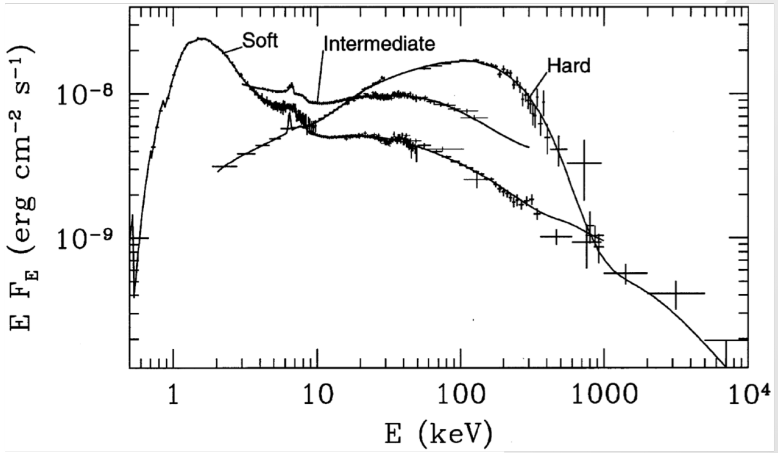
Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

Timing analysis of X-ray binaries with Stingray

Eleonora Veronica Lai, Matteo Bachetti, Daniela Huppenkothen, Matteo Lucchini, and the Stingray community

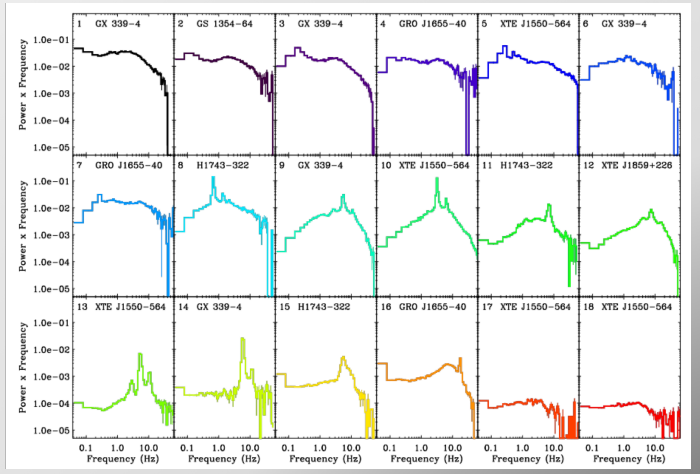
The many tones of accretion - to the memory of Tomaso Belloni - 6-11 Sept 2026

Scientific Rationale

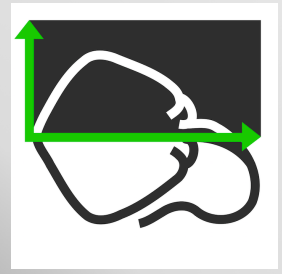


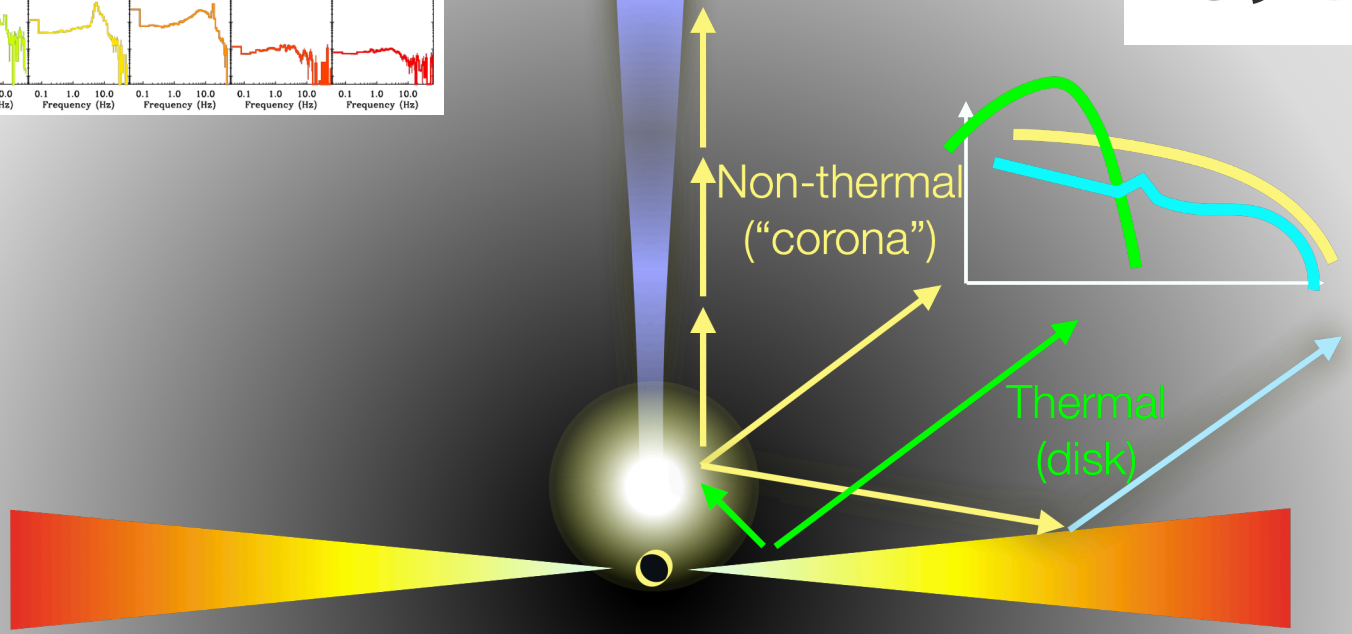
Zdziarski (2000)

Heil et al. (2015)



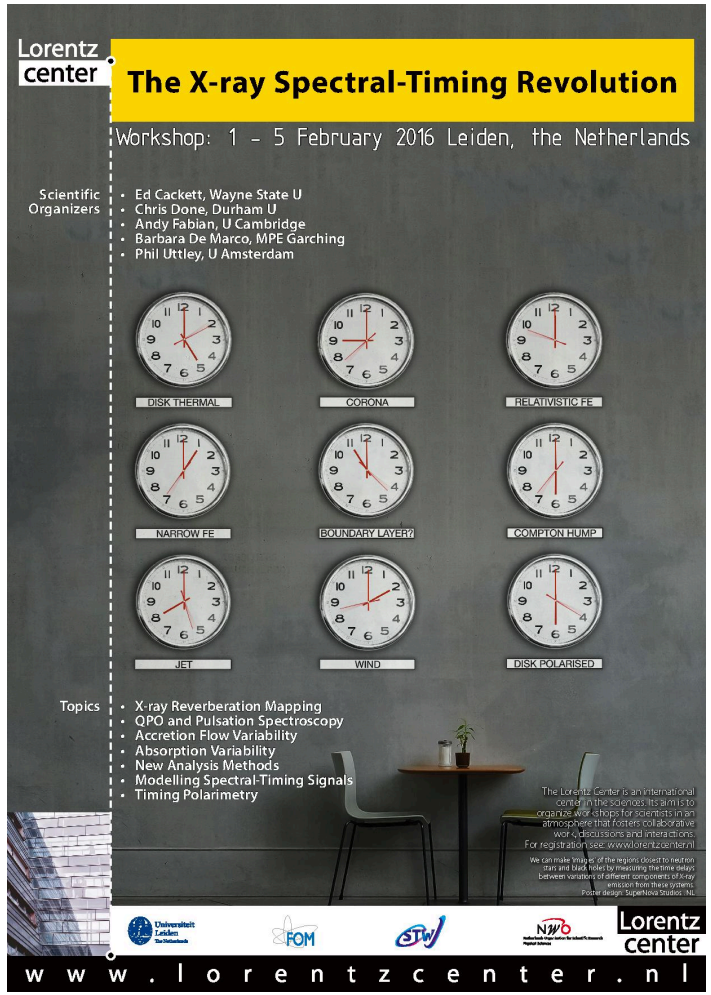
Huppenkothen et al. (2019)
Bachetti et al. (2024)
Lai et al. (2026)





Spectra or variability?
Is it possible to use the full information?

History

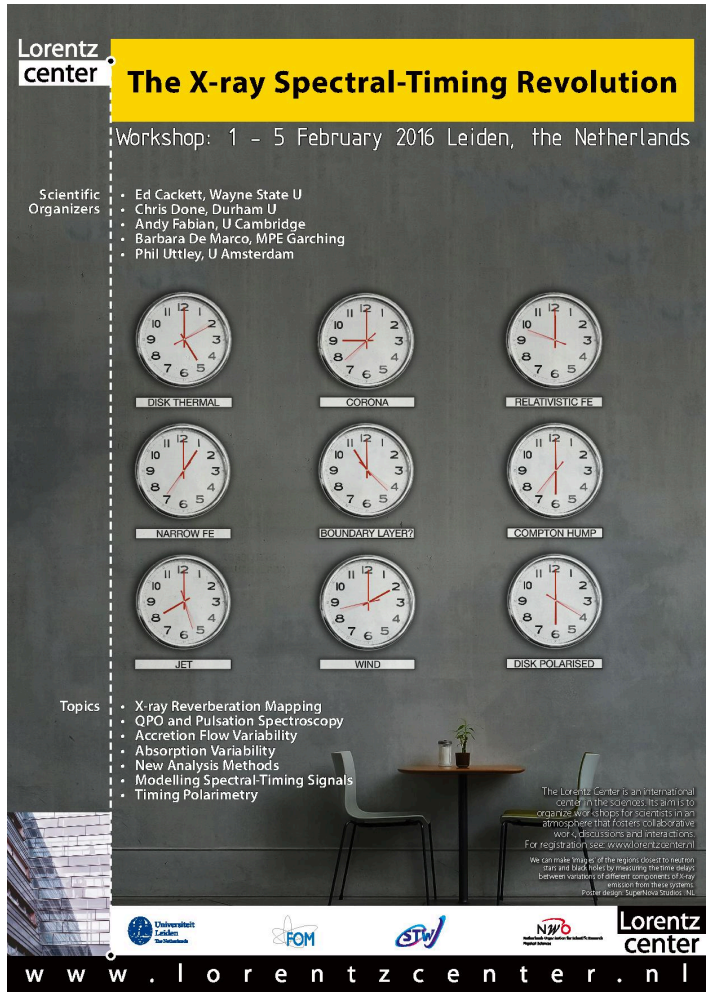


Spectral analysis	Timing analysis (+ lags)	Spectral-timing
• Xspec • Sherpa • ISIS • (…)	• (XRONOS) POWSPEC • SITAR, Isisscripts.sl	∅

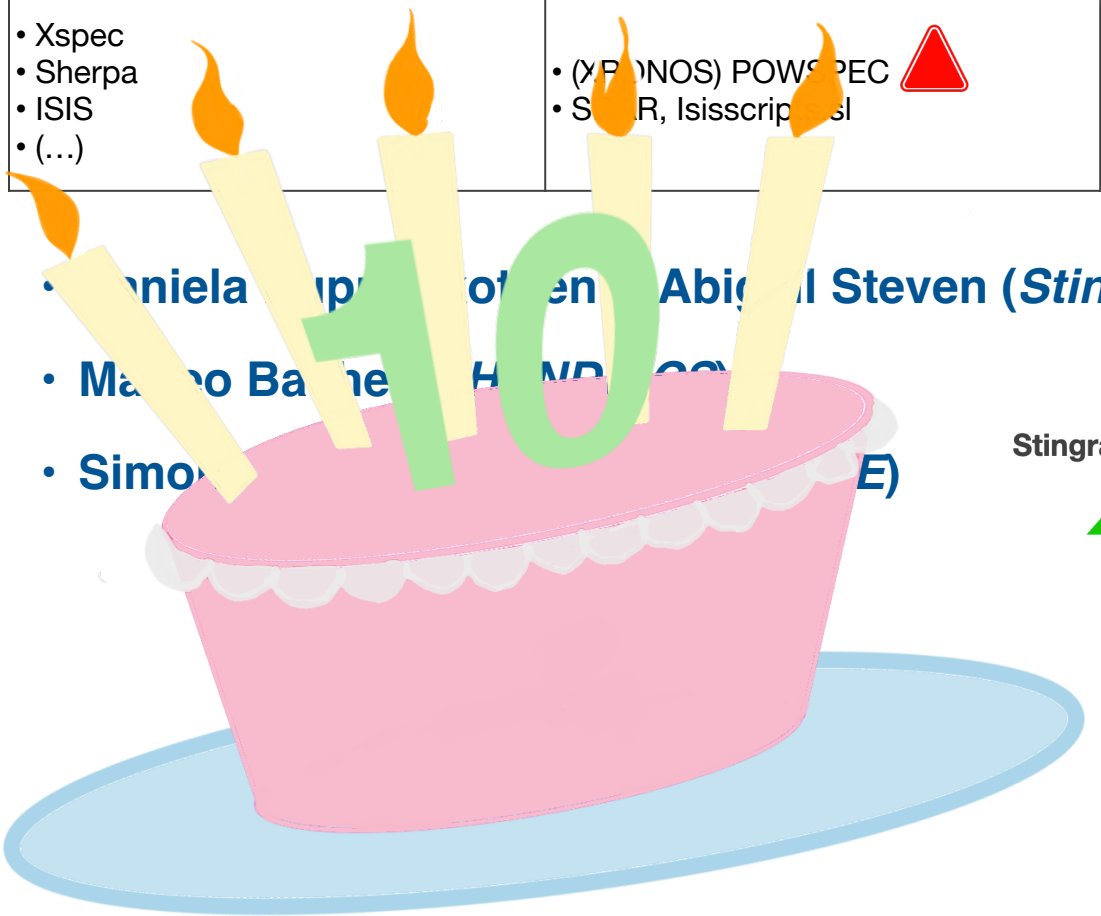
- **Daniela Huppenkothen & Abigail Steven (*Stingray's original*)**
- **Matteo Bachetti (*HENDRICS*)**
- **Simone Migliari & Paul Balm (*DAVE*)**

