
Bell_EBM Documentation

Release 1.1

Taylor James Bell

Nov 29, 2018

API Table of Contents:

1	Package Usage	3
1.1	Bell_EBM package	3
2	Indices and tables	17
3	License & Attribution	19
	Python Module Index	21

Bell_EBM is an object-oriented and flexible Energy Balance Model (EBM) that can be used to model the temperature of exoplanet atmospheres and observations of those planets. A wide range of planet compositions can be modelled: rocky planets, ocean worlds, and gas atmospheres. This is done by assuming there is a single, fully mixed layer which absorbs all of the incident radiation. The depth of this layer and the layer's heat capacity will change depending on the type of planet composition you are modelling. In the future, the hope is to have multiple layers to allow for simultaneous modelling of an atmosphere and an ocean or rock covered surface. At its core though, this is just an EBM, and no north-south (meridional) flows are modelled (e.g. no Hadley cells), and only manually selected solid-body rotation is permitted for east-west (zonal) flows (e.g. no jets).

Check out the [Quickstart Tutorial](#) to get an idea of the capabilities of this EBM, and explore the API for a more detailed description of each of the functions and objects. But the simplest, default usage of the model is:

```
import Bell_EBM as ebm

planet = ebm.Planet() # Many planetary parameters can be passed as arguments
star = ebm.Star() # Basic stellar parameters can be passed as arguments
system = ebm.System(star, planet)
```

1.1 Bell_EBM package

1.1.1 Submodules

1.1.2 Bell_EBM.H2_Dissociation_Routines module

`Bell_EBM.H2_Dissociation_Routines.cp_H2(T)`

Get the isobaric specific heat capacity of H2 as a function of temperature.

Parameters `T` (*ndarray*) – The temperature.

Returns The isobaric specific heat capacity of H2.

Return type *ndarray*

`Bell_EBM.H2_Dissociation_Routines.dDissFracApprox(T, mu=3320.680532597579, std=471.38088012739126)`

Calculate the derivative in the dissociation fraction of H2 using an erf approximation.

Parameters

- `T` (*ndarray*) – The temperature.
- `mu` (*float, optional*) – The mean for the Gaussian function.

- **std**(*float, optional*) – The standard deviation for the Gaussian function.

Returns The derivative in the dissociation fraction of H2.

Return type ndarray

`Bell_EBM.H2_Dissociation_Routines.dDissFracSaha(T, P)`

Calculate the derivative of the dissociation fraction of H2 using the Saha Equation.

Parameters

- **T**(*ndarray*) – The temperature.
- **P**(*ndarray*) – The pressure

Returns The derivative of the dissociation fraction of H2.

Return type ndarray

`Bell_EBM.H2_Dissociation_Routines.delta_cp_H2(T)`

Get the derivative of the isobaric specific heat capacity of H2 as a function of temperature.

Pretty sure cp_H2 should already include this factor...

Parameters **T**(*ndarray*) – The temperature.

Returns The derivative of the isobaric specific heat capacity of H2.

Return type ndarray

`Bell_EBM.H2_Dissociation_Routines.dissFracApprox(T, mu=3320.680532597579, std=471.38088012739126)`

Calculate the dissociation fraction of H2 using an erf approximation.

Parameters

- **T**(*ndarray*) – The temperature.
- **mu**(*float, optional*) – The mean for the error function.
- **std**(*float, optional*) – The standard deviation for the error function.

Returns The dissociation fraction of H2.

Return type ndarray

`Bell_EBM.H2_Dissociation_Routines.dissFracSaha(T, P)`

Calculate the dissociation fraction of H2 using the Saha Equation.

Parameters

- **T**(*ndarray*) – The temperature.
- **P**(*ndarray*) – The pressure

Returns The dissociation fraction of H2.

Return type ndarray

`Bell_EBM.H2_Dissociation_Routines.getSahaApproxParams(P=10132.5)`

Get the Gaussian and erf parameters used to approximate the Saha equation.

Parameters **P**(*ndarray*) – The pressure.

Returns 2 floats containing the mean and the standard deviation for the Gaussian/erf functions.

Return type list

`Bell_EBM.H2_Dissociation_Routines.nQ(mu, T)`

Calculate the quantum concentration.

