NemesisPy

Release 0.1

Jingxuan Yang, Juan Alday, Patrick Irwin

Mar 10, 2024

CONTENTS:

NemesisPy contains routines for calculating and fitting exoplanet emission spectra at arbitrary orbital phases, which can help us constrain the thermal structure and chemical abundance of exoplanet atmospheres. It is also capable of fitting emission spectra at multiple orbital phases (phase curves) at the same time. This package comes ready with some spectral data and General Circulation Model (GCM) data so you can start simulating spectra immediately. There are a few demonstration routines in the *nemesispy/examples* folder; in particular, *demo_fit_eclipse.py* contains an interactive plot routine which allows you to fit a hot Jupiter eclipse spectrum by hand by varying its chemical abundance and temperature profile. This package can be easily integrated with a Bayesian sampler such as *MultiNest* for a full spectral retrieval.

The radiative transfer calculations are done with the correlated-k approximation, and are accelerated with the *Numba* just-in-time compiler to match the speed of compiled languages such as Fortran. The radiative transfer routines are based on the well-tested Nemesis (https://github.com/nemesiscode) library developed by Patrick Irwin (University of Oxford) and collaborators.

This package has the following features:

- Written fully in Python: highly portable and customisable compared to packages written in compiled languages and can be easily installed on computer clusters.
- Fast calculation speed: the most time consuming routines are optimised with just-in-time compilation, which compiles Python code to machine code at run time.
- Radiative transfer routines are benchmarked against the extensively used Nemesis (https://github.com/nemesiscode) library.
- Contains interactive plotting routines that allow you to visualise the impact of gas abundance and thermal structure on the emission spectra.
- Contains routines to simulate spectra from General Circulation Models (GCMs).
- Contains unit tests to check if the code is working correctly after modifications.

```
$ pip install --editable .
```

To run all unit tests, change directory to the software folder and type the following in the terminal: .. code-block:: console

    $ python -m unittest discover test/

Note: This project is under active development.

CHAPTER

ONE

CONTENTS

1.1 Usage

1.1.1 Installation

To use NemesisPy, first clone the GitHub repository (https://github.com/Jingxuan97/nemesispy) to your computer. Then, navigate to the directory where you have saved the repository and run the command

```
$ pip install .
```

This will install the package and make it available to use in your Python environment. In order to install the package but still make it editable, run instead the command

```
$ pip install . --editable
```

NemesisPy can also by direcly installed from PyPI (https://pypi.org/project/nemesispy/) by running the "pip install nemesispy" command. However, we strongly recommend installing the package from the GitHub repository to make sure that you have the latest version of the package.

1.2 API

Here are the some of the functions that are available in the *NemesisPy* package.

1.2.1 common

nemesispy.**calc_hydrostat**(*P*, *T*, *mmw*, *M_plt*, *R_plt*, *H=array([], dtype=float64)*)

Calculates an altitude profile from given pressure, temperature and mean molecular weight profiles assuming hydrostatic equilibrium.

Parameters

- **P** (*ndarray*) – Pressure profile Unit: Pa
- **T** (*ndarray*) – Temperature profile Unit: K
- **mmw** (*ndarray*) – Mean molecular weight profile Unit: kg
- **M_plt** (*real*) – Planetary mass Unit: kg
- **R_plt** (*real*) – Planetary radius Unit: m
- **H** (*ndarray*) – Altitude profile to be adjusted Unit: m

Returns
adjusted_H – Altitude profile satisfying hydrostatic equlibrium. Unit: m

Return type
ndarray

nemesispy.**disc_weights**(*n*)

Generate weights for disc integration in the the emission angle direction.

Parameters
n (*int*) – Number of emission angles. Minuim 2.

Returns

- **mu** (*ndarray*) – List of cos(emission angle) for integration.
- **wtmu** (*ndarray*) – List of weights for integration.

nemesispy.**gauss_lobatto_weights**(*phase*, *nmu*)

Given the orbital phase, calculates the coordinates and weights of the points on a disc needed to compute the disc integrated radiance.

The points are chosen on a number of rings according to Gauss-Lobatto quadrature scheme, and spaced on the rings according to trapezium rule.

