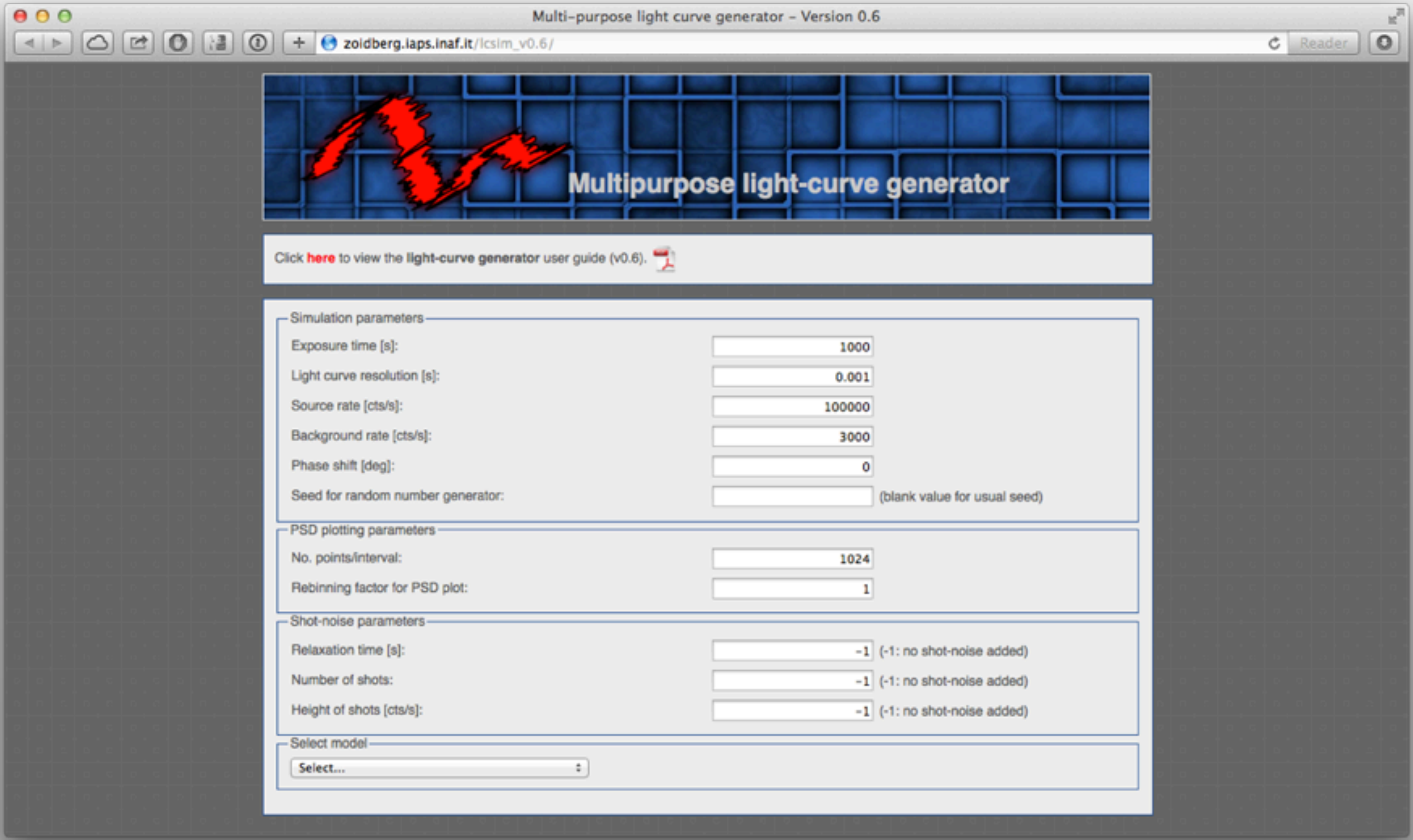
pyLCSIM
A Python package for the simulation of X-ray lightcurves

Riccardo Campana
INAF/IASF-Bologna

CNOC IX - Monte Porzio Catone - 25/09/2015

Once upon a time...

In 2010, before the original M3 LOFT proposal submission,
at IASF-Roma (RC, Imma Donnarumma, Yuri Evangelista)
we developed an online X-ray lightcurve simulator



Once upon a time...

The simulator was useful for the original LOFT/M3 proposal submission, and sometimes thereafter. Its main features were:

1. Web form with backend written in IDL
2. Possibility to simulate coherent signals (sum of sinusoids or harmonics)
3. Simulate a lightcurve from simple PSD models: powerlaw, powerlaw with 1, 2 or 3 QPOs
4. FITS output

But...

