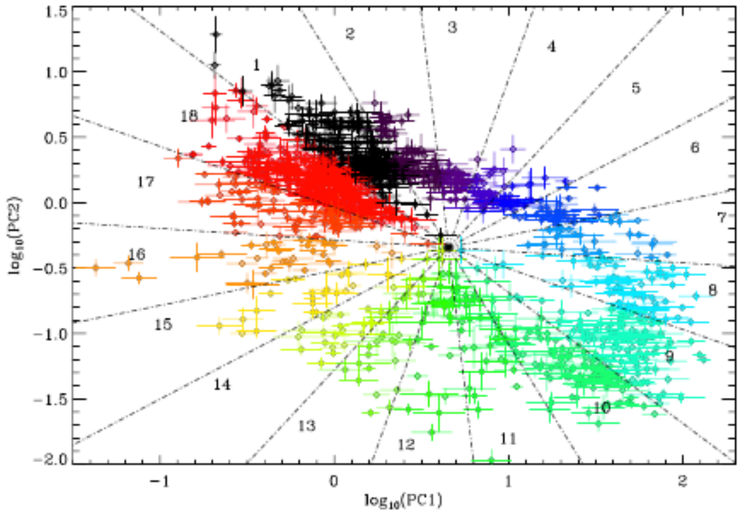
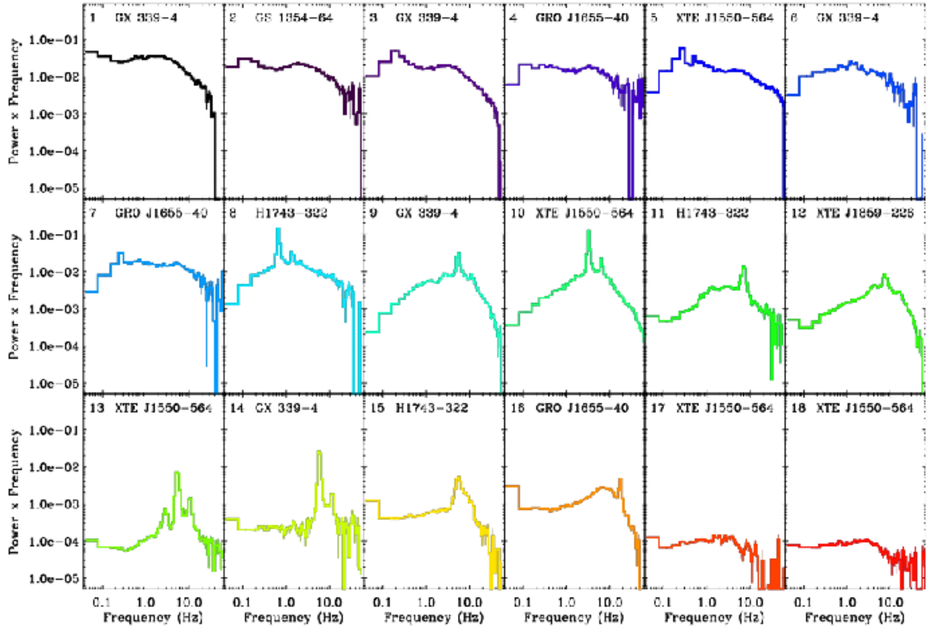
Stingray and DAVE
Spectral timing for all

Matteo Bachetti
INAF-Osservatorio Astronomico di Cagliari

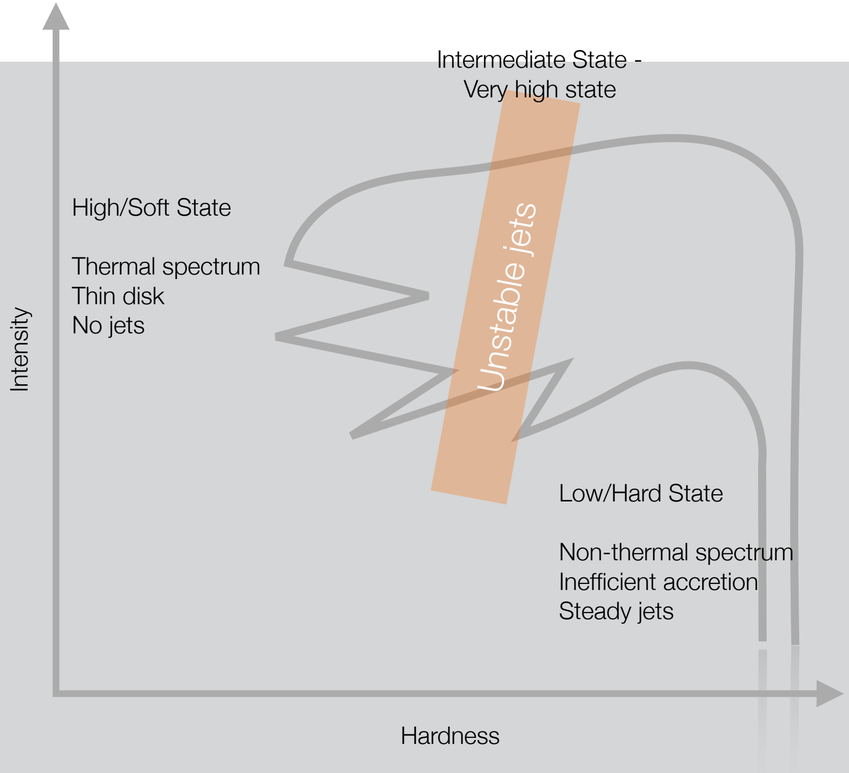
with Daniela Huppenkothen, Abigail Stevens, Paul Balm, Simone Migliari,
Ricardo Valles, Evandro Ribeiro, Himanshu Misra, Usman Khan, (...)

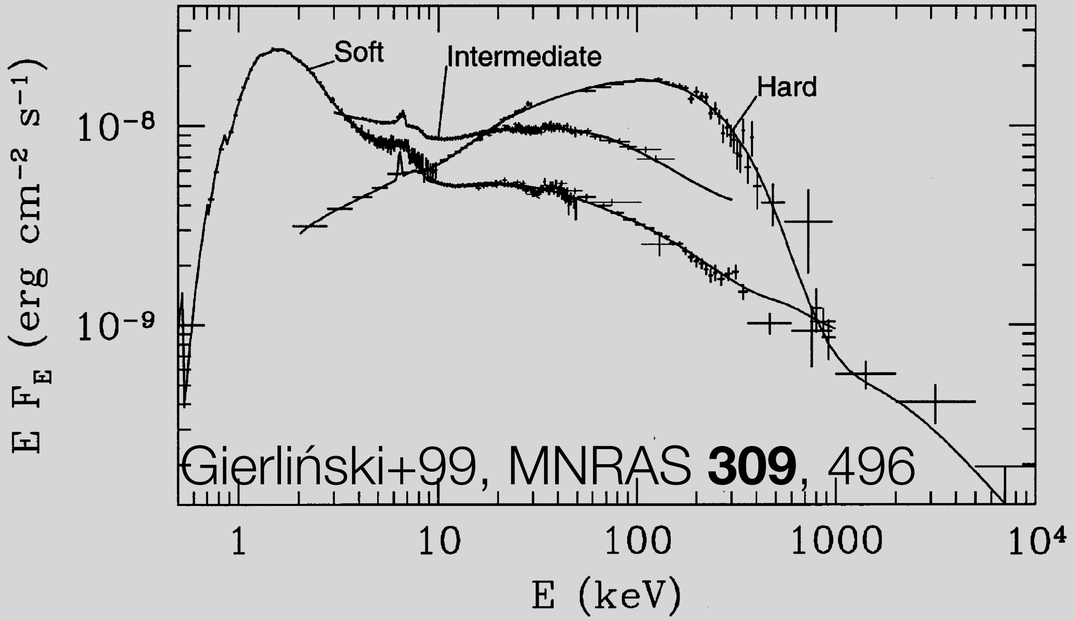


Heil, Uttley, & Klein-Wolt, *MNRAS* **448**, 3339–3347, 2015.