Parameters

- **mu** (*ndarray*) – The mean molecular weight in units of u.
- **T** (*ndarray*) – The temperature.

Returns The quantum concentration.

Return type *ndarray*

`Bell_EBM.H2_Dissociation_Routines.true_cp` (*T*, *mu*=3320.680532597579,
std=471.38088012739126)

Get the isobaric specific heat capacity of an LTE mix of H2+H as a function of temperature.

Accounts for the energy of H2 dissociation/recombination.

Parameters

- **T** (*ndarray*) – The temperature.
- **mu** (*float*) – The mean for the Gaussian/erf approximations to the Saha equation.
- **std** (*float*) – The standard deviation for the Gaussian/erf approximations to the Saha equation.

Returns The isobaric specific heat capacity of an LTE mix of H2+H.

Return type *ndarray*

1.1.3 Bell_EBM.KeplerOrbit module

class `Bell_EBM.KeplerOrbit.KeplerOrbit` (*a*=149597870700.0, *Porb*=None, *inc*=90,
t0=0, *e*=0, *Omega*=270, *argp*=90,
m1=1.9884754153381438e+30, *m2*=0)

Bases: `object`

A Keplerian orbit.

a

float – The semi-major axis in m.

Porb

float – The orbital period in days.

inc

float – The orbital inclination (in degrees above face-on)

t0

float – The linear ephemeris in days.

e

float – The orbital eccentricity.

Omega

float – The longitude of ascending node (in degrees CCW from line-of-sight).

argp

float – The argument of periastron (in degrees CCW from Omega).

m1

float – The mass of body 1 in kg.

m2

float – The mass of body 2 in kg.

distance (*t*, *xtol*=1e-10)

Find the separation between the two bodies.

Parameters

- **t** (*ndarray*) – The time in days.
- **xtol** (*float*) – tolerance on error in eccentric anomaly (calculated along the way).

Returns The separation between the two bodies.

Return type ndarray

ea_to_ma (*ea*)

Convert eccentric anomaly to mean anomaly.

Parameters **ea** (*ndarray*) – The eccentric anomaly in radians.

Returns The mean anomaly in radians.

Return type ndarray

eccentric_anomaly (*t*, *xtol*=1e-10)

Convert time to eccentric anomaly, numerically.

Parameters

- **t** (*ndarray*) – The time in days.
- **xtol** (*float*) – tolerance on error in eccentric anomaly.

Returns The eccentric anomaly in radians.

Return type ndarray

ecl_time ()

Get the time of secondary eclipse.

Returns The time of secondary eclipse.

Return type float

mean_anomaly (*t*)

Convert time to mean anomaly.

Parameters **t** (*ndarray*) – The time in days.

Returns The mean anomaly in radians.

Return type ndarray

mean_motion ()

Get the mean motion.

Returns The mean motion in radians.

Return type float

peri_time ()

Get the time of periastron.

Returns The time of periastron.

Return type float

period ()

Find the keplerian orbital period.

Returns The keplerian orbital period.

Return type float

plot_orbit ()

A convenience routine to visualize the orbit

Returns The figure containing the plot.

Return type figure

ta_to_ea (ta)

Convert true anomaly to eccentric anomaly.

Parameters **ta** (*ndarray*) – The true anomaly in radians.

Returns The eccentric anomaly in radians.

Return type ndarray

ta_to_ma (ta)

Convert true anomaly to mean anomaly.

Parameters **ta** (*ndarray*) – The true anomaly in radians.

Returns The mean anomaly in radians.

Return type ndarray

trans_time ()

Get the time of transit.

Returns The time of transit.

Return type float

true_anomaly (t, xtol=1e-10)

Convert time to true anomaly, numerically.

Parameters

- **t** (*ndarray*) – The time in days.
- **xtol** (*float*) – tolerance on error in eccentric anomaly (calculated along the way).

Returns The true anomaly in radians.

Return type ndarray

xyz (t, xtol=1e-10)

Find the coordinates of body 2 with respect to body 1.

Parameters

- **t** (*ndarray*) – The time in days.
- **xtol** (*float*) – tolerance on error in eccentric anomaly (calculated along the way).

Returns

A list of 3 ndarrays containing the x,y,z coordinate of body 2 with respect to body 1.

