GLISSANDO 3

Generated by Doxygen 1.8.6

Mon Nov 12 2018 10:36:01

Contents

Chapter 1

Main Page

GLISSANDO 3 - GLauber Initial State Simulation AND mOre... ver. 3.110, 28 October 2018

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

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@fis.agh.edu.pl)

For ver. 3, see arXiv: (∗ fill arXiv ref. ∗)

For ver. 2, see Computer Physics Communications 185 (2014) 1759, arXiv:1310.5475 [nucl-th]

For ver. 1, see Computer Physics Communications 180(2009) 69, arXiv:0710.5731 [nucl-th]

GLISSANDO is a Glauber Monte-Carlo generator for early-stages of relativistic heavy-ion collisions, written in c++ and interfaced to ROOT.

A reference manual, generated by doxygen, is supplied at the Homepage.

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

(∗ fill arXiv ref. ∗), Computer Physics Communications 180 (2009) 69, arXiv:0710.5731 [nucl-th], and Computer Physics Communications 185 (2014) 1759, arXiv:1310.5475 [nucl-th]

Chapter 2

Hierarchical 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 files with brief descriptions:

Chapter 5

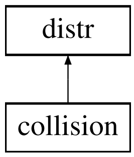
Class Documentation

5.1 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
- void shift_cmx_w_c ()

    translate to the cm reference frame in the x direction with weights, including the spactator coordinates
- void shift_cmy_w_c ()

Public Attributes

- Float_t rd2
- Int_t wc
- Int_t ∗ wwA
- Int_t ∗ wwB
- Int_t ∗ wwAB
- Int_t nwA

- Int_t nwB
- Int_t nwAB
- Int_t nzw
- Int_t nbin
- Int_t nhotspot
- Int_t specA
- Int_t specB
- Float_t xcmA
- Float_t xcmB
- Float_t ycmA
- Float_t ycmB
- Float_t rpa

5.1.1 Detailed Description

collision class

Class performing the collision of two nuclei.

5.1.2 Constructor & Destructor Documentation

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

constructor

< maximum number of wounded objects + binary

Parameters

nnA	number of objects in nucleus A
nnB	number of objects in nucleus B

5.1.2.2 collision::collision ( ) [inline]

default constructor

The default constructor assumes 208Pb-208Pb collisions with nucleons.

