GLISSANDO 2

Generated by Doxygen 1.7.6.1

Fri Oct 4 2013 11:34:40

Contents

Chapter 1

Main Page

GLISSANDO 2 - GLauber Initial State Simulation AND mOre... ver. 2.700, 15 - September 2013

Homepage: http://www.ujk.edu.pl/homepages/mryb/GLISSANDO/index.html

arXiv:xx13.xxxx [nucl-th]

Authors:

- Wojciech Broniowski (Wojciech.Broniowski@ifj.edu.pl)
- Maciej Rybczynski (Maciej.Rybczynski@ujk.edu.pl)
- Grzegorz Stefanek (Grzegorz.Stefanek@ujk.edu.pl)
- Piotr Bozek (Piotr.Bozek@ifj.edu.pl)

For the detailed description of ver. 1 program and further references to the description of the model, please, refer to

Computer Physics Communications 180(2009)69, arXiv:0710.5731 [nucl-th] http://arxiv.org.abs/0710.5731

GLISSANDO is a Glauber Monte-Carlo generator for early-stages of relativistic heavy-ion collisions, written in c++ and interfaced to ROOT. Several models are implemented: the wounded-nucleon model, the binary collisions model, the mixed model, and the model with hot-spots. The original geometric distribution of sources (i.e., wounded nucleons or binary collisions) in the transverse plane can be superimposed with a statistical distribution simulating the dispersion in the generated transverse energy in each individual collision. The program generates inter alia the variable-axes (participant) two-dimensional profiles of the density of sources in the transverse plane and their Fourier components. These profiles can be used in further analyses of physical phenomena, such as the the jet quenching, event-by-event hydrodynamics, or analyses of the elliptic flow and its fluctuations. Characteristics of the event (multiplicities, eccentricities, Fourier shape coefficients, etc.) are evaluated and stored in a ROOT file for further off-line studies. A number of scripts is provided for that purpose. The code can also be used for the proton-nucleus and deuteron-nucleus collisions.

Version 2 of GLISSANDO offers much more functionality than version 1, moreover, it is fully object-oriented, providing the user with the flexibility of inspecting and, if needed, modifying the code in a simple manner. New features involve:

- Parametrization of shape of light nuclei, useful in particular for simulations for the NA61 experiment
- Generation of distributions of deformed nuclei according to the deformed Woods-Saxon density with default deformation parameters taken from P.Moller,J.R.-Nix,W.D.Mayers, and W.J.Swiatecki [Nucl.Data Tables 59,185,1995].See also - W.Broniowski,M.Rybczynski,G.Stefanek [Phys.Rev. C87, 044908, 2013;arXiv-:1211.2537]
- The possibility of feeding into the simulations the nuclear distributions accounting for the two-body NN correlations (read from external files, see Alvioli, Drescher and Strikman, [Phys. Lett. B680, 225, 2009], the distributions can be found at http://www.phys.psu.edu/~malvioli/eventgenerator/)
- The use of the Gaussian NN wounding profile (which is more realistic than the commonly-used hard-core wounding profile, see the analysis by Bialas and Bzdak [Acta Phys. Polon. B38, 159,2007])
- The generation of the core-corona distributions (see Bozek [Acta Phys. Polon. B36, 3071,2005] and Werner [Phys. Rev. Lett. 98, 152301, 2007], see also - Becattini and Manninen [Phys. Lett. B673, 19, 2009] and Bozek [Phys. Rev. C79, 054901, 2009])
- The analysis of the triangular shape deformation parameter and profiles, relevant for the triangular flow, see Alver and Roland, [Phys. Rev. C81, 054905, 2010] and Alver, Gombeaud, Luzum,and Ollitrault, [arXiv:1007.5469]
- Generation of rapidity distributions in the wounded-nucleon picture according to the model of Bialas and Czyz [Acta Phys.Polon.B36:905-918,2005], as implemented by Bozek [arXiv:1002.4999]. This allows to obtain the fully 3-dimensional distribution of matter in the early Glauber phase of the collision.

The reference manual for ver. 2, generated by Doxygen, is supplied at the home page. The full write-up of ver. 2 is under preparation.

The code can be freely used and redistributed. However, if you decide to make modifications, the authors would appreciate notification for the record. Any publication or display of results obtained using GLISSANDO must include a reference to our papers

Computer Physics Communications 180(2009)69, arXiv:0710.5731 [nucl-th]

and arXiv:xx13.xxxx [nucl-th]

Chapter 2

Directory Hierarchy

2.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Directory Documentation

6.1 addons/ Directory Reference

Files

- file interpolation.cxx
- file retrieve.cxx

6.2 build/ Directory Reference

Directories

- directory include
- directory src

6.3 build/include/ Directory Reference

Files

- file collision.h
- file counter.h
- file distrib.h
- file functions2.h

6.4 macro/ Directory Reference

Files

- file angles.C
- file centrality.C
- file centrality2.C
- file core_mantle.C
- file corr.C
- file density.C
- file dxdy.C
- file epsilon.C
- file epsilon_b.C
- file epsilon_c.C
- file fitr.C
- file fourier.C
- file hydro.C
- file info.C
- file label.C
- file mult.C
- file overlay.C
- file profile2.C
- file profile2_deformation_63Cu.C
- file profile2_deformation_Au.C
- file profile2_deformation_U.C
- file size.C
- file tilted.C
- file wounding_profile.C

6.5 build/src/ Directory Reference

Files

- file glissando2.cxx

Chapter 7

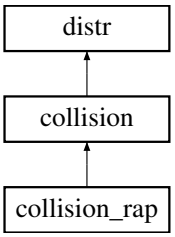
Class Documentation

7.1 collision Class Reference

collision class

`#include <collision.h>`

Inheritance diagram for collision:



Public Member Functions

- collision (Int_t nnA, Int_t nnB)

  constructor
- collision ()

  default constructor
- collision (const collision &w1)

  copying constructor
- collision & operator= (const collision &w1)

  substitution overloading
- void gen_RDS (const nucleus &nA, const nucleus &nB, Float_t d2, Float_t dbin2, Float_t mb)

  destructor

Public Attributes

- Float_t rd2
- Int_t wc
- Int_t * wwA
- Int_t * wwB
- Int_t nwA
- Int_t nwB
- Int_t nwAB
- Int_t nzw
- Int_t nbin
- Int_t nhotspot
- Float_t rpa

7.1.1 Detailed Description

collision class

Class performing the collision of two nuclei. Here the "source" is the position of the wounded nucleon (a nucleon that collided at least once) or the location of the binary collision, taken as the center-of-mass position in the nucleon pair. The weight, called RDS (relative deposited strength) depends on the adopted model. The "charge" is 0 for the binary collision, and equal to i for the wounded nucleons, where i>0 is the number of collisions experienced by the nucleon. Depending on the precompiler option _nnwp_, the wounding and binary collisions occur with a step-function distribution (_nnwp_=0), Gaussian distribution (_nnwp_=1) or gamma approximation (_nnwp_=2). The latter is much more realistic.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 collision::collision (Int_t *nnA*, Int_t *nnB*) [inline]

constructor

Parameters

nnA	mass number of nucleus A
nnB	mass number of nucleus B

7.1.2.2 collision::collision ( ) [inline]

default constructor

The default constructor assumes 208Pb-208Pb collisions.

7.1.2.3 collision::collision (const collision & *w1*) [inline]

copying constructor

7.1.3 Member Function Documentation

7.1.3.1 void collision::gen_RDS (const nucleus & *nA,* const nucleus & *nB,* Float_t *d2,* Float_t *dbin2,* Float_t *mb*) [inline]

destructor

collision between two nuclei

gen_RDS performs the collision of two nuclei, generating the wounded nucleon and the binary collisions, as well as their RDS (relative deposited strength, or weight). It is the core function of the code, implementing the specific model mechanism of the collision.

Parameters

nA	nucleus A, nA>0, nA=1 - proton, nA=2 - deuteron, nA>2 - other nuclei
nB	nucleus B, nB>0, nB=1 - proton, nB=2 - deuteron, nB>2 - other nuclei
d2	wounding distance squared
dbin2	binary-collison distance squared
mb	ratio of the wounding to binary cross sections

7.1.3.2 collision& collision::operator= (const collision & *w1*) [inline]

substitution overloading

7.1.4 Member Data Documentation

7.1.4.1 Int_t collision::nbin

number of binary collisions in the event

7.1.4.2 Int_t collision::nhotspot

number of hot spots

7.1.4.3 Int_t collision::nwA

number of wounded (collided at least once) nucleons in nucleus A

7.1.4.4 Int_t collision::nwAB

total number of wounded nucleons (nwA+nwB) generated in the event

7.1.4.5 Int_t collision::nwB

number of wounded (collided at least once) nucleons in nucleus B

7.1.4.6 Int_t collision::nzw

number of sources with non-zero weight

7.1.4.7 Float_t collision::rd2

square of the distance between the nucleons

7.1.4.8 Float_t collision::rpa

weight of the source (RDS)

7.1.4.9 Int_t collision::wc

counter of the sources

7.1.4.10 Int_t* collision::wwA

wwA[i] is the number of collisions of the i-th nucleon from nucleus A with the nucleons from nucleus B

7.1.4.11 Int_t * collision::wwB

wwB[i] is the number of collisions of the i-th nucleon from nucleus A with the nucleons from nucleus A

The documentation for this class was generated from the following file:

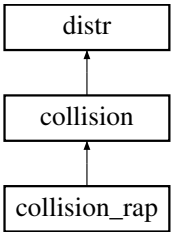
- build/include/collision.h

7.2 collision_rap Class Reference

collision_rap class

```
#include <collision.h>
```

Inheritance diagram for collision_rap:



Public Member Functions

- collision_rap (Int_t nnA, Int_t nnB)
  constructor
- collision_rap ()
  default constructor
- collision_rap (const collision_rap &w1)
  copying constructor
- collision_rap & operator= (const collision_rap &w1)
  substitution overloading
- ∼collision_rap ()
  destructor
- void gen_rap (Int_t part, Float_t maxy)
  generate the rapidity distribution in a specified gap for the y spatial coordinate
- void shift_rap (Float_t rr)
  get the rapidity distribution at shifted rapidity in the Bialas-Czyz-Bozek model

Public Attributes

- TH3D ∗ rap_distr

7.2.1 Detailed Description

collision_rap class

Collision class with an extra feature of generating the rapidity distribution. It can be used to create fully 3-dimensional distribution of sources, overlaying the (spatial) rapidity distribution over the transverse distribution.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 collision_rap::collision_rap (Int_t *nnA*, Int_t *nnB*) [inline]

constructor

Parameters

nnA	mass number of nucleus A
nnB	mass number of nucleus B

7.2.2.2 collision_rap::collision_rap ( ) [inline]

default constructor

7.2.2.3 collision_rap::collision_rap (const collision_rap & *w1*) [inline]

copying constructor

7.2.2.4 collision_rap::~collision_rap ( ) [inline]

destructor

7.2.3 Member Function Documentation

7.2.3.1 void collision_rap::gen_rap (Int_t *part*, Float_t *maxy*) [inline]

generate the rapidity distribution in a specified gap for the y spatial coordinate

Parameters

part	number of particles per unit RDS
maxy	maximum absolute value of the transverse y coordinate for histogramming in the x-y-rapidity histogram

7.2.3.2 collision_rap& collision_rap::operator= (const collision_rap & *w1*) [inline]

substitution overloading

7.2.3.3 void collision_rap::shift_rap (Float_t *rr*) [inline]

get the rapidity distribution at shifted rapidity in the Bialas-Czyz-Bozek model

Parameters

rr	amount of the shift in rapidity

7.2.4 Member Data Documentation

7.2.4.1 TH3D* collision_rap::rap_distr

histogram for storing the rapidity distribution

The documentation for this class was generated from the following file:

- build/include/collision.h

7.3 counter Class Reference

simplest counting class

`#include <counter.h>`

Public Member Functions

- counter ()
  constructor
- ~counter ()
  destructor
- void reset ()
  reset the counter
- void add (Double_t s)
  add entry to the counter
- Int_t getN ()
  get the number of entries
- Double_t get ()
  get the sum of values
- Double_t mean ()
  get the mean value

Private Attributes

- Int_t count
  number of entries
- Double_t value
  sum of values counted

7.3.1 Detailed Description

simplest counting class

7.3.2 Constructor & Destructor Documentation

7.3.2.1 counter::counter ( ) [inline]

constructor

7.3.2.2 counter::~counter ( ) [inline]

destructor

7.3.3 Member Function Documentation

7.3.3.1 void counter::add (Double_t *s*) [inline]

add entry to the counter

Parameters

s	value added

7.3.3.2 Double_t counter::get ( ) [inline]

get the sum of values

7.3.3.3 Int_t counter::getN ( ) [inline]

get the number of entries

7.3.3.4 Double_t counter::mean ( ) [inline]

get the mean value

7.3.3.5 void counter::reset ( ) [inline]

reset the counter

7.3.4 Member Data Documentation

7.3.4.1 Int_t counter::count [private]

number of entries

7.3.4.2 Double_t counter::value [private]

sum of values counted

The documentation for this class was generated from the following file:

- build/include/counter.h

7.4 counter2 Class Reference

counting class with variance

`#include <counter.h>`

Public Member Functions

- counter2 ()
  constructor
- ~counter2 ()
  destructor
- void reset ()
  reset the counter
- void add (Double_t s)
  add entry
- Int_t getN ()
  get the number of entries
- Double_t get ()
  get the sum of values
- Double_t get2 ()
  get the sum of squares of values
- Double_t mean ()
  get the mean value
- Double_t var ()
  get the variance
- Double_t vara ()
  get the variance multiplied with (N-1)/N

Private Attributes

- Int_t count
  number of entries
- Double_t value
  sum of values
- Double_t value2
  sum of squares of values

7.4.1 Detailed Description

counting class with variance

7.4.2 Constructor & Destructor Documentation

7.4.2.1 counter2::counter2 ( ) [inline]

constructor

7.4.2.2 counter2::~counter2 ( ) [inline]

destructor

7.4.3 Member Function Documentation

7.4.3.1 void counter2::add (Double_t *s*) [inline]

add entry

Parameters

s	value added

7.4.3.2 Double_t counter2::get ( ) [inline]

get the sum of values

7.4.3.3 Double_t counter2::get2 ( ) [inline]

get the sum of squares of values

7.4.3.4 Int_t counter2::getN ( ) [inline]

get the number of entries

7.4.3.5 Double_t counter2::mean ( ) [inline]

get the mean value

7.4.3.6 void counter2::reset ( ) [inline]

reset the counter

7.4.3.7 Double_t counter2::var ( ) [inline]

get the variance

7.4.3.8 Double_t counter2::vara ( ) [inline]

get the variance multiplied with (N-1)/N

7.4.4 Member Data Documentation

7.4.4.1 Int_t counter2::count [private]

number of entries

7.4.4.2 Double_t counter2::value [private]

sum of values

7.4.4.3 Double_t counter2::value2 [private]

sum of squares of values

The documentation for this class was generated from the following file:

- build/include/counter.h

7.5 counter_2D Class Reference

2-dimensional counting class

`#include <counter.h>`

Public Member Functions

- counter_2D ()
  constructor
- ~counter_2D ()
  destructor
- void reset ()
  reset the counter
- void add (Double_t sx, Double_t sy)
  add entry
- Int_t getN ()

get the number of entries

- Double_t get_x ()

  get the sum of x values

- Double_t get_y ()

  get the sum of x values

- Double_t get_x2 ()

  get the sum of x2

- Double_t get_y2 ()

  get the sum of y2

- Double_t get_xy ()

  get the sum of xy

- Double_t mean_x ()

  get the mean x value

- Double_t mean_y ()

  get the mean y value

- Double_t var_x ()

  get the variance of x

- Double_t var_y ()

  get the variance of y

- Double_t cov ()

  get the covariance

- Double_t corr ()

  get the correlation

Private Attributes

- Int_t count

  number of entries

- Double_t valuex

  sum of x values

- Double_t valuey

  sum of y values

- Double_t valuex2

  sum of x^2

- Double_t valuey2

  sum of y^2

- Double_t valuexy

  sum of x∗y

7.5.1 Detailed Description

2-dimensional counting class

7.5.2 Constructor & Destructor Documentation

7.5.2.1 counter_2D::counter_2D ( ) [inline]

constructor

7.5.2.2 counter_2D::~counter_2D ( ) [inline]

destructor

7.5.3 Member Function Documentation

7.5.3.1 void counter_2D::add (Double_t *sx*, Double_t *sy*) [inline]

add entry

Parameters

sx	x value added
sy	y value added

7.5.3.2 Double_t counter_2D::corr ( ) [inline]

get the correlation

7.5.3.3 Double_t counter_2D::cov ( ) [inline]

get the covariance

7.5.3.4 Double_t counter_2D::get_x ( ) [inline]

get the sum of x values

7.5.3.5 Double_t counter_2D::get_x2 ( ) [inline]

get the sum of x2

7.5.3.6 Double_t counter_2D::get_xy ( ) [inline]

get the sum of xy

7.5.3.7 Double_t counter_2D::get_y ( ) [inline]

get the sum of x values

7.5.3.8 Double_t counter_2D::get_y2 ( ) [inline]

get the sum of y2

7.5.3.9 Int_t counter_2D::getN ( ) [inline]

get the number of entries

7.5.3.10 Double_t counter_2D::mean_x ( ) [inline]

get the mean x value

7.5.3.11 Double_t counter_2D::mean_y ( ) [inline]

get the mean y value

7.5.3.12 void counter_2D::reset ( ) [inline]

reset the counter

7.5.3.13 Double_t counter_2D::var_x ( ) [inline]

get the variance of x

7.5.3.14 Double_t counter_2D::var_y ( ) [inline]

get the variance of y

7.5.4 Member Data Documentation

7.5.4.1 Int_t counter_2D::count [private]

number of entries

7.5.4.2 Double_t counter_2D::valuex [private]

sum of x values

7.5.4.3 Double_t counter_2D::valuex2 [private]

sum of x^2

7.5.4.4 Double_t counter_2D::valuexy [private]

sum of x*y

7.5.4.5 Double_t counter_2D::valuey [private]

sum of y values

7.5.4.6 Double_t counter_2D::valuey2 [private]

sum of y^2

The documentation for this class was generated from the following file:

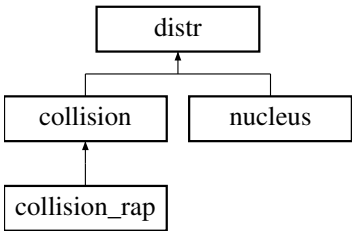
- build/include/counter.h

7.6 distr Class Reference

Distribution of sources in space.

`#include <distrib.h>`

Inheritance diagram for distr:



Public Member Functions

- distr (Int_t k)

  constructor
- distr (void)

  default constructor, 208 sources
- distr (const distr &w1)

  copying constructor

- Float_t sum_w ()

  destructor
- Float_t cmx ()

  generate the center-of-mass x coordinate, no weights
- Float_t cmy ()

  generate the center-of-mass y coordinate, no weights
- Float_t cmz ()

  generate the center-of-mass z coordinate, no weights
- Float_t cmx_w ()

  generate the center-of-mass x coordinate with weights
- Float_t cmy_w ()

  generate the center-of-mass y coordinate with weights
- Float_t cmz_w ()

  generate the center-of-mass z coordinate with weights
- void shift_x (Float_t xt)

  translate in the x direction
- void shift_y (Float_t yt)

  translate in the y direction
- void shift_z (Float_t zt)

  translate in the z direction
- void shift_cmx ()

  translate to the cm reference frame in the x direction, no weights
- void shift_cmy ()

  translate to the cm reference frame in the y direction, no weights
- void shift_cmz ()

  translate to the cm reference frame in the z direction, no weights
- void shift_cmx_w ()

  translate to the cm reference frame in the x direction with weights
- void shift_cmy_w ()

  translate to the cm reference frame in the y direction with weights
- void shift_cmz_w ()

  translate to the cm reference frame in the z direction with weights
- Float_t msx ()

  mean squared x, no weights
- Float_t msy ()

  mean squared y, no weights
- Float_t mxy ()

  mean x∗y, no weights
- Float_t msrad ()

  mean squared radius, no weights
- Float_t msrad_t ()

  mean squared transverse radius, no weights
- Float_t msrad_w ()

mean squared radius with weights

- Float_t msrad_t_w ()

  mean squared transverse radius with weights

- Float_t size ()

  size - the weighted average of the distance from the orgin (in the cm frame)

- Float_t phrot (Int_t m)

  rotation angle maximizing the m-th cosine Fourier moment with weight r^2

- Float_t phrot (Int_t m, Float_t k)

  rotation angle maximizing the m-th cosine Fourier moment with weight r^k

- void rotate (Float_t ph)

  rotate in the transverse plane by the specified angle

- void rotate_polar (Float_t costh)

  rotate in the ZX plane by the theta angle

- void writerds (ofstream &eveout)

  full event output to an external file

- Float_t eps (Int_t m, Int_t imin, Int_t imax)

  m-th cosine Fourier moment in the azimuthal angle in the transverse plane, weight r^2, limited charge range

- Float_t eps (Int_t m)

  m-th cosine Fourier moment in the azimuthal angle in the transverse plane, weight r^2, no limit on charge

- Float_t eps_s (Int_t m, Int_t imin, Int_t imax)

  m-th sine Fourier moment in the azimuthal angle in the transverse plane, weight r^2, limited charge

- Float_t eps_s (Int_t m)

  m-th sine Fourier moment in the azimuthal angle in the transverse plane, weight r^2, no limit on charge

- Float_t eps (Int_t m, Float_t k, Int_t imin, Int_t imax)

  m-th cosine Fourier moment in the azimuthal angle in the transverse plane, weight r^k, limited charge range

- Float_t eps (Int_t m, Float_t k)

  m-th cosine Fourier moment in the azimuthal angle in the transverse plane, weight r^k, no limit on charge

- Float_t eps_s (Int_t m, Float_t k, Int_t imin, Int_t imax)

  m-th sine Fourier moment in the azimuthal angle in the transverse plane, weight r^k, limited charge

- Float_t eps_s (Int_t m, Float_t k)

  m-th sine Fourier moment in the azimuthal angle in the transverse plane, weight r^k, no limit on charge

- void fill_xy (TH2D *xyh, Float_t fac, Int_t imin, Int_t imax)

  fill the histogram of the transverse distribution in cartesian coordinates, limited charge

- void fill_xy (TH2D *xyh, Float_t fac)

  fill the histogram of the transverse distribution in cartesian coordinates, no limit on charge

- void fill_polar (TH2D *polh, Int_t m, Float_t fac, Int_t imin, Int_t imax)

fill the histogram of the transverse distribution in polar coordinates, limited charge

- void fill_polar_s (TH2D *polh, Int_t m, Float_t fac, Int_t imin, Int_t imax)

  fill the histogram of the transverse sine distribution in polar coordinates, limited charge
- void fill_polar (TH2D *polh, Int_t m, Float_t fac)

  fill the histogram of the transverse distribution in polar coordinates, no limit on charge
- void fill_polar_s (TH2D *polh, Int_t m, Float_t fac)

  fill the histogram of the transverse sine distribution in polar coordinates, no limit on charge
- distr & operator= (const distr &w1)

  substitution overloading

Public Attributes

- Int_t n

  number of sources (points)
- Float_t * x

  x space coordinate
- Float_t * y

  y space coordinate
- Float_t * z

  z space coordinate
- Int_t * c

  some integer property, here called "charge"
- Float_t * w

  weight
- Float_t xcm

  center-of-mass x coordinate of the distribution
- Float_t ycm

  center-of-mass y coordinate of the distribution
- Float_t zcm

  center-of-mass z coordinate of the distribution
- Float_t msr

  mean squared radius of the distribution
- Float_t msrt

  mean squared transverse radius of the distribution
- Float_t sumw

  sum of weights of the distribution

7.6.1 Detailed Description

Distribution of sources in space.

Class for storage and basic operations (translation, rotation) of a distribution of "sources" in space. A source is a point with real weight and some additional integer property, e.g., charge.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 distr::distr (Int_t *k*) [inline]

constructor

Parameters

k	number of sources, k>0

7.6.2.2 distr::distr (void) [inline]

default constructor, 208 sources

208 corresponds to the number on nucleons in the 208Pb nucleus

7.6.2.3 distr::distr (const distr & *w1*) [inline]

copying constructor

7.6.3 Member Function Documentation

7.6.3.1 Float_t distr::cmx ( ) [inline]

generate the center-of-mass x coordinate, no weights

7.6.3.2 Float_t distr::cmx_w ( ) [inline]

generate the center-of-mass x coordinate with weights

7.6.3.3 Float_t distr::cmy ( ) [inline]

generate the center-of-mass y coordinate, no weights

7.6.3.4 Float_t distr::cmy_w ( ) [inline]

generate the center-of-mass y coordinate with weights

7.6.3.5 Float_t distr::cmz ( ) [inline]

generate the center-of-mass z coordinate, no weights

7.6.3.6 Float_t distr::cmz_w ( ) [inline]

generate the center-of-mass z coordinate with weights

7.6.3.7 Float_t distr::eps (Int_t *m,* Int_t *imin,* Int_t *imax*) [inline]

m-th cosine Fourier moment in the azimuthal angle in the transverse plane, weight r^2, limited charge range

Limited charge range may be used to get the core and mantle (corona) distributions.

Parameters

m	rank of the Fourier moment, m=0,2,3,4,5,... (m=1 does not make sense in the cm frame)
imin	lowest charge for the sources included
imax	highest charge for the sources included

7.6.3.8 Float_t distr::eps (Int_t *m*) [inline]

m-th cosine Fourier moment in the azimuthal angle in the transverse plane, weight r^2, no limit on charge

Most popular is the second moment, known as eccentricity. The third moment is relevant for the triangular flow.

Parameters

m	rank of the Fourier moment, m=0,2,3,4,5,... (m=1 does not make sense in the cm frame)

7.6.3.9 Float_t distr::eps (Int_t *m,* Float_t *k,* Int_t *imin,* Int_t *imax*) [inline]

m-th cosine Fourier moment in the azimuthal angle in the transverse plane, weight r^k, limited charge range

Limited charge range may be used to get the core and mantle (corona) distributions.

Parameters

m	rank of the Fourier moment, m=0,2,3,4,5,... (m=1 does not make sense in the cm frame)
k	power of transverse radius in the weight
imin	lowest charge for the sources included
imax	highest charge for the sources included

7.6.3.10 Float_t distr::eps (Int_t *m,* Float_t *k*) [inline]

m-th cosine Fourier moment in the azimuthal angle in the transverse plane, weight r^k, no limit on charge

Most popular is the second moment, known as eccentricity. The third moment is relevant for the triangular flow.

Parameters

m	rank of the Fourier moment, m=0,2,3,4,5,... (m=1 does not make sense in the cm frame)
k	power of transverse radius in the weight

7.6.3.11 Float_t distr::eps_s (Int_t *m,* Int_t *imin,* Int_t *imax*) [inline]

m-th sine Fourier moment in the azimuthal angle in the transverse plane, weight r^2, limited charge

Limited charge range may be used to get the core and mantle (corona) distributions.

Parameters

m	rank of the Fourier moment, m=0,2,3,4,5,... (m=1 does not make sense in the cm frame)
imin	lowest charge for the sources included
imax	highest charge for the sources included

7.6.3.12 Float_t distr::eps_s (Int_t *m*) [inline]

m-th sine Fourier moment in the azimuthal angle in the transverse plane, weight r^2, no limit on charge

Parameters

m	rank of the Fourier moment, m=2,3,4,5,... (m=1 does not make sense in the cm frame)

7.6.3.13 Float_t distr::eps_s (Int_t *m,* Float_t *k,* Int_t *imin,* Int_t *imax*) [inline]

m-th sine Fourier moment in the azimuthal angle in the transverse plane, weight r^k, limited charge

Limited charge range may be used to get the core and mantle (corona) distributions.

Parameters

m	rank of the Fourier moment, m=0,2,3,4,5,... (m=1 does not make sense in the cm frame)
k	power of transverse radius in the weight
imin	lowest charge for the sources included
imax	highest charge for the sources included

7.6.3.14 Float_t distr::eps_s (Int_t *m,* Float_t *k*) [inline]

m-th sine Fourier moment in the azimuthal angle in the transverse plane, weight r^k, no limit on charge

Parameters

m	rank of the Fourier moment, m=2,3,4,5,... (m=1 does not make sense in the cm frame)
k	power of transverse radius in the weight

7.6.3.15 void distr::fill_polar (TH2D * *polh,* Int_t *m,* Float_t *fac,* Int_t *imin,* Int_t *imax*) [inline]

fill the histogram of the transverse distribution in polar coordinates, limited charge

Limited charge range may be used to get the core and mantle (corona) distributions.

Parameters

polh	2-dim polar histogram in the transverse plane
m	rank of the Fourier moment
fac	normalization factor
imin	lowest charge for the sources included
imax	highest charge for the sources included

7.6.3.16 void distr::fill_polar (TH2D * *polh,* Int_t *m,* Float_t *fac*) [inline]

fill the histogram of the transverse distribution in polar coordinates, no limit on charge

Parameters

polh	2-dim polar histogram in the transverse plane
m	rank of the Fourier moment
fac	normalization factor

7.6.3.17 void distr::fill_polar_s (TH2D * *polh*, Int_t *m*, Float_t *fac*, Int_t *imin*, Int_t *imax*) [inline]

fill the histogram of the transverse sine distribution in polar coordinates, limited charge

Limited charge range may be used to get the core and mantle (corona) distributions.