Refer to the begining of the trig.py file for geomety and convections.

Parameters

- **phase** (*real*) – Stellar phase/orbital phase in degrees. 0=parimary transit and increase to 180 at secondary eclipse.
- **nmu** (*integer*) – Number of zenith angle ordinates

Returns

- **nav** (*int*) – Number of FOV points
- **wav** (*ndarray*) – FOV-averaging table: 0th row is lattitude, 1st row is longitude, 2nd row is stellar zenith angle, 3rd row is emission zenith angle, 4th row is stellar azimuth angle, 5th row is weight.

nemesispy.**get_gas_name**(*id*)

Find the name of the molecule given its ID number.

Parameters
id (*int*) – ID of the molecule

Returns
name – Name of the molecule.

Return type
str

nemesispy.**get_gas_id**(*name*)

Find the ID of the molecule given its name.

Parameters
name (*str*) – Name of the molecule

Returns
id – ID of the molecule

Return type
int

nemesispy.**interp_gcm_X**(*lon*, *lat*, *p*, *gcm_lon*, *gcm_lat*, *gcm_p*, *X*, *substellar_point_longitude_shift=0*)

Find the profile of X as a function of pressure at a location specified by (lon,lat) by interpolating a GCM. X can be any scalar quantity modeled in the GCM, for example temperature or chemical abundance. Note: gcm_lon and gcm_lat must be strictly increasing.

Parameters

- **lon** (*real*) – Longitude of the location
- **lat** (*real*) – Latitude of the location
- **p** (*ndarray*) – Pressure grid for the output X(P) profile
- **gcm_lon** (*ndarray*) – Longitude grid of the GCM, assumed to be [-180,180] and increasing. (1D)
- **gcm_lat** (*ndarray*) – Latitude grid of the GCM, assumed to be [-90,90] and increasing. (1D)
- **gcm_p** (*ndarrray*) – Pressure grid of the GCM. (1D)
- **X** (*ndarray*) – A scalar quantity defined in the GCM, e.g., temperature or VMR. (3D) Has dimensioin NLON x NLAT x NP
- **substellar_point_longitude_shift** (*real*) – The longitude shift from the output coordinate system to the coordinate system of the GCM. For example, if in the output coordinate system the substellar point is defined at 0 E, whereas in the GCM coordinate system the substellar point is defined at 180 E, put substellar_point_longitude_shift=180.

Returns

- **interped_X** (*ndarray*) – X interpolated to (lon,lat,p).
- *Important*
- *Longitudinal values outside of the gcm longtidinal grid are interpolated*
- *properly using the periodicity of longitude. However, latitudinal values*
- *outside of the gcm latitude grid are interpolated using the boundary*
- *values of the gcm grid. In practice, this reduces the accuracy of*
- *interpolation in the polar regions outside of the gcm grid; in particular,*
- *the interpolated value at the poles will be dependent on longitude.*
- *This is a negligible source of error in disc integrated spectroscopy since*
- *the contribution of radiance is weighted by cos(latitude).*

1.2.2 models

nemesispy.**TP_Guillot**(*P*, *g_plt*, *T_eq*, *k_IR*, *gamma*, *f*, *T_int=100*)

TP profile from eqn. (29) in Guillot 2010. DOI: 10.1051/0004-6361/200913396 Model parameters (4) : k_IR, gamma, f, T_int

Parameters

- **P** (*ndarray*) – Pressure grid (in Pa) on which the TP profile is to be constructed.
- **g_plt** (*real*) – Gravitational acceleration at the highest pressure in the pressure grid.

- **T_eq** (*real*) – Temperature corresponding to the stellar flux. T_eq = T_star * (R_star/(2*semi_major_axis))**0.5
- **k_IR** (*real*) – Range [1e-5,1e3] Mean absorption coefficient in the thermal wavelengths.
- **gamma** (*real*) – Range ~ [1e-3,1e2] gamma = k_V/k_IR, ratio of visible to thermal opacities
- **f** (*real*) – f parameter (positive), See eqn. (29) in Guillot 2010. With f = 1 at the substellar point, f = 1/2 for a day-side average and f = 1/4 for whole planet surface average.
- **T_int** (*real*) – Temperature corresponding to the intrinsic heat flux of the planet.

