CODE NAME [1] User's Manual

Marco Sortino

March 6, 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).

0.0.1 Changelog

On this date (December 08, 2023) the code has been modified:

- Fixed the previous issues on External Compton, CMB ray tracing and Disc energy loss.

- Added a convex hull algorithm to properly track the blobs subjected to CMB and External Compton

On this date (March 06, 2024) the code has been modified:

- Added `plotting.py`.

- Added RayTracing algorithm in C.

- Overhauled the general structure and collocation of libraries.

Issues

On this date (October 20, 2023), the current known issues are:

- External Compton and Synchrotron Self - Compton
  Blobs and photon interaction is not correctly tracked, resulting the source to appear more bright in the high frequencies range. —- FIXED

- Ray tracing
  The current algorithm is too slow. —- FIXED

- Disc energy loss
  Currently deactivated as it brings issues with the computation —- FIXED

- CMB ray tracing
  Disabled as there are some issues on how to correctly take the opacity into account. —- FIXED

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:

```
1    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

```
1    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.

```
1  # ***** File paths:
2  file_path = /home/user/path/to/file.hdf5
3  file_saving_path = ''
4  # ***** if you want to launch multiple session put id_launch = y
5  id_launch = n
6  file_list_path = ''
7  # ***** Indexing settings:
8  pres_Min = 1e-100 cm-1 g s-2
9  pres_Max = 1e100 cm-1 g s-2
10 dens_Min = 1e-100 g cm-3
11 dens_Max = 1e100 g cm-3
12 energy_Min = 1 erg
13 energy_Max = 1e50 erg
14 T_Min = 1 K
15 T_Max =  1e20 K
16 # ***** Jet settings:
17 # PowerLaw -> p
18 # # BrokenPowerLaw -> p1 p2 gamma_B
19 # n_e = BrokenPowerLaw
20 # p1 = 1.1
21 # p2 = 2.0
22 # gamma_B = 2
23 n_e = PowerLaw
24 p = 1.10
25 gamma_Min = 1
26 B = 1e-5 G
27 Z = 1.22
28 mu_0 = 1.4
29 E_released = 1
30 f_delta = 1
31 # ***** Processes to compute:
32 id_brem = n
33 id_syn = y
34 id_ssa = y
35 id_ssc = y
36 id_ec = n
37 # --- CMB:
38 id_cmb = n
39 # ***** Disc Settings:
40 id_disc = n
41 # M_BH = 1e45 g
42 a = 1
43 P_jet = 1e45 erg s-1
44 L_disc = 2e46 erg s-1
45 eta = 0.083
46 R_g_unit = y
47 R_in = 6
48 R_out = 200
49 # ***** Plot settings:
50 # nu_Min and nu_Max are the magnitude number, hence they must be
       float.
51 # e.g. nu_Min = 8 -> 10^8 Hz
52 nu_Min = 0
53 nu_Max = 29
54 ObserverDistance = 1e27 cm
55 Obs_theta = 0 deg
56 Obs_phi = 90 deg
```

```
57  # ***** Unit settings:
58  DistanceUnit = 4.3460247626638975e+24 cm
59  TimeUnit = 2.898016026760564e+17 s
60  MassUnit = 1.4644322980212338e+43 kg
61  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:

```
1       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.:

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

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

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

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

```
1    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:

```
1    id_brem = y
2    id_syn = n
3    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:

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

File Paths

As a brief recap it is composed by:

```
1    # ***** File paths :
2      file_path = /home/user/path/to/file.hdf5
3      file_saving_path = ''
4    # ***** if you want to launch multiple session -> id_launch = y
5      id_launch = n
6      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:

```
1    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:

```
1    runA.hdf5
2    runB.hdf5
3    runC.hdf5
```

and the `config.txt` file would be:

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

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

```
1 > 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:

```
1    n = lenght(list.txt)
2    for i in range(1, n)
3        > 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.:

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

```
1   n_e = PowerLaw
2   p = 1.1
3   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:

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

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

```
1      n_e = BrokenPowerLaw
2      p1 = 1.1
3      p2 = 2.0
4      gamma_B = 5
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:

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

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: (A.R. King, J.E. Pringle 2021 [3], DOI 10.3847/2041-8213/ac19a1).

• `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;
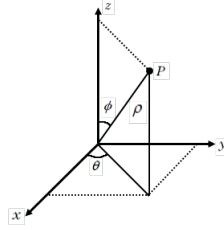
Figure 1.1 shows the correct coordinates.

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:

```
1     Initialization complete. Here a short summary:
2 ------- Summary -------
3 File name:
4 -  run1.hdf5
5 Emission processes:
6 - Bremmstrahlung
7 - Synchrotron
8 - - self-absorption
9 - - self-Compton
10 - External Compton
11 - - CMB
12 - - Disc
13 The selected power law is: PowerLaw
14 The resulting plot will be called: run1_B_S_a_c_EC_cmb_disc.png
15 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:

```
1     Attention, External Compton is set as off but the following
      processes are still set to be computed:
2 - CMB
3 - Disc
4 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.
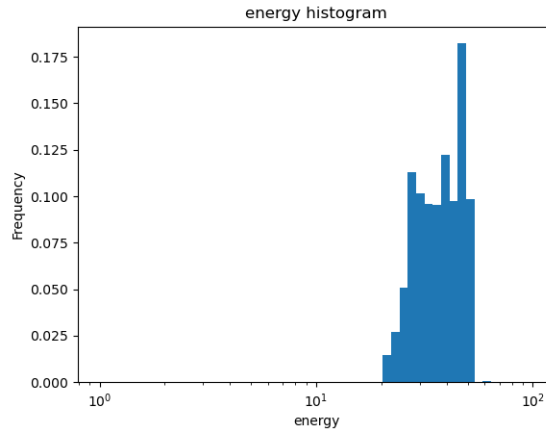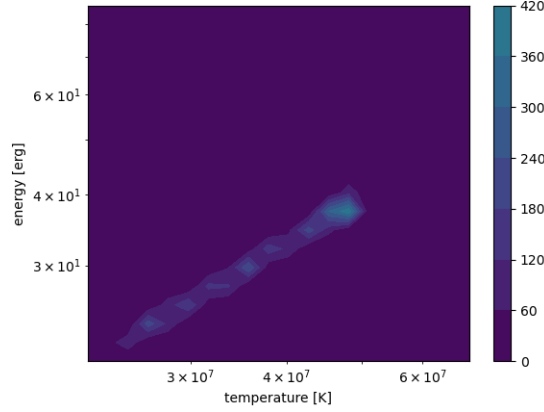

Figure 1.2: Example of histogram plot

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.

```
1  #
2  file_path = /home/user/path/to/file.hdf5
3  file_saving_path = ''
4  # If the user wants to have the arrays printed:
5  display_arrays = yes
6  # AVAILABLE LABEL:
7  # -- 'density'
8  # -- 'pressure'
9  # -- 'energy'
10 # -- 'temperature
11 kind = histogram
12 #   AVAILABLE LABEL:
13 # -- 'histogram'
14 #       if 'kind' = histogram then:
15 #           bin_rule = (one between sqrt, rice or sturges)
16 #           label1 = parameter1
17 #           label2 = parameter2 etc.
18 #       example: label1 = density
19 # -- 'contour'
20 #       if kind = counter then the following must be stated:
      x_label and y_label must be stated.
21 #           - x_label (which quantity must be on x-axis)
22 #           - y_label (which quantity must be on y-axis)
23 #           - n (number of bins to consider)
24 #           - clim (max value to display on the color bar)
25 #       n number of bins must be stated aswell.
```

```
26  #         example: x_label = density
27  #         example: y_label = pressure
28  #         example:       n = 100
29  #
30  label1 = temperature
31  label2 = energy
32  bin_rule = sqrt
33  x_label = temperature
34  y_label = energy
35  n = 20
36  clim = 1000
37  # ****** Interval settings:
38  pres_Min = 1e-20 cm-1 g s-2
39  pres_Max = 1e-5 cm-1 g s-2
40  dens_Min = 1e-30 g cm-3
41  dens_Max = 2e-20 g cm-3
42  energy_Min = 1 erg
43  energy_Max = 1e2 erg
44  T_Min = 1e1 K
45  T_Max = 1e20 K
46  # ***** Unit settings:
47  DistanceUnit = 4.3460247626638975e+24 cm
48  TimeUnit = 2.898016026760564e+17 s
49  MassUnit = 1.4644322980212338e+43 kg
50  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:

```
1      'density'
2      'pressure'
3      'energy'
4      '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:

```
1    kind = histogram
2    label1 = temperature
3    label2 = energy
4    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`

$$n_{\text{bin}} = \texttt{round}(n_{\text{blocks}}^{1/2}) \tag{1.10}$$

- `rice`

$$n_{\text{bin}} = 2 * \texttt{round}(n_{\text{blocks}}^{1/3}) \tag{1.11}$$

- `sturges`

$$n_{\text{bin}} = \texttt{round}(\log_2(n_{\text{block}}) + 1) \tag{1.12}$$

counter   Example of counter parameters:

```
1    kind = counter
2    x_label = density
3    y_label = pressure
4    n = 100
5    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:

```
1    file_contour_density_pressure.png
2    file_histogram_temperature.png
3    file_histogram_energy.png
```

Chapter 2

How it Works

WIP