`CODE NAME` [1] User's Manual

Marco Sortino

May 9, 2024

[1] Name still to be choosen

Contents

Overview

`CODE NAME` is a tool that allows user to translate FLASH simulated AGNs Jets outputs in spectra. It is capable of elaborate the Spectral Energy Distribution (SED) produced by each blob and how the light interacts with each sphere. It comes with the option to define the distance and the angle of view of the source with respect to the observer, as well as the radiative processes to consider.

Given the nature of FLASH and the block structure it comes out with, `CODE NAME` allows to select which block should take place in the emission process. For this reason `CODE NAME` comes with a small routine that produces histograms and contour plots based on density, pressure, temperature and energy of each block.

The currently available processes are:

- Non-Thermal Bremmstrahlung;
- Synchrotron;
  - Synchrotron Self Absorption;
  - Synchrotron Self Compton;
- External Compton;

The External Compton interacting region available are:

- Cosmic Microwave Background (CMB)
- Shakura Sunyaev Disc (SSDisc)

There are two codes provided: `main` and `plotting`.
The first one is the one used in order to compute the SED. The settings parameters are stored in `config.txt`. More info on how to customize it is found in section 1.3.2.
The latter is used to produce histograms and contour plots. The respective settings parameters are stored in `config_plot.txt`.
They are briefly explained in the following sections.

Soon to be available are the Dust Torus (DT) and the Broad Line Region (BLR).
This code wouldn't have seen the light of the day if not for the `agnpy` library made by Nigro et al (2022).

Chapter 1

How to Use

1.1 Installation

`CODE NAME` currently works on `python3.7` or above. The following libraries need to be installed: `numpy, matplotlib, astropy, agnpy, sherpa, gammapy`.

It has been correctly ran and tested on Ubuntu 23.02. It does **not** work on Windows since `sherpa` is not supported. Any tests on Mac has not been made yet, hence the owner cannot guarantee the correct functioning of the code.

The main directory display the following structure:

- `main.py`
- `plotting.py`
- `config.txt`
- `config_plot.txt`
- `lib/`
- `compiler.sh`

The latter is a bash script and must be run when installing for the first time. On Linux, it can be launched via terminal using:

```
sh compiler.sh
```

It will simply compile a C code that is needed for the LightRayIntersection routine that is used later. The script must be launched via terminal from the same directory in which it is located. Alternatively one can simply compile it directly in `lib/c\_routines/codes/` using

```
gcc -g -Wall raytrace.c -o raytrace -lm
```

and then move the output to `lib/c\_routines/exec/`.

1.2 How to use it

In order to compute the radiative processes we are interested in, it is necessary to edit `config.txt` and select: which radiative process must be computed, which blobs should contribute to the total SED and which structure are present. It is also necessary to specify the input file path and the directory path in which the output must be stored. `CODE NAME` automatically determines the name of the output.

Before proceeding with the computation, `CODE NAME` will print a brief summary of the starting conditions. In those cases in which some contradictory inputs are given it will ask the user whether to proceed or not.

1.3 `main` settings

`main` allows the user to translate FLASH output in SED graphs. The settings parameters are stored in `config.txt`. How it is structured and how to set it is described in 1.3.2. FLASH outputs structure to be given as inputs to `main` are described in 1.3.1.

1.3.1 Input structure

`CODE NAME` accepts as input HDF5 format files and assumes that each parameter is labeled and to be structured as:

label	(array size)
temp	$8 * 8 * 8$
dens	$8 * 8 * 8$
pres	$8 * 8 * 8$
ener	$8 * 8 * 8$
velx	$8 * 8 * 8$
vely	$8 * 8 * 8$
velz	$8 * 8 * 8$
block size	$1 * 3$
coordinates	$1 * 3$

It also looks for the time between the current input simulation time and the next one in `real runtime parameters` section and it assumes to be labeled as `dt`.

The unit measure of each of these parameters need to be stated in `config` under the Unit Settings section.

1.3.2 `config`

`config` allows user to indicate to `CODE NAME` the location of the input file, where to store the output and how to interpret the data. An example of `config.txt` is displayed below.

```
# ***** File paths:
file_path = /home/user/path/to/file.hdf5
file_saving_path = ''
# ***** if you want to launch multiple session put id_launch = y
id_launch = n
file_list_path = ''
# ***** Indexing settings:
pres_Min = 1e-100 cm-1 g s-2
pres_Max = 1e100 cm-1 g s-2
dens_Min = 1e-100 g cm-3
dens_Max = 1e100 g cm-3
energy_Min = 1 erg
energy_Max = 1e50 erg
T_Min = 1 K
T_Max =  1e20 K
# ***** Jet settings:
# PowerLaw -> p
# # BrokenPowerLaw -> p1 p2 gamma_B
# n_e = BrokenPowerLaw
# p1 = 1.1
# p2 = 2.0
# gamma_B = 2
n_e = PowerLaw
p = 1.10
gamma_Min = 1
B = 1e-5 G
Z = 1.22
mu_0 = 1.4
E_released = 1
f_delta = 1
# ***** Processes to compute:
id_brem = n
id_syn = y
id_ssa = y
id_ssc = y
id_ec = n
# --- CMB:
id_cmb = n
# ***** Disc Settings:
id_disc = n
# M_BH = 1e45 g
a = 1
P_jet = 1e45 erg s-1
L_disc = 2e46 erg s-1
eta = 0.083
R_g_unit = y
R_in = 6
R_out = 200
# ***** Plot settings:
# nu_Min and nu_Max are the magnitude number, hence they must be
    float.
# e.g. nu_Min = 8 -> 10^8 Hz
nu_Min = 0
nu_Max = 29
ObserverDistance = 1e27 cm
Obs_theta = 0 deg
Obs_phi = 90 deg
```

```
# ***** Unit settings:
DistanceUnit = 4.3460247626638975e+24 cm
TimeUnit = 2.898016026760564e+17 s
MassUnit = 1.4644322980212338e+43 kg
GridUnit = 70 kpc
```

The file is divided in 7 sections:

- File paths;
- Indexing settings;
- Jet Settings;
- Processes settings;
- Emitting Regions settings;
- Plot settings;
- Unit settings;

Each section is explained in detail below.
`config` has the following structure:

```
    parameter_name = parameter_value
```

The parameter values are classified in four categories:

- numbers;
- units;
- 'yes/no';
- path ;

Before proceeding with the computation, the code checks if the parameter values are in the correct form. An explanation on what is the correct form is given in the following paragraphs. Lines can be commented by placing `#` at the beginning of the line.

Units These inputs must be composed by a value (e.g. 1, 3.14) and an `astropy.units.Quantity`. The correct syntax for the units is the following: the exponential value of a unit must be written after the unit itself, e.g.

$$\frac{g}{s\ cm^3} \rightarrow \texttt{g s-1 cm-3} \tag{1.1}$$

The user is free to express the input using different units of measure although they must still retain the same physical quantity, e.g.:

$$erg \longleftrightarrow J \longleftrightarrow \frac{kg\ m^2}{s^2} \ \texttt{or even}\ \frac{kg\ m^2\ Hz}{s^2\ Hz} \tag{1.2}$$

The scientific notation is valid, i.e. '.' to indicate decimal quantities:

$$3.14,\ 1.27,\ 5.748 \tag{1.3}$$

and the e symbol to indicate the magnitude order, e.g.:

$$\texttt{1e3} \longleftrightarrow 1000,\ \texttt{1e-3} \longleftrightarrow 0.001 \tag{1.4}$$

For more information on these units, the user is referred to check the `astropy` doc. Example:

```
B = 1e-5 G
```

Numbers These parameters are the same of 'Unit' except that they don't come with an `astropy.units.Quantity`. Example:

```
p = 1.1
```

'yes/no' Some input must indicate for an affirmative or negative response. In order to indicate an affirmative response the user must insert `y` or `yes`. In order to indicate a negative response the user can reply with `n` or `no`. Please note that the code is case-insensitive. Any other answer leads the code to assume a negative response. However it induces to print a brief message notifying the user that the input is not clear. Example:

```
id_brem = y
id_syn = n
id_ssa = yep  <- would count as no
```

path These inputs needs to indicate the path required. The path must always be complete, i.e.: `/home/user/write/your/directory/path` The code will not assume to be based on the current directory. Example:

```
file_path = /home/user/path/to/file
```

File Paths

As a brief recap it is composed by:

```
# ***** File paths :
  file_path = /home/user/path/to/file.hdf5
  file_saving_path = ''
# ***** if you want to launch multiple session -> id_launch = y
  id_launch = n
  file_list_path = ''
```

file_path If **id_launch** is set as 'no', put the file path of the simulation file to run. If it is set to 'yes', put the path to the directory containing the list of files to launch. Check the id_launch paragraph for more information.

file_saving_path put the directory path where the user want to save the final plot. If the user don't want to specify any path then it must be leaved as:

```
    file_saving_path = ''
```

, `CODE NAME` will automatically create a `plots/seds/` directory and will store the output file there.

id_launch It is a 'yes/no' input. Put 'yes' if the user wants to launch multiple instance of the code with multiple different files coming from the same directory. In this case it is necessary to write in **file_list_path** the name of a `.txt` file containing the name of all the files to run. The input **file_path** must now indicate the *directory* containing the simulation files. Finally `main.py` must be launched specifying the number of the row as argument. An **example** is depicted below:
Suppose to have the directory `runs` containing the files: `runA.hdf5`, `runB.hdf5`, `runC.hdf5`.
The eventual `'list.txt'` file would be:

```
    runA.hdf5
    runB.hdf5
    runC.hdf5
```

and the `config.txt` file would be:

```
# ***** File paths:
file_path = /home/user/path/to/runs
file_saving_path = ''
# --- if you want to launch multiple session put id_launch = y
id_launch = y
file_list_path = list.txt
```

. Let's say we want to study `runB.hdf5` then we would issue to the terminal the command:

```
> python3 main.py 2
```

This is quite useful when dealing with multiple simulation files. In this case one simply needs to launch a shell script like the pseudo-code below:

```
    n = lenght(list.txt)
    for i in range(1, n)
        > launch python3 main.py i
```

Indexing settings

These settings allows the user to indicate which blocks must be taking into account into the SED computation. Each one of them is an 'unit' input and refers to:

- `pres` - pressure;
- `dens` - density;

- `energy` - energy;
- `T` - temperature;

They are the quantity measured in each cell of the block in the simulation file. The indexing criteria is as it follows: for each block, the main quantities are computed. For each 8x8x8 block, the quantity is calculated by averaging the value of the central 2x2x2 cells and it is then associated to the block itself. Finally, only the blocks with respective value between the `'Min'/'Max'` values are selected. E.g.:

```
temp = np.mean(data.TempSet[i, 2:4,2:4,2:4])*u.K
if temp_Min < temp < temp_Max:
    do choose block;
else
    continue;
```

Jet settings

These inputs describe the jet emitted by the AGN. Currently there is only one distribution function (d.f.) available: `PowerLaw`. Although present in code, `BrokenPowerLaw` is not available yet. However a short description is provided below.

