TERESA: Three-dimensional Exoplanet Retrieval from Eclipse Spectroscopy of Atmospheres (name not final)

Author
Ryan C. Challener

December 4, 2020

1 Overview

The goal of this package is to retrieve the three-dimensional atmosphere of an exoplanet from spectroscopic eclipse observations. It builds upon the work of Rauscher et al., 2018, where they use principal component analysis of light curves generated from spherical harmonic maps to determine a set of orthogonal light curves ("eigencurves"). These eigencurves are linearly combined to produce a best-fitting light-curve model, which is then converted to a temperature map.

This code constructs these temperature maps for each spectroscopic light curve and then places them vertically within the atmosphere. It then runs a radiative transfer calculation to produce planetary emission as a function of location on the planet. The emission is integrated over the wavelength filters and combined with the visibility function to create light curves to compare against the data. This calculation is put behind a Markov-chain Monte Carlo (MCMC) to explore parameter space.

The code is split into two operating modes: 2D and 3D. These modes are described in detail in the following sections.

2 2D Mode

2.1 Constructing 2D Maps

The "2D" mode of the code constructs the 2-dimensional thermal maps which are used in the "3D" mode. Hence, the code must be run in 2D before attempting a 3D fit.

First, the code calculates light curves from positive and negative spherical harmonics maps, up to a user-supplied complexity (l_{max}). Then, these harmonics light curves are run through a principle component analysis (PCA) to determine a set of orthogonal light curves ("eigencurves"), ordered by importance. The eigencurves are linearly combined with the uniform-map stellar and planetary light curves and fit, individually, to the spectroscopic light curves. That is, the same set of eigencurves are used for each spectroscopic bin, but they are allowed different weights. Functionally, the model is the following:

$$F_{\mathrm{sys}}(t) = c_0 Y_0^0 + \sum_i^N c_i E_i + F_{\mathrm{star}} + s_{\mathrm{corr}}, \tag{1}$$

where F_{sys} is the system flux, N is the number of eigencurves to use (set by the user), c_i are the eigencurve weights, E_i are the eigencurves, Y_0^0 is the light curve of the uniform-map planet, F_{star} is the light curve of the star (like constant and equal to unity, if the data are normalized), and s_{corr} is a constant stellar flux correction term to ensure that when the planet is eclipsed, the system flux matches the stellar flux.

This model is fit to each spectroscopic light curve using a least-squares minimization algorithm. Optionally, the user may specify that emitted fluxes must be positive. Negative fluxes are problematic because they are non-physical, and imply a negative temperature, which will cause problems for any attempts to run radiative transfer, a necessary step in 3D fitting. If this option is enabled, the model includes a check for positive fluxes; if negatives are found, the model returns a very poor fit, forcing the fit toward physically plausible thermal maps. Note that the code only enforces this condition on visible grid cells of the planet. Although the flux maps are defined on non-visible grid cells, this is only due to the continuity enforced by spherical harmonics. In reality, non-visible cells are completely unconstrained.

The code then uses the c_i weights along with the "eigenmaps" that match the eigencurves to compute a single flux map for each input light curve (Equation 4 of Rauscher et al., 2018):

$$Z_p(\theta,\phi) = c_0 Y_0^0(\theta,\phi) + \sum_i^N c_i Z_i(\theta,\phi), \tag{2}$$

where Z_p is the flux map, θ is latitude, ϕ is longitude, and Z_i are the eigenmaps.

These flux maps are converted to temperature maps using Equation 8 of Rauscher et al., 2018 (here we have included the stellar correction term as described in the appendix):

$$T_p(\theta,\phi) = (hc/\lambda k)/\ln\left[1 + \left(\frac{R_p}{R_s}\right)^2 \frac{\exp[hc/\lambda k T_s] - 1}{\pi Z_p(\theta,\phi)(1+s_{\mathrm{corr}})}\right], \tag{3}$$

where λ is the band-averaged wavelength of the filter used to observe the related light curve, R_p is the radius of the planet, R_s is the radius of the star, and T_s is the stellar temperature.

