



Finanziato
dall'Unione europea
NextGenerationEU



Ministero
dell'Università
e della Ricerca



Italiadomani
PIANO NAZIONALE
DI RIPRESA E RESILIENZA

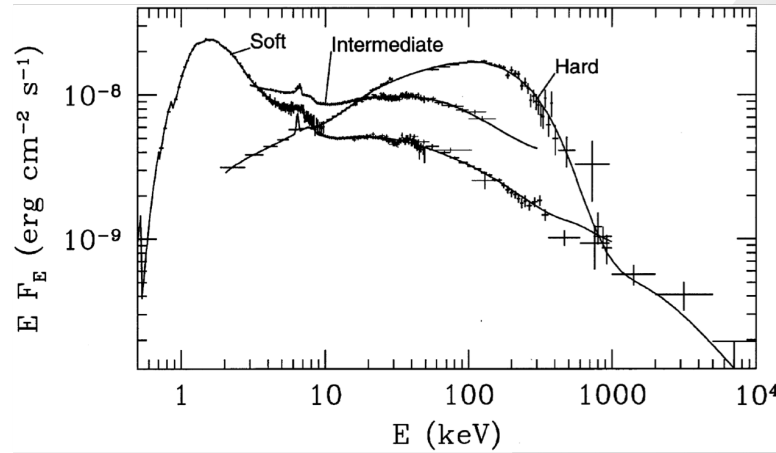


Timing analysis of X-ray binaries with Stingray

*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

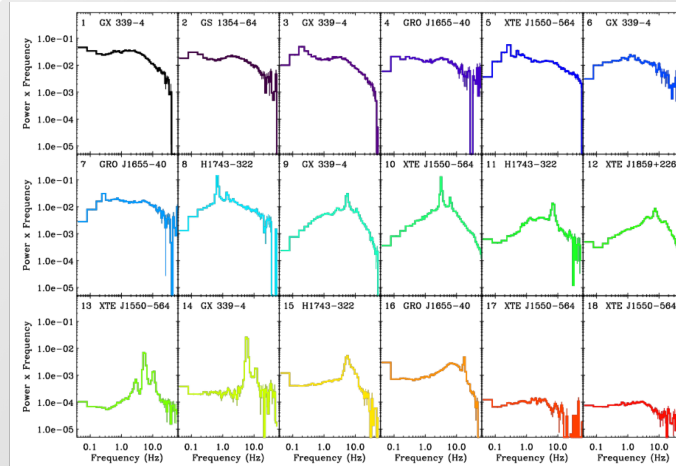
Scientific Rationale



Zdziarski (2000)

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

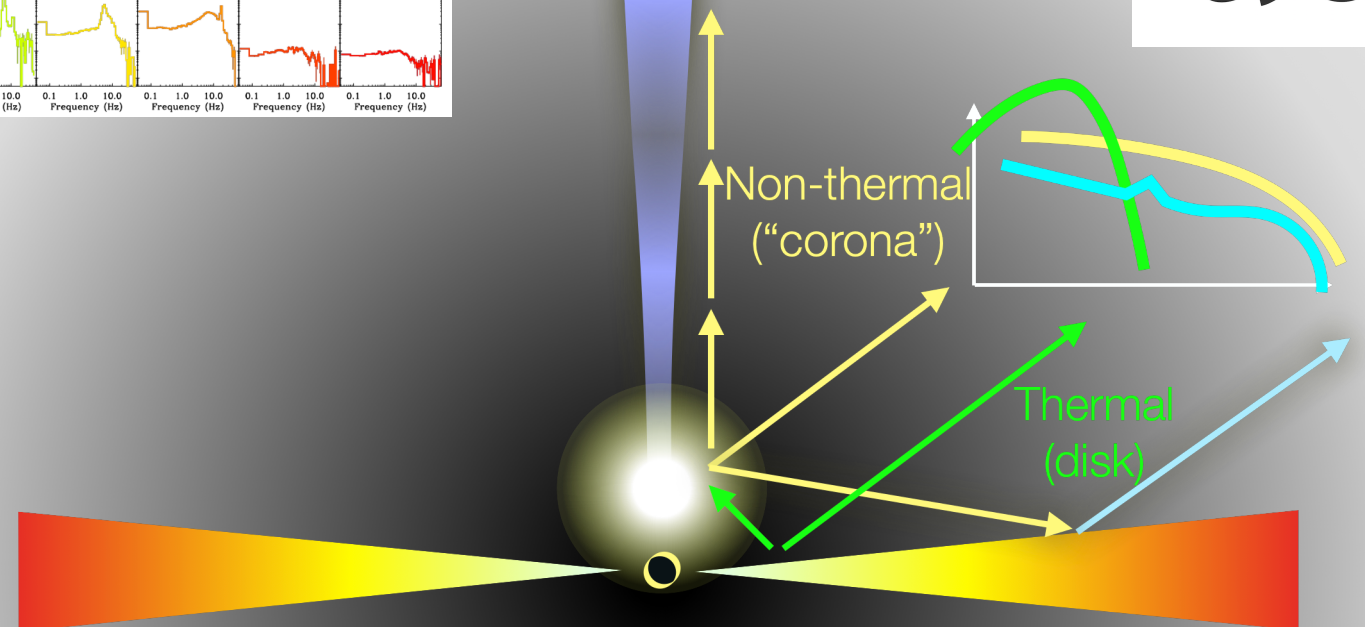
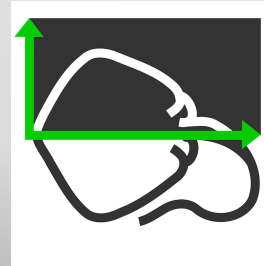
Heil et al. (2015)



Huppenkothen et al. (2019)

Bachetti et al. (2024)

Lai et al. (2026)



History

Lorentz center

The X-ray Spectral-Timing Revolution

Workshop: 1 - 5 February 2016 Leiden, the Netherlands

Scientific Organizers

- Ed Cackett, Wayne State U
- Chris Done, Durham U
- Andy Fabian, U Cambridge
- Barbara De Marco, MPE Garching
- Phil Uttley, U Amsterdam

Topics

- X-ray Reverberation Mapping
- QPO and Pulsation Spectroscopy
- Accretion Flow Variability
- Absorption Variability
- New Analysis Methods
- Modelling Spectral-Timing Signals
- Timing Polarimetry

The Lorentz Center is an international center in the power of its aim to organize workshops for scientists in an atmosphere that fosters collaborative work, discussions and interactions. For registration see www.lorentzcenter.nl

We can make images of the regions closest to neutron stars and black holes by measuring the time delays between variations of different components of x-ray emission from these systems. Peter George, SuperNova Systems, IL

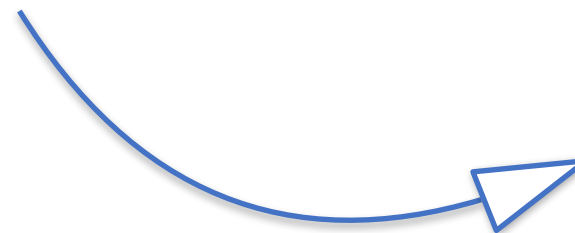
www.lorentzcenter.nl

Logos: Universiteit Leiden, FOM, STW, NWO, Lorentz center

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

Lorentz center

The X-ray Spectral-Timing Revolution

Workshop: 1 - 5 February 2016 Leiden, the Netherlands

Scientific Organizers

- Ed Cackett, Wayne State U
- Chris Done, Durham U
- Andy Fabian, U Cambridge
- Barbara De Marco, MPE Garching
- Phil Uttley, U Amsterdam



Topics

- X-ray Reverberation Mapping
- QPO and Pulsation Spectroscopy
- Accretion Flow Variability
- Absorption Variability
- New Analysis Methods
- Modelling Spectral-Timing Signals
- Timing Polarimetry

The Lorentz Center is an international center in the Netherlands that aims to organize workshops for scientists in an atmosphere that fosters collaborative work, discussions and interactions. For registration see www.lorentzcenter.nl

We can make images of the regions closest to neutron stars and black holes by measuring the time delays between variations of different components of x-ray emission from these systems. *Photo: George Stathopoulos/Science Hub*

www.lorentzcenter.nl

Logos: Universiteit Leiden, FOM, STW, NWO, Lorentz center

Spectral analysis	Timing analysis (+ lags)	Spectral-timing
- Xspec - Sherpa - ISIS (...)	(X)RONOS, POWSPEC - STIMFIT, Isiscript, etc.	

- Daniela Compton, Abigail Steven (*Stingray's original*)
- Matteo Balemi, Howard
- Simon



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

- Software:

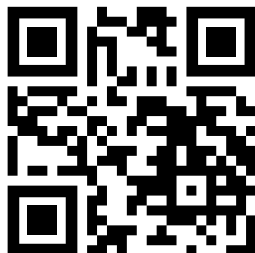
- Python **open-source** software in a Github-based workflow

- Developers:

- Astronomers
- Google Summer of Code students

- Community outreach:

- Public **Slack channel**



- Talks
- Hackatons/Tutorials

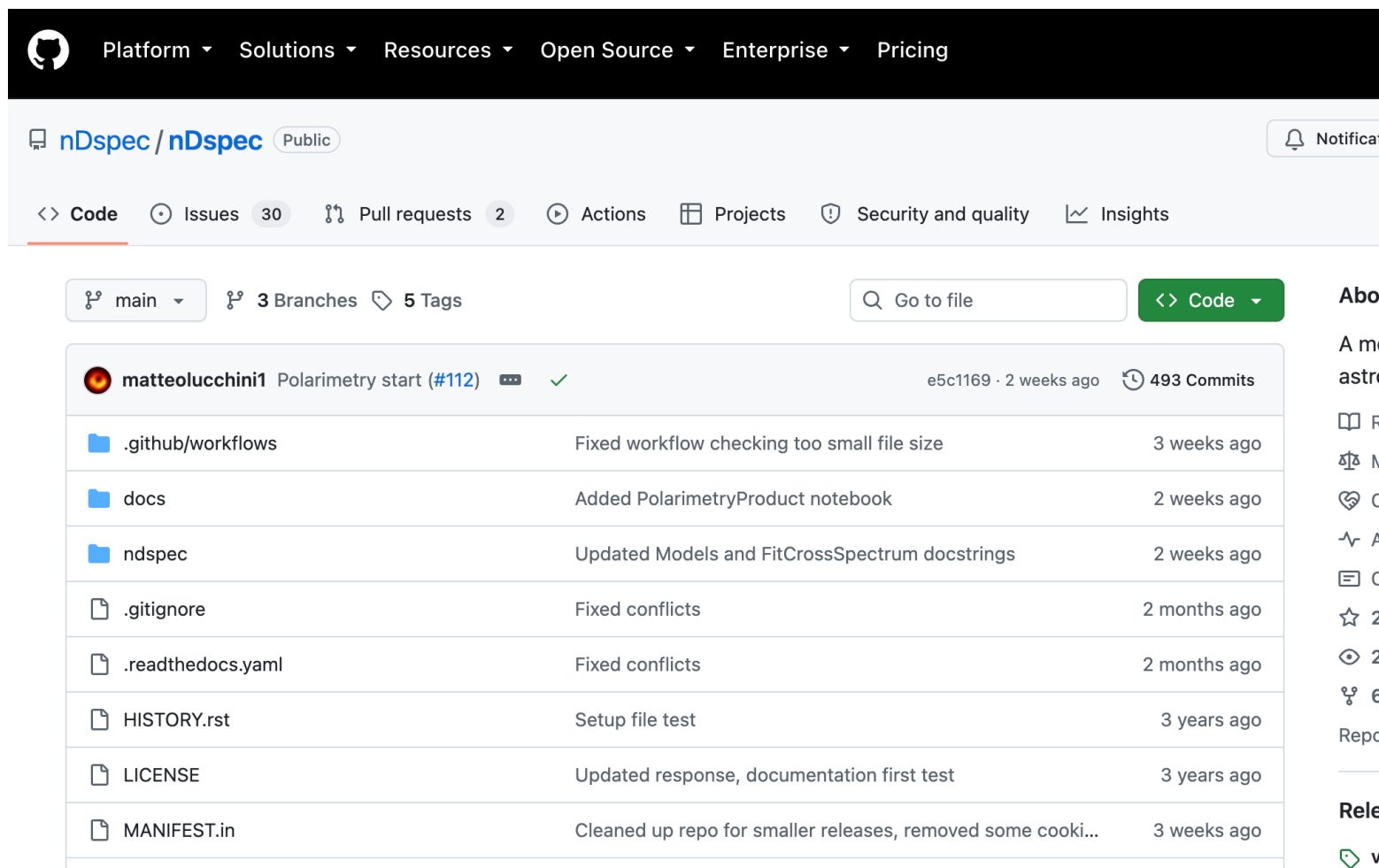
The screenshot shows the GitHub repository for 'Stingray'. The repository is described as 'Next-generation spectral-timing software for astronomy applications and beyond.' It has 38 followers and a website link 'https://stingray.science'. The page is divided into sections: Overview, Repositories (12), Projects, Packages, and People (6). Under the 'Pinned' section, there are four items:

- stingray** (Public): Anything can happen in the next half hour (including spectral timing made easy)!
 - Python
 - 221 stars
 - 188 forks
- HENDRICS** (Public): Shell scripts for spectral-timing analysis of X-ray astronomical data.
 - Jupyter Notebook
 - 26 stars
 - 18 forks
- dave** (Public): A GUI for spectral-timing analysis of X-ray astronomical data.
 - JavaScript
 - 16 stars
 - 15 forks
- notebooks** (Public): Tutorial notebooks for Stingray
 - Jupyter Notebook
 - 35 stars
 - 45 forks

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



The screenshot shows the GitHub repository for nDspec. The repository is public and has 30 issues, 2 pull requests, and 493 commits. The commit history is displayed, showing recent updates to workflows, documentation, and models.

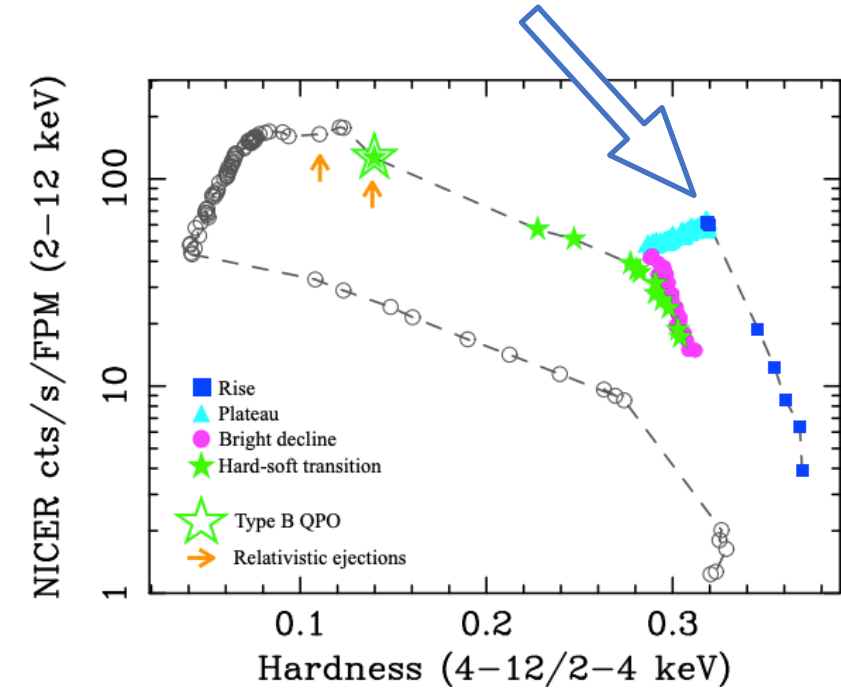
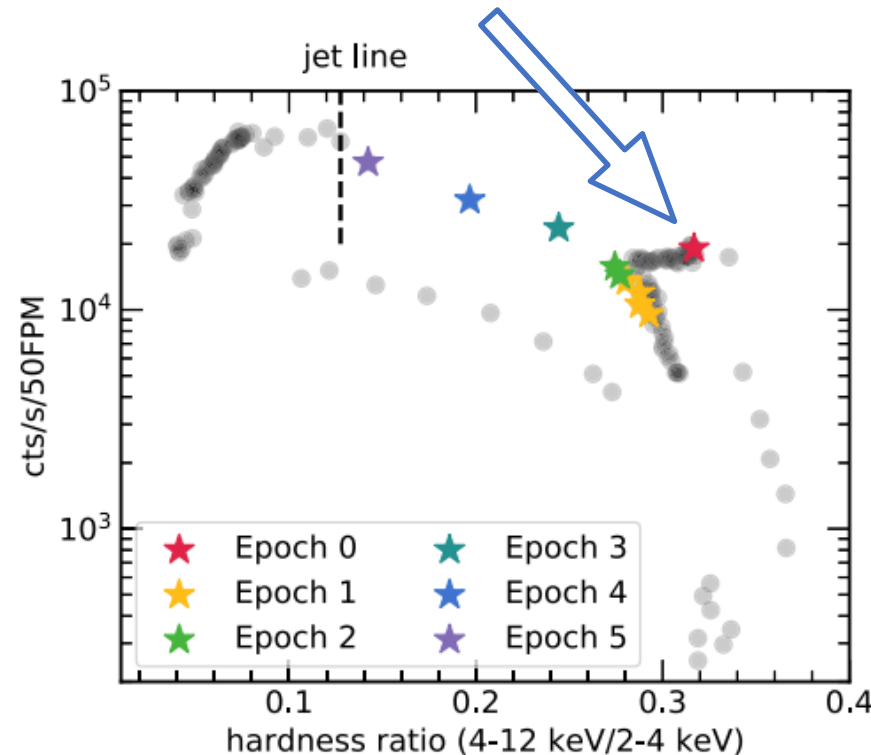
Commit	Message	Time
matteolucchini1	Polarimetry start (#112)	e5c1169 · 2 weeks ago
	Fixed workflow checking too small file size	3 weeks ago
	Added PolarimetryProduct notebook	2 weeks ago
	Updated Models and FitCrossSpectrum docstrings	2 weeks ago
	Fixed conflicts	2 months ago
	Fixed conflicts	2 months ago
	Setup file test	3 years ago
	Updated response, documentation first test	3 years ago
	Cleaned up repo for smaller releases, removed some cooki...	3 weeks ago

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

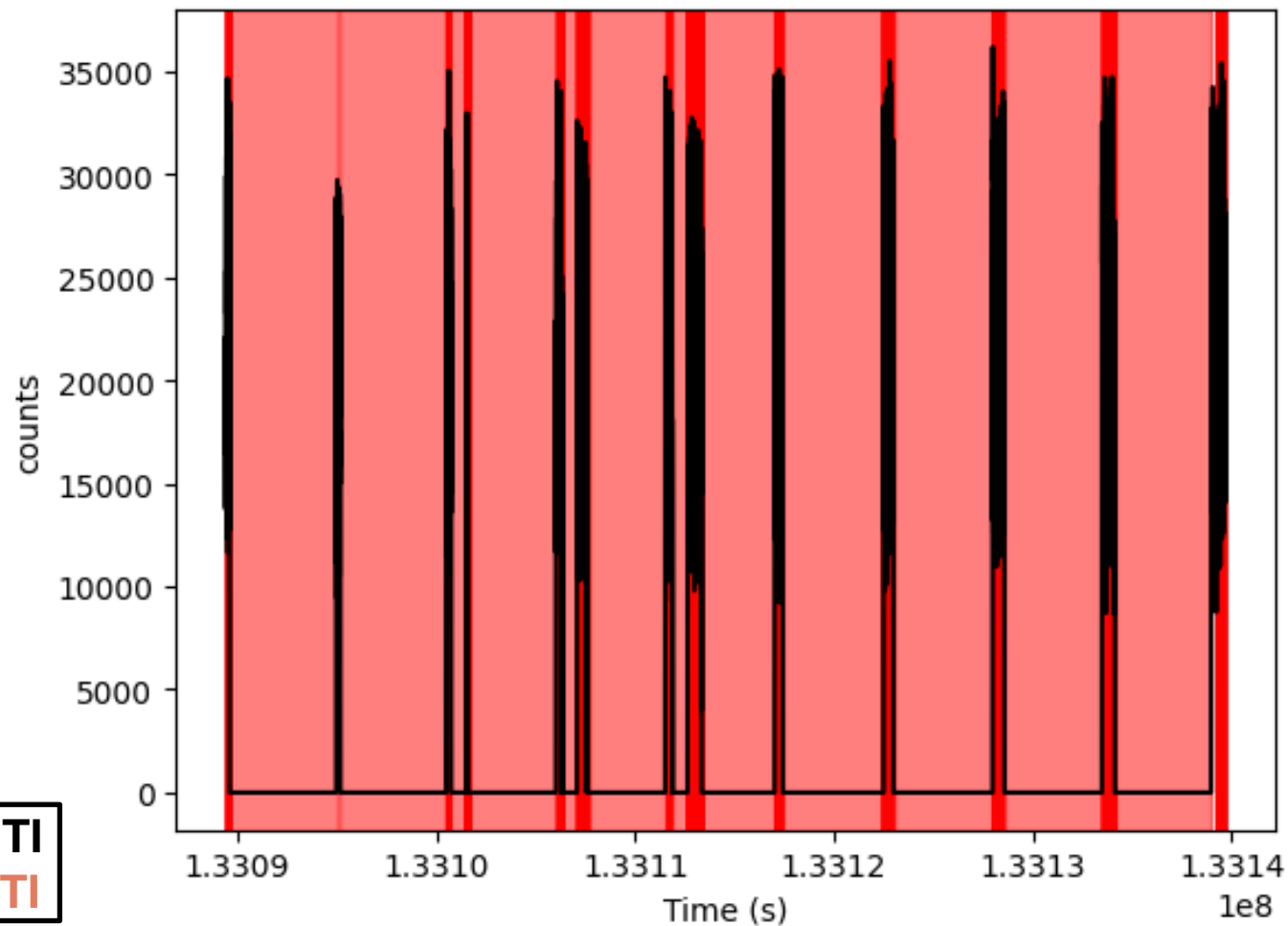
# input event file
fname = "ni1200120106_0mpu7_cl_bary.evt"

