GLISSANDO 2.07

Reference manual generated by Doxygen 1.6.3

Sat Aug 28 12:31:56 2010

Contents

Chapter 1

Main Page

GLISSANDO - GLauber Initial State Simulation AND mOre...

ver. 2.07, 25 August 2010

Authors:

- Wojciech Broniowski (Wojciech.Broniowski@ifj.edu.pl)
- Maciej Rybczynski (Maciej.Rybczynski@pu.kielce.pl)
- Piotr Bozek (Piotr.Bozek@ifj.edu.pl)

Modification of the code to ver. 2 by WB.

For the detailed description of ver. 1 program and further references to the description of the model, please, refer to our Computer Physics Communications 180(2009)69, arXiv:0710.5731 [nucl-th]

accessibile from: http://arxiv.org.abs/0710.5731

Implementation of nuclear correlations in ver. 2 as described in Phys. Rev. C81(2010)064909

Homepage: http://www.pu.kielce.pl/homepages/mryb/GLISSANDO/index.html

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 fixed axes (standard) and 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, modyfying the code in a simple manner. New features involve:

- 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 Bzadak [Acta Phys. Polon. B38, 159,2007])
- The generation of the core-mantle (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 published paper.

Chapter 2

Class Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

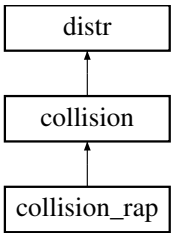
Class Documentation

5.1 collision Class Reference

collision class

`#include <distrib.h>`

Inheritance diagram for collision:



Public Member Functions

- collision (int nnA, int nnB)
  constructor
- collision ()
  default constructor
- void gen_RDS (const nucleus &nA, const nucleus &nB, float d2, float dbin2, float mb)
  collission between two nuclei

Public Attributes

- float rd2
- int wc
- int * wwA
- int * wwB

- int nwA
- int nwB
- int nwAB
- int nzw
- int nbin
- int nhotspot
- float rpa

5.1.1 Detailed Description

collision class Class performing the collision of two nuclei. Here the "source" is the position of the wounded niucleon (a nucleon that collided at least onece) or the location of the binary collision, taken as the average 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.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 collision::collision (int *nnA*, int *nnB*) [inline]

constructor

Parameters

nnA mass number of nucleus A

nnB mass number of nucleus B

5.1.2.2 collision::collision () [inline]

default constructor

assumes 208Pb-208Pb collisions

5.1.3 Member Function Documentation

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

collission 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 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 distance squared

mb ratio of the wounding to binary cross sections

5.1.4 Member Data Documentation

5.1.4.1 int collision::nbin

number of binary collisions in the event

5.1.4.2 int collision::nhotspot

number of hot spots

5.1.4.3 int collision::nwA

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

5.1.4.4 int collision::nwAB

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

5.1.4.5 int collision::nwB

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

5.1.4.6 int collision::nzw

number of sources with nonzero weight

5.1.4.7 float collision::rd2

square of the distance between the nucleons

5.1.4.8 float collision::rpa

total weight (RDS)

5.1.4.9 int collision::wc

counter of the sources

5.1.4.10 int* collision::wwA

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

5.1.4.11 int * 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:

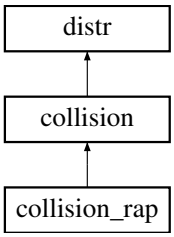
- distrib.h

5.2 collision_rap Class Reference

collision_rap class

`#include <distrib.h>`

Inheritance diagram for collision_rap:



Public Member Functions

- collision_rap (int nnA, int nnB)
  constructor
- collision_rap ()
  default constructor
- void gen_rap (int part, float maxy)
  generate the rapidity distribution

Public Attributes

- TH3D * rap_distr

5.2.1 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 transvers distribution.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 collision_rap::collision_rap (int *nnA*, int *nnB*) [inline]

constructor

Parameters

nnA mass number of nucleus A

nnB mass number of nucleus B

5.2.3 Member Function Documentation

5.2.3.1 void collision_rap::gen_rap (int *part*, float *maxy*) [inline]

generate the rapidity distribution

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

5.2.4 Member Data Documentation

5.2.4.1 TH3D* collision_rap::rap_distr

histogram for storing the rapidity distribution

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

- distrib.h

5.3 counter Class Reference

simple counting class

`#include <functions.h>`

Public Member Functions

- void reset ()
  reset the counter
- void add (long double s)
  add entry
- int getN ()
  get the number of entries
- long double get ()
  get the sum of values
- long double mean ()
  get the mean value