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

copying constructor

5.1.3 Member Function Documentation

5.1.3.1 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 wel
```

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 (for hotspots)

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

substitution overloading

5.1.3.3 void collision::shift_cmx_w_c ( ) [inline]

translate to the cm reference frame in the x direction with weights, including the spactator coordinates

5.1.3.4 void collision::shift_cmy_w_c ( ) [inline]

5.1.4 Member Data Documentation

5.1.4.1 Int_t collision::nbin

number of binary collisions in the event

5.1.4.2 Int_t collision::nhotspot

number of hot spots

5.1.4.3 Int_t collision::nwA

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

5.1.4.4 Int_t collision::nwAB

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

5.1.4.5 Int_t collision::nwB

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

5.1.4.6 Int_t collision::nzw

number of sources with non-zero weight

5.1.4.7 Float_t collision::rd2

square of the distance between the colliding objects (nucleons or partons)

5.1.4.8 Float_t collision::rpa

weight of the source (RDS)

5.1.4.9 Int_t collision::specA

number of spectators in A

5.1.4.10 Int_t collision::specB

number of spectators in B

5.1.4.11 Int_t collision::wc

counter of the sources

5.1.4.12 Int_t* collision::wwA

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

5.1.4.13 Int_t* collision::wwAB

5.1.4.14 Int_t * collision::wwB

wwB[i] is the number of collisions of the i-th object from nucleus A with the objects from nucleus A

5.1.4.15 Float_t collision::xcmA

x cm cordinate of the spectators from the A nucleus

5.1.4.16 Float_t collision::xcmB

x cm cordinate of the spectators from the B nucleus

5.1.4.17 Float_t collision::ycmA

y cm cordinate of the spectators from the A nucleus

5.1.4.18 Float_t collision::ycmB

y cm cordinate of the spectators from the B nucleus

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

- build/include/collision.h

5.2 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

5.2.1 Detailed Description

simplest counting class

Class counting the mean.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 counter::counter ( ) [inline]

constructor

5.2.2.2 counter::∼counter ( ) [inline]

destructor

5.2.3 Member Function Documentation

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

add entry to the counter

Parameters

s	value added

5.2.3.2 Double_t counter::get ( ) [inline]

get the sum of values

5.2.3.3 Int_t counter::getN ( ) [inline]

get the number of entries

5.2.3.4 Double_t counter::mean ( ) [inline]

get the mean value

5.2.3.5 void counter::reset ( ) [inline]

reset the counter

5.2.4 Member Data Documentation

5.2.4.1 Int_t counter::count [private]

number of entries

5.2.4.2 Double_t counter::value [private]

sum of values counted

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

- build/include/counter.h

5.3 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

5.3.1 Detailed Description

counting class with variance

Class counting the mean and variance.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 counter2::counter2 ( ) [inline]

constructor

5.3.2.2 counter2::~counter2 ( ) [inline]

destructor

5.3.3 Member Function Documentation

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

add entry

Parameters

s	value added

5.3.3.2 Double_t counter2::get ( ) [inline]

get the sum of values

5.3.3.3 Double_t counter2::get2 ( ) [inline]

get the sum of squares of values

5.3.3.4 Int_t counter2::getN ( ) [inline]

get the number of entries

5.3.3.5 Double_t counter2::mean ( ) [inline]

get the mean value

5.3.3.6 void counter2::reset ( ) [inline]

reset the counter

5.3.3.7 Double_t counter2::var ( ) [inline]

get the variance

5.3.3.8 Double_t counter2::vara ( ) [inline]

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

5.3.4 Member Data Documentation

5.3.4.1 Int_t counter2::count [private]

number of entries

5.3.4.2 Double_t counter2::value [private]

sum of values

5.3.4.3 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

5.4 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 Pearson correlation coefficient

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

5.4.1 Detailed Description

2-dimensional counting class

Class counting the means, variances, and covariance for a 2D sample.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 counter_2D::counter_2D ( ) [inline]

constructor

5.4.2.2 counter_2D::∼counter_2D ( ) [inline]

destructor

5.4.3 Member Function Documentation

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

add entry

Parameters

sx	x value added
sy	y value added

5.4.3.2 Double_t counter_2D::corr ( ) [inline]

get the Pearson correlation coefficient

5.4.3.3 Double_t counter_2D::cov ( ) [inline]

get the covariance

5.4.3.4 Double_t counter_2D::get_x ( ) [inline]

get the sum of x values

5.4.3.5 Double_t counter_2D::get_x2 ( ) [inline]

get the sum of x2

5.4.3.6 Double_t counter_2D::get_xy ( ) [inline]

get the sum of xy

5.4.3.7 Double_t counter_2D::get_y ( ) [inline]

get the sum of x values

5.4.3.8 Double_t counter_2D::get_y2 ( ) [inline]

get the sum of y2

5.4.3.9 Int_t counter_2D::getN ( ) [inline]

get the number of entries

5.4.3.10 Double_t counter_2D::mean_x ( ) [inline]

get the mean x value

5.4.3.11 Double_t counter_2D::mean_y ( ) [inline]

get the mean y value

5.4.3.12 void counter_2D::reset ( ) [inline]

reset the counter

5.4.3.13 Double_t counter_2D::var_x ( ) [inline]

get the variance of x

5.4.3.14 Double_t counter_2D::var_y ( ) [inline]

get the variance of y

5.4.4 Member Data Documentation

5.4.4.1 Int_t counter_2D::count [private]

number of entries

5.4.4.2 Double_t counter_2D::valuex [private]

sum of x values

5.4.4.3 Double_t counter_2D::valuex2 [private]

sum of x^2

5.4.4.4 Double_t counter_2D::valuexy [private]

sum of x*y

5.4.4.5 Double_t counter_2D::valuey [private]

sum of y values

5.4.4.6 Double_t counter_2D::valuey2 [private]

sum of y^2

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

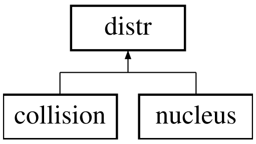
- build/include/counter.h

5.5 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 uncluster ()
  uncluster
- 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 (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 (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 qx (Int_t m, Float_t k, Float_t d)
  x-component of the Q vector in the transverse plane, weight r^k, no limit on charge, with smearing
- Float_t qy (Int_t m, Float_t k, Float_t d)
  y-component of the Q vector in the transverse plane, weight r^k, no limit on charge, with smearing
- 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
- 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

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_t *k*) [inline]

constructor

Parameters

k	number of sources, k>0

5.5.2.2 distr::distr (void) [inline]

default constructor, 208 sources

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

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

copying constructor

5.5.3 Member Function Documentation

5.5.3.1 Float_t distr::cmx ( ) [inline]

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

5.5.3.2 Float_t distr::cmx_w ( ) [inline]

generate the center-of-mass x coordinate with weights

5.5.3.3 Float_t distr::cmy ( ) [inline]

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

5.5.3.4 Float_t distr::cmy_w ( ) [inline]

generate the center-of-mass y coordinate with weights

5.5.3.5 Float_t distr::cmz ( ) [inline]

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

5.5.3.6 Float_t distr::cmz_w ( ) [inline]

generate the center-of-mass z coordinate with weights

5.5.3.7 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,...
imin	lowest charge for the sources included
imax	highest charge for the sources included

5.5.3.8 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,...

5.5.3.9 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,...
k	power of transverse radius in the weight

5.5.3.10 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,...
k	power of transverse radius in the weight
imin	lowest charge for the sources included
imax	highest charge for the sources included

5.5.3.11 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

5.5.3.12 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

5.5.3.13 Float_t distr::msrad ( ) [inline]

mean squared radius, no weights

5.5.3.14 Float_t distr::msrad_t ( ) [inline]

mean squared transverse radius, no weights

5.5.3.15 Float_t distr::msrad_t_w ( ) [inline]

mean squared transverse radius with weights

5.5.3.16 Float_t distr::msrad_w ( ) [inline]

mean squared radius with weights

5.5.3.17 Float_t distr::msx ( ) [inline]

mean squared x, no weights

5.5.3.18 Float_t distr::msy ( ) [inline]

mean squared y, no weights

5.5.3.19 Float_t distr::mxy ( ) [inline]

mean x*y, no weights

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

substitution overloading

5.5.3.21 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,...

5.5.3.22 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,...
k	power of transverse radius in the weight

5.5.3.23 Float_t distr::qx (Int_t *m,* Float_t *k,* Float_t *d*) [inline]

x-component of the Q vector in the transverse plane, weight r^k, no limit on charge, with smearing

The smearing formula is exact for k=2 and k=4, and approximate (expanded for small d) for k=3. See the Glissando 3 paper for details

Parameters

m	rank of the Fourier moment, m=0,2,3,4,5,...

k	power of transverse radius in the weight
d	smearing parameter

5.5.3.24 Float_t distr::qy (Int_t *m*, Float_t *k*, Float_t *d*) [inline]

y-component of the Q vector in the transverse plane, weight r^k, no limit on charge, with smearing

The smearing formula is exact for k=2 and k=4, and approximate (expanded for small d) for k=3. See the Glissando 3 paper for details

Parameters

m	rank of the Fourier moment, m=0,2,3,4,5,...
k	power of transverse radius in the weight
d	smearing parameter

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

rotate in the transverse plane by the specified angle

Parameters

ph	rotation angle in the transverse plane

5.5.3.26 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

5.5.3.27 void distr::shift_cmx ( ) [inline]

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

5.5.3.28 void distr::shift_cmx_w ( ) [inline]

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

5.5.3.29 void distr::shift_cmy ( ) [inline]

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

5.5.3.30 void distr::shift_cmy_w ( ) [inline]

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

5.5.3.31 void distr::shift_cmz ( ) [inline]

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

5.5.3.32 void distr::shift_cmz_w ( ) [inline]

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

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

translate in the x direction

Parameters

