Contents

1 Introduction

The charged-particle multiplicity measurement is a basic but very important analysis when studying the properties of heavy ion collisions and hence the forming quark gluon plasma (QGP). On a very basic level, the number of charged particles leaving the interaction region is counted. This measurement can be used to deduce properties of the QGP such as the energy density. This analysis follows two papers, which were performed in the past on different LHC datasets.

1. Charged-particle multiplicity measurement in proton-proton collisions at $\sqrt{s} = 0.9$ and 2.36 TeV with ALICE at LHC

2. Charged-particle multiplicity measurement with Reconstructed Tracks in pp Collisions at $\sqrt{s} = 0.9$ and 7 TeV with ALICE at the LHC

Additional information and detailed explanations of the physical background can be found in the following PhD Thesis:

3. Measurement of the Charged-Particle Multiplicity in Proton-Proton Collisions with the ALICE Detector

2 Background behind the analysis

2.1 Physical Quantities

As mentioned above, counting the charged particles is the main task of this analysis. This number is called charged-particles multiplicity and is denoted by N_{ch}. Because of the relativistic equivalence of mass and energy $E = mc^2$, N_{ch} is directly related to the energy distribution inside the QGP. The multiplicity is mostly studied as density with respect to the pseudorapidity $\eta = \ln(\tan(\frac{\theta}{2}))$, which is a pure geometric quantity as figure 1 illustrates. The resulting quantity is called multiplicity density $dN_{ch}/d\eta$ and represents the amount of particles in an certain η interval. In general, $dN_{ch}/d\eta$ can depend on η but in the following, the dependence is omitted in the notation.

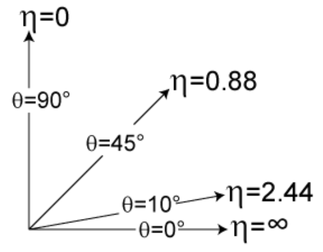


Figure 1: Illustration of the pseudorapidity η for a horizontal beamline; figure from Wikipedia

2.2 Normalization of $dN_{ch}/d\eta$

The multiplicity density can rely on different normalizations, which means, different types of events are considered in the measurement. The three types of

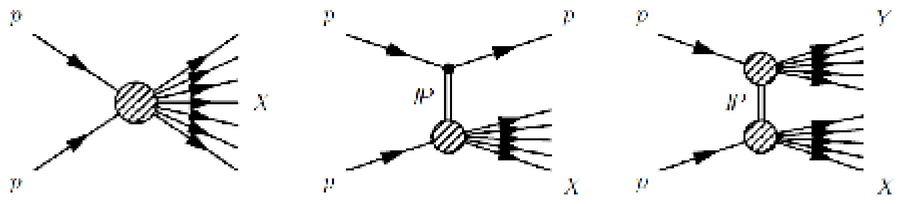


Figure 2: From left to right: ND, SD and DD; figure from InspireHEP

inelastic collisions are *non-diffractive (ND)*, *single-diffractive (SD)* and *double-diffractive (DD)*. Suggestively this corresponds to number of diffractive systems, which are created in the collision. A diffractive system is an excited incident particle, which carries the quantum numbers of the incoming particle. Figure 2 illustrates the phenomenological difference between the different types of events. The two most used normalizations are *total inelastic (INEL)* and *non-single-diffractive (NSD)*. The first one contains all inelastic events (ND, SD and DD) while the second one contains only symmetric events (ND and DD). For further information see [3].

