

Sexten Sundial, Konrad Petrides, ~ 1900

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





(Blazar) Observations

Theory & Models

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

Data / model

comparisons

Data/model comparison getting "easier"

Observational improvements



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





→ Synchrotron self-Compton model (SSC)

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

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

→ ~7 free parameters





→ Synchrotron self-Compton model (SSC)

Sketch credits: F. Tavecchio

"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)

 \rightarrow ~7 free parameters





 α_{2}

 $+T_{var}$ (constraints on R)

 v_{s}, L_{s}

 α_1

Fv [erg cm⁻²

10

10

10-1

 $v_{\rm IC}, L_{\rm IC}$

 \rightarrow Synchrotron self-Compton model (SSC)





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

Energy dependency of the polarization → complex geometry

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





10-9

Epoch B

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 ~10³ K



Sketch credits: F. Tavecchio

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) \rightarrow suited for BL Lacs and FSRQs
- Include **fitting routines** (minuit/least squares, MCMC)
- Self-consistent temporal evolution of particles also implemented

P		S J
Q Search	Ctrl+K	lets SFD modele
Documentation:		Author: Andrea Tramad
installation		JetSeT is an open source
what's new in JetSeT v1	.3.0?	jets, and galactic objects (
changelog	~	features of this framework
user guide	~	 handling observed date
code documentation (AF	PI) ~	complex numerical ra
bibliography		the CMB
New/updated in v 1.2.0-1.3.0:		 Constraining of the m trends. In particular, s
Composite Models and o	provide a good startin	
Custom emitters distribu		
Physical setup		sampling (emcee)
Hadronic pp jet model		 Self-consistent tempo
Temporal evolution, one cooling	zone, only	expansion. Both first
Temporal evolution, two cooling+acc	zones,	Important
Temporal evolution, two zones.		Acknowledgements:
cooling+acc+adb exp		Tramacoro A 20
Phenomenological mode	el constraining:	Tramacere A. 20

SSC theory

the sherpa interface

plugin with a JeSeT model

Example to use the Sherpa plugin with

Example to use the Sherpa minimizer

letSeT er and fitting Tool cere

C/Python framework to reproduce radiative and accelerative processes acting in relativistic (beamed and unbeamed), allowing to fit the numerical models to observed data. The main are:

- ata: re-binning, definition of data sets, bindings to astropy tables and quantities definition of idiative scenarios: Synchrotron Self-Compton (SSC), external Compton (EC) and EC against
- odel in the pre-fitting stage, based on accurate and already published phenomenological starting from phenomenological parameters, such as spectral indices, peak fluxes and ctral curvatures, that the code evaluates automatically, the pre-fitting algorithm is able to ng model.following the phenomenological trends that I have implemented, fitting of s using both frequentist approach (iminuit, scipy least_squares) and bayesian MCMC
- oral evolution of the plasma under the effect of radiative, accelerative processes, and adiabatic order and second order (stochastic acceleration) processes are implemented.

if you use this code in any kind of scientific publication please cite the following papers:

- t al. 2011
- Tramacere A. et al. 2009

Please, consider also citing astropy, gammapy, scipy, iminuit, emcee, and matplotlib, if you use functionalities involving the corresponding package

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

LST/MAGIC detection of OP313

ASAS-SN

AndorT90

AndorT150

LST-1

MAGIC

Montarrenti

Swift-XRT

Tuorla

ZTF

(In prep, 2025)

 10^{1}

 10^{1}

AM³

- Developers: **Several people** <u>Klinger et al. 2023 (arXiv:2312.13371)</u>.
- 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



latest

Installation Overview of AM³ List of switches Simple example Example 2: Blazar simulation including external fields Example 3: Tidal disruption event (TDE) simulation Running AM³ with Docker Running AM³ with the native C++ Adding your own terms

₭ / Welcome to the AM³ (Astrophysical Multi-Messenger Modeling) Software! View page source

Welcome to the AM³ (Astrophysical Multi-Messenger Modeling) Software!

Overview

AM³ is a software package for simulating lepto-hadronic interactions in astrophysical environments. It solves the time-dependent partial differential equations for the energy spectra of electrons, positrons, protons, neutrons, photons, neutrinos as well as charged secondaries (pions and muons), immersed in an isotropic magnetic field. Crucially, it accounts for the fact that photons and charged secondaries emitted in electromagnetic and hadronic interactions feed back into the interaction rates in a time-dependent manner, therefore grasping non-linear effects including electromagnetic cascades.

AM³ is the most computationally efficient among the state-of-the-art multi-messenger simulation tools (see Cerruti et al 2021). This makes it possible to use AM³ to scan vast source parameter scans and fit the observational data. At the time of its first public release, AM³ has been extensively used in studies of blazars, gamma-ray bursts and tidal disruption events.

With this open-source release, we are making AM³ available with all its current features. The solver consists of a C++ library that can be compiled and deployed directly. Alternatively, we provide Python users with an interface that allows to compile a shared library exposing all the AM³ highlevel functions to Python3. This means you can run simulations with AM³ in pure Python without any loss of efficiency.

Documentation

For a detailed user guide, visit the AM³ Read the Docs webpage.

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 <u>https://github.com/Axelarbetengels/Mrk421-2017-campaign-paper</u>

Welcome to Naima

Naima is a Python package for computation of non-thermal radiation from relativistic particle populations. It includes tools to perform MOMC fitting of radiative models to X-ray, GeV, and TeV spectra using <u>emcee</u>, 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 <u>Radiative Models</u>, 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 <u>Radiative Models</u> 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:



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

Naima - some examples



3. Introduction to JetSeT



Jets SED modeler and fitting Tool

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

• **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')

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

from jetset.jet_model import Jet

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

<pre>jet_env.set_par('z_cosm',val=0.03) # Source redshift</pre>
<pre>jet_env.set_par('B',val=6.le-2) # Magnetic field jet_env.set_par('R',val=1e16) # Blob radius (spherical geometry) jet_env.set_par('beam_obj', val=25) # Beaming</pre>

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

• Initialize electron distribution

from jetset.jet_model import Jet

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

9 10 11 12 13 14	<pre>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</pre>
16 17 18 19 20 21 22 23	<pre>#Initialize electron distribution jet_env.set_par('gmin',val=le3) # 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</pre>
24 25	<pre>#Alternatively, set normalization from obs. jet_env.set_N_from_nuFnu(nuFnu_obs=3.23e-10,nu_obs=2.6E17)</pre>

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

- Initialize electron distribution
- Compute the broadband emission

from jetset.jet_model import Jet

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

	<pre>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</pre>
16 17 18 19 20 21 22	<pre>#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</pre>
24	<pre>#Alternatively, set normalization from obs. jet_env.set_N_from_nuFnu(nuFnu_obs=3.23e-10,nu_obs=2.6E17)</pre>

jet env.eval()

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

- Initialize electron distribution
- Compute the broadband emission

 \rightarrow That's it!



• 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
р	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
	DT	



JetSeT - fitting tools

- Includes fitting algorithms
 - <u>Frequentists approach</u> (minuit, least-squares from scipy)
 - <u>Bayesian approach</u> (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

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

$$\rightarrow \mathbf{R}' \lesssim \mathbf{\delta} \cdot \mathbf{c} \cdot \mathbf{t}_{var} \cdot (1+z)^{-1}$$



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



JetSeT - nice additional features

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



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

"Pion bump"

- 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...



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