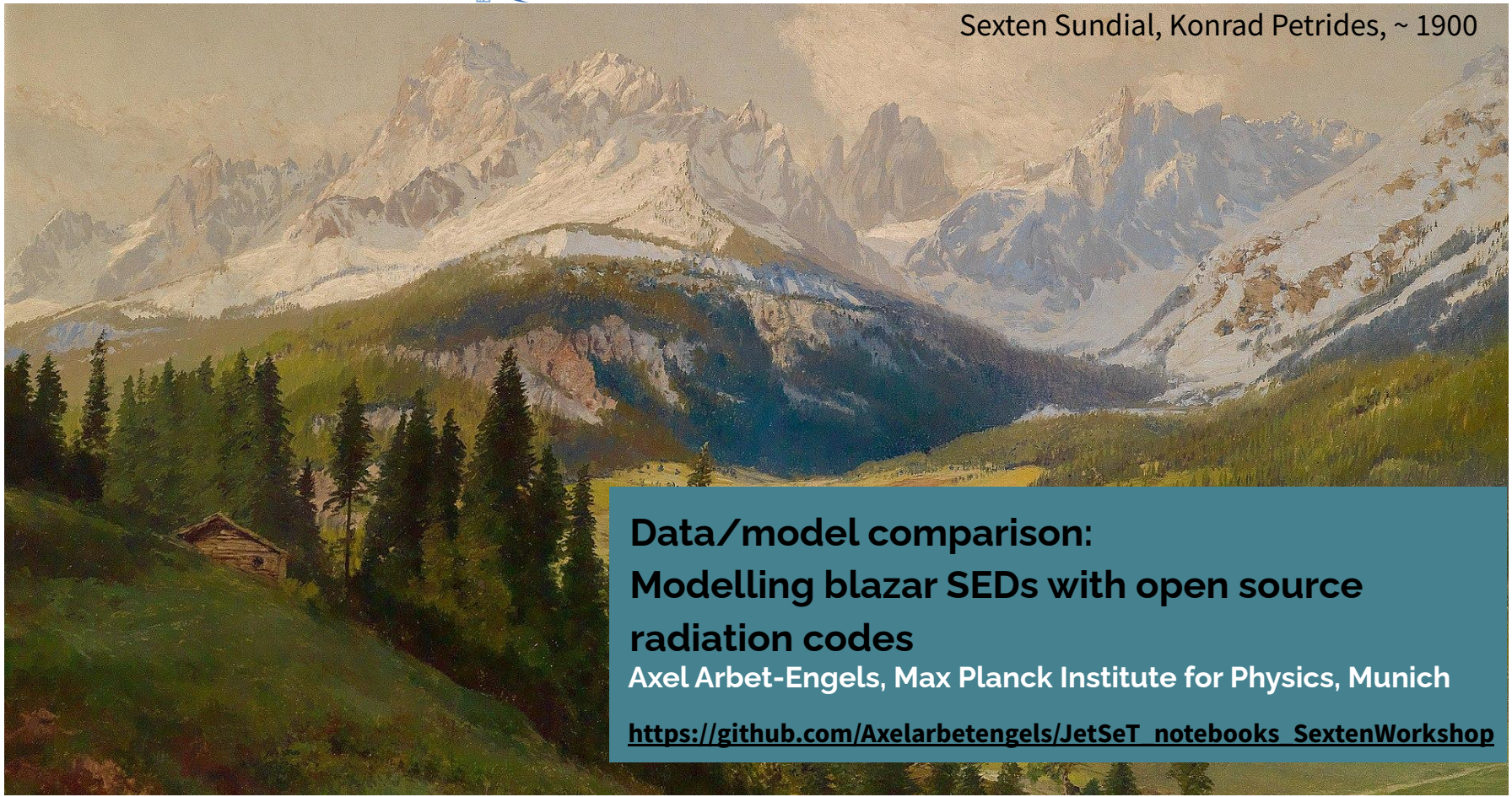
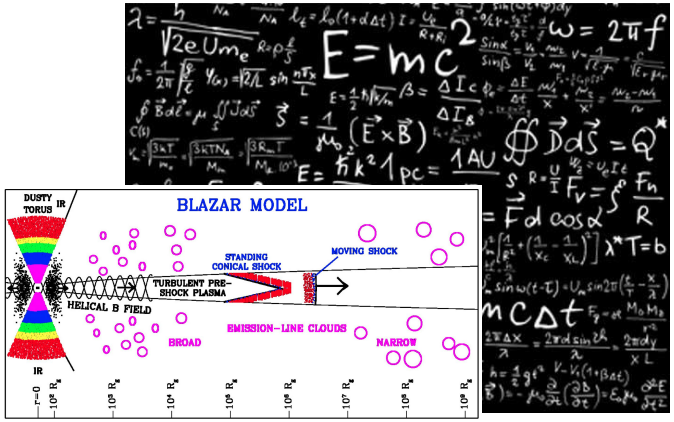


Data/model comparison: Modelling blazar SEDs with open source radiation codes

Axel Arbet-Engels, Max Planck Institute for Physics, Munich

https://github.com/Axelarbetengels/JetSeT_notebooks_SextenWorkshop



Theory & Models

(Blazar) Observations

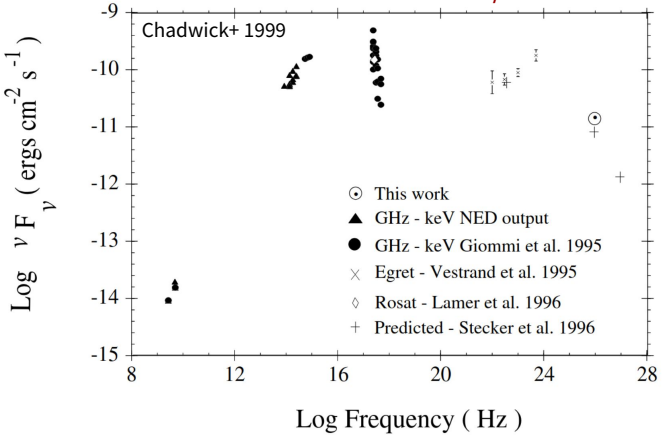
Data / model comparisons

Several (many) **open source blazar modelling codes** were developed in the past years

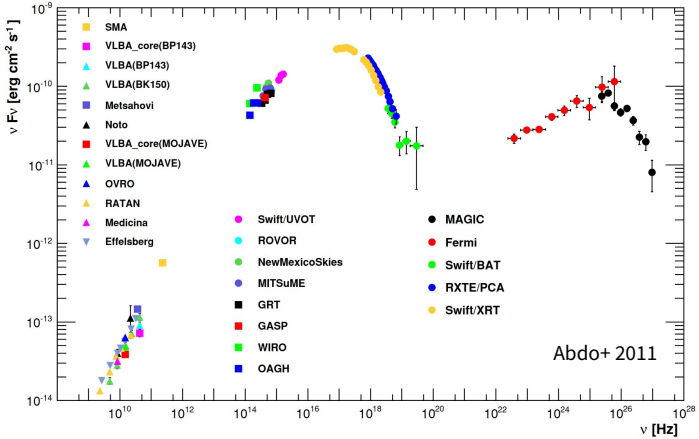
Data/model comparison getting "easier"

Observational improvements

PKS2155-304 in 1995/1997



Markarian 421 in 2008



Huge improvement in sensitivity & energy coverage in the past few decades

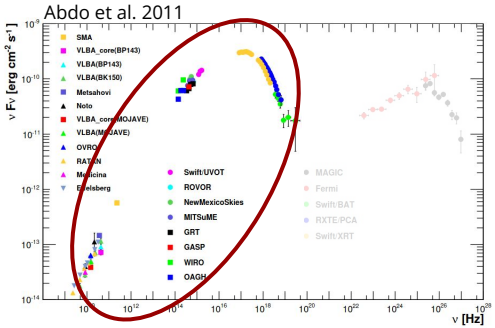
(sensitive VHE & hard X-rays observations, EHT, **X-ray polarization** etc…)

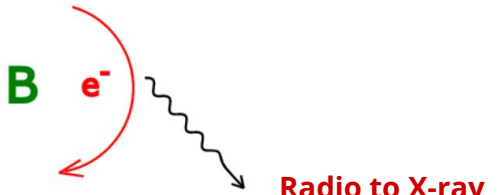
Wealth of new "informations" offer new possibilities to test models