5.3.1 Detailed Description

simple counting class

5.3.2 Member Function Documentation

5.3.2.1 void counter::add (long double *s*) [inline]

add entry

Parameters

s value added

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

- functions.h

5.4 counter2 Class Reference

counting class with variance

`#include <functions.h>`

Public Member Functions

- void reset ()
  reset the counter
- void add (long double s)
  add entry
- int getN ()
  get the number of entries
- long double get ()
  get the sum of values
- long double get2 ()
  get the sum of squares of values
- long double mean ()
  get the mean value
- long double var ()
  get the variance
- long double vara ()
  get the variance multiplied with (N-1)/N

5.4.1 Detailed Description

counting class with variance

5.4.2 Member Function Documentation

5.4.2.1 void counter2::add (long double *s*) `[inline]`

add entry

Parameters

s value added

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

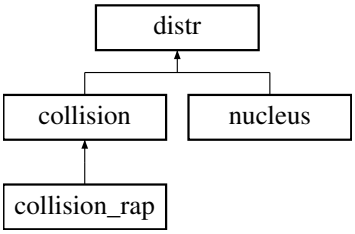
- functions.h

5.5 distr Class Reference

Distribution of sources in space.

`#include <distrib.h>`

Inheritance diagram for distr:



Public Member Functions

- distr (int k)
  constructor
- distr (void)
  default constructor, 208 sources
- distr (const distr &w1)
  copying constructor
- ~distr ()
  destructor
- float sum_w ()
  generate sum of weights
- float cmx ()
  generate the center-of-mass x coordinate, no weights
- float cmy ()
  generate the center-of-mass y coordinate, no weights
- float cmz ()
  generate the center-of-mass z coordinate, no weights
- float cmx_w ()
  generate the center-of-mass x coordinate with weights
- float cmy_w ()
  generate the center-of-mass y coordinate with weights
- float cmz_w ()
  generate the center-of-mass z coordinate with weights

- void shift_x (float xt)
  translate in the x direction
- void shift_y (float yt)
  translate in the y direction
- void shift_z (float 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 msrad ()
  mean squared radius, no weights
- float msrad_t ()
  mean squared transverse radius, no weights
- float msrad_w ()
  mean squared radius with weights
- float msrad_t_w ()
  mean squared transverse radius with weights
- float size ()
  size - the weighted average of the distance from the orgin (in the cm frame)
- float phrot (int m)
  rotation angle maximizing the m-th cosine Fourier moment
- void rotate (float ph)
  rotate in the transverse plane by the specified angle
- float eps (int m, int imin, int imax)

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

- float eps (int m)
  m-th cosine Fourier moment in the azimuthal angle in the transverse plane, no limit on charge
- float eps_s (int m, int imin, int imax)
  m-th sine Fourier moment in the azimuthal angle in the transverse plane, limited charge
- float eps_s (int m)
  m-th sine Fourier moment in the azimuthal angle in the transverse plane, no limit on charge
- void fill_xy (TH2D *xyh, float fac, int imin, int imax)
  fill the histogram of the transverse distribution in cartesian coordinates, limited charge
- void fill_xy (TH2D *xyh, float fac)
  fill the histogram of the transverse distribution in cartesian coordinates, no limit on charge
- void fill_polar (TH2D *polh, int m, float fac, int imin, int imax)
  fill the histogram of the transverse distribution in polar coordinates, limited charge
- void fill_polar_s (TH2D *polh, int m, float fac, int imin, int imax)
  fill the histogram of the transverse sine distribution in polar coordinates, limited charge
- void fill_polar (TH2D *polh, int m, float fac)
  fill the histogram of the transverse distribution in polar coordinates, no limit on charge
- void fill_polar_s (TH2D *polh, int m, float 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 n
  number of sources (points)
- float * x
  x space coordinate
- float * y
  y space coordinate
- float * z
  z space coordinate
- int * c
  some integer property, here called "charge"

- float * w
  weight
- float xcm
  center-of-mass x coordinate of the distribution
- float ycm
  center-of-mass y coordinate of the distribution
- float zcm
  center-of-mass y coordinate of the distribution
- float msr
  mean squared radius of the distribution
- float msrt
  mean squared transverse radius of the distribution
- float sumw
  sum of weights of the distribution

5.5.1 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.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 distr::distr (int *k*) [inline]

constructor

Parameters

k number of sources, k>0

5.5.2.2 distr::distr (void) [inline]

default constructor, 208 sources

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

5.5.3 Member Function Documentation

5.5.3.1 float distr::eps (int *m*) [inline]

m-th cosine Fourier moment in the azimuthal angle in the transverse plane, 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 sence in the cm frame)