The x coordinate is along the line-of-sight. The y coordinate is perpendicular to the line-of-sight and in the orbital plane. The z coordinate is perpendicular to the line-of-sight and above the orbital plane

Return type list

1.1.4 Bell_EBM.Map module

class Bell_EBM.Map.**Map** (*values=None, time=0, nlat=16, nlon=None, useHealpix=False, nside=7*)

Bases: object

A map.

lat

ndarray, optional – The unique latitude values in degrees.

latGrid

ndarray – The latitude grid in degrees.

lon

ndarray, optional – The unique longitude values in degrees.

lonGrid

ndarray – The longitude grid in degrees.

nlat

int, optional – The number of latitudinal cells to use for rectangular maps.

nlon

int, optional – The number of longitudinal cells to use for rectangular maps.

nside

int, optional – A parameter that sets the resolution of healpy maps.

pixArea

ndarray – The area of each pixel.

time

float – Time of map in days.

useHealpix

bool – Whether the planet’s map uses a healpix grid.

values

ndarray – The temperature map values.

load_custom_map (*values, time=None, lat=None, lon=None, latGrid=None, lonGrid=None, pixArea=None*)

Set the whole map object.

Parameters

- **values** (*ndarray*) – The map temperatures (in K) with a size of self.npix.
- **time** (*float, optional*) – Time of map in days.
- **lat** (*ndarray, optional*) – The unique latitude values in degrees.
- **lon** (*ndarray, optional*) – The unique longitude values in degrees.
- **latGrid** (*ndarray, optional*) – The latitude of every pixel in degrees.
- **lonGrid** (*ndarray, optional*) – The longitude of every pixel in degrees.
- **pixArea** (*ndarray, optional*) – The angular area of each pixel in steradians.

plot_dissociation (*refLon=None*)

A convenience routine to plot the H2 dissociation map.

Parameters **refLon** (*float, optional*) – The sub-stellar longitude used to de-rotate the map.

Returns The figure containing the plot.

Return type figure

plot_map (*refLon=None*)

A convenience routine to plot the temperature map

Parameters **refLon** (*float, optional*) – The sub-stellar longitude used to de-rotate the map.

Returns The figure containing the plot.

Return type figure

set_values (*values, time=None*)

Set the temperature map.

Parameters

- **values** (*ndarray*) – The map temperatures (in K) with a size of self.npix.
- **time** (*float, optional*) – Time of map in days.

1.1.5 Bell_EBM.Planet module

class Bell_EBM.Planet.**Planet** (*plType='gas', rad=1, mass=1, a=0.03, Porb=None, Prot=None, inc=90, i0=0, e=0, Omega=270, argp=90, obliq=0, argobliq=0, vWind=0, albedo=0, cp=None, mlDepth=None, mlDensity=None, T_exponent=4, emissivity=1, trasmissivity=0, nlat=16, nlon=None, useHealpix=False, nside=7*)

Bases: object

A planet.

a

float – The planet’s semi-major axis in m.

absorptivity

float – The absorptivity of the emitting layer (between 0 and 1).

albedo

float – The planet’s Bond albedo.

argobliq

float – The reference orbital angle used for the obliquity (in degrees from inferior conjunction).

argp

float – The planet’s argument of periastron (in degrees CCW from Omega).

c

float, optional – The planet’s heat capacity in J/m²/K.

cp

float or callable – The planet’s isobaric specific heat capacity in J/kg/K.

cpParams

iterable, optional – Any parameters to be passed to cp if using the bell2018 LTE H2+H mix cp

e

float – The planet’s orbital eccentricity.

emissivity

float – The emissivity of the emitting layer (between 0 and 1).

g
float – The planet’s surface gravity in m/s^2 .

inc
float – The planet’s orbital inclination (in degrees above face-on)

map
Bell_EBM.Map – The planet’s temperature map.

mass
float – The planet’s mass in kg.

mlDensity
float – The density of the planet’s mixed layer.

mlDepth
float – The depth of the planet’s mixed layer (in units of m for rock/water or Pa for gas/bell2018).

obliq
float – The planet’s obliquity (axial tilt) (in degrees toward star).

Omega
float – The planet’s longitude of ascending node (in degrees CCW from line-of-sight).

orbit
Bell_EBM.KeplerOrbit – The planet’s orbit.

plType
str – The planet’s composition.