The normalization constant K_e and the maximum energy value of the d.f. are computed by ad-hoc routines taking into account Second Order Fermi Acceleration.

- `B` - (units) - Intensity of the IGM magnetic field;
- `Z` - (numbers) - Atomic number of the blocks;
- `mu_0` - (numbers) - Average atomic mass of swept-up material in units of m_p;
- `E_released` - (numbers) - Apparent isotropy energy release of the explosion in in rest mass solar unit;
- `f_delta` - (numbers) - Shell scale of swept up material;

For more information, refer to Dermer & Menon (2012) §12.

PowerLaw In this case the `config.txt` file must state:

```
n_e = PowerLaw
p = 1.1
gamma_Min = 1
```

where:

- `n_e` - (string) - Indicates which d.f. to use. It can be 'PowerLaw' or 'BrokenPowerLaw'.

- `p` - (numbers) - Exponent of the power law distribution function (d.f.);
- `gamma_Min` - (numbers) - Lower value of the energy in the distribution function in $m_e c^2$ units;

where the d.f. is:

$$N(E) = K_e E^{-p} \text{ where gamma\_Min } < E < \text{ gamma\_Max} \tag{1.5}$$

BrokenPowerLaw (N.B. currently not available) In this case the `config.txt` file must state:

```
n_e = BrokenPowerLaw
p1 = 1.1
p2 = 2.0
gamma_B = 5
gamma_Min = 1
```

where:

- `n_e` - (string) - Indicates which d.f. to use. It can be 'PowerLaw' or 'BrokenPowerLaw';
- `p1` - (numbers) - Exponent of the broken power law distribution function (d.f.);
- `p2` - (numbers) - Exponent of the broken power law distribution function (d.f.);
- `gamma_B` - (numbers) - energy value of the break in the d.f.. In $m_e c^2$ units;
- `gamma_Min` - (numbers) - Lower value of the energy in the distribution function in $m_e c^2$ units;

Where the d.f. is:

$$N(E) = K_e E^{-p*} \text{ where } p* = \begin{cases} p_1 \text{ if } E < E_{\text{break}} \\ p_2 \text{ if } E > E_{\text{break}} \end{cases} \tag{1.6}$$

Processes to compute

This section allows the user to choose which emission process to consider. Each entry is a 'yes/no' input.

- `id_brem` refers to Non-Thermal Bremmstrahlung as computed by Blumenthal & Gould (1970).
- `id_syn` refers to Synchrotron emission and is computed using `agnpy` routines.
  - `id_ssa` refers to Synchrotron-Self Absorption and is computed using `agnpy` routines.

  - id_ssc refers to Synchrotron-Self Compton and is computed using agnpy routines. **High computational times.**

- id_ec refers to External Compton and is computed using agnpy routines. **High computational times.**

Emitting Regions

It contains all the entries for the emitting regions (i.e. CMB, Disc, Dust Torus, BLR). For the time being only CMB and Disc are available.

CMB id_cmb is a 'yes/no' input. For each block the code computes the respective redshift z w.r.t. to the observer and computes the block interaction with the Cosmic Microwave Background Radiation.

Disc id_disc is a 'yes/no' input. If left as affirmative response it is then necessary to state all the following entries of the section. Not otherwise. The user can either provide the black hole mass - commented in the initial config.txt example - via the entry:

- M_BH - (units) - Black Hole Mass

or the dimensionless spin factor and the jet kinetic power:

- a - (numbers) - is the dimensionless spin factor;
- P_jet - (units) - is the jet kinetic power;

These are used in order to compute the Black Hole Mass M_{BH} via the following equation:

$$M_{BH} = \frac{P_{jet} * 1e8 * M_{\odot}}{a^2 * 2e45 \texttt{erg s}^{-1}} \texttt{[g]} \tag{1.7}$$

Source: ([9]).

- L_disc - (units) - Disc Luminosity
- eta - (numbers) - Accretion efficiency

R_g_unit is a 'yes/no' input. In case of an affirmative input, the user provides the disc radius in units of the gravitational radius R_g computed as:

$$R_g = \frac{2GM_{BH}}{c^2} \tag{1.8}$$

hence R_in and R_out are numbers. Otherwise they must be speficied as units.

- R_in - (numbers/units) - Internal disc radius
- R_out - (numbers/units) - External disc radius

Plot settings

These entries define the plotting settings regarding the range of frequencies over which the SED must be computed and the observer parameters:

- `nu_Min` - (numbers) - indicates the magnitude number of the minimum value of the frequency range interval;
- `nu_Max` - (numbers) - indicates the magnitude number of the maximum, value of the frequency range interval;

The frequency array is defined as:

```
nu = np.logspace(nu_Min, nu_Max) * u.Hz
```

therefore both need to be numbers and to indicate the magnitude number. E.g.

$$\texttt{nu\_Min = 8} \rightarrow \nu_{min} = 10^8 \ Hz \tag{1.9}$$

- `ObserverDistance` - (units) - Distance of the observer from the source;
- `Obs_theta` - (units) - Azimuthal angle of the observer w.r.t. the source;
- `Obs_phi` - (units) - Polar angle of the observer w.r.t. the source;

Figure 1.1 shows the correct coordinates. Hence:

$$\begin{cases} x = \rho \sin(\phi) \cos(\theta) \\ y = \rho \sin(\phi) \sin(\theta) \\ z = \rho \cos(\phi) \end{cases} \tag{1.10}$$

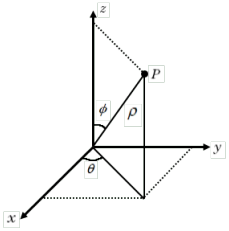

Figure 1.1: θ is the azimuthal angle, while ϕ is the polar one.

Unit settings

Here the user indicates how the block attributes have been measured and the dimension of the Grid in the original simulation file.

- `DistanceUnit` - (units) - Distance unit used in the original simulation.

- `TimeUnit` - (units) - Time unit used in the original simulation.
- `MassUnit` - (units) - Mass unit used in the original simulation.
- `GridUnit` - (units) - Grid unit dimension in the original simulation grid.

It is customary to provide these units in function of the Hubble constant H_0. However, the code is not able to determine it and thus it would return as an error. Please take into consideration to provide the correct value.

Summary

Before going on, the code will print out a short summary of the starting conditions showing the input file name, the emissions to be computed, the name of the output file and where it will be stored. An example is shown below:

```
    Initialization complete. Here a short summary:
------- Summary -------
File name:
-  run1.hdf5
Emission processes:
- Bremmstrahlung
- Synchrotron
- - self-absorption
- - self-Compton
- External Compton
- - CMB
- - Disc
The selected power law is: PowerLaw
The resulting plot will be called: run1_B_S_a_c_EC_cmb_disc.png
It will be saved in: /home/marco/coding/python/last_vers/plots
```

The output name is defined by the processes used to compute it. The structure is `name` + :

- `B` if brehmstrallung is to be considered.
- `S` if synchrotron emission is to be considered.
- `-a` if synchrotron self-absorption is to be considered.
- `-c` if synchrotron self-Compton is to be considered.
- `EC` if External Compton is to be considered.
- `-CMB` if CMB is to be considered.
- `- Disc` if the disc is to be considered.

Finally, if the user should indicate the presence of one the targets emitting sources which photons would be subject to External Compton without but the latter is set as off, the code warns the user and ask whether to continue or not:

```
    Attention, External Compton is set as off but the following
    processes are still set to be computed:
- CMB
- Disc
This may result in long computational times without any contribute
    to the final spectrum. Do you still want to proceed? (y/yes)
```

Please bear in mind that the processes of SSC and EC are time consuming.

1.4 `plotting` settings

`plotting` is used to produce the histograms and contour plots of the blobs physical quantities i.e. temperature, energy, density and pressure. It uses the same input as `main`. In order to properly set and decide which plot to produce, the user can modify `config\_plot.txt`. Before proceeding with the plots, `plotting` will print a brief summary of the starting conditions.

