Annex 1 - Computer Code

Associated with the Bachelor's Thesis *Studying Elliptic Flow in High Centrality Pb-Pb Collisions using Proton Collision Background Fitting* by Philip J. Fredholm

This annex contains all of the computer code used. Please note that all of this is also available at https://github.com/philipjfredholm/BachelorsProject at the time of writing which might provide a more convenient reading experience. Below is a list of all of the files used, the forward slash '/' denotes a file being stored in a subdirectory. Furthermore, in the file 'README.md' there is a part named *[ADD LINK HERE]*. It is required that the this report is uploaded to 'LUP Student Papers portal' on the internet after it has been passed and it is my intent to update that part of the file on GitHub with a link to this thesis. Since I of course have to hand this report in before I can upload it, the *[ADD LINK HERE]* is left as a place holder. However, if you are reading this you of course already have access to the thesis.

The purpose of the various files are explained in the file `README.md`. Note that because the LaTeX-formatting not perfect, there might be some incorrect line breaks in the presentation below.

- Makefile
- README.md
- combine.cpp
- dataFitter.cpp
- dataFitterVisualiser.cpp
- entries.cpp
- include/AliLWUtils.h
- include/LinkDef.h
- include/storeInHist.h
- protonSubtracter.cpp
- raw/AliLWUtils.cxx
- raw/AliLWUtils.h
- readData.cpp
- resultsPlotter.cpp
- runProgram.sh
- src/AliLWUtils.cxx
- src/storeInHist.cxx

Makefile

```
#C++
CXX = g++
CXXVER = #Handled by ROOT-flags below, at least C++17 is required for everything (including combine.cpp) to work
CXXWARS = -Wall -Wextra -Werror

#Linking
ROOT = `root-config --glibs --cflags --libs` #Takes care of ROOT-dependencies
ROOTYS = `root-config --incdir`
INCLUDES = -I include -I $(ROOTSYS)/include
LIBS = -L . -Wl,-rpath=.

#Shortcuts
RUN = $(CXX) $(CXXVER) $(CXXWARS) $(INCLUDES)
RUNL = $(CXX) $(CXXVER) $(CXXWARS) $(INCLUDES) $(LIBS)

#Needed for ROOT
HEADERS = include/AliLWUtils.h include/storeInHist.h
SOURCES = src/AliLWUtils.cxx src/storeInHist.cxx


#I think the 'root-config --incdirs' command is bugged. In the terminal it gives the path to the include/ directory of the ROOT
#installation, but in this Makefile it just gives the path to the ROOT-installation, hence the
#$(ROOTSYS)/include in the actual commands. I suspect that this is a bug, and if it is patched
#in the future it might cause issues.


#Commands for running the analysis
all: combine.cpp readData.cpp AliLWUtils.o storeInHist.o rootDict.cxx libROOTlibs.so
	make readData
	make entries
	make dataFitter
	make dataFitterVisualiser
	make combine
	make resultsPlotter
	make protonSubtracter


#The "make combine" above might need to be commented out on machines which do
#not have compilers which C++17. ./combine is quite fast so it may run
#on a laptop without problem even if the workstation is old.

```

```
readData: readData.cpp AliLWUtils.o storeInHist.o rootDict.cxx libROOTlibs.so
	$(RUNL) -o readData readData.cpp AliLWUtils.o storeInHist.o -lROOTlibs $(ROOT) -O3

combine: combine.cpp AliLWUtils.o storeInHist.o rootDict.cxx libROOTlibs.so
	$(RUNL) -o combine combine.cpp AliLWUtils.o storeInHist.o -lROOTlibs $(ROOT) -O3

resultsPlotter: resultsPlotter.cpp AliLWUtils.o storeInHist.o rootDict.cxx libROOTlibs.so
	$(RUNL) -o resultsPlotter resultsPlotter.cpp AliLWUtils.o storeInHist.o -lROOTlibs $(ROOT) -O3

protonSubtracter: protonSubtracter.cpp AliLWUtils.o storeInHist.o rootDict.cxx libROOTlibs.so
	$(RUNL) -o protonSubtracter protonSubtracter.cpp AliLWUtils.o storeInHist.o -lROOTlibs $(ROOT) -O3



entries: entries.cpp AliLWUtils.o rootDict.cxx libROOTlibs.so
	$(RUNL) -o entries entries.cpp AliLWUtils.o -lROOTlibs $(ROOT) -O3

dataFitter: dataFitter.cpp AliLWUtils.o storeInHist.o rootDict.cxx libROOTlibs.so
	$(RUNL) -o dataFitter dataFitter.cpp AliLWUtils.o storeInHist.o -lROOTlibs $(ROOT) -O3


dataFitterVisualiser: dataFitterVisualiser.cpp AliLWUtils.o storeInHist.o rootDict.cxx libROOTlibs.so
	$(RUNL) -o dataFitterVisualiser dataFitterVisualiser.cpp AliLWUtils.o storeInHist.o -lROOTlibs $(ROOT) -O3



#Object Files
AliLWUtils.o: include/AliLWUtils.h src/AliLWUtils.cxx
	$(RUN) -c include/AliLWUtils.h src/AliLWUtils.cxx $(ROOT)


storeInHist.o: include/storeInHist.h src/storeInHist.cxx
	$(RUN) -c include/storeInHist.h src/storeInHist.cxx $(ROOT)



#ROOT Compatibility Things

```

```
#I have no idea what -p does, but just included it as the ROOT documentation said to do so.
#From what I can see the flag is not listed on rootcling:s man-page.

# $@ refers to the name of the Make-command and $^ refers to the listed dependencies of said Make-command

#-fPIC seems to do something about "position indepent code". I don't know why it is included,
#but the compiler gives me an error message and tells me to include it when the command is ran without it.

rootDict.cxx: $(HEADERS) include/LinkDef.h
   rootcling -f $@ -c -p $^



#It would be nice to place this inside ./libs/ , but the way the dictionary is created with paths
#that does not work.
libROOTlibs.so: rootDict.cxx $(SOURCES)
   $(CXX) -shared -o$@ -I $(ROOTSYS)/include $^ $(ROOT) -fPIC



#Cleaning Options
clean1:
   rm *.o
   rm include/*.gch
   rm rootDict.cxx
   rm *.pcm
   rm *.so


clean2:
   rm readData
   rm combine
   rm dataFitter
   rm dataFitterVisualiser
   rm protonSubtracter
   rm resultsPlotter
   rm entries


clean:
   make clean1
```

```
124     make clean2
125
126
127
128
129
130 #Leftovers from earlier experimenting
131 test: test.cpp AliLWUtils.o storeInHist.o rootDict.cxx libROOTlibs.so
132     $(RUNL) -o test test.cpp AliLWUtils.o storeInHist.o -lROOTlibs $(ROOT) -O3
133     clean1
134
135
136
137 .PHONY: all clean1 clean2 clean
```

README.md

```
# Philip Fredholm's Bachelor's Project
* Requirements:
    * You must have ROOT installed (tested with v6.26/10) and have sourced the
      'thisroot.sh'-file.

    * You must have Make installed.

    * You must have at least C++17 available in your compiler (your
      ROOT-compilation must also be C++17 compatible).




# Instructions
This repository contains the files used in my (Philip Fredholm's) bachelor's
    project. For the proper context of what all the things do, please see the
    thesis/report available at [ADD LINK HERE]. The
original files which I was given were ROOT-files storing ROOT-trees according to
    the structure shown in Figure 7 of the report. The various objects were
    declared and implemented according to the files in the folder `raw`. The end
    output is the figures seen in the report. Here, I outline the general steps
    taken to produce these plots.

1. First, the command `make all` should be entered into the terminal. This will
    create all the necessary executables according to the naming scheme that a
    file called i.e. `combine.cpp`
has its executable simply named `combine`. Then, the script `runProgram.sh`
    should be used to begin processing data. It takes two arguments; the first of
    which is the path to the folder
which contains **only** the previously described ROOT-files. Note that it is
    important that there is a forward slash '/' at the end of the path, e.g.
    `/path/to/my/file/` and not `/path/to/my/file`.
The second argument is the number of cores that should be used. Note that due to
    implementation reasons, one should be careful with not using an excessive
    number of cores for files containing a very small
number of events. The output will be stored in a folder in the present working
    directory called `processedData`.

   Note that `runProgram.sh` is mostly just a wrapper. It simply starts multiple
      instances of the program associated with `readData.cpp`. This is in turn
      also a sort of wrapper. Since the code for data processing got somewhat
      complicated, the processing is done by instantiating an instance of a
      class called `storeInHist` (defined in `include/storeInHist.h` and
      implemented in `src/storeInHist.cxx`) which's constructor does all of the
      data processing and stores the results in a file. It also has some other
      functionality used later, see the actual code files for that. However, it
```

is important to note that while some parameters are defined in ‘readData.cpp‘, a lot are defined in the primary function ‘loadHistograms()‘ in the implementation of the ‘storeInHist‘ class. Some variables are even overridden so it is important that variables are changed in **both** places to get the correct results. The reason for this is that it was necessary during the development process for backwards compatibility reasons and due to time constraints, it has not been a priority to update it (since there are a lot of dependencies on these things in ‘src/storeInHist.cxx‘ it would take a lot of work).

2. Since ‘runProgram.sh‘ creates one file per core used, and since results from running the script on different files also creates different files, the data must be merged together. The file ‘combine.cpp‘ handles this. It also fills a secondary purpose in being able to plot one example out of the files combined. Note that some plotting options will have i.e. titles and axis labels and these are not updated automatically and must be changed manually. The arguments that ‘combine.cpp‘ takes are the following; firstly the path to the folder containing **only** the files to be combined (this time it is without the forward slash ’/’ at the end so be careful with the syntax), secondly which correlations are to be plotted where ‘none‘ only does the combining and no plotting (see the actual code for the different options), tertiary a drawing option which is passed the ‘Draw()‘ method of the ROOT class ‘TH2D‘ (i.e ‘colz‘ or ‘surf1‘), the fourth argument is the index of p_T (p_T = transverse momentum) interval defined in the std::vector<double> array in the ‘loadHistograms()‘ method and the final argument is ditto for the centrality interval. The program then outputs the combined results into a field called ‘totalDataProcessed.root‘.

Note that there are some other peculiarities about the ‘combine.cpp‘ file. One is that due to time efficiency, the program checks if there is already a file called ‘totalDataProcessed.root‘ before beginning the combining process. If it exists, it opens that file instead of performing the combining at the entered file path. This was implemented so that the plotting functionality could be used instantly without having to wait for the combining process every time one wanted to see a different plot. The second peculiarity is a bug, which I suspect is related to ROOT. Regardless of cause, the issue is that when combining files of i.e. 2 GB, a computer with 12 GB of RAM somehow manages to run out of memory. I have looked through my code for a possible cause of this bug but I have been unable to find it. I suspect that it has to do with ROOT. Customarily, ROOT is not used in a stand-alone mode but with its own custom interpreter. Typically, a lot of ROOT code is written with a lot of raw pointers placing things on the heap. When using ROOT with its own interpreter, it cleans up things behind the scenes to prevent memory leaks. I discovered early on in the project that there is definitely no such clean up behind the scenes going on in the stand-alone C++ version, so one has to be careful to not throw around pointers carelessly. I suspect that the issue is due to some internal implementation of ROOT

objects expecting this 'magic clean up' of the compiler to happen and when it does not in the stand-alone version, a memory leak occurs. The solution is simply to split up the file one wishes to combine into different subfolders and combine those separately. Then, the 'totalDataProcessed.root' files may be renamed and placed in another folder, on which 'combine.cpp' may be used again to combine these files.

3. The next step is to calculate v_2 as described in the report. This is done using the file called 'dataFitter.cpp'. It takes two arguments, which are the paths to the combined data from the lead-lead data and the proton-proton data. Note that as described in the report, the previous steps have to be repeated so that there is both a file for the lead-lead data and the proton-proton data for this to work. Note that this file also computes the proper uncertainties, which uses the method described in the report. The output file is a file called 'results.root'.

   Note that one very important step here is that the p_T and centrality intervals which were defined in the vectors in 'loadHistograms()' have to be manually changed to the correct values in the code before running the code for things to work correctly. The reason for this is the usage of multiple files as explained in the next paragraph.

4. The final step is to make the plots and compute the values of v_2 (I use the normal v and the italicised v to denote different things, $\text{v} \neq v$, see the report). This is done by the file called 'resultsPlotter.cpp'. It also has peculiarity due to the development process. Due to time constraints, all of the previous steps were first performed for data in the p_T intervals between 1.0 GeV and 6.0 GeV and then for the intervals between 0.2 GeV and 1.0 GeV. This resulted in two 'results.root' files (which were of course properly later renamed). Hence, 'resultsPlotter.cpp' takes the path to the file with the 1.0 GeV to 6.0 GeV data as its first argument and the other file as its second argument. As the third argument, it takes the index of the centrality interval (as defined in the vector in 'loadHistograms()') as an argument for which centrality interval to plot.

Finally, there are some other files in the repository, some of which are for producing additional plots or other dependencies. See the next section.

45

46 # Other

47 There are some files which were not mentioned above. Here, brief descriptions are given of them. For a more comprehensive view, please see the actual code in the files.

48

49 * ‘entries.cpp‘ : Takes one argument which is a path to a ROOT file with the structure as shown in Figure 7 of the report. Returns the number of events stored in the file so that

50 ‘runProgram.sh‘ can do parallelism properly.

51 * ‘dataFitterVisualiser‘ : Used for creating plots for the report. Various different options are available from the command line, but some options like the title of the graph,

52 the legend and the axis labels have to be changed manually in the code and do not update automatically.

53 * ‘protonSubtracter‘ : The same as ‘dataFitterVisualiser‘ but it subtracts the scaled (based on the fitting results) proton background and creates such plots for the report.

54 * ‘raw‘ : This folder contains the original versions of ‘AliLWUtils.cxx‘ and ‘AliLWUtils.h‘ given to me by my supervisor. They contain the definition and implementation of the classes

55 stored in the ROOT tree with the structure as shown in Figure 7 in the report.

56 * ‘include/AliLWUtils.h‘ and ‘src/AliLWUtils.cxx‘ : The same files as in ‘raw‘ but slightly modified to be compatible with some other functionality which I use.

57 * ‘LinkDef.h‘ : Necessary for making ROOT work in the stand-alone C++ without its own custom interpreter. See the comments in ‘Makefile‘ for some more comments on how this works.

58 * ‘Makefile‘ : Configuration file used by the program ‘make‘ to easily compile all of the executable files correctly.