5.5.3.2 float distr::eps (int *m*, int *imin*, int *imax*) [inline]

m-th cosine Fourier moment in the azimuthal angle in the transverse plane, 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 sence in the cm frame)

imin lowest charge for the sources included

imax highest charge for the sources included

5.5.3.3 float distr::eps_s (int *m*) [inline]

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

Parameters

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

5.5.3.4 float distr::eps_s (int *m*, int *imin*, int *imax*) [inline]

m-th sine Fourier moment in the azimuthal angle in the transverse plane, 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 sence in the cm frame)

imin lowest charge for the sources included

imax highest charge for the sources included

5.5.3.5 void distr::fill_polar (TH2D * *polh*, int *m*, float *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

5.5.3.6 void distr::fill_polar (TH2D * *polh*, int *m*, float *fac*, int *imin*, int *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

5.5.3.7 void distr::fill_polar_s (TH2D * *polh*, int *m*, float *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

5.5.3.8 void distr::fill_polar_s (TH2D * *polh*, int *m*, float *fac*, int *imin*, int *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

5.5.3.9 void distr::fill_xy (TH2D * *xyh*, float *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 normalizatin factor

5.5.3.10 void distr::fill_xy (TH2D * *xyh*, float *fac*, int *imin*, int *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 normalizatin factor

imin lowest charge for the sources included

imax highest charge for the sources included

5.5.3.11 float distr::phrot (int *m*) [inline]

rotation angle maximizing the m-th cosine Fourier moment

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 sence in the cm frame)

5.5.3.12 void distr::rotate (float *ph*) [inline]

rotate in the transverse plane by the specified angle

Parameters

ph rotation angle in the transverse plane

5.5.3.13 void distr::shift_x (float *xt*) [inline]

translate in the x direction

Parameters

xt displacement in the x direction

5.5.3.14 void distr::shift_y (float *yt*) [inline]

translate in the y direction

Parameters

yt displacement in the y direction

5.5.3.15 void distr::shift_z (float *zt*) [inline]

translate in the z direction

Parameters

zt displacement in the z direction

5.5.4 Member Data Documentation

5.5.4.1 int* 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.

5.5.4.2 float* distr::w

weight

Depending on the situation, the weigh may describe the amount of the deposited energy, entropy, etc.

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

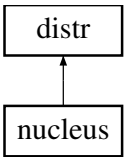
- distrib.h

5.6 nucleus Class Reference

nucleus class

`#include <distrib.h>`

Inheritance diagram for nucleus:



Public Member Functions

- nucleus (int k)
  constructor
- nucleus (const nucleus &w1)
  copying constructor
- nucleus & operator= (const nucleus &w1)
  substitution overloading
- void set_proton ()
  set the distribution for the proton
- 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_B ()
  set randomly the distribution of nucleons in the nucleus B, use the rlosB() function, no correlations
- void set_random_A (float d)
  set randomly the distribution of nucleons in the nucleus A with expulsion, use the rlosA() function
- void set_random_B (float d)
  set randomly the distribution of nucleons in the nucleus B with expulsion, use the rlosB() function
- void set_file (float *px, float *py, float *pz, int sn, int nu)
  set the distribution of nucleons in the nucleus from the tables generated earlier
- void set_file_uncor (float *px, float *py, float *pz, int sn, int nu)
  set the distribution of nucleons in the nucleus from the tables generated earlier, killing correlations
- float dist2 (int j1, int j2)

distance between two nucleons

- bool good_pair (int j1, int j2, float d)

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

- bool good_down (int j, float 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 d)

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

5.6.1 Detailed Description

nucleus class Class to store distributions of nucleons in nuclei.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 nucleus::nucleus (int *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

5.6.3 Member Function Documentation

5.6.3.1 float nucleus::dist2 (int *j1*, int *j2*) [inline]

distance between two nucleons

Parameters

j1 index of nucleon 1

j2 index of nucleon 2

5.6.3.2 bool nucleus::good_all (float *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

5.6.3.3 bool nucleus::good_down (int *j*, float *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

5.6.3.4 bool nucleus::good_pair (int *j1*, int *j2*, float *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

5.6.3.5 void nucleus::set_deuteron () [inline]

set the distribution for the deuteron

Use the Hulthen distribution.

