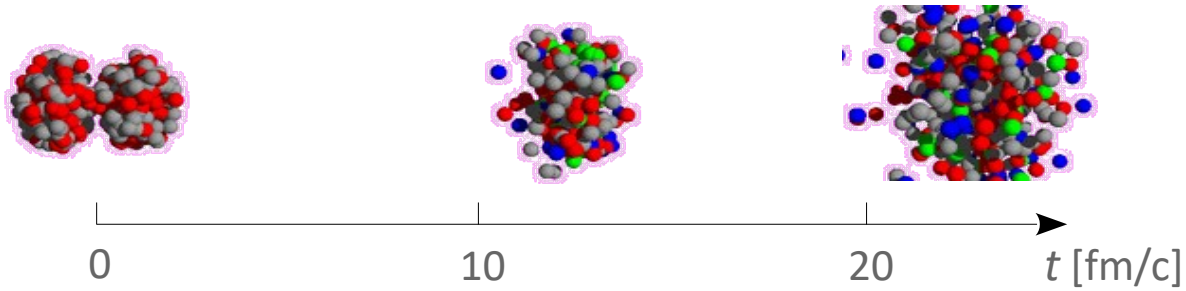
Final state of collision: statistical models

Introduction to Thermal-FIST and Therminator2

Krzysztof Piasecki

Statistical approach to heavy-ion collisions

- Sketch of a heavy-ion collision (IQMD simulation of Au+Au @ 1.5A GeV)



- Key assumption of **statistical models**: **thermodynamic equilibrium** occured at (or before) the freeze-out.
  - **Chemical equilibrium** : multiplicities of all the particles evolved to mutual equilibrium
  - **Thermal equilibrium** : phase space distributions (*p*, *E*, ...) evolved to their equilibrium forms (e.g. Fermi-Dirac or Bose-Einstein statistics, or Boltzmann statistics as simplification)
    Angular isotropy (θ, ϕ) .

- No time evolution considered. The system is at freeze-out.

  Within pure variants of statistical models, the whole motion of a system is a thermal motion.

  Sometimes hybrid approaches: energy is shared between thermal and collective motion
  (Siemens-Rasmussen model, Blast-Wave model, etc.)

- "Pure thermal" scenario within Grand-Canonical approach and assuming that hadron's mass is fixed:
  For a hadron of given type, its **multiplicity** (≡ **yield**) is assumed to have this form:

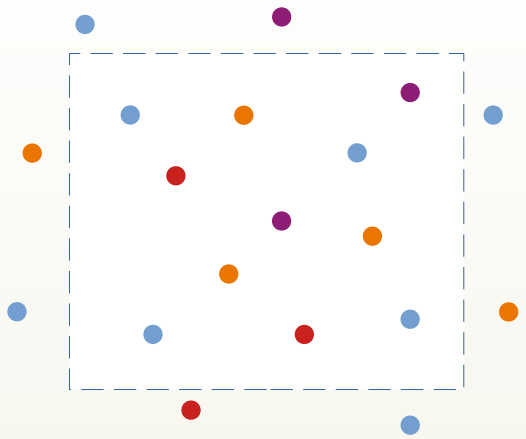
$$N_i = g_i \cdot V \cdot \int \frac{d^3 p}{(2\pi)^3} \frac{1}{e^{(E_i - \mu_i)\,/\,kT} \pm 1}$$

where: V = volume, T = temperature,
g_i = degeneracy factor (spin, isospin)

and the *chemical potential*: $\mu_i = b_i\,\mu_B + q_i\,\mu_Q + s_i\,\mu_S$

Thermodynamical ensembles

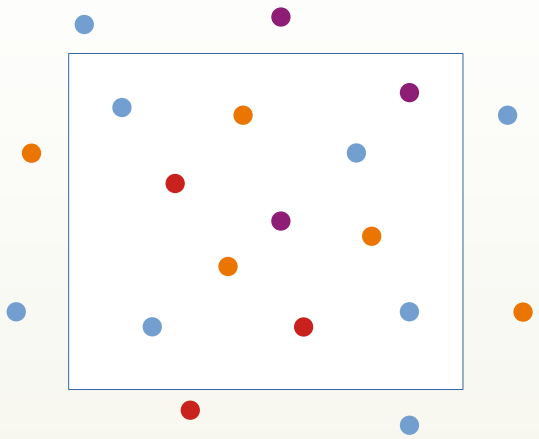
Grand Canonical Ensemble



System is in contact with reservoir, with which it exchanges energy and particles.

Form of Equilibrium:
In the system N, E = const *on average* (may fluctuate a bit)

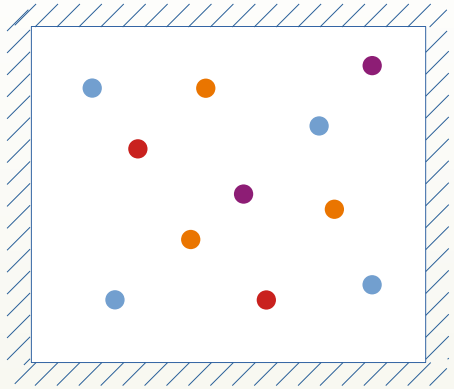
Canonical Ensemble



System is in contact with reservoir with which it exchanges energy only

Form of Equilibrium:
E = const *on average*
N = const always

Microcanonical Ensemble



System is fully isolated.

