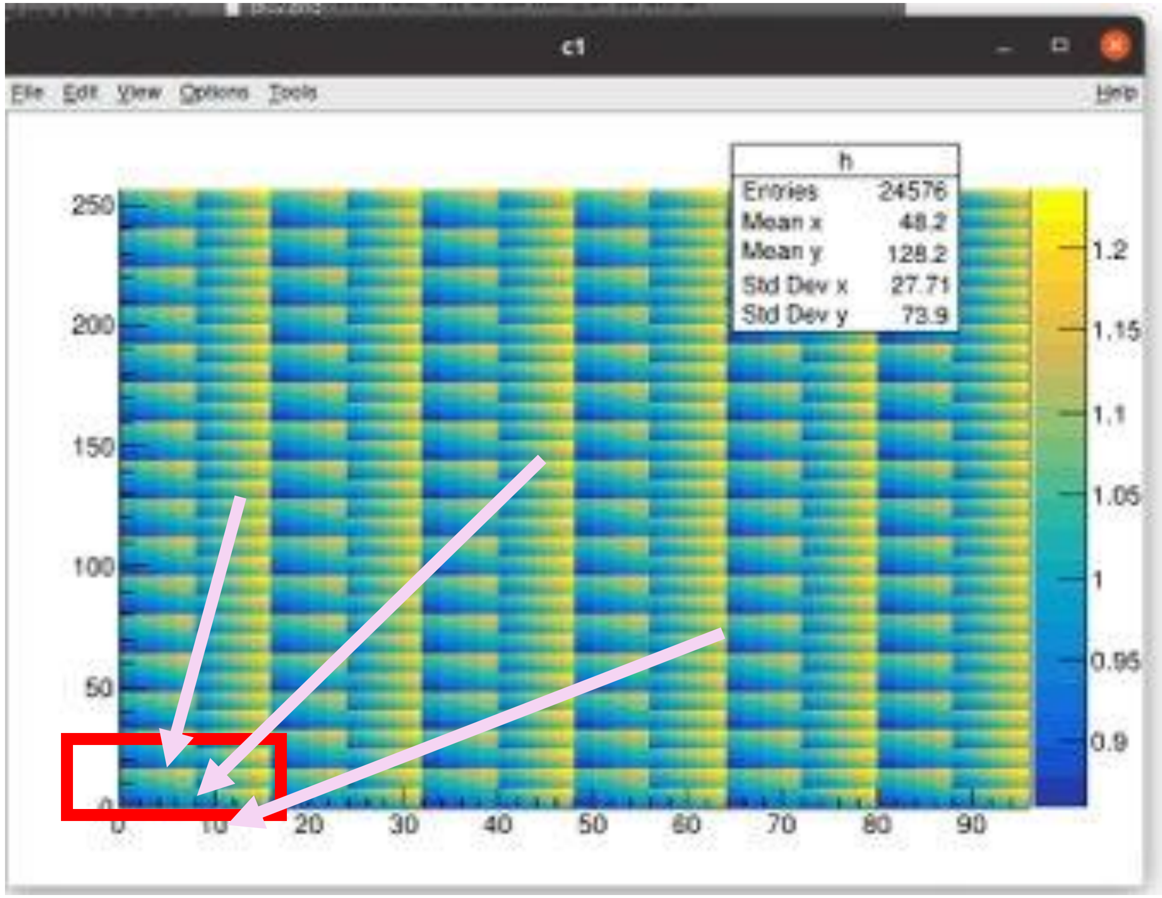
- ~10-20 M (MDC2) events is not enough statistics to test full pi0 method on all 24 K towers tower by tower
  - Previous test : pseudorapidity (eta) slices - test sensitivity to common decalibration shift value across each eta bin (same for 256) phi bins. Allows 256* more statistics when fitting invariant mass histogram
- Eta slices only share and unfold energy (and decalibration) in "1-D" eta-direction – converges faster than 2-D (eta-phi) unfolding



2D Pattern

- New test: make a single decalibration shift pattern for 2-D eta-phi 16x16 tower-unit, repeat same pattern across detector
- perform iterations as usual but to detect shift, fit invariant mass histograms with extra statistics from combining all instances of pattern for each tower in the 16x16 unit

256/16 = 16 pattern instances along phi



96/16 = 6 pattern instances along eta

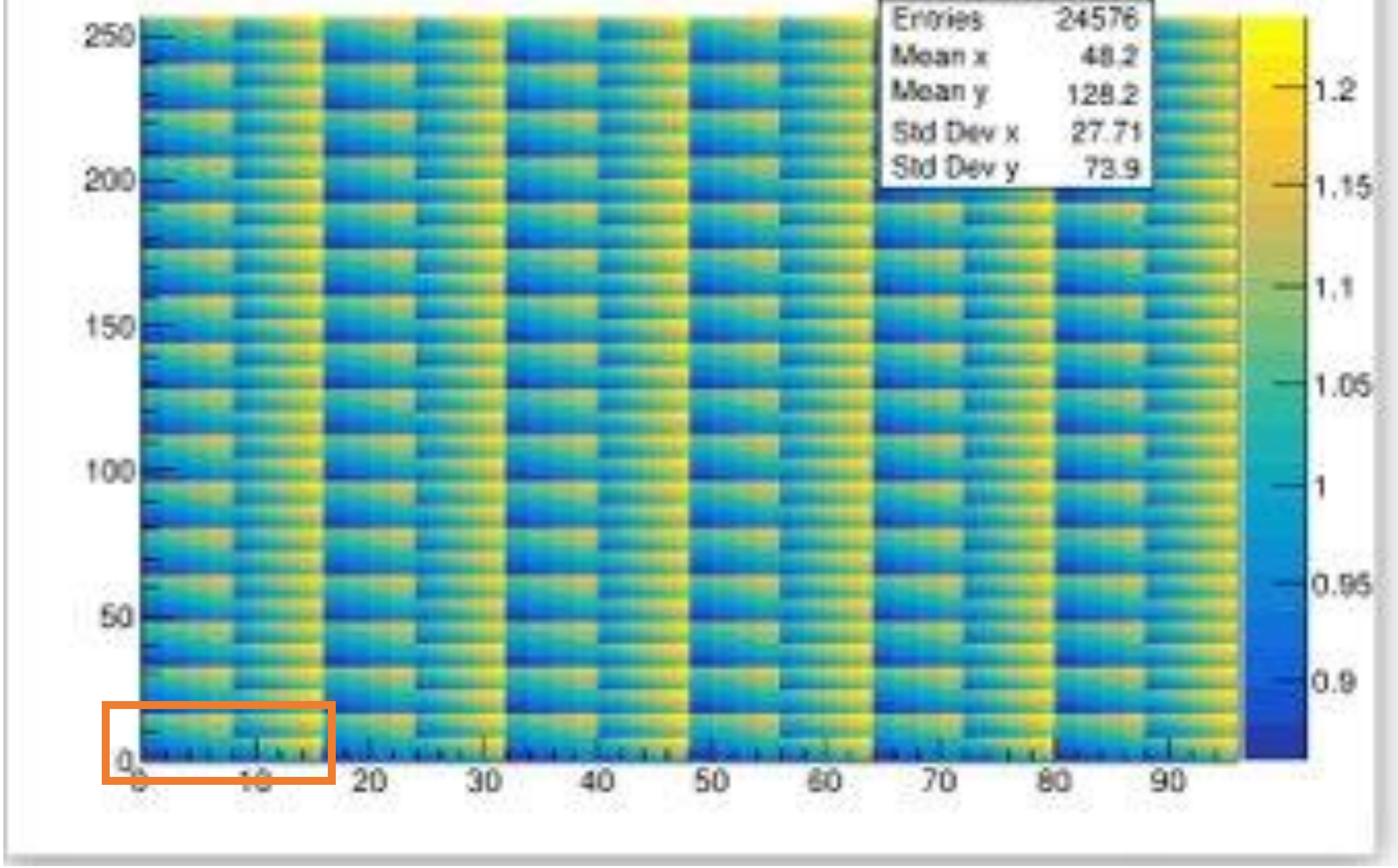
Total statistical increase for each of the 16x16 tower unit = times number of instances --> **6x16 = 96 times**

