mrspoc Documentation

Release 0.1.dev52

Brett Morris

May 22, 2018

Contents:

mrspoc is a Python toolkit for estimating stellar centroid jitter due to magnetic activity cycles for Gaia target stars.

Part I

Getting started

CHAPTER 1

Contents

1.1 Creating a spotted star

Suppose you'd like to estimate the stellar centroid jitter for a star with a spot. We begin by creating a `Star` object:

```
from mrspoc import Star
star = Star(u1=0.5, u2=0.2)
```

We've set the quadratic limb darkening parameters `u1, u2` to be similar to the Sun in the optical.

Now let's add a `Spot` to the spot list attribute `Star.spots`, which is placed half a stellar radius in the positive x-direction, with radius 10% of the radius of the star:

```
from mrspoc import Spot
spot = Spot(x=0.5, y=0, r=0.1, contrast=0.7)
star.spots.append(spot)
```

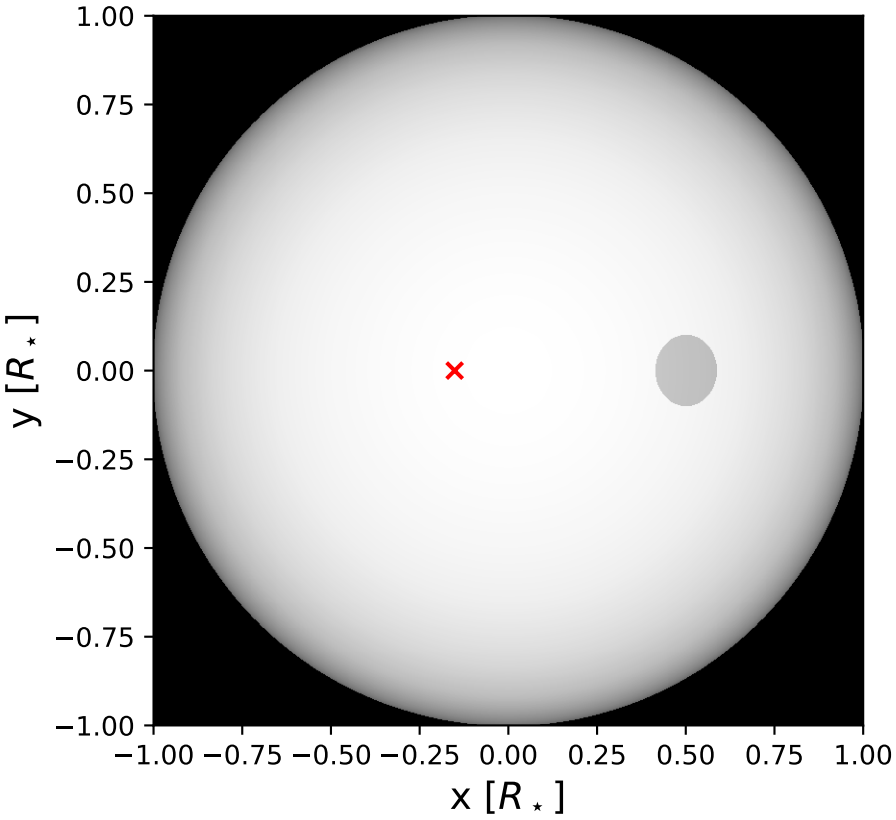
The spot contrast, set to 70% here, should be interpreted as the intensity of the atmosphere in the spot as a fraction of the flux in the quiescent photosphere.

We can print the apparent stellar centroid using the `~mrspoc.Star.center_of_light` attribute:

```
>>> star.center_of_light
(-0.0013829556756940378, 0.0)
```

The centroid is in the negative x direction since the spot is in the positive x direction. We can see what this star and spot configuration look like with the `plot` function:

```
star.plot(col_exaggerate=100)
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
y [$R_\star$]
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
x [$R_\star$]

We've used the `col_exaggerate` keyword argument to exaggerate the centroid offset by a factor of 100, so we can see it.

1.2 Gaia

`mrspoc` has a few handy functions for computing Gaia's expected astrometric precision, using the relations from Perryman et al. 2014.

You can predict the number of times Gaia will observe a given star with `Nprime_fov`, for a star at galactic latitude b:

```
>>> from mrspoc import Nprime_fov
>>> import astropy.units as u
>>> import numpy as np

>>> b = np.arange(0, 90, 10) * u.deg
>>> print(Nprime_fov(b))
[  51.9   53.7   58.5   69.1  107.4   85.    71.    65.2   62.8]
```

The results are non-integers because they are the mean number of visits for stars near each galactic latitude.

You can compute the galactic latitude b for a target given its `SkyCoord` like this:

```
>>> from astropy.coordinates import SkyCoord, Galactic
>>> import astropy.units as u
>>> from mrspoc import Nprime_fov

>>> coord = SkyCoord(ra=30*u.deg, dec=80*u.deg, frame='icrs')
>>> coord_gal = coord.transform_to(Galactic)
>>> print(coord_gal.b)
17d32m24.9039s
>>> print(Nprime_fov(coord_gal.b))
55.6
```

You can compute the expected astrometric precision on a given target as a function of its Gaia bandpass `G` magnitude with `sigma_fov`, again taking from Perryman et al. 2014, this time from Equations 1-3:

```
>>> from mrspoc import sigma_fov
>>> sigma_fov(6.5)
<Quantity 34.2301167881504 uarcsec>

>>> sigma_fov(15)
<Quantity 81.99593485858512 uarcsec>
```

Part II

Reference/API

CHAPTER 2

mrspoc Package

2.1 Functions

Nprime_fov(b)	Average number of field of view passages per star including dead time overhead for a target at galactic latitude b [deg].
draw_random_sunspot_latitudes(n[, mean_latitude])	Draw one or more random samples from the sunspot latitude distribution.
draw_random_sunspot_radii(n)	Draw one or more random samples from the sunspot radius distribution.
get_table_ms([plot, ax])	Open the TGAS catalog for all stars brighter than G < 12, make CMD cuts to flag just the main sequence stars.
sigma_fov(Gmag)	Approximate Gaia astrometric uncertainty in a single measurement, i.e.
test([package, test_path, args, plugins, ...])	Run the tests using py.test.

2.1.1 Nprime_fov

mrspoc.**Nprime_fov**(*b*)

Average number of field of view passages per star including dead time overhead for a target at galactic latitude b [deg]. Taken from Perryman et al. 2014 Table 1.

Parameters

b : {ndarray, list, float}

Array of galactic latitudes

Returns

N_fovs : ndarray

Approximate number of times that Gaia will observe a target at galactic latitude b, after including dead time. This is the <Nprime_fov> column from Perrman et al. 2014 Table 1.

2.1.2 draw_random_sunspot_latitudes

mrspoc.**draw_random_sunspot_latitudes**(*n*, *mean_latitude=15*)

Draw one or more random samples from the sunspot latitude distribution.

Parameters

n : int

Number of random sunspot latitudes to draw

Returns

lat : `Quantity`

Latitude of a sunspot drawn from the sunspot latitude distribution.

2.1.3 draw_random_sunspot_radii

mrspoc.**draw_random_sunspot_radii**(*n*)

Draw one or more random samples from the sunspot radius distribution.

Parameters

n : int

Number of random sunspot radii to draw

Returns

rspot_rstar : `ndarray`

Radii of a sunspots drawn from the sunspot radius distribution, in units of stellar radii.

2.1.4 get_table_ms

mrspoc.**get_table_ms**(*plot=True*, *ax=None*)

Open the TGAS catalog for all stars brighter than G < 12, make CMD cuts to flag just the main sequence stars.

Parameters

plot : bool (optional)

Make a plot of the CMD and color/mag cuts.

ax : `Axes` (optional)

If `ax` is not `None`, make the plot on `ax`.

Returns

table : `Table`

Table of TGAS sources, magnitudes, parallaxes, distances, and anticipated astrometric uncertainties.

ms : `ndarray`

Boolean array, `True` for rows of `table` where the star is a main sequence star within the color/mag cuts.

2.1.5 sigma_fov

mrspoc.**sigma_fov**(*Gmag*)

Approximate Gaia astrometric uncertainty in a single measurement, i.e. "the along-scan accuracy per field of view crossing", after Perryman et al. 2014, Eqn. 1 - 3.

Parameters

Gmag : float or ndarray

Gaia G band magnitude.

Returns

sigma : Quantity

Approximate astrometric uncertainty for a target of magnitude Gmag, in units of arcseconds.

2.1.6 test

mrspoc.**test**(*package=None*, *test_path=None*, *args=None*, *plugins=None*, *verbose=False*, *pastebin=None*, *remote_data=False*, *pep8=False*, *pdb=False*, *coverage=False*, *open_files=False*, ***kwargs*)

Run the tests using py.test. A proper set of arguments is constructed and passed to pytest.main.

Parameters

package : str, optional

The name of a specific package to test, e.g. 'io.fits' or 'utils'. If nothing is specified all default tests are run.

test_path : str, optional

Specify location to test by path. May be a single file or directory. Must be specified absolutely or relative to the calling directory.

args : str, optional

Additional arguments to be passed to pytest.main in the `args` keyword argument.

plugins : list, optional

Plugins to be passed to pytest.main in the `plugins` keyword argument.

verbose : bool, optional

Convenience option to turn on verbose output from py.test. Passing True is the same as specifying `'-v'` in `args`.

pastebin : {'failed','all',None}, optional

Convenience option for turning on py.test pastebin output. Set to `'failed'` to upload info for failed tests, or `'all'` to upload info for all tests.

remote_data : bool, optional

Controls whether to run tests marked with @remote_data. These tests use online data and are not run by default. Set to True to run these tests.

pep8 : bool, optional

Turn on PEP8 checking via the pytest-pep8 plugin and disable normal tests. Same as specifying `'--pep8 -k pep8'` in `args`.

pdb : bool, optional

Turn on PDB post-mortem analysis for failing tests. Same as specifying `'--pdb'` in `args`.

coverage : bool, optional

Generate a test coverage report. The result will be placed in the directory htmlcov.

open_files : bool, optional

Fail when any tests leave files open. Off by default, because this adds extra run time to the test suite. Requires the psutil package.

parallel : int, optional