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

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

lc_raw.plot()
```

GTI
BTI



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
```

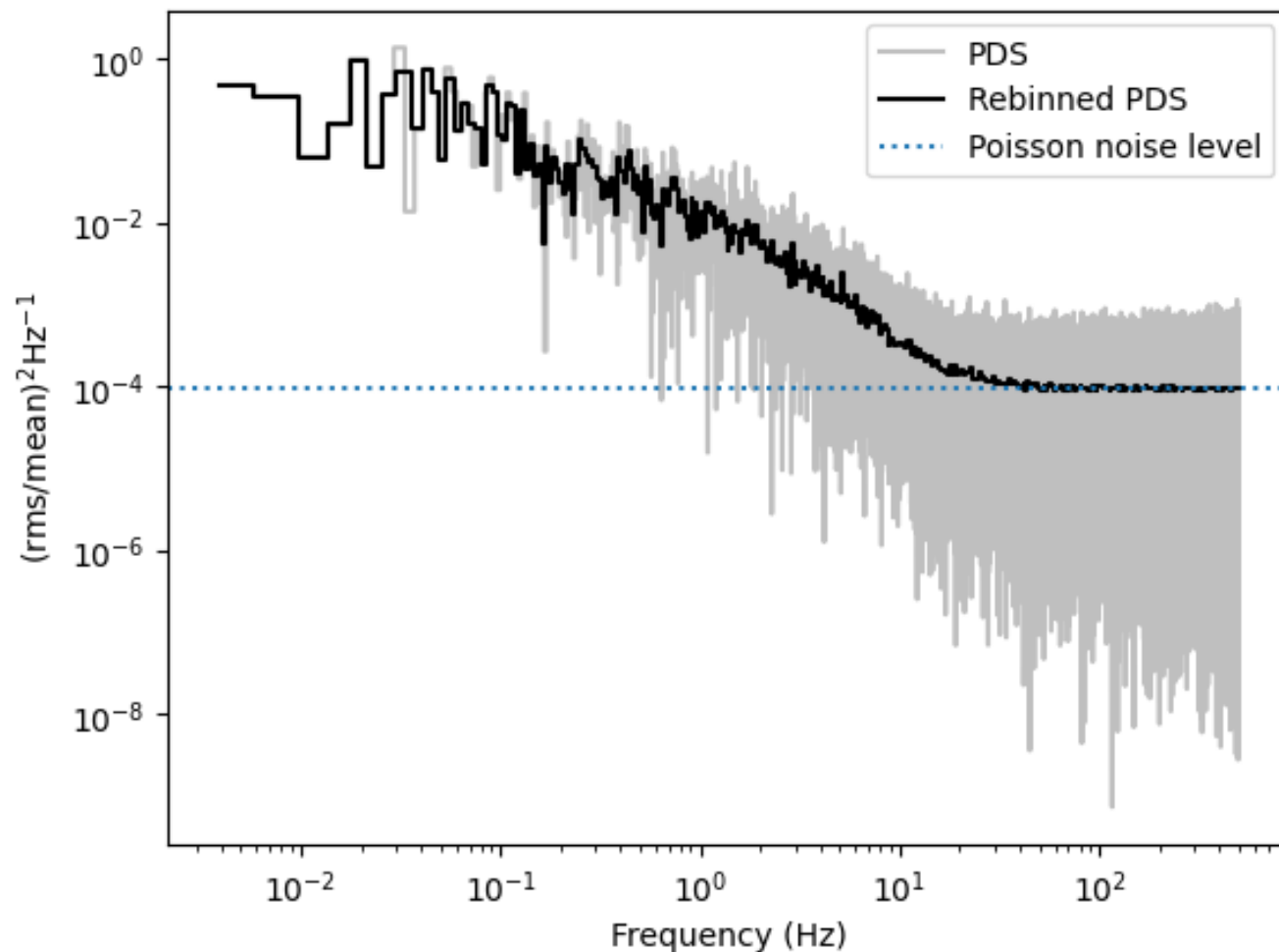
```
pds_reb = pds.rebin_log(0.02)
```

```
# 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
```

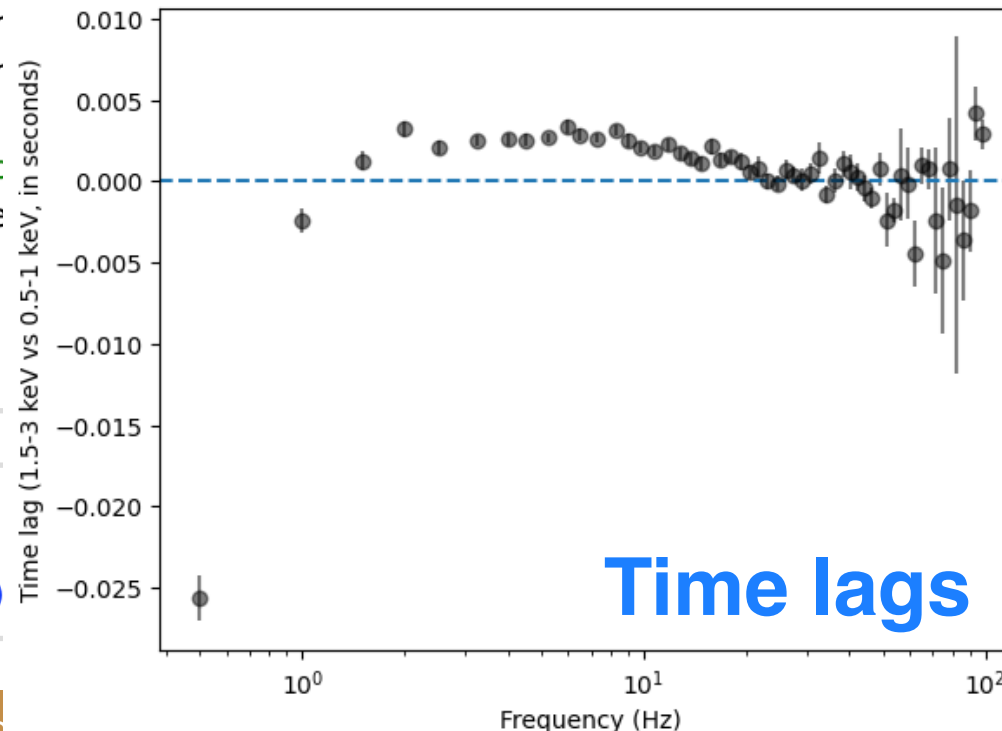
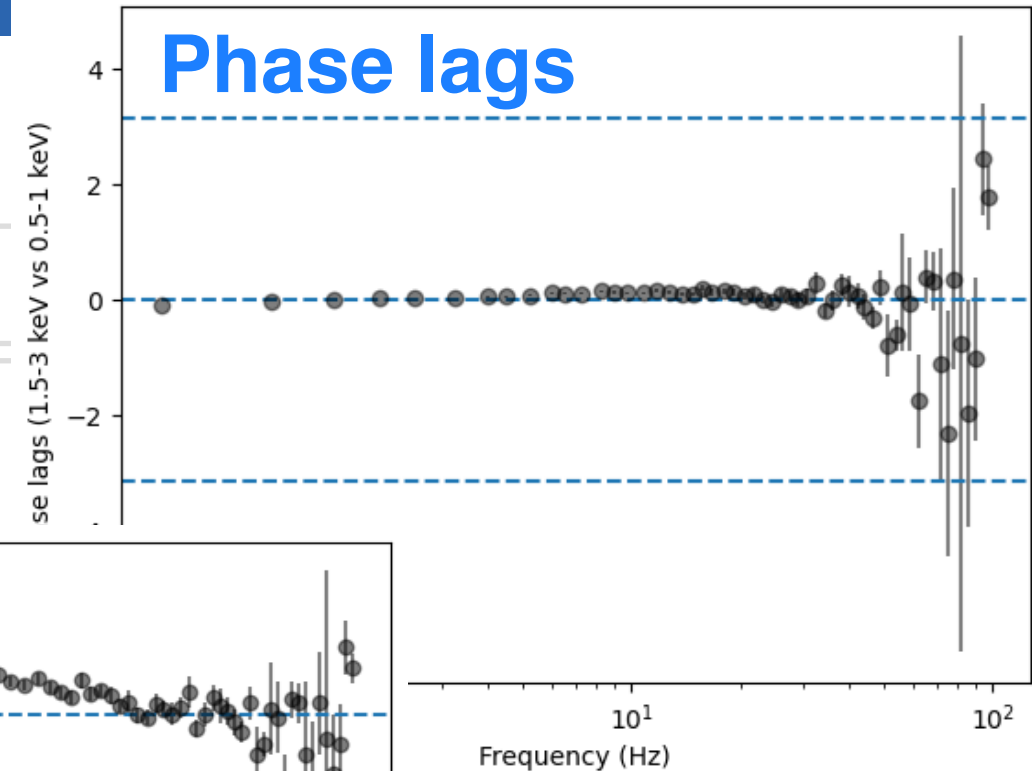
```
# Rebin
```

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

```
# Spectral-timing made easy
```

