

---

# **Bell\_EBM Documentation**

***Release 1.0.0***

**Taylor James Bell**

**Nov 02, 2018**



---

## Contents:

---

<b>1</b>	<b>Bell_EBM package</b>	<b>1</b>
1.1	Submodules	1
1.2	Bell_EBM.H2_Dissociation_Routines module	1
1.3	Bell_EBM.KeplerOrbit module	3
1.4	Bell_EBM.Map module	5
1.5	Bell_EBM.Planet module	6
1.6	Bell_EBM.Star module	9
1.7	Bell_EBM.StarPlanetSystem module	10
1.8	Module contents	13
<b>2</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>



### 1.1 Submodules

### 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**  $\mathbf{T}$  (*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**

- $\mathbf{T}$  (*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**

- $\mathbf{T}$  (*ndarray*) – The temperature.
- $\mathbf{P}$  (*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

## 1.3 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*

**eccentricAnomaly** (*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*

**meanAnomaly** (*t*)

Convert time to mean anomaly.

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

**Returns** The mean anomaly in radians.

**Return type** *ndarray*

**meanMotion** ()

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*

**show\_orbit** ()

A convenience routine to visualize the orbit

**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

**trueAnomaly** (*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

## 1.4 Bell\_EBM.Map module

**class** Bell\_EBM.Map.**Map** (*nside*=16, *values*=None, *time*=0, *useHealpix*=False)

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.

**nside**

*int* – A parameter that sets the resolution of the map.

**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.

**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.

## 1.5 Bell\_EBM.Planet module

```
class Bell_EBM.Planet.Planet (plType='gas', rad=1, mass=1, a=0.03, Porb=None, Prot=None,  
                             vWind=0, albedo=0, inc=90, t0=0, e=0, Omega=270, argp=90,  
                             obliq=0, argobliq=0, nside=16, useHealpix=False)
```

Bases: object

A planet.

**a**

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

**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/m<sup>2</sup>/K.

**cp**

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

**cpParams (**

*obj: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.

**g**

*float* – The planet’s surface gravity in m/s<sup>2</sup>.

**inc**

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

**map**

*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.

**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**

*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.

**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

**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

**showDissociation** (*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

**showMap** (*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

**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 or SOP).

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

**Return type** ndarray

## 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

## 1.7 Bell\_EBM.StarPlanetSystem module

**class** Bell\_EBM.StarPlanetSystem.**System** (*star=None, planet=None*)

Bases: object

A Star+Planet System.

**star**

*Star* – The host star.

**planet**

*Planet* – The planet.

**Finc** (*t*)

Calculate the instantaneous incident flux.

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

**Returns** The instantaneous incident flux.

**Return type** ndarray

**Firr** (*t*)

Calculate the instantaneous irradiation.

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

**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*)

Calculate the instantaneous separation between star and planet.

**Parameters** **t** (*ndarray*) – 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\_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*, *T=None*, *bolo=True*, *tStarBright=None*, *wav=4.5e-06*)

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

**Parameters**

- **t** (*ndarray*) – The time in days.
- **T** (*ndarray*) – 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).
- **tBright** (*ndarray*) – The brightness temperature to use if bolo==False.
- **wav** (*float, optional*) – The wavelength to use if bolo==False.

**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*)

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, use [self.planet.t0,self.planet.t0+self.planet.Porb].
- **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

**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, use [self.planet.t0,self.planet.t0+self.planet.Porb].
- **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

**runModel** (*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

## 1.8 Module contents



## CHAPTER 2

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### b

`Bell_EBM`, [13](#)  
`Bell_EBM.H2_Dissociation_Routines`, [1](#)  
`Bell_EBM.KeplerOrbit`, [3](#)  
`Bell_EBM.Map`, [5](#)  
`Bell_EBM.Planet`, [6](#)  
`Bell_EBM.Star`, [9](#)  
`Bell_EBM.StarPlanetSystem`, [10](#)



## A

a (Bell\_EBM.KeplerOrbit.KeplerOrbit attribute), 3  
 a (Bell\_EBM.Planet.Planet attribute), 6  
 albedo (Bell\_EBM.Planet.Planet attribute), 6  
 argobliq (Bell\_EBM.Planet.Planet attribute), 6  
 argp (Bell\_EBM.KeplerOrbit.KeplerOrbit attribute), 3  
 argp (Bell\_EBM.Planet.Planet attribute), 7

