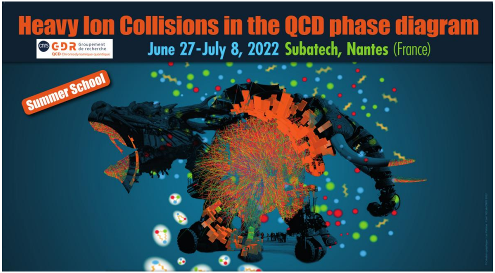
Tutorial session : running EPOS and RIVET

Johannès JAHAN (Ph.D. student) - Subatech / CNRS / Nantes Université

Damien VINTACHE - Subatech / CNRS



With the support of : Klaus WERNER - Subatech / Nantes Université

Table of Contents

Program of this tutorial

In this tutorial, we will work with **EPOS 3.259+** *(last tuned version of EPOS 3* [1] *upgraded)* and **RIVET 3.1.5** *(last version of RIVET* [2] *released so far)*.

What you will do :

- learn to write files to parametrise e^+e^- and pp/pA/AA simulations with EPOS
- run simulations with EPOS
- see the content of the main output formats from EPOS
- use *(provided)* HepMC data files from EPOS to analyse it with RIVET
- produce plots with RIVET

What you won't do :

- use EPOS 4
- run RIVET analyses for e^+e^- collisions *(issues in the HepMC files for such system)*

N.B. : *all words in blue in these slides are hyperlinks to the corresponding webpages*

[1] K. Werner et al., *Phys. Rev. C* **89** (2014), 064903

[2] C. Bierlich et al., *SciPost Phys.* **8** (2020), 026

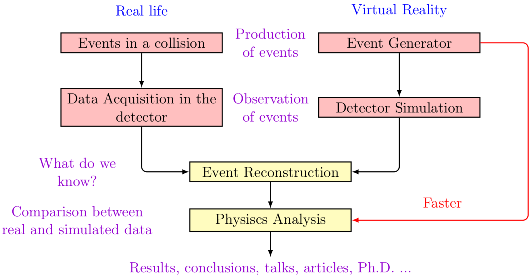
Contents

What is EPOS ?

Event generators are programs made to compute models in order to simulate every step of a collision (e.g. EPOS, PYTHIA [3]...).

Advantages : - perfect detector, as final-state particles are all listed (no uncertainties)

*(indeed, there are always **some flaws** : one has to be careful on the applicability, and phenomenological approaches generally requires parametrisation)*



[3] T. Sjöstrand et al., *Comput. Phys. Commun.* **191** (2015) 159-177

[4] K. Werner et al., *Phys. Rep.* **350** (2001) 93-289

What is EPOS ?

Event generators are programs made to compute models in order to simulate every step of a collision (e.g. EPOS, PYTHIA [3]...).

Advantages : - perfect detector, as final-state particles are all listed (no uncertainties)

*(indeed, there are always **some flaws** : one has to be careful on the applicability, and phenomenological approaches generally requires parametrisation)*

Energy conserving quantum mechanical approach, based on
Partons, parton ladders, strings,
Off-shell remnants, and
Saturation of parton ladders

Multi-purpose event generator based on parton-based Gribov-Regge Theory [4], using relativistic hydrodynamic simulation to mimic the fluid behaviour of the QGP.

Can simulate with the same formalism any type of high-energy collision consistently :

$e^- + e^+$ $e^- + p$ $p + p$ $p + A$ $A + A$

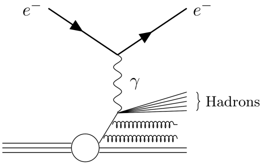
[3] T. Sjöstrand et al., *Comput. Phys. Commun.* **191** (2015) 159-177

[4] K. Werner et al., *Phys. Rep.* **350** (2001) 93-289

The motivations behind the PBGRT

Parton model

Mainly used for inclusive cross-section calculations



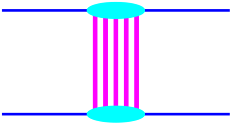
Deep Inelastic Scattering

Problems :

- can only calculate cross-section for hard processes $\rightarrow$ not suitable alone for HIC

Gribov-Regge theory

EFT for Multiple *Pomeron* Interaction



(K. Werner et al., 2000)

Inconsistencies :

- energy conserved for particle production but NOT for cross-section calculations
- although multiple scattering approach, all interactions are not treated equally

Solution : merge both into a formalism treating consistently hard and soft scattering
$\Rightarrow$ Parton-based Gribov-Regge Theory !