Decalibration Shift pattern generation

For reference

Used "worst-case" decalibration shifts ranging from ~0.86 (14% down decalibration) to 1.3 (30% upward shift)

```
void pattern_jf()
{
    TH2F *hist = new TH2F("h", "", 96,0,96, 256,0,256);

    for (int i=0; i<96; i++)
    {
        for (int j=0; j<256; j++)
        {
            float e = 1.00;
            int ir = -999;
            int jr = -999;
            if ( (i>=8 && i<16) || (i>=24 && i<32) || (i>=40 && i<48) ||
              (i>=56 && i<64) || (i>=72 && i<80) || (i>=88 && i<96) )
            {
                ir = i%8;
                jr = j%8;

                e *= 0.885+ir*0.025+jr*0.025;

            }
            else
            {
                int ib2 = i/2;
                ir = ib2%4;
                int jb2 = j/2;
                jr = jb2%8;

                e *= 0.86+ir*0.030+jr*0.030;
            }

          // e *= 0.86+ir*0.03+jr*0.03;
            hist->SetBinContent(i+1, j+1, e);
        }
    }
    hist->Draw("colz");
    //    hist->Draw("text,same");
}
```



Apply this pattern of decalibration, make invariant mass histograms for each iteration then combine pattern instances

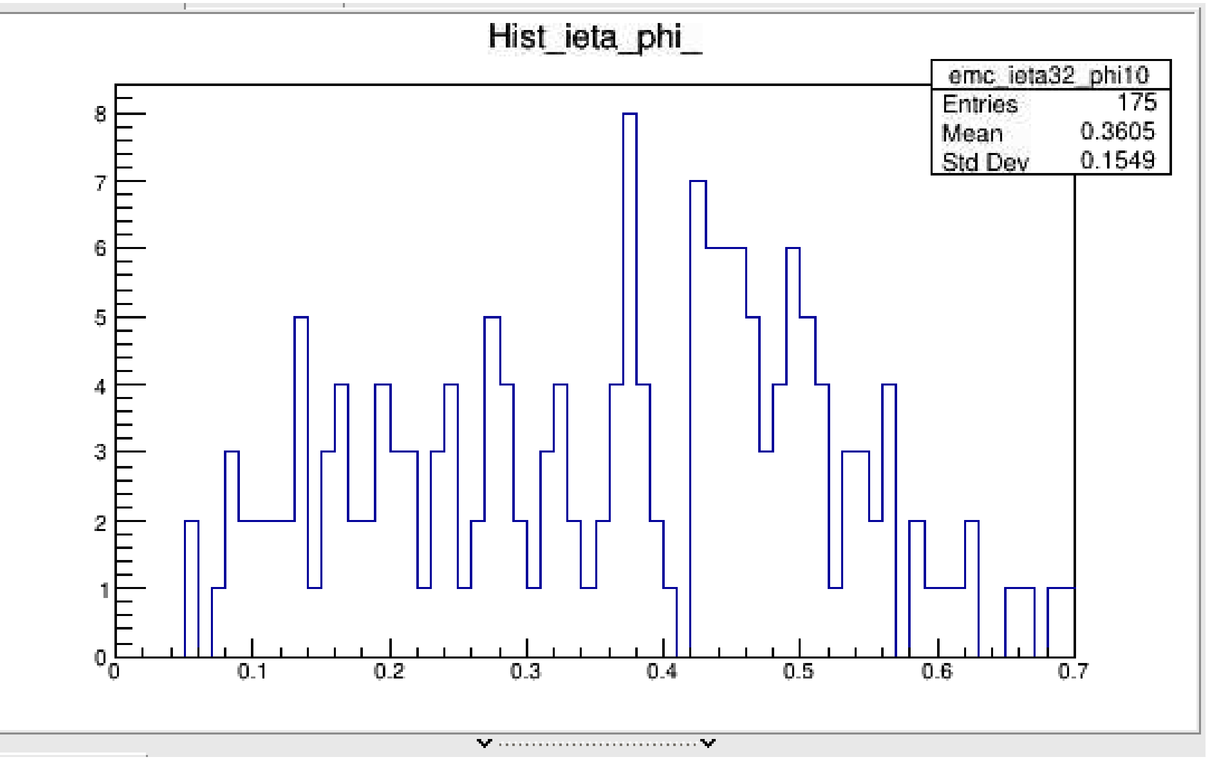
Combining statistics code:

```
//______________________________________________________________________..
void CaloCalibEmc_Pi0::Add_96()
{
  std::cout << " Inside Add_96()." << std::endl;
  for (int ieta=0; ieta<16; ieta++)
  {
    for (int iphi=0; iphi<16; iphi++)
    {
      for (int ipatt_eta=0; ipatt_eta<6; ipatt_eta++)
      {
        for (int ipatt_phi=0; ipatt_phi<16; ipatt_phi++)
        {
          if ((ipatt_eta>0) || (ipatt_phi>0))
          {
            cemc_hist_eta_phi[ieta][iphi]->Add(cemc_hist_eta_phi[ieta+ipatt_eta*16][iphi+ipatt_phi*16]);
          }
        }
      }
    }
  }
  std::cout << " Finished Add_96(). " << std::endl;
  //cemc_hist_eta_phi[0][0]->Draw();
  //cemc_hist_eta_phi[17][2]->Draw();
}

~
~
```