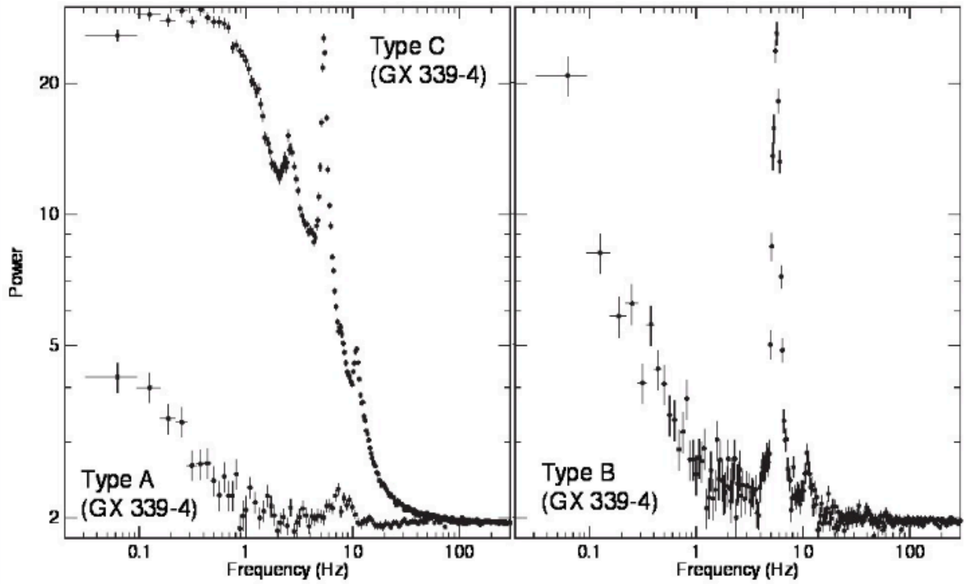


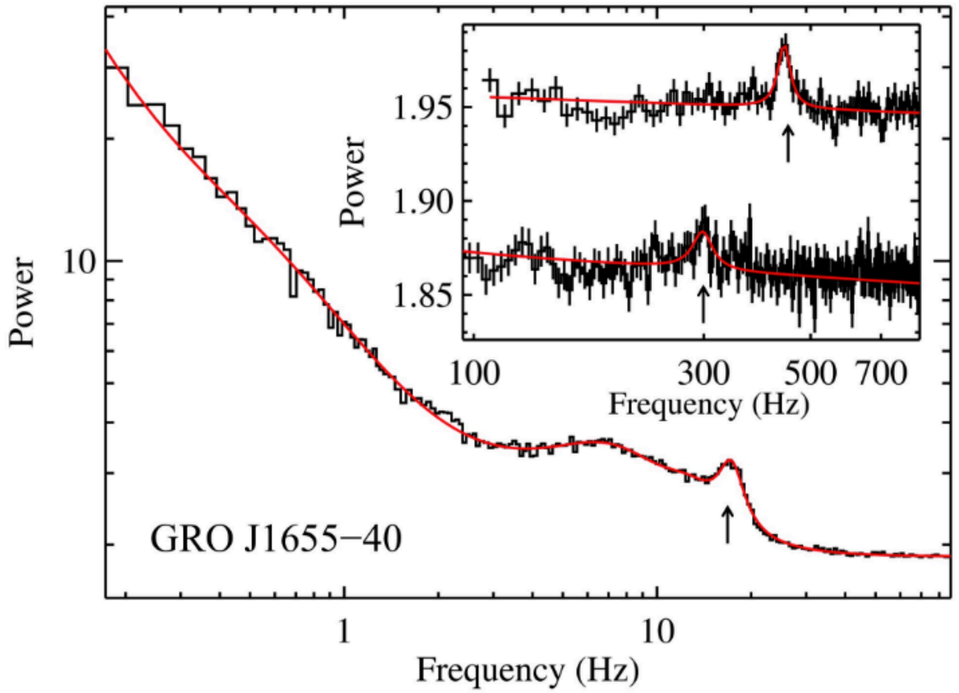
VARIABILITY

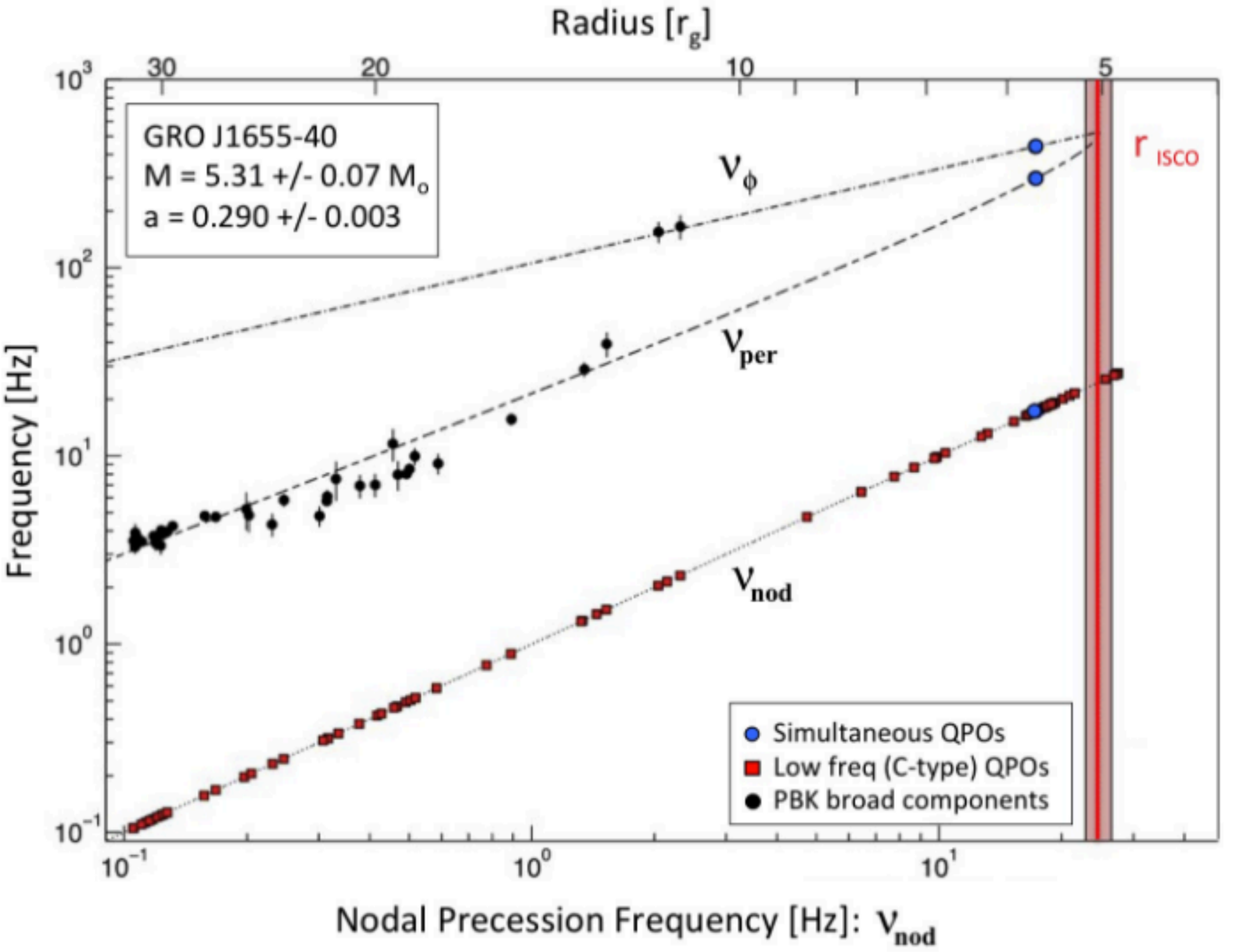




SPECTRA

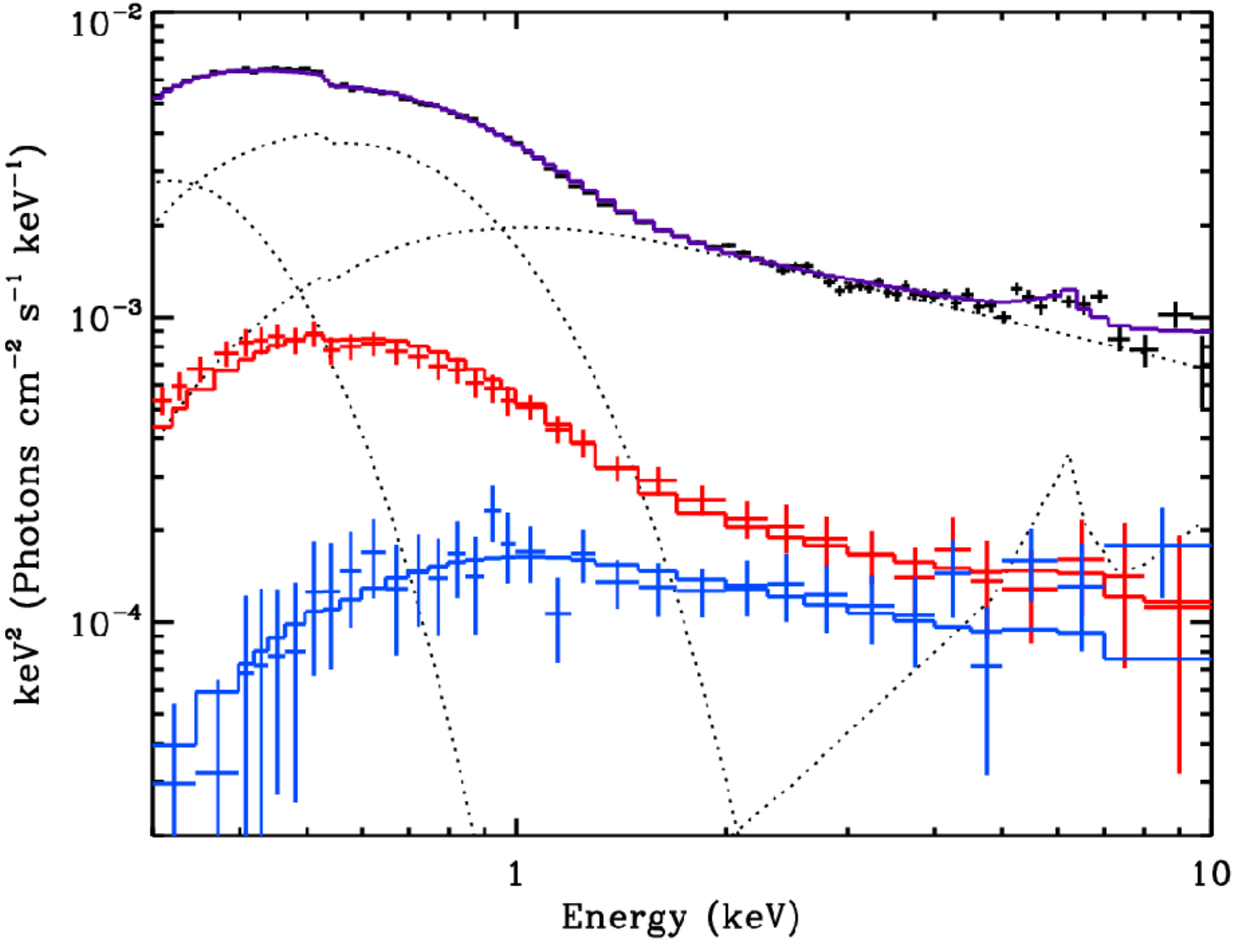


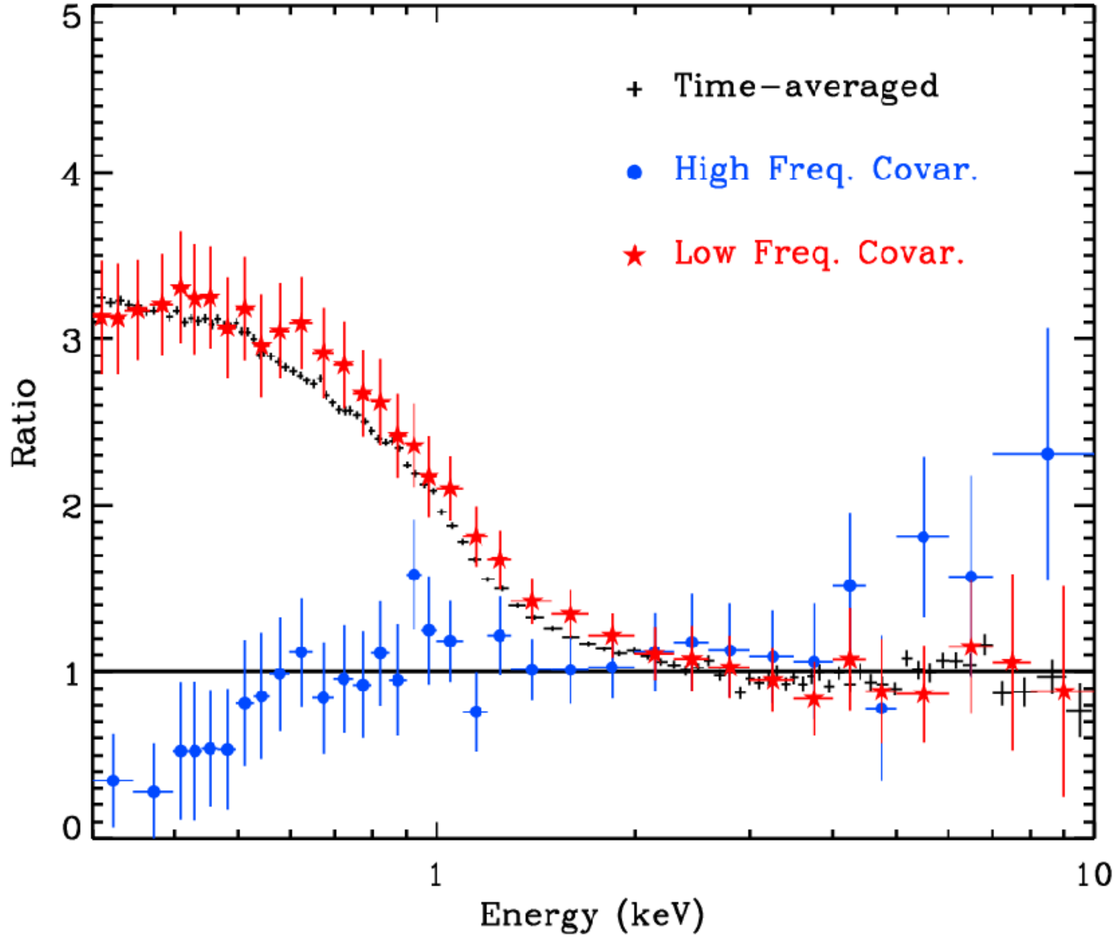




Quasi Periodic Oscillations

Motta+11,14; review: Belloni+*14,* Space Sci.Rev. **183**, 43-60



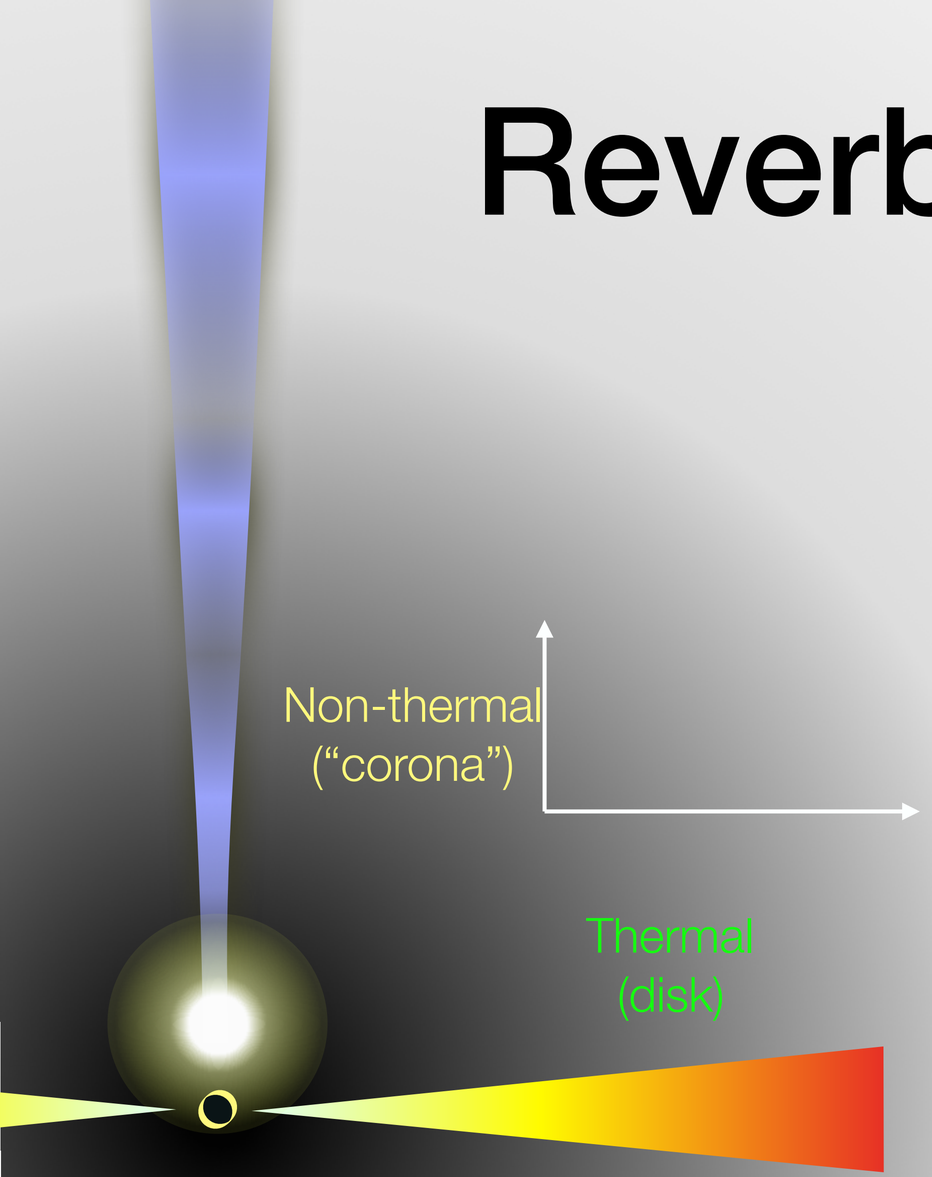


Covariance spectrum

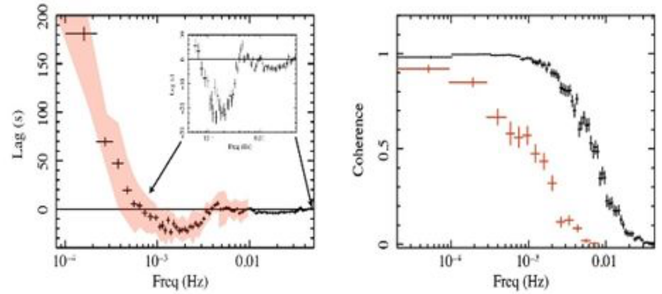
Look what spectral component is dominating the *variability*

e.g. Chichuan,et al. *MNRAS* **436**, 3173-3185, 2015.

