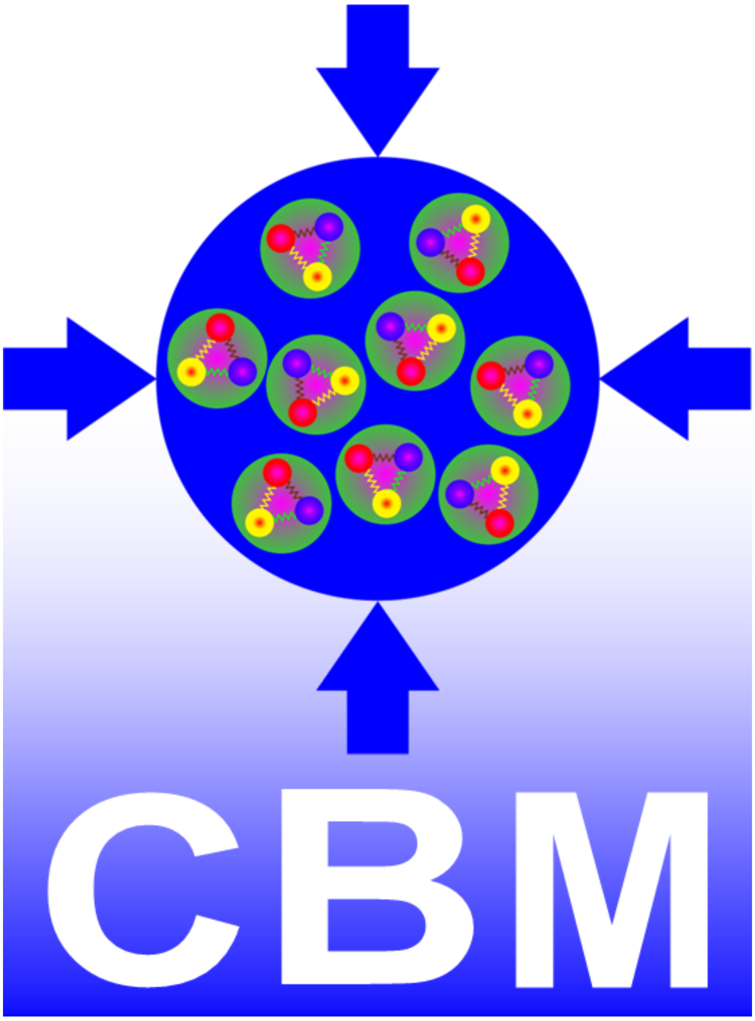
CBM simulation

Alex Olar

July 25, 2017



Introduction

I have spent one month in summer at the GSI facility, Darmstadt to study the CBM simulation which will be part of FAIR [1] . During this month I have become experienced with software such as ROOT [2] , cbmROOT [3], etc.

I have learned much about detector technologies and natural phenomena during my stay and I was happy to be part of this huge project and see how scientist work on a daily basis.

Contents

[1] Facility for Antiproton and Ion Research

[2] CERN software for particle physics

[3] CBM (Compressed Barionic Matter) software for heavy ion physics developped at GSI

I. Fundamentals

I.1. QCD

In the 20th century physicist realized that symmetries play a crucial role in further understanding the universe. Symmetries led us to conservation laws, the discovery of anti-particles, quarks and much more.

When quarks were discovered they brought order to the ever increasing particle zoo, as Niels Bohr referred to it. At first, physicist only knew three of the six existing quarks, such as: u (up), d (down), s (strange). The hadrons can be separated into two distinct groups: *mesons* and *baryons*, containing a quark-anti-quark pair and three quarks, respectively. The property that distinguishes the quarks is called *flavour*.

A unique feature of the strong interaction, which is the fundamental interaction between quarks, is confinement, meaning that quarks do not appear in isolation. The charge of the strong interaction is called colours. Due to confinement it means that an elementary particle has to be colour neutral or usually referred to as ”white”. The fundamental theory describing the strong interaction is called Quantum Chromo Dynamics - QCD.

The elementary particles of the QCD are quarks and anti-quarks which interact by gluons which also carry charge. There are eight types of gluons since they must describe every elementary colour transformation. These gluons interact with themselves as well.

I.2. CBM physics

When dealing with barionic matter the goal is to understand and map the phase diagram and its transitions. To begin with, I should first have to go through basic thermodynamics and its concepts about phases and their transitions.

The phase diagram of water shows the different phases of it according to pressure and temperature. It is well known that there is a triple point where all three forms of water can coexist such as: liquid water, solid ice, and water vapor. There are distinct lines between these phases denote, along these lines two phase can coexist at the same time but when changing conditions the substance must undergo a so called first order phase transition. There is also a critical point which is a state from where to phases differ no more, it is called a smooth *crossover* from one phase to another.