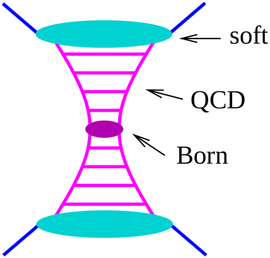
Main principle of PBGRT

Based on S-matrix approach where, from : $\langle f|\hat{S}|i\rangle = S_{fi} = \delta_{fi} + i(2\pi)^4\delta(P_f - P_i)T_{fi}$
we got the total inelastic cross-section σ_{tot} with the optical theorem (see details *here*)

$$\sigma_{tot} = \frac{1}{2s}(2\pi)^4\delta(p_f - p_i)\sum_f |T_{fi}|^2$$

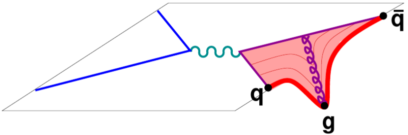
In the PBGRT, one elementary interaction is modeled as a *Pomeron*, each of them giving a contribution to the total T-matrix.

- **Soft process ($Q^2 < 1$ GeV) :** mainly elastic scatterings, parametrised T-matrix (Regge poles)
- **Hard process ($Q^2 > 1$ GeV) :** pQCD applicable, computed T-matrix (DGLAP equation)
- **Semi-hard process ($Q^2 > 1$ GeV $q_{sea}/\overline{q}_{sea}/g$) :** using both previous formalisms



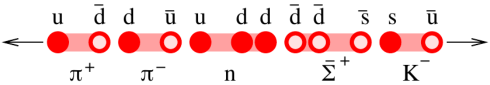
The string model

In the simplest system which is e^+e^-, the $q\bar{q}$ pair created is linked by a color field, forming what we call a relativistic string.
Such string can indeed have transverse kinks, caused by gluon emission, and evolve following the dynamics of a gauge invariant Lagrangian.



"Diagrammatic" view of a kinky string *(K. Werner, 2019)*

Eventually, it will fragment via production of $q(q) - \bar{q}(\bar{q})$ pairs, thus forming hadrons, following a so-called area law : $dP_{break} = \lambda.dA$ *(dA : infinitesimal area)*



Meson and baryon production from string breaking

Initial conditions

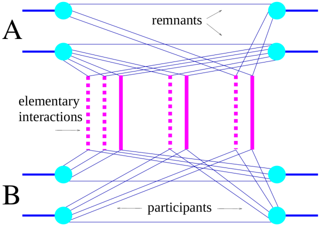
We show here the complete formalism of **EPOS 3** [5] , or how events are simulated for hadronic collisions (pp, pA or AA).

Primary interactions treated with PBGRT

Nuclei (A and B) interact by the exchange of multiple Pomerons in parallel (= MPI).

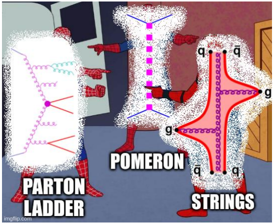
Uncut Pomerons are important as they give interference terms to σ_{tot} :

$$\sigma_{tot} = \sum_i \sum_j \int (G_i^{cut} - G_j^{uncut}).d^2 b$$



Schematic representation of a collision

(K. Werner et al., 2000)



Cut Pomerons will be used for particle production.

They can be assimilated to parton ladders *(= physical picture)*, or seen as flux tubes or kinky strings $(q - g - ... - \overline{q})$, like introduced before.

[5] K. Werner et al., *Phys. Rev. C* **89** (2014), 064903

Core-corona procedure

Multiple interactions within the PBGRT

(K. Werner, 2018)

In most of the pp collisions, the string picture is sufficient to simulate correctly the events.

However, for AA events in particular (*but not only...*), the density of strings become so important that they cannot decay independently.

We separate then, at a time τ_0 :

- core = bulk matter from high string density region ($\rho > \rho_0$)
- corona = high p_T segments (jets) escaping the core

In particular, for the string segments close to the surface with a local string density ρ, we evaluate if they have enough p_T to escape the core with :

$$p_T^{esc} = p_T - f_{E_{loss}} \int_\gamma \rho . dL$$

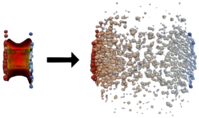
Medium evolution, hadronisation and re-scattering

Core evolution

Viscous 3D+1 hydrodynamics expansion based on a cross-over transition EoS

$+$

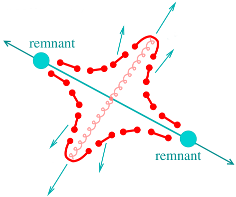
Statistical hadronisation of the medium via Cooper-Frye procedure [6]



(MADAI collaboration)

Corona evolution

Strings evolution $+$ fragmentation to produce hadrons



$\Downarrow$

Re-scatterings between formed hadrons with the UrQMD model [7] until chemical freeze-out, kinetic freeze-out

$\Downarrow$

Final state particle

[7] F. Cooper & G. Frye, *Phys. Rev. D* **10** (1974), 186

[7] M. Bleicher et al., *J. Phys. G* **25** (1999), 1859-1896

Towards the next public release

We are currently working on the development of a new version, **EPOS 4** [8], planed to be released publicly during 2022.