Stingray: Next-Generation Spectral Timing

History



Spectral analysis	Timing analysis (+ lags)	Spectral-timing
• Xspec • Sherpa • ISIS • (…)	• (XRONOS) POWSPEC • STAR, Isisscripts	∅



- Daniela [illegible] Steven (*Stingray's original*)
- Marco Ba[illegible]
- Simo[illegible]E)

Stingray: Next-Generation Spectral Timing

What is Stingray able to do?

Timing

- Light curves operations
- Periodograms
- Power density spectra
- Lomb-Scargle
- Dynamical Power spectra
- Epoch folding for pulsar search
- Periodogram modelling
- Polarimetric light curves
- *Improving Bispectrum - under review, tnx Ben Rickets :)*

Spectral

- Hardness-intensity diagrams
- Power colours

What is Stingray able to do?

Timing

- Light curves operations
- Periodograms
- Power density spectra
- Lomb-Scargle
- Dynamical Power spectra
- Epoch folding for pulsar search
- Periodogram modelling (MLE, Bayesian)
- Polarimetric light curves
- *Improving Bispectrum - under review, tnx Ben Rickets :)*

Spectral-timing

Frequency spectrum:

- Time/Phase lags
- Coherence

Energy spectrum:

- Rms-intensity diagrams
- Time/Phase lags
- Coherence
- Rms energy spectra
- Covariance spectra

Spectral

- Hardness-intensity diagrams
- Power colours

Being aware of instrumental systematics!

An open-source environment

UNIAM PRIN MIUR 2017
SpecTemPolar! PRIN INAF 2019
INAF ISTITUTO NAZIONALE DI ASTROFISICA
Google Summer of Code 2016, 2017, 2018, 2020, 2021, 2022, 2023, 2024, 2025, 2026

- Software:
  - Python **open-source** software in a Github-based workflow
- Developers:
  - Astronomers
  - Google Summer of Code students
- Community outreach:
  - Public **Slack channel**
  - **Talks**
  - Hackatons/**Tutorials**

Platform ▾ Solutions ▾ Resources ▾ Open Source ▾ Enterprise ▾ Pricing

Stingray

Next-generation spectral-timing software for astronomy applications and beyond.

38 followers https://stingray.science

Overview | Repositories 12 | Projects | Packages | People 6

Pinned

stingray Public
Anything can happen in the next half hour (including spectral timing made easy)!
Python ☆ 221 ⑂ 188

HENDRICS Public
Shell scripts for spectral-timing analysis of X-ray astronomical data.
Jupyter Notebook ☆ 26 ⑂ 18

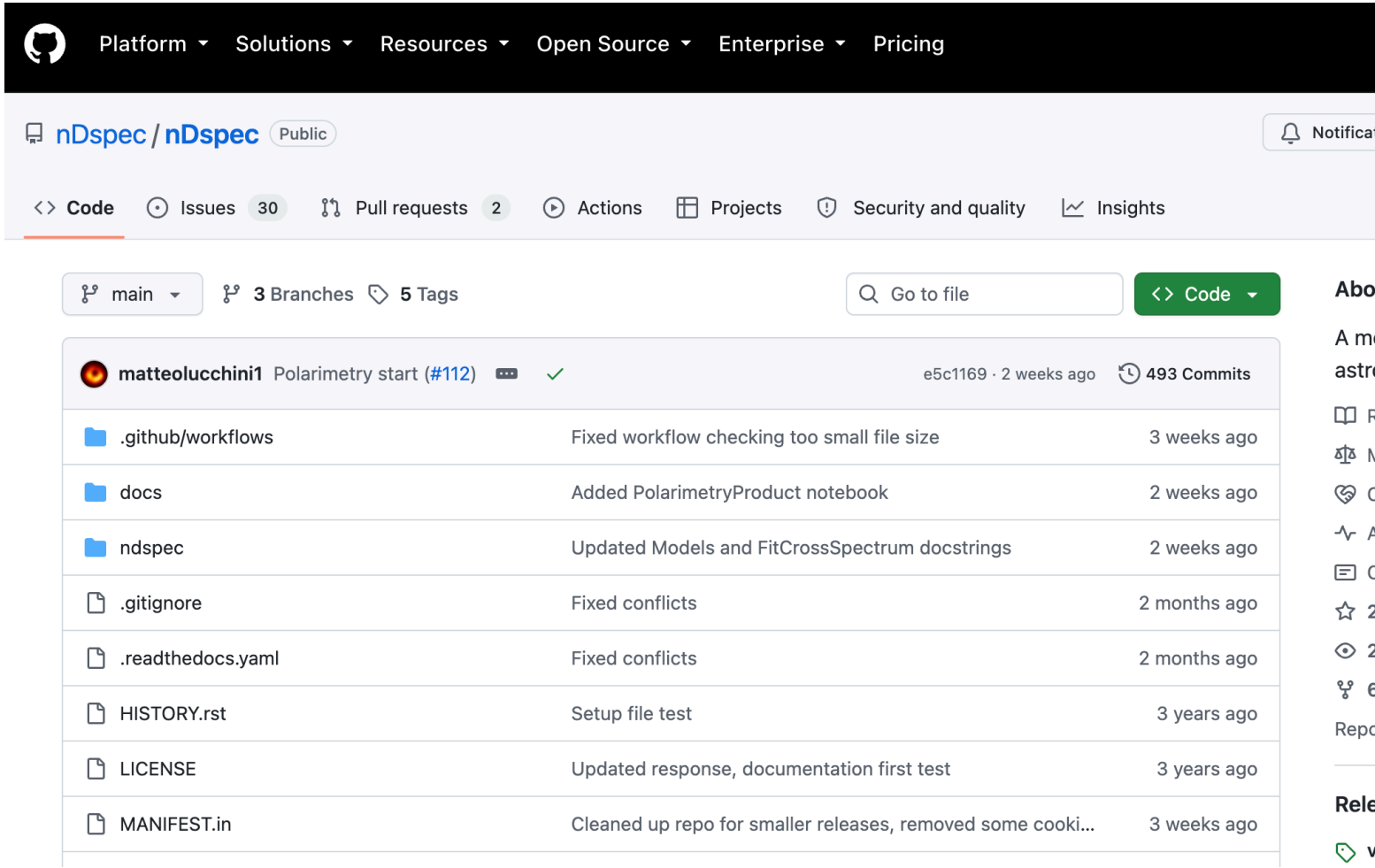
dave Public
A GUI for spectral-timing analysis of X-ray astronomical data.
JavaScript ☆ 16 ⑂ 15

notebooks Public
Tutorial notebooks for Stingray
Jupyter Notebook ☆ 35 ⑂ 45

Modelling with nDspec

https://github.com/nDspec/nDspec

- Python-based modelling package for spectral, spectral-timing and — in future releases — polarimetry data.
- 1D products as a function of photon energy/and or frequency (e.g. in output from Stingray)
- 2D data, and to joint fit of different datasets and observations
- It fully supports Bayesian sampling using common MCMC and nested sampling packages

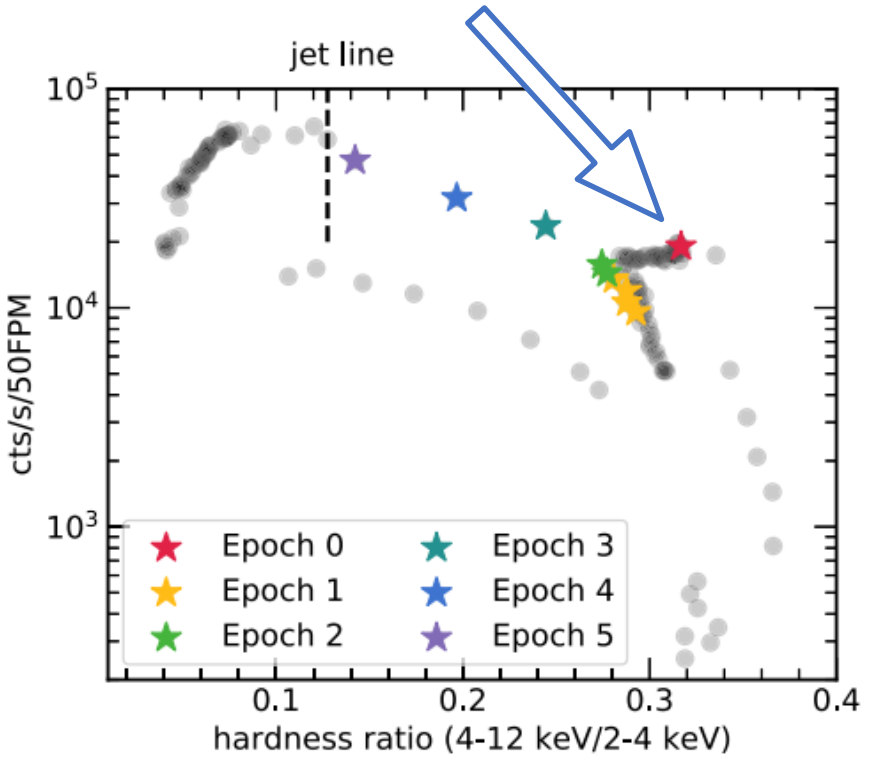


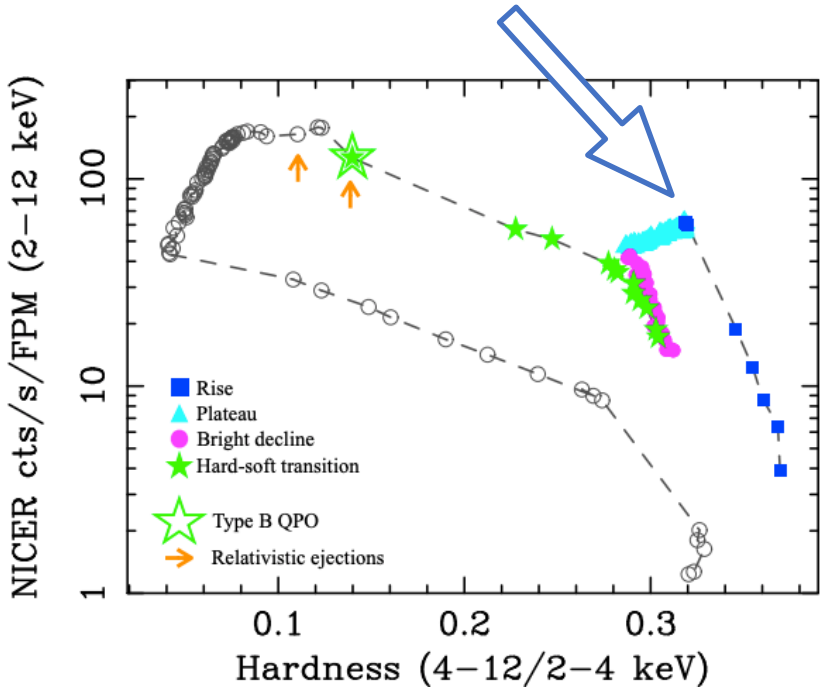
A basic tutorial

MAXI J1820+020

- ObsID: *1200120106*
- Date: 2018-03-21
- NICER
- Rise/Epoch 0

De Marco et al. (2021)