5.6.3.6 void nucleus::set_file (float * *px*, float * *py*, float * *pz*, int *sn*, int *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

5.6.3.7 void nucleus::set_file_uncor (float * *px*, float * *py*, float * *pz*, int *sn*, int *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

5.6.3.8 void nucleus::set_proton () [inline]

set the distribution for the proton

The proton is just placed in the origin

5.6.3.9 void nucleus::set_random_A (float *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

5.6.3.10 void nucleus::set_random_B (float *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

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

- distrib.h

5.7 SOURCE Struct Reference

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

`#include <functions.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

5.7.1 Detailed Description

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

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

- functions.h
- retrieve.cpp

5.8 tr_his_c Class Reference

class storing the trees and histograms

`#include <functions.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 ()

  calculates eccentricity, size, etc. and their fuctuations 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_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 * 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 * rad

  one-body radial distribution in the nucleus

- TH1D * rrel

  distance between the pair of nucleons in the nucleus

- 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

5.8.1 Detailed Description

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

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

- functions.h

Chapter 6

File Documentation

6.1 centrality.C File Reference

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

Functions

- void centrality (char *p)

  generate the centrality classes

6.1.1 Detailed Description

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

6.1.2 Function Documentation

6.1.2.1 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 parameters are generated.

Parameters

p name of ROOT input file

6.2 centrality2.C File Reference

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

Functions

- void centrality2 (char *p)

  generate the centrality classes

6.2.1 Detailed Description

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

6.2.2 Function Documentation

6.2.2.1 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.

Parameters

p name of the ROOT input file

6.3 core_mantle.C File Reference

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

Functions

- void core_mantle (char *p)

  generate the core and mantle (corona) distribution

6.3.1 Detailed Description

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

6.3.2 Function Documentation

6.3.2.1 void core_mantle (char * *p*)

generate the core and mantle (corona) distribution

Parameters

p mname of the ROOT input file

6.4 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

6.4.1 Detailed Description

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

6.4.2 Function Documentation

6.4.2.1 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 nulcleon pairs.

Parameters

p name of thr ROOT inptut file

6.5 density.C File Reference

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

Functions

- void density (char *p)

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

6.5.1 Detailed Description

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

6.5.2 Function Documentation

6.5.2.1 void density (char * *p*)

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

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

Parameters

p name of the GLISSANDO output ROOT file

6.6 distrib.h File Reference

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

Classes

- class distr
  Distribution of sources in space.
- class nucleus
  nucleus class
- class collision
  collision class
- class collision_rap
  collision_rap class

6.6.1 Detailed Description

Part of GLISSANDO

6.7 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.

6.7.1 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)

6.7.2 Function Documentation

6.7.2.1 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 ofthe GLISSANDO output ROOT file

6.8 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

6.8.1 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)

6.8.2 Function Documentation

6.8.2.1 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 ofthe GLISSANDO output ROOT file

6.9 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

6.9.1 Detailed Description

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

6.9.2 Function Documentation

6.9.2.1 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 GLISSANDO output ROOT file

6.10 fitr.C File Reference

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

Functions

- Double_t saxon (Double_t *x, Double_t *par)
  Woods-Saxon function.
- void fitr (char *p)
  fit to the Woods-Saxon form and plot

6.10.1 Detailed Description

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

6.10.2 Function Documentation

6.10.2.1 void fitr (char * *p*)

fit to the Woods-Saxon form and plot

Parameters

p name of the GLISSANDO output ROOT file

6.11 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

6.11.1 Detailed Description

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

6.11.2 Function Documentation

6.11.2.1 void fourier (char * *p*)

