Bell_EBM Documentation

Release 1.1

Taylor James Bell

Dec 12, 2018

API Table of Contents:

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).

CHAPTER 1

Package Usage

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 theisobaric 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 deviatio 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=90.0*, *t0=0.0*, *e=0.0*, *Omega=270.0*, *argp=90.0*, *obliq=0.0*, *argobliq=0.0*, *Prot=None*, *wWind=0.0*, *m1=1.9884754153381438e+30*, *m2=0.0*)

Bases: object

A Keplerian orbit.

a
float – The semi-major axis in m.

inc
float – The orbial 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).

obliq
float, optional – The obliquity (axial tilt) of body 2 (in degrees toward body 1).

argobliq
float, optional – The reference orbital angle used for obliq (in degrees from inferior conjunction).

wWind
float, optional – Body 2's wind angular velocity in revolutions/s.

t_peri
float – Time of body 2's closest approach to body 1.

t_ecl
float – Time of body 2's eclipse by body 1.

mean_motion
float – The mean motion in radians.

FSSI(*Y*, *x*, *f*, *fP*)
Fast Switch and Spline Inversion method from Tommasini+2018.

Parameters

- **Y** (*ndarray*) – The f(x) values to invert.
- **x** (*ndarray*) – x values spanning the domain (more values for higher precision).
- **f** (*callable*) – The function f.
- **fP** (*callable*) – The first derivative of the function f with respect to x.

Returns The numerical approximation of f^-(y).

Return type ndarray

FSSI_Eccentric_Inverse(*M*, *xtol=1e-10*)
Convert mean anomaly to eccentric anomaly using FSSI (Tommasini+2018).

Parameters

- **M** (*ndarray*) – The mean anomaly in radians.
- **xtol** (*float*) – tolarance on error in eccentric anomaly.

Returns The eccentric anomaly in radians.

Return type ndarray

Newton_Eccentric_Inverse(*M*, *xtol=1e-10*)
Convert mean anomaly to eccentric anomaly using Newton.

Parameters

- **M** (*ndarray*) – The mean anomaly in radians.
- **xtol** (*float*) – tolarance on error in eccentric anomaly.

Returns The eccentric anomaly in radians.

Return type ndarray

Porb
float – Body 2's orbital period in days.

Changing this will update Prot if none was provided when the orbit was initialized.

Prot
float – Body 2's rotational period in days.

distance(*t=None*, *TA=None*, *xtol=1e-10*)
Find the separation between the two bodies.

Parameters

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

- **TA** (*ndarray*) – The true anomaly in radians (if t and TA are given, only TA will be used).
- **xtol** (*float*) – tolarance 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*, *useFSSI=None*, *xtol=1e-10*)
Convert time to eccentric anomaly, numerically.

Parameters

- **t** (*ndarray*) – The time in days.
- **useFSSI** (*bool*) – Whether or not to use FSSI to invert Kepler's equation.
- **xtol** (*float*) – tolarance on error in eccentric anomaly.

Returns The eccentric anomaly in radians.

Return type ndarray

get_phase(*t*, *TA=None*)
Get the orbital phase.

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

Returns The orbital phase.

Return type ndarray

get_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

get_ssp(*t*, *TA=None*)
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

m1
float – Body 1's mass in kg.

If no period was provided when the orbit was initialized, changing this will update the period.

m2
float – Body 2's mass in kg.

If no period was provided when the orbit was initialized, changing this will update the period.

mean_anomaly(*t*)
Convert time to mean anomaly.

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

Returns The mean anomaly in radians.

Return type ndarray

phase_eclipse
float – The orbital phase of eclipse.

Read-only.

phase_periastron
float – The orbital phase of periastron.

Read-only.

phase_transit
float – The orbital phase of transit.

Read-only.

plot_orbit()
A convenience routine to visualize the orbit

Returns The figure containing the plot.

Return type figure

solve_period()
Find the Keplerian orbital period.

Returns The Keplerian orbital period.

Return type float

t_trans
float – Time of body 1's eclipse by body 2.

Read-only.

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

true_anomaly(*t*, *xtol=1e-10*)
Convert time to true anomaly, numerically.

Parameters

- **t** (*ndarray*) – The time in days.
- **xtol** (*float*) – tolarance 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*) – tolarance 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=0.0*, *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_H2_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=71492000.0*, *mass=1.8981871658715508e+27*, *a=4487936121.0*, *Porb=None*, *Prot=None*, *inc=90.0*, *t0=0.0*, *e=0.0*, *Omega=270.0*, *argp=90*, *obliq=0.0*, *argobliq=0.0*, *vWind=0.0*, *albedo=0.0*, *cp=None*, *cpParams=None*, *mlDepth=None*, *mlDensity=None*, *T_exponent=4.0*, *emissivity=1.0*, *trasmissivity=0.0*, *nlat=16*, *nlon=None*, *useHealpix=False*, *nside=7*)

Bases: object

A planet.

albedo
float – The planet's Bond albedo.

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

C
float, optional – The planet's heat capacity in J/m^2/K.

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

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

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

orbit
Bell_EBM.KeplerOrbit – The planet's orbit.

plType
str – The planet's composition.

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

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

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

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

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

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

absorptivity
float – The fraction of flux absorbed by the planet.

Read-only.

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

Changing the planet's cp may update planet.C

mass
float – The planet's mass in kg.

Changing the planet's mass will update planet.g and possibly planet.mlDensity and planet.C

mlDensity
float – The density of the planet's mixed layer

In kg/m^3 if rock/water models or the inverse of the planet's surface gravity (in s^2/m) for gas/bell2018 models.

Changing the planet's mlDensity may update planet.C

mlDepth
float – The depth of the planet's mixed layer.

In units of m for rock/water models, or Pa for gas/bell2018 models.

Changing the planet's mlDepth may update planet.C

plot_H2_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

rad
float – The planet's radius if m.

Changing the planet's radius will update planet.g and possibly planet.mlDensity and planet.C

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

weight(*t*, *TA=None*, *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.
- **EA** (*ndarray, optional*) – The eccentric anomaly in radians.
- **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=5778.0*, *rad=1.0*, *mass=1.0*)
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*, *TA=None*, *bolo=True*, *tStarBright=None*, *wav=4.5e-06*)
Calculate the instantaneous incident flux.

Parameters

- **t** (*ndarray, optional*) – The time in days.
- **TA** (*ndarray, optional*) – The true anomaly in radians.
- **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=0.0*, *TA=None*, *bolo=True*, *tStarBright=None*, *wav=4.5e-06*)
Calculate the instantaneous irradiation.

Parameters

- **t** (*ndarray, optional*) – The time in days.
- **TA** (*ndarray, optional*) – The true anomaly in radians.
- **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*, *TA=None*)
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).
- **TA** (*ndarray, optional*) – The true anomaly in radians (much faster to compute if provided).

Returns The derivative in temperature with respect to time.

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=0.0*)
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

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*, *allowReflect=False*, *allowThermal=True*)

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.
- **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

run_model(*T0=None*, *t0=0.0*, *t1=None*, *dt=None*, *verbose=True*, *intermediates=False*)

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)?
- **intermediates** (*bool, optional*) – Output the map from every time step? Otherwise just returns the last step.

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

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

Python Module Index

b

Index

I

K

L

M

N

O

P

R

S

T