Returns
TP – Temperature as a function of pressure.

Return type
ndarray

nemesispy.**TP_Guillot14**(*P*, *g_plt*, *T_eq*, *k_IR*, *gamma1*, *gamma2*, *alpha*, *beta*, *T_int*)

TP profile from eqn. (20) in Line et al. 2012. doi:10.1088/0004-637X/749/1/93, based on Parmentier and Guillot 2014. 10.1051/0004-6361/201322342. Model parameters (5) : k_IR, gamma1, gamma2, alpha, beta, T_int

Parameters

- **P** (*ndarray*) – Pressure grid (in Pa) on which the TP profile is to be constructed.
- **g_plt** (*real*) – Gravitational acceleration at the highest pressure in the pressure grid.
- **T_eq** (*real*) – Temperature corresponding to the stellar flux. T_eq = T_star * (R_star/(2*semi_major_axis))**0.5
- **k_IR** (*real*) – Range ~ [1e-5,1e3] Mean absorption coefficient in the thermal wavelengths. m^2/kg
- **gamma_1** (*real*) – Range ~ [1e-3,1e2] gamma_1 = k_V1/k_IR, ratio of mean opacity of the first visible stream to mean opacity in the thermal stream.
- **gamma_2** (*real*) – Range ~ [1e-3,1e2] gamma_2 = k_V2/k_IR, ratio of mean opacity of the second visible stream to mean opacity in the thermal stream.
- **alpha** (*real*) – Range [0,1] Percentage of the visible stream represented by opacity gamma1.
- **beta** (*real*) – Range [0,2] A catch all parameter for albedo, emissivity, day-night redistribution.
- **T_int** (*real*) – Temperature corresponding to the intrinsic heat flux of the planet.

Returns
TP – Temperature as a function of pressure.

Return type
ndarray

nemesispy.**Model4**(*P_grid*, *lon_grid*, *lat_grid*, *g_plt*, *T_eq*, *scale*, *phase_offset*, *log_kappa_day*, *log_gamma_day*, *log_f_day*, *T_int_day*, *log_kappa_night*, *log_gamma_night*, *log_f_night*, *T_int_night*, *n*)

2D temperature model consisting of two representative Guillot TP profiles. See model 4 in Yang et al. 2023. (https://doi.org/10.1093/mnras/stad2555) The atmosphere is partitioned (in longitude) into two regions: a day-side and a nightside. The dayside longitudinal span is allowed to vary.

Parameters

- **P_grid** (*ndarray*) – Pressure grid (in Pa) of the model.
- **lon_grid** (*ndarray*) – Longitude grid (in degree) of the model. Substellar point is assumed to be at 0. Range is [-180,180].

- **lat_grid** (*ndarray*) – Latitude grid (in degree) of the model. Range is [-90,90].
- **g_plt** (*real*) – Gravitational acceleration at the highest pressure in the pressure grid.
- **T_eq** (*real*) – Temperature corresponding to the stellar flux. T_eq = T_star * (R_star/(2*semi_major_axis))**0.5
- **scale** (*real*) – Scaling parameter for the longitudinal span of the dayside. Set to be between 0.5 and 1.2.
- **phase_offset** (*real*) – Central longitude of the dayside Set to be between -45 and 45.
- **log_kappa_day** (*real*) – Range [1e-5,1e3] Mean absorption coefficient in the thermal wavelengths. (dayside)
- **log_gamma_day** (*real*) – Range ~ [1e-3,1e2] gamma = k_V/k_IR, ratio of visible to thermal opacities (dayside)
- **log_f_day** (*real*) – f parameter (positive), See eqn. (29) in Guillot 2010. With f = 1 at the substellar point, f = 1/2 for a day-side average and f = 1/4 for whole planet surface average. (dayside)
- **T_int_day** (*real*) – Temperature corresponding to the intrinsic heat flux of the planet. (dayside)
- **log_kappa_night** (*real*) – Same as above definitions but for nightside.
- **log_gamma_night** (*real*) – Same as above definitions but for nightside.
- **log_f_night** (*real*) – Same as above definitions but for nightside.
- **T_int_night** (*real*) – Same as above definitions but for nightside.
- **n** (*real*) – Parameter to control how temperature vary with longitude on the dayside. Should be positive.