Wang et al. (2021)

https://zenodo.org/records/13149195

Light curve from event files

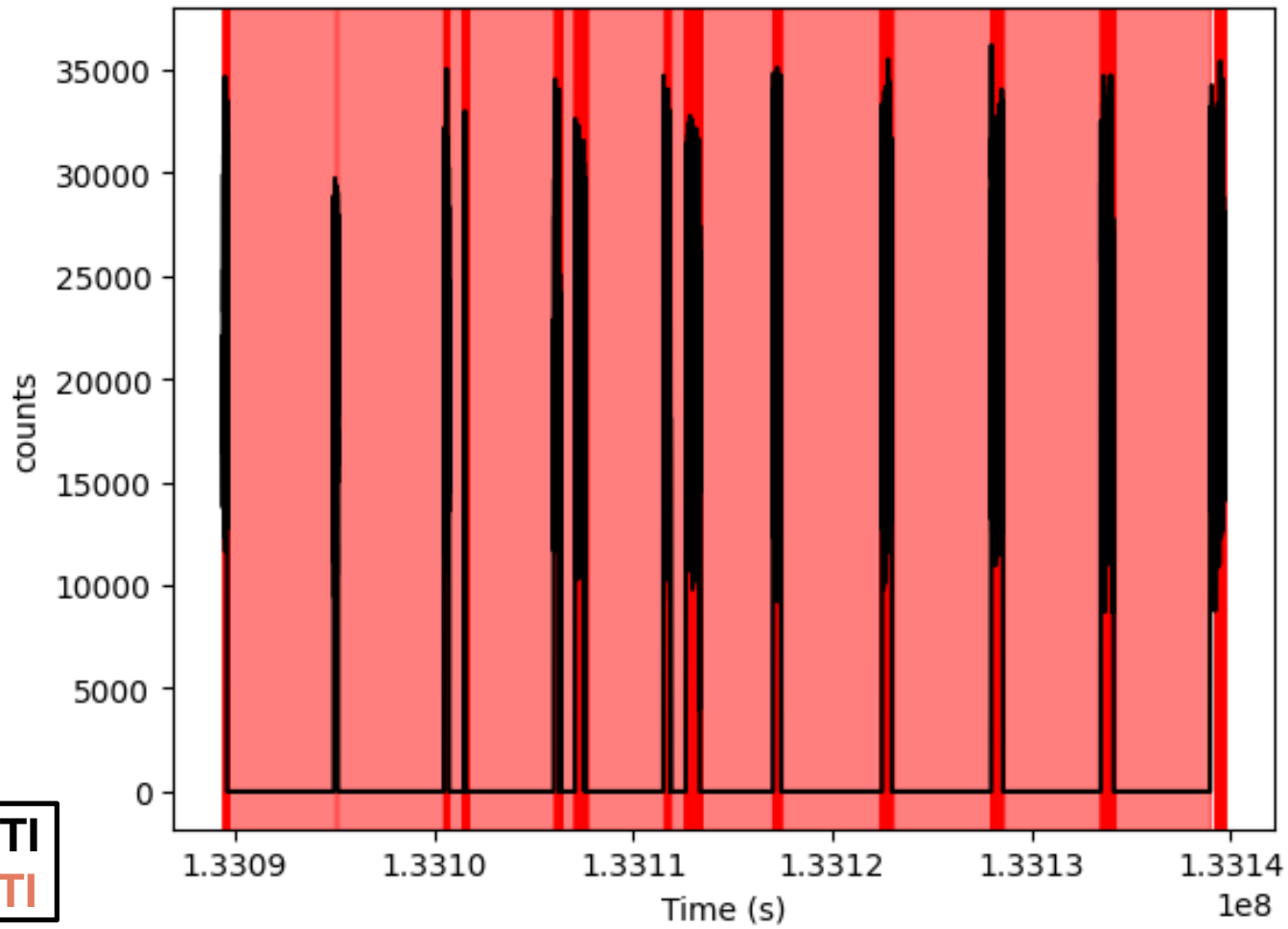
```
import numpy as np
from stingray import EventList

# input event file
fname = "ni1200120106_0mpu7_cl_bary.evt"

events = EventList.read(fname, "hea")
events.fname = fname
```

```
# Create light curve
lc_raw = events.to_lc(dt=1)

lc_raw.plot()
```

Light curve from event files

```
import numpy as np
from stingray import EventList

# input event file
fname = "ni1200120106_0mpu7_cl_bary.evt"

events = EventList.read(fname, "hea")
events.fname = fname
```

```
# Create light curve
lc_raw = events.to_lc(dt=1)

lc_raw.plot()
```



Averaged power spectrum and poisson noise level

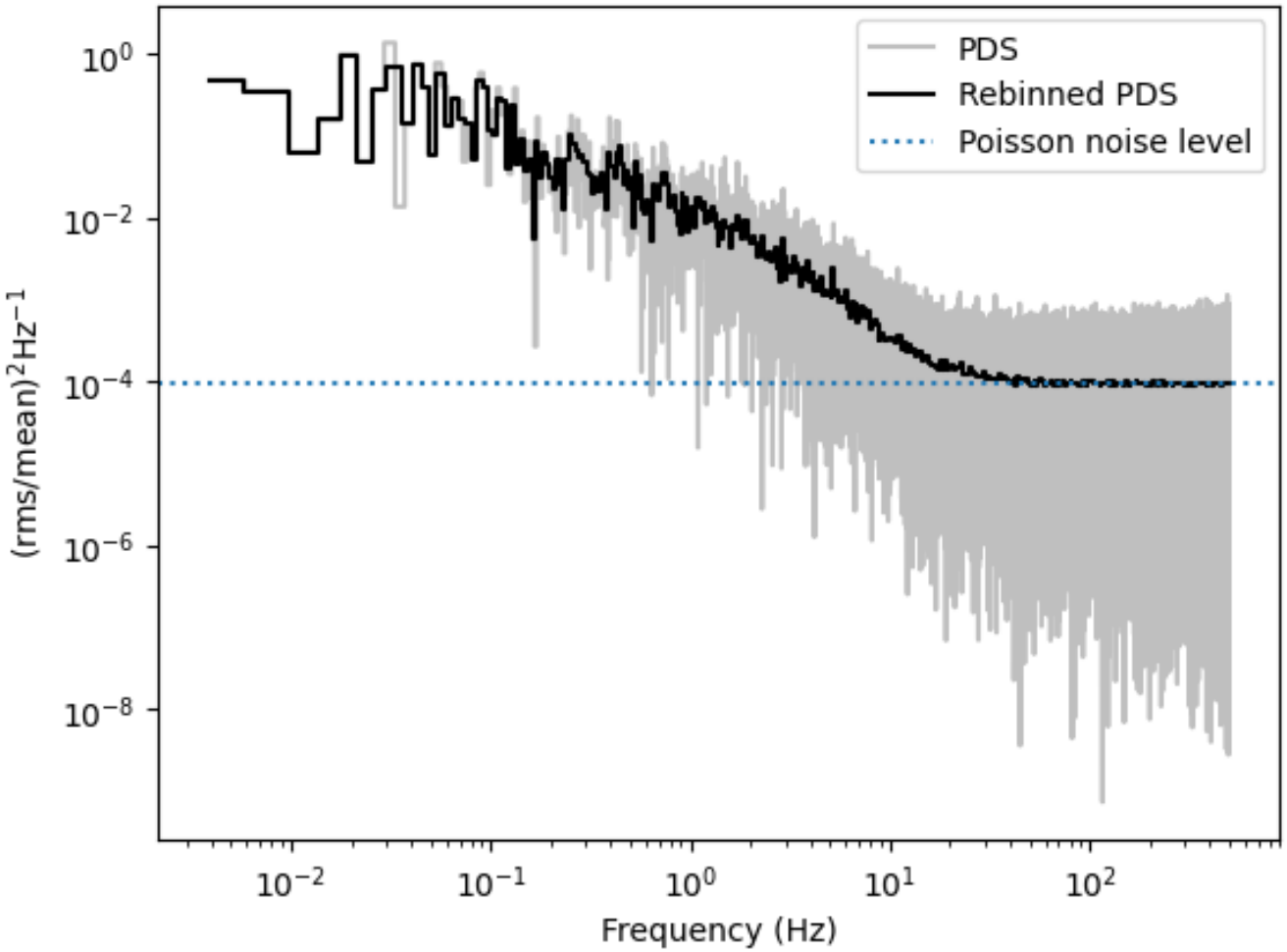
```
from stingray import AveragedPowerspectrum
from stingray.fourier import poisson_level, get_average_ctrate
```

```
segment_size = 256
dt = 0.001

# Averaged power spectrum
pds = AveragedPowerspectrum.from_events(events, segment_size=segment_size, dt=dt, norm="frac", use_common_mean=True)

# Rebin
pds_reb = pds.rebin_log(0.02)

# Calculate the mean count rate
ctrate = get_average_ctrate(events.time, events.gti, segment_size)

# Calculate the Poisson noise level
noise = poisson_level("frac", meanrate=ctrate)
```

Averaged power spectrum and poisson noise level

```
from stingray import AveragedPowerspectrum
from stingray.fourier import poisson_level, get_average_
```

```
segment_size = 256
dt = 0.001

# Averaged power spectrum
pds = AveragedPowerspectrum.from_events(events, segment_

# Rebin
pds_reb = pds.rebin_log(0.02)

# Calculate the mean count rate
ctrate = get_average_ctrate(events.time, events.gti, seg

# Calculate the Poisson noise level
noise = poisson_level("frac", meanrate=ctrate)
```



Spectral-timing analyses: Lags vs Frequency

```
from stingray import AveragedCrossspectrum
```

```
# Energy bands
ref_band = [1.5, 3]
sub_band = [0.5, 1]
events_ref = events.filter_energy_range(ref_band)
events_sub = events.filter_energy_range(sub_band)

# Computing the Averaged Cross spectrum
cs = AveragedCrossspectrum.from_events(events_sub, events_ref, segment_size=2, dt=0.005, norm="frac")

# Rebin
cs_reb = cs.rebin_log(0.04)
```

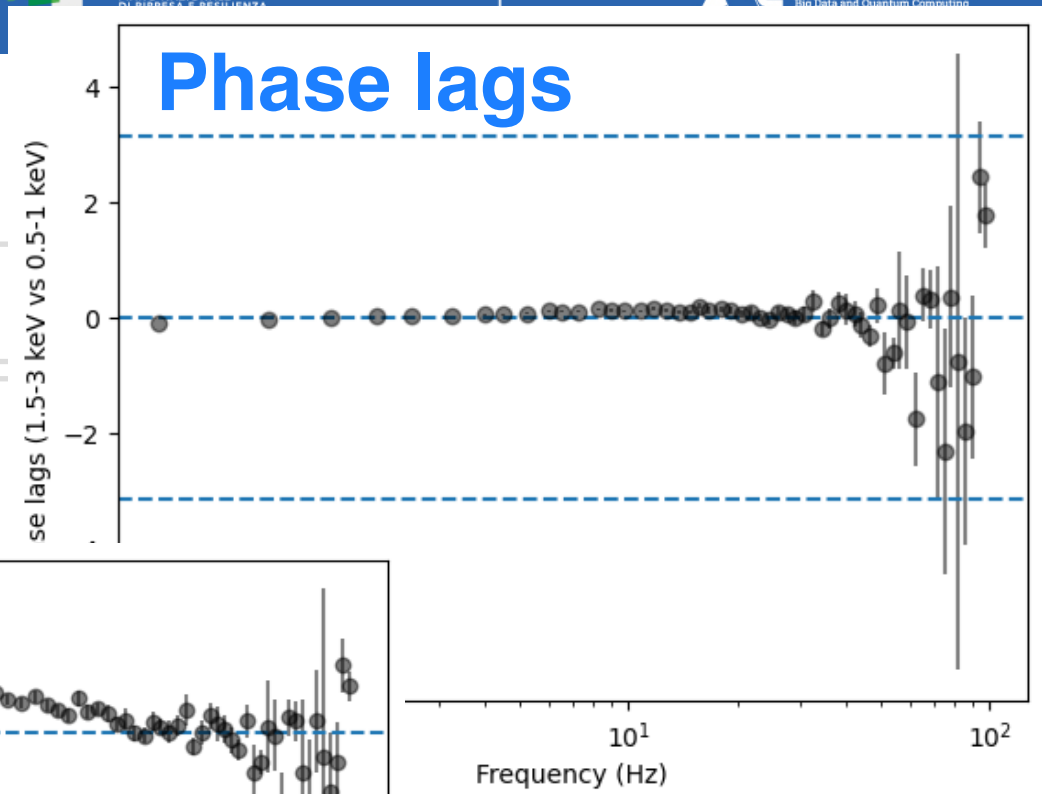
```
# Spectral-timing made easy
lag, lag_e = cs_reb.time_lag()
phlag, phlag_e = cs_reb.phase_lag()
```