Parameters

polh	2-dim polar sine histogram in the transverse plane
m	rank of the sine Fourier moment
fac	normalization factor
imin	lowest charge for the sources included
imax	highest charge for the sources included

7.6.3.18 void distr::fill_polar_s (TH2D * *polh*, Int_t *m*, Float_t *fac*) [inline]

fill the histogram of the transverse sine distribution in polar coordinates, no limit on charge

Parameters

polh	2-dim polar sine histogram in the transverse plane
m	rank of the sine Fourier moment
fac	normalization factor

7.6.3.19 void distr::fill_xy (TH2D * *xyh*, Float_t *fac*, Int_t *imin*, Int_t *imax*) [inline]

fill the histogram of the transverse distribution in cartesian coordinates, limited charge

Limited charge range may be used to get the core and mantle (corona) distributions.

Parameters

xyh	2-dim cartesian histogram in the transverse plane
fac	normalization factor
imin	lowest charge for the sources included
imax	highest charge for the sources included

7.6.3.20 void distr::fill_xy (TH2D * *xyh*, Float_t *fac*) [inline]

fill the histogram of the transverse distribution in cartesian coordinates, no limit on charge

Parameters

xyh	2-dim cartesian histogram in the transverse plane
fac	normalization factor

7.6.3.21 Float_t distr::msrad ( ) [inline]

mean squared radius, no weights

7.6.3.22 Float_t distr::msrad_t ( ) [inline]

mean squared transverse radius, no weights

7.6.3.23 Float_t distr::msrad_t_w ( ) [inline]

mean squared transverse radius with weights

7.6.3.24 Float_t distr::msrad_w ( ) [inline]

mean squared radius with weights

7.6.3.25 Float_t distr::msx ( ) [inline]

mean squared x, no weights

7.6.3.26 Float_t distr::msy ( ) [inline]

mean squared y, no weights

7.6.3.27 Float_t distr::mxy ( ) [inline]

mean x∗y, no weights

7.6.3.28 distr & distr::operator= (const distr & *w1*)

substitution overloading

7.6.3.29 Float_t distr::phrot (Int_t *m*) [inline]

rotation angle maximizing the m-th cosine Fourier moment with weight r^2

for m=2 this is the angle for passing to the "participant" frame

Parameters

m	rank of the Fourier moment, m=0,2,3,4,5,... (m=1 does not make sense in the cm frame)

7.6.3.30 Float_t distr::phrot (Int_t *m*, Float_t *k*) [inline]

rotation angle maximizing the m-th cosine Fourier moment with weight r^k

for m=2 this is the angle for passing to the "participant" frame

Parameters

m	rank of the Fourier moment, m=0,2,3,4,5,... (m=1 does not make sense in the cm frame)
k	power of transverse radius in the weight

7.6.3.31 void distr::rotate (Float_t *ph*) [inline]

rotate in the transverse plane by the specified angle

Parameters

ph	rotation angle in the transverse plane

7.6.3.32 void distr::rotate_polar (Float_t *costh*) [inline]

rotate in the ZX plane by the theta angle

Parameters

costh	cosine of the rotation angle (theta) in the ZX plane

7.6.3.33 void distr::shift_cmx ( ) [inline]

translate to the cm reference frame in the x direction, no weights

7.6.3.34 void distr::shift_cmx_w ( ) [inline]

translate to the cm reference frame in the x direction with weights

7.6.3.35 void distr::shift_cmy ( ) [inline]

translate to the cm reference frame in the y direction, no weights

7.6.3.36 void distr::shift_cmy_w ( ) [inline]

translate to the cm reference frame in the y direction with weights

7.6.3.37 void distr::shift_cmz ( ) [inline]

translate to the cm reference frame in the z direction, no weights

7.6.3.38 void distr::shift_cmz_w ( ) [inline]

translate to the cm reference frame in the z direction with weights

7.6.3.39 void distr::shift_x (Float_t *xt*) [inline]

translate in the x direction

Parameters

xt	displacement in the x direction

7.6.3.40 void distr::shift_y (Float_t *yt*) [inline]

translate in the y direction

Parameters

yt	displacement in the y direction

7.6.3.41 void distr::shift_z (Float_t *zt*) [inline]

translate in the z direction

Parameters

zt	displacement in the z direction

7.6.3.42 Float_t distr::size ( ) [inline]

size - the weighted average of the distance from the orgin (in the cm frame)

7.6.3.43 Float_t distr::sum_w ( ) [inline]

destructor

generate sum of weights

7.6.3.44 void distr::writerds (ofstream & *eveout*) [inline]

full event output to an external file

Parameters

eveout	external file for the full event data

7.6.4 Member Data Documentation

7.6.4.1 Int_t* distr::c

some integer property, here called "charge"

Depending on the situation, c is the electric charge of the nucleon in the nucleus, number of collisions of the nucleon, etc.

7.6.4.2 Float_t distr::msr

mean squared radius of the distribution

7.6.4.3 Float_t distr::msrt

mean squared transverse radius of the distribution

7.6.4.4 Int_t distr::n

number of sources (points)

7.6.4.5 Float_t distr::sumw

sum of weights of the distribution

7.6.4.6 Float_t* distr::w

weight

Depending on the situation, the weight may describe the amount of the deposited energy, entropy, etc., in the source

7.6.4.7 Float_t* distr::x

x space coordinate

7.6.4.8 Float_t distr::xcm

center-of-mass x coordinate of the distribution

7.6.4.9 Float_t* distr::y

y space coordinate

7.6.4.10 Float_t distr::ycm

center-of-mass y coordinate of the distribution

7.6.4.11 Float_t* distr::z

z space coordinate

7.6.4.12 Float_t distr::zcm

center-of-mass z coordinate of the distribution

The documentation for this class was generated from the following file:

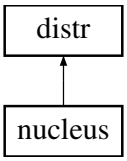
- build/include/distrib.h

7.7 nucleus Class Reference

nucleus class

`#include <distrib.h>`

Inheritance diagram for nucleus:



Public Member Functions

- nucleus (Int_t k)

  constructor
- nucleus (const nucleus &w1)

  copying constructor

- nucleus & operator= (const nucleus &w1)

  substitution overloading
- void set_proton ()

  destructor
- void set_deuteron ()

  set the distribution for the deuteron
- void set_random_A ()

  set randomly the distribution of nucleons in the nucleus A, use the rlosA() function, no correlations
- void set_random_A_hos ()

  set randomly the distribution of nucleons in the nucleus A, use the rlosA_hos() function, no correlations
- void set_random_B ()

  set randomly the distribution of nucleons in the nucleus B, use the rlosB() function, no correlations
- void set_random_B_hos ()

  set randomly the distribution of nucleons in the nucleus B, use the rlosB_hos() function, no correlations
- void set_random_A_def ()

  set randomly the distribution of nucleons in the nucleus A with deformation, no correlations
- void set_random_B_def ()

  set randomly the distribution of nucleons in the nucleus B with deformation, no correlations
- void set_random_A_def (Float_t d)
- void set_random_B_def (Float_t d)
- void set_random_A (Float_t d)

  set randomly the distribution of nucleons in the nucleus A with expulsion, use the rlosA() function
- void set_random_A_hos (Float_t d)

  similar as the above function but for harmonic oscillator shell model, use use the rlosA_hos() function.
- void set_random_B (Float_t d)

  set randomly the distribution of nucleons in the nucleus B with expulsion, use the rlosB() function
- void set_random_B_hos (Float_t d)

  similar as above function but for harmonic oscillator shell model, use use the rlosB_hos() function.
- void set_file (Float_t *px, Float_t *py, Float_t *pz, Int_t sn, Int_t nu)

  set the distribution of nucleons in the nucleus from the tables generated earlier
- void set_file_uncor (Float_t *px, Float_t *py, Float_t *pz, Int_t sn, Int_t nu)

  set the distribution of nucleons in the nucleus from the tables generated earlier, killing correlations
- Float_t dist2 (Int_t j1, Int_t j2)

  distance between two nucleons

- bool good_pair (Int_t j1, Int_t j2, Float_t d)

  the pair of nucleons is "good" when the distance between the nucleons is larger than the expulsion distance d
- bool good_down (Int_t j, Float_t d)

  nucleon j is "good" when the distance to all nucleons of index i with i<j is larger than the expulsion distance d
- bool good_all (Float_t d)

  configuration of nucleons in the nucleus is "good" when the distances between all nucleons are larger than the expulsion distance d

Private Attributes

- bool g
- Float_t cth
- Float_t sth
- Float_t phi
- Float_t r

7.7.1 Detailed Description

nucleus class

Class to store distributions of nucleons in nuclei.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 nucleus::nucleus (Int_t *k*) [inline]

constructor

Parameters

k	number of nucleons in the nucleus (the mass naumber of the nucleus), k>0, k=1 - proton, k=2 - deuteron, k>2 - other nucleus

7.7.2.2 nucleus::nucleus (const nucleus & *w1*) [inline]

copying constructor

7.7.3 Member Function Documentation

7.7.3.1 Float_t nucleus::dist2 (Int_t *j1*, Int_t *j2*) [inline]

distance between two nucleons

Parameters

j1	index of nucleon 1
j2	index of nucleon 2

7.7.3.2 bool nucleus::good_all (Float_t *d*) [inline]

configuration of nucleons in the nucleus is "good" when the distances between all nucleons are larger than the expulsion distance d

Parameters

d	expulsion distance

7.7.3.3 bool nucleus::good_down (Int_t *j*, Float_t *d*) [inline]

nucleon j is "good" when the distance to all nucleons of index i with i<j is larger than the expulsion distance d

Parameters

j	index of the nucleon
d	expulsion distance

7.7.3.4 bool nucleus::good_pair (Int_t *j1*, Int_t *j2*, Float_t *d*) [inline]

the pair of nucleons is "good" when the distance between the nucleons is larger than the expulsion distance d

Parameters

j1	index of nucleon 1
j2	index of nucleon 2
d	expulsion distance - nucleons cannot be closer to each other than d

7.7.3.5 nucleus& nucleus::operator= (const nucleus & *w1*) [inline]

substitution overloading

7.7.3.6 void nucleus::set_deuteron ( ) [inline]

set the distribution for the deuteron

Use the Hulthen distribution.

7.7.3.7 void nucleus::set_file (Float_t * *px*, Float_t * *py*, Float_t * *pz*, Int_t *sn*, Int_t *nu*) [inline]

set the distribution of nucleons in the nucleus from the tables generated earlier

The nucleon position (x,y,z) is taken from the tables read earlier. The pointers x, y, z are originally positioned at px, py, pz at a place corresponding to the beginning of a randomly selected nucleus. The tables with nuclear distributions have to be prepared externally, see, e.g., http://www.phys.psu.edu/~malvioli/eventgenerator/ for distributions involving correlations.

Parameters

px	x coordinate of distributions read from files
py	y coordinate of distributions read from files
pz	z coordinate of distributions read from files
sn	number of entries in the file (should be the mass number time the number of stored nuclei)
nu	mass number of the nucleus

7.7.3.8 void nucleus::set_file_uncor (Float_t * *px*, Float_t * *py*, Float_t * *pz*, Int_t *sn*, Int_t *nu*) [inline]

set the distribution of nucleons in the nucleus from the tables generated earlier, killing correlations

The nucleon position (x,y,z) is taken from the tables read earlier. The pointers x, y, z are positioned at px, py, pz at a place corresponding to a completely randomly selected nucleon. This procedure kills any correlations and is (probably) equivalent to the mixing technique. The tables with nuclear distributions have to be prepared externally.

Parameters

px	x coordinate of distributions read from files
py	y coordinate of distributions read from files
pz	z coordinate of distributions read from files
sn	number of entries in the file (should be the mass number time the number of stored nuclei)
nu	mass number of the nucleus

7.7.3.9 void nucleus::set_proton ( ) [inline]

destructor

set the distribution for the proton

The proton is just placed at the origin

7.7.3.10 void nucleus::set_random_A ( ) [inline]

set randomly the distribution of nucleons in the nucleus A, use the rlosA() function, no correlations

7.7.3.11 void nucleus::set_random_A (Float_t *d*) [inline]

set randomly the distribution of nucleons in the nucleus A with expulsion, use the rlosA() function

The nucleon positions are subsequntly generated acording to the spherically-symmetric Woods-Saxon distribution. If the nucleon happens to be generated closer than the expulsion distance d to any of the previously generated nucleons, it is generated anew, until it is "good". Since this leads to some swelling, the original distribution must be a bit narrower to cancel neutrilize this effect (see our original paper for a detailed discussion). The expulsion simulates in a simple manner the nuclear repulsion and generates the hard-core two-body correlations.

Parameters

d	expulsion distance, nucleons cannot be closer to each other than d

7.7.3.12 void nucleus::set_random_A_def ( ) [inline]

set randomly the distribution of nucleons in the nucleus A with deformation, no correlations

7.7.3.13 void nucleus::set_random_A_def (Float_t *d*) [inline]

Parameters

d	expulsion distance, nucleons cannot be closer to each other than d

7.7.3.14 void nucleus::set_random_A_hos ( ) [inline]

set randomly the distribution of nucleons in the nucleus A, use the rlosA_hos() function, no correlations

7.7.3.15 void nucleus::set_random_A_hos (Float_t *d*) [inline]

similar as the above function but for harmonic oscillator shell model, use use the rlosA_hos() function.

Parameters

d	expulsion distance, nucleons cannot be closer to each other than d

7.7.3.16 void nucleus::set_random_B ( ) [inline]

set randomly the distribution of nucleons in the nucleus B, use the rlosB() function, no correlations

7.7.3.17 void nucleus::set_random_B (Float_t *d*) [inline]

set randomly the distribution of nucleons in the nucleus B with expulsion, use the rlosB() function

Same as set_random_A for the case of the nucleus B, which in general is different from A

Parameters

d	expulsion distance, nucleons cannot be closer to each other than d

7.7.3.18 void nucleus::set_random_B_def ( ) [inline]

set randomly the distribution of nucleons in the nucleus B with deformation, no correlations

7.7.3.19 void nucleus::set_random_B_def (Float_t *d*) [inline]

Parameters

d	expulsion distance, nucleons cannot be closer to each other than d

7.7.3.20 void nucleus::set_random_B_hos ( ) [inline]

set randomly the distribution of nucleons in the nucleus B, use the rlosB_hos() function, no correlations

7.7.3.21 void nucleus::set_random_B_hos (Float_t *d*) [inline]

similar as above function but for harmonic oscillator shell model, use use the rlosB_hos() function.