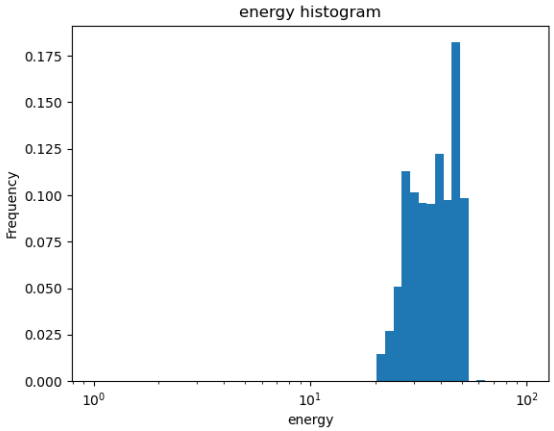

Figure 1.2: Example of histogram plot

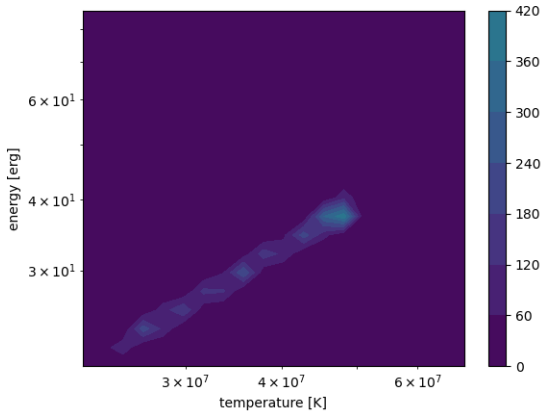

Figure 1.3: Example of contour plot

1.4.1 config_plot

`config_plot` allows user to specify the location of the input file, where to store the output, which plot to produce and which block to consider. An example is displayed below. It can be classified in file path info, graphs settings, interval settings and unit settings. The latter two works in a similar way to the one described in `config`. See 1.3.2 and 1.3.2.

```
#
file_path = /home/user/path/to/file.hdf5
file_saving_path = ''
# If the user wants to have the arrays printed:
display_arrays = yes
# AVAILABLE LABEL:
# -- 'density'
# -- 'pressure'
# -- 'energy'
# -- 'temperature
kind = histogram
#  AVAILABLE LABEL:
# -- 'histogram'
#       if 'kind' = histogram then:
#           bin_rule = (one between sqrt, rice or sturges)
#           label1 = parameter1
#           label2 = parameter2 etc.
#       example: label1 = density
# -- 'contour'
#       if kind = counter then the following must be stated:
    x_label and y_label must be stated.
#           - x_label (which quantity must be on x-axis)
#           - y_label (which quantity must be on y-axis)
#           - n (number of bins to consider)
#           - clim (max value to display on the color bar)
#       n number of bins must be stated aswell.
```

```
#       example: x_label = density
#       example: y_label = pressure
#       example:       n = 100
#
label1 = temperature
label2 = energy
bin_rule = sqrt
x_label = temperature
y_label = energy
n = 20
clim = 1000
# ****** Interval settings:
pres_Min = 1e-20 cm-1 g s-2
pres_Max = 1e-5 cm-1 g s-2
dens_Min = 1e-30 g cm-3
dens_Max = 2e-20 g cm-3
energy_Min = 1 erg
energy_Max = 1e2 erg
T_Min = 1e1 K
T_Max = 1e20 K
# ***** Unit settings:
DistanceUnit = 4.3460247626638975e+24 cm
TimeUnit = 2.898016026760564e+17 s
MassUnit = 1.4644322980212338e+43 kg
GridUnit = 70 kpc
```

File path info

Here the user must specify the location of the input file and the directory in where to save it. Similar to `main`, if the user wish not to specify any path where to save the output file, the code will automatically create a directory in `/plots/graphs`.

Graph settings

`plotting` comes up with the possibility to plot histograms or contour plot. This can be chosen by changing the parameter value `kind`. Each cases come with its own parameters to use. They are described below. The available label for the physical quantities are:

```
    'density'
    'pressure'
    'energy'
    'temperature'
```

There is no need to comment with # the unused parameters as they will be automatically ignored by the code.

With `display_arrays` the user can choose if print the name of the array printed or not on the graph.

histogram Example of histogram parameters:

```
kind = histogram
label1 = temperature
label2 = energy
bin_rule = sqrt
```

The user can choose how many label to define. In any case the rule to follow is 'label' + 'number' in increasing order. This will result in more graph to be produced. In the example above the code would produce two histograms. Each histogram is labeled accordingly.

Regarding the `bin_rule`, there are three currently laws available: `sqrt, rice, sturges`. Each of them specify the number of bin n_{bin} to consider based on the number of the blocks n_{blocks} rounded to the nearest integer.

- `sqrt`

$$n_{\text{bin}} = \texttt{round}(n_{\text{blocks}}^{1/2}) \tag{1.11}$$

- `rice`

$$n_{\text{bin}} = 2 * \texttt{round}(n_{\text{blocks}}^{1/3}) \tag{1.12}$$

- `sturges`

$$n_{\text{bin}} = \texttt{round}(\log_2(n_{\text{block}}) + 1) \tag{1.13}$$

counter Example of counter parameters:

```
kind = counter
x_label = density
y_label = pressure
n = 100
cmap = 1000
```

In this case it is necessary to specify the x and y labels, the number of bins `n` and the max value of the color map `cmap` to display on the colour bar.

Summary

Before starting to compute the graphs, the code will print a brief summary showing which plot it's going to produce, from which file, where it will be stored and which name it will have.

The name etiquette is as it follows: `name_ + kind_ + (x_) label_ ( + y_label) +.png` Example:

```
file_contour_density_pressure.png
file_histogram_temperature.png
file_histogram_energy.png
```

Chapter 2

How it Works

This chapter focuses on the functionality of `main.py` and `plotting.py`. Section 2.1 details how data is handled and reduced. Section 2.2 shows the criteria used to define the subsets of blocks taking part into Inverse Compton process. Section 2.3 explain the algorithm used to gauge the interaction of photons with the blobs who lies on the photons flux trajectory. Section 2.5 shows how blob are defined and seen. Section 2.6 explains the formulas used in the different emission processes.

2.1 Data Management

The code works by defining a `Block` class object for each block composing the original input file where it stores the necessary data and information. A list of `Block` is created and iterated through in various tasks. Each task calculates specific quantities, updates the corresponding `Block` and finally feeds the data to the emission process routines.

2.1.1 Data Read

`CODE NAME` assumes each input file to have the `HDF5` format and the data to be labeled and structured as such:

label	(array size)
temp	$8 * 8 * 8$
dens	$8 * 8 * 8$
pres	$8 * 8 * 8$
ener	$8 * 8 * 8$
velx	$8 * 8 * 8$
vely	$8 * 8 * 8$
velz	$8 * 8 * 8$
block size	$1 * 3$
coordinates	$1 * 3$

It also looks for the time between the current input simulation time and the next one in `real runtime parameters` section and it assumes to be labeled as `dt`.

2.1.2 Data reduction

FLASH computes data in a grid where each block is made by 8^3 cells. This cannot be handled efficiently in a long computation, hence `CODE NAME` computes the mean of the central 2^3 cube resulting in each `Block` being defined with a single quantity for physical quantity, as shown in Figure 2.1

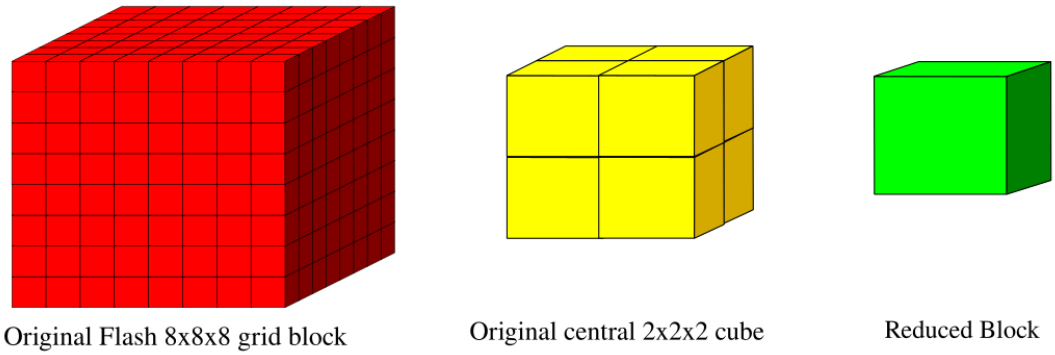

Figure 2.1: The block reduction process. From the original 8^3 grid, the code selects only the central 2^3 cube values. It makes a mean and define the final `Block` object which is used during computations.

2.1.3 `Block` Definition

With the concept of `Block` we mean the fundamental unit which stores all the information and parameters that compose the jet. While Flash divides the space in a grid composed by 8^3 blocks, `Block` is a reduced structure.

Each `Block` is composed by the following quantities.

Note: The quantities followed by an * are those which are stored dimensionless although they are computed with a certain units of measure.

Coordinate quantities:

- x* (cm)
- y* (cm)
- z* (cm)
- center* (x, y, z)

Average of physical quantities:

- temp T (K)
- dens ρ $(g\ cm^{-3})$
- pres P $(g\ cm^{-1}\ s^{-2})$

- energy E (erg)
- velx v_i $km\ s^{-1}$
- vely v_y $km\ s^{-1}$
- velz v_z $km\ s^{-1}$

Computed quantities (more info on how they are computed in the following sections):

- radius r (cm)
- distance (to the source) d (cm)
- obs_distance (distance to the observer) (cm)
- redshift
- n_0 (number density) n_0 (cm^{-3})
- opacity k

Radius

It is used to define the blob. We assume the blob to be a round sphere with radius equal to half of the block size. Hence:

$$r = 0.5\ \texttt{(block size)}_x\ [\text{cm}] \tag{2.1}$$

Distance to the source

We assume the source to be at $O \equiv (0,\ 0,\ 0)$ hence:

$$d = \sqrt{x^2 + y^2 + z^2} \tag{2.2}$$

Distance to the observer

Supposing the observer to be in $P \equiv (P_x, P_y, P_z)$ then:

$$\text{obs}_{\text{distance}} = \sqrt{(x - P_x)^2 + (y - P_y)^2 + (z - P_z)^2}\ \ [\text{cm}] \tag{2.3}$$

Redshift

Computes the redshift z associated to the distance of the block from the observer. It uses the `astropy.Coordinates.Distance` module.

