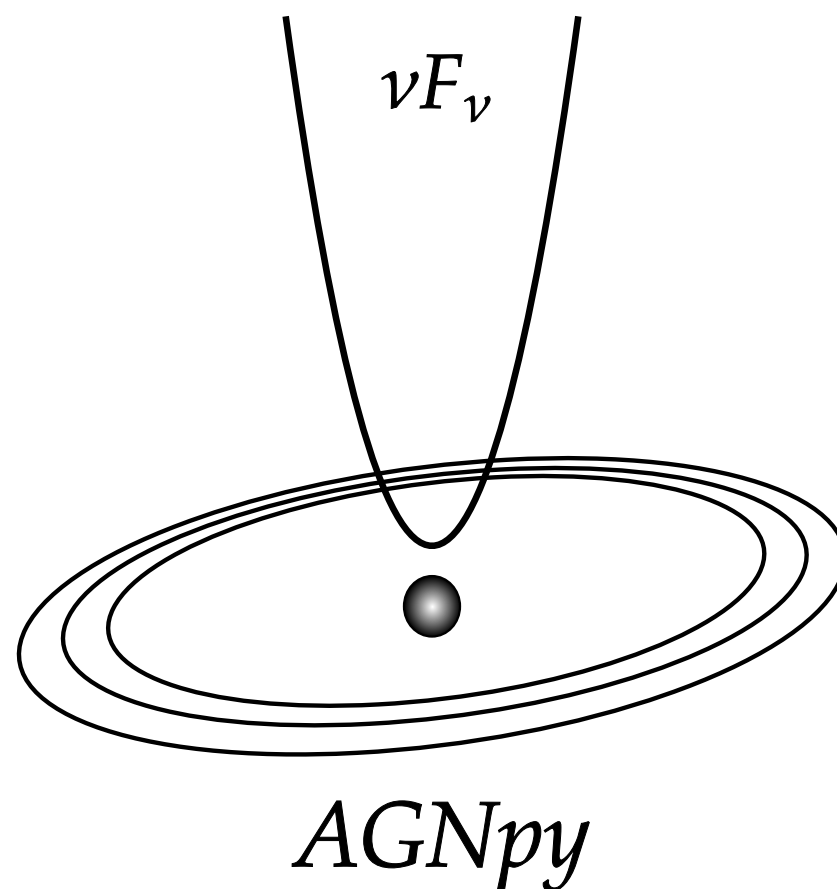




C. NIGRO

AGNPY



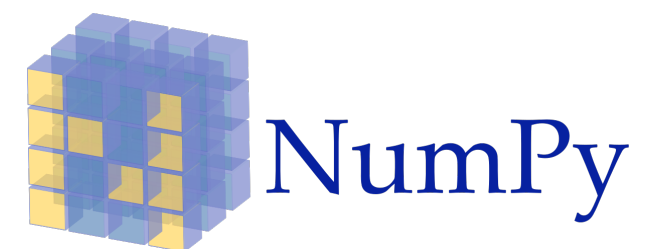
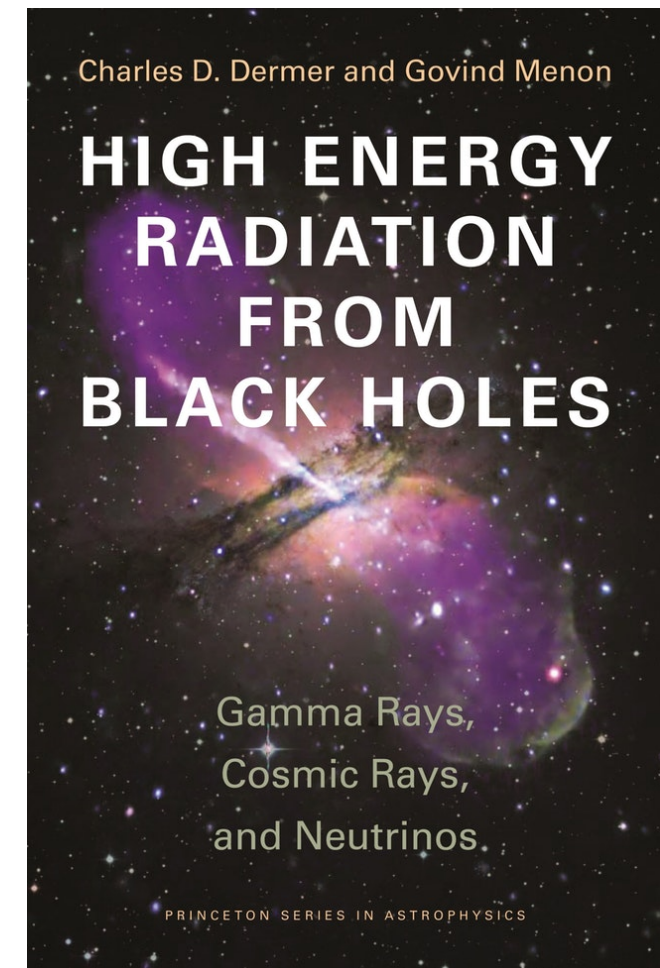
- ▶ Developed by me during my Ph.D.;
- ▶ focuses on the numerical computation of the photon spectra produced by **leptonic** radiative processes in **jetted** active galactic nuclei;
- ▶ my idea was to create a tool for AGN modelling that lived in the **numpy** + **astropy** ecosystem, increasingly dominant in astronomy.

► *REFERENCES (and processes implemented)*

- Notations and basic formulas are borrowed from [Dermer and Menon \(2009\)](#);
- the implementation of **synchrotron** and **synchrotron self-Compton (SSC)** radiative processes relies on [Dermer and Menon \(2009\)](#) and [Finke et al. \(2008\)](#);
- [Dermer et al \(2009\)](#) and [Finke \(2016\)](#) are instead the main references for the **external Compton (EC)** radiative processes.

► *IMPLEMENTATION*

- The numerical operations are delegated to [numpy arrays](#);
- all the physical quantities are casted as [astropy quantities](#).



▶ **INSTALLATION**

- ▶ The package is shipped via the Python Package Index;
- ▶ Numpy, Astropy and matplotlib are the only dependencies.
- ▶ `pip install agnpy`

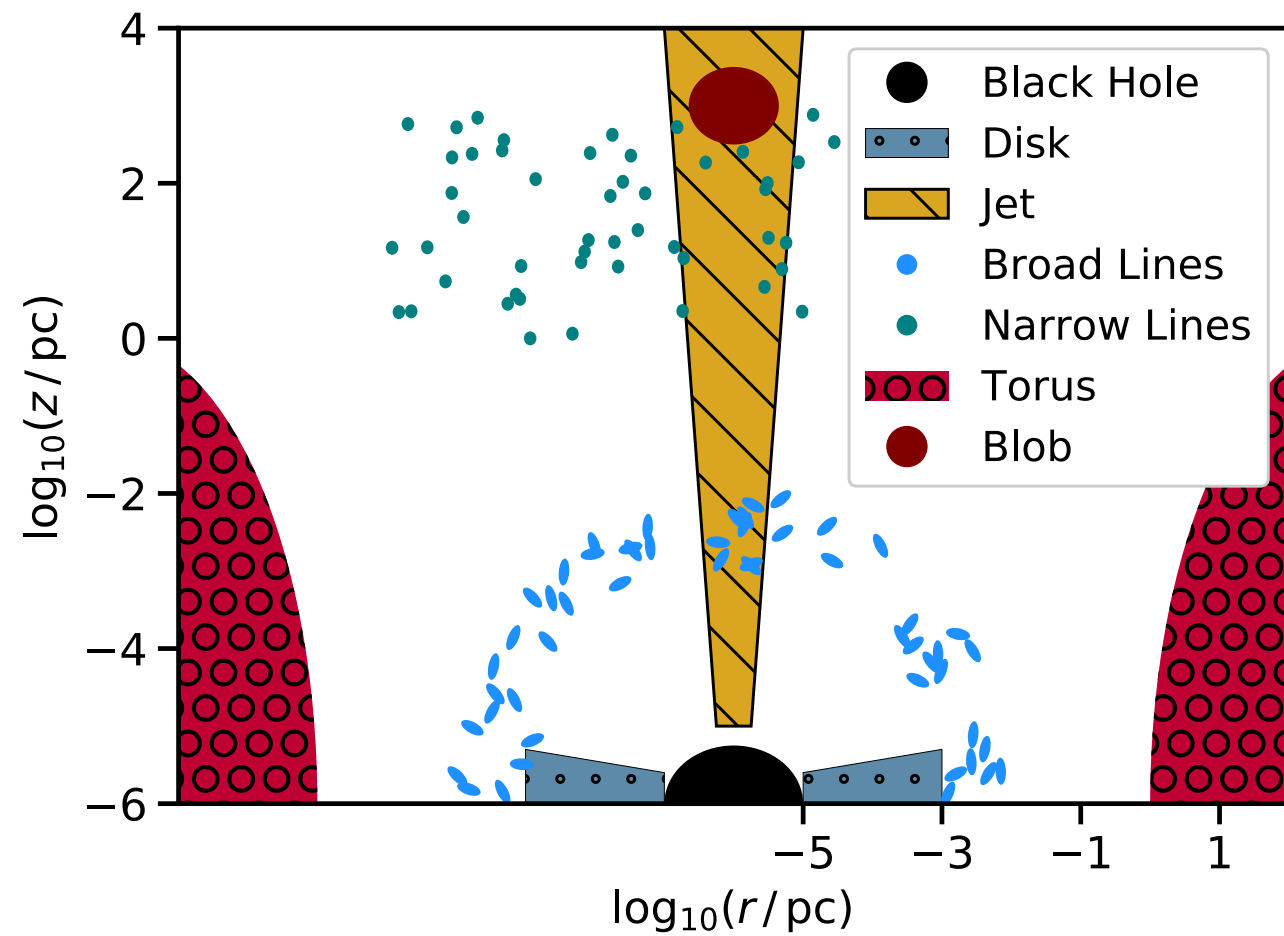
▶ **GITHUB**

- ▶ Fork the project at: <https://github.com/cosimoNigro/agnpy>.

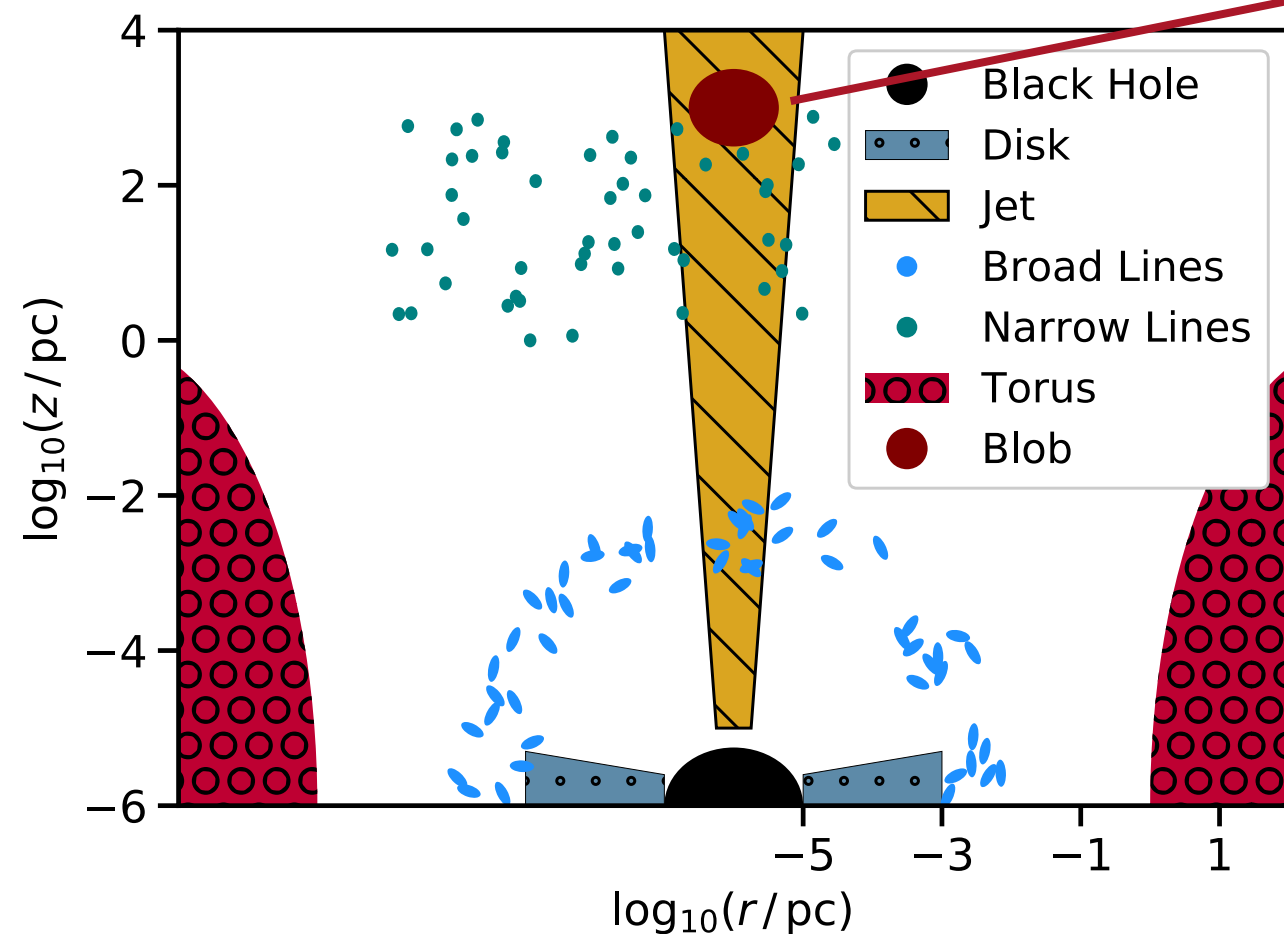
▶ **DOCUMENTATION**

- ▶ Check it at: <https://agnpy.readthedocs.io/en/latest/>.

LEPTONIC MODELLING IN A NUTSHELL



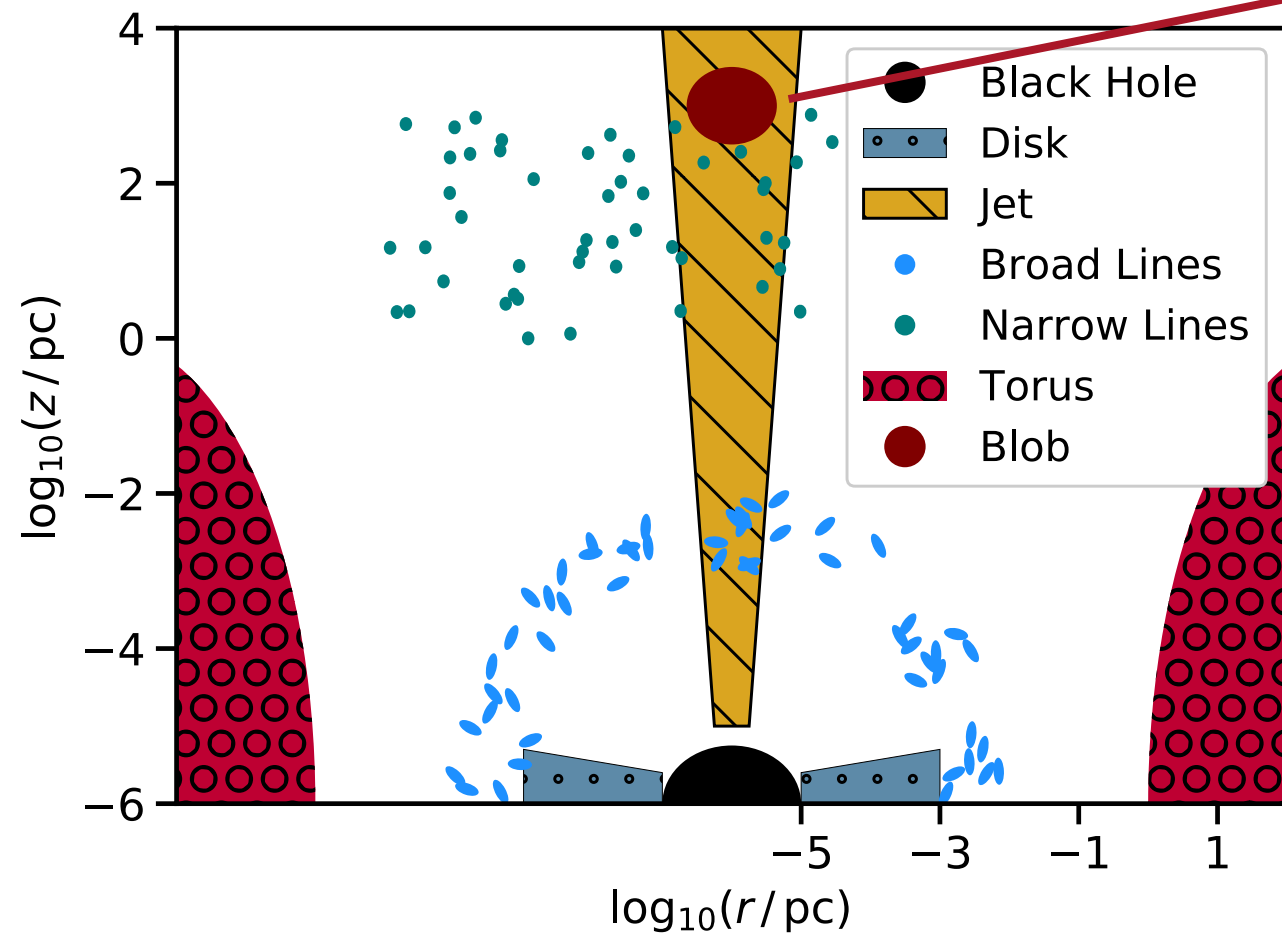
LEPTONIC MODELLING IN A NUTSHELL



Spectra of jetted AGN can be modelled considering a single plasmoid (**blob**) streaming along the jet, containing a non-thermal population of electrons (e^\pm).

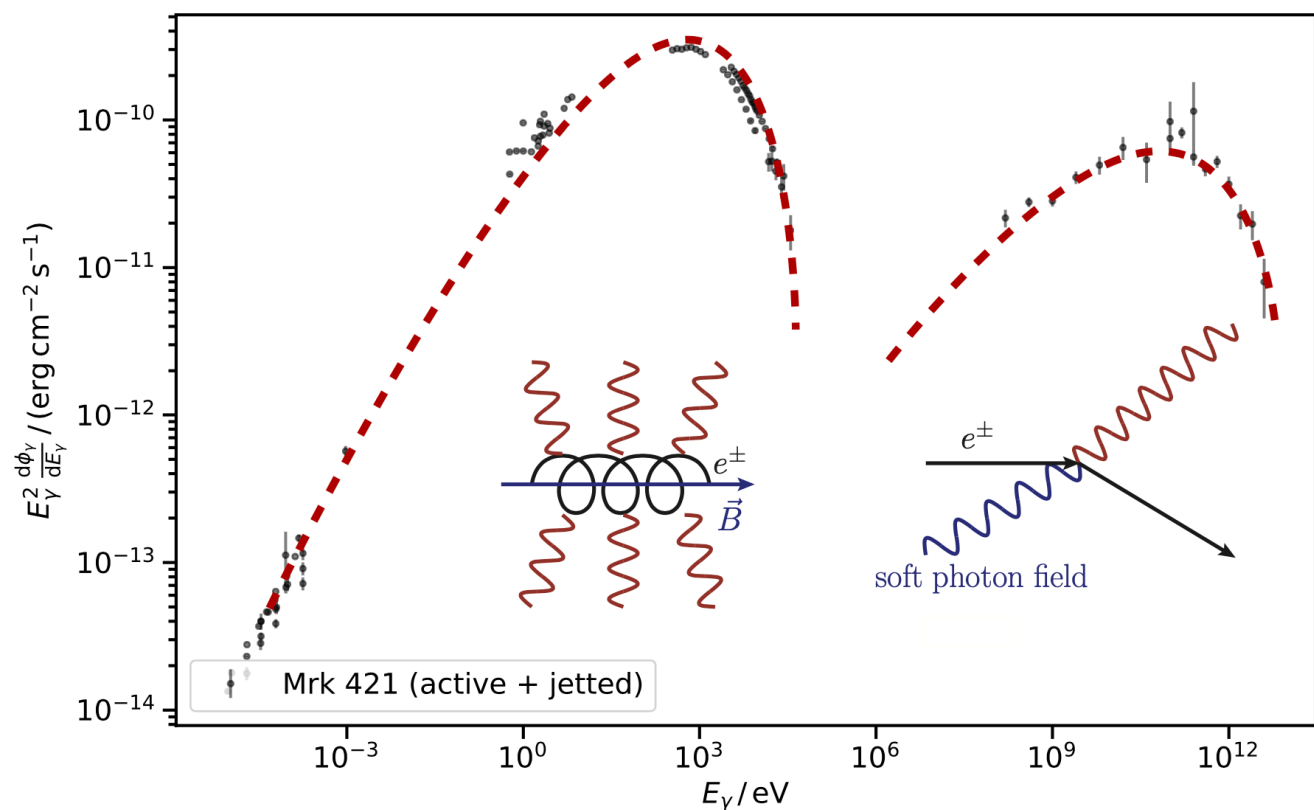
Their radiative processes produce the photon spectrum.

LEPTONIC MODELLING IN A NUTSHELL

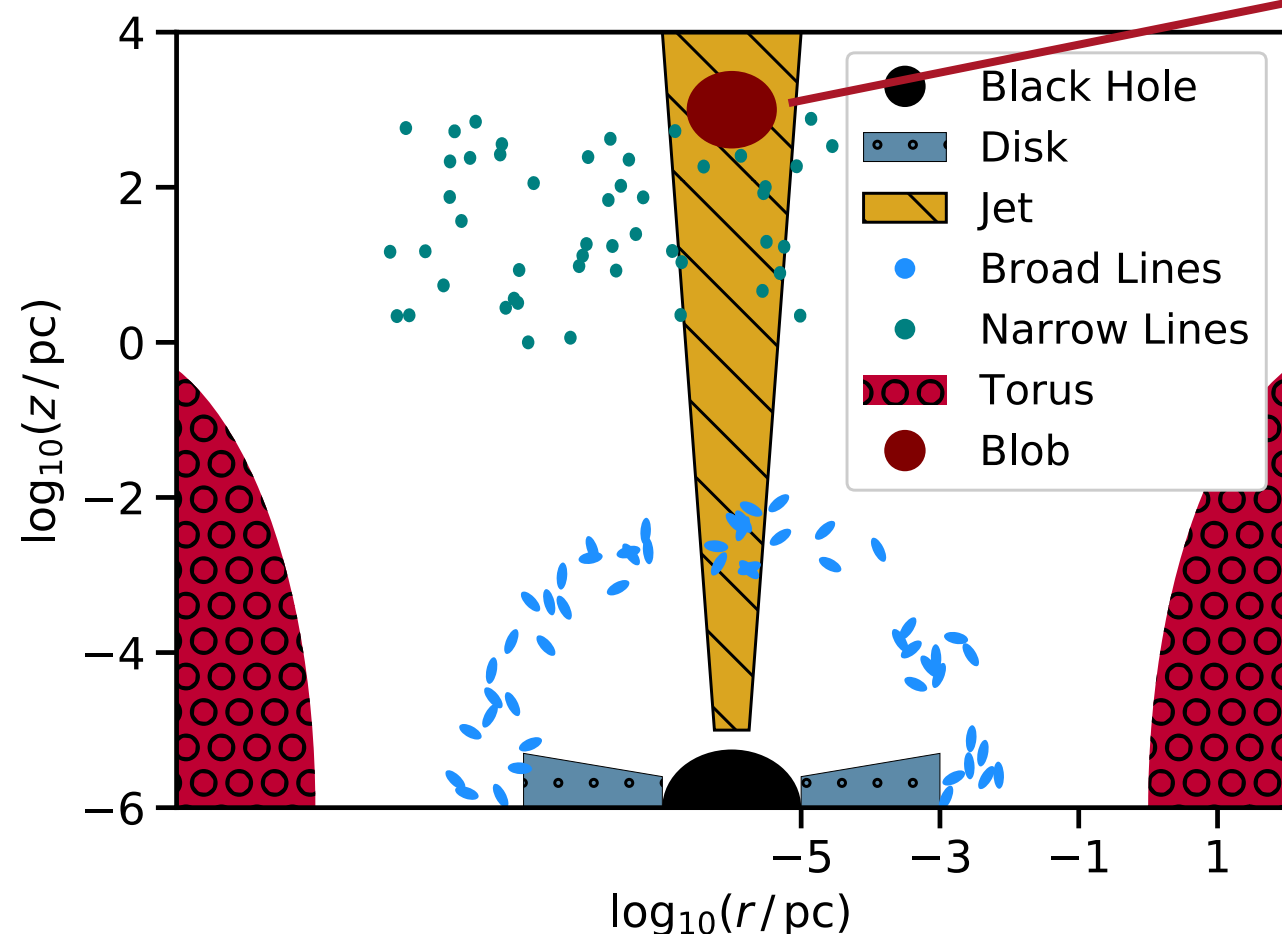


Spectra of jetted AGN can be modelled considering a single plasmoid (**blob**) streaming along the jet, containing a non-thermal population of electrons (e^\pm).

Their radiative processes produce the photon spectrum.

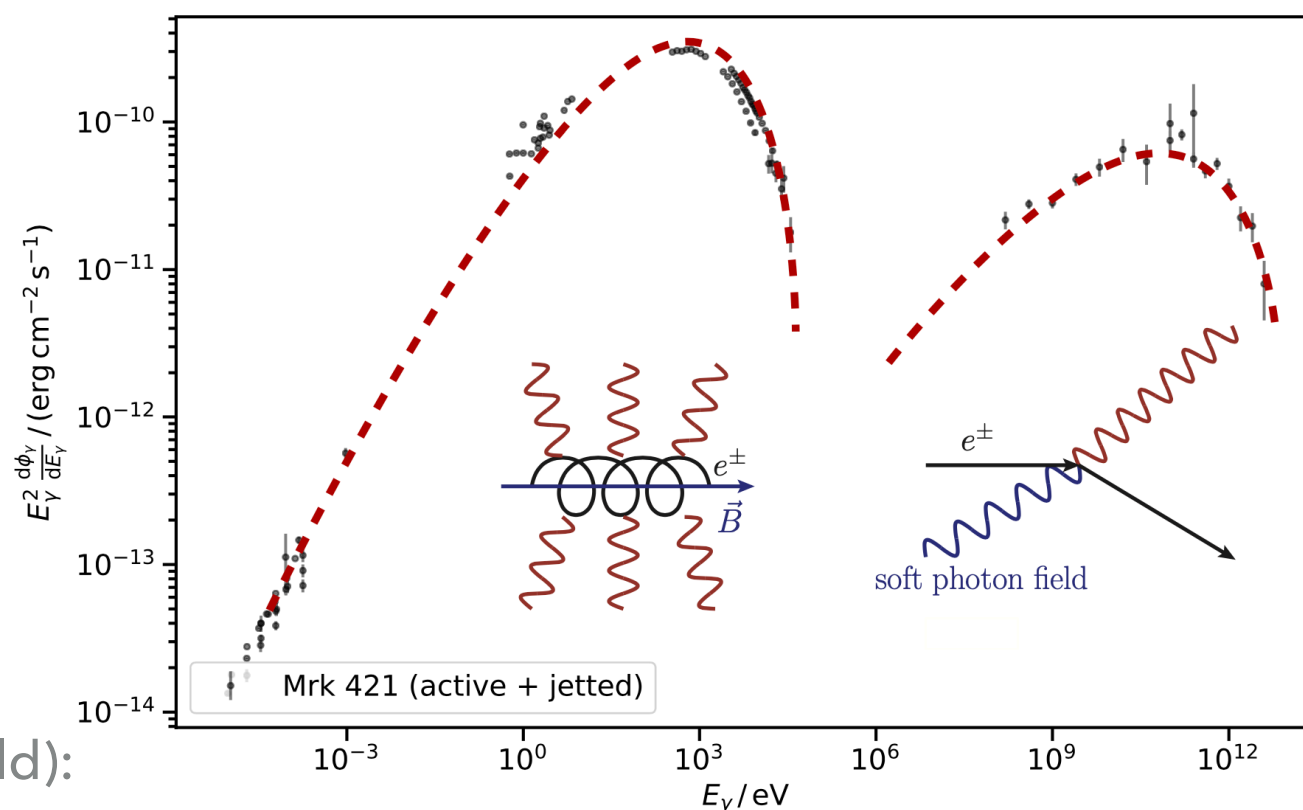


LEPTONIC MODELLING IN A NUTSHELL



Spectra of jetted AGN can be modelled considering a single plasmoid (**blob**) streaming along the jet, containing a non-thermal population of electrons (e^\pm).

Their radiative processes produce the photon spectrum.



► **Targets** for the Inverse Compton (soft photon field):

- Synchrotron photons from the blob (Synchrotron Self Compton): BL Lac blazars;
- Photon fields produced by the accretion: Disk black body, broad lines, Torus black body (External Compton): FSRQ blazars.


BASIC AGN COMPONENTS: BLOB

`from agnpy.emission_regions import Blob`

`class agnpy.emission_regions.Blob(R_b, z, delta_D, Gamma, B, spectrum_norm, spectrum_dict, spectrum_norm_type='integral', gamma_size=200)`

Simple spherical emission region.

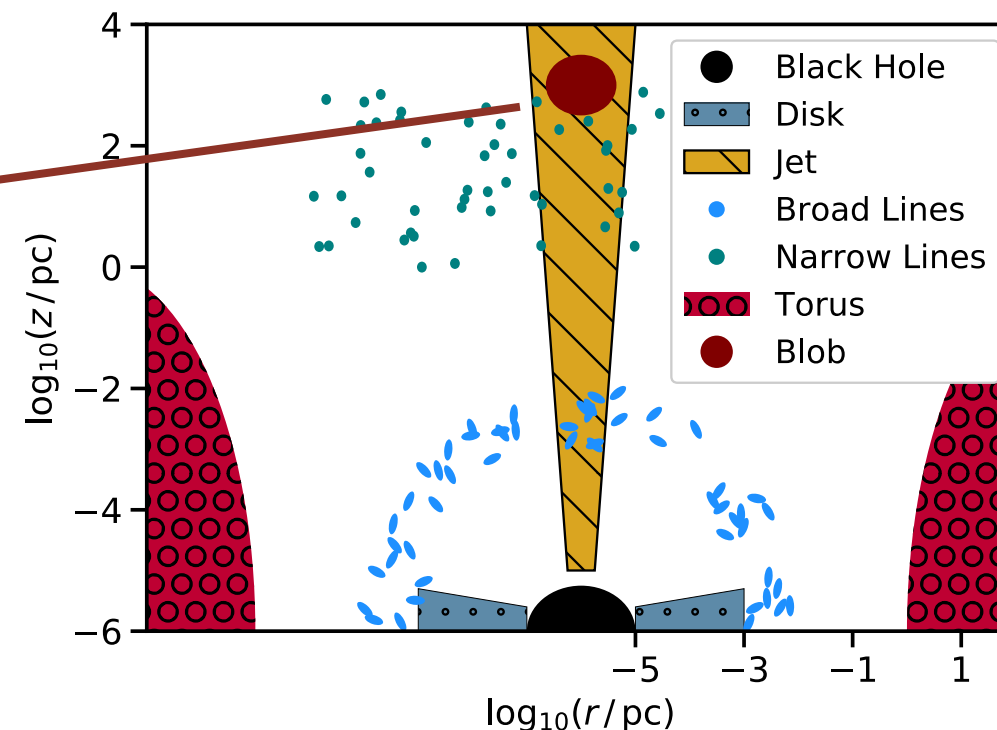
Note: all these quantities are defined in the comoving frame so they are actually primed quantities when referring the notation in [DermerMenon2009].

- Parameters:**
- **R_b** (*Quantity*) - radius of the blob
 - **z** (*float*) - redshift of the source
 - **delta_D** (*float*) - Doppler factor of the relativistic outflow
 - **Gamma** (*float*) - Lorentz factor of the relativistic outflow
 - **B** (*Quantity*) - magnetic field in the blob (Gauss)
 - **spectrum_norm** (*Quantity*) -  The emission region (Blob) contains the electron spectrum responsible for the radiation.
- normalisation of the electron spectra, by default can be, following the notation in [DermerMenon2009]:
- $n_{e, tot}$: total electrons density, in cm^{-3}
 - u_e : total electrons energy density, in erg cm^{-3}
 - W_e : total energy in electrons, in erg

see `spectrum_norm_type` for more details on the normalisation


- **spectrum_dict** (*dictionary*) - dictionary containing type and spectral shape information, e.g.:

```
spectrum_dict = {
    "type": "PowerLaw",
    "parameters": {
        "p": 2.8,
        "gamma_min": 1e2,
        "gamma_max": 1e7
    }
}
```



BASIC AGN COMPONENTS: EC TARGETS

```
from agnpy.targets import (
    SSDisk, SphericalShellBLR, RingDustTorus
)
```

`class agnpy.targets.SSDisk(M_BH, L_disk, eta, R_in, R_out)`  [Shakura1973] accretion disk.

Parameters:

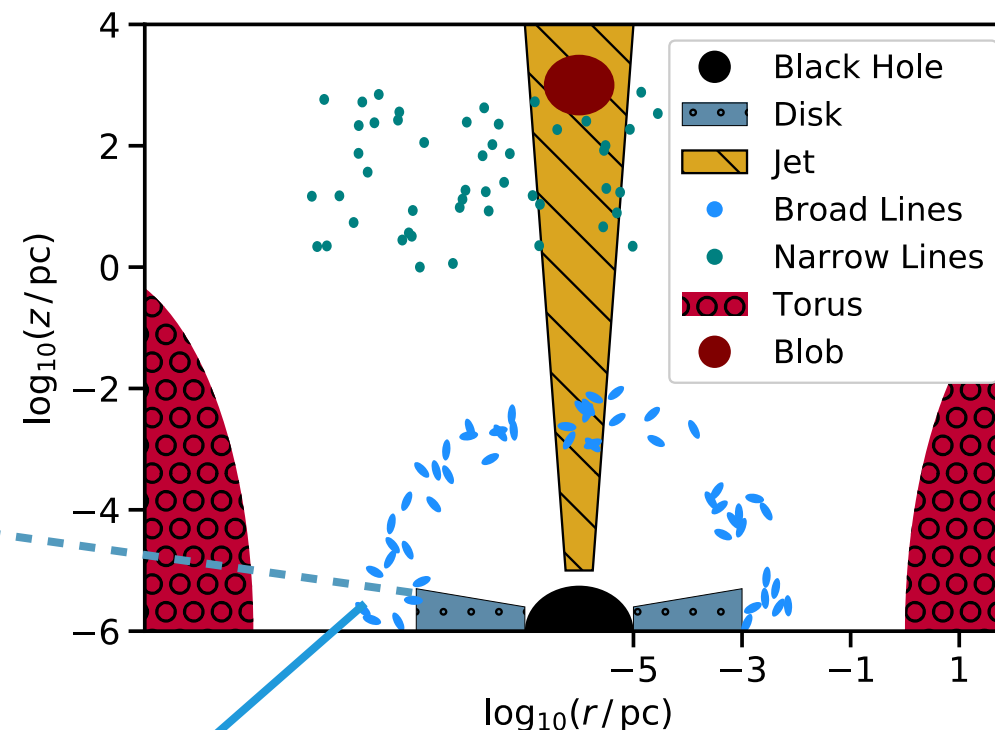
- **M_BH** (*Quantity*) - Black Hole mass
- **L_disk** (*Quantity*) - luminosity of the disk
- **eta** (*float*) - accretion efficiency
- **R_in** (*Quantity*) - inner disk radius
- **R_out** (*Quantity*) - outer disk radius

`class agnpy.targets.SphericalShellBLR(disk, xi_line, epsilon_line, R_line)` 

Spherical Shell Broad Line Region, from [Finke2016]. Each line is emitted from an infinitesimally thin spherical shell.

Parameters:

- **disk** (*SSDisk*) - disk whose radiation is being reprocessed by the BLR
- **xi_line** (*float*) - fraction of the disk radiation reprocessed by the BLR
- **epsilon_line** (*float*) - dimensionless energy of the emitted line
- **R_line** (*Quantity*) - radius of the BLR spherical shell



BASIC AGN COMPONENTS: EC TARGETS

```
from agnpy.targets import (
    SSDisk, SphericalShellBLR, RingDustTorus
)
```

`class agnpy.targets.SSDisk(M_BH, L_disk, eta, R_in, R_out)`  [Shakura1973] accretion disk.

Parameters:

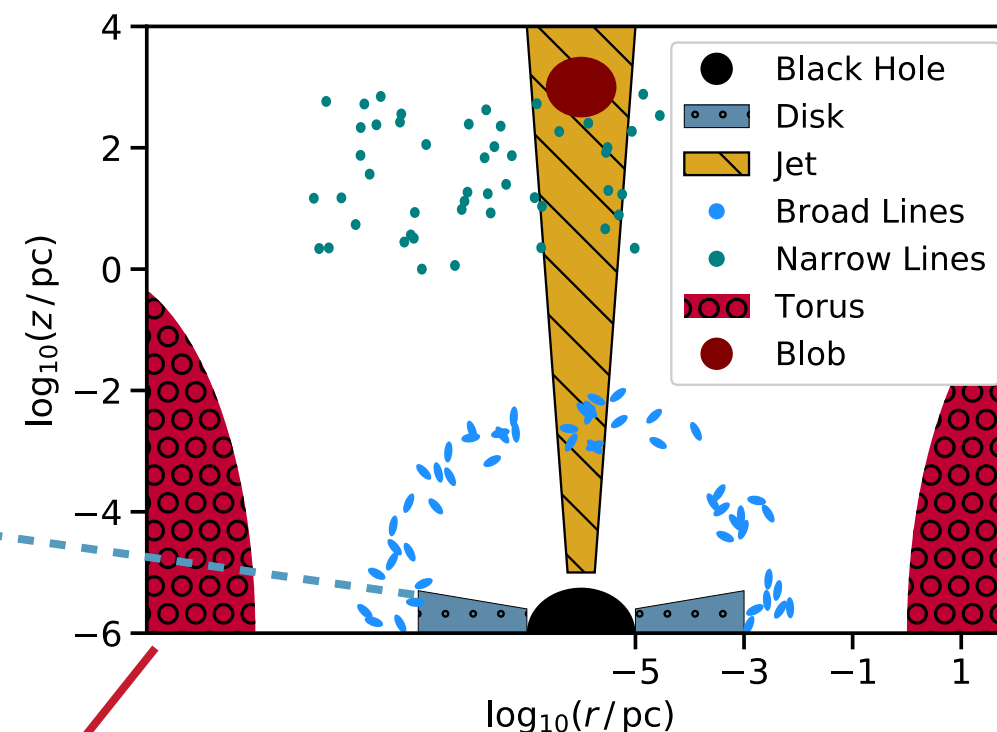
- **M_BH** (*Quantity*) - Black Hole mass
- **L_disk** (*Quantity*) - luminosity of the disk
- **eta** (*float*) - accretion efficiency
- **R_in** (*Quantity*) - inner disk radius
- **R_out** (*Quantity*) - outer disk radius

`class agnpy.targets.RingDustTorus(disk, xi_dt, epsilon_dt, R_dt=None)` 

Dust Torus as infinitesimally thin annulus, from [Finke2016]. For the Compton scattering monochromatic emission at the peak energy of the Black Body spectrum is considered.

Parameters:

- **disk** (*SSDisk*) - disk whose radiation is being reprocessed by the Torus
- **xi_dt** (*float*) - fraction of the disk radiation reprocessed
- **epsilon_dt** (*float*) - dimensionless energy peak of the black body distribution
- **R_dt** (*Quantity*) - radius of the Torus, if not specified the saturation radius of Eq. 96 in [Finke2016] will be used



RADIATIVE PROCESSES

For the **synchrotron** radiation only a Blob instance is necessary

```
from agnpy.synchrotron import Synchrotron  
synch = Synchrotron(blob)
```

For the **synchrotron self-Compton** a Blob and a Synchrotron instances are necessary

```
from agnpy.compton import SynchrotronSelfCompton  
ssc = SynchrotronSelfCompton(blob, synch)
```

For the **external Compton** a Blob and a Target instances are necessary, the distance between the blob and the BH (i.e. from the photon field) has also to be specified.

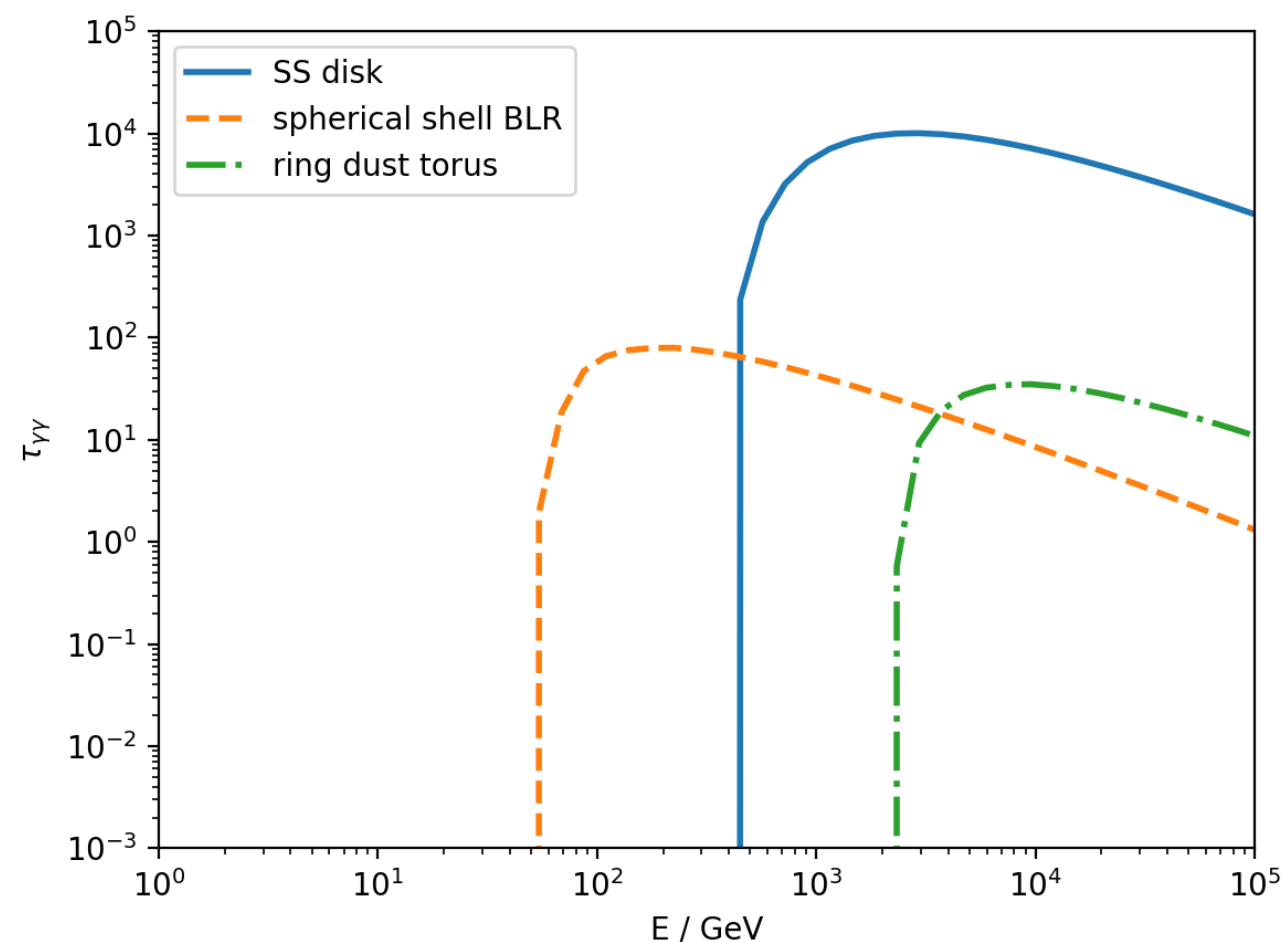
```
from agnpy.compton import ExternalCompton  
ec_disk = ExternalCompton(blob, disk, r=1e17 * u.cm)  
ec_blr = ExternalCompton(blob, blr, r=1e17 * u.cm)
```

TUTORIALS

- ▶ Tutorial with synchrotron and synchrotron self-Compton processes: https://github.com/cosimoNigro/agnpy/blob/master/tutorials/synchrotron_self_compton.ipynb.
- ▶ Tutorial with thermal targets and external Compton: https://github.com/cosimoNigro/agnpy/blob/master/tutorials/synchrotron_self_compton.ipynb.

OPTICAL DEPTH

- ▶ The code also contains a module to calculate the optical depth due to $\gamma\gamma$ pair production of the radiated photons on the fields



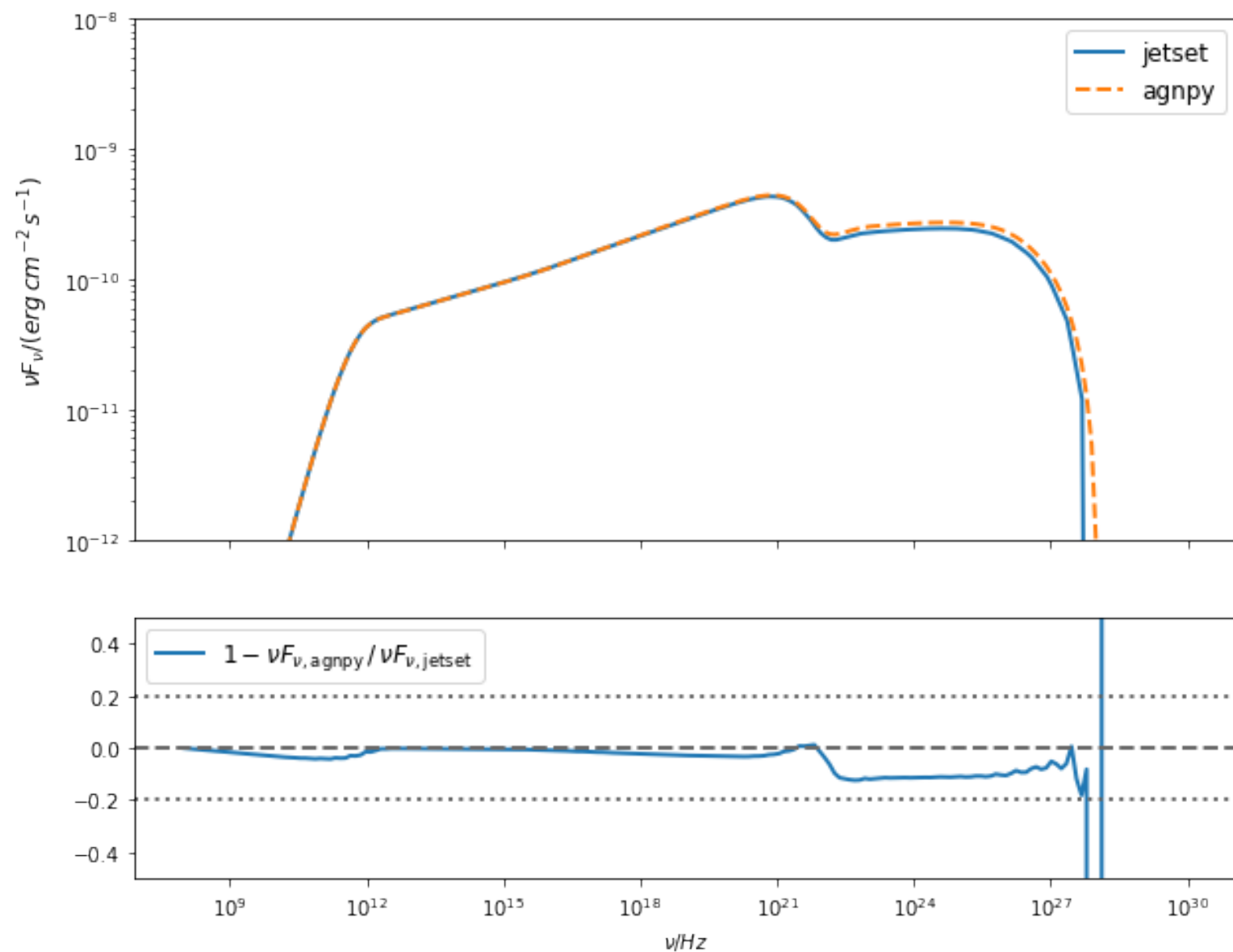
```
from agnpy.absorption import Absorption
absorption_disk = Absorption(blob, disk, r=1e17 * u.cm)
```

NEXT STEPS: INTEGRATION WITH GAMMAPY

- ▶ For the moment I simply tried to wrap a SSC sed in a SpectralModel and fit a series of FluxPoints: https://github.com/cosimoNigro/agnpy/blob/master/tutorials/gammapy_fit.ipynb

NEXT STEPS: CROSSCHECKS

- I started to do some cross checks with other codes and with some results of the literature <https://github.com/cosimoNigro/agnpy/tree/master/tests/crosschecks>



NEXT STEPS: PUBLICATION PLANS

- ▶ I will submit my code to the [journal of open source software](#);
- ▶ I asked to become [astropy affiliated package](#).



The Journal of
Open Source Software

The Journal of Open Source Software is a developer friendly, open access journal for research software packages.

Committed to publishing quality research software with zero article processing charges or subscription fees.

[Submit a paper to JOSS](#)



[Volunteer to review](#)

Astropy Affiliated Packages

A major part of the Astropy Project is the concept of “Astropy affiliated packages”. An affiliated package is an astronomy-related Python package that is not part of the astropy core package, and is not managed by the project but is a part of the Astropy Project community. These packages demonstrate a commitment to Astropy’s goals of improving reuse, interoperability, and interface standards for Python astronomy and astrophysics packages. In many (but not all) cases, affiliated packages also follow similar development processes and package templates as for the core package.

If you are developer interested in signing up as an affiliated package, details are in the [Becoming an Affiliated Package](#) section.