This new version include some major changes compare to elder versions :

- new developments on parton saturation scale, now depending on $N_{pom} + N_{part}$ and on each Pomeron's energy (instead of a simple constant Q_0 originally)
- possibility to use a new Equation of State including a critical point and a 1st order phase transition
- a microcanonical decay of the core part, replacing the grand-canonical Cooper-Frye procedure

and some minor/technical updates :

- solving issues in the list of particles decay channels (especially for some quarkonia)
- adding a new output format to make EPOS usable with RIVET

[8] K. Werner, et al., *Phys. Atom. Nucl.* **84** (2021), 1026–1029

Summary

- **EPOS** uses a unique approach to treat ALL systems (pp, pA, AA) :
  - **primary interactions** treated with a Gribov-Regge-based mutliple scattering approach, including parton saturation effects
  - **secondary interactions** based on a core-corona separation of jet hadrons and fluid, which expands using viscous hydrodynamical, with final-state hadronic cascades
- Reproduces naturally many flow-like features (even in small systems)
- Includes heavy quarks production (since EPOS 3), enabling sophisticated coupling to an energy-loss model (**EPOS-HQ**) to study interaction of HQs with a dynamical fluid evolution
- **EPOS 4** under validation, coming soon
  (many improvements on saturation, hadronisation...)

A bit more about EPOS...

These slides are coming from the following presentation : J. Jahan - QATs 2022
(more details there on collectivity in small systems and heavy flavours)

More references about EPOS :

- primary interactions & hydrodynamics in EPOS
- hydrodynamics in EPOS
- heavy flavours in EPOS (see also here *[French]*),
- EPOS-HQ project
- jet-fluid interaction in EPOS

Recent developments for EPOS 4 :

- parton saturation (*not the final version though*)
- BEST equation of state inclusion (see work from M. Stefaniak)
- microcanonical decay

Contents

Prerequisites

For simplicity, we will use **Docker** to run EPOS in a "container".
This way, you will benefit from a perfectly functional portable version, avoiding the necessity to install EPOS natively, with all its dependencies, on your computer.

Here are then the first necessary steps to follow, before to start this tutorial :

1. **Install Docker** *(for Linux, Windows or Mac)* by following instructions from : https://docs.docker.com/engine/install/
2. **Download** the image **epos3259_plus.tar.gz** and **EPOS+RIVET_Tuto.tgz** from : https://box.in2p3.fr/index.php/s/CgzXkqEDJPogcDc

You can then load this image, and check it worked (**epos3259+** should appear), with :

```
> sudo docker load --input epos3259_plus.tar.gz
> sudo docker images
```

__N.B. :__ normally, you would need to specify "`sudo`" before using any Docker command ; to get rid of this, do :

```
> sudo groupadd docker
> sudo usermod -aG docker $USER
> newgrp docker
```
(for Linux ; otherwise disconnect+reconnect your session to apply changes)

(more information here : https://docs.docker.com/engine/install/linux-postinstall/)

Simulation of e^+e^- events with EPOS

To run simulations with EPOS, we need first to define a **.optns** file, containing all the information about the system (projectiles, energy) and the run (number of events, other options...).

Create a working directory (let's call it "**Tuto_EPOS/**"), and put there **ee91.optns** extracted from **EPOS+RIVET_Tuto.tgz**.

```
application ee
  set engy 91.2               ! Center-of-mass energy of the collision (GeV)
  set nevent 10               ! Number of events simulated
  set modsho 1                ! Print a message every 'modsho' event(s) simulated
  nodecays 130 -130 -20 end   ! Block the decays of particles (K+ K- and Klong here)

  print * 2                   ! Print the events history in a .check file
```

WARNING : never use tabs in **.optns** files !