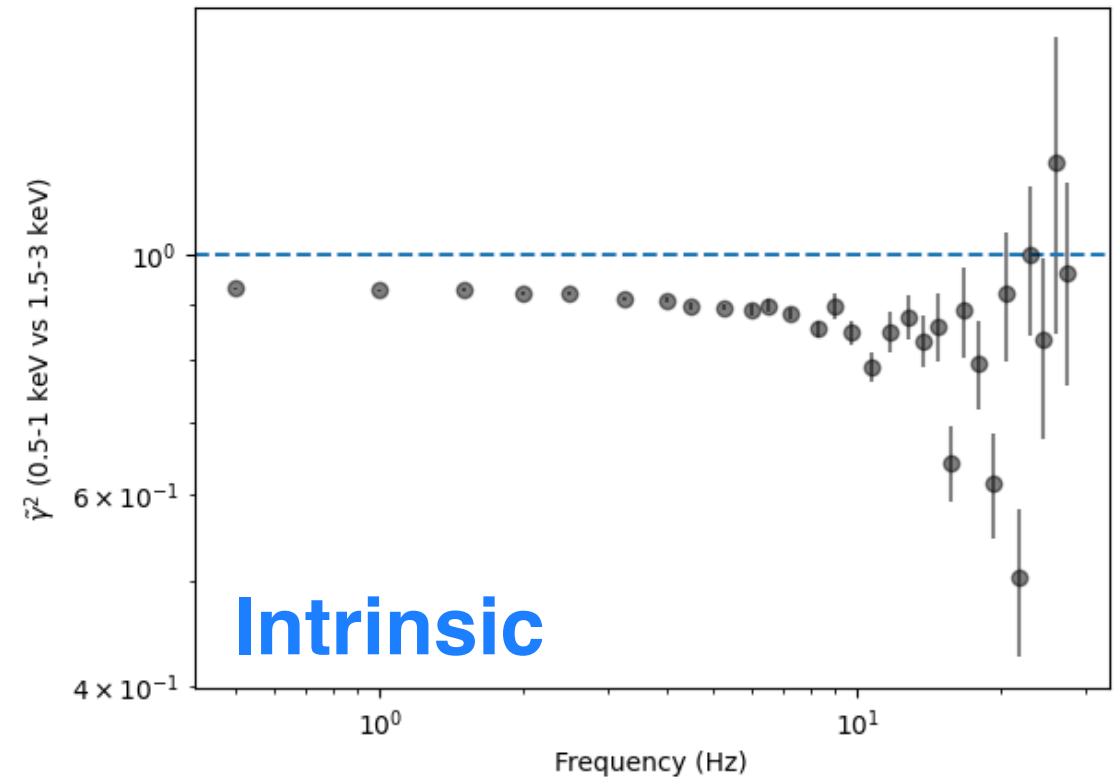
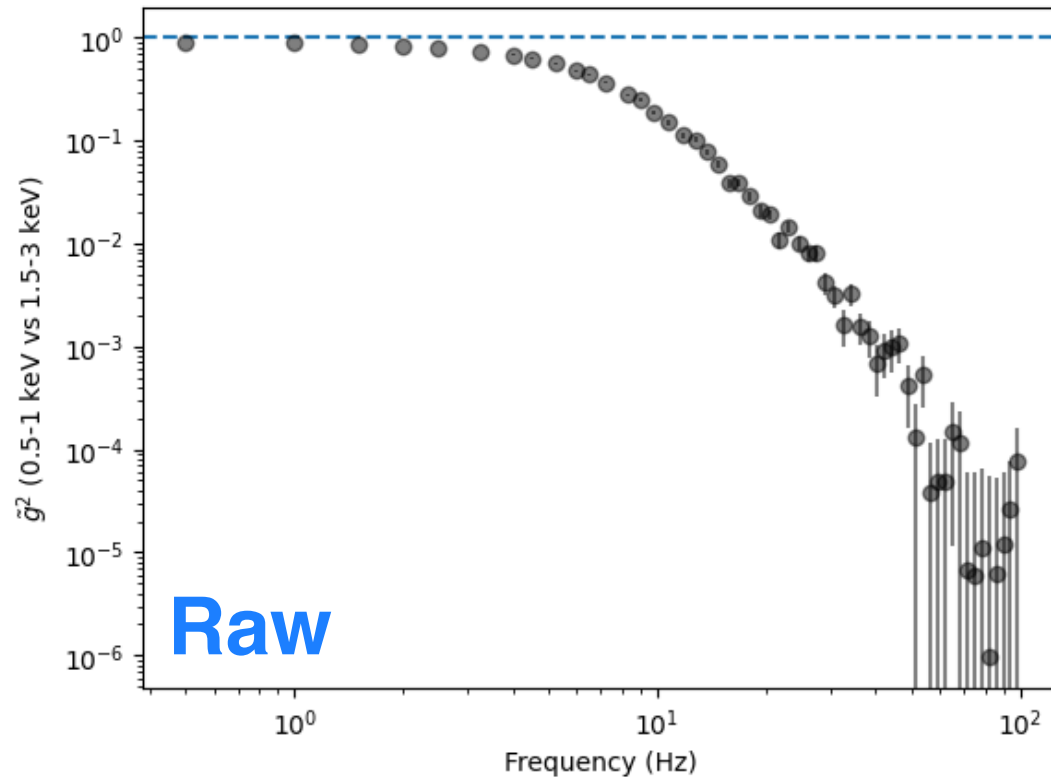
```
lag, lag_e = cs_reb.time_lag()
```

```
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()
```





ELSEVIER

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, ID, *}, 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)



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,
)
```

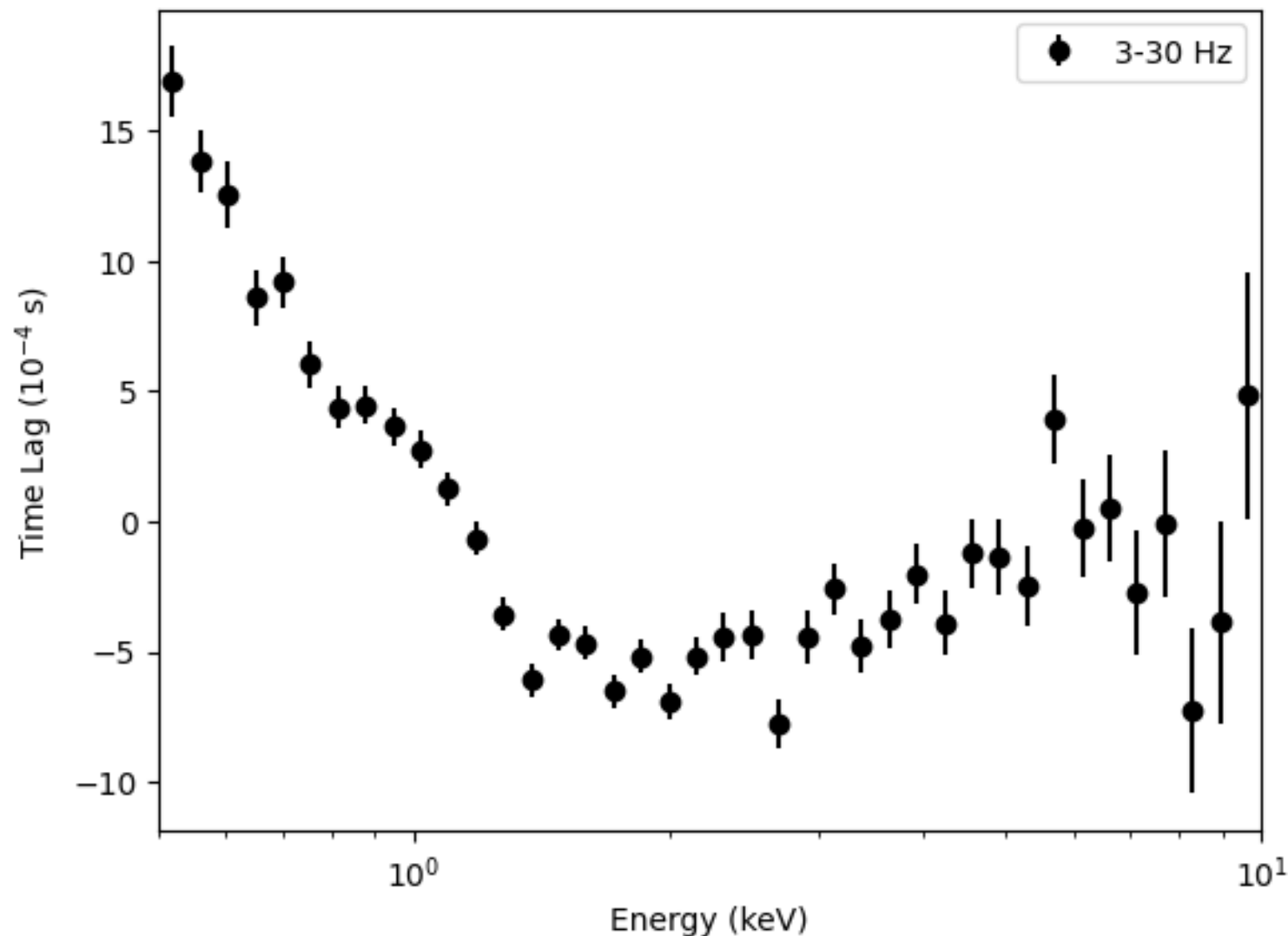


Time lags vs Energy

```
from stingray.varenergyspectrum import 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,
)
```





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",  
)
```

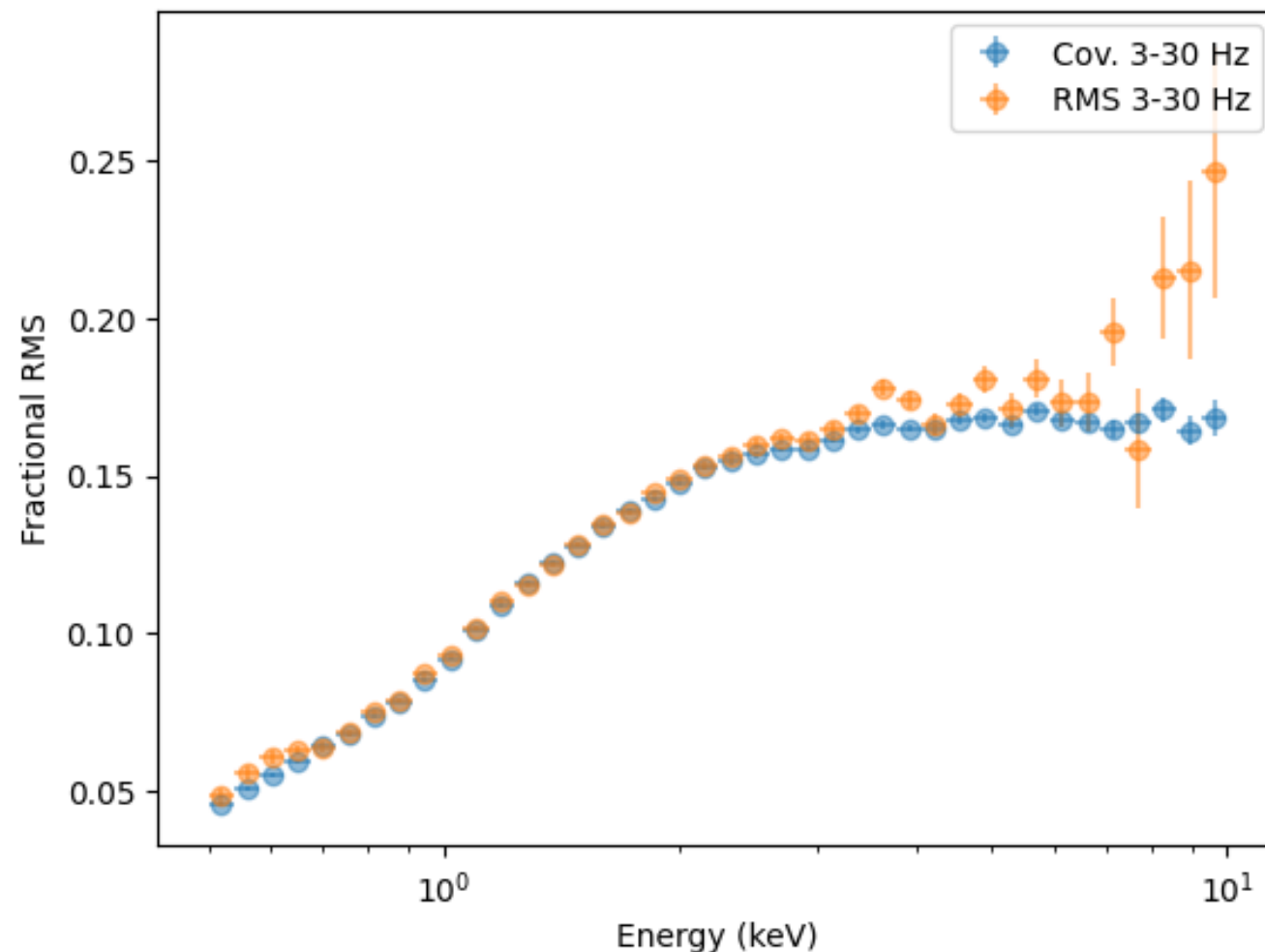
```
rmsspec_3_30 = RmsSpectrum(  
    events,  
    freq_interval=[3, 30],  
    segment_size=segment_size,  
    bin_time=bin_time,  
    energy_spec=energy_spec,  
    norm="frac",  
)
```