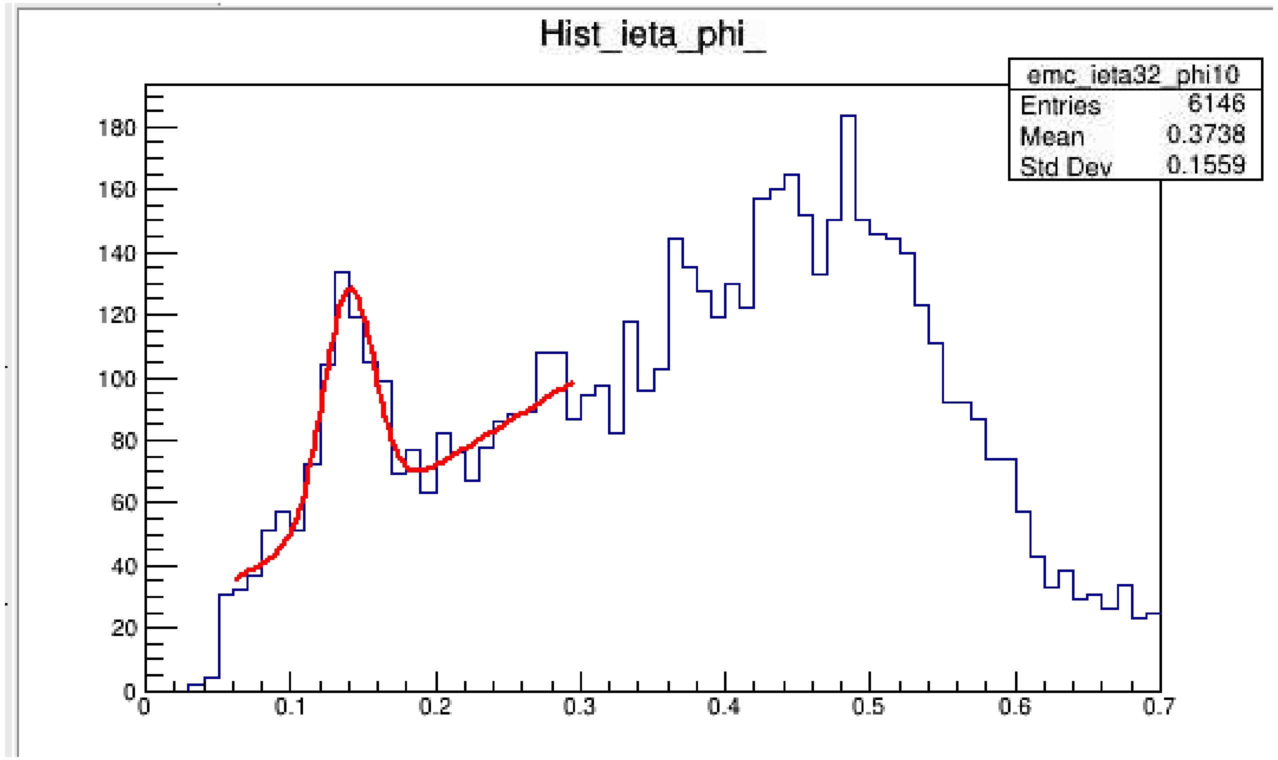
Fitting pi0 tbt added tower block's location

Example of combining statistics

Example: eta0_phi10 tower ~8M total events



Single tower from all MDC 2 events statistics

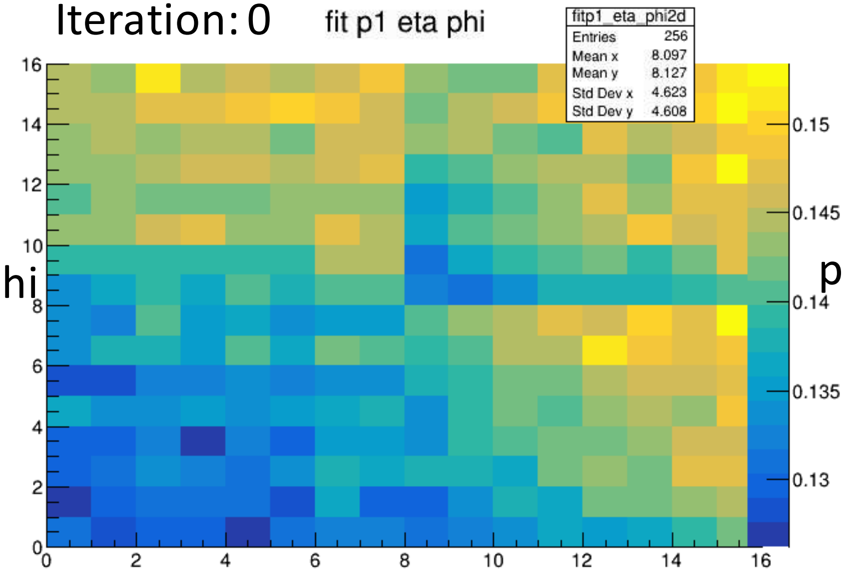


Same tower location with combined 96 times statistics

Results

Pi0 Peak Fit Mass Value Distribution (z-axis / color is mass fit value)