2.2 The Visibility Function

The 2D mode also computes the visibility function, which describes the visibility of each grid cell on the planet as a function of time. There are two sources of reduced visibility: line-of-sight (reduced visibility toward the limb) and the star. Thus,

$$V(\theta,\phi,t) = L(\theta,\phi,t)S(\theta,\phi,t), \tag{4}$$

where V is the visibility, L is line-of-sight visibility, and S is visibility due to the star. We define θ and ϕ to be locations on the planet with respect to the observer. As the planet revolves, the "sub-observer" point ($\theta = 0$, $\phi = 0$) moves in true latitude and longitude. The θ and ϕ of each grid cell change with time, but the cells' latitudes and longitudes are constant.

Line-of-sight visibility depends primarily on angular distance from the point on the planet closest to the observer. However, the planet is subdivided into discrete grid cells, which are likely large due to runtime considerations. Hence, a significant portion of the planet's flux may come from grid cells near the limb, which are only partially visibile. Therefore, we include a term to adjust these limb cells to only account for the visible portion of the cell. L is then

$$L(\theta,\phi,t) = \cos\bar{\theta}\cos\bar{\phi}A_{\mathrm{corr}}, \tag{5}$$

where $\bar{\theta}$ is the average visible θ, $\bar{\phi}$ is the average visible ϕ, and A_{corr} is the area correction term. If we define (ϕ_{min}, ϕ_{max}) as the range of latitudes of a grid cell, (θ_{min}, θ_{max}) as the range of longitudes of a grid cell, and the primed versions as the corresponding *visible* ranges, then

$$
\begin{aligned}
A_{\text{corr}} &= \frac{A_{\text{partial}}}{A_{\text{full}}} && (6)\\
&= \frac{\int_{\phi'_{\text{min}}}^{\phi'_{\text{max}}} \int_{\theta'_{\text{min}}}^{\theta'_{\text{max}}} \cos\theta d\theta d\phi}{\int_{\phi_{\text{min}}}^{\phi_{\text{max}}} \int_{\theta_{\text{min}}}^{\theta_{\text{max}}} \cos\theta d\theta d\phi} && (7)\\
&= \left(\frac{\phi'_{\text{max}} - \phi'_{\text{min}}}{\phi_{\text{max}} - \phi_{\text{min}}}\right)\left(\frac{\sin\theta'_{\text{max}} - \sin\theta'_{\text{min}}}{\sin\theta_{\text{max}} - \sin\theta_{\text{min}}}\right). && (8)
\end{aligned}
$$

For fully-visible grid cells, $A_{\text{corr}} = 1$.

Stellar visibility is the crux of eclipse mapping. As the planet moves behind the star, it is gradually eclipsed, from west to east, and then vice versa when the planet reemerges. Different grid cells are visible at different times, which enables disentangling the emission of each grid cell from the planetary emission as a whole. Currently, the code uses a very simple form of stellar visibility, where a grid cell is flagged as 100% visible or 0% visible depending on its location projected onto the plane perpendicular to the observer's line of sight. In functional form,

$$
S(\theta, \phi, t) = \begin{cases} 0 & \text{if}\, d < R_s \\ 1 & \text{otherwise} \end{cases}
$$

where d is the projected distance between the center of the visible portion of the grid cell and the center of the star, defined as

$$
\begin{aligned}
d &= \sqrt{(x_{\text{cell}} - x_s)^2 + (y_{\text{cell}} - y_s)^2} && (9)\\
&= \sqrt{(x_p + R_p\cos\bar{\theta}\sin\bar{\phi} - x_s)^2 + (y_p + R_p\sin\bar{\theta} - y_s)^2}. && (10)
\end{aligned}
$$

x_p is the x position of the planet, x_s is the x position of the star, y_p is the y position of the planet, and y_s is the y position of the star.

