IW Checkpoint Report, Spring '14

Michael A. Thomas, matthree@

March 28, 2014

Progress To Date

The goal of my project is to develop tools in Python for fitting data for the light intensity over time (called "lightcurve") of a star to a model allowing for one or more circular "starspots" which are darker than the rest of the star's photosphere, with a particular focus on the lightcurves measured by NASA's Kepler observatory. To this end, I have implemented code for generating a lightcurve from a given spot model based on the equations of Eker (1994) and code previously written in IDL by Walkowicz et al. (2012) Using this function with numerical optimization functions provided by SciPy, I have been successful at fitting the model parameters (stellar inclination and spot latitude, longitude, and radius) for synthetically generated single spot lightcurves using the LevenbergMarquardt damped least-squares algorithm. In order to deal with the possibility of numerous local minima in the parameter space, the optimization is run from a large number of different initial parameter values, and then the best fit is chosen from among the minima found. For testing, synthetic lightcurves are generated with parameters drawn uniformly at random from the parameter space. Occasionally, parameters are generated which place the spot completely out of view (e.g. a spot at very low latitude on a star with significant inclination) thus giving a constant lightcurve; these are caught automatically and new parameters generated. In the noiseless case, the fitting has been extremely successful ($\approx$100%) at generating almost exact matches to the original parameters (i.e. with differences only on the order of the machine precision). I have also tested the fitting algorithm using synthetic lightcurves including gaussian noise. In this case, the fits have still generally been quite good, but there are some inaccuracies due to the partial degeneracies between the radius, latitude, and inclination parameters.

Current Difficulties

One major current issue is the runtime of the fitting routine, which is typically 5 to 10 seconds per lightcurve. This is problematic for large scale testing and accuracy analysis of the fitting algorithm, as well as for our ultimate goal of fitting many thousands of Kepler lightcurves. Furthermore, these are only single spot fits, but nearly all real lightcurves are expected to display multiple spots. The parameter space - and thus the expected time to explore it - grows exponentially as more spots are added. The long current runtime is largely due to repeatedly running the fitting from a large number of different initial parameter values, as explained above. Instead of running each of these fits with tight convergence requirements and a large maximum number of steps, I am currently working on improving the method by initially running the L-M algorithm for only a small number of steps from each of the initial positions. This list of updated positions will then have redundancies removed (i.e. if more than one parameter set has already come to almost the same position, there is no point running the full L-M algorithm on each one) and a smaller number of the updated parameter sets with the best current fit values will be chosen and used as the initial values for running the L-M algorithm to convergence. This should significantly improve the runtime, hopefully without sacrificing accuracy of the final fits. Another option is to experiment with alternatives to the L-M algorithms in hopes of finding one with better convergence properties. SciPy provides a number of other optimization methods, and switching between them should be fairly easy now that the framework is written.

The other current issue is the partial parameter degeneracy in the case with noise. This was something of a known issue (discussed in Walkowicz et al. 2012) and there is not a lot can be done about it. Fortunately, the real Kepler data itself actually has quite low levels of noise. Additionally, the degeneracy with the stellar inclination should actually be less severe as additional spots are added, because each spot has a separate radius and latitude, but the inclination must remain fixed across all spots. Still, this highlights the importance of developing some method for gauging the uncertainty of the fitted parameters (discussed below).

Next Steps

The next major task is to extend the fitting to models with multiple spots. There are two main ways to do this. The first is, for a known number of spots, to simply run the L-M algorithm (or another optimization function) to optimize a model with a number of sets of spot parameters. Since this would fit the spots simultaneously, it should hypothetically provide the best possible fits; however, it has the disadvantage of needing to know the number of spots ahead of time. Furthermore, as mentioned above, the size of the parameter space increases exponentially with the number of parameters/spots, and thus so does the number of starting points needed to deal with the risk of local minima and the time to run the algorithm. An alternative is to fit spots sequentially by iteratively fitting a single spot and then subtracting its effect off from the lightcurve before fitting the next spot. This allows being more flexible with the number of spots, for example by letting the algorithm continue adding spots until the next spot fails to increase the goodness of fit by more than some threshold. This is expected to work well in the case where the spots are of distinct sizes, such that the biggest spot (having the most impact on the lightcurve) will be fit first, and so on. In the case of a number of similarly sized spots, however, it is expected to potentially perform rather poorly. In both cases, constraints must also be imposed to prevent spots from overlapping. The optimal method is likely to be some combination of these two, such as sequentially fitting a number of spots, and then doing a simultaneous fit using the sequential fit as the initial parameter values.

Another goal, as mentioned above, is to find a way of determining estimates for the uncertainties of the fitted model parameters. One way of doing this is by using a Markov Chain Monte Carlo algorithm, such as described by Croll (2006). There is also a Python package `lmfit` that I am looking into that is supposed to be able to propagate uncertainties through the L-M least squares algorithm. Here, it would be nice to be able to characterize both the uncertainty of which local minima characterizes the real parameters, as well as what the uncertainties on those parameters are around a given minima. The MCMC method should be able to handle both of these questions simultaneously, whereas `lmfit` will only give the latter; The former could perhaps be estimated using a simple exploration of different initial values around the parameter space, as the current fitting algorithm already uses, though it is not entirely clear how to represent that uncertainty, and, as discussed above, more initial positions means longer runtime.

Finally, once the above considerations are working, the ultimate goal would be fit real Kepler lightcurves to explore trends in the appearance of spots on real stars. It is unlikely that there will be time for much analysis before the end of the semester, but I would hope to least try fits on several real lightcurves, particularly those where some or all parameter values have been estimated through other methods, as a final proof of concept before declaring success.

References

1) Croll 2006, Markov Chain Monte Carlo methods applied to photometric spot modeling, *Pub. Ast. Soc. of the Pacific*

2) Eker 1994, "Modeling light curves of spotted stars," *ApJ*

3) Walkowicz et al. 2012, "The information content in analytic spot models of broadband precision lightcurves," *ApJ Supplement Series*