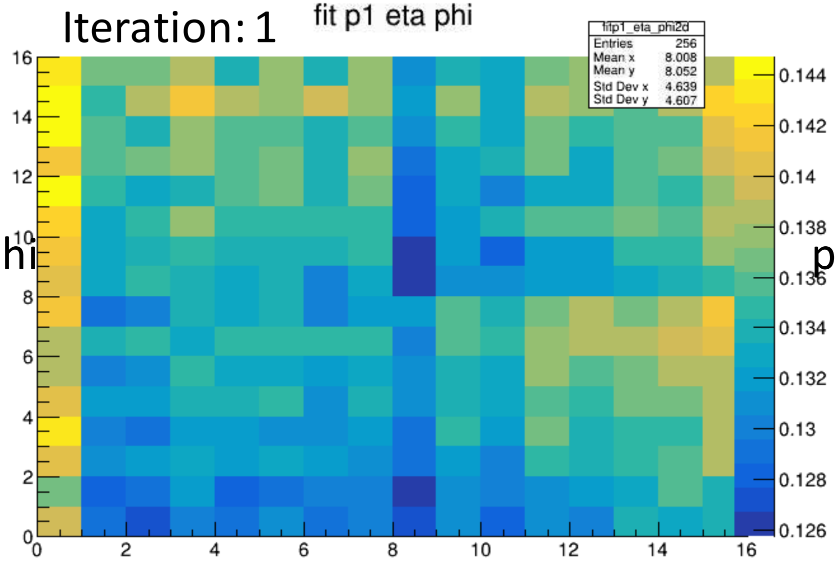
Iteration: 0



phi

eta

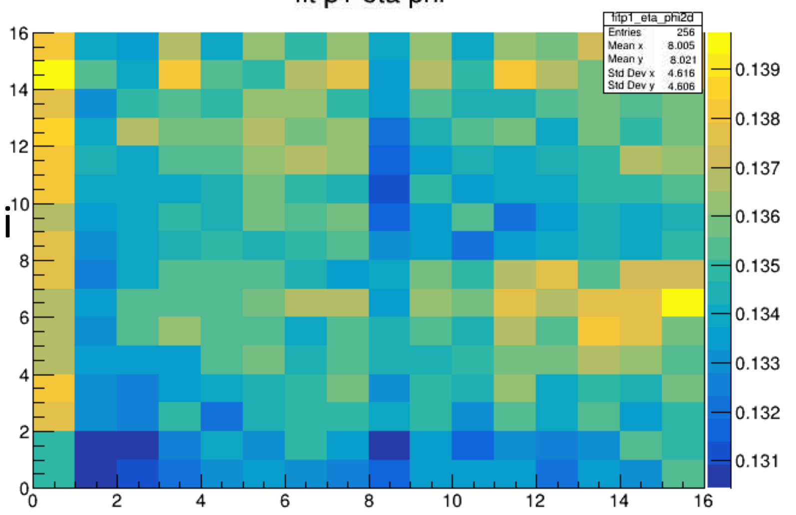
Iteration: 1



phi

eta

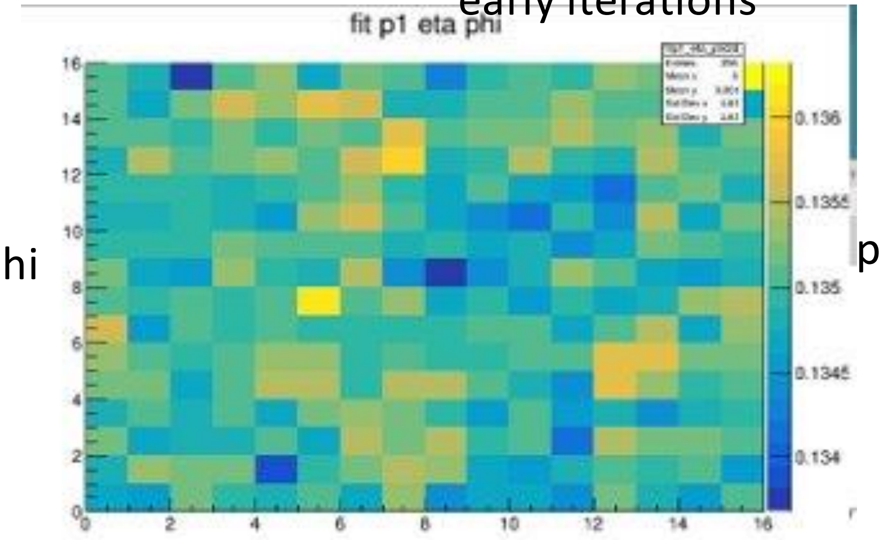
Iteration: 2



phi

eta

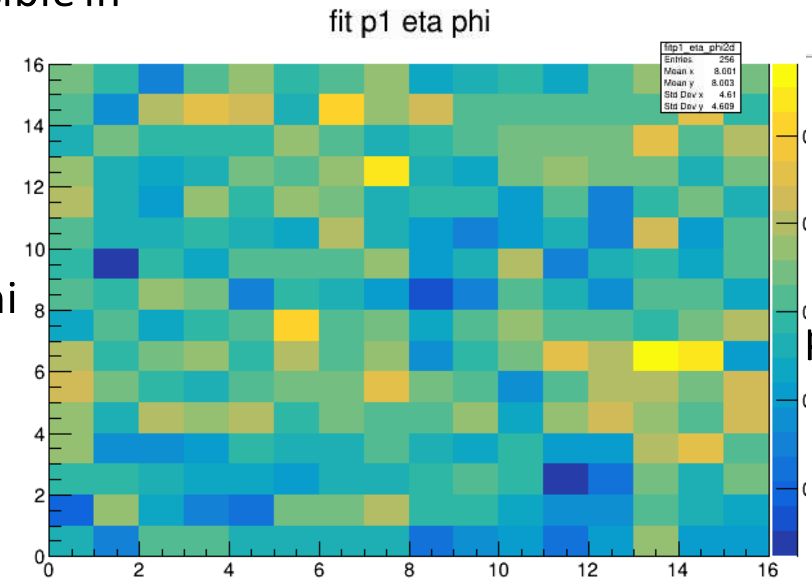
Input Pattern effect visible in early iterations