Form of Equilibrium:
N = const always.
E = const always.

- A system of colliding nuclei is in principle isolated (in vacuum).
  However, the microcanonical calculations are very CPU-consuming, so are performed very rarely.

  Rule of thumb: GCE approach is for larger-sized systems. The smaller volume → lower yields → CE or even MCE.

  Often, leading yields are large but strangeness yields are small.
  ⇒ Combined approach: GCE for bulk, CE for strangeness.

Statistical approach to heavy-ion collisions

Several codes were developed that intend to fit the predictions of the thermal model to the experiment.
Among them: **Thermus** and **Thermal-Fist** .

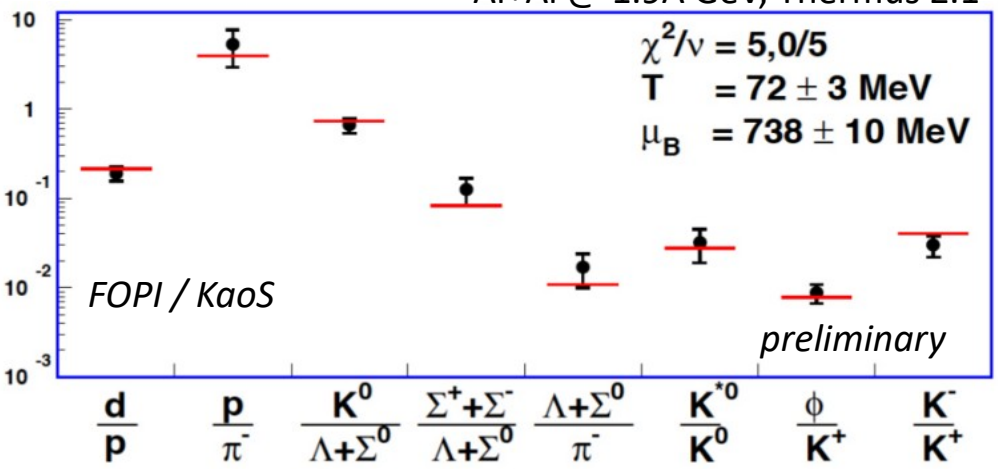
- **Input to calculation**:
  - Set of particle yields from a given experiment (A+A @ T_{Beam}, given centrality)
  - Data base of hadrons, including mass, width, isospin etc.
- **What these models do**:
  - fit temperatures, chemical potentials etc. to find the best agreement between theo and exp yields.
- **Output**:
  - found temperature, chemical potential, possibly volume etc. + list of predicted yields of hadrons
- **Caution**:
  - For particles with mass distribution:
    this distribution may or may not be modelled (Breit-Wigner + its extensions). 2B checked!
  - Most particles decay. An experimental yield may or may not include the feed-downs. 2B checked!

    Within model: non-zero yield of decayable particle does not contradict equilibrium.
    It means: decay rate is balanced with production rate.
    To compare model and experimental yields (the latter: with feeddowns from decays),
    the model yield is calculated *with* those feed-downs.

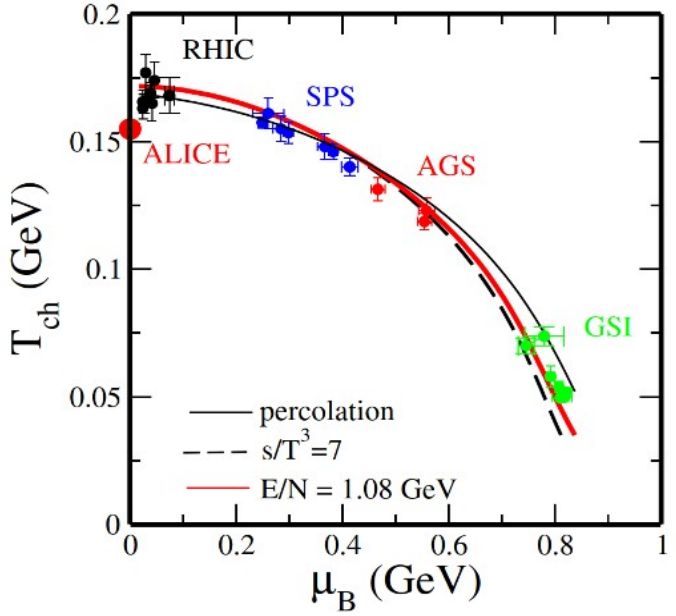
Statistical approach to heavy-ion collisions

- Despite strong assumptions, these models fit well to particle yields all throughout GeV – TeV energies:

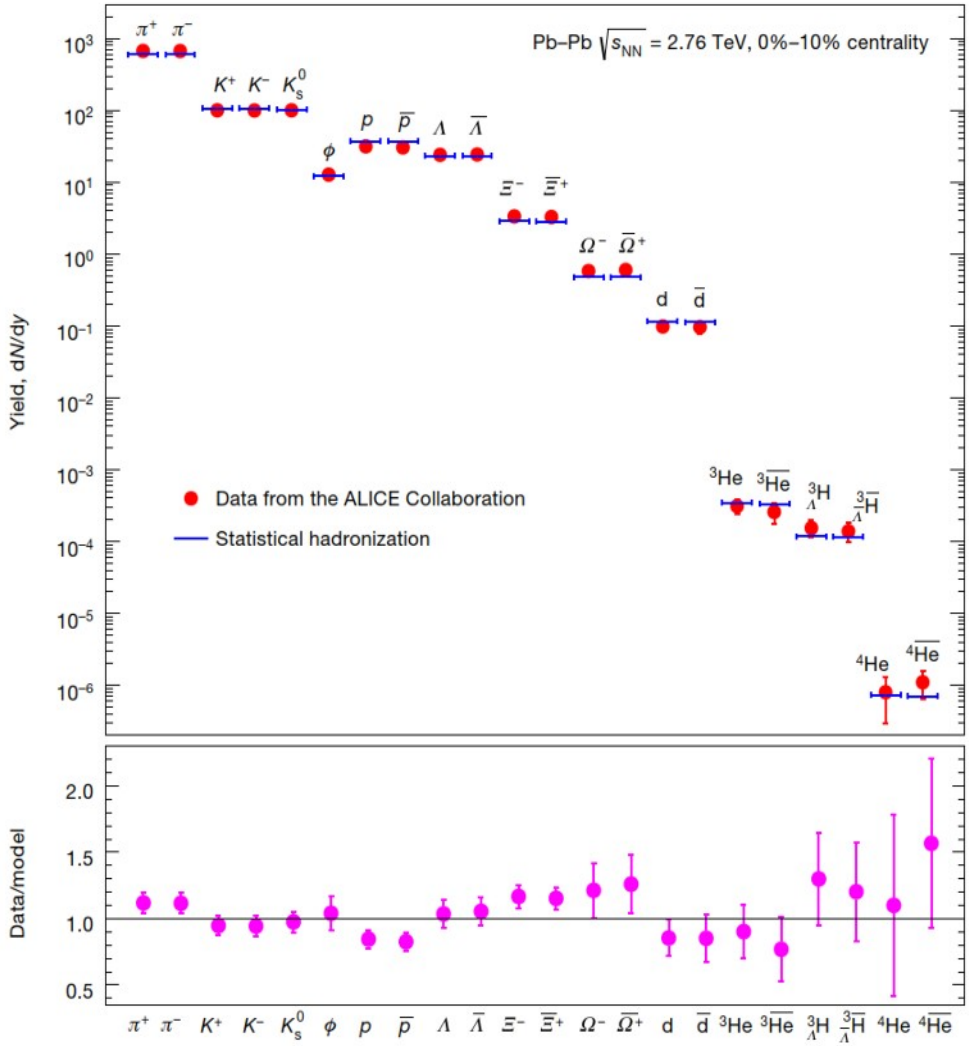
Al+Al @ 1.9A GeV, Thermus 2.1



- As a result, they point to the region on $T - \mu_B$ phase diagram reached by the colliding system(s) in the final stage:



J. Cleymans, EPJ Web of Conferences 95, 03004 (2015)



A. Andronic et al., Nature 561, 321 (2018)

Thermal-FIST : overview

- Thermal-FIST  : statistical model package for yields of particles emitted from Heavy-Ion collisions.
  Author: Volodymyr Vovchenko.

  All can be managed through a nice graphical GUI.
  This GUI can be also launched online, [here].
  Or can be included in your C++ code as a library.

  Quick start guide: [here]

  Papers: V. Vovchenko, H. Stoecker, Comput. Phys. Commun. 244, 295 (2019) [here] [arXiv]

  Sources: [GitHub] (see depencies, including Qt5)

- To run @ NPD's training computer:

  ① `mkdir tfist; cd tfist`
  ② `cp -r ~kpiasecki/soft/Thermal-FIST/ctnp/* .`
  ③ `nice ./QtThermalFIST &`