Covariance and RMS spectrum

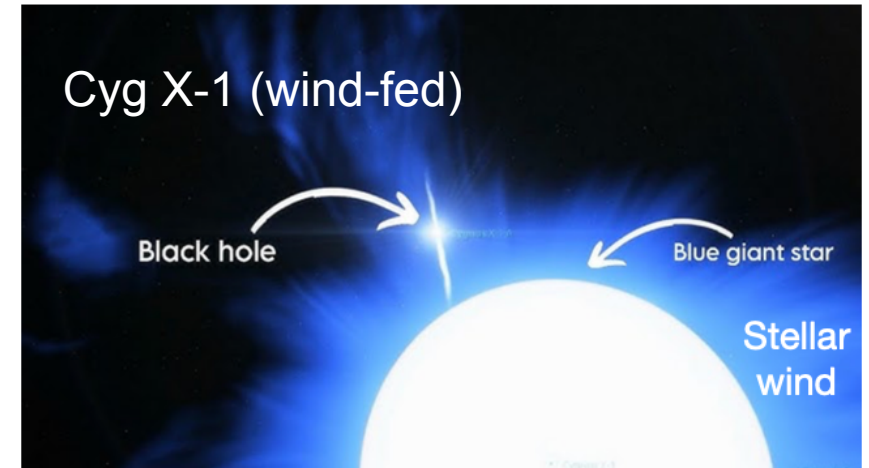
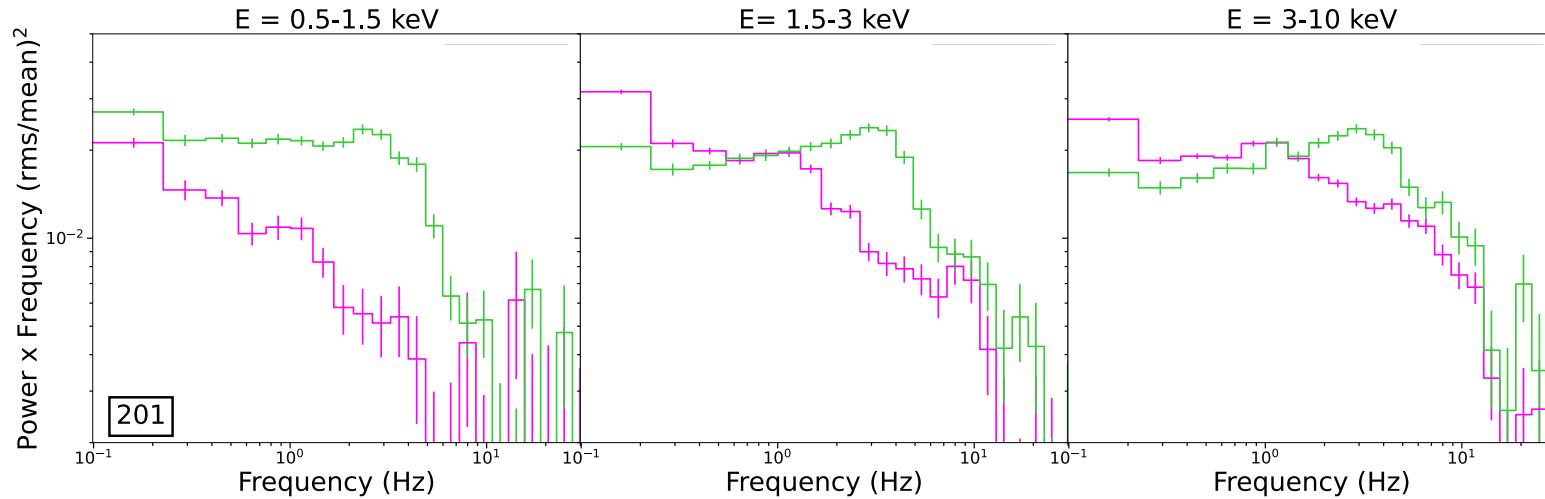
```
from stingray.varenergyspectrum import CovarianceSpec  
  
covspec_3_30 = CovarianceSpectrum(  
    events,  
    freq_interval=[3, 30],  
    segment_size=segment_size,  
    bin_time=bin_time,  
    energy_spec=energy_spec,  
    norm="frac",  
)
```

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

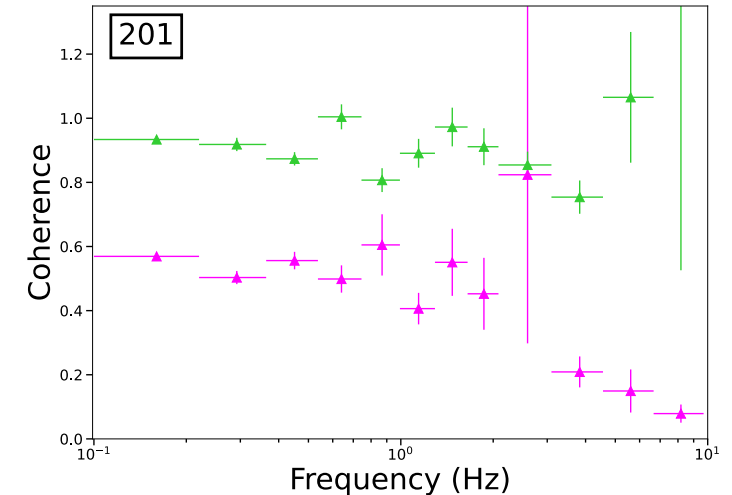
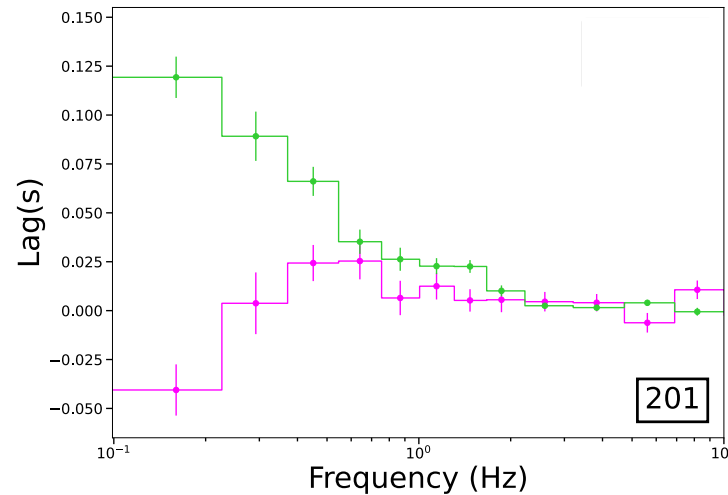