Parameters

d	expulsion distance, nucleons cannot be closer to each other than d

7.7.4 Member Data Documentation

7.7.4.1 Float_t nucleus::cth [private]

7.7.4.2 bool nucleus::g [private]

7.7.4.3 Float_t nucleus::phi [private]

7.7.4.4 Float_t nucleus::r [private]

7.7.4.5 Float_t nucleus::sth [private]

The documentation for this class was generated from the following file:

- build/include/distrib.h

7.8 SOURCE Struct Reference

structure for output of the full event - transverse coordinates, weight, number of the event

`#include <functions2.h>`

Public Attributes

- Float_t X

  x coordinate
- Float_t Y

  y coordinate
- Float_t W

  z coordinate
- UInt_t KK

  number of the event

7.8.1 Detailed Description

structure for output of the full event - transverse coordinates, weight, number of the event

7.8.2 Member Data Documentation

7.8.2.1 UInt_t SOURCE::KK

number of the event

7.8.2.2 Float_t SOURCE::W

z coordinate

7.8.2.3 Float_t SOURCE::X

x coordinate

7.8.2.4 Float_t SOURCE::Y

y coordinate

The documentation for this struct was generated from the following files:

- build/include/functions2.h
- addons/retrieve.cxx

7.9 tr_his_c Class Reference

class storing the trees and histograms

`#include <functions2.h>`

Public Member Functions

- void init ()
  initialize the histograms
- void fill ()
  fill trees param and phys
- void fill_tr ()
  fill the main tree
- void proj ()
  projects the 2-dim histograms with polar distributions on 1-dim histograms
- void fill_res ()
  calculate eccentricity, size, etc. and their fluctuations vs. number of wounded nucleons or b
- void write ()
  write out trees param, phys, full_event, and the main tree
- void write_r ()
  write out the radial density distribution an the pair distance distribution in the nucleus
- void write_w ()
  write out the overlaid distributions
- void write_wpro ()
  write out the wounding profile

- void write_d ()

  write out the histograms with the 2-dim distributions and the radial distributions of the Fourier components of the source profiles
- void gen ()

  generate histograms of eccentricities and their variance, etc., vs. the number of wounded nucleons or b

Public Attributes

- TTree * param

  parameters
- TTree * phys

  A+B cross section and other physical results.
- TTree * full_event

  full info on the event (positions and RDS of the sources)
- TTree * tree

  basic physical results
- TH2D * xyhist

  cartesian fixed-axes distribution
- TH2D * xyhist_nuclA

  cartesian fixed-axes distribution of density in nucleus A
- TH2D * xyhist_nuclB

  cartesian fixed-axes distribution of density in nucleus B
- TH2D * rcostheta_nuclA

  (r,cos(theta)) distribution of density in nucleus A
- TH2D * rcostheta_nuclB

  (r,cos(theta)) distribution of density in nucleus B
- TH2D * xyhist_mantle

  cartesian fixed-axes mantle distribution
- TH2D * xyhist_core

  cartesian fixed-axes core distribution
- TH2D * xyhistr

  cartesian variable-axes distribution
- TH2D * c0hist

  polar fixed-axes distribution of cos(phi)
- TH2D * c2hist

  polar fixed-axes distribution of cos(2 phi)
- TH2D * c3hist

  polar fixed-axes distribution of cos(3 phi)
- TH2D * c4hist

  polar fixed-axes distribution of cos(4 phi)
- TH2D * c5hist

  polar fixed-axes distribution of cos(5 phi)

- TH2D * c6hist

  polar fixed-axes distribution of cos(6 phi)
- TH2D * c0rhist

  polar variable-axes distribution of cos(phi)
- TH2D * c2rhist

  polar variable-axes distribution of cos(2 phi)
- TH2D * c3rhist

  polar variable-axes distribution of cos(3 phi)
- TH2D * c4rhist

  polar variable-axes distribution of cos(4 phi)
- TH2D * c5rhist

  polar variable-axes distribution of cos(5 phi)
- TH2D * c6rhist

  polar variable-axes distribution of cos(6 phi)
- TH2D * s3hist

  polar fixed-axes distribution of sin(3 phi)
- TH2D * s3rhist

  polar variable-axes distribution of sin(3 phi)
- TH1D * nx

  center-of-mass x coordinate of the source distribution vs. Nw
- TH1D * nx2

  square of cm x coordinate, then its variance, vs. Nw
- TH1D * ny

  center-of-mass y coordinate of the source distribution vs. Nw
- TH1D * ny2

  square of cm y coordinate, then its variance, vs. Nw
- TH1D * nsize

  size vs. Nw
- TH1D * nsize2

  square of size, then its variance/size^2, vs. Nw
- TH1D * neps

  fixed-axes eccentricity vs. Nw
- TH1D * neps2

  square of fixed-axes eccentricity, then its variance, vs. Nw
- TH1D * neps4

  fixed-axes fourth moment vs. Nw
- TH1D * nepsp

  variable-axes eccentricity vs. Nw
- TH1D * nepsp2

  square of variable-axes eccentricity, then its variance, vs. Nw
- TH1D * nepsp22

  fourth power of variable-axes eccentricity vs. Nw
- TH1D * nepsp4

variable-axes fourth moment vs. Nw

- TH1D * nuni

  frequency of Nw, i.e. histogram of unity vs. Nw
- TH1D * nepsb

  fixed-axes eccentricity vs. b
- TH1D * neps2b

  square of fixed-axes eccentricity, then its variance, vs. b
- TH1D * nepspb

  variable-axes eccentricity vs. b
- TH1D * nepsp2b

  square of variable-axes eccentricity, then its variance, vs. b
- TH1D * nunib

  frequency of b, i.e. histogram of unity vs. b
- TH1D * nwb

  number of wounded nucleons in nucleus B vs. total number of wounded nucleons
- TH1D * nw2b

  square of the number of wounded nucleons in nucleus B, then its variance, vs. total number of wounded nucleons
- TH1D * nwei

  RDS vs. number of wounded nucleons in nucleus A.
- TH1D * nwei2

  square of RDS, then its variance, vs. number of wounded nucleons in nucleus A
- TH1D * ntarg

  number of wounded nucleons in nucles B vs. number of wounded nucleons in nucleus A
- TH1D * ntarg2

  square of the number of wounded nucleons in nucles B, then its variance, vs. number of wounded nucleons in nucleus A
- TH1D * nbinar

  number of binary collisions vs. number of wounded nucleons in nucleus A
- TH1D * nbinar2

  square of the number of binary collisions, then its variance, vs. number of wounded nucleons in nucleus A
- TH1D * nunp

  frequency of the number of wounded nucleons in nucleus A
- TH1D * radA

  one-body radial distribution in the nucleus A
- TH1D * radB

  one-body radial distribution in the nucleus B
- TH1D * rrelA

  distance between the pair of nucleons in the nucleus A
- TH1D * rrelB

  distance between the pair of nucleons in the nucleus B
- TH1D * rrel_u

uncorrelated distance between the pair of nucleons in the nucleus (one nucleon from A, the other one from B)

- TH1D * weih

  the distribution overlaid on the wounded nucleons
- TH1D * weih_bin

  the distribution overlaid over binary collisions
- TH1D * wpro

  the wounding profile
- TH1D * c0hp

  fixed-axes radial profile f_0
- TH1D * c2hp

  fixed-axes radial profile f_2
- TH1D * c3hp

  fixed-axes radial profile f_3
- TH1D * s3hp

  fixed-axes radial profile for the sine moment, g_3
- TH1D * c4hp

  fixed-axes radial profile f_4
- TH1D * c5hp

  fixed-axes radial profile f_5
- TH1D * c6hp

  fixed-axes radial profile f_6
- TH1D * c0rhp

  variable-axes radial profile f_0
- TH1D * c2rhp

  variable-axes radial profile f_2
- TH1D * s3rhp

  variable-axes radial profile f_3
- TH1D * c3rhp

  variable-axes radial profile for the sine moment, g_3
- TH1D * c4rhp

  variable-axes radial profile f_4
- TH1D * c5rhp

  variable-axes radial profile f_5
- TH1D * c6rhp

  variable-axes radial profile f_6

7.9.1 Detailed Description

class storing the trees and histograms

Class for storage of ROOT structures (trees, histograms) used for later off-line analysis within ROOT or other codes

7.9.2 Member Function Documentation

7.9.2.1 void tr_his_c::fill ( ) [inline]

fill trees param and phys

7.9.2.2 void tr_his_c::fill_res ( ) [inline]

calculate eccentricity, size, etc. and their fluctuations vs. number of wounded nucleons or b

7.9.2.3 void tr_his_c::fill_tr ( ) [inline]

fill the main tree

7.9.2.4 void tr_his_c::gen ( ) [inline]

generate histograms of eccentricities and their variance, etc., vs. the number of wounded nucleons or b

7.9.2.5 void tr_his_c::init ( ) [inline]

initialize the histograms

< tree storing parameters

< tree storing some physical quantities

< tree storing full info on the events

< tree storing basic information on events

7.9.2.6 void tr_his_c::proj ( ) [inline]

projects the 2-dim histograms with polar distributions on 1-dim histograms

7.9.2.7 void tr_his_c::write ( ) [inline]

write out trees param, phys, full_event, and the main tree

7.9.2.8 void tr_his_c::write_d ( ) [inline]

write out the histograms with the 2-dim distributions and the radial distributions of the Fourier components of the source profiles

7.9.2.9 void tr_his_c::write_r ( ) [inline]

write out the radial density distribution an the pair distance distribution in the nucleus

7.9.2.10 void tr_his_c::write_w ( ) [inline]

write out the overlaid distributions

7.9.2.11 void tr_his_c::write_wpro ( ) [inline]

write out the wounding profile

7.9.3 Member Data Documentation

7.9.3.1 TH2D * tr_his_c::c0hist

polar fixed-axes distribution of cos(phi)

7.9.3.2 TH1D* tr_his_c::c0hp

fixed-axes radial profile f_0

7.9.3.3 TH2D * tr_his_c::c0rhist

polar variable-axes distribution of cos(phi)

7.9.3.4 TH1D * tr_his_c::c0rhp

variable-axes radial profile f_0

7.9.3.5 TH2D * tr_his_c::c2hist

polar fixed-axes distribution of cos(2 phi)

7.9.3.6 TH1D * tr_his_c::c2hp

fixed-axes radial profile f_2

7.9.3.7 TH2D * tr_his_c::c2rhist

polar variable-axes distribution of cos(2 phi)

7.9.3.8 TH1D * tr_his_c::c2rhp

variable-axes radial profile f_2

7.9.3.9 TH2D * tr_his_c::c3hist

polar fixed-axes distribution of cos(3 phi)

7.9.3.10 TH1D * tr_his_c::c3hp

fixed-axes radial profile f_3

7.9.3.11 TH2D * tr_his_c::c3rhist

polar variable-axes distribution of cos(3 phi)

7.9.3.12 TH1D * tr_his_c::c3rhp

variable-axes radial profile for the sine moment, g_3

7.9.3.13 TH2D * tr_his_c::c4hist

polar fixed-axes distribution of cos(4 phi)

7.9.3.14 TH1D * tr_his_c::c4hp

fixed-axes radial profile f_4

7.9.3.15 TH2D * tr_his_c::c4rhist

polar variable-axes distribution of cos(4 phi)

7.9.3.16 TH1D * tr_his_c::c4rhp

variable-axes radial profile f_4

7.9.3.17 TH2D * tr_his_c::c5hist

polar fixed-axes distribution of cos(5 phi)

7.9.3.18 TH1D * tr_his_c::c5hp

fixed-axes radial profile f_5

7.9.3.19 TH2D * tr_his_c::c5rhist

polar variable-axes distribution of cos(5 phi)

7.9.3.20 TH1D * tr_his_c::c5rhp

variable-axes radial profile f_5

7.9.3.21 TH2D * tr_his_c::c6hist

polar fixed-axes distribution of cos(6 phi)

7.9.3.22 TH1D * tr_his_c::c6hp

fixed-axes radial profile f_6

7.9.3.23 TH2D * tr_his_c::c6rhist

polar variable-axes distribution of cos(6 phi)

7.9.3.24 TH1D * tr_his_c::c6rhp

variable-axes radial profile f_6

7.9.3.25 TTree * tr_his_c::full_event

full info on the event (positions and RDS of the sources)

7.9.3.26 TH1D * tr_his_c::nbinar

number of binary collisions vs. number of wounded nucleons in nucleus A

7.9.3.27 TH1D * tr_his_c::nbinar2

square of the number of binary collisions, then its variance, vs. number of wounded nucleons in nucleus A

7.9.3.28 TH1D * tr_his_c::neps

fixed-axes eccentricity vs. Nw

7.9.3.29 TH1D * tr_his_c::neps2

square of fixed-axes eccentricity, then its variance, vs. Nw

7.9.3.30 TH1D * tr_his_c::neps2b

square of fixed-axes eccentricity, then its variance, vs. b

7.9.3.31 TH1D * tr_his_c::neps4

fixed-axes fourth moment vs. Nw

7.9.3.32 TH1D * tr_his_c::nepsb

fixed-axes eccentricity vs. b

7.9.3.33 TH1D * tr_his_c::nepsp

variable-axes eccentricity vs. Nw

7.9.3.34 TH1D * tr_his_c::nepsp2

square of variable-axes eccentricity, then its variance, vs. Nw

7.9.3.35 TH1D * tr_his_c::nepsp22

fourth power of variable-axes eccentricity vs. Nw

7.9.3.36 TH1D * tr_his_c::nepsp2b

square of variable-axes eccentricity, then its variance, vs. b

7.9.3.37 TH1D * tr_his_c::nepsp4

variable-axes fourth moment vs. Nw

7.9.3.38 TH1D * tr_his_c::nepspb

variable-axes eccentricity vs. b

7.9.3.39 TH1D * tr_his_c::nsize

size vs. Nw

7.9.3.40 TH1D * tr_his_c::nsize2

square of size, then its variance/size^2, vs. Nw

7.9.3.41 TH1D * tr_his_c::ntarg

number of wounded nucleons in nucles B vs. number of wounded nucleons in nucleus A

7.9.3.42 TH1D * tr_his_c::ntarg2

square of the number of wounded nucleons in nucles B, then its variance, vs. number of wounded nucleons in nucleus A

7.9.3.43 TH1D * tr_his_c::nuni

frequency of Nw, i.e. histogram of unity vs. Nw

7.9.3.44 TH1D * tr_his_c::nunib

frequency of b, i.e. histogram of unity vs. b

7.9.3.45 TH1D * tr_his_c::nunp

frequency of the number of wounded nucleons in nucleus A

7.9.3.46 TH1D * tr_his_c::nw2b

square of the number of wounded nucleons in nucleus B, then its variance, vs. total number of wounded nucleons

7.9.3.47 TH1D * tr_his_c::nwb

number of wounded nucleons in nucleus B vs. total number of wounded nucleons

7.9.3.48 TH1D* tr_his_c::nwei

RDS vs. number of wounded nucleons in nucleus A.