---

combine.cpp

```cpp
//Core C++
#include <iostream>
#include <string>
#include <vector>
#include <filesystem>

//Other
#include "include/storeInHist.h"
#include "AliLWUtils.h"

//ROOT
#include <TROOT.h>
#include <TFile.h>
#include <TTree.h>
#include <TObject.h>
#include <TRint.h>
#include <TApplication.h>
#include <TBranch.h>
#include <TGraph.h>
#include <TCanvas.h>
#include <TH1D.h>
#include <TH2D.h>
#include <TClonesArray.h>
#include <TMath.h>
#include <TLegend.h>
#include <TStyle.h>


/* Purpose
This program is made to plot various types of stored data for debugging purposes and to
combine all data into a single file.

It is able to plot both FMD-FMD and FMD-TPC correlations for the forward and backwards
FMD:s and the TPC. It can also plot the calculated event mixing background as well
as dividing by the background to get the true signal. It is also able to
project the data to the phi-axis and ignore the pseudorapidity values (eta)
once event mixing has been handled.
 */


TH1D projectHistogram(TH2D histogram) {
    std::string throwAwayName;
    /*
    For some odd reason ROOT needs a string as a name
```

```
45      and stores that internally instead of just the object name.
46      It results in memory leaks if I name a histogram the same thing twice,
47      so I need a new name for every histogram. The easiest way to get
48      a random throwaway garbage name is to deliberately leave a string
            uninitialised.
49      */
50      TH1D returnHistogram(throwAwayName.c_str(), "Counts", histogram.GetNbinsX(),
            0, 2*TMath::Pi());
51      double numerator;
52      double denominator;
53      double errorFactor;
54
55      for (int phiBin = 1; phiBin <= histogram.GetNbinsX(); phiBin++) { //not 0 and
            < because of ROOT's bin convention
56          numerator = 0;
57          denominator = 0;
58          errorFactor = 0;
59
60          //Weighted average
61          for (int etaBin = 1; etaBin <= histogram.GetNbinsY(); etaBin++) {
62              if (histogram.GetBinContent(phiBin, etaBin) == 0) {
63                  continue;
64              }
65              errorFactor = 1/std::pow(histogram.GetBinError(phiBin,
                    etaBin)/histogram.GetBinContent(phiBin, etaBin), 2);
66              numerator += histogram.GetBinContent(phiBin, etaBin) * errorFactor;
67              denominator += errorFactor;
68
69          }
70
71          if (denominator == 0) {continue;}
72          returnHistogram.SetBinContent(phiBin, numerator/denominator);
73          returnHistogram.SetBinError(phiBin,
                std::sqrt(1/denominator)*returnHistogram.GetBinContent(phiBin));
74
75      }
76
77      return returnHistogram;
78
79  }
80
81
82
83
84  int main(int argc, char **argv) {
85      //Makes the application
86      char myChar = 'a'; //ROOT requires argv but I do not want to give it.
87      char* myCharPtr = &myChar;
```

```
88      TRint app("app", 0, &myCharPtr);
89      TCanvas canvas("canvas", "Title", 0, 0 ,800,600);
90
91
92      //Reads in the parameters
93      std::string pathToFile = argv[1];
94      std::string drawOption = argv[2];
95      std::string drawStyle = argv[3];
96      std::string ptRegionString = argv[4];
97      std::string centralityRegionString = argv[5];
98      int ptRegion= std::stoi(ptRegionString);
99      int centralityRegion = std::stoi(centralityRegionString);
100
101
102     //Reads in the data in the histograms
103         //The number '0' in 'myHistogram(0)' is random, I needed to change the
                signature since
104         //the inheritance from TObject already does a default constructor and
                another default constructor
105         // is needed here.
106
107
108
109     storeInHist* myHistogram = new storeInHist(0);
110
111     if (std::filesystem::exists("totalDataProcessed.root")) {
112         storeInHist* myHistogram2 = new storeInHist("totalDataProcessed.root");
113         *myHistogram = *myHistogram2;
114         delete myHistogram2;
115
116     } else {
117         for (const auto& file : std::filesystem::directory_iterator(pathToFile)) {
118             std::string filename = file.path();
119             if (filename == "totalDataProcessed.root") {
120                 //A file is later stored with this name and this avoids double
                        counting
121                 continue;
122             }
123
124             //Empties unused data to save RAM
125             storeInHist dummyHistogram {filename};
126
127
128
129             myHistogram->addHistograms(dummyHistogram);
130
131
132
```

```
133
134         }
135     }
136
137
138
139
140
141
142     //Stores the merged read in data
143     myHistogram->setStorageName("totalData");
144         //All cases have stored data with normalised values after their
145         // respective event mixings have been accounted for, but this is just to
146         //be able to debug individual files. In order to get the proper
                statistics,
147         //the normalisation w.r.t the event mixing needs to be done after the
148         //measured data and the event mixing have all been added up, only then
149         //can the actual normalisation take place properly.
150         //.loadProcessed() is the member function to do this
151
152     myHistogram->loadProcessed();
153     myHistogram->storeHistogramInFile();
154
155
156
157     //Checks which plotting option has been given
158     if (drawOption == "forward") { //ForwardFMD-TPC correlations
159         std::vector<std::vector<TH2D>> histogramForwardvector =
                myHistogram->getForwardHistograms();
160         TH2D histogramForward =
                histogramForwardvector[ptRegion][centralityRegion];
161         histogramForward.Draw(drawStyle.c_str());
162
163         canvas.Modified();
164         canvas.Update();
165         app.Run();
166
167
168     }
169
170     if (drawOption == "forwardBackground") { //ForwardFMD-TPC event mixing
171         std::vector<std::vector<TH2D>> histogramForwardvector =
                myHistogram->getForwardBackgrounds();
172         TH2D histogramForward =
                histogramForwardvector[ptRegion][centralityRegion];
173         histogramForward.Draw(drawStyle.c_str());
174
175         canvas.Modified();
```

```
176         canvas.Update();
177         app.Run();
178 
179 
180     }
181 
182     if (drawOption == "backward") { //BackwardFMD-TPC correlations
183         std::vector<std::vector<TH2D>> histogramBackwardvector =
                myHistogram->getBackwardHistograms();
184         TH2D histogramBackward =
                histogramBackwardvector[ptRegion][centralityRegion];
185         histogramBackward.Draw(drawStyle.c_str());
186 
187         canvas.Modified();
188         canvas.Update();
189         app.Run();
190 
191     }
192 
193     if (drawOption == "backwardBackground") { //BackwardFMD-TPC event mixing
194         std::vector<std::vector<TH2D>> histogramBackwardvector =
                myHistogram->getBackwardBackgrounds();
195         TH2D histogramBackward =
                histogramBackwardvector[ptRegion][centralityRegion];
196         histogramBackward.Draw(drawStyle.c_str());
197 
198         canvas.Modified();
199         canvas.Update();
200         app.Run();
201 
202     }
203 
204 
205 
206     if (drawOption == "backToBack") { //ForwardFMD-BackwardFMD correlations
207         std::vector<std::vector<TH2D>> histogramBackToBackvector =
                myHistogram->getBackToBackHistograms();
208         TH2D histogramBackToBack =
                histogramBackToBackvector[ptRegion][centralityRegion];
209         histogramBackToBack.Draw(drawStyle.c_str());
210 
211         canvas.Modified();
212         canvas.Update();
213         app.Run();
214     }
215 
216 
217 
```

```
218     if (drawOption == "backToBackBackground") { //ForwardFMD-BackwardFMD event
            mixing
219         std::vector<std::vector<TH2D>> histogramBackToBackvector =
                myHistogram->getBackToBackBackgrounds();
220         TH2D histogramBackToBack =
                histogramBackToBackvector[ptRegion][centralityRegion];
221         histogramBackToBack.Draw(drawStyle.c_str());
222
223
224         canvas.Modified();
225         canvas.Update();
226         app.Run();
227     }
228
229
230
231
232
233
234
235     if (drawOption == "none") { //In case one only wants to combine the data
            files without plotting
236         //Does nothing
237     }
238
239     if (drawOption == "all") { //Plots all the different correlations, amplitudes
            may be skewed due to having different
240                                //number of tracks
241         std::vector<std::vector<TH2D>> histogramForwardvector =
                myHistogram->getForwardHistograms();
242         std::vector<std::vector<TH2D>> histogramBackwardvector =
                myHistogram->getBackwardHistograms();
243         std::vector<std::vector<TH2D>> histogramBackToBackvector =
                myHistogram->getBackToBackHistograms();
244
245         TH2D histogramForward =
                histogramForwardvector[ptRegion][centralityRegion];
246         TH2D histogramBackward =
                histogramBackwardvector[ptRegion][centralityRegion];
247         //The FMD:s have no pT-data (pT = transverse momentum), so all pT:s
248         //are in the same region and hence the [0] index. Nesting the data in
                another vector
249         //anyhow is just for purposes of consistency
250         TH2D histogramBackToBack = histogramBackToBackvector[0][centralityRegion];
251
252
253         histogramForward.Add(&histogramBackward);
254         histogramForward.Add(&histogramBackToBack);
```

```
255         histogramForward.Draw(drawStyle.c_str());
256
257
258         canvas.Modified();
259         canvas.Update();
260         app.Run();
261
262     }
263
264     if (drawOption == "allBackground") {//Plots all the different event mixings,
        amplitudes may be skewed due to having different
265                                     //number of tracks
266         std::vector<std::vector<TH2D>> histogramForwardvector =
            myHistogram->getForwardBackgrounds();
267         std::vector<std::vector<TH2D>> histogramBackwardvector =
            myHistogram->getBackwardBackgrounds();
268         std::vector<std::vector<TH2D>> histogramBackToBackvector =
            myHistogram->getBackToBackBackgrounds();
269
270         TH2D histogramForward =
            histogramForwardvector[ptRegion][centralityRegion];
271         TH2D histogramBackward =
            histogramBackwardvector[ptRegion][centralityRegion];
272         //The FMD:s have no pT-data (pT = transverse momentum), so all pT:s
273         //are in the same region and hence the [0] index. Nesting the data in
            another vector
274         //anyhow is just for purposes of consistency
275         TH2D histogramBackToBack = histogramBackToBackvector[0][centralityRegion];
276
277
278         histogramForward.Add(&histogramBackward);
279         histogramForward.Add(&histogramBackToBack);
280         histogramForward.Draw(drawStyle.c_str());
281
282
283         canvas.Modified();
284         canvas.Update();
285         app.Run();
286
287     }
288
289
290
291
292
293     if (drawOption == "allNormalised") { //Plots the calculated data for all
        cases once
294                                     // event mixing has been taken care of.
```

```
295
296         //See the comment earlier in the file about the
297         //purpose of .loadProcessed()
298         myHistogram->loadProcessed();
299         std::vector<std::vector<TH2D>> histogramForwardvector =
                myHistogram->getForwardProcessed();
300         std::vector<std::vector<TH2D>> histogramBackwardvector =
                myHistogram->getBackwardProcessed();
301         std::vector<std::vector<TH2D>> histogramBackToBackvector =
                myHistogram->getBackToBackProcessed();
302
303         TH2D histogramForward =
                histogramForwardvector[ptRegion][centralityRegion];
304         TH2D histogramBackward =
                histogramBackwardvector[ptRegion][centralityRegion];
305         //The FMD:s have no pT-data (pT = transverse momentum), so all pT:s
306         //are in the same region and hence the [0] index. Nesting the data in
                another vector
307         //anyhow is just for purposes of consistency
308         TH2D histogramBackToBack = histogramBackToBackvector[0][centralityRegion];
309
310         histogramForward.Add(&histogramBackward);
311         histogramForward.Add(&histogramBackToBack);
312         histogramForward.Draw(drawStyle.c_str());
313
314
315         canvas.Modified();
316         canvas.Update();
317         app.Run();
318
319     }
320
321     if (drawOption == "forwardNormalised") {//Plots the calculated data for
            forwardFMD-TPC correlations,
322                                             //once event mixing has been taken care
                                                    of.
323
324         //See the comment earlier in the file about the
325         //purpose of .loadProcessed()
326         myHistogram->loadProcessed();
327         std::vector<std::vector<TH2D>> histogramvector =
                myHistogram->getForwardProcessed();
328         TH2D histogram = histogramvector[ptRegion][centralityRegion];
329
330
331         histogram.Draw(drawStyle.c_str());
332
333
```

```
334         canvas.Modified();
335         canvas.Update();
336         app.Run();
337 
338     }
339 
340 
341     if (drawOption == "backwardNormalised") {//Plots the calculated data for
        backwardFMD-TPC correlations,
342                                         //once event mixing has been taken care
                                               of.
343 
344         //See the comment earlier in the file about the
345         //purpose of .loadProcessed()
346         myHistogram->loadProcessed();
347         std::vector<std::vector<TH2D>> histogramvector =
                myHistogram->getBackwardProcessed();
348         TH2D histogram = histogramvector[ptRegion][centralityRegion];
349 
350         histogram.Draw(drawStyle.c_str());
351 
352 
353         canvas.Modified();
354         canvas.Update();
355         app.Run();
356 
357     }
358 
359     if (drawOption == "backToBackNormalised") {//Plots the calculated data for
        forwardFMD-backwardFMD correlations,
360                                         //once event mixing has been taken care
                                               of.
361 
362         //See the comment earlier in the file about the
363         //purpose of .loadProcessed()
364         myHistogram->loadProcessed();
365         std::vector<std::vector<TH2D>> histogramvector =
                myHistogram->getBackToBackProcessed();
366         //The FMD:s have no pT-data (pT = transverse momentum), so all pT:s
367         //are in the same region and hence the [0] index. Nesting the data in
                another vector
368         //anyhow is just for purposes of consistency
369         TH2D histogram = histogramvector[0][centralityRegion];
370 
371 
372         histogram.Draw(drawStyle.c_str());
373 
374 
```

```
375         canvas.Modified();
376         canvas.Update();
377         app.Run();
378 
379     }
380 
381 
382 
383     if (drawOption == "forwardFinished") {//Plots the calculated data for
        forwardFMD-TPC correlations,
384                                     //once event mixing has been taken care of
                                            and projects it onto the
385                                     //the phi-axis
386 
387         //See the comment earlier in the file about the
388         //purpose of .loadProcessed()
389         myHistogram->loadProcessed();
390         myHistogram->setErrors();
391         std::vector<std::vector<TH2D>> histogramvector =
                myHistogram->getForwardProcessed();
392         TH2D histogram = histogramvector[ptRegion][centralityRegion];
393         TH1D phiProjection = projectHistogram(histogram);
394 
395         gPad->SetGrid();
396         gStyle->SetOptStat(0);
397         phiProjection.GetXaxis()->SetTitle("#Delta #varphi");
398         phiProjection.GetYaxis()->SetTitle("Scaled Counts");
399         phiProjection.GetXaxis()->SetTitleSize(0.04);
400         phiProjection.GetYaxis()->SetTitleSize(0.04);
401         phiProjection.SetTitle("Shows TPC-FMD1 Correlations");
402         phiProjection.SetFillColorAlpha(kBlue, 0.5);
403         phiProjection.Draw("HIST E");
404 
405 
406         TLegend myLegend(0.65, 0.7, 0.89, 0.9);
407         myLegend.SetTextSize(0.03);
408         myLegend.AddEntry(&phiProjection, "#it{p_{T}} 2.0 - 2.5 (GeV)", "l");
409         myLegend.Draw("same");
410 
411 
412         canvas.SetLeftMargin(0.15);
413         canvas.Print("combineForwardExample.pdf");
414         canvas.Modified();
415         canvas.Update();
416         app.Run();
417 
418     }
419 
```

```
420
421     if (drawOption == "backwardFinished") {//Plots the calculated data for
            bacmwardFMD-TPC correlations,
422                                     //once event mixing has been taken care of
                                            and projects it onto the
423                                     //the phi-axis
424
425         //See the comment earlier in the file about the
426         //purpose of .loadProcessed()
427         myHistogram->loadProcessed();
428         myHistogram->setErrors();
429         std::vector<std::vector<TH2D>> histogramvector =
                myHistogram->getBackwardProcessed();
430         TH2D histogram = histogramvector[ptRegion][centralityRegion];
431         TH1D phiProjection = projectHistogram(histogram);
432
433
434         gPad->SetGrid();
435         gStyle->SetOptStat(0);
436         phiProjection.GetXaxis()->SetTitle("#Delta #varphi");
437         phiProjection.GetYaxis()->SetTitle("Scaled Counts");
438         phiProjection.GetXaxis()->SetTitleSize(0.04);
439         phiProjection.GetYaxis()->SetTitleSize(0.04);
440         phiProjection.SetTitle("Shows TPC-FMD2 Correlations");
441         phiProjection.SetFillColorAlpha(kBlue, 0.5);
442         phiProjection.Draw("HIST E");
443
444
445         TLegend myLegend(0.65, 0.7, 0.89, 0.9);
446         myLegend.SetTextSize(0.03);
447         myLegend.AddEntry(&phiProjection, "#it{p_{T}} 5.0 - 6.0 (GeV)", "l");
448         myLegend.Draw("same");
449
450
451         canvas.SetLeftMargin(0.15);
452         canvas.Print("combineBackwardExample.pdf");
453
454
455         canvas.Modified();
456         canvas.Update();
457         app.Run();
458
459     }
460
461     if (drawOption == "backToBackFinished") {//Plots the calculated data for
            forwardFMD-backwardFMD correlations,
462                                     //once event mixing has been taken care of
                                            and projects it onto the
```

```
                                        //the phi-axis

        //See the comment earlier in the file about the
        //purpose of .loadProcessed()
        myHistogram->loadProcessed();
        myHistogram->setErrors();
        std::vector<std::vector<TH2D>> histogramvector =
            myHistogram->getBackToBackProcessed();
        //The FMD:s have no pT-data (pT = transverse momentum), so all pT:s
        //are in the same region and hence the [0] index. Nesting the data in
            another vector
        //anyhow is just for purposes of consistency
        TH2D histogram = histogramvector[0][centralityRegion];
        TH1D phiProjection = projectHistogram(histogram);



        gPad->SetGrid();
        gStyle->SetOptStat(0);
        phiProjection.GetXaxis()->SetTitle("#Delta #varphi");
        phiProjection.GetYaxis()->SetTitle("Scaled Counts");
        phiProjection.GetXaxis()->SetTitleSize(0.04);
        phiProjection.GetYaxis()->SetTitleSize(0.04);
        phiProjection.SetTitle("Shows FMD1-FMD2 Correlations");
        phiProjection.SetFillColorAlpha(kBlue, 0.5);
        phiProjection.Draw("HIST E");


        TLegend myLegend(0.65, 0.7, 0.89, 0.9);
        myLegend.SetTextSize(0.03);
        myLegend.AddEntry(&phiProjection, "#splitline{#it{p_{T}} 1.0 - 1.5
            (GeV)}{Centrality 50%-60%}", "l");
        //myLegend.Draw("same");


        canvas.SetLeftMargin(0.15);
        canvas.Print("combineBackToBackExample.pdf");
        canvas.Modified();
        canvas.Update();
        app.Run();

    }

    delete myHistogram;

    (void)argc;

}
```