☹ Poorly documented codebase

☹ Difficult to extend to different models and/or requirements

☹ Licensing problems (*IDL is not free!*)

The solution?

☺ **Refactory the code as a pure *python* module!**

Why Python?

Python has grown in the last years becoming one of the most powerful, general purpose programming languages, also for science.

- ✓ Hundreds of extension packages
  - ➡ Numerical computation (numpy)
  - ➡ Scientific libraries (scipy)
  - ➡ Astronomy utilities (astropy)
  - ➡ Data analysis frameworks (pandas)
  - ➡ Powerful plotting libraries (matplotlib)
  - ➡ http://xkcd.com/353/
- ✓ Easy to integrate with existing routines (C/C++, Fortran, R, MATLAB interfaces)
- ✓ Support different programming styles (imperative, OOP, functional...)
- ✓ **Free!** (and multiplatform)

pyLCSIM

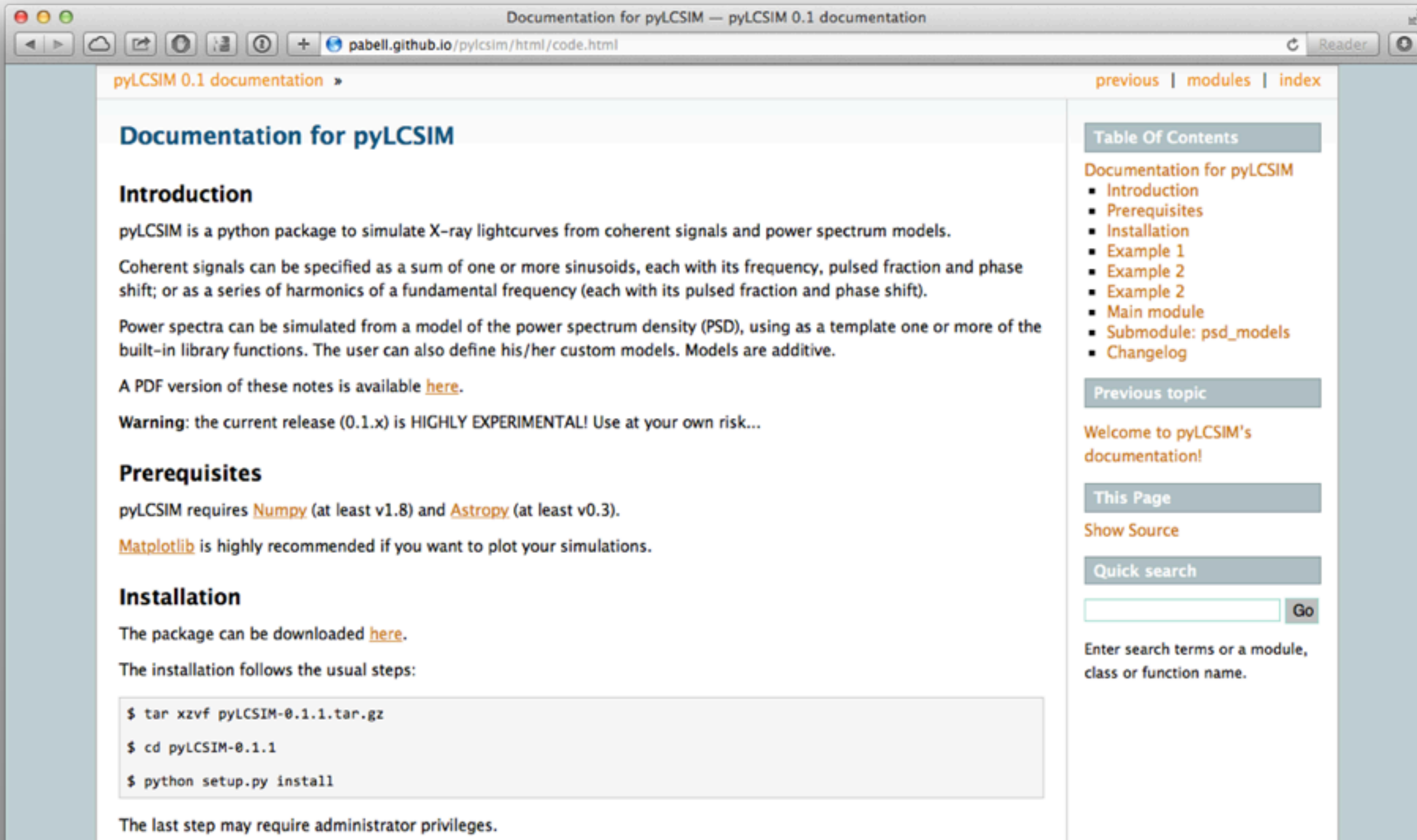
Features

- ✓ Object-oriented approach to light curve simulation
  - ➡ From a library or user-defined PSD model, lightcurve generated using the Timmer-König 1995 algorithm (with the possibility to use an arbitrary number of additive models)
  - ➡ From a coherent signal (sum of sinusoids or harmonics)
- ✓ Modular design, easily extendable to other models
- ✓ Easy of integration with existing analysis/simulation scripts
- ✓ FITS output for lightcurve and power spectrum

pyLCSIM

The latest release (0.2.2) can be downloaded from

http://pabell.github.io/pylcsim



Documentation for pyLCSIM — pyLCSIM 0.1 documentation

pabell.github.io/pylcsim/html/code.html

pyLCSIM 0.1 documentation »

previous | modules | index

Documentation for pyLCSIM

Introduction

pyLCSIM is a python package to simulate X-ray lightcurves from coherent signals and power spectrum models.