When provided, run the tests in parallel on the specified number of CPUs. If parallel is negative, it will use the all the cores on the machine. Requires the pytest-xdist plugin installed. Only available when using Astropy 0.3 or later.

kwargs

Any additional keywords passed into this function will be passed on to the astropy test runner. This allows use of test-related functionality implemented in later versions of astropy without explicitly updating the package template.

2.2 Classes

Spot([x, y, z, r, contrast, stellar_radius])	Properties of a starspot.
Star([spots, u1, u2, r, radius_threshold, ...])	Object defining a star.

2.2.1 Spot

class mrspoc.**Spot**(*x=None*, *y=None*, *z=None*, *r=None*, *contrast=0.7*, *stellar_radius=1*)

Bases: object

Properties of a starspot.

Parameters

x : float

X position [stellar radii]

y : float

Y position [stellar radii]

z : float

Z position [stellar radii], default is `z=sqrt(r_star^2 - x^2 - y^2)`.

r : float

Spot radius [stellar radii], default is `r=1`.

contrast : float (optional)

Spot contrast relative to photosphere. Default is `c=0.7`

stellar_radius : float

Radius of the star, in the same units as `x,y,z,r`. Default is 1.

Methods Summary

Methods Documentation

classmethod from_latlon(*latitude*, *longitude*, *stellar_inclination*, *radius*)
Construct a spot from latitude, longitude coordinates

Parameters
latitude : float

Spot latitude [deg]

longitude : float

Spot longitude [deg]

stellar_inclination : float

Stellar inclination angle, measured away from the line of sight, in [deg].

radius : float

Spot radius [stellar radii]

classmethod from_sunspot_distribution(*stellar_inclination*, *mean_latitude=15*, *contrast=0.7*, *radius_multiplier=1*)

Parameters
stellar_inclination : float

Stellar inclination angle, measured away from the line of sight, in [deg].

mean_latitude : float

Define the mean absolute latitude of the two symmetric active latitudes, where `mean_latitude > 0`.

contrast : float (optional)

Spot contrast relative to photosphere. Default is the area-weighted mean sunspot contrast (`c=0.7`).

2.2.2 Star

class mrspoc.**Star**(*spots=None*, *u1=0.4987*, *u2=0.1772*, *r=1*, *radius_threshold=0.1*, *inclination=None*)
Bases: `object`

Object defining a star.

Parameters
u1 : float (optional)

Quadratic limb-darkening parameter, linear term

u2 : float (optional)

Quadratic limb-darkening parameter, quadratic term

r : float (optional)

Stellar radius (default is unity)

radius_threshold : float (optional)

If all spots are smaller than this radius, use the analytic solution to compute the stellar centroid, otherwise use the numerical solution.

spots : list (optional)

List of spots on this star.

inclination : `Quantity`

Stellar inclination. Default is 90 deg.

Attributes Summary

`center_of_light`	Compute the center-of-light or centroid for this star, given its spots, and limb-darkening.
`inclination`	

Methods Summary

`derotate`()	
`limb_darkening`(r)	Compute the intensity at radius `r` for quadratic limb-darkening law with parameters `Star.u1, Star.u2`.
`limb_darkening_normed`(r)	Compute the normalized intensity at radius `r` for quadratic limb-darkening law with parameters `Star.u1, Star.u2`.
`plot`([n, ax, col, col_exaggerate])	Plot a 2D projected schematic of the star and its spots.
`rotate`(angle)	Rotate the star, by moving the spots.

Attributes Documentation

center_of_light

Compute the center-of-light or centroid for this star, given its spots, and limb-darkening.

Returns

x_centroid : float

Photocenter in the x dimension, in units of stellar radii

y_centroid : float

Photocenter in the y dimension, in units of stellar radii

inclination

Methods Documentation

derotate()

limb_darkening(*r*)
Compute the intensity at radius `r` for quadratic limb-darkening law with parameters `Star.u1, Star.u2`.

Parameters
r : float or `ndarray`

Stellar surface position in radial coords on (0, 1)

Returns
intensity : float

Intensity in un-normalized units

limb_darkening_normed(*r*)
Compute the normalized intensity at radius `r` for quadratic limb-darkening law with parameters `Star.u1, Star.u2`.

Parameters
r : float or `ndarray`

Stellar surface position in radial coords on (0, 1)

Returns
intensity : float

Intensity relative to the intensity at the center of the disk.

plot(*n=3000*, *ax=None*, *col=True*, *col_exaggerate=1*)
Plot a 2D projected schematic of the star and its spots.

Parameters
ax : `Axes`

Axis object to draw the plot on

col : bool (optional)

Show the center of light with a red "x" if `True`

col_exaggerate : float (optional)

Exaggerate the center-of-light coordinate by this factor

n : int

Number of pixels per side in the image.

Returns
ax : `Axes`

Matplotlib axis object, with the new plot on it.

rotate(*angle*)
Rotate the star, by moving the spots.

Parameters
angle : `Quantity`

2.3 Class Inheritance Diagram

Part III

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index