To link your directory **[..]/Tuto_EPOS** to the one in the Docker image :

```
> docker run --rm -v /PATH/TO/Tuto_EPOS:/home/EPOS/optns:Z
                        --entrypoint bash -it epos:3259_plus
```

It will open a new shell, in which you can now execute *(like if you had EPOS installed natively)* :

```
> ${EPO}epos ee91
```

.check output

You should now have in your repertory **Tuto_EPOS/** a new file called **z-ee91.check**.

ior	jor	i	ifr1	ifr2	id	ist	ity	pt	m	E	y
0	0	1	0	0	12	1	0	0.000E+00	0.511E-03	0.456E+02	0.121E+02
0	0	2	0	0	-12	1	0	0.000E+00	0.511E-03	0.456E+02	-.121E+02
1	2	3	4	9	10	1	0	0.000E+00	0.912E+02	0.912E+02	0.000E+00
3	4	4	10	0	4	21	30	0.442E+01	0.000E+00	0.242E+02	-.238E+01
3	4	5	10	0	9	21	30	0.473E+01	0.000E+00	0.197E+02	-.210E+01
3	4	6	10	0	9	21	30	0.355E+00	0.000E+00	0.750E+00	-.138E+01
3	-4	7	10	0	9	21	30	0.173E+01	0.000E+00	0.181E+01	0.304E+00
3	-9	8	10	0	9	21	30	0.142E+01	0.000E+00	0.144E+01	-.167E+00
3	-4	9	10	0	-4	21	30	0.236E+01	0.000E+00	0.433E+02	0.360E+01
4	9	10	11	28	800010001	29	30	0.104E-01	0.912E+02	0.912E+02	-.119E-06
10	0	11	29	30	-341	1	30	0.312E+01	0.211E+01	0.199E+02	-.235E+01
10	0	12	31	32	-131	1	30	0.142E+01	0.918E+00	0.738E+01	-.215E+01

"**ior**"/"**jor**" : *parents of particle "**i**"*
"**ifr1**"/"**ifr2**" : *children of particle "**i**"*

You can clearly see the mechanism happening here :
the colliding e^- and e^+ annihilate into a photon, which is giving birth to a chain of partons $c - g - \ldots - g - \bar{c}$ forming a string.

N.B. : *particle IDs in EPOS differ from the PDG standards ; see* ***idt.dt*** *from* ***EPOS+RIVET_Tuto.tgz***

Simulation of pp/pA/AA events with EPOS

To run simulations for any hadronic system, you need a different file.
Extract now **pp7T.optns** from **EPOS+RIVET_Tuto.tgz** and put it in **Tuto_EPOS**/.

```
application hadron
  set ecms 7000                ! Center-of-mass energy of the collision (GeV)
  set laproj 1                 ! Atomic number Z of projectile
  set maproj 1                 ! Mass number   A of projectile
  set latarg 1                 ! Atomic number Z of target
  set matarg 1                 ! Mass number   A of target

  set ninicon 1                ! Number of initial conditions used for hydro
  core  off                    ! Core-corona procedure off
  hydro off                    ! Hydrodynamics off
  hacas off                    ! Hadronic cascade (UrQMD) off

  set nfull 10                 ! Number of full hydrodynamical simulation achieved
  set nfreeze 1                ! Number of freeze-out events per full hydro event
  set modsho 1                 ! Print a message every 'modsho' event(s) simulated

  xinput KWt/icl70.optns       ! Input file containing centrality definitions
  set centrality 0             ! Define centrality class (0 = Min.Bias.)

  nodecays -20 end             ! Block the decays of particles (Klong here)

fillTree(C2)                   ! Record the events in Root files
```

If you want to activate hydro evolution of the core + final-state hadronic cascades :
*(hydro NOT recommended now, or only with "**nfull=1**" and "**nfreeze=10**")*

```
 core full                     ! Core-corona procedure ON
 hydro x3ff                    ! Hydrodynamics ON
 hacas full                    ! Hadronic cascade (UrQMD) ON
```

ROOT output

In order to produce **ROOT files** containing the simulated events (based on **ROOT 5.8**), add the flag "`-root`" before the name of the file to execute :

```
$> ${EPO}epos -root pp7T
```