Coherent signals can be specified as a sum of one or more sinusoids, each with its frequency, pulsed fraction and phase shift; or as a series of harmonics of a fundamental frequency (each with its pulsed fraction and phase shift).

Power spectra can be simulated from a model of the power spectrum density (PSD), using as a template one or more of the built-in library functions. The user can also define his/her custom models. Models are additive.

A PDF version of these notes is available here.

Warning: the current release (0.1.x) is HIGHLY EXPERIMENTAL! Use at your own risk...

Prerequisites

pyLCSIM requires Numpy (at least v1.8) and Astropy (at least v0.3).

Matplotlib is highly recommended if you want to plot your simulations.

Installation

The package can be downloaded here.

The installation follows the usual steps:

```
$ tar xzvf pyLCSIM-0.1.1.tar.gz

$ cd pyLCSIM-0.1.1

$ python setup.py install
```

The last step may require administrator privileges.

Table Of Contents

Previous topic

Welcome to pyLCSIM's documentation!

This Page

Show Source

Quick search

Go

Enter search terms or a module, class or function name.

pyLCSIM

The latest release (0.2.2) can be downloaded from

http://pabell.github.io/pylcsim

```
$ tar xzvf pyLCSIM-0.x.y.tar.gz
$ cd pyLCSIM-0.x.y
$ python setup.py install
```

...or installed/upgraded using the Python Package Index

```
$ pip install pyLCSIM --upgrade
```

(the latter is the preferred and simpler installation method, since it automatically solves any required dependency)

Example 1

Simulation of a lightcurve from a library PSD model

```
import matplotlib.pyplot as plt
import numpy as np
import pyLCSIM


rate_src     = 30000.0 # Rate of source (cts/s)
rate_bkg     = 5000.0  # Rate of background (cts/s)
t_exp        = 50.0    # Exposure time in seconds
dt           = 0.01    # Time resolution in seconds


nbins = t_exp/dt
```