We compute L and S for every grid cell at every time in the observation. Later, in the 3D operating mode, this precomputed visibility grid is multiplied with the planetary emitted flux and then summed over the grid cells at each time to compute the spectroscopic light curves.

3 3D Mode

The 3D portion of the code places the 2D thermal maps vertically in the planet's atmosphere, generates an atmospheric composition, runs radiative transfer on each grid cell, integrates the emergent flux over the observation filters, combines the flux with the visibility function, and integrates over the planet to calculate spectroscopic light curves for comparison to the data. The process is done thousands to millions of times behind an MCMC algorithm to accurately estimate parameter uncertainties.

3.1 Temperature-Pressure Mapping Functions

The manner in which the thermal maps are placed vertically in the atmosphere is one of the most important choices in the 3D model. The following options are currently available:

- Interpolate/Extrapolate – Assigns each 2D thermal map to a location-independent pressure, then linearly interpolates the temperature-pressure (TP) profiles in each grid cell in log(pressure) space. Pressure layers above and below the 2D thermal maps are assigned extrapolated temperatures. This can cause negative or extremely high temperatures, so use with caution.
- Interpolate/Isothermal – The same as Interpolate/Extrapolate, except pressure layers above and the below the 2D thermal maps, where interpolation is impossible, are instead set to the same temperature as the closest (in pressure) 2D thermal map.

3.2 Atmospheric Composition

The code also generates an atmospheric composition, as atomic and molecular abundances vs. pressure for each grid cell. Currently the only option is thermochemical equilibrium using the `rate` package. In theory, more complex schemes are possible, including options to fit to atmospheric composition.

3.3 Radiative Transfer

Once the temperature and compositional structure of the atmosphere are set, the code runs radiative transfer to calculate the emergent flux from each grid cell. The following radiative transfer packages are available:

- Tau-REx 3 – https://github.com/ucl-exoplanets/TauREx3_public/

For the sake of efficiency, radiative transfer is only run on grid cells which are visible at some point during the observation. This also prevents problems when negative temperatures are present on the non-visible portions of the planet. If negative temperatures are found in any visible grid cells, the code will return negative fluxes, which should always be a worse fit than any physical fluxes, thereby driving any fitting or MCMC algorithm toward non-negative temperatures.

The emergent flux from each grid cell is then integrated over the observation filters and combined with the visibility function to generate light curves for comparison to the data.

3.4 MCMC

The light-curve model function, described above, is run within an MCMC to explore parameter space and accurately estimate parameter uncertainties. The MCMC is done through MC3 (https://github.com/pcubillos/MC3), which offers 3 sampling algorithms: Metropolis-Hastings random walk, Differential Evolutions Markov-chain Monte Carlo, and “snooker”.

4 User Interface

The code is executed from the command line with the following commands:

```
./run.py 2d example.cfg
./run.py 3d example.cfg
```

to run the 2D and 3D operating modes, respectively. The 3rd command-line argument ("example.cfg" above) points to a configuration file, in ConfigParser format, which allows the user to supply a wide array of options to the code.

4.1 Configuration File

The configuration file follows the ConfigParser format, a standard Python format. Options are separated into several different categories (e.g., "General", "Star", "Planet", etc.). The following options are available:

- General options:
  - outdir – The name of the directory where output will be saved.
  - lmax – The maximum l value of the spherical harmonic maps which are used to generate the eigencurves and eigenmaps used in the 2D operating mode. Increasing (decreasing) this value will result in more (less) complex eigenmaps. Must be an integer.
  - plots – A boolean flag to make plots. Turning this off can reduce runtimes.
  - animations – A boolean flag to make animations. Turning this off can significantly reduce runtimes.
  - ncurves – The number of eigencurves to use in the 2D fits. A higher number of curves increases the complexity of the fit. A Bayesian Information Criterion comparison between fits with different values for ncurves can tell you how many eigencurves are warranted. The maximum value of ncurves is $(\text{lmax}+1)^2$, although using more than $(\text{lmax}+1)^2/2$ is not advised as there will be significant correlations between positive and negative versions of the same eigencurve.
  - ncpu – The number of parallel processes to use in MCMC. Setting this higher will make the code run faster, up to a limit based on your hardware.
  - nsamples – The number of iterations in the MCMC. Should be large enough to ensure that the MCMC has converged.
  - burnin – The number of iterations that are discarded from the start of each chain. Should be much less than nsamples / nchains.
  - leastsq – The type of least-squares algorithm to use when performing minimization.
  - timefile – A path to the file which contains a single column of the times of the observations.

  - fluxfile – A path to the file which contains a column of fluxes for each spectroscopic light curve.
  - ferrfile – A path to the file which contains a column of flux unvertainties for each spectroscopic light curve.
  - atmtype – Type of atmosphere. Options are: "eq" (thermochemical equilibrium).
  - nlayers – Number of pressure layers in the atmosphere. They will be calculated in log(pressure) space.
  - ptop – The pressure at the top of the atmosphere, in bars.
  - pbot – The pressure at the bottom of the atmosphere, in bars.
  - res – The number of planetary grid cells in latitude and longitude. For example, res = 9 will create a grid of 81 cells, with widths of 40° and heights of 20°.
  - posflux – A boolean instructing the code to enforce a positive-flux condition. If true, the 2D maps will be required to have positive emission in every grid cell that is visible at any point during the observation.
  - elemfile – A path to a file containing solar elemental abundances. Currently unused, but may be necessary for future radiative transfer packages.
  - filtfiles – A list of paths to files describing filter transmission. Files must have two columns: wavelength in microns and filter transmission. Transmission need not be normalized, as the filters will be normalized prior to integration.
  - mapfunc – The TP mapping function that links 2D maps to pressures in each grid cell. Current options are: "constant" (each 2D map is placed at a single, constant pressure).
  - oob – "Out-of-bounds" behavior of the TP mapping function, if applicaable. Options are: "iso" (isothermal) or "extrapolate".
  - rtfunc – The radiative transfer package to use. Current options are: "taurex".

- Star options
  - m – Stellar mass in solar masses.
  - r – Stellar radius in solar radii.
  - prot – Stellar rotation period in days.
  - t – Stellar temperature in Kelvin.
  - d – Distance to the star in parsec.
  - z – Metallicity relative to solar.

- Planet options
  - m – Planetary mass in solar masses.
  - r – Planetary radius in solar radii.
  - porb – Planetary orbital period in days.

  - prot – Planetary rotation period in days.
  - Omega – Planetary longitude of the ascending node in degrees.
  - ecc – Planetary eccentricity.
  - inc – Planetary inclination in degrees (90 is edge-on).
  - b – Planetary impact parameter.
  - a – Planetary semi-major axis in AU.
  - t0 – Time of transit in days.

- Tau-REx options
  - csxdir – Directory containing molecular cross-section opacities.
  - ciadir – Directory containing collision-induced absoprtion opacities.
  - wnlow – Minimum wavenumber (/cm) to use in the radiative transfer. Reducing the wavenumber range of the radiative transfer can significantly reduce runtimes, but be sure that your filters are fully included.
  - wnhigh – Maximum wavenumber (/cm) to use in the radiative transfer. Reducing the wavenumber range of the radiative transfer can significantly reduce runtimes, but be sure that your filters are fully included.
  - wndelt – Resolution of the wavenumber grid. Tau-REx doesn't seem to actually use this.
  - mols – Molecules which will be included in the opacities. Other molecules will be put in the atmosphere, but will be assumed to have 0 opacity. Reducing molecules can significantly decrease runtimes, but may ignore important opacity sources.