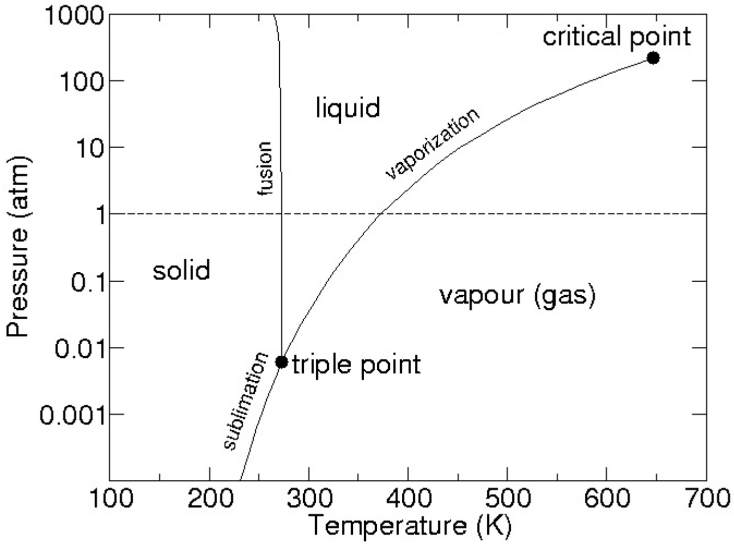
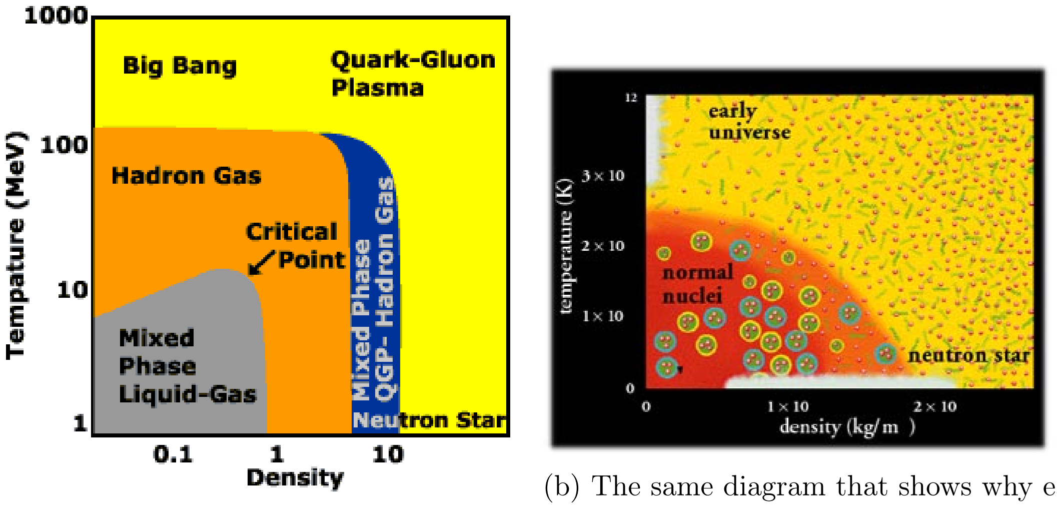


Figure 1: The previously mentioned phase diagram.

Now, that I went through the phase diagram of water (or at least a part of it) I should move on to matter governed by the strong force. The whole phase diagram of strongly interacting matter is not yet experimentally proved, purely theoretical. It depicts very different and vital phases for being able to describe the early universe or the interior of neutron stars. Each point is described by density and temperature.



(a) Temperature in MeV and densities in units of nuclear bulk density, both scales are rhythmical

(b) The same diagram that shows why each phase can be crucial to explore further.

As presented above a phase called quark-gluon plasma was present at the Big Bang and later on ceased to exist when temperature dropped. It can be seen that the interior of neutron stars supposedly contains quark-gluon plasma as well as the density is high enough to enable matter to exist in that phase although the very low temperature.

Obviously the only way to investigate strongly interacting matter on Earth is via high energy particle collisions. It is a characteristic feature of QCD that the coupling between quarks decreases as the collision energy is raised (crossover). This is referred to as asymptotic freedom. Another interesting symmetry of particle physics is chirality. It describes whether the particle's spin points to its direction of motion or to the opposite if the particles is massless. If the velocity and spin are pointing to the same direction the particle is called right-handed, otherwise left-handed. Due to the fact that the up and down quarks have such a small mass QCD has approximate chiral symmetry. However, this is spontaneously broken at low temperatures and densities because of the slight difference in the mass of the lightest quarks. For this reason one chiral direction is favored over the other.

II. CBM detector

II.1. Theory

The analysis of heavy nuclei collisions is extremely complex since the transient nature of the reaction. The goal is to find out more about phase diagram but the collision takes 10^{-22} s and the strongly interacting matter can only be measured by the byproducts of the collision.