2.3 The ALICE detector

ALICE (A Large Ion Collider Experiment), one of the four large experiments at the CERN Large Hadron Collider, has been designed to study heavy-ion collisions. It also studies proton-proton (pp) collisions, which primarily provide reference data for the heavy ion collisions. In addition, the pp-collision data allow for a number of genuine proton-proton physics studies.
The ALICE detector has been designed to cope with the highest particle multiplicities anticipated for collisions of lead nuclei at the extreme energies of the LHC. It is composed out of different sub-detectors. In the Central Barrel this are the Inner Tracking System (ITS), consisting out of three detectors having each 2 layers, namely the Silicon Pixel Detector (SPD), the Silicon Strips Detector (SSD) and the Silicon Drift Detector (SDD). The two endcaps of the ITS contain the VZERO trigger system. The ITS is followed radially by the Time Projection Chamber (TPC), the Transition Radiation Detector (TRD), the Time Of Flight detector (TOF), the High Momentum Particle Identification Detector (HMPID), the two electromagnetic calorimeters EMCAL and PHOS as well as ACORDE, the dedicated cosmic ray detector. In the following analysis the ITS as well as the TPC can be used. The TPC only covers the region $|\eta| < 0.8 \Leftrightarrow |\theta| > 48°$, so the analysis is focused mainly on this region.

2.4 The ALICE trigger system

In run 1 of the LHC, the spacing between two bunches was 50 ns with a peak luminosity of the order of $10^{33}\, cm^{-2}s^{-1}$. This led to large amounts of data, generated by the detectors, which had to be processed. As not all produced data is relevant for certain physics studies, LHC experiments have several detectors capable to act as fast trigger systems, sometimes in a correlated manner, and allowing to record mainly interesting data. For ALICE, the so called Minimum Bias trigger (MB) relies on an signal either in one of the VZEROs or in one detector of the ITS.

3 The $dN_{ch}/d\eta$ analysis

3.1 How to run the analysis

In order to run the analysis, it is recommended to use the masterclass GUI. Select the tab *dNdEta Analysis*, chose one of the datasets with a desired number of files and click on the picture. All necessary data files will be downloaded and their paths are collected inside four text files: *files.txt, filesPerugia0.txt, filesPerugia2011.txt and filesPhojet.txt*, one for the data and three for the Monte Carlos respectively. In order to run the analysis manually, just execute the Root macro *dNdEta.C* by the following line, if the mentioned text files are present:

```
aliroot dNdEta.C
```

A detailed explanation of the workflow is given in 3.4 and 3.5. The main settings of the analysis can be changed as described below. When the analysis in finished, two windows open. One contains the main result, the comparison of the data with predictions, the other is a Root browser, in which all relevant files and therefore all plots are loaded. For further details see 3.6 and 3.7.

Analysis mode

The analysis mode determines, which detector parts shall be used. By default this is TPC and ITS. The active parts can be chosen by comment in/out the lines $41 - 42$ in *run.C*:

```
41   // AliPWG0Helper::AnalysisMode analysisMode =
         AliPWG0Helper::kSPD | AliPWG0Helper::kFieldOn;
42   AliPWG0Helper::AnalysisMode analysisMode = AliPWG0Helper::
         kTPCITS | AliPWG0Helper::kFieldOn;
```

Trigger mode

The trigger mode, which is used for the analysis, can be chosen in *run.C* in lines $47 - 48$:

```
47   AliTriggerAnalysis::Trigger trigger      =
         AliTriggerAnalysis::kAcceptAll | AliTriggerAnalysis::
         kOfflineFlag;
48   // AliTriggerAnalysis::Trigger trigger      =
         AliTriggerAnalysis::kAcceptAll | AliTriggerAnalysis::
         kOfflineFlag | AliTriggerAnalysis::kOneParticle;
```

Normalization

The normalization can be chosen in *correct.C* by commenting in/out one of the block in lines 34 – 90, e.g. for INEL:

```
45
46    // All INELastic events
47    file->cd();
48    dNdEtaAnalysis* fdNdEtaAnalysis = new dNdEtaAnalysis("
          dndeta", "dndeta");
49    fdNdEtaAnalysis->LoadHistograms("fdNdEtaAnalysisESD");
50    fdNdEtaAnalysis->Finish(dNdEtaCorrection, 0.151,
          AlidNdEtaCorrection::kINEL, "ESD -> full inelastic",
          backgroundEvents);
```

Low p_T cutoff

The low transverse momentum cutoff is also defined in *correct.C*. It is the second argument, when calling *Finish* in the uncommented block, e.g. here it is set to 0.15:

```
48    dNdEtaAnalysis* fdNdEtaAnalysis = new dNdEtaAnalysis("
          dndeta", "dndeta");
```