Number Density n_0

Number density n_0 is used to compute opacity. Is defined as:

$$n_0 = \frac{\rho}{\mu_0 m_p} \ [\mathrm{cm}^{-3}] \tag{2.4}$$

where ρ is the density in the block, m_p is the mass of the proton and μ_0 is the average atomic mass of swept-up material in units of m_p.

Opacity

See 2.6.2.

2.2 Inverse Compton Selection Criteria

In the Inverse Compton Scattering process, only the blobs on the external boundary significantly influence the outcome. While inner layers might contribute to scattering, the significantly reduced free mean path of photons within them leads to higher interaction probabilities with the blobs, resulting in energy transfer through different mechanisms.

The idea to compute the Inverse Compton for each blob and consider the photon ray path has been scraped due to the inner computational overload that the computation of the scattering process brings. Therefore, focusing solely on the external layer blobs becomes a more practical approach due to the significantly reduced influence of inner layers. This means that if we have n points scattered in the space, we would like to compute the Inverse Compton only on those blobs which lie on the external boundary. This reduces the task effectively to a *convex hull* problem.

2.2.1 Convex Hull Problem

Suppose to have a set X of n points scattered in an Euclidean space. The convex hull of X is the minimal convex set containing X. The code uses the routine `ConvexHull` from `scipy`, which uses the program `QHull` [1] to return the vertices of X, i.e. all the points $P_i \in \partial X$. The idea is to feed a list of points to `ConvexHull` and take the vertices list. The Inverse Compton is computed only on the corresponding blobs.

This is shown in Figure 2.2 where the function returns the coordinates of the points lying on the red line. The number of points returned depends on the depth of the algorithm. However, the `qhull` implementation found in `scipy` does not allow to define the depth (i.e. the number of points returned) of the algorithm. The actual code returns too few points (e.g. in some preliminary tests, for a dataset of around 3000 points, the code returned around 20 points) hence as a provisional approach it has been decided to iterate the algorithm until it returns the 10% of the total number of blobs.

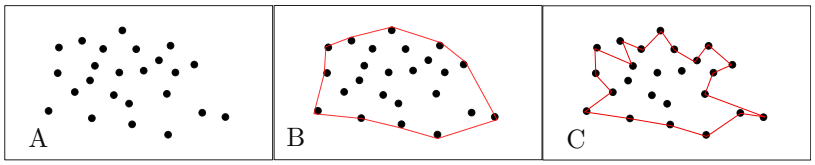

Figure 2.2: An application of the Convex Hull algorithm. Given a set of points, the convex hull is the minimal convex set containing the points. The algorithm returns the vertices of the set. I.e. the set which lies on the frontier of the set. Figure A shows the set of points $\in X$. B shows the result of the Convex Hull algorithm with a low depth while C shows the result with an higher depth.

2.3 Raytracing

Along the trajectory of a photons flux may lie the other blobs. In these cases, in order to access the proper SED, it is necessary to quantify the number of interactions of the photons when going from a point A to a point B. Some examples when this becomes necessary are the routines of Inverse Compton Scattering where the photons are scattered by the blobs which lie on the external boundary of the jet and may need to go through the whole jet in order to reach the observer. Similarly, (although the code takes a subset of the blobs which lie on the external layers, see 2.2), blobs on the inner part of the jet may see a "dimmer" light coming from the disk and other structures.

2.3.1 Generalization of the problem

Suppose to have n round spheres scattered in an empty space. Given an arbitrary point P, from each sphere's center starts a segment having as extremes respectively the center of the sphere and the point P. For each segment it is necessary to assess if it intersects with other spheres and if so, compute the fraction of the segment intersected with the sphere.

2.3.2 Algorithm

A scheme is shown in Figure 2.3. Here explained step by step:

1. Compute the line l passing through two points (P and the center C_i of the i-sphere).
2. Compute the plane $\hat{A}_p$ perpendicular to the line l and passing through P.
3. Compute the plane $\hat{A}_{C_i}$ perpendicular to the line l and passing through C_i
4. Consider only the spheres lying between the two planes;
5. For each sphere $j \neq i$ that lies between the two planes, compute the distance between C_j and l

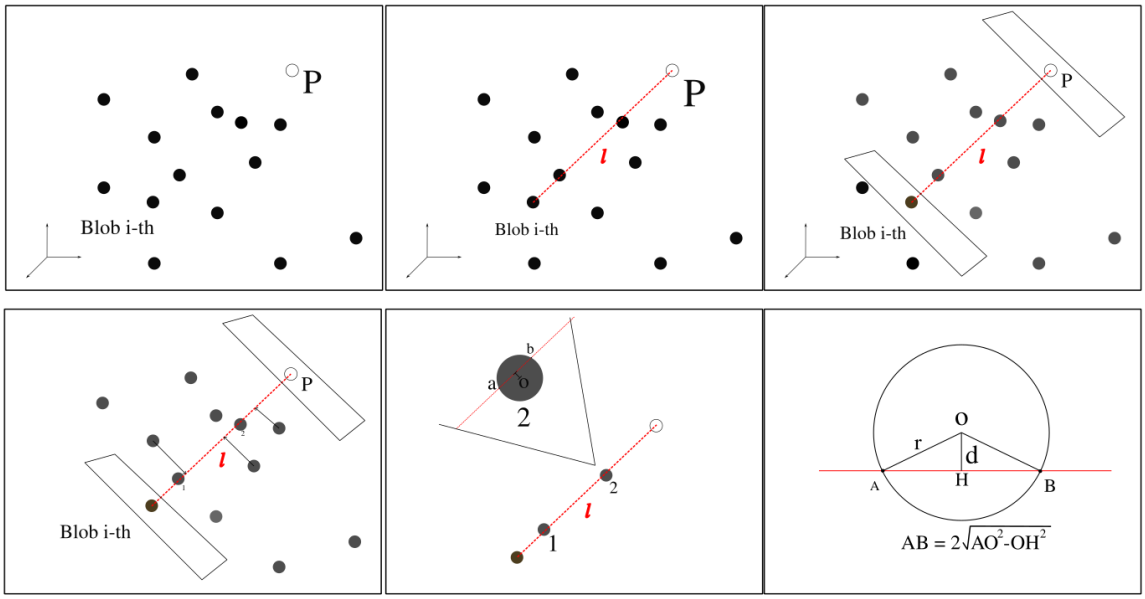

Figure 2.3: The different steps of the Raytracing algorithm. Given a set of points, it draws a line l going from P to the i-th blob. It then defines two planes containing P and the blob, both $\perp$ l. It then select only those blobs j between the two plane and computes the distance d to l. If the distance is lower or equal to the respective blob radius, i.e. $d <= r_j$ then it computes the segment $AB = l \cap j$

6. If the distance $d(C_j, i) \leq r_j$ where r_j is the radius of the sphere j, then l intersects with the sphere j

2.3.3 Mathematical solution

Here is shown the mathematical approach to the algorithm. For formatting purposes, C_i is indicated with C, all other block centers C_j with $j \neq i$ are indicated with K:

1. The direction vector $\vec{l}$ of the line going from P to C is:

$$\vec{l} = (C_x - P_x, C_y - P_y, C_z - P_z) \tag{2.5}$$

2. Given the plane $\hat{A_C}$ passing through C and $\perp \vec{l}$ we want to see in which side of the space P resides. This translates to compute the distance point-plane $d(P, \hat{A})$. Since we are not interested in the whole number but only on the sign of the distance, we do the following:

$$\vec{r} = (C_x - P_x, C_y - P_y, C_z - P_z) \tag{2.6}$$

   The distance should be computed as:

$$d_1 = d(P, \hat{A}_C) = \frac{\vec{l} \cdot \vec{r}}{\left\|\vec{l}\right\|} \tag{2.7}$$

   but since we need only the sign of d_1 then:

$$d_C^{\text{sign}}(P) = \frac{\vec{l} \cdot \vec{r}}{d_1} \tag{2.8}$$

   i.e. the sign of the distance of P with respect to the plane which passes through C.

3. We iterate 2. for each other block K. This is done both for the plane passing through C $\hat{A}_C$ and through $\hat{A}_P$. Therefore, we compute $d_C^{\text{sign}}(K)$ and $d_P^{\text{sign}}(K)$.

4. We consider only those points K positioned between the two planes. This means we consider the set X:

$$X = \{ \forall\ K : d_C^{\text{sign}}(K) = d_C^{\text{sign}}(P) \wedge d_P^{\text{sign}}(K) = -d_C^{\text{sign}}(P)\} \tag{2.9}$$

5. For each point in the set X we consider the distance point-line $d(K, l)$ (line as defined by passing through two points). This is done in several step. First we compute the direction vector $\vec{D}$ of the line same as (2.5):

$$\vec{D} = C - P \tag{2.10}$$

Then the vector from C to K, same as above:

$$\vec{E} = C - K \tag{2.11}$$

We then compute the projection of $\vec{E}$ on $\vec{D}$:

$$\vec{proj} = \frac{\vec{E} \cdot \vec{D}}{\vec{D} \cdot \vec{D}} D \tag{2.12}$$

Finally, it returns the distance between $\vec{E}$ and $\vec{proj}$:

$$d(K, l) = \sqrt{(E_x - proj_x)^2 + (E_y - proj_y)^2 + (E_z - proj_z)^2} \tag{2.13}$$

2.3.4 Code solution

The code can be divided in two structure: `Python` and in `C` . `Python` feeds the input to the `C` code and handles the output. `C` do the effective computation. This solution effectively reduces the computational times needed. The various inputs and outputs are in `txt` format and are stored in `/lib/tmp_files`.

After the computation, `Python` updates each block with the respective distance computed, i.e. the source of the Jet or the observer.