phi

Iteration: 5

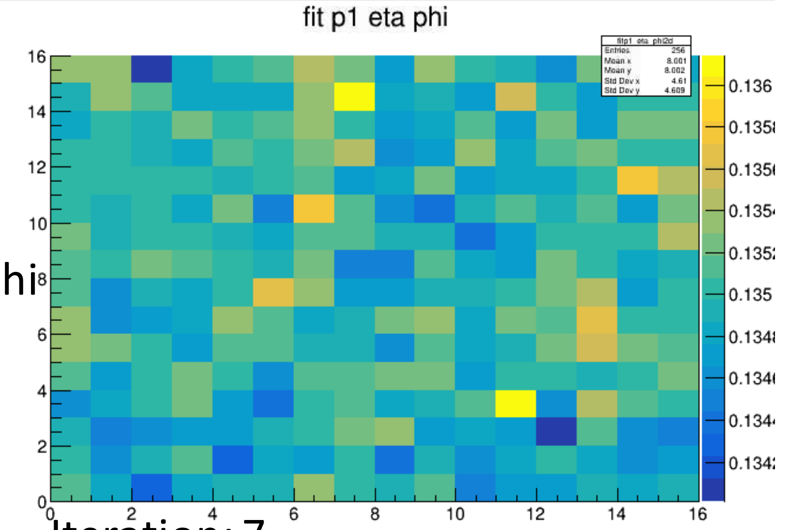
eta



phi

Iteration: 6

eta

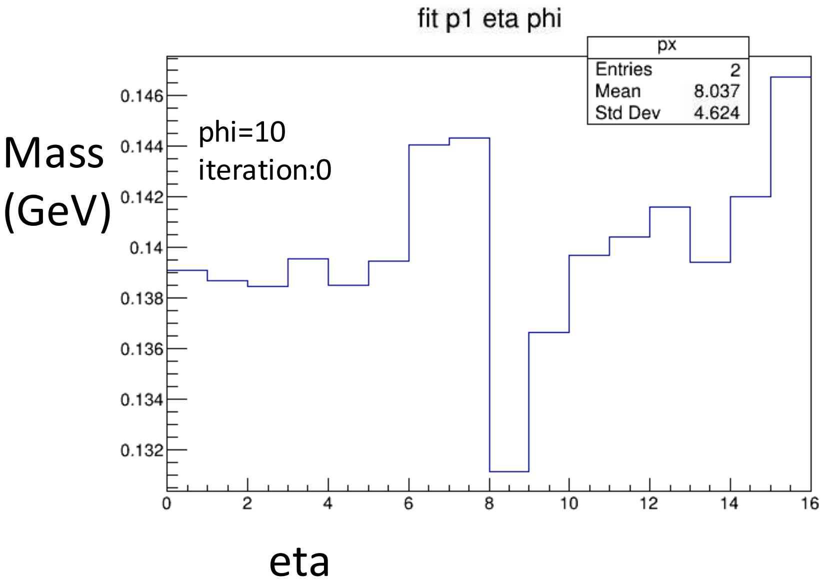


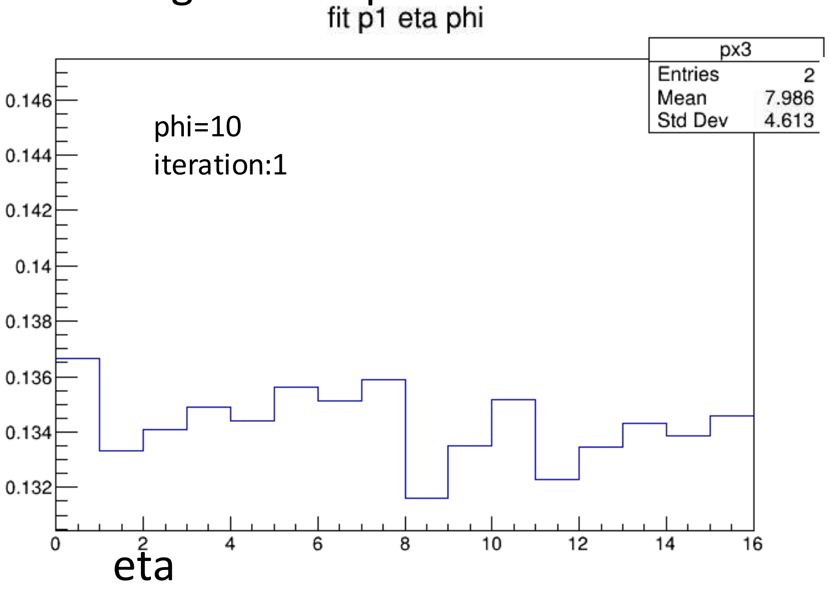
phi

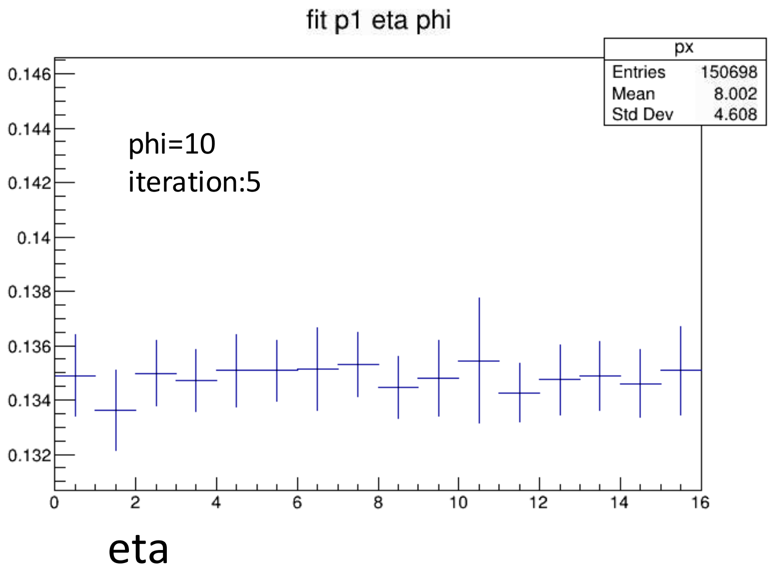
Iteration: 7

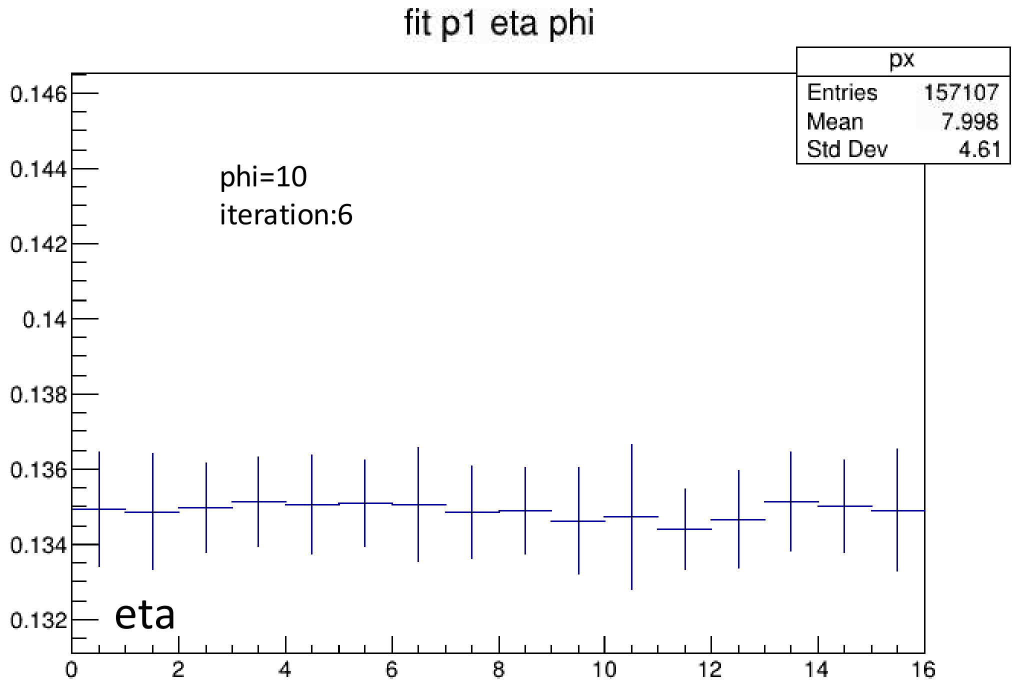
eta

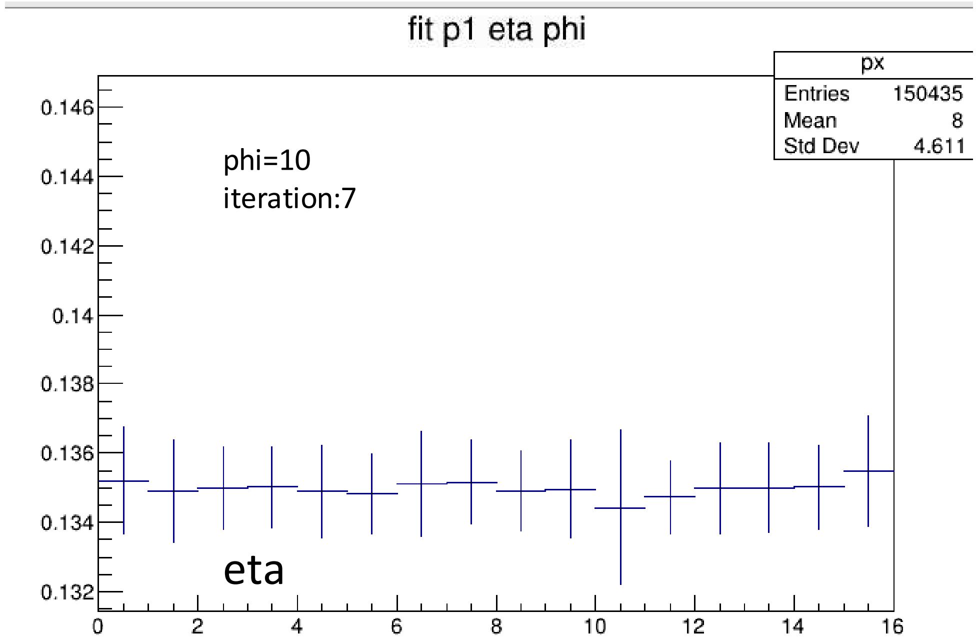
1-Phi Bin Projection: ProjectionX of previous histograms for phi=10:











Conclusions and Outlook-Part1:

- Adding the shift patterns into the cluster energy and pt should dramatically change the pi0 invariant mass peak locations, and our code should still be able to bring the shifts to the true pi0 mass value

- In order to test individual towers ~25000 in EMCal, we needed lots of statistics, in order to make the higher statistics so that the fit don’t fail, we added the tower blocks in various ways (96, and 32 blocks)

- We were able to see that each tower (in our chosen block) has increased statistics and that the fit function is fitting them well.

- Also, even the possibly worst shifts implemented in the EMCal towers is being corrected