Thermal-FIST : preparation for fitting

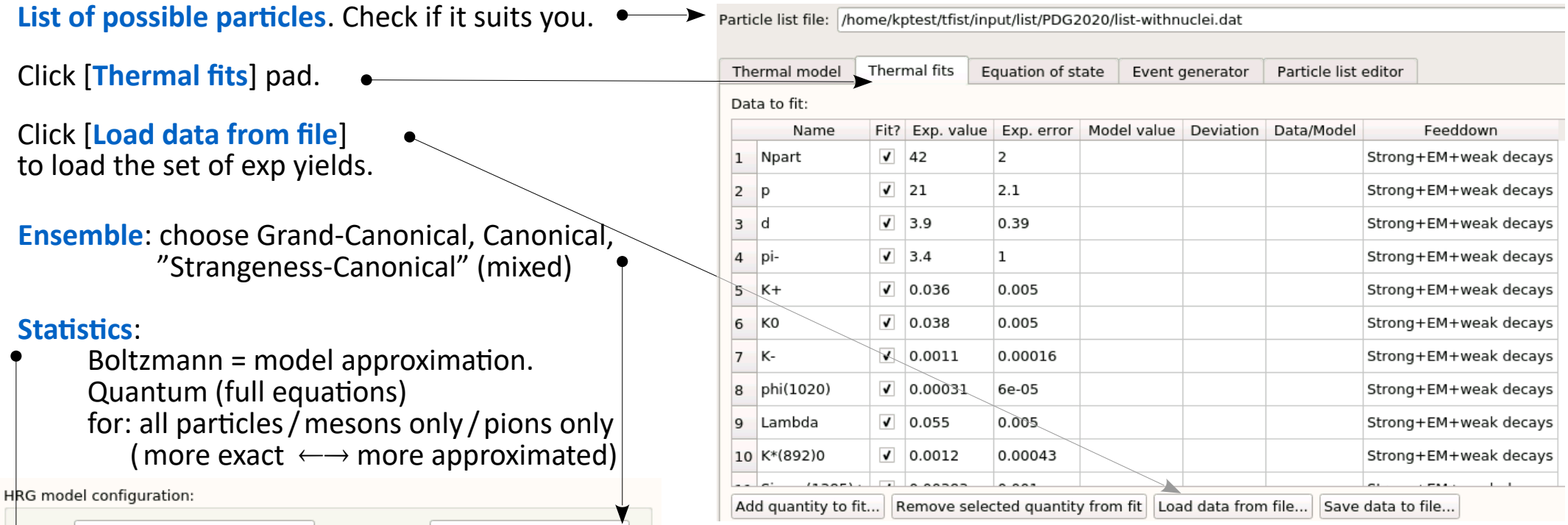
- **List of possible particles**. Check if it suits you.

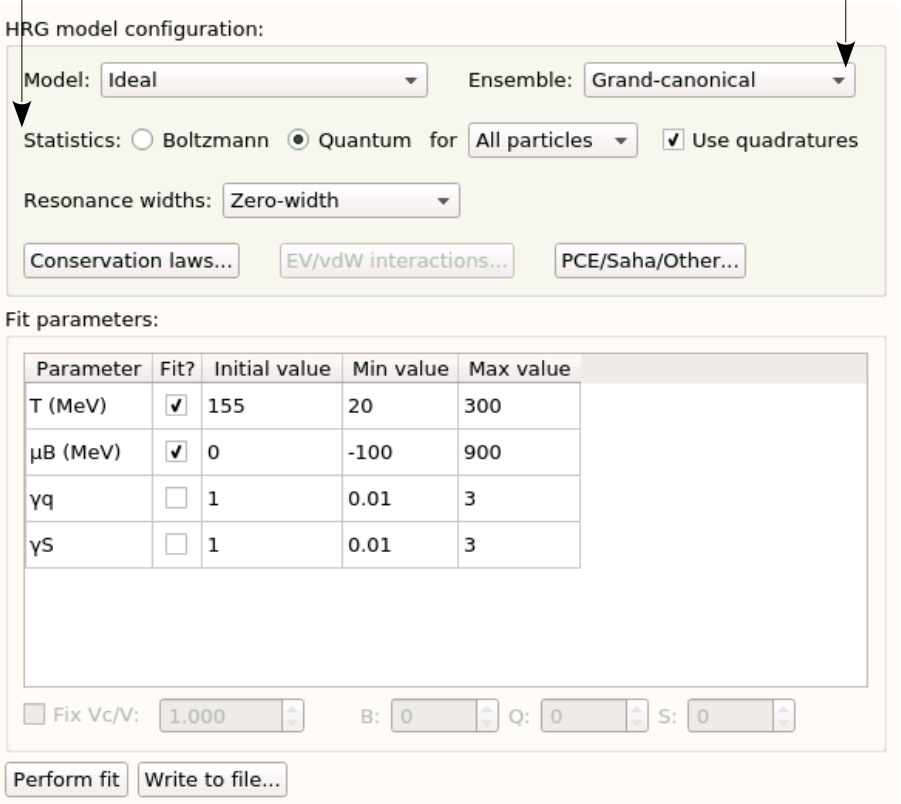
  Click [**Thermal fits**] pad.

  Click [**Load data from file**]
  to load the set of exp yields.

- **Ensemble**: choose Grand-Canonical, Canonical,
  "Strangeness-Canonical" (mixed)

  Statistics:
  Boltzmann = model approximation.
  Quantum (full equations)
  for: all particles / mesons only / pions only
  (more exact ⟷ more approximated)





Resonance widths:
- Zero-width [all masses are fixed at centroids]
- Const Breit-Wigner [mass = Breit-Wigner, but Γ = const]
- eBW [mass = Breit-Wigner, and Γ = f(mass)]

Conservation laws:
For given colliding nuclei (*A*,*Z*) the *Q*/*B* ratio must be conserved.
Make sure to type in this ratio.