Porb
float – The planet’s orbital period in days.

Prot
float – The planet’s rotational period in days.

rad
float – The planet’s radius in m.

t0
float – The planet’s linear ephemeris in days.

T_exponent
float – The exponent which determines the rate at which the planet cools (4 for blackbody cooling, 1 for Newtonian cooling) when calculating *Fout* with *bolo=True*.

transmissivity
float – The transmissivity of the emitting layer (between 0 and 1).

useHealpix
bool – Whether the planet’s map uses a healpix grid.

vWind
float – The planet’s wind velocity in m/s.

Fout (*T=None, bolo=True, wav=1e-06*)
 Calculate the instantaneous total outgoing flux.

Parameters

- **T** (*ndarray*) – The temperature (if None, use *self.map.values*).
- **bolo** (*bool, optional*) – Determines whether computed flux is bolometric (True, default) or wavelength dependent (False).

- **wav** (*float, optional*) – The wavelength to use if bolo==False.

Returns The emitted flux in the same shape as T.

Return type ndarray

Fp_vis (*t, T=None, bolo=True, wav=4.5e-06*)

Calculate apparent outgoing planetary flux (used for making phasecurves).

Weight flux by visibility/illumination kernel, assuming the star/observer are infinitely far away for now.

Parameters

- **t** (*ndarray*) – The time in days.
- **T** (*ndarray*) – The temperature (if None, use self.map.values).
- **bolo** (*bool, optional*) – Determines whether computed flux is bolometric (True, default) or wavelength dependent (False).
- **wav** (*float, optional*) – The wavelength to use if bolo==False

Returns The apparent emitted flux. Has shape (t.size, self.map.npix).

Return type ndarray

plot_dissociation (*tempMap=None, time=None*)

A convenience routine to plot the planet's H2 dissociation map.

Parameters

- **tempMap** (*ndarray, optional*) – The temperature map (if None, use self.map.values).
- **time** (*float, optional*) – The time corresponding to the map used to de-rotate the map.

Returns The figure containing the plot.

Return type figure

plot_map (*tempMap=None, time=None*)

A convenience routine to plot the planet's temperature map.

Parameters

- **tempMap** (*ndarray*) – The temperature map (if None, use self.map.values).
- **time** (*float, optional*) – The time corresponding to the map used to de-rotate the map.

Returns The figure containing the plot.

Return type figure

sop (*t*)

Calculate the sub-observer longitude and latitude.

Parameters **t** (*ndarray*) – The time in days.

Returns

A list of 2 ndarrays containing the sub-observer longitude and latitude.

Each ndarray is in the same shape as t.

Return type list

ssp (*t*)

Calculate the sub-stellar longitude and latitude.

Parameters *t* (*ndarray*) – The time in days.

Returns

A list of 2 *ndarrays* containing the sub-stellar longitude and latitude.

Each *ndarray* is in the same shape as *t*.

Return type *list*

update ()

Update the planet's properties

Used to propagate any manual changes to the planet's attributes through the other, dependent attributes.

weight (*t*, *refPos*=*'SSP'*)

Calculate the weighting of map pixels.

Weight flux by visibility/illumination kernel, assuming the star/observer are infinitely far away for now.

Parameters

- *t* (*ndarray*) – The time in days.
- *refPos* (*str*, *optional*) – The reference position; SSP (sub-stellar point) or SOP (sub-observer point).

Returns The weighting of map mixels at time *t*. Has shape (*t.size*, *self.map.npix*).

Return type *ndarray*

1.1.6 Bell_EBM.Star module

class Bell_EBM.Star.Star (*teff*=5778, *rad*=1, *mass*=1)

Bases: *object*

A star.

teff

float – The star's effective temperature in K.

rad

float – The star's radius in solar radii.

mass

float – The star's mass in solar masses.

Fstar (*bolo*=*True*, *tBright*=*None*, *wav*=*4.5e-06*)

Calculate the stellar flux for lightcurve normalization purposes.

Parameters

- *bolo* (*bool*, *optional*) – Determines whether computed flux is bolometric (*True*, default) or wavelength dependent (*False*).
- *tBright* (*ndarray*) – The brightness temperature to use if *bolo*==*False*.
- *wav* (*float*, *optional*) – The wavelength to use if *bolo*==*False*.