3.2 AliRoot

The $dN_{ch}/d\eta$ analysis is using the AliRoot framework, which contains the software libraries allowing to to perform detector simulation, reconstruction and physics analysis for the ALICE collaboration. The main workflow is event based and modular. This means each physics interaction event is processed separately and can be looped through several tasks. This leads to a very convenient way of plugging an analysis together while reducing the amount source code as well as its complexity. The single tasks are contained in one of the two main parts of the AliRoot framework, AliCore and AliPhysics. The first one contains basic tasks, e.g. for event processing or simulations, while the latter one contains mainly tasks for physical analysis. The tasks are chained together in a so called train model using a Root macro. The configuration to be defined by the user contains a set of environment variables, input and output data containers as well as the connection between the different analysis tasks.

3.3 Workflow of the analysis

As described in the previous section, the analysis is split into several tasks and steering macros. The top-level macro *dNdEta.C* controls the main workflow, which looks as follows:

1. Run the analysis on simulated Monte Carlo data

2. Run the analysis on real data
3. Correct raw results with MC
4. Plot the final results as well as intermediate steps

The first step is to predict the expected results by using Monte Carlo simulations. For this analysis, there are three different MC datasets available, the first one is *Pythia* with tune *Perugia0*, the second one is the same with rune *Perugia2011* while the third one is *Phojet*. Both datasets are processed with the macro *run.C*, which is explained in detail below:

```
13      run(2, 1, "filesPerugia0.txt");
22      run(2, 1, "filesPerugia2011.txt");
31      run(2, 1, "filesPhojet.txt");
```

For each of this runs, the result is both, a prediction of what will be measured (*analysis_mc.root*) based on the MC truth, as well as a correction map (*correction_map.root*), which can be used to correct the measured data with respect to detector effects like geometry or efficiency. After each run, the results are saved in separate folders (*perugia0. perugia2011* and *phojet*).
Afterwards, the measured data from the ALICE detector is analyzed. This is done with the same script as for the MC, but the details differ a little bit, see below:

```
31      run(2, 1, "filesPhojet.txt");
```

Of course, there is no correction map created in this run. Furthermore, the results are raw and are not corrected for the mentioned detector effects (*analysis_esd_raw.root*).
This correction are applied in the third step of the analysis. There are three different correction maps for each of the different MC generators. Thus, the raw data is corrected three times, once for each generator (lines 50, 59 and 68). The results (*analysis_esd.root*) are saved in the folders of the MC.
The last step of the analysis and the second half of the macro *dNdEta.C* (l. 56 ff) contains the plotting of intermediate steps and the final result of the analysis, which is a comparison between the MC prediction and the measured data. This is the key to determine, if the underlying models of the MC and therefore the understanding of the QGP is correct or if the models have to be modified are replaced. Nevertheless, the intermediate plots are as important as the final result because they deliver information about the correctness of the analysis as well as the quality of the measurement.

3.4 The data analysis

Following the workflow of the $dN_{ch}/d\eta$ analysis, *run.C* is the first macro, which directly uses AliRoot on the data. As described above, the macro is used for analyzing the MC as well as the measured data. It can be used in two different modes, one for MC and one for data.

In the first case, one wants to run the analysis task itself as well as the correction task. This is done by setting the first argument $runWhat = 2$. Furthermore one indicates, that the data contains MC information by calling the macro with $requiredData = 1$. The third argument is a text file, which contains paths to the data files. The last argument is optional and can contain some additional settings for the tasks (for further information see AliRoot). Thus, the calls for the MC are as seen above. For the data analysis, the macro call is slightly different. One only wants to do the analysis, not the correction task, so $runWhat = 0$. Furthermore, the required data is raw data $requiredData = 2$. The text file is modified respectively.
The first lines of the macro (l. $22 - 33$) are setting the basic structure of the analysis.

```
22    // add aliroot include path
23    gROOT->ProcessLine(Form(".include %s/include", gSystem->
          ExpandPathName("$ALICE_ROOT")));
24    gROOT->SetStyle("Plain");
25    // analysis manager
26    AliAnalysisManager* mgr = new AliAnalysisManager("dNdEta")
          ;
27    // create the alien handler and attach it to the manager
28    AliAnalysisGrid *plugin = CreateAlienHandler(file);
29    mgr->SetGridHandler(plugin);
30    // set input handler as standard input handler and
          deactivate unnecessary detector parts
31    AliESDInputHandler* iH = new AliESDInputHandlerRP();
32    iH->SetInactiveBranches("FMD AliRawDataErrorLogs
          CaloClusters Cascades EMCALCells EMCALTrigger ESDfriend
           Kinks MuonTracks TrdTracks");
33    mgr->SetInputEventHandler(iH);
```

