blazar++

Generated by Doxygen 1.8.6

Sun Dec 14 2014 21:11:54

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

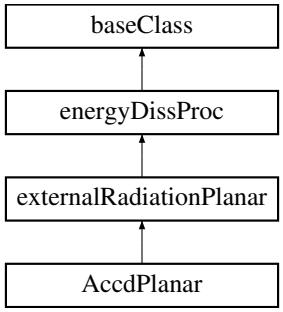
Chapter 4

Class Documentation

4.1 AccdPlanar Class Reference

`#include <AccdPlanar.hpp>`

Inheritance diagram for AccdPlanar:



4.1.1 Detailed Description

Parameters used by class (defaults)

Parameters

alpha	(externalRadiationPlanar) (alpha = -0.3333)
R1	- (externalRadiationPlanar) accd lower boundary radius ('NEW' stratification) (R1 = 0.1∗R_-sub)
R2	- (externalRadiationPlanar) accd upper boundary radius ('NEW' stratification) (R2 = R_sub)
s	- (externalRadiationPlanar) stratification index ('NEW' stratification) (s = 2.0)

Public Member Functions

- AccdPlanar (scfgp ∗_cfg, jetGeometry ∗_r, electrons ∗_ele, std::string _id)
- ∼AccdPlanar ()
- void printInfo ()
- void setRadius ()
- double getvext (double _R)
- void setdLdR ()

Additional Inherited Members

4.1.2 Constructor & Destructor Documentation

4.1.2.1 AccdPlanar::AccdPlanar (scfgp * *_cfg*, jetGeometry * *_r*, electrons * *_ele*, std::string *_id*)

constructor

Parameters

_cfg	- scfgp class object
_r	- jetGeometry class object
_ele	- electrons class object
_id	

```
9                                                                                  :
     externalRadiationPlanar( cfg, r, ele, id ) {
10   /* request parameters */
11   /* -- */
12
13   if( R1 == 0.0 || R2 == 0.0 ) { setRadius( ); }
14   else { bazinga::warning(id,"Overwritting R1 and R2 values from config file!"); }
15
16   vpMin = 0.01*DSQR( ele -> getGammaMin( ) )*getvext( R2 );
17   vpMax = 100.0*DSQR( ele -> getGammaMax( ) )*getvext( R1 );
18
19   /* initialize disk radius R;
20      here we use N-1 to use the same Upe as in other processes */
21   R = new logGeometry( R1, R2, N-1 );
22
23   allocateUpe( ); // this is for storing matrix information about dUpe(r)/dR (R)
24   allocateUpeR( ); // this is for storing vector information about Upe(r)
25   allocateLpv( );
26   allocateLvPoint( );
27   allocateLvPointAvg( );
28
29   for( int i=0;i<N;i++ ) { set_vp( i, vpMin*pow( vpMax/vpMin,(double)i/((double)N-1)) ); }
30
31   /* set the values of dLdR */
32   setdLdR( ); }
```

4.1.2.2 AccdPlanar::~AccdPlanar ( )

destructor

```
34                          {
35   freeUpe( );
36   freeLpv( );
37   freeLvPoint( );
38   freeLvPointAvg( ); }
```

4.1.3 Member Function Documentation

4.1.3.1 double AccdPlanar::getvext (double *_R*) [virtual]

get v_ext value in Hz

Parameters

_R	- radius in accretion disk plane

Returns

v_ext

Reimplemented from externalRadiationPlanar.

```
77                                      {
78   double Ledd = 1.3e47*mBH;
79   double Ldisk = eDisk*mDot*Ledd;
80   double vext = 0.0;
81   double Fdisk;
82   if( luminosityConstNu ) { Fdisk = (3.0*G_CONST*mBH*1e09*MSUN)*mDot*Ledd/( DSQR(
      LIGHT_SPEED)*8.0*M_PI*pow(R2,3.0) ); }
83   else { Fdisk = (3.0*G_CONST*mBH*1e09*MSUN)*mDot*Ledd/( DSQR(LIGHT_SPEED)*8.0*M_PI*pow(_R,3.0) ); }
84   double Teff = pow( Fdisk/SIGMA_SB, 0.25 );
85   vext = 3.92*Teff*K_BOLTZMAN/PLANCK_H;
86   return vext; }
```

4.1.3.2 void AccdPlanar::printInfo ( ) [virtual]

print basic information about myself (virtual)

Reimplemented from externalRadiationPlanar.

```
40                              {
41   bazinga::info(id,"Info");
42   bazinga::print_info(id,"N",N);
43   bazinga::info(id,"Using stratification version NEW (only option)");
44   bazinga::print_info(id,"radius R1",R1,"cm");
45   bazinga::print_info(id,"radius R2",R2,"cm");
46   bazinga::print_info(id,"stratification index",s);
47   bazinga::print_info(id,"alpha",alpha);
48   bazinga::print_info(id,"vpMin",vpMin,"Hz");
49   bazinga::print_info(id,"vpMax",vpMax,"Hz");
50   if( approx ) { bazinga::info(id,"Using approximate dependence on R"); }
51   if( luminosityConstU ) { bazinga::warning(id,"Using constant u' to calculate luminosity."
      ); }
52   if( luminosityConstNu ) { bazinga::warning(id,"Using constant v_ext to calculate
       luminosity."); }
53 }
```

4.1.3.3 void AccdPlanar::setdLdR ( ) [virtual]

set dL/dR for particulat source

Reimplemented from externalRadiationPlanar.

```
63                              {
64   double Ledd = 1.3e47*mBH;
65   double Ldisk = eDisk*mDot*Ledd;
66   /* fill in dLdR vector that will be used for the rest of the calculations */
67   bazinga::info(id,"Seeting dLdR.");
68   for( int i=0;i<R->getMaxIndex();i++ ) {
69       R->update( i );
70       double val = 0.0;
71       val = (3.0*G_CONST*mBH*1e09*MSUN)*mDot*Ledd*pow(R->get(),-s)/( 2.0*DSQR(LIGHT_SPEED) );
72       gsl_vector_set( dLdR, R->getIndex( ), val ); }
73   /* Now when we have all set let us save what we have just calculated - dLdR */
74   bazinga::info(id,"Saving dLdR.");
75   bazinga::save_GSLVector( "dLdR_"+this->whoAmI( ), R->getRadius_GSLVector( ), dLdR,
      cfg->get<std::string>("output") ); }
```

4.1.3.4 void AccdPlanar::setRadius ( ) [virtual]

sets rext if not provided by config file

Reimplemented from externalRadiationPlanar.

```
55                              {
56   double Ledd = 1.3e47*mBH;
57   double Ldisk = eDisk*mDot*Ledd;
58   double R_sub = 1.6e-5*sqrt( Ldisk );
59   double Rg = mBH*1e09*MSUN*G_CONST/DSQR(LIGHT_SPEED);
60   R1 = Rg/(2.0*eDisk);
61   R2 = R_sub; }
```

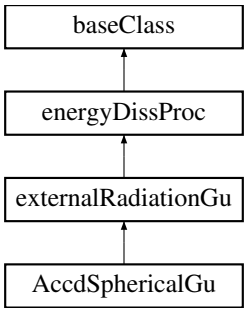
The documentation for this class was generated from the following files:

- /home/mjaniak/Soft/blazar++/include/AccdPlanar.hpp
- /home/mjaniak/Soft/blazar++/src/AccdPlanar.cpp

4.2 AccdSphericalGu Class Reference

`#include <AccdSphericalGu.hpp>`

Inheritance diagram for AccdSphericalGu:



4.2.1 Detailed Description

Parameters used by class (defaults)

Parameters

rext	- accretion disk inner radius

Public Member Functions

- AccdSphericalGu (scfgp ∗_cfg, jetGeometry ∗_r, electrons ∗_ele, std::string _id)
- ∼AccdSphericalGu ()
- void printInfo ()
- void update ()
- double getvext (double _r)
- void setRadius ()

Additional Inherited Members

4.2.2 Constructor & Destructor Documentation

4.2.2.1 AccdSphericalGu::AccdSphericalGu (scfgp ∗ *_cfg,* jetGeometry ∗ *_r,* electrons ∗ *_ele,* std::string *_id*)

constructor

Parameters

_cfg	- scfgp class object
_r	- jetGeometry class object
_ele	- electrons class object
_id	

```
9                                                                                :
      externalRadiationGu( cfg, r, ele, id ) {
```

```
10   /* requested parameters */
11 //  cfg -> request<double>(id + "e", 10.0, &e );
12 //  cfg -> request<double>(id + "cf", 0.1, &cf );
13   cfg -> request<double>(id + "rext", 0.0, &rext );
14 //  cfg -> request<double>(id + "k", 3.0, &k );
15 //
16   cfg -> updateRequests( );
17
18   if( rext == 0.0 ) { setRadius( ); }
19   else { bazinga::warning(id,"Overwritting rext value from config file!"); }
20
21   vpMin = 0.01*DSQR( ele -> getGammaMin( ) )*getvext( r->getRMax() );
22   vpMax = 100.0*DSQR( ele -> getGammaMax( ) )*getvext( r->getR0() );
23
24   allocateUpeR( ); // this is for storing vector information about Upe(r)
25   allocateLpv( );
26   allocateLvPoint( );
27   allocateLvPointAvg( );
28
29   for( int i=0;i<N;i++ ) { set_vp( i, vpMin*pow( vpMax/vpMin,(double)i/((double)N-1)) ); }
30 }
```

4.2.2.2 AccdSphericalGu::~AccdSphericalGu ( ) [inline]

destructor

```
33 { }
```

4.2.3 Member Function Documentation

4.2.3.1 double AccdSphericalGu::getvext (double _r) [virtual]

get v_ext value in Hz

Parameters

_r	- radius

Returns

v_ext

Reimplemented from externalRadiationGu.

```
32                                          {
33   double Ledd = 1.3e47*mBH;
34   double Ldisk = eDisk*mDot*Ledd;
35   double vext = 0.0;
36   double Fdisk;
37   if( luminosityConstNu ) { Fdisk = (3.0*G_CONST*mBH*1e09*MSUN)*mDot*Ledd/( DSQR(
      LIGHT_SPEED)*8.0*M_PI*pow(_r,3.0) ); }
38   else { Fdisk = (3.0*G_CONST*mBH*1e09*MSUN)*mDot*Ledd/( DSQR(LIGHT_SPEED)*8.0*M_PI*pow(_r,3.0) ); }
39   double Teff = pow( Fdisk/SIGMA_SB, 0.25 );
40   vext = 3.92*Teff*K_BOLTZMAN/PLANCK_H;
41   return vext; }
```

4.2.3.2 void AccdSphericalGu::printInfo ( ) [virtual]

print basic information about myself (virtual)

Reimplemented from externalRadiationGu.

```
55                                  {
56   bazinga::info(id,"Info");
57   bazinga::print_info(id,"N",N);
58   bazinga::print_info(id,"Avg energy (in external frame)",getvext( r->getR0())*PLANCK_H,"eV");
59   bazinga::print_info(id,"Avg frequency (in external frame)",getvext( r->getR0()),"Hz");
```

```
60   bazinga::print_info(id,"radius rext",rext,"cm");
61   bazinga::print_info(id,"vpMin",vpMin,"Hz");
62   bazinga::print_info(id,"vpMax",vpMax,"Hz");
63 }
```

4.2.3.3 void AccdSphericalGu::setRadius ( ) [virtual]

sets rext if not provided by config file

Reimplemented from externalRadiationGu.

```
65                                  {
66   double Ledd = 1.3e47*mBH;
67   double Ldisk = eDisk*mDot*Ledd;
68   double R_sub = 1.6e-5*sqrt( Ldisk );
69   double Rg = mBH*1e09*MSUN*G_CONST/DSQR(LIGHT_SPEED);
70   rext = Rg; }
```

4.2.3.4 void AccdSphericalGu::update ( ) [virtual]

calculate and set Upe every time with new radius r

Reimplemented from externalRadiationGu.

```
43                                  {
44   double Ledd = 1.3e47*mBH;
45   double Ldisk = eDisk*mDot*Ledd;
46   double temp_upe = 0.0;
47   double temp_cf = 0.0;
48   temp_cf = 0.75*(((0.28*rext)/r->get())+(1.0/(4.0*DSQR(Gamma)*DSQR(Gamma))));
49   temp_upe = (temp_cf*Ldisk*DSQR(Gamma))/(3.0*M_PI*DSQR(r->get())*LIGHT_SPEED);
50   gsl_vector_set( upe_r, r->getIndex( ), temp_upe );
51   bazinga::print_info(id,"Upe",gsl_vector_get( upe_r, r->getIndex( ) ) );
52   set_upe_r( gsl_vector_get( upe_r, r->getIndex( ) ) );
53   flag_upe_r = false; }
```

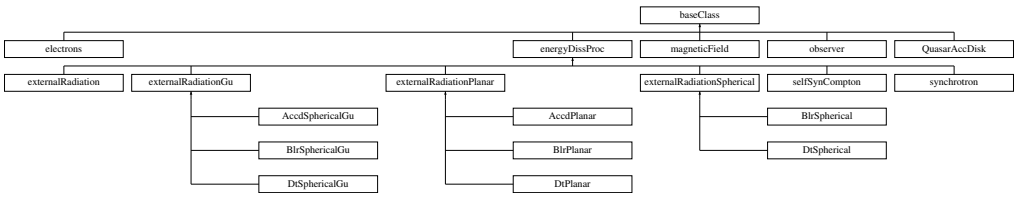
The documentation for this class was generated from the following files:

- /home/mjaniak/Soft/blazar++/include/AccdSphericalGu.hpp
- /home/mjaniak/Soft/blazar++/src/AccdSphericalGu.cpp

4.3 baseClass Class Reference

`#include <baseClass.hpp>`

Inheritance diagram for baseClass:



4.3.1 Detailed Description

Base class for all other classes used in blazar++. Provides unified way of sharing scfgp, id and jetGeometry

Public Member Functions

- **baseClass** (scfgp *cfg, jetGeometry *r, std::string id)
- std::string whoAmI ()
- virtual void printInfo ()
- double beta (double x)
- int getN ()

Public Attributes

- std::string id
- scfgp * cfg
- jetGeometry * r
- int N

4.3.2 Member Function Documentation

4.3.2.1 double baseClass::beta (double *x*)

get beta from Lorentz factor ; it shouldn't probably be here

```
16 { return sqrt(1.0-1.0/(x*x)); }
```

4.3.2.2 int baseClass::getN ( ) [inline]

get N

```
51 { return N; }
```

4.3.2.3 virtual void baseClass::printInfo ( ) [inline], [virtual]

print basic information about myself (virtual)

Reimplemented in electrons, energyDissProc, observer, externalRadiationPlanar, magneticField, QuasarAccDisk, DtSpherical, BlrPlanar, synchrotron, BlrSpherical, externalRadiationSpherical, DtPlanar, selfSynCompton, AccdPlanar, externalRadiationGu, BlrSphericalGu, DtSphericalGu, AccdSphericalGu, and externalRadiation.

```
45 { };
```

4.3.2.4 std::string baseClass::whoAmI ( )

tell me what is my id

```
14 { return id; }
```

4.3.3 Member Data Documentation

4.3.3.1 scfgp* baseClass::cfg

every class needs to have an information about parameters available - this is realized by attaching a pointer to scfgp class

4.3.3.2 std::string baseClass::id

id characterizing object

4.3.3.3 int baseClass::N

vectors size - since this is often used will be copied from model at the beginning

4.3.3.4 jetGeometry* baseClass::r

every class needs to have an information of current radius and overal geomtery - this is realized by attaching a pointer to radius class

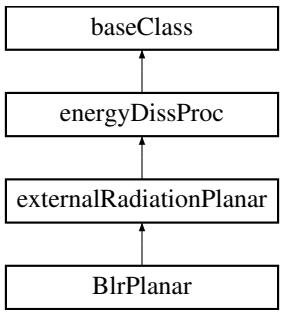
The documentation for this class was generated from the following files:

- /home/mjaniak/Soft/blazar++/include/baseClass.hpp
- /home/mjaniak/Soft/blazar++/src/baseClass.cpp

4.4 BlrPlanar Class Reference

`#include <BlrPlanar.hpp>`

Inheritance diagram for BlrPlanar:



4.4.1 Detailed Description

Parameters used by class (defaults)

Parameters

alpha	(externalRadiationPlanar) (alpha = 0.0)
R1	- (externalRadiationPlanar) blr lower boundary radius ('NEW' stratification) (R1 = 0.1*R_sub)
R2	- (externalRadiationPlanar) blr upper boundary radius ('NEW' stratification) (R2 = R_sub)
s	- (externalRadiationPlanar) stratification index ('NEW' stratification) (s = 2.0)
e	- energy (ext) in eV (e = 10.0)
CF	- covering factor (CF = 0.1)
q1	- lower index in stratification ('M03' stratification) (q1 = 1.0)
q2	- upper index in stratification ('M03' stratification) (q2 = 1.0)
Rext	- blr upper radius ('M03' stratification) (R2 = 0.1*R_sub)
strat	- sets stratification ('NEW' or 'M03') (strat = 'NEW')

Parameters used by class (defaults)

Parameters

alpha	(externalRadiationPlanar) (alpha = 0.0)
R1	- (externalRadiationPlanar) dt lower boundary radius (R1 = R_sub)
R2	- (externalRadiationPlanar) dt upper boundary radius (R2 = 10.0∗R_sub)
s	- (externalRadiationPlanar) stratification index (s = 1.0)
e	- energy (ext) in eV (e = 0.6203 @ Rext ∼ 1.5e14 Hz)
CF	- covering factor (CF = 0.1)

Public Member Functions

- BlrPlanar (scfgp ∗_cfg, jetGeometry ∗_r, electrons ∗_ele, std::string _id)
- ∼BlrPlanar ()
- void printInfo ()
- void setRadius ()
- double getvext (double _R)
- void setdLdR ()

Additional Inherited Members

4.4.2 Constructor & Destructor Documentation

4.4.2.1 BlrPlanar::BlrPlanar (scfgp ∗ *_cfg,* jetGeometry ∗ *_r,* electrons ∗ *_ele,* std::string *_id*)

constructor

Parameters

_cfg	- scfgp class object
_r	- jetGeometry class object
_ele	- electrons class object
_id	

```
9                                                                                    :
     externalRadiationPlanar( cfg, r, ele, id ) {
10   /* request parameters */
11   cfg -> request<double>( id+"e", 10.0, &e );
12   cfg -> request<double>( id+"cf", 0.1, &cf );
13   cfg -> request<double>( id+"q1", 1.0, &q1 );
14   cfg -> request<double>( id+"q2", 1.0, &q2 );
15   cfg -> request<double>( id+"R", 0.0, &Rext );
16   cfg -> request<std::string>( id+"Strat", "NEW", &strat );
17
18   cfg -> updateRequests( );
19
20   if( strat == "NEW" ) {
21     if( R1 == 0.0 || R2 == 0.0 ) { setRadius( ); }
22     else { bazinga::warning(id,"Overwritting R1 and R2 values from config file!"); }
23   }
24   else if( strat == "M03" ) {
25     if( Rext == 0.0 ) { setRadius( ); }
26     else { bazinga::warning(id,"Overwritting Rext value from config file!"); }
27   }
28
29   vpMin = 0.01*DSQR( ele -> getGammaMin( ) )*getvext( Rext );
30   vpMax = 100.0*DSQR( ele -> getGammaMax( ) )*getvext( Rext );
31
32   /* initialize disk radius R;
33      here we use N-1 to use the same Upe as in other processes */
34   R = new logGeometry( R1, R2, N-1 );
35
36   allocateUpe( ); // this is for storing matrix information about dUpe(r)/dR (R)
37   allocateUpeR( ); // this is for storing vector information about Upe(r)
38   allocateLpv( );
39   allocateLvPoint( );
40   allocateLvPointAvg( );
41
```

```
42   for( int i=0;i<N;i++ ) { set_vp( i, vpMin*pow( vpMax/vpMin,(double)i/((double)N-1)) ); }
43
44   /* set the values of dLdR */
45   setdLdR( ); }
```

4.4.2.2 BlrPlanar::~BlrPlanar ( )

destructor

```
47                          {
48   freeUpe( );
49   freeLpv( );
50   freeLvPoint( );
51   freeLvPointAvg( ); }
```

4.4.3 Member Function Documentation

4.4.3.1 double BlrPlanar::getvext (double _R) [virtual]

get v_ext value in Hz

Parameters

_R	- radius in accretion disk plane

Returns

v_ext

Reimplemented from externalRadiationPlanar.

```
115                                     {
116   vext = e*eV2erg/PLANCK_H;
117   return vext; }
```

4.4.3.2 void BlrPlanar::printInfo ( ) [virtual]

print basic information about myself (virtual)

Reimplemented from externalRadiationPlanar.

```
53                          {
54   bazinga::info(id,"Info");
55   bazinga::print_info(id,"N",N);
56   bazinga::print_info(id,"Using stratification version",strat);
57   if( strat == "NEW" ) {
58     bazinga::print_info(id,"radius R1",R1,"cm");
59     bazinga::print_info(id,"radius R2",R2,"cm");
60     bazinga::print_info(id,"stratification index",s); }
61   if( strat == "M03" ) {
62     bazinga::print_info(id,"radius Rext",Rext,"cm");
63     bazinga::print_info(id,"radius R1",R1,"cm");
64     bazinga::print_info(id,"radius R2",R2,"cm");
65     bazinga::print_info(id,"stratfication index q1",q1);
66     bazinga::print_info(id,"stratfication index q2",q2); }
67
68   bazinga::print_info(id,"Avg energy (in external frame)",e,"eV");
69   bazinga::print_info(id,"Avg frequency (in external frame)",getvext( Rext ),"Hz");
70   bazinga::print_info(id,"Clouds covering factor cf",cf);
71   bazinga::print_info(id,"alpha",alpha);
72   bazinga::print_info(id,"vpMin",vpMin,"Hz");
73   bazinga::print_info(id,"vpMax",vpMax,"Hz");
74   if( approx ) { bazinga::print_info(id,"Using approximate dependence on R"); }
75   if( luminosityConstU ) { bazinga::warning(id,"Using constant u' to calculate luminosity."
       ); }
76   if( luminosityConstNu ) { bazinga::warning(id,"Using constant v_ext to calculate
       luminosity."); }
77 }
```

4.4.3.3 void BlrPlanar::setdLdR ( ) [virtual]

set dL/dR for particulat source

Reimplemented from externalRadiationPlanar.

```
95                          {
96   double Ledd = 1.3e47*mBH;
97   double Ldisk = eDisk*mDot*Ledd;
98   /* fill in dLdR vector that will be used for the rest of the calculations */
99   bazinga::info(id,"Seeting dLdR.");
100   for( int i=0;i<R->getMaxIndex();i++ ) {
101     R -> update( i );
102     double val = 0.0;
103     if( strat == "NEW" ) { val = cf*Ldisk*ctau( )*pow(R->get( ),-s); }
104     if( strat == "M03" ) {
105       val = cf*Ldisk/(R->get()*(q1+q2)/(q1*q2));
106       if( R->get() <= Rext ) { val *= pow(R->get()/Rext,q1); }
107       if( R->get() > Rext ) { val *= pow(R->get()/Rext,-q2); }
108     }
109     gsl_vector_set( dLdR, R->getIndex( ), val );
110   }
111   /* Now when we have all set let us save what we have just calculated - dLdR */
112   bazinga::info(id,"Saving dLdR.");
113   bazinga::save_GSLVector( "dLdR_"+this->whoAmI( ), R->getRadius_GSLVector( ), dLdR,
      cfg->get<std::string>("output") ); }
```

4.4.3.4 void BlrPlanar::setRadius ( ) [virtual]

sets rext if not provided by config file

Reimplemented from externalRadiationPlanar.

```
79                            {
80   double Ledd = 1.3e47*mBH;
81   double Ldisk = eDisk*mDot*Ledd;
82   double R_sub = 1.6e-5*sqrt( Ldisk );
83   if( strat == "NEW" ) {
84     R1 = 0.1*R_sub;
85     R2 = R_sub;
86       /* Rext is set only to avoid 'if's in constructor when setting vpMIN and vpMAX; Rext is just set to
      R1 */
87     Rext = R1; }
88   else if( strat == "M03" ) {
89     Rext = 0.1*R_sub;
90     /* values R1 and r2 in M03 are only bounds for numerical reasons */
91     R1 = 1.0e-5*R_sub; // ???
92     R2 = 10.0*R_sub; }
93 }
```

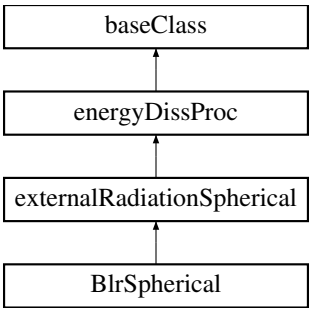
The documentation for this class was generated from the following files:

- /home/mjaniak/Soft/blazar++/include/BlrPlanar.hpp
- /home/mjaniak/Soft/blazar++/src/BlrPlanar.cpp

4.5 BlrSpherical Class Reference

`#include <BlrSpherical.hpp>`

Inheritance diagram for BlrSpherical:



4.5.1 Detailed Description

Parameters used by class (defaults)

Parameters

e	(externalRadiationSpherical) - energy (ext) in eV (e = 10.0)
cf	(externalRadiationSpherical) - covering factor (cf = 0.1)
q1	- lower index in stratification (q1 = 1.0)
q2	- upper index in stratification (q2 = 3.0)
rext	- blr radius (rext = 0.1∗r_sub)

Public Member Functions

- BlrSpherical (scfgp ∗_cfg, jetGeometry ∗_r, electrons ∗_ele, std::string _id)
- ∼BlrSpherical ()
- void update ()
- void printInfo ()
- void setRadius ()
- double getvext (double _r)

Additional Inherited Members

4.5.2 Constructor & Destructor Documentation

4.5.2.1 BlrSpherical::BlrSpherical (scfgp ∗ *_cfg,* jetGeometry ∗ *_r,* electrons ∗ *_ele,* std::string *_id*)

constructor

Parameters

_cfg	- scfgp class object
_r	- jetGeometry class object
_ele	- electrons class object
_id	

!!

```
9                                                                                :
     externalRadiationSpherical( cfg, r, ele, id ) {
10   /* request parameters */
11   cfg -> request<double>( id+"q1", 1.0, &q1 );
12   cfg -> request<double>( id+"q2", 1.0, &q2 );
13   cfg -> request<double>( id+"r", 0.0, &rext );
14   cfg -> request<double>("mBH",1.0,&mBH);
15   cfg -> request<double>("eDisk",0.1,&eDisk);
16   cfg -> request<double>("mDot",1.0,&mDot);
17   cfg -> request<double>("Gamma",10.0,&Gamma);
18
```

```
19   cfg -> updateRequests( );
20
21   if( rext == 0.0 ) { setRadius( ); }
22   else { bazinga::warning(id,"Overwritting rext value from config file!"); }
23
24   vpMin = 0.01*DSQR( ele -> getGammaMin() )*getvext( rext );
25   vpMax = 100.0*DSQR( ele -> getGammaMax() )*getvext( rext );
26
27   allocateUpe( );
28   allocateLpv( );
29   allocateLvPoint( );
30   allocateLvPointAvg( );
31
32   for( int i=0;i<N;i++ ) { set_vp( i, vpMin*pow( vpMax/vpMin,(double)i/((double)N-1)) ); }
33 }
```

4.5.2.2 BlrSpherical::~BlrSpherical ( ) [inline]

destructor

```
39 { }
```

4.5.3 Member Function Documentation

4.5.3.1 double BlrSpherical::getvext (double _r) [virtual]

get v_ext value in Hz

Parameters

_r	- radius

Returns

v_ext

Reimplemented from externalRadiationSpherical.

```
65                                      {
66   vext = e*eV2erg/PLANCK_H;
67   return vext; }
```