dataFitter.cpp

```cpp
//Core C++
#include <iostream>
#include <string>
#include <vector>
#include <filesystem>
#include <exception>

//Other
#include "include/storeInHist.h"
#include "AliLWUtils.h"

//ROOT
#include <TROOT.h>
#include <TFile.h>
#include <TTree.h>
#include <TObject.h>
#include <TRint.h>
#include <TApplication.h>
#include <TBranch.h>
#include <TGraph.h>
#include <TCanvas.h>
#include <TH1D.h>
#include <TH2D.h>
#include <TClonesArray.h>
#include <TMath.h>
#include <TF1.h>
#include <TLegend.h>
#include <TGraph.h>
#include <TStyle.h>
#include <TSystem.h>
#include <TRandom2.h>




/* Purpose:
This program is intended to able to extract the the various flow harmonic from
correlations taken from the lead-lead (PbPb) collisions by fitting them to
    Fourier harmonics
and background from proton-proton (pp) collisions to account for non flow effects
    of
QCD.
 */


//ROOT is a bit messy so it is easier to do things with global variables in this
```

```
    case.
TH1D data;
TH1D background;
TRandom2 randomValue(1);


//Returns the pp-background value for a given angle
double ppHistogramValue(Double_t x) {
    //Check that the input is valid
    if ((x < 0) || (x > 2*TMath::Pi())) {
        throw(std::invalid_argument("The value must be between 0 and 2pi"));
    }

    //Finds the correct value to return
    int resultIndex = background.FindBin(x);
    double returnValue = background.GetBinContent(resultIndex);

    return returnValue;
}



//Function that the data will be fitted to.
double fitFunction(double* values, double* parameters) {
    int numberOfHarmonics = sizeof(parameters)-2;
    double backgroundNumber = parameters[0]*ppHistogramValue(values[0]);

    double ellipticFlow = 1 ;
    for (int harmonic = 2; harmonic < numberOfHarmonics; harmonic++) {
        ellipticFlow += 2*parameters[harmonic]*std::cos(harmonic*values[0]);

    }
    ellipticFlow *= parameters[1];



    return backgroundNumber+ellipticFlow;

}


//Projects 2D histograms to 1D and does the proper weighted-average/error
    propagation
TH1D projectHistogram(TH2D histogram) {
    std::string throwAwayName;
    /*
    For some odd reason ROOT needs a string as a name
    and stores that internally instead of just the object name.
```

```
91      It results in memory leaks if I name a histogram the same thing twice,
92      so I need a new name for every histogram. The easiest way to get
93      a random throwaway garbage name is to deliberately leave a string
            uninitialised.
94      */
95      TH1D returnHistogram(throwAwayName.c_str(), "Counts", histogram.GetNbinsX(),
            0, 2*TMath::Pi());
96      double numerator;
97      double denominator;
98      double errorFactor;
99
100     for (int phiBin = 1; phiBin <= histogram.GetNbinsX(); phiBin++) { //not 0 and
            < because of ROOT's bin convention
101         numerator = 0;
102         denominator = 0;
103         errorFactor = 0;
104
105         //Weighted average
106         for (int etaBin = 1; etaBin <= histogram.GetNbinsY(); etaBin++) {
107             if (histogram.GetBinContent(phiBin, etaBin) == 0) {
108                 continue;
109             }
110             errorFactor = 1/std::pow(histogram.GetBinError(phiBin,
                    etaBin)/histogram.GetBinContent(phiBin, etaBin), 2);
111             numerator += histogram.GetBinContent(phiBin, etaBin) * errorFactor;
112             denominator += errorFactor;
113
114         }
115
116         if (denominator == 0) {continue;}
117         returnHistogram.SetBinContent(phiBin, numerator/denominator);
118         returnHistogram.SetBinError(phiBin,
                std::sqrt(1/denominator)*returnHistogram.GetBinContent(phiBin));
119
120     }
121
122     return returnHistogram;
123
124 }
125
126
127
128 //My own implementation of the standard deviation function because reading C++
        documentation
129 //is a lot more time consuming than just writing my own function.
130 double calculateStandardDev(std::vector<double> values) {
131     int entries = static_cast<int>(values.size());
132
```

```
133     double average = 0;
134     for (auto entry : values) {
135         average += entry;
136     }
137 
138 
139     average /= entries;
140     double sumOfSquares = 0;
141     for (auto entry : values) {
142         sumOfSquares += std::pow(entry - average , 2);
143     }
144 
145     return std::sqrt(sumOfSquares/entries);
146 
147 }
148 
149 
150 
151 //Randomises the value of each bin entry
152 //according to a normal distribution centred on its value
153 //with its own uncertainty.
154 TH1D randomiseHistogram(TH1D histogram) {
155     TH1D returnHistogram = histogram;
156     int entries = histogram.GetNbinsX();
157     double localValue;
158     double standardDev;
159     double newValue;
160 
161 
162 
163     for (int entry = 1; entry <= entries; entry++) {
164         localValue = histogram.GetBinContent(entry);
165         standardDev = histogram.GetBinError(entry);
166 
167         newValue = randomValue.Gaus(localValue, standardDev);
168         returnHistogram.SetBinContent(entry, newValue);
169         /*
170         The histogram is randomised with the help of the
171         standard deviation and we then get the uncertainty in
172         v2 from doing multiple fits to randomised histograms.
173         Since we do not want ROOT to go vigilante
174         mode and take the uncertainty into account when it is doing
175         its own fitting, we set it to 0 manually here.
176         */
177         returnHistogram.SetBinError(entry, 0.001);
178 
179     }
180 
```

```
181
182
183
184     return returnHistogram;
185
186 }
187
188
189 //My own mean value function because reading C++ documentation
190 //is more time consuming than making my own version.
191 double meanValue(std::vector<double> vector) {
192     /*
193     Writing my own function is faster than learning how to use the
194     functions in the <algorithm> header :)
195     */
196
197    double average = 0;
198    for (auto entry : vector) {
199         average += entry;
200    }
201
202    return average/static_cast<int>(vector.size());
203
204
205 }
206
207
208
209 int main(int argc, char **argv) {
210     //Argument Processing
211     std::string pathToFile = argv[1];
212     std::string pathToBackground = argv[2];
213
214     std::vector<double> startOfCentralityIntervals {50, 60, 65, 70, 75, 80, 85,
            90};
215     std::vector<double> startOfPtIntervals {1, 1.5, 2, 2.5, 3, 3.5, 4, 5, 6};
            //{0.2, 0.5, 1.0}; //
216
217
218     //Reads in the data
219     storeInHist dataHistograms(pathToFile);
220     storeInHist backgroundHistograms(pathToBackground);
221     dataHistograms.setErrors(); //Calculates the correct errors
222     backgroundHistograms.setErrors();
223
224     dataHistograms.loadProcessed(); //Handles the event mixing properly
225     backgroundHistograms.loadProcessed();
226
```

```
227
228     std::vector<std::vector<TH2D>> dataVectorForward =
            dataHistograms.getForwardProcessed();
229     std::vector<std::vector<TH2D>> backgroundVectorForward =
            backgroundHistograms.getForwardProcessed();
230     std::vector<std::vector<TH2D>> dataVectorBackward =
            dataHistograms.getBackwardProcessed();
231     std::vector<std::vector<TH2D>> backgroundVectorBackward =
            backgroundHistograms.getBackwardProcessed();
232     std::vector<std::vector<TH2D>> dataVectorBackToBack =
            dataHistograms.getBackToBackProcessed();
233     std::vector<std::vector<TH2D>> backgroundVectorBackToBack =
            backgroundHistograms.getBackToBackProcessed();
234
235
236
237     //Vectors for storing the results
238     std::vector<std::vector<std::vector<double>>> v2ForwardList;
239     std::vector<std::vector<std::vector<double>>> v2BackwardList;
240     std::vector<std::vector<std::vector<double>>> v2BackToBackList;
241
242     std::vector<std::vector<double>> v2ForwardFinalList;
243     std::vector<std::vector<double>> v2BackwardFinalList;
244     std::vector<std::vector<double>> v2BackToBackFinalList;
245
246     std::vector<std::vector<double>> v2ErrorForwardList;
247     std::vector<std::vector<double>> v2ErrorBackwardList;
248     std::vector<std::vector<double>> v2ErrorBackToBackList;
249
250     std::vector<std::vector<double>> placeHolder;
251     std::vector<double> placeHolder2;
252
253
254
255     //Defines the fitting function
256     TF1 v2Finder("v2Finder", fitFunction, 0, 2*TMath::Pi()-0.0001, 4);
257     v2Finder.SetParNames("Background Amplitude", "Scaling", "v2", "v3", "v4",
            "v5", "v6", "v7");
258     v2Finder.SetParameter(2,0.5); //Initial Guesses
259     v2Finder.SetParameter(2,0.05);
260     v2Finder.SetParLimits(2, 0, 1);
261     v2Finder.SetParLimits(3, 0, 1);
262
263
264     //Loops over all different cases
265     int numberOfDraws = 5;
266     TH2D dataTemp;
267     TH2D backgroundTemp;
```

```
268
269
270     //Calculates v2 many times for many different random variations to
271     //so that the uncertainty of the final value/fit may be
272     //determined
273     for (int ptNumber = 0; ptNumber < static_cast<int>(dataVectorForward.size());
            ptNumber++) {
274         v2ForwardList.push_back(placeHolder);
275         v2BackwardList.push_back(placeHolder);
276
277         if (ptNumber == 0) {
278             v2BackToBackList.push_back(placeHolder);
279         }
280
281         for (int centralityNumber = 0; centralityNumber <
                static_cast<int>(dataVectorForward[0].size()); centralityNumber++) {
282             v2ForwardList[ptNumber].push_back(placeHolder2);
283             v2BackwardList[ptNumber].push_back(placeHolder2);
284
285             if (ptNumber == 0) {
286                 v2BackToBackList[ptNumber].push_back(placeHolder2);
287             }
288
289
290             //Forward
291             dataTemp = dataVectorForward[ptNumber][centralityNumber];
292             backgroundTemp = backgroundVectorForward[ptNumber][0];
293             for (int pull = 0; pull < numberOfDraws; pull++) {
294                 data = randomiseHistogram(projectHistogram(dataTemp));
295                 background = randomiseHistogram(projectHistogram(backgroundTemp));
296                 data.Fit("v2Finder", "RQ0");
297                 v2ForwardList[ptNumber][centralityNumber].push_back(v2Finder.GetParameter(2));
298
299             }
300
301
302             //Backward
303             dataTemp = dataVectorBackward[ptNumber][centralityNumber];
304             backgroundTemp = backgroundVectorBackward[ptNumber][0];
305             for (int pull = 0; pull < numberOfDraws; pull++) {
306                 data = randomiseHistogram(projectHistogram(dataTemp));
307                 background = randomiseHistogram(projectHistogram(backgroundTemp));
308
309                 data.Fit("v2Finder", "RQ0");
310                 v2BackwardList[ptNumber][centralityNumber].push_back(v2Finder.GetParameter(2));
311
312             }
313
```

```
314
315             //BackToBack
316             if (ptNumber == 0 ) {
317                 dataTemp = dataVectorBackToBack[ptNumber][centralityNumber];
318                 backgroundTemp = backgroundVectorBackToBack[ptNumber][0];
319                 for (int pull = 0; pull < numberOfDraws; pull++) {
320                     data = randomiseHistogram(projectHistogram(dataTemp));
321                     background =
                            randomiseHistogram(projectHistogram(backgroundTemp));
322
323                     data.Fit("v2Finder", "RQ0");
324                     v2BackToBackList[ptNumber][centralityNumber].push_back(v2Finder.GetParameter(
325
326
327                 }
328             }
329
330
331
332
333
334         }
335     }
336
337
338     //Calculates the mean values and standard deviations.
339     for (int ptNumber = 0; ptNumber < static_cast<int>(dataVectorForward.size());
            ptNumber++) {
340         //Initialisation
341         v2ForwardFinalList.push_back(placeHolder2);
342         v2BackwardFinalList.push_back(placeHolder2);
343
344         v2ErrorForwardList.push_back(placeHolder2);
345         v2ErrorBackwardList.push_back(placeHolder2);
346
347         if (ptNumber == 0) {
348             v2BackToBackFinalList.push_back(placeHolder2);
349             v2ErrorBackToBackList.push_back(placeHolder2);
350         }
351
352
353         for (int centralityNumber = 0; centralityNumber <
                static_cast<int>(dataVectorForward[ptNumber].size());
                centralityNumber++) {
354             v2ForwardFinalList[ptNumber].push_back(meanValue(v2ForwardList[ptNumber][centralityN
355             v2BackwardFinalList[ptNumber].push_back(meanValue(v2BackwardList[ptNumber][centralit
356
357             v2ErrorForwardList[ptNumber].push_back(calculateStandardDev(v2ForwardList[ptNumber][
```

```
358             v2ErrorBackwardList[ptNumber].push_back(calculateStandardDev(v2BackwardList[ptNumber
359
360             if (ptNumber == 0) {
361                 v2BackToBackFinalList[ptNumber].push_back(meanValue(v2BackToBackList[ptNumber][c
362                 v2ErrorBackToBackList[ptNumber].push_back(calculateStandardDev(v2BackToBackList[
363             }
364
365
366         }
367     }
368
369
370
371     //Saves the data
372     TFile results("results.root", "RECREATE");
373     results.WriteObject(&v2ForwardFinalList, "v2ForwardList");
374     results.WriteObject(&v2BackwardFinalList, "v2BackwardList");
375     results.WriteObject(&v2BackToBackFinalList, "v2BackToBackList");
376
377     results.WriteObject(&v2ErrorForwardList, "v2ErrorForwardList");
378     results.WriteObject(&v2ErrorBackwardList, "v2ErrorBackwardList");
379     results.WriteObject(&v2ErrorBackToBackList, "v2ErrorBackToBackList");
380
381     results.WriteObject(&startOfCentralityIntervals,
            "startOfCentralityIntervals");
382     results.WriteObject(&startOfPtIntervals, "startOfPtIntervals");
383
384     results.Close();
385     (void)argc;
386
387
388 }
```

dataFitterVisualiser.cpp

```
1  //Core C++
2  #include <iostream>
3  #include <string>
4  #include <vector>
5  #include <filesystem>
6  #include <exception>
7
8  //Other
9  #include "include/storeInHist.h"
10 #include "AliLWUtils.h"
11
12 //ROOT
13 #include <TROOT.h>
14 #include <TFile.h>
15 #include <TTree.h>
16 #include <TObject.h>
17 #include <TRint.h>
18 #include <TApplication.h>
19 #include <TBranch.h>
20 #include <TGraph.h>
21 #include <TCanvas.h>
22 #include <TH1D.h>
23 #include <TH2D.h>
24 #include <TClonesArray.h>
25 #include <TMath.h>
26 #include <TF1.h>
27 #include <TLegend.h>
28 #include <TGraph.h>
29 #include <TStyle.h>
30 #include <TSystem.h>
31 #include <TVirtualFitter.h>
32
33
34 /* Purpose:
35 This program is intended to able to extract the the various flow harmonic from
36 correlations taken from the lead-lead (PbPb) collisions by fitting them to
       Fourier harmonics
37 and background from proton-proton (pp) collisions to account for non flow effects
       of
38 QCD.
39  */
40
41
42 //ROOT is a bit messy so it is easier to do things with global variables in this
       case.
43 TH1D data;
```

```
44 TH1D background;
45
46 //Returns the pp-background value for a given angle
47 double ppHistogramValue(Double_t x) {
48     //Check that the input is valid
49     if ((x < 0) || (x > 2*TMath::Pi())) {
50         throw(std::invalid_argument("The value must be between 0 and 2pi"));
51     }
52
53     //Finds the correct value to return
54     int resultIndex = background.FindBin(x);
55     double returnValue = background.GetBinContent(resultIndex);
56
57     return returnValue;
58 }
59
60
61 //Function that the data will be fitted to.
62 double fitFunction(double* values, double* parameters) {
63     int numberOfHarmonics = sizeof(parameters)-2;
64     double backgroundNumber = parameters[0]*ppHistogramValue(values[0]);
65     double ellipticFlow = 1 ;
66
67     for (int harmonic = 2; harmonic < numberOfHarmonics; harmonic++) {
68         ellipticFlow += 2*parameters[harmonic]*std::cos(harmonic*values[0]);
69     }
70
71     ellipticFlow *= parameters[1];
72
73     return backgroundNumber+ellipticFlow;
74 }
75
76
77 //Projects 2D histograms to 1D and does the proper weighted-average/error
       propagation
78 TH1D projectHistogram(TH2D histogram) {
79     std::string throwAwayString;
80     /*
81     For some odd reason ROOt needs a string as a name
82     and stores that internally instead of just the object name.
83     It results in memory leaks if I name a histogram the same thing twice,
84     so I need a new name for every histogram. The easiest way to get
85     a random throwaway garbage name is to deliberately leave a string
           uninitialised.
86     */
87     TH1D returnHistogram(throwAwayString.c_str(), "Counts",
           histogram.GetNbinsX(), 0, 2*TMath::Pi());
88     double numerator;
```

```
89      double denominator;
90      double errorFactor;
91
92      for (int phiBin = 1; phiBin <= histogram.GetNbinsX(); phiBin++) { //not 0 and
            < because of ROOT's bin convention
93          numerator = 0;
94          denominator = 0;
95          errorFactor = 0;
96
97          //Weighted average
98          for (int etaBin = 1; etaBin <= histogram.GetNbinsY(); etaBin++) {
99              if (histogram.GetBinContent(phiBin, etaBin) == 0) {
100                 continue;
101             }
102             errorFactor = 1/std::pow(histogram.GetBinError(phiBin,
                    etaBin)/histogram.GetBinContent(phiBin, etaBin), 2);
103             numerator += histogram.GetBinContent(phiBin, etaBin) * errorFactor;
104             denominator += errorFactor;
105
106         }
107
108         if (denominator == 0) {continue;}
109         returnHistogram.SetBinContent(phiBin, numerator/denominator);
110
111         returnHistogram.SetBinError(phiBin,
                std::sqrt(1/denominator)*returnHistogram.GetBinContent(phiBin));
112
113     }
114
115     return returnHistogram;
116
117 }
118
119
120
121 int main(int argc, char **argv) {
122     //Argument Processing
123     std::string pathToFile = argv[1];
124     std::string pathToBackground = argv[2];
125     std::string plotOption = argv[3];
126     std::string ptIndexString = argv[4];
127     std::string centralityIndexString = argv[5];
128     int ptIndex = std::stoi(ptIndexString);
129     int centralityIndex = std::stoi(centralityIndexString);
130
131
132     std::vector<Double_t> startOfCentralityIntervals {50, 60, 65, 70, 75, 80, 85,
            90};
```

```
133     std::vector<Double_t> startOfPtIntervals {1, 1.5, 2, 2.5, 3, 3.5, 4, 5, 6};
134
135
136     //Creates the application
137     char myChar = 'a'; //ROOT requires argv but I do not want to give it.
138     char* myCharPtr = &myChar;
139     TRint app("app", 0, &myCharPtr);
140     TCanvas canvas("canvas", "Title", 0, 0 ,800,600);
141
142
143     //Reads in the data
144     storeInHist dataHistograms(pathToFile);
145     storeInHist backgroundHistograms(pathToBackground);
146     dataHistograms.setErrors();
147     backgroundHistograms.setErrors();
148     dataHistograms.loadProcessed();
149     backgroundHistograms.loadProcessed();
150
151
152     if (plotOption == "forward") {
153         std::vector<std::vector<TH2D>> dataVectorForward =
                dataHistograms.getForwardProcessed();
154         std::vector<std::vector<TH2D>> backgroundVectorForward =
                backgroundHistograms.getForwardProcessed();
155
156         data = projectHistogram(dataVectorForward[ptIndex][centralityIndex]);
157         background = projectHistogram(backgroundVectorForward[ptIndex][0]);
158
159     }
160     if (plotOption == "backward") {
161         std::vector<std::vector<TH2D>> dataVectorBackward =
                dataHistograms.getBackwardProcessed();
162         std::vector<std::vector<TH2D>> backgroundVectorBackward =
                backgroundHistograms.getBackwardProcessed();
163
164         data = projectHistogram(dataVectorBackward[ptIndex][centralityIndex]);
165         background = projectHistogram(backgroundVectorBackward[ptIndex][0]);
166
167     }
168     if (plotOption == "backToBack") {
169         std::vector<std::vector<TH2D>> dataVectorBackToBack =
                dataHistograms.getBackToBackProcessed();
170         std::vector<std::vector<TH2D>> backgroundVectorBackToBack =
                backgroundHistograms.getBackToBackProcessed();
171
172         data = projectHistogram(dataVectorBackToBack[ptIndex][centralityIndex]);
173         background = projectHistogram(backgroundVectorBackToBack[ptIndex][0]);
174
```

```
175     }
176
177     if ((plotOption != "forward") && (plotOption != "backward") && (plotOption !=
            "backToBack")) {
178         throw(std::invalid_argument("Valid options for the third argument are
                'forward', 'backward' and 'backToBack'!"));
179     }
180
181
182     //std::cout << data.GetBinError(10)/data.GetBinContent(10) << std::endl;
183
184     //Necessary for plotting
185 /*    double dataMinimum = data.GetMinimum();
186     double dataMaximum = data.GetMaximum();        //Uncommenting this block
            'zooms out' the plot so that the vertical axis starts at 0.
187     data.SetMinimum(dataMinimum*0);
188     data.SetMaximum(dataMaximum*1.1); */
189     TH1D dataCopy = data;
190     dataCopy.SetName("dataCopy");
191     TH1D protonBackground = background;
192
193
194
195     //Plotting Options
196     gROOT->ForceStyle();
197     gStyle->SetOptStat(0);
198     dataCopy.SetFillColorAlpha(kBlue, 0.5);
199
200
201     dataCopy.GetXaxis()->CenterTitle(true);
202     dataCopy.GetYaxis()->CenterTitle(true);
203     dataCopy.GetXaxis()->SetTitle("#Delta #varphi");
204     dataCopy.GetYaxis()->SetTitle("Scaled Counts");
205     dataCopy.GetXaxis()->SetTitleSize(0.04);
206     dataCopy.GetYaxis()->SetTitleSize(0.04);
207     dataCopy.SetTitle("Shows the Measured Signal and the Fit (TPC-FMD2)");
208
209     protonBackground.SetFillColorAlpha(kGreen, 0.2);
210     protonBackground.SetTitle("protonBackground");
211     protonBackground.SetLineColor(kGreen);
212
213     gPad->SetGrid();
214     gStyle->SetTitleFontSize(0.04);
215
216     TGraph fourierBackground;
217     fourierBackground.SetLineColor(kOrange);
218     fourierBackground.SetLineWidth(2);
219
```

```
220
221
222
223
224     //Fitting
225     TF1 fitFunctionROOTBackground("fitFunctionROOTBackground", fitFunction,
            0.0001, 2*TMath::Pi()-0.0001, 4); // +- 0001 to avoid underflow and
            overflow bins
226     fitFunctionROOTBackground.SetParNames("Background Scale Factor", "Fourier
            Harmonic Scale Factor", "v2", "v3", "v4", "v5");
227     fitFunctionROOTBackground.SetRange(0.0001, 2*TMath::Pi()-0.0001);
228
229     fitFunctionROOTBackground.SetParameters(1, 1, 0.05, 0.005);
230     fitFunctionROOTBackground.SetParLimits(2, 0, 1);
231     fitFunctionROOTBackground.SetParLimits(3, 0, 1);
232     for (int n = 0; n < 4; n++) {
233         fitFunctionROOTBackground.SetParError(n, 0);
234     }
235
236
237
238     dataCopy.Fit("fitFunctionROOTBackground", "R same");
239
240
241
242     double protonScale = fitFunctionROOTBackground.GetParameter(0);
243     double fourierScale = fitFunctionROOTBackground.GetParameter(1);
244     double v2 = fitFunctionROOTBackground.GetParameter(2);
245     //double v3 = fitFunctionROOTBackground.GetParameter(3);
246
247
248     protonBackground = protonScale * protonBackground;
249     int numberOfPoints = 5000;
250     double stepLength = 2*TMath::Pi()/numberOfPoints;
251     double fourierValue;
252
253
254     for (int position = 0; position < numberOfPoints; position ++) {
255         fourierValue = protonScale*ppHistogramValue(stepLength*position) +
                fourierScale*(1+2*v2*std::cos(2*stepLength*position));//+2*v3*std::cos(3*stepLength*
256         fourierBackground.AddPoint(stepLength*position, fourierValue);
257     }
258
259
260
261     dataCopy.Draw("HIST same");
262     protonBackground.Draw("HIST same");
263     fourierBackground.Draw("same");
```

```
264
265
266     TLegend myLegend(0.62, 0.7, 0.83, 0.9);
267     myLegend.AddEntry(&data, "Measured Data", "l");
268     myLegend.AddEntry(&fitFunctionROOTBackground, "#splitline{Full
            Fit}{#mbox{}}", "l");
269     myLegend.AddEntry(&fourierBackground, "#splitline{pp Background +}{Only
            Elliptic Flow}", "l");
270     myLegend.SetTextSize(0.03);
271
272
273     myLegend.AddEntry(&protonBackground, "pp Background", "l");
274     myLegend.Draw("same");
275
276
277
278     //Runs the application
279     canvas.SetLeftMargin(0.12);
280     std::string filename = "fitExamplePT" +
            std::to_string(startOfPtIntervals[ptIndex]).substr(0,4) + "Centrality" +
            std::to_string(startOfCentralityIntervals[centralityIndex]).substr(0,4) +
            ".pdf";
281     canvas.Print(filename.c_str());
282     canvas.Modified();
283     canvas.Update();
284     app.Run();
285     (void)argc;
286
287
288 }
```