This contains technical features like defining include paths, as well as the initialization of the analysis manager, which contains all tasks and settings. Afterwards, a handler is defined, which forces the analysis to run on the local machine. Further details can be found at the end of the macro, where the functionality is explained in the comments:

```
165  AliAnalysisGrid* CreateAlienHandler(const Char_t* file)
166  {
167    AliAnalysisAlien *plugin = new AliAnalysisAlien();
168    // Set the run mode (can be "full", "test", "offline", "
           submit" or "terminate")
169    // "test" let the analysis run on the local machine
170    plugin->SetRunMode("test");
171
172    // Plugin test mode works only providing a file containing
            test file locations.
173    // The file should contain paths to local files like "*
           ESDs.root" etc.
174    plugin->SetFileForTestMode(file);
175   Int_t nFiles = gSystem->GetFromPipe(Form("grep -c \".\" %s
          ", file));
176   plugin->SetNtestFiles(nFiles);
177   return plugin;
178 }
```

The last three lines of this paragraph contain the definition of the input file handler. Especially line 32 is important, because there the unused parts of the detector deactivated for the analysis.
Lines 35 – 53 are one of the most important part of the analysis. Firstly, it is defined, which parts of the detector shall be used. For this analysis, either only the ESD is used or the whole ITS together with the TPC. By default, both systems are used. In order to use only the SPD, one has to comment line 41 in and 42 out:

```
41    AliPWG0Helper::AnalysisMode analysisMode = AliPWG0Helper::
          kSPD | AliPWG0Helper::kFieldOn;
42    // AliPWG0Helper::AnalysisMode analysisMode =
          AliPWG0Helper::kTPCITS | AliPWG0Helper::kFieldOn;
```

The second important option is the offline trigger. This is an additional trigger to the minimum bias trigger, which is applied online during data taking. There are two options available:

kAcceptAll No further trigger is applied to the data

kOneParticle At least one particle in the SPD is required

As seen above, on has to comment in the desired setting, e.g. to accept all particles:

```
47    AliTriggerAnalysis::Trigger trigger      =
          AliTriggerAnalysis::kAcceptAll | AliTriggerAnalysis::
          kOfflineFlag;
48    // AliTriggerAnalysis::Trigger trigger      =
          AliTriggerAnalysis::kAcceptAll | AliTriggerAnalysis::
          kOfflineFlag | AliTriggerAnalysis::kOneParticle;
```

The last definition of the variable *diffTreatement* is more technical and is used for the MC analysis.
Lines 55 – 73 contain another important setting of the analysis, the track cuts.

```
59    AliESDtrackCuts* esdTrackCuts = 0;
60    if (!(analysisMode & AliPWG0Helper::kSPD))
61    {
62      // execute macro CreateStandardCuts.C
63      gROOT->ProcessLine(".L CreateStandardCuts.C");
64      esdTrackCuts = CreateTrackCuts(analysisMode);
65      // print if an error occured
66      if (!esdTrackCuts)
67      {
68        printf("ERROR: esdTrackCuts could not be created\n");
69        return;
70      }
71      // Enable histograms for cuts in order to controll them
           after the analysis
72      esdTrackCuts->SetHistogramsOn(kTRUE);
73    }
```