## B

Bell\_EBM (module), 13  
 Bell\_EBM.H2\_Dissociation\_Routines (module), 1  
 Bell\_EBM.KeplerOrbit (module), 3  
 Bell\_EBM.Map (module), 5  
 Bell\_EBM.Planet (module), 6  
 Bell\_EBM.Star (module), 9  
 Bell\_EBM.StarPlanetSystem (module), 10

## C

C (Bell\_EBM.Planet.Planet attribute), 7  
 cp (Bell\_EBM.Planet.Planet attribute), 7  
 cp\_H2() (in module Bell\_EBM.H2\_Dissociation\_Routines),  
 1

## D

dDissFracApprox() (in module  
 Bell\_EBM.H2\_Dissociation\_Routines), 1  
 dDissFracSaha() (in module  
 Bell\_EBM.H2\_Dissociation\_Routines), 1  
 delta\_cp\_H2() (in module  
 Bell\_EBM.H2\_Dissociation\_Routines), 2  
 dissFracApprox() (in module  
 Bell\_EBM.H2\_Dissociation\_Routines), 2  
 dissFracSaha() (in module  
 Bell\_EBM.H2\_Dissociation\_Routines), 2  
 distance() (Bell\_EBM.KeplerOrbit.KeplerOrbit method),  
 3  
 distance() (Bell\_EBM.StarPlanetSystem.System  
 method), 10

## E

e (Bell\_EBM.KeplerOrbit.KeplerOrbit attribute), 3  
 e (Bell\_EBM.Planet.Planet attribute), 7  
 ea\_to\_ma() (Bell\_EBM.KeplerOrbit.KeplerOrbit  
 method), 3  
 eccentricAnomaly() (Bell\_EBM.KeplerOrbit.KeplerOrbit  
 method), 4  
 ecl\_time() (Bell\_EBM.KeplerOrbit.KeplerOrbit method),  
 4

## F

Finc() (Bell\_EBM.StarPlanetSystem.System method), 10  
 Firr() (Bell\_EBM.StarPlanetSystem.System method), 10  
 Fout() (Bell\_EBM.Planet.Planet method), 8  
 Fp\_vis() (Bell\_EBM.Planet.Planet method), 8  
 Fstar() (Bell\_EBM.Star.Star method), 9

## G

g (Bell\_EBM.Planet.Planet attribute), 7  
 get\_phase() (Bell\_EBM.StarPlanetSystem.System  
 method), 10  
 get\_phase\_eclipse() (Bell\_EBM.StarPlanetSystem.System  
 method), 11  
 get\_phase\_periastron() (Bell\_EBM.StarPlanetSystem.System  
 method), 11  
 get\_phase\_transit() (Bell\_EBM.StarPlanetSystem.System  
 method), 11  
 get\_xyzPos() (Bell\_EBM.StarPlanetSystem.System  
 method), 11  
 getSahaApproxParams() (in module  
 Bell\_EBM.H2\_Dissociation\_Routines), 2

## I

inc (Bell\_EBM.KeplerOrbit.KeplerOrbit attribute), 3  
 inc (Bell\_EBM.Planet.Planet attribute), 7  
 invert\_lc() (Bell\_EBM.StarPlanetSystem.System  
 method), 11

## K

KeplerOrbit (class in Bell\_EBM.KeplerOrbit), 3

## L

lat (Bell\_EBM.Map.Map attribute), 5  
latGrid (Bell\_EBM.Map.Map attribute), 5  
lightcurve() (Bell\_EBM.StarPlanetSystem.System method), 11  
lon (Bell\_EBM.Map.Map attribute), 5  
lonGrid (Bell\_EBM.Map.Map attribute), 5