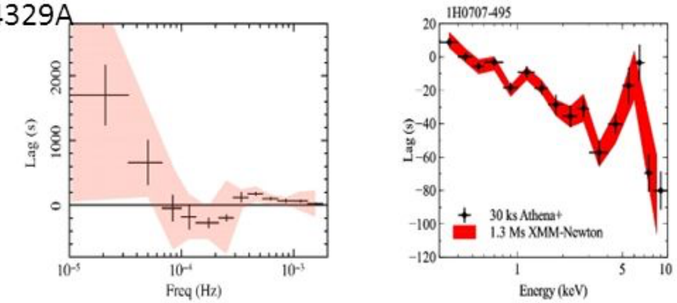
Reverberation mapping



1H0707-495

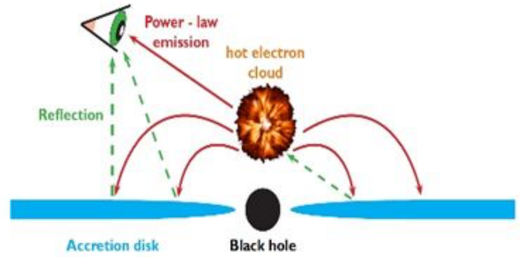


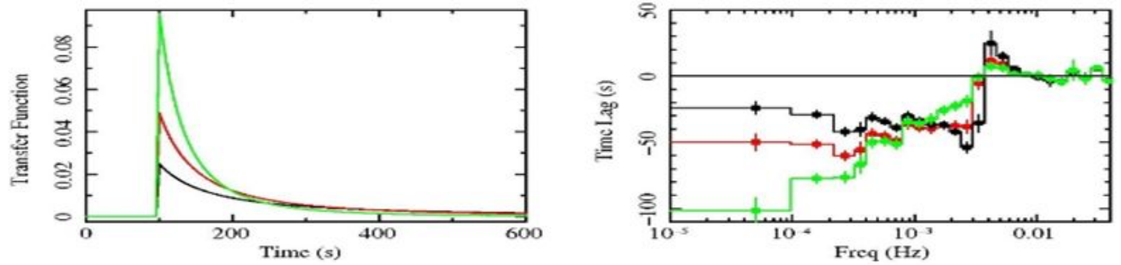
IC4329A



Corona/disk structure of (Reverberation mapping)

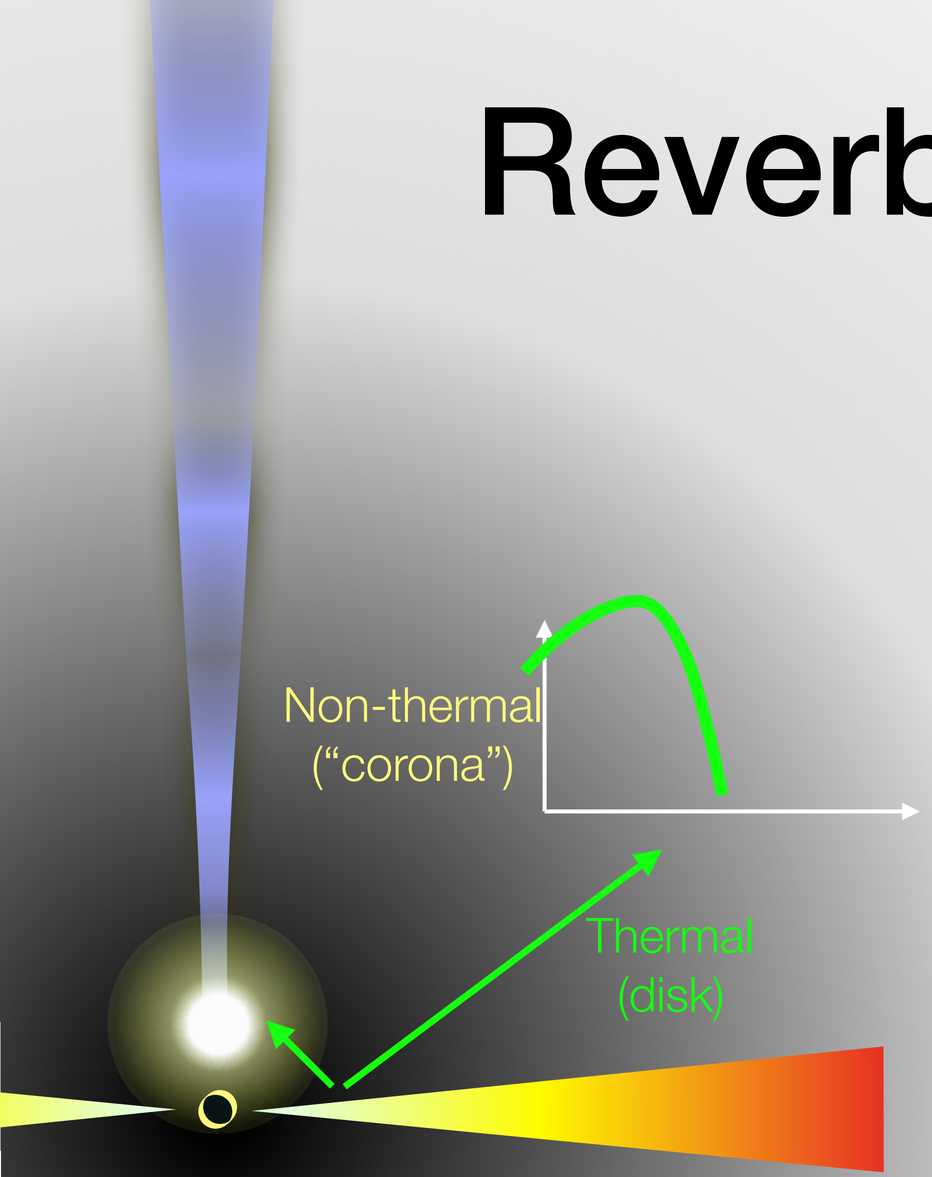
Time lag between 1-4keV and 0.3-1keV, Comparison between Athena and XMM



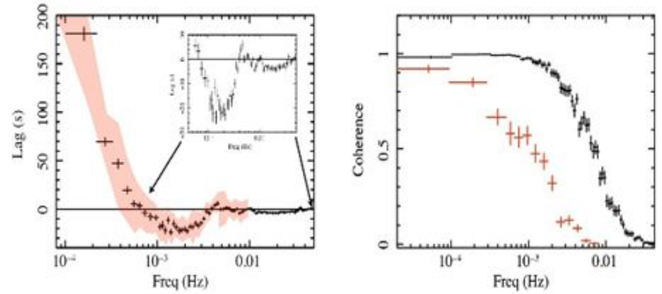


1H0707-495: time lag to probe the transfer structure of corona

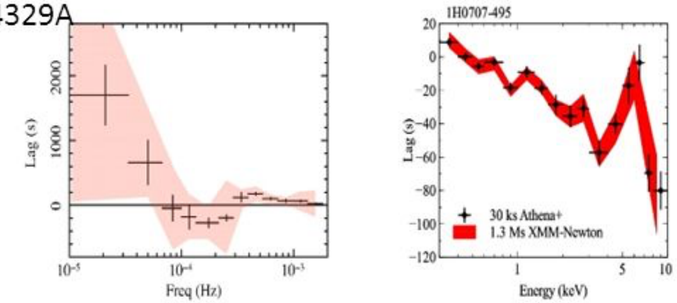
Reverberation mapping



1H0707-495

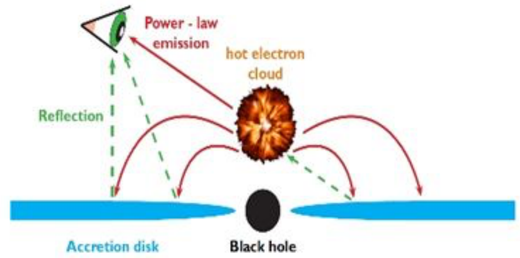


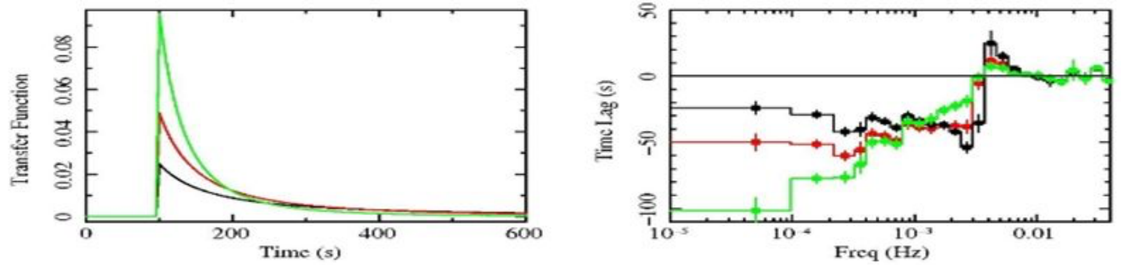
IC4329A



Corona/disk structure of (Reverberation mapping)

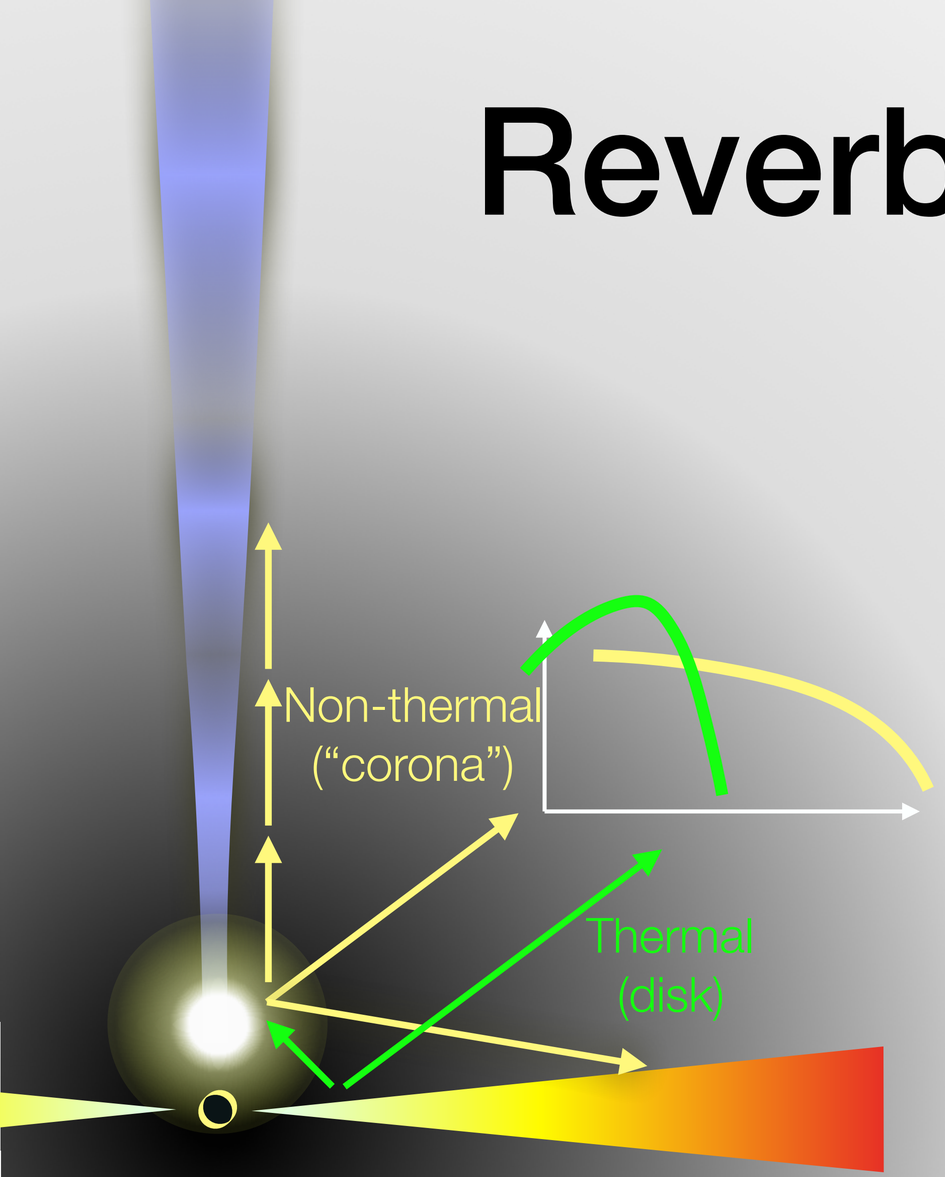
Time lag between 1-4keV and 0.3-1keV, Comparison between Athena and XMM



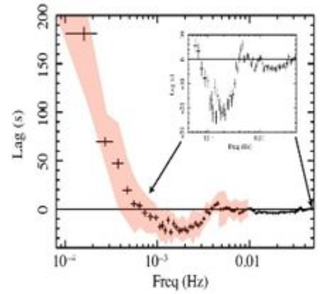


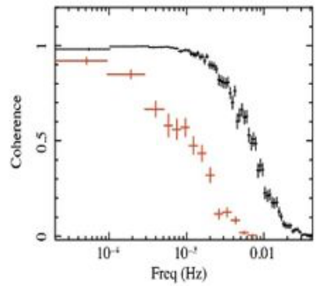
1H0707-495: time lag to probe the transfer structure of corona

Reverberation mapping

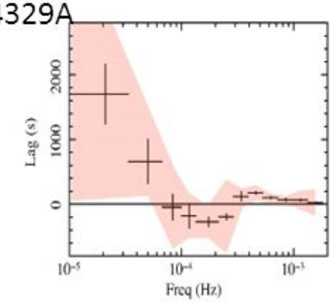


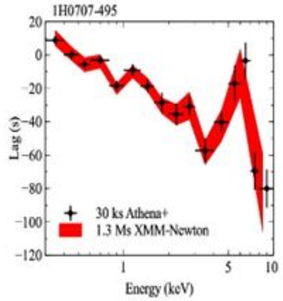
1H0707-495





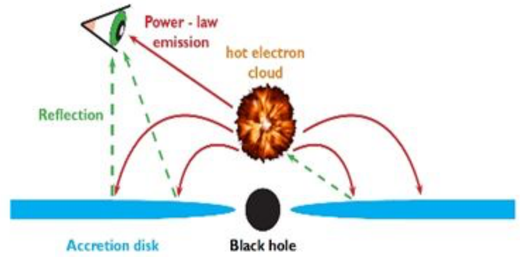
IC4329A

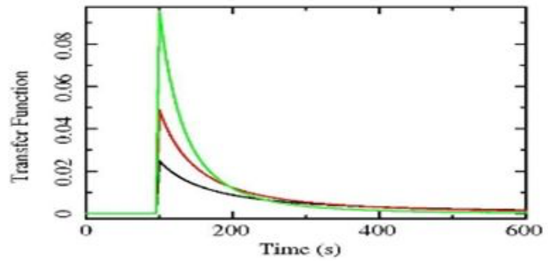


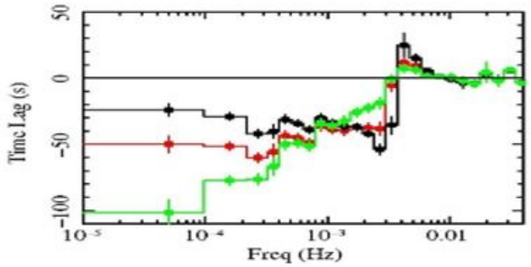


Corona/disk structure of (Reverberation mapping)

