Cheetah: Modeling Starspots from Kepler Lightcurves

Michael A. Thomas
advised by Adam Finkelstein and Lucianne Walkowicz

The phenomenon of relatively cool, darker areas of a star's photosphere – called starspots – is closely linked with the star's internal magnetic field activity. These magnetic fields can locally inhibit the star's internal convection, preventing heat flow towards the surface and thus causing these comparatively cooler areas. Full models describing stellar magnetic fields are as yet incomplete, but their development could be aided by constraints provided by an understanding of spot characteristics. NASA's Kepler space observatory continually measures the brightness of over 100,000 stars, and periodic variation in these photometric "lightcurves" is commonly attributed to spots rotating into and out of view. As yet, however, no large-scale modeling efforts have been focused on determining spot properties from this data. Here, we are attempting to develop computer algorithms for fitting spot patterns from such lightcurves. Although optimization of algorithm parameters and investigation of performance characteristics is ongoing, preliminary results from testing against artificially generated lightcurves with known spot parameters have been promising in both single and double spot cases.

Keywords: starspots; stellar dynamo; stars: magnetic activity; techniques: photometric; methods: data analysis; methods: numerical optimization

Introduction

Stars are known to generate strong internal magnetic fields due to the convection of electrically conducting plasma. At locations where these magnetic fields penetrate the star's photosphere, they inhibit that convection, preventing heat from flowing towards the surface and thus generating cooler areas. These areas emit less light, and thus appear as dark "starspots" against the brighter surrounding photosphere (see Figure 1a and 1c). The processes governing these magnetic fields are described by what is commonly referred to as the stellar dynamo theory, but the theory is incomplete and a full model has yet to be developed. This is due in part to a lack of data, since few stars are close enough to take relevant measurements. Because of the known connection between surface features such as starspots and a star's internal magnetic field activity, however, data on the appearance of starspots could provide constraints on theoretical models of the stellar dynamo. That is, a full model of such field activity could be used to predict various spot properties, and these could be compared to observed spot patterns as a check on the validity of the model. (Berdyugina 2005)

Unfortunately, little is known about starspots on stars besides the Sun. Most other stars are too far away to be imaged with any spatial resolution, so we generally cannot "see" spots on other stars. There has been some success using Doppler imaging to study spots on a small number of stars. For example, a Doppler image of the main sequence solar-analogue EK Draconis reveals cooler "spots" which are much larger and occur in a wider variety of locations than the spots we typically see on the Sun (see Figure 1b). This suggests the possibility that spots on other stars are not necessarily similar to those on Sun, and implies the need to look further than the Sun for a general theory of starspots. The effectiveness of Doppler imaging is however limited to stars with certain properties (namely those rotating at a fast enough rate and of high enough temperature) and the process is too time consuming to admit very large scale studies. (Strassmeier 2009) Fortunately, another option presents itself in the form of data from NASA's Kepler space observatory. Kepler continually measures the overall brightness of over 100,000 stars. There has been much success in recent years in determining the existence of exoplanets orbiting these stars by analyzing periodic sharp dips in their brightness as planets pass in front of the stars. These measurements, tracking the total amount of light over time and thus called lightcurves, also show more gradual periodic fluctuations, and these are thought to be due to starspots rotating in and out of view. It should thus be possible to determine information about the spot patterns present on a large number of stars from these lightcurves. As yet, however, no large scale modeling efforts that we know of have been undertaken. Here, we attempt to solve several issues with determining spot pattern information from such lightcurves to enable a large scale survey of spot characteristics from the Kepler dataset.