Spectral-timing for stellar wind studies

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



Wind affected
No Wind Affected

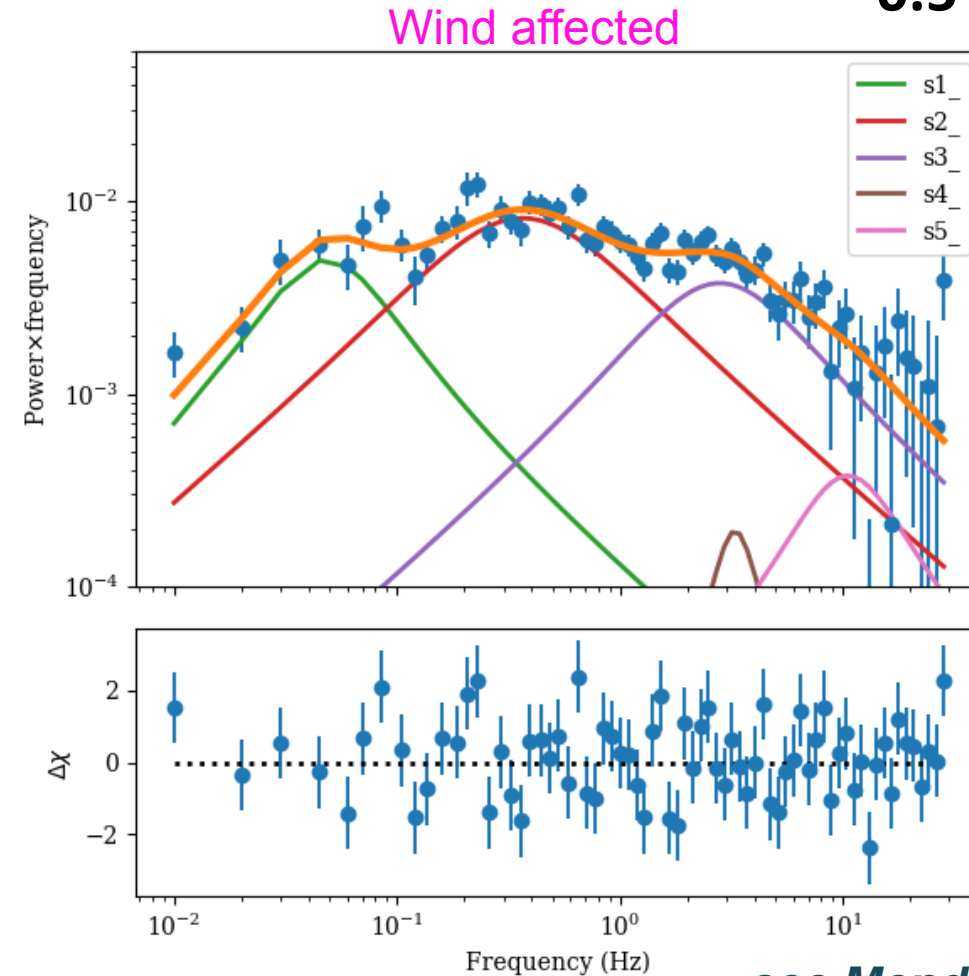
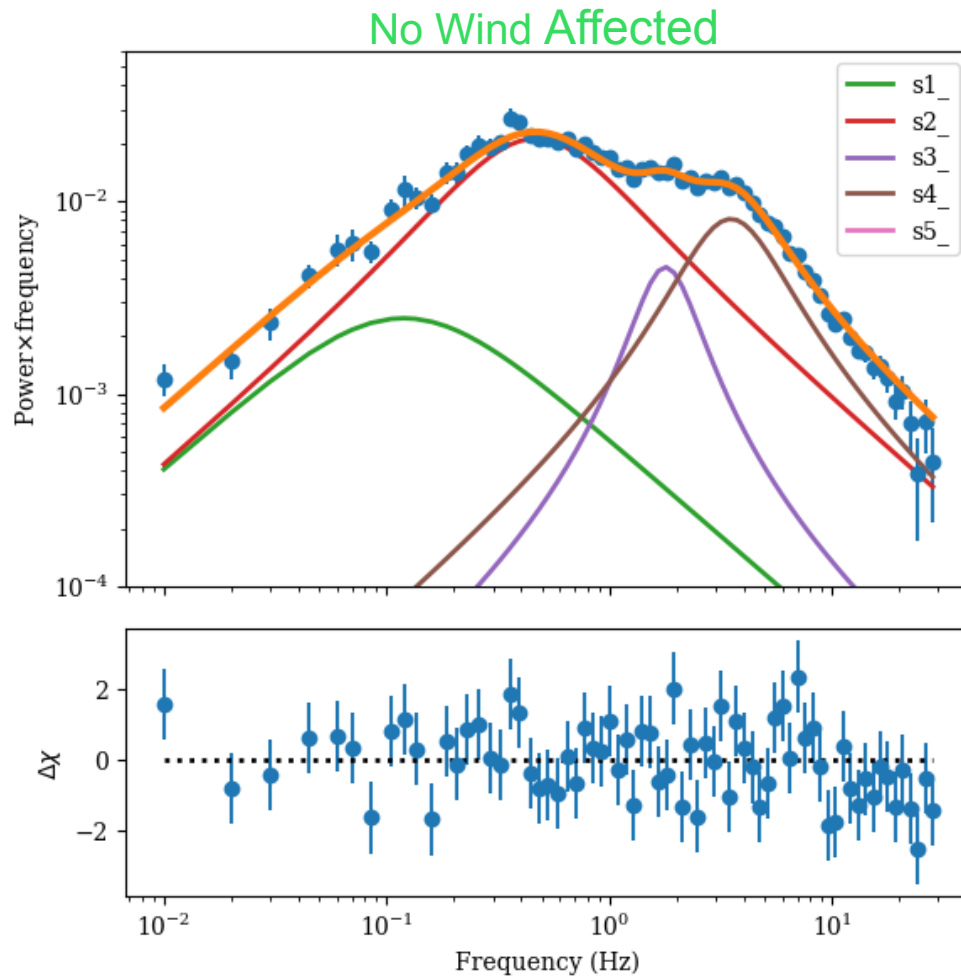


XMM-Newton, ObsID: 0745250201

Spectral-timing for stellar wind studies

Nicer, ObsID: 0100320106

0.5 - 3 keV

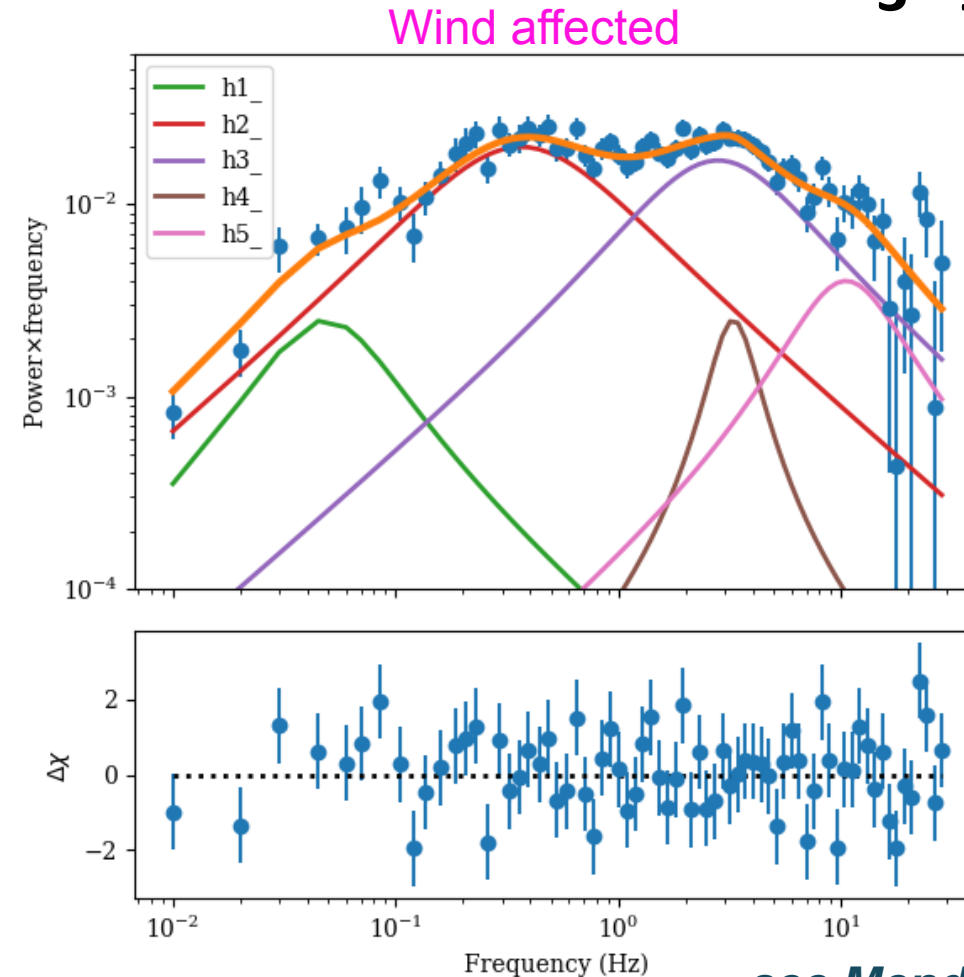
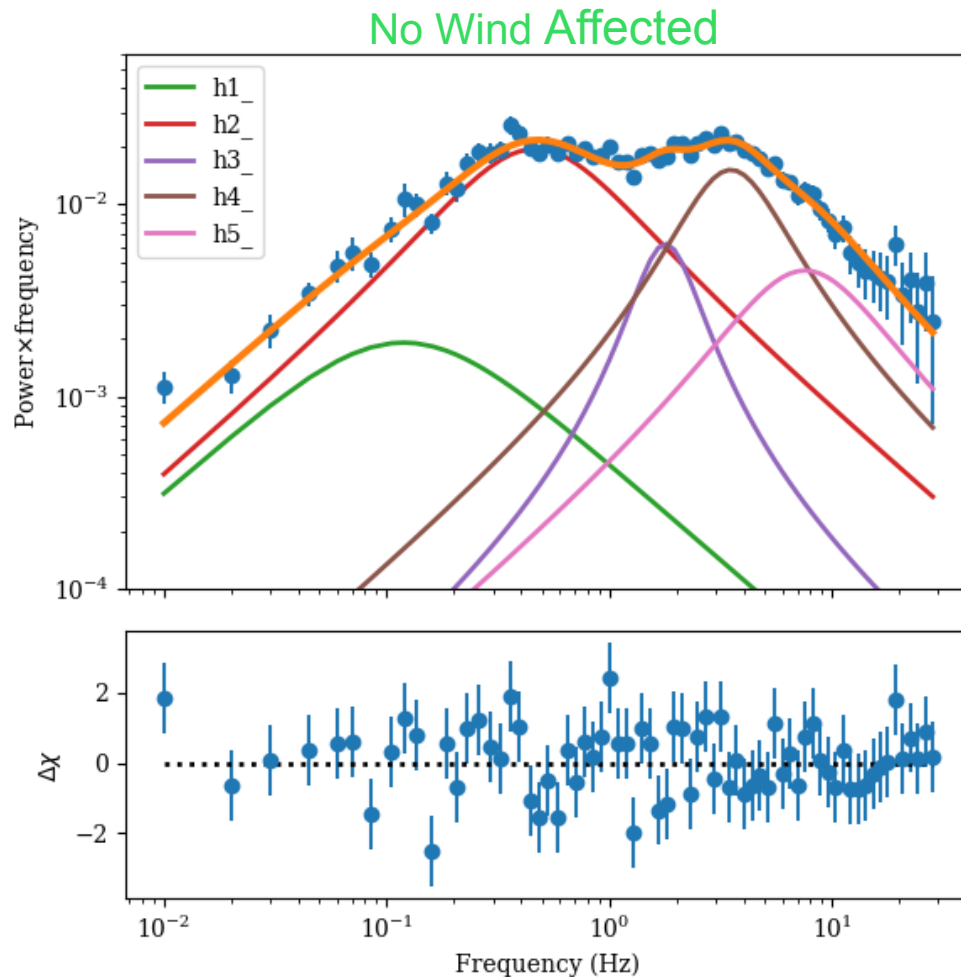


see Mendez et al. (2024)

Spectral-timing for stellar wind studies

Nicer, ObsID: 0100320106

3 - 10 keV



see Mendez et al. (2024)