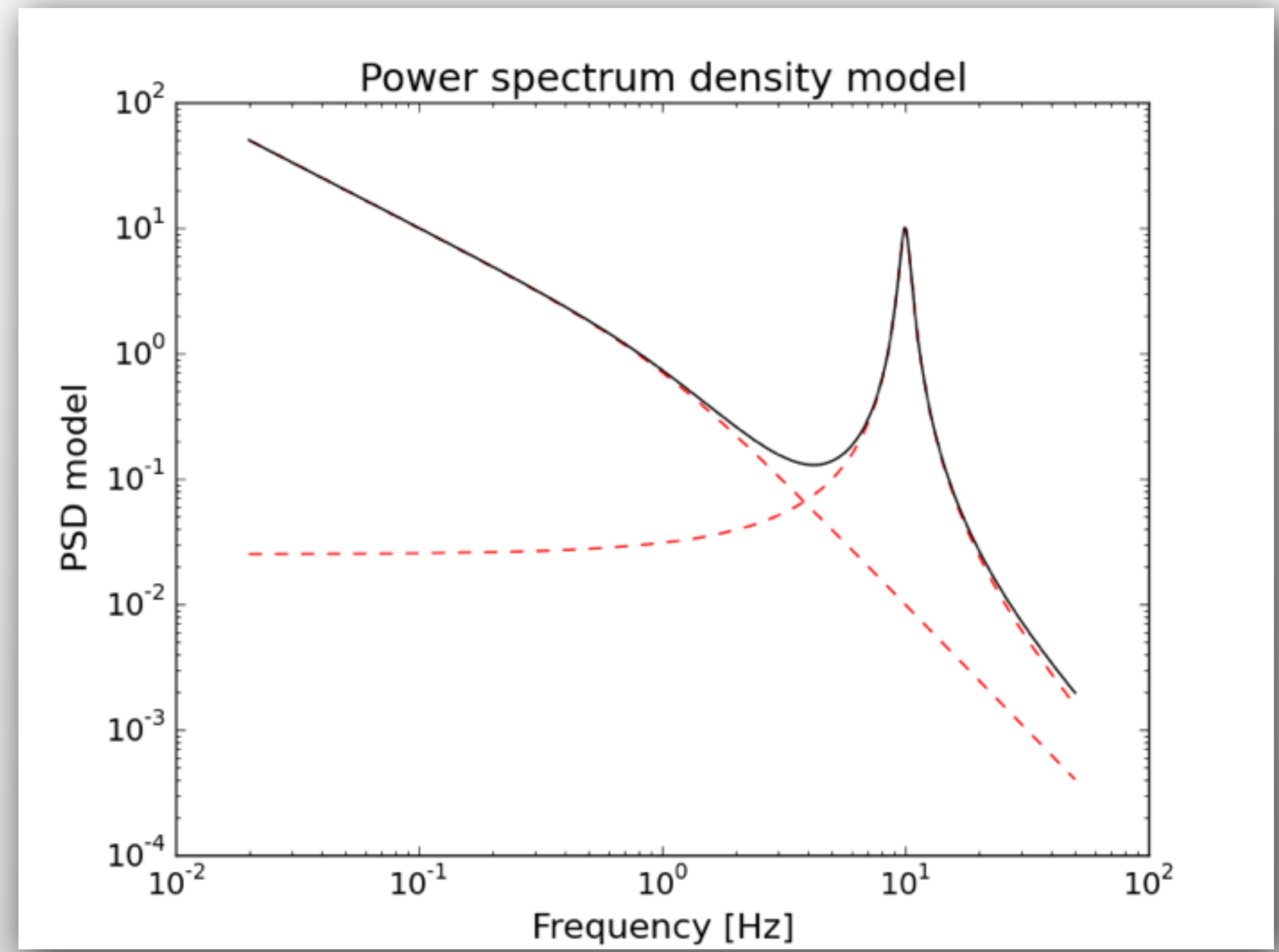
Example 1

Simulation of a lightcurve from a library PSD model

```
# Instantiate a simulation object
sim = pyLCSIM.Simulation()

# Add two PSD models: a smooth broken power law
# and a Lorentzian representing a QPO.
# See the documentation for details.
sim.addModel('smoothbknpo', [1., 1, 2, 1])
sim.addModel('lorentzian', [10., 1., 10, 2])

# Run the simulation
sim.run(dt, nbins, rate_src, rms=frms)

# Add Poisson noise to the light curve
sim.poissonRandomize(dt, rate_bkg)

# Get lightcurve and power spectrum as 1-D arrays
time, rate = sim.getLightCurve()
f, psd = sim.getPowerSpectrum()
```

