FHCal Tutorial

v.1 (December, 2022) by S. Musin

1. Geometry

Current version of FHCal geometry is v1 no overlap pipe magnet: `zdc_oldnames_7sect_v1_no_overlaps_w_pipe_magnet.root`

Please make sure that this version of FHCal geometry is called in `mpdroot/macro/mpd/geometry_stage1.C`

The following lines should be verified:

```
FairDetector *Zdc = new MpdZdc("ZDC", kTRUE);

Zdc->SetGeometryFileName(
        "zdc_oldnames_7sect_v1_no_overlaps_w_pipe_magnet.root");

fRun->AddModule(Zdc);
```

2. Running FHCal simulations.

There is a FHCal ZDC digitizer available in the MpdRoot: `mpdroot/detectors/zdc/MpdZdcDigi.*`

When to comparing to real data it is advisable to use adjusting `MpdZdcDigiProducer` class parameters to better suit the experiment.

In order to run the code the following lines need to be added (if not present) in `reco.C`:

```
FairTask *tdigi= new MpdZdcDigiProducer("MpdZdcDigiProducer");
fRun->AddTask(tdigi);
```

3. Data Format

Produced DST files have structure of `MpdZdcDigi`:

```
#include "MpdZdcDigi.h"
```

```
TTree *inTree;
inTree = (TTree*) inFile->Get("mpdsim");

MpdZdcDigi* mpdzdcdigi;
inTree->SetBranchAddress("mpdzdcdigi", &mpdzdcdigi);

for (long int event=0; event < inTree->GetEntries(); event++){
        inTree->GetEntry(event);
        // do what you need with mpdzdcdigi variables here
}
```

4. Digitizer Variables

For each collision event FHCal can detect energy deposition in a number of fired sections/modules. Energy deposition and coordinates of modules are written into a TTree:

```
/// returns ID of FHCal detector arm (1 - left, 2 - right).
mpdzdcdigi->GetDetectorID();

/// returns ID of current module (1-45 for each arm).
mpdzdcdigi->GetModuleID();

/// Hardware channel number (Section number) (int in range [0,6]).
mpdzdcdigi->GetChannelID();

/// Sum of the energy losses as analog signal accumulated prior to
digitalization
/// GeV
mpdzdcdigi->GetELoss();

/// Sum of the energy losses as analog simulated response of the
detector/channel
/// GeV
mpdzdcdigi->GetELossReco();

/// Sum of the energy losses as digital response of the detector/channel
mpdzdcdigi->GetELossDigi();

// X/Y coordinate of FHCal module center ([-45,45] cm, module size is 15
cm).
mpdzdcdigi->GetModuleX();
mpdzdcdigi->GetModuleY();
```

Constructor of `MpdZdcDigiProducer` defines the following values:

```
/// in average 15 pixels of utilized SiPMs are fired for each detected MIP
fPix2Mip = 15;
```

```
/// energy depoistion of MIP in FHCal cells is measured to be 5MeV
fMIPEnergy = 0.005;

// 0.3 of MIP noise level — one should adjust this value to meet
experimental data
fMIPNoise = 0.3;
```

5. Simple analysis of DST file

The code for simple analysis of DST file can be found in:
`/eos/nica/bmn/users/musin/FHCal_tutorial/FHCal.cpp`

To run the program:

```
$ root -l
root [0] .L FHCal.cpp
root [1] FHCal("dst_file.root", "output_file.root")
```

The output file contains six 2D histograms.

Histograms named `pXY_1` and `pXY_2` are the plots showing the module loading per event (number of events with energy deposition in module / to total number of events).
`pXY` shows the same for both arms of FHCal calorimeter.

Histograms `pXY_weighted_Edep_1` and `pXY_weighted_Edep_2` show the energy deposition in FHCal modules in the left and right arm normalized to number of events.
`pXY_weighted_Edep` shows the same for both arms of FHCal calorimeter.