These cuts are important to ensure a certain quality for the tracks of particles, which are used in the analysis. For the default analysis with TPC and ITS, the ALICE standard cuts from 2010 are used. For SPD only, no cuts are required. For further information look into *CreateStandardCuts.C* or at ALICE Track Selection.
Lines 75−78 are adding the first task to the analysis, the physical event selection. Afterwards, lines 80 − 135 are adding the main tasks of the analysis. This depends on the given data and is controlled by the given value of $runWhat$, see above. Both definitions are similar, so only one will be explained in detail:

```
84    if (runWhat == 0 || runWhat == 2)
85    {
86      task = new AlidNdEtaTask(option);
87
88      if (requiredData == 1) task->SetReadMC();
89      // physicsSelectionTask->GetPhysicsSelection()->
           SetBin0Callback("AlidNdEtaTask");
90
91      // INEL>0 definition
92      if (trigger & AliTriggerAnalysis::kOneParticle) task->
           SetMultAxisEta1();
93      task->SetTrigger(trigger);
94      task->SetAnalysisMode(analysisMode);
95      task->SetTrackCuts(esdTrackCuts);
96      // task->SetDeltaPhiCut(0.064);
97      task->SetDiffTreatment(diffTreatment);
98
99      // add task to analysis chain
100     mgr->AddTask(task);
101     // set output root file name for different analysis
102     TString outfilename = "results.root";
103     // Attach input
104     mgr->ConnectInput(task, 0, mgr->GetCommonInputContainer
           ());
105     // Attach output
106     cOutput = mgr->CreateContainer("cOutput", TList::Class()
           , AliAnalysisManager::kOutputContainer);
107     mgr->ConnectOutput(task, 1, cOutput);
108   }
```

Firstly, the task is defined. Lines 88 − 97 are applying the previously defined options, such as the trigger, to the task. Afterwards, lines 99 − 107 add the task to the manager, set output filename and connect in- and output.

In the next two sections, some final settings are set. Lines 137 – 144 are used only for MC runs while lines 146 – 154 are loading the magnetic field from *OCDB.root* file, which is the Offline Condition DataBase.

```
146   // Load the magnatic field from the OCDB file of the run
147   AliCDBManager *cdb = AliCDBManager::Instance();
148   cdb->SetDefaultStorage("local://");
149   cdb->SetRun(0);
150   cdb->InitFromSnapshot("OCDB.root");
151   AliGRPManager *grp= new AliGRPManager();
152   printf("#### Loading GRP to init B-field...\n");
153   if(!grp->ReadGRPEntry()) AliFatal("Cannot get GRP entry");
154   if(!grp->SetMagField())  AliFatal("Problem with magnetic
          field setup");
```

Finally, the analysis can be started:

```
156   // Initialize the analysis
157   if (!mgr->InitAnalysis()) return;
158   mgr->PrintStatus();
159   // And start it!
160   Printf("Starting Analysis....");
161   mgr->StartAnalysis("local");
```

3.5 The correction task

After MC and data are analyzed, the correction task must be performed. For that, three files must be in the same folder:

- *analysis_mc.root*
- *correction_map.root*
- *analysis_esd_raw.root*

The correction task is also controlled by a macro, *correct.C*, which takes the above mentioned files as standard input parameters. The core algorithm is *AlidNdEtaCorrection*, which is intantiated and configured in lines 4 – 10:

```
 4   AlidNdEtaCorrection* dNdEtaCorrection = 0;
 5   if (correctionMapFile)
 6   {
 7     dNdEtaCorrection = new AlidNdEtaCorrection(
           correctionMapFolder, correctionMapFolder);
 8     if (!TFile::Open(correctionMapFile)) return;
 9     dNdEtaCorrection->LoadHistograms();
10   }
```

After defining the background, the correction can be done. The correction can be done for different normalizations (see above) and the different trigger sets. As an example, the INEL correction is explained in detail, the others can be commented out as needed:

```
  // All INELastic events
  file->cd();
  dNdEtaAnalysis* fdNdEtaAnalysis = new dNdEtaAnalysis("
      dndeta", "dndeta");
  fdNdEtaAnalysis->LoadHistograms("fdNdEtaAnalysisESD");
  fdNdEtaAnalysis->Finish(dNdEtaCorrection, 0.151,
      AlidNdEtaCorrection::kINEL, "ESD -> full inelastic",
      backgroundEvents);
  file2->cd();
  fdNdEtaAnalysis->SaveHistograms();
```

