ThERESA

Release 0.1

Ryan Challener, Emily Rauscher

Feb 03, 2022

CONTENTS

ThERESA is a Python package for three-dimensional exoplanet retrieval from spectroscopic eclipse observations. It uses principle component analysis to fit a two-dimensional thermal map to light curves at each wavelength, or filter, then combines these maps into a three-dimensional atmosphere, computes emission across the planet using radiative transfer, and integrates over the planet to calculate light curves for comparison with observations.

Author Ryan Challener and Emily Rauscher

Contact rchallen@umich.edu

CHAPTER

ONE

DOCUMENTATION

1.1 Installation

Install ThERESA from GitHub with the following:

```
git clone --recursive https://github.com/rychallener/theresa
```

ThERESA comes with a conda environement to install its dependencies, which you install with:

```
cd theresa
conda env create -f environment.yml
```

1.2 Getting Started

There is an example application to the ultra-hot Jupiter WASP-76b included in the ThERESA installation. First, download the required ExoTransmit opacities. There is a script included to do this, found in the example directory. Assuming you are in the top level directory, do:

```
cd example
./fetchopac.sh
```

This will populate example/opac with the necessary files. Then, ThERESA can be run from the command line with the following commands:

```
cd ../theresa
theresa.py 2d wasp76-example.cfg
theresa.py 3d wasp76-example.cfg
```

The available configuration options are discussed in more detail in *Configuration*.

1.3 Configuration

Users supply inputs and setting to ThERESA through a configuration file, following the ConfigParser format. The file is split into several sections:

- General
- 2D
- 3D
- Star
- Planet
- taurex

1.3.1 General Settings

This section has one option – the location of output (for 2D mode) or input and output (for 3D mode).

- outdir – The directory where output will be written. The directory is created if it does not exist.

1.3.2 2D Settings

This section contains options for the 2D fits.

- ncpu – Number of CPUs to use in parallel. Also sets the number of chains in the MCMC.
- nsamples – Number of total iterations in the MCMC.
- burnin – Number of "burned-in" (discarded) iterations per Markov chain.
- lmax – Largest l value to use when generating the spherical harmonic basis maps.
- ncurves – Number of eigencurves used in the fits.
- pca – Type of principle compoenent analysis to use. Options are 'pca' for typical PCA and 'tsvd' for truncated singular-value decomposition, which does not do mean subtraction to ensure physically plausible eigencurves. 'tsvd' is the recommended option.
- ncalc – Number of calculations for post-MCMC analysis. If higher than nsamples, it will be reduced to equal nsamples.
- leastsq – Controls whether to do a least-squares minimization prior to fitting, and if so, which kind. Options are 'None', 'trf' (respect parameter boundaries), and 'lm' (faster, does not respect boundaries).
- nlat, nlon – Number of evenly spaced latitude and longitude bins for plotting, visibility function calculation, and, later, 3D modeling.
- posflux – Boolean to enforce positive flux in visible grid cells. Setting this to False may prevent future 3D modeling.
- timefile – Path to a file that lists the times of observation.
- fluxfile – Path to a file that lists the fluxes of the observation.
- ferrfile – Path to a file that lists the flux uncertainties of the observation.
- filtfiles – List of files that contain transmission information for the filters that correspond to the fluxes of the observation.
- plots – Boolean to turn plotting on or off.

- animations – Boolean to turn animations on or off.

1.3.3 3D Settings

This section contains options for the 3D fit.

