# ASTR4260: Problem Set #3

Due: Wednesday, February 19

## Problem 1

In this problem, you will read in light curve data from the Kepler satellite and plot it.

(a) Download the light curve data associated with Kepler Object of Interest (KOI) 97, the host star of Kepler 7b, one of the first Kepler exoplanets detected (see the ApJ paper describing it - ApJL, 2010, 713, L140). If you wish, you may use a different KOI, but note that not all KOI's correspond to confirmed planets. As usual, this can be done a number of ways.

One nice way is to use the `kplr` python package (developed by an NYU graduate student), which is described at https://github.com/dfm/kplr/blob/master/docs/index.rst If you have pip working (which you should if you used the anaconda install recommended in problem set 0, or otherwise have a relatively recent version of python), then you can just use `pip install kplr`. To confirm successful installation (if you used anaconda), use `conda list`.

Once you have the package installed, you can use it to fetch the data from MAST, the online data repository, using something like the following code:[1]

```
import kplr
client = kplr.API()
koi = client.koi(97.01) # Find the target KOI.
lcs = koi.get_light_curves(short_cadence=False) # Get list of datasets.
f = lcs[0].open() # open the first light-curve dataset
hdu_data = f[1].data
time = hdu_data["time"]  # get the time of each observation
flux = hdu_data["sap_flux"] # get the flux
flux_err = hdu_data["sap_flux_err"] # get the error in the flux
f.close()
```

As an alternative, you could visit the MAST archive and download the data directly, at: archive.stsci.edu/kepler. This takes a bit of work as the files are in FITS format (which can be read using the `pyfits` package – more on this in a later problem set).

Finally, as a fall back, I have placed the data for this one object, titled `KOI97.01_1.out` in your github template repository. You can read it with the techniques discussed in class, or a special tool of numpy (loadtxt) that can directly read in columns of text numbers:

```
import numpy as np
time, flux, flux_err = np.loadtxt('KOI97.01_1.out', unpack=True)
```

However, you read it in, you should have three arrays: `time`, `flux` and `flux_err` for one set of observations of a single star observed by Kepler. The time of each observation is Coordinated Universal Time (UTC) - 2454833 (this arbitrary number is subtracted in order to reduce round-off errors and also to make it more convenient to deal with), and the flux is in counts/s, although we won't worry too much about the units of the flux since we're primarily interested in the *ratio* of fluxes (eclipsed to not eclipsed).

[1] Note that this requires an active internet connection. See the `kplr` documentation for more information about using the package.

(b) Plot the flux vs. time, including error bars using the `matplotlib` package[2] . Please label the axes accurately and include plot with submission. The `errorbar` function from matplotlib will be useful.

## Problem 2

In this problem, we want to compare the data to a theoretical light curve. To do this you will need to use the machinery from Problems 1 and 2. First, examine the data you have plotted above and find a section that looks like it corresponds to an eclipse. Then extract a section of the data corresponding to a single eclipse (include some time before and after the eclipse to show the baseline clearly). These are the data that we will try to fit with a theoretical eclipse curve.

(a) Begin by trying to fit the eclipse using the flux ratio code from problem set #1 with uniform stellar intensity. Make an estimate of the required parameters by eye; we'll get into statistical model fitting later in the course.

You will need to convert from time $t$ to $z$, which you can do with

$$z(t) = (t - t_0)/\tau \tag{1}$$

where $\tau$ and $t_0$ are constants: $\tau$ is related to the duration of the eclipse and $t_0$ is the time of maximal eclipse. Guess values for $p$, $\tau$ and $t_0$ and calculate $F(p, z(t))$ for the same $t_i$ values as the data for KOI 97 that you downloaded in problem 1. Experiment to find an eclipse shape that approximately matches the data. Generate a labelled plot and include it with your submission. If you are having trouble getting reasonable parameters, ask me for a hint.

(b) Use the code from problem set #2 with the realistic limb-darkening function

$$I(r) = 1 - (1 - \mu^{3/2}) \tag{2}$$

where $\mu = \cos\theta = (1 - r^2)^{1/2}$.

## Problem 3

There are many definitions of the habitable zone around a star, but typically they require liquid water on the surface of planets orbiting within it. A recent theoretical determination of the habitable zone range is given in Kopparapu et al (2013, ApJ 765, 131). They determine that the inner edge of the habitable zone is given in Astronomical Units (AU):

$$d = \left(\frac{L/L_\odot}{S_{\rm eff}}\right)^{1/2} \text{AU},$$

where $L/L_\odot = (T_{\rm eff}/5780\ \text{K})^4$ is the luminosity of the star in terms of its effective temperature $T_{\rm eff}$, $S_{\rm eff}$ is given approximately by

$$S_{\rm eff} = S_{\rm eff\odot} + aT_* + bT_*^2$$

and $T_* = T_{\rm eff} - 5780$ K. For the inner edge of the habitable zone, $S_{\rm eff\odot} = 1.014$, $a = 8.177 \times 10^{-5}$ and $b = 1.706 \times 10^{-9}$. One AU is the mean distance from the sun to the earth.

Using one of the root finders described in class, determine the value of $T_{\rm eff}$ for $d = 0.5$ AU (i.e. find $T_{\rm eff}$ for which $f(T_{\rm eff}) = d(T_{\rm eff}) - 0.5\ \text{AU} = 0$). This marks the point beyond which it is challenging to find exoplanets through the transit technique and so, if we want to find planets in the habitable zone using this technique, we are forced to focus on stars with $T_{\rm eff}$ values smaller than this. (Of course, this yields "habitable" planets so close to their stars that they risk being tidally locked and heavily irradiated by stellar activity, but that's a discussion for another time.)

[2] You may either create an image containing the plot or, better yet, include the image inline in your jupyter notebook – to do so, be sure to include `%matplotlib inline` at the top of your file)