xt	displacement in the x direction

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

translate in the y direction

Parameters

yt	displacement in the y direction

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

translate in the z direction

Parameters

zt	displacement in the z direction

5.5.3.36 Float_t distr::size ( ) [inline]

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

5.5.3.37 Float_t distr::sum_w ( ) [inline]

destructor

generate sum of weights

5.5.3.38 void distr::uncluster ( ) [inline]

uncluster

5.5.4 Member Data Documentation

5.5.4.1 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.

5.5.4.2 Float_t distr::msr

mean squared radius of the distribution

5.5.4.3 Float_t distr::msrt

mean squared transverse radius of the distribution

5.5.4.4 Int_t distr::n

number of sources (points)

5.5.4.5 Float_t distr::sumw

sum of weights of the distribution

5.5.4.6 Float_t* distr::w

weight

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

5.5.4.7 Float_t* distr::x

x space coordinate

5.5.4.8 Float_t distr::xcm

center-of-mass x coordinate of the distribution

5.5.4.9 Float_t* distr::y

y space coordinate

5.5.4.10 Float_t distr::ycm

center-of-mass y coordinate of the distribution

5.5.4.11 Float_t* distr::z

z space coordinate

5.5.4.12 Float_t distr::zcm

center-of-mass z coordinate of the distribution

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

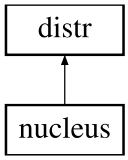
- build/include/distrib.h

5.6 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_parton_A (const nucleus &nucA, Int_t np)

  destructor
- void set_parton_B (const nucleus &nucB, Int_t np)

  set randomly the distribution of partons around centres of nucleons inside nucleus B
- void set_proton ()

  set the distribution for the proton
- void set_deuteron ()

  set the distribution for the deuteron
- void set_deuteron_s1 ()
- void set_deuteron_s0 ()
- void set_random_A ()

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

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

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

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

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

  set randomly the distribution of nucleons in nucleus B with deformation, no correlations
- void set_random_A_def (Float_t d)
- void set_random_B_def (Float_t d)
- void set_tritium (Float_t scale)
- void set_alpha_cluster (Float_t d, Float_t sigma, Float_t bp, Float_t dp)
- void set_berillium_7_cluster (Float_t scale, Float_t d, Float_t sigma, Float_t sigmabis)
- void set_berillium_8_cluster (Float_t scale, Float_t d, Float_t sigma)
- void set_berillium_9_cluster (Float_t scale, Float_t d, Float_t sigma, Float_t sigmabis)
- void set_carbon_cluster (Float_t scale, Float_t d, Float_t sigma)
- void set_oxygen_cluster (Float_t scale, Float_t d, Float_t sigma)

- void set_oxygen_cluster_square (Float_t scale, Float_t d, Float_t sigma)
- 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

Additional Inherited Members

5.6.1 Detailed Description

nucleus class

Class to store distributions of nucleons (or partons) in nuclei.

5.6.2 Constructor & Destructor Documentation

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

constructor

Parameters

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

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

copying constructor

5.6.3 Member Function Documentation

5.6.3.1 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

5.6.3.2 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

5.6.3.3 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

5.6.3.4 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

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

substitution overloading

5.6.3.6 void nucleus::set_alpha_cluster (Float_t *d*, Float_t *sigma*, Float_t *bp*, Float_t *dp*) [inline]

set alpha cluster

Parameters

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

bp	not used
dp	not used

5.6.3.7 void nucleus::set_berillium_7_cluster (Float_t *scale*, Float_t *d*, Float_t *sigma*, Float_t *sigmabis*) [inline]

set clustered 7Be (dumbbell of alpha and 3He)

Parameters

scale	scale parameter for the size of the nucleus
d	expulsion distance, nucleons cannot be closer to each other than d
sigma	standard deviation of x,y,z coordinates of nucleons in the alpha cluster
sigmabis	standard deviation of x,y,z coordinates of nucleons in the 3He cluster

5.6.3.8 void nucleus::set_berillium_8_cluster (Float_t *scale*, Float_t *d*, Float_t *sigma*) [inline]

set clustered 8Be (dumbbell)

Parameters

scale	scale parameter for the size of the nucleus
d	expulsion distance, nucleons cannot be closer to each other than d
sigma	standard deviation of x,y,z coordinates of nucleons in the cluster

5.6.3.9 void nucleus::set_berillium_9_cluster (Float_t *scale*, Float_t *d*, Float_t *sigma*, Float_t *sigmabis*) [inline]

set clustered 9Be (dumbbell + extra neutron)

Parameters

scale	scale parameter for the size of the nucleus
d	expulsion distance, nucleons cannot be closer to each other than d
sigma	standard deviation of x,y,z coordinates of nucleons in the cluster
sigmabis	standard deviation of x,y,z coordinates of nucleon no. 9

5.6.3.10 void nucleus::set_carbon_cluster (Float_t *scale*, Float_t *d*, Float_t *sigma*) [inline]

set clustered 12C (triangle)

Parameters

scale	scale parameter for the size of the nucleus
d	expulsion distance, nucleons cannot be closer to each other than d
sigma	standard deviation of x,y,z coordinates of nucleons in the cluster

5.6.3.11 void nucleus::set_deuteron ( ) [inline]

set the distribution for the deuteron

Use the Hulthen distribution.

5.6.3.12 void nucleus::set_deuteron_s0 ( ) [inline]

5.6.3.13 void nucleus::set_deuteron_s1 ( ) [inline]

set the distributions of the polarized deuteron

5.6.3.14 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

5.6.3.15 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

5.6.3.16 void nucleus::set_oxygen_cluster (Float_t *scale,* Float_t *d,* Float_t *sigma*) [inline]

set clustered 16O (tetrahedron)

Parameters

scale	scale parameter for the size of the nucleus
d	expulsion distance, nucleons cannot be closer to each other than d
sigma	standard deviations of x,y,z coordinates of nucleons in the cluster

5.6.3.17 void nucleus::set_oxygen_cluster_square (Float_t *scale,* Float_t *d,* Float_t *sigma*) [inline]

set clustered 16O (square)

Parameters

scale	scale parameter for the size of the nucleus
d	expulsion distance, nucleons cannot be closer to each other than d
sigma	standard deviation of x,y,z coordinates of nucleons in the cluster

5.6.3.18 void nucleus::set_parton_A (const nucleus & *nucA*, Int_t *np*) [inline]

destructor

set randomly the distribution of np partons around centre of nucleon A

Parameters

nucA	nucleus A
np	number of partons in nucleon

5.6.3.19 void nucleus::set_parton_B (const nucleus & *nucB*, Int_t *np*) [inline]

set randomly the distribution of partons around centres of nucleons inside nucleus B

Parameters

nucB	nucleus B
np	number of partons in nucleon

5.6.3.20 void nucleus::set_proton ( ) [inline]

set the distribution for the proton

The proton is just placed at the origin

5.6.3.21 void nucleus::set_random_A ( ) [inline]

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

5.6.3.22 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

5.6.3.23 void nucleus::set_random_A_def ( ) [inline]

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

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

Parameters

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

5.6.3.25 void nucleus::set_random_A_hos ( ) [inline]

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

5.6.3.26 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

5.6.3.27 void nucleus::set_random_B ( ) [inline]

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

5.6.3.28 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

5.6.3.29 void nucleus::set_random_B_def ( ) [inline]

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

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

Parameters

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

5.6.3.31 void nucleus::set_random_B_hos ( ) [inline]

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

5.6.3.32 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

5.6.3.33 void nucleus::set_tritium (Float_t *scale*) [inline]

set triangular tritium

Parameters

scale	scale of the triangle

5.6.4 Member Data Documentation

5.6.4.1 Float_t nucleus::cth [private]

5.6.4.2 bool nucleus::g [private]

5.6.4.3 Float_t nucleus::phi [private]

5.6.4.4 Float_t nucleus::r [private]

5.6.4.5 Float_t nucleus::sth [private]

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

- build/include/distrib.h

5.7 SOURCE Struct Reference

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

`#include <functions.h>`

Public Attributes

- Float_t X

  x coordinate
- Float_t Y

  y coordinate
- Float_t W

  weight
- UInt_t KK

  number of the event

5.7.1 Detailed Description

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

5.7.2 Member Data Documentation

5.7.2.1 UInt_t SOURCE::KK

number of the event

5.7.2.2 Float_t SOURCE::W

weight

5.7.2.3 Float_t SOURCE::X

x coordinate

5.7.2.4 Float_t SOURCE::Y

y coordinate

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

- build/include/functions.h

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 fill_res ()

  calculate eccentricity, size, etc. and their fluctuations vs. number of wounded objects 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 objects 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 participant-plane distribution
- 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 ∗ nepsp

  participant-plane eccentricity vs. Nw
- TH1D ∗ nepsp2

  square of participant-plane eccentricity, then its variance, vs. Nw
- TH1D ∗ nepsp4

  participant-plane fourth moment vs. Nw
- TH1D ∗ nuni

  frequency of Nw, i.e. histogram of unity vs. Nw
- TH1D ∗ nuniRDS

  frequency of RDS, i.e. histogram of unity vs. RDS
- TH1D ∗ nunib

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

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

- TH1D * nw2b

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

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

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

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

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

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

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

  frequency of the number of wounded objects 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 objects in the nucleus A
- TH1D * rrelB

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

  uncorrelated distance between the pair of objects in the nucleus (one nucleon from A, the other one from B)
- TH1D * weih

  the distribution overlaid on the wounded objects
- TH1D * weih_bin

  the distribution overlaid over binary collisions