Over the past decade the main experimental activity occurred in RHIC [4] and at the LHC [5], On one hand these facilities produce important results concerning the phase diagram. These facilities are mapping the low density, high temperature area of the phase diagram and can analyze the crossover between hadron gas phase and quark-gluon plasma. On the other hand, the FAIR project is going to create much higher baryon densities to be able to explore the first-order phase transition and the critical end point of it.

[4] Relativistic Heavy Ion Collider - Brookhaven

[5] CERN - Large Hadron Collider

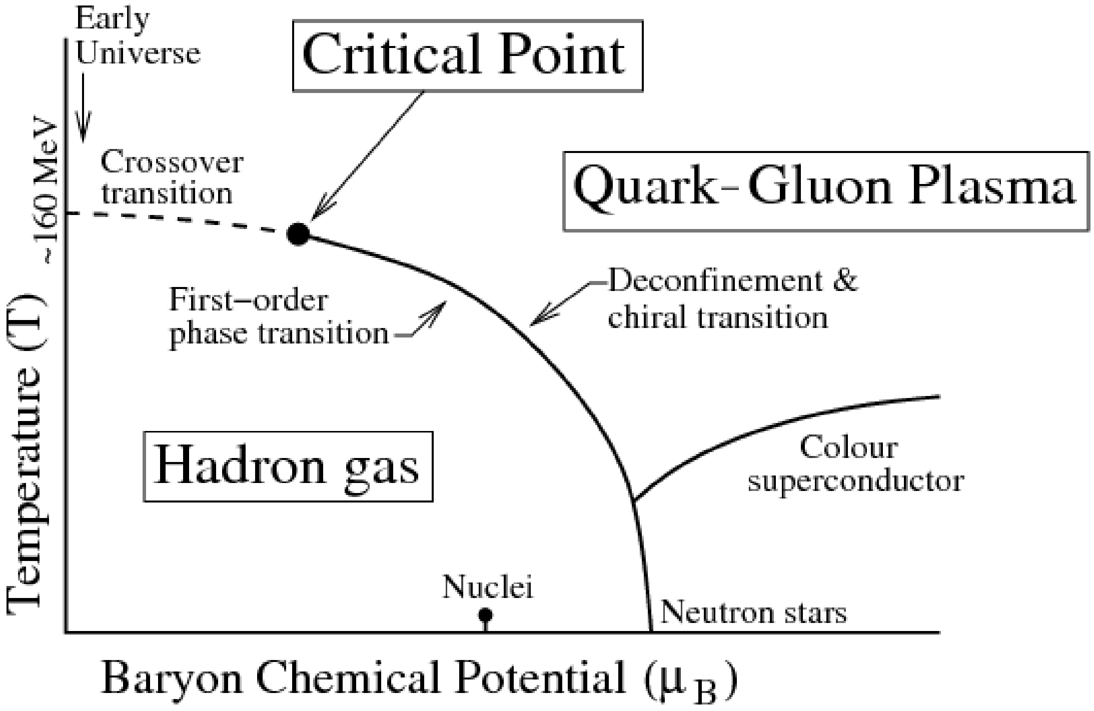


Figure 3: The above discussed hypothetical phase diagram and the transitions between phases.

II.2. Setup and simulation

The setup of the future CBM detector is the following from left to right.

- CBM superconducting magnet with silicon spectrometer
- the micro vertex detector (MVD) is inside of this
- the silicon tracking system (STS) is as well
- ring imaging Cherenkov detector (RICH - light blue)
- followed by four layers of transition radiation (TRD) detectors
- and a time-of-flight (TOF) wall
- after the main detectors there's a muon spectrometer and a projectile spectator detector (PSD)

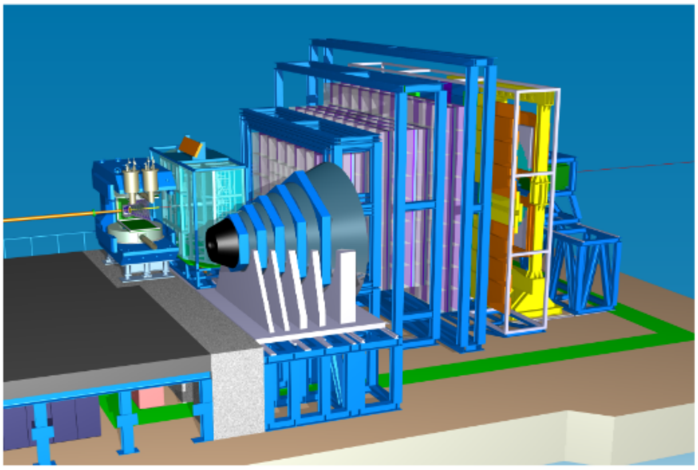


Figure 4: View of the setup

The silicon tracking system is intended to reconstruct the trajectories of the particles it detects. It can only detect charged particles but also able to detect their momenta as well. The time-of-flight wall can achieve high resolution time measurements of about 60 ps.

The CBM project is currently only a plan. The construction of FAIR has started this summer and the first beams are expected by 2022. Until then CBM intends to install a reduced setup at the accelerator available at the GSI to test the systems and prepare for data processing.