Several analytical models, generally assuming circular spots, have been developed for calculating the expected shape of a lightcurve for a star with a known spot pattern. The model we use here was developed by Eker, but all of the well-known the analytical models are essentially equivalent. (Eker 1994) Several previous attempts have been made at developing computer programs for fitting spot parameters from a given lightcurve shape. Ribárik et al. developed the SpotModeL program, using Levenberg-Marquardt nonlinear least squares to fit a similar analytic model to lightcurve data. (Ribárik 2002) Croll later expanded this into the StarSpotz program, which notably also includes the option to use a Markov Chain Monte Carlo method for assessing the model goodness of fit. (Croll 2006) Several issues, however, make these unsuitable for a large scale study of the Kepler lightcurves. Both focus largely on model visualization over speed, making them difficult to apply to a large number of lightcurves. Furthermore, in both cases, they are written in a variety of programming and scripting languages, none of which are widely used or understood in the astrophysics community, making them difficult to modify or apply to the specific task at hand. Finally, to our knowledge, no performance or accuracy results were ever published for either program. Here, we seek to develop efficient algorithms for fitting spot parameters from given lightcurves which can be scaled up to a large scale study of the Kepler dataset, and implement these algorithms in Python, which is one of the most prominent languages used today for computational data analysis. Additionally, we look to evaluate the method's accuracy by comparing fits to known parameter values for a set of synthetically generated lightcurves.

Analytical Model

We base our work on the analytical model for computing the expected lightcurve of a star with a given spot pattern developed in (Eker 1994). The model assumes circular spots, characterized simply by latitude, longitude, and radius parameters, as well as allowing for the star's rotational pole to have some inclination angle towards the point of measurement. The general idea behind the model is to calculate the projected spot areas using geometry, and then integrate the specific intensity of light over the star's visible area. This essentially gives the amount of light "lost" due to spot. Since spots are assumed to be non-overlapping, multi-spot curves can be formed by linear combination of single-spot curves. Specifically, each curve is measured as a fraction of the unspotted intensity, i.e. a completely unspotted lightcurve would have a constant value of 1. Thus, given two lightcurves $l_1(t)$ and $l_2(t)$, their combined intensity is given by $l_c(t) = l_1(t) + l_2(t) - 1$. In the analytic case, the maximum of this curve can be < 1 if there are spots visible at all rotational phases. In experimental data, however, we have no way of knowing unspotted intensity of star, so we generally normalize the maximum of each lightcurve to equal 1. Thus, we tend to underestimate spot coverage.

For simplicity of the model, we have made several simplifying assumptions. In addition to the parameters mentioned above, the model also includes a parameter describing the contrast ratio between the spot and the surrounding photosphere, as well as coefficients allowing for a limb darkening effect on the star. These, however, are effectively degenerate with spot radius (a smaller, darker spot is essentially equivalent to a larger, lighter one) and thus would be difficult to fit. Therefore, we arbitrarily fix them at approximately their solar values. Furthermore, being a fluid body, a true star exhibits differential rotation, such that spots at different latitudes would have somewhat different rotational periods. In fitting spots to a lightcurve, one would need to determine each individual spot period. Here, we assume all spots have the same period, which is already known. `lcspot.py` provides straightforward implementation of Eker equations, based on an earlier version written in IDL by Walkowicz et al. (2012)

Single Spot Fitting

The main single spot problem we consider involves fitting the stellar inclination as well as the longitude, latitude, and radius of a single spot based on a given lightcurve, which consists of (potentially noisy) intensity data taken at specified phase values. The core of our process is the Levenberg-Marquardt nonlinear least squares algorithm, which we use to minimize the sum squared error between the model and the lightcurve that we wish to fit. In particular, we use the implementation of the L-M algorithm provided by the `leastsq` function of the `scipy.optimize` package. The L-M algorithm (like essentially any numerical optimization method) only finds local minima around the region specified by the starting parameter values. Unfortunately, changes to a number of the model parameters (in particular, inclination, latitude, and radius) can have similar effects on the resulting lightcurve, and thus different sets of values can produce very similar (though in principle not identical) lightcurves. (A much more detailed discussion of these parameter "degeneracies" can be found in Walkowicz et al. 2012.) This implies the potential for numerous local minima throughout the parameter space. In order to find a global minimum, we must therefore run the optimization using a variety of starting points throughout the parameter space. Thus far, we have had success simply using an evenly-spaced grid of starting points, though one could also consider using randomly selected values.