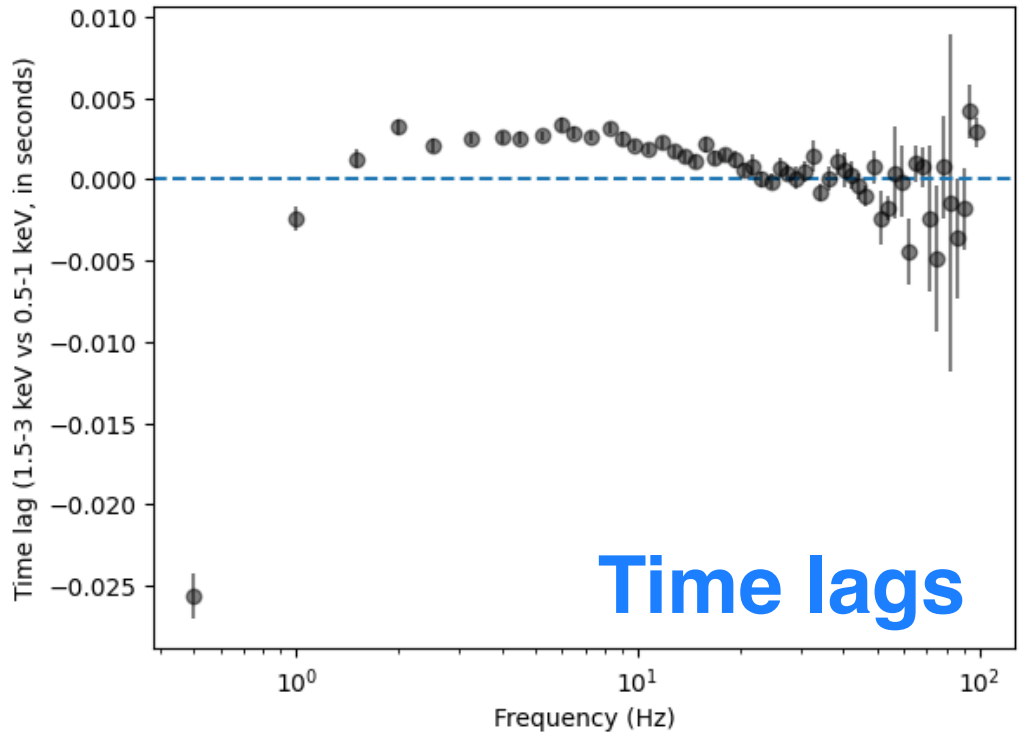
Spectral-timing analyses: Lags vs

```
from stingray import AveragedCrossspectrum
```

```
# Energy bands
ref_band = [1.5, 3]
sub_band = [0.5, 1]
events_ref = events.filter_energy_r
events_sub = events.filter_energy_r

# Computing the Averaged Cross spec
cs = AveragedCrossspectrum.from_eve

# Rebin
cs_reb = cs.rebin_log(0.04)
```

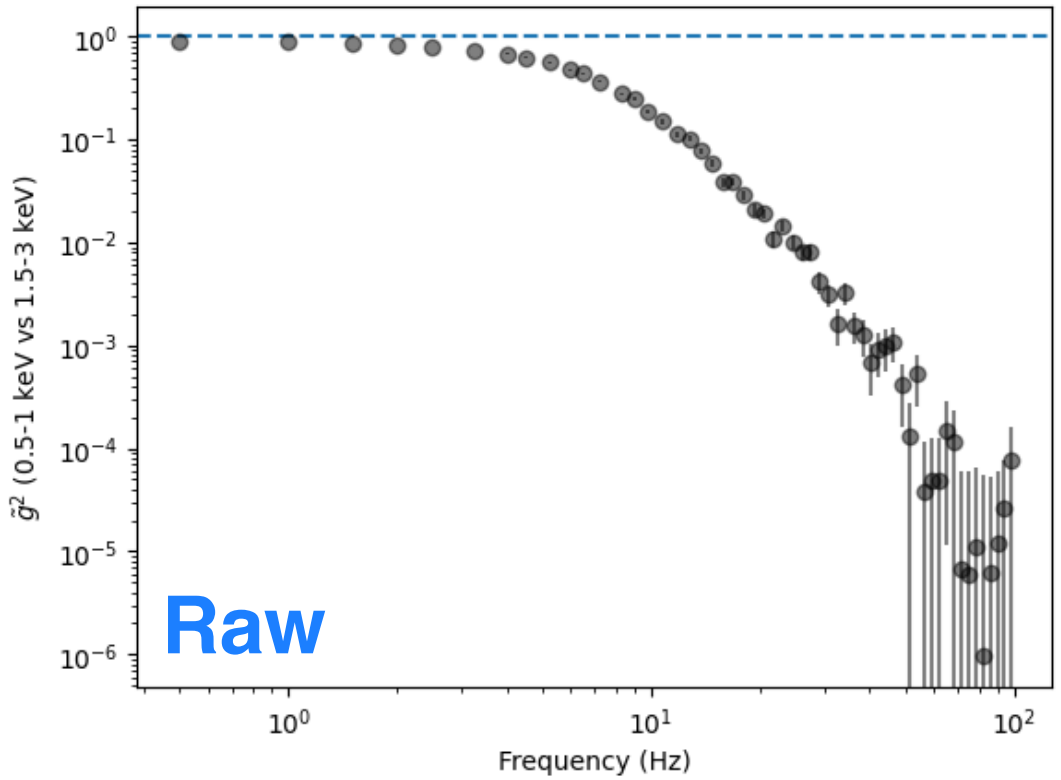
```
# Spectral-timing made easy
lag, lag_e = cs_reb.time_lag()
phlag, phlag_e = cs_reb.phase_lag()
```

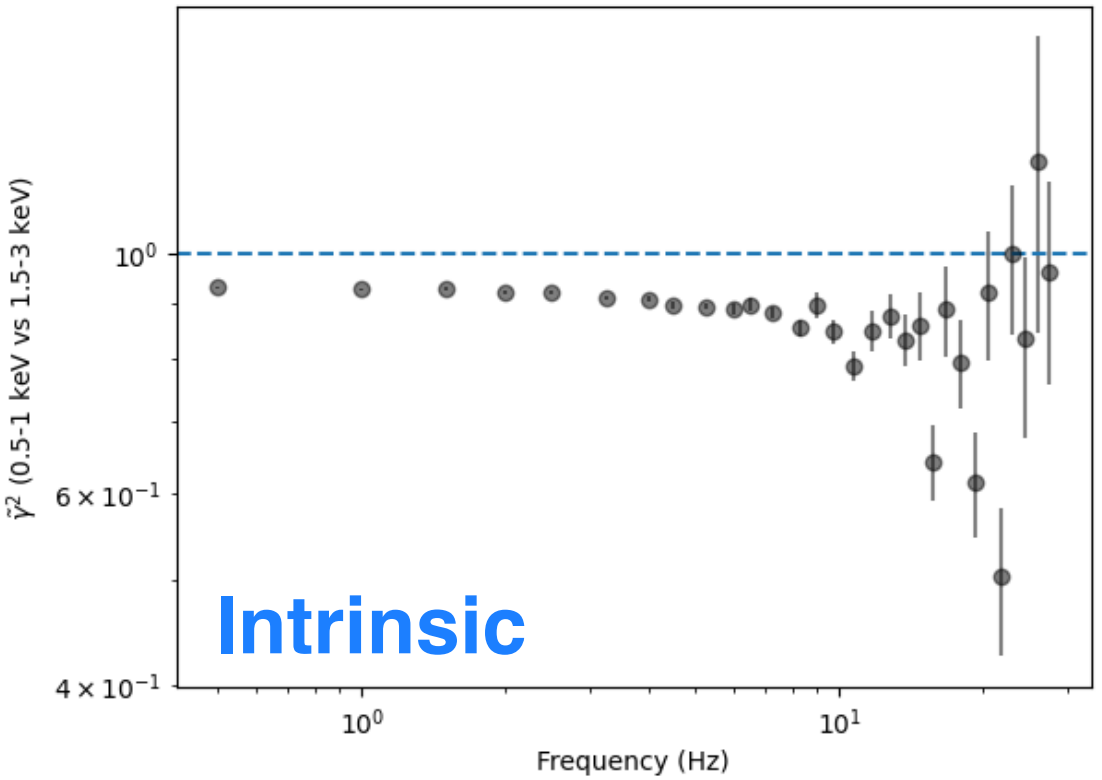




Spectral-timing analyses: Coherence

```
coh_raw, coh_e_raw = cs_reb.raw_coherence()
coh_intr, coh_e_intr = cs_reb.intrinsic_coherence()
```





Contents lists available at ScienceDirect

Astronomy and Computing

journal homepage: www.elsevier.com/locate/ascom

Full Length Paper

High performance Stingray☆

Eleonora Veronica Lai [a,*], Matteo Bachetti [a], Maura Pilia [a], Daniela Huppenkothen [b], Matteo Lucchini [b], Guglielmo Mastroserio [c], Hamza El Byad [a,d]

[a] INAF-Osservatorio Astronomico di Cagliari, via della Scienza 5, Selargius, I-09047, (CA), Italy
[b] Anton Pannekoek Institute for Astronomy, University of Amsterdam, Science Park 904, Amsterdam, 1098, XH, The Netherlands
[c] Scuola Universitaria Superiore IUSS Pavia, Palazzo del Broletto, Piazza della Vittoria 15, Pavia, I-27100, PV, Italy
[d] Department of Mathematics and Computer Science, University of Cagliari, Italy

Lai et al. (2026)

ARTICLE INFO

Keywords:
Timeseries
Python
Open-source
Variability
High energy astronomy
High performance computing

ABSTRACT

X-ray astrophysical objects show variability on a wide range of timescales, i.e. from fractions of second to years. The open-source Python library `stingray` is able to perform time series analyses with a focus on high-energy astrophysics. Comprising the most commonly used Fourier analyses techniques, it also supports a range of additional extensions able to analyse pulsar data, simulate data sets and perform statistical modelling. With the latest release of the code, new Fourier methods and support for additional missions have been implemented, making Stingray more easily adaptable and extendable to other use cases. In this paper we focus on testing the performance and robustness of the latest version of `stingray`. We consider the possible different behaviour of the code when dealing with data sets smaller or larger than the RAM. We address the problem of dealing with large data sets and implemented parallel versions of the slowest methods in this regime. For small data sets, we investigate the possibility of a porting in GPU of key functions of the code.