4.5.3.2 void BlrSpherical::printInfo ( ) [virtual]

print basic information about myself (virtual)

Reimplemented from externalRadiationSpherical.

```
35                              {
36   bazinga::info(id,"Info");
37   bazinga::print_info(id,"N",N);
38   bazinga::print_info(id,"Avg energy (in external frame)",e,"eV");
39   bazinga::print_info(id,"Avg frequency (in external frame)",getvext( rext ),"Hz");
40   bazinga::print_info(id,"Clouds covering factor cf",cf);
41   bazinga::print_info(id,"radius r",rext,"cm");
42   bazinga::print_info(id,"q1",q1);
43   bazinga::print_info(id,"q2",q2);
44   bazinga::print_info(id,"vpMin",vpMin,"Hz");
45   bazinga::print_info(id,"vpMax",vpMax,"Hz");
46   if( luminosityConstU ) { bazinga::warning(id,"Using constant u' to calculate luminosity."
      ); }
47 }
```

4.5.3.3 void BlrSpherical::setRadius ( ) [virtual]

sets rext if not provided by config file

Reimplemented from externalRadiationSpherical.

```
49                                {
50   double Ledd = 1.3e47*mBH;
51   double Ldisk = eDisk*mDot*Ledd;
52   double R_sub = 1.6e-5*sqrt(Ldisk);
53   rext = 0.1*R_sub; }
```

4.5.3.4 void BlrSpherical::update ( ) [virtual]

calculate and set Upe every time with new radius r

Reimplemented from externalRadiationSpherical.

```
69                               {
70   double Ledd = 1.3e47*mBH;
71   double Ldisk = eDisk*mDot*Ledd;
72   double u_ext = Gamma*Gamma*cf*Ldisk/(6.*M_PI*LIGHT_SPEED*rext*rext);
73   if( r->get( ) <= rext ) { u_ext *= pow( r->get( )/rext, -q1 ); }
74   if( r->get( ) > rext ) { u_ext *= pow( r->get( )/rext, -q2 ); }
75   gsl_vector_set( upe, r->getIndex( ), u_ext );
76   bazinga::print_info(id,"Upe",gsl_vector_get( upe, r->getIndex( ) ) );
77   set_upe_r( gsl_vector_get( upe, r->getIndex( ) ) );
78   flag_upe_r = false; }
```

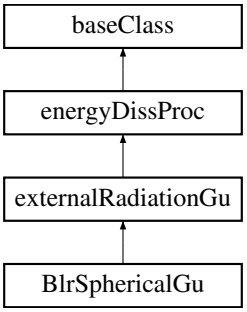
The documentation for this class was generated from the following files:

- /home/mjaniak/Soft/blazar++/include/BlrSpherical.hpp
- /home/mjaniak/Soft/blazar++/src/BlrSpherical.cpp

4.6 BlrSphericalGu Class Reference

`#include <BlrSphericalGu.hpp>`

Inheritance diagram for BlrSphericalGu:



4.6.1 Detailed Description

Parameters used by class (defaults)

Parameters

e	(externalRadiationGu) - energy (ext) in eV (e = 10.0)
cf	(externalRadiationSphericalGu) - covering factor (cf = 0.1)
rext	- blr radius (rext = 0.1∗r_sub)
k	- u_ext vs r index = 3

Public Member Functions

- BlrSphericalGu (scfgp ∗_cfg, jetGeometry ∗_r, electrons ∗_ele, std::string _id)
- ∼BlrSphericalGu ()
- void printInfo ()
- void update ()
- double getvext (double _r)
- void setRadius ()

Additional Inherited Members

4.6.2 Constructor & Destructor Documentation

4.6.2.1 BlrSphericalGu::BlrSphericalGu (scfgp ∗ *_cfg*, jetGeometry ∗ *_r*, electrons ∗ *_ele*, std::string *_id*)

constructor

Parameters

_cfg	- scfgp class object
_r	- jetGeometry class object
_ele	- electrons class object
_id	

```
9                                                                                                    :
     externalRadiationGu( cfg, r, ele, id ) {
10   /* requested parameters */
11   cfg -> request<double>(id + "e", 10.0, &e );
12   cfg -> request<double>(id + "cf", 0.1, &cf );
13   cfg -> request<double>(id + "rext", 0.0, &rext );
14   cfg -> request<double>(id + "k", 3.0, &k );
15
16   cfg -> updateRequests( );
17
18   if( rext == 0.0 ) { setRadius( ); }
19   else { bazinga::warning(id,"Overwritting rext value from config file!"); }
20
21   vpMin = 0.01*DSQR( ele -> getGammaMin( ) )*getvext( r->getRMax() );
22   vpMax = 100.0*DSQR( ele -> getGammaMax( ) )*getvext( r->getR0() );
23
24   allocateUpeR( ); // this is for storing vector information about Upe(r)
25   allocateLpv( );
26   allocateLvPoint( );
27   allocateLvPointAvg( );
28
29   for( int i=0;i<N;i++ ) { set_vp( i, vpMin*pow( vpMax/vpMin,(double)i/((double)N-1)) ); }
30 }
```

4.6.2.2 BlrSphericalGu::∼BlrSphericalGu ( ) [inline]

destructor

```
35 { }
```

4.6.3 Member Function Documentation

4.6.3.1 double BlrSphericalGu::getvext (double *_r*) [virtual]

get v_ext value in Hz

Parameters

_r	- radius

Returns

v_ext

Reimplemented from externalRadiationGu.

```
32 { return e*eV2erg/PLANCK_H; }
```

4.6.3.2 void BlrSphericalGu::printInfo ( ) [virtual]

print basic information about myself (virtual)

Reimplemented from externalRadiationGu.

```
47                                {
48   bazinga::info(id,"Info");
49   bazinga::print_info(id,"N",N);
50   bazinga::print_info(id,"Avg energy (in external frame)",e,"eV");
51   bazinga::print_info(id,"Avg frequency (in external frame)",getvext( r->getR0()),"Hz");
52   bazinga::print_info(id,"Clouds covering factor cf",cf);
53   bazinga::print_info(id,"radius rext",rext,"cm");
54   bazinga::print_info(id,"strat index k",k);
55   bazinga::print_info(id,"vpMin",vpMin,"Hz");
56   bazinga::print_info(id,"vpMax",vpMax,"Hz");
57 }
```

4.6.3.3 void BlrSphericalGu::setRadius ( ) [virtual]

sets rext if not provided by config file

Reimplemented from externalRadiationGu.

```
59                                 {
60   double Ledd = 1.3e47*mBH;
61   double Ldisk = eDisk*mDot*Ledd;
62   double R_sub = 1.6e-5*sqrt( Ldisk );
63   rext = 0.1*R_sub; }
```

4.6.3.4 void BlrSphericalGu::update ( ) [virtual]

calculate and set Upe every time with new radius r

Reimplemented from externalRadiationGu.

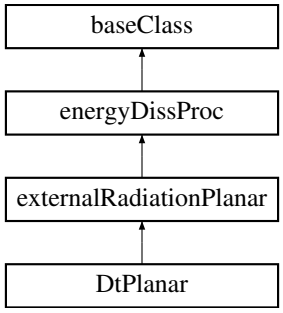
```
34                               {
35   double Ledd = 1.3e47*mBH;
36   double Ldisk = eDisk*mDot*Ledd;
37   double temp_upe = 0.0;
38   double temp_cf = 0.0;
39   temp_cf = cf;
40   temp_cf *= pow(r->get()/rext,2)/(1.0+pow(r->get()/rext,k));
41   temp_upe = (temp_cf*Ldisk*DSQR(Gamma))/(3.0*M_PI*DSQR(r->get())*LIGHT_SPEED);
42   gsl_vector_set( upe_r, r->getIndex( ), temp_upe );
43   bazinga::print_info(id,"Upe",gsl_vector_get( upe_r, r->getIndex( ) ) );
44   set_upe_r( gsl_vector_get( upe_r, r->getIndex( ) ) );
45   flag_upe_r = false; }
```

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

- /home/mjaniak/Soft/blazar++/include/BlrSphericalGu.hpp
- /home/mjaniak/Soft/blazar++/src/BlrSphericalGu.cpp

4.7 DtPlanar Class Reference

Inheritance diagram for DtPlanar:



Public Member Functions

- DtPlanar (scfgp ∗_cfg, jetGeometry ∗_r, electrons ∗_ele, std::string _id)
- ∼DtPlanar ()
- void printInfo ()
- void setRadius ()
- double getvext (double _R)
- void setdLdR ()

Additional Inherited Members

4.7.1 Constructor & Destructor Documentation

4.7.1.1 DtPlanar::DtPlanar (scfgp ∗ *_cfg,* jetGeometry ∗ *_r,* electrons ∗ *_ele,* std::string *_id*)

constructor

Parameters

_cfg	- scfgp class object
_r	- jetGeometry class object
_ele	- electrons class object
_id	

```
9                                                                                :
     externalRadiationPlanar( cfg, r, ele, id ) {
10   /* request parameters */
11   cfg -> request<double>( id+"e", 10.0, &e );
12   cfg -> request<double>( id+"cf", 0.1, &cf );
13
14   cfg -> updateRequests( );
15
16   if( R1 == 0.0 || R2 == 0.0 ) { setRadius( ); }
17   else { bazinga::warning(id,"Overwritting R1 and R2 values from config file!"); }
18
19   vpMin = 0.01*DSQR( ele -> getGammaMin( ) )*getvext( R2 );
20   vpMax = 100.0*DSQR( ele -> getGammaMax( ) )*getvext( R1 );
21
22   /* initialize disk radius R;
23      here we use N-1 to use the same Upe as in other processes */
24   R = new logGeometry( R1, R2, N-1 );
25
26   allocateUpe( ); // this is for storing matrix information about dUpe(r)/dR (R)
27   allocateUpeR( ); // this is for storing vector information about Upe(r)
28   allocateLpv( );
29   allocateLvPoint( );
30   allocateLvPointAvg( );
31
```

```
32   for( int i=0;i<N;i++ ) { set_vp( i, vpMin*pow( vpMax/vpMin,(double)i/((double)N-1)) ); }
33
34   /* set the values of dLdR */
35   setdLdR( ); }
```

4.7.1.2 DtPlanar::~DtPlanar ( )

destructor

```
37                      {
38   freeUpe( );
39   freeLpv( );
40   freeLvPoint( );
41   freeLvPointAvg( ); }
```

4.7.2 Member Function Documentation

4.7.2.1 double DtPlanar::getvext (double _R) [virtual]

get v_ext value in Hz

Parameters

_R	- radius in accretion disk plane

Returns

v_ext

Reimplemented from externalRadiationPlanar.

```
82                                      {
83   vext = e*eV2erg/PLANCK_H;
84   if( luminosityConstNu ) { return vext; }
85   else { return vext*R1/_R; }
86 }
```

4.7.2.2 void DtPlanar::printInfo ( ) [virtual]

print basic information about myself (virtual)

Reimplemented from externalRadiationPlanar.

```
43                          {
44   bazinga::info(id,"Info");
45   bazinga::print_info(id,"N",N);
46   bazinga::info(id,"Using stratification version NEW (only option)");
47   bazinga::print_info(id,"radius R1",R1,"cm");
48   bazinga::print_info(id,"radius R2",R2,"cm");
49   bazinga::print_info(id,"stratification index",s);
50   bazinga::print_info(id,"Avg energy (in external frame)",e,"eV");
51   bazinga::print_info(id,"Avg frequency (in external frame)",getvext( R1 ),"Hz");
52   bazinga::print_info(id,"Clouds covering factor cf",cf);
53   bazinga::print_info(id,"alpha",alpha);
54   bazinga::print_info(id,"vpMin",vpMin,"Hz");
55   bazinga::print_info(id,"vpMax",vpMax,"Hz");
56   if( approx ) { bazinga::info(id,"Using approximate dependence on R"); }
57   if( luminosityConstU ) { bazinga::warning(id,"Using constant u' to calculate luminosity."
      ); }
58   if( luminosityConstNu ) { bazinga::warning(id,"Using constant v_ext to calculate
       luminosity."); }
59 }
```

4.7.2.3 void DtPlanar::setdLdR ( ) [virtual]

set dL/dR for particulat source

Reimplemented from externalRadiationPlanar.

```
68                          {
69   double Ledd = 1.3e47*mBH;
70   double Ldisk = eDisk*mDot*Ledd;
71   /* fill in dLdR vector that will be used for the rest of the calculations */
72   bazinga::info(id,"Seeting dLdR.");
73   for( int i=0;i<R->getMaxIndex();i++ ) {
74       R -> update( i );
75       double val = 0.0;
76       val = cf*Ldisk*ctau( )*pow(R->get( ),-s);
77       gsl_vector_set( dLdR, R->getIndex(), val ); }
78   /* Now when we have all set let us save what we have just calculated - dLdR */
79   bazinga::info(id,"Saving dLdR.");
80   bazinga::save_GSLVector( "dLdR_"+this->whoAmI( ), R->getRadius_GSLVector( ), dLdR,
      cfg->get<std::string>("output") ); }
```

4.7.2.4 void DtPlanar::setRadius ( ) [virtual]

sets rext if not provided by config file

Reimplemented from externalRadiationPlanar.

```
61                           {
62   double Ledd = 1.3e47*mBH;
63   double Ldisk = eDisk*mDot*Ledd;
64   double R_sub = 1.6e-5*sqrt( Ldisk );
65   R1 = R_sub;
66   R2 = 10.0*R_sub; }
```

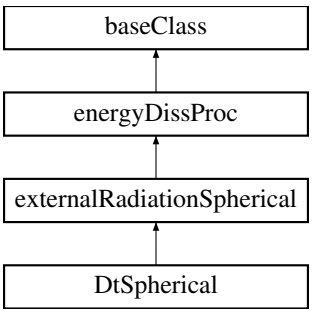
The documentation for this class was generated from the following files:

- /home/mjaniak/Soft/blazar++/include/DtPlanar.hpp
- /home/mjaniak/Soft/blazar++/src/DtPlanar.cpp

4.8 DtSpherical Class Reference

`#include <DtSpherical.hpp>`

Inheritance diagram for DtSpherical:



4.8.1 Detailed Description

Parameters used by class (defaults)

Parameters

e	(externalRadiationSpherical) - energy (ext) in eV (e = 0.4136 which corresponds to 1.5e14Hz @ rext)
cf	(externalRadiationSpherical) - covering factor (cf = 0.1)
q1	- lower index in stratification (q1 = 0.0)
q2	- upper index in stratification (q2 = -3.0)
rext	- dt radius (rext = 1.0∗r_sub)

Public Member Functions

- DtSpherical (scfgp ∗_cfg, jetGeometry ∗_r, electrons ∗_ele, std::string _id)
- ∼DtSpherical ()
- void update ()
- void printInfo ()
- void setRadius ()
- double getvext (double _r)

Additional Inherited Members

4.8.2 Constructor & Destructor Documentation

4.8.2.1 DtSpherical::DtSpherical (scfgp ∗ *_cfg*, jetGeometry ∗ *_r*, electrons ∗ *_ele*, std::string *_id*)

constructor

Parameters

_cfg	- scfgp class object
_r	- jetGeometry class object
_ele	- electrons class object
_id	

```
9                                                                                    :
     externalRadiationSpherical( cfg, r, ele, id ) {
10   /* request parameters */
11   cfg -> request<double>( id+"q1", 1.0, &q1 );
12   cfg -> request<double>( id+"q2", 1.0, &q2 );
13   cfg -> request<double>( id+"r", 0.0, &rext );
14   cfg -> request<double>("mBH",1.0,&mBH);
15   cfg -> request<double>("eDisk",0.1,&eDisk);
16   cfg -> request<double>("mDot",1.0,&mDot);
17   cfg -> request<double>("Gamma",10.0,&Gamma);
18
19   cfg -> updateRequests( );
20
21   if( rext == 0.0 ) { setRadius( ); }
22   else { bazinga::warning(id,"Overwritting rext value from config file!"); }
23
24   vpMin = 0.01*DSQR( ele -> getGammaMin() )*getvext( 10.*rext );
25   vpMax = 100.0*DSQR( ele -> getGammaMax() )*getvext( rext );
26
27   allocateUpe( );
28   allocateLpv( );
29   allocateLvPoint( );
30   allocateLvPointAvg( );
31
32   for( int i=0;i<N;i++ ) { set_vp( i, vpMin*pow( vpMax/vpMin,(double)i/((double)N-1)) ); }
33 }
```

4.8.2.2 DtSpherical::∼DtSpherical ( ) [inline]

destructor

```
45 { }
```

4.8.3 Member Function Documentation

4.8.3.1 double DtSpherical::getvext (double _r) [virtual]

get v_ext value in Hz

Parameters

_r	- radius

Returns

v_ext

Reimplemented from externalRadiationSpherical.

```
74                                      {
75   vext = e*eV2erg/PLANCK_H;
76   if( luminosityConstNu ) { return vext; }
77   else { return vext*rext/_r; }
78 }
```

4.8.3.2 void DtSpherical::printInfo ( ) [virtual]

print basic information about myself (virtual)

Reimplemented from externalRadiationSpherical.

```
35                              {
36   bazinga::info(id,"Info");
37   bazinga::print_info(id,"id",id);
38   bazinga::print_info(id,"N",N);
39   bazinga::print_info(id,"Avg energy (in external frame) @ rext",e,"eV");
40   bazinga::print_info(id,"Avg frequency (in external frame) @ rext",getvext( rext ),"Hz");
41   bazinga::print_info(id,"Clouds covering factor cf",cf);
42   bazinga::print_info(id,"radius r",rext,"cm");
43   //  bazinga::print_info(id,"radius r2",r2,"cm");
44   bazinga::print_info(id,"q1",q1);
45   bazinga::print_info(id,"q2",q2);
46   bazinga::print_info(id,"vpMin",vpMin,"Hz");
47   bazinga::print_info(id,"vpMax",vpMax,"Hz");
48   if( luminosityConstU ) { bazinga::warning(id,"Using constant u' to calculate luminosity."
      ); }
49   if( luminosityConstNu ) { bazinga::warning(id,"Using constant v' to calculate
       luminosity."); }
50  }
```

4.8.3.3 void DtSpherical::setRadius ( ) [virtual]

sets rext if not provided by config file

Reimplemented from externalRadiationSpherical.

```
52                              {
53   double Ledd = 1.3e47*mBH;
54   double Ldisk = eDisk*mDot*Ledd;
55   double R_sub = 1.6e-5*sqrt(Ldisk);
56   rext = R_sub; }
```

4.8.3.4 void DtSpherical::update ( ) [virtual]

calculate and set Upe every time with new radius r

Reimplemented from externalRadiationSpherical.

```
80                          {
81   double Ledd = 1.3e47*mBH;
82   double Ldisk = eDisk*mDot*Ledd;
83   double u_ext = Gamma*Gamma*cf*Ldisk/(3.*M_PI*LIGHT_SPEED*rext*rext);
84   if( r->get( ) <= rext ) { u_ext *= pow( r->get( )/rext, -q1 ); }
85   if( r->get( ) > rext ) { u_ext *= pow( r->get( )/rext, -q2 ); }
86   gsl_vector_set( upe, r->getIndex( ), u_ext );
87   bazinga::print_info(id,"Upe",gsl_vector_get( upe, r->getIndex( ) ) );
88   set_upe_r( gsl_vector_get( upe, r->getIndex( ) ) );
89   flag_upe_r = false; }
```

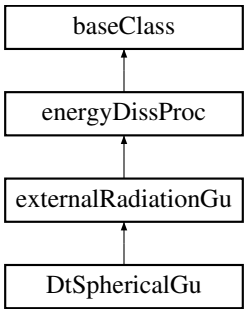
The documentation for this class was generated from the following files:

- /home/mjaniak/Soft/blazar++/include/DtSpherical.hpp
- /home/mjaniak/Soft/blazar++/src/DtSpherical.cpp

4.9 DtSphericalGu Class Reference

`#include <DtSphericalGu.hpp>`

Inheritance diagram for DtSphericalGu:



4.9.1 Detailed Description

Parameters used by class (defaults)

Parameters

e	(externalRadiationGu) - energy (ext) in eV (e = 0.6203)
cf	(externalRadiationSphericalGu) - covering factor (cf = 0.1)
rext	- blr radius (rext = 1.0∗r_sub)
k	- u_ext vs r index = 2

Public Member Functions

- DtSphericalGu (scfgp ∗_cfg, jetGeometry ∗_r, electrons ∗_ele, std::string _id)
- ∼DtSphericalGu ()
- void printInfo ()
- void update ()
- double getvext (double _r)
- void setRadius ()

Additional Inherited Members

4.9.2 Constructor & Destructor Documentation

4.9.2.1 DtSphericalGu::DtSphericalGu (scfgp * *_cfg*, jetGeometry * *_r*, electrons * *_ele*, std::string *_id*)

constructor

Parameters

_cfg	- scfgp class object
_r	- jetGeometry class object
_ele	- electrons class object
_id	

```
9                                                                                                  :
     externalRadiationGu( cfg, r, ele, id ) {
10   /* requested parameters */
11   cfg -> request<double>(id + "e", 0.6203, &e );
12   cfg -> request<double>(id + "cf", 0.1, &cf );
13   cfg -> request<double>(id + "rext", 0.0, &rext );
14   cfg -> request<double>(id + "k", 2.0, &k );
15
16   cfg -> updateRequests( );
17
18   if( rext == 0.0 ) { setRadius( ); }
19   else { bazinga::warning(id,"Overwritting rext value from config file!"); }
20
21   vpMin = 0.01*DSQR( ele -> getGammaMin( ) )*getvext( r->getRMax() );
22   vpMax = 100.0*DSQR( ele -> getGammaMax( ) )*getvext( r->getR0() );
23
24   allocateUpeR( ); // this is for storing vector information about Upe(r)
25   allocateLpv( );
26   allocateLvPoint( );
27   allocateLvPointAvg( );
28
29   for( int i=0;i<N;i++ ) { set_vp( i, vpMin*pow( vpMax/vpMin,(double)i/((double)N-1)) ); }
30 }
```

4.9.2.2 DtSphericalGu::~DtSphericalGu ( ) [inline]

destructor

```
35 { }
```

4.9.3 Member Function Documentation

4.9.3.1 double DtSphericalGu::getvext (double *_r*) [virtual]

get v_ext value in Hz

Parameters

_r	- radius

Returns

v_ext

Reimplemented from externalRadiationGu.

```
32                                        {
33   double vext;
34   vext = e*eV2erg/PLANCK_H;
35   if( luminosityConstNu ) { return vext; }
36   else { return vext*rext/_r; }
37 }
```

4.9.3.2 void DtSphericalGu::printInfo ( ) [virtual]

print basic information about myself (virtual)

Reimplemented from externalRadiationGu.

```
52                                {
53   bazinga::info(id,"Info");
54   bazinga::print_info(id,"N",N);
55   bazinga::print_info(id,"Avg energy (in external frame)",e,"eV");
56   bazinga::print_info(id,"Avg frequency (in external frame)",getvext( r->getR0()),"Hz");
57   bazinga::print_info(id,"Clouds covering factor cf",cf);
58   bazinga::print_info(id,"radius rext",rext,"cm");
59   bazinga::print_info(id,"strat index k",k);
60   bazinga::print_info(id,"vpMin",vpMin,"Hz");
61   bazinga::print_info(id,"vpMax",vpMax,"Hz");
62 }
```

4.9.3.3 void DtSphericalGu::setRadius ( ) [virtual]

sets rext if not provided by config file

Reimplemented from externalRadiationGu.

```
64                                 {
65   double Ledd = 1.3e47*mBH;
66   double Ldisk = eDisk*mDot*Ledd;
67   double R_sub = 1.6e-5*sqrt( Ldisk );
68   rext = R_sub; }
```

4.9.3.4 void DtSphericalGu::update ( ) [virtual]

calculate and set Upe every time with new radius r

Reimplemented from externalRadiationGu.

```
39                                {
40   double Ledd = 1.3e47*mBH;
41   double Ldisk = eDisk*mDot*Ledd;
42   double temp_upe = 0.0;
43   double temp_cf = 0.0;
44   temp_cf = cf;
45   temp_cf *= pow(r->get()/rext,2)/(1.0+pow(r->get()/rext,k));
46   temp_upe = (temp_cf*Ldisk*DSQR(Gamma))/(3.0*M_PI*DSQR(r->get())*LIGHT_SPEED);
47   gsl_vector_set( upe_r, r->getIndex( ), temp_upe );
48   bazinga::print_info(id,"Upe",gsl_vector_get( upe_r, r->getIndex( ) ) );
49   set_upe_r( gsl_vector_get( upe_r, r->getIndex( ) ) );
50   flag_upe_r = false; }
```

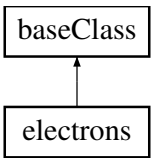
The documentation for this class was generated from the following files:

- /home/mjaniak/Soft/blazar++/include/DtSphericalGu.hpp
- /home/mjaniak/Soft/blazar++/src/DtSphericalGu.cpp

4.10 electrons Class Reference

`#include <electrons.hpp>`

Inheritance diagram for electrons:



4.10.1 Detailed Description

Class to hold all detaild for electrons in the jet (all calculated in jet co-moving frame); it is responsible for calculating energy losses, solving propagation equation, electron energy spectrum etc.