- TH1D * wpro

  the wounding profile

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 or other codes

5.8.2 Member Function Documentation

5.8.2.1 void tr_his_c::fill ( ) [inline]

fill trees param and phys

5.8.2.2 void tr_his_c::fill_res ( ) [inline]

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

5.8.2.3 void tr_his_c::fill_tr ( ) [inline]

fill the main tree

5.8.2.4 void tr_his_c::gen ( ) [inline]

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

5.8.2.5 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

5.8.2.6 void tr_his_c::write ( ) [inline]

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

5.8.2.7 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

5.8.2.8 void tr_his_c::write_r ( ) [inline]

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

5.8.2.9 void tr_his_c::write_w ( ) [inline]

write out the overlaid distributions

5.8.2.10 void tr_his_c::write_wpro ( ) [inline]

write out the wounding profile

5.8.3 Member Data Documentation

5.8.3.1 TTree * tr_his_c::full_event

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

5.8.3.2 TH1D * tr_his_c::nbinar

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

5.8.3.3 TH1D $*$ tr_his_c::nbinar2

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

5.8.3.4 TH1D $*$ tr_his_c::nepsp

participant-plane eccentricity vs. Nw

5.8.3.5 TH1D $*$ tr_his_c::nepsp2

square of participant-plane eccentricity, then its variance, vs. Nw

5.8.3.6 TH1D $*$ tr_his_c::nepsp4

participant-plane fourth moment vs. Nw

5.8.3.7 TH1D $*$ tr_his_c::nsize

size vs. Nw

5.8.3.8 TH1D $*$ tr_his_c::nsize2

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

5.8.3.9 TH1D $*$ tr_his_c::ntarg

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

5.8.3.10 TH1D $*$ tr_his_c::ntarg2

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

5.8.3.11 TH1D $*$ tr_his_c::nuni

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

5.8.3.12 TH1D $*$ tr_his_c::nunib

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

5.8.3.13 TH1D $*$ tr_his_c::nuniRDS

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

5.8.3.14 TH1D $*$ tr_his_c::nunp

frequency of the number of wounded objects in nucleus A

5.8.3.15 TH1D ∗ tr_his_c::nw2b

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

5.8.3.16 TH1D ∗ tr_his_c::nwb

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

5.8.3.17 TH1D∗ tr_his_c::nwei

RDS vs. number of wounded objects in nucleus A.

5.8.3.18 TH1D ∗ tr_his_c::nwei2

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

5.8.3.19 TH1D∗ tr_his_c::nx

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

5.8.3.20 TH1D ∗ tr_his_c::nx2

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

5.8.3.21 TH1D ∗ tr_his_c::ny

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

5.8.3.22 TH1D ∗ tr_his_c::ny2

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

5.8.3.23 TTree∗ tr_his_c::param

parameters

5.8.3.24 TTree ∗ tr_his_c::phys

A+B cross section and other physical results.

5.8.3.25 TH1D∗ tr_his_c::radA

one-body radial distribution in the nucleus A

5.8.3.26 TH1D ∗ tr_his_c::radB

one-body radial distribution in the nucleus B

5.8.3.27 TH2D * tr_his_c::rcostheta_nuclA

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

5.8.3.28 TH2D * tr_his_c::rcostheta_nuclB

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

5.8.3.29 TH1D * tr_his_c::rrel_u

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

5.8.3.30 TH1D * tr_his_c::rrelA

distance between the pair of objects in the nucleus A

5.8.3.31 TH1D * tr_his_c::rrelB

distance between the pair of objects in the nucleus B

5.8.3.32 TTree * tr_his_c::tree

basic physical results

5.8.3.33 TH1D * tr_his_c::weih

the distribution overlaid on the wounded objects

5.8.3.34 TH1D * tr_his_c::weih_bin

the distribution overlaid over binary collisions

5.8.3.35 TH1D * tr_his_c::wpro

the wounding profile

5.8.3.36 TH2D* tr_his_c::xyhist

cartesian fixed-axes distribution

5.8.3.37 TH2D * tr_his_c::xyhist_core

cartesian fixed-axes core distribution

5.8.3.38 TH2D * tr_his_c::xyhist_mantle

cartesian fixed-axes mantle distribution

5.8.3.39 TH2D ∗ tr_his_c::xyhist_nuclA

cartesian fixed-axes distribution of density in nucleus A

5.8.3.40 TH2D ∗ tr_his_c::xyhist_nuclB

cartesian fixed-axes distribution of density in nucleus B

5.8.3.41 TH2D ∗ tr_his_c::xyhistr

cartesian participant-plane distribution

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

- build/include/functions.h

Chapter 6

File Documentation

6.1 build/include/collision.h File Reference

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

Classes

- class collision
  collision class

6.1.1 Detailed Description

Part of GLISSANDO 3

6.2 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

6.2.1 Detailed Description

Part of GLISSANDO 3

6.3 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

6.3.1 Detailed Description

Part of GLISSANDO 3

6.4 build/include/functions.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

- Double_t fsNN (Double_t ecm)

  NN inelastic cross section.
- Double_t fsQQ (Double_t ecm)

  parton-parton inelastic cross section
- Double_t fqscale (Double_t ecm)

  parton separation parameter
- Double_t fgama (Double_t ecm)
- Double_t fomega (Double_t ecm)
- void readpar (TString inpfile)

  process the input file containing 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 console output
- void epilog ()

  print epilog to the output
- Int_t time_start ()

  start the time measurement and print the stamp
- void time_stop (Int_t ts)

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

  random number generator using the built-in ROOT generator, uniform on (0,1)
- Float_t los32 ()

  random number generator for spin (3/2,3/2) projection
- Float_t los12 ()

  random number generator for spin (3/2,1/2) projection
- 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)

  random number generator for the deformed Woods-Saxon or Fermi distribution
- Float_t rlosB_def (Float_t ∗cth_pointerB, Float_t beta2, Float_t beta4)

  random number generator for the deformed Woods-Saxon or Fermi distribution
- Float_t rlos_hole (Float_t s)

  random number generator for distribution with a hole in the middle
- Float_t rlos_alpha (Float_t s, Float_t bp, Float_t dp)

  random number generator for distribution in the alpha nucleus - not used
- Float_t rlos_abf (Float_t sc, Float_t scp)

  random number generator for distribution of nucleons in the alpha cluster
- 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
- Double_t gamgen (Float_t a)

  random number generator for the Gamma distribution
- Int_t negbin (Double_t m, Double_t v)

  random number generator for the negative binomial distribution
- Float_t rlos_parton ()

  random radial coordinate of a parton in the nucleon
- Float_t dist (Int_t m, Float_t u, Float_t v)

  statistial distribition overlaid on RDS
- Float_t disp (Float_t w)

  random shift of the source location

Variables

- Int_t PARTONS
- Int_t CLUSTERS
- Int_t ARGC
- Int_t EVENTS =1000

  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 NCS =3

  number of partons in the nucleon
- Int_t WMIN =2

  minimum number of wounded nucleons to record the event
- Int_t MODEL =0

  switch for the superimposed multiplicity distribution: 0 - scale, 1 - Poisson, 2 - Gamma, 3 - Negative Binomial
- 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 objects in the acceptance window
- Int_t W1 =10000

  maximum allowed number of wounded objects in the acceptance window
- Int_t SHIFT =1

  1 - shift the coordinates of the fireball to c.m., 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 wounding profile, 1 - Gaussian wounding profile, 2 - gamma 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 =-1.

  Woods-Saxon radius for nucleus A.
- Float_t AWSA =0.459

  Woods-Saxon width for nucleus A.
- Float_t BETA2A =-1

  Deformation parameter beta2 of deformed Woods-Saxon distribution for nucleus A.
- Float_t BETA4A =-1

  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 =-1.

  Woods-Saxon radius for nucleus B.
- Float_t AWSB =0.459

  Woods-Saxon width for nucleus B.
- Float_t BETA2B =-1

  Deformation parameter beta2 of deformed Woods-Saxon distribution for nucleus B.
- Float_t BETA4B =-1

  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 =-1

  NN "wounding" cross section in milibarns.
- Float_t SBIN =-1

  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 =1.

  Poisson or Gamma parameters for superimposed distribution, wounded nucleons.
- Float_t Ubin =1.

  Poisson or Gamma parameters for superimposed distribution, binary collisions.
- Float_t Vw =2.

  Negative binomial variance, wounded nucleons.
- Float_t Vbin =2.

  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 =1

  Gaussian wounding profile parameter (height at the origin)
- Float_t RAPRANGE =5.

  range in rapidity
- Float_t ETA0 =2.5

  2∗ETA0 is the width of the plateau in eta
- Float_t ETAM =8.58

  parameter of the Bialas-Czyz-Bozek model
- Float_t SIGETA =1.4

  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 =-1

  relative variance of cross-section fluctuations
- Float_t GAMA =-1

  gamma wounding profile parameter (height at the origin)
- Float_t SCALEA =1.

  scale parameter for the size of the nucleus (cluster version)
- Float_t SIGMAA =1.

  standard deviation of x,y,z coordinates of nucleons in the alpha cluster
- Float_t SIGMABISA =1.

  standard deviation of x,y,z coordinates of nucleons in the 3He cluster or nucleon no. 9
- Float_t SCALEB =1.

  scale parameter for the size of the nucleus (cluster version)
- Float_t SIGMAB =1.

  standard deviation of x,y,z coordinates of nucleons in the alpha cluster
- Float_t SIGMABISB =1.

  standard deviation of x,y,z coordinates of nucleons in the 3He cluster or nucleon no. 9
- Float_t QSCALE =-1

  scale parameter in the parton distribution function, f(r)=r^2∗exp(-r/QSCALE), QSCALE=1/(4.27/Sqrt(3/2))
- Float_t DS =-1

  source smearing parameter
- Float_t ECM =-1.

  center of mass energy of the colliding system [GeV]
- counter2 epart1

  counter for participant-plane <r^3 cos(phi)>/<r^3>

- counter2 epart

  counter for participant-plane <r^n cos(2 phi)>/<r^n>
- counter2 epart3

  counter for participant-plane <r^n cos(3 phi)>/<r^n>
- counter2 epart4

  counter for participant-plane <r^n cos(4 phi)>/<r^n>
- counter2 epart5

  counter for participant-plane <r^n cos(5 phi)>/<r^n>
- counter2 epart6

  counter for participant-plane <r^n cos(6 phi)>/<r^n>
- counter2 nwounded

  counter for number of wounded objects
- 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 objects in A
- Float_t rwB

  number of wounded objects in B
- Float_t rwAB

  number of all wounded objects
- 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 ep1s

  participant-plane eccentricities
- Float_t eps
- Float_t ep3s
- Float_t ep4s
- Float_t ep5s
- Float_t ep6s
- Float_t phirot1

  angles of the principal axis
- Float_t phirot
- Float_t phirot3
- Float_t phirot4
- Float_t phirot5