Scientist of the FAIR project has developed a robust simulation so far built on ROOT. They call it cbmROOT and is available to the public. They use different well-known heavy ion collision simulation methods such as: UrQMD [6] or PHSD [7]. These models are widely used among scientist in the field.

III. My project

III.1. Φ-meson reconstruction

CBM is going to be a general-purpose heavy ion experiment to study the phase diagram of strongly interacting matter. Resonances are highly useful to study the high density matter created during the collisions. One of these is the Φ-meson which has a small hadronic cross section so it's very unlikely to interact with the

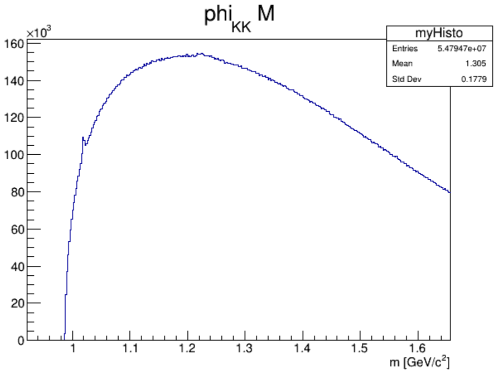
[6] Ultra Relativistic Quantum Molecular Dynamics

[7] Parton Hadron String Dynamics

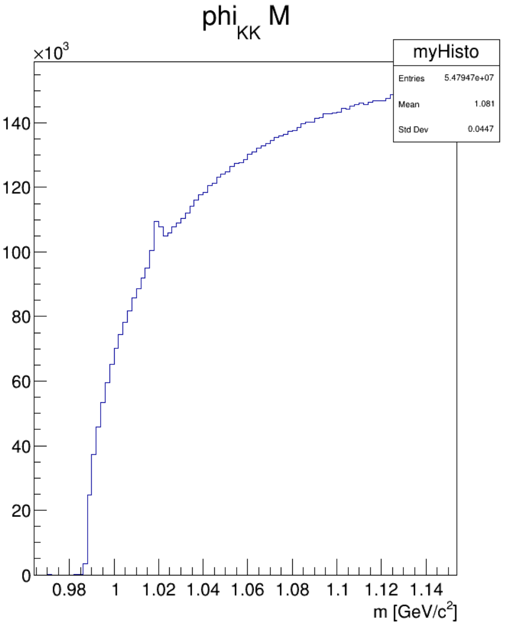
huge amount of hadrons. The Φ-meson is made of a strange and an anti-strange quark and can be a tool to study strangeness production in the partonic phase. The Φ-meson decays into K^+,K^- pairs with an approximate of 50 % chance and to dileptons. The mean lifetime of a Φ-meson is really short, $1.55 \cdot 10^{-22}$ s so it might decay into kaons and dileptons during the collision and only the byproducts would be detected by the detectors. It's mass is 1.019 MeV which can be seen in the invariant mass of the kaons as a peak.

I analyzed data acquired by PHSD simulation. It was an Au+Au central collision at $\sqrt{s} = 10$ GeV. The output of the simulation was passed through cbmROOT reconstruction. There were more than 5$million$ events in the $.root$ file I used for analysis.

On a histogram where the x-axis represents the invariant mass of K^+,K^- pairs and the y-axis the entries, a clear peak can be seen on the combinatorial background at approximately 1.02 GeV. Those were the produced Φ-mesons during the collision.



(a) The combinatorial background and a small but recognizable peak.



(b) Zoomed in on the region of interest.

I tried to approximate the background with a second order polynomial. The acquired parameters for this were $(ax^2 + bx + c)$:

Parameter name	Value []	Error
a	-7.70559e+06	78750.5
b	1.42738e+07	150055
c	-6.49147e+06	71438.9

Here the fit of the background:

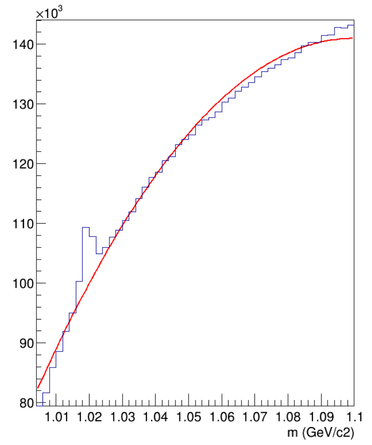
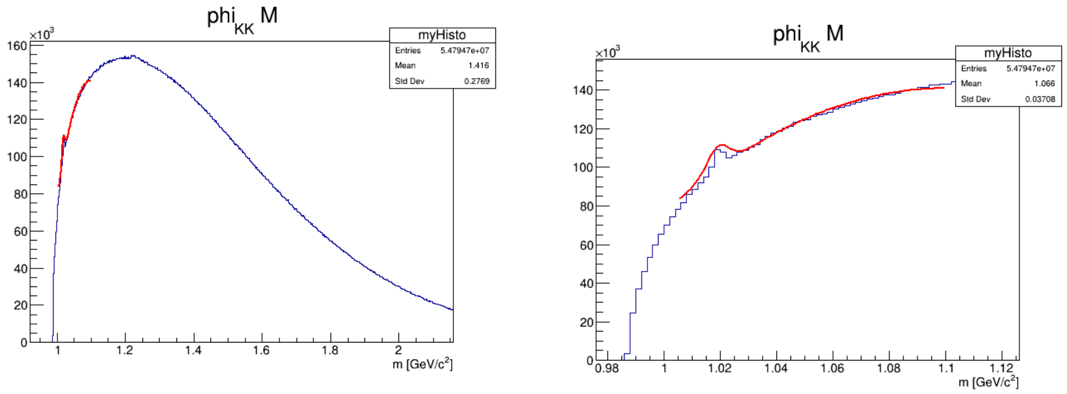