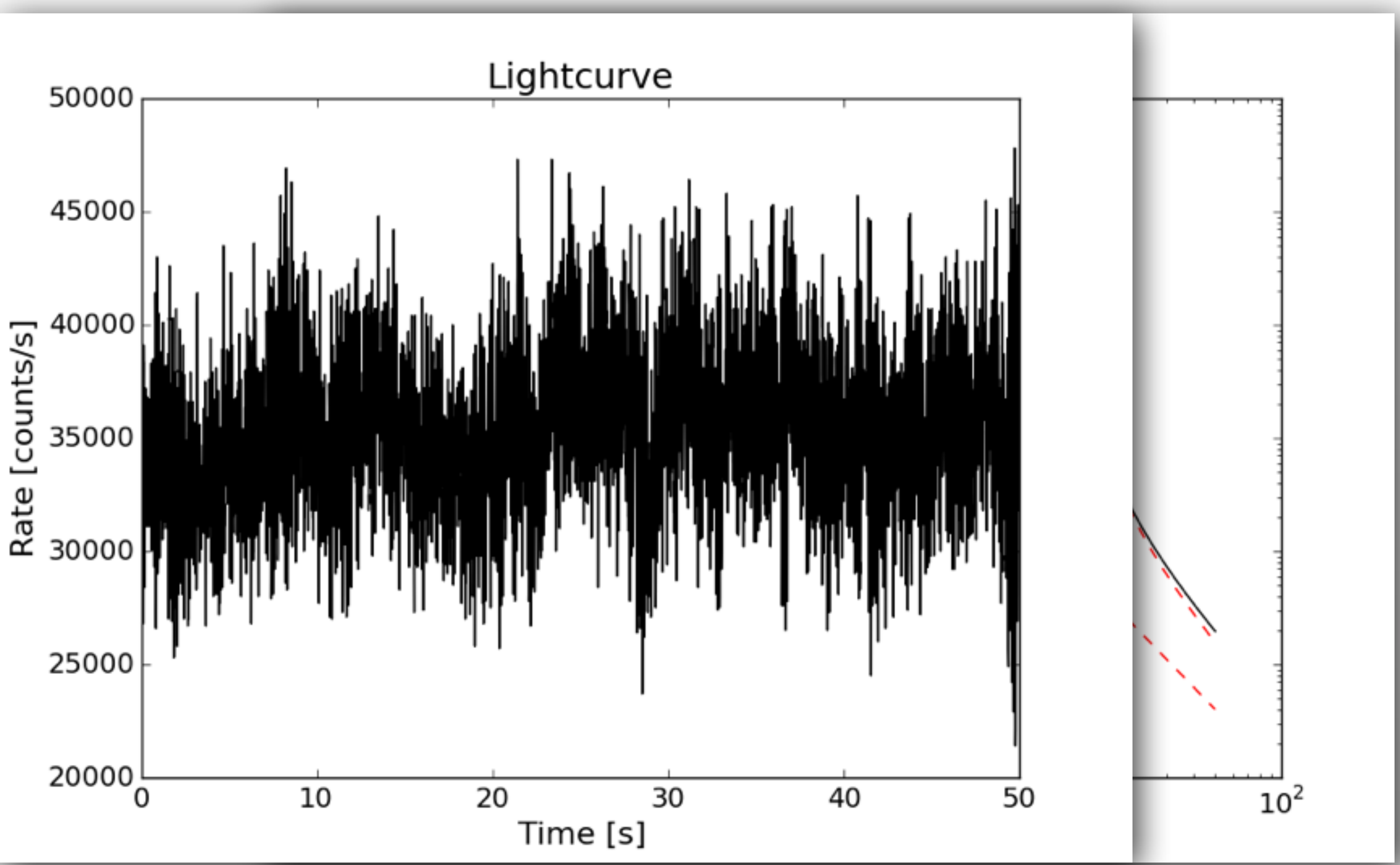
Example 1

Simulation of a lightcurve from a library PSD model



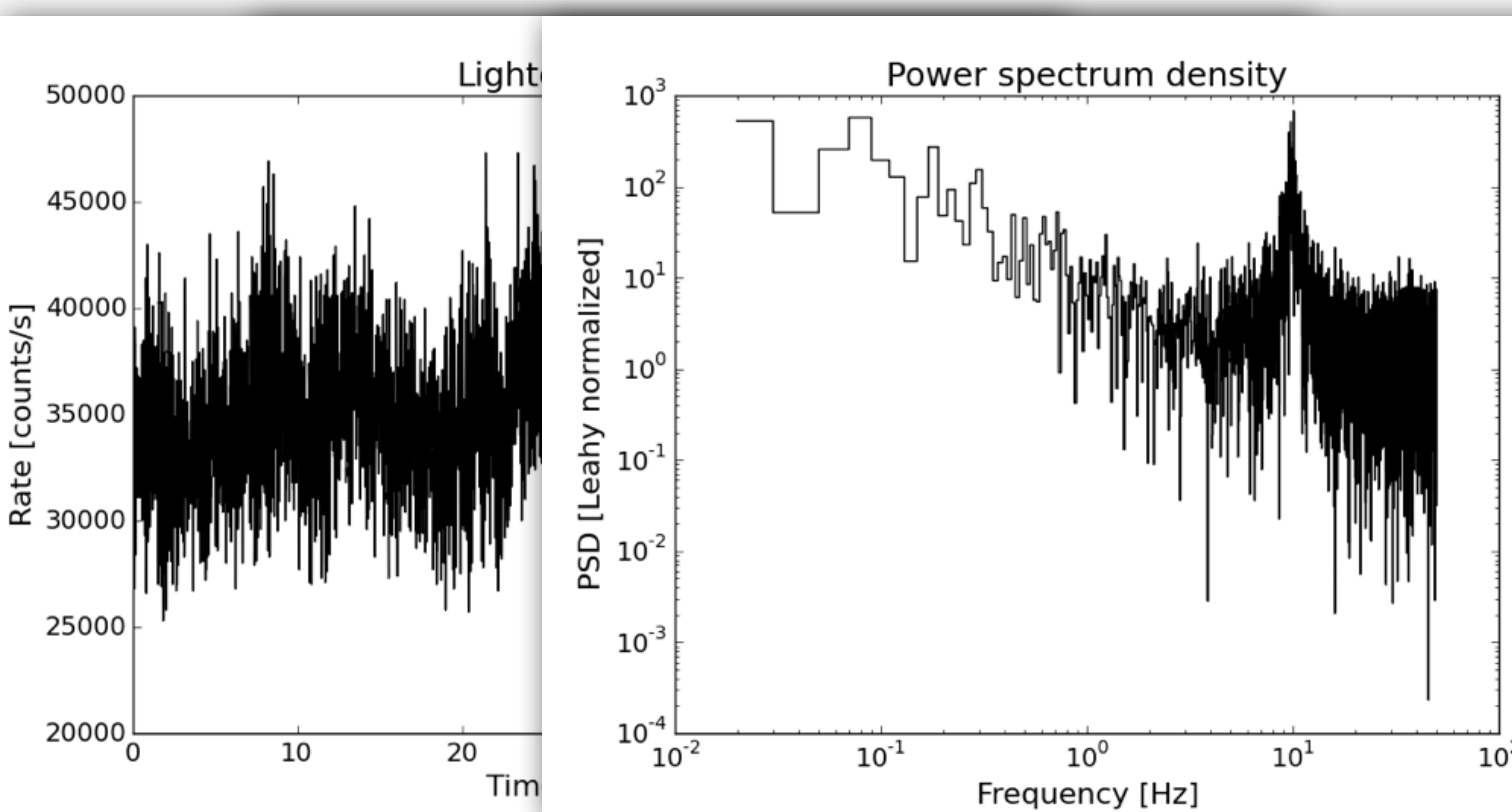
Example 1

Simulation of a lightcurve from a library PSD model



Example 1

Simulation of a lightcurve from a library PSD model