Returns The emitted flux in the same shape as *T*.

Return type *ndarray*

1.1.7 Bell_EBM.StarPlanetSystem module

class Bell_EBM.StarPlanetSystem.**System** (*star=None, planet=None*)

Bases: object

A Star+Planet System.

star

Bell_EBM.Star – The host star.

planet

Bell_EBM.Planet – The planet.

Fin (*t=0, bolo=True, tStarBright=None, wav=4.5e-06*)

Calculate the instantaneous incident flux.

Parameters

- **t** (*ndarray, optional*) – The time in days.
- **bolo** (*bool, optional*) – Determines whether computed flux is bolometric (True, default) or wavelength dependent (False).
- **tStarBright** (*ndarray*) – The stellar brightness temperature to use if bolo==False.
- **wav** (*float, optional*) – The wavelength to use if bolo==False.

Returns The instantaneous incident flux.

Return type ndarray

Firr (*t=0, bolo=True, tStarBright=None, wav=4.5e-06*)

Calculate the instantaneous irradiation.

Parameters

- **t** (*ndarray, optional*) – The time in days.
- **bolo** (*bool, optional*) – Determines whether computed flux is bolometric (True, default) or wavelength dependent (False).
- **tStarBright** (*ndarray*) – The stellar brightness temperature to use if bolo==False.
- **wav** (*float, optional*) – The wavelength to use if bolo==False.

Returns The instantaneous irradiation.

Return type ndarray

ODE (*t, T*)

The derivative in temperature with respect to time.

Used by scipy.integrate.ode to update the map

Parameters

- **t** (*ndarray*) – The time in days.
- **T** (*ndarray*) – The temperature map with shape (self.planet.map.npix).

Returns The derivative in temperature with respect to time.

Return type ndarray

distance (*t=0*)

Calculate the instantaneous separation between star and planet.

Parameters **t** (*ndarray, optional*) – The time in days.

Returns The separation between the star and planet in m.

Return type ndarray

get_phase (*t*)

Get the orbital phase.

Parameters *t* (ndarray) – The time in days.

Returns The orbital phase.

Return type ndarray

get_phase_eclipse ()

Get the orbital phase of eclipse.

Returns The orbital phase of eclipse.

Return type float

get_phase_periastron ()

Get the orbital phase of periastron.

Returns The orbital phase of periastron.

Return type float

get_phase_transit ()

Get the orbital phase of transit.

Returns The orbital phase of transit.

Return type float

get_teq (*t=0*)

Get the planet's equilibrium temperature.

Parameters *t* (ndarray, optional) – The time in days.

Returns The planet's equilibrium temperature at time(s) *t*.

Return type ndarray

get_tirr (*t=0*)

Get the planet's irradiation temperature.

Parameters *t* (ndarray, optional) – The time in days.

Returns The planet's irradiation temperature at time(s) *t*.

Return type ndarray

get_xyzPos (*t*)

Get the x,y,z coordinate(s) of the planet.

Parameters *t* (ndarray) – The time in days.

Returns

A list of 3 ndarrays containing the x,y,z coordinate of the planet with respect to the star.

The x coordinate is along the line-of-sight. The y coordinate is perpendicular to the line-of-sight and in the orbital plane. The z coordinate is perpendicular to the line-of-sight and above the orbital plane

Return type list

invert_lc (*fp_fstar*, *bolo=True*, *tStarBright=None*, *wav=4.5e-06*)

Invert the fp/fstar phasecurve into an apparent temperature phasecurve.

Parameters

- **fp_fstar** (*ndarray*) – The observed planetary flux normalized by the stellar flux.
- **bolo** (*bool*, *optional*) – Determines whether computed flux is bolometric (True, default) or wavelength dependent (False).
- **tBright** (*ndarray*) – The brightness temperature to use if *bolo==False*.
- **wav** (*float*, *optional*) – The wavelength to use if *bolo==False*.

Returns The apparent, disk-integrated temperature.

Return type *ndarray*

lightcurve (*t=None*, *T=None*, *bolo=True*, *tStarBright=None*, *wav=4.5e-06*, *allowReflect=True*, *allowThermal=True*)

Calculate the planet's lightcurve (ignoring any occultations).