Returns
tp_out – Temperature model defined on a (longitude, laitude, pressure) grid.

Return type
ndarray

1.2.3 radtran

nemesispy.**calc_layer**(*planet_radius*, *H_model*, *P_model*, *T_model*, *VMR_model*, *ID*, *NLAYER*, *path_angle*, *H_0=0.0*, *layer_type=1*, *custom_path_angle=0.0*, *custom_H_base=None*, *custom_P_base=None*)

Top level routine that calculates absorber-amount-weighted average layer properties from an atmospehric model.

Parameters

- **planet_radius** (*real*) – Reference planetary planet_radius where H_model is set to be 0. Usually set at surface for terrestrial planets, or at 1 bar pressure level for gas giants.
- **H_model(NMODEL)** (*ndarray*) – Altitudes of the atmospheric model points. Assumed to be increasing. Unit: m
- **P_model(NMODEL)** (*ndarray*) – Pressures of the atmospheric model points. Unit: Pa
- **T_mode(NMODEL)** (*ndarray*) – Temperature of the atmospheric model points. Unit: K

- **VMR_model(NMODEL** (*ndarray*) – Volume mixing ratios of gases defined in the atmospheric model. VMR_model[i,j] is Volume Mixing Ratio of jth gas at ith profile point. The jth column corresponds to the gas with RADTRANS ID ID[j].
- **NGAS)** (*ndarray*) – Volume mixing ratios of gases defined in the atmospheric model. VMR_model[i,j] is Volume Mixing Ratio of jth gas at ith profile point. The jth column corresponds to the gas with RADTRANS ID ID[j].
- **ID** (*ndarray*) – Gas identifiers.
- **NLAYER** (*int*) – Number of layers to split the atmospheric model into.
- **path_angle** (*real*) – Zenith angle in degrees defined at H_0.
- **H_0** (*real, optional*) – Altitude of the lowest point in the atmospheric model. This is defined with respect to the reference planetary radius, i.e. the altitude at planet_radius is 0. The default is 0.0.
- **layer_type** (*int, optional*) – Integer specifying how to split up the layers. 0 = split by equal changes in pressure 1 = split by equal changes in log pressure 2 = split by equal changes in height 3 = split by equal changes in path length 4 = split by given layer base pressures P_base 5 = split by given layer base heights H_base Note 4 and 5 force NLAYER = len(P_base) or len(H_base). The default is 1.
- **custom_path_angle** (*real, optional*) – Required only for layer type 3. Zenith angle in degrees defined at the base of the lowest layer. The default is 0.0.
- **custom_H_base(NLAYER)** (*ndarray, optional*) – Required only for layer type 5. Altitudes of the layer bases defined by user. The default is None.
- **custom_P_base(NLAYER)** (*ndarray, optional*) – Required only for layer type 4. Pressures of the layer bases defined by user. The default is None.

Returns

- **H_layer(NLAYER)** (*ndarray*) – Averaged layer altitudes. Unit: m
- **P_layer(NLAYER)** (*ndarray*) – Averaged layer pressures. Unit: Pa
- **T_layer(NLAYER)** (*ndarray*) – Averaged layer temperatures. Unit: K
- **U_layer(NLAYER)** (*ndarray*) – Total gaseous absorber amounts along the line-of-sight path, i.e. total number of gas molecules per unit area. Unit: no of absorber per m^2
- **VMR_layer(NLAYER, NGAS)** (*ndarray*) – Averaged layer volume mixing ratios. VMR_layer[i,j] is averaged VMR of gas j in layer i.
- **scale(NLAYER)** (*ndarray*) – Layer scaling factor, i.e. ratio of path length through each layer to the layer thickness.
- **dS(NLAYER)** (*ndarray*) – Path lengths. Unit: m

nemesispy.**calc_mmw**(*ID*, *VMR*, *ISO=[]*)

Calculate mean molecular weight in kg given a list of molecule IDs and a list of their respective volume mixing ratios.