Time lag between 1-4keV and 0.3-1keV, Comparison between Athena and XMM

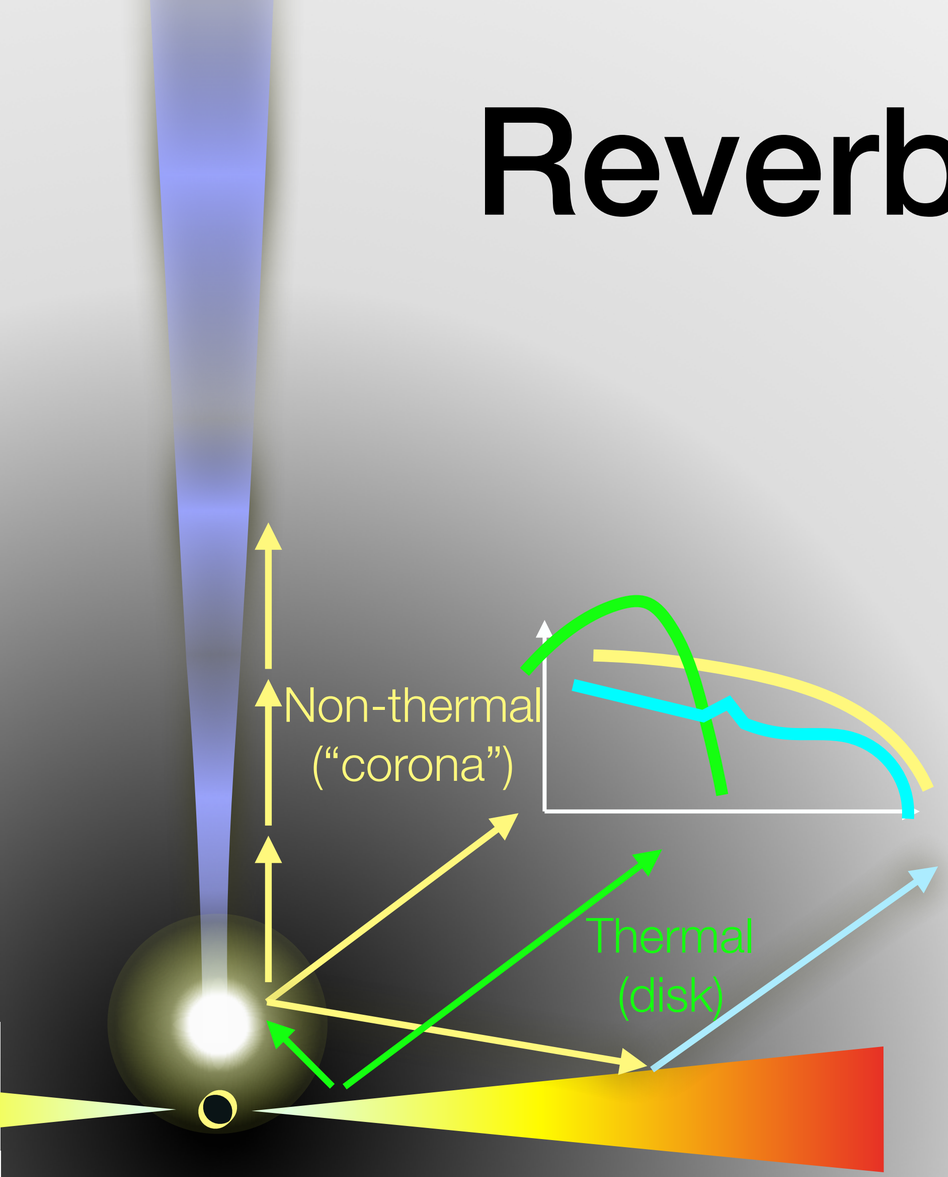




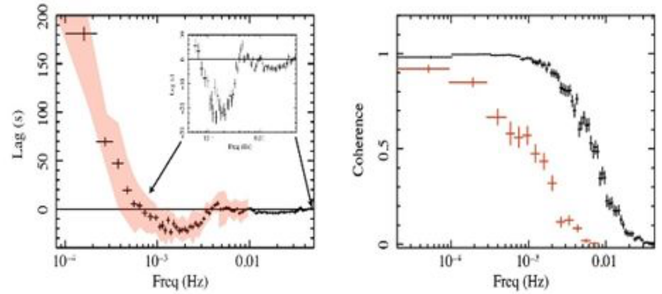


1H0707-495: time lag to probe the transfer structure of corona

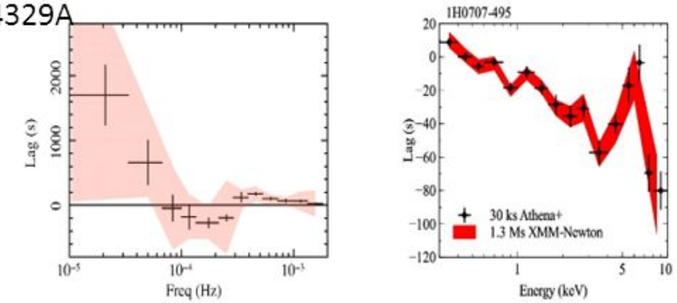
Reverberation mapping



1H0707-495

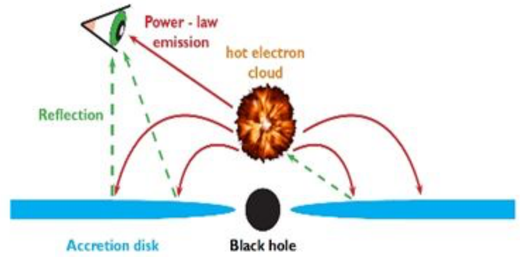


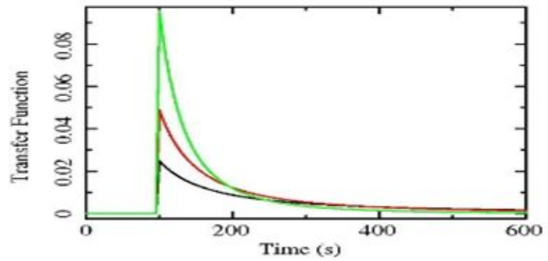
IC4329A

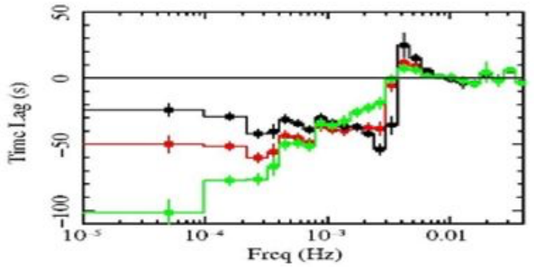


Corona/disk structure of (Reverberation mapping)

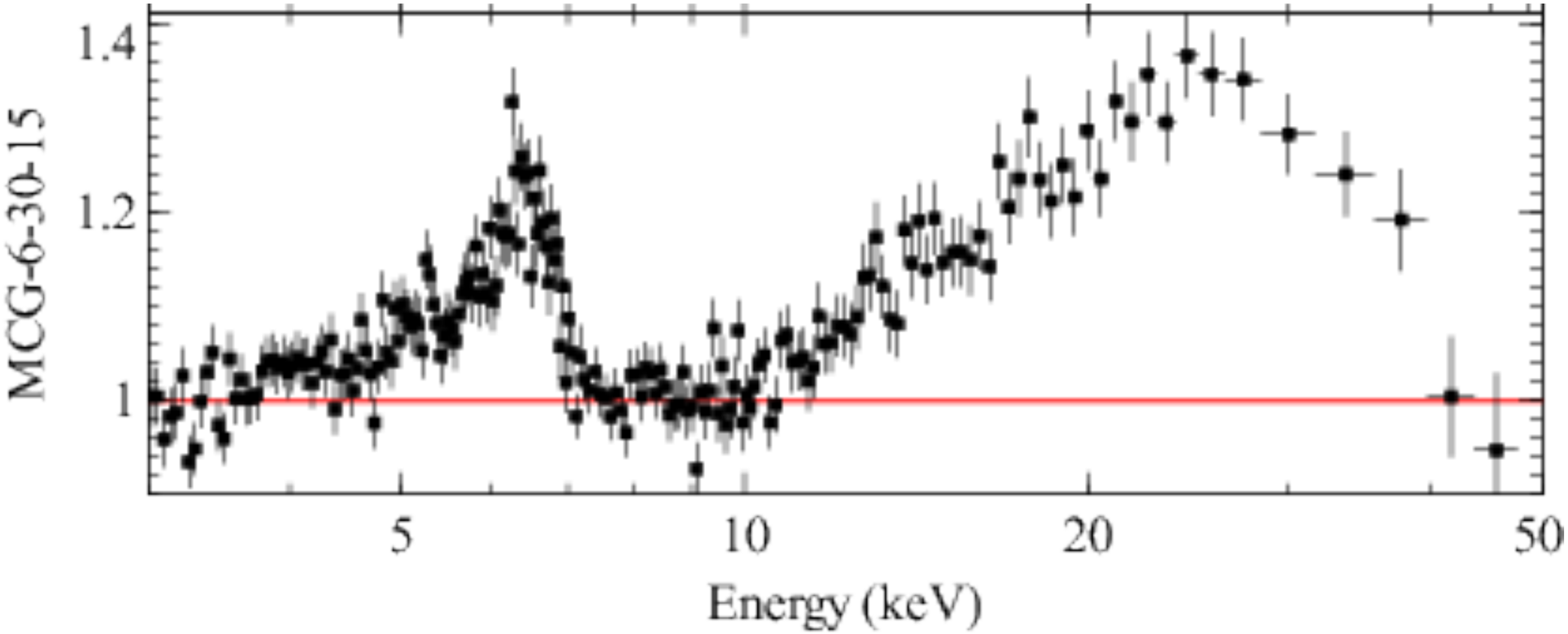
Time lag between 1-4keV and 0.3-1keV, Comparison between Athena and XMM