entries.cpp

```
//Core C++
#include <iostream>
#include <string>
#include <vector>

//Other
#include "AliLWUtils.h"

//ROOT
#include <TROOT.h>
#include <TFile.h>
#include <TTree.h>
#include <TObject.h>
#include <TBranch.h>
#include <TClonesArray.h>


/* Purpose:
The only purpose of this program is to give the bash script
runProgram.sh the number of entries (events) in a given
file so that it may properly divide (and hence parallelise) the intervals
that the readData.cpp file will handle.
*/

int main(int argc, char** argv) {
    //Reads in input data
    std::string pathToFile = argv[1];

    //Reads in the file
    TFile dataFile(pathToFile.c_str(), "dataFile", "READ");
    TTree* dataTree = (TTree*)dataFile.Get("LWTree");
    Int_t dataCount = dataTree->GetEntries();
    dataFile.Close();


    //Returns the value to the bash script
    std::cout << dataCount << std::endl;
        //Apparently the only way to return a value to a
        //bash-script is with 'cout' and not with a return statement for some
            reason
    //return dataCount;



    //Deliberately not using static_cast. I do not want
    //to turn off compiler warnings about
```

```
46      //unused variables but this is actually
47      //supposed to be discarded.
48      (void)argc;
49  }
```

include/AliLWUtils.h

```
#ifndef AliLWUtils__h
#define AliLWUtils__h
#include <TObject.h>


/*
This file contains class definitions which were given to me by my
supervisor to be able to read in the data in the data stored in ROOT-trees.
The original unmodified versions are stored in the folder 'raw'
as I had to modify some syntax in some places to get things to work.

*/

class AliLWTPCTrack : public TObject {
  public:
    AliLWTPCTrack();
    AliLWTPCTrack(Float_t pt, Float_t phi, Float_t eta, Short_t trFlag=1);
    ~AliLWTPCTrack();
    Bool_t IsEqual(TObject* obj) const;
    Bool_t IsSortable() const;
    Int_t Compare(TObject *obj) const;
    Float_t fPt;
    Float_t fPhi;
    Float_t fEta;
    Short_t fTrFlag;
  ClassDef(AliLWTPCTrack, 1);
};
class AliLWFMDTrack : public TObject {
  public:
    AliLWFMDTrack();
    AliLWFMDTrack(Float_t phi, Float_t eta, Short_t mult);
    ~AliLWFMDTrack();
    Float_t fPhi;
    Float_t fEta;
    Short_t fMult;
  ClassDef(AliLWFMDTrack, 1);
};
class AliLWEvent : public TObject {
  public:
    AliLWEvent();
    ~AliLWEvent();
    AliLWEvent(UInt_t runNo, Float_t vz, Float_t cent, Short_t evFlag=1);
    void Setup(UInt_t runNo, Float_t vz, Float_t cent, Short_t evFlag=1);
    UInt_t fRunNo;
    Float_t fVz;
    Float_t fCent;
```

```
47      Short_t fEvFlag;
48    ClassDef(AliLWEvent,1);
49  };
50  #endif
```

include/LinkDef.h

```
#ifdef __CLING__

#pragma link C++ class AliLWTPCTrack+;
#pragma link C++ class AliLWFMDTrack+;
#pragma link C++ class AliLWEvent+;
#pragma link C++ class storeInHist+;

#endif


/*This file is only here to allow ROOT to run in normal
C++ while also using custom ROOT-classes, see
the comments in the file 'Makefile'.
*/
```

include/storeInHist.h

```
#ifndef __storeInHistFredholm__
#define __storeInHistFredholm__

#include <iostream>
#include <vector>
#include <string>
#include <exception>
#include <algorithm>
#include <tuple>
#include <exception>


#include "AliLWUtils.h"

#include <TFile.h>
#include <TTree.h>
#include <TH1D.h>
#include <TH2D.h>
#include <TObject.h>
#include <TClonesArray.h>
#include <TMath.h>
#include <TClonesArray.h>


/*
This class is used for reading in data and processing it. It was implemented as a class
to keep all of the moving parts nicely working together.
*/


class storeInHist {
    private:
        //Member Variables
        std::string _pathToFile; //Where the file is to be stored if storeHistogramInFile() is called
        Int_t _initialised; //For re-implmenting a default constructor since inheritance from
                            //TObject stole it in an earlier implementation.

        //Raw correlation data
        std::vector<std::vector<TH2D>> _storedForwardList;
        std::vector<std::vector<TH2D>> _storedBackwardList;
        std::vector<std::vector<TH2D>> _storedBackToBackList;

        //Backgrounds due to event-mixing
```

```
44          std::vector<std::vector<TH2D>> _noCorrelationForwardList;
45          std::vector<std::vector<TH2D>> _noCorrelationBackwardList;
46          std::vector<std::vector<TH2D>> _noCorrelationBackToBackList;
47
48          //Raw correlation data which has been 'normalised' w.r.t
49          //event-mixing and number of tracks. Note that
50          //the data stored here is not acutally useful
51          //until combine.cpp calculates things
52          //properly with respect to all tracks
53          std::vector<std::vector<TH2D>> _processedForwardList;
54          std::vector<std::vector<TH2D>> _processedBackwardList;
55          std::vector<std::vector<TH2D>> _processedBackToBackList;
56
57          //Stores number of tracks for different categories so
58          //proper normalisation may take place in loadProcessed()
59          std::vector<std::vector<int>> _eventNumberList; //TPC-values
60          std::vector<std::vector<int>> _eventNumberListFMD; //FMD-values
61
62
63          //Member functions
64              //This is the main 'work horse' of the program
65          std::tuple< std::vector<std::vector<std::vector<TH2D>>>,
                std::vector<std::vector<int>>, std::vector<std::vector<int>> >
                loadHistograms(std::string pathToFile, Short_t cutOption,
66                                 Double_t centralityMin, Double_t
                                       centralityMax,
67                                 Double_t ptMin, Double_t ptMax,
68                                 Double_t etaMin, Double_t etaMax,
69                                 Int_t countsPhi, Int_t countsEta,
70                                 Int_t start, Int_t stop);
71
72
73
74          //Various methods for calculting correlations for different cases
75              //FMD-FMD
76          void calculateCorrelation(TH2D& myHistogram, const std::vector<Double_t>&
                phi1, const std::vector<Double_t>& eta1,
77                                    const std::vector<Double_t>& phi2, const
                                          std::vector<Double_t>& eta2,
78                                    const std::vector<Int_t>& mult1, const
                                          std::vector<Int_t>& mult2);
79
80
81
82              //FMD-FMD
83          void calculateSingleCorrelation(TH2D& myHistogram, const Double_t& phi1,
                const Double_t& eta1,
84                                    const std::vector<Double_t>& phi2, const
```

```
                                std::vector<Double_t>& eta2,
85                              const Int_t& mult1, const std::vector<Int_t>&
                                mult2);
86
87          //FMD-TPC
88      void calculateSingleCorrelation(TH2D& myHistogram, const Double_t& phi1,
            const Double_t& eta1,
89                              const std::vector<Double_t>& phi2, const
                                std::vector<Double_t>& eta2,
90                              const Int_t& mult2);
91
92
93
94  public:
95
96      //Member Functions
97          //Primary constructor
98      storeInHist(std::string pathToFile, Short_t cutOption,
99                                  Double_t centralityMin, Double_t
                                        centralityMax,
100                                 Double_t ptMin, Double_t ptMax,
101                                 Double_t etaMin, Double_t etaMax,
102                                 Int_t countsPhi, Int_t countsEta,
103                                 Int_t start, Int_t stop);
104         //Constructs by reading in old saved data
105     storeInHist(std::string pathToFile);
106
107     //A default constructor is necessary in a later part of the code. However,
108     //inheritance from ROOT's TObject interferes with this so I just make one
            with a number instead.
109     storeInHist(Int_t number);
110
111     //Getters
112     std::string getFilePath();
113     const std::vector<std::vector<TH2D>> getForwardHistograms();
114     const std::vector<std::vector<TH2D>> getBackwardHistograms();
115     const std::vector<std::vector<TH2D>> getBackToBackHistograms();
116
117     const std::vector<std::vector<TH2D>> getForwardBackgrounds();
118     const std::vector<std::vector<TH2D>> getBackwardBackgrounds();
119     const std::vector<std::vector<TH2D>> getBackToBackBackgrounds();
120
121     const std::vector<std::vector<TH2D>> getForwardProcessed();
122     const std::vector<std::vector<TH2D>> getBackwardProcessed();
123     const std::vector<std::vector<TH2D>> getBackToBackProcessed();
124
125     const std::vector<std::vector<int>> getEventNumberList();
126     const std::vector<std::vector<int>> getEventNumberListFMD();
```

```
127
128         //Setters
129
130         void setForwardHistograms(std::vector<std::vector<TH2D>> newVector);
131         void setBackwardHistograms(std::vector<std::vector<TH2D>> newVector);
132         void setBackToBackHistograms(std::vector<std::vector<TH2D>> newVector);
133
134         void setForwardBackgrounds(std::vector<std::vector<TH2D>> newVector);
135         void setBackwardBackgrounds(std::vector<std::vector<TH2D>> newVector);
136         void setBackToBackBackgrounds(std::vector<std::vector<TH2D>> newVector);
137
138         void setForwardProcessed(std::vector<std::vector<TH2D>> newVector);
139         void setBackwardProcessed(std::vector<std::vector<TH2D>> newVector);
140         void setBackToBackProcessed(std::vector<std::vector<TH2D>> newVector);
141
142         void setEventNumberList(std::vector<std::vector<int>> newVector);
143         void setEventNumberListFMD(std::vector<std::vector<int>> newVector);
144
145
146
147         void setStorageName(std::string location);
148
149         //Other
150         void addHistograms(storeInHist secondHistogram);
151         void storeHistogramInFile();
152         void loadProcessed(); //Creates the histograms which are compensated for
                event mixing effects.
153         void setErrors(); //Sets the error to the square root of the value of the
                bin for each bin.
154         void setErrors0(); //Sets the errors to 0 for all bins.
155
156
157
158
159
160
161
162
163
164
165 };
166
167
168
169 #endif //__storeInHistFredholm__
```

protonSubtracter.cpp

```
//Core C++
#include <iostream>
#include <string>
#include <vector>
#include <filesystem>
#include <exception>

//Other
#include "include/storeInHist.h"
#include "AliLWUtils.h"

//ROOT
#include <TROOT.h>
#include <TFile.h>
#include <TTree.h>
#include <TObject.h>
#include <TRint.h>
#include <TApplication.h>
#include <TBranch.h>
#include <TGraph.h>
#include <TCanvas.h>
#include <TH1D.h>
#include <TH2D.h>
#include <TClonesArray.h>
#include <TMath.h>
#include <TF1.h>
#include <TLegend.h>
#include <TGraph.h>
#include <TStyle.h>
#include <TSystem.h>
#include <TVirtualFitter.h>




//ROOT is a bit messy so it is easier to do things with global variables in this
    case.
TH1D data;
TH1D background;

//Returns the pp-background value for a given angle
double ppHistogramValue(Double_t x) {
    //Check that the input is valid
    if ((x < 0) || (x > 2*TMath::Pi())) {
        throw(std::invalid_argument("The value must be between 0 and 2pi"));
    }
```

```
46
47      //Finds the correct value to return
48      int resultIndex = background.FindBin(x);
49      double returnValue = background.GetBinContent(resultIndex);
50
51      return returnValue;
52  }
53
54
55  //Function that the data will be fitted to.
56  double fitFunction(double* values, double* parameters) {
57      int numberOfHarmonics = sizeof(parameters)-2;
58      double backgroundNumber = parameters[0]*ppHistogramValue(values[0]);
59      double ellipticFlow = 1 ;
60
61      for (int harmonic = 2; harmonic < numberOfHarmonics; harmonic++) {
62          ellipticFlow += 2*parameters[harmonic]*std::cos(harmonic*values[0]);
63      }
64
65      ellipticFlow *= parameters[1];
66
67      return backgroundNumber+ellipticFlow;
68  }
69
70
71  //Projects 2D histograms to 1D and does the proper weighted-average/error
        propagation
72  TH1D projectHistogram(TH2D histogram) {
73      std::string throwAwayString;
74      /*
75      For some odd reason ROOt needs a string as a name
76      and stores that internally instead of just the object name.
77      It results in memory leaks if I name a histogram the same thing twice,
78      so I need a new name for every histogram. The easiest way to get
79      a random throwaway garbage name is to deliberately leave a string
            uninitialised.
80      */
81      TH1D returnHistogram(throwAwayString.c_str(), "Counts",
            histogram.GetNbinsX(), 0, 2*TMath::Pi());
82      double numerator;
83      double denominator;
84      double errorFactor;
85
86      for (int phiBin = 1; phiBin <= histogram.GetNbinsX(); phiBin++) { //not 0 and
            < because of ROOT's bin convention
87          numerator = 0;
88          denominator = 0;
89          errorFactor = 0;
```

```
90
91          //Weighted average
92          for (int etaBin = 1; etaBin <= histogram.GetNbinsY(); etaBin++) {
93              if (histogram.GetBinContent(phiBin, etaBin) == 0) {
94                  continue;
95              }
96              errorFactor = 1/std::pow(histogram.GetBinError(phiBin,
                    etaBin)/histogram.GetBinContent(phiBin, etaBin), 2);
97              numerator += histogram.GetBinContent(phiBin, etaBin) * errorFactor;
98              denominator += errorFactor;
99
100         }
101
102         if (denominator == 0) {continue;}
103         returnHistogram.SetBinContent(phiBin, numerator/denominator);
104
105         returnHistogram.SetBinError(phiBin,
                std::sqrt(1/denominator)*returnHistogram.GetBinContent(phiBin));
106
107     }
108
109     return returnHistogram;
110
111 }
112
113
114
115 int main(int argc, char **argv) {
116     //Argument Processing
117     std::string pathToFile = argv[1];
118     std::string pathToBackground = argv[2];
119     std::string plotOption = argv[3];
120     std::string ptIndexString = argv[4];
121     std::string centralityIndexString = argv[5];
122     int ptIndex = std::stoi(ptIndexString);
123     int centralityIndex = std::stoi(centralityIndexString);
124
125
126     std::vector<Double_t> startOfCentralityIntervals {50, 60, 65, 70, 75, 80, 85,
            90};
127     std::vector<Double_t> startOfPtIntervals {1, 1.5, 2, 2.5, 3, 3.5, 4, 5, 6};
128
129
130     //Creates the application
131     char myChar = 'a'; //ROOT requires argv but I do not want to give it.
132     char* myCharPtr = &myChar;
133     TRint app("app", 0, &myCharPtr);
134     TCanvas canvas("canvas", "Title", 0, 0 ,800,600);
```

```
135
136
137     //Reads in the data
138     storeInHist dataHistograms(pathToFile);
139     storeInHist backgroundHistograms(pathToBackground);
140     dataHistograms.setErrors();
141     backgroundHistograms.setErrors();
142     dataHistograms.loadProcessed();
143     backgroundHistograms.loadProcessed();
144
145
146     if (plotOption == "forward") {
147         std::vector<std::vector<TH2D>> dataVectorForward =
                dataHistograms.getForwardProcessed();
148         std::vector<std::vector<TH2D>> backgroundVectorForward =
                backgroundHistograms.getForwardProcessed();
149
150         data = projectHistogram(dataVectorForward[ptIndex][centralityIndex]);
151         background = projectHistogram(backgroundVectorForward[ptIndex][0]);
152
153     }
154     if (plotOption == "backward") {
155         std::vector<std::vector<TH2D>> dataVectorBackward =
                dataHistograms.getBackwardProcessed();
156         std::vector<std::vector<TH2D>> backgroundVectorBackward =
                backgroundHistograms.getBackwardProcessed();
157
158         data = projectHistogram(dataVectorBackward[ptIndex][centralityIndex]);
159         background = projectHistogram(backgroundVectorBackward[ptIndex][0]);
160
161     }
162     if (plotOption == "backToBack") {
163         std::vector<std::vector<TH2D>> dataVectorBackToBack =
                dataHistograms.getBackToBackProcessed();
164         std::vector<std::vector<TH2D>> backgroundVectorBackToBack =
                backgroundHistograms.getBackToBackProcessed();
165
166         data = projectHistogram(dataVectorBackToBack[ptIndex][centralityIndex]);
167         background = projectHistogram(backgroundVectorBackToBack[ptIndex][0]);
168
169     }
170
171     if ((plotOption != "forward") && (plotOption != "backward") && (plotOption !=
            "backToBack")) {
172         throw(std::invalid_argument("Valid options for the third argument are
                'forward', 'backward' and 'backToBack'!"));
173     }
174
```

```
175
176     //std::cout << data.GetBinError(10)/data.GetBinContent(10) << std::endl;
177
178     //Necessary for plotting
179 /*    double dataMinimum = data.GetMinimum();
180     double dataMaximum = data.GetMaximum();  //Uncommenting this block 'zooms
            out' the plot so that the vertical axis starts at 0.
181     data.SetMinimum(dataMinimum*0);
182     data.SetMaximum(dataMaximum*1.1); */
183     TH1D dataCopy = data;
184     dataCopy.SetName("dataCopy");
185     TH1D protonBackground = background;
186
187
188
189     //Plotting Options
190     gROOT->ForceStyle();
191     gStyle->SetOptStat(0);
192     dataCopy.SetFillColorAlpha(kBlue, 0.5);
193
194
195     dataCopy.GetXaxis()->CenterTitle(true);
196     dataCopy.GetYaxis()->CenterTitle(true);
197     dataCopy.GetXaxis()->SetTitle("#Delta #varphi");
198     dataCopy.GetYaxis()->SetTitle("Scaled Counts");
199     dataCopy.GetXaxis()->SetTitleSize(0.04);
200     dataCopy.GetYaxis()->SetTitleSize(0.04);
201     dataCopy.SetTitle("#splitline{Shows the Measured Signal Subtracted by
            the}{Proton-data Scaled by the Fit Result (TPC-FMD2)}");
202
203     protonBackground.SetFillColorAlpha(kGreen, 0.2);
204     protonBackground.SetTitle("protonBackground");
205     protonBackground.SetLineColor(kGreen);
206
207     gPad->SetGrid();
208     gStyle->SetTitleFontSize(0.035);
209
210     TGraph fourierBackground;
211     fourierBackground.SetLineColor(kOrange);
212     fourierBackground.SetLineWidth(2);
213
214
215
216
217
218     //Fitting
219     TF1 fitFunctionROOTBackground("fitFunctionROOTBackground", fitFunction,
            0.0001, 2*TMath::Pi()-0.0001, 4); // +- 0001 to avoid underflow and
```

```
        overflow bins
220     fitFunctionROOTBackground.SetParNames("Background Scale Factor", "Fourier
            Harmonic Scale Factor", "v2", "v3", "v4", "v5");
221     fitFunctionROOTBackground.SetRange(0.0001, 2*TMath::Pi()-0.0001);
222
223     fitFunctionROOTBackground.SetParameters(1, 1, 0.05, 0.005);
224     fitFunctionROOTBackground.SetParLimits(2, 0, 1);
225     fitFunctionROOTBackground.SetParLimits(3, 0, 1);
226     for (int n = 0; n < 4; n++) {
227         fitFunctionROOTBackground.SetParError(n, 0);
228     }
229
230
231
232     dataCopy.Fit("fitFunctionROOTBackground", "RQ0 same");
233     double protonScale = fitFunctionROOTBackground.GetParameter(0);
234     protonBackground = protonScale * protonBackground;
235
236     dataCopy = dataCopy - protonBackground;
237     dataCopy.Draw("HIST same");
238
239
240
241     TLegend myLegend(0.62, 0.7, 0.83, 0.9);
242     myLegend.AddEntry(&data, "Measured Data", "l");
243     myLegend.AddEntry(&fitFunctionROOTBackground, "#splitline{Full
            Fit}{#mbox{}}", "l");
244     myLegend.SetTextSize(0.03);
245
246
247
248     //Runs the application
249     canvas.SetLeftMargin(0.12);
250     std::string filename = "fitExamplePT" +
            std::to_string(startOfPtIntervals[ptIndex]).substr(0,4) + "Centrality" +
            std::to_string(startOfCentralityIntervals[centralityIndex]).substr(0,4) +
            ".pdf";
251     canvas.Print(filename.c_str());
252     canvas.Modified();
253     canvas.Update();
254     app.Run();
255     (void)argc;
256
257
258 }
```