Firstly, an analysis object is created with a suggestive naming. The third line is the call of the function *Finish*, which then performs the correction. Its arguments are as follows:

1. The correction object
2. The pT cut which has to be used
3. The kind of correction, which has to be performed
4. A naming tag for the correction
5. The number of background events

Afterwards, the histograms after the correction are saved into the output file, which is by default *analysis_esd.root*. As mentioned above, the correction is done with all MC datasets. Hence, there are three potentially different corrected versions of the data available. These fluctuations are studied in context of systematic errors and can reach up to 4.1% in total, which is used here. If the amount of used MC files is small, these fluctuations can be much larger. Nevertheless, in the rest of the analysis, an arithmetic average of both corrections is used as single result.

3.6 Results...

After the correction is done, the analysis is finished. The macro *dNdEta.C* produces now a summary and comparison plot between the MC and the data. Such a plot can look like 3. The plot is divided into two parts. The upper part shows the analyzed data as black points with error bars and with a gray error band. The error bars are due to statistical fluctuations. In figure 3, the errors on $dN_{ch}/d\eta$ are too small to be visible while the error on η is due to the binning of the data. The gray band represents the mentioned systematic errors.

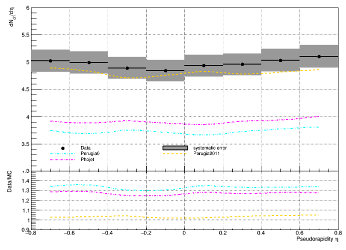


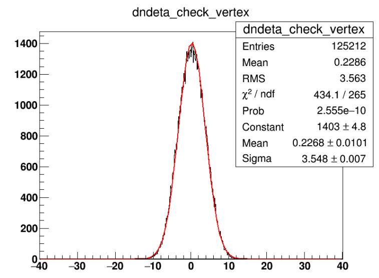
Figure 3: Example results of the $dN_{ch}/d\eta$ analysis

As mentioned above, the amount of used MC files has a large influence on the correction. If only a few files are used, the visible error band is too small and does not represent the correct error estimation. Additionally to the data, the predictions of the three MC generators are plotted in the same diagram as colored lines. Their statistical errors are omitted here for simplicity but can be estimated by the change of the curvature and the asymmetry around $\eta = 0$. The lower plots show explicitly the difference between MC and data as a data/MC ratio. For accurate MC, this would be a straight line at 1.

3.7 ...and intermediate steps

Besides the final results, intermediate steps are interesting to control the single steps of the analysis as well as their quality. First of all, every plot is available in the concerning Root-files, which are produced during the analysis. After the analysis is finished, a Root browser is opened, in which the Root-files are already loaded. They can be found in the file tree on the left side in the folder *ROOT Files*, which is the third from above. In the following part, some interesting intermediate plots are discussed. The headers represent the file and the path to the plot inside, followed by a description of it.

data/analysis_esd_raw.root/dndeta_check_vertex



The plots contains the reconstructed vertex positions of all events along the beam line axis in cm. This is an important criteria to check the quality of the reconstruction. The distribution should follow a Gaussian with $\mu \approx 0$. This can be tested by fitting such a distribution. In order to do so, right-click on the black curve of the histogram, click *Fit*, type *gaus* into the field *formula* and press OK. The width of the distribution is $\sigma \approx 3.5\,\text{cm}$ and the mean is $\mu < 0.3\,\text{cm}$. This is an intrinsic property of the beam at the interactions point of ALICE.

data/analysis_esd_raw.root/fMult(Vtx)

The two plots show the multiplicity distribution in the raw data. The first one (fMult) contains all events while the latter one (fMultVtx) contains only those events, which have a reconstructed vertex. Normalized to 1, this is equal to the multiplicity probability density $P(N_{ch})$, which is analyzed in the same paper as $dN_{ch}/d\eta$ and contains information about how probable a certain multiplicity is. This density is another basic property to study the QGP.