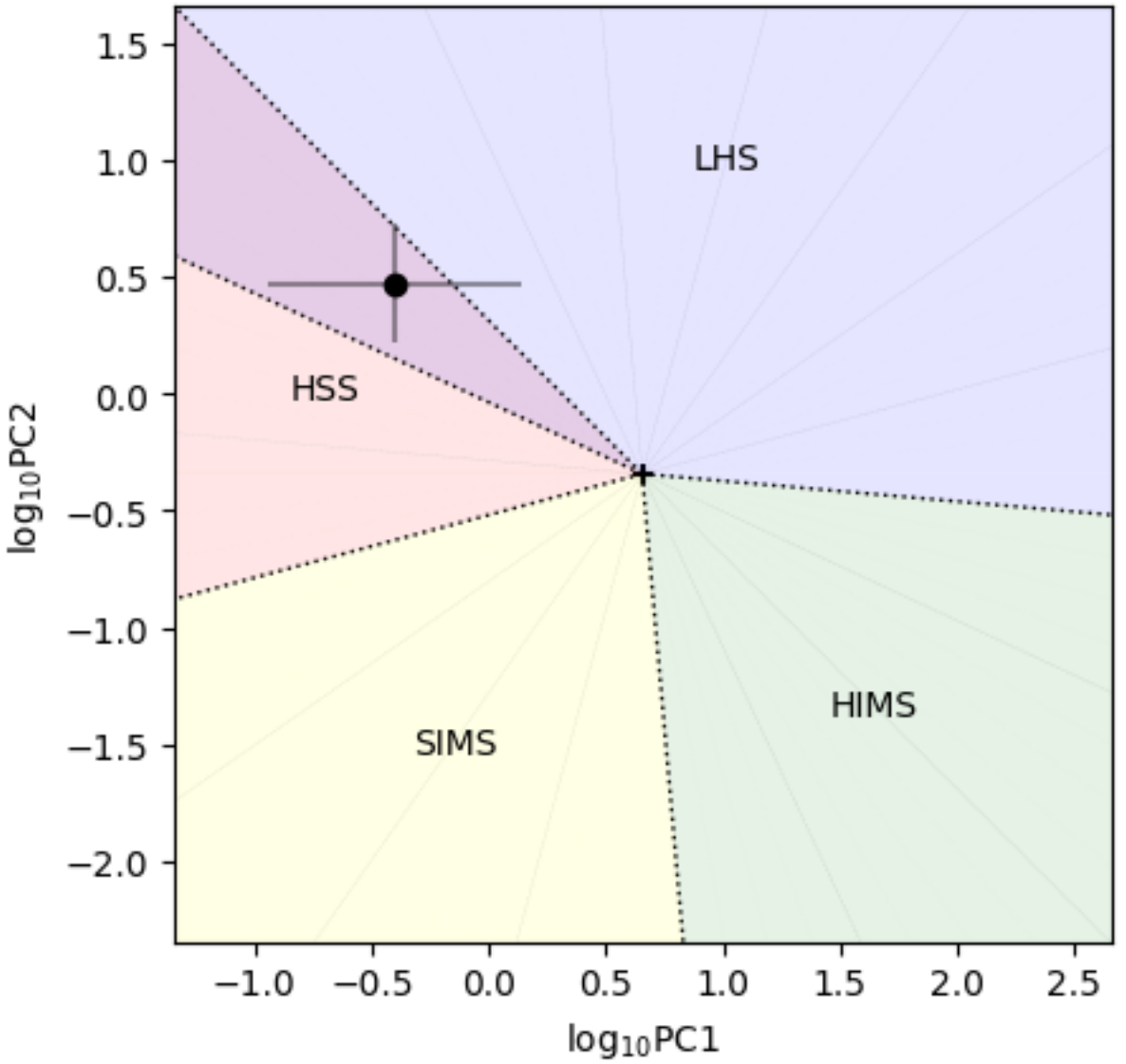
Power colours

```
from stingray.power_colors import power_color, plot_power_colors, DEFAULT_COLOR_CONFIGURATION

segment_size = 256
dt = 0.001

# Averaged power spectrum
pds = AveragedPowerspectrum.from_events(events, segment_size=segment_size, dt=dt, norm="frac", use_common_mean=True)
poisson = poisson_level("frac", meanrate=ctrate)


p1, p1e, p2, p2e = power_color(pds.freq, pds.power, pds.power_err,
                               freq_edges=[1/256, 1/32, 0.25, 2, 16], poisson_power=poisson)


configuration=DEFAULT_COLOR_CONFIGURATION
plot_power_colors(p1, p1e, p2, p2e, plot_spans=True, configuration=configuration)
```

Heil et al. (2015)

Power colours

```
from stingray.power_colors import power_color, plot_power_col

segment_size = 256
dt = 0.001

# Averaged power spectrum
pds = AveragedPowerspectrum.from_events(events, segment_size=
poisson = poisson_level("frac", meanrate=ctrate)


p1, p1e, p2, p2e = power_color(pds.freq, pds.power, pds.power
                               freq_edges=[1/256, 1/32, 0.25,


configuration=DEFAULT_COLOR_CONFIGURATION
plot_power_colors(p1, p1e, p2, p2e, plot_spans=True, configu
```

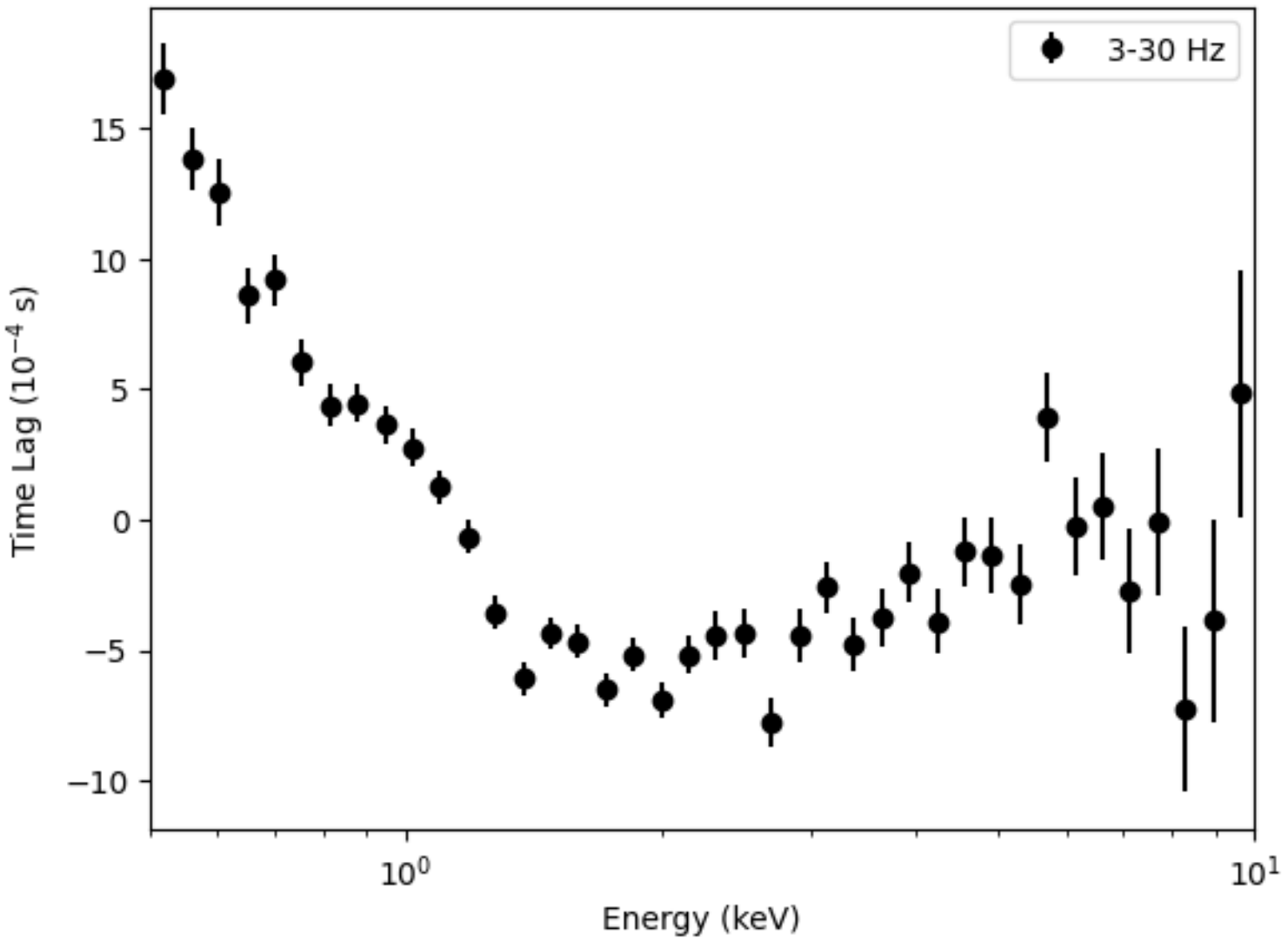


Time lags vs Energy

```
from stingray.varenergyspectrum import LagSpectrum

energy_spec = np.geomspace(0.5, 10, 41)
segment_size = 10
bin_time = 0.001
freq_interval = [3, 30]
ref_band = [0.5, 10]

lagspec_3_30 = LagSpectrum(
    events,
    freq_interval=freq_interval,
    segment_size=segment_size,
    bin_time=bin_time,
    energy_spec=energy_spec,
    ref_band=ref_band,
)
```

Time lags vs Energy

```
from stingray.varenergyspectrum import Lags

energy_spec = np.geomspace(0.5, 10, 41)
segment_size = 10
bin_time = 0.001
freq_interval = [3, 30]
ref_band = [0.5, 10]

lagspec_3_30 = LagSpectrum(
    events,
    freq_interval=freq_interval,
    segment_size=segment_size,
    bin_time=bin_time,
    energy_spec=energy_spec,
    ref_band=ref_band,
)
```



Covariance and RMS spectrum

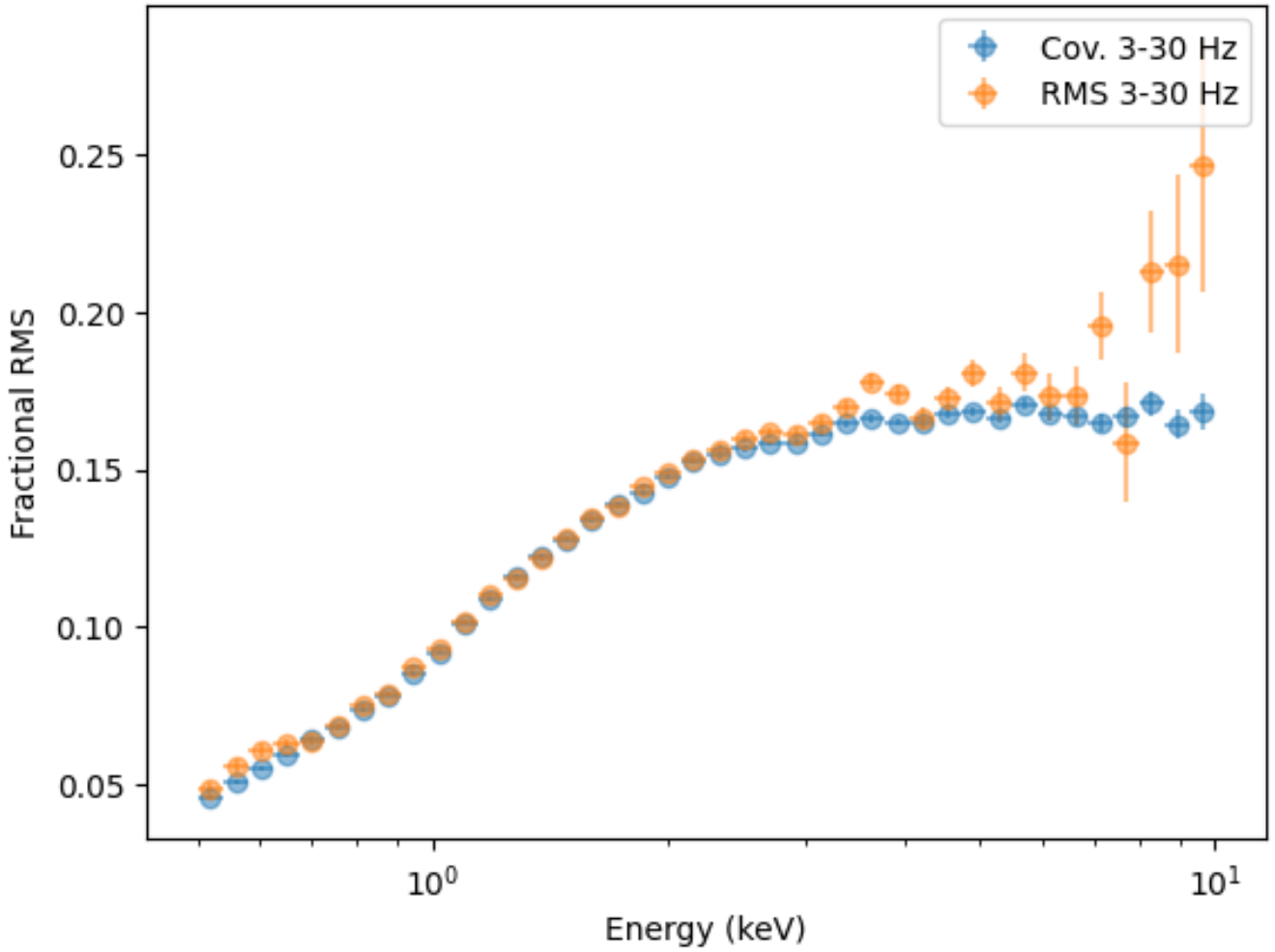
```
from stingray.varenergyspectrum import CovarianceSpectrum, RmsSpectrum

covspec_3_30 = CovarianceSpectrum(
    events,
    freq_interval=[3, 30],
    segment_size=segment_size,
    bin_time=bin_time,
    energy_spec=energy_spec,
    norm="frac",
)
```

```
rmsspec_3_30 = RmsSpectrum(
    events,
    freq_interval=[3, 30],
    segment_size=segment_size,
    bin_time=bin_time,
    energy_spec=energy_spec,
    norm="frac",
)
```