- Float_t phirot6
- Float_t xx

  center-of-mass x coordinate
- Float_t yy

  center-of-mass y coordinate
- Float_t xepp

  average ep
- Float_t xsepp

  standard deviation of ep
- Float_t wfqA

  number of wounded nucleons evaluated from the number of wounded partons in nucleus A
- Float_t wfqB

  number of wounded nucleons evaluated from the number of wounded partons in nucleus B
- Float_t wfq

  total number of wounded nucleons evaluated from the number of wounded partons

6.4.1 Detailed Description

Part of GLISSANDO 3

6.4.2 Function Documentation

6.4.2.1 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

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

statistial distribition overlaid on RDS

Parameters

m	case: 0 - none, 1 - Poisson, 2 - Gamma distribution, 3 - Negative binomial distribution
u	average of the distribution in cases 1, 2 and 3
v	variance of the distribution in case 3, v>u

6.4.2.3 void echopar ( )

echo parameters to the output

set the deformation parameters

6.4.2.4 void epilog ( )

print epilog to the output

6.4.2.5 Double_t fgama (Double_t *ecm*)

6.4.2.6 Double_t fomega (Double_t *ecm*)

6.4.2.7 Double_t fqscale (Double_t *ecm*)

parton separation parameter

parton-parton separation that (together with fsQQ) reproduces the NN inelastic cross section and the NN wounding profile

Parameters

ecm	sqrt(s_NN)

6.4.2.8 Double_t fsNN (Double_t *ecm*)

NN inelastic cross section.

NN inelastic cross section from our parametrization of the COMPETE model

Parameters

ecm	sqrt(s_NN)

6.4.2.9 Double_t fsQQ (Double_t *ecm*)

parton-parton inelastic cross section

parton-parton inelastic cross section that (together with fqscale) reproduces the NN inelastic cross section and the NN wounding profile

Parameters

ecm	sqrt(s_NN)

6.4.2.10 Double_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)

6.4.2.11 void header ( )

print the header in the console output

6.4.2.12 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)

6.4.2.13 Float_t los ( )

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

6.4.2.14 Float_t los12 ( )

random number generator for spin (3/2,1/2) projection

6.4.2.15 Float_t los32 ( )

random number generator for spin (3/2,3/2) projection

6.4.2.16 Int_t negbin (Double_t *m*, Double_t *v*)

random number generator for the negative binomial distribution

6.4.2.17 void readpar (TString *inpfile*)

process the input file containing parameters

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

reset number of constituents to 1 in the wounded nucleon case

correct wrong input

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

set the range for the histograms

Parameters

inpfile	name of the input file

6.4.2.18 void reset_counters ( )

reset the counters used to store physical quantities in the event

6.4.2.19 Float_t rlos_abf (Float_t *sc*, Float_t *scp*)

random number generator for distribution of nucleons in the alpha cluster

Parameters

sc	scale parameter
scp	scale parameter

6.4.2.20 Float_t rlos_alpha (Float_t *s*, Float_t *bp*, Float_t *dp*)

random number generator for distribution in the alpha nucleus - not used

6.4.2.21 Float_t rlos_hole (Float_t *s*)

random number generator for distribution with a hole in the middle

Parameters

s	size scale parameter

6.4.2.22 Float_t rlos_hult ( )

random number generator for the Hulthen distribution

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

6.4.2.23 Float_t rlos_parton ( )

random radial coordinate of a parton in the nucleon

gamma distribution f(r)=r^2*exp(-r/wid)

6.4.2.24 Float_t rlosA ( )

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

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

random number generator for the deformed Woods-Saxon or Fermi distribution

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

Parameters

cth_pointerA	cos(theta)
beta2	beta_2 nuclear deformation parameter
beta4	beta_4 nuclear deformation parameter

6.4.2.26 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.)

6.4.2.27 Float_t rlosB ( )

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

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

random number generator for the deformed Woods-Saxon or Fermi distribution

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

Parameters

cth_pointerB	cos(theta)
beta2	beta_2 nuclear deformation parameter
beta4	beta_4 nuclear deformation parameter

6.4.2.29 Float_t rlosB_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.)

6.4.2.30 Int_t time_start ( )

start the time measurement and print the stamp

6.4.2.31 void time_stop (Int_t *ts*)

stop the time measurement and print the stamp

Parameters

ts	time at start

6.4.3 Variable Documentation

6.4.3.1 Float_t ALPHA =0.15

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

6.4.3.2 counter_2D angles

counter for forward-backward reaction-plane angle correlations

6.4.3.3 Int_t ARANK =2

rank of the Fourier moment for the forward-backward analysis

6.4.3.4 Int_t ARGC

6.4.3.5 Float_t AWSA =0.459

Woods-Saxon width for nucleus A.

6.4.3.6 Float_t AWSB =0.459

Woods-Saxon width for nucleus B.

6.4.3.7 Float_t b

impact parameter

6.4.3.8 Float_t BETA2A =-1

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

6.4.3.9 Float_t BETA2B =-1

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

6.4.3.10 Float_t BETA4A =-1

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

6.4.3.11 Float_t BETA4B =-1

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

6.4.3.12 Float_t BMAX =25.

maximum value of the impact parameter in the acceptance window

6.4.3.13 Float_t BMIN =0.

minimum value of the impact parameter in the acceptance window

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

maximum coordinate value for some histograms

6.4.3.15 Float_t CD =0.9

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

6.4.3.16 Int_t CLUSTERS

6.4.3.17 Float_t d

the wounding distance

6.4.3.18 Float_t DBIN =0.

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

6.4.3.19 Float_t dbin

the binary-collision distance

6.4.3.20 Int_t DOBIN =0

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

6.4.3.21 Float_t DS =-1

source smearing parameter

6.4.3.22 Float_t DW =0.

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

6.4.3.23 Float_t ECM =-1.

center of mass energy of the colliding system [GeV]

6.4.3.24 Float_t ep1s

participant-plane eccentricities

6.4.3.25 Float_t ep3s

6.4.3.26 Float_t ep4s

6.4.3.27 Float_t ep5s

6.4.3.28 Float_t ep6s

6.4.3.29 counter2 epart

counter for participant-plane <r^n cos(2 phi)>/<r^n>

6.4.3.30 counter2 epart1

counter for participant-plane <r^3 cos(phi)>/<r^3>

6.4.3.31 counter2 epart3

counter for participant-plane <r^n cos(3 phi)>/<r^n>

6.4.3.32 counter2 epart4

counter for participant-plane <r^n cos(4 phi)>/<r^n>

6.4.3.33 counter2 epart5

counter for participant-plane <r^n cos(5 phi)>/<r^n>

6.4.3.34 counter2 epart6

counter for participant-plane <r^n cos(6 phi)>/<r^n>

6.4.3.35 Float_t eps

6.4.3.36 Float_t ETA0 =2.5

2*ETA0 is the width of the plateau in eta

6.4.3.37 Float_t ETAM =8.58

parameter of the Bialas-Czyz-Bozek model

6.4.3.38 Int_t evall

number of all attempted event

6.4.3.39 Int_t EVENTS =1000

number of generated events

6.4.3.40 Int_t FBIN =72

number of bins for histogramming in the azimuthal angle

6.4.3.41 Float_t FBRAP =2.5

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

6.4.3.42 Int_t FILES =0

1 - read distribution from files, 0 - do not

6.4.3.43 Int_t FULL =0

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

6.4.3.44 Float_t GA =1

Gaussian wounding profile parameter (height at the origin)

6.4.3.45 Float_t GAMA =-1

gamma wounding profile parameter (height at the origin)

6.4.3.46 UInt_t ISEED

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

6.4.3.47 UInt_t ISEED1

copy of ISEED

6.4.3.48 Int_t kk

number of the current event

6.4.3.49 Float_t MAXYRAP =10.

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

6.4.3.50 Int_t MODEL =0

switch for the superimposed multiplicity distribution: 0 - scale, 1 - Poisson, 2 - Gamma, 3 - Negative Binomial

6.4.3.51 Int_t NBIN =40

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

6.4.3.52 counter2 nbinary

counter for number of binary collisions

6.4.3.53 Int_t NCS =3

number of partons in the nucleon

6.4.3.54 counter2 nhot

counter for number of hot-spots

6.4.3.55 Int_t NNWP =0

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

6.4.3.56 Int_t NUMA =208

mass number of nucleus A

6.4.3.57 Int_t NUMB =208

mass number of nucleus B

6.4.3.58 Int_t NUMRAP =10

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

6.4.3.59 counter2 nweight

counter for relative deposited strength (RDS)

6.4.3.60 counter2 nwounded

counter for number of wounded objects

6.4.3.61 Float_t OMEGA =-1

relative variance of cross-section fluctuations

6.4.3.62 Int_t PARTONS

6.4.3.63 Float_t phirot

6.4.3.64 Float_t phirot1

angles of the principal axis

6.4.3.65 Float_t phirot3

6.4.3.66 Float_t phirot4

6.4.3.67 Float_t phirot5

6.4.3.68 Float_t phirot6

6.4.3.69 Float_t PI =4.∗atan(1.)

the number pi

6.4.3.70 Int_t PP =-1

power of the transverse radius in the Fourier moments

6.4.3.71 Int_t ppp

power of the weight in eccentricty definition

6.4.3.72 Float_t QSCALE =-1

scale parameter in the parton distribution function, f(r)=r^2∗exp(-r/QSCALE), QSCALE=1/(4.27/Sqrt(3/2))

6.4.3.73 Float_t RAPRANGE =5.

range in rapidity

6.4.3.74 Float_t rbin

number of binary collisions

6.4.3.75 Float_t RCHA =5.66

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

6.4.3.76 Float_t RCHB =5.66

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

6.4.3.77 Float_t RCHP =0.7714

harmonic oscillator shell model density mean squared charge radii of proton

6.4.3.78 Float_t RDS0 =0.

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

6.4.3.79 Float_t RDS1 =100000

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

6.4.3.80 Int_t RET =0

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

6.4.3.81 Float_t rhotspot

number of hot-spots

6.4.3.82 Int_t RO =0

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

6.4.3.83 Int_t roo

Fourier rank for the rotation axis.

6.4.3.84 Float_t ROTA_PHI =-1.0

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

6.4.3.85 Float_t ROTA_THETA =-1.0

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

6.4.3.86 Float_t ROTB_PHI =-1.0

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

6.4.3.87 Float_t ROTB_THETA =-1.0

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

6.4.3.88 Float_t rpa

relative deposited strength (RDS)

6.4.3.89 Float_t rwA

number of wounded objects in A

6.4.3.90 Float_t rwAB

number of all wounded objects

6.4.3.91 Float_t rwB

number of wounded objects in B

6.4.3.92 Float_t RWSA =-1.

Woods-Saxon radius for nucleus A.

6.4.3.93 Float_t RWSB =-1.

Woods-Saxon radius for nucleus B.

6.4.3.94 Float_t SBIN =-1

NN binary cross section in milibarns.

6.4.3.95 Float_t SCALEA =1.

scale parameter for the size of the nucleus (cluster version)

