scRRpy

Release 0.1.0

Feb 26, 2019

Contents

CHAPTER 1

scRRpy

Scalar Resonant Relaxation around a massive black hole

- Free software: BSD license

1.1 Installation

1.1.1 PYTHON VERSIONS AND DEPENDENCIES

`scRRpy` supports both Python 3.5, and 3.6

This package requires:

- Numpy
- Scipy
- Matplotlib
- astropy
- numba
- vegas

```
python setup.py install
```

1.2 Documentation

https://scrrpy.readthedocs.io/

CHAPTER 2

Installation

At the command line:

```
pip install scrrpy
```

CHAPTER 3

Usage

To use scRRpy in a project

3.1 Examples

Plotting the diffusion coefficient for $a = 0.1$ pc in a Milky-Way like galactic center with Bahcall–Wolf cusp.

```
from scrrpy import DRR
import matplotlib.pyplot as plt


drr = DRR(0.1, gamma=1.75, mbh_mass=4.3e6, rh=2.0, star_mass=1.0, j_grid_size=32)

djj, djj_err = drr(l_max=5)

plt.loglog(drr.j, djj)
plt.xlabel(r'$J/J_\mathrm{c}$')
plt.ylabel(r'$D^{\mathrm{RR}}_{JJ}/J_\mathrm{c}^2$ [1/Myr]')

plt.show()
```

CHAPTER 4

Reference

4.1 scrrpy

class scrrpy.**DRR**(*sma*, *gamma=1.75*, *mbh_mass=4000000.0*, *star_mass=1.0*, *j_grid_size=128*, *rh=2.0*, *seed=None*)

Bases: `scrrpy.cusp.Cusp`

Resonant relaxation diffusion coefficient (DRR). Assuming a power law stellar cusp around a massive black hole (MBH). The cusp is assumed to have an isotropic distribution function $f(E) \propto |E|^p$ corresponding ro a stellar density $n(r) \propto r^{-\gamma}$ where $\gamma = \frac{3}{2} + p$

Parameters

- **sma** (*float*) – The semi-mahor axis along which DRR will be computed
- **gamma** (*float, int, optional*) – The slope of the density profile. Default: 7/4 (Bahcall wolf cusp)
- **mbh_mass** (*float, int, optional*) – Mass of the MBH [solar mass]. Default: 4.3×10^6 (Milky Way MBH)
- **star_mass** (*float, int, optional*) – Mass of individual stars [solar mass]. Default: 1.0
- **rh** (*float, int, optional*) – Radius of influence [pc]. Define as the radius in which the velocity dispersion of the stellar cusp σ is equal to the Keplerian velocity due to the MBH $\sigma(r_\mathrm{h})^2 = GM_\bullet/r_\mathrm{h}$. Default: 2.0

Methods Summary

`__call__`(l_max[, neval, threads, . . .])	Returns the RR diffusion coefficient D_{JJ}/J_c^2 [1/yr].
`save`(file_name)	Save the current instance to an hdf5 file.
`from_file`(file_name)	Load from file and return an instance

Methods Documentation

__call__(*l_max*, *neval=1000.0*, *threads=1*, *progress_bar=True*, *seed=None*)
Returns the RR diffusion coefficient D_{JJ}/J_c^2 [1/yr].

Parameters

- **l_max** (*int*) – Maximal order of spherical harmonics to compute
- **neval** (*int*) – The maximum number of integrand evaluations in each iteration of the *vegas* algorithm. Default: 1000
- **threads** (*int*) – Number of parallel threads to use. Default: 1 (no parallelization)
- **progress_bar** (*bool*) – Show progress bar. Default: `True`

save(*file_name*)
Save the current instance to an hdf5 file.

Example

```
>>> drr = DRR(0.1, j_grid_size=32)
>>> d, d_err = drr(l_max=3)
>>> drr.save('example.hdf5')
>>> drr = DRR.from_file('example.hdf5')
>>> d, d_err = drr(l_max=drr.l_max, neval=drr.neval)
```

classmethod from_file(*file_name*)
Load from file and return an instance

Example

```
>>> drr = DRR(0.1, j_grid_size=32)
>>> d, d_err = drr(l_max=3)
>>> drr.save('example.hdf5')
>>> drr = DRR.from_file('example.hdf5')
>>> d, d_err = drr(l_max=drr.l_max, neval=drr.neval)
```

class scrrpy.**Cusp**(*gamma=1.75*, *mbh_mass=4000000.0*, *star_mass=1.0*, *rh=2.0*)
A power law stellar cusp around a massive black hole (MBH). The cusp is assumed to have an isotropic distribution function $f(E) \propto |E|^p$ corresponding ro a stellar density $n(r) \propto r^{-\gamma}$ where $\gamma = \frac{3}{2} + p$