7.9.3.49 TH1D * tr_his_c::nwei2

square of RDS, then its variance, vs. number of wounded nucleons in nucleus A

7.9.3.50 TH1D* tr_his_c::nx

center-of-mass x coordinate of the source distribution vs. Nw

7.9.3.51 TH1D * tr_his_c::nx2

square of cm x coordinate, then its variance, vs. Nw

7.9.3.52 TH1D * tr_his_c::ny

center-of-mass y coordinate of the source distribution vs. Nw

7.9.3.53 TH1D * tr_his_c::ny2

square of cm y coordinate, then its variance, vs. Nw

7.9.3.54 TTree* tr_his_c::param

parameters

7.9.3.55 TTree * tr_his_c::phys

A+B cross section and other physical results.

7.9.3.56 TH1D* tr_his_c::radA

one-body radial distribution in the nucleus A

7.9.3.57 TH1D * tr_his_c::radB

one-body radial distribution in the nucleus B

7.9.3.58 TH2D * tr_his_c::rcostheta_nuclA

(r,cos(theta)) distribution of density in nucleus A

7.9.3.59 TH2D * tr_his_c::rcostheta_nuclB

(r,cos(theta)) distribution of density in nucleus B

7.9.3.60 TH1D * tr_his_c::rrel_u

uncorrelated distance between the pair of nucleons in the nucleus (one nucleon from A, the other one from B)

7.9.3.61 TH1D * tr_his_c::rrelA

distance between the pair of nucleons in the nucleus A

7.9.3.62 TH1D * tr_his_c::rrelB

distance between the pair of nucleons in the nucleus B

7.9.3.63 TH2D * tr_his_c::s3hist

polar fixed-axes distribution of sin(3 phi)

7.9.3.64 TH1D * tr_his_c::s3hp

fixed-axes radial profile for the sine moment, g_3

7.9.3.65 TH2D * tr_his_c::s3rhist

polar variable-axes distribution of sin(3 phi)

7.9.3.66 TH1D * tr_his_c::s3rhp

variable-axes radial profile f_3

7.9.3.67 TTree * tr_his_c::tree

basic physical results

7.9.3.68 TH1D * tr_his_c::weih

the distribution overlaid on the wounded nucleons

7.9.3.69 TH1D * tr_his_c::weih_bin

the distribution overlaid over binary collisions

7.9.3.70 TH1D * tr_his_c::wpro

the wounding profile

7.9.3.71 TH2D* tr_his_c::xyhist

cartesian fixed-axes distribution

7.9.3.72 TH2D * tr_his_c::xyhist_core

cartesian fixed-axes core distribution

7.9.3.73 TH2D * tr_his_c::xyhist_mantle

cartesian fixed-axes mantle distribution

7.9.3.74 TH2D * tr_his_c::xyhist_nuclA

cartesian fixed-axes distribution of density in nucleus A

7.9.3.75 TH2D * tr_his_c::xyhist_nuclB

cartesian fixed-axes distribution of density in nucleus B

7.9.3.76 TH2D * tr_his_c::xyhistr

cartesian variable-axes distribution

The documentation for this class was generated from the following file:

- build/include/functions2.h

Chapter 8

File Documentation

8.1 addons/interpolation.cxx File Reference

```
#include <math.h> #include <cstring> #include <iostream> ×
#include <fstream> #include <cstdlib> #include <sstream> ×
#include <TH1F.h>  #include <TH1D.h>  #include <TH2F.h> ×
#include <TFile.h> #include <TTree.h> #include <TChain.-
h>
```

Functions

- Double_t w1D (Int_t n, Double_t dx, Double_t x[4], Double_t y[4])

  1D Lagrange interpolation, calculation of value of the Lagrange polynomial f(x) at given x
- Double_t w2D (Double_t x, Double_t y, Double_t x0, Double_t y0, Double_t u0, Double_t x1, Double_t y1, Double_t u1, Double_t x2, Double_t y2, Double_t u2)

  2D Lagrange interpolation, calculation of value of Lagrange polynomial (first order)
- Double_t lin (Double_t dx, Double_t x1, Double_t y1, Double_t x2, Double_t y2)

  Two nodes. Linear regression method. Calculation of f(x)=a+x∗b.
- Double_t inter1D (Double_t dx, TFile &f, char ∗histname)

  1D interpolation, determination of the node values from the given histogram.
- Double_t inter2D (Double_t dx, Double_t dy, TFile &f, char ∗histname)

  2D interpolation, determination of the node values from the given histogram.
- void d1 (TFile &f)

  1D interpolation
- void d2 (TFile &f)

  2D interpolation
- void start ()

  Start menu.
- Double_t mean_weight (TString &inpfile)

Determination of value of mean weight from the npa branch stored in GLISSANDO Root file.

- Double_t integral1D (Double_t nbc, TFile &f, char ∗histname)

  Calculation the normalization integral from the given 1-dim histogram stored in the GLISSANDO Root file. Rectangle method.

- Double_t integral2D (Double_t nbc, TFile &f, char ∗histname)

  Calculation of value of integral: f(rho_x,rho_y) drho_x drho_y from the given histogram stored in the GLISSANDO Root file. Rectangle method.

- Int_t main (Int_t argc, char ∗∗argv)

Variables

- char hname [100]

  name of the histogram from the GLISSANDO output ROOT file

8.1.1 Detailed Description

auxilliary file, part of GLISSANDO 2

8.1.2 Function Documentation

8.1.2.1 void d1 (TFile & *f*)

1D interpolation

8.1.2.2 void d2 (TFile & *f*)

2D interpolation

8.1.2.3 Double_t integral1D (Double_t *nbc,* TFile & *f,* char ∗ *histname*)

Calculation the normalization integral from the given 1-dim histogram stored in the GLISSANDO Root file. Rectangle method.

8.1.2.4 Double_t integral2D (Double_t *nbc,* TFile & *f,* char ∗ *histname*)

Calculation of value of integral: f(rho_x,rho_y) drho_x drho_y from the given histogram stored in the GLISSANDO Root file. Rectangle method.

8.1.2.5 Double_t inter1D (Double_t *dx,* TFile & *f,* char ∗ *histname*)

1D interpolation, determination of the node values from the given histogram.

8.1.2.6 Double_t inter2D (Double_t *dx,* Double_t *dy,* TFile & *f,* char * *histname*)

2D interpolation, determination of the node values from the given histogram.

8.1.2.7 Double_t lin (Double_t *dx,* Double_t *x1,* Double_t *y1,* Double_t *x2,* Double_t *y2*)

Two nodes. Linear regression method. Calculation of f(x)=a+x*b.

8.1.2.8 Int_t main (Int_t *argc,* char ** *argv*)

The code computes the values of the one-dimensional or two-dimansional profiles at a specified point via Lagrange interpolation. Useful, if exporting of these data for other applications is needed.

Parameters

argc	number of command line parameters
argv	name of the GLISSANDO output ROOT file

8.1.2.9 Double_t mean_weight (TString & *inpfile*)

Determination of value of mean weight from the npa branch stored in GLISSANDO Root file.

8.1.2.10 void start ( )

Start menu.

8.1.2.11 Double_t w1D (Int_t *n,* Double_t *dx,* Double_t *x[4],* Double_t *y[4]*)

1D Lagrange interpolation, calculation of value of the Lagrange polynomial f(x) at given x

8.1.2.12 Double_t w2D (Double_t *x,* Double_t *y,* Double_t *x0,* Double_t *y0,* Double_t *u0,* Double_t *x1,* Double_t *y1,* Double_t *u1,* Double_t *x2,* Double_t *y2,* Double_t *u2*)

2D Lagrange interpolation, calculation of value of Lagrange polynomial (first order)

8.1.3 Variable Documentation

8.1.3.1 char hname[100]

name of the histogram from the GLISSANDO output ROOT file

8.2 addons/retrieve.cxx File Reference

```
#include <math.h>  #include <TH1D.h>  #include <TFile.h>
#include <TTree.h> #include <TChain.h> #include <fstream> ×
#include <iostream> #include <sstream>
```

Classes

- struct SOURCE

  structure for output of the full event - transverse coordinates, weight, number of the event

Functions

- Int_t main (Int_t argc, char **argv)

8.2.1 Detailed Description

auxilliary file, part of GLISSANDO 2

8.2.2 Function Documentation

8.2.2.1 Int_t main (Int_t *argc*, char ** *argv*)

The code serves as an example of using the full spatial distribution of sources, which optionally may be generated in the simulation.

Parameters

argc	number of command line parameters
argv	name of the GLISSANDO output ROOT file

8.3 build/include/collision.h File Reference

```
#include "distrib.h" #include <TMath.h>
```

Classes

- class collision

  collision class

- class collision_rap

  collision_rap class

8.3.1 Detailed Description

Part of GLISSANDO 2

8.4 build/include/counter.h File Reference

Classes

- class counter

  simplest counting class
- class counter2

  counting class with variance
- class counter_2D

  2-dimensional counting class

8.4.1 Detailed Description

Part of GLISSANDO

8.5 build/include/distrib.h File Reference

```
#include <TH1D.h> #include <TH3D.h> #include "counter.h"
```

Classes

- class distr

  Distribution of sources in space.
- class nucleus

  nucleus class

8.5.1 Detailed Description

Part of GLISSANDO 2

8.6 build/include/functions2.h File Reference

```
#include <math.h> #include <time.h> #include <string.h>
#include <iostream> #include <fstream> #include <cstdlib> ×
#include <TH1D.h> #include <TH2D.h> #include <TFile.-
h> #include <TTree.h> #include <TRandom3.h> #include
"counter.h"
```

Classes

- struct SOURCE

  structure for output of the full event - transverse coordinates, weight, number of the event
- class tr_his_c

  class storing the trees and histograms

Functions

- void readpar (TString inpfile)

  process the input file with parameters
- void echopar ()

  echo parameters to the output
- void reset_counters ()

  reset the counters used to store physical quantities in the event
- void helper (Int_t argc, char *str)

  print the version or brief help
- void header ()

  print the header in the output
- void epilog ()

  print epilog to the output
- Int_t time_start ()

  start the time measurement
- void time_stop (Int_t ts)

  stop the time measurement and print stamp
- Float_t los ()

  random number generator using the built-in ROOT generator, uniform on (0,1)
- Float_t rlosA ()

  random number generator for the Woods-Saxon (or Fermi) distribution - nucleus A
- Float_t rlosB ()

  random number generator for the Woods-Saxon (or Fermi) distribution - nucleus B
- Float_t rlosA_def (Float_t *cth_pointerA, Float_t beta2, Float_t beta4)
- Float_t rlosB_def (Float_t *cth_pointerB, Float_t beta2, Float_t beta4)
- Float_t rlosA_hos ()

  random number generator for the harmonic oscillator shell model density - nucleus A
- Float_t rlosB_hos ()

  random number generator for the harmonic oscillator shell model density - nucleus B
- Float_t rlos_hult ()

  random number generator for the Hulthen distribution
- Float_t fg (Float_t rr)

  rapidity distribution with the plateau
- Float_t fpm (Float_t rr)

function f +/- from Bozek, arXiv:1002.4999v2 [nucl-th]

- Float_t los_rap_A ()

  random number generator for the rapidity distribution - wounded nucleons from nucleus A

- Float_t los_rap_B ()

  random number generator for the rapidity distribution - wounded nucleons from nucleus B

- Float_t los_rap_bin ()

  random number generator for the rapidity distribution - binary collisions

- Float_t gamgen (Float_t a)

  random number generator for the Gamma distribution

- Int_t negbin (Float_t m, Float_t v)
- Float_t dist (Int_t m, Float_t u, Float_t v)
- Float_t disp (Float_t w)

  random shift of the source location

Variables

- Int_t EVENTS = 50000

  number of generated events

- Int_t NBIN = 40

  number of bins for histogramming in x, y, and r

- Int_t FBIN = 72

  number of bins for histogramming in the azimuthal angle

- Int_t NUMA = 208

  mass number of nucleus A

- Int_t NUMB = 208

  mass number of nucleus B

- Int_t WMIN = 2

  minimum number of wounded nucleons to record the event

- Int_t MODEL = 0

  switch for the superimposed multiplicity distribution: 0 - uniform, 1 - Poisson, 2 - - Gamma

- Int_t DOBIN = 0

  1 - count binary collisions even in the pure wounded-nucleon model. 0 - do not

- Int_t W0 = 2

  minimum allowed number of wounded nucleons in the acceptance window

- Int_t W1 = 1000

  maximum allowed number of wounded nucleons in the acceptance window

- Int_t SHIFT = 1

  1 - shift the coordinates of the fireball to c.m. in the fixed-axes case (preferred), 0 - do not

- Int_t RET = 0

0 - fix-last algorithm (preferred), 1 - return-to-beginning algorithm for the generation of the nuclear distribution

- Int_t FULL = 0

  1 - generate the full event tree (large output file), 0 - do not

- Int_t FILES = 0

  1 - read distribution from files, 0 - do not

- Int_t NNWP = 0

  0 - hard-sphere NN wounding profile, 1 - Gaussian NN wounding profile, 2 - gamma NN wounding profile

- Int_t NUMRAP = 10

  number of particles per unit weight generated in the whole rapidity range

- Int_t ARANK = 2

  rank of the Fourier moment for the forward-backward analysis

- Int_t PP = -1

  power of the transverse radius in the Fourier moments

- Int_t RO = 0

  rank of the rotation axes (0 - rotation rank = rank of the Fourier moment)

- UInt_t ISEED

  read seed for the ROOT random number generator, if 0 - random seed generated

- UInt_t ISEED1

  copy of ISEED

- Float_t BMIN = 0.

  minimum value of the impact parameter in the acceptance window

- Float_t BMAX = 25.

  maximum value of the impact parameter in the acceptance window

- Float_t RDS0 = 0.

  minimum value of the relative deposited strength (RDS) in the acceptance window

- Float_t RDS1 = 100000

  maximum value of the relative deposited strength (RDS) in the acceptance window

- Float_t RWSA = 6.407

  Woods-Saxon radius for nucleus A.

- Float_t AWSA = 0.459

  Woods-Saxon width for nucleus A.

- Float_t BETA2A = 0.0

  Deformation parameter beta2 of deformed Woods-Saxon distribution for nucleus A.

- Float_t BETA4A = 0.0

  Deformation parameter beta4 of deformed Woods-Saxon distribution for nucleus A.

- Float_t ROTA_THETA = -1.0

  Parameter controlling the rotation of nucleus A in XZ plane (polar angle THETA, -1 means random rotation)

- Float_t ROTA_PHI = -1.0

  Parameter controlling the rotation of nucleus A in XY plane (azimuthal angle PHI, -1 means random rotation)

- Float_t RWSB = 6.407

Woods-Saxon radius for nucleus B.