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

Parameters

p mname of the ROOT input file

6.12 functions.cpp 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 "functions.h"
```

Functions

- float los ()
  random number generator using the built-in ROOT generator, uniform on (0,1)
- float rlosA ()
  random number generator for the Woods-Saxon (or Fermi) distribution - nucleus A
- float rlosB ()
  random number generator for the Woods-Saxon (or Fermi) distribution - nucleus B
- float rlos_hult ()
  random number generator for the Hulthen distribution
- float fg (float rr)
  rapidity distribution with the plateau
- float fpm (float rr)
  function f +/- from Bozek, arXiv:1002.4999v2 [nucl-th]
- float los_rap_A ()
  random number generator for the rapidity distribution - wounded nucleons from nucleus A
- float los_rap_B ()
  random number generator for the rapidity distribution - wounded nucleons from nucleus B
- float los_rap_bin ()
  random number generator for the rapidity distribution - binary collisions
- float gamgen (float a)

random number generator for the Gamma distribution

- float dist (int m, float u)
  overlaid distribution
- float disp (float w)
  random shift of the source location
- void helper (int argc, char ∗str)
  print the help
- void header ()
  print the header in the output
- int time_start ()
  start the time measurement
- void time_stop (int ts)
  stop the time measurement
- 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 epilog ()
  print epilog to the output

Variables

- float ver
  version of the code
- int EVENTS = 50000
  number of generated events
- int NBIN = 40
  number of bins for histogramming in x, y, and r
- int FBIN = 72
  number of bins for histogramming in the azimuthal angle
- int NUMA = 197
  mass number of nucleus A

- int NUMB = 197

  mass number of nucleus B

- int WMIN = 2

  minimum number of wounded nucleons to record the event

- int MODEL = 0

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

- int DOBIN = 0

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

- int W0 = 2

  minimum allowed number of wounded nucleons in the acceptance window

- int W1 = 1000

  maximum allowed number of wounded nucleons in the acceptance window

- int SHIFT = 1

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

- int RET = 0

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

- int FULL = 0

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

- int FILES = 0

  1 - read distribution from files, 0 - do not

- int GAUSS = 0

  1 - Gaussian wounding profile, 0 - hard-sphere wounding profile

- int NUMRAP = 10

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

- UInt_t ISEED

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

- UInt_t ISEED1

  copy of ISEED

- float BMIN = 0.

  minimum value of the impact parameter in the acceptance window

- float BMAX = 25.

  maximum value of the impact parameter in the acceptance window

- float RDS0 = 0.

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

- float RDS1 = 100000

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

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

  maximum impact parameter value for histogramming

- float RWSA = 6.43

  Woods-Saxon radius for nucleus A.

- float AWSA = 0.45

  Woods-Saxon width for nucleus A.

- float RWSB = 6.43

  Woods-Saxon radius for nucleus B.

- float AWSB = 0.45

  Woods-Saxon width for nucleus B.

- float WFA = 0.

  the w parameter for the Fermi distribution for nucleus A

- float WFB = 0.

  the w parameter for the Fermi distribution for nucleus B

- float SNN = 42.

  NN "wounding" cross section in millibarns.

- float SBIN = 42.

  NN binary cross section in millibarns.

- float ALPHA = 0.145

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

- float Uw = 2.

  Poisson or Gamma parameters for superimposed distribution, wounded nucleons.

- float Ubin = 2.

  Poisson or Gamma parameters for superimposed distribution, binary collisions.

- float PI = 4.∗atan(1.)

  the number pi

- float CD = 0.9

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

- float DW = 0.

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

- float DBIN = 0.

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

- float GA = 0.92

  Gaussian wounding profile parameter (hight at the origin).

- float RAPRANGE = 5.

  range in rapidity

- float ETA0 = 1.

  2∗ETA0 is the width of the plateau in eta

- float ETAM = 3.36

  parameter of the Bialas-Czyz-Bozek model

- float SIGETA = 1.3

  parameter controling the witdth of the rapidity distribution

- float MAXYRAP = 1000

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

- counter2 estd

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

- counter2 epart

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

- counter2 estd3

  counter for fixed-axes <r^2 cos(3 phi)>/<r^2>

- counter2 epart3

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

- counter2 estd4

  counter for fixed-axes <r^2 cos(4 phi)>/<r^2>

- counter2 epart4

  counter for variable-axes <r^2 cos(4 phi)>/<r^2>

- counter2 estd5

  counter for fixed-axes <r^2 cos(5 phi)>/<r^2>

- counter2 epart5

  counter for variable-axes <r^2 cos(5 phi)>/<r^2>

- counter2 estd6

  counter for fixed-axes <r^2 cos(6 phi)>/<r^2>

- counter2 epart6

  counter for variable-axes <r^2 cos(6 phi)>/<r^2>

- 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)
- int evall = 0
  number of all attempted event
- int kk
  number of the current event
- float d
  the wounding distance
- float 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 ep

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

- Float_t eps

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

- Float_t es3

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

- Float_t es3s

  fixed-axes sine moment, <r^2 sin(3 phi)>/<r^2>

- Float_t ep3

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

- Float_t ep3s

  variable-axes sine moment, <r^2 sin(3 phi)>/<r^2>

- Float_t es4

  fixed-axes <r^2 cos(4 phi)>/<r^2>

- Float_t es4s

  fixed-axes sine moment, <r^2 sin(4 phi)>/<r^2>

- Float_t ep4

  variable-axes <r^2 cos(4 phi)>/<r^2>

- Float_t ep4s

  variable-axes sine moment, <r^2 sin(4 phi)>/<r^2>

- Float_t es5

  fixed-axes <r^2 cos(5 phi)>/<r^2>

- Float_t es5s

  fixed-axes sine moment, <r^2 sin(5 phi)>/<r^2>

- Float_t ep5

  variable-axes <r^2 cos(5 phi)>/<r^2>

- Float_t ep5s

  variable-axes sine moment, <r^2 sin(5 phi)>/<r^2>

- Float_t es6

  fixed-axes <r^2 cos(6 phi)>/<r^2>

- Float_t es6s

  fixed-axes sine moment, <r^2 sin(6 phi)>/<r^2>

- Float_t ep6

  variable-axes <r^2 cos(6 phi)>/<r^2>

- Float_t ep6s

  variable-axes sine moment, <r^2 sin(6 phi)>/<r^2>

- Float_t phirot

  rotation angle maximizing the second Fourier moment

- 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

6.12.1 Detailed Description

Part of GLISSANDO

6.12.2 Function Documentation

6.12.2.1 float disp (float *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 width w.

Parameters

w average shift, Gaussian distribution

6.12.2.2 float dist (int *m*, float *u*)

overlaid distribution

distribution of RDS overlayed over the number of sources

The sources are supplied with a random weight, generated according to the distribution selected with m.

Parameters

m case: 0 - uniform, 1 - Poisson, 2 - Gamma

u average of the distribution in cases 1 and 2

6.12.2.3 float gamgen (float *a*)

random number generator for the Gamma distribution

Parameters

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

6.12.2.4 float 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].

6.12.2.5 float 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].

6.12.2.6 void readpar (TString *inpfile*)

process the input file with parameters

read parameter values from the input file

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

correct wrong input

see the paper for the discussion of parametrizatiations of nuclear distributions

Parameters

inpfile name of the input file

6.12.2.7 float rlos_hult ()

random number generator for the Hulthen distribution

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

6.12.2.8 float rlosA ()

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

random number generator for the Woods-Saxon distribution - nucleus A

6.12.2.9 float rlosB ()

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

random number generator for the Woods-Saxon distribution - nucleus B

6.12.2.10 void time_stop (int *ts*)

stop the time measurement

Parameters

ts time at start

6.13 functions.h File Reference