Parameters

- **ID** (*ndarray or list*) – A list of Radtran gas identifiers.
- **VMR** (*ndarray or list*) – A list of VMRs corresponding to the gases in ID.
- **ISO** (*ndarray or list*) – If ISO=[], assume terrestrial relative isotopic abundance for all gases. Otherwise, if ISO[i]=0, then use terrestrial relative isotopic abundance for the ith gas. To specify particular isotopologue, input the corresponding Radtran isotopologue identifiers.

Returns
mmw – Mean molecular weight. Unit: kg

Return type
real

Notes

Cf mol_id.py and mol_info.py.

nemesispy.**calc_planck**(*wave_grid*, *T*, *ispace=1*)

Calculates the blackbody radiance.

Parameters

- **wave_grid(nwave)** (*ndarray*) – Wavelength or wavenumber array Unit: um or cm-1
- **T** (*real*) – Temperature of the blackbody (K)
- **ispace** (*int*) – Flag indicating the spectral units (0) Wavenumber (cm-1) (1) Wavelength (um)

Returns

- **bb(nwave)** (*ndarray*)
- *Spectral radiance.*
- **Unit** (*(0) W cm-2 sr-1 (cm-1)-1*) –

(1) W cm-2 sr-1 um-1

nemesispy.**calc_radiance**(*wave_grid*, *U_layer*, *P_layer*, *T_layer*, *VMR_layer*, *k_gas_w_g_p_t*, *P_grid*, *T_grid*, *del_g*, *ScalingFactor*, *R_plt*, *solspec*, *k_cia*, *ID*, *cia_nu_grid*, *cia_T_grid*, *dH*)

Calculate emission spectrum using the correlated-k method.

Parameters

- **wave_grid(NWAVE)** (*ndarray*) – Wavelengths (um) grid for calculating spectra.
- **U_layer(NLAYER)** (*ndarray*) – Surface density of gas particles in each layer. Unit: no. of particle/m^2
- **P_layer(NLAYER)** (*ndarray*) – Atmospheric pressure grid. Unit: Pa
- **T_layer(NLAYER)** (*ndarray*) – Atmospheric temperature grid. Unit: K
- **VMR_layer(NLAYER** (*ndarray*) – Array of volume mixing ratios for NGAS. Has dimensioin: NLAYER x NGAS
- **NGAS)** (*ndarray*) – Array of volume mixing ratios for NGAS. Has dimensioin: NLAYER x NGAS
- **k_gas_w_g_p_t(NGAS** (*ndarray*) – k-coefficients. Has dimension: NGAS x NWAVE x NG x NPRESSK x NTEMPK.
- **NWAVE** (*ndarray*) – k-coefficients. Has dimension: NGAS x NWAVE x NG x NPRESSK x NTEMPK.
- **NG** (*ndarray*) – k-coefficients. Has dimension: NGAS x NWAVE x NG x NPRESSK x NTEMPK.
- **NPRESSK** (*ndarray*) – k-coefficients. Has dimension: NGAS x NWAVE x NG x NPRESSK x NTEMPK.

- **NTEMPK)** (*ndarray*) – k-coefficients. Has dimension: NGAS x NWAVE x NG x NPRESSK x NTEMPK.
- **P_grid(NPRESSK)** (*ndarray*) – Pressure grid on which the k-coeff's are pre-computed. We want SI unit (Pa) here.
- **T_grid(NTEMPK)** (*ndarray*) – Temperature grid on which the k-coeffs are pre-computed. In Kelvin
- **del_g** (*ndarray*) – Quadrature weights of the g-ordinates.
- **ScalingFactor(NLAYER)** (*ndarray*) – Scale stuff to line of sight
- **R_plt** (*real*) – Planetary radius Unit: m
- **solspec** (*ndarray*) – Stellar spectra, used when the unit of the output is in fraction of stellar irradiance.

  Stellar flux at planet's distance (W cm-2 um-1 or W cm-2 (cm-1)-1)

Returns

spectrum – Output spectrum (W cm-2 um-1 sr-1)

Return type

ndarray

INDEX