Parameters

- **t** (*ndarray*, *optional*) – The time in days. If None, will use 1000 time steps around orbit.
- **T** (*ndarray*, *optional*) – The temperature map (either shape (1, self.planet.map.npix) and constant over time or shape is (t.shape, self.planet.map.npix). If None, use self.planet.map.values instead (default).
- **bolo** (*bool*, *optional*) – Determines whether computed flux is bolometric (True, default) or wavelength dependent (False).
- **tStarBright** (*ndarray*) – The stellar brightness temperature to use if *bolo==False*.
- **wav** (*float*, *optional*) – The wavelength to use if *bolo==False*.
- **allowReflect** (*bool*, *optional*) – Account for the contribution from reflected light.
- **allowThermal** (*bool*, *optional*) – Account for the contribution from thermal emission.

Returns The observed planetary flux normalized by the stellar flux.

Return type *ndarray*

plot_lightcurve (*t=None*, *T=None*, *bolo=True*, *tStarBright=None*, *wav=4.5e-06*, *allowReflect=True*, *allowThermal=True*)

A convenience plotting routine to show the planet's phasecurve.

Parameters

- **t** (*ndarray*, *optional*) – The time in days with shape (t.size,1). If None, will use 1000 time steps around orbit.
- **T** (*ndarray*, *optional*) – The temperature map in K with shape (1, self.planet.map.npix) if the map is constant or (t.size,self.planet.map.npix). If None, use self.planet.map.values instead.
- **bolo** (*bool*, *optional*) – Determines whether computed flux is bolometric (True, default) or wavelength dependent (False).
- **tBright** (*ndarray*) – The brightness temperature to use if *bolo==False*.
- **wav** (*float*, *optional*) – The wavelength to use if *bolo==False*.

- **allowReflect** (*bool, optional*) – Account for the contribution from reflected light.
- **allowThermal** (*bool, optional*) – Account for the contribution from thermal emission.

Returns The figure containing the plot.

Return type figure

plot_tempcurve (*t=None, T=None, bolo=True, tStarBright=None, wav=4.5e-06*)

A convenience plotting routine to show the planet’s phasecurve in units of temperature.

Parameters

- **t** (*ndarray, optional*) – The time in days with shape (t.size,1). If None, will use 1000 time steps around orbit.
- **T** (*ndarray, optional*) – The temperature map in K with shape (1, self.planet.map.npix) if the map is constant or (t.size,self.planet.map.npix). If None, use self.planet.map.values instead.
- **bolo** (*bool, optional*) – Determines whether computed flux is bolometric (True, default) or wavelength dependent (False).
- **tBright** (*ndarray*) – The brightness temperature to use if bolo==False.
- **wav** (*float, optional*) – The wavelength to use if bolo==False.

Returns The figure containing the plot.

Return type figure

run_model (*T0=None, t0=0, t1=None, dt=None, verbose=True*)

Evolve the planet’s temperature map with time.

Parameters

- **T0** (*ndarray*) – The initial temperature map with shape (self.planet.map.npix). If None, use self.planet.map.values instead (default).
- **t0** (*float, optional*) – The time corresponding to T0 (default is 0).
- **t1** (*float, optional*) – The end point of the run (default is 1 orbital period later).
- **dt** (*float, optional*) – The time step used to evolve the map (default is 1/100 of the orbital period).
- **verbose** (*bool, optional*) – Output comments of the progress of the run (default = False).

Returns A list of 2 ndarrays containing the time and map of each time step.

Return type list

1.1.8 Module contents

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`

CHAPTER 3

License & Attribution

Copyright (c) 2018 Taylor James Bell.

Bell_EBM is free software made available under the MIT License. For details see the [LICENSE](#).