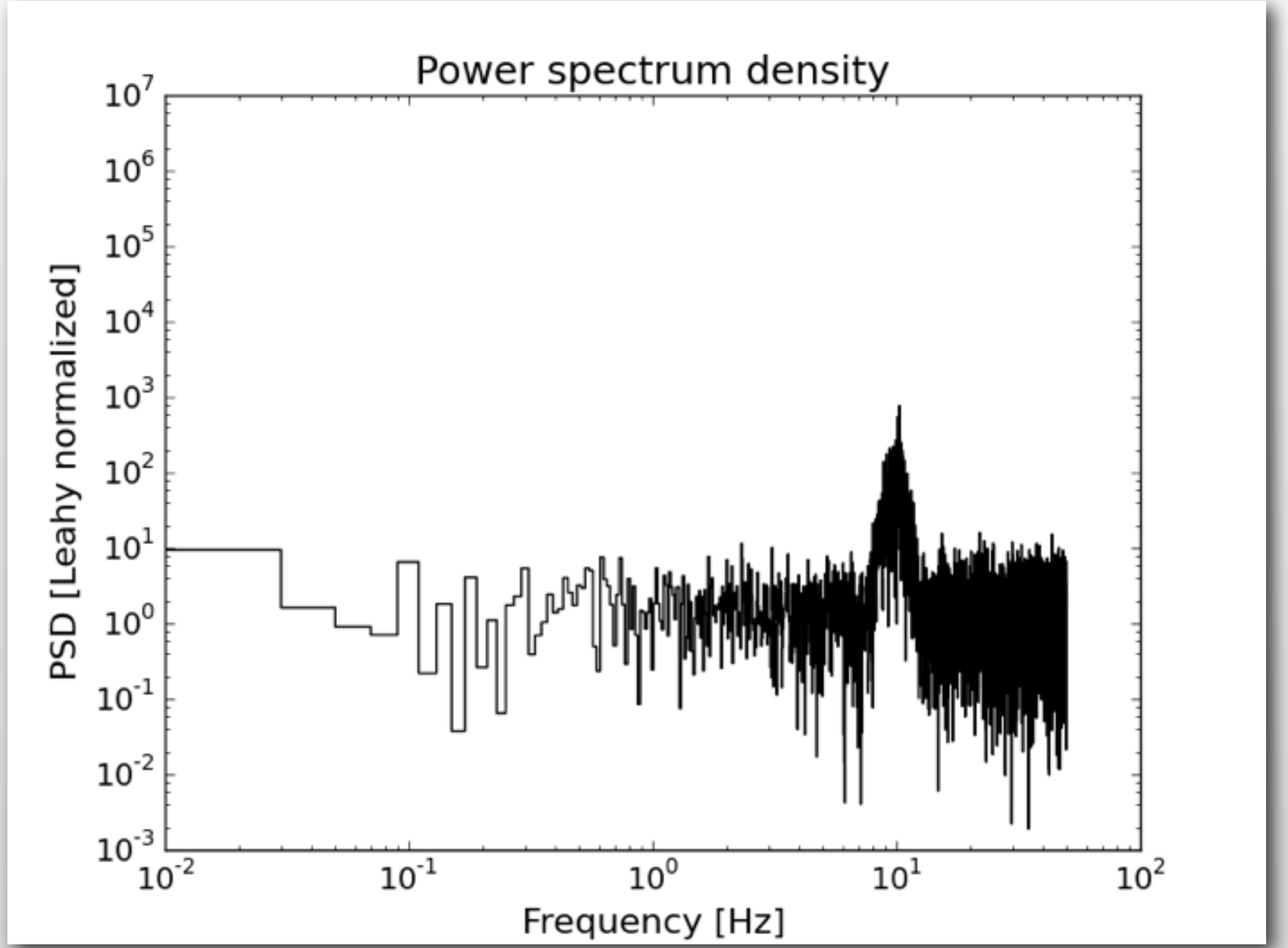
Example 2

Simulation of a lightcurve from an user-defined PSD model

```
def myFunc(f, p):
    """
    Example of user-defined function: a Gaussian.
    User-defined PSD models should be positive-valued!
    Moreover, in this example the output is clipped at 1e-32
    to avoid too small values.
    """
    f = p[0]*np.exp(-(f-p[1])**2/p[2]**2)
    return np.clip(f, 1e-32, np.max(f))

sim = pyLCSIM.Simulation()
sim.addModel('smoothbknpo', [1., 1, 2, 1])
sim.addModel(myFunc, [1000., 10, 1.])  ←

# Run the simulation
sim.run(dt, nbins, rate_src, rms=frms)
```