Public Member Functions

- electrons (scfgp ∗_cfg, jetGeometry ∗_r, std::string _id)
- ∼electrons ()
- double tQ (double g, double _radius)
- double gauss (double _radius)
- double gaussmod (double _radius)
- double tri (double _radius)
- double inject ()
- void evolve ()
- double dgdr (double g)
- double getGamma (int i)
- double getNgamma (int i)
- void addEnDissProc (energyDissProc ∗_obj)
- void listEnDissProc ()
- void printInfo ()
- void printCoolingInfo (double g)
- void avgNgamma ()
- void setInjectionParameters ()
- void setEnergetics ()
- double **getGammaMin** ()
- double **getGammaMax** ()
- double beta (double x)
- double getdLogGamma ()
- int ifEvol ()
- void saveInjection ()
- void saveNgamma ()
- void saveNgammaAvg ()

Public Attributes

- std::vector< energyDissProc ∗ > EnDissProc
- gsl_vector ∗ gamma
- gsl_vector ∗ **Ngamma**
- gsl_vector ∗ **NgammaAvg**
- gsl_vector ∗ A
- gsl_vector ∗ **B**
- gsl_vector ∗ **C**
- gsl_vector ∗ **S**
- gsl_vector ∗ **U**

4.10.2 Constructor & Destructor Documentation

4.10.2.1 electrons::electrons (scfgp ∗ *_cfg,* jetGeometry ∗ *_r,* std::string *_id*)

constructor

Parameters

scfgp	
jetGeometry	
id	

```
10                                                                  : baseClass(_cfg, _r, _id ) {
11   /* requested parameters */
12   cfg -> request<int>("N"+id,200,&N);
13   cfg -> request<std::string>("eleModel","blob",&eleModel);
14   cfg -> request<std::string>("injModel","REC",&injModel);
15   cfg -> request<std::string>("lumModel","blob",&lumModel);
16   cfg -> request<double>("p1",2.0,&p1);
17   cfg -> request<double>("p2",2.0,&p2);
18   cfg -> request<double>("gammaMin",1.0,&gammaMin);
19   cfg -> request<double>("gammaBreak",0.0,&gammaBreak);
20   cfg -> request<double>("gammaMax",1.0e2,&gammaMax);
21   cfg -> request<double>("injRm",2.0e17,&injRm);
22   cfg -> request<double>("injSigma",0.6,&injSigma);
23   cfg -> request<double>("eleK",0.0,&eleK);
24   cfg -> request<double>("eEle",0.1,&eEle);
25   cfg -> request<double>("eDiss",0.1,&eDiss);
26   cfg -> request<double>("Gamma",10.0,&Gamma);
27   cfg -> request<double>("NeNp",0.1,&NeNp);
28   cfg -> request<double>("mBH",1.0,&mBH);
29   cfg -> request<double>("eDisk",0.1,&eDisk);
30   cfg -> request<double>("mDot",1.0,&mDot);
31   cfg -> request<double>("eJet",0.3,&eJet);
32   cfg -> request<double>("sigmaB",0.1,&sigmaB);
33   cfg -> request<int>("evol",0,&evol);
34   cfg -> request<int>("saveElectrons",1,&saveElectrons);
35   cfg -> request<int>("saveElectronsAvg",1,&saveElectronsAvg);
36
37   cfg -> updateRequests( );
38
39   /* check if spectral indices are OK; for steady model p1 must be lower than 1 and p2 must be greater than
       2; otherwise calculation of break electron energy will fail;
40      for blob model it is all good to set any indices you like */
41   if( (p1 > 1.0 || p2 < 2.0) && eleModel == "steady" && gammaBreak == 0.0 ) {
42     bazinga::warning(id,"Can't use steady model with p1>1.0 or p2<2.0!");
43     bazinga::warning(id,"To overwrite that you can set gammaBreak explicity.");
44     bazinga::warning(id,"Quit.");
45     exit(0); }
46
47   allocateGamma( );
48   allocateTridiag( );
49
50   for( int i=0;i<=N+1;i++ ) { gsl_vector_set( gamma, i, pow( gammaMax,(double)i/(double)
      N )); }
51
52   setEnergetics( );
53   setInjectionParameters( );
54   dLogGamma = log( gsl_vector_get(gamma,1)/gsl_vector_get(gamma,0) ); }
```

4.10.2.2 electrons::~electrons ( )

destructor

```
140                         {
141   freeGamma( );
142   freeTridiag( ); }
```

4.10.3 Member Function Documentation

4.10.3.1 void electrons::addEnDissProc (energyDissProc * *_obj*)

add process to energy dissipation

Parameters

energyDissProc	

```
234 { EnDissProc.push_back( _obj ); }
```

4.10.3.2 void electrons::avgNgamma ( )

calculate averaged electron distribution

```
361                             {
362   for( int i=0;i<N;i++ ) { gsl_vector_set( NgammaAvg, i, gsl_vector_get( NgammaAvg, i ) +
      getNgamma( i ) ); }
363 }
```

4.10.3.3 double electrons::beta (double *x*)

probably doubled from baseClass??

```
373 { return sqrt(1.0-1.0/(x*x)); }
```

4.10.3.4 double electrons::dgdr (double *g*)

get electron cooling details

Parameters

g	- electron Lorentz factor

Returns

d gamma \ d t

```
222                                 {
223   double dotg = 0.0;
224   for( int i=0; i<EnDissProc.size(); i++ ) { dotg += EnDissProc[i]->dotg( g ); }
225   dotg *= A43sigTmec; /* multiply by common factor */
226   dotg *= g*g;
227   dotg *= 1.0/(beta(Gamma)*LIGHT_SPEED*Gamma); /* convert from time to radius */
228   if( cfg->get<int>("Adiabatic") ) { dotg += 0.666667*g/r->get(); } /* adiabatic cooling */
229   return dotg; }
```

4.10.3.5 void electrons::evolve ( )

main electron evolution function; used gsl_tridiag to solve evolution equation

```
145                         {
146   double err = 0.0;
147   bazinga::info(id,"Cooling details:");
148   /* print info about cooling */
149   bazinga::info(id,"@ gammaMIN:");
150   printCoolingInfo( gsl_vector_get(gamma,0) );
151   bazinga::info(id,"@ gammaMAX:");
152   printCoolingInfo( gsl_vector_get(gamma,N-1) );
153
154   if( cfg->get<int>("Ssc") ) { bazinga::info(id,"Ssc loop:"); }
155   do {
156     /* solve equations for electrons */
157     for( int i=0;i<N;i++ )
158       { gsl_vector_set( B, i, 1.0+r->getDr()*dgdr( 0.5*(gsl_vector_get(
      gamma,i)+gsl_vector_get(gamma,i+1)) )/(0.5*(gsl_vector_get(gamma,i+2)-gsl_vector_get(
```

```
      gamma,i))) ); }
159     for( int i=0;i<N-1;i++ )
160       { gsl_vector_set( C, i, -r->getDr()*dgdr(0.5*(gsl_vector_get(gamma,i+1)+gsl_vector_get(
      gamma,i+2)) )/(0.5*(gsl_vector_get(gamma,i+2)-gsl_vector_get(gamma,i))));    }
161
162     gsl_linalg_solve_tridiag(B,C,A,S,U);
163
164     gsl_vector_memcpy( Ngamma, U );
165
166     if ( cfg->get<int>("Ssc") ) { /* in order to calculate corrected model for ssc after electron
       evolution we need to first identify which pointer in array corresponds to synchrotron */
167       for( int j=0; j<EnDissProc.size(); j++ )
168     if( EnDissProc[j]->whoAmI( ) == "syn" ) {
169       err = ( static_cast<synchrotron*>( EnDissProc[j] ) ) -> iterate( );
170       bazinga::print_info(id,"  iteration error",err); }
171     }
172   } while ( err > ESP ); }
```

4.10.3.6 double electrons::gauss (double *_radius*)

gaussian radial injection profile

Parameters

radius	

```
204                                       {
205   double x = _radius/injRm;
206   return exp( -pow(x-1.0,2)/(pow( injSigma,2)) ); }
```

4.10.3.7 double electrons::gaussmod (double *_radius*)

modified gaussian radial injection profile

Parameters

radius	

```
208                                          {
209   double x = _radius/injRm;
210   return x*exp( (1-x)*(x+pow( injSigma,2)-1.0 )/pow( injSigma,2) ) ; }
```

4.10.3.8 double electrons::getdLogGamma ( ) [inline]

get d gamma in log scale

Returns

dLogGamma

```
130 { return dLogGamma; }
```

4.10.3.9 double electrons::getGamma (int *i*)

get electron gamma factor

Parameters

vector	index

Returns

electron gamma factor

```
231 { return gsl_vector_get( gamma,i+1 ); }
```

4.10.3.10 double electrons::getNgamma (int *i*)

get electron Ngamma

Parameters

vector	index

Returns

electron Ngamma

```
232 { return gsl_vector_get( Ngamma,i ); }
```

4.10.3.11 int electrons::ifEvol ( ) [inline]

check if we should do electron evolution

```
133 { return evol; }
```

4.10.3.12 double electrons::inject ( )

electron injection function

```
175                          {
176   for( int i=1;i<=N;i++ )
177     if( evol ) { gsl_vector_set( S, i-1, gsl_vector_get(Ngamma,i-1)+r->getDr()*
      tQ( gsl_vector_get(gamma, i), r->get( )) ); }
178     else { gsl_vector_set( Ngamma, i-1, gsl_vector_get(Ngamma,i-1)+r->getDr()*
      tQ( gsl_vector_get(gamma, i), r->get()) ); }
179 }
```

4.10.3.13 void electrons::listEnDissProc ( )

list available and active energy dissipation processes

```
236                                {
237   std::stringstream s;
238   for( int i=0; i<EnDissProc.size(); i++ ) { s << EnDissProc[i]->whoAmI() << " "; }
239   bazinga::print_info(id,"Processes used in electron cooling",s.str( )); }
```

4.10.3.14 void electrons::printCoolingInfo (double *g*)

print information about cooling details for active processes electron Lorentz factor

```
217                                          {
218   for( int i=0; i<EnDissProc.size(); i++ ) { bazinga::print_info(id,
      EnDissProc[i]->whoAmI(),EnDissProc[i]->dotg( g )*A43sigTmec*g*g/(
      beta(Gamma)*LIGHT_SPEED*Gamma)); }
219   if( cfg->get<int>("Adiabatic") ) { bazinga::print_info(id,"adiabatic",cfg->get<double>("
      AdiabaticABG")*g/r->get( )); }
220 }
```

4.10.3.15 void electrons::printInfo ( ) [virtual]

print basic information about myself (virtual)

Reimplemented from baseClass.

```
99                          {
100   bazinga::info(id,"Info");
101   bazinga::print_info(id,"evol",evol);
102   bazinga::print_info(id,"Electron model",eleModel);
103   bazinga::print_info(id,"Injection model",injModel);
104   bazinga::print_info(id,"N",N);
105   bazinga::print_info(id,"Index p 1",p1);
106   bazinga::print_info(id,"Index p 2",p2);
107   bazinga::print_info(id,"Gamma MIN",gammaMin);
108   bazinga::print_info(id,"Gamma MAX",gammaMax);
109
110   if( injModel == "GAUSS" || injModel == "GAUSSMOD" ) {
111     bazinga::print_info(id,"injection maximium R_m", injRm);
112     bazinga::print_info(id,"injection disspersion sigma",injSigma); }
113   else if( injModel == "TRI") {
114     bazinga::print_info(id,"injection maximium R_m",injRm); }
115
116   if( eleModel == "steady" && gammaBreak ) { bazinga::print_info(id,"Injection average gamma",avgGamma); }
117
118   bazinga::print_info(id,"Gamma break",gammaBreak);
119   bazinga::print_info(id,"Electron normalization",eleK);
120
121   if( eleModel == "steady") {
122     bazinga::print_info(id,"Luminosity model",lumModel);
123     bazinga::print_info(id,"eta electrons",eEle);
124     bazinga::print_info(id,"eta dissipation",eDiss);
125     bazinga::print_info(id,"Jet Lorentz factor",Gamma);
126     bazinga::print_info(id,"n_e/n_p",NeNp);
127     bazinga::print_info(id,"BH mass",mBH);
128     bazinga::print_info(id,"eta disk",eDisk);
129     bazinga::print_info(id,"m dot",mDot);
130     bazinga::print_info(id,"eta jet",eJet);
131     bazinga::print_info(id,"magnetic sigma",sigmaB);
132     bazinga::print_info(id,"Eddington luminosity",Ledd,"erg/s");
133     bazinga::print_info(id,"Accretion disk luminosity",Ldisk,"erg/s");
134     bazinga::print_info(id,"Jet power before dissipation",Ljet,"erg/s");
135     bazinga::print_info(id,"Jet power after dissipation",LjetDiss,"erg/s");
136     bazinga::print_info(id,"Magnetic energy flux",Lb,"erg/s");
137     bazinga::print_info(id,"Kinetic energy of cold protons",Lprot,"erg/s"); }
138 }
```

4.10.3.16 void electrons::saveInjection ( )

save injected electron spectrum

```
375                              {
376   if( saveElectrons && r->ifSaveRadius( ) ) {
377     bazinga::info(id,"Saving injection");
378     bazinga::save_GSLVectorEle( "Injection", gamma, S, r->getPosition( ),
      cfg->get<std::string>("output") ); }
379 }
```

4.10.3.17 void electrons::saveNgamma ( )

save current electron spectrum

```
381                             {
382   if( saveElectrons && r->ifSaveRadius( ) ) {
383     bazinga::info(id,"Saving Ngamma");
384     bazinga::save_GSLVectorEle( "Ngamma", gamma, Ngamma, r->getPosition( ),
      cfg->get<std::string>("output") ); }
385 }
```

4.10.3.18 void electrons::saveNgammaAvg ( )

save averaged electron spectrum

```
387                                {
388   if( saveElectronsAvg ) {
389     bazinga::info(id,"Saving NgammaAvg");
390     bazinga::save_GSLVectorEle( "Ngamma_Avg", gamma, NgammaAvg, cfg->get<std::string>("output") );
      }
391 }
```

4.10.3.19 void electrons::setEnergetics ( )

set jet energetics

```
365                                {
366    Ledd = 1.3e47*mBH;
367    Ldisk = eDisk*mDot*Ledd;
368    Ljet = 0.5*eJet*mDot*Ledd;
369    LjetDiss = (1.0-eDiss)*Ljet;
370    Lprot = (1.0-eDiss)*Ljet/(1.0+sigmaB);
371    Lb = sigmaB*(1.0-eDiss)*Ljet/(1.0+sigmaB); }
```

4.10.3.20 void electrons::setInjectionParameters ( )

set electron injection parameters

```
281                                        {
282   if( eleModel == "steady" ) {
283     if( !gammaBreak || !eleK ) {
284       avgGamma = mpme*eEle*eDiss*(Gamma-1.0)*(1.0+sigmaB)/( NeNp*(1.0-eDiss)*Gamma );
285       avgGamma += 1.0;
286
287       gammaBreak = solveGammaBreak( p1, p2, gammaMin, gammaMax, avgGamma );
288
289       eleK = eEle*eDiss*Ljet/( (avgGamma-1.0)*mec2 );
290       eleK /= f2( p1, p2, gammaMin, gammaMax, gammaBreak ); }
291     else { bazinga::error(id,"Set both gammaBreak and eleK or do not set any of them."); }
292
293     if( lumModel == "steady" ) { eleK /= (r->getNinj( )); }
294   }
295 }
```

4.10.3.21 double electrons::tQ (double *g*, double *_radius*)

electron injection function

Parameters

g	- gamma Lorentz factor
_radius	- radius at which injection occurs

Returns

N_gamma

```
181                                                {
182   double x;
183
184   if( eleModel == "steady" ) {
185     if( g >= gammaMin && g <= gammaMax && _radius >= r->getR0( ) && _radius <=
      r->getRInjMax( ) ) {
186       if( g <= gammaBreak ) { x = eleK*pow( gammaBreak, p1-p2 )*pow( g, -p1 ); }
187       else { x = eleK*pow( g, -p2 ); }
188     }
189     else { x = 0.0; }
```

```
190     return x/(LIGHT_SPEED*beta(Gamma)*Gamma); }
191
192   if( eleModel == "blob" ) {
193     double corr = 1.0;
194     if( injModel == "GAUSS" ) { corr = gauss( _radius ); }
195     if( injModel == "GAUSSMOD" ) { corr = gaussmod( _radius ); }
196     if( injModel == "TRI" ) { corr = tri( _radius ); }
197
198     if( g >= gammaMin && g <= gammaMax && _radius >= r->getR0( ) && _radius <=
      r->getRInjMax( ) ) {
199       x = eleK*pow(g,-p1)*pow(1.0 + pow(g/gammaBreak,4.0),(p1-p2)/4.0)/(LIGHT_SPEED*
      beta(Gamma)*Gamma); }
200     else { x = 0.0; }
201     return corr*x; }
202 }
```

4.10.3.22 double electrons::tri (double *_radius*)

triangle radial injection profile

Parameters

radius	

```
212                                     {
213   if( _radius < injRm ) { return 0.5+0.5*(_radius-r->getR0( ))/(injRm-r->getR0( )); }
214   else if( _radius >= injRm ) { return 1.0+0.5*(_radius-injRm)/(injRm-r->getRInjMax( )); }
215 }
```

4.10.4 Member Data Documentation

4.10.4.1 gsl_vector* electrons::A

tridag vectors

4.10.4.2 std::vector<energyDissProc*> electrons::EnDissProc

vector to hold active energy dissipation processes

4.10.4.3 gsl_vector* electrons::gamma

vectors to store electron spectrum data

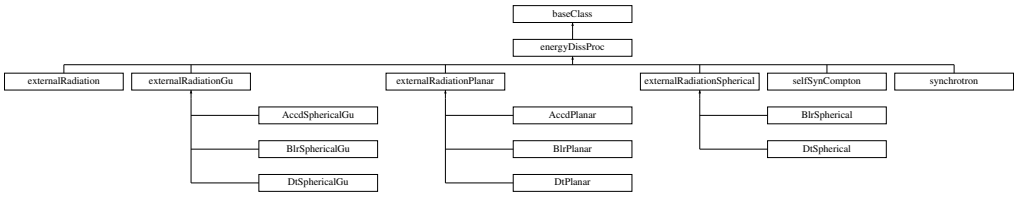
The documentation for this class was generated from the following files:

- /home/mjaniak/Soft/blazar++/include/electrons.hpp
- /home/mjaniak/Soft/blazar++/src/electrons.cpp

4.11 energyDissProc Class Reference

`#include <energyDissProc.hpp>`

Inheritance diagram for energyDissProc:



4.11.1 Detailed Description

Interface class for every energy dissipation process; provides memory management and other common methods shared by every process

Public Member Functions

- energyDissProc (scfgp ∗cfg, jetGeometry ∗r, electrons ∗ele, std::string _id)
- ∼energyDissProc ()
- virtual void setLpv ()
- virtual double calculateLpv (double v)
- virtual void update ()
- virtual double dotg (double g)
- virtual void printInfo ()
- void saveLuminosity ()
- void saveUpeR ()
- void **allocateLpv** ()
- void **allocateLvPoint** ()
- void **allocateLvPointAvg** ()
- void **allocateUpeR** ()
- void **allocateUpe** ()
- void **allocateGammaKN** ()
- void **freeLpv** ()
- void **freeLvPoint** ()
- void **freeLvPointAvg** ()
- void **freeUpe** ()
- void **freeUpeR** ()
- void **freeGammaKN** ()
- double **get_ep** (int i)
- double **get_upe** (int i)
- double **get_upe_r** ()
- double **get_vPoint** (int i)
- double **get_LvPoint** (int i)
- double **get_LvPointAvg** (int i)
- double **get_vp** (int i)
- double **get_Lpv** (int i)
- double **get_gammaKN** ()
- void **set_ep** (int i, double val)
- void **set_upe** (int i, double val)
- void **set_upe_r** (double val)
- void **set_vPoint** (int i, double val)
- void **set_LvPoint** (int i, double val)
- void **set_LvPointAvg** (int i, double val)
- void **set_vp** (int i, double val)
- void **set_Lpv** (int i, double val)
- double **set_gammaKN** (double val)
- void set_KN_info (double g)
- void print_KN_info ()
- double getdLogE ()

Public Attributes

- electrons * ele
- gsl_vector * ep
- gsl_matrix * upe
- gsl_vector * vp
- gsl_vector * Lpv
- gsl_vector * vPoint
- gsl_vector * LvPoint
- gsl_vector * LvPointAvg
- gsl_vector * upe_r
- double injRm
- int luminosityConstU
- int luminosityConstNu
- gsl_vector * gammaKN
- double epMin
- double **epMax**
- double vpMin
- double **vpMax**
- double dLogE
- bool flag_upe_r

4.11.2 Constructor & Destructor Documentation

4.11.2.1 energyDissProc::energyDissProc (scfgp * *cfg,* jetGeometry * *r,* electrons * *ele,* std::string *_id*)

constructor

Parameters

scfgp	
jetGeometry	
electrons	
id	

```
9                                                                                                    :
      baseClass(_cfg, _r, _id ), ele( _ele ), flag_upe_r( false ) {
10
11    /* request parameters */
12    cfg -> request<int>("saveLum",0,&saveLum);
13    cfg -> request<int>("saveUpeVsR",0,&saveUpeVsR);
14    cfg -> request<double>("injRm",2.0e17,&injRm);
15
16    cfg -> updateRequests( );
17
18    /* allocate common memory */
19    allocateGammaKN(); }
```

4.11.2.2 energyDissProc::~energyDissProc ( )

destructor

```
21                                    {
22    freeGammaKN( );
23    ele = NULL; }
```

4.11.3 Member Function Documentation

4.11.3.1 virtual double energyDissProc::calculateLpv (double *v*) [inline],[virtual]

calculate intrinsic luminosity

Parameters

v	- frequency (jet co-moving frame)

Returns

L'_v

Reimplemented in synchrotron, and selfSynCompton.

```
86 { };
```

4.11.3.2 virtual double energyDissProc::dotg (double *g*) [inline], [virtual]

get d gamma \ d t for particular process

Parameters

g	- electron Lorentz factor

Returns

d gamma \ d t

Reimplemented in externalRadiationPlanar, externalRadiationSpherical, externalRadiationGu, synchrotron, selfSynCompton, and externalRadiation.

```
94 { };
```

4.11.3.3 double energyDissProc::getdLogE ( ) [inline]

get log-energy step

Returns

log-energy step

```
152 { return dLogE; }
```

4.11.3.4 void energyDissProc::print_KN_info ( )

print info whether particular process for particular electron energy is withing Klein-Nishina region already

```
96                                      {
97   std::cout << " KN info "+this->whoAmI() << "\tradius: " << r->get() << "\tgammaKN: ";
98   if( get_gammaKN() > ele->getGammaMax( ) || get_gammaKN( ) == 0.0 ) {
99     std::cout << "> gammaMax = " << std::scientific << std::setprecision(2) <<
      ele->getGammaMax( ) << std::endl; }
100   else { std::cout << std::scientific << std::setprecision(2) << get_gammaKN() << std::endl; }
101 }
```

4.11.3.5 virtual void energyDissProc::printInfo ( ) [inline], [virtual]

print basic information about myself (virtual)

Reimplemented from baseClass.

Reimplemented in externalRadiationPlanar, DtSpherical, BlrPlanar, synchrotron, BlrSpherical, externalRadiationSpherical, DtPlanar, selfSynCompton, AccdPlanar, externalRadiationGu, BlrSphericalGu, DtSphericalGu, AccdSphericalGu, and externalRadiation.

```
95 { };
```

4.11.3.6 void energyDissProc::saveLuminosity ( )

save calulated luminosity

```
103                                    {
104   std::string type = "Lpv_"+this->whoAmI( );
105   bazinga::info(this->whoAmI( ),"Saving luminosity.");
106   if( saveLum && r->ifSaveRadius( ) ) { bazinga::save_GSLVector( type, this->vp,
      ele->gamma, this->Lpv, r->getPosition( ), cfg->get<std::string>("output") ); }
107 }
```

4.11.3.7 void energyDissProc::saveUpeR ( )

save energy density vs radius info

```
109                                  {
110   std::string type = "UpeR_"+this->whoAmI( );
111   bazinga::info(this->whoAmI( ),"Saving upe vs radius info.");
112   if( saveUpeVsR ) { bazinga::save_GSLVector( type, r->getRadius_GSLVector( ), this->
      upe_r, cfg->get<std::string>("output") ); }
113 }
```

4.11.3.8 void energyDissProc::set_KN_info (double *g*)

set information about Klein-Nishina regime

Parameters

-	electron Lorentz factor

```
91                                          {
92   if( get_gammaKN( ) == 0 ) { set_gammaKN( g ); }
93   else if( get_gammaKN( ) > g ) { set_gammaKN( g ); }
94 }
```

4.11.3.9 virtual void energyDissProc::setLpv ( ) [inline], [virtual]

set intrinsic luminosities

Reimplemented in externalRadiationPlanar, externalRadiationSpherical, externalRadiationGu, synchrotron, selfSynCompton, and externalRadiation.

```
81 { };
```

4.11.3.10 virtual void energyDissProc::update ( ) [inline], [virtual]

update all process internal parameters to current radius

Reimplemented in externalRadiationPlanar, synchrotron, DtSpherical, externalRadiationSpherical, selfSynCompton, BlrSpherical, externalRadiationGu, BlrSphericalGu, DtSphericalGu, AccdSphericalGu, and externalRadiation.

```
89 { };
```

4.11.4 Member Data Documentation

4.11.4.1 double energyDissProc::dLogE

ep integration step

4.11.4.2 electrons* energyDissProc::ele

class needs to have access to electrons to calculate luminosities etc

4.11.4.3 gsl_vector* energyDissProc::ep

data matrices and vectors; first index is radius photon field energy - primed

4.11.4.4 double energyDissProc::epMin

boundary energies

4.11.4.5 bool energyDissProc::flag_upe_r

set this to true after setting upe_r

4.11.4.6 gsl_vector* energyDissProc::gammaKN

data vector to store information on electrons energies that cool in the Klein-Nishina regime for each radius a data on electrons in KN in being saved

4.11.4.7 double energyDissProc::injRm

used to specify radial maximum of injected electrons - used only for non-uniform injections

4.11.4.8 gsl_vector* energyDissProc::Lpv

monochromatic apparent luminosity - primed

4.11.4.9 int energyDissProc::luminosityConstNu

set this to 1 if luminosity is to be calculated with constant v'; in such case v' will be set to a constant value given by v'(injrm)

4.11.4.10 int energyDissProc::luminosityConstU

set this to 1 if luminosity is to be calculated with constant u_ext; in such case u' will be set to a constant value given by u'(injrm)

4.11.4.11 gsl_vector* energyDissProc::LvPoint

monochromatic apparent luminosity for point source

4.11.4.12 gsl_vector* energyDissProc::LvPointAvg

monochromatic apparent luminosity for point source averaged over radius

4.11.4.13 gsl_matrix* energyDissProc::upe

photon field monochromatic energy density - primed

4.11.4.14 gsl_vector* energyDissProc::upe_r

data vector for storing energy density vs radius

4.11.4.15 gsl_vector* energyDissProc::vp

frequency - primed

4.11.4.16 double energyDissProc::vpMin

boundary frequencies

4.11.4.17 gsl_vector* energyDissProc::vPoint

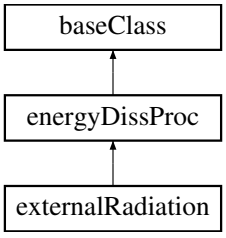
freqency - point source

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

- /home/mjaniak/Soft/blazar++/include/energyDissProc.hpp
- /home/mjaniak/Soft/blazar++/src/energyDissProc.cpp

4.12 externalRadiation Class Reference

Inheritance diagram for externalRadiation:



Public Member Functions

- **externalRadiation** (scfgp *_cfg, jetGeometry *_r, electrons *_ele, std::string _id)
- double dotg (double g)
- void update ()
- void setLpv ()
- double **calculateLpv** (double v, double theta)
- void printInfo ()

Additional Inherited Members

4.12.1 Member Function Documentation

4.12.1.1 double externalRadiation::dotg (double *g*) [virtual]

get d gamma \ d t for particular process

Parameters

g	- electron Lorentz factor

Returns

d gamma \ d t

Reimplemented from energyDissProc.

```
72                                          {
73   double b, sum, val = 0.0;
74   for( int i=0;i<N;i++ ) {
75       b = 4.0*get_ep(i)*g;
76       if( b > 1.0 ) { set_KN_info( g ); }
77       sum += bazinga::IntCor(i,N)*get_ep(i)*get_upe(i)*inverseCompton::fKN(b,KN); }
78
79   val = dLogE*sum;
80   set_upe_r( val );
81   return val; }
```

4.12.1.2 void externalRadiation::printInfo ( ) [virtual]

print basic information about myself (virtual)

Reimplemented from energyDissProc.

```
57                                      {
58   bazinga::info(id,"Info");
59   bazinga::print_info(id,"N",N);
60   bazinga::print_info(id,"radius R",extr,"cm");
61   bazinga::print_info(id,"Index kERC",extk);
62   bazinga::info(id,"(in electrons co-moving frame)");
63   bazinga::print_info(id,"Avg energy",exte,"eV");
64   bazinga::print_info(id,"Energy density",extu,"erg cm-3"); }
```

4.12.1.3 void externalRadiation::setLpv ( ) [virtual]

set intrinsic luminosities

Reimplemented from energyDissProc.

```
83                                  {
84   for( int i=0;i<N;i++ ) { set_Lpv( i, calculateLpv( get_vp(i), thetaObs ) ); }
85 }
```

4.12.1.4 void externalRadiation::update ( ) [virtual]

update all process internal parameters to current radius

Reimplemented from energyDissProc.

```
66                                  {
67   for (int i=0;i<N;i++ ) {
68     ksi = get_ep(i)/TempX;
69     set_upe( i, normA*pow(ksi,3)/(exp(ksi)-1.0)/(1.0e0+pow(r->get()/extr,extk) ) ); }
70   flag_upe_r = false; }
```