- Float_t AWSB = 0.459

  Woods-Saxon width for nucleus B.

- Float_t BETA2B = 0.0

  Deformation parameter beta2 of deformed Woods-Saxon distribution for nucleus B.

- Float_t BETA4B = 0.0

  Deformation parameter beta4 of deformed Woods-Saxon distribution for nucleus B.

- Float_t ROTB_THETA = -1.0

  Parameter controlling the rotation of nucleus B in XZ plane (polar angle THETA, -1 means random rotation)

- Float_t ROTB_PHI = -1.0

  Parameter controlling the rotation of nucleus B in XY plane (azimuthal angle PHI, -1 means random rotation)

- Float_t BTOT = fmax(RWSA,RWSB)+AWSA+AWSB

  maximum coordinate value for some histograms

- Float_t WFA = 0.

  the w parameter for the Fermi distribution for nucleus A

- Float_t WFB = 0.

  the w parameter for the Fermi distribution for nucleus B

- Float_t SNN = 73.5

  NN "wounding" cross section in milibarns.

- Float_t SBIN = 73.5

  NN binary cross section in milibarns.

- Float_t ALPHA = 0.15

  the mixing parameter: 0 - wounded, 1 - binary, 0.145 - mixed (PHOBOS)

- Float_t Uw = 2.

  Poisson or Gamma parameters for superimposed distribution, wounded nucleons.

- Float_t Ubin = 2.

  Poisson or Gamma parameters for superimposed distribution, binary collisions.

- Float_t Vw = 4.

  Negative binomial variance, wounded nucleons.

- Float_t Vbin = 4.

  Negative binomial variance, binary collisions.

- Float_t PI = 4.∗atan(1.)

  the number pi

- Float_t CD = 0.9

  closest allowed distance (expulsion distance) between nucleons in the nucleus in fm (simulation of repulsion)

- Float_t DW = 0.

  dispersion of the location of the source for wounded nucleons (in fm)

- Float_t DBIN = 0.

  dispersion of the location of the source for binary collisions (in fm)

- Float_t GA = 0.92

  Gaussian wounding profile parameter (height at the origin)

- Float_t RAPRANGE = 5.

  range in rapidity
- Float_t ETA0 = 1.

  2∗ETA0 is the width of the plateau in eta
- Float_t ETAM = 3.36

  parameter of the Bialas-Czyz-Bozek model
- Float_t SIGETA = 1.3

  parameter controlling the width of the rapidity distribution
- Float_t MAXYRAP = 10.

  maximum absolute value of the y coordinate in the x-y-rapidity histogram
- Float_t FBRAP = 2.5

  forward rapidity for the forward-backward analysis (backward rapidity = - FBRAP)
- Float_t RCHA = 5.66

  harmonic oscillator shell model density mean squared charge radii of 12C-nucleus
- Float_t RCHB = 5.66

  harmonic oscillator shell model density mean squared charge radii of 12C-nucleus
- Float_t RCHP = 0.7714

  harmonic oscillator shell model density mean squared charge radii of proton
- Float_t OMEGA = 0.4

  relative variance of cross-section fluctuations
- Float_t GAMA = 1.0
- counter2 estd

  counter for epsilon standard (fixed-axes), <r^2 cos(2 phi)>/<r^2>
- counter2 epart1

  counter for epsilon participant (variable-axes), <r^3 cos(phi)>/<r^3>
- counter2 epart

  counter for epsilon participant (variable-axes), <r^n cos(2 phi)>/<r^n>
- counter2 estd3

  counter for fixed-axes <r^n cos(3 phi)>/<r^n>
- counter2 epart3

  counter for variable-axes <r^n cos(3 phi)>/<r^n>
- counter2 estd4

  counter for fixed-axes <r^n cos(4 phi)>/<r^n>
- counter2 epart4

  counter for variable-axes <r^n cos(4 phi)>/<r^n>
- counter2 estd5

  counter for fixed-axes <r^n cos(5 phi)>/<r^n>
- counter2 epart5

  counter for variable-axes <r^n cos(5 phi)>/<r^n>
- counter2 estd6

  counter for fixed-axes <r^n cos(6 phi)>/<r^n>
- counter2 epart6

  counter for variable-axes <r^n cos(6 phi)>/<r^n>

- counter2 nwounded

  counter for number of wounded nucleons
- counter2 nbinary

  counter for number of binary collisions
- counter2 nhot

  counter for number of hot-spots
- counter2 nweight

  counter for relative deposited strength (RDS)
- counter_2D angles

  counter for forward-backward reaction-plane angle correlations
- Int_t evall

  number of all attempted event
- Int_t roo

  Fourier rank for the rotation axis.
- Int_t ppp

  power of the weight in eccentricty definition
- Int_t kk

  number of the current event
- Float_t d

  the wounding distance
- Float_t dbin

  the binary-collision distance
- Float_t b

  impact parameter
- Float_t sitot

  the total A+B cross section in the acceptance window
- Float_t sirad

  equivalent hard-sphere radius for the cross section
- Float_t rwA

  number of wounded nucleons in A
- Float_t rwB

  number of wounded nucleons in B
- Float_t rwAB

  number of all wounded nucleons
- Float_t rbin

  number of binary collisions
- Float_t rhotspot

  number of hot-spots
- Float_t rpa

  relative deposited strength (RDS)
- Float_t sizeav

  size
- Float_t es

epsilon standard (fixed-axes), <r^2 cos(2 phi)>/<r^2>

- Float_t ess

  fixed-axes sine moment, <r^2 sin(2 phi)>/<r^2>

- Float_t ep1

  epsilon participant (variable-axes), <r^3 cos(phi)>/<r^3>

- Float_t ep1s

  variable-axes sine moment, <r^3 sin(phi)>/<r^3>

- Float_t ep

  epsilon participant (variable-axes), <r^n cos(2 phi)>/<r^n>

- Float_t eps

  variable-axes sine moment, <r^n sin(2 phi)>/<r^n>

- Float_t es3

  fixed-axes <r^n cos(3 phi)>/<r^n>

- Float_t es3s

  fixed-axes sine moment, <r^n sin(3 phi)>/<r^n>

- Float_t ep3

  variable-axes <r^2 cos(3 phi)>/<r^2>

- Float_t ep3s

  variable-axes sine moment, <r^n sin(3 phi)>/<r^n>

- Float_t es4

  fixed-axes <r^n cos(4 phi)>/<r^n>

- Float_t es4s

  fixed-axes sine moment, <r^n sin(4 phi)>/<r^n>

- Float_t ep4

  variable-axes <r^n cos(4 phi)>/<r^n>

- Float_t ep4s

  variable-axes sine moment, <r^n sin(4 phi)>/<r^n>

- Float_t es5

  fixed-axes <r^n cos(5 phi)>/<r^n>

- Float_t es5s

  fixed-axes sine moment, <r^n sin(5 phi)>/<r^n>

- Float_t ep5

  variable-axes <r^n cos(5 phi)>/<r^n>

- Float_t ep5s

  variable-axes sine moment, <r^n sin(5 phi)>/<r^n>

- Float_t es6

  fixed-axes <r^n cos(6 phi)>/<r^n>

- Float_t es6s

  fixed-axes sine moment, <r^n sin(6 phi)>/<r^n>

- Float_t ep6

  variable-axes <r^n cos(6 phi)>/<r^n>

- Float_t ep6s

  variable-axes sine moment, <r^n sin(6 phi)>/<r^n>

- Float_t phirot

  rotation angle maximizing the second Fourier moment
- Float_t phirot_plus

  rotation angle maximizing the second Fourier moment - increased rapidity
- Float_t phirot_minus

  rotation angle maximizing the second Fourier moment - decreased rapidity
- Float_t phirot3

  rotation angle maximizing the third Fourier moment
- Float_t phirot4

  rotation angle maximizing the fourth Fourier moment
- Float_t phirot5

  rotation angle maximizing the fifth Fourier moment
- Float_t phirot6

  rotation angle maximizing the sixth Fourier moment
- Float_t xx

  center-of-mass x coordinate
- Float_t yy

  center-of-mass y coordinate
- Float_t xeps

  average es
- Float_t xseps

  standard deviation of es
- Float_t xepp

  average ep
- Float_t xsepp

  standard deviation of ep

8.6.1 Detailed Description

Part of GLISSANDO 2

8.6.2 Function Documentation

8.6.2.1 Float_t disp (Float_t *w*)

random shift of the source location

The location of the source may by shifted randomly when DW>0 od DBIN>0, with the Gaussian distribution of the width w.

Parameters

w	average shift of the source location, Gaussian distribution

8.6.2.2 Float_t dist (Int_t *m,* Float_t *u,* Float_t *v*)

Parameters

m	case: 0 - uniform, 1 - Poisson, 2 - Gamma distribution
u	average of the distribution in cases 1 and 2 and 3
v	variance of the distribution in case 3, v>u

8.6.2.3 void echopar ( )

echo parameters to the output

8.6.2.4 void epilog ( )

print epilog to the output

8.6.2.5 Float_t fg (Float_t *rr*)

rapidity distribution with the plateau

Parameters

rr	spatial rapidity

8.6.2.6 Float_t fpm (Float_t *rr*)

function f +/- from Bozek, arXiv:1002.4999v2 [nucl-th]

Parameters

rr	spatial rapidity

8.6.2.7 Float_t gamgen (Float_t *a*)

random number generator for the Gamma distribution

Parameters

a	the parameter a in f(x) = x^(a-1) exp(-x)/Gamma(a)

8.6.2.8 void header ( )

print the header in the output

8.6.2.9 void helper (Int_t *argc,* char * *str*)

print the version or brief help

Parameters

argc	number of command line parameters
str	string parameter (-v for version, -h for brief help)

8.6.2.10 Float_t los ( )

random number generator using the built-in ROOT generator, uniform on (0,1)

8.6.2.11 Float_t los_rap_A ( )

random number generator for the rapidity distribution - wounded nucleons from nucleus A

Formula for the wounded quark rapidity profile from Bozek, arXiv:1002.4999v2 [nucl-th].

8.6.2.12 Float_t los_rap_B ( )

random number generator for the rapidity distribution - wounded nucleons from nucleus B

Formula for the wounded quark rapidity profile from Bozek, arXiv:1002.4999v2 [nucl-th].

8.6.2.13 Float_t los_rap_bin ( )

random number generator for the rapidity distribution - binary collisions

Formula for the wounded quark rapidity profile from Bozek, arXiv:1002.4999v2 [nucl-th].

8.6.2.14 Int_t negbin (Float_t *m,* Float_t *v*)

8.6.2.15 void readpar (TString *inpfile*)

process the input file with parameters

scan the input file for the parameters reset from the default values

correct wrong input

see the paper for the discussion of parametrizatiations of the nuclear distributions

Parameters

inpfile	name of the input file

8.6.2.16 void reset_counters ( )

reset the counters used to store physical quantities in the event

8.6.2.17 Float_t rlos_hult ( )

random number generator for the Hulthen distribution

The Hulthen distribution used to generate the distance between nucleons in the deuteron

8.6.2.18 Float_t rlosA ( )

random number generator for the Woods-Saxon (or Fermi) distribution - nucleus A

8.6.2.19 Float_t rlosA_def (Float_t * *cth_pointerA,* Float_t *beta2,* Float_t *beta4*)

random number generator for the Woods-Saxon (deformed with beta2,beta4 parameters of deformation and spherical harmonics Y20, Y40 (or Fermi) distribution - nucleus A

8.6.2.20 Float_t rlosA_hos ( )

random number generator for the harmonic oscillator shell model density - nucleus A

The harmonic oscillator shell distribution used to generate the distance between nucleons in light (2<NUMA<17) nuclei (Nuclear Sizes. L. R. B. Elton, Oxford University Press, New York, 1961.)

8.6.2.21 Float_t rlosB ( )

random number generator for the Woods-Saxon (or Fermi) distribution - nucleus B

8.6.2.22 Float_t rlosB_def (Float_t * *cth_pointerB,* Float_t *beta2,* Float_t *beta4*)

random number generator for the Woods-Saxon (deformed with beta2,beta4 parameters of deformation and spherical harmonics Y20, Y40 (or Fermi) distribution - nucleus B

8.6.2.23 Float_t rIosB_hos ( )

random number generator for the harmonic oscillator shell model density - nucleus B

The harmonic oscillator shell distribution used to generate the distance between nucleons in light (2<NUMB<17) nuclei (Nuclear Sizes. L. R. B. Elton, Oxford University Press, New York, 1961.)

8.6.2.24 Int_t time_start ( )

start the time measurement

8.6.2.25 void time_stop (Int_t *ts*)

stop the time measurement and print stamp

Parameters

ts	time at start

8.6.3 Variable Documentation

8.6.3.1 Float_t ALPHA = 0.15

the mixing parameter: 0 - wounded, 1 - binary, 0.145 - mixed (PHOBOS)

8.6.3.2 counter_2D angles

counter for forward-backward reaction-plane angle correlations

8.6.3.3 Int_t ARANK = 2

rank of the Fourier moment for the forward-backward analysis

8.6.3.4 Float_t AWSA = 0.459

Woods-Saxon width for nucleus A.

8.6.3.5 Float_t AWSB = 0.459

Woods-Saxon width for nucleus B.

8.6.3.6 Float_t b

impact parameter

8.6.3.7 Float_t BETA2A = 0.0

Deformation parameter beta2 of deformed Woods-Saxon distribution for nucleus A.

8.6.3.8 Float_t BETA2B = 0.0

Deformation parameter beta2 of deformed Woods-Saxon distribution for nucleus B.

8.6.3.9 Float_t BETA4A = 0.0

Deformation parameter beta4 of deformed Woods-Saxon distribution for nucleus A.

8.6.3.10 Float_t BETA4B = 0.0

Deformation parameter beta4 of deformed Woods-Saxon distribution for nucleus B.