In order to reduce the time required for a single fit, which must be reasonably fast for large-scale modeling of many lightcurves to be attempted, we initially run the L-M algorithm on each of the points in the full grid for only a small number of iterations. We then cluster the resulting points using the K-means clustering algorithm and keep only the best (lowest SSE) point from each cluster. The goal of this operation is to avoid running the L-M algorithm for a large number of steps on points in the parameter space which are likely to converge to the same local minima (those already close together) while still keeping a diversity of points throughout the space. Finally, we run the L-M algorithm to convergence on each of the points in this reduced list. For the single spot results, given below, we consider only the final point with the best SSE. In testing using artificially generated lightcurves with known parameter values, however, particularly those including some random noise added to the intensity values, we have noticed that although a point close to the "true" values is almost always present in the final list, it is not always the one with the absolute best SSE. Since we use this single spot fitting algorithm as a component step in the multi-spot algorithms (discussed below) we would like to avoid discarding the "correct" parameter value set. In this case, we keep the parameter set with the lowest SSE and any other points with SSE's within a fixed multiple of the lowest (known as the threshold ratio, see discussion of algorithm parameters, below).

We also use this same process for fitting a single spot in the case with a given fixed stellar inclination as well as target lightcurve data. This arises in particular in the multi-spot fitting algorithms, described below. `lcsinglefit.py` provides the single-spot fitting algorithms.

Multi-spot Fitting

The most straightforward way to fit lightcurves with multiple spots would be to simply use an algorithm similar to the single-spot algorithm described above, simultaneously fitting all spot parameters. Unfortunately, the grid of starting points for the L-M optimization grows exponentially with the number of dimensions. For example, a grid spanning 5 points in each dimension contains 625 points for a single-spot fit (4 dimensions: inclination plus 3 spot parameters) but 78,125 points for a double-spot fit (7 dimensions). Thus, this method quickly becomes intractable for multi-spot fitting. Instead, we initially treat the lightcurve as single-spot curve, fitting it as such using the algorithm given above. Since multi-spot curves are simply linear combinations of the single-spot curves corresponding to each spot. We can fit additional spots by simply separating out the previously fitted lightcurve from the given data curve and fitting the residual again as a single spot curve. We do this for each set of spot parameters kept from the initially fit (i.e. within the threshold ratio of the best fit, as described above) and spots after the first, we of course must hold the stellar inclination fixed to its previously fitted value. Finally, we take the list of potential fits generated by this process and use it as the initial set of possible points for running simultaneous fits of all spots (and the inclination) using the

L-M algorithm, and select the best fit. This uses a much smaller list of starting points than a full grid, but since they have already been selected by a fitting process, we hope that at least some are in the vicinity of the true parameters.

For double-spot fitting, this seems to work reasonably well (see Results, below). Unfortunately but unsurprisingly, however, we notice that as more spots are added, the sequential fits of the latter spots get worse and worse. Thus, we have developed what we refer to as the "ratchet" method of multi-spot fitting. As before, we fit the first two spots sequentially, and then perform a simultaneous double-spot fit. For each additional spot, we then perform an additional sequential single-spot fit of the residual lightcurve, followed by a new simultaneous fit of all spots thus far, ultimately selecting the best fit parameter set from the final simultaneous fitting. This method has not yet been tested enough to draw significant conclusions, but the results from several triple-spot fits have shown some promise. `lcmultifit.py` provides both of these multi-spot fitting algorithms.

Algorithm Parameters

There are a number of adjustable parameters involved in the described fitting algorithms, including: the set of initial points for the L-M algorithm, the initial number of L-M iterations before clustering in the single-spot fitting process, the number of clusters to create, and the threshold ratio for keeping additional fits with SSEs somewhat higher than the lowest. A full investigation of algorithm performance in terms of both accuracy and speed using different settings of these values is ongoing, particularly with respect to varying levels of noise in the lightcurve intensity data. Here, we simply give preliminary results showing proof-of-concept functionality of the algorithms with arbitrarily chosen parameter values: 20 initial L-M iterations, 30 clusters, and a threshold ratio of 2. For starting points of the L-M optimization in the single-spot algorithm, we use an evenly-spaced grid of points, spanning 5 possible values in each dimension. This comes to $5^4 = 625$ starting points in the variable inclination case, and 125 in the fixed inclination case.