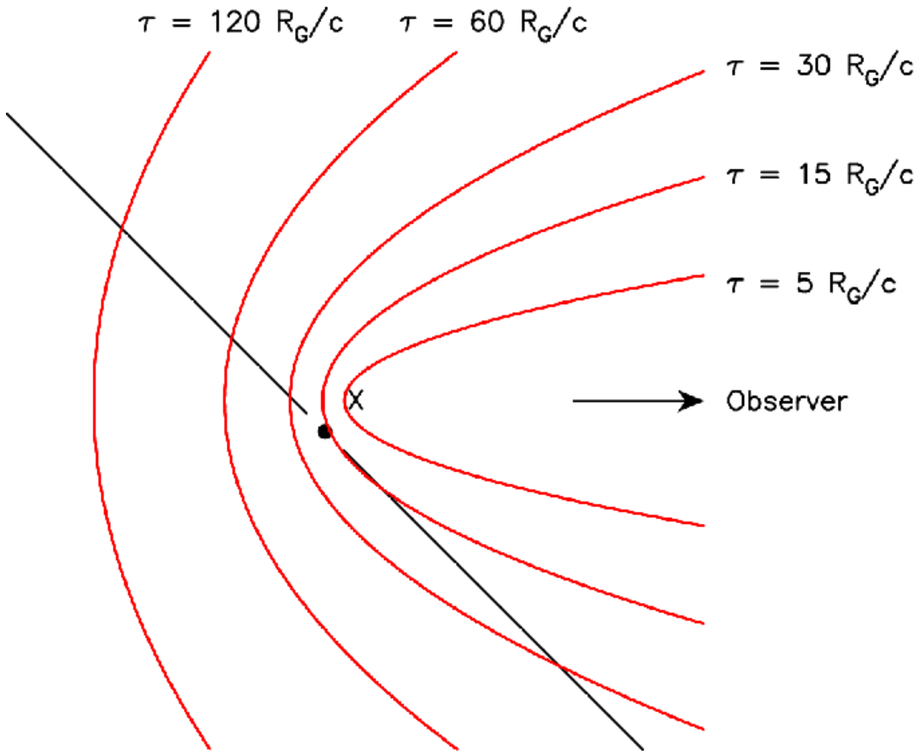


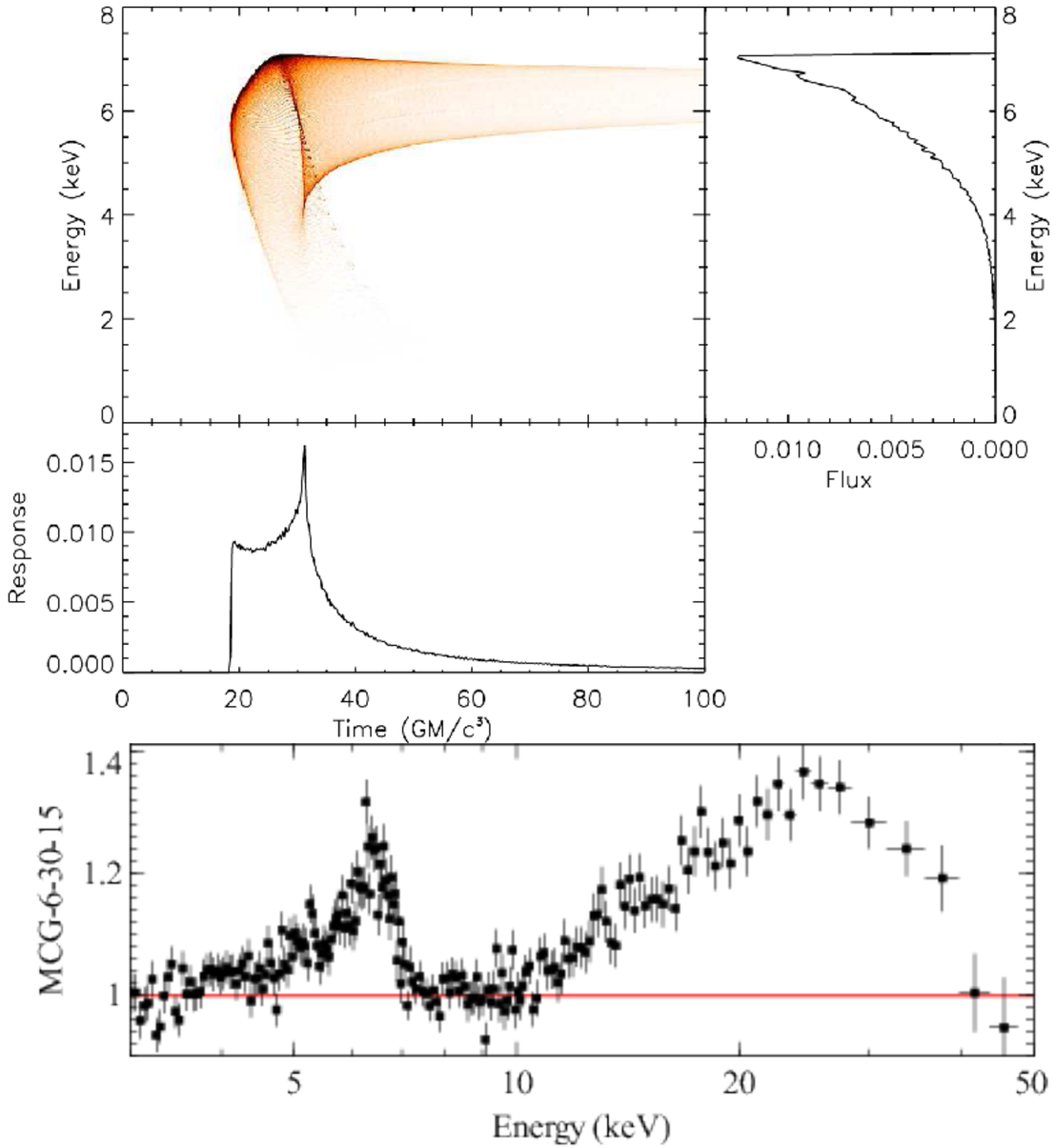


1H0707-495: time lag to probe the transfer structure of corona



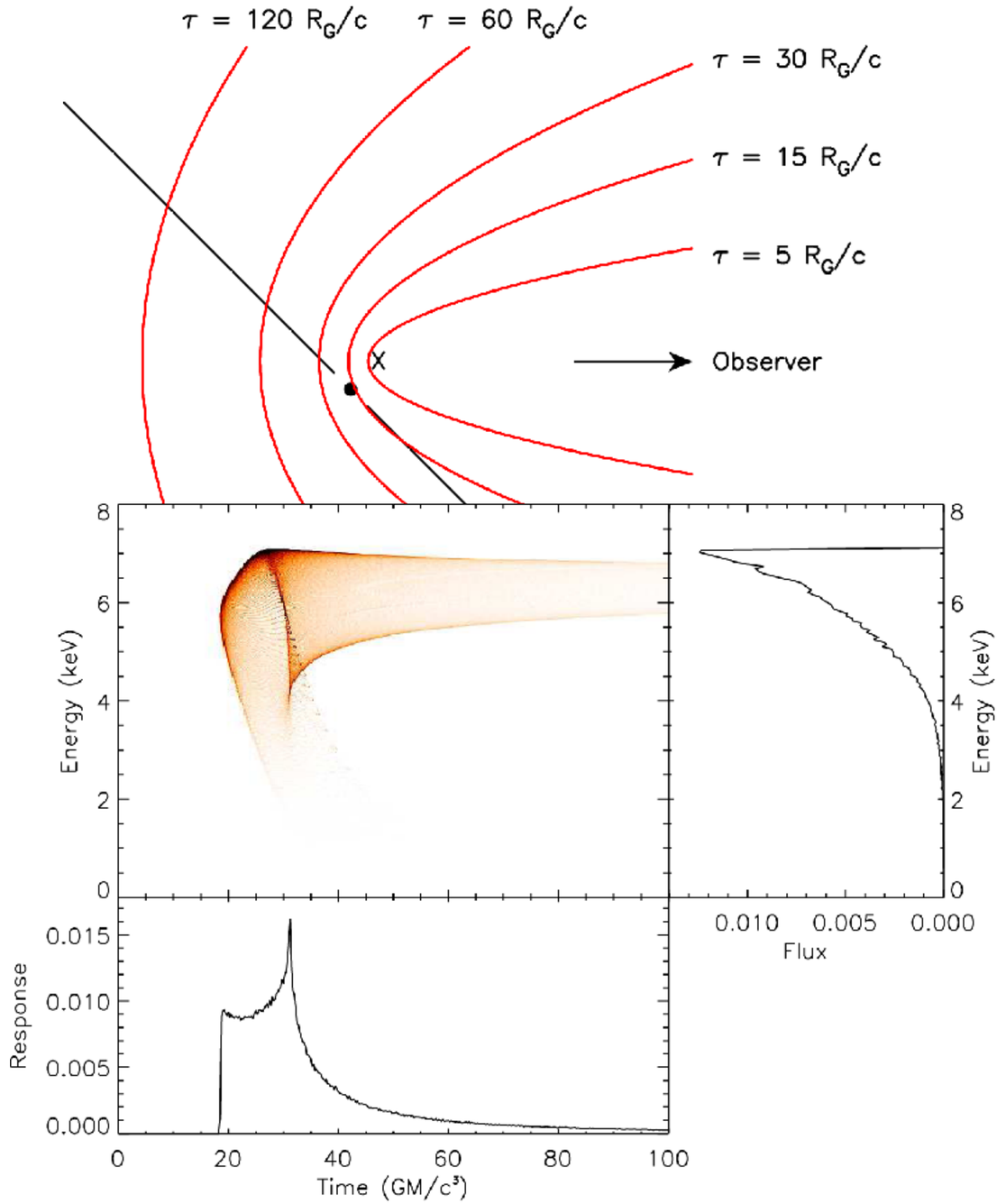
Fabian, Astron.Nachr. **337**, 375-380, 2017

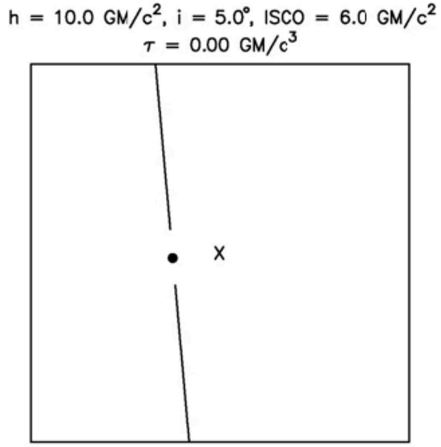


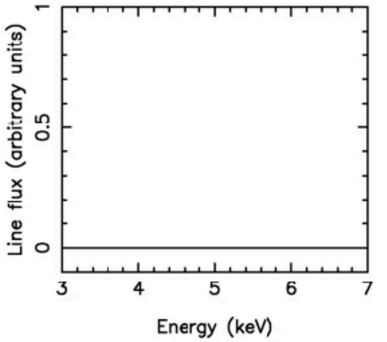


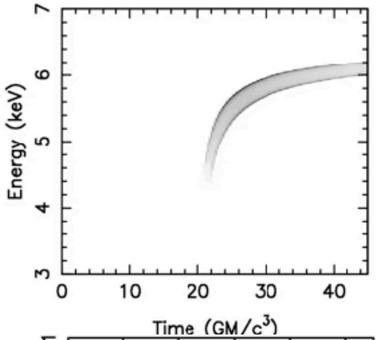
Fabian, Astron.Nachr. **337**, 375-380, 2017

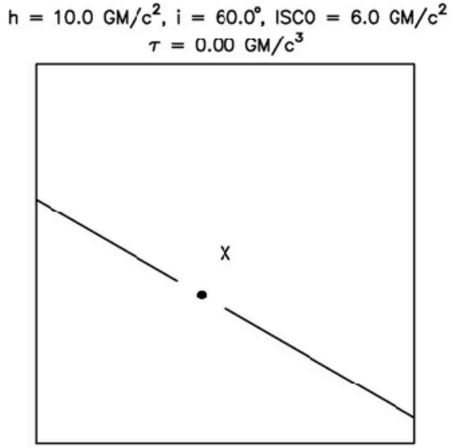
Uttley et al. *A&ARv* **22**, 72–66, 2014.