Figure 6: The fit of the combinatorial background.

For the peak I used a different method. I used a low multiplicity signal to approximate the shape of the peak with a Gaussian function and then scaled that shape up to the peak with the background still remaining the same. Here are my results:



(a) The combinatorial background and a small but recognizable peak fitted.

(b) Zoomed in on the region of interest.

I include the ROOT macro I used for the analysis.

```
void fit(){

    TFile* srcFile = TFile::Open("KFParticleFinder_phsdwocsr_auau_10gev_centr_sis100_electron_5M_ToF.root");
    TDirectory* phi = (TDirectory*)srcFile->Get("KFTopoReconstructor/KFParticlesFinder/Particles/phi_{KK}/Parameters");
```

```
  TDirectory* phi_signal = (TDirectory*)srcFile->Get("KFTopoReconstructor/KFParticlesFinder/Particles/phi_{KK}/Parameters/Signal");

  TH1F* M = (TH1F*)phi->Get("M");
  TH1F* Msignal = (TH1F*)phi_signal->Get("M");
  Msignal->SetName("Msignal");

  TFile* myFile = new TFile("phi_fit.root","recreate");
  TH1F* myHisto = (TH1F*)M->Clone();
  myHisto->SetName("myHisto");
  TH1F* myBackground = (TH1F*)M->Clone();
  myBackground->SetName("myBackground");
  TH1F* mySignal = (TH1F*)Msignal->Clone();
  mySignal->SetName("mySignal");
  myFile->cd();

  srcFile->Close();

  TCanvas* canv = new TCanvas("canv","Total fit",640,480);

  TF1* background = new TF1("background","pol2",1.004,1.1);
  TF1* signal = new TF1("signal","gaus",1.011,1.033);

  TF1* total = new TF1("total","gaus(0) + pol2(3)",1.005,1.1);

  background->SetParameters(0.2,1.01,108000.);
  background->SetParNames("landau_1","landau_2","landau_3");
  myBackground->Fit("background","R+");

  Double_t params[6];
  background->GetParameters(&params[3]);

  signal->SetParameters(3200., 1.021, 3000.);
  signal->SetParNames("scale","mean","sigma");
  mySignal->Fit("signal","R+");
  signal->GetParameters(&params[0]);

  params[0]*=250.; // rescale
  total->SetParameters(params);
  total->SetParNames("scale","mean","sigma",
          "la","b","c");
  myHisto->Draw();
  total->Draw("same");

  signal->Write();
  background->Write();
  total->Write();
  myHisto->Write();
  canv->Write();

```

```
    myFile->Close();

}
```

III.2. Φ-meson detection in the CBM

It is going to be a really hard task to efficiently detect Φ-mesons in the future CBM detector system. Not only does the these particles are short live but also a huge combinatorial background is making it hard to detect that small peak in multiplicity. However, it is reasonable to say that CBM will be capable of Φ-meson detection and of understanding strange production in strongly interacting matter more in depth.

The projected efficiency in different momenta is not very high but steady enough. These data was acquired by the simulation.