raw/AliLWUtils.cxx

```
#include "AliLWUtils.h"
AliLWTPCTrack::AliLWTPCTrack():fPt(-1),fPhi(-999),fEta(-999),fTrFlag(1) {};
AliLWTPCTrack::AliLWTPCTrack(Float_t pt, Float_t phi, Float_t eta, Short_t
    trFlag):fPt(pt),fPhi(phi),fEta(eta),fTrFlag(trFlag) {};
AliLWTPCTrack::~AliLWTPCTrack() {};
Bool_t AliLWTPCTrack::IsEqual(TObject *obj) const
{
  AliLWTPCTrack *l_Tr = (AliLWTPCTrack*)obj;
  if(!l_Tr) return kFALSE;
  if(fPhi == l_Tr->fPhi &&
     fEta == l_Tr->fEta &&
     fPt  == l_Tr->fPt &&
     fTrFlag== l_Tr->fTrFlag)
      return kTRUE;
  return kFALSE;
};
Int_t AliLWTPCTrack::Compare(TObject *obj) const
{
  AliLWTPCTrack *l_Tr = (AliLWTPCTrack*)obj;
  if(fPt < l_Tr->fPt) return -1;
  else if(fPt > l_Tr->fPt) return 1;
  else return 0;
};
Bool_t AliLWTPCTrack::IsSortable() const {return kTRUE; };
AliLWFMDTrack::AliLWFMDTrack():fPhi(-999),fEta(-999),fMult(-1) {};
AliLWFMDTrack::AliLWFMDTrack(Float_t phi, Float_t eta, Short_t
    mult):fPhi(phi),fEta(eta),fMult(mult) {};
AliLWFMDTrack::~AliLWFMDTrack() {};
AliLWEvent::AliLWEvent():fRunNo(0),fVz(-999),fCent(-1),fEvFlag(1) {};
AliLWEvent::AliLWEvent(UInt_t runNo, Float_t vz, Float_t cent, Short_t
    evFlag):fRunNo(runNo),fVz(vz),fCent(cent),fEvFlag(evFlag) {};
AliLWEvent::~AliLWEvent() {};
void AliLWEvent::Setup(UInt_t runNo, Float_t vz, Float_t cent, Short_t evFlag) {
  fRunNo = runNo;
  fVz = vz;
  fCent = cent;
  fEvFlag = evFlag;
}
```

raw/AliLWUtils.h

```
#ifndef AliLWUtils__h
#define AliLWUtils__h
#include "TObject.h"
class AliLWTPCTrack : public TObject {
  public:
    AliLWTPCTrack();
    AliLWTPCTrack(Float_t pt, Float_t phi, Float_t eta, Short_t trFlag=1);
    ~AliLWTPCTrack();
    Bool_t IsEqual(TObject* obj) const;
    Bool_t IsSortable() const;
    Int_t Compare(TObject *obj) const;
    Float_t fPt;
    Float_t fPhi;
    Float_t fEta;
    Short_t fTrFlag;
  ClassDef(AliLWTPCTrack, 1);
};
class AliLWFMDTrack : public TObject {
  public:
    AliLWFMDTrack();
    AliLWFMDTrack(Float_t phi, Float_t eta, Short_t mult);
    ~AliLWFMDTrack();
    Float_t fPhi;
    Float_t fEta;
    Short_t fMult;
  ClassDef(AliLWFMDTrack, 1);
};
class AliLWEvent : public TObject {
  public:
    AliLWEvent();
    ~AliLWEvent();
    AliLWEvent(UInt_t runNo, Float_t vz, Float_t cent, Short_t evFlag=1);
    void Setup(UInt_t runNo, Float_t vz, Float_t cent, Short_t evFlag=1);
    UInt_t fRunNo;
    Float_t fVz;
    Float_t fCent;
    Short_t fEvFlag;
  ClassDef(AliLWEvent,1);
};
#endif
```

readData.cpp

```cpp
//Core C++
#include <iostream>
#include <string>
#include <vector>
#include <thread>
#include <future>
#include <mutex>


//Other
#include "include/storeInHist.h"
#include "AliLWUtils.h"

//ROOT
#include <TROOT.h>
#include <TFile.h>
#include <TTree.h>
#include <TObject.h>
#include <TRint.h>
#include <TApplication.h>
#include <TBranch.h>
#include <TGraph.h>
#include <TCanvas.h>
#include <TH1D.h>
#include <TH2D.h>
#include <TClonesArray.h>
#include <TMath.h>



/* Purpose:
The purpose of this file is to be able to read in parts of files
and process data as well as the event mixing background.

These are just some simple running instructions, and a lot of them
are left for backwards compatibility reasons.

The input parameters are handled by runProgram.sh
*/


int main(int argc, char **argv) {
    //Reads in the input variables
    std::string pathToFile = argv[1];
    std::string startString = argv[2];
    std::string stopString = argv[3];
```

```
47      int start = std::stoi(startString);
48      int stop = std::stoi(stopString);
49
50      //Program Parameters
51      int phiBins = 20; //20 bins for the FMD:s
52      int etaBins = 320; //400 bins over -10 to 10 in the FMD:s
53      double etaMin = -6; //eta is the pseudorapidity
54      double etaMax = 10;
55
56
57
58      /*
59      These values below are just left for backwards compatiblity, and
60      changing them might have unintended consequences. If I had more time
61      I could had cleaned up the syntax more. The real values are set
62      in the file src/storeInHist.cxx in the member function loadHistograms().
63      */
64      Short_t cutOption = 3;
65      double centralityStart = 50;
66      double centralityStop = 90;
67      double ptStart = 0.2;
68      double ptStop = 6;
69
70
71      //Runs the program
72      storeInHist myHistogram {pathToFile, cutOption,
73                               centralityStart, centralityStop,
74                               ptStart, ptStop,
75                               etaMin, etaMax,
76                               phiBins, etaBins,
77                               start, stop};
78
79
80
81      (void)argc;
82
83  }
```

resultsPlotter.cpp

```cpp
//Core C++
#include <iostream>
#include <string>
#include <vector>
#include <filesystem>
#include <exception>


//Other
#include "include/storeInHist.h"
#include "AliLWUtils.h"

//ROOT
#include <TROOT.h>
#include <TFile.h>
#include <TTree.h>
#include <TObject.h>
#include <TRint.h>
#include <TApplication.h>
#include <TBranch.h>
#include <TGraph.h>
#include <TCanvas.h>
#include <TH1D.h>
#include <TH2D.h>
#include <TClonesArray.h>
#include <TMath.h>
#include <TF1.h>
#include <TLegend.h>
#include <TGraph.h>
#include <TGraphErrors.h>
#include <TStyle.h>
#include <TSystem.h>



int main(int argc, char** argv) {
    //Argument processing
    std::string pathToFile = argv[1];
    std::string pathToFile2 = argv[2];
    std::string centralityIntervalString = argv[3];
    int centralityInterval = std::stoi(centralityIntervalString);


    //Creates the application
    char myChar = 'a'; //ROOT requires argv but I do not want to give it.
    char* myCharPtr = &myChar;
```

```
47      TRint app("app", 0, &myCharPtr);
48      TCanvas canvas("canvas", "Title", 0, 0 ,800,600);
49
50      //Reads in the data from the first run
51      TFile dataset1(pathToFile.c_str(), "READ");
52
53      std::vector<std::vector<double>> v2ForwardList =
            *(std::vector<std::vector<double>>*)dataset1.Get("v2ForwardList");
54      std::vector<std::vector<double>> v2BackwardList =
            *(std::vector<std::vector<double>>*)dataset1.Get("v2BackwardList");;
55      std::vector<std::vector<double>> v2BackToBackList =
            *(std::vector<std::vector<double>>*)dataset1.Get("v2BackToBackList");
56
57      std::vector<std::vector<double>> v2ErrorForwardList =
            *(std::vector<std::vector<double>>*)dataset1.Get("v2ErrorForwardList");
58      std::vector<std::vector<double>> v2ErrorBackwardList =
            *(std::vector<std::vector<double>>*)dataset1.Get("v2ErrorBackwardList");;
59      std::vector<std::vector<double>> v2ErrorBackToBackList =
            *(std::vector<std::vector<double>>*)dataset1.Get("v2ErrorBackToBackList");
60
61      std::vector<double> startOfCentralityIntervals =
            *(std::vector<double>*)dataset1.Get("startOfCentralityIntervals");
62      std::vector<double> startOfPtIntervals =
            *(std::vector<double>*)dataset1.Get("startOfPtIntervals");
63
64      dataset1.Close();
65
66
67      //Reads in data from the second run
68      TFile dataset2(pathToFile2.c_str(), "READ");
69
70      std::vector<std::vector<double>> v2ForwardList2 =
            *(std::vector<std::vector<double>>*)dataset2.Get("v2ForwardList");
71      std::vector<std::vector<double>> v2BackwardList2 =
            *(std::vector<std::vector<double>>*)dataset2.Get("v2BackwardList");;
72      std::vector<std::vector<double>> v2BackToBackList2 =
            *(std::vector<std::vector<double>>*)dataset2.Get("v2BackToBackList");
73
74      std::vector<std::vector<double>> v2ErrorForwardList2 =
            *(std::vector<std::vector<double>>*)dataset2.Get("v2ErrorForwardList");
75      std::vector<std::vector<double>> v2ErrorBackwardList2 =
            *(std::vector<std::vector<double>>*)dataset2.Get("v2ErrorBackwardList");;
76      std::vector<std::vector<double>> v2ErrorBackToBackList2 =
            *(std::vector<std::vector<double>>*)dataset2.Get("v2ErrorBackToBackList");
77
78      std::vector<double> startOfCentralityIntervals2 =
            *(std::vector<double>*)dataset2.Get("startOfCentralityIntervals");
79      std::vector<double> startOfPtIntervals2 =
```

```
        *(std::vector<double>*)dataset2.Get("startOfPtIntervals");
80
81      dataset2.Close();
82
83
84      //Merges the two datasets
85
86      for (int elementNumber = static_cast<int>(startOfPtIntervals2.size()) - 2;
            elementNumber >= 0; elementNumber--) { //-2 is intentional
87          startOfPtIntervals.insert(startOfPtIntervals.begin(),
                startOfPtIntervals2[elementNumber]);
88
89      }
90
91      for (int elementNumber = static_cast<int>(v2ForwardList2.size()) - 1;
            elementNumber >= 0; elementNumber--) { //-1 is intentional
92          v2ForwardList.insert(v2ForwardList.begin(),
                v2ForwardList2[elementNumber]);
93          v2BackwardList.insert(v2BackwardList.begin(),
                v2BackwardList2[elementNumber]);
94
95          v2ErrorForwardList.insert(v2ErrorForwardList.begin(),
                v2ErrorForwardList2[elementNumber]);
96          v2ErrorBackwardList.insert(v2ErrorBackwardList.begin(),
                v2ErrorBackwardList[elementNumber]);
97
98          //The fmd-fmd data does not need to be merged as these results are the
                same
99      }
100
101
102     if (centralityInterval >
            static_cast<int>(startOfCentralityIntervals.size())-2) {
103         throw std::invalid_argument("That centrality interval is not available!");
104     }
105
106
107     //Calculates the final values
108     std::vector<std::vector<double>> v2FinalList;
109     std::vector<std::vector<double>> v2ErrorFinalList;
110     std::vector<double> placeHolder;
111
112     double v2Forward;
113     double v2Backward;
114     double v2BackToBack;
115
116     double v2ErrorForward;
117     double v2ErrorBackward;
```

```
118     double v2ErrorBackToBack;
119
120     double v2Final;
121     double v2ErrorFinal;
122
123
124     //Calculates the actual v2 values
125     for (int ptNumber = 0; ptNumber < static_cast<int>(startOfPtIntervals.size())
            - 1; ptNumber++) {
126         v2FinalList.push_back(placeHolder);
127         v2ErrorFinalList.push_back(placeHolder);
128
129         for (int centralityNumber = 0; centralityNumber <
                static_cast<int>(startOfCentralityIntervals.size()) - 1;
                centralityNumber++) {
130             v2Forward = v2ForwardList[ptNumber][centralityNumber];
131             v2Backward = v2BackwardList[ptNumber][centralityNumber];
132             v2BackToBack = v2BackToBackList[0][centralityNumber];
133
134             v2ErrorForward = v2ErrorForwardList[ptNumber][centralityNumber];
135             v2ErrorBackward = v2ErrorBackwardList[ptNumber][centralityNumber];
136             v2ErrorBackToBack = v2ErrorBackToBackList[0][centralityNumber];
137
138             //Final value and error propagation
139             v2Final = std::sqrt(v2Forward*v2Backward/v2BackToBack);
140             v2ErrorFinal =
                    0.5*v2Final*std::sqrt(std::pow(v2ErrorForward/v2Forward,
                    2)+std::pow(v2ErrorBackward/v2Backward,
                    2)+std::pow(v2ErrorBackToBack/v2BackToBack, 2));
141
142
143
144
145             v2FinalList[ptNumber].push_back(v2Final);
146             v2ErrorFinalList[ptNumber].push_back(v2ErrorFinal);
147
148
149         }
150
151     }
152
153
154
155     //Plotting
156     TGraphErrors finalPlot;
157     finalPlot.SetName("finalPlot");
158     finalPlot.SetTitle("Shows values of #nu_{2} for various values of #it{p_{T}};
            #it{p_{T}} (GeV); #it{#nu_{2}}");
```

```
159
160
161     int counter = 0;
162     double ptMid;
163     double ptDiff;
164     double v2value;
165     double v2ErrorValue;
166
167     for (int ptNumber = 0; ptNumber < static_cast<int>(startOfPtIntervals.size())
            - 1; ptNumber++) {
168
169         ptMid = (startOfPtIntervals[ptNumber+1] + startOfPtIntervals[ptNumber])/2;
170         ptDiff = (startOfPtIntervals[ptNumber+1] -
                startOfPtIntervals[ptNumber])/2;
171
172         v2value = v2FinalList[ptNumber][centralityInterval];
173         v2ErrorValue = v2ErrorFinalList[ptNumber][centralityInterval];
174         //std::cout << "$" << v2value << "\\pm " << v2ErrorValue << "$" <<
                std::endl;
175
176
177         finalPlot.AddPoint(ptMid, v2value);
178         finalPlot.SetPointError(counter, ptDiff, v2ErrorValue);
179
180         counter++;
181
182
183
184     }
185
186
187
188
189     gPad->SetGrid();
190     gStyle->SetCanvasPreferGL(kTRUE);
191     finalPlot.SetMarkerColor(2);
192     finalPlot.SetMarkerStyle(8);
193     finalPlot.SetFillColorAlpha(kRed,0.4);
194
195
196
197     TLegend myLegend(0.6, 0.70, 0.87, 0.85);
198     std::string centralityStart =
            std::to_string(startOfCentralityIntervals[centralityInterval]);
199     centralityStart = centralityStart.substr(0,2);
200     std::string centralityStop =
            std::to_string(startOfCentralityIntervals[centralityInterval+1]);
201     centralityStop = centralityStop.substr(0,2);
```

```
202     std::string stringCentralityIntervalString2 =centralityStart+"% -
            "+centralityStop + "% Centrality";
203     myLegend.AddEntry(&finalPlot, stringCentralityIntervalString2.c_str(), "pf");
204     finalPlot.Draw("AP2");
205     myLegend.Draw();
206
207
208
209     std::string filename = "resultsCentrality" +
            std::to_string(startOfCentralityIntervals[centralityInterval]).substr(0,4)
            + ".pdf";
210     gPad->Print(filename.c_str());
211
212     //Runs the application
213     canvas.Modified();
214     canvas.Update();
215     app.Run();
216
217
218
219     (void)argc;
220     (void)argv;
221 }
```

runProgram.sh

```
#!/bin/bash


#This script runs the program over all files in a given directory.
#To parallelise the process, it runs multiple instances of the same
#program over different events (which in total covers all events in a given file).
#To make sure that it does not start running on new files before all
#files have finished running, the program keeps track of the PID:s
#it has started and waits for all of them to finish.


pids="" #Stores active process id:s so that we can wait for all to finish
maxcores=$2
fileextension=".root"

#Makes sure there is a new empty folder
rm -rfd processedData
rm -rfd "temp"
mkdir processedData
mkdir temp
mkdir -p processedData/disk/DataSets_PbPb/TPCFMDTrees/LHC15o/WithFMD/ #For
    backwards compatiblity

#Stores a list of all files the directory to be read in ascending file size order
#so that some files may be used while waiting for program to finish
ls -S ${1}*.root > fileListDescending.txt
tac fileListDescending.txt > fileListAscending.txt
rm fileListDescending.txt



#Starts the reading in of each files
for entry in $(cat fileListAscending.txt); do
    echo "Starting process for $entry"
    echo $(date) #Prints start time
    numberOfEvents=$(./entries $entry)
    eventsPerCore=$((numberOfEvents / maxcores))

    for task in $(seq 1 $maxcores); do
        newfolder="temp"
        filename="${entry#"$1"}"
        filename="${filename%".root"}"
        newname="${newfolder}/${filename}part${task}.root"
        cp $entry $newname

        start=$(( eventsPerCore * (task-1) ))
```

```
46          stop=$(( eventsPerCore * task ))
47          if [[ $task = $maxcores ]]; then
48              stop=$numberOfEvents
49          fi
50
51          ./readData $newname $start $stop &
52          processId="$!"
53          pids="$pids $processId"
54
55
56      done
57
58      for id in $pids; do
59          wait $id
60      done
61
62      for task in $(seq 1 $maxcores); do
63          newfolder="temp"
64          filename="${entry#"$1"}"
65          filename="${filename%".root"}"
66          newname="${newfolder}/${filename}part${task}.root"
67          rm $newname
68
69      done
70
71      echo $(date) #Prints finish time
72
73
74  done
75
76
77  rm fileListAscending.txt
78  echo "All files have now been processed"
```

src/AliLWUtils.cxx

```
#include "../include/AliLWUtils.h"



/*
This file contains class implementations which were given to me by my
supervisor to be able to read in the data in the data stored in ROOT-trees.
The original unmodified versions are stored in the folder 'raw'
as I had to modify some syntax in some places to get things to work.

*/


ClassImp(AliLWTPCTrack);
ClassImp(AliLWFMDTrack);
ClassImp(AliLWEvent);


AliLWTPCTrack::AliLWTPCTrack():fPt(-1),fPhi(-999),fEta(-999),fTrFlag(1) {};
AliLWTPCTrack::AliLWTPCTrack(Float_t pt, Float_t phi, Float_t eta, Short_t
    trFlag):fPt(pt),fPhi(phi),fEta(eta),fTrFlag(trFlag) {};
AliLWTPCTrack::~AliLWTPCTrack() {};
Bool_t AliLWTPCTrack::IsEqual(TObject *obj) const
{
  AliLWTPCTrack *l_Tr = (AliLWTPCTrack*)obj;
  if(!l_Tr) return kFALSE;
  if(fPhi == l_Tr->fPhi &&
     fEta == l_Tr->fEta &&
     fPt  == l_Tr->fPt &&
     fTrFlag== l_Tr->fTrFlag)
      return kTRUE;
  return kFALSE;
};
Int_t AliLWTPCTrack::Compare(TObject *obj) const
{
  AliLWTPCTrack *l_Tr = (AliLWTPCTrack*)obj;
  if(fPt < l_Tr->fPt) return -1;
  else if(fPt > l_Tr->fPt) return 1;
  else return 0;
};
Bool_t AliLWTPCTrack::IsSortable() const {return kTRUE; };
AliLWFMDTrack::AliLWFMDTrack():fPhi(-999),fEta(-999),fMult(-1) {};
AliLWFMDTrack::AliLWFMDTrack(Float_t phi, Float_t eta, Short_t
    mult):fPhi(phi),fEta(eta),fMult(mult) {};
AliLWFMDTrack::~AliLWFMDTrack() {};
AliLWEvent::AliLWEvent():fRunNo(0),fVz(-999),fCent(-1),fEvFlag(1) {};
```

```
45 AliLWEvent::AliLWEvent(UInt_t runNo, Float_t vz, Float_t cent, Short_t
       evFlag):fRunNo(runNo),fVz(vz),fCent(cent),fEvFlag(evFlag) {};
46 AliLWEvent::~AliLWEvent() {};
47 void AliLWEvent::Setup(UInt_t runNo, Float_t vz, Float_t cent, Short_t evFlag) {
48   fRunNo = runNo;
49   fVz = vz;
50   fCent = cent;
51   fEvFlag = evFlag;
52 }
```