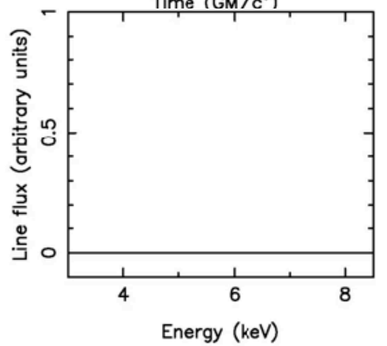


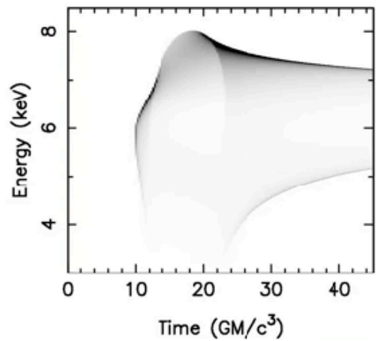




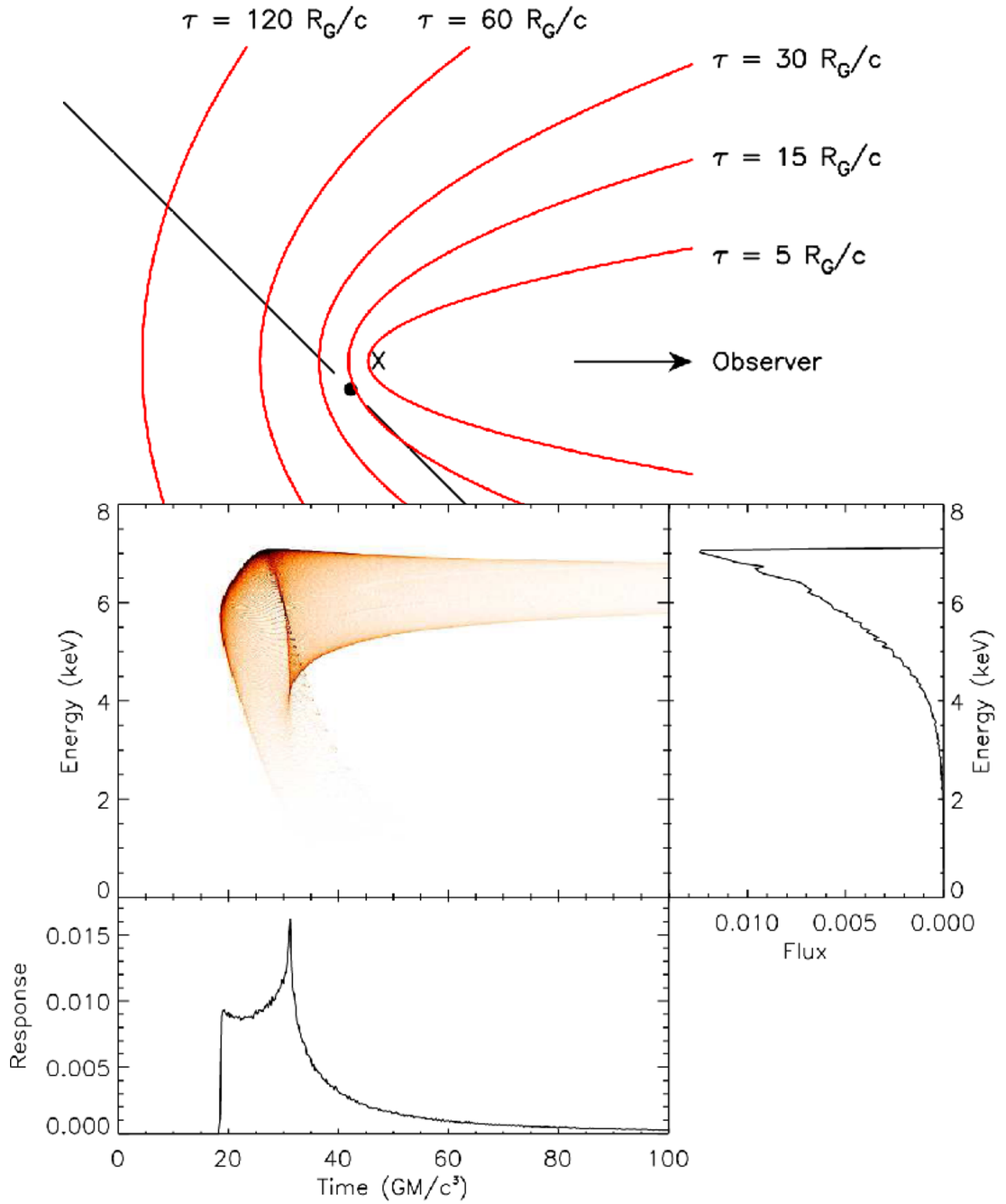


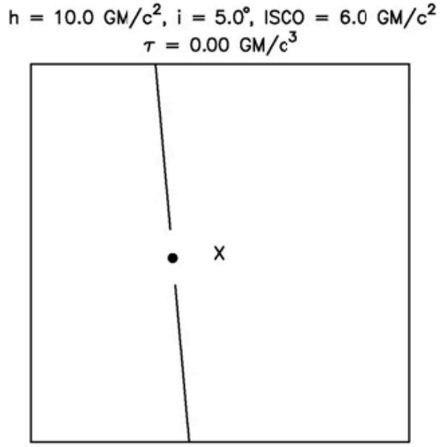


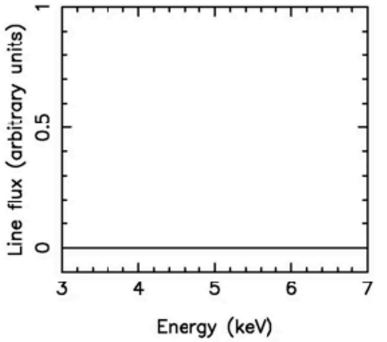


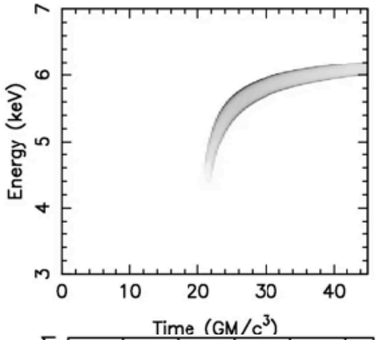


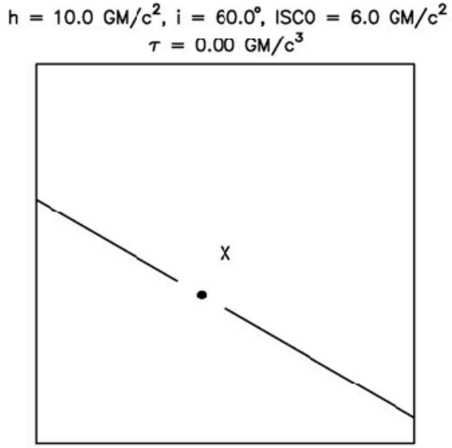
Courtesy of Ed Cackett

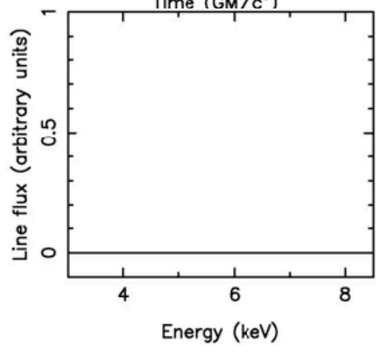


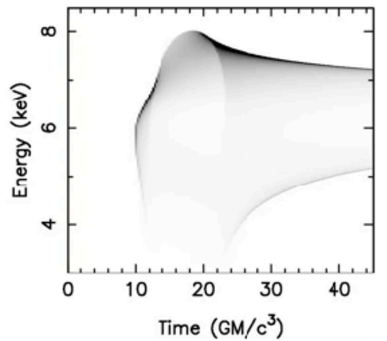












Courtesy of Ed Cackett

Timing analysis

Periodograms
Power density spectra
Lomb-Scargle
Epoch folding search
Z-search, H-search
Cross spectra
Coherence spectra
(…)

Cross-correlation
Autocorrelation

Wavelets
Dynamical power spectra
Phaseogram

Spectral analysis

Continuum modeling
Color-color, Hardness-intensity
Absorption-emission features
High-resolution spectroscopy

Timing analysis

Periodograms
Power density spectra
Lomb-Scargle
Epoch folding search
Z-search, H-search
Cross spectra
Coherence spectra
(...)

Cross-correlation
Autocorrelation

Wavelets
Dynamical power spectra
Phaseogram

Spectral timing

phase-resolved spectra
rms-intensity diagrams
Time lags vs Energy
rms vs Energy
Covariance spectrum
(...)

Spectral analysis

Continuum modeling
Color-color, Hardness-intensity
Absorption-emission features
High-resolution spectroscopy

Existing “public” software
(i.e. you can look at the code and read docs)

Spectral analysis	Timing analysis (+ lags)	Spectral timing

- Xspec
- Sherpa
- ISIS
- (…)

Existing “public” software
(i.e. you can look at the code and read docs)

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

Existing “public” software
(i.e. you can look at the code and read docs)

Spectral analysis	Timing analysis (+ lags)	Spectral timing

- Xspec
- Sherpa
- ISIS
- (…)

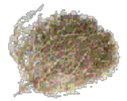
- (XRONOS) POWSPEC 
- SITAR, lsisscripts.sl

Existing “public” software
(i.e. you can look at the code and read docs)

Spectral analysis	Timing analysis (+ lags)	Spectral timing

- Xspec
- Sherpa
- ISIS
- (…)

- (XRONOS) POWSPEC
- SITAR, lsisscripts.sl

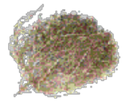


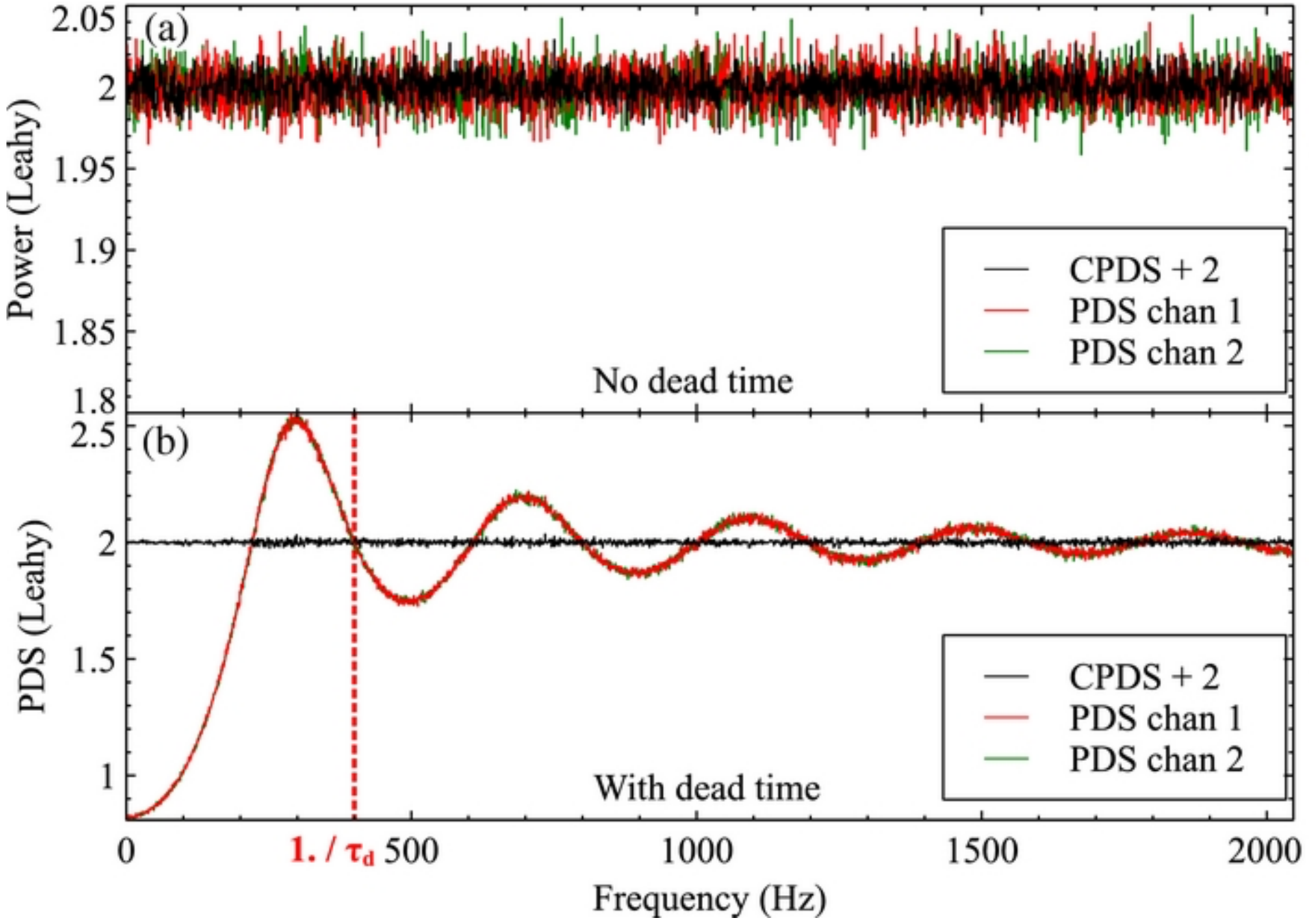
Existing “public” software
(i.e. you can look at the code and read docs)

Spectral analysis	Timing analysis (+ lags)	Spectral timing

- Xspec
- Sherpa
- ISIS
- (…)

- (XRONOS) POWSPEC
- SITAR, lsisscripts.sl





Bachetti et al. 2015, *ApJ* **800**,109, 2015

Page Contents
MaLTPyNT documentation
- What's new
- Preliminary notes
  - MaLTPyNT vs FTOOLS (and together with FTOOLS)
    - vs POWSPEC
    - Clarification about dead time treatment
  - License and notes for the users
  - Acknowledgements
- Getting started
- Command line interface
- API documentation

Indices and tables

ascl:1502.021

MaLTPyNT documentation

The MaLTPyNT (Matteo's Libraries and Tools in Python for NuSTAR Timing) suite is designed for the **quick-look timing analysis** of NuSTAR data. It treats properly orbital gaps (e.g., occultation, SAA passages) and performs the standard aperiodic timing analysis (power density spectrum, lags, etc.), plus the **cospectrum**, a proxy for the power density spectrum that uses the signals from two detectors instead of a single one (for an explanation of why this is important in NuSTAR, look at Bachetti et al., *ApJ*, **800**, 109 -arXiv:1409.3248). The output of the analysis, be it a cospectrum, a power density spectrum, or a lag spectrum, can be fitted with Xspec, Isis or any other spectral fitting program.

Despite its main focus on NuSTAR, the software can be used to make standard spectral analysis on X-ray data from, in principle, any other satellite (for sure XMM-Newton and RXTE). Input files can be any event lists in FITS format, provided that they meet certain minimum standard. Also, light curves in FITS format or text format can be used. See the documentation of `MPlcurve` for more information.

What's new

Note

MaLTPyNT provisionally accepted as an Astropy affiliated package

In preparation for the 2.0 release, the API has received some visible changes. Names do not have the `mp_` prefix anymore, as they were very redundant; the structure of the code base is now based on the AstroPy structure; tests have been moved and the documentation improved.

`MPexposure` is a new livetime correction script on sub-second timescales for NuSTAR. It will be able to replace `nulccorr`, and get results on shorter bin times, in observations done with a specific observing mode, where the observer has explicitly requested to telemeter all events (including rejected) and the user has run `nupipeline` with the `CLEANCOLS = NO` option. This tool is under testing.

`MPfake` is a new script to create fake observation files in FITS format, for testing. New functions to create fake data will be added to `maltpynt.fake`.

Preliminary notes

2014-2016

In-development astropy-affiliated package.

Things to work on to be accepted:
1. The name
2. Better documented API

Page Contents
MaLTPyNT documentation
- What's new
- Preliminary notes
  - MaLTPyNT vs FTOOLS (and together with FTOOLS)
    - vs POWSPEC
    - Clarification about dead time treatment
  - License and notes for the users
  - Acknowledgements
- Getting started
- Command line interface
- API documentation

Indices and tables

MaLTPyNT
Matteo's
Library and Tools in Python
for NuSTAR Timing

ascl:1502.021

MaLTPyNT documentation

The MaLTPyNT (Matteo's Libraries and Tools in Python for NuSTAR Timing) suite is designed for the **quick-look timing analysis** of NuSTAR data. It treats properly orbital gaps (e.g., occultation, SAA passages) and performs the standard aperiodic timing analysis (power density spectrum, lags, etc.), plus the **cospectrum**, a proxy for the power density spectrum that uses the signals from two detectors instead of a single one (for an explanation of why this is important in NuSTAR, look at Bachetti et al., *ApJ*, **800**, 109 -arXiv:1409.3248). The output of the analysis, be it a cospectrum, a power density spectrum, or a lag spectrum, can be fitted with Xspec, Isis or any other spectral fitting program.

Despite its main focus on NuSTAR, the software can be used to make standard spectral analysis on X-ray data from, in principle, any other satellite (for sure XMM-Newton and RXTE). Input files can be any event lists in FITS format, provided that they meet certain minimum standard. Also, light curves in FITS format or text format can be used. See the documentation of `MPlcurve` for more information.

What's new

Note

MaLTPyNT provisionally accepted as an Astropy affiliated package

In preparation for the 2.0 release, the API has received some visible changes. Names do not have the `mp_` prefix anymore, as they were very redundant; the structure of the code base is now based on the AstroPy structure; tests have been moved and the documentation improved.

`MPexposure` is a new livetime correction script on sub-second timescales for NuSTAR. It will be able to replace `nulccorr`, and get results on shorter bin times, in observations done with a specific observing mode, where the observer has explicitly requested to telemeter all events (including rejected) and the user has run `nupipeline` with the `CLEANCOLS = NO` option. This tool is under testing.

`MPfake` is a new script to create fake observation files in FITS format, for testing. New functions to create fake data will be added to `maltpynt.fake`.

Preliminary notes

2014-2016

In-development astropy-affiliated package.

Things to work on to be accepted:
1. The name
2. Better documented API

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

Stingray
Daniela Huppenkothen,
Abigail Stevens

MaLTPyNT
(in-development
Astropy-affiliated)
Matteo Bachetti

GUI for timing
Simone Migliari, Paul Balm

Stingray
Python API

“ex-MaLTPyNT”
CLI

DAVE
GUI

README.rst

Stingray

Master	build passing	docs latest	slack 4/54	coverage 97%

X-Ray Spectral Timing Made Easy

Stingray is an in-development spectral-timing software package for astrophysical X-ray (and more) data. Stingray merges existing efforts for a (spectral-)timing package in Python, and is structured with the best guidelines for modern open-source programming, following the example of Astropy .