Covariance and RMS spectrum

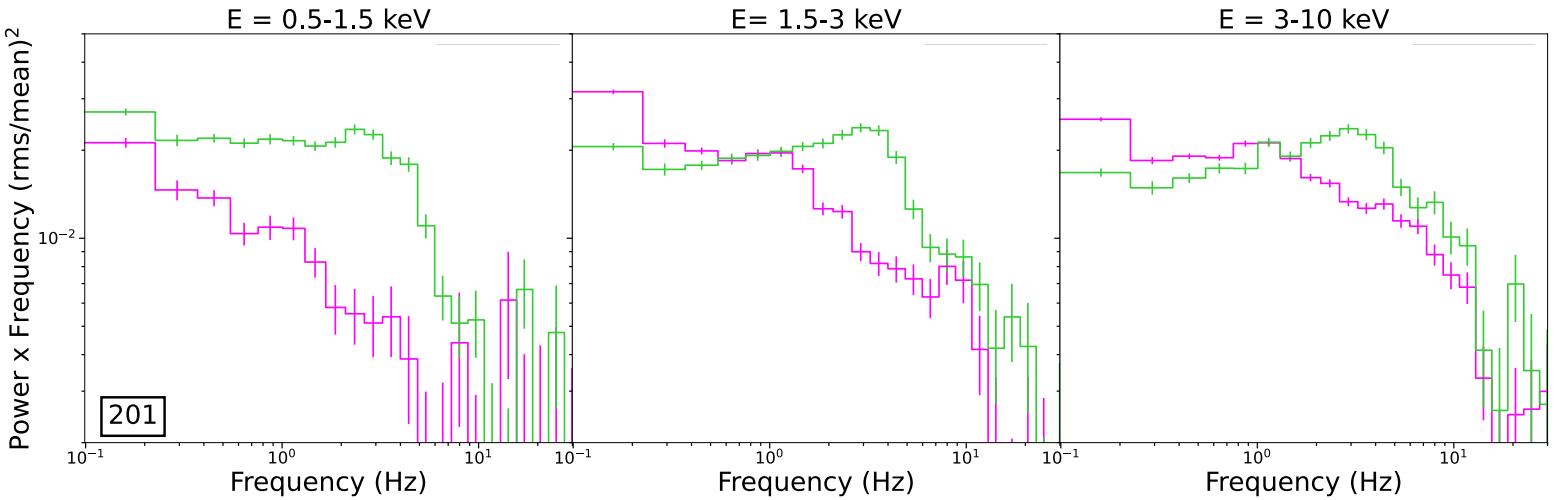
```
from stingray.varenergyspectrum import CovarianceSpec

covspec_3_30 = CovarianceSpectrum(
    events,
    freq_interval=[3, 30],
    segment_size=segment_size,
    bin_time=bin_time,
    energy_spec=energy_spec,
    norm="frac",
)
```

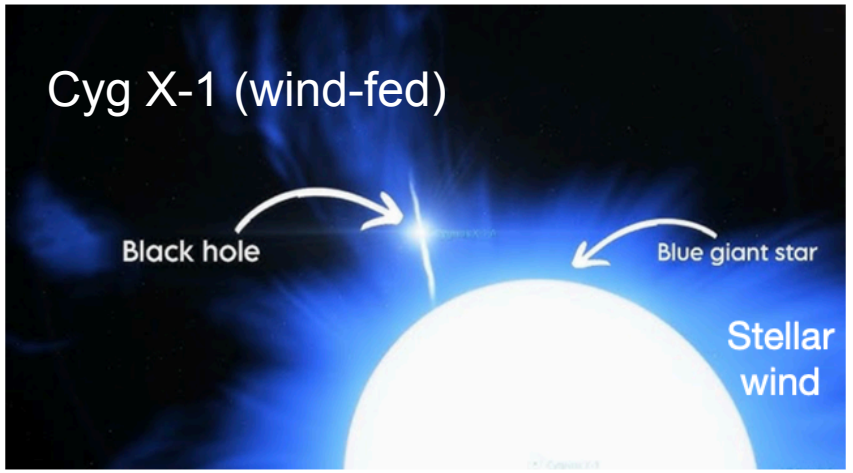
```
rmsspec_3_30 = RmsSpectrum(
    events,
    freq_interval=[3, 30],
    segment_size=segment_size,
    bin_time=bin_time,
    energy_spec=energy_spec,
    norm="frac",
)
```



Spectral-timing for stellar wind studies

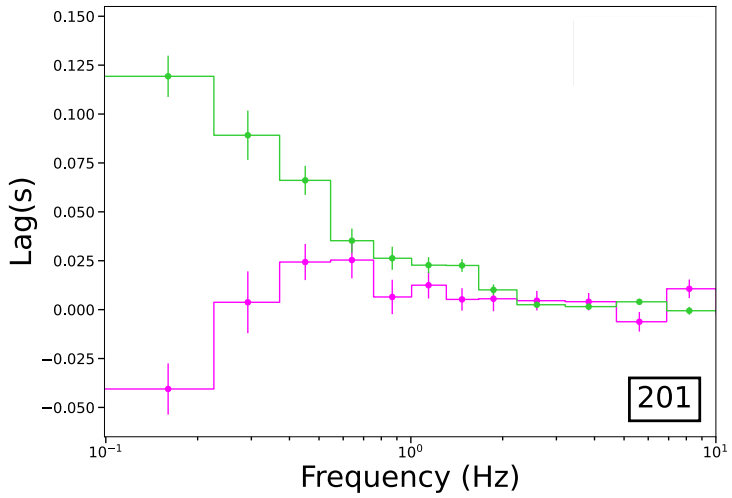
Lai E.V. , De Marco B., Zdziarski A.A., Belloni T. M. et al. (2022)