8.6.3.11 Float_t BMAX = 25.

maximum value of the impact parameter in the acceptance window

8.6.3.12 Float_t BMIN = 0.

minimum value of the impact parameter in the acceptance window

8.6.3.13 Float_t BTOT = fmax(RWSA,RWSB)+AWSA+AWSB

maximum coordinate value for some histograms

8.6.3.14 Float_t CD = 0.9

closest allowed distance (expulsion distance) between nucleons in the nucleus in fm (simulation of repulsion)

8.6.3.15 Float_t d

the wounding distance

8.6.3.16 Float_t DBIN = 0.

dispersion of the location of the source for binary collisions (in fm)

8.6.3.17 Float_t dbin

the binary-collision distance

8.6.3.18 Int_t DOBIN = 0

1 - count binary collisions even in the pure wounded-nucleon model. 0 - do not

8.6.3.19 Float_t DW = 0.

dispersion of the location of the source for wounded nucleons (in fm)

8.6.3.20 Float_t ep

epsilon participant (variable-axes), <r^n cos(2 phi)>/<r^n>

8.6.3.21 Float_t ep1

epsilon participant (variable-axes), <r^3 cos(phi)>/<r^3>

8.6.3.22 Float_t ep1s

variable-axes sine moment, <r^3 sin(phi)>/<r^3>

8.6.3.23 Float_t ep3

variable-axes <r^2 cos(3 phi)>/<r^2>

8.6.3.24 Float_t ep3s

variable-axes sine moment, <r^n sin(3 phi)>/<r^n>

8.6.3.25 Float_t ep4

variable-axes <r^n cos(4 phi)>/<r^n>

8.6.3.26 Float_t ep4s

variable-axes sine moment, <r^n sin(4 phi)>/<r^n>

8.6.3.27 Float_t ep5

variable-axes <r^n cos(5 phi)>/<r^n>

8.6.3.28 Float_t ep5s

variable-axes sine moment, <r^n sin(5 phi)>/<r^n>

8.6.3.29 Float_t ep6

variable-axes <r^n cos(6 phi)>/<r^n>

8.6.3.30 Float_t ep6s

variable-axes sine moment, <r^n sin(6 phi)>/<r^n>

8.6.3.31 counter2 epart

counter for epsilon participant (variable-axes), <r^n cos(2 phi)>/<r^n>

8.6.3.32 counter2 epart1

counter for epsilon participant (variable-axes), <r^3 cos(phi)>/<r^3>

8.6.3.33 counter2 epart3

counter for variable-axes <r^n cos(3 phi)>/<r^n>

8.6.3.34 counter2 epart4

counter for variable-axes <r^n cos(4 phi)>/<r^n>

8.6.3.35 counter2 epart5

counter for variable-axes <r^n cos(5 phi)>/<r^n>

8.6.3.36 counter2 epart6

counter for variable-axes <r^n cos(6 phi)>/<r^n>

8.6.3.37 Float_t eps

variable-axes sine moment, <r^n sin(2 phi)>/<r^n>

8.6.3.38 Float_t es

epsilon standard (fixed-axes), <r^2 cos(2 phi)>/<r^2>

8.6.3.39 Float_t es3

fixed-axes <r^n cos(3 phi)>/<r^n>

8.6.3.40 Float_t es3s

fixed-axes sine moment, <r^n sin(3 phi)>/<r^n>

8.6.3.41 Float_t es4

fixed-axes <r^n cos(4 phi)>/<r^n>

8.6.3.42 Float_t es4s

fixed-axes sine moment, <r^n sin(4 phi)>/<r^n>

8.6.3.43 Float_t es5

fixed-axes <r^n cos(5 phi)>/<r^n>

8.6.3.44 Float_t es5s

fixed-axes sine moment, <r^n sin(5 phi)>/<r^n>

8.6.3.45 Float_t es6

fixed-axes <r^n cos(6 phi)>/<r^n>

8.6.3.46 Float_t es6s

fixed-axes sine moment, <r^n sin(6 phi)>/<r^n>

8.6.3.47 Float_t ess

fixed-axes sine moment, <r^2 sin(2 phi)>/<r^2>

8.6.3.48 counter2 estd

counter for epsilon standard (fixed-axes), <r^2 cos(2 phi)>/<r^2>

8.6.3.49 counter2 estd3

counter for fixed-axes <r^n cos(3 phi)>/<r^n>

8.6.3.50 counter2 estd4

counter for fixed-axes <r^n cos(4 phi)>/<r^n>

8.6.3.51 counter2 estd5

counter for fixed-axes <r^n cos(5 phi)>/<r^n>

8.6.3.52 counter2 estd6

counter for fixed-axes <r^n cos(6 phi)>/<r^n>

8.6.3.53 Float_t ETA0 = 1.

2∗ETA0 is the width of the plateau in eta

8.6.3.54 Float_t ETAM = 3.36

parameter of the Bialas-Czyz-Bozek model

8.6.3.55 Int_t evall

number of all attempted event

8.6.3.56 Int_t EVENTS = 50000

number of generated events

8.6.3.57 Int_t FBIN = 72

number of bins for histogramming in the azimuthal angle

8.6.3.58 Float_t FBRAP = 2.5

forward rapidity for the forward-backward analysis (backward rapidity = - FBRAP)

8.6.3.59 Int_t FILES = 0

1 - read distribution from files, 0 - do not

8.6.3.60 Int_t FULL = 0

1 - generate the full event tree (large output file), 0 - do not

8.6.3.61 Float_t GA = 0.92

Gaussian wounding profile parameter (height at the origin)

8.6.3.62 Float_t GAMA = 1.0

gamma wounding profile parameter (height at the origin)

8.6.3.63 UInt_t ISEED

read seed for the ROOT random number generator, if 0 - random seed generated

8.6.3.64 UInt_t ISEED1

copy of ISEED

8.6.3.65 Int_t kk

number of the current event

8.6.3.66 Float_t MAXYRAP = 10.

maximum absolute value of the y coordinate in the x-y-rapidity histogram

8.6.3.67 Int_t MODEL = 0

switch for the superimposed multiplicity distribution: 0 - uniform, 1 - Poisson, 2 - Gamma

8.6.3.68 Int_t NBIN = 40

number of bins for histogramming in x, y, and r

8.6.3.69 counter2 nbinary

counter for number of binary collisions

8.6.3.70 counter2 nhot

counter for number of hot-spots

8.6.3.71 Int_t NNWP = 0

0 - hard-sphere NN wounding profile, 1 - Gaussian NN wounding profile, 2 - gamma NN wounding profile

8.6.3.72 Int_t NUMA = 208

mass number of nucleus A

8.6.3.73 Int_t NUMB = 208

mass number of nucleus B

8.6.3.74 Int_t NUMRAP = 10

number of particles per unit weight generated in the whole rapidity range

8.6.3.75 counter2 nweight

counter for relative deposited strength (RDS)

8.6.3.76 counter2 nwounded

counter for number of wounded nucleons

8.6.3.77 Float_t OMEGA = 0.4

relative variance of cross-section fluctuations

8.6.3.78 Float_t phirot

rotation angle maximizing the second Fourier moment

8.6.3.79 Float_t phirot3

rotation angle maximizing the third Fourier moment

8.6.3.80 Float_t phirot4

rotation angle maximizing the fourth Fourier moment

8.6.3.81 Float_t phirot5

rotation angle maximizing the fifth Fourier moment

8.6.3.82 Float_t phirot6

rotation angle maximizing the sixth Fourier moment

8.6.3.83 Float_t phirot_minus

rotation angle maximizing the second Fourier moment - decreased rapidity

8.6.3.84 Float_t phirot_plus

rotation angle maximizing the second Fourier moment - increased rapidity

8.6.3.85 Float_t PI = 4.∗atan(1.)

the number pi

8.6.3.86 Int_t PP = -1

power of the transverse radius in the Fourier moments

8.6.3.87 Int_t ppp

power of the weight in eccentricty definition

8.6.3.88 Float_t RAPRANGE = 5.

range in rapidity

8.6.3.89 Float_t rbin

number of binary collisions

8.6.3.90 Float_t RCHA = 5.66

harmonic oscillator shell model density mean squared charge radii of 12C-nucleus

8.6.3.91 Float_t RCHB = 5.66

harmonic oscillator shell model density mean squared charge radii of 12C-nucleus

8.6.3.92 Float_t RCHP = 0.7714

harmonic oscillator shell model density mean squared charge radii of proton

8.6.3.93 Float_t RDS0 = 0.

minimum value of the relative deposited strength (RDS) in the acceptance window

8.6.3.94 Float_t RDS1 = 100000

maximum value of the relative deposited strength (RDS) in the acceptance window

8.6.3.95 Int_t RET = 0

0 - fix-last algorithm (preferred), 1 - return-to-beginning algorithm for the generation of the nuclear distribution

8.6.3.96 Float_t rhotspot

number of hot-spots

8.6.3.97 Int_t RO = 0

rank of the rotation axes (0 - rotation rank = rank of the Fourier moment)

8.6.3.98 Int_t roo

Fourier rank for the rotation axis.

8.6.3.99 Float_t ROTA_PHI = -1.0

Parameter controlling the rotation of nucleus A in XY plane (azimuthal angle PHI, -1 means random rotation)

8.6.3.100 Float_t ROTA_THETA = -1.0

Parameter controlling the rotation of nucleus A in XZ plane (polar angle THETA, -1 means random rotation)

8.6.3.101 Float_t ROTB_PHI = -1.0

Parameter controlling the rotation of nucleus B in XY plane (azimuthal angle PHI, -1 means random rotation)

8.6.3.102 Float_t ROTB_THETA = -1.0

Parameter controlling the rotation of nucleus B in XZ plane (polar angle THETA, -1 means random rotation)

8.6.3.103 Float_t rpa

relative deposited strength (RDS)

8.6.3.104 Float_t rwA

number of wounded nucleons in A

8.6.3.105 Float_t rwAB

number of all wounded nucleons

8.6.3.106 Float_t rwB

number of wounded nucleons in B

8.6.3.107 Float_t RWSA = 6.407

Woods-Saxon radius for nucleus A.

8.6.3.108 Float_t RWSB = 6.407

Woods-Saxon radius for nucleus B.

8.6.3.109 Float_t SBIN = 73.5

NN binary cross section in milibarns.

8.6.3.110 Int_t SHIFT = 1

1 - shift the coordinates of the fireball to c.m. in the fixed-axes case (preferred), 0 - do not

8.6.3.111 Float_t SIGETA = 1.3

parameter controlling the width of the rapidity distribution

8.6.3.112 Float_t sirad

equivalent hard-sphere radius for the cross section

8.6.3.113 Float_t sitot

the total A+B cross section in the acceptance window

8.6.3.114 Float_t sizeav

size

8.6.3.115 Float_t SNN = 73.5

NN "wounding" cross section in milibarns.

8.6.3.116 Float_t Ubin = 2.

Poisson or Gamma parameters for superimposed distribution, binary collisions.

8.6.3.117 Float_t Uw = 2.

Poisson or Gamma parameters for superimposed distribution, wounded nucleons.

8.6.3.118 Float_t Vbin = 4.

Negative binomial variance, binary collisions.

8.6.3.119 Float_t Vw = 4.

Negative binomial variance, wounded nucleons.

8.6.3.120 Int_t W0 = 2

minimum allowed number of wounded nucleons in the acceptance window

8.6.3.121 Int_t W1 = 1000

maximum allowed number of wounded nucleons in the acceptance window

8.6.3.122 Float_t WFA = 0.

the w parameter for the Fermi distribution for nucleus A

8.6.3.123 Float_t WFB = 0.

the w parameter for the Fermi distribution for nucleus B

8.6.3.124 Int_t WMIN = 2

minimum number of wounded nucleons to record the event

8.6.3.125 Float_t xepp

average ep

8.6.3.126 Float_t xeps

average es

8.6.3.127 Float_t xsepp

standard deviation of ep

8.6.3.128 Float_t xseps

standard deviation of es

8.6.3.129 Float_t xx

center-of-mass x coordinate

8.6.3.130 Float_t yy

center-of-mass y coordinate

8.7 build/src/glissando2.cxx File Reference

```
#include <math.h>  #include <time.h>  #include <string.-
h>  #include <iostream>  #include <fstream>  #include <T-
H1D.h>  #include <TH2D.h>  #include <TH3D.h>  #include <T-
File.h> #include <TTree.h> #include <TRandom3.h> #include
"counter.h"  #include "functions2.h"  #include "distrib.h"
#include "collision.h"
```

Defines

- #define _VER_ 2.700

Functions

- Int_t main (Int_t argc, char *argv[])
  the main function of GLISSANDO 2

Variables

- Float_t ver = _VER_
  current version of the code

- TRandom3 raa

  ROOT random number generator.

8.7.1 Detailed Description

The main file of GLISSANDO 2

8.7.2 Define Documentation

8.7.2.1 #define _VER_ 2.700

8.7.3 Function Documentation

8.7.3.1 Int_t main (Int_t *argc,* char * *argv[]*)

the main function of GLISSANDO 2

The main function of GLISSANDO 2 contains the basic structure of the Glauber Monte Carlo simulation, i.e., declarations and definitions of basic objects of the nucleus and collision classes, the main loop over events, evaluation of basic quantities, etc. It is meant to be taylored by the user to meet his needs. For speed of the execution, some switches of the code are controlled by the preprocessor variables (_nnwp_, _files_, _profile_, _weight_, _rapidity_). (the units for all dimensionful quantities in GLISSANDO 2 are powers of fm)

start time

print basic info

print header

set the input file

process input parameters

set the ROOT output file

seed the ROOT random-number generator

reset counters used for some basic physical quantities

declare and initialize trees and histograms for storage of data

set the minimum wounding and binary-collision distances (hard-sphere profile) or the Gaussian wounding parameters (Gaussian profile)

echo basic parameters to the console

#if(_files_) then initialize the nucleon distributions from external tables

declare nuclei A nad B

declare the collision

--- start the main loop over events

generate the distributions of nucleons in nuclei A and B

shift nuclei to the center-of-mass frame

rotate deformed nuclei by theta (zx plane) and phi (xy plane) angles The proper order of rotations is important. Opposite one (phi,theta) has no sense.

#(if_profile_) generate the histograms of one-body density in x-y and r-cos(theta) coordinate systems for nucleus A, and for the relative NN distance

generate the impact parameter b with the distribution proportional to b^2 in the range (BMAX, BMIN)

shift the coordinates of the nucleons in nucleus A such that the center of mass is at the point (b∗NUMB/(NUMA+NUMB),0)

shift the coordinates of the nucleons in nucleus B such that the center of mass is at the point (-b∗NUMA/(NUMA+NUMB),0)

collide the nuclei, create the sources (wounded nucleons, binary collisions) and RDS

#if(_rapidity_) generate the rapidity distribution

#if(_weight_) generate the histograms for the NN collision profiles

generate various 2-dim histograms with the distributions of sources

generate the fixed-axes Fourier moments (obsolete) weight r^2

generate the variable-axes Fourier moments (up to 6-th moment)

get some basic properties of the event

fill the data in trees and histograms

if(FULL) write the full event info to the file (added for comparability reasons, takes a lot of space)

--- end of main loop over events

output of results

the total cross section and the equivalent hard-sphere radius

project out the marginal distribution in the radial variable (generate the radial Fourier profiles)

generate and write some histograms with physical quantities

#if(_weight_) normalize to the wounding and the binary cross sections and write

#if(_rapidity_) write rapidity distribution

closing ROOT file

write exit info

stop time and print stamp

Parameters

argc	number of command line parameters
argv	used for passing input and output file names first argument: <input.dat> second argument: <output.root> third argument: <nucl_A.dat> (only when _files_=1) fourth argument: <nucl_B.dat> (only when _files_=1)

8.7.4 Variable Documentation

8.7.4.1 TRandom3 raa

ROOT random number generator.