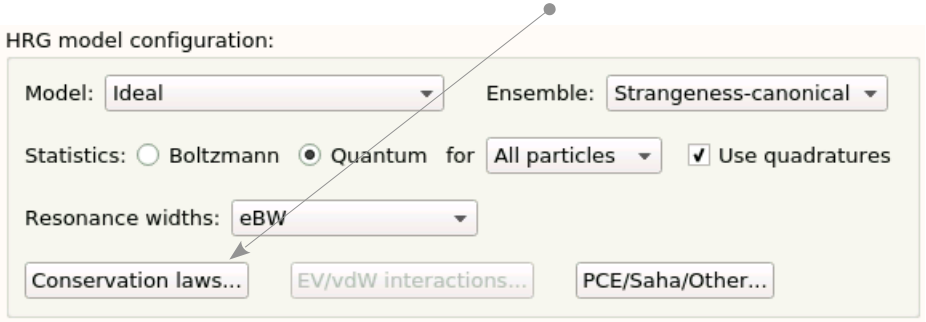
Fit parameters: propose starting values and fitting range.
☑ = parameter will be fitted
☐ = parameter is fixed

Click [**Perform fit**] and wait ...

Thermal-FIST : preparation for fitting

- Example: yields from Al+Al @ 1.9A GeV.

  Proposition of initial settings. Set Q/B to 13/27 = 0.482



Fit parameters:

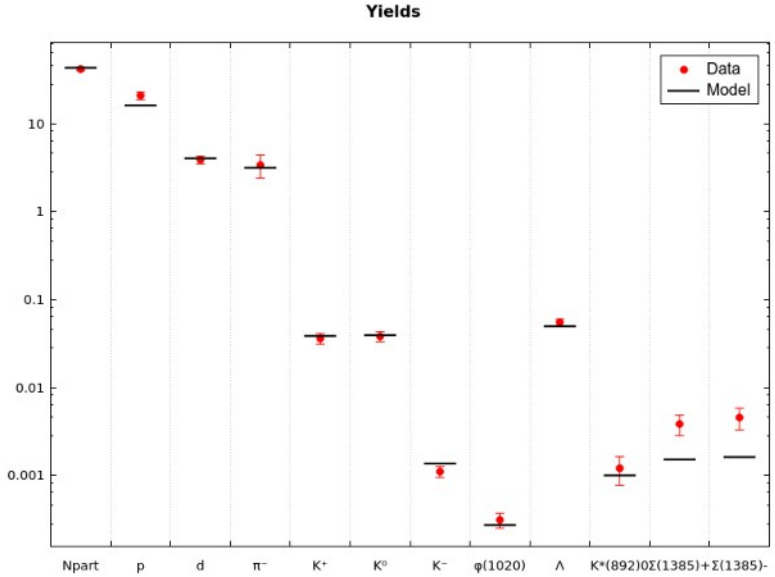
Parameter	Fit?	Initial value	Min value	Max value
T (MeV)	☑	70	20	160
R (fm)	☑	7	2	16
Rc (fm)	☑	3	2	16
μB (MeV)	☑	750	100	900
γq	☐	1	0.01	3
γS	☐	1	0.01	3

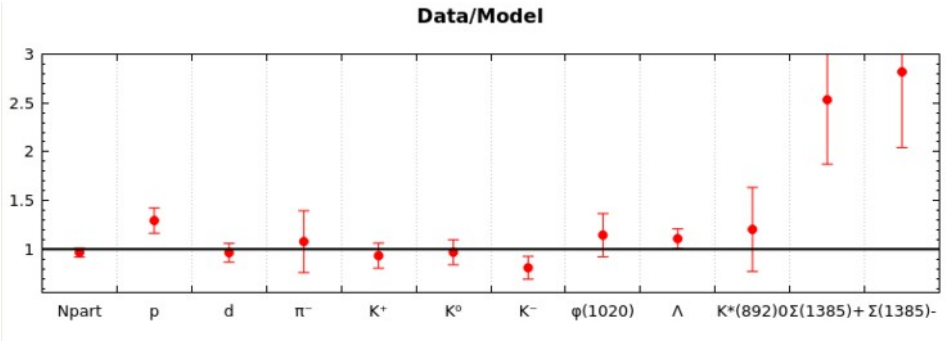
Fix Vc/V: 1.000 B: 0 Q: 0 S: 0

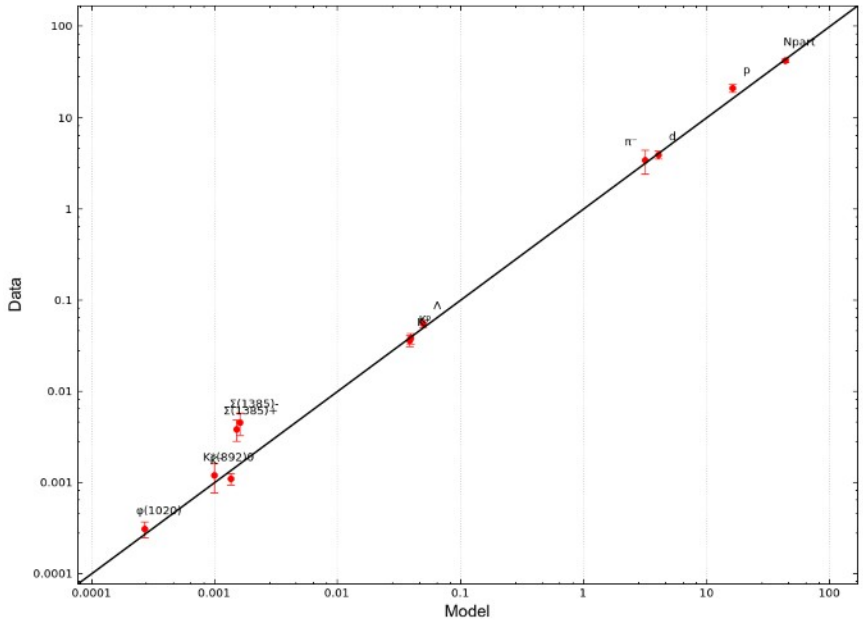
- You should get the following results:

Extracted parameters:

Parameter	Value	Error
T (MeV)	71.5834	1.48121
μB (MeV)	743.66	6.22032
γq	1	--
γs	1	--
R (fm)	5.6295	0.370721
V (fm^3)	747.304	147.637
Rc (fm)	3.17618	0.246619
Vc (fm^3)	134.216	31.2642
chi2/dof	21.5041/8	
chi2/dof	2.68801	



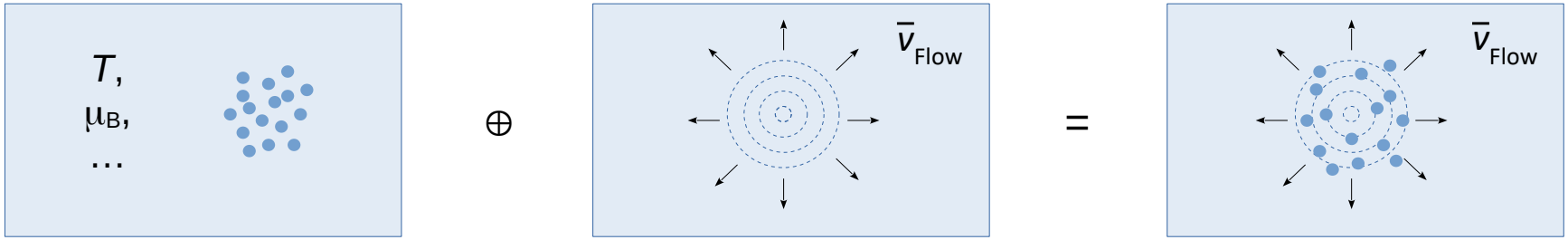




- Click [Write to file] to get the file with the list of yields according to the model.

Therminator2 : overview

- Program to generate distributions of particles emitted from ultra-relativistic heavy-ion collisions, at the final (freeze-out) stage. It makes following assumptions:
  - particles are in **thermodynamical equilibrium** (in their local frame)
  - but their thermal motion is superimposed on the **collective expansion** of a system as a whole.



Thermal motion in local frames (Bose-Einstein / Fermi-Dirac)

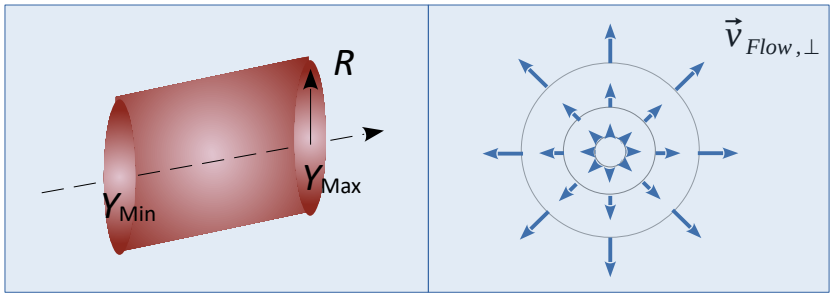
In CM, local frames expand, depending on their placement

Superposition as seen in CM (Global Frame)

A picture above of **radial** expansion (angular isotropy) is valid for $T_{Beam} \sim 1$ ... few A GeV.

- At higher energies (UrHIC, Ultra-relativistic Heavy-Ion Collisions) the experimental spectra agree more with the picture of cylindrically-expanding zone, parametrized by:

⇨ R (max. radius) , ΔY (longitudinal) rapidity range , $\bar{v}_{Flow,\perp}$.



$\bar{v}_{Flow,\perp}$ is not constant, but often an onion-like expansion is assumed:

$$\vec{\beta}_{Flow,\perp}(\rho) = \beta_{Flow,Surf} \cdot \left(\frac{\rho}{R}\right)^2 \hat{e}_\rho$$

This ansatz is called: **Blast-Wave model** (Ref. , [arXiv]).

Therminator2 : overview

- All these models assume that the freeze-out occurs at the same t_{Local} (moment of time in Local frame).
  But the Local → Global Lorentz transformation imposes that for different space points,
  the freeze-out occurs at different t_{Global} (moments of time in Global frame).
  A curve in Global space-time, describing the freeze-out points, is called the **freeze-out hypersurface** σ.

  ⇨ Within Therminator2 you can select one of the predefined hypersurfaces or model it yourself.