Here is the list of variables stored in the Trees of ROOT files produced with **EPOS 3** :

```
iversn = new int ;          // EPOS version number
laproj = new int ;          // atomic number projectile
maproj = new int ;          // mass number projectile
latarg = new int ;          // atomic number target
matarg = new int ;          // mass number target
engy   = new float ;        // energy in the cms in GeV
nfull  = new int ;          // number of full events
nfreeze= new int ;          // number of freeze-outs per full event (to increase stat)

np = new int ;              // number of particles in the event
bim = new float ;           // impact parameter (usually; other choices are possible)
zus = new float [nmax];     // private use
px = new float [nmax];      // p_x of particle
py = new float [nmax];      // p_y of particle
pz = new float [nmax];      // p_z of particle
e = new float [nmax];       // energy of particle
x = new float [nmax];       // x component of formation point
y = new float [nmax];       // y component of formation point
z = new float [nmax];       // z component of formation point
t = new float [nmax];       // formation time
id = new int [nmax];        // particle id  (see file "KWt/idt.dt")
ist = new int [nmax];       // particle status (hadron last generation (0) or not (1); other numbers refer to partons, Pomerons, etc)
ity = new int [nmax];       // type of particle origin (20-29 from soft strings, 30-39 from hard strings, 40-59 from remnants, 60 from fluid)
ior = new int [nmax];       // index of father  (resonance decay products have only a father)
jor = new int [nmax];       // index of mother  (mother needed for exemple for strings: partons between ior and jor constitute the string)
```

+ list of variables added in **EPOS 4** (also in **3.259+**) :

```
sigtot    = new float ;     // corresponding pp cross-section
nhard     = new int ;       // number of elementary hard parton-parton scatterings (set to 0)
npartproj = new int ;       // number of projectile's nucleons participants according to Glauber
nparttarg = new int ;       // number of target's nucleons participants according to Glauber
nspecp    = new int ;       // number of spectators protons according to EPOS
nspecn    = new int ;       // number of spectators neutrons according to EPOS
```

HepMC output

With **EPOS 4**, it is possible to produce **ASCII HepMC files** *(based on **HepMC 2.06.09**[9])* which contain some part of the simulated events, making it compatible with **RIVET**.

To produce such files (also possible here with **EPOS 3259+**), you shall add "`set ihepmc 1`" in the **.optns** file and run :

```
$> ${EPO}epos -hepmc pp7T
```

You can also extract **PbPb3T.optns** from **EPOS+RIVET_Tuto.tgz** and run it. In the **.hepmc** file you will get, there is one more line "`H`" at the head of each event, which contain some information for heavy-ion collisions :

```
E 1 -1 -1.0000000000000000e+00 -1.0000000000000000e+00 -1.0000000000000000e+00 0 -1 235 10001 10002 0 0
U GEV MM
C 8.4970962524414062e+10 0.0000000000000000e+00
H 0 11 16 44 235 154 0 0 0 1.3447725296020508e+01 0 0 0
V -1 0 0 0 0 0 2 2451 0
P 10001 1000822080 0 0 -2.8703993462420581e+05 2.8704000000000000e+05 1.9372901600000000e+02 4 0 0 -1 0
P 10002 1000822080 0 0 2.8703993462420581e+05 2.8704000000000000e+05 1.9372901600000000e+02 4 0 0 -1 0
P 10003 2212 0 0 1.3799996337890625e+03 1.3799999527566974e+03 9.3826997280120850e-01 1 0 0 0 0
P 10004 2212 0 0 1.3799996337890625e+03 1.3799999527566974e+03 9.3826997280120850e-01 1 0 0 0 0
  .
  .
  .
P 10384 -211 4.1446900367736816e-01 1.5941139459609985e+00 2.2229127883911133e+01 2.2290504482655265e+01 1.3956999778747559e-01 1 0 0 0 0
P 10385 2212 2.4639999866485596e-01 -2.7039399743080139e-01 1.3799184570312500e+03 1.3799188245081380e+03 9.3826997280120850e-01 1 0 0 0 0
P 10386 2212 -5.7721100747585297e-02 -8.2814702764153481e-03 1.2756362304687500e+03 1.2756365768648102e+03 9.3826997280120850e-01 1 0 0 0 0
```

N.B. : *detailed information about the structure of such files are given in HepMC 2 user manual (Sec.6)*

[9] https://hepmc.web.cern.ch/hepmc/

HepMC output

We provide in these files the following general information :

- the cross-section (in pb) of an equivalent p-p collision
- the numbers of projectile and target participants as well as *NN* collisions from Glauber model[10] calculation
- the numbers of neutron and proton spectators from EPOS
  (correspond to the number of spectators listed in the file : $N_{part}^{proj+targ} + N_{spec}^{n+p} \neq A^{proj+targ}$*)*
- the impact parameter (in fm)

We could also record the whole history of each event,
but it's not really relevant for HIC, and it would take a lot of disk space.
Thus, we "only" record the target + projectile, the final-state particles, as well as their parents (+grand-parents) when they decay with a lifetime $\tau > 10^{-20} s$
(+ some exceptions, i.e. J/Ψ*,* $\Psi(2S)$ *and* $\Upsilon(1S,2S,3S,4S)$*).*