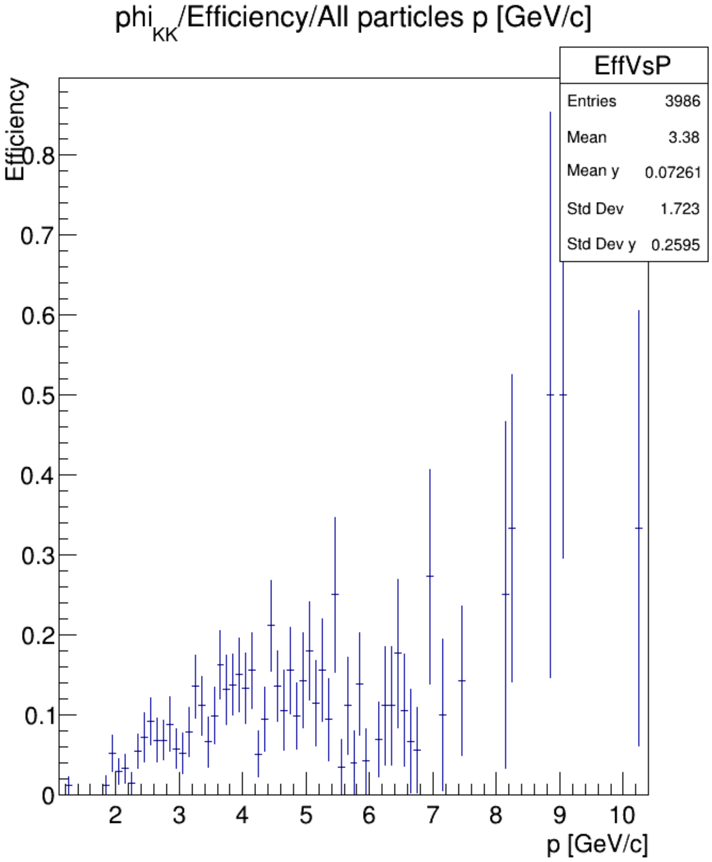


Figure 8: Efficiency in means of momentum

IV. The simulation

IV.1. Installation

There are three main components to the CBM simulation such as: FairROOR, Fairsoft and cbmROOT. They all can be installed on any machine from this link: `https://redmine.cbm.gsi.de/projects/cbmroot/wiki/InstallCbmRootAuto`

It is advisory to do it this way because otherwise one might have serious problems with the software pack. It takes up around 25 GB (probably a tiny bit more with its dependencies which are necessary for it to function properly).

IV.2. Introduction

The collision is simulated via UrQMD and PHSD. These models have been developed for ALICE, RHIC, LHC and last but not least CBM. I mostly used UrQMD data for the time I spent here but I came across PHSD files as well.

At first, one has to run the Monte Carlo simulation to be able to match the 'real' data with generated events. It is designed to filter out hits and find points that can be reconstructed into trajectories.

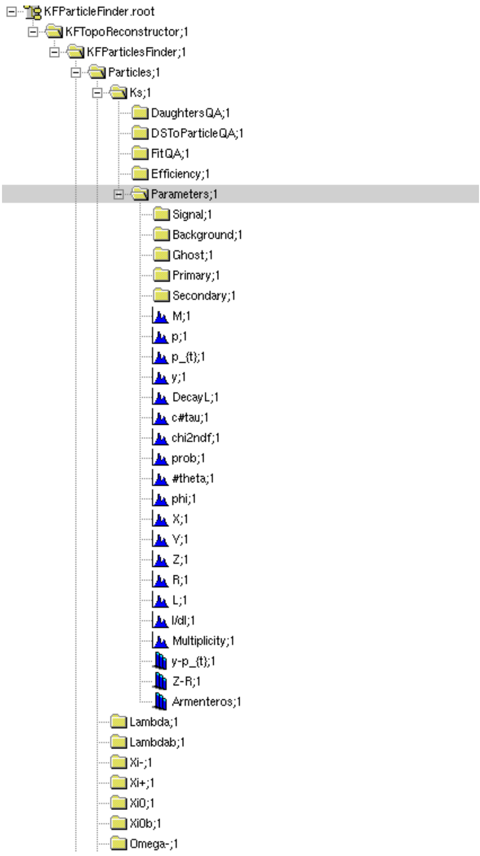
They use Geant3 and Geant4 to simulate the passage of elementary particles through matter. It is part of the Monte Carlo simulation that needs to be ran first.

The output of this macro is the simulation file which contains true hits detected by the STS and MVD detectors as well as by the others. Using these the reconstruction of trajectories is due. Stating with the MVD the program tries to find clusters within the detector then moves on to the STS to do the same thing. These methods spread to other detectors as well which I won't mention here because I mainly concentrated on these two. Moving on to the track finding in MVD and STS the trajectories are being reconstructed and the primary vertex is distinguished during the process.

Obviously the hits and paths are being matched several times during the run to make sure that the inquired data is correct in every way and there are no mismatched results.

After the successful finish of reconstruction a reconstructed file is acquired from the raw data. Now there is only one remaining task to be done which is matching the trajectories to particles and identifying most of them. This is done by KF-ParticleFinder which is substantial package of complex code to identify particles.