2.4 Maximum Electrons energy

Each input file represents the simulation at an instant t_i and it is separated by the next one at instant t_{i+1} by an interval Δt_i. This value is automatically retrieved by the code by looking at `'real parameters units'` in the input file. `CODE NAME` computes the emission at $t_i + \Delta t_i$, i.e. an instant before the next time-step in the original simulation code. Therefore it is necessary to compute the maximum energy reached by electrons. The code supposes that the main energy feeding mechanism to particles is Second Order Fermi Acceleration. [4] shows that the maximum energy achievable by electrons via this mechanism scales as:

$$QB_* f_\Delta x_d P_0 \cong 7.7 \times 10^{20} Z \epsilon_B^{1/2} (\mu_0 n_0)^{1/6} f_\Delta \beta_0 (\mathcal{E}_\odot \Gamma_0)^{1/3} \text{ eV} \tag{2.14}$$

where:

- Q is the total charge;
- B_* is the magnetic field;
- f_Δ is the hydrodynamic width of the blast wave;
- x_d is the deceleration radius;
- P_0 is the adimensional fluid motion $P_0 = \beta_0 \Gamma_0 = \sqrt{\Gamma_0^2 - 1}$;
- Z is the atomic number;
- $\epsilon_B = B^2/(8\pi)$ is the fraction of total energy of the magnetic field;
- $\mathcal{E}_\odot$ is the apparent isotropic energy release of the explosion expressed in units of Solar rest-mass energy;
- μ_0 is the average atomic mass of swept-up material in terms of proton mass m_p;
- n_0 is the number density of the material.

The main task is to assess which `Block` still emits at $t_i + \Delta t_i$, this means to have the condition $E_f(t_i + \Delta t_i) \neq 0$. If such block would reach a null final energy it gets flagged and don't take part into the emission computation process. In order to do the code compute the time-scale of energy loss t_{cool} per process and compares it with Δt_i. If greater it means that in that conditions the emitting process is extremely efficient and the block would dissipate all of their energy by emission before reaching the observation instant.

t_{cool} can be computed supposing the final energy to be null $E_f = 0$, the initial energy to be the maximum achievable energy via Second Order Fermi Acceleration, therefore using (2.14): $E_i = E_{\max}$ and results in:

$$\frac{E_f - E_i}{t_{\text{cool}}} = \frac{\mathrm{d}E}{\mathrm{d}t} \rightarrow t_{\text{cool}} = \frac{E_{\max}}{-\left(\frac{\mathrm{d}E}{\mathrm{d}t}\right)} \tag{2.15}$$

Each emission process has a different rate of energy loss $-\mathrm{d}E/\mathrm{d}t$.

[2] shows that the non-thermal Bremsstrahlung energy loss can be computed by:

$$-\left(\frac{\mathrm{d}E}{\mathrm{d}t}\right) = 4\alpha r_0 c^2 \sum_Z n_Z Z(Z+1)\left(\ln 2E_i - \frac{1}{3}\right)E_i \tag{2.16}$$

where α is the fine-structure constant, r_0 the electron radius, c the speed of light, Z is the atomic number and n_Z is the number density per species, E_i is the initial energy in $m_e c^2$ units.

The Synchrotron rate energy loss for an isotropic distribution as shown by [10]) is:

$$-\left(\frac{\mathrm{d}E}{\mathrm{d}t}\right) = \frac{4}{3}\sigma_T c U_{\mathrm{mag}} \gamma^2 \left(\frac{v}{c}\right)^2 \tag{2.17}$$

where σ_T is the coefficient of Thomson scattering, c is the speed of light, U_{mag} is the energy density of the magnetic field. In this case γ is the energy of the electrons in eV, hence in order to get the time scale of loss energy in (2.15) we divide by a $m_e c^2$ factor. We suppose that $v \sim c$.

The External Compton emission as shown in [10] is:

$$-\frac{\mathrm{d}E}{\mathrm{d}t} = \frac{4}{3}\sigma_T c U_{\mathrm{rad}} \left(\frac{v}{c}\right)^2 \gamma^2 \tag{2.18}$$

Same as above with U_{rad} being the energy density of the radiation field and γ the energy of the electrons in eV. We suppose that $v \sim c$. In case of the Cosmic Background Radiation the code uses equation (2.35). In case of a Shakura-Sunyaev accretion the equation is (2.42).

$$t_{\mathrm{cool,brem}} = \left\{4\alpha r_0^2 c \left[\sum_Z n_z Z(Z+1)(\ln(2E_{\max}) - 1/3\right]\right\}^{-1} \tag{2.19}$$

$$t_{\mathrm{cool,syn}} = \left(\frac{m_e c}{\sigma_T}\right)\left(\frac{B^2}{\mu_0}\right)^{-1} \gamma_0^{-1} \tag{2.20}$$

$$t_{\mathrm{cool,\ IC}} = \frac{3}{4}\left(\frac{m_e c}{\sigma_T}\right) U_{\mathrm{rad}}^{-1} \gamma_0^{-1} \tag{2.21}$$

2.5 System and blob geometry

Here we discuss the geometry of the system considered in the calculations. For more information check [11], the developers of `agnpy` [12].

In 2.5.1 the main computations done for the blob geometry are shown. More information can be found in [4].

2.5.1 Blob Geometry

Quantities in the stationary frame at rest in the Hubble flow are starred (K^*), comoving quantities are primed (K'), observed quantities are unscripted (K).

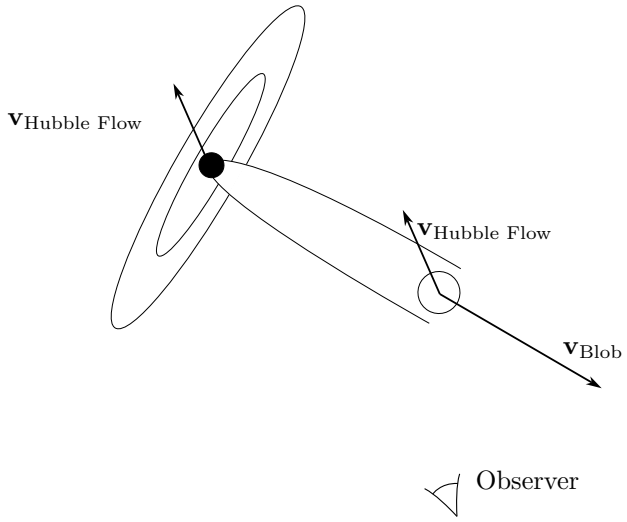

Figure 2.4: In the Laboratory frame, a distant object appears to be receding due to the expansion of the Universe. This effect, known as Hubble flow, results in a redshift z.

Fluxes of relativistic cosmological sources

The energy flux Φ of an isotropically emitting source in the stationary frame, with a luminosity $L_{*,\text{iso}}\left[\text{ergs s}^{-1}\right]$ is:

$$\Phi = \frac{L_{*,\text{iso}}}{4\pi d_L^2} \tag{2.22}$$

where d_L is the source distance from the observer. For an emitting source having luminosity $L_*(\epsilon_*, \Omega_*)$, the emitting flux would be:

$$F(\epsilon)d\epsilon = \frac{L_*(\epsilon_*, \Omega_*)}{d_L^2}\,\mathrm{d}\epsilon \tag{2.23}$$

where ϵ is the photon energy.
The photon energy in frame K is related to the photon energy in frame K' through the relation:

$$\frac{\epsilon}{\epsilon'} = \frac{\delta_D}{1+z} \tag{2.24}$$

where $\delta_D \equiv [\Gamma(1-\beta\mu)]^{-1}$ is the Doppler factor and z is the redshift of the source, see Figure 2.4. The energies in the three frames are related accordingly as:

$$\epsilon = \frac{\epsilon_*}{(1+z)} = \frac{\delta_D \epsilon'}{(1+z)} \tag{2.25}$$

Here $\epsilon, \epsilon_*, \epsilon'$ are the energy densities in the observer's, central source and blob's frame, respectively. For a homogeneous, isotropic universe, the direction vector

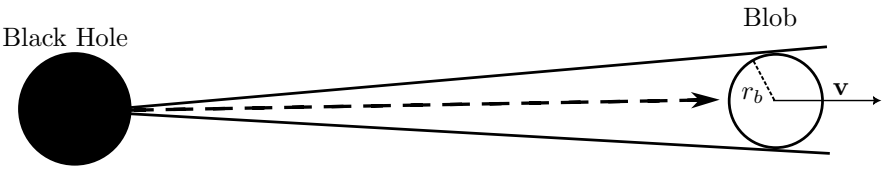

Figure 2.5: Jets emitted by black holes can be seen as composed by blobs, moving away from the source with a given velocity **v**.

of the observer frame is equal to the one in the cosmological frame, thus: $\Omega_* = \Omega$. Hence:

$$F(\epsilon;\Omega) = \frac{(1+z)}{d_L^2} L_*(\epsilon_*,\Omega) \tag{2.26}$$

The differential directional luminosity is:

$$\mathrm{d}L_*\left(\epsilon_*,\Omega_*\right) = \mathrm{d}V_* j_*(\epsilon_*,\Omega_*) \tag{2.27}$$

where $\mathrm{d}V_*$ is a differential volume element, $j(\epsilon,\Omega)$ is the directional emissivity which divided by ϵ is a Lorentz invariant. Thus:

$$j_*(\epsilon_*,\Omega_*) = \left(\frac{\epsilon_*}{\epsilon'}\right)^2 j'(\epsilon',\Omega') \tag{2.28}$$

and

$$\mathrm{d}[\epsilon F(\epsilon;\Omega,t)] = \mathrm{d}f_\epsilon(t) = \frac{\delta_D^4}{d_L^2}\epsilon' j'(\epsilon',\Omega';\mathbf{r}',t')\mathrm{d}V' \tag{2.29}$$

where the reception time t and emission time t' have to be properly related. Finally f_ϵ is the energy flux νF_ν $\left[\text{ergs cm}^{-2}\ \text{s}^{-1}\right]$ and is defined as:

$$f_\epsilon = \epsilon F(\epsilon) = \epsilon \int_{\Omega_s} \mathrm{d}\Omega \mu I_\epsilon \tag{2.30}$$