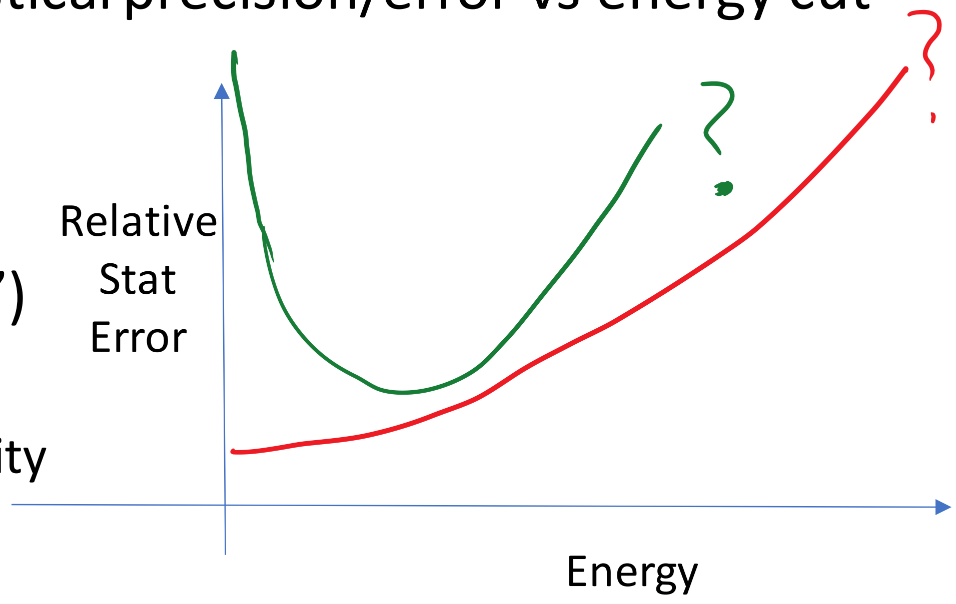
Part-2

Optimizing Cut Values – v1

Optimizing energy cuts

1) not addressed today but rough ideas from current fit method developments

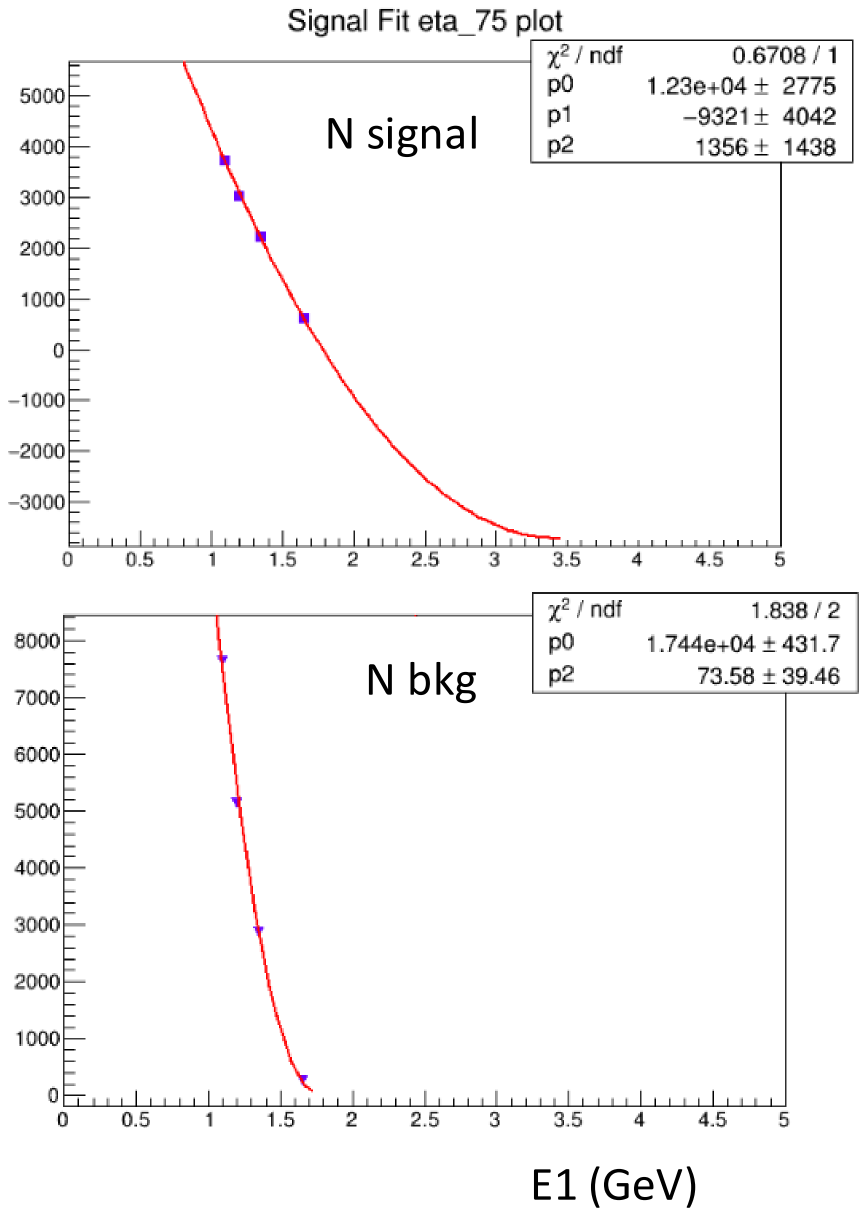
- Two goals:
- 1) Peak fitting robustness to eliminate/minimize human interaction each iteration
  - Most important consideration is S/B (B is mostly combinatoric background)
  - Drives cuts towards high energy decay photons to get good S/B
- 2) Maximize statistical precision per event to minimize number of events needed
  - Drives cuts towards lower energy … e.g. lower than fit considerations
  - statistical precision of fit: S/B: if Comb. Bkg increases fast enough, faster than Signal, as energy is lowered, there may be optimal energy cut values for optimizing statistical precision
- Address 2): want a fairly simple estimator of relative statistical precision/error vs energy cut values without relying on *actual* fitting
  - too slow to do actual analyses for many cut values
  - also, for low pt (< 1 GeV), fit likely to fail,
- With MDC2 pass used so far, minimum photon E cut for either photon is ~1.1 GeV—call this photon 2 E ("E2") cut (new pass 0.6 GeV cut being analyzed)
  - There are also energy asymm α cuts to vary (α= |E1-E2|/(E1+E2)) but leave this as const < 0.5 for simplicity
- **First look: Vary only photon1 E ("E1") cuts**
  - Pi0 Pt/energy ~ E1+1.1 e.g. for 1.3, pi0 pt/E ~= 2.4 GeV



Fit Signal and Background as fn of energy

- For four E1 cut values, estimate S and B at all cut values with simple sideband estimates
  - Avg low and high band = Bkg
  - Peak integral = S+B
  - S = Peak integral - Bkg
- Fit dependence of N signal and N bkg vs E1
  - For now using simple quadratic, not sure how kinematic cuts affect spectra
  - To do: use more realistic e.g. power law/exponential fit functions

- Extract Fit Fns of S(E1) and B(E1) at all E1.
- We want to minimize stat error of peak mean (gaus “p1” param—but need formula to avoid actual fit)
  - not directly minimizing S/B
  - Need estimator fn f(S,B) of peak mean error



Estimator fn

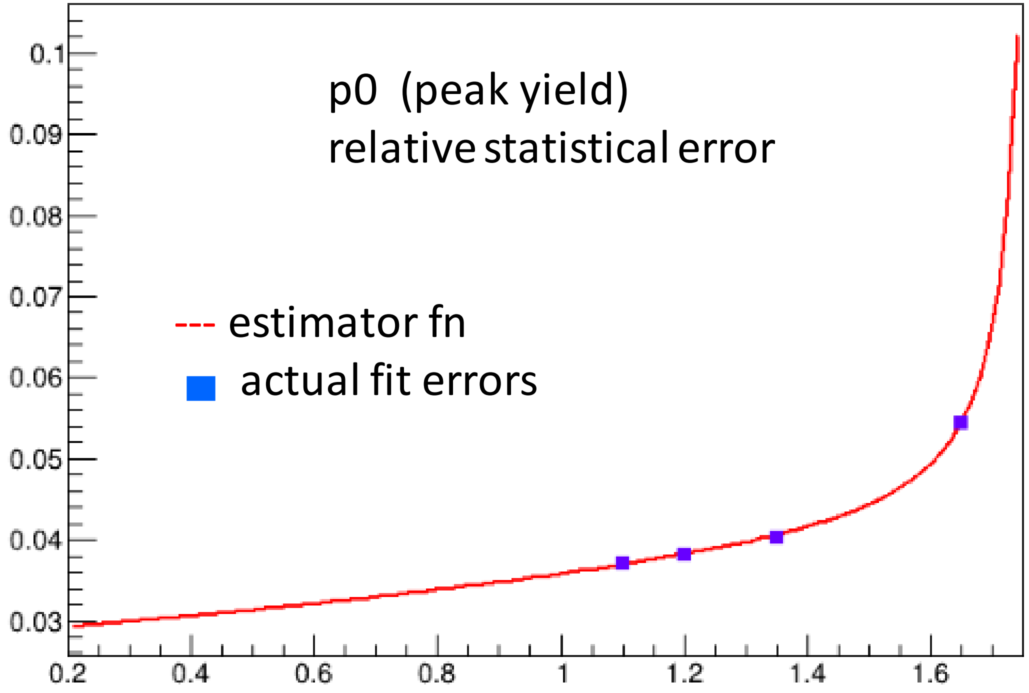
- Naïve relative error of peak **yield** assuming poisson sqrt(S+B)/S
- Better formula known (e.g. CKB thesis*) to be ∝ sqrt(S+2B)
- Found that sqrt(S+2B)/S best estimator of peak **yield** relative error = 1/ sqrt("N_measments")
- For peak **mean** error, use std error of mean formula : std_dev / sqrt("N_measurements")
- peak mean relative error → divided by ~0.135 gaus fit parameters : f(S,B) = **[p2* sqrt(S+2B)/S]/p1**
  - comparison to p1 error from fit, relatively good, ~10% below actual errors– across several cut, eta values