Outline

1. Introduction
2. Some **open source** modelling tools
3. Introduction to the **JetSeT code**
4. Hands-on session using **JetSeT**
5. Questions & Answers

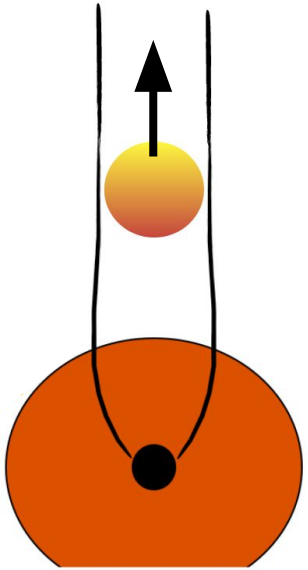
Emission processes - simplified view





electron synchrotron

Bulk lorentz factor
$\Gamma \gtrsim 10$

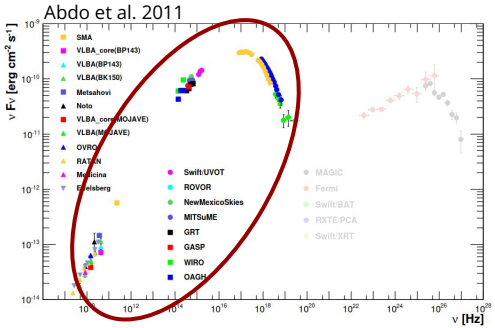


Sketch credits: F. Tavecchio

Synchrotron origin of first component is well established:

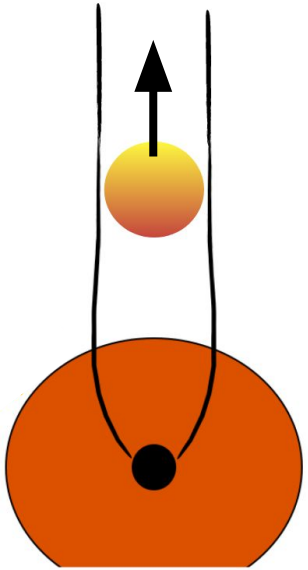
→ From polarization measurements
→ Spectral shape
→ Absorption measured at lowest frequencies

Emission processes - simplified view

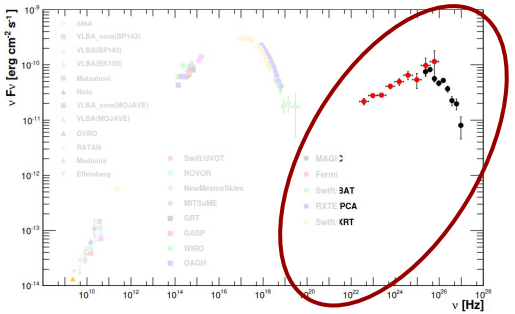


electron synchrotron

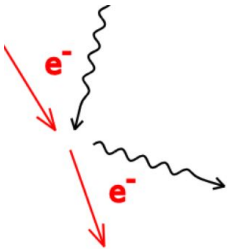
Bulk lorentz factor
Γ ≳ 10



Sketch credits: F. Tavecchio



Synchrotron Photon (radio to UV)



Inverse Compton photon (X-rays to gamma-rays)

Inverse Compton scattering

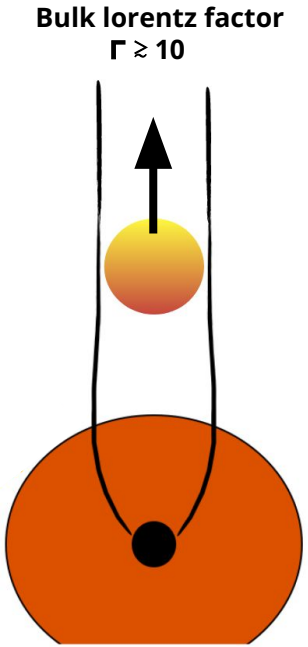
→ Synchrotron self-Compton model (SSC)

Emission processes - simplified view

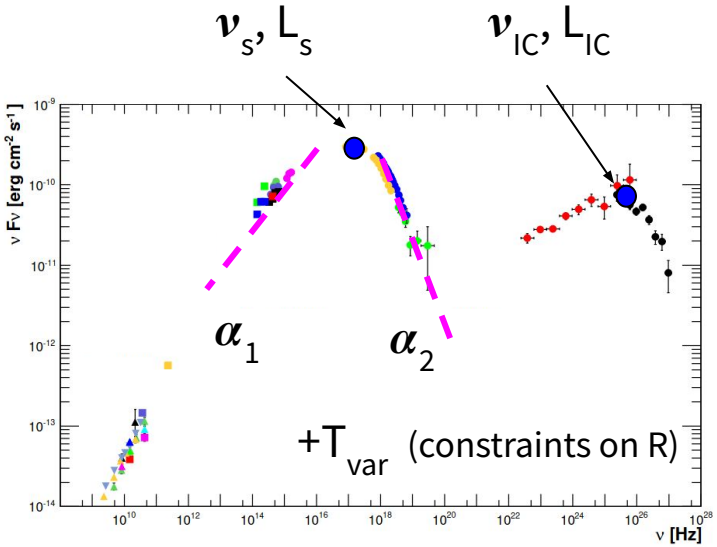
"Inject" e+/- distribution
Power law *or* Broken power law

Other parameters:
Doppler factor (~10-50),
B (~0.1-0.01 G),
R (~1e15-1e17 cm)

→ ~7 free parameters



Sketch credits: F. Tavecchio



7 observables

→ 1-zone SSC is a ~closed system

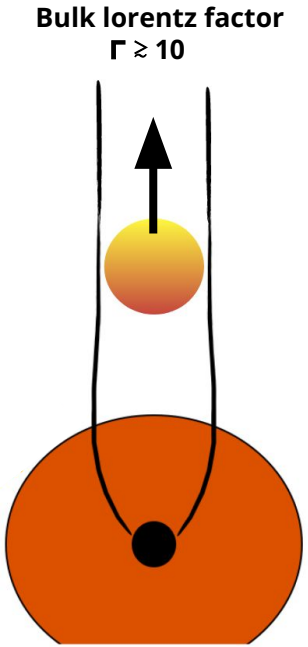
→ Synchrotron self-Compton model (SSC)

Emission processes - simplified view

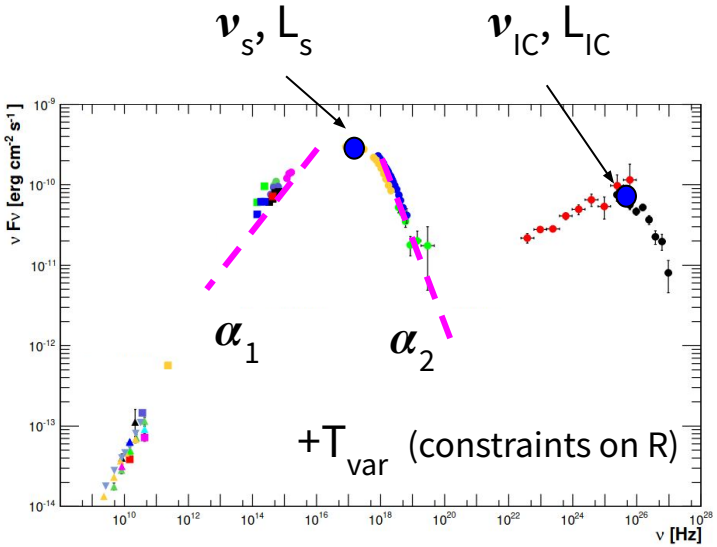
"Inject" e+/- distribution
Power law *or* Broken power law

Other parameters:
Doppler factor (~10-50),
B (~0.1-0.01 G),
R (~1e15-1e17 cm)

→ ~7 free parameters



Sketch credits: F. Tavecchio

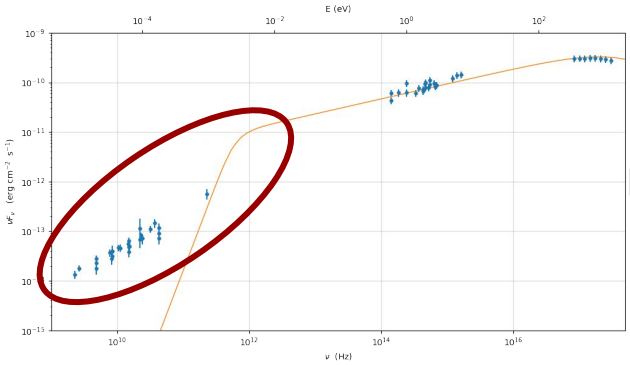


7 observables

→ 1-zone SSC is a ~closed system
there is a but…

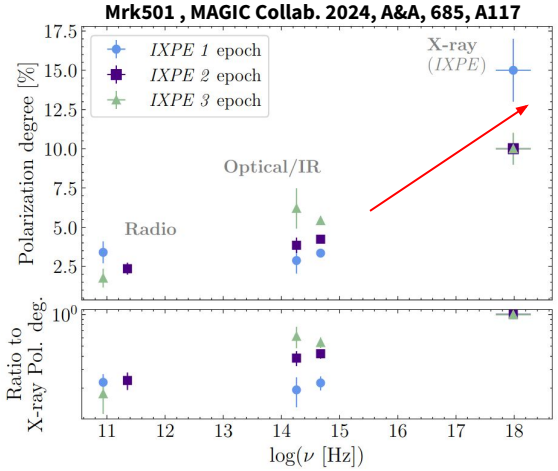
→ Synchrotron self-Compton model (SSC)

Emission processes - simplified view

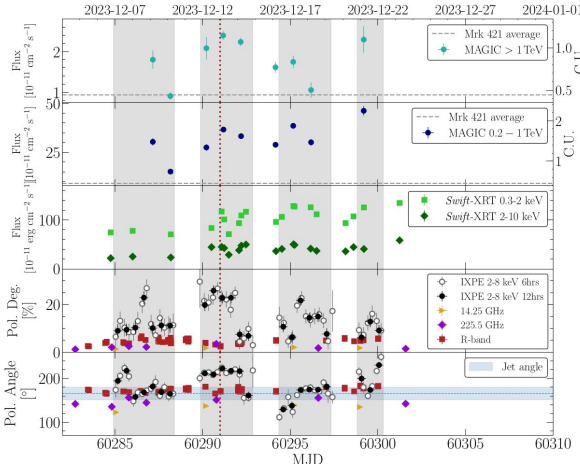


Self-absorption of radio flux in compact X-ray / VHE emitting jet regions

Mrk501 , MAGIC Collab. 2024, A&A, 685, A117

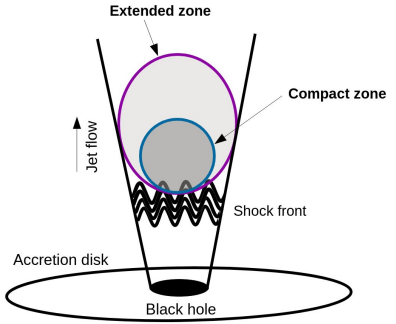


Mrk421 , MAGIC Collab. 2025, A&A, 695, A217

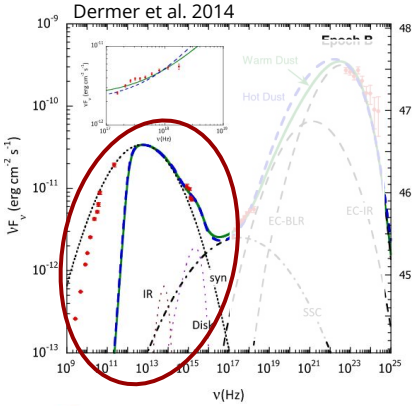


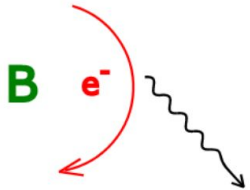
Energy dependency of the **polarization**
→ complex geometry

Data suggest complexity
→ likely several emitting regions
→ drastically increase the free parameters



Emission processes - simplified view

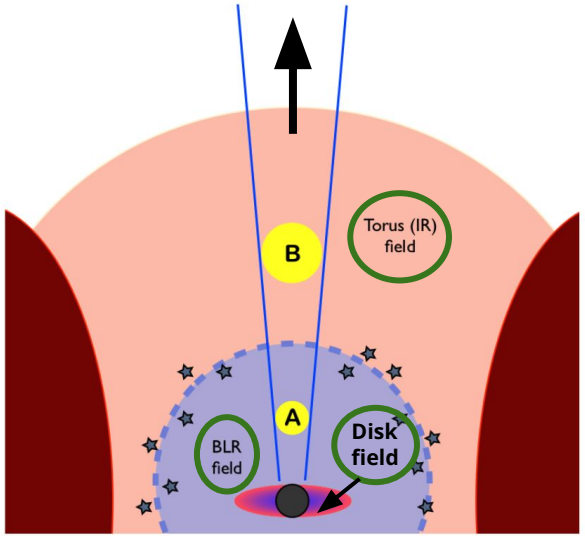




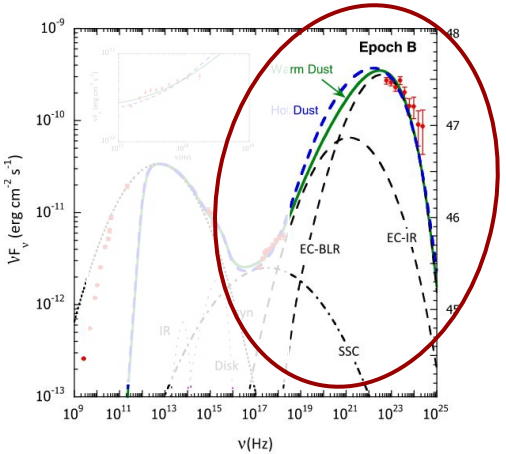
Radio to X-ray

electron synchrotron

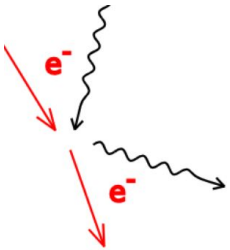
Bulk lorentz factor
$\Gamma \gtrsim 10$



Sketch credits: F. Tavecchio



Synchrotron + BLR + Disk + Torus photon field



Inverse Compton photon (X-rays to gamma-rays)

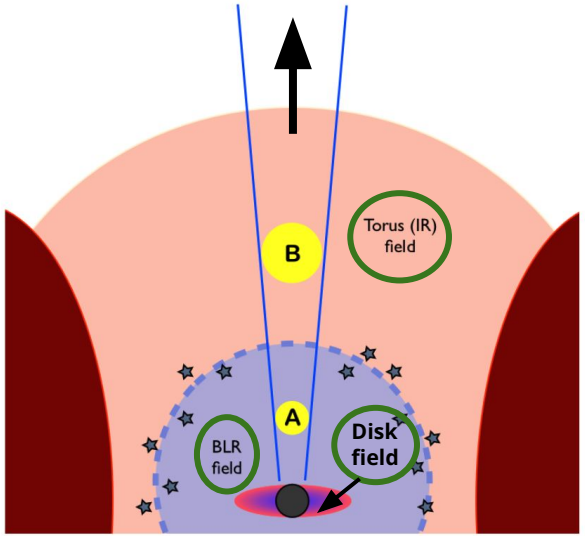
Inverse Compton scattering

Emission processes - simplified view

Main target photon fields:

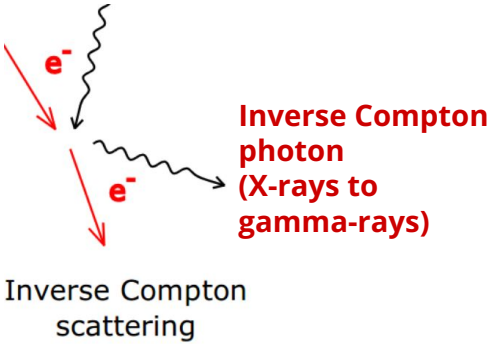
- **Disk:** ~UV thermal photons from accretion disk
  → due to doppler de-boosting important only if region close to BH
- **BLR lines:** reprocessing of disk UV photons by gas clouds in BLR regions (Lyα is the dominant line)
  → usually modelled as monochromatic fields
- **Torus:** IR photons emitted due to thermal excitation of dust clouds surrounding the disk
  → Typically modelled as a blackbody with T ~10^3 K

Bulk lorentz factor
$\Gamma \gtrsim 10$



Sketch credits: F. Tavecchio

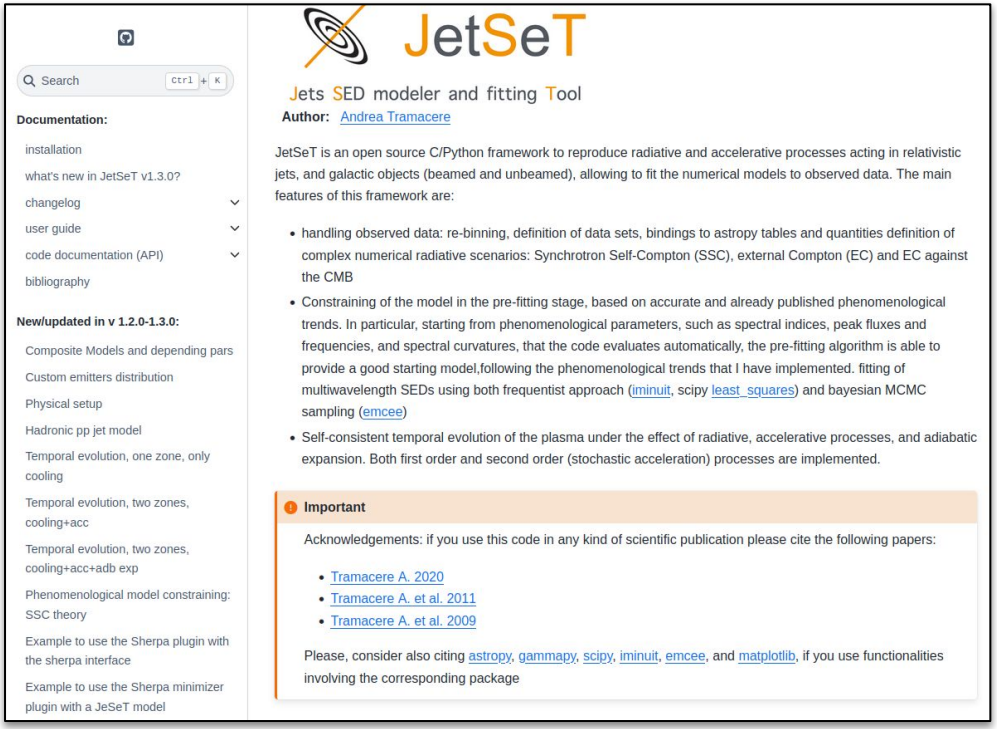
Synchrotron + BLR + Disk + Torus photon field



2. Open source modelling tools

JetSeT

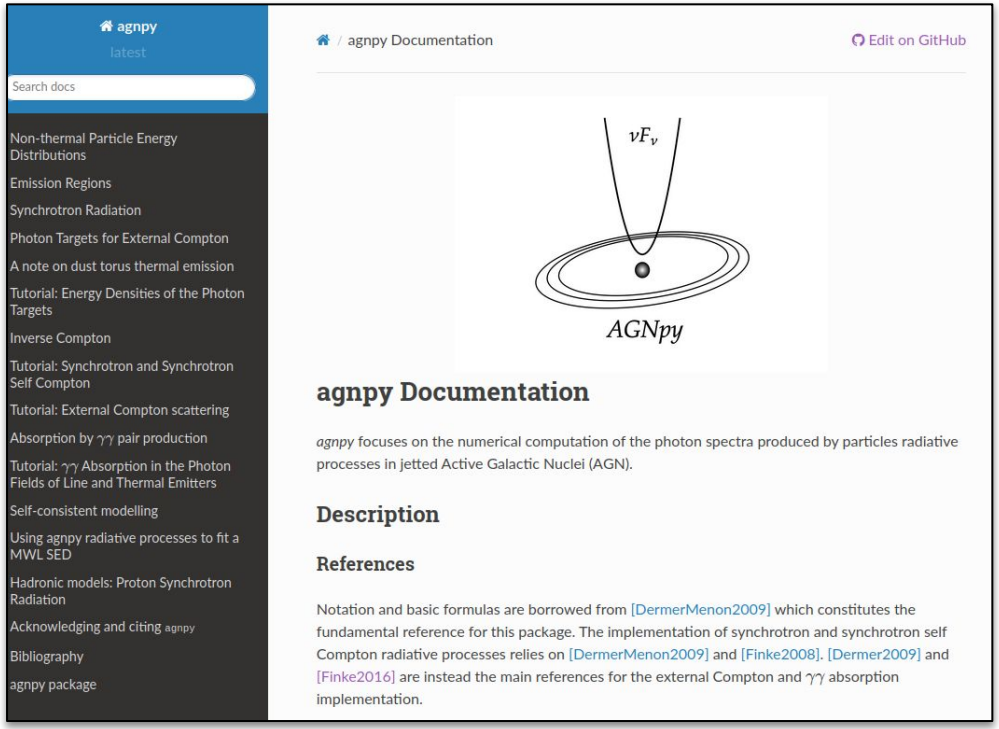
- Main developer: **Andrea Tramacere**
- **Python interface**
  (radiative processes computed in C)
- Include **SSC** and **EC** (on torus, BLR and disc fields)
  → suited for BL Lacs and FSRQs
- Include **fitting routines**
  (minuit/least squares, MCMC)
- **Self-consistent temporal evolution** of particles
  also implemented



GitHub: **https://github.com/andreatramacere/jetset**
Readthedocs: **https://jetset.readthedocs.io/en/1.3.0/**

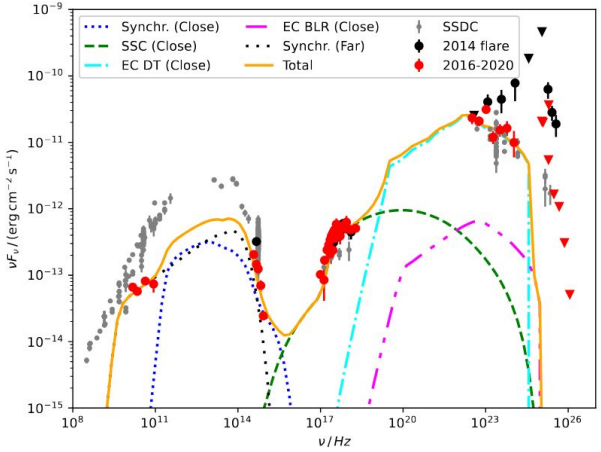
agnpy

- Main developer: **Cosimo Nigro**
- Fully written in **Python**
- Include **SSC** and **EC** (on torus, BLR and disc fields)
  → suited for BL Lacs and FSRQs
- Easy to **build wrapper** to **fit** model to data
  (e.g., using **sherpa** and **Gammapy**)
- **Self-consistent temporal evolution** of particles
  being implemented



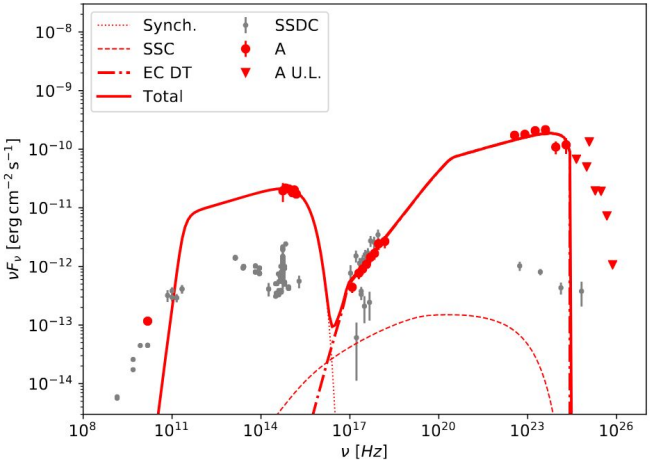
GitHub: **https://github.com/cosimoNigro/agnpy**
Readthedocs: **https://agnpy.readthedocs.io/en/latest/**

agnpy - some examples



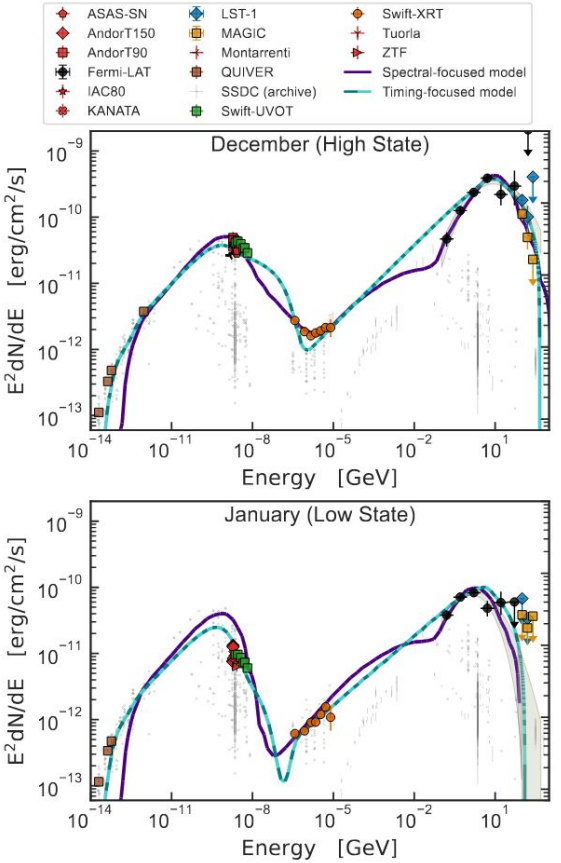
Gravitationally lensed blazar QSO B0218+357

(Acciari et al. 2022, MNRAS, 510, 2344)



Detection of QSO B1420+326

(Acciari et al. 2021, A&A, 647, A163)



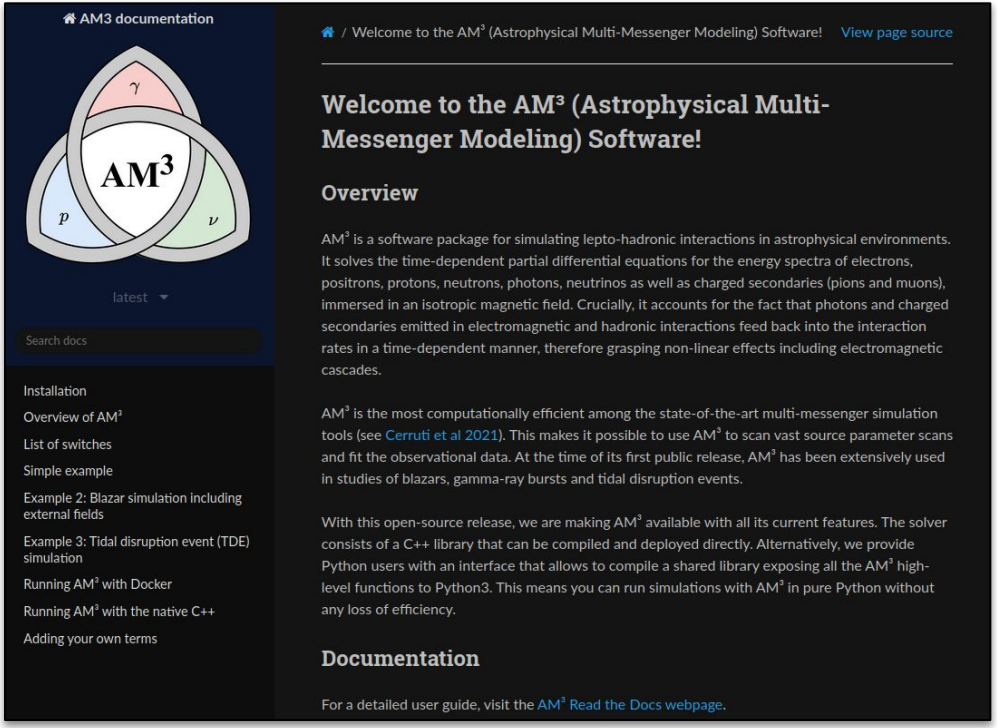
LST/MAGIC detection of OP313

(*In prep*, 2025)

AM³

- Developers: **Several people**
  Klinger et al. 2023 (arXiv:2312.13371).
- **Python with C++ backend**
- Include **SSC** and **EC**
  → suited for BL Lacs and FSRQs
- **Lepto-hadronic**
- **No fitting routines**
- **Solves time-dependent equations**
  for the energy spectra of electrons
  positrons, protons, neutrons, photons,
  neutrinos as well as charged secondaries

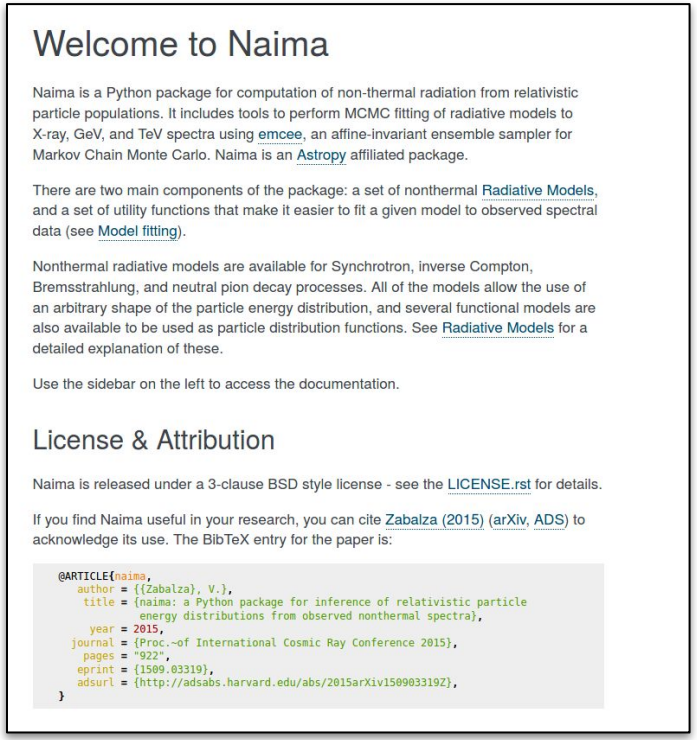
Among the most advanced tools to model the blazar SEDs



GitLab: **https://gitlab.desy.de/am3/am3**
Readthedocs: **https://am3.readthedocs.io/en/latest/index.html**

Naima
(the pioneer)

- Main developer: **Víctor Zabalza**
- Fully written in **Python**
- Developed for **galactic sources** (PWN, SNR, etc..)
  → Implement yourself relativistic corrections
  → Synchrotron self-abs. & internal pair prod.
  not implemented
- Compute **SSC**, and other processes
  (e.g., Bremsstrahlung, pion decay from p-p interactions,
  or IC on any photon field)
- I have made a notebook showing how to modify
  Naima to get a simple SSC model
  https://github.com/Axelarbetengels/Mrk421-2017-campaign-paper



Welcome to Naima

Naima is a Python package for computation of non-thermal radiation from relativistic particle populations. It includes tools to perform MCMC fitting of radiative models to X-ray, GeV, and TeV spectra using emcee, an affine-invariant ensemble sampler for Markov Chain Monte Carlo. Naima is an Astropy affiliated package.

There are two main components of the package: a set of nonthermal Radiative Models, and a set of utility functions that make it easier to fit a given model to observed spectral data (see Model fitting).

Nonthermal radiative models are available for Synchrotron, inverse Compton, Bremsstrahlung, and neutral pion decay processes. All of the models allow the use of an arbitrary shape of the particle energy distribution, and several functional models are also available to be used as particle distribution functions. See Radiative Models for a detailed explanation of these.

Use the sidebar on the left to access the documentation.

License & Attribution

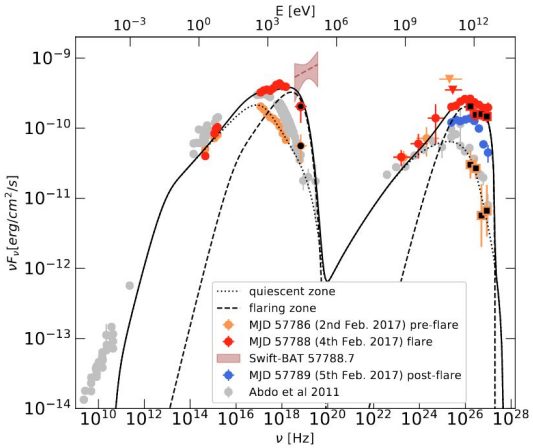
Naima is released under a 3-clause BSD style license - see the LICENSE.rst for details.

If you find Naima useful in your research, you can cite Zabalza (2015) (arXiv, ADS) to acknowledge its use. The BibTeX entry for the paper is:

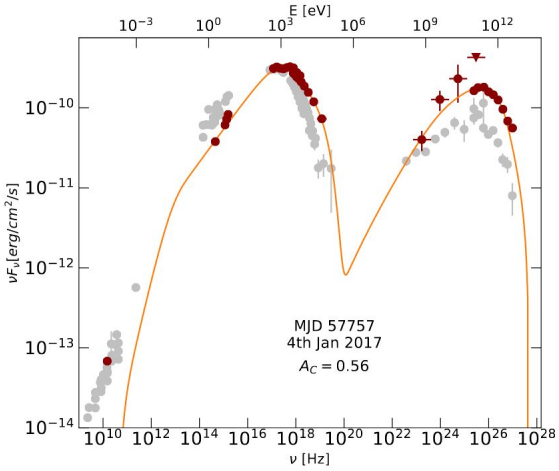
```
@ARTICLE{naima,
   author = {{Zabalza}, V.},
    title = {naima: a Python package for inference of relativistic particle
             energy distributions from observed nonthermal spectra},
     year = 2015,
  journal = {Proc.~of International Cosmic Ray Conference 2015},
    pages = "922",
   eprint = {1509.03319},
   adsurl = {http://adsabs.harvard.edu/abs/2015arXiv150903319Z},
}
```

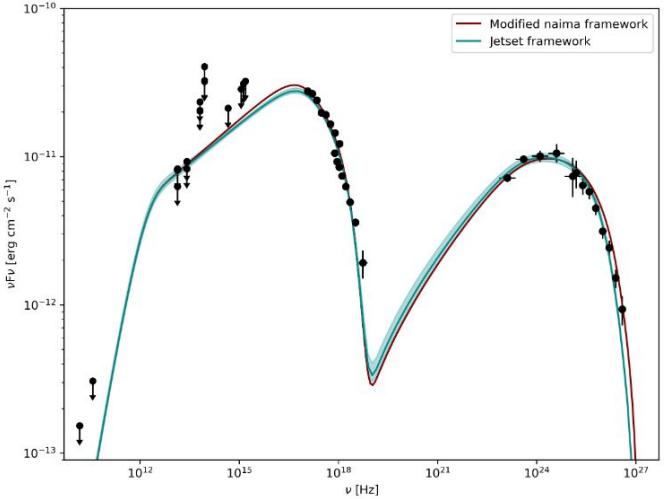
GitHub: **https://github.com/zblz/naima**
Readthedocs: **https://naima.readthedocs.io/en/latest/**

Naima - some examples



Mrk 421 in 2017 with *NuSTAR*
(Acciari et al. 2021, A&A, 655, A89)





(a) Low state with B=0.03

Mrk 501 in *low* X-ray/VHE state
(Abe et al. 2023, ApJS, 266, 37)

3. Introduction to JetSeT

JetSeT - Installation

With anaconda:

- create a virtual environment (not necessary, but suggested):

```
conda create --name jetset python=3.10 ipython jupyter
```

```
conda activate jetset
```

- install the code

```
conda install -c andreatramacere -c astropy -c conda-forge 'jetset>=1.3'
```

With PIP:

```
pip install jetset>=1.3
```

JetSeT - Basic Setup

- **Create** a `Jet` instance & select **electron distribution shape** (here broken power-law)

```
1  from jetset.jet_model import Jet
2
3  jet_env = Jet(name='My_AGN', electron_distribution='bkn')
```

JetSeT - Basic Setup

- **Create** a **`Jet`** instance & select **electron distribution shape** (here broken power-law)

```
1  from jetset.jet_model import Jet
2
3  jet_env = Jet(name='My_AGN', electron_distribution='bkn')
```

- **Define** the **`Jet`** environment

```
9  jet_env.set_par('z_cosm',val=0.03) # Source redshift
10
11 jet_env.set_par('B',val=6.1e-2) # Magnetic field
12 jet_env.set_par('R',val=1e16) # Blob radius (spherical geometry)
13 jet_env.set_par('beam_obj', val=25) # Beaming
14
```

JetSeT - Basic Setup

- **Create** a **`Jet`** instance & select **electron distribution shape** (here broken power-law)

```
from jetset.jet_model import Jet

jet_env = Jet(name='My_AGN', electron_distribution='bkn')
```

- **Define** the **`Jet`** environment

```
jet_env.set_par('z_cosm',val=0.03) # Source redshift

jet_env.set_par('B',val=6.1e-2) # Magnetic field
jet_env.set_par('R',val=1e16) # Blob radius (spherical geometry)
jet_env.set_par('beam_obj', val=25) # Beaming
```

- **Initialize electron distribution**

```
#Initialize electron distribution
jet_env.set_par('gmin',val=1e3) # minimum Lorentz factor
jet_env.set_par('gamma_break', val=2.1e5) # break Lorentz factor
jet_env.set_par('gmax',val=1.5e6) # maximum Lorentz factor
jet_env.set_par('p',val=2.2) # slope below break
jet_env.set_par('p_1',val=3.8) # slope above break
jet_env.set_par('N',val=3) # normalization

#Alternatively, set normalization from obs.
jet_env.set_N_from_nuFnu(nuFnu_obs=3.23e-10,nu_obs=2.6E17)
```

JetSeT - Basic Setup

- **Create** a **`Jet`** instance & select **electron distribution shape** (here broken power-law)

```
from jetset.jet_model import Jet

jet_env = Jet(name='My_AGN', electron_distribution='bkn')
```

- **Define** the **`Jet`** environment

```
jet_env.set_par('z_cosm',val=0.03) # Source redshift

jet_env.set_par('B',val=6.1e-2) # Magnetic field
jet_env.set_par('R',val=1e16) # Blob radius (spherical geometry)
jet_env.set_par('beam_obj', val=25) # Beaming
```

- **Initialize electron distribution**

```
#Initialize electron distribution
jet_env.set_par('gmin',val=1e3) # minimum Lorentz factor
jet_env.set_par('gamma_break', val=2.1e5) # break Lorentz factor
jet_env.set_par('gmax',val=1.5e6) # maximum Lorentz factor
jet_env.set_par('p',val=2.2) # slope below break
jet_env.set_par('p_1',val=3.8) # slope above break
jet_env.set_par('N',val=3) # normalization

#Alternatively, set normalization from obs.
jet_env.set_N_from_nuFnu(nuFnu_obs=3.23e-10,nu_obs=2.6E17)
```

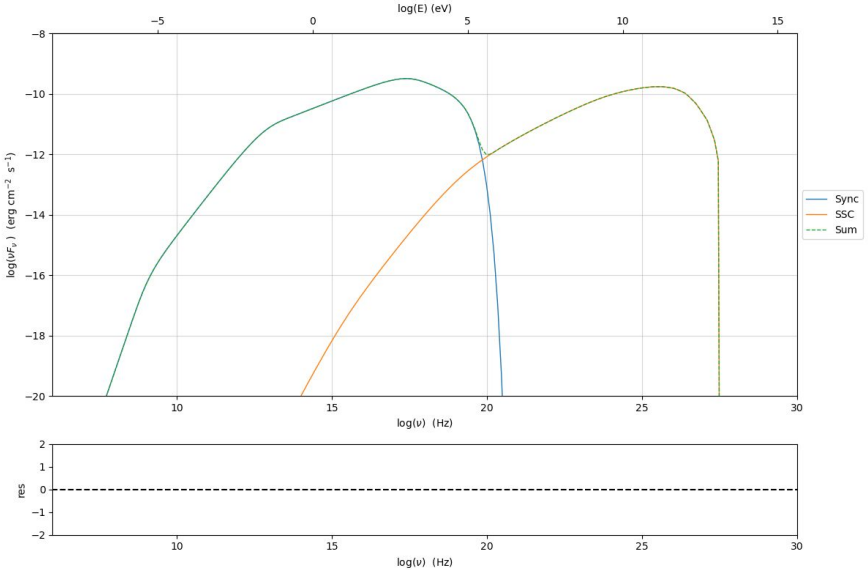
- **Compute the broadband emission**

```
jet_env.eval()
```

JetSeT - Basic Setup

- **Create** a **`Jet`** instance & select **electron distribution shape** (here broken power-law)
- **Define** the **`Jet`** environment
- **Initialize electron distribution**
- **Compute the broadband emission**

→ That's it!

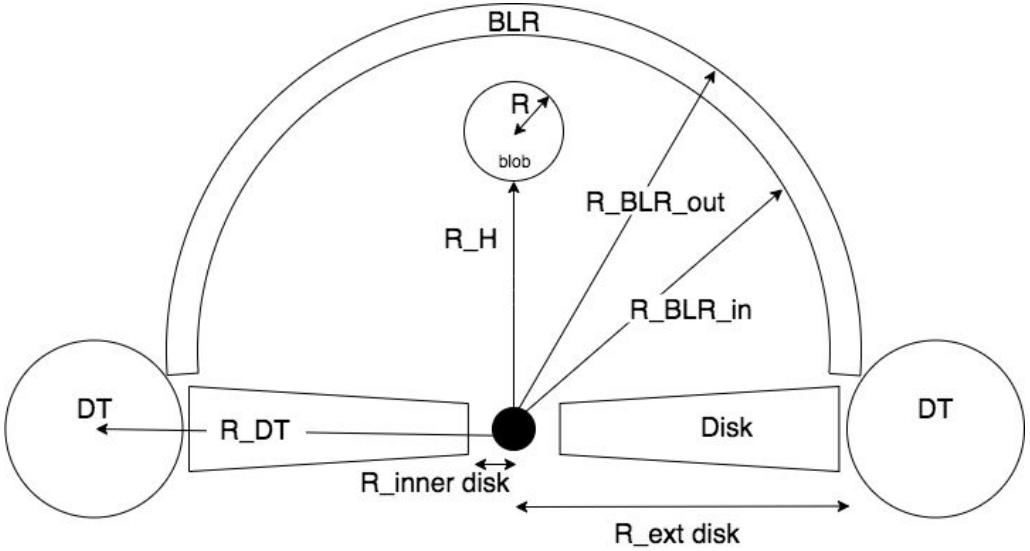



```
29
30  jet_env.eval()
31
32  jet_env.plot_model()
33
```

JetSeT - Basic Setup

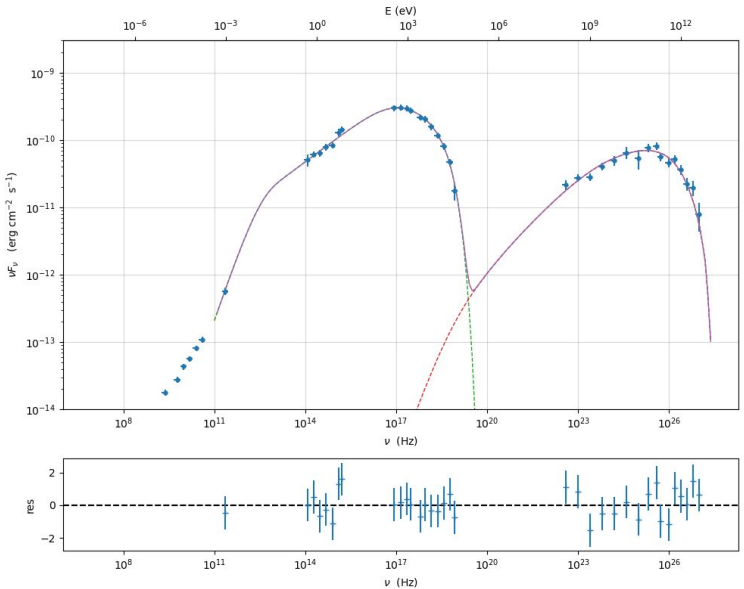
- For **FSRQs,** the **external photon field** (Disk, Torus, BLR) **easily configurable**

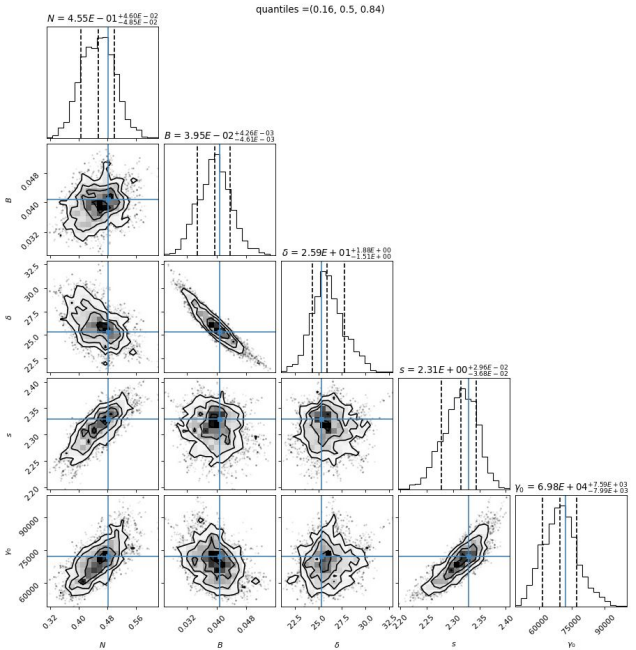
name	par type	units
gmin	low-energy-cut-off	lorentz-factor*
gmax	high-energy-cut-off	lorentz-factor*
N	emitters_density	1 / cm3
p	LE_spectral_slope	
p_1	HE_spectral_slope	
gamma_break	turn-over-energy	lorentz-factor*
R	region_size	cm
R_H	region_position	cm
B	magnetic_field	G
theta	jet-viewing-angle	deg
BulkFactor	jet-bulk-factor	Lorentz-factor*
z_cosm	redshift	
tau_BLR	BLR	
R_BLR_in	BLR	cm
R_BLR_out	BLR	cm
L_Disk	Disk	erg / s
T_Disk	Disk	K
T_DT	DT	K
R_DT	DT	cm
tau_DT	DT	



JetSeT - fitting tools

- Includes **fitting algorithms**
  - Frequentists approach (minuit, least-squares from scipy)
  - Bayesian approach (MCMC)



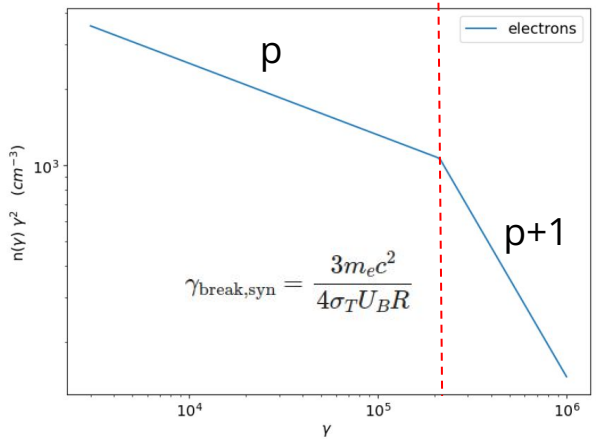


JetSeT - fitting tools

- **Fitted parameters** can be made **interdependent**
  - **Set physical constraints**,
    e.g. cooling break, emitting region radius

For example:

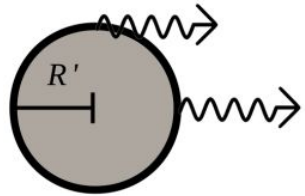
Make the particle distribution compatible with the equilibrium state



Make the radius of the emitting region compatible with the variability timescale

$$t'_{var} \gtrsim R'/c$$

$$\rightarrow \mathbf{R'} \lesssim \boldsymbol{\delta} \cdot c \cdot t_{var} \cdot (1+z)^{-1}$$



R' : emitting zone radius

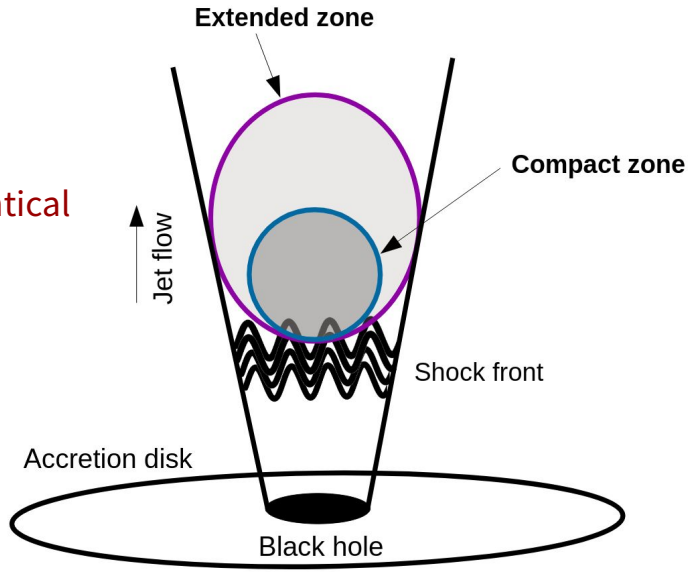
δ: doppler factor of emitting region
t_{var} : observed variability time scale

JetSeT - fitting tools

- **Fitted parameters** can be **linked** between emitting components

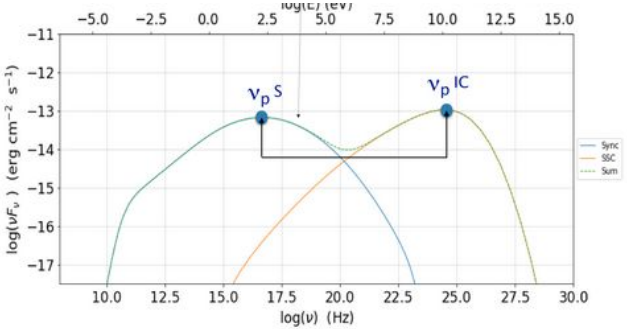
For example:

Keep the **Doppler factor** or
slope of radiating particles ~identical
for *nearby* regions



JetSeT - nice additional features

- Convenient **API to access** some **observables**
  - e.g. peak luminosities, energetics, etc…



$$\epsilon' = \frac{h\nu'}{m_e c^2}$$

$\epsilon' << m_e c^2$ **TH regime**

ν_p IC / ν_p S ~(4/3) γ_p^2

$\epsilon' \geq m_e c^2$ **KN regime**

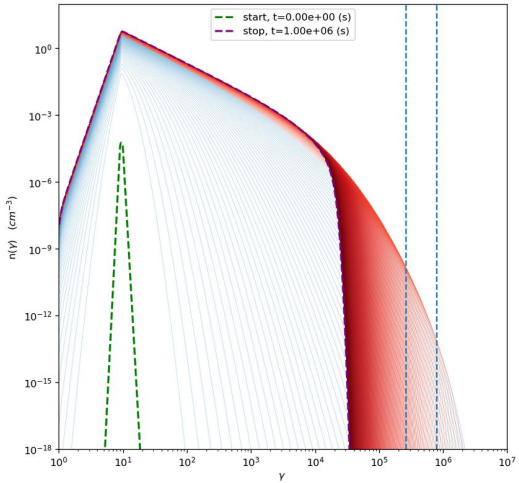
ν_p IC / ν_p S ~γ_p^{-1}

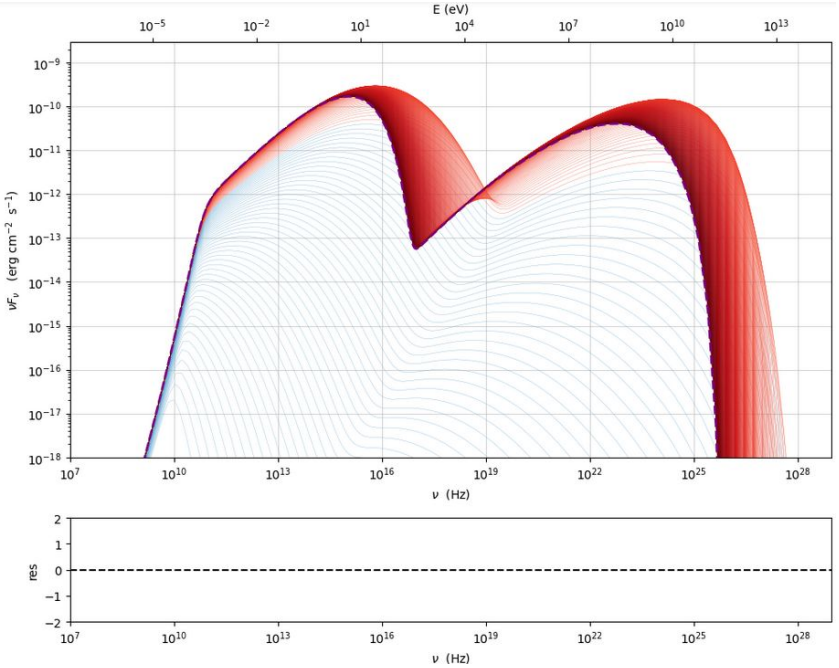
$h\nu_p$ IC~$m_e c^2 \gamma_p$

JetSeT - nice additional features

- Convenient **API to access** some **observables**
  - e.g. peak luminosities, energetics, etc…

- **Temporal evolution**, including:
  - Acceleration (Fermi I & II)
  - Cooling (synchrotron / inverse-Compton / adiabatic)
  - Particle escape





Very useful to simulate flares!

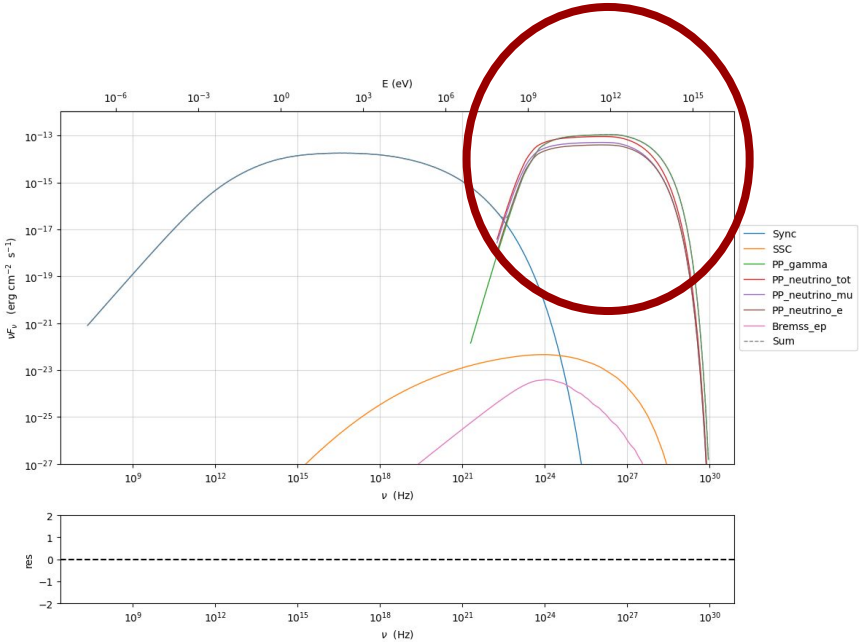
JetSeT - nice additional features

- Convenient **API to access** some **observables**
  - e.g. peak luminosities, energetics, etc…

- **Temporal evolution**, including:
  - Acceleration (Fermi I & II)
  - Cooling (synchrotron / inverse-Compton / adiabatic)
  - Particle escape

- Radiation processes suitable for **galactic sources**
  - p-p interaction (incl. neutrino output)
  - electron-ion Bremsstrahlung, etc…

“Pion bump”



JetSeT - Hands on session

- We will go together through **notebooks** presenting the basic functionalities of JetSeT
- The idea is to give examples **complemented with some physical insights**
  → basis to develop “meaningful” models describing your data
- Visit **https://github.com/Axelarbetengels/JetSeT_notebooks_SextenWorkshop to access the notebooks**