```
#include <TRandom3.h>
#include <iostream>
```

Classes

- struct SOURCE
  structure for output of the full event - transverse coordinates, weight, number of event
- class counter
  simple counting class
- class counter2
  counting class with variance
- class tr_his_c
  class storing the trees and histograms

Functions

- void helper (int argc, char ∗str)
  print the help
- void header ()
  print the header in the output
- void echopar ()
  echo parameters to the output
- void epilog ()
  print epilog to the output
- int time_start ()
  start the time measurement
- void time_stop (int ts)
  stop the time measurement
- void readpar (TString inpfile)
  read parameter values from the input file
- void reset_counters ()
  reset the counters used to store physical quantities in the event
- float los ()
  random number generator using the built-in ROOT generator, uniform on (0,1)

- float rlosA ()

  random number generator for the Woods-Saxon distribution - nucleus A

- float rlosB ()

  random number generator for the Woods-Saxon distribution - nucleus B

- float rlos_hult ()

  random number generator for the Hulthen distribution

- float los_rap_A ()

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

- float los_rap_B ()

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

- float los_rap_bin ()

  random number generator for the rapidity distribution - binary collisions

- float gamgen (float a)

  random number generator for the Gamma distribution

- float dist (int m, float u)

  distribution of RDS overlayed over the number of sources

- float disp (float x)

  random shift of the source location

Variables

- TRandom3 raa

  random number generator from ROOT

- float ver

  version of the code

- int EVENTS

  number of generated events

- int NBIN

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

- int FBIN

  number of bins for histogramming in the azimuthal angle

- int NUMA

  mass number of nucleus A

- int NUMB

  mass number of nucleus B

- int WMIN

  minimum number of wounded nucleons to record the event

- int MODEL

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

- int DOBIN

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

- int W0

  minimum allowed number of wounded nucleons in the acceptance window

- int W1

  maximum allowed number of wounded nucleons in the acceptance window

- int SHIFT

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

- int RET

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

- int FULL

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

- int FILES

  1 - read distribution from files, 0 - do not

- int GAUSS

  1 - Gaussian wounding profile, 0 - hard-sphere wounding profile

- int NUMRAP

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

- UInt_t ISEED

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

- UInt_t ISEED1

  copy of ISEED

- float BMIN

  minimum value of the impact parameter in the acceptance window

- float BMAX

  maximum value of the impact parameter in the acceptance window

- float RDS0

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

- float RDS1

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

- float BTOT

  maximum impact parameter value for histogramming

- float RWSA

  Woods-Saxon radius for nucleus A.

- float AWSA

  Woods-Saxon width for nucleus A.

- float RWSB

  Woods-Saxon radius for nucleus B.

- float AWSB

  Woods-Saxon width for nucleus B.

- float WFA

  the w parameter for the Fermi distribution for nucleus A

- float WFB

  the w parameter for the Fermi distribution for nucleus B

- float SNN

  NN "wounding" cross section in millibarns.

- float SBIN

  NN binary cross section in millibarns.

- float ALPHA

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

- float Uw

  Poisson or Gamma parameters for superimposed distribution, wounded nucleons.

- float Ubin

  Poisson or Gamma parameters for superimposed distribution, binary collisions.

- float PI

  the number pi

- float CD

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

- float DW

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

- float DBIN

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

- float GA

  Gaussian wounding profile parameter (hight at the origin).

- float RAPRANGE

  range in rapidity

- float ETA0

  2∗ETA0 is the width of the plateau in eta

- float ETAM

  parameter of the Bialas-Czyz-Bozek model

- float SIGETA

  parameter controling the witdth of the rapidity distribution

- float MAXYRAP

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

- counter2 estd

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

- counter2 epart

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

- counter2 estd3

  counter for fixed-axes <r^2 cos(3 phi)>/<r^2>

- counter2 epart3

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

- counter2 estd4

  counter for fixed-axes <r^2 cos(4 phi)>/<r^2>

- counter2 epart4

  counter for variable-axes <r^2 cos(4 phi)>/<r^2>

- counter2 estd5

  counter for fixed-axes <r^2 cos(5 phi)>/<r^2>

- counter2 epart5

  counter for variable-axes <r^2 cos(5 phi)>/<r^2>

- counter2 estd6

  counter for fixed-axes <r^2 cos(6 phi)>/<r^2>

- counter2 epart6

  counter for variable-axes <r^2 cos(6 phi)>/<r^2>

- 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)
- int evall
  number of all attempted event
- int kk
  number of the current event
- float d
  the wounding distance
- float 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 ep

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

- Float_t eps

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

- Float_t es3

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

- Float_t es3s

  fixed-axes sine moment, <r^2 sin(3 phi)>/<r^2>

- Float_t ep3

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

- Float_t ep3s

  variable-axes sine moment, <r^2 sin(3 phi)>/<r^2>

- Float_t es4

  fixed-axes <r^2 cos(4 phi)>/<r^2>

- Float_t es4s

  fixed-axes sine moment, <r^2 sin(4 phi)>/<r^2>

- Float_t ep4

  variable-axes <r^2 cos(4 phi)>/<r^2>

- Float_t ep4s

  variable-axes sine moment, <r^2 sin(4 phi)>/<r^2>

- Float_t es5

  fixed-axes <r^2 cos(5 phi)>/<r^2>

- Float_t es5s

  fixed-axes sine moment, <r^2 sin(5 phi)>/<r^2>

- Float_t ep5

  variable-axes <r^2 cos(5 phi)>/<r^2>

- Float_t ep5s

  variable-axes sine moment, <r^2 sin(5 phi)>/<r^2>

- Float_t es6

  fixed-axes <r^2 cos(6 phi)>/<r^2>

- Float_t es6s

  fixed-axes sine moment, <r^2 sin(6 phi)>/<r^2>

- Float_t ep6

  variable-axes <r^2 cos(6 phi)>/<r^2>

- Float_t ep6s

  variable-axes sine moment, <r^2 sin(6 phi)>/<r^2>

- Float_t phirot

  rotation angle maximizing the second Fourier moment

- 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

6.13.1 Detailed Description

Part of GLISSANDO

6.13.2 Function Documentation

6.13.2.1 float disp (float *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 width w.

Parameters

w average shift, Gaussian distribution

6.13.2.2 float dist (int *m*, float *u*)

distribution of RDS overlayed over the number of sources

distribution of RDS overlayed over the number of sources

The sources are supplied with a random weight, generated according to the distribution selected with m.

Parameters

m case: 0 - uniform, 1 - Poisson, 2 - Gamma

u average of the distribution in cases 1 and 2

6.13.2.3 float gamgen (float *a*)

random number generator for the Gamma distribution

Parameters

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

6.13.2.4 float 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].

6.13.2.5 float 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].

6.13.2.6 void readpar (TString *inpfile*)

read parameter values from the input file

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

correct wrong input

see the paper for the discussion of parametrizatiations of nuclear distributions

Parameters

inpfile name of the input file

6.13.2.7 float rlos_hult ()

random number generator for the Hulthen distribution

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

6.13.2.8 void time_stop (int *ts*)

stop the time measurement

Parameters

ts time at start

6.14 glissando.cpp File Reference