8.7.4.2 Float_t ver = _VER_

current version of the code

8.8 macro/angles.C File Reference

```
#include "label.C"
```

Functions

- void angles (char *p)

  produces plots of the correlation between principal axes in the forward and backward rapidities

8.8.1 Detailed Description

Script generating the plot of the correlation between principal axes in the forward and backward rapidities (part of GLISSANDO 2)

8.8.2 Function Documentation

8.8.2.1 void angles (char * *p*)

produces plots of the correlation between principal axes in the forward and backward rapidities

Produce plots of the correlation between principal axes in the forward and backward rapidities.

Parameters

p	name of the ROOT input file

8.9 macro/centrality.C File Reference

`#include "label.C" #include <iostream> #include <fstream>` ×

Functions

- void centrality (char *p)

  generate the centrality classes

8.9.1 Detailed Description

Script generating the centrality classes (alternative to centrality2.C) (part of GLISSANDO 2)

8.9.2 Function Documentation

8.9.2.1 void centrality (char * *p*)

generate the centrality classes

Centrality classes are generated in the total number of wounded nucleons, relative deposited strenth (RDS), and the impact parameter. Plots of centrality vs. these variables are generated. The script makes sense for the minimum-bias simulations.

Parameters

p	name of the ROOT input file

8.10 macro/centrality2.C File Reference

`#include "label.C"`

Functions

- void centrality2 (char *p)

  generate the centrality classes

8.10.1 Detailed Description

Script generating the centrality classes (alternative to centrality.C) (part of GLISSANDO 2)

8.10.2 Function Documentation

8.10.2.1 void centrality2 (char * p)

generate the centrality classes

Centrality classes are generated in the total number of wounded nucleons, relative deposited strenth (RDS), and the impact parameter. Plots of distributions divided into classes are produced. The script makes sense for the minimum-bias simulations.

Parameters

p	name of the ROOT input file

8.11 macro/core_mantle.C File Reference

```
#include "label.C"
```

Functions

- void core_mantle (char *p)

  generate the core and corona distributions

8.11.1 Detailed Description

Script generating the core-corona densities (part of GLISSANDO 2)

8.11.2 Function Documentation

8.11.2.1 void core_mantle (char * p)

generate the core and corona distributions

Parameters

p	name of the ROOT input file

8.12 macro/corr.C File Reference

```
#include "label.C"
```

Functions

- void corr (char *p)

  generating the plot of the radial NN correlation function, C(r), for nucleus A

8.12.1 Detailed Description

Script generating the NN correlation plot for nucleus A (part of GLISSANDO 2)

8.12.2 Function Documentation

8.12.2.1 void corr (char * *p*)

generating the plot of the radial NN correlation function, C(r), for nucleus A

C(r) is obtained via division of the correlated and uncorrelated distributions of the relative distance in the nucleon pairs. Requires _profile_=1.

Parameters

p	name of the ROOT input file

8.13 macro/density.C File Reference

```
#include "label.C"
```

Functions

- void density (char *p)

  produce plots of the fixed-axes and variable-axes densities

8.13.1 Detailed Description

Script generating the fixed- and variable-axes densities (part of GLISSANDO 2)

8.13.2 Function Documentation

8.13.2.1 void density (char * p)

produce plots of the fixed-axes and variable-axes densities

Produces plots of the fixed-axes and variable-axes (participant) densities

Parameters

p	name of the ROOT input file

8.14 macro/dxdy.C File Reference

```
#include "label.C"
```

Functions

- void dxdy (char *p)

  produce the plot of the dispersion of the center-of-mass x and y coordinates as a function of the number of wounded nucleons.

8.14.1 Detailed Description

Script generating the dispersion of the center-of-mass x and y coordinates as a function of the number of wounded nucleons (part of GLISSANDO 2)

8.14.2 Function Documentation

8.14.2.1 void dxdy (char * p)

produce the plot of the dispersion of the center-of-mass x and y coordinates as a function of the number of wounded nucleons.

Produces the plot of the standard deviation of the x and y center-of-mass coordinates as a function of the number of wounded nucleons.

Parameters

p	name of the ROOT input file

8.15 macro/epsilon.C File Reference

```
#include "label.C"
```

Functions

- void epsilon (char *p)

  produces plots of the mean and the scaled standard deviation of the eccentricities as functions of the number of wounded nucleons

8.15.1 Detailed Description

Script generating the plots of eccentricities and their scaled standard deviations as functions of the number of wounded nucleons (part of GLISSANDO 2)

8.15.2 Function Documentation

8.15.2.1 void epsilon (char * p)

produces plots of the mean and the scaled standard deviation of the eccentricities as functions of the number of wounded nucleons

Produce plots of the mean and the scaled standard deviation of the eccentricities as functions of the number of wounded nucleons

Parameters

p	name of the ROOT input file

8.16 macro/epsilon_b.C File Reference

```
#include "label.C"
```

Functions

- void epsilon_b (char *p)

  produce plots of the mean and the scaled standard deviation of the eccentricities as functions of the impact parameter b

8.16.1 Detailed Description

Script generating the plots of eccentricities and their scaled standard deviations as functions of the impact parameter (part of GLISSANDO 2)

8.16.2 Function Documentation

8.16.2.1 void epsilon_b (char * p)

produce plots of the mean and the scaled standard deviation of the eccentricities as functions of the impact parameter b

Produces plots of the mean and the scaled standard deviation of the fixed- and variable-axes eccentricities as functions of the impact parameter b

Parameters

p	name of the ROOT input file

8.17 macro/epsilon_c.C File Reference

```
#include "label.C"
```

Functions

- void epsilon_c (char *p)

  produces plots of the mean and the scaled standard deviation of the participant eccentricities as functions of centrality

8.17.1 Detailed Description

Script generating the plots of participant eccentricities and their scaled standard deviations as functions of centrality (part of GLISSANDO 2)

8.17.2 Function Documentation

8.17.2.1 void epsilon_c (char * p)

produces plots of the mean and the scaled standard deviation of the participant eccentricities as functions of centrality

Produce plots of the mean and the scaled standard deviation of the participant eccentricities as functions of centrality. Centrality is determined from the number of wounded nucleons.

Parameters

p	name of the ROOT input file

8.18 macro/fitr.C File Reference

```
#include "label.C"
```

Functions

- Double_t saxon (Double_t *x, Double_t *par)

  Woods-Saxon radial density function.
- void fitr (char *p)

  fit the obtained density to the Woods-Saxon form and plot the result

8.18.1 Detailed Description

Script fitting the density profile of nucleus A to the Woods-Saxon form (part of GLISSANDO 2)

8.18.2 Function Documentation

8.18.2.1 void fitr (char * *p*)

fit the obtained density to the Woods-Saxon form and plot the result

Fit the obtained density of nucleons in nucleus A to the Woods-Saxon form and plot the result. Requires _profile_=1.

Parameters

p	name of the ROOT input file

8.18.2.2 Double_t saxon (Double_t * *x*, Double_t * *par*)

Woods-Saxon radial density function.

Parameters

x	radial coordinate
par	Woods-Saxon parameters (par[0]=norm, par[1]=R, par[2]=a)

8.19 macro/fourier.C File Reference

```
#include "label.C"
```

Functions

- void fourier (char *p)

  generate the plot of epsilon_n^* vs. Nw, n=2,3,4,5,6

8.19.1 Detailed Description

Script generating the epsilon_n^∗ vs. N_w plot (part of GLISSANDO 2)

8.19.2 Function Documentation

8.19.2.1 void fourier (char ∗ p)

generate the plot of epsilon_n^∗ vs. Nw, n=2,3,4,5,6

Parameters

p	name of the ROOT input file

8.20 macro/hydro.C File Reference

Functions

- void hydro (char ∗p, char ∗pp)

  generate the grid for input to hydrodynamic calculations

8.20.1 Detailed Description

Script generating grid for input to hydrodynamic calculations, to be used as the initial condition (part of GLISSANDO 2)

8.20.2 Function Documentation

8.20.2.1 void hydro (char ∗ p, char ∗ pp)

generate the grid for input to hydrodynamic calculations

The grid with the initial condition for hydro is generated from the c0rhist histogram in the format (rho, phi, density).

Parameters

p	the ROOT input file
pp	an ASCII output file in the format (rho, phi, density)

8.21 macro/info.C File Reference

Functions

- void info (char ∗p)

  print info on the stored GLISSANDO 2 ROOT file

8.21.1 Detailed Description

Script printing info on the GLISSANDO 2 ROOT file (part of GLISSANDO 2)

8.21.2 Function Documentation

8.21.2.1 void info (char ∗ *p*)

print info on the stored GLISSANDO 2 ROOT file

Print the parameters of the simulation stored in the ROOT file. < relative variance of cross-section fluctuations

< gamma approximation wounding profile parameter (height at the origin)

Parameters

p	name of the ROOT file

8.22 macro/label.C File Reference

Functions

- void label (char ∗inpfile)

  generate the label for graphics, containing the basic information on the simulation
- void label_fit (char ∗inpfile)

  generate the label for plots referring to distributions within the nucleus

8.22.1 Detailed Description

Script generating the label for the graphics (part of GLISSANDO 2)

8.22.2 Function Documentation

8.22.2.1 void label (char ∗ *inpfile*)

generate the label for graphics, containing the basic information on the simulation

Generate the label for graphics, containing the basic information on the simulation.

Parameters

inpfile	name of the ROOT input file

8.22.2.2 void label_fit (char * *inpfile*)

generate the label for plots referring to distributions within the nucleus

Parameters

inpfile	name of the ROOT input file

8.23 macro/mult.C File Reference

```
#include "label.C"
```

Functions

- void mult (char *p)

  produces plots of the scaled variance of RDS vs. the number of wounded nucleons in the projectile

8.23.1 Detailed Description

Script plotting the event-by-event scaled variance of RDS vs. the number of wounded nucleons in the projectile (part of GLISSANDO 2)

8.23.2 Function Documentation

8.23.2.1 void mult (char * *p*)

produces plots of the scaled variance of RDS vs. the number of wounded nucleons in the projectile

Produce the plot of the scaled variance of RDS vs. the number of wounded nucleons in the projectile.

Parameters

p	name of the ROOT input file

8.24 macro/overlay.C File Reference

```
#include "label.C"
```

Functions

- void overlay (char *p)

  generate the overlaid distribution for the wounded and binary-collision sources

8.24.1 Detailed Description

Script generating the overlaid distribution for the wounded and binary-collision sources (part of GLISSANDO 2)

8.24.2 Function Documentation

8.24.2.1 void overlay (char * *p*)

generate the overlaid distribution for the wounded and binary-collision sources

Parameters

p	name of the ROOT input file

8.25 macro/profile2.C File Reference

```
#include "label.C"
```

Functions

- void profile2 (char *p)

  produces plots of the fixed-axes and variable-axes Fourier profiles of the density of sources.

8.25.1 Detailed Description

Script generating the radial profiles of subsequent Fourier components of the density (part of GLISSANDO 2)

8.25.2 Function Documentation

8.25.2.1 void profile2 (char * *p*)

produces plots of the fixed-axes and variable-axes Fourier profiles of the density of sources.

The plots are obtained by Fourier-projecting the 2-dim distribution of sources.

Parameters

p	name of the GLISSANDO input ROOT file

8.26 macro/profile2_deformation_63Cu.C File Reference

```
#include "label.C"
```

Functions

- void profile2_deformation_63Cu (char *p, char *ps)

  produces plots of the fixed-axes x-y and r-cos(theta) distributions of the nucleons in the nuleus.

8.26.1 Function Documentation

8.26.1.1 void profile2_deformation_63Cu (char * *p*, char * *ps*)

produces plots of the fixed-axes x-y and r-cos(theta) distributions of the nucleons in the nuleus.

The plots are obtained as 2D x-y and r-cos(theta) histograms of nucleons positions

Parameters

p	name of the GLISSANDO input ROOT file

8.27 macro/profile2_deformation_Au.C File Reference

```
#include "label.C"
```

Functions

- void profile2_deformation_Au (char *p, char *ps)

  produces plots of the fixed-axes x-y and r-cos(theta) distributions of the nucleons in the nuleus.

8.27.1 Function Documentation

8.27.1.1 void profile2_deformation_Au (char * *p*, char * *ps*)

produces plots of the fixed-axes x-y and r-cos(theta) distributions of the nucleons in the nuleus.

The plots are obtained as 2D x-y and r-cos(theta) histograms of nucleons positions

Parameters

p	name of the GLISSANDO input ROOT file

8.28 macro/profile2_deformation_U.C File Reference

```
#include "label.C"
```

Functions

- void profile2_deformation_U (char *p, char *ps)

  produces plots of the fixed-axes x-y and r-cos(theta) distributions of the nucleons in the nuleus.

8.28.1 Function Documentation

8.28.1.1 void profile2_deformation_U (char * *p*, char * *ps*)

produces plots of the fixed-axes x-y and r-cos(theta) distributions of the nucleons in the nuleus.

The plots are obtained as 2D x-y and r-cos(theta) histograms of nucleons positions

Parameters

p	name of the GLISSANDO input ROOT file

8.29 macro/size.C File Reference

```
#include "label.C"
```

Functions

- void size (char *p)

  Produces the plot of the scaled standard deviation of the size parameter.

8.29.1 Detailed Description

Script plotting the event-by-event scaled standard deviation of the size parameter <r>, defined as the average distance of sources from the center of mass (part of GLISSANDO 2)

8.29.2 Function Documentation

8.29.2.1 void size (char * p)

Produces the plot of the scaled standard deviation of the size parameter.

Produces the plot of the scaled standard deviation of the size parameter. Used for the transverse momentum fuctuations.

Parameters

p	name of the ROOT input file

8.30 macro/tilted.C File Reference

```
#include "label.C"
```

Functions

- void tilted (char *p)

  generating the tilted initial profile in the x-rapidity space near y=0

8.30.1 Detailed Description

Script generating the tilted initial profile in the x-rapidity space at y=0 (part of GLISSANDO 2)

8.30.2 Function Documentation

8.30.2.1 void tilted (char * p)

generating the tilted initial profile in the x-rapidity space near y=0

Generates the tilted initial profile in the x-rapidity space near y=0. Useful for testing the initial conditions to hydrodynamics. Requires _rrapidity_=1.

Parameters

p	name of the ROOT input file

8.31 macro/wounding_profile.C File Reference

```
#include "label.C"
```

Functions

- void wounding_profile (char *p)

  generate the plots of the wounding and binary-collision profiles

8.31.1 Detailed Description

Script generating the wounding and binary-collision profiles (part of GLISSANDO 2)

8.31.2 Function Documentation

8.31.2.1 void wounding_profile (char * *p*)

generate the plots of the wounding and binary-collision profiles

Parameters

p	name of the ROOT input file