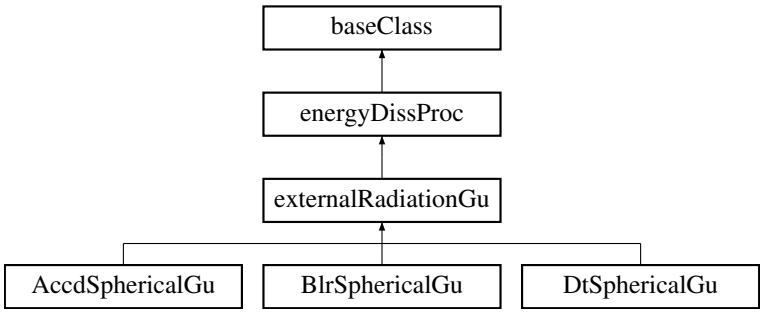
The documentation for this class was generated from the following files:

- /home/mjaniak/Soft/blazar++/include/externalRadiation.hpp
- /home/mjaniak/Soft/blazar++/src/externalRadiation.cpp

4.13 externalRadiationGu Class Reference

`#include <externalRadiationGu.hpp>`

Inheritance diagram for externalRadiationGu:



4.13.1 Detailed Description

Class provides electron energy losses and luminosity calculation for inverse Compton process (ERC) for seed photons comming from BLR and HDR modelled as quasi-spherical sources with u_ext modified by g_u parameter

Public Member Functions

- externalRadiationGu (scfgp *_cfg, jetGeometry *_r, electrons *_ele, std::string _id)
- ~externalRadiationGu ()
- virtual void printInfo ()
- virtual void update ()
- virtual double getvext (double _r)
- virtual void setRadius ()
- double dotg (double g)
- double calculateLpv (double v, double theta)
- void setLpv ()

Public Attributes

- double **Gamma**
- double **thetaObs**

- double **mBH**
- double **eDisk**
- double **mDot**
- int **KN**

4.13.2 Constructor & Destructor Documentation

4.13.2.1 externalRadiationGu::externalRadiationGu (scfgp * *_cfg,* jetGeometry * *_r,* electrons * *_ele,* std::string *_id*)

constructor

Parameters

_cfg	- scfgp class object
_r	- jetGeometry class object
_ele	- electrons class object
_id	

```
9                                                                                                                    :
     energyDissProc( _cfg, _r, _ele, _id ) {
10   /* requested parameters */
11   cfg -> request<int>("N" + id, 200, &N );
12   cfg -> request<double>("Gamma", 10.0, &Gamma );
13   cfg -> request<double>("thetaObs", 0.1, &thetaObs );
14   cfg -> request<int>("KN", 0, &KN );
15   cfg -> request<double>("mBH",1.0,&mBH);
16   cfg -> request<double>("eDisk",0.1,&eDisk);
17   cfg -> request<double>("mDot",1.0,&mDot);
18   cfg -> request<int>(id+"LuminosityConstU",0,&luminosityConstU);
19   cfg -> request<int>(id+"LuminosityConstNu",0,&luminosityConstNu);
20
21   cfg -> updateRequests( ); }
```

4.13.2.2 externalRadiationGu::~externalRadiationGu ( )

destructor

```
23                                                {
24   freeLpv( );
25   freeLvPoint( );
26   freeLvPointAvg( );
27   freeUpeR( ); }
```

4.13.3 Member Function Documentation

4.13.3.1 double externalRadiationGu::calculateLpv (double *v,* double *theta*)

calculate intrinsic luminosity

Parameters

v	- frequency (jet co-moving frame)

Returns

L'_v

```
36                                                                {
37   double Int = 0.0;
38   double e, miu, ep;
39
40   e = PLANCK_H*v/mec2;
41   miu = -( cos(theta)-beta(Gamma) )/( 1.0-beta(Gamma)*cos(theta) );
```

```
42   ep = getvext( r->get() )*PLANCK_H*Gamma/mec2;
43
44   for( int k=0;k<ele->getN( );k++ )
45     Int  += bazinga::IntCor( k, ele->getN( ) )*inverseCompton::f( ele->
      getGamma(k), ep, e, miu )*ele->getNgamma(k)/ele->
      getGamma(k);
46
47   if( luminosityConstU )
48     { Int *= gsl_vector_get( upe_r, 0 )*ele->getdLogGamma( )/pow(
      getvext( r->get() ), 2.0 ); }
49   else
50     { Int *= gsl_vector_get( upe_r, r->getIndex( ) )*ele->getdLogGamma( )/pow(
      getvext( r->get( ) ), 2.0 ); }
51
52   Int *= (3.0*SIGMA_T*LIGHT_SPEED)/(4.0*pow(Gamma,2.0));
53   Int *= v;
54   return Int; }
```

4.13.3.2 double externalRadiationGu::dotg (double *g*) [virtual]

get d gamma \ d t for particular process

Parameters

g	- electron Lorentz factor

Returns

d gamma \ d t

Reimplemented from energyDissProc.

```
29                                          {
30   double b, val = 0.0;
31   b = 4.0*g*getvext( r->get() )*PLANCK_H*Gamma/mec2;
32   if( b > 1.0 ) { set_KN_info( g ); }
33   val = gsl_vector_get( upe_r, r->getIndex( ) )*inverseCompton::fKN(b,KN);
34   return val; }
```

4.13.3.3 virtual double externalRadiationGu::getvext (double *_r*) [inline], [virtual]

get v_ext value in Hz

Parameters

_r	- radius

Returns

v_ext

Reimplemented in BlrSphericalGu, DtSphericalGu, and AccdSphericalGu.

```
46 { }
```

4.13.3.4 virtual void externalRadiationGu::printInfo ( ) [inline], [virtual]

print basic information about myself (virtual)

Reimplemented from energyDissProc.

Reimplemented in BlrSphericalGu, DtSphericalGu, and AccdSphericalGu.

```
38 { }
```

4.13.3.5 void externalRadiationGu::setLpv ( ) [virtual]

set intrinsic luminosities

Reimplemented from energyDissProc.

```
56                                  {
57   for( int i=0;i<N;i++ ) { set_Lpv( i, calculateLpv( get_vp(i),thetaObs ) ); }
58 }
```

4.13.3.6 virtual void externalRadiationGu::setRadius ( ) [inline], [virtual]

set radial boundaries for processes

Reimplemented in BlrSphericalGu, DtSphericalGu, and AccdSphericalGu.

```
49 { }
```

4.13.3.7 virtual void externalRadiationGu::update ( ) [inline], [virtual]

calculate and set Upe every time with new radius r

Reimplemented from energyDissProc.

Reimplemented in BlrSphericalGu, DtSphericalGu, and AccdSphericalGu.

```
41 { }
```

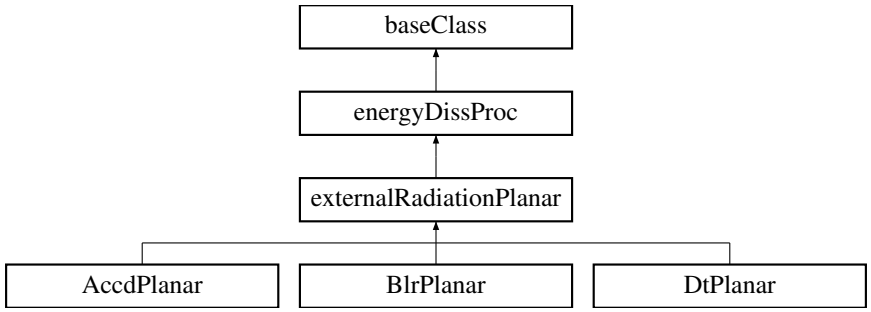
The documentation for this class was generated from the following files:

- /home/mjaniak/Soft/blazar++/include/externalRadiationGu.hpp
- /home/mjaniak/Soft/blazar++/src/externalRadiationGu.cpp

4.14 externalRadiationPlanar Class Reference

`#include <externalRadiationPlanar.hpp>`

Inheritance diagram for externalRadiationPlanar:



4.14.1 Detailed Description

Class provides electron energy losses and luminosity calculation for inverse Compton process (ERC) for seed photons comming from accretion disk, BLR and HDR modelled as planar source. Incoming photons are Doppler shifted depending on the geometry. Energy density of seed photons is calculated consistently based on the source characteristics in the accretion disk plane

Parameters

alpha	- radiation vs R slope index
R1	- inner edge
R2	- outer edge
s	- radiation vs R slope index
approx	- approximation calculations switch
fixedUpe	- fix u_ext vs r
saveRm	- switch to save information about R_m
R	- hirozontal disk plane radius
rm	- R_m vector info
dLdR	- vector to store information about dLdR

Public Member Functions

- externalRadiationPlanar (scfgp ∗_cfg, jetGeometry ∗_r, electrons ∗_ele, std::string _id)
- ∼externalRadiationPlanar ()
- virtual void printInfo ()
- virtual void setRadius ()
- virtual double getvext (double _R)
- virtual void setdLdR ()
- void update ()
- double calculateUpeFull ()
- double calculateUpeApprox (int index, bool flag)
- double function_upe (int index)
- double ctau ()
- double getdLdR (int index)
- double dotg (double g)
- void setLpv ()
- double calculateLpv (double v, double theta)
- double fsc_integral (double v, double theta, double R)
- double function_lum (int index, double v, double theta)
- void **allocateRm** ()
- void **allocatedLdR** ()
- void **freeRm** ()
- double getRm ()
- void setRm (double val)
- gsl_vector ∗ getRmVector ()
- void saveRmVsR ()

Protected Attributes

- double **alpha**
- double **R1**
- double **R2**
- double **s**
- int **approx**
- double **fixedUpe**
- double **Gamma**
- double **thetaObs**
- int **KN**
- double **mBH**
- double **eDisk**
- double **mDot**

- int **saveRm**
- logGeometry * **R**
- gsl_vector * **rm**
- gsl_vector * **dLdR**
- double **Rm**
- double **Upe**
- int **Rm_index**

Additional Inherited Members

4.14.2 Constructor & Destructor Documentation

4.14.2.1 externalRadiationPlanar::externalRadiationPlanar (scfgp * *_cfg,* jetGeometry * *_r,* electrons * *_ele,* std::string *_id*)

constructor

Parameters

_cfg	- scfgp class object
_r	- jetGeometry class object
_ele	- electrons class object
_id	

```
9
          : energyDissProc( _cfg, _r, _ele, _id ) {
10    /* requested parameters */
11    /* R1, R2, s, alpha - needed by Accd, BlrPl, DtPl */
12    cfg -> request<int>("N" + id, 200, &N );
13    cfg -> request<double>(id +  "R1", 0.0, &R1 );
14    cfg -> request<double>(id +  "R2", 0.0, &R2 );
15    cfg -> request<double>(id +  "s", 0.0, &s );
16    cfg -> request<double>(id +  "alpha", 0.0, &alpha );
17    cfg -> request<double>("Gamma", 10.0, &Gamma );
18    cfg -> request<double>("thetaObs", 0.1, &thetaObs );
19    cfg -> request<int>("KN", 0, &KN );
20    cfg -> request<double>("mBH",1.0,&mBH);
21    cfg -> request<double>("eDisk",0.1,&eDisk);
22    cfg -> request<double>("mDot",1.0,&mDot);
23    cfg -> request<int>(id+"LuminosityConstU",0,&luminosityConstU);
24    cfg -> request<int>(id+"LuminosityConstNu",0,&luminosityConstNu);
25
26    /* works for both stratifications and for Accd */
27    cfg -> request<int>("extPlApp",0,&approx);
28    cfg -> request<int>("saveRmVsR",0,&saveRm);
29
30    cfg -> updateRequests( );
31
32    allocatedLdR( );
33    allocateRm( ); }
```

4.14.2.2 externalRadiationPlanar::~externalRadiationPlanar ( )

destructor

```
35                                                  {
36    freeLpv( );
37    freeLvPoint( );
38    freeLvPointAvg( );
39    freeUpe( );
40    freeRm( );
41
42    delete R;
43    R = NULL; }
```

4.14.3 Member Function Documentation

4.14.3.1 double externalRadiationPlanar::calculateLpv (double *v*, double *theta*)

calculate intrinsic luminosity

Parameters

v	- frequency (jet co-moving frame)

Returns

L'_v

```
152                                                                    {
153   double val = 0.0;
154   if( approx ) { val = function_lum( Rm_index, v, theta )*Rm; }
155   else {
156     for( int i=0;i<R->getMaxIndex();i++ ) {
157       R -> update( i );
158       val += bazinga::IntCor( i, R->getN( ) )*function_lum( i, v, theta )*R->getDr(i); }
159   }
160
161   val *= 3.0*SIGMA_T/(16.0*M_PI);
162   val *= v;
163   return val; }
```

4.14.3.2 double externalRadiationPlanar::calculateUpeApprox (int *index*, bool *flag*)

caculate u'_ext in the jet co-moving frame using approximate equations

Parameters

i	- index in radius R at which u'_ext is calculated
flag	- obsolete ?

Returns

u'_ext

```
89 { return function_upe( index )*R->get(index); }
```

4.14.3.3 double externalRadiationPlanar::calculateUpeFull ( )

caculate u'_ext in the jet co-moving frame using full equations

Returns

u'_ext

```
72                                                {
73   double _Int = 0.0;
74   gsl_vector *_temp_function_upe_vector = gsl_vector_alloc( N );
75   double _function_upe_value = 0;
76
77   for( int i=0;i<R->getMaxIndex();i++ ) {
78     gsl_vector_set( _temp_function_upe_vector, i, function_upe( i ) );
79     _Int += bazinga::IntCor( i, R->getN( ) )*gsl_vector_get( _temp_function_upe_vector, i)*R->getDr(i);
80   }
81
82   /* before quit save _temp_function_upe_vector */
83   bazinga::save_GSLVector( "dUpedR_"+this->whoAmI( ), R->getRadius_GSLVector( ),
      _temp_function_upe_vector, R->getPosition( ), cfg->get<std::string>("output") );
84   gsl_vector_free( _temp_function_upe_vector );
85   _temp_function_upe_vector = NULL;
86
87   return _Int; }
```

4.14.3.4 double externalRadiationPlanar::ctau ( )

auxillary function to alculate u'_ext - normalization factor

```
98                                          {
99   double val = 0.0;
100   if( s == 1 ) { val = 1.0/log(R2/R1); }
101   else { val = (1.0-s)/(pow(R2,1.0-s)-pow(R1,1.0-s)); }
102   return val; }
```

4.14.3.5 double externalRadiationPlanar::dotg (double *g*) [virtual]

get d gamma \ d t for particular process

Parameters

g	- electron Lorentz factor

Returns

d gamma \ d t

Reimplemented from energyDissProc.

```
106                                                 {
107   double val = 0.0;
108   double b = 0.0;
109   double r2R2 = pow(r->get(),2.0)+pow(Rm,2.0);
110
111   double doppler_ext = 1.0/( Gamma*( 1.0-beta(Gamma)*(r->get()/sqrt(r2R2)) ) );
112   /* as e_ext max we take e_ext @ Rm where it is maximum; in case of BLRPL it makes no difference at all */
113   b = 4.0*g*getvext( Rm )*PLANCK_H/(mec2*doppler_ext);
114
115   if( b > 1.0 ) { set_KN_info( g ); }
116   val = Upe*inverseCompton::FKN( b, alpha, KN ); /* fKN not used anymore; using FKN instead */
117
118   return val; }
```

4.14.3.6 double externalRadiationPlanar::fsc_integral (double *v*, double *theta*, double *R*)

function that calculates intrinsic ERC luminosity

Parameters

v	- frequency
theta	- observer angle
R	- radius in the accretion disk plane

Returns

L'

```
122                                                                                 {
123   double Int = 0.0;
124   double ee = PLANCK_H*v/(ELECTRON_MASS*LIGHT_SPEED*LIGHT_SPEED);
125   double miu = -(cos(theta)-beta(Gamma))/(1.0-beta(Gamma)*cos(theta));
126
127   double r2R2, doppler_ext;
128   if( luminosityConstU ) {
129     r2R2 = pow(r->getRInjMax( ),2.0)+pow(R,2.0);
130     doppler_ext = 1.0/( Gamma*( 1.0-beta(Gamma)*(r->getRInjMax( )/sqrt(r2R2)) ) ); }
131   else {
132     r2R2 = pow(r->get(),2.0)+pow(R,2.0);
133     doppler_ext = 1.0/( Gamma*( 1.0-beta(Gamma)*(r->get()/sqrt(r2R2)) ) ); }
134
135   /* integration loop over electrons */
136   for( int k=0;k<ele->getN( );k++ ) { Int += bazinga::IntCor( k, ele->
```

```
      getN( ) )*inverseCompton::f(ele->getGamma(k), PLANCK_H*getvext( R )/(mec2*doppler_ext
      ), ee, miu )*ele->getNgamma(k)/ele->getGamma(k); }
137
138   Int *= ele->getdLogGamma( );
139   return Int; }
```

4.14.3.7 double externalRadiationPlanar::function_lum (int *index*, double *v*, double *theta*)

auxillary function to calculate L'

Parameters

index	- index in radius R at which L' is calculated
v	- frequency
theta	- observer angle

```
141                                                                          {
142   double r2R2, doppler_ext;
143   if( luminosityConstU ) {
144     r2R2 = pow(r->getRInjMax( ),2.0)+pow(R->get(index),2.0);
145     doppler_ext = 1.0/( Gamma*( 1.0-beta(Gamma)*(r->getRInjMax( )/sqrt(r2R2)) ) ); }
146   else {
147      r2R2 = pow(r->get(),2.0)+pow(R->get(index),2.0);
148     doppler_ext = 1.0/( Gamma*( 1.0-beta(Gamma)*(r->get()/sqrt(r2R2)) ) ); }
149
150   return getdLdR(index)*fsc_integral( v, theta, R->get(index) )/(r2R2*pow(
      getvext( R->get(index) ), 2 )); }
```

4.14.3.8 double externalRadiationPlanar::function_upe (int *index*)

auxillary function to calculate u'_ext

Parameters

index	- index in radius R at which u'_ext is calculated

```
91                                                    {
92   double val;
93   double r2R2 = pow(r->get(),2.0)+pow(R->get(index),2);
94   val = getdLdR(index)*pow(1.0-((beta(Gamma)*r->get())/sqrt(r2R2)),2)/r2R2;
95   val *= pow(Gamma,2)/(4.0*M_PI*LIGHT_SPEED);
96   return val; }
```

4.14.3.9 double externalRadiationPlanar::getdLdR (int *index*)

get fdLdR

Parameters

index	in radius R at which dLdR is calculated

```
104 { return gsl_vector_get( dLdR, index ); }
```

4.14.3.10 double externalRadiationPlanar::getRm ( ) [inline]

get R_m - radius at which contribution of u'_ext is maximal

Returns

R_m

```
126 { return gsl_vector_get( rm, r->getIndex( ) ); }
```

4.14.3.11 gsl_vector* externalRadiationPlanar::getRmVector ( ) [inline]

get vector with R_m values stored

```
134 { return rm; }
```

4.14.3.12 virtual double externalRadiationPlanar::getvext (double _R) [inline], [virtual]

get v_ext value in Hz

Parameters

_R	- radius in accretion disk plane

Returns

v_ext

Reimplemented in BlrPlanar, DtPlanar, and AccdPlanar.

```
68 { return 0; }
```

4.14.3.13 virtual void externalRadiationPlanar::printInfo ( ) [inline], [virtual]

print basic information about myself (virtual)

Reimplemented from energyDissProc.

Reimplemented in BlrPlanar, DtPlanar, and AccdPlanar.

```
60 { }
```

4.14.3.14 void externalRadiationPlanar::saveRmVsR ( )

save vector with R_m values stored

```
175                                          {
176  if( saveRm ) { bazinga::save_GSLVector( "Rm_"+ this->whoAmI( ), r->getRadius_GSLVector( ),
      getRmVector( ), cfg->get<std::string>("output") ); }
177 }
```

4.14.3.15 virtual void externalRadiationPlanar::setdLdR ( ) [inline], [virtual]

set dL/dR for particulat source

Reimplemented in BlrPlanar, DtPlanar, and AccdPlanar.

```
71 { }
```

4.14.3.16 void externalRadiationPlanar::setLpv ( ) [virtual]

set intrinsic luminosities

Reimplemented from energyDissProc.

```
120 { for( int i=0;i<N;i++ ) set_Lpv( i, calculateLpv( get_vp(i), thetaObs )); }
```

4.14.3.17 virtual void externalRadiationPlanar::setRadius ( ) [inline], [virtual]

sets rext if not provided by config file

Reimplemented in BlrPlanar, DtPlanar, and AccdPlanar.

```
63 { }
```

4.14.3.18 void externalRadiationPlanar::setRm (double *val*) [inline]

set R_m - radius at which contribution of u'_ext is maximal

Parameters

val	- current R_m

```
131 { gsl_vector_set( rm, r->getIndex( ), val ); }
```

4.14.3.19 void externalRadiationPlanar::update ( ) [virtual]

update all process internal parameters to current radius

Reimplemented from energyDissProc.

```
46                                      {
47   /* even though we use full version we need to calculate the Rm for gamma_dot */
48   double oldUpe, currentUpe = 0.0;
49   for( int i=0;i<R->getMaxIndex();i++ ) {
50     R -> update( i );
51     oldUpe = currentUpe;
52     currentUpe = calculateUpeApprox( i, false );
53     set_upe( i, currentUpe );
54     if( currentUpe > oldUpe ) {
55       Rm = R->get( );
56       Rm_index = i; }
57
58     if( Rm < R1 ) { Rm = R1; Rm_index = 0; }
59     if( Rm > R2 ) { Rm = R2; Rm_index = R -> getMaxIndex( ); }
60   }
61
62   setRm( Rm );
63   bazinga::print_info(id,"Radius Rm", getRm( ) );
64
65   if( approx ) { Upe = calculateUpeApprox( Rm_index, true ); }  /* approximate version */
66   else { Upe = calculateUpeFull( ); }  /* full integral version */
67   bazinga::print_info(id,"Upe",Upe);
68
69   set_upe_r( Upe ); /* this is to make a plot gamma_dot vs r */
70   flag_upe_r = false; }
```

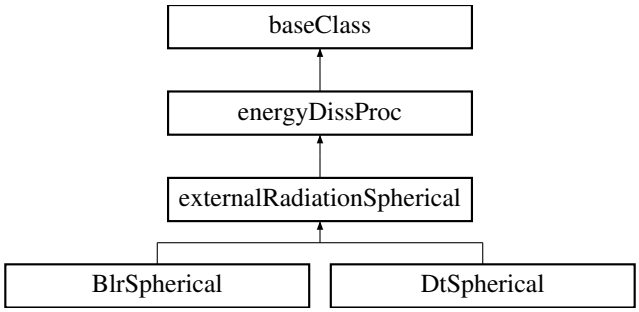
The documentation for this class was generated from the following files:

- /home/mjaniak/Soft/blazar++/include/externalRadiationPlanar.hpp
- /home/mjaniak/Soft/blazar++/src/externalRadiationPlanar.cpp

4.15 externalRadiationSpherical Class Reference

`#include <externalRadiation.hpp>`

Inheritance diagram for externalRadiationSpherical:



4.15.1 Detailed Description

Class provides electron energy losses and luminosity calculation for inverse Compton process (ERC) for seed photons comming from BLR and HDR modelled as spherical sources. Implementation provided by the class is identical to this in original BLAZAR code from Moderski et al. (2003). it is not modified OR checked for consistency. Use with caution!

Class provides electron energy losses and luminosity calculation for inverse Compton process (ERC) for seed photons comming from BLR and HDR modelled as spherical sources

Public Member Functions

- externalRadiationSpherical (scfgp ∗_cfg, jetGeometry ∗_r, electrons ∗_ele, std::string _id)
- ∼externalRadiationSpherical ()
- virtual x void setRadius ()
- virtual void printInfo ()
- virtual void update ()
- virtual double getvext (double _r)
- double dotg (double g)
- double calculateLpv (double v, double theta)
- void setLpv ()

Public Attributes

- double **cf**
- double **e**
- double **Gamma**
- double **thetaObs**
- int **KN**
- gsl_vector ∗ **upe**

4.15.2 Constructor & Destructor Documentation

4.15.2.1 externalRadiationSpherical::externalRadiationSpherical (scfgp ∗ *_cfg,* jetGeometry ∗ *_r,* electrons ∗ *_ele,* std::string *_id*)

constructor

Parameters

_cfg	- scfgp class object
_r	- jetGeometry class object

_ele	- electrons class object
_id	

```
9
                : energyDissProc( _cfg, _r, _ele, _id ) {
10   /* requested parameters */
11   cfg -> request<int>("N" + id, 200, &N );
12   cfg -> request<double>(id + "e", 1.0, &e );
13   cfg -> request<double>(id + "cf", 0.1, &cf );
14   cfg -> request<double>("Gamma", 10.0, &Gamma );
15   cfg -> request<double>("thetaObs", 0.1, &thetaObs );
16   cfg -> request<int>("KN", 0, &KN );
17   cfg -> request<int>(id+"LuminosityConstU",0,&luminosityConstU);
18   cfg -> request<int>(id+"LuminosityConstNu",0,&luminosityConstNu);
19
20   cfg -> updateRequests( );
21
22   /* allocate space for upe vector */
23   upe = gsl_vector_alloc( r->getMaxIndex( ) );
24   gsl_vector_set_zero( upe );
25
26   allocateUpeR( ); }
```

4.15.2.2 externalRadiationSpherical::~externalRadiationSpherical ( )

destructor

```
28                                                          {
29   freeUpe( );
30   freeLpv( );
31   freeLvPoint( );
32   freeLvPointAvg( );
33   freeUpeR( );
34
35   /* free space for upe vector */
36   gsl_vector_free( upe );
37   upe = NULL; }
```

4.15.3 Member Function Documentation

4.15.3.1 double externalRadiationSpherical::calculateLpv (double *v*, double *theta*)

calculate intrinsic luminosity

Parameters

v	- frequency (jet co-moving frame)

Returns

L'_v

```
46                                                                  {
47   double Int = 0.0;
48   double e, miu, ep;
49
50   e = PLANCK_H*v/mec2;
51   miu = -( cos(theta)-beta(Gamma) )/( 1.0-beta(Gamma)*cos(theta) );
52   ep = getvext( r->get( ) )*PLANCK_H*Gamma/mec2;
53
54   for( int k=0;k<ele->getN( );k++ )
55     Int  += bazinga::IntCor( k, ele->getN( ) )*inverseCompton::f( ele->
      getGamma(k), ep, e, miu )*ele->getNgamma(k)/ele->
      getGamma(k);
56
57   if( luminosityConstU )
58     Int *= gsl_vector_get( upe, 0 )*ele->getdLogGamma( )/pow(
      getvext( r->get( ) ), 2.0 );
59   else
60     Int *= gsl_vector_get( upe, r->getIndex( ) )*ele->getdLogGamma( )/pow(
      getvext( r->get( ) ), 2.0 );
```

```
61
62   Int *= (3.0*SIGMA_T*LIGHT_SPEED)/(4.0*pow(Gamma,2.0));
63   Int *= v;
64   return Int; }
```

4.15.3.2 double externalRadiationSpherical::dotg (double *g*) [virtual]

get d gamma \ d t for particular process

Parameters

g	- electron Lorentz factor

Returns

d gamma \ d t

Reimplemented from energyDissProc.

```
39                                                  {
40   double b, val = 0.0;
41   b = 4.0*g*getvext( r->get( ) )*PLANCK_H*Gamma/mec2;
42   if( b > 1.0 ) { set_KN_info( g ); }
43   val = gsl_vector_get( upe, r->getIndex( ) )*inverseCompton::fKN(b,KN);
44   return val; }
```

4.15.3.3 virtual double externalRadiationSpherical::getvext (double *_r*) [inline], [virtual]

now obsolete virtual double getdLdlnr(double _r) { return 0; } get v_ext value in Hz

Parameters

_r	- radius

Returns

v_ext

Reimplemented in DtSpherical, and BlrSpherical.

```
52 { return 0; }
```

4.15.3.4 virtual void externalRadiationSpherical::printInfo ( ) [inline], [virtual]

print basic information about myself (virtual)

Reimplemented from energyDissProc.