6. Centrality_NICA

6.1 Introduction

This files implemented to determine centrality at MPD/NICA experiment with FHCal.

Source code can be found here.

For a general introduction to the topic, see this and this.

Also useful links: HADES, ALICE.

The preliminary version of the code, requires more automatization, however, is currently working.

6.2 Files description

`twod_gauss_energy_uni_distr.cpp`. Creating different histograms with dependencies (e.g. impact vs. angle, edep vs emax etc.).

`FitIt.cpp` is for fitting histograms in order to divide it into sectors (10%, 5% etc.)

`RunFits.sh` is needed if the defaults do not result in a fitting with the 1st time.

`Ellipse.cpp`. FitIt uses this file to fit with an ellipse.

`Impact.cpp` is a macro for splitting the histograms into centrality classes.

6.3 How to

1. `twod_gauss_energy_uni_distr.cpp`

Set path to your data file here:

```
TFile *_file0 = TFile::Open("/path/datafile.root");
```

Uncomment this line if you want to try pion subtraction:

```
 //hFin->Add(hPionsFit, -1.);
```

You can draw a lot of histograms, some of them will be already available after the first run, you can see the list in the "draw" part of the code. For futher steps we will use Edep_Emax histo as an example.

```
if (radius_con > 0) //here you can set any cut you need
{
        hEdepEmax->Fill(f8->GetParameter(2) / 1000, edep_7sect_1 +
edep_7sect_2);
        //f8->GetParameter(2) is a cone height
    myfile << f8->GetParameter(2) / 1000 << " "
    << edep_7sect_1 + edep_7sect_2 << " " << impPar << endl;
    //write data to the .txt
}
```

You'll need the .txt file for the next step, its name can be changed here:

```
myfile.open("file_name.txt");
```

To run:

```
root [0] .L twod_gauss_energy_uni_distr.cpp
root [1] twod_gauss_energy()
```

2. `Ellipse.cpp`

Now you have a histogram, you need to roughly determine the center of the ellipse and the size of the axes. You need to paste these estimates into the FitIt.cpp file to make the fit more accurate. Put this values (x0, y0, a, b) into the lines 131-134.

Specify the path to the folder (or create it) where the pictures with the fit results will be saved:

```
gSystem->cd("/mnt/d/Work/INR/centrality/pics");
gSystem->Exec("mkdir fits_ell");
gSystem->cd("/mnt/d/Work/INR/centrality/pics/fits_ell");
```

3. `RunFits.sh`

Here just set number of `FitIt.cpp` iterations. The difference between iterations lies in the fact that probably at a single iteration fit may not work, so each run 1 bin in the x and y axis (from the left bottom) will be cutted. You can monitor the quality of the fit "online" by watching the output of Minuit in the terminal, or just look at the pictures and select the fit after finishing.

To run:

