Pandora-moon Documentation

Release stable

Jun 26, 2022

CONTENTS

CHAPTER

ONE

PYTHON INTERFACE

This describes the Python interface of Pandora. There are actually two ways to obtain a model: A class-based and a function-based approach.

- The class-based version is more "pythonic" and in line with what users know from packages like batman, Wotan, TLS, and others. It offers various gimicks such as video animations of transits.
- The function-based version is simpler, procedural, but "ugly" due to passing along many parameters. Depending on the size of the model, it can be a few times faster due to the lack of class instantiation. Its main use case is for model evaluation with MCMC and nested samplers.

Both versions must yield perfectly identical results. If you find any differences, please open a bug ticket on Github.

1.1 Define data for a model

class model_params(*params*)

Star parameters:

R_star
(float) Stellar radius [m]

u1
(float) Quadratic limb-darkening [0..1]. Can be a free parameter.

u2
(float) Quadratic limb-darkening [0..1]. Can be a free parameter.

Barycenter parameters:

per_bary
(float) Period of planet-moon barycenter. Unit: [days]. Should be a free parameter when fitting.

a_bary
(float) Semimajor axis of planet-moon barycenter. Unit: [$R_\odot$]. Should be a free parameter when fitting.

r_planet
(float) Planet radius. Unit: [$R_\odot$]. Should be a free parameter when fitting.

b_bary
(float) Impact parameter of planet-moon barycenter [0..2]. Should be a free parameter when fitting.

ecc_bary
(float) Eccentricity of barycenter [0..1]. Optional, default *0*.

w_bary
(float) Argument of periapsis of barycenter [deg]. Optional, default *0*.

t0_bary
(float) Time of inferior conjunction of planet-moon barycenter [days]. Should NOT be a free parameter when fitting.

t0_bary_offset
(float) Time difference to inferior conjunction of planet-moon barycenter [days]. Should be a free parameter when fitting.

M_planet
(float) Planetary mass [kg]. Should be a free parameter when fitting.

Moon parameters:

r_moon
(float) Radius. Unit: [$R_\odot$]. Should be a free parameter.

per_moon
(float) Semimajor axis [days]. Should be a free parameter.

tau_moon
(float) Time of periapsis passage [0..1] normalized by period. Should be a free parameter.

Omega_moon
(float) Longitude of the ascending node [0..360 deg]. Should be a free parameter.

i_moon
(float) Orbit inclination [0..360 deg]. Should be a free parameter.

ecc_moon
(float) [0..1] Moon eccentricity. Optional. Optional, default *0*.

w_moon
(*float*) [0..360 deg] Argument of periapsis of barycenter. Optional, default *0*.

M_moon
(float) Mass [kg]. Should be a free parameter when fitting.

Other model parameters:

epoch_distance
(*float*) Time distance between each epoch [days]. Should NOT be a free parameter.

supersampling_factor
(*int*) (Optional parameter, default value: 1, which represents no supersampling). Higher values are integer multiples for higher supersampling. This compensates for morphological deformation at the cost of computational expense.

occult_small_threshold
(*float*) (Optional parameter, default value: 0.01) If the moon radius ($R_S/R_\odot$) is smaller than this value, its occultation is approximated with constant limb darkening under its area. To obtain a precise estimate even for very small moons, set *occult_small_threshold* to a very small value (e.g., 1^{-8}).

hill_sphere_threshold
(*float*) (Optional parameter, default value: 1.1) If the moon semimajor axis is larger than *hill_sphere_threshold*, the moon is considered unphysical. Then, a planet-only model is returned. The usual threshold should be close to *hill_sphere_threshold=1*. To keep unphysical systems, set a high value, e.g. *hill_sphere_threshold=100*.

numerical_grid
(*int*) (Optional parameter, default value: 25) Diameter in pixels of numerical grid to estimate planet-moon occultation in case at least one body is on the stellar limb. A value of 25 (100) pixels corresponds to an accuracy < 1 ppm (<0.25 ppm).

time
(*1D array of floats*) Time stamps to evaluate model at.

cache
(*2d array of floats*) Optional. Can be used to speed-up model calculation in case of fixed limb-darkening parameters.

Time grid:

epochs
(*int*) Number of transit epochs in time series.

epoch_duration
(*float*) Duration of each epoch, centered at planetary transit [days]

cadences_per_day
(*int*) Number of exposured (cadences) per day. Should be constant.

Note: There are 4 values related to planetary period: *t0_bary*, *t0_bary_offset*, *epoch_distance*, and *per_bary*. *epoch_distance* is the true planetary period, which you may (or may not) know. If you use Pandora's *time.grid* to generate time stamps, this parameter is used as the planetary period. For sampling, the parameter should be used from your prior information, in combination with *epoch_duration*, so that the time segments are wide enough to contain all relevant transit data. For sampling, *t0_bary* is the fixed single point prior information. The uncertainty and free parameter for fitting is given as *t0_bary_offset*. Similarly, *per_bary* is the free period parameter for sampling.