6.4.3.96 Float_t SCALEB =1.

scale parameter for the size of the nucleus (cluster version)

6.4.3.97 Int_t SHIFT =1

1 - shift the coordinates of the fireball to c.m., 0 - do not

6.4.3.98 Float_t SIGETA =1.4

parameter controlling the width of the rapidity distribution

6.4.3.99 Float_t SIGMAA =1.

standard deviation of x,y,z coordinates of nucleons in the alpha cluster

6.4.3.100 Float_t SIGMAB =1.

standard deviation of x,y,z coordinates of nucleons in the alpha cluster

6.4.3.101 Float_t SIGMABISA =1.

standard deviation of x,y,z coordinates of nucleons in the 3He cluster or nucleon no. 9

6.4.3.102 Float_t SIGMABISB =1.

standard deviation of x,y,z coordinates of nucleons in the 3He cluster or nucleon no. 9

6.4.3.103 Float_t sirad

equivalent hard-sphere radius for the cross section

6.4.3.104 Float_t sitot

the total A+B cross section in the acceptance window

6.4.3.105 Float_t sizeav

size

6.4.3.106 Float_t SNN =-1

NN "wounding" cross section in milibarns.

6.4.3.107 Float_t Ubin =1.

Poisson or Gamma parameters for superimposed distribution, binary collisions.

6.4.3.108 Float_t Uw =1.

Poisson or Gamma parameters for superimposed distribution, wounded nucleons.

6.4.3.109 Float_t Vbin =2.

Negative binomial variance, binary collisions.

6.4.3.110 Float_t Vw =2.

Negative binomial variance, wounded nucleons.

6.4.3.111 Int_t W0 =2

minimum allowed number of wounded objects in the acceptance window

6.4.3.112 Int_t W1 =10000

maximum allowed number of wounded objects in the acceptance window

6.4.3.113 Float_t WFA =0.

the w parameter for the Fermi distribution for nucleus A

6.4.3.114 Float_t WFB =0.

the w parameter for the Fermi distribution for nucleus B

6.4.3.115 Float_t wfq

total number of wounded nucleons evaluated from the number of wounded partons

6.4.3.116 Float_t wfqA

number of wounded nucleons evaluated from the number of wounded partons in nucleus A

6.4.3.117 Float_t wfqB

number of wounded nucleons evaluated from the number of wounded partons in nucleus B

6.4.3.118 Int_t WMIN =2

minimum number of wounded nucleons to record the event

6.4.3.119 Float_t xepp

average ep

6.4.3.120 Float_t xsepp

standard deviation of ep

6.4.3.121 Float_t xx

center-of-mass x coordinate

6.4.3.122 Float_t yy

center-of-mass y coordinate

6.5 build/src/glissando3.cxx File Reference

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

Macros

- #define _VER_ 3.111

Functions

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

  the main function of GLISSANDO 3

Variables

- Float_t ver = _VER_

  current version of the code
- TRandom3 raa

  ROOT random number generator.

6.5.1 Detailed Description

The main file of GLISSANDO 3

6.5.2 Macro Definition Documentation

6.5.2.1 #define _VER_ 3.111

6.5.3 Function Documentation

6.5.3.1 Int_t main (Int_t *argc*, char ∗ *argv[]*)

the main function of GLISSANDO 3

The main function of GLISSANDO 3 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*, etc.). (the units for all dimensionful quantities in GLISSANDO 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 from other sources the third command line argument is the file for nucleus A and the fourth (optional) for nucleus B - if absent, B is generated randomly in Glissando3. Format of reading is adjusted to the format of files, which is different for A=3 (helium, tritium) and A>3.

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 the nuclei by theta (zx plane) and phi (xy plane) angles The proper order of rotations is important

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 objects, binary collisions) and RDS

#if(*weight*) generate the histograms for the NN collision profiles

generate various 2-dim histograms with the distributions of sources

generate the participant-plane 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 some final results to the console 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

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> (present only when *files*=1) fourth argument: <nucl_B.dat> (esent pronly when *files*=1) command line parameters (file names)

6.5.4 Variable Documentation

6.5.4.1 TRandom3 raa

ROOT random number generator.

6.5.4.2 Float_t ver = _VER_

current version of the code

6.6 macro/demo_2/centrality2.C File Reference

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

Functions

- void centrality2 (char *p)

  generates the centrality classes

6.6.1 Detailed Description

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

6.6.2 Function Documentation

6.6.2.1 void centrality2 (char * *p*)

generates 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

6.7 macro/demo_2/core_mantle.C File Reference

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

Functions

- void core_mantle (char *p)

    generates the core and corona distributions

6.7.1 Detailed Description

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

6.7.2 Function Documentation

6.7.2.1 void core_mantle (char * *p*)

generates the core and corona distributions

Parameters

p	name of the ROOT input file

6.8 macro/demo_2/corr.C File Reference

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

Functions

- void corr (char *p)

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

6.8.1 Detailed Description

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

6.8.2 Function Documentation

6.8.2.1 void corr (char * *p*)

generates 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

6.9 macro/demo_2/density.C File Reference

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

Functions

- void density (char ∗p)

  produces plots of the fixed-axes and participant-plane densities

6.9.1 Detailed Description

Script generating the fixed- and participant-plane densities (part of GLISSANDO 2)

6.9.2 Function Documentation

6.9.2.1 void density (char ∗ *p*)

produces plots of the fixed-axes and participant-plane densities

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

Parameters

p	name of the ROOT input file

6.10 macro/demo_2/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

6.10.1 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)

6.10.2 Function Documentation

6.10.2.1 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

Produces 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

6.11 macro/demo_2/fourier.C File Reference

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

Functions

- void fourier (char *p)

  generates the plot of epsilon_n^* vs. Nw, n=1,2,3,4,5,6

6.11.1 Detailed Description

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

Script generating the epsilon_n vs. Q_w plot (part of GLISSANDO 3)

6.11.2 Function Documentation

6.11.2.1 void fourier (char * p)

generates the plot of epsilon_n^* vs. Nw, n=1,2,3,4,5,6

Parameters

p	name of the ROOT input file

6.12 macro/demo_2/info.C File Reference

Functions

- void info (char *p)

  prints info on the stored GLISSANDO 2 ROOT file

6.12.1 Detailed Description

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

6.12.2 Function Documentation

6.12.2.1 void info (char * p)

prints info on the stored GLISSANDO 2 ROOT file

Prints 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

6.13 macro/demo_2/label.C File Reference

Functions

- void label (char *inpfile)

  generates 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

6.13.1 Function Documentation

6.13.1.1 void label (char * *inpfile*)

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

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

Parameters

inpfile	name of the ROOT input file

6.13.1.2 void label_fit (char * *inpfile*)

generate the label for plots referring to distributions within the nucleus

Parameters

inpfile	name of the ROOT input file

6.14 macro/demo_3/label.C File Reference

Functions

- void label (char *inpfile)

  generates 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

6.14.1 Function Documentation

6.14.1.1 void label (char * *inpfile*)

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

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

Parameters

inpfile	name of the ROOT input file

6.14.1.2 void label_fit (char * *inpfile*)

generate the label for plots referring to distributions within the nucleus

Parameters

inpfile	name of the ROOT input file

6.15 macro/demo_2/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

6.15.1 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)

6.15.2 Function Documentation

6.15.2.1 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

6.16 macro/demo_2/profile2_deformation_63Cu.C File Reference

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

Functions

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

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

6.16.1 Detailed Description

Script generating r-cos(theta) distributions of the nucleons in the nuleus (part of GLISSANDO 2)

6.16.2 Function Documentation

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

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

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

Parameters

p	name of the GLISSANDO input ROOT for the deformed case
ps	name of the GLISSANDO input ROOT for the spherical case

6.17 macro/demo_2/profile2_deformation_U.C File Reference

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

Functions

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

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

6.17.1 Detailed Description

Script generating r-cos(theta) distributions of the nucleons in the nuleus (part of GLISSANDO 2)

6.17.2 Function Documentation

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

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

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

Parameters

p	name of the GLISSANDO input ROOT for the deformed case
ps	name of the GLISSANDO input ROOT for the spherical case

6.18 macro/demo_2/size.C File Reference

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

Functions

- void size (char *p)

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

6.18.1 Detailed Description

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

6.18.2 Function Documentation

6.18.2.1 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

6.19 macro/demo_2/wounding_profile.C File Reference

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

Functions

- void wounding_profile (char *p)

  generates the plots of the wounding and binary-collision profiles

6.19.1 Detailed Description

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

Script generating the wounding (inelasticity) profile (part of GLISSANDO 3)

6.19.2 Function Documentation

6.19.2.1 void wounding_profile (char * *p*)

generates the plots of the wounding and binary-collision profiles

Parameters

p	name of the ROOT input file

6.20 macro/demo_3/b_dist.C File Reference

Functions

- void b_dist (char ∗p)

6.20.1 Function Documentation

6.20.1.1 void b_dist (char ∗ *p*)

Parameters

p	name of the ROOT input file

6.21 macro/demo_3/b_distr.C File Reference