src/storeInHist.cxx

```
#include "../include/storeInHist.h"

/*
This file contains the implementation of the class
which handles reading in data, processing the data
and storing the data for later use.
*/



//Getters and Setters
void storeInHist::setStorageName(std::string location) {
    this->_pathToFile = location;
}

std::string storeInHist::getFilePath() {
    return this->_pathToFile;
}

const std::vector<std::vector<TH2D>> storeInHist::getForwardHistograms() {
    return this->_storedForwardList;
}

const std::vector<std::vector<TH2D>> storeInHist::getBackwardHistograms() {
    return this->_storedBackwardList;
}

const std::vector<std::vector<TH2D>> storeInHist::getBackToBackHistograms() {
    return this->_storedBackToBackList;
}

const std::vector<std::vector<TH2D>> storeInHist::getForwardBackgrounds() {
    return this->_noCorrelationForwardList;
}

const std::vector<std::vector<TH2D>> storeInHist::getBackwardBackgrounds() {
    return this->_noCorrelationBackwardList;
}

const std::vector<std::vector<TH2D>> storeInHist::getBackToBackBackgrounds() {
    return this->_noCorrelationBackToBackList;
}

const std::vector<std::vector<TH2D>> storeInHist::getForwardProcessed() {
    return this->_processedForwardList;
}
```

```
47
48  const std::vector<std::vector<TH2D>> storeInHist::getBackwardProcessed() {
49      return this->_processedBackwardList;
50  }
51
52  const std::vector<std::vector<TH2D>> storeInHist::getBackToBackProcessed() {
53      return this->_processedBackToBackList;
54  }
55
56  const std::vector<std::vector<int>> storeInHist::getEventNumberList() {
57      return this->_eventNumberList;
58  }
59
60  const std::vector<std::vector<int>> storeInHist::getEventNumberListFMD() {
61      return this->_eventNumberListFMD;
62  }
63
64
65
66
67  void storeInHist::setForwardHistograms(std::vector<std::vector<TH2D>> newVector) {
68      this->_storedForwardList = newVector;
69  }
70  void storeInHist::setBackwardHistograms(std::vector<std::vector<TH2D>> newVector)
        {
71      this->_storedBackwardList = newVector;
72  }
73  void storeInHist::setBackToBackHistograms(std::vector<std::vector<TH2D>>
        newVector) {
74      this->_storedBackToBackList = newVector;
75  }
76
77  void storeInHist::setForwardBackgrounds(std::vector<std::vector<TH2D>> newVector)
        {
78      this->_noCorrelationForwardList = newVector;
79  }
80  void storeInHist::setBackwardBackgrounds(std::vector<std::vector<TH2D>>
        newVector) {
81      this->_noCorrelationBackwardList = newVector;
82  }
83  void storeInHist::setBackToBackBackgrounds(std::vector<std::vector<TH2D>>
        newVector) {
84      this->_noCorrelationBackToBackList = newVector;
85  }
86
87  void storeInHist::setForwardProcessed(std::vector<std::vector<TH2D>> newVector) {
88      this->_processedForwardList = newVector;
89  }
```

```
90  void storeInHist::setBackwardProcessed(std::vector<std::vector<TH2D>> newVector) {
91      this->_processedBackwardList = newVector;
92  }
93  void storeInHist::setBackToBackProcessed(std::vector<std::vector<TH2D>>
        newVector) {
94      this->_processedBackToBackList = newVector;
95  }
96
97  void storeInHist::setEventNumberList(std::vector<std::vector<int>> newVector) {
98      this->_eventNumberList = newVector;
99  }
100 void storeInHist::setEventNumberListFMD(std::vector<std::vector<int>> newVector) {
101     this->_eventNumberListFMD = newVector;
102 }
103
104
105
106
107
108 //Methods
109     //Various methods to save space later when many correlations have to be
            calculated
110 void storeInHist::calculateCorrelation(TH2D& myHistogram, const
        std::vector<Double_t>& phi1, const std::vector<Double_t>& eta1,
111                                 const std::vector<Double_t>& phi2, const
                                        std::vector<Double_t>& eta2,
112                                 const std::vector<Int_t>& mult1, const
                                        std::vector<Int_t>& mult2) {
113
114
115     //This is just for my own debugging, in a public member function we of course
            properly throw errors
116     if (phi1.size() != eta1.size()) {
117         std::cout << "Unequal Sizes detect" << std::endl;
118
119
120     }
121
122     if (phi2.size() != eta2.size()) {
123         std::cout << "Unequal Sizes detect" << std::endl;
124
125
126     }
127
128
129     Double_t phiDiff;
130     Double_t etaDiff;
131     Double_t multiplicity;
```

```
132
133
134     for (int detector1Track = 0; detector1Track < static_cast<int>(phi1.size());
            detector1Track++) {
135         for (int detector2Track = 0; detector2Track <
                static_cast<int>(phi2.size()); detector2Track++) {
136
137             phiDiff = phi1[detector1Track] - phi2[detector2Track] - 0.0001; //This
                    is for the FMD-FMD correlation so -0.001 is necessary to avoid
                    binning problems
138             if (phiDiff < 0) {
139                 phiDiff += 2*TMath::Pi();
140             }
141
142             etaDiff = eta1[detector1Track] - eta2[detector2Track] - 0.0001;
143             multiplicity = mult1[detector1Track] * mult2[detector2Track];
144             myHistogram.Fill(phiDiff, etaDiff, multiplicity);
145
146
147         }
148
149     }
150
151
152 }
153
154
155
156
157
158     //FMD-FMD
159 void storeInHist::calculateSingleCorrelation(TH2D& myHistogram, const Double_t&
        phi1, const Double_t& eta1,
160                                  const std::vector<Double_t>& phi2, const
                                        std::vector<Double_t>& eta2,
161                                  const Int_t& mult1, const std::vector<Int_t>&
                                        mult2) {
162
163
164     if (phi2.size() != eta2.size()) {
165         std::cout << "Unequal Sizes detect" << std::endl;
166
167
168     }
169
170
171     Double_t phiDiff;
172     Double_t etaDiff;
```

```
173     Double_t multiplicity;
174
175
176     for (int detectorTrack = 0; detectorTrack < static_cast<int>(phi2.size());
            detectorTrack++) {
177         phiDiff = phi1 - phi2[detectorTrack] -0.0001; //-0.0001 Necessary to
                avoid binning problems in FMD-FMD correlations
178
179         if (phiDiff < 0) {
180             phiDiff += 2*TMath::Pi();
181         }
182
183         etaDiff = eta1 - eta2[detectorTrack] - 0.0001; //-0.0001 Necessary to
                avoid binning problems in FMD-FMD correlations
184         multiplicity = mult1 * mult2[detectorTrack];
185         myHistogram.Fill(phiDiff, etaDiff, multiplicity);
186
187
188     }
189
190
191
192 }
193
194
195     //FMD-TPC
196 void storeInHist::calculateSingleCorrelation(TH2D& myHistogram, const Double_t&
        phi1, const Double_t& eta1,
197                                  const std::vector<Double_t>& phi2, const
                                        std::vector<Double_t>& eta2,
198                                  const Int_t& mult1) {
199     if (phi2.size() != eta2.size()) {
200         std::cout << "Unequal Sizes detect" << std::endl;
201
202
203     }
204
205
206     Double_t phiDiff;
207     Double_t etaDiff;
208     Double_t multiplicity;
209
210
211     for (int detectorTrack = 0; detectorTrack < static_cast<int>(phi2.size());
            detectorTrack++) {
212
213         phiDiff = phi1 - phi2[detectorTrack];
214         if (phiDiff < 0) {
```

```
215             phiDiff += 2*TMath::Pi();
216         }
217 
218         etaDiff = eta1 - eta2[detectorTrack];
219         multiplicity = mult1;
220         myHistogram.Fill(phiDiff, etaDiff, multiplicity);
221 
222 
223     }
224 
225 
226 }
227 
228 
229 
230     //Adds class instances of storeInHist, in hindsight maybe I should had just
        defined the operator+ instead
231 void storeInHist::addHistograms(storeInHist secondHistogram) {
232     if (this->_initialised == 0) { //For compatibility with the dummy default
        constructor that is implmented later
233         //Raw
234         this->_storedForwardList = secondHistogram.getForwardHistograms();
235         this->_storedBackwardList = secondHistogram.getBackwardHistograms();
236         this->_storedBackToBackList = secondHistogram.getBackToBackHistograms();
237 
238 
239         //Background
240         this->_noCorrelationForwardList = secondHistogram.getForwardBackgrounds();
241         this->_noCorrelationBackwardList =
                secondHistogram.getBackwardBackgrounds();
242         this->_noCorrelationBackToBackList =
                secondHistogram.getBackToBackBackgrounds();
243 
244 
245         //Normalised
246         this->_processedForwardList = secondHistogram.getForwardProcessed();
247         this->_processedBackwardList = secondHistogram.getBackwardProcessed();
248         this->_processedBackToBackList = secondHistogram.getBackToBackProcessed();
249         this->_eventNumberList = secondHistogram.getEventNumberList();
250         this->_eventNumberListFMD = secondHistogram.getEventNumberListFMD();
251 
252         //Initialisation
253         this->_pathToFile = secondHistogram.getFilePath();
254         this->_initialised = 1;
255 
256 
257 
258     } else {
```

```
259
260         /*
261         The combination of ROOT and C++ is quite annoying here. ROOT only accepts
                pointers/addresses for the native
262         .Add() method instead of something nice like passing by reference. Since
                the compiler complains about
263         'reference to r-value' when I try to directly take the address in shorter
                syntax, I have to make completely
264          new objects just to use the native .Add() method for the TH2D class.
265         */
266
267         std::vector<std::vector<int>> secondEventNumberList =
                secondHistogram.getEventNumberList();
268         std::vector<std::vector<int>> secondEventNumberListFMD =
                secondHistogram.getEventNumberListFMD();
269
270
271         std::vector<std::vector<TH2D>> secondForwardHistogramCopy =
                secondHistogram.getForwardHistograms();
272         std::vector<std::vector<TH2D>> secondBackwardHistogramCopy =
                secondHistogram.getBackwardHistograms();
273         std::vector<std::vector<TH2D>> secondBackToBackHistogramCopy =
                secondHistogram.getBackToBackHistograms();
274
275         std::vector<std::vector<TH2D>> secondForwardBackgroundCopy =
                secondHistogram.getForwardBackgrounds();
276         std::vector<std::vector<TH2D>> secondBackwardBackgroundCopy =
                secondHistogram.getBackwardBackgrounds();
277         std::vector<std::vector<TH2D>> secondBackToBackBackgroundCopy =
                secondHistogram.getBackToBackBackgrounds();
278
279         std::vector<std::vector<TH2D>> secondForwardProcessed =
                secondHistogram.getForwardProcessed();
280         std::vector<std::vector<TH2D>> secondBackwardProcessed =
                secondHistogram.getBackwardProcessed();
281         std::vector<std::vector<TH2D>> secondBackToBackProcessed =
                secondHistogram.getBackToBackProcessed();
282
283
284         //Unfortunately I did not get std::transform() (from include <algorithm>)
                to work
285         //for element wise addition of nested vectors to work so I go with a
                nested for loop.
286         for (int ptNumber = 0; ptNumber <
                static_cast<int>(this->_storedForwardList.size()); ptNumber++) {
287
288             int numberOfEntries =
                    static_cast<int>(this->_storedForwardList[ptNumber].size());
```

```
289             for (int centralityNumber = 0 ; centralityNumber < numberOfEntries;
                    centralityNumber++) {
290
291                 secondEventNumberList[ptNumber][centralityNumber] =
                        secondEventNumberList[ptNumber][centralityNumber] +
                        this->_eventNumberList[ptNumber][centralityNumber];
292
293                 //Raw
294                 this->_storedForwardList[ptNumber][centralityNumber].Add(&secondForwardHistogram
295                 this->_storedBackwardList[ptNumber][centralityNumber].Add(&secondBackwardHistogr
296
297
298                 //Combinatorial and defficiency background (event mixing)
299                 this->_noCorrelationForwardList[ptNumber][centralityNumber].Add(&secondForwardBa
300                 this->_noCorrelationBackwardList[ptNumber][centralityNumber].Add(&secondBackward
301
302
303                 //Normalised data
304                 this->_processedForwardList[ptNumber][centralityNumber].Add(&secondForwardProces
305                 this->_processedBackwardList[ptNumber][centralityNumber].Add(&secondBackwardProc
306
307
308                 if (ptNumber == 0 ) {
309                     this->_storedBackToBackList[0][centralityNumber].Add(&secondBackToBackHistogr
310                     this->_noCorrelationBackToBackList[0][centralityNumber].Add(&secondBackToBack
311                     this->_processedBackToBackList[0][centralityNumber].Add(&secondBackToBackProc
312                     secondEventNumberListFMD[0][centralityNumber] =
                            secondEventNumberListFMD[0][centralityNumber] +
                            this->_eventNumberList[0][centralityNumber];
313                 }
314
315
316
317
318
319             }
320         }
321         this->_eventNumberList = secondEventNumberList;
322
323
324
325     }
326
327
328
329
330 }
331
```

```
332
333     //Stores the various hitograms in a file for later read-in
334 void storeInHist::storeHistogramInFile() {
335     //Sets the new filename
336     const std::string filename =
            _pathToFile.substr(_pathToFile.find_last_of("/")+1, _pathToFile.length());
337     std::string storageLocation = filename.substr(0, filename.find_last_of("."))
            +"Processed"+ ".root";
338
339     //Stores the histogram
340     std::vector<std::vector<TH2D>>* histogramForwardPointer =
            &(this->_storedForwardList);
341     std::vector<std::vector<TH2D>>* histogramBackwardPointer =
            &(this->_storedBackwardList);
342     std::vector<std::vector<TH2D>>* histogramBackToBackPointer =
            &(this->_storedBackToBackList);
343
344     std::vector<std::vector<TH2D>>* backgroundForwardPointer =
            &(this->_noCorrelationForwardList);
345     std::vector<std::vector<TH2D>>* backgroundBackwardPointer =
            &(this->_noCorrelationBackwardList);
346     std::vector<std::vector<TH2D>>* backgroundBackToBackPointer =
            &(this->_noCorrelationBackToBackList);
347
348     std::vector<std::vector<TH2D>>* processedForwardPointer =
            &(this->_processedForwardList);
349     std::vector<std::vector<TH2D>>* processedBackwardPointer =
            &(this->_processedBackwardList);
350     std::vector<std::vector<TH2D>>* processedBackToBackPointer =
            &(this->_processedBackToBackList);
351
352     std::vector<std::vector<int>>* eventNumberListPointer =
            &(this->_eventNumberList);
353     std::vector<std::vector<int>>* eventNumberListFMDPointer =
            &(this->_eventNumberListFMD);
354
355
356     TFile writeData(storageLocation.c_str(), "RECREATE");
357
358     writeData.WriteObject(histogramForwardPointer, "dataForwardHistogram");
359     writeData.WriteObject(histogramBackwardPointer, "dataBackwardHistogram");
360     writeData.WriteObject(histogramBackToBackPointer, "dataBackToBackHistogram");
361
362     writeData.WriteObject(backgroundForwardPointer,
            "dataForwardHistogramBackground");
363     writeData.WriteObject(backgroundBackwardPointer,
            "dataBackwardHistogramBackground");
364     writeData.WriteObject(backgroundBackToBackPointer,
```

```
        "dataBackToBackHistogramBackground");
365
366
367     writeData.WriteObject(processedForwardPointer,
            "dataForwardHistogramProcessed");
368     writeData.WriteObject(processedBackwardPointer,
            "dataBackwardHistogramProcessed");
369     writeData.WriteObject(processedBackToBackPointer,
            "dataBackToBackHistogramProcessed");
370
371     writeData.WriteObject(eventNumberListPointer, "eventNumberList");
372     writeData.WriteObject(eventNumberListFMDPointer, "eventNumberListFMD");
373
374     writeData.Close();
375
376 }
377
378
379     //Sets the error of each bin equal to the square root of that bin.
380 void storeInHist::setErrors() {
381     for (int ptNumber = 0; ptNumber <
            static_cast<int>(this->_storedForwardList.size()); ptNumber++) {
382         for (int centralityNumber = 0; centralityNumber <
                static_cast<int>(this->_storedForwardList[ptNumber].size());
                centralityNumber++) {
383
384             for (int phiBin = 1; phiBin <=
                    this->_storedForwardList[ptNumber][centralityNumber].GetNbinsX();
                    phiBin++) {
385                 for (int etaBin = 1; etaBin <=
                        this->_storedForwardList[ptNumber][centralityNumber].GetNbinsY();
                        etaBin++) {
386
387                     this->_storedForwardList[ptNumber][centralityNumber].SetBinError(phiBin,
                            etaBin,
                            std::sqrt(this->_storedForwardList[ptNumber][centralityNumber].GetBinCont
                            etaBin)));
388                     this->_storedBackwardList[ptNumber][centralityNumber].SetBinError(phiBin,
                            etaBin,
                            std::sqrt(this->_storedBackwardList[ptNumber][centralityNumber].GetBinCon
                            etaBin)));
389
390                     this->_noCorrelationForwardList[ptNumber][centralityNumber].SetBinError(phiBi
                            etaBin,
                            std::sqrt(this->_noCorrelationForwardList[ptNumber][centralityNumber].Get
                            etaBin)));
391                     this->_noCorrelationBackwardList[ptNumber][centralityNumber].SetBinError(phiB
                            etaBin,
```

```
                        std::sqrt(this->_noCorrelationBackwardList[ptNumber][centralityNumber].Ge
                        etaBin)));
392
393                     if (ptNumber == 0) {
394                         this->_storedBackToBackList[ptNumber][centralityNumber].SetBinError(phiBi
                                etaBin,
                                std::sqrt(this->_storedBackToBackList[ptNumber][centralityNumber].GetE
                                etaBin)));
395                         this->_noCorrelationBackToBackList[ptNumber][centralityNumber].SetBinErro
                                etaBin,
                                std::sqrt(this->_noCorrelationBackToBackList[ptNumber][centralityNumbe
                                etaBin)));
396                     }
397                 }
398             }
399         }
400     }
401
402
403 }
404
405
406
407     //Sets the error of all bins to 0 (mostly just for debugging purposes)
408 void storeInHist::setErrors0() {
409     for (int ptNumber = 0; ptNumber <
            static_cast<int>(this->_storedForwardList.size()); ptNumber++) {
410         for (int centralityNumber = 0; centralityNumber <
                static_cast<int>(this->_storedForwardList[ptNumber].size());
                centralityNumber++) {
411
412             for (int phiBin = 1; phiBin <=
                    this->_storedForwardList[ptNumber][centralityNumber].GetNbinsX();
                    phiBin++) {
413                 for (int etaBin = 1; etaBin <=
                        this->_storedForwardList[ptNumber][centralityNumber].GetNbinsY();
                        etaBin++) {
414
415                     this->_storedForwardList[ptNumber][centralityNumber].SetBinError(phiBin,
                            etaBin, 1);
416                     this->_storedBackwardList[ptNumber][centralityNumber].SetBinError(phiBin,
                            etaBin, 1);
417
418                     this->_noCorrelationForwardList[ptNumber][centralityNumber].SetBinError(phiBi
                            etaBin, 1);
419                     this->_noCorrelationBackwardList[ptNumber][centralityNumber].SetBinError(phiB
                            etaBin, 1);
420
```

```
421                     if (ptNumber == 0) {
422                         this->_storedBackToBackList[ptNumber][centralityNumber].SetBinError(phiBi
                                etaBin, 1);