Example:

```
import pandoramoon as pandora
params = pandora.model_params()
R_sun = 696342000.0           # [m]
params.R_star = 1 * R_sun     # [m]
params.u1 = 0.4089            # [0..1]
params.u2 = 0.2556            # [0..1]

# Planet parameters
params.per_bary = 365.25      # [days]
params.a_bary = 215           # [R_star]
params.r_planet = 0.1         # [R_star]
params.b_bary = 0.3           # [0..1]
params.t0_bary = 11           # [days]
params.t0_bary_offset = 0     # [days]
params.M_planet = 1.8986e+27  # [kg]
params.w_bary = 20            # [deg]
params.ecc_bary = 0.2         # [0..1]

# Moon parameters
params.r_moon = 0.03526       # [R_star]
params.per_moon = 0.3         # [days]
params.tau_moon = 0.07        # [0..1]
params.Omega_moon = 0         # [0..360]
```

(continues on next page)

(continued from previous page)

```
params.i_moon = 80            # [0..360]
params.e_moon = 0.9           # [0..1]
params.w_moon = 20            # [deg]
params.mass_ratio = 0.05395   # [0..1]

# Other model parameters
params.epochs = 3  # [int]
params.epoch_duration = 0.6  # 5     # [days]
params.cadences_per_day = 250        # [int]
params.epoch_distance = 365.26       # [days]
params.supersampling_factor = 1      # [int]
params.occult_small_threshold = 0.1  # [0..1]
params.hill_sphere_threshold = 1.2
```

1.2 Get time grid

For a model comparison, the time series from the real data should be used. For a pure model, Pandora can create a suitable time grid:

class model_params(*params*)

Returns:

time
(*array*) Timestamps of the model

Example:

```
model = pandora.moon_model(params)
time = pandora.time(params).grid()
```

1.3 Evaluate model and obtain lightcurve

class model.**light_curve**

Parameters: None

Returns:

flux_total
(*array*) Lightcurve of planet and moon model

flux_planet
(*array*) Only contributions by the planet

flux_moon
(*array*) Only contributions by the moon

Example:

```
model = pandora.moon_model(params)
flux_total, flux_planet, flux_moon = model.light_curve(time)
```

1.4 Evaluate model and obtain positions

class model.**coordinates**(*time*)

Parameters: None

Returns:

px
(*array*) Planet X position at each timestamp

py
(*array*) Planet Y position at each timestamp

mx
(*array*) Moon X position at each timestamp

my
(*array*) Moon Y position at each timestamp

Example:

```
model = pandora.moon_model(params)
px_bary, py_bary, mx_bary, my_bary = model.coordinates(time)
```

1.5 Evaluate model and obtain transit video

class model.**video**(*time*)

Parameters:

limb_darkening
(*boolean*) If *True* (default), a limb-darkened star is painted using the model parameters u1, u2. If *False*, a uniformely yellow star is painted.

teff
(*float*) Star temperature in [2300..12000] K to draw the star color according to "Digital color codes of stars" (Harre & Heller 2021).

planet_color
(*string*) A matplotlib color for the planet. Default: "black".

moon_color
(*string*) A matplotlib color for the moon. Default: "black".

ld_circles
(*int*) Number of concentric circles used to paint the limb-darkened star. Default: *100*.

Returns: Matplotlib FuncAnimation object which can be viewed or saved to disk.

Example:

```
model = pandora.moon_model(params)
video = model.video(
    limb_darkening=True,
    teff=3000,
    planet_color="black",
```

(continues on next page)

(continued from previous page)

```
    moon_color="black",
    ld_circles=200
)
video.save(filename="video.mp4", fps=10, dpi=200)
```

Note: Creation takes considerable time. A progress bar is shown during video creation.

Note: Sizes of planet and moon may not be pixel-perfect due to scaling done by Matplotlib.

1.6 Convert quadratic limb darkening priors

helpers.**ld_convert**()

To sample the quadratic limb darkening coefficients more efficiently, Pandora offers a conversion routine to calculate $u_1 = 2\sqrt{q_1}q_2$ and $u_2 = \sqrt{q_1}(1 - 2q_2)$ based on q_1 and q_2 from the unit hypercube. This procedure has been shown to reduce the prior volume Kipping (2013).

Parameters:

q1

q2
(*float*): Priors [0..1] as provided by the sampler's unit hypercube

Returns:

u1

u2
(*float*) Limb darkening parameter u1, u2 for quadratic limb darkening calculation

Example:

```
from pandoramoon.helpers import ld_convert
u1, u2 = ld_convert(q1=0.4, q2=0.6)
```

The inverse is also provided, e.g. for verification:

helpers.**ld_invert**()

Parameters:

u1

u2
(*float*) Limb darkening parameter u1, u2 for quadratic limb darkening calculation

Returns:

q1

q2
(*float*): Limb darkening parameter as defined by Kipping (2013),

Example:

```
from pandoramoon.helpers import ld_invert
q1, q2 = ld_invert(u1=0.5, u2=0.5)
```

INDEX