```
#include "TCanvas.h"
#include "TStyle.h"
#include "TH1.h"
```

Functions

- void b_distr (char ∗p)

6.21.1 Function Documentation

6.21.1.1 void b_distr (char ∗ *p*)

Parameters

p	name of the ROOT input file

6.22 macro/demo_3/cent.C File Reference

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

Functions

- void cent (char ∗p)

  generates the centrality distribution of wounded objects

6.22.1 Detailed Description

Script generating the centrality distribution of wounded objects (part of GLISSANDO 3)

6.22.2 Function Documentation

6.22.2.1 void cent (char ∗ *p*)

generates the centrality distribution of wounded objects

generates the centrality distribution of wounded objects (nucleons or quarks)

Parameters

p	name of the ROOT input file

6.23 macro/demo_3/cent_comp.C File Reference

Functions

- void cent_comp (char ∗p_qq, char ∗p_NN)
  generates the centrality distribution of wounded objects

6.23.1 Function Documentation

6.23.1.1 void cent_comp (char ∗ *p_qq*, char ∗ *p_NN*)

generates the centrality distribution of wounded objects

generates the centrality distribution of wounded objects (nucleons or quarks)

Parameters

p_qq	name of the ROOT input file

6.24 macro/demo_3/clusters.C File Reference