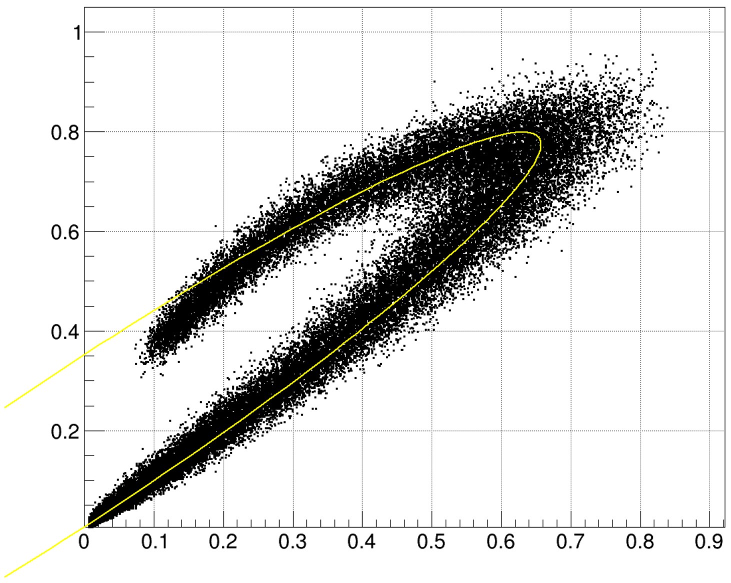
```
./RunFits.sh
```

4. `Impact.cpp`

Final stage. Not very user-friendly and requires manual actions. Once you have finished fitting and selected the right fit, you have the parameters of a curve that envelope the histogram data. The output looks like this:

```
EXT PARAMETER                                           STEP         FIRST
NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
1  x0           5.00000e-04   2.51883e-04   3.74524e-05** at limit **
2  y0           2.37348e-01   3.36057e-03  -4.11668e-06  -6.81629e-02
3  a            7.54648e-01   6.39113e-03  -1.85627e-06   1.10214e-02
4  b            1.75969e-01   2.27828e-03  -4.68362e-06   5.60232e-02
5  theta        3.80269e+01   2.85929e-01   5.07917e-06  -1.90082e-02
top x 0.594961
top y 0.702224
theta deg 0.663675
```

Example of the fit picture:



This way you have all the data to go on dividing the histogram into sectors (in this code 5% sectors). However, there are still a couple of steps left, it is necessary to make changes (enter these parameters) in the program code (at this stage it is organized manually, in the future everything will be automated). In case your fit is obtained at zero iteration of `FitIt.cpp` you should do the following:

Set paths and name of your .txt with data from step 1:

```
TFile *f_input = new
TFile("/mnt/d/Work/root/builddir/macros/EdepEmax_QGSM_full_1_to_1.root");
TH2F *hist = (TH2F *)f_input->Get("EdepEmax");
ifstream fp2("QGSM_11_Edep_Emax.txt");
```

Set parameters of the curve in lines 64 - 68. Example:

```
const double y0 = 0.17992;
const double x0 = 0.0005;
const double a = 0.891876;
const double b = 0.127786;
const double th = 0.754254;
```

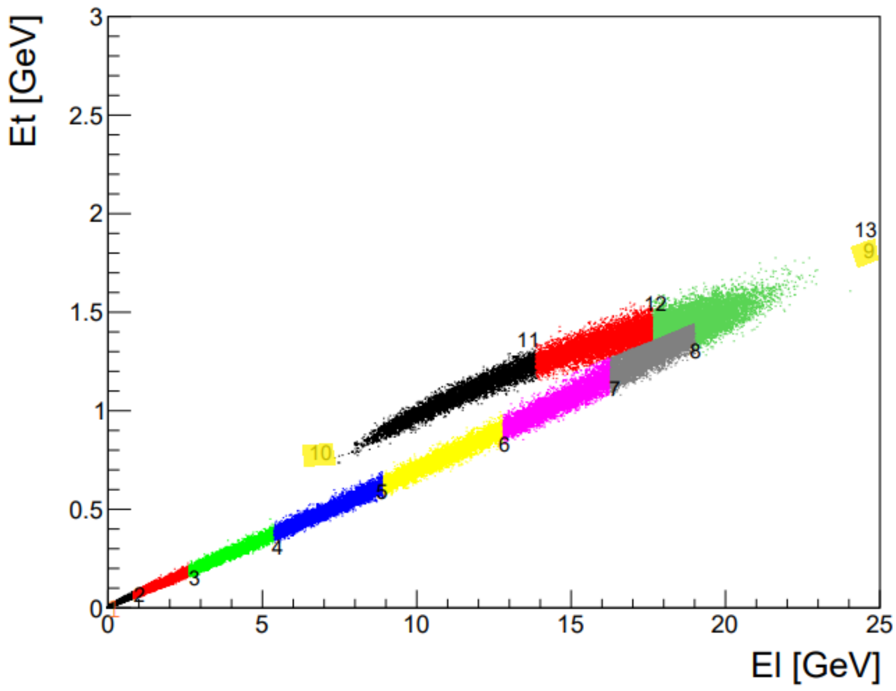
Then you have to determine k and b parameters from the y=kx+b equation. You should solve the system of 2 equations putting x0, y0 and top_x, top_y. Then replace all the k and b in program to your own (ctrl+H).