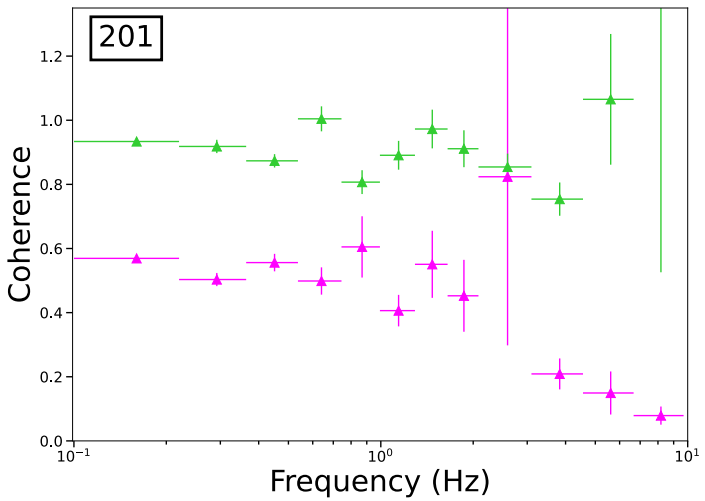




Wind affected

No Wind Affected





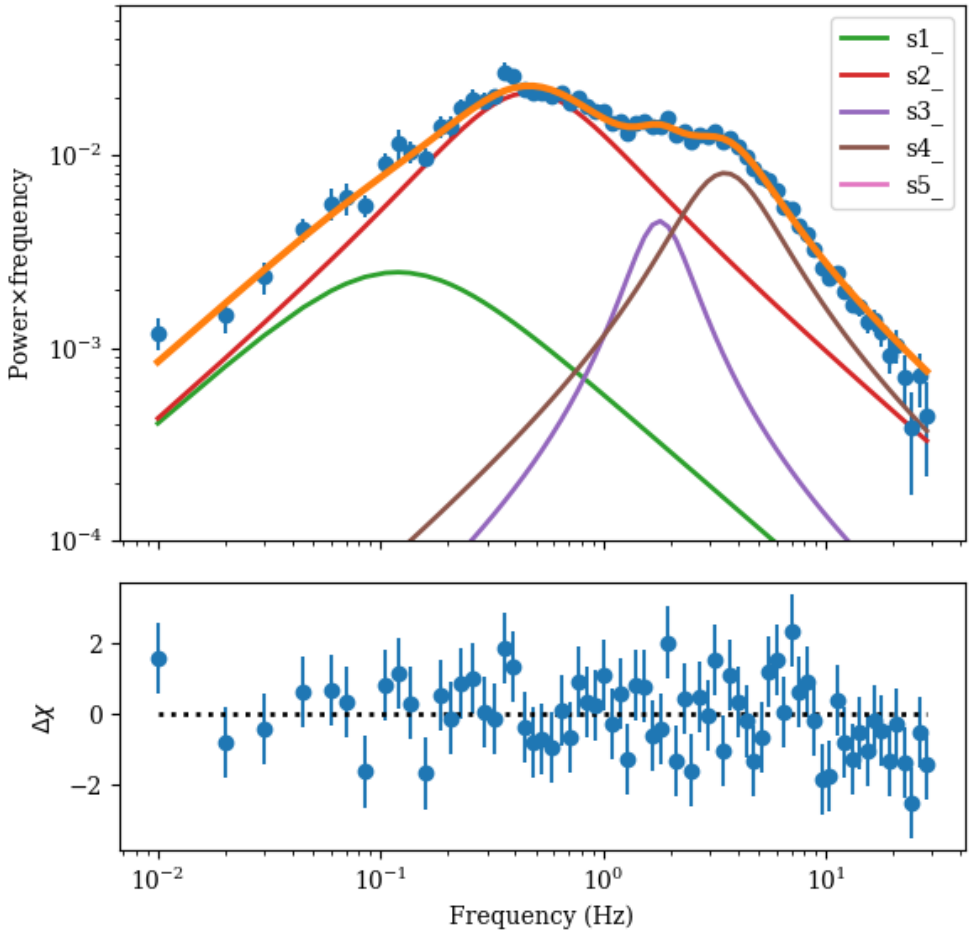
XMM-Newton, ObsID: 0745250201

Spectral-timing for stellar wind studies

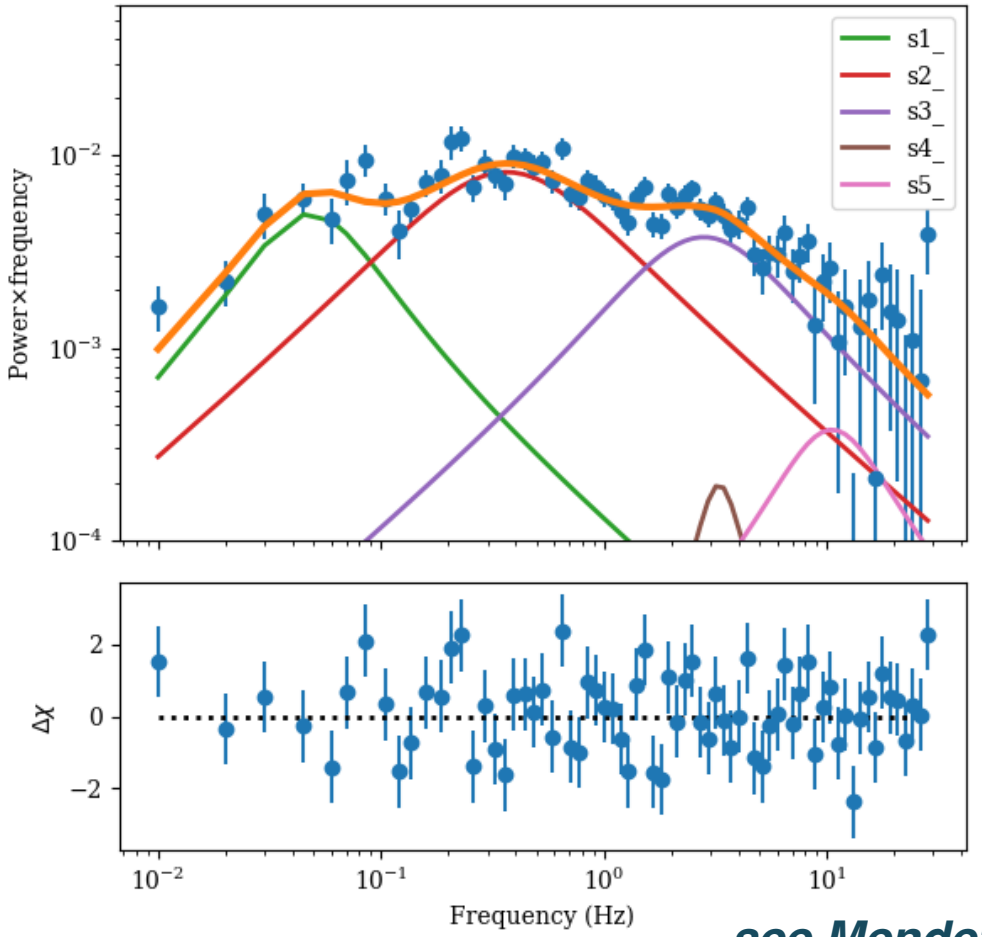
Nicer, ObsID: 0100320106

0.5 - 3 keV

No Wind Affected



Wind affected



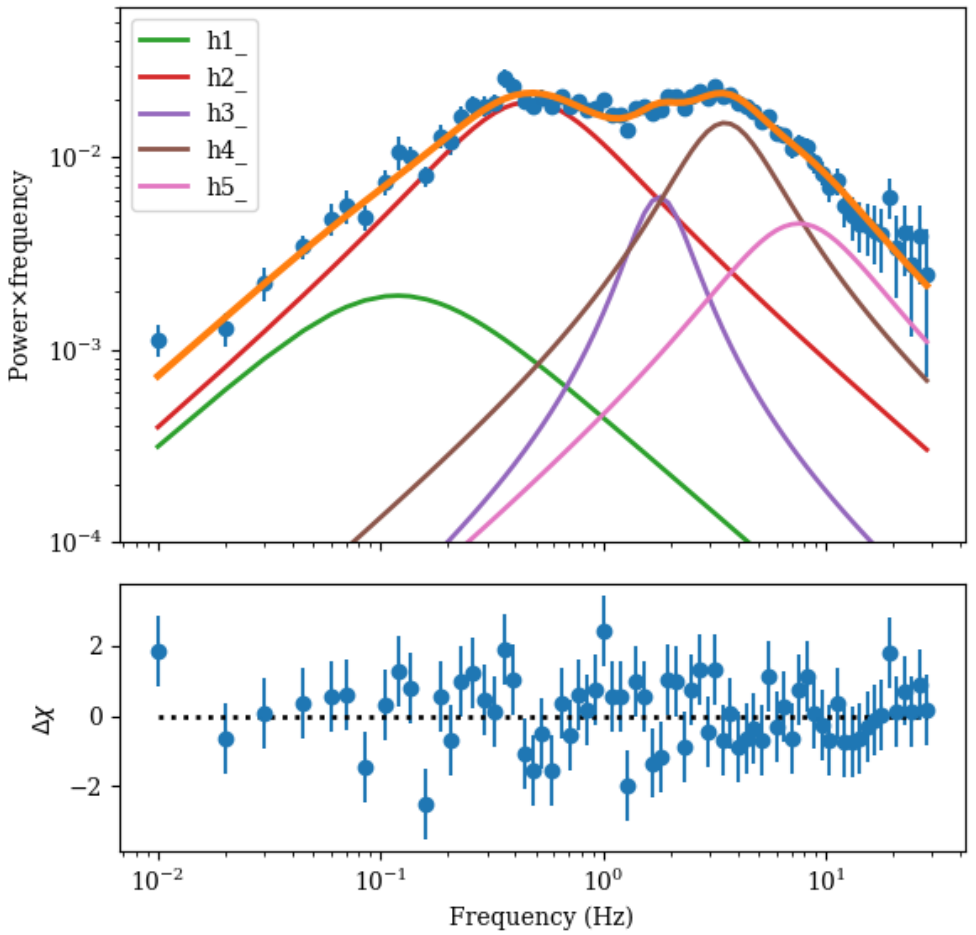
see Mendez et al. (2024)

Spectral-timing for stellar wind studies

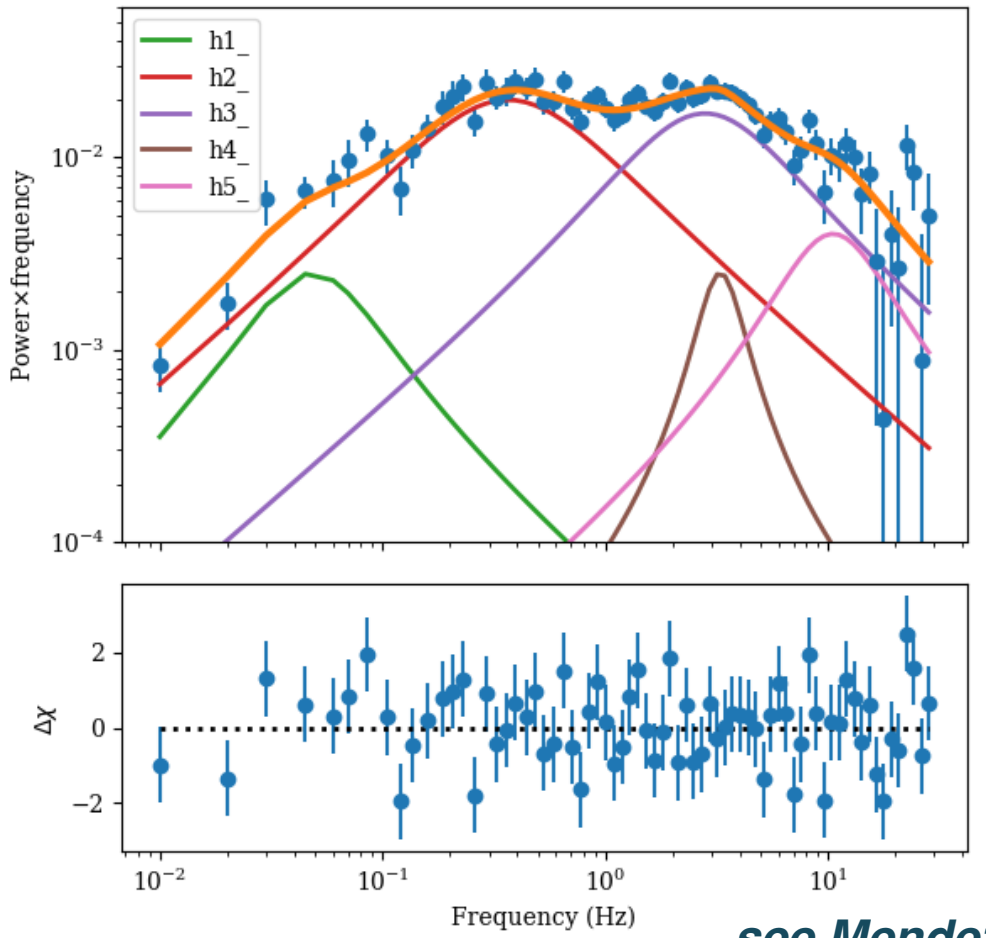
Nicer, ObsID: 0100320106

3 - 10 keV

No Wind Affected



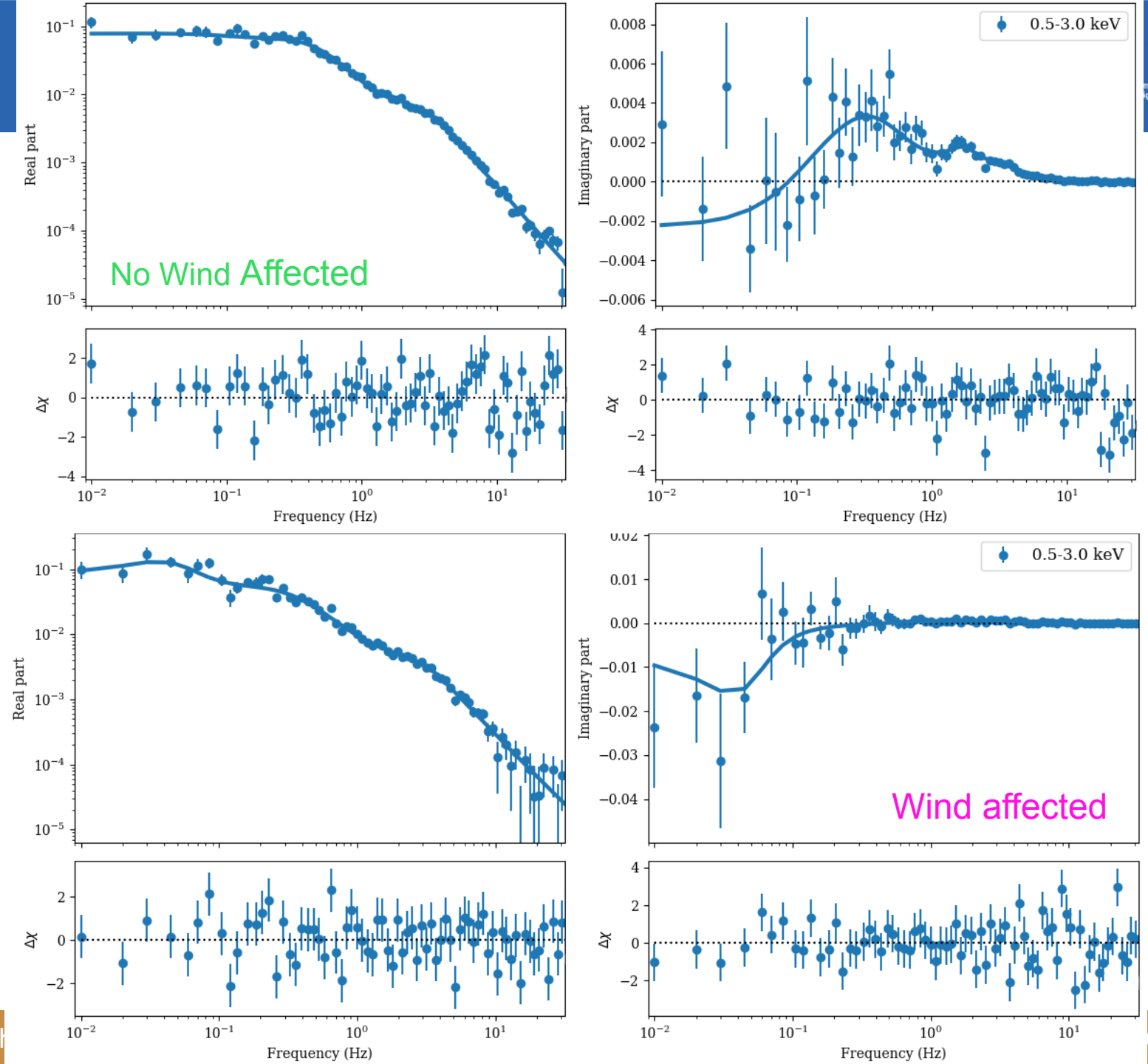
Wind affected



see Mendez et al. (2024)