It is composed of:

1. a library of time series methods, including power spectra, cross spectra, covariance spectra, lags, and so on;
2. a set of scripts to load FITS data files from different missions;
3. a simulator of light curves and event lists, that includes different kinds of variability and more complicated phenomena based on the impulse response of given physical events (e.g. reverberation);
4. finally, an in-development GUI to ease the learning curve for new users.

README.rst

Stingray

Master	build passing	docs latest	slack 4/54	coverage 97%

X-Ray Spectral Timing Made Easy

Stingray is an in-development spectral-timing software package for astrophysical X-ray (and more) data. Stingray merges existing efforts for a (spectral-)timing package in Python, and is structured with the best guidelines for modern open-source programming, following the example of Astropy .

It is composed of:

1. a library of time series methods, including power spectra, cross spectra, covariance spectra, lags, and so on;
2. a set of scripts to load FITS data files from different missions;
3. a simulator of light curves and event lists, that includes different kinds of variability and more complicated phenomena based on the impulse response of given physical events (e.g. reverberation);
4. finally, an in-development GUI to ease the learning curve for new users.

Stingray API

Stingray

EventList

Lightcurve

Powerspectrum
Averagedpowerspectrum
Crossspectrum
Averagedcrossspectrum

Covariancespectrum
Crosscorrelation
RmsEnergySpectrum
LagEnergySpectrum

Phase-resolved spectra

Stingray API

modeling

simulator

pulse

Stingray

EventList

Lightcurve

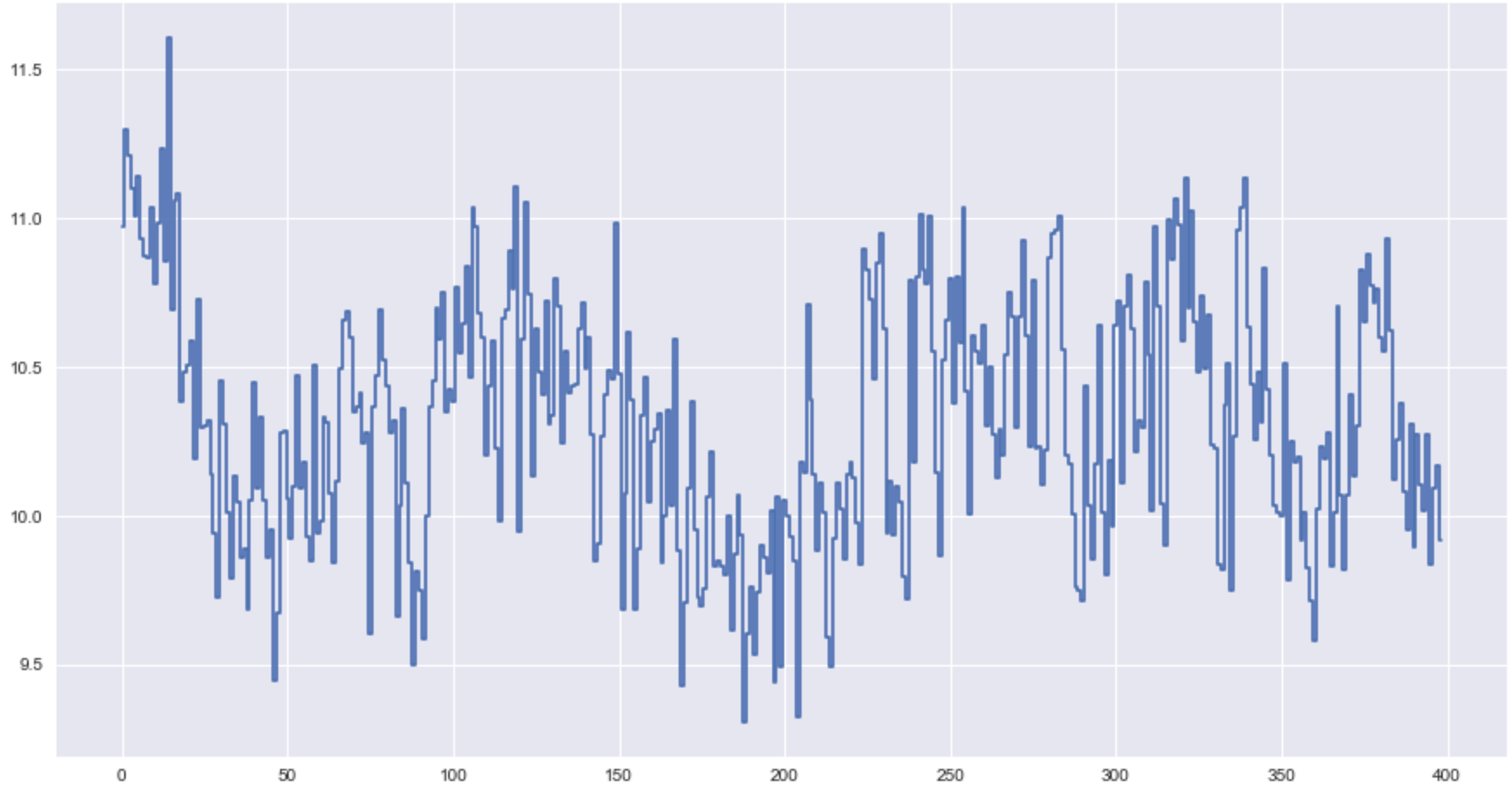
Powerspectrum
Averagedpowerspectrum
Crossspectrum
Averagedcrossspectrum

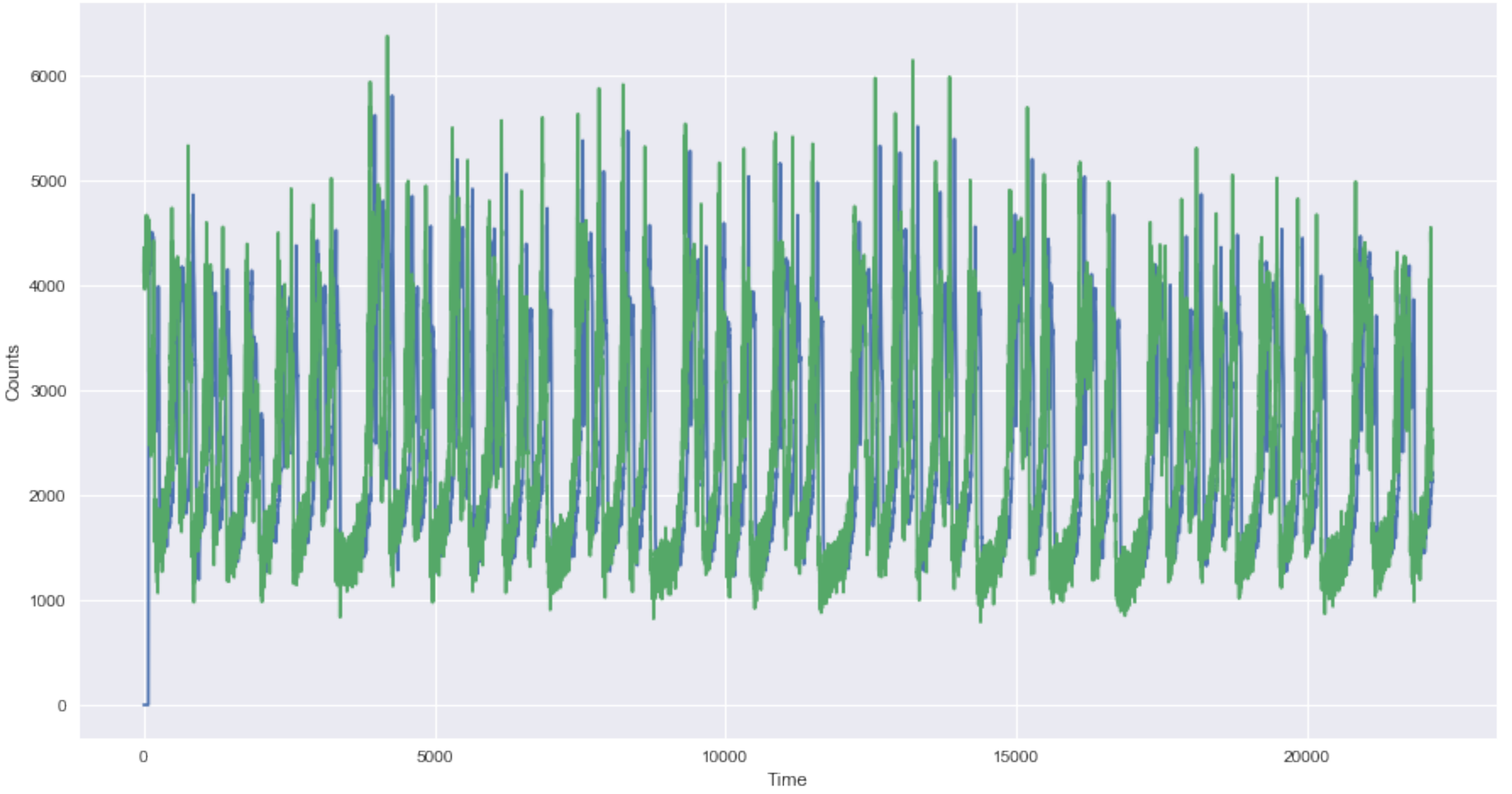
Covariancespectrum
Crosscorrelation
RmsEnergySpectrum
LagEnergySpectrum

Phase-resolved spectra

In [8]:
```
lc = sim.simulate('smoothbknpo', [.6, 0.9, .2, 4])
plt.plot(lc.counts[1:400], drawstyle='steps-mid')
```

Out[8]:
```
[<matplotlib.lines.Line2D at 0x11d260630>]
```




6000
5000
4000
3000
2000
1000
0
Counts
0
5000
10000
15000
20000
Time

usmanwardag Adds real 2-d plot — a3ebf31 on Aug 22, 2016

1 contributor

515 lines (514 sloc) | 79.5 KB | Raw | Blame | History

Contents

This notebook covers the basics of creating TransferFunction object, obtaining time and energy resolved responses, plotting them and using IO methods available. Finally, artificial responses are introduced which provide a way for quick testing.

Setup

Set up some useful libraries.

In [39]:
```
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

Import relevant stingray libraries.

In [40]:
```
from stingray.simulator.transfer import TransferFunction
from stingray.simulator.transfer import simple_ir, relativistic_ir
```

Creating TransferFunction

ex-MaLTPyNT

```
$ MPreadevents 002A.evt 002B.evt
$ MPcalibrate 002A_ev.nc 002B_ev.nc -r file.rmf
$ MPlcurve 002A_ev_calib.nc 002B_ev_calib.nc -e 3 30
$ MPfspec 002A_E3-30_lc.nc 002B_E3-30_lc.nc \
    -k CPDS -o cpds_002_3-30 --norm rms
$ MPrebin cpds_002_3-30_0.nc -r 1.03
```

Creativity needed!

Help us choose the new name for MaLTPyNT!

The acronym should mention synonyms of timing/ variability/spectral timing, compact objects/black holes/ pulsars, X-rays/high energies. Mentions to music are highly appreciated

Saving the spectra in a format readable to XSpec

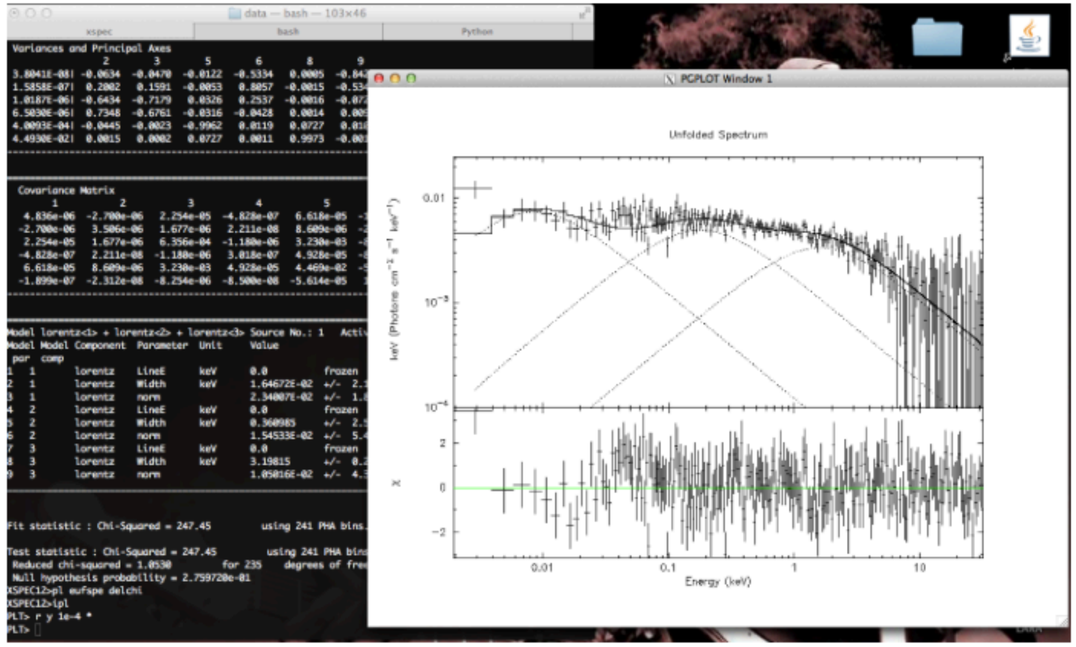
To save the cospectrum in a format readable to XSpec it is sufficient to give the command

