Programming for Astronomy and Astrophysics

Assignment 2: Fitting TESS Exoplanet Transits with Python's Numerical Libraries

Deadline: Thursday 5th October at 23:59 – submitted via Canvas as a Jupyter notebook.

Goals of the assignment

By doing this assignment you will show that you can:

- Structure your code into functions which are easy to read and well-commented or explained using docstrings.
- Read data into numpy arrays and edit or manipulate the arrays as required to do the analysis.
- Use appropriate functions from the numpy and scipy libraries, to analyse time-series data.
- Write model functions, and fit simple models to data using curve-fit to obtain the model parameters.
- Plot the results you obtain, using clear labels on your plots.
- Set out your analysis in a clear and logical way in a Jupyter notebook.

Introduction

The two text files associated with this assignment contain time-series data for two stars with transiting exoplanets, observed by NASA's Transiting Exoplanet Survey Satellite (TESS), which was launched in April 2018. During its initial 2-year survey mission, TESS has used its array of wide-field cameras to monitor 200 000 of the brightest stars near the Earth to look for small dips in the light from the star that correspond to (exo)planets transiting the face of the star and blocking some of the light.

The two 'light curves' of TESS target stars provided here are given as time series with the first column showing the time in days (the values are given as Barycentred Julian Date – 2457000, although this is not relevant for this Assignment) and the second column showing the calibrated photoelectron count rates for the star (as electrons/s although again, the exact flux units are not relevant for this Assignment). The light curves each cover about a month and the individual data points correspond to 2-minute intervals. There are gaps of a few days in the middle of the observations due to observing constraints on the satellite.

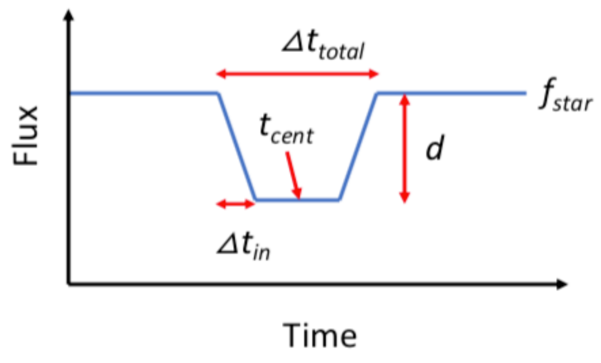
The main purpose of this assignment is to practice and demonstrate programming to solve a data analysis problem using Python's numerical libraries, numpy and scipy. **You should try to use these libraries throughout. Use loops and lists where it is really necessary, but not to do large numerical calculations, when you should be using arrays with the appropriate pre-compiled library functions.**

What you need to do

You should use a Jupyter Notebook to complete this assignment – you can write the functions in the first cell and use them to read in the data and make plots in subsequent cells. Be sure to include comments (#) or docstrings (```   ```) in the code to explain it.

You should make sure that you define your own functions for carrying out repeated tasks. But you can carry out your analysis (calling functions as necessary) in the notebook cells below the function definitions.

1. Read in the data using `genfromtxt` and 'clean' the data by removing rows from the resulting array which contain `nan` values (note that both the time and flux columns may contain these values). Plot your cleaned light curve.
2. Use the cleaned data to do the following analysis to help you confirm the period of the transiting exoplanet: Smooth the light curve by averaging each data point together with the two (or more – see what works) data points on either side (i.e. for smoothing with the 2 points on each side you measure the average of a 'sliding window' of five data points centered on each data point). Plot the smoothed light curve along with 'zoomed-in' segment of the light curve showing an eclipse. Use the full smoothed light curve and a periodogram (e.g., a Lomb-Scargle periodogram, see Lesson 17: https://philuttley.github.io/prog4aa/17-scipy/index.html) to determine the likely orbital period of the transiting exoplanet (plot the periodogram and identify the highest peak).
3. Finally, you will try to fit a model to the shape of the eclipse in the light curve, to constrain its parameters and measure a more accurate orbital period:
    a. Write a function which will calculate the following functional form for the light curve flux (e.g. count rate) when there is an eclipse, when given the time values:



    Your function should use the variables shown in the diagram above. Δt_{total} is the total duration of the eclipse (from start to finish), Δt_{in} is the ingress time, i.e. the time from the start of the eclipse until the maximum depth (d) of the eclipse is reached, t_{cent} is the time corresponding to the mid-point of the eclipse and f_{star} is the flux level of the star when not eclipsed. You can assume that the eclipse egress time (time of rising flux at the end of the eclipse) is the same as the ingress time.

    b. Use your function with a subset of the cleaned (but unsmoothed) light curve, to fit one of the eclipses using `scipy.optimize.curve_fit` (you should not include any error bars in the fit – it's fine to assume all the data errors are the same for the purpose of the Assignment) and obtain the fit parameters described above. Plot the data and the model together on the same plot.

The following c and d questions are ***optional*** 'stretch goals', in case you finish early and want an additional challenge:

c. Now modify your function to include the orbital period, i.e. the separation between successive eclipses, as an additional parameter, and fit multiple eclipses simultaneously to obtain a best-fit estimate of the orbital period.
d. Finally, use the fitted t_{cent} values of the eclipses to shift segments containing the eclipse so that they all have the same t_{cent} (i.e. they overlay one another), and plot the model on the same plot.

4. Try your method on one light curve eclipse first (whichever shows the clearest signal is probably the best one to start with), then apply it to the other light curve eclipses, adapting the starting arguments/parameter values for the fits as needed.

Hints and tips

- Some parts of the light curves may show additional variations in the flux of the star itself. These may be real stellar variability or related to calibration issues. If the light curve segments you choose to fit show this variability, consider first fitting a spline function to the parts on either side of the eclipse (excluding the eclipse itself) and subtract it from the light curve (so the intrinsic level of stellar flux becomes zero), before you fit your eclipse model.
- You can use conditional statements to build a 'piecewise' function with sudden changes in gradient.
- When fitting multiple eclipses together, you don't need to include all the parts of the light curve that are in-between the narrow eclipse segments. Your fits will go quicker if you only fit a light curve consisting of the eclipses and small regions around them.