Blob Geometry

Figure 2.5 shows a blob moving away from the central Black Hole. It has a comoving radius r'_b and comoving volume $V'_b = 4\pi r'^3_b/3$. It is assumed that the timescale for variation in the radiation t'_{var} is longer than the light-crossing time $t'_{lc} \approx r'_b/c$ of the blob. In this way one can avoid performing temporal integrations over different portions of the blob.
This requirement translates into a constraint on the blob's size:

$$r'_b \underset{\sim}{<} ct'_{var} = c\delta_D \frac{\Delta t_{\mathrm{var}}}{1+z} \tag{2.31}$$

where the relation $\Delta t = [t'_{\mathrm{var}}(1+z)]/\delta_D$ has been used. Assuming the blob moving with constant speed, using (2.29) one gets:

$$f_\epsilon(t) \cong \frac{\delta_D^4 V'_b}{d_L^2}\epsilon' j'(\epsilon'\Omega';t') \tag{2.32}$$

and assuming that the blob radiates isotropically in the comoving frame:

$$f_\epsilon(t) \cong \frac{\delta_D^4 V_b'}{4\pi d_L^2} \epsilon' j'(\epsilon';t') \cong \frac{\delta_D^4 \epsilon' L'(\epsilon';t')}{4\pi d_L^2} \tag{2.33}$$

where the comoving spectral luminosity (2.27) has been used.

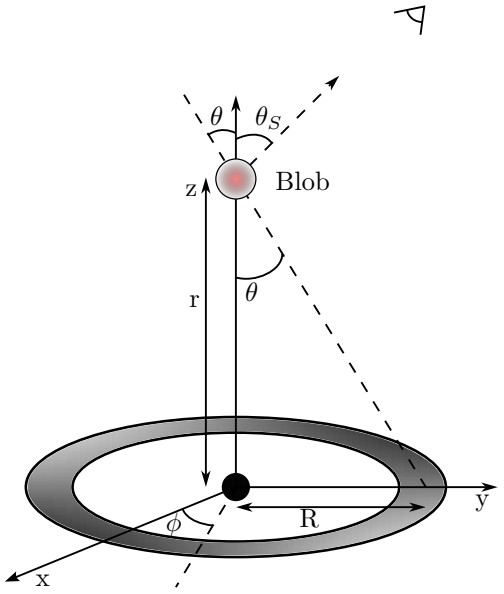

Figure 2.6: Geometry used for the energy density and external Compton scattering calculations in `agnpy`

2.6 Radiative Process

In the following paragraphs we will follow `agnpy` [12] structure. Hence we consider the blob moving along the z axis and the main angles θ and ϕ defined as in Figure 2.6.

Section 2.6.2 describes the calculations made in order to compute the opacity of each blob. Section 2.6.3 shows how the code computes the Non-Thermal Bremsstrahlung emission. Section 2.6.4 dives in the synchrotron emission, as well as showing the computations made for the Self-Synchrotron Absorption and for Self-Synchrotron Compton. Finally, section 2.6.5 shows the computation done for the Inverse Compton in the case of the Cosmic Background Radiation (CMB) and for the Shakura-Sunyaev disk.

2.6.1 Energy densities

The energy density of a photon field at a given distance r above the central Black Hole $u(r)[\text{ erg cm}^{-3}]$ is computed by integrating the specific spectral energy density[1] $\underline{u}(\epsilon, \Omega; r)$ over (dimensionless) energy and solid angle:

$$u(r) = \int \mathrm{d}\epsilon \int \mathrm{d}\Omega \; \underline{u}(\epsilon, \Omega; r) \tag{2.34}$$

[1] I.e. the energy density per unit volume.

In case of an isotropic monochromatic radiation field with dimensionless energy ϵ_0, the specific spectral energy density is:

$$\underline{u}(\epsilon, \Omega) = \frac{u_0 \delta(\epsilon - \epsilon_0)}{4\pi} \tag{2.35}$$

such that the energy density in the galactic frame is trivially $u = u_0$, while in the blob frame:

$$u' = u_0 \Gamma \left(1 + \frac{\mathcal{B}^2}{3}\right) \tag{2.36}$$

where $\mathcal{B}$ is the relativistic dimensionless velocity of the blob in the galactic (laboratory) frame.

As shown in figure 2.6, in case of an accretion disc with inner and outer radii $R_{\rm in}$ and $R_{\rm out}$, the cosine of the photon incoming angle $\mu = \cos\theta$ and the coordinate along the disc radius R are related via:

$$\mu = \left(1 + \frac{R^2}{r^2}\right)^{-1/2} \text{ and } R = r\sqrt{\mu^{-2} - 1} \tag{2.37}$$

Using Dermer & Schlickeiser (2002) and Dermer (2009) [5][4] the specific spectral energy density of photons produced by an accretion disc can be parameterized as:

$$\underline{u}(\epsilon, \Omega; r) = \frac{3GM\dot{m}}{(4\pi)^2 c R^3} \phi(R) \delta(\epsilon - \epsilon_0(R)) \tag{2.38}$$

where $\phi(R)$ represents the variation of energy flux along the radius

$$\phi(R) = 1 - \sqrt{\frac{R_{\rm in}}{R}} \tag{2.39}$$

and $\epsilon_0(R)$ is the monochromatic approximation for the mean photon energy emitted from the disc at radius R:

$$\epsilon_0(R) = 2.7 \times 10^{-4} \left(\frac{L_{\rm Edd}}{M_8 \eta}\right)^{1/4} \left(\frac{R}{R_g}\right)^{-3/4} \tag{2.40}$$

where

- $G = 6.67259 \times 10^{-8}$ is the gravitational constant (in cgs units);
- $M = M_8 \times 10^8 M_\odot$ is the mass of the black hole in solar mass unit;
- $\dot{m}$ [g s^{-1}] is the black hole mass accretion rate;
- η the fraction of gravitational energy converted to radiant energy;
- $l_{\rm Edd}$ is the ratio of the disc luminosity to the Eddington luminosity;
- $R_g = 2GM/c^2$ is the gravitational radius of the black hole.

Figure 2.7: An "head-on" and "tail-on" collision.

When using the equations (2.39) and (2.40), one implicitly takes into account the different incidence angles of the photons emitted at a given radius for different angles (see eqs. (2.37) and being scattered by the blob at position r, hence:

$$\varphi(R) \to \varphi(\mu; r) \text{ and } \epsilon_0(R) \to \epsilon_0(\mu; r) \tag{2.41}$$

Therefore, the integral energy density in the galactic frame is:

$$u(r) = \frac{3}{8\pi c}\frac{GM\dot{m}}{r^3}\int_{\mu_{\min}}^{\mu_{\max}} \mathrm{d}\mu \frac{\varphi(\mu; r)}{(\mu^{-2}-1)^{3/2}} \tag{2.42}$$

while in the blob frame the energy density is:

$$u'(r) = \frac{3}{8\pi c}\frac{GM\dot{m}}{r^3}\int_{\mu_{\min}}^{\mu_{\max}} \mathrm{d}\mu \frac{\varphi(\mu; r)}{\Gamma^6(1-\mathcal{B}\mu)^2(1+\mathcal{B}\mu')^4(\mu^{-2}-1)^{-3/2}} \tag{2.43}$$

2.6.2 Opacity

Every medium encountered along the path of a photon flux interacts with the photons, leading to a reduction in the overall flux. This decrease in flux is determined by both the opacity of the medium and the cross section of interaction between the photons and the constituent particles of the medium. The opacity can be expressed as follows:

$$k_\nu = \frac{\sigma_c n_0}{\rho} \tag{2.44}$$

where ρ and n_0 are, respectively, the density and the number density of the gas. Finke (2016) [6] provides a way to compute the cross section: in the "head-on" approximation (see figure 2.7), the photons are scattered in the same direction as the incident electrons, thus:

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\epsilon_s \mathrm{d}\Omega_s} \approx \frac{\mathrm{d}\sigma}{\mathrm{d}\epsilon_s}\delta(\Omega_s - \Omega_e) \tag{2.45}$$

where ϵ_S is the scattered photon energy, Ω_s it is the direction that the delta ensured to be equal to the electron direction Ω_e.

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\epsilon_s} = \int \mathrm{d}\Omega_s \frac{\mathrm{d}\sigma}{\mathrm{d}\epsilon_s \mathrm{d}\Omega_s} = \frac{3\sigma_T}{8\gamma\bar{\epsilon}}\Xi(\epsilon, y) H\left(\epsilon_s; \frac{\bar{\epsilon}}{2\gamma}, \frac{2\gamma\bar{\epsilon}}{1+2\bar{\epsilon}}\right) \tag{2.46}$$

with

$$\Xi(\bar{\epsilon}, y) = 1 - y + \frac{1}{1-y} - \frac{2y}{\bar{\epsilon}(1-y)} + \left[\frac{y}{\bar{\epsilon}(1-y)}\right]^2 \tag{2.47}$$

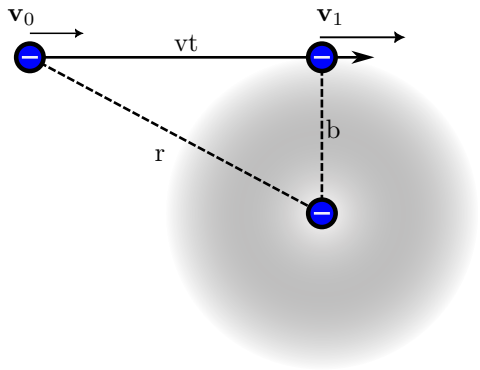

Figure 2.8: Example of moving charged particle interacting with the electromagnetic field produced by another charge. In this case the interaction results in an acceleration.

and

$$\bar{\epsilon} = \gamma\epsilon(1 - \cos\psi) \tag{2.48}$$

$$y = \frac{\epsilon_s}{\gamma} \tag{2.49}$$

Let ϕ and $\theta = \arccos(\mu)$ be, respectively, the azimuthal and polar angles of the incident photon (figure 2.6) and ϕ_s, $\theta_s = \arccos(\mu_s)$ the angles of the scattered photons, then:

$$\cos\psi = \mu\mu_s + \sqrt{1-\mu^2}\sqrt{1-\mu_s^2}\cos(\phi - \phi_s) \tag{2.50}$$

The total Compton-scattering cross-section is:

$$\sigma(\bar{\epsilon}) = \int \mathrm{d}\epsilon_s \frac{\mathrm{d}\sigma}{\mathrm{d}\epsilon_s} = \sigma_T S_0(\bar{\epsilon}) \tag{2.51}$$

where:

$$S_0(x) = \frac{3}{8x^2}\left[4 + \frac{2x^2(1+x}{(1+2x)^2} + \frac{x^2 - 2x - x}{x}\ln(1+2x)\right] \tag{2.52}$$

with x being the energy of the photon in $m_e c^2$ units.