- **What Therminator2 does and what does not**:

  → It's not a fitter of thermal parameters to a set of particle yields.

  → You give T, μ_B, ..., type of collective expansion + its parameters.
  The code generates events with particles from this model.

- **Homepage**: therminator2.ifj.edu.pl

- **Papers**: [v1] A. Kisiel et al., Comput. Phys. Commun. 174, 669 (2006) [arXiv]
  [v2] M. Chojnacki et al., Comput. Phys. Commun. 183, 846 (2012) [arXiv]

- **Installation**: therminator2.ifj.edu.pl/install.html (requires ROOT)

  Particles and their decays are taken from the PDG database.
  Particle list file: `share/particles.data`. File with list of decays: `share/decays.data`

- **How to run** @ NPD's training computer:

```
①  mkdir therminator2;  cd therminator2
②  cp -r ~kpiasecki/soft/therminator2/ctnp/* .
③  . ./therminator2_start.sh                    ← each time you want to run
④  less events.ini
⑤  less fomodel/blastwave.ini
⑥  nice ./therm2_events
```

Therminator2 : overview

- **Steps of program operation**:

① Generation of average multiplicities of all the particle types. (about 15')
Particle types and their decays are listed in: `share/` → `particles.data` → `decays.data`
Average multiplicities are stored in: `events/{model}/multiplicity_NNN.txt`

② Generation of events. For each event:

▻ No. of particles of each type is pulled from Poisson distribution, with average obtained in ①

▻ Particles are generated with local momentum pulled from Fermi-Dirac / Bose-Einstein thermal distribution ...

▻ … and dispatched on the freeze-out hypersurface.
Their 4-momenta are boosted to the CM frame, according to flow at given position.

▻ Unstable particles are decayed according to the decay law with given lifetime τ
(2-body and 3-body decays are implemented)
The decay products are positioned just after the parent particle traversed straight line for given time.
Any unstable product is again subject to decay law, etc.

▻ Particles are written into output files. Both the decayed and final ones.

- **Types of output files**:

→ `fmultiplicity_NNN.txt` : List of particle types with average yields

→ `event.txt` : Text file with listing of events ⊕ their particles

→ `eventNNN.root` : Root file with TTree's of: events, particles and model information.
500 events are packed into 1 file. Next events are stored in further files.

Therminator2 : overview

- Let's look at events.ini input file. Here the basic settings of a simulation are defined.

```
[FreezeOut]
FreezeOutModel = fomodel/blastwave.ini

[Event]
# Number of events to generate
# default:   50000
NumberOfEvents = 10

# Event output file format
# available: root, root&text, text
# default:   root
EventFileType = root

[Primordial]
# Distribution of primordial particles multiplicity
# available: Poisson
# default:   Poisson
MultiplicityDistribution = Poisson

# Number of samples used in determination of
# primordial multiplicity and max. integrand value
# default:   5000000
IntegrateSamples = 5000000

[Random]
# Start each event with a new random seed taken
from current time (1)
# or do a constant seed (0)
# default:   1
Randomize = 1
```

```
[Directories]
# Directory with SHARE input files
# default:   share/
ShareDir = share/

# Directory with Freeze-Out Model
parameter files
# default:   fomodels/
FreezeOutDir = fomodel/

# Directory with ROOT macro files *.C
# default:   macro/
MacroDir = macro/

# Directory to write the events
# default:   events/
EventDir = events/

[Logging]
# Log file
# default:   therminator.log
LogFile = therminator.log
```

Therminator2 : overview

- Let's look at `blastwave.ini` input file. Here you can define the Blast-Wave model parameters.

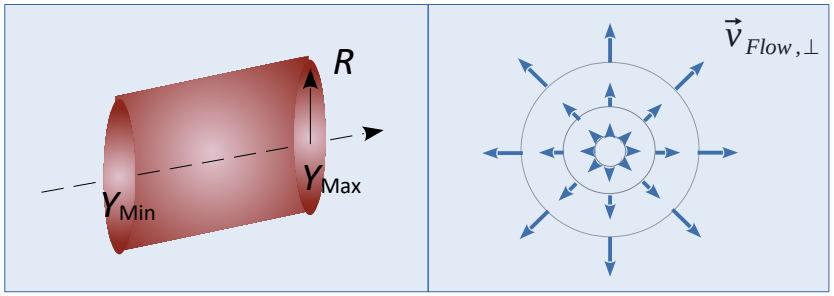
```
[Ranges]
# Rapidity range
# default: 4.0
RapPRange = 4.0

# Spatial rapidity range
# default: 8.0
RapSRange = 8.0

[Model_parameters]
# proper time at freeze-out [fm]
# default: 8.17
Tau = 8.17

# maximum transverse radius [fm]
# default: 8.21
RhoMax = 8.21
```

$$y_{Spatial,Z} = atanh\left(\frac{z}{t}\right)$$




```
# Transverse velocity [c]
# default: 0.341
VelT = 0.341

# Freeze-Out Temperature [MeV]
# default: 165.6
Temperature = 165.6

# Chemical potentials for Barion, Isospin (I_3),
  Strangeness and Charm [MeV]
# default: 28.5, -0.9, 6.9, 0.0
MuB = 28.5
MuI = -0.9
MuS =  6.9
MuC =  0.0

[Subdirectory]
# subdirectory to store events of this model
# default: blastwave/
EventSubDir = blastwave/
```