Set the range in for cycles:

```
for (int i = -200000; i < 200000; i++) // along x
```

and further.

The last thing to do is to determine at what point the transition from the edge of the lower branch to the edge of the upper branch occurs (see the picture illustrating this transition).



CUTS

№	x1	y1	x2	y2
1.	0.00625	0.222403	0.00625	0
2.	0.86875	0.278082	0.86875	0
3.	2.66875	0.394281	2.66875	0
4.	5.43125	0.572614	5.43125	0
5.	8.95625	0.800171	8.95625	0
6.	12.8313	1.05032	12.8313	0
7.	16.3063	1.27465	16.3063	0
8.	19.0688	1.45298	19.0688	0
9.	25	1.83587	25	0
10.	7.38125	10	7.38125	0.698497
11.	13.8938	10	13.8938	1.11891
12.	17.6563	10	17.6563	1.3618
13.	25	3	25	1.83587

To determinate it run the program for 1 time, you will have such an output in the terminal (example):

```
CUTS 0 | 2.61855e-322 6.95331e-310 6.92198e-310 6.92199e-310
CUTS 1 | -2.35397 2.80689 0.00131725 0
CUTS 2 | -1.32037 3.69122 1.84388 0
CUTS 3 | 0.628014 5.35822 5.41378 0
CUTS 4 | 3.59403 7.89588 11.1476 0
CUTS 5 | 7.28269 11.0518 19.1294 0
CUTS 6 | 11.1003 14.3181 30.7557 0
CUTS 7 | 13.9407 16.7483 139.87 0
CUTS 8 | 30 30.4883 30 0
CUTS 9 | 0 30 0 4
CUTS 10 | -17.5241 30 1.21605 5.86133
CUTS 11 | -3.88682 30 7.77095 11.4696
CUTS 12 | 11.9038 30 12.6132 15.6125
CUTS 13 | 30 30 30 30.4883
```

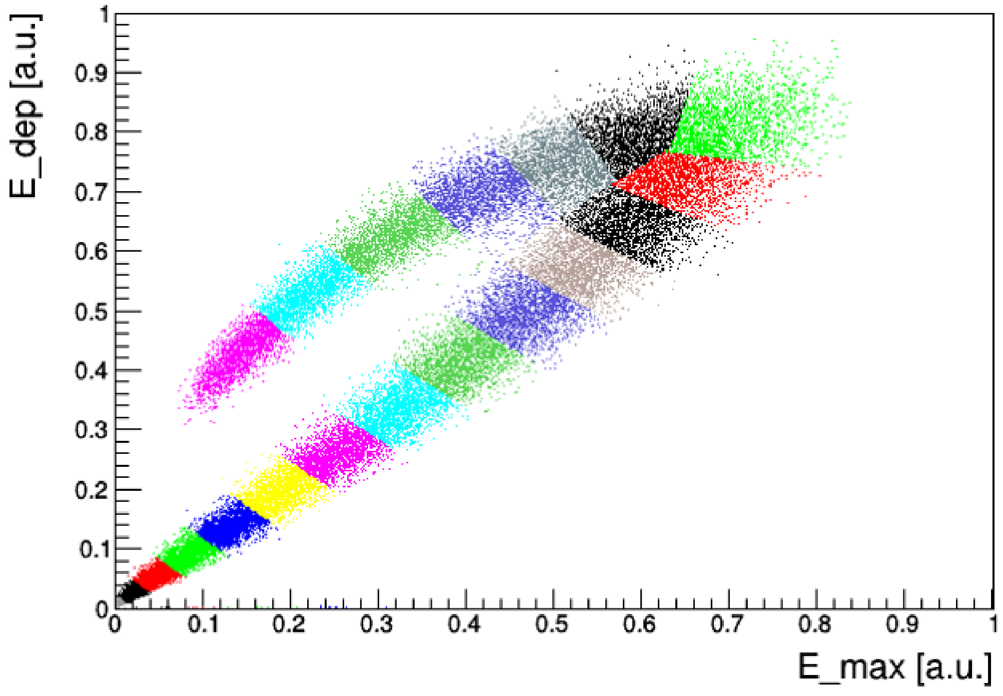
Hence, you should set the following parameters in the code:

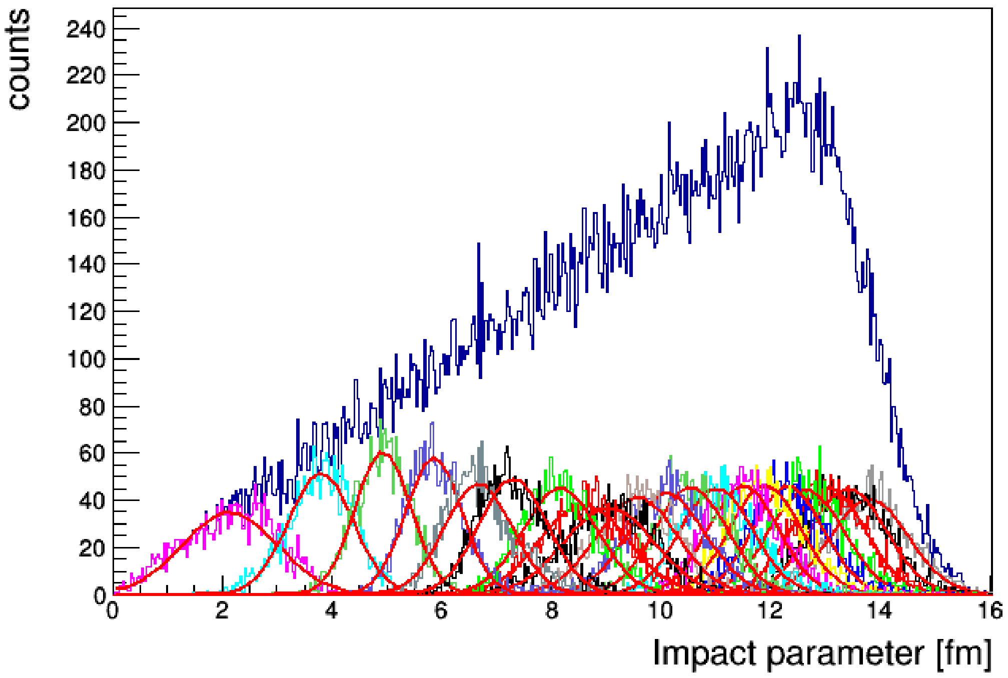
```
for (int iii = 0; iii < 13; iii++)
{
    if (iii == 8) continue;
    <...>

    if (graph_cut->IsInside(emax, edep_read))
    {
                if (iii == 12)
                {
                        ImpPar[7]->Fill(impPar);
                        pElEt[7]->Fill(emax, edep_read);
                }

        <...>

        for (int i_impact = 0; i_impact < 13; i_impact++)
        {
            if (i_impact != 8)
        <...>

        for (int i_imp = 0; i_imp < 23; i_imp++)
        {
            if (i_imp == 8) continue;

                <...>

                for (int i_pelet = 0; i_pelet < 23; i_pelet++)
                {
                if (i_pelet != 8)

                <...>
```

To run:

```
root [0] .L Impact.cpp
root [1] ImpactIt()
```

In the end you should have such pictures:


E_dep [a.u.]
E_max [a.u.]


counts
Impact parameter [fm]