423                         this->_noCorrelationBackToBackList[ptNumber][centralityNumber].SetBinErro
                                etaBin, 1);
424                     }
425                 }
426             }
427         }
428     }
429
430
431 }
432
433
434
435
436     //Performs a normalisation w.r.t event mixing and number of tracks
437     //for the raw data and stores the results in a separate member variable.
438     //See the general explanation in the readme file for why [0] instead of
            [ptIndex]
439     //is used in some places.
440     //Also note that .Sumw2() from ROOT was not used on purpose.
441 void storeInHist::loadProcessed() {
442     if (this->_initialised == 0) {
443         throw(std::logic_error("Error: Trying to process unloaded histograms"));
444     }
445
446     std::vector<std::vector<TH2D>> histogramForward =
            this->getForwardHistograms();
447     std::vector<std::vector<TH2D>> histogramBackward =
            this->getBackwardHistograms();
448     std::vector<std::vector<TH2D>> histogramBackToBack =
            this->getBackToBackHistograms();
449
450     std::vector<std::vector<TH2D>> histogramForwardBackground =
            this->getForwardBackgrounds();
451     std::vector<std::vector<TH2D>> histogramBackwardBackground =
            this->getBackwardBackgrounds();
452     std::vector<std::vector<TH2D>> histogramBackToBackBackground =
            this->getBackToBackBackgrounds();
453
454     std::vector<std::vector<TH2D>> processedForward;
455     std::vector<std::vector<TH2D>> processedBackward;
456     std::vector<std::vector<TH2D>> processedBackToBack;
457
458     std::vector<TH2D> placeHolderVector;
459     TH2D placeHolderHistogram;
```

```
460     TH2D histogramForwardCopy;
461     TH2D histogramBackwardCopy;
462     TH2D histogramBackToBackCopy;
463
464     double errorFactor;
465
466     for (int ptNumber = 0; ptNumber <
            static_cast<int>(this->_storedForwardList.size()); ptNumber++) {
467         processedForward.push_back(placeHolderVector);
468         processedBackward.push_back(placeHolderVector);
469         if (ptNumber == 0) {
470             processedBackToBack.push_back(placeHolderVector);
471         }
472
473
474         int numberOfEntries =
                static_cast<int>(this->_storedForwardList[ptNumber].size());
475         for (int centralityNumber = 0 ; centralityNumber < numberOfEntries;
                centralityNumber++) {
476             Double_t maxValueForward =
                    histogramForwardBackground[ptNumber][centralityNumber].GetMaximum();
477             Double_t maxValueBackward =
                    histogramBackwardBackground[ptNumber][centralityNumber].GetMaximum();
478
479
480             // *= syntax does not seem to be implemented for TH2D
481             TH2D normalisedForwardBackground =
                    (1/maxValueForward)*histogramForwardBackground[ptNumber][centralityNumber];
482             TH2D normalisedBackwardBackground =
                    (1/maxValueBackward)*histogramBackwardBackground[ptNumber][centralityNumber];
483
484
485             histogramForwardCopy = histogramForward[ptNumber][centralityNumber];
486             histogramBackwardCopy = histogramBackward[ptNumber][centralityNumber];
487             histogramForward[ptNumber][centralityNumber].Divide(&normalisedForwardBackground);
488             histogramBackward[ptNumber][centralityNumber].Divide(&normalisedBackwardBackground);
489
490
491
492             for (int phiBin = 1; phiBin <= histogramForwardCopy.GetNbinsX();
                    phiBin++) {
493                 for (int etaBin = 1; etaBin <= histogramForwardCopy.GetNbinsY();
                        etaBin++) {
494                     if ((histogramForwardCopy.GetBinContent(phiBin, etaBin) != 0)
                            && (normalisedForwardBackground.GetBinContent(phiBin,
                            etaBin) != 0)) {
495                         errorFactor =
                                std::pow(histogramForwardCopy.GetBinError(phiBin,
```

```
                    etaBin)/histogramForwardCopy.GetBinContent(phiBin,
                    etaBin),2);
496                 errorFactor +=
                    std::pow(normalisedForwardBackground.GetBinError(phiBin,
                    etaBin)/normalisedForwardBackground.GetBinContent(phiBin,
                    etaBin), 2) ;
497                 errorFactor = std::sqrt(errorFactor);
498
499
500                 histogramForward[ptNumber][centralityNumber].SetBinError(phiBin,
                    etaBin,
                    histogramForward[ptNumber][centralityNumber].GetBinContent(phiBin,
                    etaBin) * errorFactor);
501
502
503
504             }
505
506             if ((histogramBackwardCopy.GetBinContent(phiBin, etaBin) != 0)
                && (normalisedBackwardBackground.GetBinContent(phiBin,
                etaBin) != 0)) {
507                 errorFactor =
                    std::pow(histogramBackwardCopy.GetBinError(phiBin,
                    etaBin)/histogramBackwardCopy.GetBinContent(phiBin,
                    etaBin),2);
508                 errorFactor +=
                    std::pow(normalisedBackwardBackground.GetBinError(phiBin,
                    etaBin)/normalisedBackwardBackground.GetBinContent(phiBin,
                    etaBin), 2) ;
509                 errorFactor = std::sqrt(errorFactor);
510
511                 histogramBackward[ptNumber][centralityNumber].SetBinError(phiBin,
                    etaBin,
                    histogramBackward[ptNumber][centralityNumber].GetBinContent(phiBin,
                    etaBin) * errorFactor);
512             }
513
514         }
515
516     }
517
518
519
520     int tpcTracksNormalisation =
            this->_eventNumberList[ptNumber][centralityNumber];
521     //int fmdTracksNormalisation =
            this->_eventNumberListFMD[0][centralityNumber]; //For debugging
            purposes
```

```
522
523             TH2D forwardTemp = histogramForward[ptNumber][centralityNumber];
524             TH2D backwardTemp = histogramBackward[ptNumber][centralityNumber];
525
526             forwardTemp = forwardTemp*(1.0/tpcTracksNormalisation);
527             backwardTemp = backwardTemp*(1.0/tpcTracksNormalisation);
528
529
530
531             processedForward[ptNumber].push_back(forwardTemp);
532             processedBackward[ptNumber].push_back(backwardTemp);
533
534             //Stores the processed histograms
535
536             if (ptNumber == 0) {
537                 Double_t maxValueBackToBack =
                        histogramBackToBackBackground[ptNumber][centralityNumber].GetMaximum();
538                 TH2D normalisedBackToBackBackground =
                        (1/maxValueBackToBack)*histogramBackToBackBackground[ptNumber][centralityNumb
539
540                 histogramBackToBackCopy =
                        histogramBackToBack[ptNumber][centralityNumber];
541                 histogramBackToBack[ptNumber][centralityNumber].Divide(&normalisedBackToBackBack
542
543
544
545
546                 for (int phiBin = 1; phiBin <=
                        histogramBackToBackCopy.GetNbinsX(); phiBin++) {
547                     for (int etaBin = 1; etaBin <=
                            histogramBackToBackCopy.GetNbinsY(); etaBin++) {
548                         if ((histogramBackToBackCopy.GetBinContent(phiBin, etaBin)
                                != 0) &&
                                (normalisedBackToBackBackground.GetBinContent(phiBin,
                                etaBin) != 0)) {
549                             errorFactor =
                                    std::pow(histogramBackToBackCopy.GetBinError(phiBin,
                                    etaBin)/histogramBackToBackCopy.GetBinContent(phiBin,
                                    etaBin),2);
550                             errorFactor +=
                                    std::pow(normalisedBackToBackBackground.GetBinError(phiBin,
                                    etaBin)/normalisedBackToBackBackground.GetBinContent(phiBin,
                                    etaBin), 2) ;
551                             errorFactor = std::sqrt(errorFactor);
552
553
554                             histogramBackToBack[ptNumber][centralityNumber].SetBinError(phiBin,
                                    etaBin,
```

```
                                histogramBackToBack[ptNumber][centralityNumber].GetBinContent(phiB
                                etaBin) * errorFactor);
555
556
557                         }
558
559
560                     }
561
562                 }
563
564                 TH2D backToBackTemp =
                        histogramBackToBack[ptNumber][centralityNumber];
565                 backToBackTemp = backToBackTemp*(1.0/tpcTracksNormalisation);
566                 processedBackToBack[ptNumber].push_back(backToBackTemp);
567
568             }
569
570
571         }
572
573     }
574
575     this->_processedForwardList = processedForward;
576     this->_processedBackwardList = processedBackward;
577     this->_processedBackToBackList = processedBackToBack;
578
579
580
581
582 }
583
584
585
586
587
588 //Constructors
589
590
591     //Dummy default constructor since the default constructor used to be taken by
           inheritance from TObject in an earlier implementation
592 storeInHist::storeInHist(Int_t number) {
593     this->_initialised = 0;
594     this->_pathToFile = "";
595     (void)number; //Deliberately not using static_cast<void>(number) since I do
           not care what happens to the number,
596                   //it is just to turn off warnings of unused variables.
597 }
```

```
598
599
600     //Initialisation by reading a file
601 storeInHist::storeInHist(std::string pathToFile) : _pathToFile{pathToFile} {
602     TFile dataFile(pathToFile.c_str(), "dataFile", "READ");
603
604     std::vector<std::vector<TH2D>>* histogramForward =
            (std::vector<std::vector<TH2D>>*)dataFile.Get("dataForwardHistogram");
605     std::vector<std::vector<TH2D>>* histogramBackward =
            (std::vector<std::vector<TH2D>>*)dataFile.Get("dataBackwardHistogram");
606     std::vector<std::vector<TH2D>>* histogramBackToBack =
            (std::vector<std::vector<TH2D>>*)dataFile.Get("dataBackToBackHistogram");
607
608     std::vector<std::vector<TH2D>>* histogramForwardBackground =
            (std::vector<std::vector<TH2D>>*)dataFile.Get("dataForwardHistogramBackground");
609     std::vector<std::vector<TH2D>>* histogramBackwardBackground =
            (std::vector<std::vector<TH2D>>*)dataFile.Get("dataBackwardHistogramBackground");
610     std::vector<std::vector<TH2D>>* histogramBackToBackBackground =
            (std::vector<std::vector<TH2D>>*)dataFile.Get("dataBackToBackHistogramBackground");
611     std::vector<std::vector<int>>* eventNumberListPtr =
            (std::vector<std::vector<int>>*)dataFile.Get("eventNumberList");
612     std::vector<std::vector<int>>* eventNumberListFMDPtr =
            (std::vector<std::vector<int>>*)dataFile.Get("eventNumberListFMD");
613
614     this->_storedForwardList = *histogramForward;
615     this->_storedBackwardList = *histogramBackward;
616     this->_storedBackToBackList = *histogramBackToBack;
617
618     this->_noCorrelationForwardList = *histogramForwardBackground;
619     this->_noCorrelationBackwardList = *histogramBackwardBackground;
620     this->_noCorrelationBackToBackList = *histogramBackToBackBackground;
621
622     this->_eventNumberList = *eventNumberListPtr;
623     this->_eventNumberListFMD = *eventNumberListFMDPtr;
624
625
626     //Old versions of the program did not save the processed histograms as member
            variables
627     // nor in the file. Not reading in processed histograms directly is for
            backwards compatibility.
628     try {
629         std::vector<std::vector<TH2D>>* histogramForwardProcessed =
                (std::vector<std::vector<TH2D>>*)dataFile.Get("dataForwardHistogramProcessed");
630         std::vector<std::vector<TH2D>>* histogramBackwardProcessed =
                (std::vector<std::vector<TH2D>>*)dataFile.Get("dataBackwardHistogramProcessed");
631         std::vector<std::vector<TH2D>>* histogramBackToBackProcessed =
                (std::vector<std::vector<TH2D>>*)dataFile.Get("dataBackToBackHistogramProcessed");
632
```

```
633         this->_processedForwardList = *histogramForwardProcessed;
634         this->_processedBackwardList = *histogramBackwardProcessed;
635         this->_processedBackToBackList = *histogramBackToBackProcessed;
636 
637 
638 
639 
640 
641 
642     } catch (...) {
643 
644         this->loadProcessed();
645 
646     }
647 
648 
649 
650     this->_pathToFile = pathToFile.c_str();
651     this->_initialised = 1;
652     dataFile.Close();
653 
654 
655 }
656 
657 
658 
659 
660 
661     //Primary Constructor
662 storeInHist::storeInHist(std::string pathToFile, Short_t cutOption,
663                                                  Double_t centralityMin, Double_t
                                                         centralityMax,
664                                                  Double_t ptMin, Double_t ptMax,
665                                                  Double_t etaMin, Double_t etaMax,
666                                                  Int_t countsPhi, Int_t countsEta,
667                                                  Int_t start, Int_t stop) :
                                                         _pathToFile{pathToFile} {
668 
669 
670     //Gets the number of entries in the tree
671     TFile dataFile(pathToFile.c_str(), "dataFile", "READ");
672     TTree* dataTree = (TTree*)dataFile.Get("LWTree");
673     Int_t dataCount = dataTree->GetEntries();
674     dataFile.Close();
675     if (stop > dataCount) {
676         stop = dataCount; //To avoid segmentation errors
677         throw(std::invalid_argument("The stop number cannot be greater than the
                 number of events. Please compile entries.cpp to check the number of
```

```
        entries"));
678 }
679
680 if (start < 0) {
681     start = 0;
682     throw(std::invalid_argument("The start number cannot be smaller than 0"));
683
684 }
685
686
687 //loadHistogram creates the histograms that are wanted for the different
        cases. See the readme file for
688 //a more detailed explanation. loadHistograms() is a separate method as it is
        very long and involved.
689 std::tuple< std::vector<std::vector<std::vector<TH2D>>>,
690     std::vector<std::vector<int>>, std::vector<std::vector<int>> >
        returnVector = loadHistograms(pathToFile, cutOption,
691                                                                          centr
                                                                              c
692                                                                          ptMir
                                                                              p
693                                                                          etaMi
                                                                              e
694                                                                          count
                                                                              c
695                                                                          start
                                                                              s
696
697
698
699
700 this->_storedForwardList = std::get<0>(returnVector)[0];
701 this->_storedBackwardList = std::get<0>(returnVector)[1];
702 this->_storedBackToBackList = std::get<0>(returnVector)[2];
703
704 this->_noCorrelationForwardList = std::get<0>(returnVector)[3];
705 this->_noCorrelationBackwardList = std::get<0>(returnVector)[4];
706 this->_noCorrelationBackToBackList = std::get<0>(returnVector)[5];
707
708 this->_eventNumberList = std::get<1>(returnVector);
709 this->_eventNumberListFMD = std::get<2>(returnVector);
710
711
712
713 this->_initialised = 1;
714 this->loadProcessed(); //This needs to go after _initialised is set to 1
715                        //since loadProcessed() checks the initialisation status.
716 storeHistogramInFile();
```

```
717
718
719 }
720
721
722
723 //This is the main work horse of the program. It reads in all of the desired
       histograms as well
724 //as the number of tracks. See the comments in readData.cpp if you want to know
       what the arguments do.
725 //See the readme file for an actual explanation of what is going on, since it is
       somewhat involved
726 //I will not put an explanation as inline comments.
727 std::tuple< std::vector<std::vector<std::vector<TH2D>>>,
       std::vector<std::vector<int>>, std::vector<std::vector<int>> >
       storeInHist::loadHistograms(std::string pathToFile, Short_t cutOption,
728                                         Double_t centralityMin, Double_t
                                             centralityMax,
729                                         Double_t ptMin, Double_t ptMax,
730                                         Double_t etaMin, Double_t etaMax,
731                                         Int_t countsPhi, Int_t countsEta,
732                                         Int_t start, Int_t stop) {
733
734     //Parameters
735     Int_t numberOlderEventsToSave = 5;
736
737
738
739     /*
740     As mentioned in readData.cpp, a lot of the arguments passed along as
           arguments to this function
741     are deprecated. If there was more time, I should of course had cleaned up the
           syntax but right
742     now cleaning up the long list of dependencies is not a prioirty.
743     */
744
745
746     (void)cutOption; //outdated variable, left in case it should be reimplemented
           later. Deliberaltely not using static_cast
747                      //as it is supposed to be discarded.
748
749
750     //If there was more time these intervals should probably be determined by an
           argument
751     //given in readData.cpp.
752     std::vector<Double_t> startOfPtIntervals {1, 1.5, 2, 2.5, 3, 3.5, 4, 5, 6};
           //{0.2, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 5, 6
753     if (ptMin < startOfPtIntervals[0]) {
```

```
754         ptMin = startOfPtIntervals[0];
755     }
756     if (ptMax > startOfPtIntervals[startOfPtIntervals.size()-1]) {
757         ptMax = startOfPtIntervals[startOfPtIntervals.size()-1];
758     }
759 
760     std::vector<Double_t> startOfCentralityIntervals {50, 60, 65, 70, 75, 80, 85,
        90};
761     int numberOfEntriesCentrality =
        static_cast<int>(startOfCentralityIntervals.size());
762     int numberOfEntriesPt = static_cast<int>(startOfPtIntervals.size());
763 
764 
765 
766 
767 
768 
769 
770 
771 
772 
773 
774 
775 
776 
777 
778 
779 
780     /*
781     IIRC the vector class is just a wrapper around pointers, so making nested
        vectors
782     to represent matrices does not actually declare objects. Unfortunately, this
        means that
783     there has to be an entire block of code below to assign pointers to objects
        just to
784     make the syntax workable later on.
785     */
786 
787         //Data to be stored
788     std::vector<std::vector<TH2D>> forwardVector;
789     std::vector<std::vector<TH2D>> backwardVector;
790     std::vector<std::vector<TH2D>> backToBackVector;
791 
792     std::vector<std::vector<TH2D>> forwardBackgroundVector;
793     std::vector<std::vector<TH2D>> backwardBackgroundVector;
794     std::vector<std::vector<TH2D>> backToBackBackgroundVector;
795 
796     std::vector<std::vector<int>> eventNumbers;
```

```
797     std::vector<std::vector<int>> eventNumbersFMD;
798
799     std::vector<int> eventNumbersPlaceHolder;
800     std::vector<TH2D> placeHolderVector;
801     TH2D placeHolderHistogram("histogram", "Counts", countsPhi, 0, 2*TMath::Pi(),
            countsEta, etaMin, etaMax);
802
803         //Things needed for the event mixing
804     std::vector<std::vector<std::vector<std::vector<Double_t>>>>
            oldPhiTracksTPCvector;
805     std::vector<std::vector<std::vector<std::vector<Double_t>>>>
            oldEtaTracksTPCvector;
806     std::vector<std::vector<std::vector<std::vector<Double_t>>>>
            oldPhiTracksBackwardFMDvector;
807     std::vector<std::vector<std::vector<std::vector<Double_t>>>>
            oldEtaTracksBackwardFMDvector;