$$\vec{\beta}_{Flow,\perp}(\rho) = \beta_{Flow,Surf} \cdot \left(\frac{\rho}{R}\right)^2 \hat{e}_\rho$$

$$N_i = g_i \cdot V \cdot \int \frac{d^3p}{(2\pi)^3} \frac{1}{e^{(E_i - \mu_i)/kT} \pm 1}$$

$$\mu_i = b_i \mu_B + q_i \mu_Q + s_i \mu_S$$

- Exemplary contents of **ascii output file**:

```
# THERMINATOR 2 text output
#<EVENT_ENTRY>eid fathereid pid fatherpid rootpid decayed mass e px py pz t x y z</EVENT_ENTRY>
#<EVENT_ID>0x65679F6</EVENT_ID>
#<NO_OF_PARTICLES>9684</NO_OF_PARTICLES>
0 -1 9001 9001 9001 0 2.350000e+00 5.423351e+00 5.105272e-01 8.400604e-01 -4.787891e+00
3.012708e+01 3.797786e+00 1.937673e+00 -2.789553e+01
1 -1 5218 5218 5218 1 2.250000e+00 5.035699e+00 1.177745e+00 -3.007674e-01 -4.337997e+00
2.272015e+01 2.773765e+00 -1.740429e+00 -1.985397e+01
```

- **Structure of the ROOT output file**.

Variables in the "events" TTree :

eventID	(UInt_t)	–	unique ID of the event
entries	(Uint_t)	–	No. of particle entries in this event

Variables in the "particles" TTree characterising a single particle:

mass	(Float_t)	–	mass
t, x, y, z	(Float_t)	–	space-time coordinates of creation (in fm/c)
e,px,py,pz	(Float_t)	–	energy and momentum coordinates (in GeV)
decayed	(Int_t)	–	1 if this particle decayed, 0 if survived.
pid	(Int_t)	–	PDG identification number
fatherpid	(Int_t)	–	parent's PDG number
rootpid	(Int_t)	–	root (primordial) particle's PDG number
eid	(Int_t)	–	particle's sequence number in the event
fathereid	(Int_t)	–	parent's sequence number in the event (-1 = primordial particle)
eventid	(Uint_t)	–	unique id of the event

Therminator2 : overview

- A primordial particle has: `fathereid == -1`

  Note: the particle TTree stores also the decayed particles (`decayed == 1`).

- **Different PIDs**. A particle's entry in tree has 3 pid codes: `pid`, `fatherpid` and `rootpid`.

  A primordial particle has `pid == fatherpid == rootpid`.
  A product of a primordial particle's decay has `fatherpid == rootpid`.
  A further product has 3 different PIDs.

- **Examples of plotting the spectra**:

```
$ root -l event001.root
root[0] particles->Draw ("sqrt(px*px + py*py)", "pid == 2212")        – pT spectrum of protons
root[1] particles->Draw ("0.5*log( (e+pz)/(e-pz) )", "pid == 3122")   – rapidity spectrum of Λ
```

  To draw (on the same plot) the rapidity spectrum of primordial pions and those from decays of other hadrons:

```
root[2] particles->Draw ("0.5*log( (e+pz)/(e-pz) )", "pid == 211 && fathereid>-1")
root[3] particles->Draw ("0.5*log( (e+pz)/(e-pz) )", "pid == 211 && fathereid==-1", "same")
```

- You can merge the collection of tree files using TChain:

```
$ root -l
root[0] TChain t ("particles")
root[1] t.Add ("event000.root") ; t.Add ("event001.root")
root[2] particles->Draw ("mass", "fathereid == -1")
```

  In `macros/` directory you can find examples of full event file joining and plotting different distributions.

- **HOWTO start analysing output in Macro?**

In Therminator2's `include` path you can find the necessary classes.
Among them, the `ParticleCoor.h` header file includes the most important `ParticleCoor` class.
Its members just match the list of variables of each entry of the `particles` tree.

```
#include "/home/kpiasecki/soft/therminator2/build/include/ParticleCoor.h"

int partLoop ()
{
  ParticleCoor P;
  Float_t Pt, Y;

  TFile* fin = new TFile ("events/blastwave/event000.root");
  TTree* tin = (TTree*) fin->Get ("particles");
  tin->SetBranchAddress ("particle", &P );

  TH2F* hpty = new TH2F ("hpty", "", 20, -4., 4., 20, 0., 2.);

  for (int iev = 0; iev < tin->GetEntries() ; iev++) {
    tin->GetEvent (iev);
    if (! (P.pid == 321 && P.decayed == false) ) continue;

    Pt = sqrt (P.px*P.px + P.py*P.py);
    Y = 0.5 * log((P.e + P.pz) / (P.e - P.pz));

    hpty->Fill ( Y, Pt );
  }
  hpty->Draw ("colz");
  return 0;
}
```