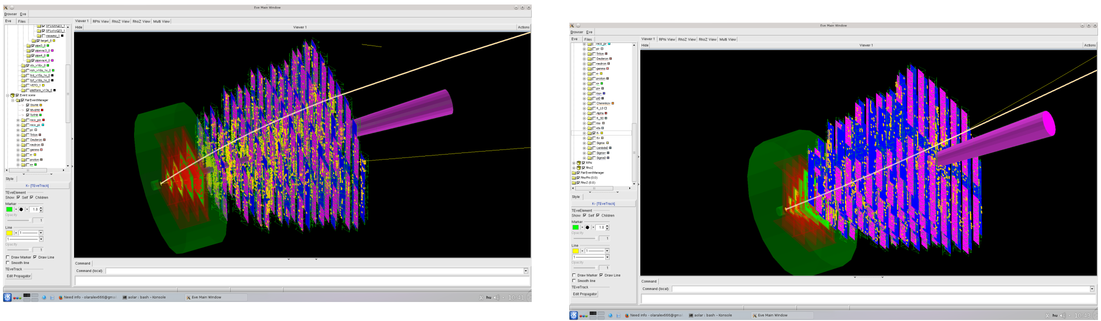
The output of this program is a .root file which contains numerous particles and accompanying histograms of efficiencies of reconstruction, background, armenteros diagrams, signals, etc. The structure looks like this:



```
KFParticleFinder.root
  KFTopoReconstructor;1
    KFParticlesFinder;1
      Particles;1
        Ks;1
          DaughtersQA;1
          DSToParticleQA;1
          FitQA;1
          Efficiency;1
          Parameters;1
            Signal;1
            Background;1
            Ghost;1
            Primary;1
            Secondary;1
            M;1
            p;1
            p_{t};1
            y;1
            DecayL;1
            c#tau;1
            chi2ndf;1
            prob;1
            #theta;1
            phi;1
            X;1
            Y;1
            Z;1
            R;1
            L;1
            l/dl;1
            Multiplicity;1
            y-p_{t};1
            Z-R;1
            Armenteros;1
        Lambda;1
        Lambdab;1
        Xi-;1
        Xi+;1
        Xi0;1
        Xi0b;1
        Omega-;1
```

Figure 9: The top portion of the data in the file.

The CBM project has a powerful visualization tool as well to visualize the detector geometry and the reconstructed tracks inside. The following pictures demonstrates this:



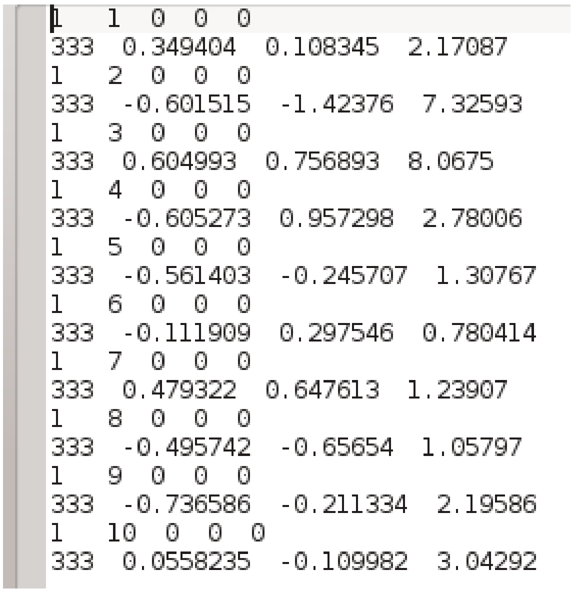
(a) Reconstructed track in the MVD and STS detectors.

(b) From a different angle.

IV.3. How-tos

As I mentioned before, at first one should use the Monte Carlo simulation with some input file. This can be a .root or a simple ASCII file which contains particle IDs and momentum distribution. The output of a PHSD or UrQMD simulation is mainly a .root file but other generators like HIJING use plain text for example. In the CBM simulation software there are tools to process both ASCII and .root input files.

I learned to use the signal generator with which one can create any type of input file for the simulation. I mainly used it to create Φ-s and see how they decay into various particles. The input file looks like the following:



```
1   1  0  0  0
333  0.349404  0.108345  2.17087
1  2  0  0  0
333  -0.601515  -1.42376  7.32593
1  3  0  0  0
333  0.604993  0.756893  8.0675
1  4  0  0  0
333  -0.605273  0.957298  2.78006
1  5  0  0  0
333  -0.561403  -0.245707  1.30767
1  6  0  0  0
333  -0.111909  0.297546  0.780414
1  7  0  0  0
333  0.479322  0.647613  1.23907
1  8  0  0  0
333  -0.495742  -0.65654  1.05797
1  9  0  0  0
333  -0.736586  -0.211334  2.19586
1  10  0  0  0
333  0.0558235  -0.109982  3.04292
```

333 is the particle ID for the Φ-meson. The simulation then knows how to process this particle and is able to decay it with the suitable probabilities. To be able to

process plain text one has to use the

```
FairAsciiGenerator *SignalGen = new FairAsciiGenerator(inFile);
primGen->AddGenerator(SignalGen);
```

above generator instead the one for .root input called *CbmUnigenGenerator*. There's another important step if someone would like to visualize the output of a simulation-reconstruction-identification cycle. In the simulation file one has to enable trajectory saving. It is not efficient if there are too many events so it is advised to not use it just for low event counts.

```
 // -Trajectories Visualization (TGeoManager Only )
 run->SetStoreTraj(kTRUE);  //->
 // ------------------------------------------------------------
```

So after finishing the reconstruction and particle identification on any input signal it is possible to display the events one-by-one with the *eventDisplay.C* macro. This macro will output the whole CBM geometry and the found tracks as well. Here is a picture of a Φ-decay into kaons.