808     std::vector<std::vector<std::vector<std::vector<Int_t>>>>
            oldMultiplicityTracksBackwardFMDvector;
809
810     std::vector<std::vector<std::vector<Double_t>>>
            oldPhiTracksTPCvectorPlaceHolder;
811     std::vector<std::vector<std::vector<Double_t>>>
            oldEtaTracksTPCvectorPlaceHolder;
812     std::vector<std::vector<std::vector<Double_t>>>
            oldPhiTracksBackwardFMDvectorPlaceHolder;
813     std::vector<std::vector<std::vector<Double_t>>>
            oldEtaTracksBackwardFMDvectorPlaceHolder;
814     std::vector<std::vector<std::vector<Int_t>>>
            oldMultiplicityTracksBackwardFMDvectorPlaceHolder;
815
816
817     std::vector<std::vector<Double_t>> oldPhiTracksTPC;
818     std::vector<std::vector<Double_t>> oldEtaTracksTPC;
819     std::vector<std::vector<Double_t>> oldPhiTracksBackwardFMD;
820     std::vector<std::vector<Double_t>> oldEtaTracksBackwardFMD;
821     std::vector<std::vector<Int_t>> oldMultiplicityTracksBackwardFMD;
822
823
824     std::vector<std::vector<std::vector<Double_t>>> tracksPhiTPCvector;
825     std::vector<std::vector<std::vector<Double_t>>> tracksEtaTPCvector;
826
827     std::vector<std::vector<Double_t>> tracksPhiTPCvectorPlaceHolder;
828     std::vector<std::vector<Double_t>> tracksEtaTPCvectorPlaceHolder;
829
830     //To avoid excessive copying of the same data over and over again which takes
            time,
831     //I predefine the length of the vectors to somewhat above what the largest
            number of tracks seem to be
```

```
832     //in the FMD and TPC respectively
833     int reserveNumberFMD = 500;
834     int reserveNumberTPC = 150000;
835 
836     std::vector<Double_t> forwardTracksPhi;
837     std::vector<Double_t> backwardTracksPhi;
838     std::vector<Double_t> forwardTracksEta;
839     std::vector<Double_t> backwardTracksEta;
840     std::vector<Int_t> forwardTracksMult;
841     std::vector<Int_t> backwardTracksMult;
842     forwardTracksPhi.reserve(reserveNumberFMD);
843     backwardTracksPhi.reserve(reserveNumberFMD);
844     forwardTracksEta.reserve(reserveNumberFMD);
845     backwardTracksEta.reserve(reserveNumberFMD);
846     forwardTracksMult.reserve(reserveNumberFMD);
847     backwardTracksMult.reserve(reserveNumberFMD);
848 
849     std::vector<Double_t> tracksPhiTPC;
850     std::vector<Double_t> tracksEtaTPC;
851     tracksPhiTPC.reserve(reserveNumberTPC);
852     tracksEtaTPC.reserve(reserveNumberTPC);
853 
854 
855 
856     for (int ptNumber = 0; ptNumber < numberOfEntriesPt -1 ; ptNumber++) { //The
            last interval is 85-90, hence the -1
857         forwardVector.push_back(placeHolderVector);
858         backwardVector.push_back(placeHolderVector);
859         forwardBackgroundVector.push_back(placeHolderVector);
860         backwardBackgroundVector.push_back(placeHolderVector);
861 
862 
863         tracksPhiTPCvector.push_back(tracksPhiTPCvectorPlaceHolder);
864         tracksEtaTPCvector.push_back(tracksEtaTPCvectorPlaceHolder);
865         oldPhiTracksTPCvector.push_back(oldPhiTracksTPCvectorPlaceHolder);
866         oldEtaTracksTPCvector.push_back(oldEtaTracksTPCvectorPlaceHolder);
867 
868         eventNumbers.push_back(eventNumbersPlaceHolder);
869 
870 
871         if (ptNumber == 0) { //There are no pT cuts in the FMD:s, structure is
                kept for consistency
872             backToBackVector.push_back(placeHolderVector);
873             backToBackBackgroundVector.push_back(placeHolderVector);
874 
875             oldPhiTracksBackwardFMDvector.push_back(oldPhiTracksBackwardFMDvectorPlaceHolder);
876             oldEtaTracksBackwardFMDvector.push_back(oldEtaTracksBackwardFMDvectorPlaceHolder);
877             oldMultiplicityTracksBackwardFMDvector.push_back(oldMultiplicityTracksBackwardFMDvec
```

```
878             eventNumbersFMD.push_back(eventNumbersPlaceHolder);
879         }
880 
881 
882 
883         for (int centralityNumber = 0 ; centralityNumber <
                numberOfEntriesCentrality -1; centralityNumber++) { //-1 is intentional
884 
885             forwardVector[ptNumber].push_back(placeHolderHistogram);
886             backwardVector[ptNumber].push_back(placeHolderHistogram);
887             forwardBackgroundVector[ptNumber].push_back(placeHolderHistogram);
888             backwardBackgroundVector[ptNumber].push_back(placeHolderHistogram);
889 
890 
891             oldPhiTracksTPCvector[ptNumber].push_back(oldPhiTracksTPC);
892             oldEtaTracksTPCvector[ptNumber].push_back(oldEtaTracksTPC);
893             tracksPhiTPCvector[ptNumber].push_back(tracksPhiTPC);
894             tracksEtaTPCvector[ptNumber].push_back(tracksEtaTPC);
895 
896 
897             eventNumbers[ptNumber].push_back(0);
898 
899 
900             if (ptNumber == 0) {
901                 backToBackVector[0].push_back(placeHolderHistogram);
902                 backToBackBackgroundVector[0].push_back(placeHolderHistogram);
903 
904                 oldPhiTracksBackwardFMDvector[0].push_back(oldPhiTracksBackwardFMD);
905                 oldEtaTracksBackwardFMDvector[0].push_back(oldEtaTracksBackwardFMD);
906                 oldMultiplicityTracksBackwardFMDvector[0].push_back(oldMultiplicityTracksBackwar
907 
908                 eventNumbersFMD[ptNumber].push_back(0);
909             }
910 
911         }
912 
913 
914     }
915 
916 
917 
918 
919 
920 
921 
922 
923 
924
```

```
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939     //Opens the data
940     TFile dataFile(pathToFile.c_str(), "dataFile", "READ");
941     TTree* dataTree = (TTree*)dataFile.Get("LWTree");
942
943
944     //Creates variables to write the read-in variables to.
945     AliLWEvent* event = new AliLWEvent; //Not great with raw pointers, but ROOT
            requires pointers and refuses smart pointers for some reason
946     TClonesArray* tpcTrack = new TClonesArray("AliLWTPCTrack");
947     TClonesArray* fmdTrack = new TClonesArray("AliLWFMDTrack");
948     dataTree->SetBranchAddress("Event", &event);
949     dataTree->SetBranchAddress("TPCTracks", &tpcTrack);
950     dataTree->SetBranchAddress("FMDTracks", &fmdTrack);
951
952
953     //Variables for looping
954     Int_t trackCountTPC;
955     Int_t trackCountFMD;
956     AliLWTPCTrack* currentTrackTPC;
957     AliLWFMDTrack* currentTrackFMD;
958
959     //Variables for keeping track of which category data is to be saved to
960     int centralityIndex;
961     int ptIndex;
962
963
964     //Event-variables
965     Double_t centrality;
966
967     //TPC-variables
968     Double_t phiValTPC;
969     Double_t etaValTPC;
970     Short_t cutFlag;
971     Double_t pT; //Tranvsverse momentum
```

```
972
973
974     //FMD-variables
975     Double_t phiValFMD; //For TPC-FMD correlations
976     Double_t etaValFMD;
977     Int_t fmdMultiplicity;
978
979
980     //Values to be stored in the histograms
981     Double_t etaDiff; //For TPC-FMD correlations
982     Double_t phiDiff;
983
984
985
986
987
988
989
990     //Event-loop
991     for (Int_t eventNumber = start; eventNumber < stop; eventNumber++) {
992         //Reads in the data for the event
993         dataTree->GetEntry(eventNumber);
994
995
996         //Determines which interval the centrality belongs to and skips it if it
                isn't wanted.
997         centrality = event->fCent;
998         if ((centrality < centralityMin) || (centrality > centralityMax)) {
999             continue;
1000        } else {
1001            for (int centralityNumber = 0; centralityNumber <
                    numberOfEntriesCentrality -1; centralityNumber++) {
1002                if ((centrality >= startOfCentralityIntervals[centralityNumber])
                        && (centrality <
                        startOfCentralityIntervals[centralityNumber+1])) {
1003                    centralityIndex = centralityNumber;
1004                    break;
1005                }
1006            }
1007
1008        }
1009
1010
1011        //Clears data from previous events
1012        forwardTracksEta.clear();
1013        backwardTracksEta.clear();
1014        forwardTracksPhi.clear();
1015        backwardTracksPhi.clear();
```

```
1016        forwardTracksMult.clear();
1017        backwardTracksMult.clear();
1018        for (int ptNumber = 0; ptNumber < numberOfEntriesPt - 1; ptNumber++) {
                //-1 is intentional
1019            tracksPhiTPCvector[ptNumber][centralityIndex].clear();
1020            tracksEtaTPCvector[ptNumber][centralityIndex].clear();
1021        }
1022
1023
1024        //Number of events to loop over in the FMD and TPC
1025        trackCountTPC = tpcTrack->GetEntries();
1026        trackCountFMD = fmdTrack->GetEntries();
1027
1028
1029        //FMD loop start
1030        for (Int_t fmdTrackNumber = 0; fmdTrackNumber < trackCountFMD;
                fmdTrackNumber++) {
1031            //Gets details about the track
1032            currentTrackFMD =
                    static_cast<AliLWFMDTrack*>((*fmdTrack)[fmdTrackNumber]);
1033            etaValFMD = currentTrackFMD->fEta;
1034
1035            //Cutting away data where the resolution is low,
1036            //if there was more time this should probably had been defined in
                    readData.cpp
1037            if ((etaValFMD < -3.1) || (etaValFMD > -2)) {
1038                if ((etaValFMD < 3.8) || (etaValFMD > 4.7)) {
1039                    if ((etaValFMD < 2.5) || (etaValFMD > 3.1)) {
1040                        continue;
1041
1042                    }
1043                }
1044            }
1045
1046            //Reads in these values after doing the eta check to save time
1047            //if there are values which are skipped
1048            fmdMultiplicity = currentTrackFMD->fMult;
1049            phiValFMD = currentTrackFMD->fPhi;
1050
1051            //Stores values for the forward and backward FMD-tracks.
1052            if (etaValFMD >= 0) {
1053                forwardTracksPhi.push_back(phiValFMD);
1054                forwardTracksEta.push_back(etaValFMD);
1055                forwardTracksMult.push_back(fmdMultiplicity);
1056
1057            } else {
1058                backwardTracksPhi.push_back(phiValFMD);
1059                backwardTracksEta.push_back(etaValFMD);
```

```
1060            backwardTracksMult.push_back(fmdMultiplicity);
1061
1062        }
1063
1064
1065        /*
1066        The code grew sort of organically as more and more features had to be
                implemented.
1067        Originally, there was no event mixing so I made a nested for loop of
                FMD and TPC
1068        tracks. In the current implementation, tracks have to be saved anyhow
                since
1069        they are needed in the event mixing, so the faster option would be to
                'un-nest'
1070        these loops and have both the TPC and event mixing in separate loops.
1071        Since the event mixing is by far the biggest time sink, and since that
                would take equally
1072        long unnested, I have not bothered with un-nesting the loops. If one
                were to write
1073        it from scatch with all the hindsight, these loops should of course be
                written in an unnested
1074        way as that would be much simpler and the logic would be much easier
                to follow for un-nested loops.
1075        */
1076
1077        //Loops through all tracks in the TPC so TPC-FMD correlations can be
                calculated
1078        for (Int_t tpcTrackNumber = 0; tpcTrackNumber < trackCountTPC;
                tpcTrackNumber++) {
1079            currentTrackTPC =
                    static_cast<AliLWTPCTrack*>((*tpcTrack)[tpcTrackNumber]);
1080            cutFlag = currentTrackTPC->fTrFlag;
1081            //Cutting away data where with the wrong flag(s).
1082            if (!(cutFlag & 2)) {
1083                continue;
1084            }
1085
1086            //Cutting away data where the resolution is low
1087            etaValTPC = currentTrackTPC->fEta;
1088            if ((etaValTPC < -0.75) || (etaValTPC > 0.75)) {
1089                continue;
1090            }
1091
1092            //Cuts away unwanted pT:s
1093            pT = currentTrackTPC->fPt;
1094            if ((pT < ptMin) || (pT > ptMax)) {
1095                continue;
1096            }
```

```
1097
1098            for (int ptNumber = 0; ptNumber < numberOfEntriesPt -1;
                    ptNumber++) {
1099                if ( (pT >= startOfPtIntervals[ptNumber]) && (pT <
                        startOfPtIntervals[ptNumber+1]) ) {
1100                    ptIndex = ptNumber;
1101                    break;
1102                }
1103            }
1104
1105
1106
1107            phiValTPC = currentTrackTPC->fPhi;
1108            phiDiff = phiValFMD - phiValTPC;
1109            etaDiff = etaValFMD - etaValTPC;
1110
1111            //Makes sures all values are positive (we want all values within
                    one period)
1112            if (phiDiff < 0) {phiDiff += 2*TMath::Pi();}
1113
1114
1115            if (fmdTrackNumber == 0) {
1116                eventNumbers[ptIndex][centralityIndex] += 1;
1117            }
1118
1119            //Stores the read-in and approved values for later event mixing
1120
1121            tracksPhiTPCvector[ptIndex][centralityIndex].push_back(phiValTPC);
1122            tracksEtaTPCvector[ptIndex][centralityIndex].push_back(etaValTPC);
1123
1124
1125            //Fills the correct histogram. The sign determines if it is the
                    forwards or backwards FMD
1126            if (etaValFMD > 0) {
1127                forwardVector[ptIndex][centralityIndex].Fill(phiDiff, etaDiff,
                        fmdMultiplicity);
1128
1129            } else {
1130                backwardVector[ptIndex][centralityIndex].Fill(phiDiff, etaDiff,
                        fmdMultiplicity);
1131
1132            }
1133
1134
1135
1136        } //TPC-loop end
1137
1138
```

```
1139
1140
1141
1142            //Event mixing start
1143            //FMD-oldTPC correlations
1144            if (etaValFMD > 0) {
1145                for (int ptNumber = 0; ptNumber < numberOfEntriesPt-1; ptNumber++)
                        {
1146                    for (int oldEventNumber = 0; oldEventNumber <
                            static_cast<int>(oldPhiTracksTPCvector[ptNumber][centralityIndex].size())
                            oldEventNumber++) {
1147
1148                        calculateSingleCorrelation(forwardBackgroundVector[ptNumber][centralityIn
                                phiValFMD, etaValFMD,
1149                                    oldPhiTracksTPCvector[ptNumber][centralityIndex][oldEventNu
1150                                    oldEtaTracksTPCvector[ptNumber][centralityIndex][oldEventNu
1151                                    fmdMultiplicity);
1152
1153                    }
1154
1155                }
1156            } else {
1157                for (int ptNumber = 0; ptNumber < numberOfEntriesPt-1; ptNumber++)
                        {
1158                    for (int oldEventNumber = 0; oldEventNumber <
                            static_cast<int>(oldPhiTracksTPCvector[ptNumber][centralityIndex].size())
                            oldEventNumber++) {
1159                        calculateSingleCorrelation(backwardBackgroundVector[ptNumber][centralityI
                                phiValFMD, etaValFMD,
1160                                    oldPhiTracksTPCvector[ptNumber][centralityIndex][oldEventNu
1161                                    oldEtaTracksTPCvector[ptNumber][centralityIndex][oldEventNu
1162                                    fmdMultiplicity);
1163                    }
1164
1165                }
1166            }
1167
1168            //forwardFMD-oldBackwardFMD
1169            if (etaValFMD > 0 ) {
1170                for (int oldEventNumber = 0; oldEventNumber <
                        static_cast<int>(oldPhiTracksBackwardFMDvector[0][centralityIndex].size());
                        oldEventNumber++) {
1171                    calculateSingleCorrelation(backToBackBackgroundVector[0][centralityIndex],
                            phiValFMD, etaValFMD,
1172                                    oldPhiTracksBackwardFMDvector[0][centralityIndex][oldEventN
                                            oldEtaTracksBackwardFMDvector[0][centralityIndex][oldEv
1173                                    fmdMultiplicity,
                                            oldMultiplicityTracksBackwardFMDvector[0][centralityInd
```

```
1174
1175                }
1176            }
1177
1178
1179            //Event Mixing End
1180
1181            //Keeping track of how many tracks that are approved for later
                    normalisation
1182            eventNumbersFMD[0][centralityIndex] += 1;
1183        } //FMD-loop end
1184
1185
1186
1187        //Calculates FMD-FMD correlations
1188        calculateCorrelation(backToBackVector[0][centralityIndex],
                forwardTracksPhi, forwardTracksEta,
1189                             backwardTracksPhi, backwardTracksEta,
1190                             forwardTracksMult, backwardTracksMult);
1191
1192
1193
1194
1195
1196
1197
1198        //Updates the stored old events with the current one and removes the
                oldest one
1199        for (int ptNumber = 0; ptNumber < numberOfEntriesPt-1; ptNumber++) {
1200            if
                    (static_cast<int>(tracksPhiTPCvector[ptNumber][centralityIndex].size())
                    >= 1) {
1201                oldPhiTracksTPCvector[ptNumber][centralityIndex].push_back(tracksPhiTPCvector[pt
1202                oldEtaTracksTPCvector[ptNumber][centralityIndex].push_back(tracksEtaTPCvector[pt
1203            }
1204
1205            if
                    (static_cast<int>(oldPhiTracksTPCvector[ptNumber][centralityIndex].size())
                    > numberOlderEventsToSave) {
1206                oldPhiTracksTPCvector[ptNumber][centralityIndex].erase(oldPhiTracksTPCvector[ptN
1207                oldEtaTracksTPCvector[ptNumber][centralityIndex].erase(oldEtaTracksTPCvector[ptN
1208            }
1209        }
1210
1211        oldPhiTracksBackwardFMDvector[0][centralityIndex].push_back(backwardTracksPhi);
1212        oldEtaTracksBackwardFMDvector[0][centralityIndex].push_back(backwardTracksEta);
1213        oldMultiplicityTracksBackwardFMDvector[0][centralityIndex].push_back(backwardTracksMult
1214
```

```
1215
1216        if
                (static_cast<int>(oldPhiTracksBackwardFMDvector[0][centralityIndex].size())
                > numberOlderEventsToSave) {
1217            oldPhiTracksBackwardFMDvector[0][centralityIndex].erase(oldPhiTracksBackwardFMDvecto
1218            oldEtaTracksBackwardFMDvector[0][centralityIndex].erase(oldEtaTracksBackwardFMDvecto
1219            oldMultiplicityTracksBackwardFMDvector[0][centralityIndex].erase(oldMultiplicityTrac
1220
1221        }
1222
1223
1224
1225
1226    } //Event-loop end
1227
1228
1229
1230
1231
1232
1233
1234    //Returns the results
1235    std::vector<std::vector<std::vector<TH2D>>> returnVector;
1236    returnVector.push_back(forwardVector);
1237    returnVector.push_back(backwardVector);
1238    returnVector.push_back(backToBackVector);
1239    returnVector.push_back(forwardBackgroundVector);
1240    returnVector.push_back(backwardBackgroundVector);
1241    returnVector.push_back(backToBackBackgroundVector);
1242
1243    //Sets the minimum number of track to 1 to avoid division by 0 errors later
            with needing to have
1244    //a lot of if statements in the .loadProcessed() member function.
1245    for (int ptNumber = 0; ptNumber < numberOfEntriesPt -1 ; ptNumber++) {
1246        for (int centralityNumber = 0; centralityNumber <
                numberOfEntriesCentrality -1 ; centralityNumber++) {
1247            if (eventNumbers[ptNumber][centralityNumber] == 0) {
1248                eventNumbers[ptNumber][centralityNumber] = 1;
1249            }
1250
1251            if (eventNumbersFMD[0][centralityNumber] == 0) {
1252                eventNumbersFMD[0][centralityNumber] = 1;
1253            }
1254
1255        }
1256    }
1257
1258
```

```
1259     auto returnTuple = std::make_tuple(returnVector, eventNumbers,
             eventNumbersFMD);
1260
1261     dataFile.Close();
1262     delete event;
1263     delete tpcTrack;
1264     delete fmdTrack;
1265
1266
1267     return returnTuple;
1268
1269
1270
1271 }
```