Quick tutorial on MC-Glauber based centrality determination procedure in MPD (NICA)

Petr Parfenov, Dim Idrisov, Vinh Ba Luong and Arkadiy Taranenko

March 2021

1 Glauber Monte Carlo

1.1 Installation

The latest version of Glauber Monte Carlo (MC-Glauber) package is available here [1]:
https://tglaubermc.hepforge.org/downloads/
To download it on the cluster one can use wget command:

```
wget https://tglaubermc.hepforge.org/downloads/?f=TGlauberMC-3.2.tar.gz
```

To unpack and clean up the downloaded archive one can use the following command:

```
tar -xf *.tar.gz
rm *.tar.gz
```

The main ROOT script is `runlauber_v3.2.C`.

1.2 Usage

Firstly, one need to set up ROOT environment:

```
source /opt/fairsoft/mpd/new/bin/thisroot.sh
```

Use following commands to run Glauber Monte Carlo:

```
root -l
.L runlauber_v3.2.C
runAndSaveNtuple(Nev, sysA, sysB, signn)
.q
```

Here `Nev` denotes number of events, `sysA` and `sysB` – colliding nuclei, and `signn` – inelastic nucleon-nucleon cross section which defines energy of the colliding nuclei.

In general, it is reasonable to generate (5-20)$\cdot 10^6$ events to have enough statistics for the centrality determination procedure. The ratio between MC-Glauber and data statistics is 10/1, so for the centrality determination based on $5\cdot 10^5$ events from the data one must have at least $5\cdot 10^6$ MC-Glauber events.

The first colliding systems on the MPD experiment is planned to be Bi+Bi and Au+Au. Those nuclei must be manually set up in the `runlauber_v3.2.C` (under the line number 1172):

```
else if (TString(name) == "Au3")
    {fN = 197; fR = 6.5541; fA = 0.523; fW = 0; fF = 1; fZ=79;}
else if (TString(name) == "Bi")
    {fN = 209; fR = 6.75; fA = 0.468; fW = 0; fF = 1; fZ=83;}
```

Here `fN` denotes mass number, `fR` – radius of the nucleus, `fA` – skin thickness of the nucleus, `fW` – deformation parameter of the nucleus, `fF` – type of the nuclear density function (F = 1 means 3-parameter Fermi function) and `fZ` – is the atomic number. The data was taken from refs. [2], [3].

The last parameter `signn` is set accordingly to the system energy ($\sqrt{s_{NN}}$) [4]:

$\sqrt{s_{NN}}$, GeV	4.5	7.7	9.5	11
σ_{NN}^{inel}, mb	29.3	29.7	30.8	31.2

After adding those modification in the code, one can generate MC-Glauber data. For example, to generate $5{\cdot}10^6$ Au+Au events at $\sqrt{s_{NN}} = 11$ GeV, one can use the following command:

```
root -l
.L runlauber_v3.2.C+
runAndSaveNtuple(5000000, "Au3", "Au3", 31.2)
.q
```

MC-Glauber will then write `gmc-Au3Au3-snn31.2--md0.4-nd-1.0-rc1-smax99.0.root` file with `TNtuple nt_Au3_Au3` in it. This file will be used in the centrality procedure.

2 Centrality Framework

2.1 Installation

To download Centrality framework from the git one can use git clone command and install the project:

```
git clone https://github.com/FlowNICA/CentralityFramework.git
cd CentralityFramework/Framework/centrality-master/
mkdir build/
cd build/
cmake ..
make
```

2.2 Usage

2.2.1 Preparing the data

To begin centrality determination procedure one needs multiplicity distribution from the data. Template reader for MpdDst data format `CentralityFramework/Readers/MpdDstReader.C` can be modified and used to fill histogram with multiplicity distribution from MpdDst file format. Generally, the framework was tested on multiplicity distributions with the following track selection criteria:

- $p_T > 0.15$ GeV/c
- $|\eta| < 0.5$
- Only charged particles (for model data only)
- $N_{hits} > 16$ (for reconstructed data only)
- DCA < 0.5 cm (for reconstructed data only).

2.3 Submitting centrality determination jobs

2.3.1 Configuring `config.txt.template`

Once the input files from MC-Glauber and data are ready, one can start the procedure. Template of the main script is stored in `CentralityFramework/scripts/template/` directory. First, change `config.txt.template`: put full path to the MC-Glauber file in the first line, name of the `TNtuple` in the second line (`nt_Au3_Au3` or `nt_Bi_Bi`) and full path to the file with multiplicity distribution from the previous step in the third line. If the name of the multiplicity histogram was modified it should be set in the line 4 of the `config.txt.template`.

There are 4 parametrizations for fit function available. They set number of ancestors N_a as a function of N_{part} and N_{coll}:

- Default : $N_a = fN_{part} + (1-f)N_{coll}$
- PSD : $N_a = f - N_{part}$
- Npart : $N_a = (N_{part})^f$
- Ncoll : $N_a = (N_{coll})^f$
- STAR : $N_a = \frac{(1-f)}{2}N_{part} + fN_{coll}$.

One can change parametrization in line 14 of `config.txt.template`.

2.3.2 Configuring `parameter.list`

Next step is to configure `parameter.list` file. It has 200 lines containing parameters configuration for each job in the following pattern:

```
f_min:f_max:k_min:k_max:Mult_min:Mult_max
```

Those parameters then parsed and placed into copy of `config..txt.template` for each job. Parameters f, k are set within ranges $0 < f < 1$, $0 < k < 100$ accordingly and do not require any changes. So the only change that is required are `Mult_min:Mult_max` which define multiplicity range within which the fit procedure is applied. To change those values one can use sed command:

```
sed -i "s/20:360/Mult_min:Mult_max/" parameter.list
```

Here `Mult_min` and `Mult_max` are chosen based on the multiplicity distribution: `Mult_min` generally is set to 10 or 20 and `Mult_max` is set close to the values of maximum of the multiplicity distribution.