TODO - Implement normalization Total mass at r_h

TODO - Implement normalization $N(a)$ vs $N(r)$

Parameters

- **gamma** (*float, int, optional*) – The slope of the density profile. Default: 7/4 (Bahcall-Wolf cusp)
- **mbh_mass** (*float, int*) – Mass of the MBH [solar mass]. Default: 4.3×10^6 (Milky Way MBH)
- **star_mass** (*float, int*) – Mass of individual stars [solar mass]. Default: 1.0
- **rh** (*float, int*) – Radius of influence [pc]. Define as the radius in which the velocity dispersion of the stellar cusp σ is equal to the Keplerian velocity due to the MBH $\sigma(r_\mathrm{h})^2 = GM_\bullet/r_\mathrm{h}$. Default: 2.0

a_gr1
The sma below which $\nu_{\rm p}$ is only positive, that is $\nu_{\rm p}(a, j=1) = 0$

d_nu_p (a, j)
The derivative of $\nu_{\rm p}$ with respect to j, defined to be positive

inverse_cumulative_a (x)
The inverse of $N(a)$. Useful to generate a random sample of semi-major axis.

Parameters **x** (*float, array*) – x in [0, 1]

Example

```
>>> cusp = Cusp(gamma=1.75)
>>> np.random.seed(1234)
>>> sma = cusp.inverse_cumulative_a(np.random.rand(100))
>>> print("{:0.10}, {:0.10}, {:0.10}".format(sma.min(), sma.mean(), sma.
↪max()))
0.03430996478, 1.147418232, 1.987320281
```

jlc (a)
Relativistic loss cone

Minimal normalized angular momentum on which orbits are stable.

$j_{\rm lc} = J_{\rm lc}/J_{\rm c}$, where $J_{\rm lc} = 4GM_\bullet/c$ is the last stable orbit in the parabolic limit and $J_{\rm c} = \sqrt{GM_\bullet a}$ is the maximal (circular) stable orbit.

This is an approximation which works when the orbital binding energy E is much smaller than rest energy of the MBH Mc^2.

Parameters **a** (*float, array*) – Semi-major axis [pc].

mass_ratio
MBH to star mass ratio

nu_gr (a, j)
Precession frequency [rad/year] due to general relativity (first PN term)

Parameters

- **a** (*float, array*) – Semi-major axis [pc].
- **j** (*float, array*) – Normalized angular momentum $j = J/J_{\rm c} = \sqrt{1-e^2}$.

nu_mass (a, j)
Precession frequency [rad/year] due to stellar mass.

Parameters

- **a** (*float, array*) – Semi-major axis [pc].
- **j** (*float, array*) – Normalized angular momentum $j = J/J_{\rm c} = \sqrt{1-e^2}$.

nu_p (a, j)
Precession frequency [rad/year]

$\nu_{\rm p}(a, j) = \nu_{\rm gr}(a, j) + \nu_{\rm mass}(a, j)$

Parameters

- **a** (*float, array*) – Semi-major axis [pc].
- **j** (*float, array*) – Normalized angular momentum $j = J/J_{\rm c} = \sqrt{1-e^2}$.

nu_p1 (a)
Precession frequency at $j = 1$

nu_r (a)
The orbital frequency in rad/year at a [pc]

Parameters **a** (*float, array*) – Semi-major axis [pc].

number_of_stars (a)
Number of stars with semi-major axis smaller than a [pc]

Parameters **a** (*float, array*) – Semi-major axis [pc].

period (a)
The orbital period in years at a [pc]

Parameters **a** (*float, array*) – Semi-major axis [pc].

rg
Gravitational radius of the MBH [pc]

stellar_mass (a)
Enclosed mass within $r = a$ [pc].

TODO - check $M(r)$ vs $M(a)$

Parameters **a** (*float, array*) – Semi-major axis [pc].

tg
Light crossing time of the MBH [sec]

total_number_of_stars
Number of stars within the radius of influence $r_{\rm h}$

total_stellar_mass
Total mass within the radius of influence $r_{\rm h}$ [solar mass]

TODO - Implement normalization

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

5.1 Bug reports

When reporting a bug please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.2 Documentation improvements

scRRpy could always use more documentation, whether as part of the official scRRpy docs, in docstrings, or even on the web in blog posts, articles, and such.

5.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/benbaror/scrrpy/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

5.4 Development

To set up *scrrpy* for local development:

1. Fork scrrpy (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/scrrpy.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. When you're done making changes, run all the checks, doc builder and spell checker with tox one command:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

5.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

5.4.2 Tips

To run a subset of tests:

```
tox -e envname -- py.test -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.

It will be slower though ...

CHAPTER 6

Authors

- Ben Bar-Or - benbaror@ias.edu
- Jean-Baptiste Fouvry - fouvry@ias.edu

CHAPTER 7

Changelog

7.1 0.1.0 (2017-10-15)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Index