If you make use of Bell_EBM in your work, please cite the Bell & Cowan (2018) paper ([arXiv](#), [ADS](#), [BibTeX](#)).

b

`Bell_EBM`, [16](#)
`Bell_EBM.H2_Dissociation_Routines`, [3](#)
`Bell_EBM.KeplerOrbit`, [5](#)
`Bell_EBM.Map`, [8](#)
`Bell_EBM.Planet`, [9](#)
`Bell_EBM.Star`, [12](#)
`Bell_EBM.StarPlanetSystem`, [13](#)

A

a (Bell_EBM.KeplerOrbit.KeplerOrbit attribute), 5
a (Bell_EBM.Planet.Planet attribute), 9
absorptivity (Bell_EBM.Planet.Planet attribute), 9
albedo (Bell_EBM.Planet.Planet attribute), 9
argobliq (Bell_EBM.Planet.Planet attribute), 9
argp (Bell_EBM.KeplerOrbit.KeplerOrbit attribute), 5
argp (Bell_EBM.Planet.Planet attribute), 9

B

Bell_EBM (module), 16
Bell_EBM.H2_Dissociation_Routines (module), 3
Bell_EBM.KeplerOrbit (module), 5
Bell_EBM.Map (module), 8
Bell_EBM.Planet (module), 9
Bell_EBM.Star (module), 12
Bell_EBM.StarPlanetSystem (module), 13

C

C (Bell_EBM.Planet.Planet attribute), 9
cp (Bell_EBM.Planet.Planet attribute), 9
cp_H2() (in module Bell_EBM.H2_Dissociation_Routines), 3
cpParams (Bell_EBM.Planet.Planet attribute), 9

D

dDissFracApprox() (in module Bell_EBM.H2_Dissociation_Routines), 3
dDissFracSaha() (in module Bell_EBM.H2_Dissociation_Routines), 4
delta_cp_H2() (in module Bell_EBM.H2_Dissociation_Routines), 4
dissFracApprox() (in module Bell_EBM.H2_Dissociation_Routines), 4
dissFracSaha() (in module Bell_EBM.H2_Dissociation_Routines), 4
distance() (Bell_EBM.KeplerOrbit.KeplerOrbit method), 5

distance() (Bell_EBM.StarPlanetSystem.System method), 13

E

e (Bell_EBM.KeplerOrbit.KeplerOrbit attribute), 5
e (Bell_EBM.Planet.Planet attribute), 9
ea_to_ma() (Bell_EBM.KeplerOrbit.KeplerOrbit method), 6
eccentric_anomaly() (Bell_EBM.KeplerOrbit.KeplerOrbit method), 6
ecl_time() (Bell_EBM.KeplerOrbit.KeplerOrbit method), 6
emissivity (Bell_EBM.Planet.Planet attribute), 9

F

Fin() (Bell_EBM.StarPlanetSystem.System method), 13
Firr() (Bell_EBM.StarPlanetSystem.System method), 13
Fout() (Bell_EBM.Planet.Planet method), 10
Fp_vis() (Bell_EBM.Planet.Planet method), 11
Fstar() (Bell_EBM.Star.Star method), 12

G

g (Bell_EBM.Planet.Planet attribute), 9
get_phase() (Bell_EBM.StarPlanetSystem.System method), 14
get_phase_eclipse() (Bell_EBM.StarPlanetSystem.System method), 14
get_phase_periastron() (Bell_EBM.StarPlanetSystem.System method), 14
get_phase_transit() (Bell_EBM.StarPlanetSystem.System method), 14
get_teq() (Bell_EBM.StarPlanetSystem.System method), 14
get_tirr() (Bell_EBM.StarPlanetSystem.System method), 14
get_xyzPos() (Bell_EBM.StarPlanetSystem.System method), 14
getSahaApproxParams() (in module Bell_EBM.H2_Dissociation_Routines), 4

I

inc (Bell_EBM.KeplerOrbit.KeplerOrbit attribute), 5
inc (Bell_EBM.Planet.Planet attribute), 10
invert_lc() (Bell_EBM.StarPlanetSystem.System method), 14

K

KeplerOrbit (class in Bell_EBM.KeplerOrbit), 5

L

lat (Bell_EBM.Map.Map attribute), 8
latGrid (Bell_EBM.Map.Map attribute), 8
lightcurve() (Bell_EBM.StarPlanetSystem.System method), 15
load_custom_map() (Bell_EBM.Map.Map method), 8
lon (Bell_EBM.Map.Map attribute), 8
lonGrid (Bell_EBM.Map.Map attribute), 8