3 - 10 keV
vs
0.5 - 3 keV

Nicer, ObsID: 0100320106

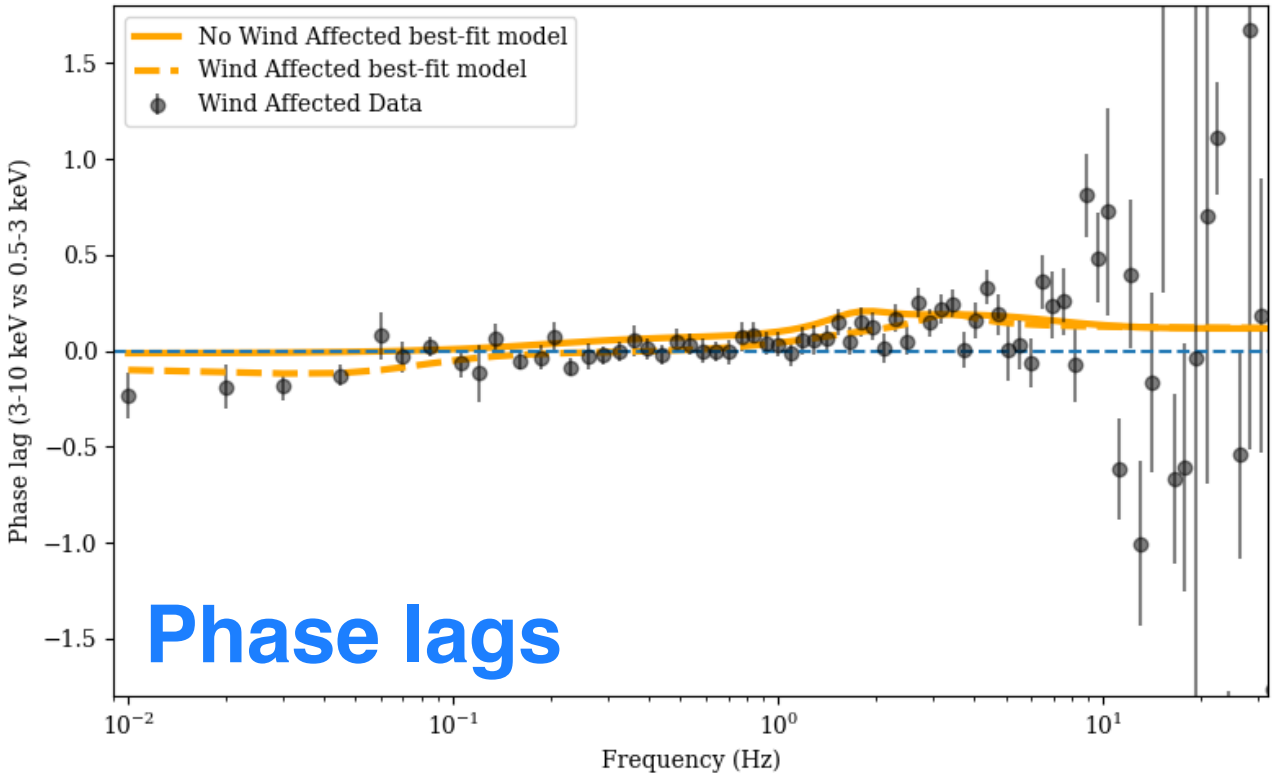


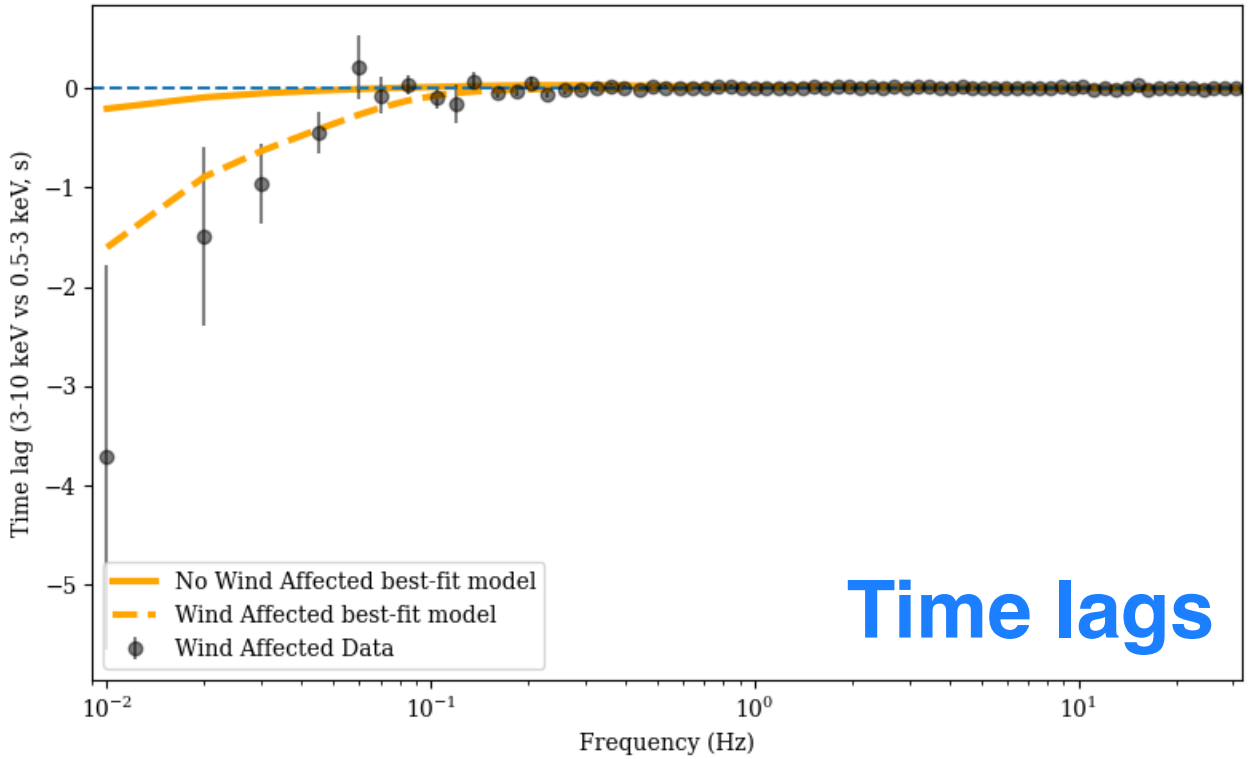
see Mendez et al. (2024)

Spectral-timing for stellar wind studies

Nicer, ObsID: 0100320106

3 - 10 keV vs 0.5 - 3 keV

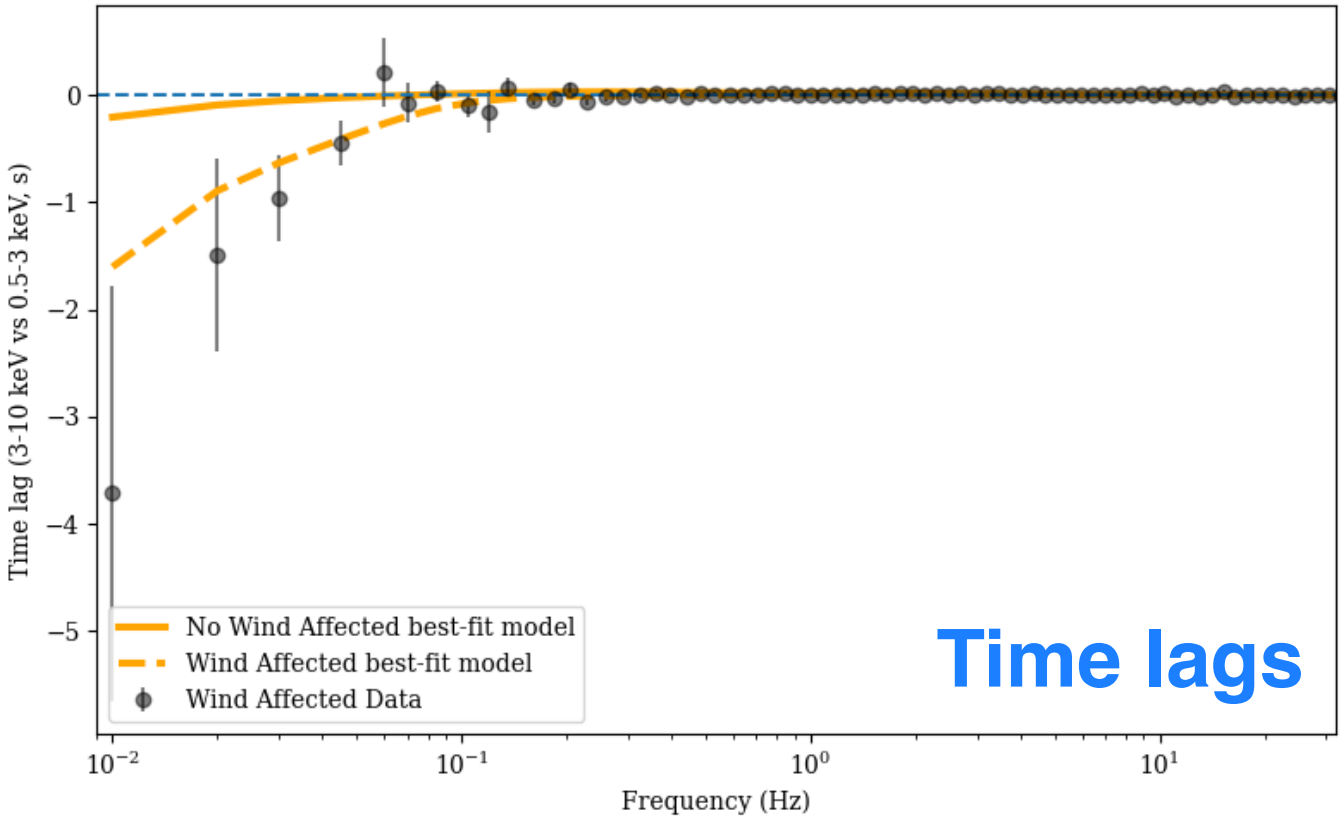




Spectral-timing for stellar wind studies

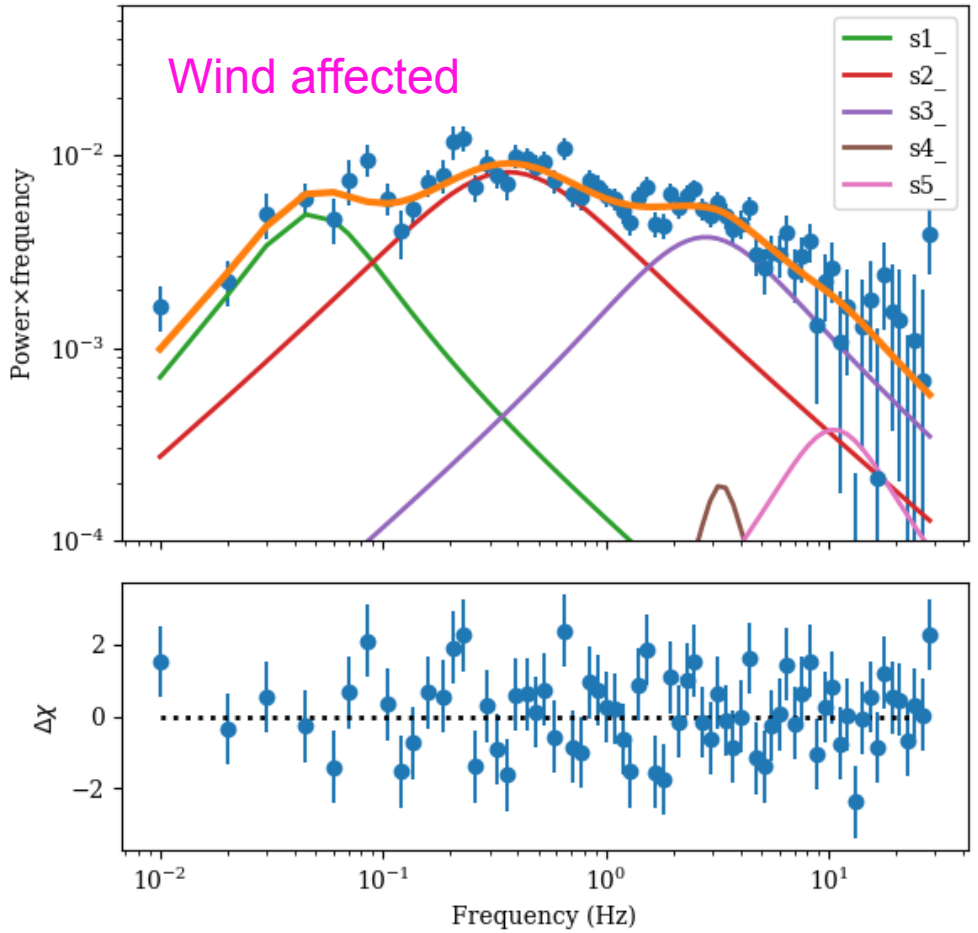
Nicer, ObsID: 0100320106

3 - 10 keV vs 0.5 - 3 keV



Time lags

0.5 - 3 keV



Docs and info

Information

How to install it

Tutorials

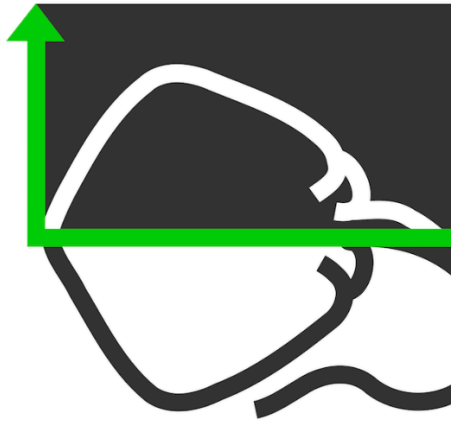
https://docs.stingray.science/en/stable/

Stingray:docs astropy Index M

stingray v2.3.2 »

Page Contents

Stingray: Next-Generation Spectral Tim



Usage	Release	Development
python >=3.8	release v2.3.2	CI Tests pass
docs latest	JOSS 10.21105/joss.07389	repo status Acti
License MIT	DOI 10.5281/zenodo.1490116	codecov 95%

Stingray is a Python library designed to perform times series analysis and rel

Installation

Messages to bring home…

Stingray: Next-Generation Spectral Timing

- Stingray is an **easy, free** to use, and **well-mantained** tool to perform spectral-timing
- **Great performance** (*Lai et al. 2026*)
- **Complicated analyses** in a **simple way**:
  - Check your results
  - Get acquainted with such methods
- Available **tutorials** (basic to advanced) and **fast communication** with developers
- Not an X-rays lover? Stingray works also in other energy bands

Stingray: your friendly Python library for spectral-timing


Thank you!