Finanziato
dall'Unione europea
NextGenerationEU

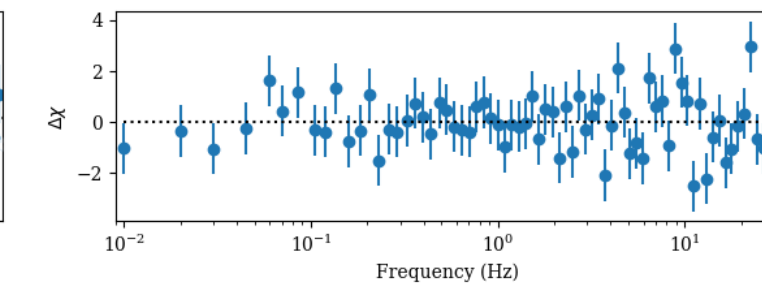
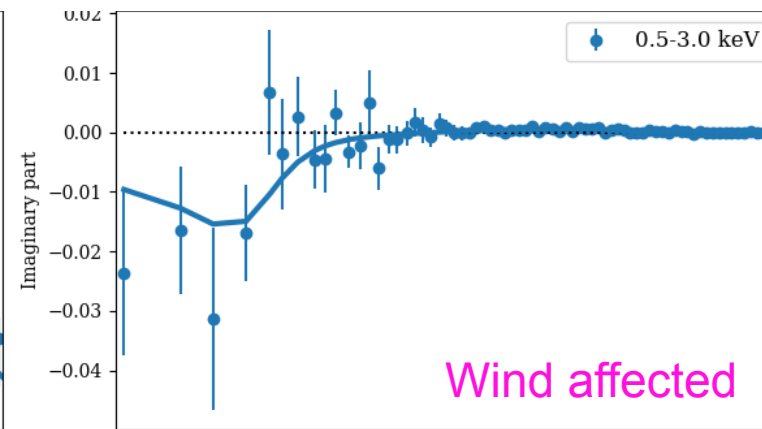
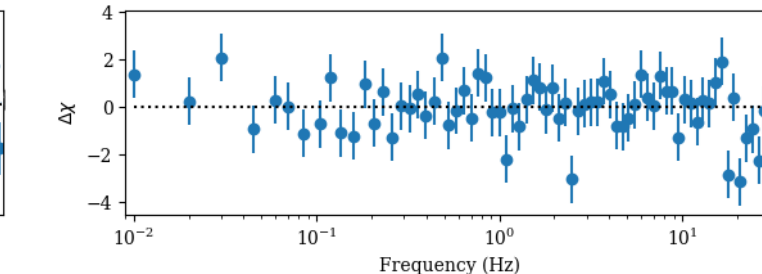
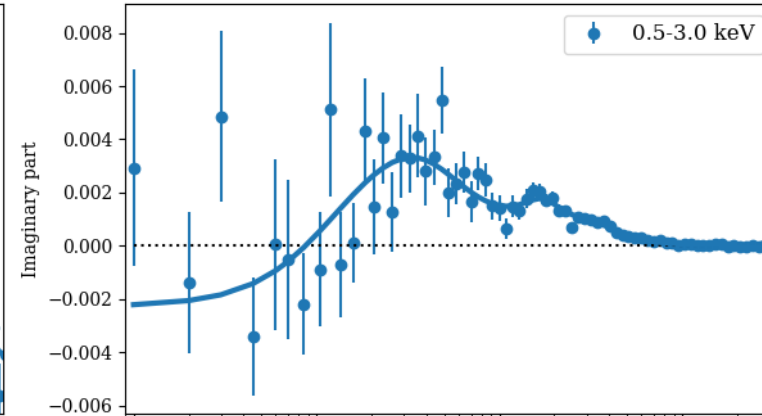
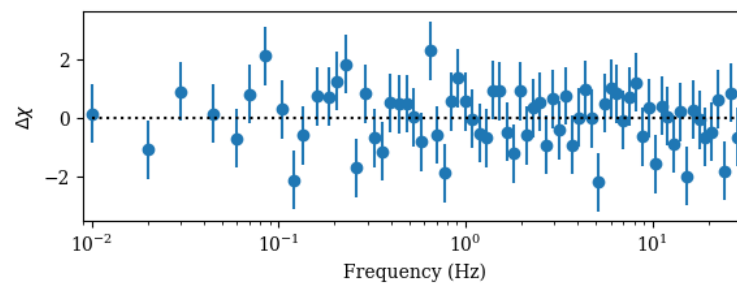
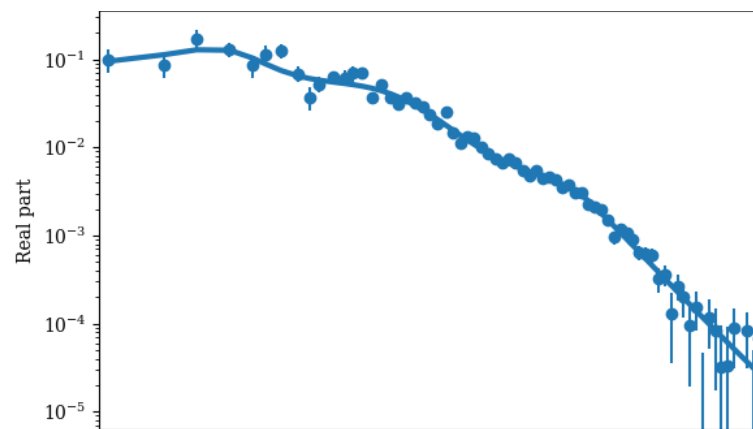
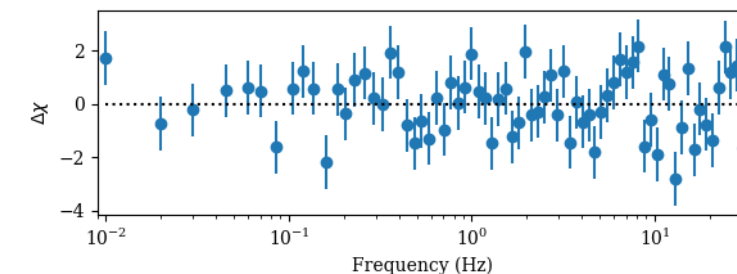
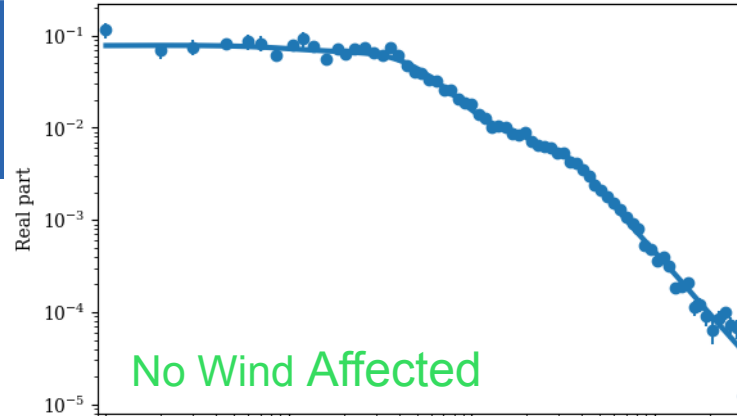
CSC

Centro Nazionale di Ricerca in HPC,
Big Data and Quantum Computing

**3 - 10 keV
vs
0.5 - 3 keV**

Nicer, ObsID: 0100320106

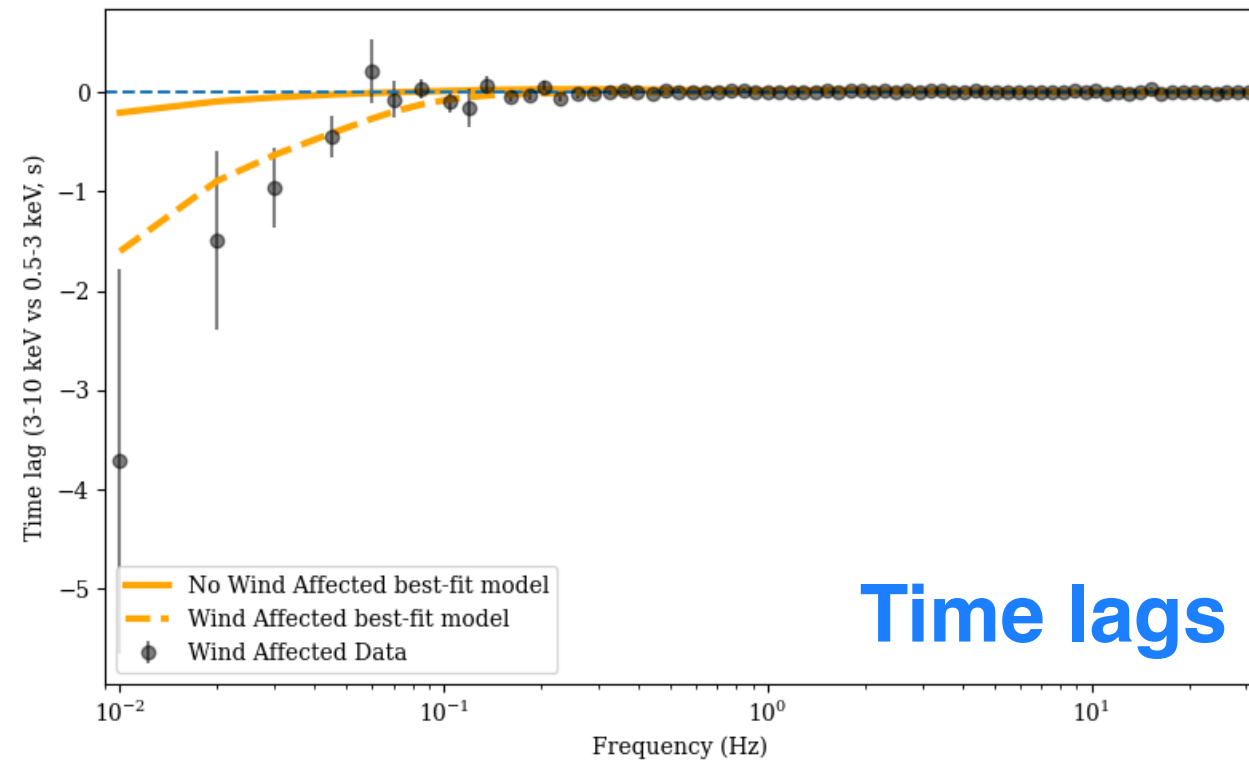
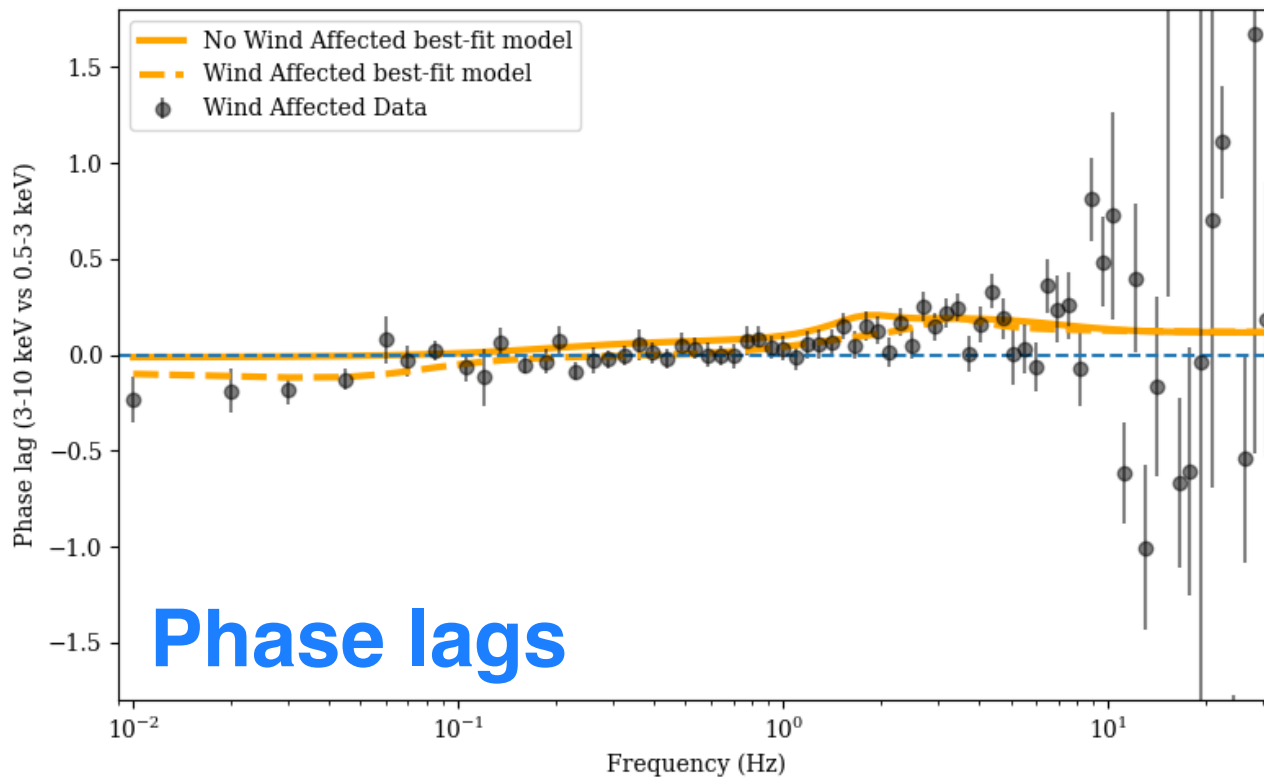
see Mendez et al. (2024)