```
#include <math.h>
#include <time.h>
#include <string.h>
#include <iostream>
#include <fstream>
#include <TH1D.h>
#include <TH2D.h>
#include <TH3D.h>
#include <TFile.h>
#include <TTree.h>
#include <TRandom3.h>
#include "functions.h"
#include "distrib.h"
```

Functions

- int main (int argc, char ∗argv[ ])
  the main function of GLISSANDO

Variables

- float ver = 2.07
  version of the code
- TRandom3 raa
  declare the ROOT random number generator

6.14.1 Detailed Description

The main file of GLISSANDO 2

6.14.2 Function Documentation

6.14.2.1 int main (int *argc*, char ∗ *argv*[ ])

the main function of GLISSANDO
(the units for all dimensionful quantities 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

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 main loop over events

generate the distributions of nucleons in nuclei A and B

shift nuclei to the center-of-mass frame

if(_profile_) generate the histograms of one-body density in nucleus A and 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

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 Fourier moments

get some basic properties of the event

fill the data in trees and histograms

if(FULL) write the full event info to the file

--- end of main loop over events

output of results

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

generate and write some histograms with physical quantities

write exit info

stop time

Parameters

argv used for passing input and output file names

6.14.3 Variable Documentation

6.14.3.1 TRandom3 raa

declare the ROOT random number generator

random number generator from ROOT

6.15 info.C File Reference

Functions

- void info (char *p)

  print info on the stored GLISSANDO 2 ROOT file

6.15.1 Detailed Description

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

6.15.2 Function Documentation

6.15.2.1 void info (char * *p*)

print info on the stored GLISSANDO 2 ROOT file

Parameters

p input file name

6.16 interpolation.cpp 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>
```