```
$ MP2xspec cpds_002_3-30_0_rebin1.03.nc --flx2xsp
```

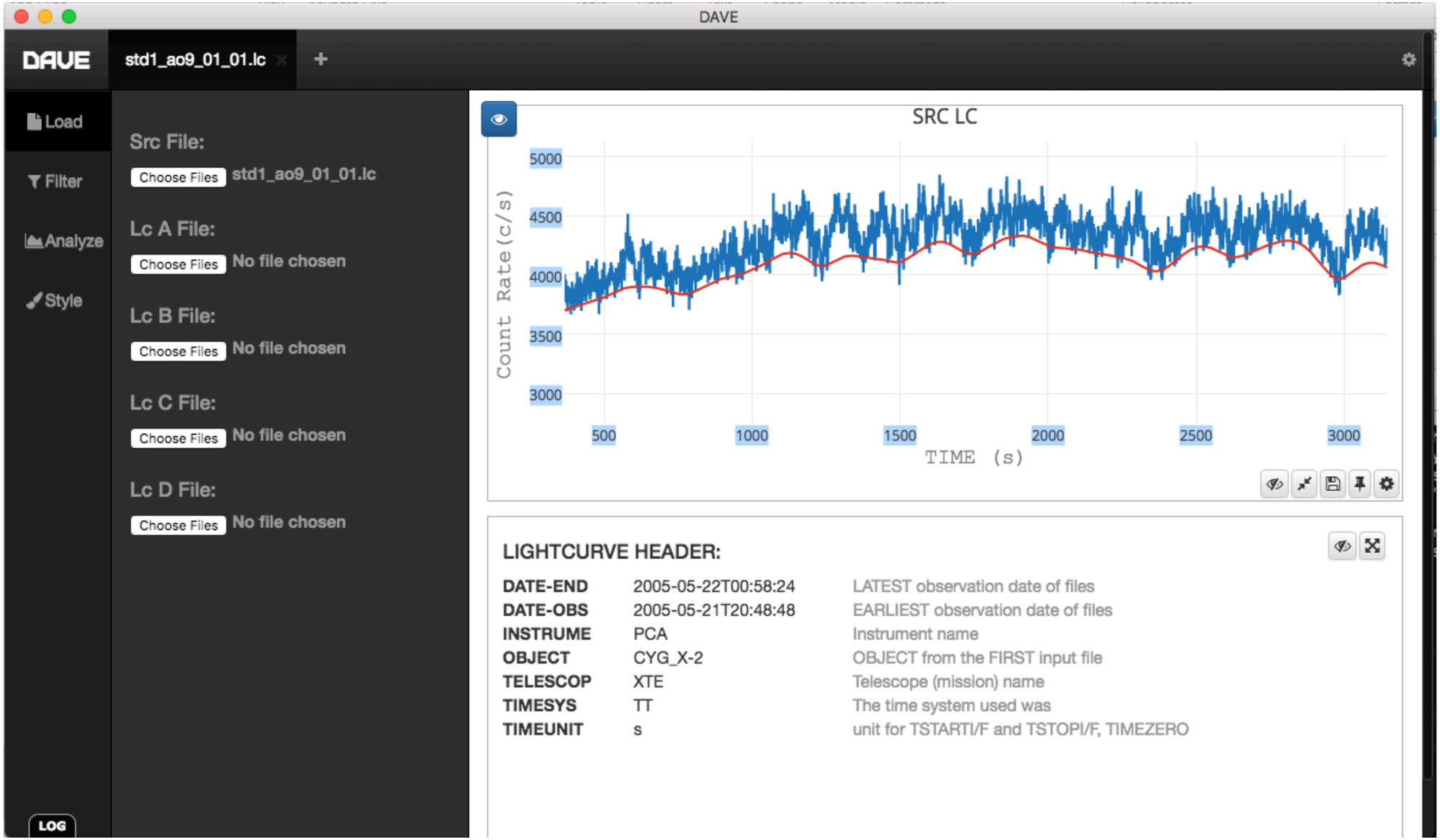
Open and fit in XSpec!

```
$ xspec
XSPEC> data cpds.pha
XSPEC> cpd /xw; setp ener; setp comm log y
XSPEC> mo lore + lore + lore
(...)
XSPEC> fit
XSPEC> pl eufspe delchi
```

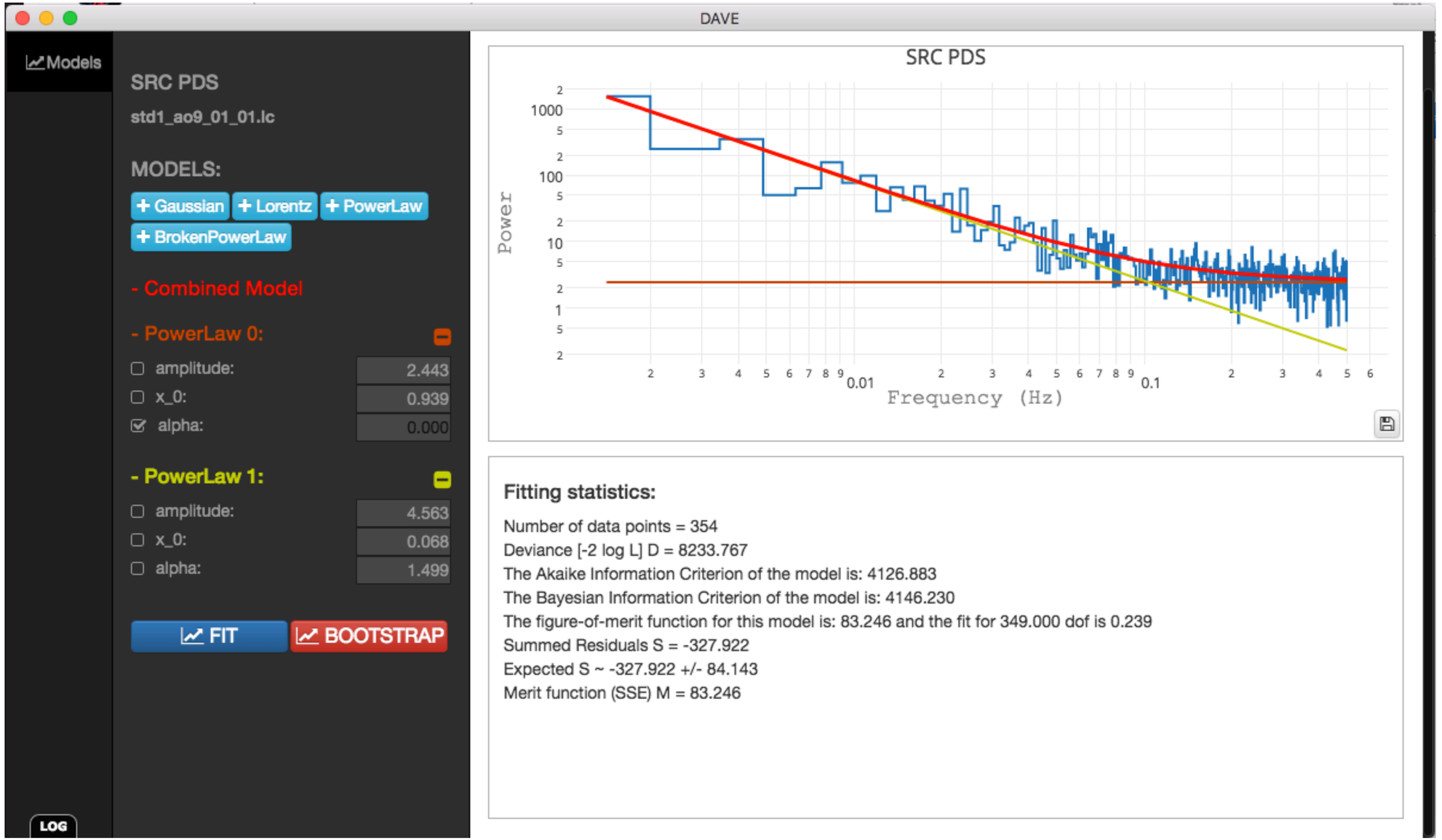
etc.



DAVE


DAVE
DAVE
std1_ao9_01_01.lc
Load
Filter
Analyze
Style
LOG
Src File:
Choose Files
std1_ao9_01_01.lc
Lc A File:
Choose Files
No file chosen
Lc B File:
Choose Files
No file chosen
Lc C File:
Choose Files
No file chosen
Lc D File:
Choose Files
No file chosen
SRC LC
Count Rate(c/s)
5000
4500
4000
3500
3000
500
1000
1500
2000
2500
3000
TIME (s)
LIGHTCURVE HEADER:
DATE-END 2005-05-22T00:58:24 LATEST observation date of files
DATE-OBS 2005-05-21T20:48:48 EARLIEST observation date of files
INSTRUME PCA Instrument name
OBJECT CYG_X-2 OBJECT from the FIRST input file
TELESCOP XTE Telescope (mission) name
TIMESYS TT The time system used was
TIMEUNIT s unit for TSTARTI/F and TSTOPI/F, TIMEZERO

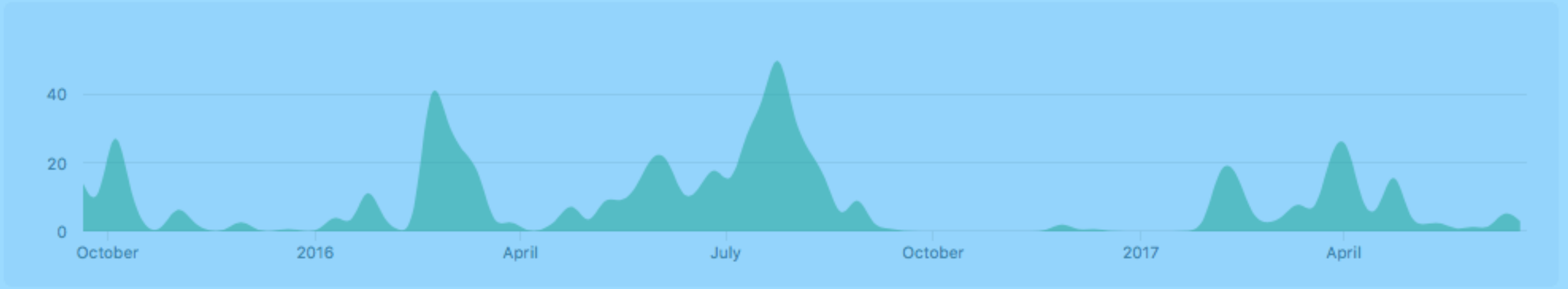
DAVE


DAVE
Models
SRC PDS
std1_ao9_01_01.lc
MODELS:
+ Gaussian
+ Lorentz
+ PowerLaw
+ BrokenPowerLaw
- Combined Model
- PowerLaw 0:
amplitude: 2.443
x_0: 0.939
alpha: 0.000
- PowerLaw 1:
amplitude: 4.563
x_0: 0.068
alpha: 1.499
FIT
BOOTSTRAP
LOG
SRC PDS
Power
Frequency (Hz)
0.01
0.1
Fitting statistics:
Number of data points = 354
Deviance [-2 log L] D = 8233.767
The Akaike Information Criterion of the model is: 4126.883
The Bayesian Information Criterion of the model is: 4146.230
The figure-of-merit function for this model is: 83.246 and the fit for 349.000 dof is 0.239
Summed Residuals S = -327.922
Expected S ~ -327.922 +/- 84.143
Merit function (SSE) M = 83.246

Google Summer of Code

- 2016 (as Timelab)
  Himanshu Misra
  Usman Khan
  Danish Sodhi

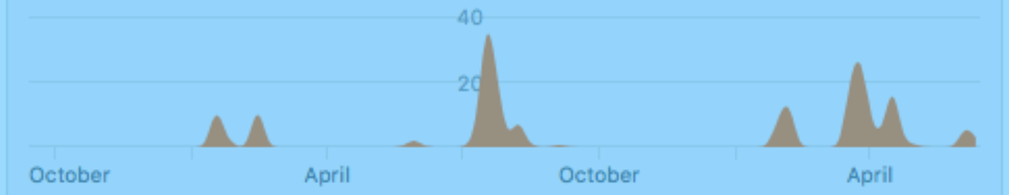
- 2017 (Python Software Foundation)
  Omar Hammad
  Haroon Rashid



matteobachetti

#1

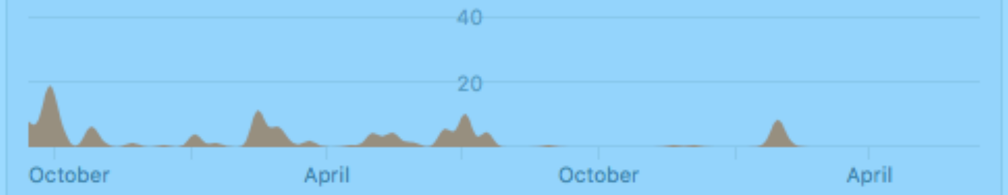
244 commits / 6,239 ++ / 1,824 --



dhuppenkothen

#2

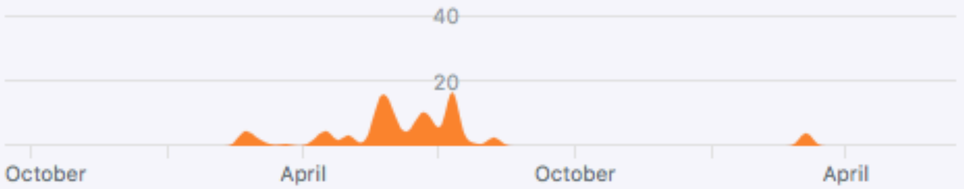
170 commits / 17,252 ++ / 9,263 --



usmanwardag

#3

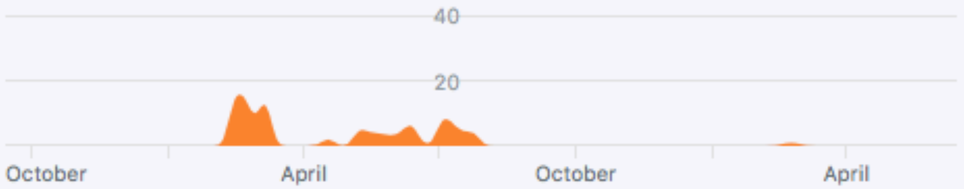
128 commits / 28,831 ++ / 3,716 --



OrkoHunter

#4

118 commits / 5,927 ++ / 2,458 --





- **Stingray** + **DAVE** + “**ex-MaLTPyNT**”: Spectral timing, for everyone:
  1. Python **API**: for the brave
  2. **GUI**: shallow learning curve
  2. **Shell Scripts**: batch processing

- Open source, **BSD** and **Apache 2**

- In development! Help us meet the first milestone (0.1, in August)!