data/analysis_esd_raw.root/dndeta_check_0

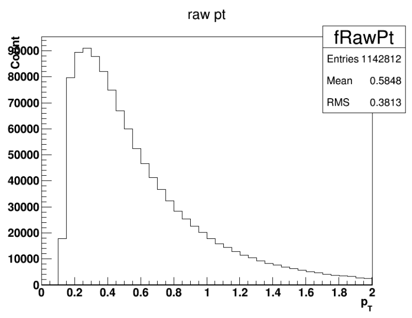
This is the raw distribution of $dN_{ch}/d\eta$. It is not corrected for any detector effects and can contains asymmetries and other effects. This distribution has to be corrected by using the MC generators, which simulates the whole detector response.

data/analysis_esd_raw.root/fPhi

The plots shows the distribution of reconstructed tracks in ϕ-direction, which is the plane orthogonal to the beam line. In general, physics must be the same in ϕ, the experiment is symmetric around the beam axis. Thus, a flat distribution is expected. In reality, fluctuations or even dips can be observed because the detector is not perfectly symmetric or even some parts of the detector can be damaged or offline.

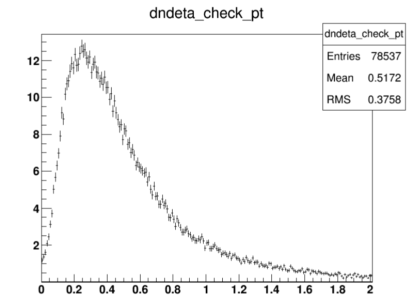
data/analysis_esd_raw.root/fRawPt

This histogram contains the distribution of the transverse momentum of produced particles p_T [GeV/c]. It can be useful to estimate a low-p_T-cutoff, which is used in the analysis.



***/analysis_mc.root/dndeta_check_pt**

This is the same as above but for the MC dataset, which is chosen in the path (*perugia0*, *perugia2011* or *phojet*). The combination of this with the distribution above can be used to estimate a low-p_T-cutoff, which has to be applied to the data. This cutoff is important because the ALICE detector is not sensitive to low p_T particles and a cutoff therefore improves the quality of the analysis. The default value is $p_T > 0.15\,\mathrm{GeV/c}$ above which the distributions are compatible.



***/analysis_mc.root/dndeta*/dNdEta**

The plot shows the observed $dN_{ch}/d\eta$ distributions of the MC simulations. The first * in the path stand again for the MC tool, while the second * indicates a special normalization/selection. The distribution is in general symmetric around $\eta = 0$ within its statistical uncertainty.

***/correction_map.root/fEtaCorrelation**

This 2D-histogram show the correlation between the generated η (x-axis) and the simulated measurement of η in the detector (y-axis). This distribution should be as close and as sharp as possible to a line through the origin in order to make a precise measurement.

***/correction_map.root/fEtaResolution**

Another form of representation of the above distribution is given in this plot. It is an histogram of the difference between generated η and its simulated measurement. For a perfect detector, this would be a infinite sharp peak at exactly 0, but in reality this is always smeared out.

/correction_map.root/fpT

In analogy to the two plots above, the generated and the simulated measurement of p_T can be compared with the plots *fpTCorrelation* and *fpTResolution*. The first one is again a 2D-histogram while the second one represents the relative deviation (y-axis) in dependence of the generated p_T (x-axis). This distribution should also be as narrow as possible to 0.

***/analysis_esd.root/dndeta*/dNdEta_corrected**

This is the final result of the analysis with one specific MC. The second * in the path represents again one specific normalization/selection. This histogram is used in order to calculate the average of the different MC and also to estimate the systematic uncertainty coming from the MC. The latter one can be estimated by comparing this plot for the different available MC generators.

***/analysis_esd.root/dndeta*/Pt**

The final distribution of transverse momenta can be obtained in this histogram. In contrast to the raw histograms of p_T, there are no entries below the cutoff, which has been applied during the analysis.