Functions

- double w1D (int n, double dx, double x[4], double y[4])

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

- double w2D (double x, double y, double x0, double y0, double u0, double x1, double y1, double u1, double x2, double y2, double u2)

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

- double lin (double dx, double x1, double y1, double x2, double y2)

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

- double inter1D (double dx, TFile &f, char ∗histname)

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

- double inter2D (double dx, double 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 mean_weight (TFile &f)

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

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

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

- double integral2D (double 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 main (int argc, char **argv)

Variables

- char hname [25]

  name of the histogram from the GLISSANDO output ROOT file

6.16.1 Detailed Description

auxilliary file, part of GLISSANDO 2

6.16.2 Function Documentation

6.16.2.1 int main (int *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

argv name of the GLISSANDO output ROOT file

6.17 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

6.17.1 Detailed Description

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

6.17.2 Function Documentation

6.17.2.1 void label (char * *inpfile*)

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

Parameters

inpfile name of the input file

6.17.2.2 void label_fit (char * *inpfile*)

generate the label for plots referring to distributions within the nucleus

Parameters

inpfile input file name

6.18 overlay.C File Reference

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

Functions

- void overlay (char *p)

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

6.18.1 Detailed Description

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

6.18.2 Function Documentation

6.18.2.1 void overlay (char * *p*)

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

Parameters

p name of the ROOT input file

6.19 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.

6.19.1 Detailed Description

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

6.19.2 Function Documentation

6.19.2.1 void profile2 (char * *p*)

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

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

Parameters

p name of the GLISSANDO input ROOT file

6.20 retrieve.cpp 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 event

Functions

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

6.20.1 Detailed Description

auxilliary file, part of GLISSANDO 2

6.20.2 Function Documentation

6.20.2.1 int main (int *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

argv name of the GLISSANDO output ROOT file

6.21 size.C File Reference

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

Functions

- void size (char *p)

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

6.21.1 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)

6.21.2 Function Documentation

6.21.2.1 void size (char * *p*)

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

Used for the transverse momentum fuctuations.

Parameters

p name of the GLISSANDO input ROOT file

6.22 tilted.C File Reference

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

Functions

- void tilted (char *p)

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

6.22.1 Detailed Description

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

6.22.2 Function Documentation

6.22.2.1 void tilted (char * *p*)

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

Parameters

p mname of the ROOT input file

6.23 wounding_profile.C File Reference

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

Functions

- void wounding_profile (char *p)

  generate the plots of the wounding and binary-collision profiles

6.23.1 Detailed Description

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

6.23.2 Function Documentation

6.23.2.1 void wounding_profile (char * *p*)

generate the plots of the wounding and binary-collision profiles

Parameters

p name of the ROOT input file

Index