$\Rightarrow$ enables to identify EM/weak feed-down contributions
+ particle reconstruction via specific branchings

N.B. : *detailed information about the content of EPOS* ***.hepmc*** *files is given in* ***HepMC.txt*** *from* ***EPOS+RIVET_Tuto.tgz***

[10] Michael L. Miller et al., *Ann.Rev.Nucl.Part.Sci.* **57** (2007), 205-243

Contents

Prerequisites

For this tutorial, we will use the latest version released of **RIVET** [11], version 3.1.6.
Follow these steps to download and make RIVET ready to use on your machine :

To download a Docker image of RIVET version 3.1.6 (work the same for any version) :

```
> docker pull hepstore/rivet:3.1.6
```

Then, in order to link your /**Tuto_EPOS** directory to the one in the RIVET image (like we did for EPOS) :

```
> docker run --rm -v /PATH/TO/Tuto_EPOS:/work:Z
                        --entrypoint bash -it hepstore/rivet:3.1.6
```

Finally, you can test that RIVET works properly from the new shell opened with :

```
> rivet --version
```

N.B. : otherwise, you can define aliases in order to work from your directory, without opening a new shell, by following instructions from : https://gitlab.com/hepcedar/rivet/-/blob/release-3-1-x/doc/tutorials/docker.md

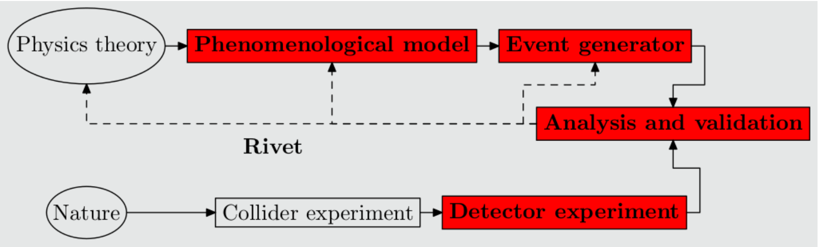
[11] https://gitlab.com/hepcedar/rivet

Robust Independant Validation of Experiment and Theory

RIVET is a ”software” based on C++ libraries, installed with different packages :

- YODA : Python libraries and classes used for analyses and histogramming
- HepMC : simulations recording and reading for analyses
- FastJet : recombination algorithms, mainly used for jet analyses

Purpose : offer a simple and standardised tool to automatise comparison between event generators simulations and experimental data



(C. Bierlich, 2019)

Robust Independant Validation of Experiment and Theory

RIVET is a "software" based on C++ libraries, installed with different packages :

- YODA : Python libraries and classes used for analyses and histogramming
- HepMC : simulations recording and reading for analyses
- FastJet : recombination algorithms, mainly used for jet analyses

Purpose : offer a simple and standardised tool to automatise comparison between event generators simulations and experimental data

RIVET contains many analyses based on publications from many different experiments (data included), and develops thanks to contributions from the users community (from both experiments & theory).

Advantages :

- provides huge and constantly growing library of data and analyses
- easy to handle (a lot of documentation + helpful reactive developers)
- don't have to "think about" the analysis details anymore

$\Rightarrow$ **RIVET is a very useful tool for us !**

Composition of an analysis

Each analysis is corresponding to a given publication, and based on 4 files :

```
#include "Rivet/Tools/Percentile.hh"
#include <complex>
#include <iostream>
#include <string>

namespace Rivet {

    class STAR_BES_Centrality: public SingleValueProjection{
    public:
      STAR_BES_Centrality () {
        declare(ChargedFinalState(Cuts::abseta < 0.5 && Cuts::absrap < 0.1 && Cuts::pT > 0.2*GeV)
      }
      /// Clone on the heap.
      DEFAULT_RIVET_PROJ_CLONE(STAR_BES_Centrality);
    protected:
      void project(const Event& e) {
        clear();
        double estimate = apply<FinalState>(e, "STAR_BES_Centrality").particles().size();
        set(estimate);
```

