Time variability @ gammapy

Claudio Galelli - LUTh

The current state

What we have:

- Convenience functions for some useful evaluators of variability
- Doubling time and structure function for flare shapes
- Possibility of using the rebinning functionality on the time axis (ty Atreyee)
- A first algorithm for simulations based on the psd (to be modified during coding sprint)

What is in progress:

- More complex algorithms of simulation
- Envelopes of the power spectral distribution
- A fitting algorithm for the psd model (gamma2024)
- Lightcurve class

Emmanoulopoulos algorithm

```
def Emmanoulopoulos_lightcurve_simulator(pdf, psd, npoints, spacing, pdf_params=None, psd_params=None, random_state="random-seed", imax = 1000,
    npoints_ext = npoints*nchunks
    lc_norm, taxis = TimmerKonig_lightcurve_simulator(psd, npoints_ext, spacing, power_spectrum_params=psd_params, random_state=random_state, m

    random_state = get_random_state(random_state)

    ndiv = npoints_ext//(2*nchunks)
    lc_norm = lc_norm[npoints_ext//2-ndiv : npoints_ext//2+ndiv]
    taxis = taxis[: npoints]

    fft_norm = np.fft.rfft(lc_norm)

    a_norm = np.abs(fft_norm)/npoints
    phi_norm = np.angle(fft_norm)

    if "scale" in pdf_params: scale = pdf_params.get("scale")
    else: scale = 1

    xx = np.linspace(0, scale*10, 1000)
    lc_sim = np.interp(random_state.rand(npoints), np.cumsum(pdf(xx, **pdf_params))/np.sum(pdf(xx, **pdf_params)), xx)

    lc_sim = lc_sim - lc_sim.mean()
    lc_sim = (lc_sim/lc_sim.std())*lc_norm.std()
    lc_sim += lc_norm.mean()

    nconv = True
    i=0
    while nconv and i<imax:
        i+=1
        fft_sim = np.fft.rfft(lc_sim)
        a_sim = np.abs(fft_sim)/npoints
```

Pro: adds PDF model in the simulation

Con: is not “universal”

PSD envelopes and fitting

[53]:
```
def lightcurve_psd_envelope(psd, npoints, spacing, pdf=None, nsims=10000, pdf_params=None, psd_params=None, simulator="TK", mean=None, std=None
    if simulator== "TK" : tseries, taxis = TimmerKonig_lightcurve_simulator(psd, npoints, spacing, power_spectrum_params=psd_params, mean=mean,
    elif simulator== "EMM": tseries, taxis = Emmanoulopoulos_lightcurve_simulator(pdf, psd, npoints, spacing, pdf_params=pdf_params, psd_params
    freqs, pg = periodogram(tseries, spacing.value)
    envelopes_psd = np.empty((nsims, len(pg)-1))
    envelopes_psd[0] = pg[1:]

    for _ in range(1, nsims):
        if simulator== "TK" : tseries, taxis = TimmerKonig_lightcurve_simulator(psd, npoints, spacing, power_spectrum_params=psd_params, mean=m
        else: tseries, taxis = Emmanoulopoulos_lightcurve_simulator(pdf, psd, npoints, spacing, pdf_params=pdf_params, psd_params=psd_params, m

        freqs, pg = periodogram(tseries, spacing.value)
        envelopes_psd[_] = pg[1:]

    return envelopes_psd, freqs[1:]
```

Based on Max-Morbeck, 2014

Basically a x2 on the envelope

[54]:
```
def prob_fit_moerbeck(psd_params_list, pgram, npoints, spacing, psd, pdf=None, pdf_params=None, simulator="TK", nsims=10000, mean=None, std=Non

    psd_params_keys = list(inspect.signature(psd).parameters.keys())

    if len(psd_params_keys[1:]) != len(psd_params_list): raise ValueError("parameter values do not correspond to the request from the psd funct

    psd_params = dict(zip(psd_params_keys[1:], psd_params_list))

    envelopes, freqs = lightcurve_psd_envelope(psd, npoints, spacing, pdf=pdf, pdf_params=pdf_params, psd_params=psd_params, simulator=simulato

    if len(envelopes[0])!= len(pgram): raise ValueError("required length is different than data length!")

    obs = (pgram - envelopes.mean(axis=0))**2/envelopes.std(axis=0)**2
    sim = (envelopes - envelopes.mean(axis=0))**2/envelopes.std(axis=0)**2
    sumobs = np.sum(obs)
    sumsim = np.sum(sim, axis=-1)
    sign = len(np.where(sumobs>=sumsim)[0])/nsims

    return sign
```

UNNAMED GAMMAPY TIME DOMAIN ABSTRACT

Gamma-ray astrophysics increasingly focuses on time-domain studies of variable sources like GRBs and AGNs. As the foundation for CTAO's science analysis tools, the open Python analysis library Gammapy must adapt to support these advancements. This contribution highlights the recent expansion of Gammapy's time-domain capabilities and outlines its near-future plans, particularly regarding power spectrum analysis in Fourier space. Estimating the spectral behavior of a light curve in the frequency domain is crucial for studying variable astrophysical sources. We present Gammapy's novel framework for simulating light curves from a power spectrum model using the Timmer & Koenig and Emmanoulopoulos algorithms. This framework includes a fitting procedure to assess the behavior of observed light curves in the frequency domain. We will also demonstrate the expected reconstruction power of CTAO using this method under various observational scenarios.

Lightcurve

The final idea is to have something like stingray, ctaagnvar and similar packages focused on time-domain

```
[ ]: lightcurve.periodogram ##based on scipy
     lightcurve.psd ## {"model": function, "parameters": dict of parameters}
     lightcurve.envelope() ## statistical distribution of periodogram around each frequency
     lightcurve.fit_psd()
```

Discussion point: the philosophy of Lightcurve:

- Keep the strong line from FluxPoints?
- Separate completely?
- Rewrite and specialize only some tools?

  Example problem: slicing lightcurves to one time bin does not work with simple inheritance - fundamental step in to_table() and write()

Bonus 1

Next week I will be presenting gammapy @ CRIS-MAC 2024

https://agenda.infn.it/event/36661/

Suggestions on points to highlight or structure are well accepted

Bonus 2

What is the correct treatment for an upper limit?