- ncpu – Number of CPUs to use in parallel. Also sets the number of chains in the MCMC.
- nsamples – Number of total iterations in the MCMC.
- burnin – Number of "burned-in" (discarded) iterations per Markov chain.
- leastsq – Controls whether to do a least-squares minimization prior to fitting, and if so, which kind. Options are 'None', 'trf' (respect parameter boundaries), and 'lm' (faster, does not respect boundaries). The complexity of the 3D model usually makes a least-squares optimization difficult.
- elemfile – Path to a file that describes elemental ratios.
- atmtype – Type of atomspheric composition to use. Options are 'rate' and 'GGchem'. See *Code Description* for more information.
- atmfile – If using a GGchem composition, this is a path to the GGchem output file.
- nlayers – Number of layers in the fitted atmosphere, equally spaced in log-pressure.
- ptop – Pressure at the top of the atmosphere, in bars.
- pbot – Pressure at the bottom of the atmosphere, in bars.
- rtfunc – Radiative transfer function to use. Currently limited to 'taurex'.
- mapfunc – Function to link 2D maps to pressure levels in the 3D model. Options are 'isobaric', 'sinusoidal', and 'flexible'. See *Code Description* for more information.
- oob – Out-of-bounds behavior when extrapolating the 3D thermal structure. Options are 'isothermal', 'top' (add a parameter for top-of-the-atmosphere temperature), 'bot' (add a parameter for bottom-of-the-atmosphere temperature), and 'both'.
- interp – Method to use when interpolating the 3D thermal structure. Options are 'linear', 'quadratic', and 'cubic'.
- fitcf – Boolean to turn on or off enforced contribution function consistency.
- mols – List of molecules which will have opacity in the radiative transfer calculation.
- params (optional) – Sets the starting values for the parameters in the MCMC.
- pmin (optional) – Sets the lower boundaries for the parameters in the MCMC.
- pmax (optional) – Sets the upper boundaries for the parameters in the MCMC.
- pnames (optional) – Sets the names of the MCMC parameters, for plotting.
- plots – Boolean to turn plotting on or off.
- animations – Boolean to turn animations on or off.

1.3.4 Star

Stellar parameters.

- m – Mass in solar masses.
- r – Radius in solar radii.
- prot – Rotational period in days.
- t – Temperature in K.
- d – Distance in parsecs.
- z – Metallicity (dex relative to solar).

1.3.5 Planet

Planetary parameters.

- m – Mass in solar masses.
- r – Radius in solar radii.
- p0 – Pressure at r (bars).
- porb – Orbital period in days.
- prot – Rotational period in days.
- Omega – Longitude of ascending node in degrees.
- ecc – Eccentricity.
- inc – Inclination in degrees. 90 is considered edge-on.
- b – Impact parameter.
- w – Longitude of periastron in degrees.
- a – Semi-major axis in AU.
- t0 – Time of transit in days.

1.3.6 taurex

Tau-REx specific options.

- csxdir – Directory containing molecular opacity data. This directory must contain a file or files for each molecule in the 3D fit.
- ciadir – Directory containing collision-induced absorption cross section files.
- wnlow – Minimum wavenumber to calculate radiative transfer.
- wnhigh – Maximum wavenumber to calculate radiative transfer.

1.4 Code Description

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.

1.4.1 2D Mode

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:

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

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 (likely constant and equal to unity, if the data are normalized), and s_{corr} is a constant term to correct for any errors in stellar normalization.

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):

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

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):

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

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.

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,

$$V(\theta, \phi, t) = L(\theta, \phi, t)S(\theta, \phi, t),$$

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 on angular distance from the point on the planet closest to the observer and the area of the discrete grid cell, integrated over the visible portion of the grid cell. L is then

$$L(\theta, \phi, t) = \int_{\theta_i}^{\theta_f} \int_{\phi_i}^{\phi_f} R_p^2 \cos^2\theta \cos\phi d\phi d\theta$$

where (θ_i, θ_f) is the range of visible θ and (ϕ_i, ϕ_f) is the range of visible ϕ for each grid cell.

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,

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

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

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

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. $\bar{\theta}$ is the average visible θ and $\bar{\phi}$ is the average visible ϕ.

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.

1.4.2 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.

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:

- Isobaric – Each 2D thermal map is placed at a single pressure for all grid cells. There is one free parameter, a log-pressure level, for each thermal map.
- Sinusoidal – Each 2D thermal map is placed according to a sinusoid, in both longitude and latitude. The longitudinal phase can vary. Functionally, the model is:

$$\log p(\theta, \phi) = a_1 + a_2 \cos\theta + a_3 \cos(\phi - a_4)$$

  where a_i are free parameters. There are four free parameters per thermal map.
- Flexible – Each visible grid cell of each thermal map has its own parameter for its pressure level. The number of free parameters depends on the latitude-longitude resolution of the 3D map.