Reimplemented in DtSpherical, and BlrSpherical.

```
41 { }
```

4.15.3.5 void externalRadiationSpherical::setLpv ( ) [virtual]

set intrinsic luminosities

Reimplemented from energyDissProc.

```
66                                          {
67   for( int i=0;i<N;i++ ) { set_Lpv( i, calculateLpv( get_vp(i),thetaObs ) ); }
68 }
```

4.15.3.6 virtual x void externalRadiationSpherical::setRadius ( ) [inline], [virtual]

sets rext if not provided by config file

Reimplemented in DtSpherical, and BlrSpherical.

```
40 { }
```

4.15.3.7 virtual void externalRadiationSpherical::update ( ) [inline], [virtual]

calculate and set Upe every time with new radius r

Reimplemented from energyDissProc.

Reimplemented in DtSpherical, and BlrSpherical.

```
44 { }
```

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

- /home/mjaniak/Soft/blazar++/include/externalRadiationSpherical.hpp
- /home/mjaniak/Soft/blazar++/src/externalRadiationSpherical.cpp

4.16 gamma_break_params Struct Reference

Public Attributes

- double **p1**
- double **p2**
- double **gamma_min**
- double **gamma_max**
- double **avg_gamma**

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

- /home/mjaniak/Soft/blazar++/include/electrons.hpp

4.17 jetGeometry Class Reference

Public Member Functions

- **jetGeometry** (scfgp *_cfg)
- void **update** (int index)
- double **show** ()
- double **get** ()
- double **get** (int index)
- int **getIndex** ()
- int **getMaxIndex** ()
- double **getDr** ()
- double **getDr** (int index)
- double **getPosition** ()
- double **getPosition** (int index)
- void **printInfo** ()

- int **getNinj** ()
- double **getR0** ()
- double **getRInjMax** ()
- double **getRMax** ()
- gsl_vector ∗ **getRadius_GSLVector** ()
- int **ifSaveRadius** ()

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

- /home/mjaniak/Soft/blazar++/include/jetGeometry.hpp
- /home/mjaniak/Soft/blazar++/src/jetGeometry.cpp

4.18 logGeometry Class Reference

Public Member Functions

- **logGeometry** (double _r1, double _r2, double _N)
- void **update** (int index)
- double **show** ()
- double **get** ()
- double **get** (int index)
- int **getIndex** ()
- int **getMaxIndex** ()
- double **getDr** ()
- double **getDr** (int index)
- double **getPosition** ()
- double **getPosition** (int index)
- void **printInfo** ()
- double **getR0** ()
- double **getRMax** ()
- int **getN** ()
- gsl_vector ∗ **getRadius_GSLVector** ()

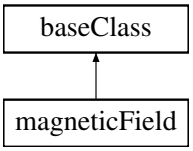
The documentation for this class was generated from the following files:

- /home/mjaniak/Soft/blazar++/include/logGeometry.hpp
- /home/mjaniak/Soft/blazar++/src/logGeometry.cpp

4.19 magneticField Class Reference

`#include <magneticField.hpp>`

Inheritance diagram for magneticField:



4.19.1 Detailed Description

Class defining magnetic field across the jet

Public Member Functions

- magneticField (scfgp ∗_cfg, jetGeometry ∗_r, std::string _id)
- ∼magneticField ()
- double getB ()
- double getB (double _r)
- double getMaxB ()
- double get_uB ()
- double get_uB (double _r)
- void printInfo ()

Additional Inherited Members

4.19.2 Constructor & Destructor Documentation

4.19.2.1 magneticField::magneticField (scfgp ∗ *_cfg,* jetGeometry ∗ *_r,* std::string *_id*)

consructor

Parameters

scfgp	
jetGeometry	
id	

```
8                                                                                :
      baseClass(_cfg, _r, _id ) {
9
10   /* request parameters */
11   cfg -> request<double>("B0",0.0,&B0);
12   cfg -> request<double>("B1",1.0,&B1);
13   cfg -> request<double>("sigmaB",0.01,&sigmaB);
14   cfg -> request<double>("kB",2.0,&kB);
15   cfg -> request<std::string>("magModel","blob",&magModel);
16   cfg -> request<double>("Gamma",10.0,&Gamma);
17   cfg -> request<double>("thetaJ",0.1,&thetaJ);
18   cfg -> request<double>("injRm",2.0e17,&injRm);
19   cfg -> request<double>("eDiss",0.1,&eDiss);
20   cfg -> request<double>("eEle",0.1,&eEle);
21   cfg -> request<double>("mBH",1.0,&mBH);
22   cfg -> request<double>("eJet",0.3,&eJet);
23   cfg -> request<double>("mDot",1.0,&mDot);
24
25   cfg -> updateRequests( );
26
27   /* set energetics and magnetic flux value if in 'steady' model */
28   if( magModel == "steady" ) {
29     double Ledd, Ljet;
30     Ledd = 1.3e47*mBH;
31     Ljet = 0.5*eJet*mDot*Ledd;
32     Lb = sigmaB*(1.0-eDiss)*Ljet/(1.0+sigmaB);
33     B0steady = sqrt(6.0*Lb/(LIGHT_SPEED*beta(Gamma)))/(thetaJ*Gamma); }
34 }
```

4.19.2.2 magneticField::∼magneticField ( )

destructor

4.19.3 Member Function Documentation

4.19.3.1 double magneticField::get_uB ( )

get current value of magnetic energy density

```
65 { return( DSQR( getB( ) )/(8.0*M_PI) ); }
```

4.19.3.2 double magneticField::get_uB (double _r)

get value of magnetic energy density at aspecific radius

Parameters

radius	

```
66 { return( DSQR( getB( _r ) )/(8.0*M_PI) ); }
```

4.19.3.3 double magneticField::getB ( )

get current value of magnetic field

```
50                              {
51   if( magModel == "steady" ) { return B0steady/r->get( ); };
52   if( magModel == "blob" ) { return B0+(B1*pow(injRm/r->get( ),0.5*kB)); }
53 }
```

4.19.3.4 double magneticField::getB (double _r)

get value of magnetic field at specific radius

Parameters

radius	

```
55                                      {
56   if( magModel == "steady" ) { return B0steady/_r; };
57   if( magModel == "blob" ) { return B0+(B1*pow(injRm/_r,0.5*kB)); }
58 }
```

4.19.3.5 double magneticField::getMaxB ( )

get maximum value of magnetic field (closest radius

```
60                                 {
61   if( magModel == "steady" ) { return B0steady/r->getR0( ); }
62   if( magModel == "blob" ) { return B0+(B1*pow( injRm/r->getR0( ),0.5*kB)); }
63 }
```

4.19.3.6 void magneticField::printInfo ( ) [virtual]

print basic information about myself (virtual)

Reimplemented from baseClass.

```
36                                 {
37   bazinga::info(id,"Info");
38   bazinga::print_info(id,"Magnetic field model",magModel);
39   if( magModel == "blob" ) {
40       bazinga::print_info(id,"B0",B0);
41       bazinga::print_info(id,"B1 @ InjRm",B1);
42       bazinga::print_info(id,"index kB",kB); }
43
44   if( magModel == "steady" ) {
45       bazinga::print_info(id,"sigmaB",sigmaB);
46       bazinga::print_info(id,"B @ injRm",B0steady/injRm);
47       bazinga::print_info(id,"Magnetic field flux", Lb);
48       bazinga::print_info(id,"thetaJ", thetaJ); } }
```

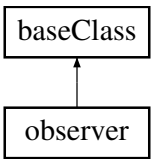
The documentation for this class was generated from the following files:

- /home/mjaniak/Soft/blazar++/include/magneticField.hpp
- /home/mjaniak/Soft/blazar++/src/magneticField.cpp

4.20 observer Class Reference

`#include <observer.hpp>`

Inheritance diagram for observer:



4.20.1 Detailed Description

Class is mainly used to calculate observed values from intrinsic jet properties such as observed luminosities instead of luminosities in jet co-moving frame or flare

Public Member Functions

- observer (scfgp ∗_cfg, jetGeometry ∗_r, std::string _id)
- ∼observer ()
- void printInfo ()
- int ifCalculateFlares ()
- int getFreqListSize ()
- double getFreqList (int i)
- void addEnDissProc (energyDissProc ∗_obj)
- void listEnDissProc ()
- int sizeEnDissProc ()
- void addEnDissProcPoint (energyDissProc ∗_obj)
- void listEnDissProcPoint ()
- int sizeEnDissProcPoint ()
- void addExtPl (energyDissProc ∗_obj)
- void listExtPl ()
- int sizeExtPl ()
- void addExtSp (energyDissProc ∗_obj)
- void listExtSp ()
- int sizeExtSp ()
- void addExtGu (energyDissProc ∗_obj)
- void listExtGu ()
- int sizeExtGu ()
- void calculatePointLuminosity (energyDissProc ∗_obj)
- void sumPointLuminosity ()
- void avgPointLuminosity (energyDissProc ∗_obj)
- void sumPointAvgLuminosity ()
- void addQAccdLuminosity (QuasarAccDisk ∗QAccd)
- double calculateFlare (energyDissProc ∗_obj, double nu)
- void calculateAllFlares ()

- double interpolate (gsl_vector *x, gsl_vector *y, double xToInterpolate)
- void savePointLuminosity (energyDissProc *x)
- void savePointLuminositySum ()
- void savePointLuminositySum (gsl_vector *gamma)
- void saveAveragedPointLuminosity (energyDissProc *x)
- void saveAveragedPointLuminositySum ()
- void saveAveragedPointLuminositySum (gsl_vector *gamma)

Public Attributes

- std::vector< energyDissProc * > EnDissProc
- std::vector< energyDissProc * > **EnDissProcPoint**
- std::vector< energyDissProc * > **ExtPl**
- std::vector< energyDissProc * > **ExtSp**
- std::vector< energyDissProc * > **ExtGu**
- gsl_vector * vPointSum
- gsl_vector * **LvPointSum**
- gsl_vector * **LvPointAvgSum**
- std::vector< double > freqList

4.20.2 Constructor & Destructor Documentation

4.20.2.1 observer::observer (scfgp * *_cfg,* jetGeometry * *_r,* std::string *_id*)

constructor

Parameters

scfgp	
jetgeometry	
id	

```
10                                                          : baseClass(_cfg, _r, _id ) {
11   /* request parameters */
12   cfg -> request<int>("N"+id,1000,&N);
13   cfg -> request<double>("thetaObs",0.1,&thetaObs);
14   cfg -> request<double>("thetaJ",0.1,&thetaJ);
15   cfg -> request<double>("nu1",0,&nu1);
16   cfg -> request<double>("nu2",0,&nu2);
17   cfg -> request<double>("nu3",0,&nu3);
18   cfg -> request<double>("nu4",0,&nu4);
19   cfg -> request<double>("nu5",0,&nu5);
20   cfg -> request<double>("nu6",0,&nu6);
21   cfg -> request<double>("Gamma",10.0,&Gamma);
22   cfg -> request<std::string>("lumModel","blob",&lumModel);
23   cfg -> request<int>("saveLumPoint",0,&saveLumPoint);
24   cfg -> request<int>("saveLumPointAvg",0,&saveLumPointAvg);
25   cfg -> request<int>("saveExtPlVsRm",0,&saveExtPlVsRm);
26
27   cfg -> updateRequests( );
28
29   allocateLvPointSum( );
30   allocateLvPointAvgSum( ); }
```

4.20.2.2 observer::~observer ( )

destructor

```
32                      {
33   freeLvPointSum( );
34   freeLvPointAvgSum( ); }
```

4.20.3 Member Function Documentation

4.20.3.1 void observer::addEnDissProc (energyDissProc * _obj)

add energy dissipation process to list of active ones

Parameters

energyDissProc	

```
79 { EnDissProc.push_back( _obj ); }
```

4.20.3.2 void observer::addEnDissProcPoint (energyDissProc * *_obj*)

add energy dissipation process to list of those for which we have to calculate 'point source' luminosity

Parameters

energyDissProc	

```
80 { EnDissProcPoint.push_back( _obj ); }
```

4.20.3.3 void observer::addExtGu (energyDissProc * *_obj*)

add ExtGu process to list of active ones

Parameters

energyDissProc	

```
83 { ExtGu.push_back( _obj ); }
```

4.20.3.4 void observer::addExtPl (energyDissProc * *_obj*)

add ExtPl process to list of active ones

Parameters

energyDissProc	

```
81 { ExtPl.push_back( _obj ); }
```

4.20.3.5 void observer::addExtSp (energyDissProc * *_obj*)

add ExtSp process to list of active ones

Parameters

energyDissProc	

```
82 { ExtSp.push_back( _obj ); }
```

4.20.3.6 void observer::addQAccdLuminosity (QuasarAccDisk * *QAccd*)

add Quasar Template to calculated luminosities

Parameters

QuasarAccDisk	

```
171                                                    {
172   bazinga::info(id,"Adding QAccd spectra to averaged point source luminosities ...");
173
174   double QAccd_vmin = gsl_vector_min( QAccd -> vTemplate );
175   double QAccd_vmax = gsl_vector_max( QAccd -> vTemplate );
176   bazinga::print_info(id, "QAccd vmin", QAccd_vmin );
177   bazinga::print_info(id, "QAccd vmax", QAccd_vmax );
178
179   double interpolatedValue = 0.0;
180   for( int i=0;i<N;i++ ) {
181     interpolatedValue = gsl_vector_get( LvPointAvgSum, i );
182     if( gsl_vector_get( vPointSum, i ) > QAccd_vmin && gsl_vector_get(
      vPointSum, i ) < QAccd_vmax )
183       interpolatedValue += interpolate( QAccd -> vTemplate, QAccd -> LvTemplate, gsl_vector_get(
       vPointSum, i ) );
184
185     gsl_vector_set( LvPointAvgSum, i, interpolatedValue );
186   }
187 }
```

4.20.3.7 void observer::avgPointLuminosity (energyDissProc * *_obj*)

calculate averaged 'point-source' luminosity; avergaed over all s-cells in the jet avtive region

Parameters

energyDissProc	

```
125                                                  {
126   bazinga::info(id,"Averaging spectra", x -> whoAmI( ) );
127   for( int i=0;i<x->N;i++ ) {
128     if( lumModel == "blob" ) { x->set_LvPointAvg( i, x->get_LvPointAvg( i ) + x->get_LvPoint( i )/
      r->getNinj( ) ); }
129     if( lumModel == "steady" ) { x->set_LvPointAvg( i, x->get_LvPointAvg( i ) + x->get_LvPoint( i ) ); }
130   }
131 }
```

4.20.3.8 void observer::calculateAllFlares ( )

calculate all flares

```
40                                    {
41   if( nu1 || nu2 || nu3 || nu4 || nu5 || nu6 ) {
42     bazinga::info(id,"Calculating flares");
43     for( int j=0; j<sizeEnDissProcPoint(); j++ ) {
44       bazinga::info(id,"Calculating point source flare",EnDissProcPoint[j]->
      whoAmI( ));
45       for( int k=0; k<getFreqListSize(); k++ ) {
46     calculateFlare( EnDissProc[j], getFreqList(k) );
47     bazinga::print_info(id,"frequency",getFreqList(k),"Hz"); }
48     }
49   }
50 }
```

4.20.3.9 double observer::calculateFlare (energyDissProc * *_obj*, double *nu*)

calculate flares

Parameters

energyDissProc	
frequency	

```
229                                                                    {
230   gsl_vector* temp_x = gsl_vector_alloc( obj->N );
231   gsl_vector* temp_y = gsl_vector_alloc( obj->N );
232
233   gsl_vector* rad = gsl_vector_alloc( r->getMaxIndex() );
234   gsl_vector* flare = gsl_vector_alloc( r->getMaxIndex() );
235
236   for( int i=0;i<r->getMaxIndex();i++ ) {
237     gsl_vector_set_zero( temp_x );
238     gsl_vector_set_zero( temp_y );
239
240     r->update( i );
241
242     gsl_vector_memcpy( temp_x, obj->vPoint );
243     gsl_vector_memcpy( temp_y, obj->LvPoint );
244     gsl_vector_mul( temp_y, temp_x );
245
246     gsl_vector_set( rad, i, r->get() );
247     gsl_vector_set( flare, i, interpolate( temp_x, temp_y, nu ) ); }
248
249   std::string type = "FlarePoint_";
250   type += obj->whoAmI( );
251   bazinga::info(id,"Saving flare.");
252   bazinga::save_GSLVectorFlare( type, r->getRadius_GSLVector( ), flare, nu, cfg->get<std::string>("
      output") );
253
254   gsl_vector_free( temp_x );
255   gsl_vector_free( temp_y );
256
257   gsl_vector_free( rad );
258   gsl_vector_free( flare ); }
```

4.20.3.10 void observer::calculatePointLuminosity (energyDissProc * *_obj*)

caluclate 'point-source' luminosity

Parameters

energyDissProc	

```
115                                                          {
116   bazinga::info(id,"Calculating point source luminosity", x -> whoAmI( ) );
117   double Doppler = 1.0/(Gamma*(1.0-cos( thetaObs )*beta(Gamma)));
118   for( int i=0;i<x->N;i++ ) {
119     x->set_vPoint( i, x->get_vp( i )*Doppler );
120     if( lumModel == "blob" ) { x->set_LvPoint( i, x->get_Lpv( i )*pow(Doppler, 3) ); }
121     if( lumModel == "steady" ) { x->set_LvPoint( i, x->get_Lpv( i )*pow(Doppler, 2)/Gamma ); }
122   }
123 }
```

4.20.3.11 double observer::getFreqList (int *i*)

get frequency

Parameters

index	

Returns

i-th frequency

```
38 { return freqList[i]; }
```

4.20.3.12 int observer::getFreqListSize ( )

check how many frequencies to look at wen calculating flares

Returns

number of frequencies

```
36 { return freqList.size(); }
```

4.20.3.13 int observer::ifCalculateFlares ( )

check if you should calculate flares

Returns

1 if yes, 0 if no

```
357                                 {
358   if( nu1 || nu2 || nu3 || nu4 || nu5 || nu6 ) { return 1; }
359   else { return 0; }
360 }
```

4.20.3.14 double observer::interpolate (gsl_vector * *x*, gsl_vector * *y*, double *xToInterpolate*)

technical: interpolate data y(x)

Parameters

gsl_vector	x
gsl_vector	y
x_value	to calculate y(x_value)

```
260                                                                 {
261   double interpolatedValue;
262   double xmin, xmax = 0.0;
263 
264   if( _x < gsl_vector_min( x ) || _x > gsl_vector_max( x ) ) { interpolatedValue = 1.0; }
265 
266   for( int i=0; i<x->size-1; i++ ) {
267     if( gsl_vector_get( x, i ) > _x ) {
268       if( gsl_vector_get( y, i ) == 0 || gsl_vector_get( y, i+1 ) == 0 ) { interpolatedValue = 0.0; }
269       else {
270     interpolatedValue = log( gsl_vector_get( y, i ) ) + log(gsl_vector_get( y, i+1 )/gsl_vector_get( y, i )
    )/log(gsl_vector_get( x, i+1 )/gsl_vector_get( x, i ))*log(_x/gsl_vector_get( x, i ) );
271       }
272       interpolatedValue = exp(interpolatedValue);
273       //      interpolatedValue = 0.5*( gsl_vector_get( y, i ) + gsl_vector_get( y, i+1 ) );
274       break; }
275   }
276 
277   return interpolatedValue; }
```

4.20.3.15 void observer::listEnDissProc ( )

show active energy dissipation processes

```
105                                 {
106   std::stringstream s;
107   for( int i=0; i<EnDissProc.size(); i++ ) s << EnDissProc[i]->
    whoAmI() << " ";
108   bazinga::print_info(id,"Processes used to calculate intrinsic lumonisities",s.str()); }
```

4.20.3.16 void observer::listEnDissProcPoint ( )

show active energy dissipation processes for which we have to calculate 'point source' luminosity

```
110                                     {
111   std::stringstream s;
112   for( int i=0; i<EnDissProcPoint.size(); i++ ) s << EnDissProcPoint[i]->whoAmI() << " ";
113   bazinga::print_info(id,"Processes used to calculate point lumonisities",s.str()); }
```

4.20.3.17 void observer::listExtGu ( )

show active ExtGu energy dissipation processes

```
100                             {
101   std::stringstream s;
102   for( int i=0; i<ExtGu.size(); i++ ) s << ExtGu[i]->whoAmI() << " ";
103   bazinga::print_info(id,"External Radiation Gu Sources",s.str()); }
```

4.20.3.18 void observer::listExtPl ( )

show active ExtPl energy dissipation processes

```
90                              {
91   std::stringstream s;
92   for( int i=0; i<ExtPl.size(); i++ ) s << ExtPl[i]->whoAmI() << " ";
93   bazinga::print_info(id,"External Radiation Planar Sources",s.str()); }
```

4.20.3.19 void observer::listExtSp ( )

show active ExtSp energy dissipation processes

```
95                              {
96   std::stringstream s;
97   for( int i=0; i<ExtSp.size(); i++ ) s << ExtSp[i]->whoAmI() << " ";
98   bazinga::print_info(id,"External Radiation Spherical Sources",s.str()); }
```

4.20.3.20 void observer::printInfo ( ) [virtual]

print basic information about myself (virtual)

Reimplemented from baseClass.

```
52                              {
53   bazinga::info(id,"Info");
54   bazinga::print_info(id,"N",N);
55   bazinga::print_info(id,"Observer @",thetaObs,"rad");
56   if( nu1 || nu2 || nu3 || nu4 || nu5 || nu6 ) {
57       bazinga::info(id,"Calculating flares");
58       if( nu1 ) {
59     freqList.push_back( nu1 );
60     bazinga::print_info(id,"@ frequency",nu1); }
61       if( nu2 ) {
62     freqList.push_back( nu2 );
63     bazinga::print_info(id,"@ frequency",nu2); }
64       if( nu3 ) {
65     freqList.push_back( nu3 );
66     bazinga::print_info(id,"@ frequency",nu3); }
67       if( nu4 ) {
68     freqList.push_back( nu4 );
69     bazinga::print_info(id,"@ frequency",nu4); }
70       if( nu5 ) {
71     freqList.push_back( nu5 );
72     bazinga::print_info(id,"@ frequency",nu5); }
73       if( nu6 ) {
74     freqList.push_back( nu6 );
75     bazinga::print_info(id,"@ frequency",nu6); }
76   }
77 }
```

4.20.3.21 void observer::saveAveragedPointLuminosity (energyDissProc * x)

save averaged 'point-source' luminosity

Parameters

energyDissProc	

```
336                                                                 {
337   if( saveLumPointAvg ) {
338     bazinga::info(id,"Saving averaged point luminosities.");
339     std::string type = "LvPointAvg_" + x->whoAmI( );
340     bazinga::save_GSLVector( type, x->vPoint, x->ele->gamma, x->
      LvPointAvg, cfg->get<std::string>("output") ); }
341 }
```

4.20.3.22 void observer::saveAveragedPointLuminositySum ( )

save summed averaged 'point-source' luminosity

```
343                                             {
344   if( saveLumPointAvg  ) {
345     bazinga::info(id,"Saving summed averaged point luminosities.");
346     std::string type = "LvPointAvg_" + this->whoAmI( );
347     bazinga::save_GSLVector( type, vPointSum, LvPointAvgSum, cfg->get<std::string>("output") );
       }
348 }
```

4.20.3.23 void observer::saveAveragedPointLuminositySum (gsl_vector * gamma)

save summed averaged 'point-source' luminosity along with electron gamma values (obsolete)

Parameters

gsl_vector	gamma

```
350                                                                 {
351   if( saveLumPointAvg  ) {
352     bazinga::info(id,"Saving summed averaged point luminosities.");
353     std::string type = "LvPointAvg_" + this->whoAmI( );
354     bazinga::save_GSLVector( type, vPointSum, gamma, LvPointAvgSum, cfg->get<std::string>("
      output") ); }
355 }
```

4.20.3.24 void observer::savePointLuminosity (energyDissProc * x)

save 'point-source' luminosity

Parameters

energyDissProc	

```
315                                                 {
316   if( saveLumPoint && r->ifSaveRadius( ) ) {
317     bazinga::info(id,"Saving point luminosities.");
318     std::string type = "LvPoint_" + x -> whoAmI( );
319     bazinga::save_GSLVector( type, x->vPoint, x->ele->gamma, x->
      LvPoint, r->getPosition( ), cfg->get<std::string>("output") ); }
320 }
```

4.20.3.25 void observer::savePointLuminositySum ( )

save summed 'point-source' luminosity

```
322                                     {
323   if( saveLumPoint && r->ifSaveRadius( ) ) {
324     bazinga::info(id,"Saving summed point luminosities.");
325     std::string type = "LvPoint_" + this -> whoAmI( );
326     bazinga::save_GSLVector( type, vPointSum, LvPointSum, r->getPosition( ),
      cfg->get<std::string>("output") ); }
327 }
```

4.20.3.26 void observer::savePointLuminositySum (gsl_vector * *gamma*)

save summed 'point-source' luminosity along with electron gamma values (obsolete)

Parameters

gsl_vector	gamma

```
329                                                     {
330   if( saveLumPoint && r->ifSaveRadius( ) ) {
331     bazinga::info(id,"Saving summed point luminosities.");
332     std::string type = "LvPoint_" + this -> whoAmI( );
333     bazinga::save_GSLVector( type, vPointSum, gamma, LvPointSum, r->getPosition( ),
      cfg->get<std::string>("output") ); }
334 }
```

4.20.3.27 int observer::sizeEnDissProc ( )

show how many active energy dissipation processes there are

Returns

int

```
84 { return EnDissProc.size(); }
```

4.20.3.28 int observer::sizeEnDissProcPoint ( )

show how many active energy dissipation processes for which we have to calculate 'point source' luminosity there are

Returns

int

```
85 { return EnDissProcPoint.size(); }
```

4.20.3.29 int observer::sizeExtGu ( )

show how many active ExtGu energy dissipation processes there are

Returns

int

```
88 { return ExtGu.size(); }
```

4.20.3.30 int observer::sizeExtPl ( )

show how many active ExtPl energy dissipation processes there are

Returns

int

```
86 { return ExtPl.size(); }
```

4.20.3.31 int observer::sizeExtSp ( )

show how many active ExtSp energy dissipation processes there are

Returns

int

```
87 { return ExtSp.size(); }
```

4.20.3.32 void observer::sumPointAvgLuminosity ( )

caculate a sum of all averaged 'point-source' luminosities from every process involved

```
190 {
191   bazinga::info(id,"Summing calculated averaged point source luminosities ...");
192
193   /* preparing vector for interpolated data */
194   gsl_vector* xx = gsl_vector_alloc( N );
195   gsl_vector* yy = gsl_vector_alloc( N );
196
197   gsl_vector_set_zero( xx );
198   gsl_vector_set_zero( yy );
199
200   for( int i=0;i<N;i++ ) { gsl_vector_set( xx, i, vmin*pow(vmax/vmin,(double)i/(double)N) ); }
201
202   for( int i=0; i<sizeEnDissProcPoint(); i++ ) {
203     bazinga::print_info(id,"Add",EnDissProcPoint[i]->whoAmI());
204
205     /* we will copy actual values stores in matrices to gsl_vectors; y vector is nu L nu!! */
206     gsl_vector* temp_x = gsl_vector_alloc( EnDissProcPoint[i]->N );
207     gsl_vector* temp_y = gsl_vector_alloc( EnDissProcPoint[i]->N );
208
209     gsl_vector_memcpy( temp_x, EnDissProcPoint[i]->vPoint );
210     gsl_vector_memcpy( temp_y, EnDissProcPoint[i]->LvPointAvg );
211
212     /* nu F nu! */
213     gsl_vector_mul( temp_y, temp_x );
214
215     /* fill in the final vector and add only those values which are within boundaries */
216     for( int k=0; k<N; k++ ) { gsl_vector_set( yy, k, gsl_vector_get( yy, k ) +
    interpolate( temp_x, temp_y, gsl_vector_get( xx, k ) ) ); }
217
218     gsl_vector_free( temp_x );
219     gsl_vector_free( temp_y ); }
220
221   /* copy vector to matrix */
222   for( int i=0;i<N;i++ ) {
223     gsl_vector_set( vPointSum, i, gsl_vector_get( xx, i ) );
224     gsl_vector_set( LvPointAvgSum, i, gsl_vector_get( yy, i ) ); }
225
226   gsl_vector_free( xx );
227   gsl_vector_free( yy ); }
```