ANALYSIS_NAME.cc
(analysis code)

```
BEGIN YODA_SCATTER2D_V2 /REF/STAR_2017_PRC96_044904/d20-x04-y01
Variations: [""]
IsRef: 1
Path: /REF/STAR_2017_PRC96_044904/d20-x04-y01
Title: ~
Type: Scatter2D
---
# xval   xerr-   xerr+   yval    yerr-   yerr+
1.390000e+01   4.400000e+00   4.400000e+00   4.410000e-01   6.000000e-02   6.000000e-02
2.560000e+01   7.100000e+00   7.100000e+00   4.560000e-01   6.000000e-02   6.000000e-02
4.470000e+01   8.700000e+00   8.700000e+00   4.910000e-01   6.400000e-02   6.400000e-02
7.180000e+01   1.020000e+01   1.020000e+01   5.300000e-01   6.900000e-02   6.900000e-02
1.099000e+02   1.100000e+01   1.100000e+01   5.850000e-01   7.400000e-02   7.400000e-02
1.602000e+02   1.020000e+01   1.020000e+01   5.730000e-01   7.300000e-02   7.300000e-02
2.262000e+02   7.900000e+00   7.900000e+00   5.680000e-01   7.100000e-02   7.100000e-02
2.904000e+02   6.000000e+00   6.000000e+00   5.910000e-01   7.500000e-02   7.500000e-02
3.374000e+02   2.100000e+00   2.100000e+00   5.880000e-01   7.500000e-02   7.500000e-02
END YODA_SCATTER2D_V2
```

ANALYSIS_NAME.yoda
(experimental data)

```
Name: STAR_2017_PRC96_044904
Year: 2017
Summary: Bulk Properties of the medium produced in Relativistic Heavy-Ion Collisic
Experiment: STAR
Collider: RHIC
InspireID: 1510593
Status: UNVALIDATED
Authors:
 - Johannes Jahan <johannes.jahan@etu.univ-nantes.fr>
 - Gabriela Pokropska <g.pokropska@gmail.com>
 - Maria Stefaniak <maria.stefaniak@subatech.in2p3.fr>
References:
```

ANALYSIS_NAME.info
(information about paper, beam, author...)

```
BEGIN PLOT /STAR_2017_PRC96_044904/d20-x04-y01
Title=$p/\pi^+$ (Au+Au 7.7 GeV/c)
XLabel=$\langle N_{part} \rangle$
YLabel=$p/\pi^+$
LogY=0
YMin=0
YMax=0.8
XMax=360
ConnectBins=0
LegendYPos=0.2
# + any additional plot settings you might like, see make-plots documentation
END PLOT
```

ANALYSIS_NAME.plot
(plotting options)

Analyses in RIVET

Catalogue of the analyses available in RIVET : https://rivet.hepforge.org/analyses.html

or by typing :

```
> rivet --list-analyses
```

> **WANTED: Analysis code**
>
> We need your analyses! Preserving analysis logic in a re-runnable, re-interpretable form is a key part of scientific reproducibility and impact at the LHC and other HEP experiments. If you are member of an experimental collaboration, please have a look at our wishlist and help us by providing us with Rivet analyses for your publications. This will also ensure that your measurements get used (and cited)!

Proposition of a list of relevant analyses to try during this tutorial :

For p-p @ 7 TeV :

- ALICE_2010_S8625980 (charged mult.)
- ALICE_2015_I1357424 (light hadrons)
- ALICE_2012_I944757 (D mesons)
- ALICE_2017_I1645239 (Λ_c^+)
- ALICE_2016_I1471838 (strange hadrons)

For Pb-Pb @ 2.76 TeV/A :

- ALICE_2010_I880049 (charged mult.)
- ALICE_2013_I1225979 (charged $dN/d\eta$)
- ALICE_2012_I1126966 (light hadrons)
- ALICE_2014_I1243865 (strange hadrons)
- ALICE_2019_I1723697 (flow)

Running analyses with RIVET : p-p

We'll first run some analyses using a set of 50k events of p-p collisions at $\sqrt{s} = 7$ TeV, obtained with **EPOS 4.32.8**, stored in **EPOS-4.32.8_pp7T_50k-evts.hepmc**.

IMPORTANT : preliminary version of EPOS 4, tuning not complete yet

In order to run any analysis of the catalogue with RIVET, you need to type :

```
$> rivet DATA_FILE.hepmc -a ANALYSIS_NAME -o ANALYSIS_NAME.yoda
```