Atmospheric Composition

The code also generates an atmospheric composition, as atomic and molecular abundances vs.pressure for each grid cell. ThERESA offers two schemes for calculating atmospheric composition:

- rate – Thermochemical abundances are computed analytically as needed.
- GGchem – The user supplies a file describing thermochemical equilibrium over a range of temperatures and pressures, which is then interpolated as needed.

There is no significant difference in runtime between the two. GGchem requires slightly more work by the user, but is valid over a larger range of temperatures and pressures. In theory, more complex schemes are possible, including options to fit to atmospheric composition.

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

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.

Contribution Function Fitting

ThERESA has an option to enforce some consistency in the 3D model by penalizing the goodness-of-fit based on how close the 2D thermal maps are placed, in pressure, to the pressures which contribute to the emergent flux at the wavelengths corresponding to each thermal map. This check is done for every observational filter and every visible grid cell. Without this check enable, ThERESA may find a "good" fit which is physically implausible. For example, a 2D map may be buried deep in the atmosphere, where it has no effect on the emergent spectrum.

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, which offers 3 sampling algorithms: Metropolis-Hastings random walk, Differential Evolutions Markov-chain Monte Carlo, and "snooker".

1.5 Scripts

This section contains some scripts that may be useful for your applications of the ThERESA code. They are not needed to run ThERESA.

1.5.1 synthlc

This script generates synthetic light curves, given a 3D description of an exoplanet atmosphere temperature structure (a GCM) and parameters for the planet, star, radiative transfer, and observation. Many of the parameters are similar to those used by ThERSA. Note that in order to run this code you need a 3D temperature structure in the right format and a set of opacities/CIAs for Tau-REx, which can be fetched from a number of locations (e.g., ExoTransmit, HITRAN/HITEMP). The tutorial (*Getting Started*) includes a script to fetch some of the ExoTransmit opacities.

To run synthlc, do the following:

```
cd scripts
./synthlc.py synthlc.cfg
```

If the configuration file is set up correctly, this will read the GCM temperature structure, calculate thermochemical equilibrium, run radiative transfer, and integrate over the planet at the observation times requested. It will produce time.txt, flux.txt, and ferr.txt, which can be used in ThERESA.

Options

- planetname – Name of the simulated planet.
- outdir – Where to store the output. Will be created if it does not exist.
- ms – Stellar mass in solar masses.
- rs – Stellar radius in solar radii.
- ts – Stellar temperature [K].
- ds – Distance to star [pc]. Has no effect at the moment.
- zs – Stellar metallicity [dex].
- mp – Planetary mass in Jupiter masses.
- rp – Planetary radius in Jupiter radii.
- ap – Planet semimajor axis [au].
- bp – Planet impact parameter.
- porb – Planet orbital period [days]
- prot – Planet rotational period [days]
- t0 – Planet time of transit [days]

- ecc – Planet eccentricity.
- inc – Planet inclination [deg].
- atmtype – Same as ThERESA. See *Configuration*.
- atmfile – Same as ThERESA. See *Configuration*.
- mols – Same as ThERESA. See *Configuration*.
- opacdir – Location of TauREx molecular opacities.
- ciadir – Location of TauREx CIA opacities.
- filtdir – Location of filter files.
- filters – List of filter file names in filtdir.
- phasestart – When to start the observation in orbital phase.
- phaseend – When to end the observation in orbital phase.
- dt – Length of one exposure or integration of the observation [s].
- noise – List of star-normalized noise estimates for each filter.
- necl – Number of stacked eclipses. Uncertainties scale as $\sqrt{\text{necl}}$.
- oom – Number of orders of magnitude of pressure change in the GCM.
- surfp – Surface pressure of the GCM.
- gcmfile – File describing GCM output.