M

m1 (Bell_EBM.KeplerOrbit.KeplerOrbit attribute), 5
m2 (Bell_EBM.KeplerOrbit.KeplerOrbit attribute), 5
map (Bell_EBM.Planet.Planet attribute), 10
Map (class in Bell_EBM.Map), 8
mass (Bell_EBM.Planet.Planet attribute), 10
mass (Bell_EBM.Star.Star attribute), 12
mean_anomaly() (Bell_EBM.KeplerOrbit.KeplerOrbit method), 6
mean_motion() (Bell_EBM.KeplerOrbit.KeplerOrbit method), 6
mlDensity (Bell_EBM.Planet.Planet attribute), 10
mlDepth (Bell_EBM.Planet.Planet attribute), 10

N

nlat (Bell_EBM.Map.Map attribute), 8
nlon (Bell_EBM.Map.Map attribute), 8
nQ() (in module Bell_EBM.H2_Dissociation_Routines), 4
nside (Bell_EBM.Map.Map attribute), 8

O

obliq (Bell_EBM.Planet.Planet attribute), 10
ODE() (Bell_EBM.StarPlanetSystem.System method), 13
Omega (Bell_EBM.KeplerOrbit.KeplerOrbit attribute), 5
Omega (Bell_EBM.Planet.Planet attribute), 10
orbit (Bell_EBM.Planet.Planet attribute), 10

P

peri_time() (Bell_EBM.KeplerOrbit.KeplerOrbit method), 6
period() (Bell_EBM.KeplerOrbit.KeplerOrbit method), 6
pixArea (Bell_EBM.Map.Map attribute), 8

planet (Bell_EBM.StarPlanetSystem.System attribute), 13

Planet (class in Bell_EBM.Planet), 9
plot_dissociation() (Bell_EBM.Map.Map method), 8
plot_dissociation() (Bell_EBM.Planet.Planet method), 11
plot_lightcurve() (Bell_EBM.StarPlanetSystem.System method), 15
plot_map() (Bell_EBM.Map.Map method), 9
plot_map() (Bell_EBM.Planet.Planet method), 11
plot_orbit() (Bell_EBM.KeplerOrbit.KeplerOrbit method), 7
plot_tempcurve() (Bell_EBM.StarPlanetSystem.System method), 16
plType (Bell_EBM.Planet.Planet attribute), 10
Porb (Bell_EBM.KeplerOrbit.KeplerOrbit attribute), 5
Porb (Bell_EBM.Planet.Planet attribute), 10
Prot (Bell_EBM.Planet.Planet attribute), 10

R

rad (Bell_EBM.Planet.Planet attribute), 10
rad (Bell_EBM.Star.Star attribute), 12
run_model() (Bell_EBM.StarPlanetSystem.System method), 16

S

set_values() (Bell_EBM.Map.Map method), 9
sop() (Bell_EBM.Planet.Planet method), 11
ssp() (Bell_EBM.Planet.Planet method), 11
star (Bell_EBM.StarPlanetSystem.System attribute), 13
Star (class in Bell_EBM.Star), 12
System (class in Bell_EBM.StarPlanetSystem), 13

T

t0 (Bell_EBM.KeplerOrbit.KeplerOrbit attribute), 5
t0 (Bell_EBM.Planet.Planet attribute), 10
T_exponent (Bell_EBM.Planet.Planet attribute), 10
ta_to_ea() (Bell_EBM.KeplerOrbit.KeplerOrbit method), 7
ta_to_ma() (Bell_EBM.KeplerOrbit.KeplerOrbit method), 7
teff (Bell_EBM.Star.Star attribute), 12
time (Bell_EBM.Map.Map attribute), 8
trans_time() (Bell_EBM.KeplerOrbit.KeplerOrbit method), 7
trasmissivity (Bell_EBM.Planet.Planet attribute), 10
true_anomaly() (Bell_EBM.KeplerOrbit.KeplerOrbit method), 7
true_cp() (in module Bell_EBM.H2_Dissociation_Routines), 5

U

update() (Bell_EBM.Planet.Planet method), 12
useHealpix (Bell_EBM.Map.Map attribute), 8
useHealpix (Bell_EBM.Planet.Planet attribute), 10

V

values (Bell_EBM.Map.Map attribute), [8](#)

vWind (Bell_EBM.Planet.Planet attribute), [10](#)

W

weight() (Bell_EBM.Planet.Planet method), [12](#)

X

xyz() (Bell_EBM.KeplerOrbit.KeplerOrbit method), [7](#)