4.20.3.33 void observer::sumPointLuminosity ( )

caculate a sum of all 'point-source' luminosities from every process involved

```
133                                 {
134   /* preparing vector for interpolated data */
135   gsl_vector* xx = gsl_vector_alloc( N );
136   gsl_vector* yy = gsl_vector_alloc( N );
137
138   gsl_vector_set_zero( xx );
139   gsl_vector_set_zero( yy );
140
141   for( int i=0;i<N;i++ ) { gsl_vector_set( xx, i, vmin*pow(vmax/vmin,(double)i/(double)N) ); }
142
143   for( int i=0; i<sizeEnDissProcPoint(); i++ ) {
144     bazinga::print_info(id,"Add",EnDissProcPoint[i]->whoAmI());
145
146     /* we will copy actual values stores in matrices to gsl_vectors; y vector is nu L nu!! */
147     gsl_vector* temp_x = gsl_vector_alloc( EnDissProcPoint[i]->N );
148     gsl_vector* temp_y = gsl_vector_alloc( EnDissProcPoint[i]->N );
149
150     gsl_vector_memcpy( temp_x, EnDissProcPoint[i]->vPoint );
151     gsl_vector_memcpy( temp_y, EnDissProcPoint[i]->LvPoint );
152
153     /* nu F nu! */
154     gsl_vector_mul( temp_y, temp_x );
155
156     /* fill in the final vector and add only those values which are within boundaries */
157     for( int k=0; k<N; k++ ) { gsl_vector_set( yy, k, gsl_vector_get( yy, k ) +
    interpolate( temp_x, temp_y, gsl_vector_get( xx, k ) ) ); }
158
159     gsl_vector_free( temp_x );
160     gsl_vector_free( temp_y ); }
161
162   /* copy vector to matrix */
163   for( int i=0;i<N;i++ ) {
164     gsl_vector_set( vPointSum, i, gsl_vector_get( xx, i ) );
165     gsl_vector_set( LvPointSum, i, gsl_vector_get( yy, i ) );
166  }
167
168   gsl_vector_free( xx );
169   gsl_vector_free( yy ); }
```

4.20.4 Member Data Documentation

4.20.4.1 std::vector<energyDissProc*> observer::EnDissProc

vectors to hold processes being used in calculations

4.20.4.2 std::vector<double> observer::freqList

frequency list

4.20.4.3 gsl_vector* observer::vPointSum

vectors to store summed luminosities

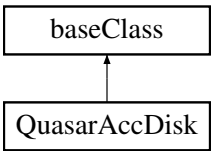
The documentation for this class was generated from the following files:

- /home/mjaniak/Soft/blazar++/include/observer.hpp
- /home/mjaniak/Soft/blazar++/src/observer.cpp

4.21 QuasarAccDisk Class Reference

`#include <QuasarAccDisk.hpp>`

Inheritance diagram for QuasarAccDisk:



4.21.1 Detailed Description

Class used to process quasar radiation template and add it to calculated jet radiation; at the moment quasar radiation is not stricty calculated but a radius-loud quasar radiation template from Elvis et al. (1994) is used. Class has some basic methods to calculate standard multitemperature Shakura-Sunyayev disk radiation but it is not used currently as it need much, much detailed study and modelling of dusty torus, coronae etc

Public Member Functions

- QuasarAccDisk (scfgp ∗_cfg, std::string _id)
- ∼QuasarAccDisk ()
- void printInfo ()
- double getTemperature (double _r)
- double calculateLv (double v, int radius_index)
- void calculateLuminosity ()
- void setRadius ()
- void setEnergetics ()
- int readTemplate ()
- void scaleTemplate ()
- int saveTemplate ()
- void allocateLv ()
- void **freeLv** ()
- void **allocateLvTemplate** ()
- void **freeLvTemplate** ()
- void **set_v** (int i, double val)
- void **set_Lv** (int i, double val)
- int **getN** ()
- gsl_vector ∗ **get_v** ()
- gsl_vector ∗ **get_Lv** ()
- int **get_save_Lr** ()

Public Attributes

- gsl_vector ∗ vTemplate
- gsl_vector ∗ **LvTemplate**
- logGeometry ∗ R

4.21.2 Constructor & Destructor Documentation

4.21.2.1 QuasarAccDisk::QuasarAccDisk (scfgp ∗ *_cfg,* std::string *_id*)

constructor

Parameters

scfgp	
id	

```
8                                                                   : baseClass( _cfg, NULL, _id ) {
9   /* requested parameters */
10   cfg -> request<double>("mBH",1.0,&mBH);
11   cfg -> request<double>("mDot",1.0,&mDot);
12   cfg -> request<double>("eDisk",0.1,&eDisk);
13   cfg -> request<int>("N" + id, 200, &N );
14   cfg -> request<double>(id + "R1", 0.0, &R1 );
15   cfg -> request<double>(id + "R2", 0.0, &R2 );
16   cfg -> request<std::string>(id + "TemplateFilename", "dat/elvisRL.dat", &templateFilename );
17
18   cfg -> request<int>(id + "SaveLr", 0, &save_Lr);
19
20   cfg -> updateRequests( );
21
22   setEnergetics( );
23
24   if( R1 == 0.0 || R2 == 0.0 ) { setRadius( ); }
25
26   vMin = 1.0e12;
27   vMax = 1.0e18;
28
29   numLines = 0;
30
31   allocateLv( );
32
33   for( int i=0;i<N;i++ ) { set_v( i, vMin*pow( vMax/vMin,(double)i/((double)N-1)) ); }
34   R = new logGeometry( R1, R2, 1000 );
35 }
```

4.21.2.2 QuasarAccDisk::~QuasarAccDisk ( )

destructor

```
37                                {
38   freeLv( );
39   freeLvTemplate( );
40 }
```

4.21.3 Member Function Documentation

4.21.3.1 void QuasarAccDisk::allocateLv ( )

methods to allocate and free memory

```
173                                 {
174   v = gsl_vector_alloc( N );
175   Lv = gsl_vector_alloc( N );
176   bazinga::print_GSLVector_allocated_memory( id, v );
177   bazinga::print_GSLVector_allocated_memory( id, Lv );
178   gsl_vector_set_zero( v );
179   gsl_vector_set_zero( Lv ); }
```

4.21.3.2 void QuasarAccDisk::calculateLuminosity ( )

calculate a whole disk luminosity

```
197                                       {
198   bazinga::info(id,"Calculating luminosity.");
199   gsl_vector* tempLv = gsl_vector_alloc( N );
200
201   for( int i=0;i<R->getMaxIndex();i++ ) {
202     R -> update( i );
203     gsl_vector_set_zero( tempLv );
204
```

```
205     for( int j=0;j<N;j++ ) {
206       if( save_Lr ) {
207     gsl_vector_set( tempLv, j, calculateLv( gsl_vector_get(v,j), i ) ); }
208       gsl_vector_set( Lv, j, gsl_vector_get( Lv, j ) + calculateLv( gsl_vector_get(v,j), i ) );
209     }
210
211     if( save_Lr ) {
212       std::string type = "Lv_";
213       type += this->whoAmI( );
214       bazinga::print_info(id,"Saving QLuminosity.",R->get( )/R->getR0( ));
215       bazinga::save_GSLVector( type, v, tempLv, R->get( )/R->getR0( ), cfg->get<std::string>("output")
      ); }
216   }
217
218   gsl_vector_free( tempLv );
219   tempLv = NULL;
220
221   std::string type = "Lv_";
222   type += this->whoAmI( );
223   bazinga::info(id,"Saving QLuminosity.");
224   bazinga::save_GSLVector( type, v, Lv, cfg->get<std::string>("output") );
225
226   double integral = 0.0;
227   tempLv = gsl_vector_alloc( N );
228   gsl_vector_set_zero( tempLv );
229   gsl_vector_memcpy( tempLv, Lv );
230   gsl_vector_mul( tempLv, v );
231   integral = loglogIntegrate( v, tempLv );
232   bazinga::print_info( id, "Acc Disk Lbol", integral );
233 }
```

4.21.3.3 double QuasarAccDisk::calculateLv (double *v*, int *radius_index*)

calculate disk luminosity

Parameters

frequency	v
radius_index	

```
235                                                 {
236   double val = ( 16.0*pow(M_PI,2)*PLANCK_H*cos(0.0)*pow(v,3) )/( pow(LIGHT_SPEED,2) );
237   return val*( R->get(i)*R->getDr(i) )/( exp( (PLANCK_H*v)/(K_BOLTZMAN*
      getTemperature(R->get(i)) ) ) - 1.0 );
238 }
```

4.21.3.4 double QuasarAccDisk::getTemperature (double *_r*)

get BB temperature at distance r (SS disk)

Parameters

radius	r

```
168                                             {
169   double Risco = 6.0*Rg;
170   return pow( ( 3.0*G_CONST*1.3e56*pow(mBH,2)*MSUN*mDot )*( 1.0-sqrt(Risco/_r) )/( 8.0*M_PI*pow(_r, 3)*
      SIGMA_SB*pow(LIGHT_SPEED,2) ), 0.25 );
171 }
```

4.21.3.5 void QuasarAccDisk::printInfo ( ) [virtual]

print basic information about myself (virtual)

Reimplemented from baseClass.

```
154                                {
155   bazinga::info( id, "Info" );
156   bazinga::print_info( id, "N", N );
157   bazinga::print_info( id, "Gravitational radius", Rg, "cm" );
158   bazinga::print_info( id, "R1", R1, "cm" );
159   bazinga::print_info( id, "R2", R2, "cm" );
160   bazinga::print_info( id, "Save luminosity", save_Lr );
161   bazinga::print_info(id,"BH mass",mBH);
162   bazinga::print_info(id,"eta disk",eDisk);
163   bazinga::print_info(id,"m dot",mDot);
164   bazinga::print_info(id,"Eddington luminosity",Ledd,"erg/s");
165   bazinga::print_info( id, "Accretion disk luminosity", Ldisk, "erg/s");
166 }
```

4.21.3.6 int QuasarAccDisk::readTemplate ( )

read quasar radiation template

Returns

1 if success; 0 otherwise

```
115                                  {
116   std::ifstream input( templateFilename.c_str() );
117   bazinga::print_info( id, "Reading quasar template data from file ",templateFilename.c_str());
118      if( input == NULL )
119     {
120       bazinga::error(id,"Opening file:",templateFilename);
121       return 1;
122     }
123       else
124     {
125       // check how many lines there are
126       std::string unused;
127       while ( std::getline(input, unused) )
128         ++numLines;
129
130       bazinga::print_info(id,"Found",numLines,"lines");
131
132       // allocate space for vectors
133       allocateLvTemplate( );
134
135       // read data from template
136       input.clear();
137       input.seekg(0, std::ios::beg);
138       double x, y;
139       int i = 0; // gsl counter
140       while( std::getline(input, unused) ) {
141       std::istringstream ss(unused);
142       std::istream_iterator<std::string> begin(ss), end;
143
144       //putting all the tokens in the vector
145       std::vector<std::string> arrayTokens(begin, end);
146
147       gsl_vector_set( vTemplate, i, atof( arrayTokens[0].c_str() ) );
148       gsl_vector_set( LvTemplate, i, atof( arrayTokens[1].c_str() ) );
149       ++i; }
150       return 0;
151     }
152 }
```

4.21.3.7 int QuasarAccDisk::saveTemplate ( )

save scaled quasar radiation template

```
101                                  {
102   std::string type = "LvTemplate_"+this->whoAmI( );
103   bazinga::info(this->whoAmI( ),"Saving luminosity.");
104
105   gsl_vector* LvTemplateTemp = gsl_vector_alloc( numLines );
106   gsl_vector_set_zero( LvTemplateTemp );
107   gsl_vector_memcpy( LvTemplateTemp, LvTemplate );
108   gsl_vector_div( LvTemplateTemp, vTemplate );
109
110   bazinga::save_GSLVector( type, this->vTemplate, LvTemplateTemp, cfg->get<std::string>("output
```

```
      ") );
111   gsl_vector_free( LvTemplateTemp );
112   LvTemplateTemp = NULL;
113  }
```

4.21.3.8 void QuasarAccDisk::scaleTemplate ( )

scale quasar radiation template to match set accretion disk efficiency

```
49                                      {
50   double integral = loglogIntegrate( vTemplate, LvTemplate );
51   double correctionFactor = Ldisk/integral;
52   bazinga::print_info( id, "Template correction factor", correctionFactor );
53   gsl_vector_scale( LvTemplate, correctionFactor );
54   integral = loglogIntegrate( vTemplate, LvTemplate );
55   bazinga::print_info( id, "Template Lbol after correction", integral );
56
57   // Find peak in IR, Optical and X-ray
58   double IRmin = 1.0e10;
59   double IRmax = 2.0e14;
60   double OPTmin = IRmax;
61   double OPTmax = 7.0e16;
62   double Xmin = OPTmax;
63   double Xmax = 1.0e20;
64
65   double vIRpeak = 0.0;
66   double vOPTpeak = 0.0;
67   double vXpeak = 0.0;
68
69   double vLvIRpeak = 0.0;
70   double vLvOPTpeak = 0.0;
71   double vLvXpeak = 0.0;
72
73   for( int i=0; i<vTemplate->size-1; ++i ) {
74     if( gsl_vector_get( vTemplate, i ) >= IRmin && gsl_vector_get(
      vTemplate, i ) <= IRmax )
75       if( gsl_vector_get( LvTemplate, i ) > vLvIRpeak ) {
76     vLvIRpeak = gsl_vector_get( LvTemplate, i );
77     vIRpeak = gsl_vector_get( vTemplate, i ); }
78   }
79   bazinga::print_info( id, "IR template v peak", vIRpeak );
80   bazinga::print_info( id, "IR template vLv peak", vLvIRpeak );
81
82   for( int i=0; i<vTemplate->size-1; ++i ) {
83     if( gsl_vector_get( vTemplate, i ) >= OPTmin && gsl_vector_get(
      vTemplate, i ) <= OPTmax )
84       if( gsl_vector_get( LvTemplate, i ) > vLvOPTpeak ) {
85     vLvOPTpeak = gsl_vector_get( LvTemplate, i );
86     vOPTpeak = gsl_vector_get( vTemplate, i ); }
87   }
88   bazinga::print_info( id, "OPT template v peak", vOPTpeak );
89   bazinga::print_info( id, "OPT template vLv peak", vLvOPTpeak );
90
91   for( int i=0; i<vTemplate->size-1; ++i ) {
92     if( gsl_vector_get( vTemplate, i ) >= Xmin && gsl_vector_get(
      vTemplate, i ) <= Xmax )
93       if( gsl_vector_get( LvTemplate, i ) > vLvXpeak ) {
94     vLvXpeak = gsl_vector_get( LvTemplate, i );
95     vXpeak = gsl_vector_get( vTemplate, i ); }
96   }
97   bazinga::print_info( id, "X template v peak", vXpeak );
98   bazinga::print_info( id, "X template vLv peak", vLvXpeak );
99 }
```

4.21.3.9 void QuasarAccDisk::setEnergetics ( )

set energetics for further calculations

```
247                                     {
248    Ledd = 1.3e47*mBH;
249    Ldisk = eDisk*mDot*Ledd; }
```

4.21.3.10 void QuasarAccDisk::setRadius ( )

set accretion disk radial boundaries

```
240                                {
241   /* gravitational radius */
242   Rg = (2.0*G_CONST*mBH*1.0e9*MSUN)/pow(LIGHT_SPEED,2);
243   R1 = 6.0*Rg;
244   double R_sub = 1.6e-5*sqrt( Ldisk );
245   R2 = R_sub; }
```

4.21.4 Member Data Documentation

4.21.4.1 logGeometry* QuasarAccDisk::R

logGeometry to store log-spaced data for loglog-interpolation

4.21.4.2 gsl_vector* QuasarAccDisk::vTemplate

vectors to store data

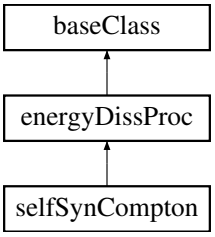
The documentation for this class was generated from the following files:

- /home/mjaniak/Soft/blazar++/include/QuasarAccDisk.hpp
- /home/mjaniak/Soft/blazar++/src/QuasarAccDisk.cpp

4.22 selfSynCompton Class Reference

`#include <selfSynCompton.hpp>`

Inheritance diagram for selfSynCompton:



4.22.1 Detailed Description

Class provides SSC electron energy losses and SSC luminosity calculation

Public Member Functions

- selfSynCompton (scfgp *_cfg, jetGeometry *_r, electrons *_ele, magneticField *_B, std::string _id)
- ~selfSynCompton ()
- void printInfo ()
- double dotg (double g)
- void update ()
- double calculateLpv (double v)
- void setLpv ()

Public Attributes

- magneticField * B
- energyDissProc * syn

4.22.2 Constructor & Destructor Documentation

4.22.2.1 selfSynCompton::selfSynCompton (scfgp * *_cfg,* jetGeometry * *_r,* electrons * *_ele,* magneticField * *_B,* std::string *_id*)

constructor

Parameters

_cfg	- scfgp class object
_r	- jetGeometry class object
_ele	- electrons class object
_B	- magneticField class object
_id	

```
9
          : energyDissProc( _cfg, _r, _ele, _id  ), B(_B) {
10   /* requested parameters */
11   cfg -> request<int>("N"+id,200,&N);
12   cfg -> request<int>("KN",0,&KN);
13
14   cfg -> updateRequests( );
15
16   vpMin = 1.0;
17   /* here we set vpMAX to a maximum value specified by maximal value of magnetic field at R0 */
18   vpMax = 100.0*A43*DSQR(ele->getGammaMax( ))*DSQR(ele->getGammaMax( ))*(
      B->getMaxB( )/B_CR )*mec2h;
19
20   allocateUpe( );
21   allocateUpeR( );
22   allocateLpv( );
23   allocateLvPoint( );
24   allocateLvPointAvg( );
25
26   for( int i=0;i<N;i++ ) { set_vp( i, vpMin*pow( vpMax/vpMin,(double)i/((double)N-1)) ); }
27  }
```

4.22.2.2 selfSynCompton::~selfSynCompton ( )

destructor

```
29                                {
30   freeLpv( );
31   freeLvPoint( );
32   freeLvPointAvg( );
33   freeUpe( ); }
```

4.22.3 Member Function Documentation

4.22.3.1 double selfSynCompton::calculateLpv (double *v*) [virtual]

calculate intrinsic luminosity

Parameters

v	- frequency (jet co-moving frame)

Returns

L'_v

Reimplemented from energyDissProc.

```
59                                                {
60   double dg,de,j;
61   double Int1;
62   double e;
63   Int1 = 0.0;
64   j = 0.0;
65
66   e = PLANCK_H*v/(ELECTRON_MASS*LIGHT_SPEED*LIGHT_SPEED);
67
68   dg = ele->getdLogGamma( );
69   de = syn->getdLogE( );
70
71   for( int i=0; i<syn->getN( );i++ ) {
72     Int1 = 0.0;
73     /* integration over electron energy (gamma) */
74     for( int k=0;k<ele->getN( );k++ ) {
75       Int1  += bazinga::IntCor( k, ele->getN( ) )*inverseCompton::fiso(
      ele->getGamma(k), syn->get_ep(i),e )*ele->getNgamma(k)/
      ele->getGamma(k); }
76
77       /* integration over seed photon energy */
78     j += bazinga::IntCor( i, syn->getN( ) )*syn->get_upe(i)/syn->get_ep(i)*Int1*dg; }
79
80   j *= constC1*e*de;
81   return j; }
```

4.22.3.2 double selfSynCompton::dotg (double *g*) [virtual]

get d gamma \ d t for particular process

Parameters

g	- electron Lorentz factor

Returns

d gamma \ d t

Reimplemented from energyDissProc.

```
42                                        {
43   double b,sum, val = 0.0;
44
45   for( int i=0;i<syn->getN( );i++ ) {
46     b = 4.0*syn->get_ep(i)*g;
47     if( b > 1.0 ) { set_KN_info( g ); }
48     sum += bazinga::IntCor( i, syn->getN( ) )*syn->get_ep(i)*syn->get_upe(i)*
      inverseCompton::fKN( b, KN ); }
49
50   val = syn->getdLogE( )*sum;
51
52   set_upe_r( val );
53   return( val ); }
```

4.22.3.3 void selfSynCompton::printInfo ( ) [virtual]

print basic information about myself (virtual)

Reimplemented from energyDissProc.

```
35                                {
36   bazinga::info(id,"Info");
37   bazinga::print_info(id,"N",N);
38   bazinga::print_info(id,"KN",KN); }
```

4.22.3.4 void selfSynCompton::setLpv ( ) [virtual]

set intrinsic luminosities

Reimplemented from energyDissProc.

```
55                                {
56   for( int i=0;i<N;i++ ) { set_Lpv( i, calculateLpv( get_vp(i) ) ); }
57 }
```

4.22.3.5 void selfSynCompton::update ( ) [virtual]

update all process internal parameters to current radius

Reimplemented from energyDissProc.

```
40 { flag_upe_r = false; }
```

4.22.4 Member Data Documentation

4.22.4.1 magneticField* selfSynCompton::B

magenetic field B'

4.22.4.2 energyDissProc* selfSynCompton::syn

synchrotron process pointer

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

- /home/mjaniak/Soft/blazar++/include/selfSynCompton.hpp
- /home/mjaniak/Soft/blazar++/src/selfSynCompton.cpp

4.23 struct Class Reference

4.23.1 Detailed Description

used to calculate break in electron energy

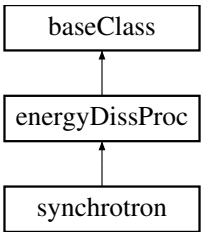
The documentation for this class was generated from the following file:

- /home/mjaniak/Soft/blazar++/include/electrons.hpp

4.24 synchrotron Class Reference

`#include <synchrotron.hpp>`

Inheritance diagram for synchrotron:



4.24.1 Detailed Description

Class provides synchrotron electron energy losses and synchrotron luminosity calculation

Public Member Functions

- synchrotron (scfgp ∗_cfg, jetGeometry ∗_r, electrons ∗_ele, magneticField ∗_B, std::string _id)
- ∼synchrotron ()
- void printInfo ()
- double iterate ()
- double dotg (double g)
- void update ()
- void setLpv ()
- double calculateLpv (double v)

Additional Inherited Members

4.24.2 Constructor & Destructor Documentation

4.24.2.1 synchrotron::synchrotron (scfgp ∗ *_cfg,* jetGeometry ∗ *_r,* electrons ∗ *_ele,* magneticField ∗ *_B,* std::string *_id*)

constructor

Parameters

_cfg	- scfgp class object
_r	- jetGeometry class object
_ele	- electrons class object
_B	- magnetic field class object
_id	

```
9                                                                                                                    :
      energyDissProc( _cfg, _r, _ele, _id ), B(_B) {
10    /* requested parameters */
11    cfg -> request<int>("N"+id,200,&N);
12    cfg -> request<std::string>("lumModel","blob",&lumModel);
13    cfg -> request<double>("thetaJ",0.1,&thetaJ);
14    cfg -> request<double>("Gamma",10.0,&Gamma);
15    cfg -> request<int>("SABS",1,&SABS);
16    cfg -> request<int>(id+"LuminosityConstU",0,&luminosityConstU);
17    cfg -> request<int>(id+"LuminosityConstNu",0,&luminosityConstNu);
18    cfg -> request<int>("setSSCBlob",0,&setSSCBlob);
19
20    cfg -> updateRequests( );
21
22    vpMin = 1.0;
23    /* here we set vpMAX to a maximum value specified by maximal value of magnetic field at R0 */
24    vpMax = 100.0*A43*DSQR(ele->getGammaMax( ))*( B->getMaxB( )/B_CR )*mec2h;
25    epMin = PLANCK_H/mec2;
26    epMax = vpMax/mec2h;
27
28    allocateUpe( );
```

```
29   allocateUpeR( );
30   allocateLpv( );
31   allocateLvPoint( );
32   allocateLvPointAvg( );
33
34   for( int i=0;i<N;i++ ) {
35     set_ep( i, epMax*pow( epMax/epMin,(double)i/(double)(N-1)) );
36     set_vp( i, vpMin*pow( vpMax/vpMin,(double)i/((double)N-1)) ); }
37
38   dLogE = log(get_ep(1)/get_ep(0)); }
```

4.24.2.2 synchrotron::∼synchrotron ( )

destructor

```
40                              {
41   freeLpv( );
42   freeLvPoint( );
43   freeLvPointAvg( );
44   freeUpe( ); }
```

4.24.3 Member Function Documentation

4.24.3.1 double synchrotron::calculateLpv (double *v*) [virtual]

calculate intrinsic luminosity

Parameters

v	- frequency (jet co-moving frame)

Returns

L'_v

Reimplemented from energyDissProc.

```
91                                          {
92   double a,jS,sigmaS,corr;
93   double vB,h,t,L,tau;
94
95   vB = ELECTRON_CHARGE*B->getB( )/(2.0*M_PI*ELECTRON_MASS*LIGHT_SPEED);
96   h = ele->getdLogGamma( );
97   L = tau = 0.0e0;
98
99   for( int i=0;i<ele->getN( );i++) {
100       t = v/(3.0*DSQR( ele->getGamma(i))*vB );
101       FS(t,&jS,&sigmaS);
102       tau += bazinga::IntCor( i, ele->getN( ) )*sigmaS*ele->getNgamma(i)/pow(
     ele->getGamma(i),4);
103       L += bazinga::IntCor( i, ele->getN( ) )*ele->getGamma(i)*
     ele->getNgamma(i)*jS; }
104
105
106    if( SABS && (r->get( )>=r->getR0( )) ) {
107       /* since in a there is r-r0 then first luminosity wont have absorption; original blazar code had
      different naming:
108       all quantities calculated at r were tagged as r+rd - we do not follow this approach thus r>*
     (model->R0) and not
109       r>=R0 (it gives errors) */
110       /* I cheat; I set DR to be dr for r=R0; is it a bad approximation? I dunno */
111       double DR = 0;
112       if( r->get() == r->getR0( ) ) { DR = r->getDr(); }
113       else { DR = r->get( )-(r->getR0( )); }
114
115       a = 2.0*DR/(beta(Gamma)*Gamma);
116       tau *= h*3.0/(4.0*M_PI*a*a)*2.0*sqrt(3.0)*M_PI/15.0*ELECTRON_CHARGE/B->
     getB( );
117       corr = (tau>1.0e-5) ? (1.0-exp(-tau))/tau : (1.0-0.5*tau+1.0/6.0*tau*tau-1.0/24.0*pow(tau,3)+1.0/120.
     0*pow(tau,4)-1.0/720.0*pow(tau,5));
118    } else { corr = 1.0e0; }
119
```

```
120   if( luminosityConstU || luminosityConstNu ) {   L *= h*3.0*sqrt(3.0)*
      SIGMA_T*LIGHT_SPEED*B->get_uB( injRm )/(M_PI*vB)*corr; }
121   else {   L *= h*3.0*sqrt(3.0)*SIGMA_T*LIGHT_SPEED*B->get_uB( )/(M_PI*vB)*corr; }
122
123   return L; }
```