2.6.3 Bremsstrahlung

When a moving charged particle interacts with the potential field of the nearby particles, it gets accelerated resulting in emissions of photons. This process is called Bremsstrahlung or free-free emission and it is responsible of the emission in the X-ray.

In this section it is described how to compute the bremsstrahlung emission produced by a non-thermal population. Source [2].

In this case the emission comes from a non-thermal distribution of electrons the total bremsstrahlung spectrum is found from:

$$\frac{\mathrm{d}N_{\mathrm{tot}}}{\mathrm{d}t\,\mathrm{d}k} = \int \mathrm{d}E_i\, N_e(E_i)\left(\frac{\mathrm{d}N_i}{\mathrm{d}t\,\mathrm{d}k}\right) \tag{2.53}$$

where:

$$N_e(E_i) = K_e E_i^{-p} H(E_i; E_0, E_m) \tag{2.54}$$

with $H(E_i; E_0, E_m)$ is the Heaviside function and K_e is the normalization constant:

$$K_e = \frac{\rho}{\left(\int_{E_0}^{E_m} N(E_i)\,\mathrm{d}E_i\right)\mu m_H} \tag{2.55}$$

with:

$$\int_{E_0}^{E_m} N(E_i)\,\mathrm{d}E_i = \frac{E_m^{-p+1} - E_0^{-p+1}}{-p+1} \tag{2.56}$$

Supposing that $k \ll E_m$ in order to simplify the computation, the lower limit is:

$$E_L = \max(k, E_0) \tag{2.57}$$

Using (2.58)

$$\frac{\mathrm{d}N_i}{\mathrm{d}t\,\mathrm{d}k} = c\sum_s n_s\left(\frac{\mathrm{d}\sigma_s}{\mathrm{d}k}\right) \tag{2.58}$$

and (2.59)

$$\mathrm{d}\sigma = \Phi(k)\,\mathrm{d}k = \alpha r_0^2\left(\frac{\mathrm{d}k}{k}\right)\left(E_i^2\right)^{-1}\left[\left(E_i^2 + E_f^2\right)\phi_1 - \frac{2}{3}E_i E_f \phi_2\right] \tag{2.59}$$

one obtains:

$$\begin{aligned}\frac{\mathrm{d}N_{\mathrm{tot}}}{\mathrm{d}t\,\mathrm{d}k} =& \alpha r_0 K_e k^{-1}\sum_s n_s \times \\ &\int_{E_L} \mathrm{d}E_i\, E_i^{-(p+2)}\left[\left(2E_i^2 - 2E_i k + k^2\right)\phi_{1(s)} - \frac{2}{3}E_i(E_i - k)\phi_{2(s)}\right]\end{aligned} \tag{2.60}$$

Referring to the whole integral as simply $I_s(k, E_L; p)$, in case of weak shielding, complete ionization, ϕ_1 and ϕ_2 are well represented by the value:

$$\phi_{\mathrm{weak}} = 4\left(Z^2 + Z_{\mathrm{el}}\right)\left\{\ln\left[2E_i\left(\frac{E_i - k}{k}\right)\right] - \frac{1}{2}\right\} \tag{2.61}$$

thus reducing I_s to:

$$I_s(\text{weak shielding}) \approx I'\bar{\phi}_{\mathrm{weak}} \tag{2.62}$$

where

$$\begin{aligned} I' &= \int_{E_L} \mathrm{d}E_i\, E_i^{-(p+2)}\left(\frac{4}{3}E_i^2 - \frac{4}{3}E_i k + k^2\right) \\ &= \frac{4}{3}\frac{E_L^{-(p-1)}}{p-1} - \frac{4}{3}\frac{kE_L^{-p}}{p} + \frac{k^2 E_L^{-(p+1)}}{p+1} \end{aligned} \tag{2.63}$$

For $E_L = E_0$ one can set $E_i = E_0$, on the other hand, if $E_L = E_0$ or $k \approx E_0$ one can set $E_i \approx 2k$ in (2.61), thus obtaining:

$$\bar{\phi}_{\rm weak} = \begin{cases} 4(Z^2 + Z_{el})\{\ln\left[2E_0(E_0 - k)/k\right] - 1/2\} \text{ for } k < E_0 \\ 4(Z^2 + Z_{el})[\ln(4k) - 1/2] \text{ for } k > E_0 \end{cases} \tag{2.64}$$

2.6.4 Synchrotron

Relativistic moving charge particles, in presence of a magnetic field, are accelerated resulting in emission of photons in the X-ray. This process is known as synchrotron emission.

Synchrotron emission from isotropic electrons

A complete calculation of the synchrotron emissivity from a population of isotropic electrons in a randomly oriented magnetic field has been presented by Crusius & Schlickeiser (1986) [3]

$$\epsilon' J'_{\rm syn}(\epsilon') = \frac{\sqrt{3}\epsilon' e^3 B}{h} \int_1^\infty {\rm d}\gamma' N'_e(\gamma') R(x) \tag{2.65}$$

where for a homogeneous distribution

$$\epsilon' J'_{\rm syn}(\epsilon') = V' b \epsilon' j'_{\rm syn}(\epsilon'_s) \tag{2.66}$$

e is the fundamental charge, h is Planck's constant,

$$x = \frac{4\pi\epsilon' m_e^2 c^3}{3eBh\gamma'^2} \tag{2.67}$$

and

$$R(x) = \frac{x}{2} \int_0^\pi {\rm d}\theta \sin\theta \int_{x/\sin\theta}^\infty {\rm d}t K_{5/3}(t) \tag{2.68}$$

where $K_{5/3}(t)$ is the modified Bessel function of the second kind of order 5/3. It represents the pitch-angle-averaged synchrotron spectral power of a single electron. Substituting in (2.33):

$$\hat{f}^{\rm synch}_{\hat{\epsilon}} = \frac{\delta_D^4}{4\pi d_L^2} \epsilon' \frac{\sqrt{3}e^3 B}{h} \int_1^\infty {\rm d}\gamma' \underline{N}'_e(\gamma)' R(x) \tag{2.69}$$

$\delta-$function approximation for Synchrotron Radiation In problems involving isotropic electron pitch-angle distributions in a randomly oriented magnetic field, a simple δ-function approximation can be employed to estimate the synchrotron radiation.

In terms of photon emissivity $\dot{n}^{\rm syn}(\epsilon)$ [ph cm^{-3}s$^{-1}\epsilon^{-1}$], the $\delta-$function approximation for synchrotron emission is given by:

$$\dot{n}^{\rm syn}(\epsilon) \cong \frac{2}{3} c\sigma_T u_B \epsilon^{-1/2} \varepsilon_B^{-3/2} n_e \left(\sqrt{\frac{\epsilon}{\varepsilon_B}}\right) \tag{2.70}$$

where:

$$u_B = \frac{U_B}{m_e c^2} = \frac{B^2}{8\pi m_e c^2} \tag{2.71}$$

For a uniform-emitting region of volume V_b, the $\epsilon L(\epsilon)$ spectrum is given by:

$$\epsilon L_\delta^{\rm syn}(\epsilon) = m_e c^2 V_b \epsilon^2 \dot{n}^{\rm syn}(\epsilon) \cong \frac{2}{3} c\sigma_T U_b \gamma_s^3 N_e(\gamma_s) \tag{2.72}$$

where (2.70) has been used and $\gamma_s = \sqrt{\epsilon/\varepsilon_B}$.

From (2.33) and (2.72), the νF_ν synchrotron radiation spectrum can be approximated by the expression:

$$f_\epsilon^{\rm syn} \cong \frac{\delta_D^4}{6\pi d_L^2} c\sigma_T U_b \gamma_s'^3 N_e'(\gamma_s') \tag{2.73}$$

where

$$\gamma_s' = \sqrt{\frac{\epsilon(1+z)}{\delta_D \varepsilon_B}} \tag{2.74}$$

SSC emissivity In order to compute the SSC emissivity, it is necessary to determine the comoving electron distribution $N_e'(\gamma_s')$ and the comoving radiation energy density $u'(\epsilon')$. The comoving electron distribution can be found by explicitly stating it from (2.73):

$$N_e'(\gamma_s') = V_b' n_e'(\gamma_s') \cong \frac{6\pi d_L^2 f_\epsilon^{\rm syn}}{c\sigma_T U_b \delta_D^4 \gamma' 3_s} \tag{2.75}$$

The synchrotron photon number density is then determined using (2.75) and (2.70)

$$n_{\rm syn}'(\epsilon') \cong \frac{R_b'}{c} \dot{n}_{\rm syn}'(\epsilon') \cong \frac{3 d_L^2 f_\epsilon^{\rm syn}}{m_e c^3 R_b' \delta_D^4 \epsilon_B^2 \gamma_s'^4} \tag{2.76}$$

This can be converted to a radiation energy density:

$$u'(\epsilon') = \epsilon' m_e c^2 n_{\rm syn}'(\epsilon') \cong \frac{3}{4} \frac{3 d_L^2 \hat{f}_\epsilon^{\rm syn}}{c R_b'^2 \delta_D^4 \epsilon'} \tag{2.77}$$

where the term 3/4 is due to averaging the radiation in a sphere.