## M

m1 (Bell\_EBM.KeplerOrbit.KeplerOrbit attribute), 3  
m2 (Bell\_EBM.KeplerOrbit.KeplerOrbit attribute), 3  
map (Bell\_EBM.Planet.Planet attribute), 7  
Map (class in Bell\_EBM.Map), 5  
mass (Bell\_EBM.Planet.Planet attribute), 7  
mass (Bell\_EBM.Star.Star attribute), 9  
meanAnomaly() (Bell\_EBM.KeplerOrbit.KeplerOrbit method), 4  
meanMotion() (Bell\_EBM.KeplerOrbit.KeplerOrbit method), 4  
mlDensity (Bell\_EBM.Planet.Planet attribute), 7  
mlDepth (Bell\_EBM.Planet.Planet attribute), 7

## N

nQ() (in module Bell\_EBM.H2\_Dissociation\_Routines), 2  
nside (Bell\_EBM.Map.Map attribute), 6

## O

obliq (Bell\_EBM.Planet.Planet attribute), 7  
ODE() (Bell\_EBM.StarPlanetSystem.System method), 10  
Omega (Bell\_EBM.KeplerOrbit.KeplerOrbit attribute), 3  
Omega (Bell\_EBM.Planet.Planet attribute), 7  
orbit (Bell\_EBM.Planet.Planet attribute), 7

## P

peri\_time() (Bell\_EBM.KeplerOrbit.KeplerOrbit method), 4  
period() (Bell\_EBM.KeplerOrbit.KeplerOrbit method), 4  
pixArea (Bell\_EBM.Map.Map attribute), 6  
planet (Bell\_EBM.StarPlanetSystem.System attribute), 10  
Planet (class in Bell\_EBM.Planet), 6  
plot\_dissociation() (Bell\_EBM.Map.Map method), 6  
plot\_lightcurve() (Bell\_EBM.StarPlanetSystem.System method), 12  
plot\_map() (Bell\_EBM.Map.Map method), 6  
plot\_tempcurve() (Bell\_EBM.StarPlanetSystem.System method), 12  
plType (Bell\_EBM.Planet.Planet attribute), 7  
Porb (Bell\_EBM.KeplerOrbit.KeplerOrbit attribute), 3  
Porb (Bell\_EBM.Planet.Planet attribute), 7  
Prot (Bell\_EBM.Planet.Planet attribute), 7

## R

rad (Bell\_EBM.Planet.Planet attribute), 7  
rad (Bell\_EBM.Star.Star attribute), 9  
runModel() (Bell\_EBM.StarPlanetSystem.System method), 12

## S

set\_values() (Bell\_EBM.Map.Map method), 6  
show\_orbit() (Bell\_EBM.KeplerOrbit.KeplerOrbit method), 4  
showDissociation() (Bell\_EBM.Planet.Planet method), 8  
showMap() (Bell\_EBM.Planet.Planet method), 9  
SOP() (Bell\_EBM.Planet.Planet method), 8  
SSP() (Bell\_EBM.Planet.Planet method), 8  
star (Bell\_EBM.StarPlanetSystem.System attribute), 10  
Star (class in Bell\_EBM.Star), 9  
System (class in Bell\_EBM.StarPlanetSystem), 10

## T

t0 (Bell\_EBM.KeplerOrbit.KeplerOrbit attribute), 3  
t0 (Bell\_EBM.Planet.Planet attribute), 7  
ta\_to\_ea() (Bell\_EBM.KeplerOrbit.KeplerOrbit method), 4  
ta\_to\_ma() (Bell\_EBM.KeplerOrbit.KeplerOrbit method), 5  
teff (Bell\_EBM.Star.Star attribute), 9  
time (Bell\_EBM.Map.Map attribute), 6  
trans\_time() (Bell\_EBM.KeplerOrbit.KeplerOrbit method), 5  
true\_cp() (in module Bell\_EBM.H2\_Dissociation\_Routines), 2  
trueAnomaly() (Bell\_EBM.KeplerOrbit.KeplerOrbit method), 5

## U

update() (Bell\_EBM.Planet.Planet method), 9  
useHealpix (Bell\_EBM.Map.Map attribute), 6  
useHealpix (Bell\_EBM.Planet.Planet attribute), 7

## V

values (Bell\_EBM.Map.Map attribute), 6  
vWind (Bell\_EBM.Planet.Planet attribute), 7

## W

weight() (Bell\_EBM.Planet.Planet method), 9

## X

xyz() (Bell\_EBM.KeplerOrbit.KeplerOrbit method), 5