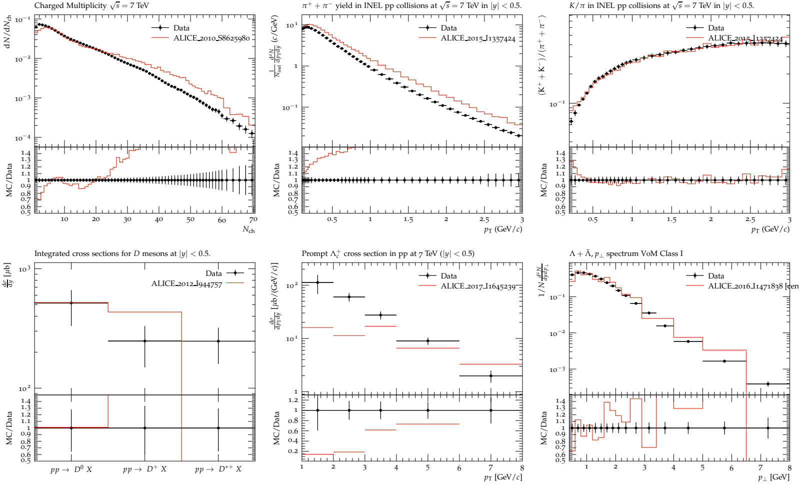
To get plots from the **.yoda** file obtained then, you can type :

```
$> rivet-mkhtml ANALYSIS_NAME.yoda -o ANALYSIS_NAME
```

It will create a folder called "**ANALYSIS_NAME**", containing all plots in **.pdf** and **.png**, with a generated URL where all the figures can be displayed.

In case of error message "*All analyses were incompatible with the first event's beam*", or if you want to run an analysis which is made for other systems/energies, you can enforce it by using the flag : `--ignore-beams`.

Some results you should obtain...



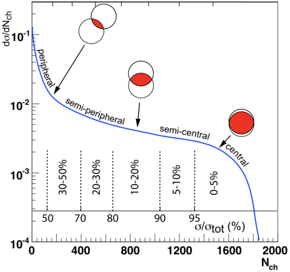
N.B. : *these data are only used for illustration ; they've been produced with a preliminary version of EPOS 4 (as already highlighted before), which explains why they don't fit well to the experimental results*

Running analyses with RIVET : A-A

Let's run now some analyses for Pb-Pb collisions on **EPOS-4.32.8_PbPb2.76T_2k-evts.hepmc**.

Heavy-ion analyses generally require a centrality calibration (as well as some p-p ones for high-multiplicity events, like **ALICE_2016_I1471838**).

Hence, when running your main analysis, you need to provide the results of the calibration **CALIB.yoda** (***ALICE_2015_PPCentrality** or **ALICE_2015_PBPBCentrality** for the suggested analyses here)* by using the "`-p`" flag :



J. Jia (2018)

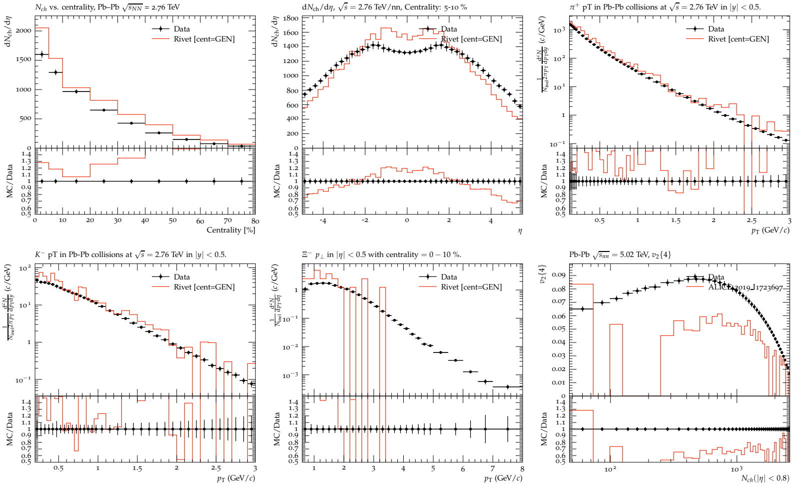
```
> rivet DATA_FILE.hepmc -a ANALYSIS_NAME:cent=GEN
                          -p CALIB.yoda -o ANALYSIS_NAME.yoda
```

("`EXP`" : based on experimental mult. distr. / "`GEN`" : based on simulated mult. distr. / "`IMP`" : ...)

Last little tip if your analysis takes time : you can specify how many events maximum you want to analyse with `-n MAX` *(or simply use **Ctrl+C** to stop the running analysis).*

N.B. : *for more information on how to use RIVET, look at RIVET's ReadMe or C. Bierlich's "Hands-on session" (COST workshop 2019)*

Some other results you should obtain...



Thanks for your attention !

All remarks and suggestions will be welcome :)

Contacts : johannes.jahan@subatech.in2p3.fr / damien.vintache@subatech.in2p3.fr
klaus.werner@subatech.in2p3.fr