Results

In order to test algorithm performance, we use synthetic lightcurves with randomly generated known inclination and spot parameters. To generate a single-spot curve, we simply choose uniformly distributed random values from the valid ranges for stellar inclination ($0° - 90°$) and spot longitude ($0° - 360°$) and latitude ($-90° - 90°$). For spot radius in the lightcurves considered here, we choose a uniform random value in the range ($10° - 25°$) as smaller spots and multi-spot curves with too much disparity between spot sizes are more difficult to fit. Finally, we test to make sure that there is a significant amount of variation in the generated lightcurve, as configurations of the inclination and spot parameters where the spot never rotates into or out of view are possible, and would obviously be unfittable. For multi-spot curves, we simply add spots as above, checking to make sure that each new spot does not overlap with any of the previously added spots. Strictly speaking, this process potentially makes some configurations of spot parameters more likely than others. For small numbers of spots of the sizes given above, however, collisions between spots in generation are rare, and thus the spot parameters of the generated lightcurves may still be considered to be approximately uniformly distributed. Plans for more evaluation of algorithm performance given different lightcurve parameters are described below. For testing with noise, the range of intensities of the generated lightcurve is calculated, and then random noise terms are generated and added to each intensity measurement, distributed gaussian with mean zero and standard deviation proportional to the calculated range. The constant of proportionality is termed the noise factor, and is equal to .02 in the single-spot tests and .01 in the double-spot tests given below. `lcgenerate.py` provides the synthetic lightcurve generation methods.

For evaluating algorithm accuracy, we need a method of comparing the fit lightcurve parameters with the known values. The scales of each parameter, however, are not directly comparable; for example, a $1°$ error in the radius is much worse than a $1°$ error in the longitude. Using percent errors would be the standard solution to this, but percentages are problematic for the location parameters. For example, a 1° error in the longitude should be considered the same no matter whether the true longitude is 5° or 355°. Thus, to compare among different parameters, we instead normalize each by a sort of "average" magnitude. Since the parameters of our synthetic lightcurves are drawn from uniform distributions, we take this to be equal to half the range of each parameter: 45° for the inclination, 180° for the longitude, 90° for the latitude, and 7.5° for the radius. Note that these are not the mean values for each parameter, but rather what the mean values would be if the minimum value of each parameter was shifted to zero. We can then average the errors among the various parameters to asses the goodness of a certain fit. For the multi-spot fits, however, we separate out the inclination error, as we gain information about the inclination from each spot, and therefore expect its error to be lower than that of the parameters for any single spot. For display purposes, we take the logarithm of the mean errors and bin them to create frequency histograms. Results are shown in Figures 2-6 for noisy single- and both noiseless and noisy double-spot fits. (Noiseless single spot fits have an extremely low rate of error, and are thus not shown.)

Further Work

Our immediate priority for further work is experimentation with the algorithm parameter values to attain a more thorough understanding of their effects on performance characteristics, as we would of course like to minimize the computational runtime while still maintaining a high degree of accuracy. Alternative numerical optimization methods to the L-M algorithm could also be swapped into the code relatively easily to determine whether some are better suited to dealing with the shape of the sum squared error function in our specific case and thus might display faster convergence to the minima. We would also like to examine performance characteristics on lightcurves with different amounts of noise, including more realistic noise levels determined from analysis of real lightcurves. Finally, we should evaluate whether and how the accuracy of the algorithm varies depending on the actual spot parameters. For example, one might expect the algorithm to most accurately identify the largest of a set of spots, or to perform better on spots closer to the equator than those at extreme latitudes, which may never fully rotate into or out of view. Experimenting with these possibilities will give us a better idea of how much confidence we can have in the fit parameters for an individual lightcurve, which will also be important in interpreting the results and detecting trends in spot characteristics in surveys across many lightcurves.

In order to apply our algorithm to real lightcurves, we also need to develop methods of determining the rotational period of spots, which we have thus far taken to be known, as well as incorporate the possibility of differential rotation. For certain stars, stellar inclination has been estimated by other methods, and for some, doppler imaging has revealed other pieces of spot information. We plan to test our model on such stars before attempting a larger-scale study of stars with completely unknown parameters.