```
#include "TCanvas.h"
#include "TStyle.h"
#include "TH1.h"
```

Functions

- void clusters (char ∗p)

6.24.1 Function Documentation

6.24.1.1 void clusters (char ∗ *p*)

Parameters

p	name of the ROOT input file

6.25 macro/demo_3/eps_dist.C File Reference

Functions

- void eps_dist (char ∗p)

6.25.1 Function Documentation

6.25.1.1 void eps_dist (char ∗ *p*)

Parameters

p	name of the ROOT input file

6.26 macro/demo_3/epsilon_c.C File Reference

Functions

- void epsilon_c (char ∗p)

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

6.26.1 Detailed Description

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

6.26.2 Function Documentation

6.26.2.1 void epsilon_c (char ∗ *p*)

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

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

Parameters

p	name of the ROOT input file

6.27 macro/demo_3/fourier_Q.C File Reference

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

Functions

- void fourier_Q (char ∗p, Int_t up)
  generates the plot of epsilon_n vs. Qw, n=2,3

6.27.1 Function Documentation

6.27.1.1 void fourier_Q (char ∗ *p*, Int_t *up*)

generates the plot of epsilon_n vs. Qw, n=2,3

Parameters

p	name of the ROOT input file
up	x-range of the plot

6.28 macro/demo_3/inel_prof.C File Reference

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

Functions

- void inel_prof (char ∗p)
  generates the plots of the wounding profile

6.28.1 Function Documentation

6.28.1.1 void inel_prof (char ∗ *p*)

generates the plots of the wounding profile

Parameters

p	name of the ROOT input file

6.29 macro/demo_3/mult_pPb_Q.C File Reference

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

Functions

- void mult_pPb_Q (char ∗p)
  generates the plot of epsilon_n vs. Qw, n=2,3

6.29.1 Function Documentation

6.29.1.1 void mult_pPb_Q (char ∗ *p*)

generates the plot of epsilon_n vs. Qw, n=2,3

Parameters

p	name of the ROOT input file

6.30 macro/demo_3/mult_Q.C File Reference

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

Functions

- void mult_Q (char ∗p)

  generates the plot of epsilon_n vs. Qw, n=2,3

6.30.1 Function Documentation

6.30.1.1 void mult_Q (char ∗ *p*)

generates the plot of epsilon_n vs. Qw, n=2,3

Parameters

p	name of the ROOT input file

6.31 macro/demo_3/part_dist_pp.C File Reference

Functions

- void part_dist_pp (char ∗p)

6.31.1 Function Documentation

6.31.1.1 void part_dist_pp (char ∗ *p*)

Parameters

p	name of the ROOT input file

6.32 macro/demo_3/rad_dis.C File Reference

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

Functions

- void rad_dis (char ∗p)

  radial distribution of nucleons or partons

6.32.1 Detailed Description

Script yielding the radial distribution of nucleons or partons (part of GLISSANDO 3)

6.32.2 Function Documentation

6.32.2.1 void rad_dis (char ∗ *p*)

radial distribution of nucleons or partons

for *profile*=0 - nucleons, for *profile*=1 - partons

Parameters

p	name of the ROOT input file

6.33 macro/demo_3/sc_comp.C File Reference

Functions

- void sc_comp (char ∗p_clust, char ∗p_unif)

  generates the plot of the symmetric cumulant SC(2,3)

6.33.1 Detailed Description

(part of GLISSANDO 3)

6.33.2 Function Documentation

6.33.2.1 void sc_comp (char ∗ *p_clust,* char ∗ *p_unif*)

generates the plot of the symmetric cumulant SC(2,3)

$<$ names of the ROOT input files

determination of centrality bins

determination of centrality bins

6.34 macro/demo_3/w_prof_norm_AA.C File Reference

```
#include "TCanvas.h"
#include "TStyle.h"
#include "TH1.h"
```

Functions

- void w_prof_norm_AA (char ∗p_qq, char ∗p_NN)

6.34.1 Function Documentation

6.34.1.1 void w_prof_norm_AA (char * *p_qq*, char * *p_NN*)

Parameters

p_qq	name of the ROOT input file

6.35 macro/demo_3/w_prof_norm_pp.C File Reference

Functions

- Double_t nn (Double_t ∗x, Double_t ∗par)
- Float_t nn_1 (Double_t gama, Double_t omega, Float_t x)
- void w_prof_norm_pp (char ∗p)

6.35.1 Function Documentation

6.35.1.1 Double_t nn (Double_t ∗ *x*, Double_t ∗ *par*)

6.35.1.2 Float_t nn_1 (Double_t *gama*, Double_t *omega*, Float_t *x*)

6.35.1.3 void w_prof_norm_pp (char ∗ *p*)

Parameters

p	name of the ROOT input file

Index