The SSC emissivity, integrated over volume, for isotropic and homogeneous photon and distribution is then given using ((2.75) and (2.77):

$$\epsilon_s' J_{\rm SSC}' = \frac{3}{4} c\sigma_T \epsilon_S'^2 \int_0^\infty {\rm d}\epsilon' \frac{u'(\epsilon')}{\epsilon'^2} \int_{\gamma_{\rm min}'}^{\gamma_{\rm max}'} {\rm d}\gamma' \frac{N_e'(\gamma')}{\gamma'^2} F_C(q, \Gamma_e) \tag{2.78}$$

where for an homogeneous distribution $\epsilon_s' J_{SSC}'(\epsilon_s') = V_b' \epsilon_s' j_{\rm SSC}'(\epsilon_s')$. F_C is an integration kernel and it represents the Compton cross section for electrons and

photons with a uniform spatial distribution (Jones 1968, Blumenthal & Gould 1970) [8][2]

$$F_C(q,\Gamma_e)=\left[2q'\ln q'+(1+2q)(1-q')+\frac{1}{2}\frac{(\Gamma'_e q')^2}{(1+\Gamma'_e q')}(1-q')\right]H\left(q';\frac{1}{4\gamma'^2},1\right) \tag{2.79}$$

with

$$q'\equiv\frac{\epsilon'_s/\gamma'}{\Gamma'_e(1-\epsilon'_s/\gamma')}\text{ and }\Gamma'_e=4\epsilon'_s\gamma' \tag{2.80}$$

The Heaviside functions on q' represents the kinematic limits on ϵ' and γ'.

Substituting (2.78) in (2.33) one obtains:

$$\hat{f}^{\rm SSC}_{\hat{\epsilon}_s}=\frac{\delta_D^4}{4\pi d_L^2}\frac{3}{4}c\sigma_T\epsilon'^2_s\int_0^\infty {\rm d}\epsilon'\frac{u'_{\rm synch}(\epsilon')}{\epsilon'^2}\int_0^\infty {\rm d}\gamma\frac{N'_e(\gamma')}{\gamma'^2}F_c(q',\Gamma'_e) \tag{2.81}$$

2.6.5 Inverse Compton

Here we focus on how to compute the energy flux for a generic radiation field and the case in which it is produced by the CMB or by the disc.

Inverse Compton energy flux

Assuming to be in a "head-on" approximation, i.e. $\gamma'\gg 1$, thanks to the relativistic transformations:

$$\sin\theta'=\frac{\sin\theta}{\gamma[1+(v/c)\cos\theta]}\text{ and }\cos\theta'=\frac{\cos\theta+v/c}{1+(v/c)\cos\theta} \tag{2.82}$$

we see that the photon is scattered back from its initial propagation direction. Under this condition the cross section simplifies:

$$\frac{{\rm d}\sigma_C}{{\rm d}\epsilon_s{\rm d}\Omega_S}\approx\frac{{\rm d}\sigma_C}{{\rm d}\epsilon_s}\delta(\Omega_s-\Omega_e) \tag{2.83}$$

where ϵ_S is the scattered photon energy, Ω_s is the scattered photon's solid angle, and the Dirac delta forces it to coincide with the electron direction Ω_e. To transform the electron and photon densities in the same reference frame, the approach proposed by Georganopoulos (2001) [7]is used.

Consider the figure 2.6. For an isotropic electron distribution the equation is:

$$N_e(\gamma,\Omega_e)=\delta_D^3 N'_e\left(\frac{\gamma}{\delta_D}\right) \tag{2.84}$$

Therefore, the energy flux produced by the blob electrons Compton scattering an external radiation field reads:

$$\hat{f}^{\rm EC}_{\hat{\epsilon}_s}=\frac{c\pi r_e^2}{4\pi d_L^2}\epsilon_s^2\delta_D^3\int{\rm d}\Omega\int_0^{\epsilon_{\rm high}}{\rm d}\epsilon\,\frac{u(\epsilon,\Omega;r)}{\epsilon^2}\int_{\gamma_{\rm low}}^\infty{\rm d}\gamma\frac{N'_e(\gamma/\delta_D)}{\gamma^2}\Xi_C \tag{2.85}$$

where Ξ_C is the integration kernel, i.e.

$$\Xi_C = y + y^{-1} + \frac{2\epsilon_s}{\gamma\bar{\epsilon}y} + \left(\frac{\epsilon_s}{\gamma\bar{\epsilon}y}\right)^2 \tag{2.86}$$

This coincides with the one previously discussed, i.e. (2.47), although here the argument of the kernel is different:

$$y = 1 - \frac{\epsilon_s}{\gamma} \tag{2.87}$$

while $\bar{\epsilon}$ and $\cos\psi$ are the same as previously defined in (2.48) and (2.50), with ψ being the angle between the direction of the incident photon and the scattering electron.

The extrema of integration are:

$$\epsilon_{\rm high} = \frac{2\epsilon_s}{1-\cos\phi} \tag{2.88}$$

and

$$\gamma_{\rm low} = \frac{\epsilon_s}{2}\left[1 + \sqrt{1 + \frac{2}{\epsilon\epsilon_s(1-\cos\phi)}}\right] \tag{2.89}$$

CMB Cosmic Microwave Background radiation is an isotropic radiation field, often represented by an isotropic monochromatic external radiation field, whose photons originate from the last-scattering surface. They are subject to inverse Compton scattering when interacting with the electrons of the blob. In this case using (2.35) in (2.85) one gets:

$$\hat{f}^{\rm EC}_{\hat{\epsilon}_s} = \frac{3}{2^7\pi^2}\frac{c\sigma_T u_0}{d_L^2}\left(\frac{\epsilon_s}{\epsilon_0}\right)^2 \delta_D^3 \int_0^{2\pi} {\rm d}\phi \int_{-1}^{1} {\rm d}\mu \int_{\gamma_{\rm low}}^{\infty} {\rm d}\gamma \frac{N'_e(\gamma/\delta_D)}{\gamma^2}\Xi_C \tag{2.90}$$

Shakura-Sunyaev accretion disc In the case the source is a Shakura-Sunyaev accretion disc then the flux is computed using (2.38) in (2.85)

$$\begin{aligned}\hat{f}^{\rm EC}_{\hat{\epsilon}_s}(r) = {} & \frac{3}{2^9\pi^3}\frac{\sigma_T G M\dot{m}}{d_L^2 r^3}\epsilon_s^2\delta_D^3 \times \\ & \int_0^{2\pi} {\rm d}\phi \int_{\mu_{\rm min}}^{\mu_{\rm max}} {\rm d}\mu \frac{\varphi(\mu;r)}{\epsilon_0^2(\mu;r)(\mu^{-2}-1)^{3/2}} \int_{\gamma_{\rm low}}^{\infty} {\rm d}\gamma \frac{N'_e(\gamma/\delta_D)}{\gamma^2}\Xi_C\end{aligned} \tag{2.91}$$

Bibliography

[1] C. Bradford Barber and Hannu Huhdanpaa. *qhull*. URL: `http://www.qhull.org/`. (accessed: 11.03.24).

[2] George R Blumenthal and Robert J Gould. "Bremsstrahlung, synchrotron radiation, and compton scattering of high-energy electrons traversing dilute gases". In: *Reviews of modern Physics* 42.2 (1970), p. 237.

[3] A Crusius and R Schlickeiser. "Synchrotron radiation in random magnetic fields". In: *Astronomy and Astrophysics* 164 (1986), pp. L16–L18.

[4] Charles D Dermer and Govind Menon. *High energy radiation from black holes: gamma rays, cosmic rays, and neutrinos*. Vol. 17. Princeton University Press, 2009.

[5] Charles D Dermer and Reinhard Schlickeiser. "Transformation properties of external radiation fields, energy-loss rates and scattered spectra, and a model for blazar variability". In: *The astrophysical journal* 575.2 (2002), p. 667.

[6] Justin D Finke. "External compton scattering in blazar jets and the location of the gamma-ray emitting region". In: *The Astrophysical Journal* 830.2 (2016), p. 94.

[7] M Georganopoulos, JG Kirk, and A Mastichiadis. "The beaming pattern and spectrum of radiation from inverse Compton scattering in blazars". In: *The Astrophysical Journal* 561.1 (2001), p. 111.

[8] Frank C Jones. "Calculated spectrum of inverse-Compton-scattered photons". In: *Physical Review* 167.5 (1968), p. 1159.

[9] A. R. King and J. E. Pringle. "Can the Blandford–Znajek Mechanism Power Steady Jets?" In: *The Astrophysical Journal Letters* 918.1 (Sept. 2021), p. L22. DOI: `10.3847/2041-8213/ac19a1`. URL: `https://dx.doi.org/10.3847/2041-8213/ac19a1`.

[10] Malcolm S Longair. *High energy astrophysics*. Cambridge university press, 2010.

[11] C Nigro et al. "agnpy: An open-source python package modelling the radiative processes of jetted active galactic nuclei". In: *Astronomy & Astrophysics* 660 (2022), A18.

[12] Cosimo Nigro et al. *agnpy (0.4.0)*. 2023. DOI: 10.5281/zenodo.7633553. URL: https://doi.org/10.5281/zenodo.7633553.