2.3.3 Configuring and running `start.sh`

First, one should change the paths to the temporary directory of their choosing: lines 4, 12, 13 of the `start.sh`. Then, in line 25, full path to the framework core:
`CentralityFramework/Framework/centrality-master/`
In line 32, one can set short and usable name for this centrality determination run. Generally, it contains collision system and beam energy under investigation with additional optional short information. After all necessary modifications are done, one can start the procedure:

```
qsub start.sh
```

In most cases, it takes up to 2-3 hours for all jobs to finish. Results will be stored in the output directory `OUT` alongside `start.sh` script.

2.4 Final steps of the centrality determination analysis

After the fitting from the previous step is complete one can finish the analysis.

2.4.1 Finding the best fit

First of all, one has to merge all fit results:

```
cd CentralityFramework/scripts/template/OUT/COMMIT/jobid/file/root/
hadd -k -f -j 20 fit_merged.root fit/*.root
```

Then use `CentralityFramework/Framework/Chi2.C` macro:

```
root -l -b -q Chi2.C'("........./fit_merged.root")'
```

The macro may process for several minutes and result with a line:

```
f = 0.34+/-0.122 mu = 0.221908+/-0.197214
k = 41+/-8.243 chi2 = 0.991703+/-0.0813411
```

One has to find the corresponding file in `glauber_qa/` directory. To do so, find the line in `parameter.list` which contains found f and k parameters - let's say line number N. The resulting best fit then is `glauber_qa/glauber_qa_jobid_N.root`. `Chi2.C` macro also will result with `Fit_Errors_RPC.root` file which contains quality of the fit information.

2.4.2 Dividing multiplicity into centrality classes using multiplicity cuts

This procedure is available once the file with optimal fit is found (see previous step). In file `HistoCut.C` one should modify the lines 2 and 3 with full path to the file `glauber_qa/glauber_qa_jobid_N.root` from the previous step. If the name of the multiplicity histogram was changed, it needs to be modified as well in line 5. Once the correct path to the file is set, simply run the macro:

```
root -l -b -q HistoCut.C'(10)'
```

Number of the centrality classes is an argument of the macro. The argument is set `10` which produces 10 centrality classes within 0-100% range (0-10%, 10-20%, ..., 90-100%). Resulting file `HistoCutResult.root` contains multiplicity distributions for each centrality cuts for both fitted multiplicity function (`CentralityClass_Fit`) and multiplicity distribution (`CentralityClass`) as well as the `TTree Borders`. This `TTree` contains the following information about each centrality class:

- `Ncc` – number of entry
- `MinPercent` – minimum value of centrality in the given centrality class
- `MaxPercent` – maximum value of centrality in the given centrality class
- `MinBorder` – lower cut on multiplicity for the given centrality class
- `MaxBorder` – upper cut on multiplicity for the given centrality class.

2.4.3 Mapping parameters from MC-Glauber with centrality classes

Similarly to the previous step, in file `CentralityClasses.C` one has to modify lines 2 and 3 with full path to the `HistoCutResult.root` and `glauber_qa/glauber_qa_jobid_N.root` files correspondingly. After that, run the macro:

```
root -l -b -q CentralityClasses.C'(10)'
```

The argument here is the same as for `HistoCut.C` macro: total number of centrality classes within 0-100% centrality range. Resulting file `FINAL.root` contains impact parameter (`B_VS_CentralityClass`), number of participants (`Npart_VS_CentralityClass`), number of binary nucleon-nucleon collisions (`Ncoll_VS_CentralityClass`) distributions for each centrality class and centrality dependence of their mean values (`B_average_VS_Centrality`, `Npart_average_VS_Centrality`, `Ncoll_average_VS_Centrality`). Additionally, `FINAL.root` contains `TTree Result` which is a direct copy of `TTree Borders` from the previous step.

2.5 Using centrality classes provided from the framework in the analysis

As was mentioned above, files `HistoCutResult.root` and `FINAL.root` have all needed information. Use macro `printFinal.C` to display this information in a simple and readable way:

```
root -l -b -q printFinal.C'("path-to-FINAL.root")'
```

This will print out all needed information for each centrality class. This macro also can save output information in several formats: latex, csv tables and C++ code.
Example of `printFinal.C` saving in latex table:

```
root -l -b -q printFinal.C'("path-to-FINAL.root","./example.tex")'
```

Example of `printFinal.C` saving in csv table (compatible with LibreOffice and MS Excel):

```
root -l -b -q printFinal.C'("path-to-FINAL.root","./example.csv")'
```

Example of `printFinal.C` saving in C++ code:

```
root -l -b -q printFinal.C'("path-to-FINAL.root","./example.C")'
```

After `printFinal.C` generates output C++ code, one can use `Float_t GetCentMult(Int_t)` as a function which returns centrality percent (for example, for 0-10% centrality class, the function will return 15.) based on input multiplicity value.

References

[1] C. Loizides, J. Nagle, and P. Steinberg, *SoftwareX*, vol. 1-2, pp. 13–18, 2015.

[2] C. De Jager, H. De Vries, and C. De Vries, *Atomic Data and Nuclear Data Tables*, vol. 14, no. 5, pp. 479–508, 1974, Nuclear Charge and Moment Distributions.

[3] H. De Vries, C. W. De Jager, and C. De Vries, *Atom. Data Nucl. Data Tabl.*, vol. 36, pp. 495–536, 1987.

[4] P. D. Group, P. A. Zyla, R. M. Barnett, *et al.*, *Progress of Theoretical and Experimental Physics*, vol. 2020, no. 8, 2020, 083C01.