4.24.3.2 double synchrotron::dotg (double *g*) [virtual]

get d gamma \ d t for particular process

Parameters

g	- electron Lorentz factor

Returns

d gamma \ d t

Reimplemented from energyDissProc.

```
58 { return B->get_uB( ); }
```

4.24.3.3 double synchrotron::iterate ( )

iterate synchrotron and SSC calculation to achieve steady state and balance between synchrotron and SSC luminosities

```
125                              {
126   double sumN = 0.0;
127   double err = 0.0;
128   gsl_vector* temp_upe = gsl_vector_alloc( N );
129
130   /* copy current upe to temp_upe */
131   for( int i=0;i<N;i++ ) { gsl_vector_set( temp_upe, i, get_upe( i ) ); }
132
133   /* calculate synchrotron energy densities and luminosities with new Ngamma*/
134   update( );
135
136   for( int i=0;i<N;i++ ) {
137       sumN += get_upe( i );
138       err += fabs( get_upe( i ) - gsl_vector_get( temp_upe, i ) ); }
139   err /= sumN;
140   gsl_vector_free( temp_upe );
141   temp_upe = NULL;
142
143   return err; }
```

4.24.3.4 void synchrotron::printInfo ( ) [virtual]

print basic information about myself (virtual)

Reimplemented from energyDissProc.

```
46                             {
47   bazinga::info(id,"Info");
48   bazinga::print_info(id,"N",N);
49   bazinga::print_info(id,"SABS",SABS);
50   if( setSSCBlob ) {
51     bazinga::print_info(id,"setSSCBlob",setSSCBlob); }
52
53   if( luminosityConstU || luminosityConstNu ) {
54     bazinga::warning(id,"Using constant u' to calculate luminosity.");
55     bazinga::warning(id,"Using constant v' to calculate luminosity."); }
56 }
```

4.24.3.5 void synchrotron::setLpv ( ) [virtual]

set intrinsic luminosities

Reimplemented from energyDissProc.

```
74                          {
75   for( int i=0;i<N;i++ ) {
76     set_vp( i, vpMin*pow( vpMax/vpMin,(double)i/((double)N-1)) );
77     set_Lpv( i, calculateLpv( get_vp(i) ) ); }
78 }
```

4.24.3.6 void synchrotron::update ( ) [virtual]

update all process internal parameters to current radius

Reimplemented from energyDissProc.

```
60                          {
61   setLpv( );
62   double lum_to_upe = 0.0;
63   if( lumModel == "blob" || setSSCBlob ) { lum_to_upe = 2.0*M_PI*thetaJ*thetaJ*r->get( )*
      r->get( )*LIGHT_SPEED; }
64   if( lumModel == "steady" && !setSSCBlob ) { lum_to_upe = 2.0*M_PI*r->get( )*r->getDr( )*thetaJ*Gamma*
      LIGHT_SPEED; }
65
66   for( int i=0;i<N;i++ ) {
67       set_ep( i, epMin*pow(epMax/epMin,(double)i/((double)N-1)) );
68       set_upe( i, mec2h*get_Lpv(i)/lum_to_upe ); }
69
70   set_upe_r( B->get_uB( ) );
71   //  std::cout << "BFIELD " << r->get() << " " << B->getB();
72   flag_upe_r = false; }
```

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

- /home/mjaniak/Soft/blazar++/include/synchrotron.hpp
- /home/mjaniak/Soft/blazar++/src/synchrotron.cpp

Chapter 5

File Documentation

5.1 /home/mjaniak/Soft/blazar++/include/AccdPlanar.hpp File Reference

```
#include "externalRadiationPlanar.hpp"
```

5.1.1 Detailed Description

Author

Mateusz Janiak

Classes

- class AccdPlanar

5.2 /home/mjaniak/Soft/blazar++/include/AccdSphericalGu.hpp File Reference

```
#include "externalRadiationGu.hpp"
#include "inverseCompton.hpp"
```

5.2.1 Detailed Description

Author

Mateusz Janiak

Classes

- class AccdSphericalGu

5.3 /home/mjaniak/Soft/blazar++/include/baseClass.hpp File Reference

```
#include <string>
```

```
#include <sstream>
#include <iostream>
#include <iomanip>
#include <cmath>
#include <scfgp.hpp>
#include <bazinga.hpp>
#include "jetGeometry.hpp"
```

5.3.1 Detailed Description

Author

Mateusz Janiak

Classes

- class baseClass

5.4 /home/mjaniak/Soft/blazar++/include/blazar.hpp File Reference

```
#include <scfgp.hpp>
#include <bazinga.hpp>
#include "electrons.hpp"
```

5.4.1 Detailed Description

Main blazar++'s header file

Author

Mateusz Janiak

Functions

- void getConfigSwitches (scfgp ∗_cfg)
- void safetyConfigSwitches (scfgp ∗_cfg, electrons ∗_ele)

5.4.2 Function Documentation

5.4.2.1 void getConfigSwitches (scfgp ∗ *_cfg*)

Read main config switches from configuration file

Parameters

scfgp	

```
20                                      {
21   _cfg -> add<int>("lumSyn",0);
22   _cfg -> add<int>("lumSsc",0);
23   _cfg -> add<int>("lumExt1",0);
24   _cfg -> add<int>("lumExt2",0);
25   _cfg -> add<int>("lumBlrPl",0);
```

```
26   _cfg -> add<int>("lumDtPl",0);
27   _cfg -> add<int>("lumAccd",0);
28   _cfg -> add<int>("lumBlrSp",0);
29   _cfg -> add<int>("lumDtSp",0);
30   _cfg -> add<int>("lumBlrGu",0);
31   _cfg -> add<int>("lumDtGu",0);
32   _cfg -> add<int>("lumAccdGu",0);
33   _cfg -> add<int>("lumSynPoint",0);
34   _cfg -> add<int>("lumSscPoint",0);
35   _cfg -> add<int>("lumExt1Point",0);
36   _cfg -> add<int>("lumExt2Point",0);
37   _cfg -> add<int>("lumBlrPlPoint",0);
38   _cfg -> add<int>("lumDtPlPoint",0);
39   _cfg -> add<int>("lumAccdPoint",0);
40   _cfg -> add<int>("lumBlrSpPoint",0);
41   _cfg -> add<int>("lumDtSpPoint",0);
42   _cfg -> add<int>("lumBlrGuPoint",0);
43   _cfg -> add<int>("lumDtGuPoint",0);
44   _cfg -> add<int>("lumAccdGuPoint",0);
45   _cfg -> add<int>("Adiabatic",0);
46   _cfg -> add<int>("Syn",0);
47   _cfg -> add<int>("Ssc",0);
48   _cfg -> add<int>("Ext1",0);
49   _cfg -> add<int>("Ext2",0);
50   _cfg -> add<int>("BlrPl",0);
51   _cfg -> add<int>("DtPl",0);
52   _cfg -> add<int>("BlrSp",0);
53   _cfg -> add<int>("DtSp",0);
54   _cfg -> add<int>("Accd",0);
55   _cfg -> add<int>("BlrGu",0);
56   _cfg -> add<int>("DtGu",0);
57   _cfg -> add<int>("AccdGu",0);
58   _cfg -> add<int>("QAccd",0);
59   _cfg -> add<int>("SABS",0);
60   _cfg -> add<double>("AdiabaticABG",0.666667);
61   _cfg -> add<std::string>("output","blazar_out");
62   _cfg -> add<int>("saveElectrons",1);
63   _cfg -> add<int>("saveElectronsAvg",1);
64   _cfg -> add<int>("saveLum",1);
65   _cfg -> add<int>("saveLumPoint",1);
66   _cfg -> add<int>("root",1);
67   _cfg -> add<int>("printRoot",1);
68
69   _cfg -> updateAddedParameters( ); }
```

5.4.2.2 void safetyConfigSwitches (scfgp * *_cfg*, electrons * *_ele*)

After reading parameters apply some necessary changes to them

Parameters

scfgp	
electrons	

```
77                                                          {
78   /* some safety features */
79   if( _ele -> ifEvol( ) &&  _cfg->get<int>("Syn") ) { _cfg->modify<int>("lumSyn",1); }
80   if( _ele -> ifEvol( ) &&  _cfg->get<int>("Ssc") ) { _cfg->modify<int>("lumSsc",1); }
81
82   if( _cfg->get<int>("lumSynPoint") ) { _cfg->modify<int>("lumSyn",1); }
83   if( _cfg->get<int>("lumSscPoint") ) { _cfg->modify<int>("lumSsc",1); }
84   if( _cfg->get<int>("lumExt1Point") ) { _cfg->modify<int>("lumExt1",1); }
85   if( _cfg->get<int>("lumExt2Point") ) { _cfg->modify<int>("lumExt2",1); }
86   if( _cfg->get<int>("lumBlrPlPoint") ) { _cfg->modify<int>("lumBlrPl",1); }
87   if( _cfg->get<int>("lumDtPlPoint") ) { _cfg->modify<int>("lumDtPl",1); }
88   if( _cfg->get<int>("lumAccdPoint") ) { _cfg->modify<int>("lumAccd",1); }
89   if( _cfg->get<int>("lumBlrSpPoint") ) { _cfg->modify<int>("lumBlrSp",1); }
90   if( _cfg->get<int>("lumDtSpPoint") ) { _cfg->modify<int>("lumDtSp",1); }
91   if( _cfg->get<int>("lumBlrGuPoint") ) { _cfg->modify<int>("lumBlrGu",1); }
92   if( _cfg->get<int>("lumDtGuPoint") ) { _cfg->modify<int>("lumDtGu",1); }
93   if( _cfg->get<int>("lumAccdGuPoint") ) { _cfg->modify<int>("lumAccdGu",1); }
94
95   if( _cfg->get<int>("lumSsc") ) { _cfg->modify<int>("Syn",1); }
96   if( _cfg->get<int>("lumSsc") ) { _cfg->modify<int>("lumSyn",1); }
97
98   if( _cfg->get<int>("lumSyn") ) { _cfg->modify<int>("Syn",1); }
99   if( _cfg->get<int>("lumSsc") ) { _cfg->modify<int>("Ssc",1); }
100   if( _cfg->get<int>("lumExt1") ) { _cfg->modify<int>("Ext1",1); }
101   if( _cfg->get<int>("lumExt2") ) { _cfg->modify<int>("Ext2",1); }
102   if( _cfg->get<int>("lumBlrPl") ) { _cfg->modify<int>("BlrPl",1); }
```

```
103   if( _cfg->get<int>("lumDtPl") ) { _cfg->modify<int>("DtPl",1); }
104   if( _cfg->get<int>("lumAccd") ) { _cfg->modify<int>("Accd",1); }
105   if( _cfg->get<int>("lumBlrSp") ) { _cfg->modify<int>("BlrSp",1); }
106   if( _cfg->get<int>("lumDtSp") ) { _cfg->modify<int>("DtSp",1); }
107   if( _cfg->get<int>("lumBlrGu") ) { _cfg->modify<int>("BlrGu",1); }
108   if( _cfg->get<int>("lumDtGu") ) { _cfg->modify<int>("DtGu",1); }
109   if( _cfg->get<int>("lumAccdGu") ) { _cfg->modify<int>("AccdGu",1); }
110
111   if( _cfg->get<int>("lumSscPoint") ) { _cfg->modify<int>("lumSynPoint",1); }
112 }
```

5.5 /home/mjaniak/Soft/blazar++/include/BlrPlanar.hpp File Reference

```
#include "externalRadiationPlanar.hpp"
```

5.5.1 Detailed Description

Author

Mateusz Janiak

Classes

- class BlrPlanar

5.6 /home/mjaniak/Soft/blazar++/include/BlrSpherical.hpp File Reference

```
#include "externalRadiationSpherical.hpp"
#include "inverseCompton.hpp"
```

5.6.1 Detailed Description

Author

Mateusz Janiak

Classes

- class BlrSpherical

5.7 /home/mjaniak/Soft/blazar++/include/BlrSphericalGu.hpp File Reference

```
#include "externalRadiationGu.hpp"
#include "inverseCompton.hpp"
```

5.7.1 Detailed Description

Author

Mateusz Janiak

Classes

- class BlrSphericalGu

5.8 /home/mjaniak/Soft/blazar++/include/DtPlanar.hpp File Reference

```
#include "externalRadiationPlanar.hpp"
```

5.8.1 Detailed Description

Author

Mateusz Janiak

Classes

- class DtPlanar

5.9 /home/mjaniak/Soft/blazar++/include/DtSpherical.hpp File Reference

```
#include "externalRadiationSpherical.hpp"
#include "inverseCompton.hpp"
```

5.9.1 Detailed Description

Author

Mateusz Janiak

Classes

- class DtSpherical

5.10 /home/mjaniak/Soft/blazar++/include/DtSphericalGu.hpp File Reference

```
#include "externalRadiationGu.hpp"
#include "inverseCompton.hpp"
```

5.10.1 Detailed Description

Author

Mateusz Janiak

Classes

- class DtSphericalGu

5.11 /home/mjaniak/Soft/blazar++/include/electrons.hpp File Reference

```
#include "baseClass.hpp"
#include <gsl/gsl_errno.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_roots.h>
#include <gsl/gsl_linalg.h>
```

5.11.1 Detailed Description

Author

Mateusz Janiak

Classes

- class electrons
- struct gamma_break_params

Functions

- double functionGammaBreak (double x, void *params)
- double solveGammaBreak (double p1, double p2, double gamma_min, double gamma_max, double avg_gamma)
- double f1 (double p1, double p2, double gamma_min, double gamma_max, double gamma_break)
- double f2 (double p1, double p2, double gamma_min, double gamma_max, double gamma_break)

5.11.2 Function Documentation

5.11.2.1 double f1 (double *p1*, double *p2*, double *gamma_min*, double *gamma_max*, double *gamma_break*)

auxillary for calculating gamma_break

```
297                                                                                                 {
298     return ( (1.0-pow(gamma_min/gamma_break,2.0-p1))/(2.0-p1) + (1.0-pow(gamma_break/gamma_max,p2-2.0))/(p2-2
        .0) )*pow(gamma_break,1.0-p2); }
```

5.11.2.2 double f2 (double *p1*, double *p2*, double *gamma_min*, double *gamma_max*, double *gamma_break*)

auxillary for calculating gamma_break

```
300                                                                                                 {
301     return ( (1.0-pow(gamma_min/gamma_break,1.0-p1))/(1.0-p1) + (1.0-pow(gamma_break/gamma_max,p2-1.0))/(p2-1
        .0) )*pow(gamma_break,1.0-p2); }
```

5.11.2.3 double functionGammaBreak (double *x*, void * *params*)

defines function to be solved to get break in electron energy spectrum

Parameters

x	- electron Lorentz gamma * - other parameters defined by struct gamma_break_params

Returns

function value

```
303                                                 {
304   /* x <- gamma_break */
305   struct gamma_break_params *p = (struct gamma_break_params *) params;
306
307   double p1 = p->p1;
308   double p2 = p->p2;
309   double gamma_min = p->gamma_min;
310   double gamma_max = p->gamma_max;
311   double avg_gamma = p->avg_gamma;
312
313   return x*f1(p1, p2, gamma_min, gamma_max, x)/f2(p1, p2, gamma_min, gamma_max, x) - avg_gamma; }
```

5.11.2.4 double solveGammaBreak (double *p1*, double *p2*, double *gamma_min*, double *gamma_max*, double *avg_gamma*)

method to calculate gamma break

Parameters

p1	- spectral index p1
p2	- spectral index p2 - minimal electron Lorentz factor - max electron Lorentz factor - average electron injecton Lorentz factor

Returns

gamma_break

```
315                                                                                                 {
316   bazinga::info("","Entering solve_gamma_break");
317   //  std::cout << "Using p1: " << p1 << " p2: " << p2 << " gmin: " << gamma_min << " gmax: " << gamma_max
       << " avg gamma: " << avg_gamma << std::endl;
318   int status;
319   int iter = 0;
320   int max_iter = 100;
321
322   double r = 0;
323   double x_low = 0;
324   double x_high = 0;
325
326   const gsl_root_fsolver_type *T = gsl_root_fsolver_brent;
327   gsl_root_fsolver *s = gsl_root_fsolver_alloc( T );
328
329   gsl_function F;
330   struct gamma_break_params params = { p1, p2, gamma_min, gamma_max, avg_gamma };
331
332   F.function = &functionGammaBreak;
333   F.params = &params;
334
335   gsl_root_fsolver_set( s, &F, gamma_min, gamma_max );
336
337   bazinga::print_header();
338   bazinga::info("","Trying to find gamma_break ...");
339
340   printf("%5s [%8s, %7s] %8s\n", "iter", "lower", "upper", "root");
341
342   do
343     {
344       iter++;
345       status = gsl_root_fsolver_iterate( s );
346       r = gsl_root_fsolver_root( s );
347       x_low = gsl_root_fsolver_x_lower( s );
348       x_high = gsl_root_fsolver_x_upper( s );
349       status = gsl_root_test_interval( x_low, x_high, 0, 0.001 );
350
351       if( status == GSL_SUCCESS ) { bazinga::print_info("Converged:"); }
352
353       printf("%5d [%.2e %.2e] %.2e\n", iter, x_low, x_high, r);
354     }
```

```
355
356   while( status == GSL_CONTINUE && iter <= max_iter );
357
358   gsl_root_fsolver_free( s );
359   return r; }
```

5.12 /home/mjaniak/Soft/blazar++/include/energyDissProc.hpp File Reference

```
#include "baseClass.hpp"
```

5.12.1 Detailed Description

Author

Mateusz Janiak

Classes

- class energyDissProc

5.13 /home/mjaniak/Soft/blazar++/include/externalRadiation.hpp File Reference

```
#include "energyDissProc.hpp"
#include "inverseCompton.hpp"
```

5.13.1 Detailed Description

Author

Mateusz Janiak

Classes

- class externalRadiation

5.14 /home/mjaniak/Soft/blazar++/include/externalRadiationGu.hpp File Reference

```
#include "energyDissProc.hpp"
#include "inverseCompton.hpp"
```

5.14.1 Detailed Description

Author

Mateusz Janiak

Classes

- class externalRadiationGu

5.15 /home/mjaniak/Soft/blazar++/include/externalRadiationPlanar.hpp File Reference

```
#include "energyDissProc.hpp"
#include "logGeometry.hpp"
#include "inverseCompton.hpp"
```

5.15.1 Detailed Description

Author

Mateusz Janiak

Classes

- class externalRadiationPlanar

5.16 /home/mjaniak/Soft/blazar++/include/externalRadiationSpherical.hpp File Reference

```
#include "energyDissProc.hpp"
#include "inverseCompton.hpp"
```

5.16.1 Detailed Description

Author

Mateusz Janiak

Classes

- class externalRadiationSpherical

5.17 /home/mjaniak/Soft/blazar++/include/magneticField.hpp File Reference

```
#include "baseClass.hpp"
```

5.17.1 Detailed Description

Author

Mateusz Janiak

Classes

- class magneticField

5.18 /home/mjaniak/Soft/blazar++/include/observer.hpp File Reference

```
#include "baseClass.hpp"
#include <gsl/gsl_errno.h>
#include <gsl/gsl_spline.h>
#include "QuasarAccDisk.hpp"
```

5.18.1 Detailed Description

Author

Mateusz Janiak

Classes

- class observer

5.19 /home/mjaniak/Soft/blazar++/include/QuasarAccDisk.hpp File Reference

```
#include "baseClass.hpp"
#include "logGeometry.hpp"
```

5.19.1 Detailed Description

Author

Mateusz Janiak

Classes

- class QuasarAccDisk

5.20 /home/mjaniak/Soft/blazar++/include/selfSynCompton.hpp File Reference

```
#include "energyDissProc.hpp"
#include "magneticField.hpp"
#include "inverseCompton.hpp"
```

5.20.1 Detailed Description

Author

Mateusz Janiak

Classes

- class selfSynCompton

5.21 /home/mjaniak/Soft/blazar++/include/synchrotron.hpp File Reference

```
#include "energyDissProc.hpp"
#include "magneticField.hpp"
#include <gsl/gsl_sf_bessel.h>
```

5.21.1 Detailed Description

Author

Mateusz Janiak

Classes

- class synchrotron

5.22 /home/mjaniak/Soft/blazar++/src/AccdPlanar.cpp File Reference

```
#include "AccdPlanar.hpp"
#include "electrons.hpp"
```

5.22.1 Detailed Description

Author

Mateusz Janiak

5.23 /home/mjaniak/Soft/blazar++/src/baseClass.cpp File Reference

```
#include "baseClass.hpp"
```

5.23.1 Detailed Description

Author

Mateusz Janiak

5.24 /home/mjaniak/Soft/blazar++/src/blazar.cpp File Reference

```
#include <scfgp.hpp>
#include <bazinga.hpp>
#include "blazar.hpp"
#include "baseClass.hpp"
#include "jetGeometry.hpp"
#include "electrons.hpp"
#include "magneticField.hpp"
#include "observer.hpp"
#include "synchrotron.hpp"
#include "plots.hpp"
#include "selfSynCompton.hpp"
#include "externalRadiation.hpp"
#include "BlrPlanar.hpp"
#include "DtPlanar.hpp"
#include "AccdPlanar.hpp"
#include "BlrSpherical.hpp"
#include "DtSpherical.hpp"
#include "BlrSphericalGu.hpp"
#include "DtSphericalGu.hpp"
#include "AccdSphericalGu.hpp"
#include "QuasarAccDisk.hpp"
```

5.24.1 Detailed Description

Main blazar++ file

Author

Mateusz Janiak

Functions

- int main (int argc, char ∗argv[])

