ASTR 4260: Problem Set #7

Due: Wednesday, November 8, 2023

In this problem set, we're going to return to the Kepler data set from problem set #3 (Kepler 7b) and do a proper statistical treatment, rather than just fitting by eye. As before, you can use `lightkurve` to read the data from MAST. (Don't forget to activate your virtual environment if you used that before and to use the `jupyter` installed in that virtual environment. Ask me how I know.)

Problem 1

First, extract the data in the time range $261 < t < 262$ (in units of UTC-2454833 days). Include only data between these times, and be sure to get the time (t_i), the flux (F_i) and the uncertainty in the flux ($\sigma_{F,i}$). The numpy `where` command can be useful here, or the use of a mask, as we discussed in an earlier lecture. (You may also use numpy methods to compute the mean and standard deviation in this problem set.)

In order to compare to the predicted light curve, we will need to normalize the stellar flux, that is compute $F_i/\overline{F}$ where $\overline{F}$ is the mean unobscured stellar flux, excluding the transit. Last time I wasn't too concerned about how you did this, but now we need to be more careful. We will use a "two-sigma clipping" algorithm. To do this, compute the average and standard deviation of the extracted flux values – we'll call these $\overline{F}'$ and σ'. This average includes the transit so it is not exactly what we are looking for – to exclude the transit, remove the points for which $|F_i - \overline{F}'|/\sigma' > 2$, that is, those points which lie more than 2σ away from the mean. The remaining points should not include the transit, although usually we need to iterate this a few times: using the remaining points, recompute the mean and standard deviation ($\overline{F}''$ and σ'') and then repeat the procedure, again excluding the points which lie more than 2 sigma (using σ'') away from the new mean $\overline{F}''$. Repeat this five times and the resulting mean should be a pretty good estimate of the flux excluding the transit. From here on, normalize both F_i and $\sigma_{F,i}$ by this mean value.

As a first step in the model fitting, let's compute a measure of the fit for a given set of parameters. Use the ratio of obscured to unobscured flux computed in problem set #2. There is no constraint on how to compute the integrals for this problem – you may use your own routines, my routines, or `scipy.integrate` (scipy will probably be fastest). Use the following limb-darkening relation:

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

where $\mu = (1-r^2)^{1/2}$. (Note: I would like you to use this law, which is more simply expressed as $I(r) = \mu^{1/2}$ rather than the revised one I gave in Problem Set #3).

Let's take a guess at the parameters which fit this light curve: $p = 0.0775, \tau = 0.1, t_0 = 261.3$ (see Problem Set #2 for the meaning of these parameters). Compute χ^2 for this choice of parameters:

$$\chi^2 = \sum_i^N \left(\frac{F_i - F(t_i; p, \tau, t_0)}{\sigma_{F,i}} \right)^2, \tag{1}$$

where the sum is over the N points in the light curve (i.e. between $261 < t < 262$).

Problem 2

Is this a good fit or not? To answer this question, first plot the data and the predicted transit curve on top of each other. Before going any further, determine (by "eye") if you think the fit is good or not (no points off for guessing wrong here – I just want you to get some intuition into what a good fit looks like). Next, compute how likely this value of χ^2 is, assuming that the errors are normally distributed. This can be done with the incomplete Γ function, as was discussed in class. You will need to know the number of degrees of freedom $\nu = N - M$, where N is the number of data points and M is the number of fitted parameters (in this case, take $M = 3$). The probability of getting this χ^2 by chance is known as the p-value (not to be confused with the normalized planet radius p). Is it a good fit?

Problem 3

For the next step, we are going to allow one parameter to vary. Let's take the transit width parameter τ and vary it between 0.08 and 0.13, while keeping the other parameters (p and t_0) constant. Find the minimum in χ^2 and report this best fitting τ value. Again, plot the result and determine the corresponding p-value for the fit. Finally, determine the one-sigma uncertainty in this parameter from the chi-squared plot.

Problem 4 (graduates)

So far, we have only varied one parameter at a time. Now, allow all three parameters (p, τ, t_0) to vary and find (a) the best fit parameter set, and (b) an estimate of the uncertainty of the parameters. There are many ways to do this – you could do a brute force search of the three-dimensional parameter space, you could use some minimization routine to find the minimum of χ^2, or you could use a Markov chain Monte Carlo (MCMC) technique—the python `emcee` package is a particularly good choice (if you use this, you can plot the results in a standard format using the python package `corner`.) Note: you do *not* need to write your own algorithm, just find, understand, and apply an existing module to complete this problem. If you use `emcee`, note that the required probability $\ln P = -(1/2)\chi^2$ for a normal distribution.