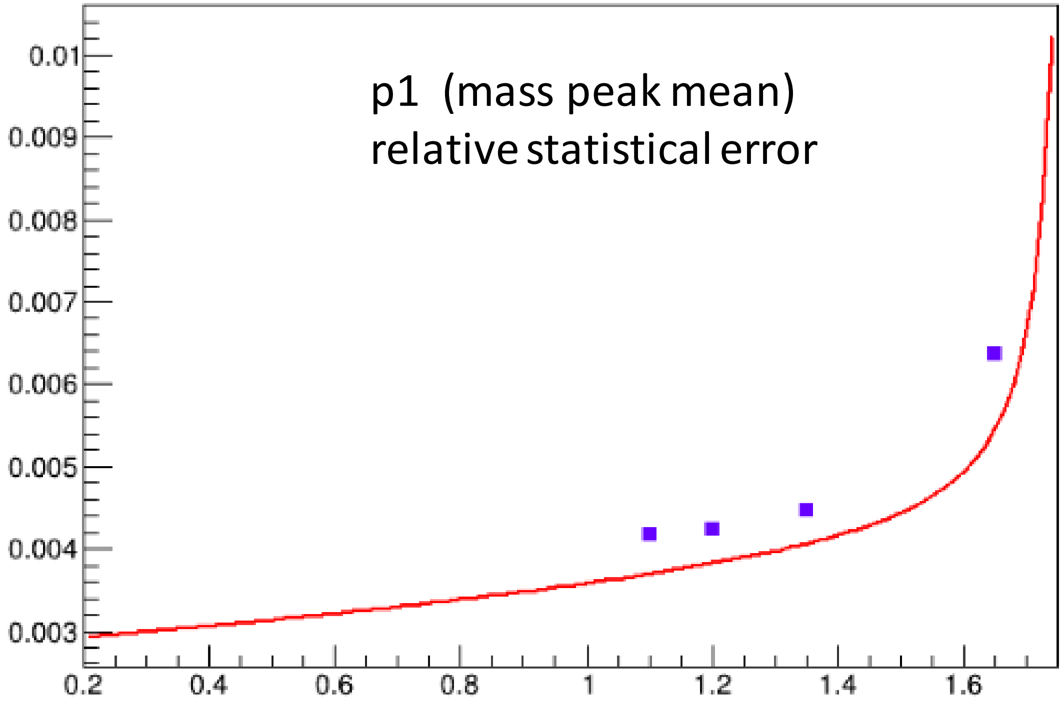
	rison of p0			Comparison of p1					
histo number	relerr3 = sqrt(S+B)/(S+B)	relerr4 = sqrt(S+2B) / S	relerr5 = sqrt(S+2B) / (S+2B)	relerr1_p1 = (p2/p1) * (1/sqrt(S))	rel_errfit = fit error on p1) / p1	relerr2_p1 = p2/p1)*(sqrt(S +B)/S)	relerr3_p1 = (p2/p1) * (sqrt(S+B)/(S+B))	relerr4_p1 = (p2/p1) * sqrt(S+2B)/S)	relerr5_p1 = (p2/p1) * (sqrt(S+2B)/(S+2B))
eta_25 (1.1, 1.1)	0.009970	0.043895	0.007638	0.0019744	0.0567241	.0036263059	0.00107499	.004733070	0.00082362
eta_25 (1.2, 1.2)	0.011793	0.045553	0.009137	0.0021144	0.0056165	.0036578294	0.00122224	.004721088	0.00094698
eta_25 (1.1, 1.3) ??	0.009970	0.043895	0.007638	0.0019744	0.0056724	.0036263059	0.00107499	.004733070	0.00082362
eta_25 (1.35, 1.35)	0.014904	0.046890	0.011789	0.0024238	0.0056957	.0038237317	0.00153646	.004833927	0.00121537
eta_25 (agg4)	0.033352	0.061947	0.028496	0.0043933	0.0075036	.0055344146	0.00348748	.006477524	0.00297971
eta_50 (1.1, 1.1)	0.007549	0.050817	0.005626	0.0017593	0.0067539	.0039405244	0.00078550	.005287739	0.00058537
eta_50 (1.2, 1.2)	0.008849	0.049880	0.006662	0.0018941	0.0064750	.0039019886	0.00091944	.005182992	0.00069219
eta_50 (1.1, 1.3) ??	0.007549	0.050817	0.005626	0.0017593	0.0067539	.0039405244	0.00078550	.005287739	0.00058537
eta_50 (1.35, 1.35)	0.011171	0.052735	0.008513	0.0021244	0.0067229	.0040292016	0.00112006	.005287341	0.00085354
eta_50 (agg4)	0.029617	0.062865	0.024720	0.0042102	0.0076105	.0056038257	0.00316321	.006714137	0.00264011
eta_75 (1.1, 1.1)	0.009381	0.037073	0.007253	0.0018497	0.0053252	.0032333813	0.00105816	.004181874	0.00081816
eta_75 (1.2, 1.2)	0.011078	0.038195	0.008678	0.0020205	0.0054065	.0033204617	0.00122948	.004238924	0.00096309
eta_75 (1.1, 1.3) ??	0.009381	0.037073	0.007253	0.0018497	0.0053252	.0032333813	0.00105816	.004181874	0.00081816
eta_75 (1.35, 1.35)	0.014028	0.040179	0.011219	0.0023581	0.0053513	.0035689605	0.00155800	.004462570	0.00124601
eta_75 (agg4)	0.034021	0.054416	0.029954	0.0047251	0.0069596	.0056074301	0.00398166	.006368645	0.00350575
eta_90 (1.1, 1.1)	0.013238	0.040645	0.010501	0.0022450	0.0048840	.0035034480	0.00143859	.004416818	0.00114110
eta_90 (1.2, 1.2)	0.014807	0.039247	0.011960	0.0023939	0.0049712	.0035025978	0.00163611	.004336561	0.00132147
eta_90 (1.1, 1.3) ??	0.013238	0.040645	0.010501	0.0022450	0.0048840	.0035034480	0.00143859	.004416818	0.00114110
eta_90 (1.35, 1.35)	0.018206	0.040847	0.015061	0.0028361	0.0054051	.0038638335	0.00208176	.004670638	0.00172216
eta_90 (agg4)	0.042258	0.064684	0.037569	0.0068702	0.0099632	.0080144949	0.00588922	.009014716	0.00523579