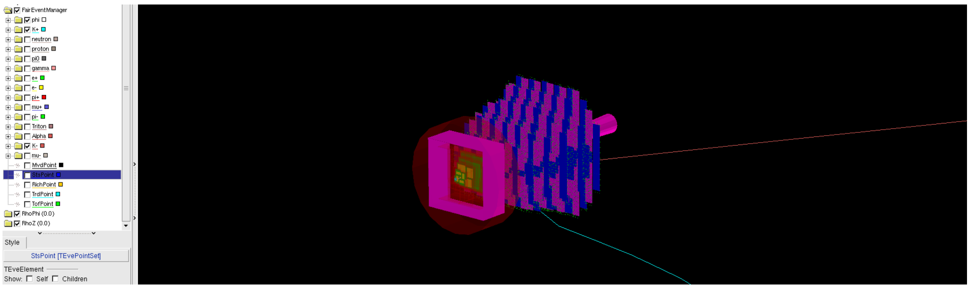


Figure 11: Event visualization in the MVD and STS detectors.

IV.4. Generating Φ-mesons and analyzing the output

I ran the signal generator to generate 10000 events of phi right in the center of the target. It is useful to analyze such data because with the results one can separate the facts from the fancy when experimenting later. With known Φ multiplicity and efficiencies it is possible to test theoretical forecast according the outcome.

In the generator macro it is possible to modify the beam energy (in GeV), the temperature (in GeV) and of course the particles which we want to generate.

```
double fSlope = .154; // temperature
...
double eBeam = 10.; // beam energy
double pBeam = TMath::Sqrt(eBeam*eBeam - kProtonMass*kProtonMass);
...
 const int NParticlesPerEvent = 1;
 const double kSignalMass[NParticlesPerEvent] = {1.019455};     //  mass
    in GeV
 const int    kSignalID[NParticlesPerEvent] =  {333};
 ...
   for (int i=0; i<NEvent; i++){
   // Generate rapidity, pt and azimuth
   outputfile<<NParticlesPerEvent<<"  "<<i + 1<<"  "<<0.<<"  "<<0.<<"  "<<0.<<endl;
   for(int j=0;j<NParticlesPerEvent;++j) {
   double yD   = gRandom->Gaus(fYcm, fRapSigma);
   double ptD  = fThermal[j].GetRandom();
   double phiD = gRandom->Uniform(0., kTwoPi);

   // Calculate momentum, energy, beta and gamma
   double pxD    = ptD * TMath::Cos(phiD);
   double pyD    = ptD * TMath::Sin(phiD);
   double mtD    = TMath::Sqrt(kSignalMass[j]*kSignalMass[j] + ptD*ptD);
   double pzD    = mtD * TMath::SinH(yD);

   outputfile<<kSignalID[j]<<"  "<<pxD<<"  "<<pyD<<"  "<<pzD<<endl;

   }
   }
```

It can be easily seen that modifying this file is fairly easy and one can generate any desirable input. After generating the ASCII signal file it should be provided to the simulation macro with the event count.

```
void run_mc_phi(TString inFile="Signal_phi_10000.txt", const char* setupName = "sis100_electron", Int_t nEvents = 10000)
{
  TString outFile = "sim_phi_10000.root";
  TString parFile = "param_phi_10000.root";
...
  // --- Define the target geometry -------------------------------------------
  //
  // The target is not part of the setup, since one and the same setup can
  // and will be used with different targets.
  // The target is constructed as a tube in z direction with the specified
  // diameter (in x and y) and thickness (in z). It will be placed at the
  // specified position as daughter volume of the volume present there. It is
  // in the responsibility of the user that no overlaps or extrusions are
  // created by the placement of the target.
  //
  TString  targetElement   = "Gold";
  Double_t targetThickness = 0.025;  // full thickness in cm
  Double_t targetDiameter  = 2.5;    // diameter in cm
  Double_t targetPosX        = 0.;      // target x position in global c.s. [cm]
  Double_t targetPosY        = 0.;      // target y position in global c.s. [cm]
  Double_t targetPosZ        = 0.;      // target z position in global c.s. [cm]
  Double_t targetRotY        = 0.;      // target rotation angle around the y axis [deg]
}
```

The output is a .root simulation file with data of the hits in the detectors. It is worth mentioning that the target is self-defined, so it is completely up to the user to place it correctly and define it suitably.

After these the reconstruction file must be modified slightly with the event number and the appropriate input files. This file as I mentioned above reconstructs the trajectories of particles by trying to fit circles on the hit points provided be the simulation. Lastly it is due to run the physics macro to identify the tracks with real life particles and create the *KFParticleFinder.root* file which contains the appropriate histograms. From this I have plots to show: