JetSeT

Jets SED modeler and fitting Tool

Andrea Tramacere

https://jetset.readthedocs.io/en/latest/
https://github.com/andreatramacere/jetset
https://www.facebook.com/jetsetastro/

Numerical modeling meeting Paris 21/02/2024

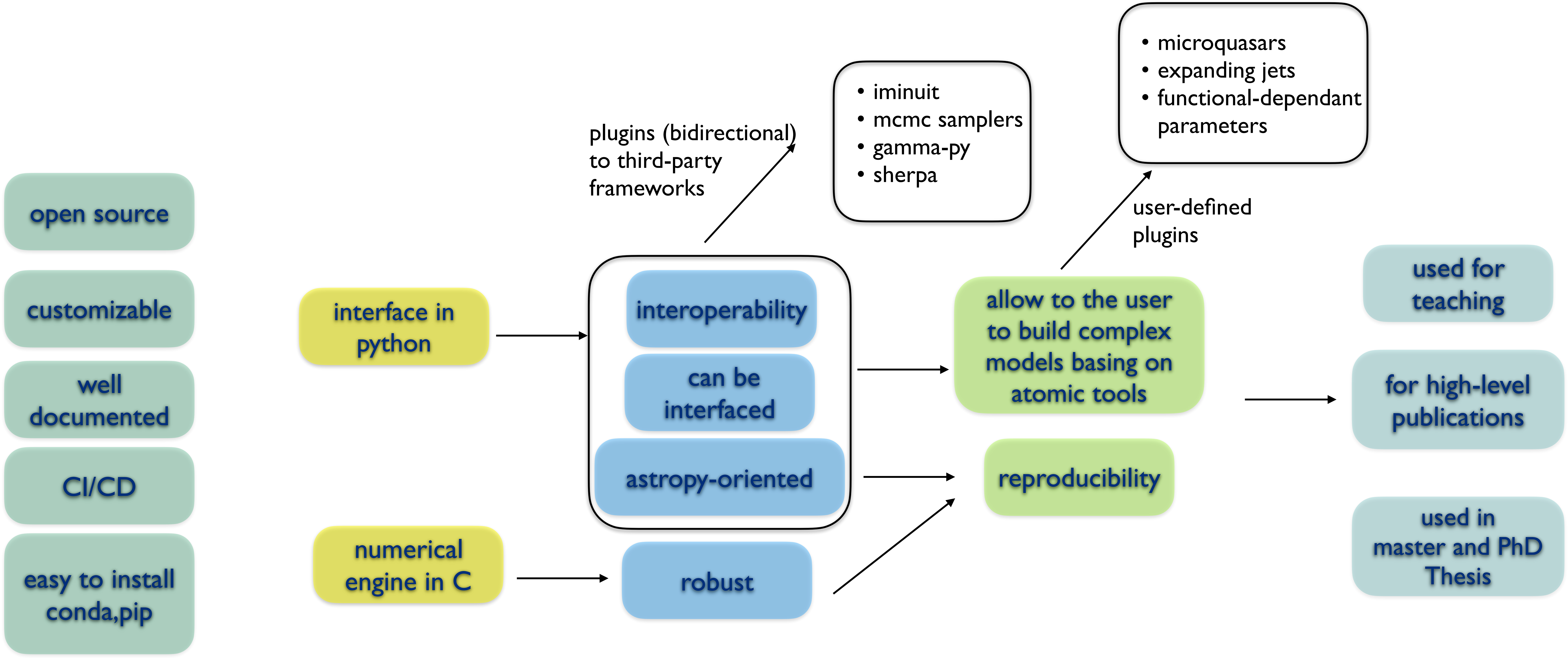
outline

- Philosophy and architecture of Jetset
- Current features and next developments
- Recent publications done with Jetset
- Conclusions

JetSeT

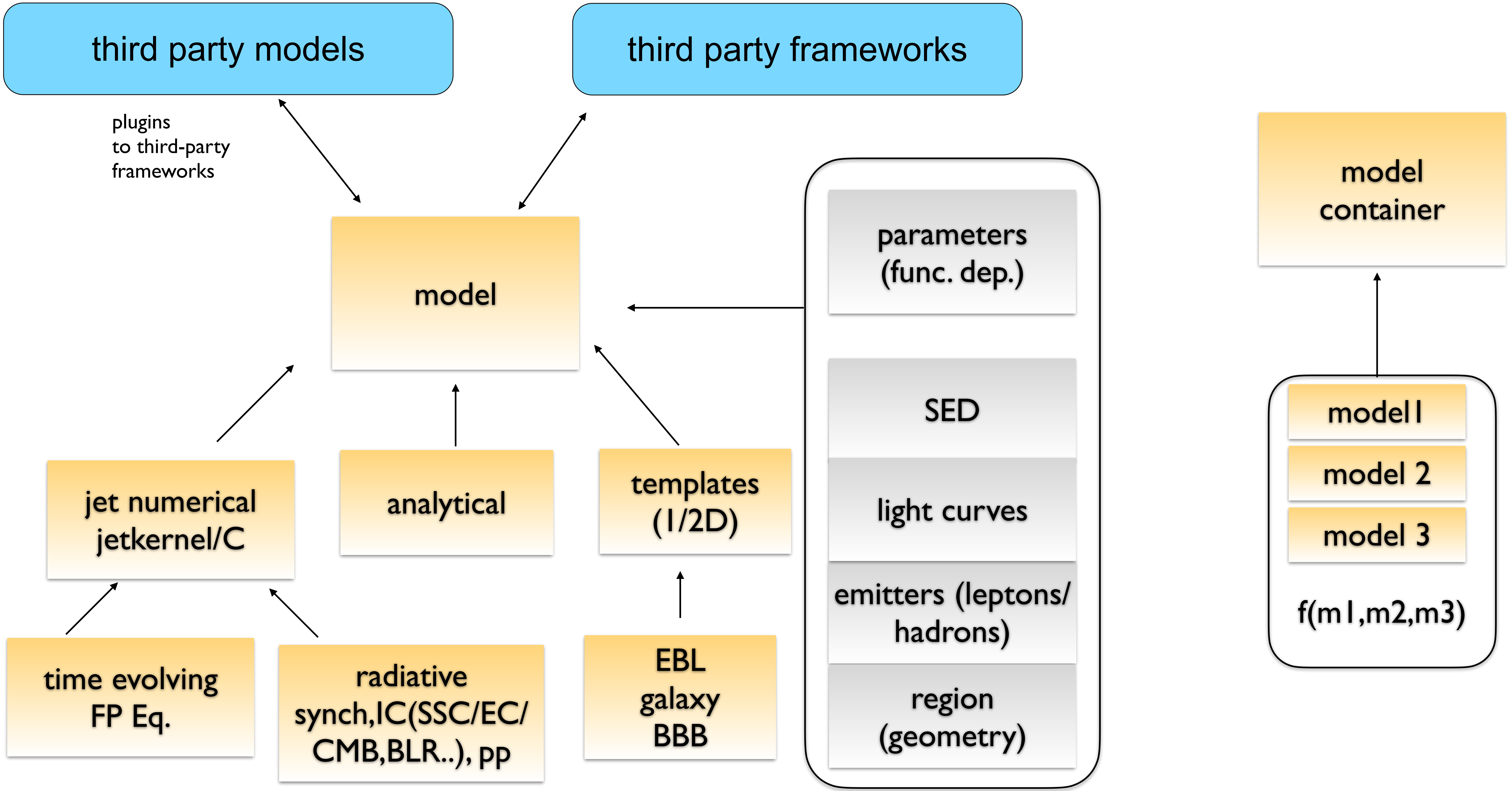
the philosophy

JetSeT is an open source C/Python framework to reproduce radiative and accelerative processes acting in relativistic jets, and galactic objects (with/without jet), allowing to fit the numerical models to observed data.



JetSeT

the architecture (obj oriented)



releases

- the code can easily installed using pip or conda (single line instruction)
- easily recompiled cloning the repo (just run setup.py)
- on git I provide both releases of sources and binaries

Release 1.2.2 Latest

Compare

github-actions released this Sep 15, 2023 · 54 commits to master since this release 1.2.2 6fa9445

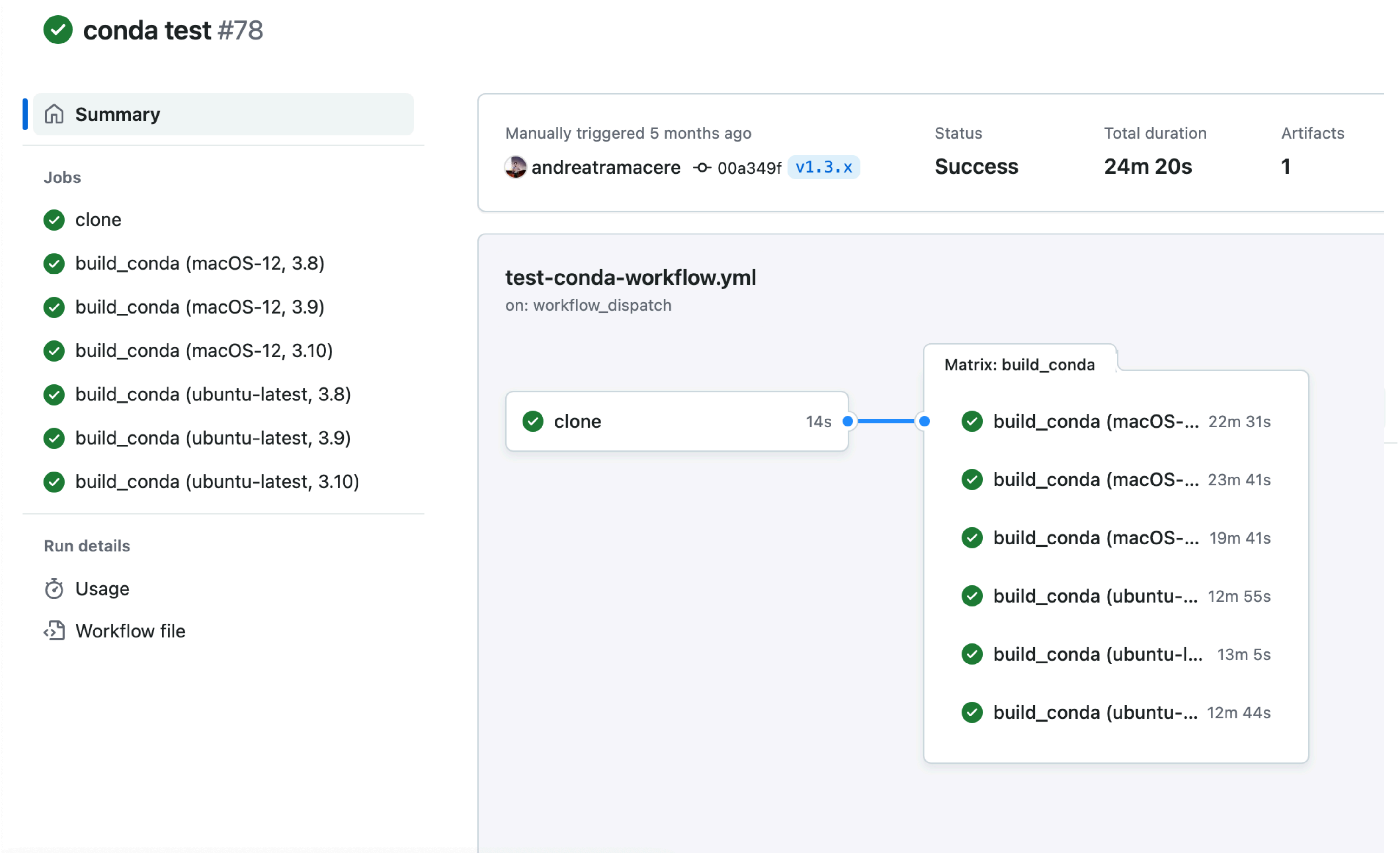
1.2.2

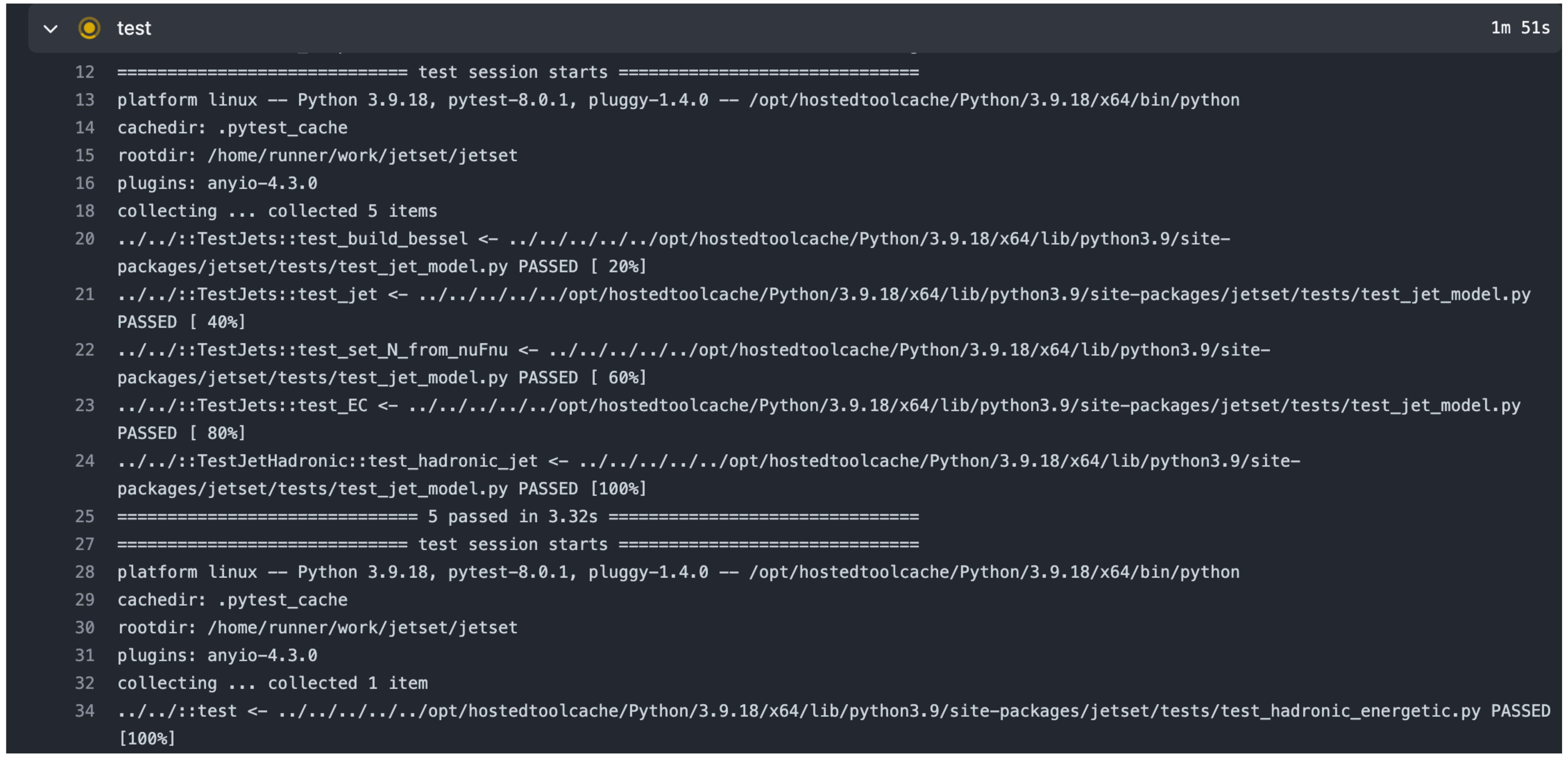
Assets 16

conda-binary-1.2.2-macos-10.15-py-3.10.tar	1.32 MB	Jun 6, 2022
conda-binary-1.2.2-macos-10.15-py-3.8.tar	1.31 MB	Jun 6, 2022
conda-binary-1.2.2-macos-10.15-py-3.9.tar	1.32 MB	Jun 6, 2022
conda-binary-1.2.2-ubuntu-latest-py-3.10.tar	1.32 MB	Jun 6, 2022
conda-binary-1.2.2-ubuntu-latest-py-3.8.tar	1.31 MB	Jun 6, 2022
conda-binary-1.2.2-ubuntu-latest-py-3.9.tar	1.32 MB	Jun 6, 2022
pip-binary-1.2.2-macos-10.15-py-3.7.tar	1.25 MB	Jun 6, 2022
pip-binary-1.2.2-macos-10.15-py-3.8.tar	1.25 MB	Jun 6, 2022
pip-binary-1.2.2-macos-10.15-py-3.9.tar	1.25 MB	Jun 6, 2022
pip-binary-1.2.2-ubuntu-latest-py-3.7.tar	2.27 MB	Jun 6, 2022
pip-binary-1.2.2-ubuntu-latest-py-3.8.tar	2.28 MB	Jun 6, 2022
pip-binary-1.2.2-ubuntu-latest-py-3.9.tar	2.31 MB	Jun 6, 2022
pip-src-1.2.2-macos-10.15.tar	1.03 MB	Jun 6, 2022
pip-src-1.2.2-ubuntu-latest.tar	1.04 MB	Jun 6, 2022
Source code (zip)		Sep 7, 2023
Source code (tar.gz)		Sep 7, 2023

releases

- the code is built for various linux/mac os and three different python version
- linux binaries are built on many_linux distro to ensure no issues when in installed on any linux distro
- git Actions pipelines are used for CI/CD with extensive testing of the code





an example of workflow

interoperability

- You can both turn your model into a sherpa or gamma-py model,
- our just use the sherpa or gamma-py (or iminuit or emcee) fitting frontend!
- with sherpa and gamma-py you can do forward-folded model fitting taking into account responses of different datasets

JetSeT

interoperability: using model fitting plugins is trivial

Sherpa

```
sherpa_model_jet=JetsetSherpaModel(prefit_jet)
sherpa_model_gal=JetsetSherpaModel(my_shape.host_gal)
sherpa_model_ebl=JetsetSherpaModel(ebl_franceschini)
```

```
jetset model name R renamed to  R_sh due to sherpa internal naming convention
```

```
sherpa_model=(sherpa_model_jet+sherpa_model_gal)*sherpa_model_ebl
```

```
sherpa_model
```

▼ Model

Expression: (jet_leptonic + host_galaxy) * Franceschini_2008

Component	Parameter	Thawed	Value	Min	Max	Units
jet_leptonic	gmin	☑	470.39174855643597	1.0	1000000000.0	lorentz-factor*
	gmax	☑	2310708.197406515	1.0	1000000000000000.0	lorentz-factor*
	N	☑	7.087120469822453	0.0	MAX	1 / cm3
	gamma0_log_parab	☑	10458.36315393129	1.0	1000000000.0	lorentz-factor*
	s	☑	2.2487867709713574	-10.0	10.0	
	r	☑	0.320557142636666	-15.0	15.0	
	R_sh	☑	1.0569580559768326e+16	1000.0	1e+30	cm
	R_H	☑	1e+17	0.0	MAX	cm
	B	☑	0.0505	0.0	MAX	gauss
	beam_obj	☑	25.0	0.0001	MAX	lorentz-factor*
	z_cosm	☑	0.0336	0.0	MAX	
host_galaxy	nuFnu_p_host	☑	-10.062787651081644	-12.254122641095535	-8.254122641095535	erg / (cm2 s)
	nu_scale	☑	0.017307503006438463	-0.5	0.5	Hz
Franceschini_2008	z_cosm	☑	0.0336	0.0	MAX	

```
sherpa_model_ebl.z_cosm  = sherpa_model_jet.z_cosm
```

γπ

```
gammapy_jet_model=GammapyJetsetModelFactory(jet)
gammapy_jet_model.parameters.to_table()
```

Table length=9

type	name	value	unit	error	min	max	frozen	link
str8	str9	float64	str4	int64	float64	float64	bool	str1
spectral	gmin	2.0000e+00		0.000e+00	1.000e+00	1.000e+09	False	
spectral	gmax	1.0000e+06		0.000e+00	1.000e+00	1.000e+15	False	
spectral	N	1.0000e+02	cm-3	0.000e+00	0.000e+00	nan	False	
spectral	gamma_cut	1.0000e+04		0.000e+00	1.000e+00	1.000e+09	False	
spectral	R	5.0000e+15	cm	0.000e+00	1.000e+03	1.000e+30	False	
spectral	R_H	1.0000e+17	cm	0.000e+00	0.000e+00	nan	True	
spectral	B	1.0000e-01	G	0.000e+00	0.000e+00	nan	False	
spectral	beam_obj	1.0000e+01		0.000e+00	1.000e-04	nan	False	
spectral	z_cosm	1.0000e-01		0.000e+00	0.000e+00	nan	False	

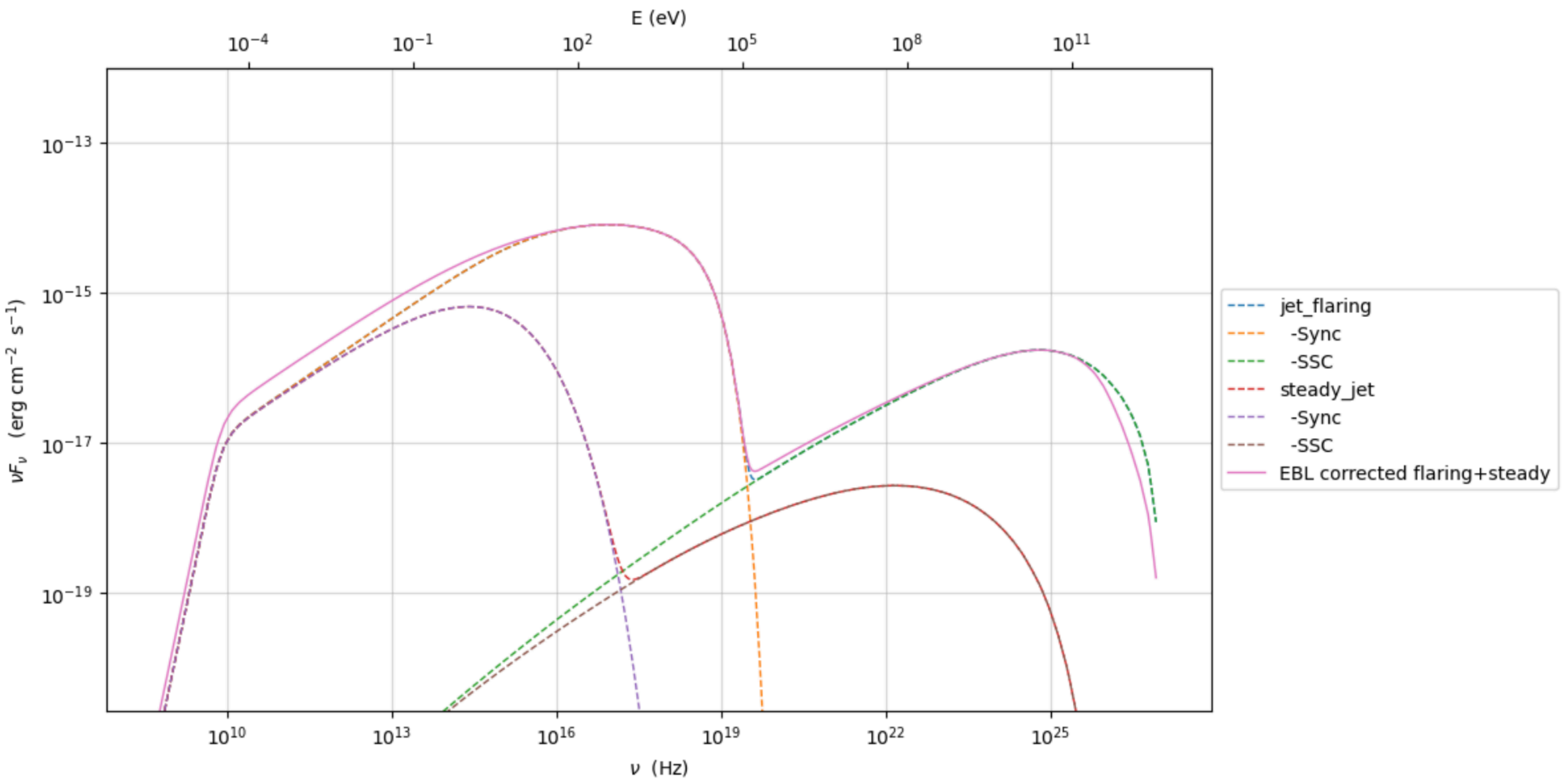
```
print(gammapy_jet_model)
```

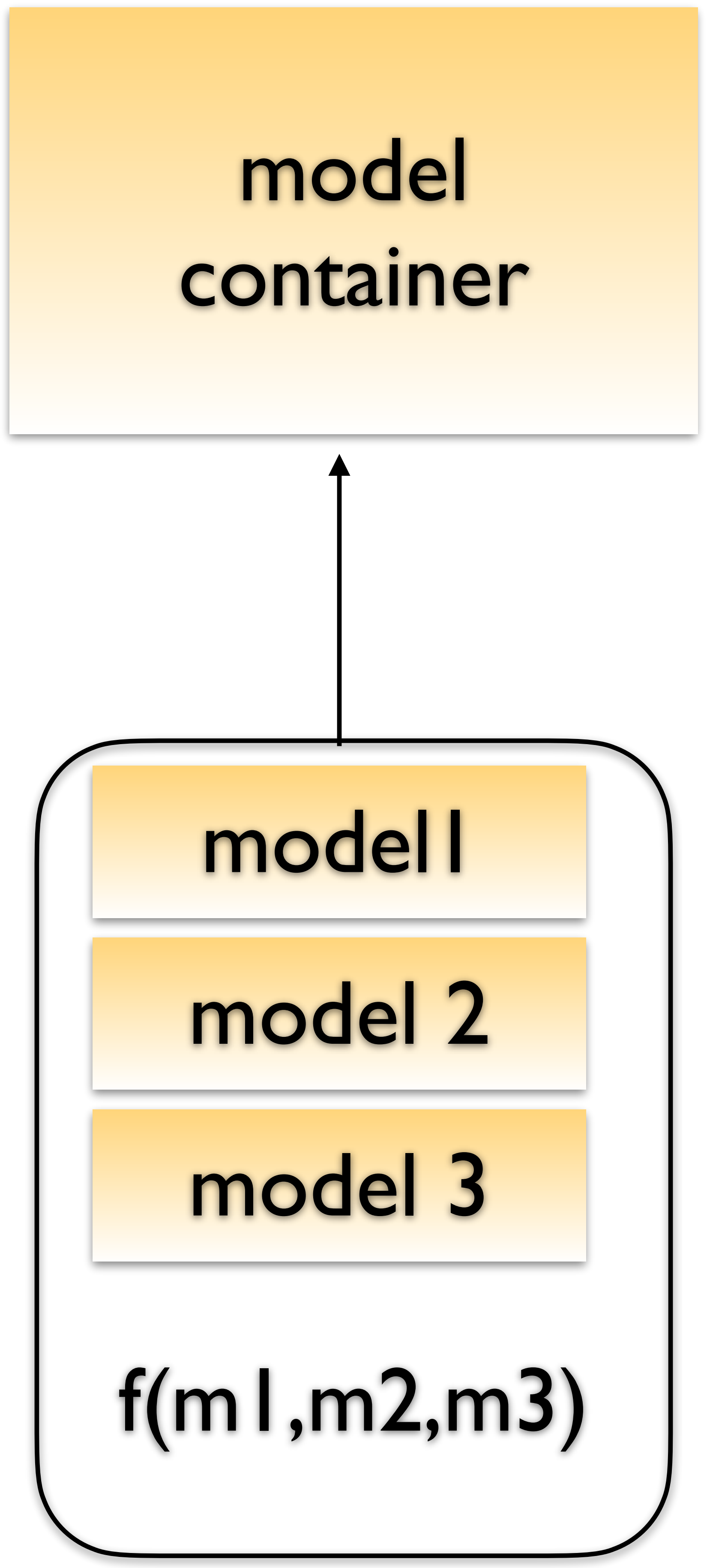
```
GammapyJetsetModel

  type      name     value    unit   error      min       max    frozen link
-------- --------- ---------- ---- --------- --------- --------- ------ ----
spectral      gmin 2.0000e+00      0.000e+00 1.000e+00 1.000e+09  False
spectral      gmax 1.0000e+06      0.000e+00 1.000e+00 1.000e+15  False
spectral         N 1.0000e+02 cm-3 0.000e+00 0.000e+00       nan  False
spectral gamma_cut 1.0000e+04      0.000e+00 1.000e+00 1.000e+09  False
spectral         R 5.0000e+15   cm 0.000e+00 1.000e+03 1.000e+30  False
spectral       R_H 1.0000e+17   cm 0.000e+00 0.000e+00       nan   True
spectral         B 1.0000e-01    G 0.000e+00 0.000e+00       nan  False
spectral  beam_obj 1.0000e+01      0.000e+00 1.000e-04       nan  False
spectral    z_cosm 1.0000e-01      0.000e+00 0.000e+00       nan  False
```

User customization

- model con be easily combined using math expression

```
composite_model.composite_expr='(jet_flaring + steady_jet) * Franceschini_2008'
```



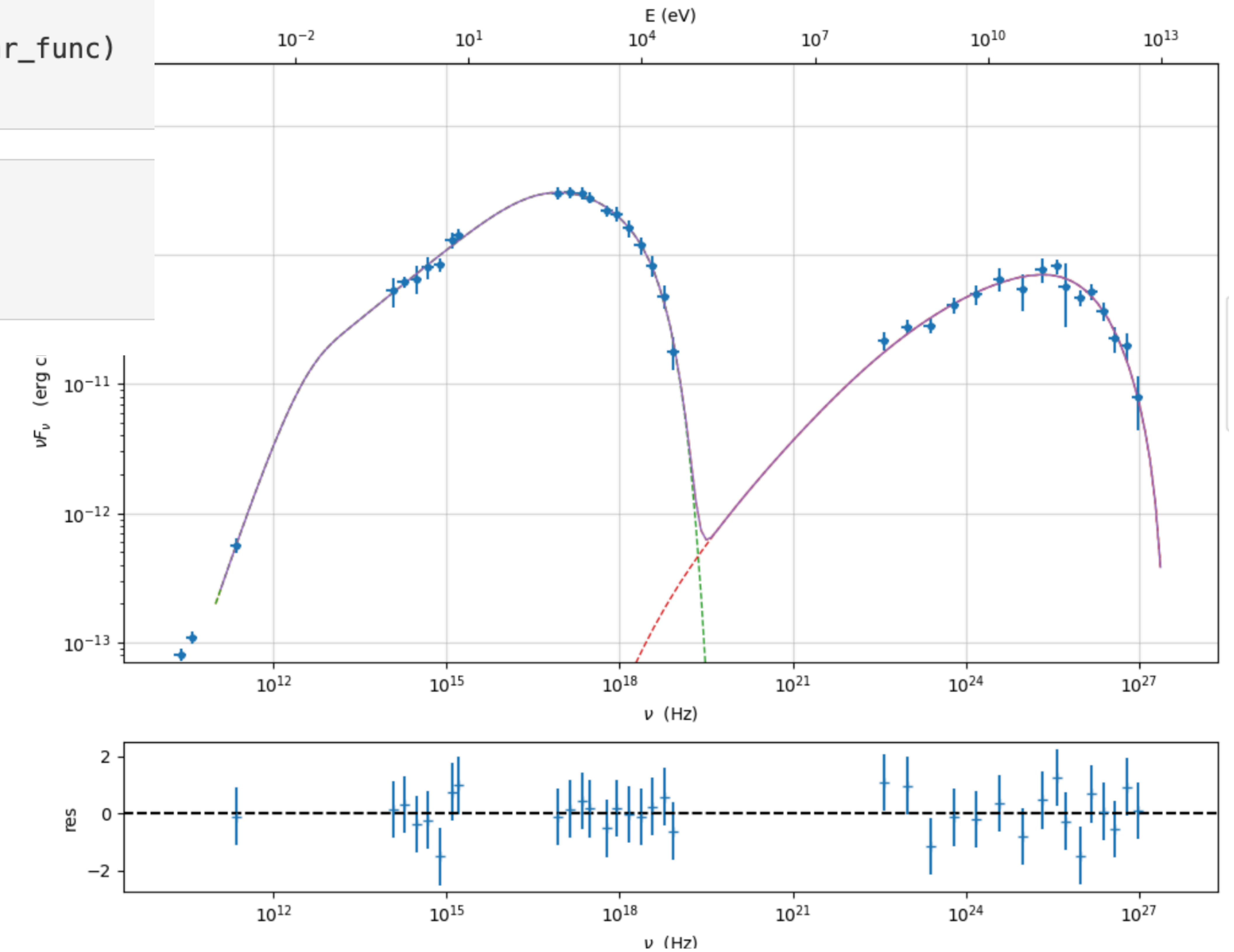


User customization

- parameters can be easily linked with functions

```
fit_model.jet_leptonic.add_user_par(name='B0',units='G',val=1E3,val_min=0,val_max=None)
fit_model.jet_leptonic.add_user_par(name='R0', units='cm', val=5E13, val_min=0, val_max=None)
fit_model.jet_leptonic.add_user_par(name='m_B', val=1, val_min=1, val_max=2)
fit_model.jet_leptonic.parameters.R0.frozen=True
fit_model.jet_leptonic.parameters.B0.frozen=True

def par_func(R0,B0,R_H,m_B):
    return B0*np.power((R0/R_H),m_B)

fit_model.jet_leptonic.make_dependent_par(par='B', depends_on=['B0', 'R0', 'R_H','m_B'], par_expr=par_func)
fit_model.parameters
```

```
==> par B is now depending on ['B0', 'R0', 'R_H', 'm_B'] according to expr:B =
def par_func(R0,B0,R_H,m_B):
    return B0*np.power((R0/R_H),m_B)
```



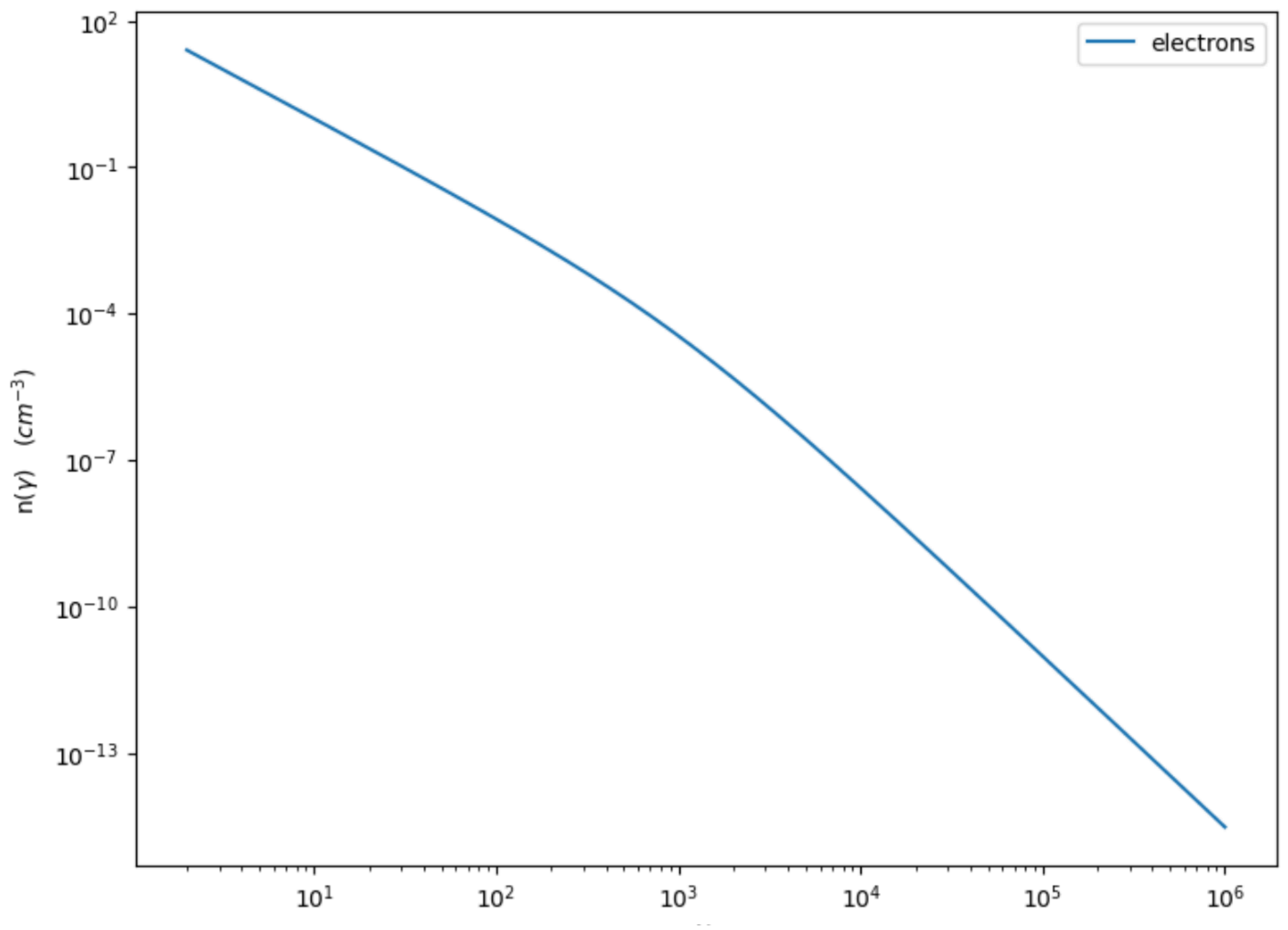
User customization

- users can easily define custom distribution for electrons and protons

```
def distr_func_bkn(gamma_break,gamma,s1,s2):
    return np.power(gamma,-s1)*(1.+(gamma/gamma_break))**(-(s2-s1))

n_e_bkn=EmittersDistribution('bkn',spectral_type='bkn')
n_e_bkn.add_par('gamma_break',par_type='turn-over-energy',val=1E3,vmin=1., vmax=None, unit='lorentz-factor')
n_e_bkn.add_par('s1',par_type='LE_spectral_slope',val=2.5,vmin=-10., vmax=10, unit='')
n_e_bkn.add_par('s2',par_type='HE_spectral_slope',val=3.2,vmin=-10., vmax=10, unit='')
n_e_bkn.set_distr_func(distr_func_bkn)
n_e_bkn.parameters.show_pars()
n_e_bkn.parameters.s1.val=2.0
n_e_bkn.parameters.s2.val=3.5
p=n_e_bkn.plot()
```

Table length=6

name	par type	units	val	phys. bound. min	phys. bound. max	log	frozen
gmin	low-energy-cut-off	lorentz-factor*	2.000000e+00	1.000000e+00	1.000000e+09	False	False
gmax	high-energy-cut-off	lorentz-factor*	1.000000e+06	1.000000e+00	1.000000e+15	False	False
N	emitters_density	1 / cm3	1.000000e+02	0.000000e+00	--	False	False
gamma_break	turn-over-energy	lorentz-factor*	1.000000e+03	1.000000e+00	--	False	False
s1	LE_spectral_slope		2.500000e+00	-1.000000e+01	1.000000e+01	False	False
s2	HE_spectral_slope		3.200000e+00	-1.000000e+01	1.000000e+01	False	False



The FP approach: Numerically solved Chang&Cooper

JetSeT

here we add adiabatic cooling
(t=time elapsed from the expansion)

$$|\dot{\gamma}_{ad}| = \frac{1}{3}\frac{\dot{V}}{V}\gamma = \frac{\dot{R}(t)}{R(t)}\gamma = \frac{\beta_{\exp} c}{R(t)}\gamma$$

Parker 2006, Stawartz&Petrosian 2008 Tramacere+2011

JetSeT

injection term

$$L_{inj} = V_{\rm acc} \int \gamma m_e c^2 Q(\gamma,t) d\gamma \quad (erg/s)$$

systematic term

$$S(\gamma,t) = -C(\gamma,t) + A(\gamma,t)$$

cooling term

$$C(\gamma) = |\dot{\gamma}_{synch}| + |\dot{\gamma}_{IC}| + |\dot{\gamma}_{ad}|$$

syst. acc. term

$$A(\gamma) = A_{p0}\gamma, t_A = \frac{1}{A_0}$$

$$\frac{\partial n(\gamma,t)}{\partial t} = \frac{\partial}{\partial \gamma}\left\{-[S(\gamma,t) + D_A(\gamma,t)]n(\gamma,t)\right\} + \frac{\partial}{\partial \gamma}\left\{D_p(\gamma,t)\frac{\partial n(\gamma,t)}{\partial \gamma}\right\} - \frac{n(\gamma,t)}{T_{esc}(\gamma)} - \frac{n(\gamma,t)}{T_{ad}} + Q(\gamma,t)$$

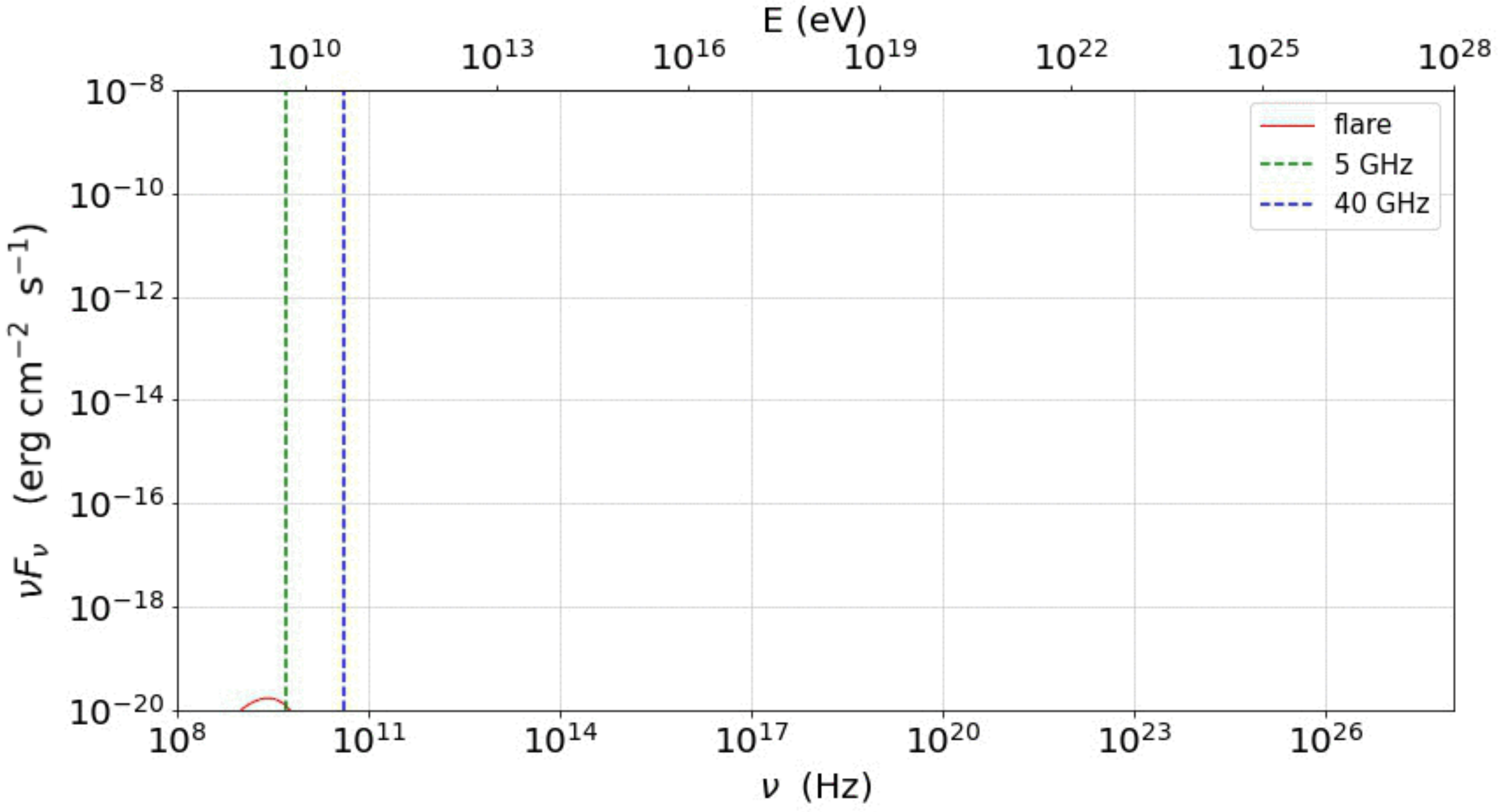
$$T_{ad} = \frac{1}{3}\frac{R(t)}{\beta_{\exp} c} \quad \text{(Gould 1975)}$$

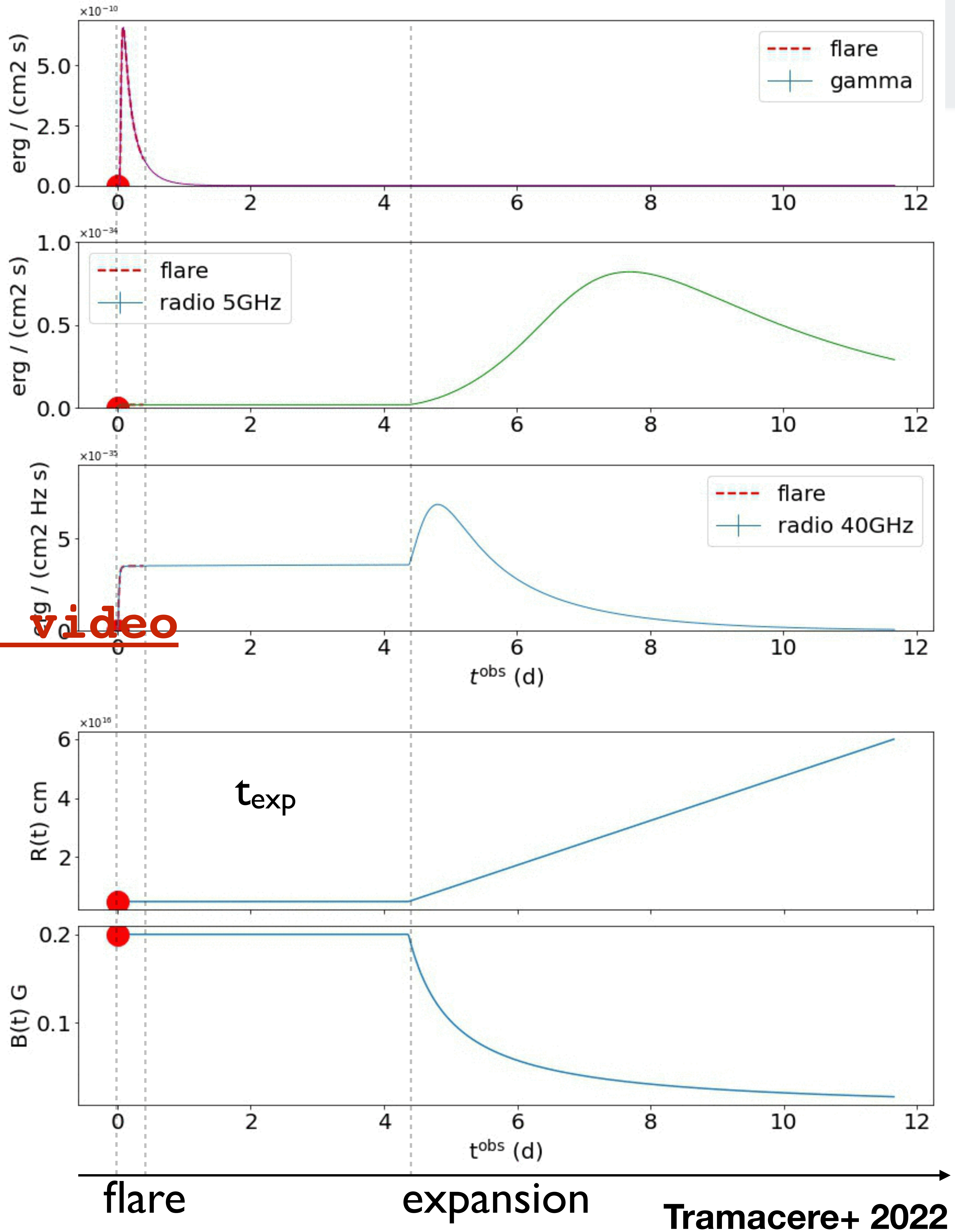
Turbulent magnetic field → momentum diffusion term

$$W(k) = \frac{\delta B(k_0^2)}{8\pi}\left(\frac{k}{k_0}\right)^{-q}$$

$$D_p \approx \beta_A^2 \left(\frac{\delta B}{B_0}\right)^2 \left(\frac{\rho_g}{\lambda_{\max}}\right)^{q-1} \frac{p^2 c^2}{\rho_g c}$$

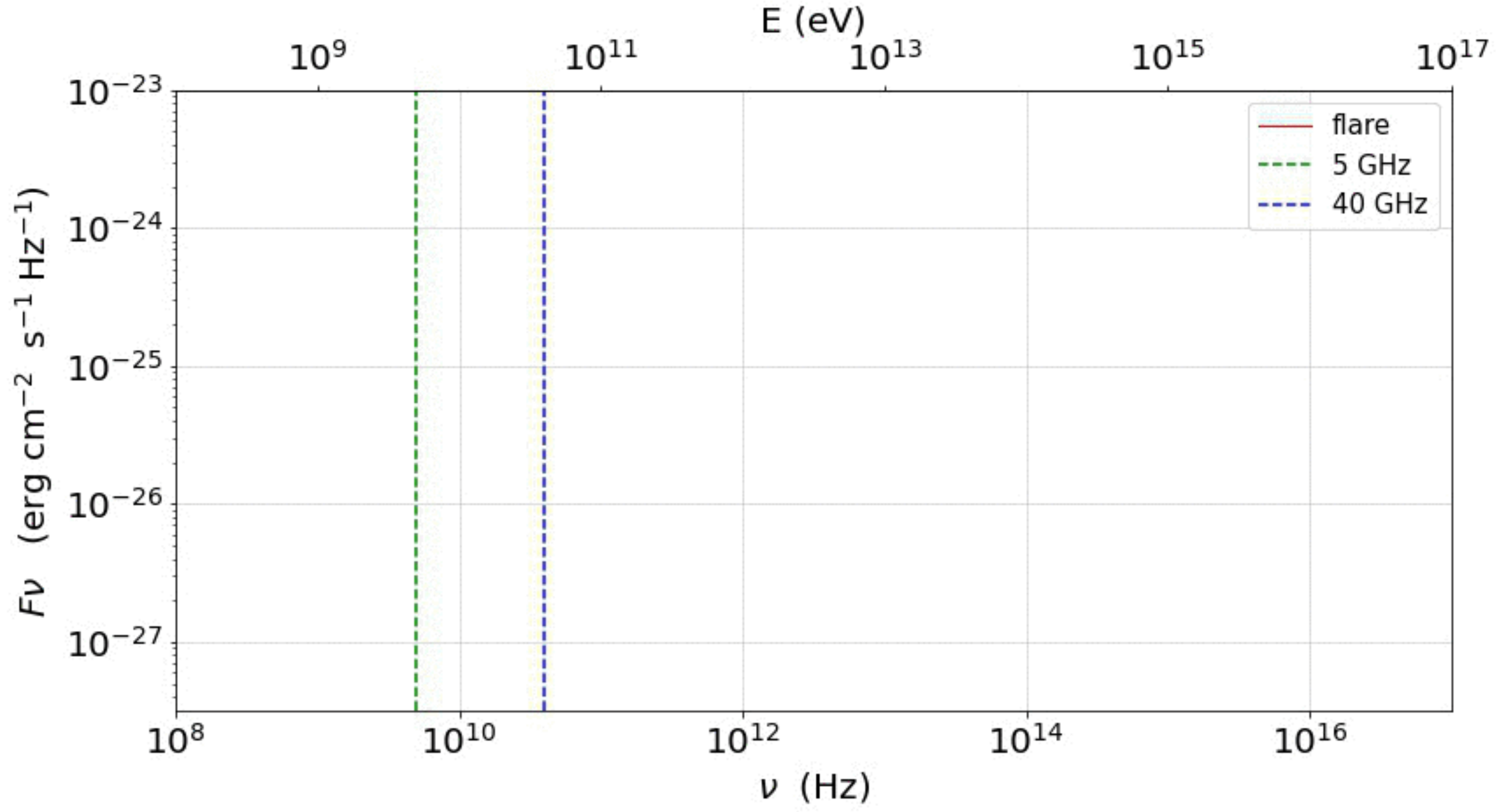
JetSeT radio-gamma delay and adb. exp. self-consistent approach





- duration~$3\text{x}10^7$ s (blob frame)~ 11 d obs
- t_{exp}=$1\text{x}10^7$ s
- β_{exp}=0.1c

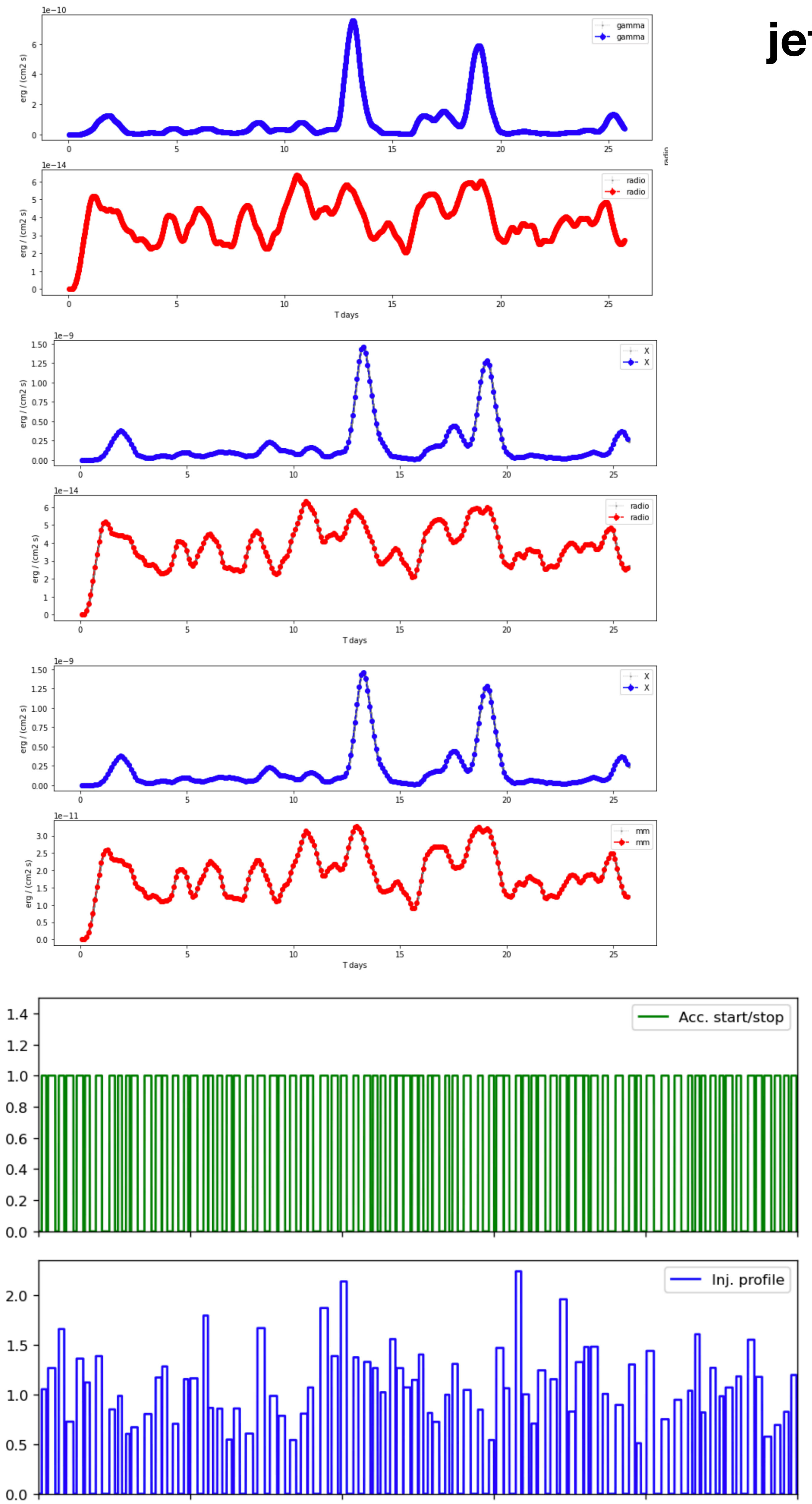
click here for the video



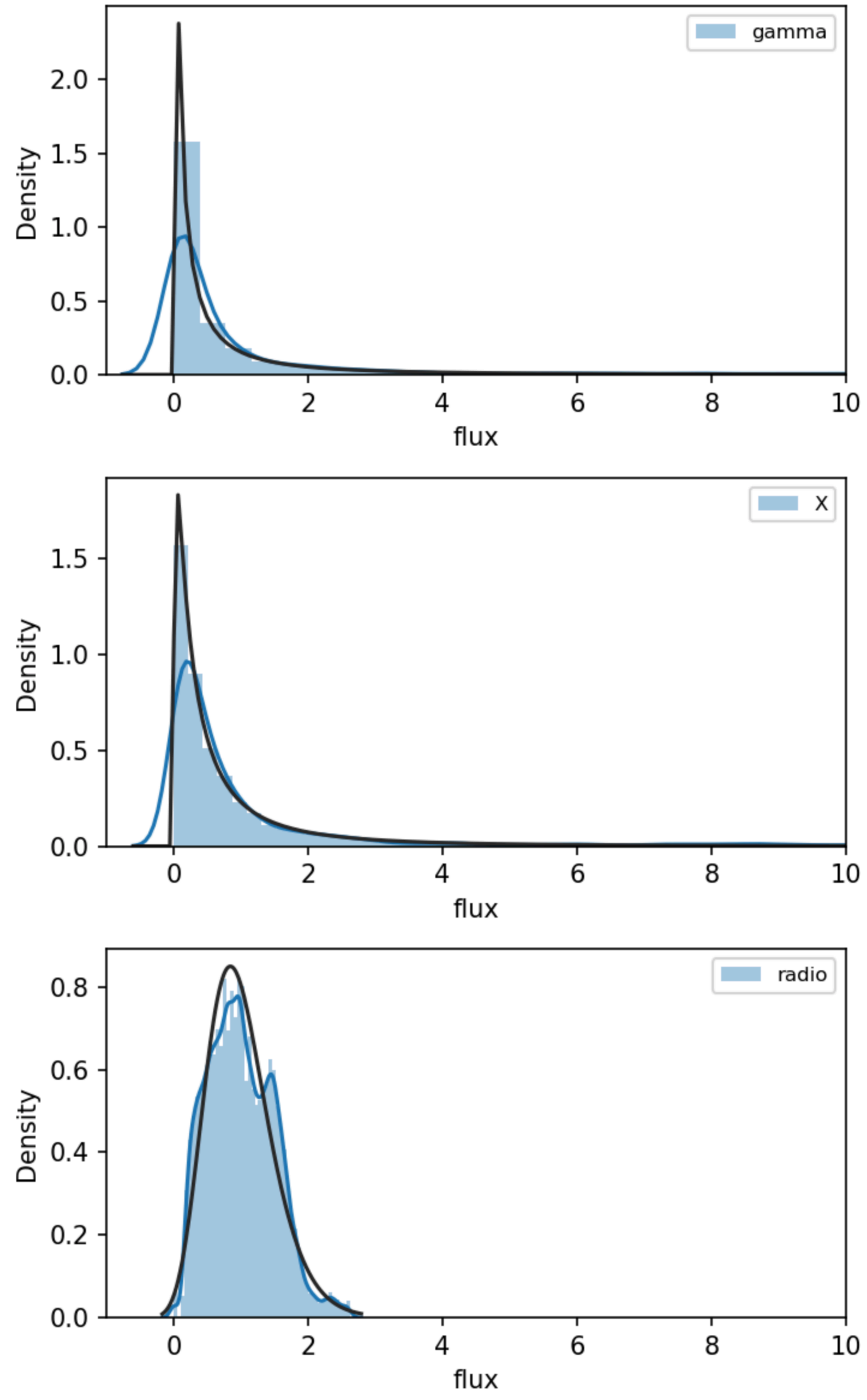
Tramacere+ 2022

JetSeT

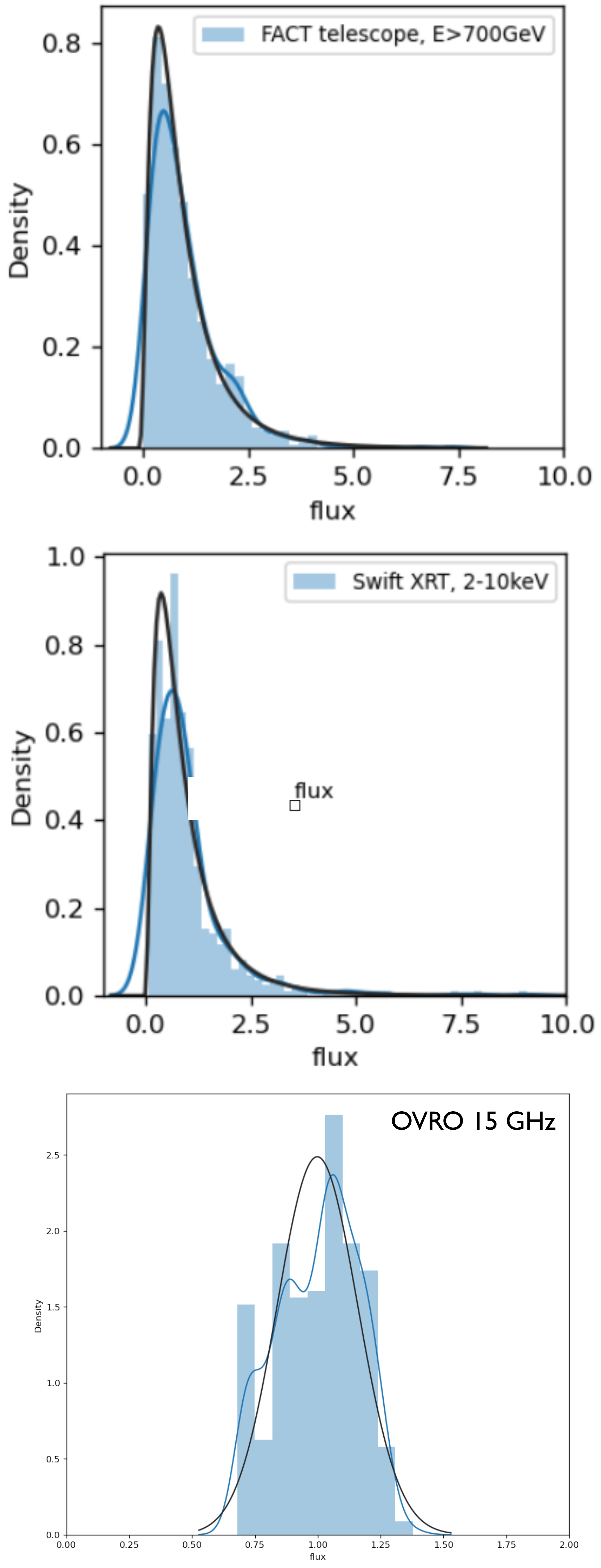
an examplem of temporal evolution with multiple injections



jetset simulation

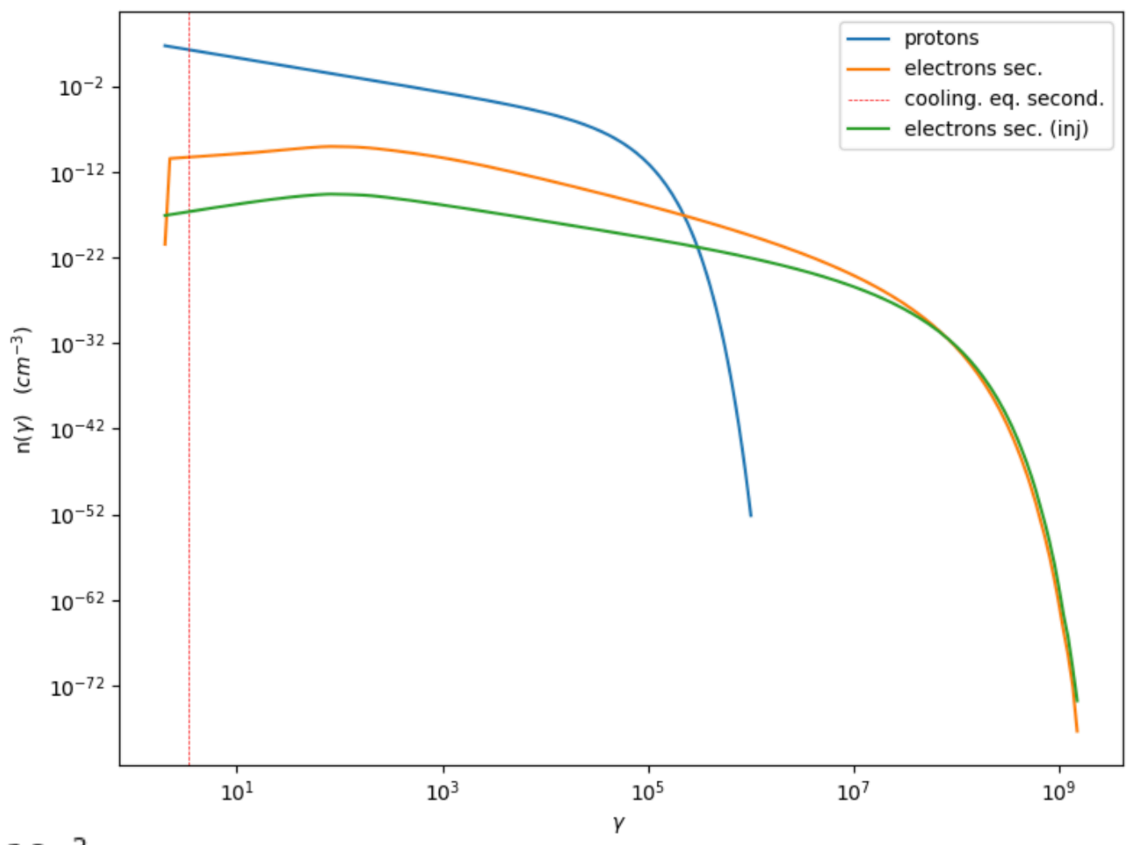


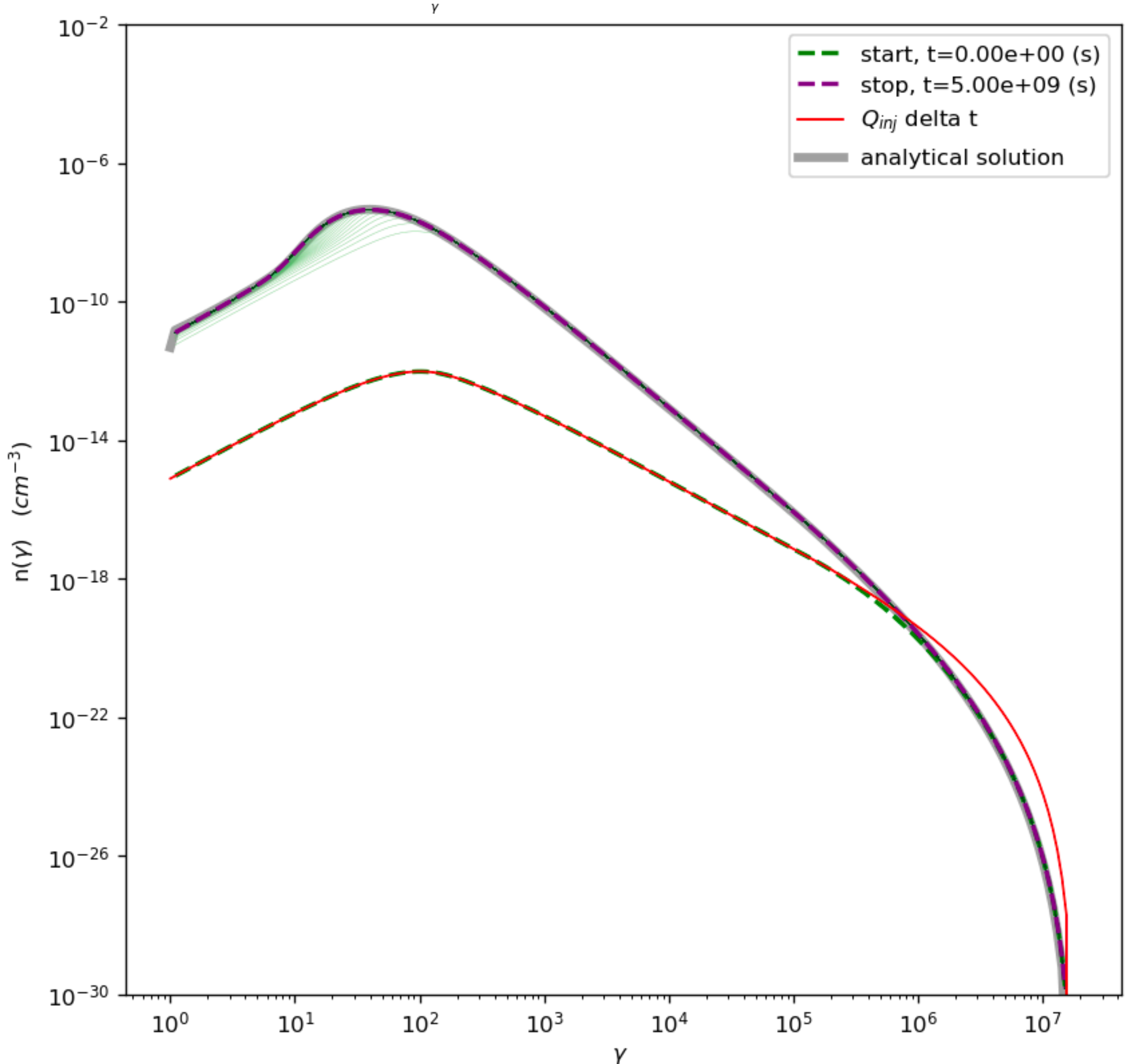
Observed datasets (Mrk 421)

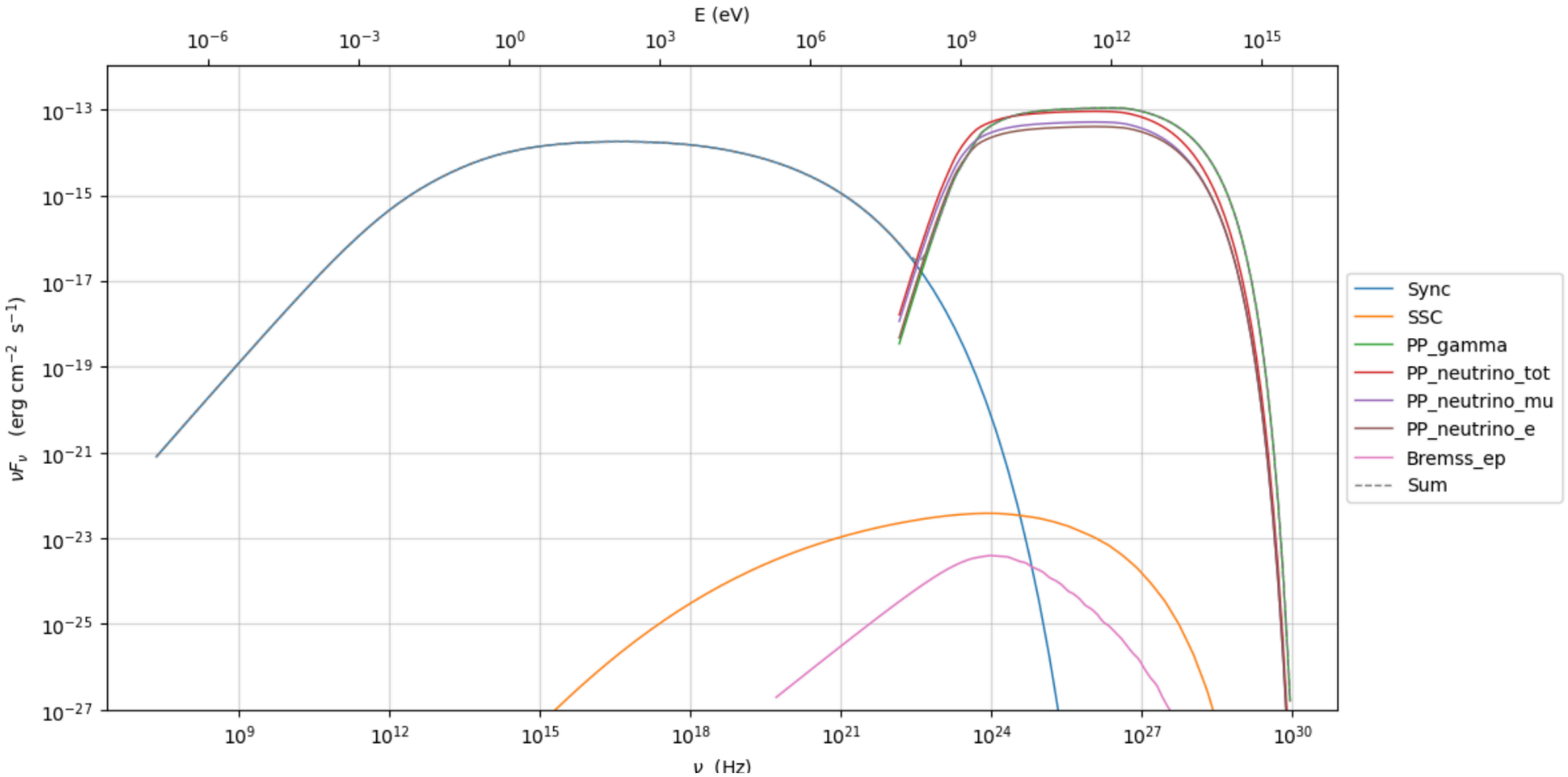


JetSeT

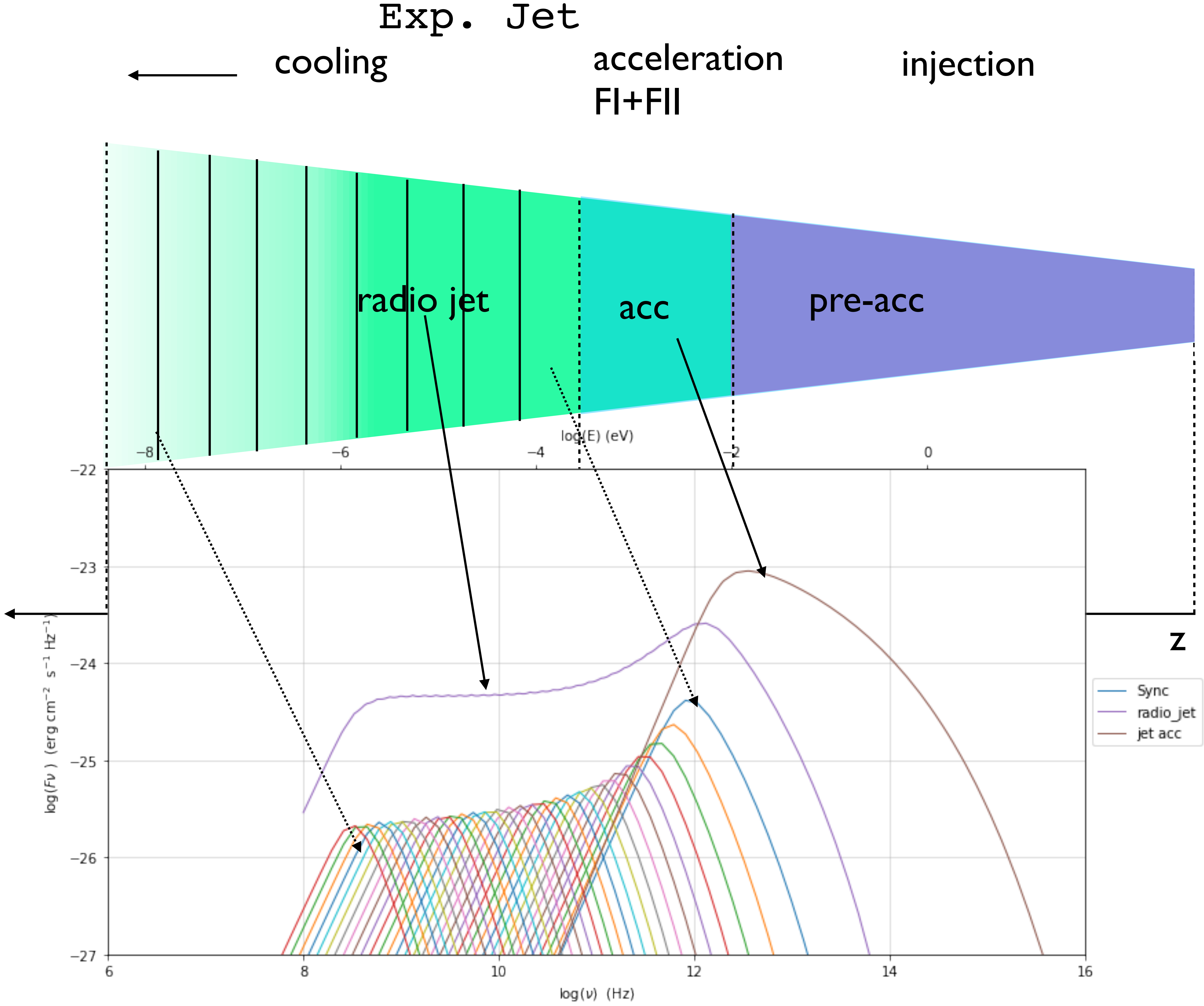
hadronic pp model (following Kelner&Aharonian 2006)

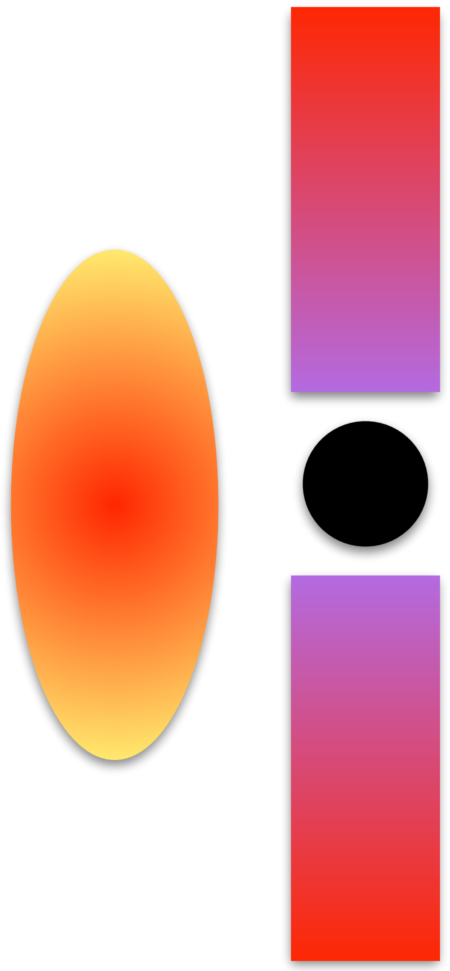






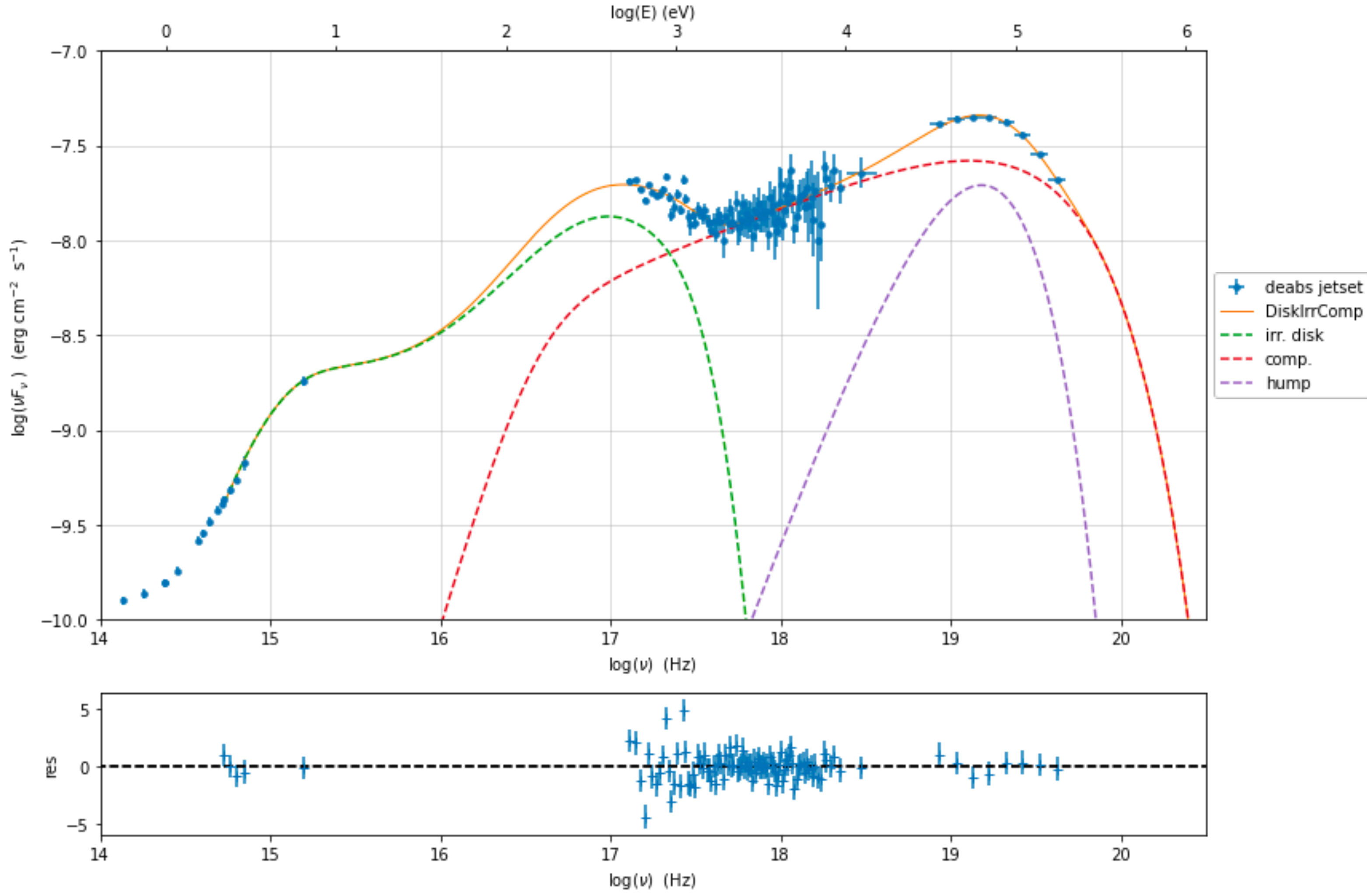
MQ plugin





DiksIrrComp Model +Corona implemented in JetSeT following

- Gierlinski, Done & Page 2009

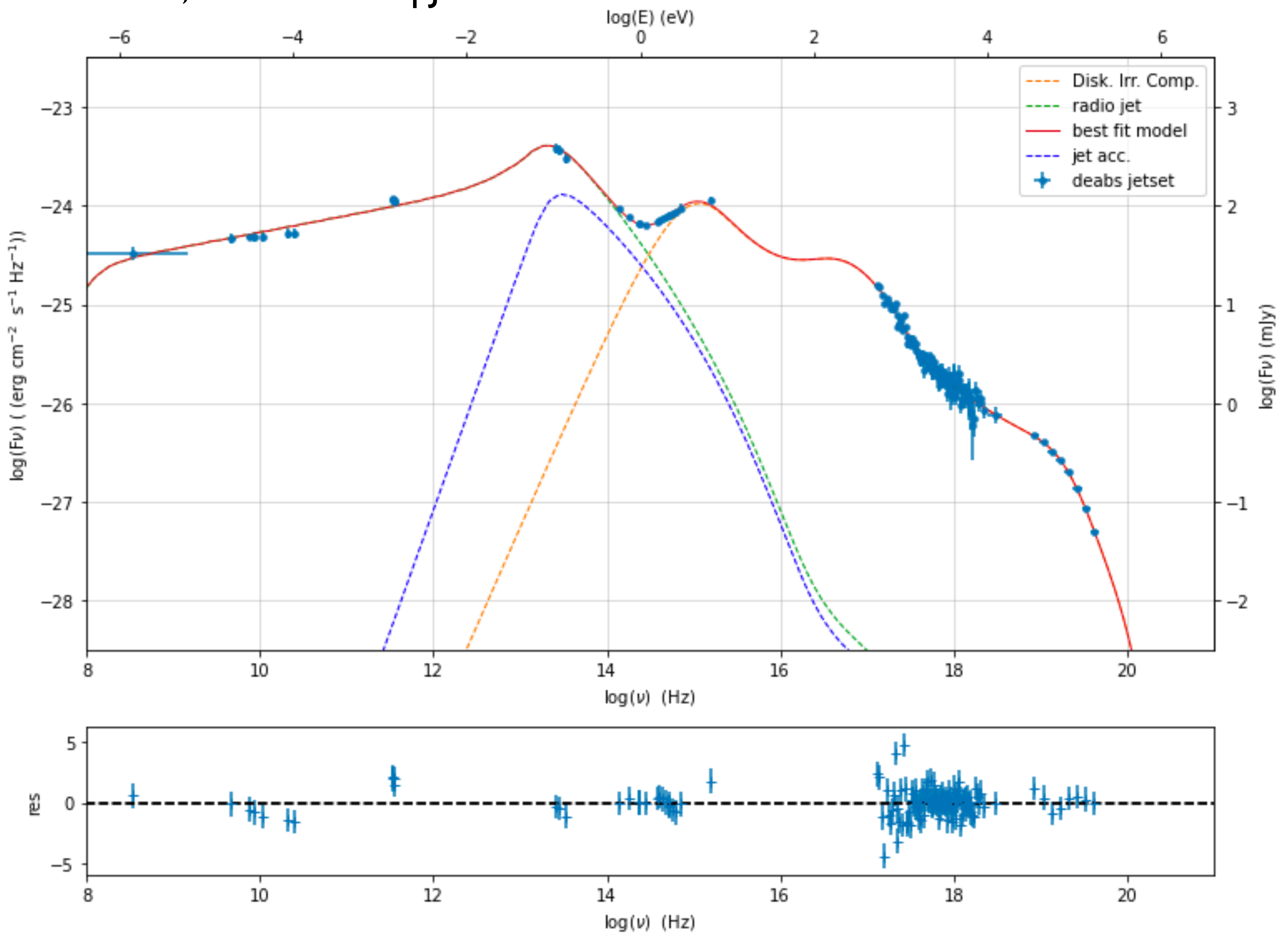


Rodi, Tramacere+ ApJ2020

JetSeT

MQ plugin: Best fit model

Rodi,Tramacere+ ApJ2020



Best fit model parameters					
model name	par. name	units	best fit value	error	starting value
JetAcc	$N_{e,acc}$	cm^{-3}	9.998×10^{11}	0.001×10^{11}	1.0×10^{12}
"	s		2.082	0.007	2.1
"	γ_{cut}		65.4	1.7	60
"	R_{acc}	cm	2.6×10^{9}	1.0×10^{1}	2.6×10^{9}
"	z_{acc}	cm			2.8×10^{10}
"	B_{acc}	G	17986	1.0×10^{-3}	17986
"	θ_{jet}	deg			63
"	Γ_{jet}				2.19
RadioJet	z_{inj}				2.5×10^{10}
"	N_{frac}				1
"	K_R^{start}				1
"	K_R^{end}				30000
"	m_{jet}		1.203	0.001	1.1

par. name	units	value	setup value
q_{jet}		> 0.15	0.20
U_e/U_B		0.18	–
N_e^{acc}/N_p^{cold}		< 94	–
L_{jet}^{acc}	erg s^{-1}	$> 8.0 \times 10^{37}$	–
L_{rad}^{acc}	erg s^{-1}	1.1×10^{36}	–
L_B^{acc}	erg s^{-1}	3.6×10^{37}	3.6×10^{37}
L_e^{acc}	erg s^{-1}	6.6×10^{36}	–
L_p^{acc}	erg s^{-1}	$> 3.6 \times 10^{37}$	$> 3.6 \times 10^{37}$

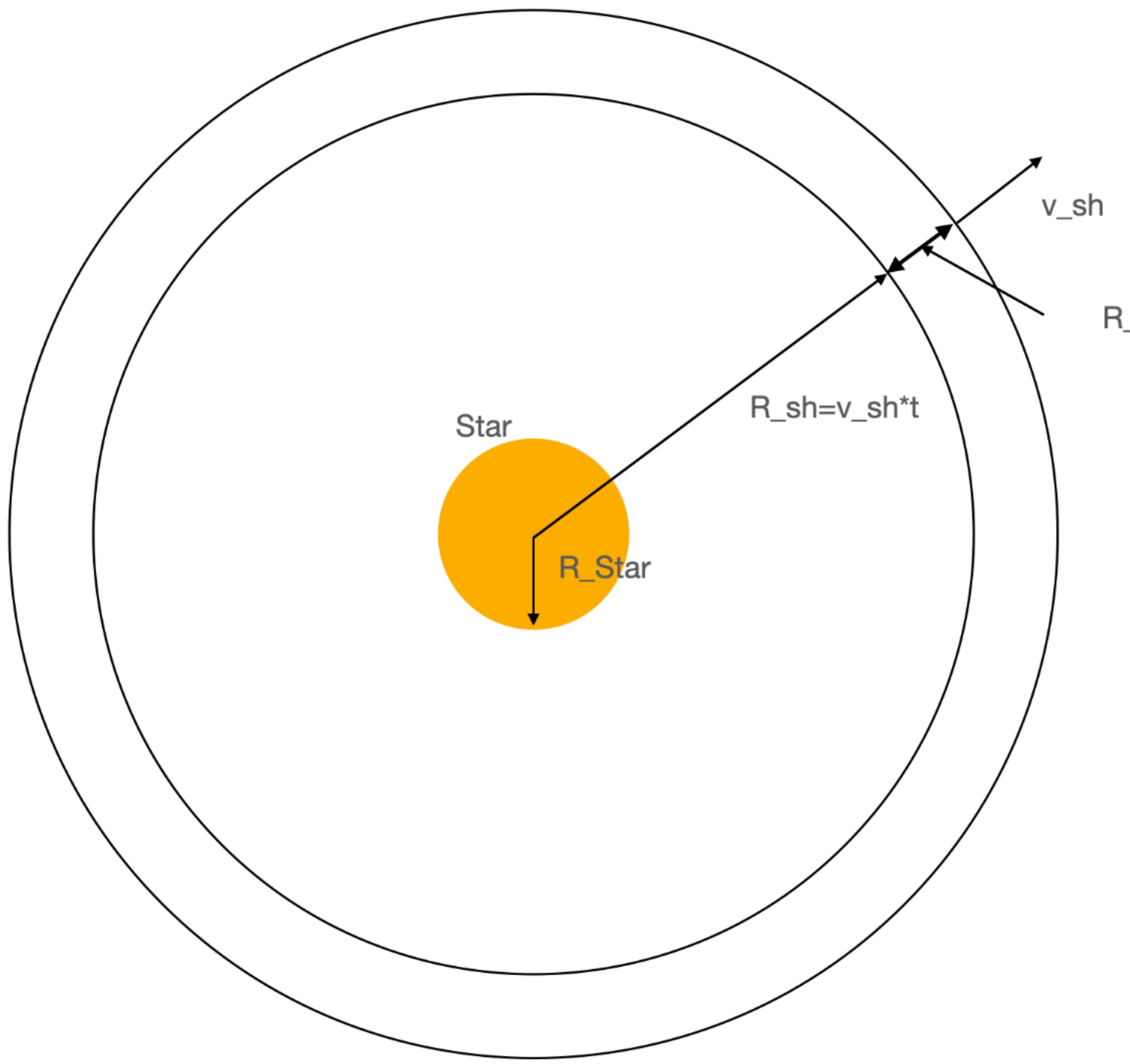
JetSeT

Short term: new features in version 1.3.x

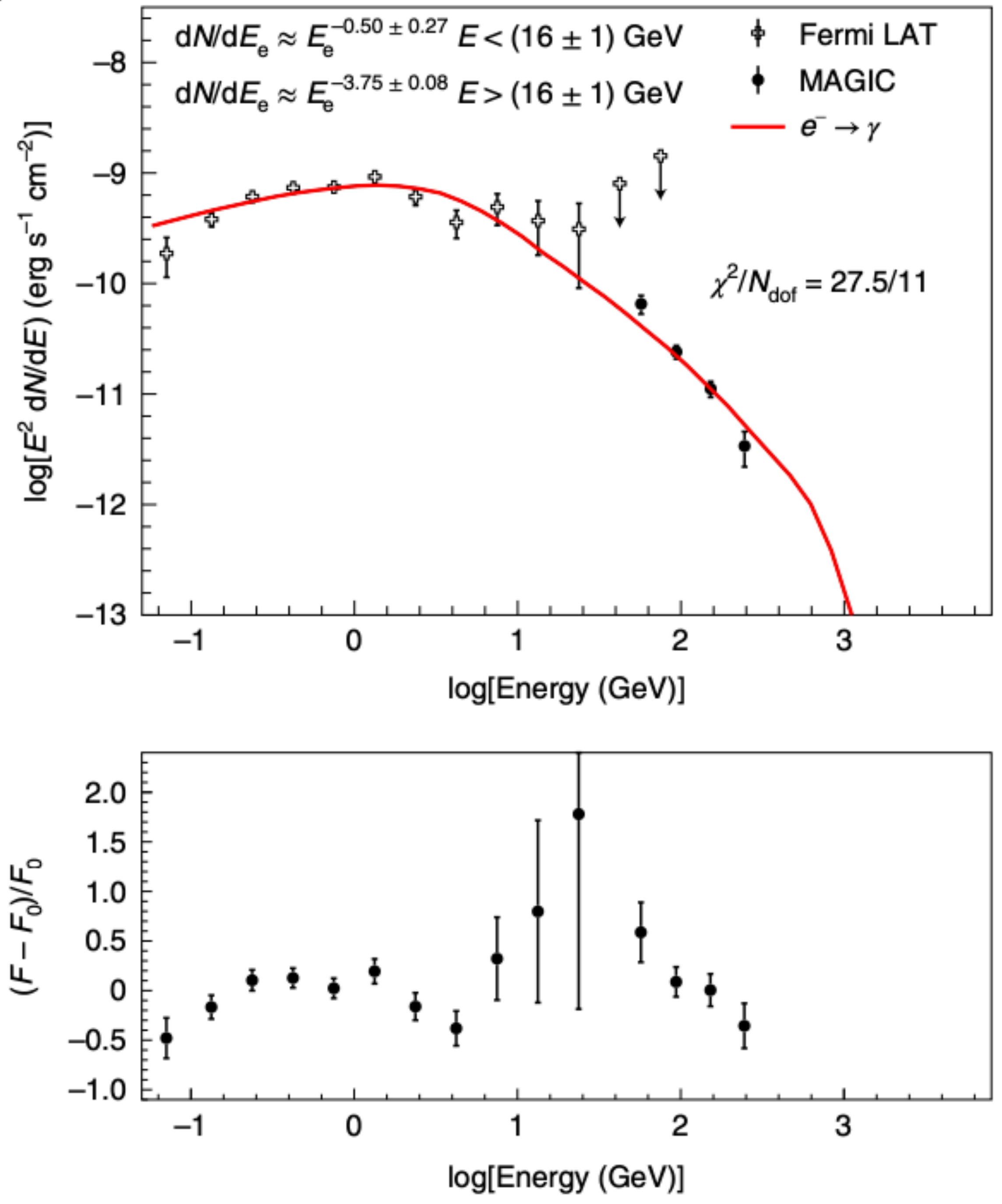
- ☑ Updated EBL models (Dominguez+23, Saldana-Lopez+21)
- ☑ C threads (speed up more effective compare to python threads, at least 3x!)
- ☑ optical depth (BLR, DT, and generic fields)
- ☑ bulk Compton emissions
- ☑ Synchrotron polarization (one region and multi-region, stochastic model with a dedicated plugin)
- ☑ Improved serialization (removed pickle issues, e.g. astropy, numba etc…)
- WORK IN PROGRESS: 3ML plugin
- WORK IN PROGRESS: GRB plugin
- WORK IN PROGRESS: p-gamma emission (following Kelner&Aharonian 2008, in collaboration with Ankur Sharma) in progress…

Short term: new features in version 1.3.x

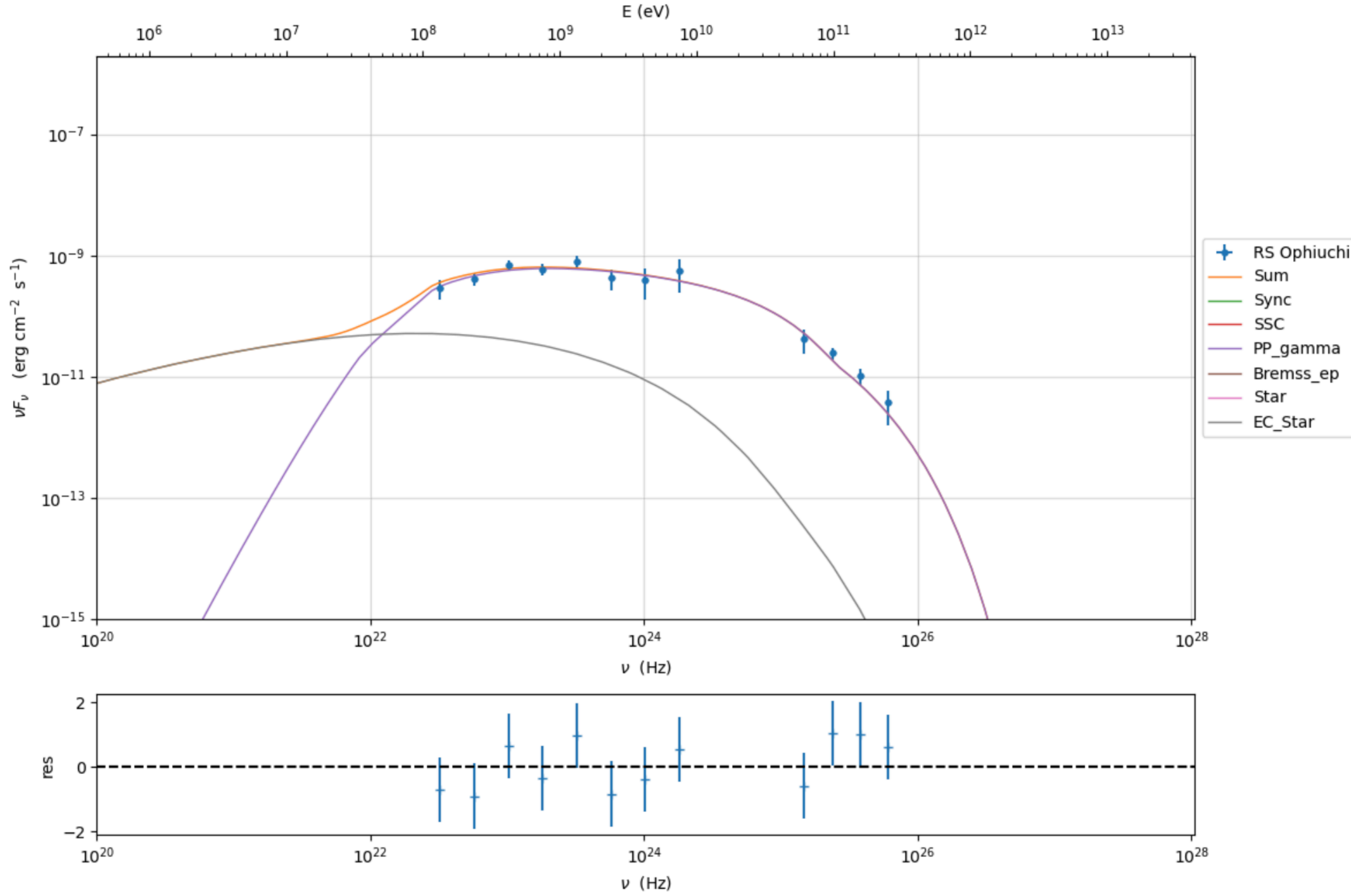
- Extended plugins for Galactic objects (this can be fully user implemented/modified): reproduction of MAGIC paper Nature Astronomy, Volume 6, p. 689-697



MAGIC paper
nova RS Ophiuchi

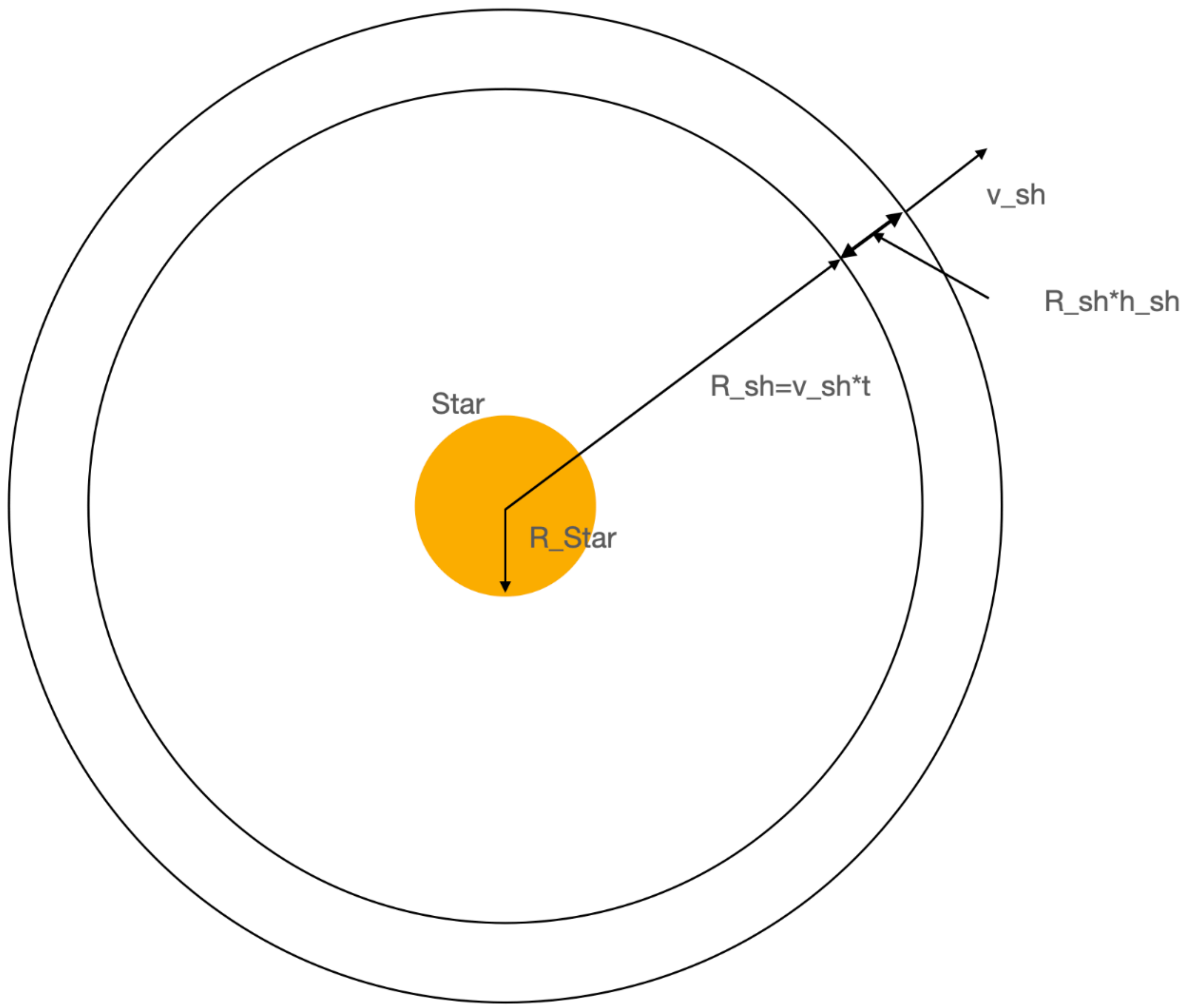


jetset



JetSeT Short term: new features in version 1.3.x

- Extended plugins for Galactic objects (this can be fully user implemented/modified): reproduction of MAGIC paper Nature Astronomy, Volume 6, p. 689-697




```python
def build_nova_expanding_shell(E_k=1E40,emitters_type='electrons',emitters_distribution='bkn'):
    j_spherical_shell=build_expanding_spherical_shell(add_EC_star=True,sync='self-abs',emitters_type=emitters_type,emitters_distribution=emitters_distribution)

    j_spherical_shell.parameters.B.val=1E-3
    j_spherical_shell.parameters.T_Star.val=6000
    j_spherical_shell.parameters.gmax.val=1E3
    j_spherical_shell.parameters.v_sh.val=4500
    j_spherical_shell.parameters.L_Star.val=4E43

    j_spherical_shell.set_external_field_transf('disk')
    x=u.M_sun
    j_spherical_shell.add_user_par(name='M_ej',units=x,val=0.1,val_min=0,val_max=None)


    def par_func_N(M_ej,R_sh,h_sh):
        """
        determines the accelerated proton density from M_ej and shell volume
        Eq. 4 in the paper

        """
        vol=4/3*np.pi*R_sh**3*((1+h_sh)**3 -1)*u.Unit('cm3')

        N= M_ej.cgs/(vol*constants.m_p.cgs)

        return N

    if emitters_type == 'protons':
        j_spherical_shell.add_user_par('NH_pp_ratio',val=1E-5,val_min=0, val_max=1000)

        def par_func_NH_pp(N,NH_pp_ratio):
            """
            This should come form Eq. 5, but I did not find the relevant parameters in the paper.
            Anyhow, the n_ej>>n_target, I paramtrize it as function of n_je i.e. accelerated proton density
            """
            return N*NH_pp_ratio

        j_spherical_shell.make_dependent_par(par='NH_pp', depends_on=['N','NH_pp_ratio'], par_expr=par_func_NH_pp)

    j_spherical_shell.make_dependent_par(par='N', depends_on=['M_ej', 'R_sh','h_sh'], par_expr=par_func_N)

    def par_func_M_ej(E_k,v_sh):
        """
        mass of the ejecta from Eq. 11
        """
        return (2*E_k*u.erg/(v_sh.cgs)**2)/((u.M_sun).to('g'))

    j_spherical_shell.add_user_par(name='E_k',units='erg',val=1E40,val_min=0,val_max=None)


    j_spherical_shell.make_dependent_par(par='M_ej', depends_on=['E_k','v_sh'], par_expr=par_func_M_ej)
    j_spherical_shell.parameters.E_k.val=1E42
    return j_spherical_shell
```

Mid term: v2.x.x

- Transition form C to C++. Already working minimal version.

JetSeT

Documentation

JetSeT doc 1.2.2 JetSeT ▾ Page ▾ Installation » Source

Search

JetSeT

Jets SED modeler and fitting Tool

Author: Andrea Tramacere

JetSeT is an open source C/Python framework to reproduce radiative and accelerative processes acting in relativistic jets, and galactic objects (beamed and unbeamed), allowing to fit the numerical models to observed data. The main features of this framework are:

- handling observed data: re-binning, definition of data sets, bindings to astropy tables and quantities definition of complex numerical radiative scenarios: Synchrotron Self-Compton (SSC), external Compton (EC) and EC against the CMB
- Constraining of the model in the pre-fitting stage, based on accurate and already published phenomenological trends. In particular, starting from phenomenological parameters, such as spectral indices, peak fluxes and frequencies, and spectral curvatures, that the code evaluates automatically, the pre-fitting algorithm is able to provide a good starting model,following the phenomenological trends that I have implemented. fitting of multiwavelength SEDs using both frequentist approach (iminuit) and bayesian MCMC sampling (emcee)
- Self-consistent temporal evolution of the plasma under the effect of radiative, accelerative processes, and adiabatic expansion. Both first order and second order (stochastic acceleration) processes are implemented.

Important

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

- Tramacere A. 2020
- Tramacere A. et al. 2011
- Tramacere A. et al. 2009

Documentation:

- installation
- what's new in jetset 1.2.2 and bug fixing
- user guide
- code documentation (API)
- bibliography

New/updated in v 1.2.2-1.2.0:

NO THUMB NAIL	NO THUMB NAIL	NO THUMB NAIL	NO THUMB NAIL	NO THUMB NAIL	NO THUMB NAIL

Documentation

- The documentation leads users to non-black-box usage
- extensive theory background added here

JetSeT

Some recent scientific results

JetSeT

hadronic pp model

A&A 673, A75 (2023)
https://doi.org/10.1051/0004-6361/202245086

Astronomy & Astrophysics

Multiwavelength study of the galactic PeVatron candidate LHAASO J2108+5157

S. Abe[1], A. Aguasca-Cabot[2], I. Agudo[3], N. Alvarez Crespo[4], L. A. Antonelli[5], C. Aramo[6], A. Arbet-Engels[7], M. Artero[8], K. Asano[1], P. Aubert[9], A. Baktash[10], A. Bamba[11], A. Baquero Larriva[12], L. Baroncelli[13],

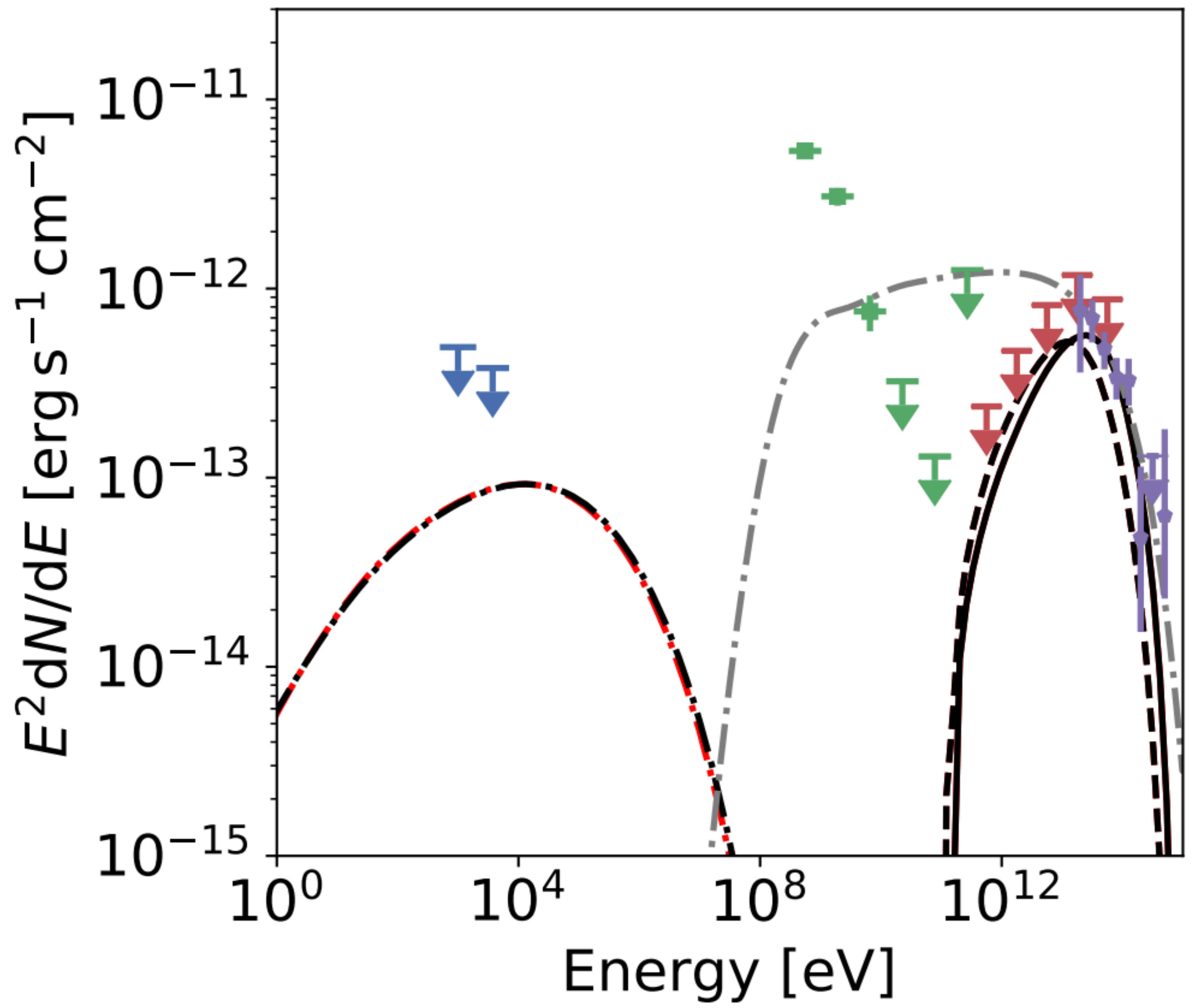


Table 5. Best-fit parameters of π^0 decay-dominated VHE-UHE emission of LHAASO J2108+5157 for both molecular clouds in the direction of the source.

Parameter	Best fit value		Frozen
	Cloud 1	Cloud 2	
n [cm^{-3}]	115	240	True
d [kpc]	3.1	2.0	True
R [pc]	7.1	4.5	True
$\gamma_{\rm min}$ [$\times10^5$]	1.6 ± 0.5	1.6 ± 5	False
$\gamma_{\rm max}$ [$\times10^6$]	1.0	1.0	True
B [mG]	9 ± 5	≤ 8	False
N [$\times10^{-15}$ cm^{-3}]	4 ± 1	3 ± 1	False
α	2.75	2.75	True

Notes. The injected protons are assumed to be distributed according to ECPL with γ-factor in the range ($\gamma_{\rm min}$, $\gamma_{\rm max}$), cutoff at $\gamma_{\rm cut}$, spectral index α, and total numeric density N.

- pp MW model fitting (iminuit plugin) with temporal evolution of secondaries
- both and gamma and neutrino emission

JetSeT

MAGIC paper I

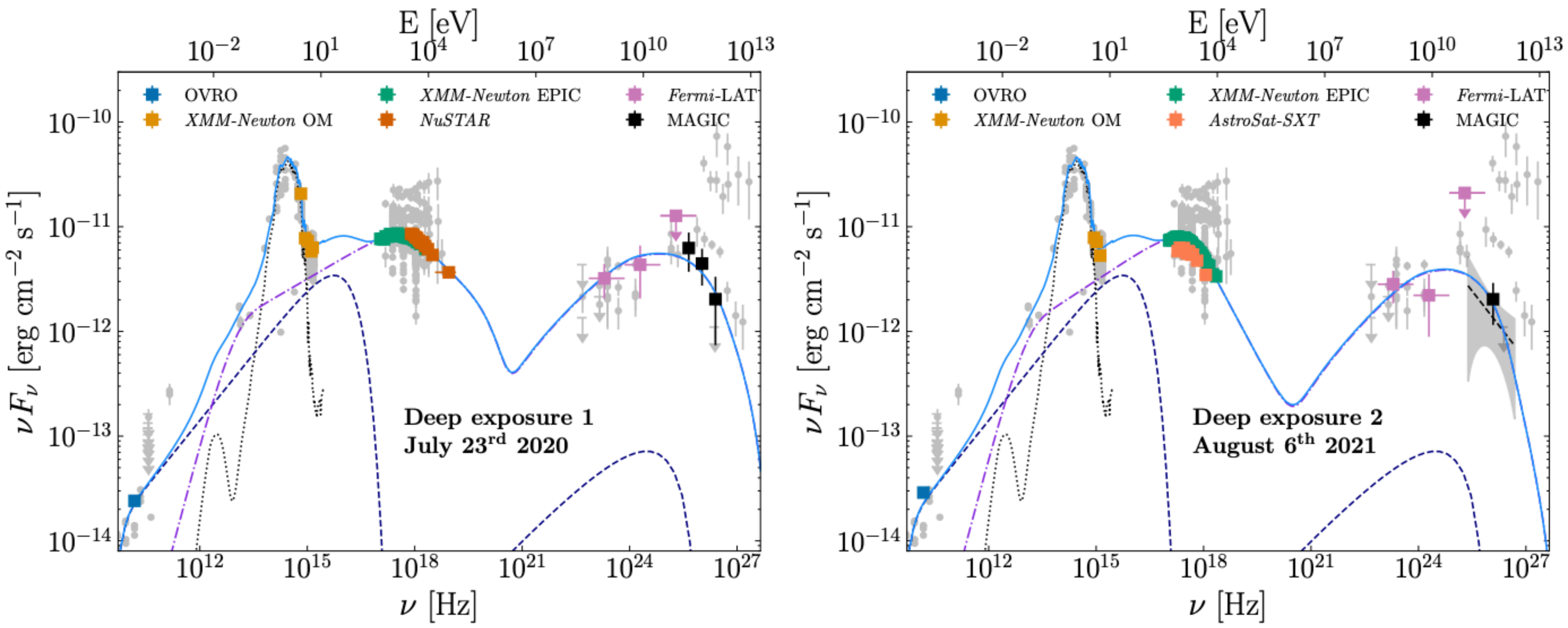
A&A, 682, A114 (2024)
https://doi.org/10.1051/0004-6361/202347845

Astronomy & Astrophysics

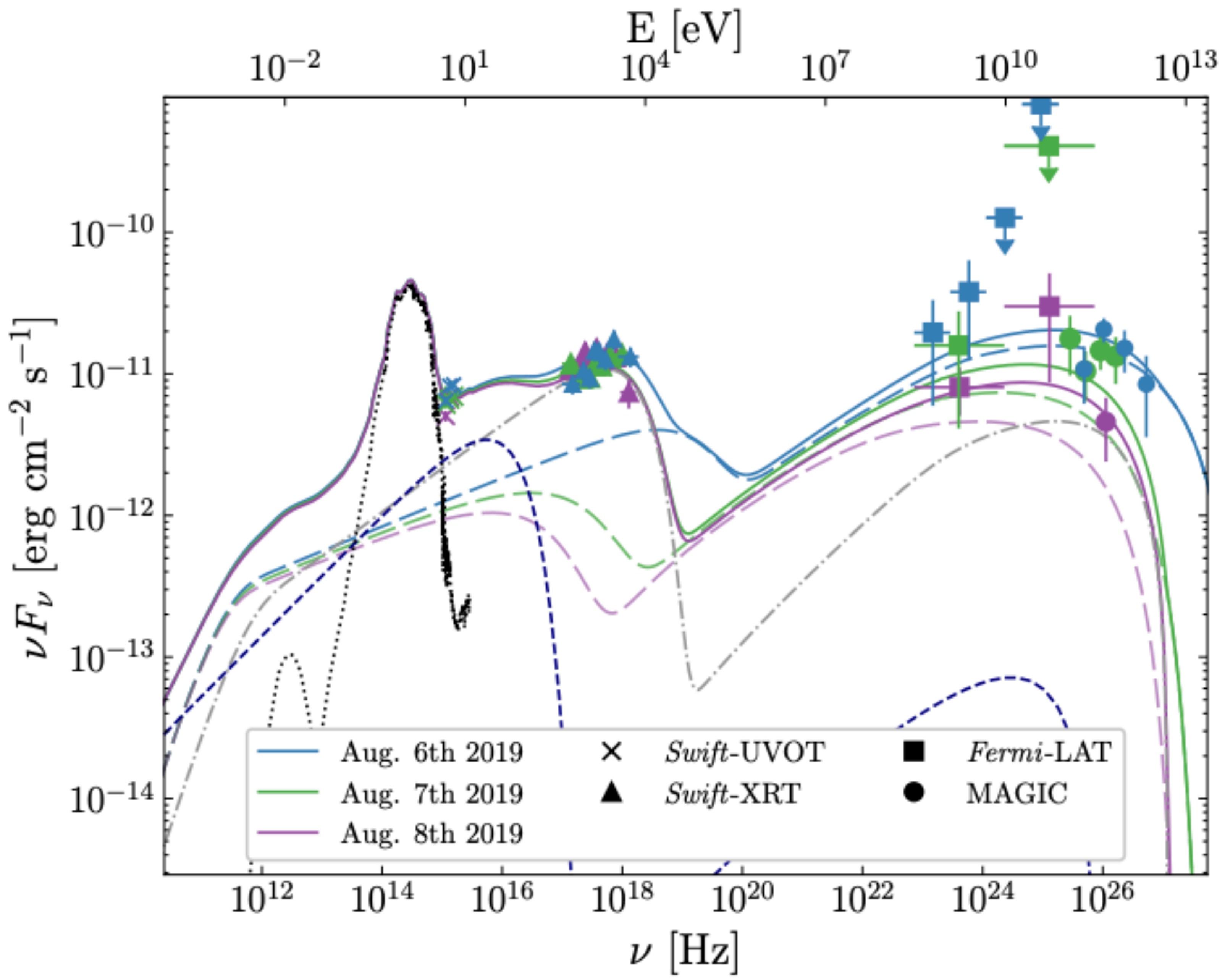
Multi-year characterisation of the broad-band emission from the intermittent extreme BL Lac 1ES 2344+514

MAGIC Collaboration: H. Abe[1], S. Abe[1], V. A. Acciari[2], I. Agudo[3], T. Aniello[4], S. Ansoldi[5,42], L. A. Antonelli[4], A. Arbet Engels[6,★], C. Arcaro[7], M. Artero[8], K. Asano[1], D. Baack[9], A. Babić[10], A. Baquero[11], U. Barres de Almeida[12], I. Batković[7], J. Baxter[1], J. Becerra González[2], E. Bernardini[7], J. Bernete[13], A. Berti[6], J. Besenrieder[6], C. Bigongiari[4],

SED Model fitting



Time-evolved model fitting



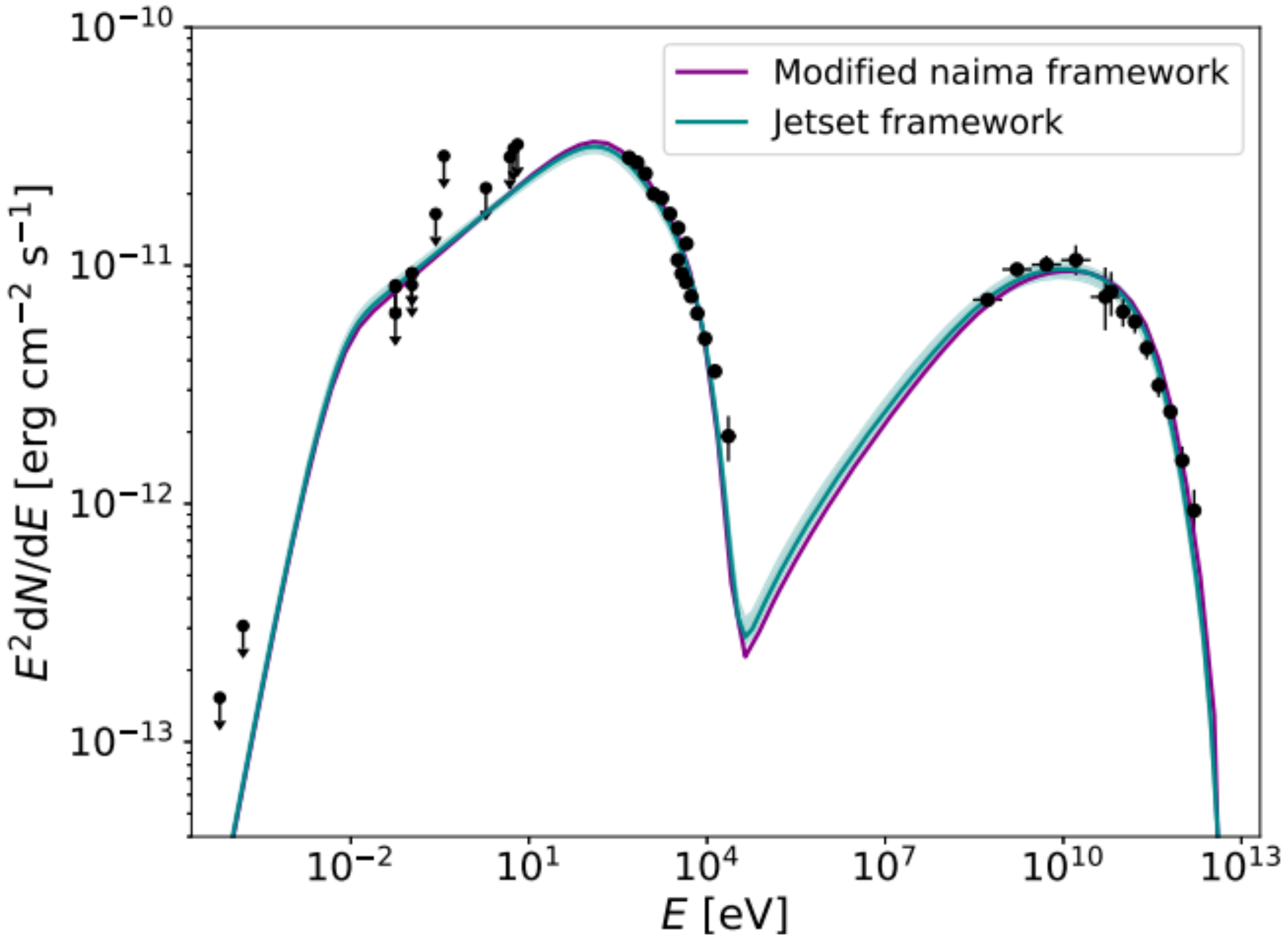
MAGIC paper II

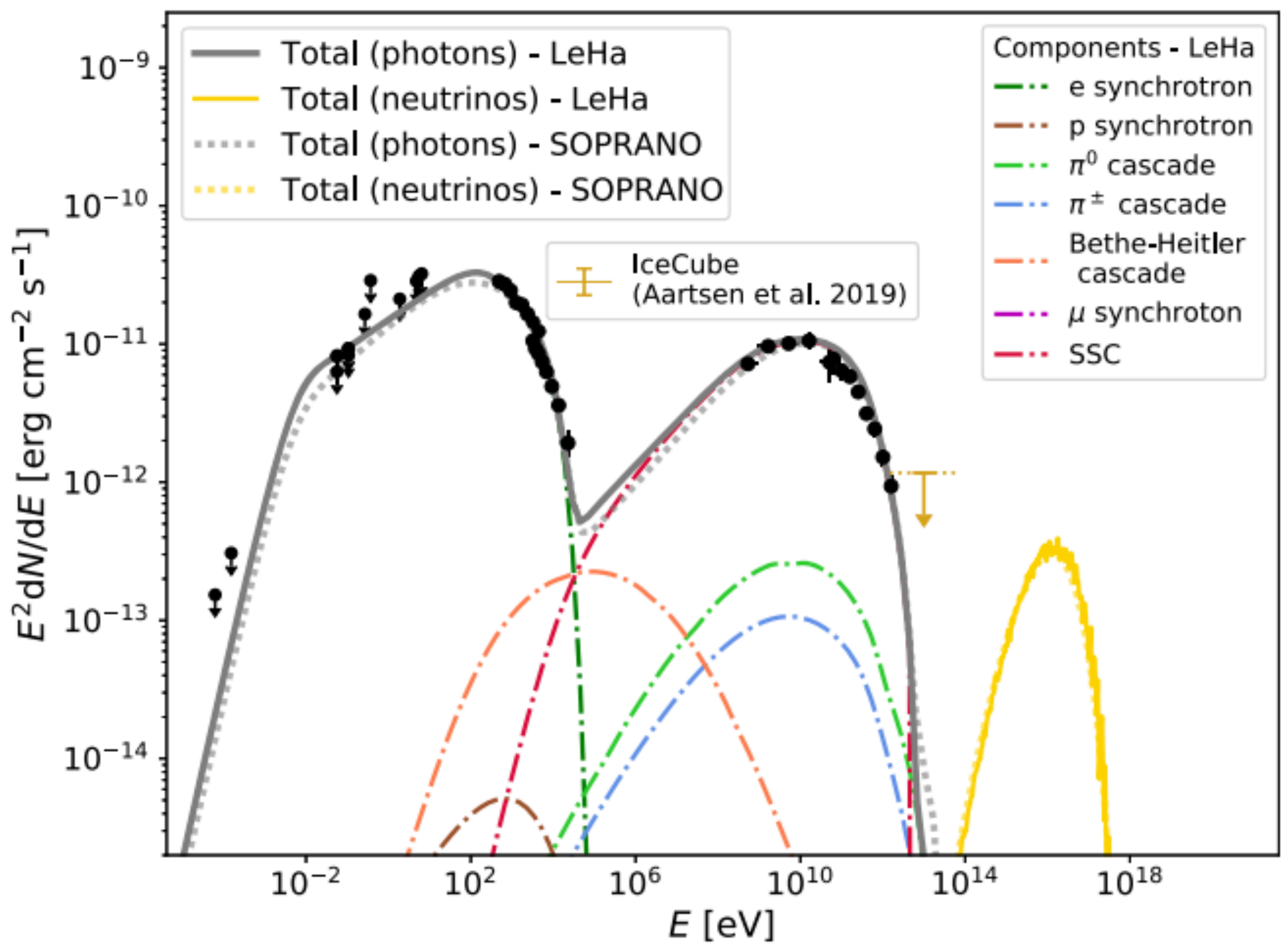
The Astrophysical Journal Supplement Series, 266:37 (43pp), 2023 June https://doi.org/10.3847/1538-4365/acc181

Multimessenger Characterization of Markarian 501 during Historically Low X-Ray and γ-Ray Activity

H. Abe[1], S. Abe[1], V. A. Acciari[2], I. Agudo[3], T. Aniello[4], S. Ansoldi[5,98], L. A. Antonelli[4], A. Arbet-Engels[6], C. Arcaro[7], M. Artero[8], K. Asano[1], D. Baack[9], A. Babić[10], A. Baquero[11], U. Barres de Almeida[12], J. A. Barrio[11], I. Batković[7], J. Baxter[1], J. Becerra González[2], W. Bednarek[13], E. Bernardini[7], M. Bernardos[3],

Leptonic modeling (Jetset and Naima), and comparison with leptohadronic codes (LeHa and Soaprano)



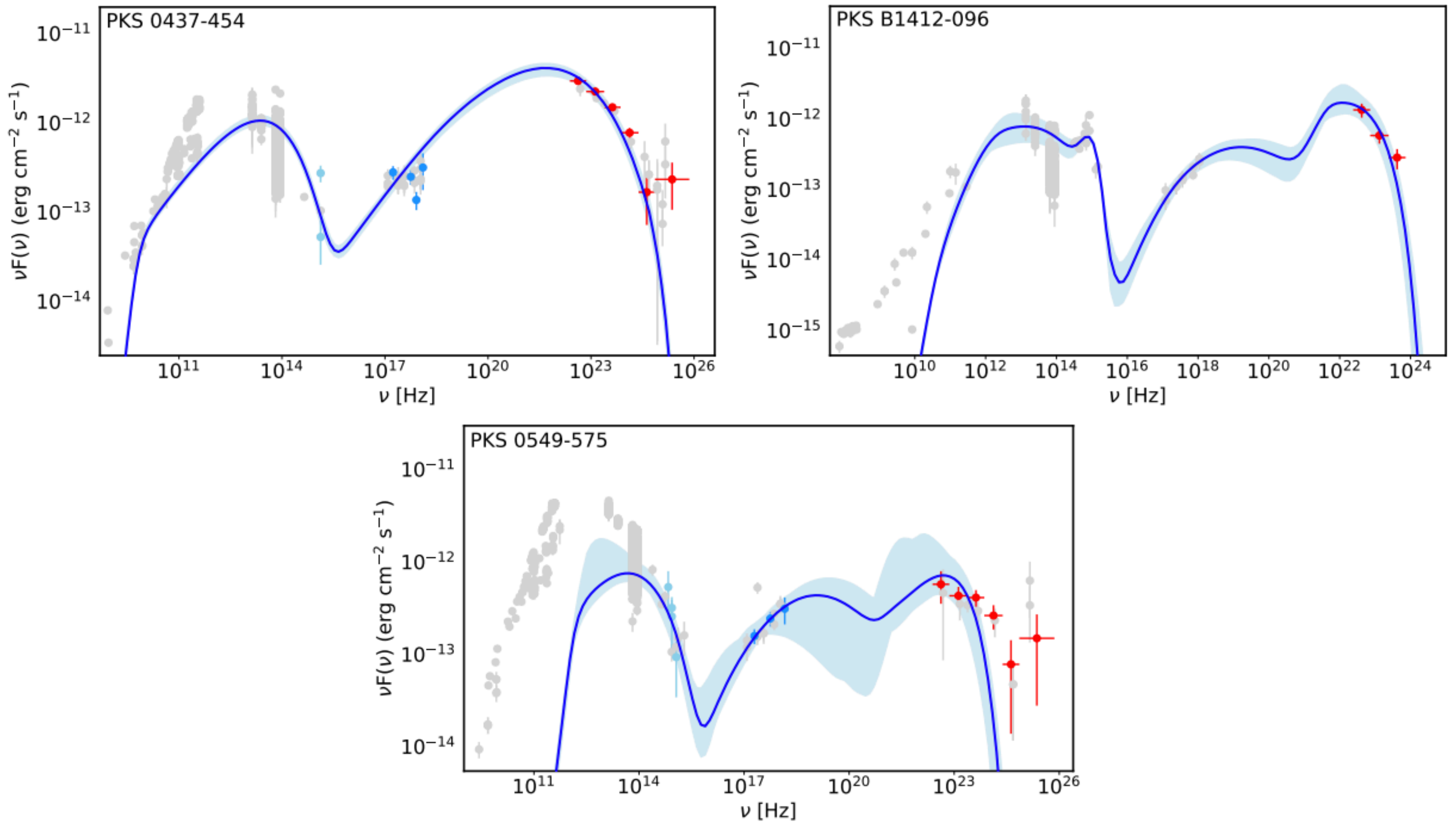


JetSeT

MNRAS 000, 1–20 (2022) Preprint 29 January 2024 Compiled using MNRAS LaTeX style file v3.0

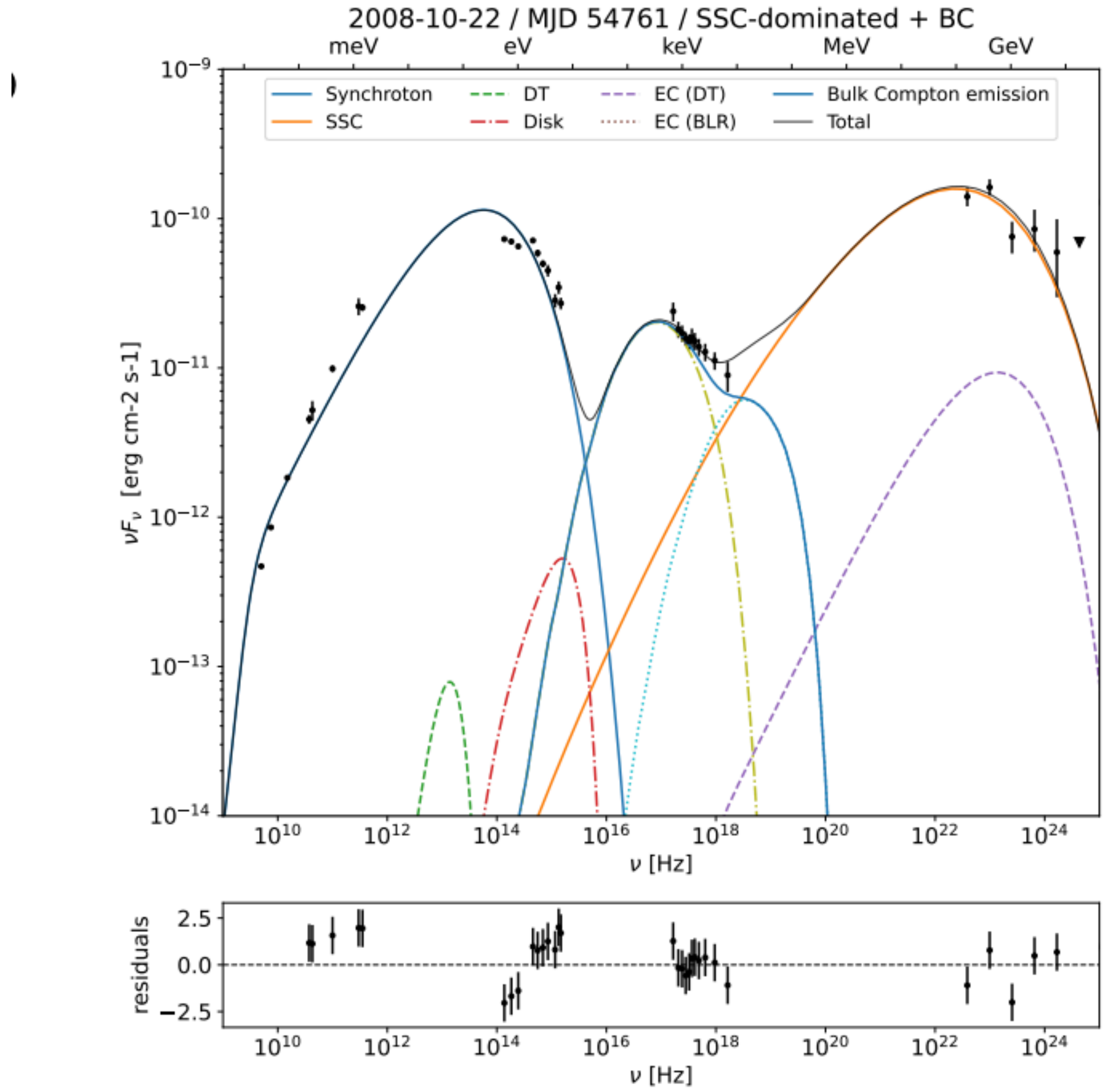
Broadband Study of Gamma-Ray Blazars at Redshifts $z = 2.0 - 2.5$

N. Sahakyan[1], ⋆ G. Harutyunyan[1], S. Gasparyan[1], D. Israyelyan[1]
[1]ICRANet-Armenia, Marshall Baghramian Avenue 24a, Yerevan 0019, Armenia



Repeating flaring activity of the blazar AO 0235+164

J. Escudero Pedrosa[1], I. Agudo[1], A. Tramacere[2], A. P. Marscher[3], S. Jorstad[3,12], Z. R. Weaver[3], C. Casadio[5,4], C. Thum[6], I. Myserlis[6], A. Fuentes[1], E. Traianou[1], J.-Y. Kim[7,8], J. Kramer[7], R. López-Coto[1], F. D'Ammando[9], M. Bernardos[1], G. Bonnoli[10,1], D. A. Blinov[4,5], G. A. Borman[11], T. S. Grishina[12], V. A. Hagen-Thorn[12], E. N. Kopatskaya[12], E. G. Larionova[12], V. M. Larionov[12], L. V. Larionova[12], D. A. Morozova[12], S. S. Savchenko[12,13,14], I. S. Troitskiy[12], Y. V. Troitskaya[12], and A. A. Vasilyev[12]



Conclusions

Pro

- So far Jetset is a successful project
- Used for publications by large collaboration and small group os scientists, for teaching and PhD/Master thesis
- Great fun in turning a personal code (start date Nov. 2000 for my master thesis) into an open source project used by the community

Conclusions

Cons

- Time consuming (coding, CI/CD, documentation…)
- So far the project was developed only by me, but contributors are super welcome (thanks Ankur for being the first active contributors!)
- Overlapping! As of today, we have plenty of code, with plenty of overlapping features

Conclusions

The future: a collaborative effort!

- Would be great to turn (at leas a fraction) of the active projects into a single effort (a la astropy), combining the strength of different codes
- In particular building basic blocks with a flexible interface (in python), and strong underlying C/C++ engine
- leaving to the user the capability to customize their final models