*Christian K. Boesing PHENIX thesis https://zenodo.org/record/4267413

First results

- **Ignoring energy dependence of peak sigma p2** (expected from energy resolution) by setting p2/p1 = const = 0.1 (true for all blue points), test minimal error from energy dependence of S and combinatoric B alone:
- First Results: No local minimum in E1, but error flattens as E1 decreases—**gain in statistical precision from going lower in E1 small, outweighed by fit robustness**
  - sqrt(S+2B)/S very good estimator of p0 error values
  - p2*sqrt(S+2B)/S*p1: ~correct shape for p1 error
- Lower than ~0.9 not trustable…we are studying effects of cuts (new pass with much lower min photon E2 cut), S/B functional forms, and energy resolution to push lower, see if local min

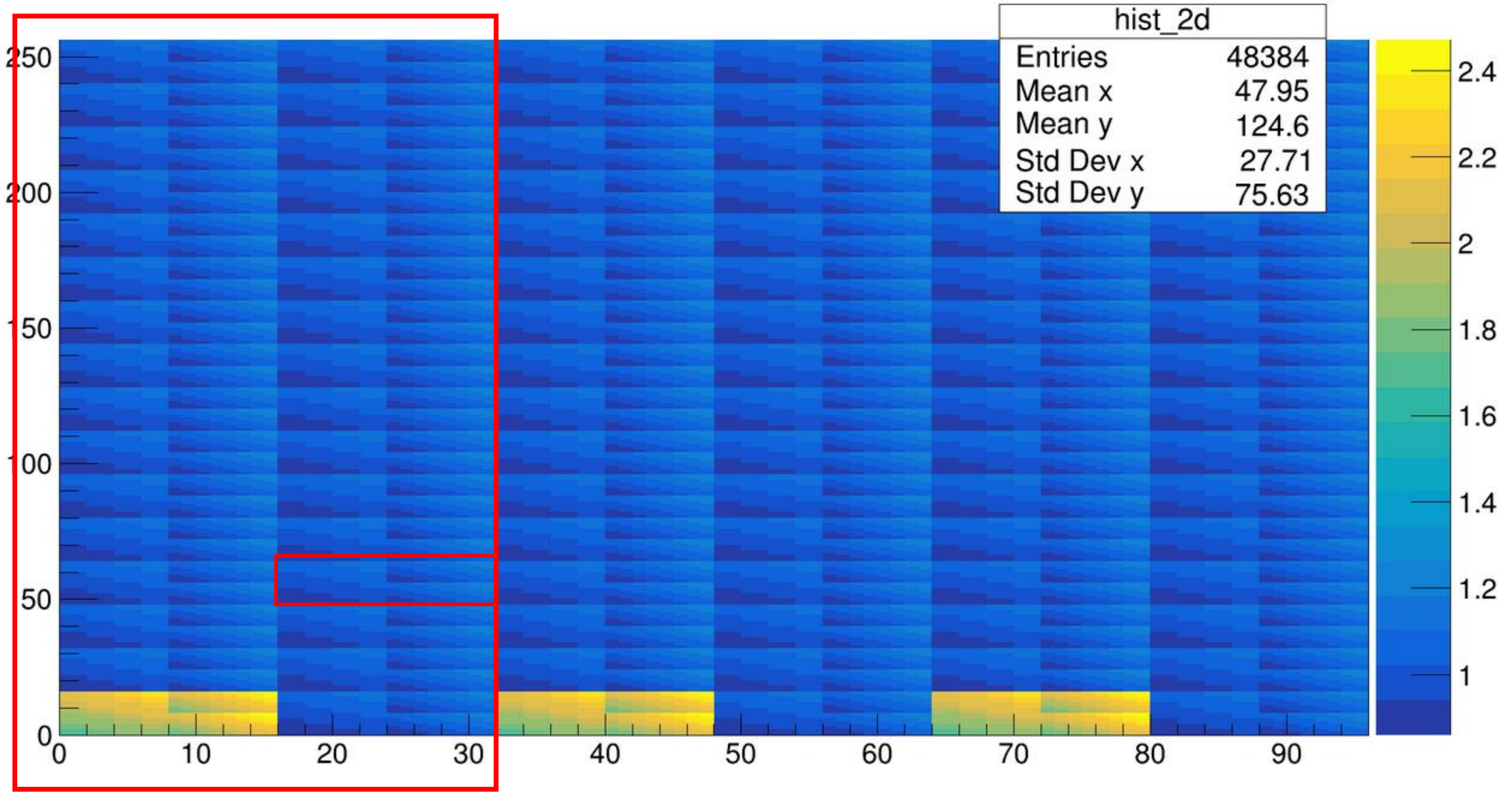




Backup

Adding 32 pieces into one location: peek on code

```
//________________________________________________________________..
void CaloCalibEmc_Pi0::Add_32()
{
  for (int ithirds=0; ithirds<3; ithirds++)
  {
        for (int ieta=0+ithirds*32; ieta<(ithirds*32+16); ieta++)
        {
            for (int iphi=0; iphi<16; iphi++)
            {
                for (int ipatt_eta=0; ipatt_eta<2; ipatt_eta++)
                {
                    for (int ipatt_phi=0; ipatt_phi<16; ipatt_phi++)
                    {
                        if ((ipatt_eta>0) || (ipatt_phi>0))
                        {
                          cemc_hist_eta_phi[ieta][iphi]->Add(cemc_hist_eta_phi[ieta+ipatt_eta*16][iphi+ipatt_phi*16]);
                        }
                    }
                }
            }
        }
  }
}

```


hist_2d
Entries 48384
Mean x 47.95
Mean y 124.6
Std Dev x 27.71
Std Dev y 75.63
0
50
100
150
200
250
0
10
20
30
40
50
60
70
80
90
1
1.2
1.4
1.6
1.8
2
2.2
2.4

```
#include <calib_emc_pi0/CaloCalibEmc_Pi0.h>
#include "GetTChainMacro.C"

void runLCELoop(int nevents = -1, const char *ifile="", const char *ofile="",const char *incorr="")
{
  //    gSystem->Load("libcalibCaloEmc_pi0.so");
  R__LOAD_LIBRARY(libcalibCaloEmc_pi0.so)
  CaloCalibEmc_Pi0 obj_LCE("CaloCalibEmc_Pi0", ofile);
  obj_LCE.InitRun(0);

  //  obj_LCE.Loop(nevents,ifile,0);

//  TTree * intree1 =  get_tchain();
  TTree * intree1 =  GetTChainMacro(ifile);

//          obj_LCE.Loop(nevents,"",intree1,"rerestart1_v7v1.root");
  obj_LCE.Loop(nevents, ifile, intree1, incorr);
  //----  obj_LCE.End(0);
  obj_LCE.End(0);
  //obj_LCE.FittingHistos();


}
```

::Loop macro

```
#include <calib_emc_pi0/CaloCalibEmc_Pi0.h>

void save_corr_value2(const char* ifile = "", const char* ofile = "",const char * infilent = "")
{
  R__LOAD_LIBRARY(libcalibCaloEmc_pi0.so)

  CaloCalibEmc_Pi0 calo_obj("CaloCalibEmc_Pi0" /* this name goes to SubsysReco  */, ofile );
  //  calo_obj.InitRun(0); // to declare the eta_hist[96]
  calo_obj.Get_Histos(ifile,ofile); // open the fun4al file and Get eta_hist[96] histos
  calo_obj.Add_96();
  //calo_obj.Add_32();
  calo_obj.Fit_Histos_Eta_Phi(infilent);  // do the fittings
  calo_obj.End(0);  // save the output file
}
```

::Fitting macro