5.24.2 Function Documentation

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

Main programme starts here

```
32 {
33   bazinga::print_section( );
34   bazinga::print_info("\t\tBlazar++");
35   bazinga::print_section( );
36
37   scfgp *cfg = new scfgp( ); /* Initialize config file reader */
38
39   if( argc != 1 ) { cfg->addConfigFile( argv[1] ); }
40   else {
41     bazinga::error("main","No config file given. You forgot about it?");;
42     exit( 0 ); }
43
```

```
44   /* Gather information about processes used */
45   getConfigSwitches( cfg );
46
47   /* initialize jet geometry */
48   jetGeometry* rJet = new jetGeometry( cfg );
49   rJet -> printInfo( );
50
51   /* initialize electrons*/
52   electrons* ele = new electrons( cfg, rJet, "ele" );
53   ele -> printInfo( );
54
55   /* initialize magnetic field */
56   magneticField* B = new magneticField( cfg, rJet, "mag" );
57   B -> printInfo( );
58
59   /* initialize observer to caluclate luminosities  */
60   observer* obs = new observer( cfg, rJet, "obs" );
61   obs -> printInfo( );
62
63   /* gather information about processes used */
64   safetyConfigSwitches( cfg, ele );
65
66   /* initialize Synchroton */
67   energyDissProc* Syn = cfg->get<int>("Syn") ? new synchrotron( cfg, rJet, ele, B,
       "syn" ) : NULL;
68
69   if( cfg->get<int>("Syn") ) { Syn -> printInfo( ); }
70   if( cfg->get<int>("Syn") ) { ele -> addEnDissProc( Syn ); }
71   if( cfg->get<int>("lumSyn") ) { obs -> addEnDissProc( Syn ); }
72   if( cfg->get<int>("lumSynPoint") ) { obs -> addEnDissProcPoint( Syn ); }
73
74   /* initialize synchrotron self Compton */
75   energyDissProc* Ssc = cfg->get<int>("Ssc") ? new selfSynCompton( cfg, rJet,
      ele, B, "ssc" ) : NULL;
76
77   /* I probably should not be doing that dynamic_cast but I really have no better idea ... */
78   if( cfg->get<int>("Ssc") ) { dynamic_cast<selfSynCompton*>(Ssc)->syn=Syn; }
79
80   if( cfg->get<int>("Ssc") ) { Ssc -> printInfo( ); }
81   if( cfg->get<int>("Ssc") ) { ele -> addEnDissProc( Ssc ); }
82   if( cfg->get<int>("lumSsc") ) { obs -> addEnDissProc( Ssc ); }
83   if( cfg->get<int>("lumSscPoint") ) { obs -> addEnDissProcPoint( Ssc ); }
84
85   /* initialize External Radiation - Blr */
86   energyDissProc* Ext1 = cfg->get<int>("Ext1") ? new
      externalRadiation( cfg, rJet, ele, "ext1" ) : NULL;
87   if( cfg->get<int>("Ext1") ) { Ext1 -> printInfo( ); }
88   if( cfg->get<int>("Ext1") ) { ele->addEnDissProc( Ext1 ); }
89   if( cfg->get<int>("lumExt1") ) { obs->addEnDissProc( Ext1 ); }
90   if( cfg->get<int>("lumExt1Point") ) { obs->addEnDissProcPoint( Ext1 ); }
91
92   /* initialize External Radiation - Dt */
93   energyDissProc* Ext2 = cfg->get<int>("Ext2") ? new
      externalRadiation( cfg, rJet, ele, "ext2" ) : NULL;
94   if( cfg->get<int>("Ext2") ) { Ext2 -> printInfo( ); }
95   if( cfg->get<int>("Ext2") ) { ele->addEnDissProc( Ext2 ); }
96   if( cfg->get<int>("lumExt2") ) { obs->addEnDissProc( Ext2 ); }
97   if( cfg->get<int>("lumExt2Point") ) { obs->addEnDissProcPoint( Ext2 ); }
98
99   /* initialize External Radiation Planar Geometry  - BlrPl */
100   energyDissProc* BlrPl = cfg->get<int>("BlrPl") ? new BlrPlanar( cfg, rJet, ele, "
      blrPl" ) : NULL;
101   if( cfg->get<int>("BlrPl") ) { BlrPl -> printInfo( ); }
102   if( cfg->get<int>("BlrPl") ) { ele -> addEnDissProc( BlrPl ); }
103   if( cfg->get<int>("BlrPl") ) { obs -> addExtPl( BlrPl ); }
104   if( cfg->get<int>("lumBlrPl") ) { obs -> addEnDissProc( BlrPl ); }
105   if( cfg->get<int>("lumBlrPlPoint") ) { obs -> addEnDissProcPoint( BlrPl ); }
106
107   /* initialize External Radiation Planar Geometry - DtPl */
108   energyDissProc* DtPl = cfg->get<int>("DtPl") ? new DtPlanar( cfg, rJet, ele, "dtPl"
       ) : NULL;
109   if( cfg->get<int>("DtPl") ) { DtPl -> printInfo( ); }
110   if( cfg->get<int>("DtPl") ) { ele -> addEnDissProc( DtPl ); }
111   if( cfg->get<int>("DtPl") ) { obs -> addExtPl( DtPl ); }
112   if( cfg->get<int>("lumDtPl") ) { obs -> addEnDissProc( DtPl ); }
113   if( cfg->get<int>("lumDtPlPoint") ) { obs -> addEnDissProcPoint( DtPl ); }
114
115   /* initialize External Radiation Planar Geometry - Accd */
116   energyDissProc* Accd = cfg->get<int>("Accd") ? new AccdPlanar( cfg, rJet, ele, "
      accd" ) : NULL;
117   if( cfg->get<int>("Accd") ) { Accd -> printInfo( ); }
118   if( cfg->get<int>("Accd") ) { ele -> addEnDissProc( Accd ); }
119   if( cfg->get<int>("Accd") ) { obs -> addExtPl( Accd ); }
120   if( cfg->get<int>("lumAccd") ) { obs -> addEnDissProc( Accd ); }
121   if( cfg->get<int>("lumAccdPoint") ) { obs -> addEnDissProcPoint( Accd ); }
122
123   /* initialize External Radiation Spherical Geometry  - BlrSp */
```

```
124   energyDissProc* BlrSp = cfg->get<int>("BlrSp") ? new
      BlrSpherical( cfg, rJet, ele, "blrSp" ) : NULL;
125   if( cfg->get<int>("BlrSp") ) { BlrSp -> printInfo( ); }
126   if( cfg->get<int>("BlrSp") ) { ele -> addEnDissProc( BlrSp ); }
127   if( cfg->get<int>("BlrSp") ) { obs -> addExtSp( BlrSp ); }
128   if( cfg->get<int>("lumBlrSp") ) { obs -> addEnDissProc( BlrSp ); }
129   if( cfg->get<int>("lumBlrSpPoint") ) { obs -> addEnDissProcPoint( BlrSp ); }
130
131   /* initialize External Radiation Spherical Geometry - DtSp */
132   energyDissProc* DtSp = cfg->get<int>("DtSp") ? new DtSpherical( cfg, rJet, ele,
      "dtSp" ) : NULL;
133   if( cfg->get<int>("DtSp") ) { DtSp -> printInfo( ); }
134   if( cfg->get<int>("DtSp") ) { ele -> addEnDissProc( DtSp ); }
135   if( cfg->get<int>("DtSp") ) { obs -> addExtSp( DtSp ); }
136   if( cfg->get<int>("lumDtSp") ) { obs -> addEnDissProc( DtSp ); }
137   if( cfg->get<int>("lumDtSpPoint") ) { obs -> addEnDissProcPoint( DtSp ); }
138
139   /* initialize External Radiation Gu - Blr */
140   energyDissProc* BlrGu = cfg->get<int>("BlrGu") ? new
      BlrSphericalGu( cfg, rJet, ele, "blrGu" ) : NULL;
141   if( cfg->get<int>("BlrGu") ) { BlrGu -> printInfo( ); }
142   if( cfg->get<int>("BlrGu") ) { ele->addEnDissProc( BlrGu ); }
143   if( cfg->get<int>("BlrGu") ) { obs -> addExtGu( BlrGu ); }
144   if( cfg->get<int>("lumBlrGu") ) { obs->addEnDissProc( BlrGu ); }
145   if( cfg->get<int>("lumBlrGuPoint") ) { obs->addEnDissProcPoint( BlrGu ); }
146
147   /* initialize External Radiation Gu - Dt */
148   energyDissProc* DtGu = cfg->get<int>("DtGu") ? new DtSphericalGu( cfg, rJet,
      ele, "dtGu" ) : NULL;
149   if( cfg->get<int>("DtGu") ) { DtGu -> printInfo( ); }
150   if( cfg->get<int>("DtGu") ) { ele->addEnDissProc( DtGu ); }
151   if( cfg->get<int>("DtGu") ) { obs -> addExtGu( DtGu ); }
152   if( cfg->get<int>("lumDtGu") ) { obs->addEnDissProc( DtGu ); }
153   if( cfg->get<int>("lumDtGuPoint") ) { obs->addEnDissProcPoint( DtGu ); }
154
155   /* initialize External Radiation Gu - Accd */
156   energyDissProc* AccdGu = cfg->get<int>("AccdGu") ? new
      AccdSphericalGu( cfg, rJet, ele, "accdGu" ) : NULL;
157   if( cfg->get<int>("AccdGu") ) { AccdGu -> printInfo( ); }
158   if( cfg->get<int>("AccdGu") ) { ele->addEnDissProc( AccdGu ); }
159   if( cfg->get<int>("AccdGu") ) { obs -> addExtGu( AccdGu ); }
160   if( cfg->get<int>("lumAccdGu") ) { obs->addEnDissProc( AccdGu ); }
161   if( cfg->get<int>("lumAccdGuPoint") ) { obs->addEnDissProcPoint( AccdGu ); }
162
163   /* initialize Quasar Accretion Disk */
164   QuasarAccDisk* QAccd = cfg->get<int>("QAccd") ? new QuasarAccDisk( cfg, "QAccd"
       ) : NULL;
165   if( cfg->get<int>("QAccd") ) { QAccd -> printInfo( ); }
166
167   /* read QAccd template data */
168   if( cfg->get<int>("QAccd") ) { QAccd -> readTemplate( ); }
169   if( cfg->get<int>("QAccd") ) { QAccd -> scaleTemplate( ); }
170   if( cfg->get<int>("QAccd") ) { QAccd -> saveTemplate( ); }
171
172   bazinga::print_section( );
173   /* Before proceeding further show us what you're going to do */
174   ele -> listEnDissProc( );
175
176   /* Before proceeding further show us what luminosities are you going to calculate */
177   obs -> listEnDissProc( );
178   obs -> listEnDissProcPoint( );
179   obs -> listExtPl( );
180   obs -> listExtSp( );
181   obs -> listExtGu( );
182
183   /* ------------------------------------------------------------------------------------------ */
184   /* BEGIN root Canvases and Graphs */
185   /* ------------------------------------------------------------------------------------------ */
186
187 #ifdef USE_ROOT
188 /* Create TApplication */
189 TApplication* theApp = cfg->get<int>("root") ? new TApplication("App",0,0) : NULL;
190
191   /* Create all the necessary TCanvas */
192   /* First row - electrons*/
193 TCanvas *eleInjCanvas = cfg->get<int>("root") && ele->ifEvol() ? new TCanvas("EleInj","Electron
       injection",0,0,400,400) : NULL;
194   TCanvas *eleCanvas = cfg->get<int>("root") ? new TCanvas("EleEvo","Electron evolution",400,0,400,400) :
      NULL;
195   /* Second row - luminosities */
196   TCanvas *intLumCanvas = cfg->get<int>("root") && obs->sizeEnDissProc( ) ? new TCanvas("
      IntLum","Intrinsic luminosities",0,430,400,400) : NULL;
197   TCanvas *lumPointCanvas = cfg->get<int>("root") && obs->sizeEnDissProcPoint( ) ? new
      TCanvas("LumPoint","Point luminosities",400,430,400,400) : NULL;
198   TCanvas *lumPointAvgCanvas = cfg->get<int>("root") && obs->sizeEnDissProcPoint( ) ?
      new TCanvas("LumPointAvg","Avg Point luminosities",800,430,400,400) : NULL;
199
```

```
200   gSystem -> ProcessEvents();
201
202   /* Create all the necessary TMultiGraph */
203   TMultiGraph* mgInjNgamma = cfg->get<int>("root") && ele->ifEvol( ) ? new TMultiGraph("EleInj","
      Electron injection;log #gamma;log #gamma^{2} N_{#gamma}") : NULL;
204   TMultiGraph* mgNgamma = cfg->get<int>("root") ? new TMultiGraph("EleEvo","Electron evolution;log
       #gamma;log #gamma^{2} N_{#gamma}") : NULL;
205   TMultiGraph* mgIntLum = cfg->get<int>("root") && obs->sizeEnDissProc( ) ? new TMultiGraph("
      IntLum","Intrinsic luminosities;log #nu;log #nu L_{#nu}") : NULL;
206   TMultiGraph* mgLumPoint = cfg->get<int>("root") && obs->sizeEnDissProcPoint( ) ? new
      TMultiGraph("LumPoint","Point luminosities;log #nu;log #nu L_{#nu}") : NULL;
207   TMultiGraph* mgLumAvgPoint = cfg->get<int>("root") && obs->sizeEnDissProcPoint( ) ?
      new TMultiGraph("LumPointAvg","Avg Point luminosities;log #nu;log #nu L_{#nu}") : NULL;
208 #endif
209
210   /* ------------------------------------------------------------------------------------------ */
211   /* END root Canvases and Graphs */
212   /* ------------------------------------------------------------------------------------------ */
213
214   bazinga::print_section( );
215
216   if( ele->ifEvol( ) ) {
217     bazinga::info( "main", "Starting electron evolution loop ...");
218     bazinga::print_section( ); }
219   else {
220     bazinga::info("main", "No electron evolution. Injection only.");
221     bazinga::print_section( ); }
222
223   /* ------------------------------------------------------------------------------------------ */
224   /* MAIN LOOP STARTS HERE */
225   /* ------------------------------------------------------------------------------------------ */
226
227   for( int i=0;i<rJet->getMaxIndex( );i++ ) {
228     rJet->update( i );
229     rJet->show( );
230     /* inject electrons */
231     ele -> inject( );
232
233     /* update the data about processes that will cool electrons */
234     for( int i=0; i<ele->EnDissProc.size(); i++ ) {
235       bazinga::info( ele->EnDissProc[i]->whoAmI( ), "Updating" );
236       ele -> EnDissProc[i] -> update( ); }
237
238     /* show some info about magnetic field */
239     bazinga::print_info( B->whoAmI(), "Magnetic field", B->getB(), "Gauss");
240
241     /* electrons evolution taking part here*/
242     if( ele -> ifEvol( ) ) {
243       /* save injected electrons */
244       ele -> saveInjection( );
245
246 #ifdef USE_ROOT
247       /* make ROOT plot */
248       if( cfg->get<int>("root") && ele -> ifEvol( ) ) { makeNgammaPlotROOT( ele->
      gamma, ele->S, eleInjCanvas, mgInjNgamma, ele->whoAmI(), rJet ); }
249 #endif
250
251       /* solve evolution equations for electrons */
252       ele -> evolve( ); }
253
254     /* save electrons */
255     ele -> saveNgamma( );
256     /* calculate AvgNgamma; this is just an update with the newest Ngamma so is has to be run everytime in
       a loop */
257     ele -> avgNgamma();
258
259 #ifdef USE_ROOT
260     /* make ROOT plot */
261     if( cfg->get<int>("root") ) { makeNgammaPlotROOT( ele->gamma, ele->Ngamma, eleCanvas, mgNgamma,
      ele->whoAmI(), rJet ); }
262 #endif
263
264     /* ------------------------------------------------------------------------------------------ */
265     /* BEGIN Calculate intrinsic luminosities */
266     /* ------------------------------------------------------------------------------------------ */
267     /* take electrons from the previous step - either evolved, only injected or read from a file */
268     for( int j=0; j<obs->sizeEnDissProc(); j++ ) {
269       bazinga::info( obs -> EnDissProc[j ]-> whoAmI( ), "Calculating intrinsic luminosity." );
270       /* set luminosities */
271       obs -> EnDissProc[j] -> setLpv( );
272       obs -> EnDissProc[j] -> saveLuminosity( );
273
274 #ifdef USE_ROOT
275       if( cfg->get<int>("root") ) { makeLPlotROOT( obs->EnDissProc[j]->vp, obs->
      EnDissProc[j]->Lpv, intLumCanvas, mgIntLum, obs->EnDissProc[j]->whoAmI( ), rJet ); }
276 #endif
277     }
```

```
278     /* ------------------------------------------------------------------------------------------------ */
279     /* END Calculate intrinsic luminosities */
280     /* ------------------------------------------------------------------------------------------------ */
281 
282     /* ------------------------------------------------------------------------------------------------ */
283     /* BEGIN Calculate point luminosities */
284     /* ------------------------------------------------------------------------------------------------ */
285     for( int j=0; j<obs->sizeEnDissProcPoint(); j++ ) {
286       obs -> calculatePointLuminosity( obs->EnDissProcPoint[j] );
287       obs -> savePointLuminosity( obs->EnDissProcPoint[j] );
288 
289 #ifdef USE_ROOT
290       if( cfg->get<int>("root") ) { makeLPlotROOT( obs->EnDissProcPoint[j]->vPoint, obs->EnDissProcPoint[j]
      ->LvPoint, lumPointCanvas, mgLumPoint, obs->EnDissProcPoint[j]->whoAmI( ), rJet ); }
291 #endif
292 
293       /* Averaging spectra */
294       obs -> avgPointLuminosity( obs->EnDissProcPoint[j] ); }
295 
296     /* Calculate accretion disk luminosity - QAccd */
297     if( cfg->get<int>("QAccd") ) { QAccd -> calculateLuminosity( ); }
298 
299     /* ------------------------------------------------------------------------------------------------ */
300     /* END Calculate point luminosities */
301     /* ------------------------------------------------------------------------------------------------ */
302 
303     /* ------------------------------------------------------------------------------------------------ */
304     /* BEGIN Sum point luminosities */
305     /* ------------------------------------------------------------------------------------------------ */
306     if( obs->sizeEnDissProcPoint( ) ) {
307       obs -> sumPointLuminosity( );
308       obs -> savePointLuminositySum( ele->gamma );
309 
310 #ifdef USE_ROOT
311       if( cfg->get<int>("root") ) { makeLSumPlotROOT( obs->vPointSum, obs->LvPointSum,
    lumPointCanvas, mgLumPoint, obs->whoAmI( ), rJet ); }
312 #endif
313     }
314     /* ------------------------------------------------------------------------------------------------ */
315     /* END Sum point luminosities */
316     /* ------------------------------------------------------------------------------------------------ */
317 
318     /* save information on upe vs radius */
319     for( int j=0; j < obs->sizeEnDissProc( ); j++ ) { obs ->  EnDissProc[j] -> saveUpeR( ); }
320 
321     /* save information on Rm vs radius for ExtPl */
322     for( int j=0; j < obs->sizeExtPl( ); j++ ) { dynamic_cast<
    externalRadiationPlanar*>( obs->ExtPl[j] ) -> saveRmVsR( ); }
323   }
324   /* ---------------------------------------------------------------------------------------------------- */
325   /* MAIN LOOP ENDS HERE */
326   /* ---------------------------------------------------------------------------------------------------- */
327 
328   if( ele->ifEvol( ) ) {
329     bazinga::info( ele->whoAmI( ), "Ending electron evolution loop.");
330     bazinga::print_section( ); }
331   else {
332     bazinga::info( ele->whoAmI( ), "Ending electron loop.");
333     bazinga::print_section( ); }
334 
335   /* ---------------------------------------------------------------------------------------------------- */
336 
337   /* save information about NgammaAvg */
338   ele -> saveNgammaAvg( );
339 
340   /* ---------------------------------------------------------------------------------------------------- */
341   /* BEGIN Save average spectra */
342   /* ---------------------------------------------------------------------------------------------------- */
343   for( int j=0; j < obs -> sizeEnDissProcPoint(); j++ ) {
344     obs -> saveAveragedPointLuminosity( obs -> EnDissProcPoint[j] );
345 
346 #ifdef USE_ROOT
347   if( cfg->get<int>("root") ) { makeLPlotROOT( obs->EnDissProc[j]->vPoint, obs->
    EnDissProc[j]->LvPointAvg, lumPointAvgCanvas, mgLumAvgPoint, obs->EnDissProcPoint[j]->whoAmI( ),
    rJet ); }
348 #endif
349   }
350 
351   /* ---------------------------------------------------------------------------------------------------- */
352   /* END Save average spectra */
353   /* ---------------------------------------------------------------------------------------------------- */
354 
355   /* ---------------------------------------------------------------------------------------------------- */
356   /* BEGIN Sum and save averaged point sum spectra */
357   /* ---------------------------------------------------------------------------------------------------- */
358 
359     if( obs -> sizeEnDissProcPoint( ) ) {
```

```
360       obs -> sumPointAvgLuminosity( );
361
362       if( cfg->get<int>("QAccd") ) { obs -> addQAccdLuminosity( QAccd ); }
363
364       obs -> saveAveragedPointLuminositySum( ele->gamma );
365
366 #ifdef USE_ROOT
367     if( cfg->get<int>("root") ) { makeLSumPlotROOT( obs->vPointSum, obs->LvPointAvgSum,
    lumPointAvgCanvas, mgLumAvgPoint, obs->whoAmI( ), rJet ); }
368 #endif
369     }
370
371   /* ------------------------------------------------------------------------------------------------ */
372   /* END Sum and save averaged sum spectra */
373   /* ------------------------------------------------------------------------------------------------ */
374
375   /* ------------------------------------------------------------------------------------------------ */
376   /* BEGIN Calculating flares */
377   /* ------------------------------------------------------------------------------------------------ */
378     obs -> calculateAllFlares( );
379   /* ------------------------------------------------------------------------------------------------ */
380   /* END Calculating flares */
381   /* ------------------------------------------------------------------------------------------------ */
382
383   /* ------------------------------------------------------------------------------------------------ */
384   /* BEGIN make gnuplot plots */
385   /* ------------------------------------------------------------------------------------------------ */
386     bazinga::print_section( );
387     bazinga::info("main","Preparing gnuplot files.");
388     if( cfg->get<int>("saveElectrons") ) { makeNgammaPlots( ele, cfg, rJet ); } /* electron plot */
389     if( cfg->get<int>("saveLumPoint") ) { makeLPointPlots( obs, cfg, rJet ); } /* point source luminosities
     */
390     if( cfg->get<int>("saveLumPoint") ) { makeLPointVsElePlots( obs, cfg, rJet ); } /* point source
    luminosities */
391     if( cfg->get<int>("saveLumPoint") ) { makeLPointSumPlots( obs, cfg, rJet ); } /* point source
    luminosity sum */
392     if( obs->sizeEnDissProc( ) ) { makeUpeRPlots( obs, cfg, rJet ); } /* make Upe vs radius
    plots */
393     if( cfg->get<int>("QAccd") ) { makeLQAccdPlots( QAccd, cfg ); } /* make QAccd plots -  template and
    accretion disk multicolor blackbody */
394
395     if( obs->sizeExtPl( ) ) {
396       makeRmPlots( obs, cfg, rJet ); /* make Rm vs radius plots */
397       makeExtPldLdRPlots( obs, cfg, rJet ); } /* make dLdR vs R plots for ExtPl */
398
399     /* flare plots */
400     if( obs -> ifCalculateFlares( ) ) { makePointFlare( obs, cfg, rJet ); }
401
402     bazinga::info("main","Gnuplot files ready.");
403   /* ------------------------------------------------------------------------------------------------ */
404   /* END make gnuplot plots */
405   /* ------------------------------------------------------------------------------------------------ */
406
407 #ifdef USE_ROOT
408     if( cfg->get<int>("root") && cfg->get<int>("printRoot") ) {
409     bazinga::print_section( );
410     bazinga::info("main","Priniting ROOT canvases");
411     eleCanvas -> Print();
412     if( ele->ifEvol( ) ) { eleInjCanvas -> Print(); }
413     if( obs->sizeEnDissProc( ) ) { intLumCanvas -> Print(); }
414     if( obs->sizeEnDissProcPoint( ) ) { lumPointCanvas -> Print(); }
415     if( obs->sizeEnDissProcPoint( ) ) { lumPointAvgCanvas -> Print(); }
416   }
417 #endif
418
419   bazinga::print_section( );
420     /* print KN info */
421     /* go along the jet once again */
422   for( int i=0;i<rJet->getMaxIndex();i++ ) {
423     rJet -> update( i );
424     for( int j=0; j < ele->EnDissProc.size(); j++ ) { ele->EnDissProc[j]->print_KN_info
    ( ); }
425   }
426
427 bazinga::print_section( );
428 bazinga::info("main","Merci beaucoup. I'm done here.");
429
430 #ifdef USE_ROOT
431  if( cfg->get<int>("root") ) { bazinga::info("main","PRESS 'ENTER' TO QUIT.");
432   std::cin.ignore(); }
433 #endif
434
435   /* free memory */
436  bazinga::info("BLAZAR","Let's free some memory:");
437   for( int j=0; j<ele->EnDissProc.size(); j++ ) { delete ele->
    EnDissProc[j]; }
438   delete obs;
```

```
439   delete ele;
440   delete rJet;
441   delete cfg;
442   delete QAccd;
443 #ifdef USE_ROOT
444   delete theApp;
445 #endif
446 return 0;
447 }
```

5.25 /home/mjaniak/Soft/blazar++/src/BlrPlanar.cpp File Reference

```
#include "BlrPlanar.hpp"
#include "electrons.hpp"
```

5.25.1 Detailed Description

Author

Mateusz Janiak

5.26 /home/mjaniak/Soft/blazar++/src/BlrSpherical.cpp File Reference

```
#include "BlrSpherical.hpp"
#include "electrons.hpp"
```

5.26.1 Detailed Description

Author

Mateusz Janiak

5.27 /home/mjaniak/Soft/blazar++/src/BlrSphericalGu.cpp File Reference

```
#include "BlrSphericalGu.hpp"
#include "electrons.hpp"
```

5.27.1 Detailed Description

Author

Mateusz Janiak

5.28 /home/mjaniak/Soft/blazar++/src/DtPlanar.cpp File Reference

```
#include "DtPlanar.hpp"
#include "electrons.hpp"
```

5.28.1 Detailed Description

Author

Mateusz Janiak

5.29 /home/mjaniak/Soft/blazar++/src/DtSpherical.cpp File Reference

```
#include "DtSpherical.hpp"
#include "electrons.hpp"
```

5.29.1 Detailed Description

Author

Mateusz Janiak

5.30 /home/mjaniak/Soft/blazar++/src/DtSphericalGu.cpp File Reference

```
#include "DtSphericalGu.hpp"
#include "electrons.hpp"
```

5.30.1 Detailed Description

Author

Mateusz Janiak

5.31 /home/mjaniak/Soft/blazar++/src/electrons.cpp File Reference

```
#include "electrons.hpp"
#include "energyDissProc.hpp"
#include "synchrotron.hpp"
```

5.31.1 Detailed Description

Author

Mateusz Janiak

Functions

- double f1 (double p1, double p2, double gamma_min, double gamma_max, double gamma_break)
- double f2 (double p1, double p2, double gamma_min, double gamma_max, double gamma_break)
- double functionGammaBreak (double x, void ∗params)
- double solveGammaBreak (double p1, double p2, double gamma_min, double gamma_max, double avg_-gamma)

5.31.2 Function Documentation

5.31.2.1 double f1 (double *p1*, double *p2*, double *gamma_min*, double *gamma_max*, double *gamma_break*)

auxillary for calculating gamma_break

```
297                                                                                                 {
298   return ( (1.0-pow(gamma_min/gamma_break,2.0-p1))/(2.0-p1) + (1.0-pow(gamma_break/gamma_max,p2-2.0))/(p2-2
      .0) )*pow(gamma_break,1.0-p2); }
```

5.31.2.2 double f2 (double *p1*, double *p2*, double *gamma_min*, double *gamma_max*, double *gamma_break*)

auxillary for calculating gamma_break

```
300                                                                                                 {
301   return ( (1.0-pow(gamma_min/gamma_break,1.0-p1))/(1.0-p1) + (1.0-pow(gamma_break/gamma_max,p2-1.0))/(p2-1
      .0) )*pow(gamma_break,1.0-p2); }
```

5.31.2.3 double functionGammaBreak (double *x*, void * *params*)

defines function to be solved to get break in electron energy spectrum

Parameters

x	- electron Lorentz gamma * - other parameters defined by struct gamma_break_params

Returns

function value

```
303                                                 {
304   /* x <- gamma_break */
305   struct gamma_break_params *p = (struct gamma_break_params *) params;
306
307   double p1 = p->p1;
308   double p2 = p->p2;
309   double gamma_min = p->gamma_min;
310   double gamma_max = p->gamma_max;
311   double avg_gamma = p->avg_gamma;
312
313   return x*f1(p1, p2, gamma_min, gamma_max, x)/f2(p1, p2, gamma_min, gamma_max, x) - avg_gamma; }
```

5.31.2.4 double solveGammaBreak (double *p1*, double *p2*, double *gamma_min*, double *gamma_max*, double *avg_gamma*)

method to calculate gamma break

Parameters

p1	- spectral index p1
p2	- spectral index p2 - minimal electron Lorentz factor - max electron Lorentz factor - average electron injecton Lorentz factor

Returns

gamma_break

```
315                                                                                                 {
316   bazinga::info("","Entering solve_gamma_break");
317   //  std::cout << "Using p1: " << p1 << " p2: " << p2 << " gmin: " << gamma_min << " gmax: " << gamma_max
       << " avg gamma: " << avg_gamma << std::endl;
318   int status;
319   int iter = 0;
```

```
320   int max_iter = 100;
321
322   double r = 0;
323   double x_low = 0;
324   double x_high = 0;
325
326   const gsl_root_fsolver_type *T = gsl_root_fsolver_brent;
327   gsl_root_fsolver *s = gsl_root_fsolver_alloc( T );
328
329   gsl_function F;
330   struct gamma_break_params params = { p1, p2, gamma_min, gamma_max, avg_gamma };
331
332   F.function = &functionGammaBreak;
333   F.params = &params;
334
335   gsl_root_fsolver_set( s, &F, gamma_min, gamma_max );
336
337   bazinga::print_header();
338   bazinga::info("","Trying to find gamma_break ...");
339
340   printf("%5s [%8s, %7s] %8s\n", "iter", "lower", "upper", "root");
341
342   do
343     {
344       iter++;
345       status = gsl_root_fsolver_iterate( s );
346       r = gsl_root_fsolver_root( s );
347       x_low = gsl_root_fsolver_x_lower( s );
348       x_high = gsl_root_fsolver_x_upper( s );
349       status = gsl_root_test_interval( x_low, x_high, 0, 0.001 );
350
351       if( status == GSL_SUCCESS ) { bazinga::print_info("Converged:"); }
352
353       printf("%5d [%.2e %.2e] %.2e\n", iter, x_low, x_high, r);
354     }
355
356   while( status == GSL_CONTINUE && iter <= max_iter );
357
358   gsl_root_fsolver_free( s );
359   return r; }
```

5.32 /home/mjaniak/Soft/blazar++/src/energyDissProc.cpp File Reference

```
#include "energyDissProc.hpp"
#include "electrons.hpp"
```

5.32.1 Detailed Description

Author

Mateusz Janiak

5.33 /home/mjaniak/Soft/blazar++/src/externalRadiation.cpp File Reference

```
#include "externalRadiation.hpp"
#include "electrons.hpp"
```

5.33.1 Detailed Description

Author

Mateusz Janiak

5.34 /home/mjaniak/Soft/blazar++/src/externalRadiationGu.cpp File Reference

```
#include "externalRadiationGu.hpp"
#include "electrons.hpp"
```

5.34.1 Detailed Description

Author

Mateusz Janiak

5.35 /home/mjaniak/Soft/blazar++/src/externalRadiationPlanar.cpp File Reference

```
#include "externalRadiationPlanar.hpp"
#include "electrons.hpp"
```

5.35.1 Detailed Description

Author

Mateusz Janiak

5.36 /home/mjaniak/Soft/blazar++/src/externalRadiationSpherical.cpp File Reference

```
#include "externalRadiationSpherical.hpp"
#include "electrons.hpp"
```

5.36.1 Detailed Description

Author

Mateusz Janiak

5.37 /home/mjaniak/Soft/blazar++/src/magneticField.cpp File Reference

```
#include "magneticField.hpp"
```

5.37.1 Detailed Description

Author

Mateusz Janiak

5.38 /home/mjaniak/Soft/blazar++/src/observer.cpp File Reference

```
#include "observer.hpp"
#include "electrons.hpp"
#include "energyDissProc.hpp"
```

5.38.1 Detailed Description

Author

Mateusz Janiak

5.39 /home/mjaniak/Soft/blazar++/src/QuasarAccDisk.cpp File Reference

```
#include "QuasarAccDisk.hpp"
```

5.39.1 Detailed Description

Author

Mateusz Janiak

5.40 /home/mjaniak/Soft/blazar++/src/selfSynCompton.cpp File Reference

```
#include "selfSynCompton.hpp"
#include "electrons.hpp"
```

5.40.1 Detailed Description

Author

Mateusz Janiak

5.41 /home/mjaniak/Soft/blazar++/src/synchrotron.cpp File Reference

```
#include "synchrotron.hpp"
#include "electrons.hpp"
```

5.41.1 Detailed Description

Author

Mateusz Janiak

Index