Example 2

Simulation of a lightcurve from an user-defined PSD model



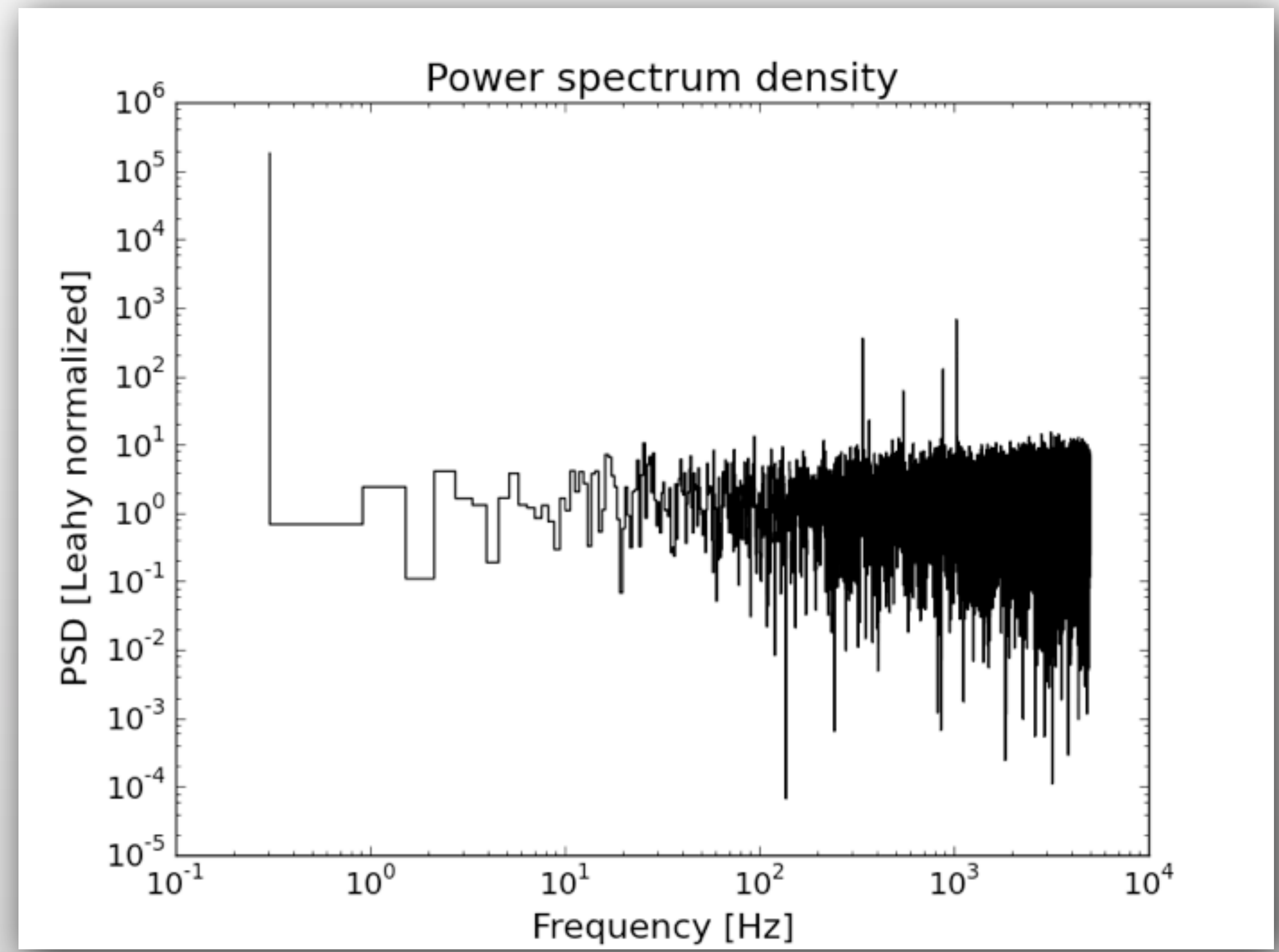
Example 3

Coherent signal as a sum of sinusoids

```
# Instantiate a simulation object, this time as coherent
sim = pyLCSIM.Simulation(kind='coherent')

# Run the simulation, using:
# four sinusoidal frequencies: 340, 550, 883, 1032 Hz;
# with pulsed fractions 10%, 5%, 7% and 15% respectively;
# the third frequency has a 35 degree phase shift
# with respect to the others
sim.run(dt, nbins, rate_src, freq=[340, 550, 883, 1032], \
                          amp=[0.1, 0.05, 0.07, 0.15],\
                          phi=[0., 0, 35., 0.])

# Add Poisson noise to the light curve
sim.poissonRandomize(dt, rate_bkg)

# Get lightcurve and power spectrum as 1-D arrays
time, rate = sim.getLightCurve()
f, psd = sim.getPowerSpectrum()
```

Example 2

Coherent signal as a sum of sinusoids



Example 4

Coherent signal as a sum of harmonics

```
# Instantiate a simulation object, this time as coherent
sim = pyLCSIM.Simulation(kind='coherent')


# Run the simulation:
# Fundamental at 500 Hz, 3 harmonics (500, 1000, 1500 Hz)
# with pulsed fractions 10%, 5% and 15% respectively
sim.run(dt, nbins, rate_src, freq=500, nha=3, amp=[0.1, 0.05, 0.15])

# Add Poisson noise to the light curve
sim.poissonRandomize(dt, rate_bkg)

# Get lightcurve and power spectrum as 1-D arrays
time, rate = sim.getLightCurve()
f, psd = sim.getPowerSpectrum()
```

Future prospects

Features that will (or can) be added in future versions

1. Multiplicative PSD models
2. More refined simulation algorithms (e.g. Emmanoulopoulos et al. 2013)
3. Add more examples and documentation
4. Add a few analysis utilities (e.g. for the PSD)

Conclusions

pyLCSIM is at a **very early** stage of development!

(i.e. use at your own risk)

If you are interested to:

1. Help beta-testing
2. Suggest new features
3. Collaborate to the code

...check it out!

http://pabell.github.io/pylcsim

And contact me at:

campana@iasfbo.inaf.it