Spectral-timing for stellar wind studies

Nicer, ObsID: 0100320106

3 - 10 keV vs 0.5 - 3 keV

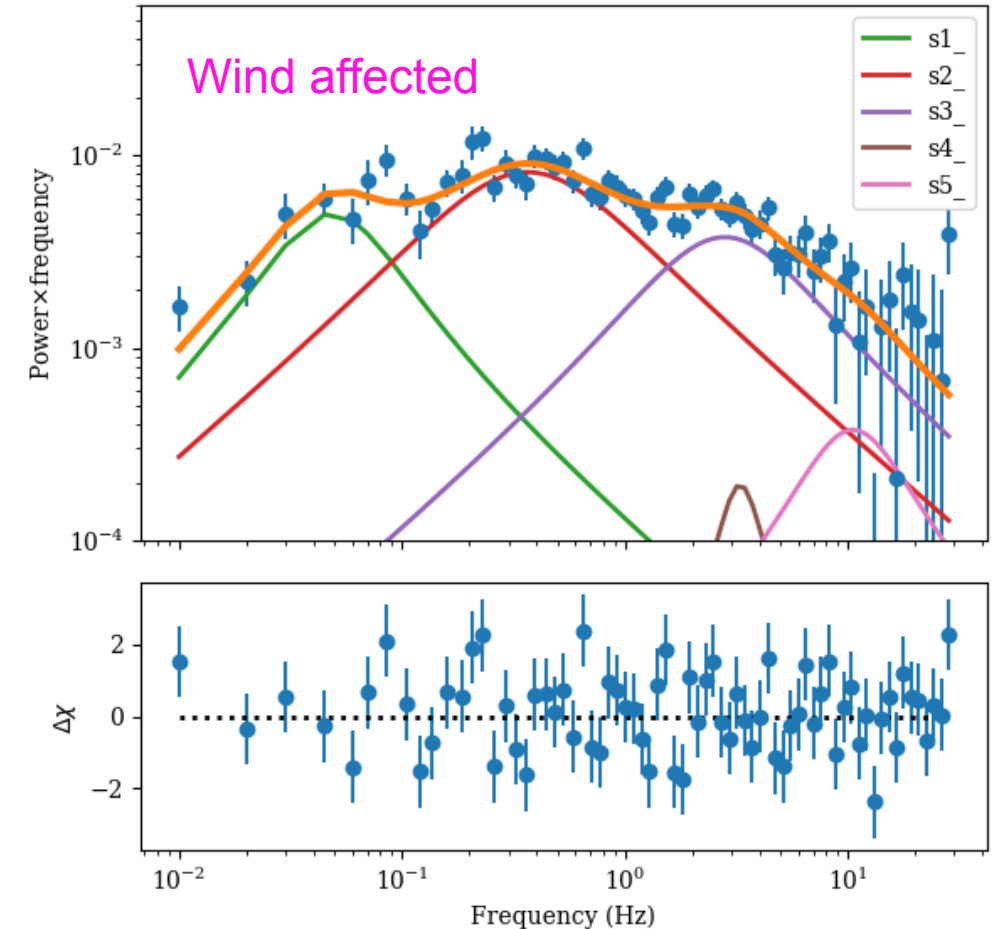
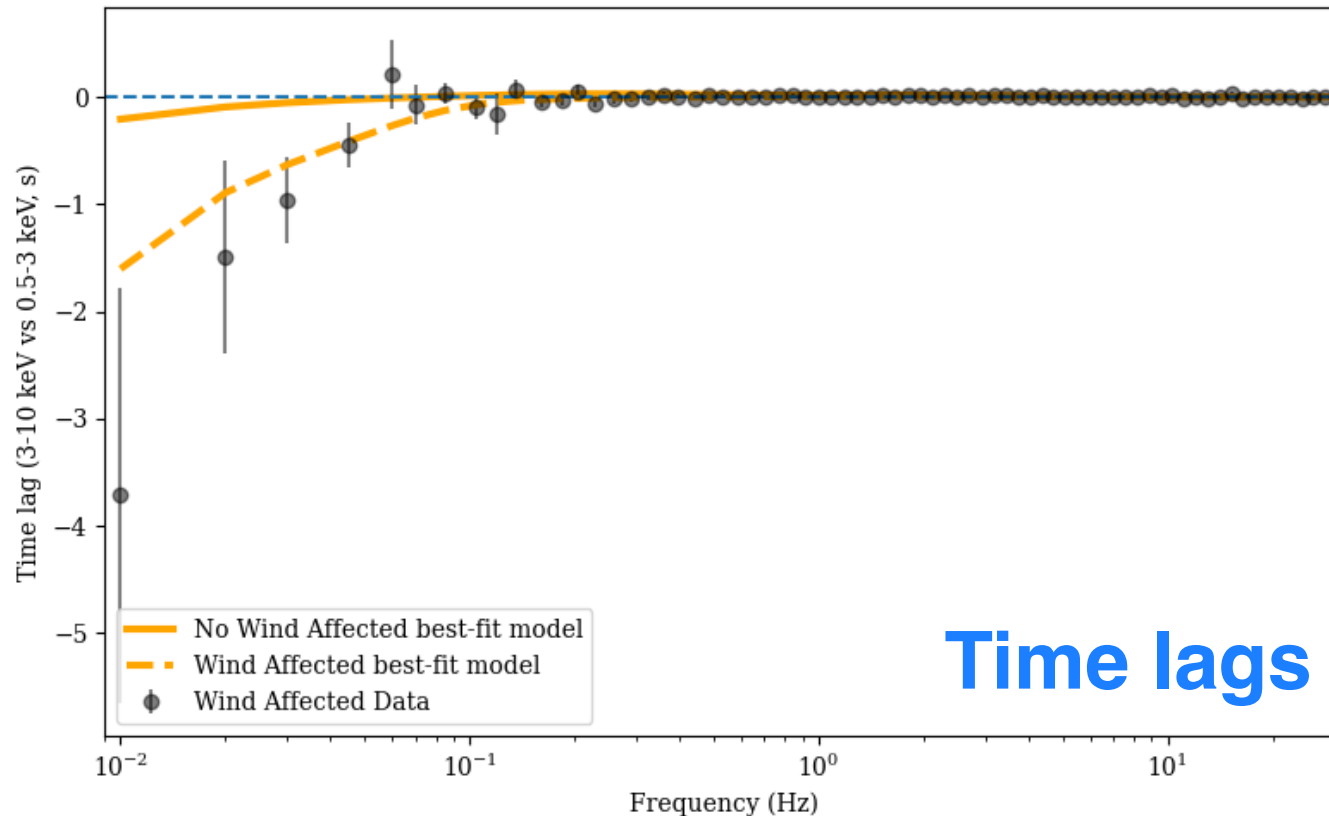


Spectral-timing for stellar wind studies

Nicer, ObsID: 0100320106

0.5 - 3 keV

3 - 10 keV vs 0.5 - 3 keV



Docs and info

Information



How to install it



Tutorials



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

 **Stingray**:docs

stingray v2.3.2 »

Page Contents

Stingray: Next-Generation Spectral Timing

Features

Current Capabilities

1. Data handling and simulation
2. Fourier methods
3. Other time series methods

Future Plans

Platform-specific issues

Installation instructions

Dependencies

Installation

Installing via **conda**

Installing via **pip**

Installing from source (bleeding edge version)

Installing development environment (for new contributors)

Test Suite

Building the Documentation

Using Stingray

A Spectral timing exploration

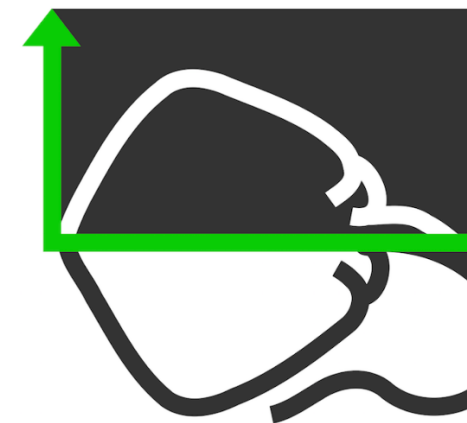
Stingray fundamentals

Advanced

Additional information

Indices and tables

Stingray: Next-Generation Spectral Timing

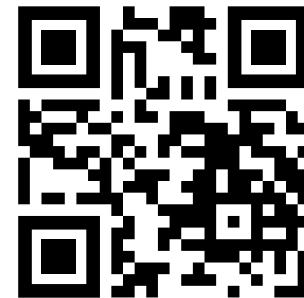


Usage	Release	Development
python >=3.8	release v2.3.2	CI Tests pass
docs latest	JOSS 10.21105/joss.07389	repo status Active
License MIT	DOI 10.5281/zenodo.1490116	codecov 95%

Stingray is a Python library designed to perform times series analysis and related tasks. [Installation](#)

Messages to bring home...

- Stingray is an **easy, free** to use, and **well-maintained** 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: Next-Generation Spectral Timing



Stingray:
*your friendly Python library
for spectral-timing*

The background of the image is a collage of numerous small, overlapping photographs of diverse individuals. The photos vary in style, some being professional headshots and others more casual. In the lower right quadrant, there is a white square containing a black line-art icon of a megaphone. A green arrow points upwards from the top of the megaphone, and another green arrow points to the right from the middle of the megaphone. The word "Thank you!" is written in a large, bold, black sans-serif font, centered horizontally and partially overlapping the collage and the megaphone icon.

Thank you!