References

1. Berdyugina, S.V. 2005. Starspots: a key to the stellar dynamo. Living Reviews in Solar Physics.
2. Croll, B. 2006. Markov chain monte carlo methods applied to photometric spot modeling. Publications of the Astronomical Society of the Pacific.
3. Eker, Z. 1994. Modeling light curves of spotted stars. The Astrophysical Journal.
4. Ribárik, G., Oláh, K., and Strassmeier, K.G. 2002. A new computer code for time-series photometric spot modelling. 1st Potsdam Thinkshop Poster Proceedings.
5. Strassmeier, K.G. 2009. Starspots. Astronomy and Astrophysics Review.
6. Walkowicz, L.M., Basri, G.S., and Valenti, J.A. 2012. The information content in analytic spot models of broadband precision lightcurves. The Astrophysical Journal Supplement Series.

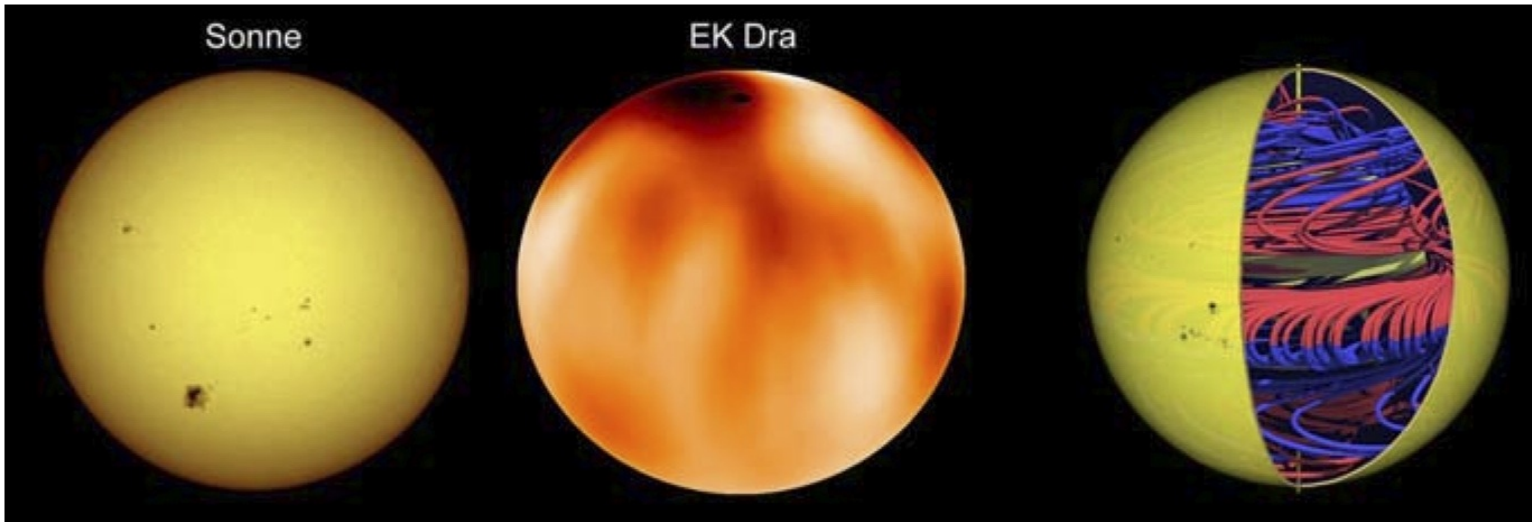

Figure 1. from Strassmeier 2009, “Starspots,” *Astron Astrophys Rev*

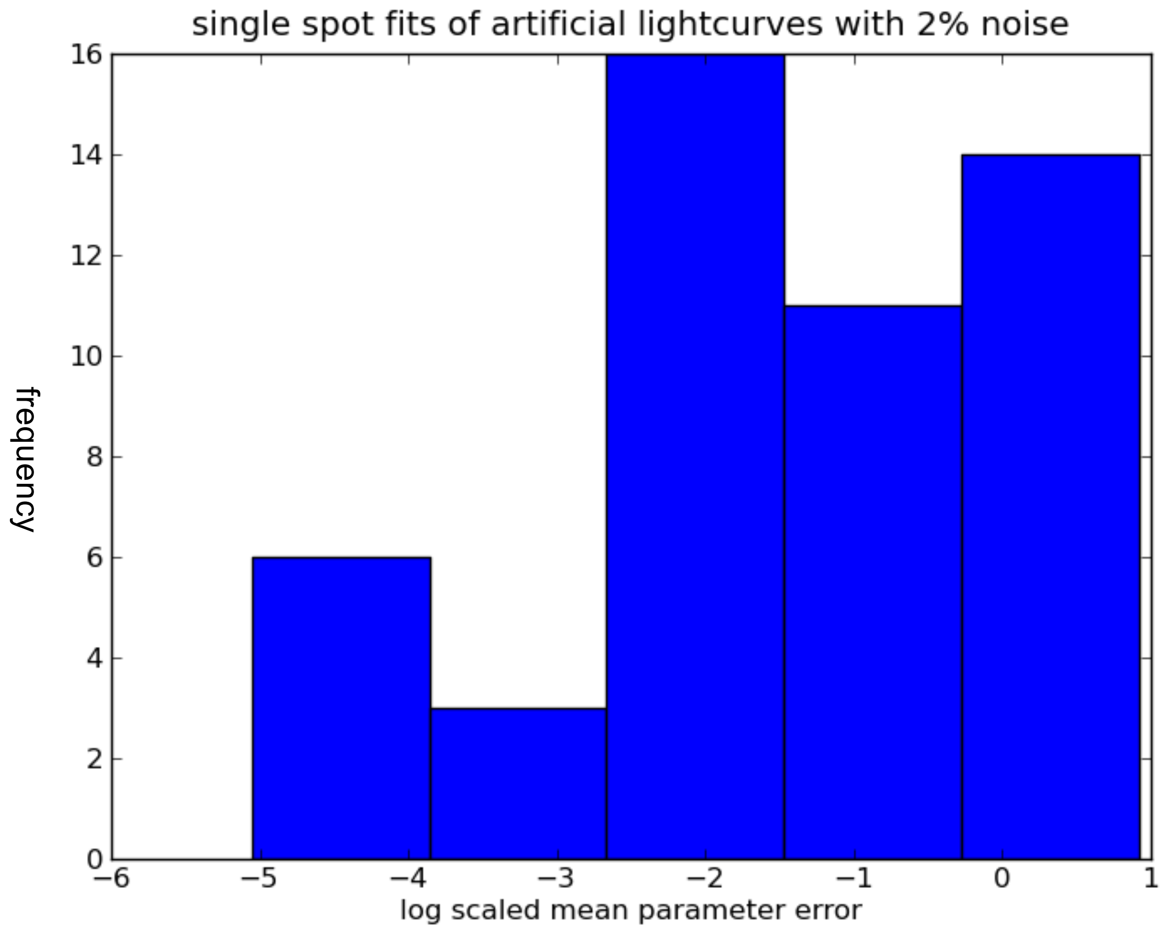

Figure 2

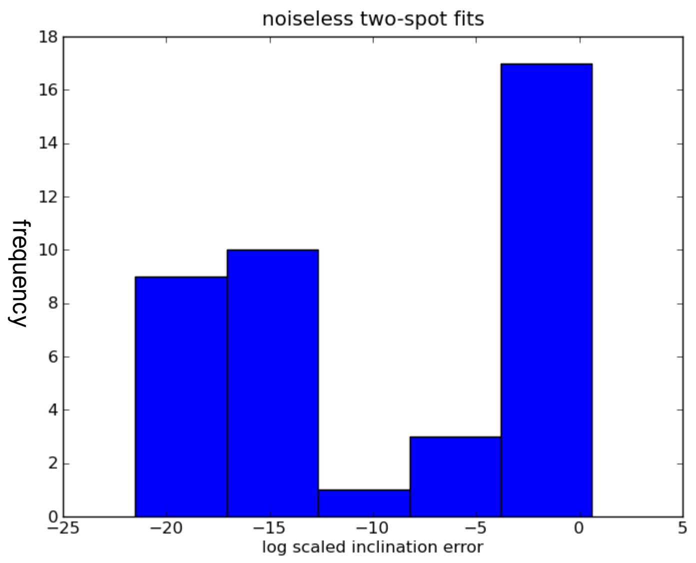

Figure 3

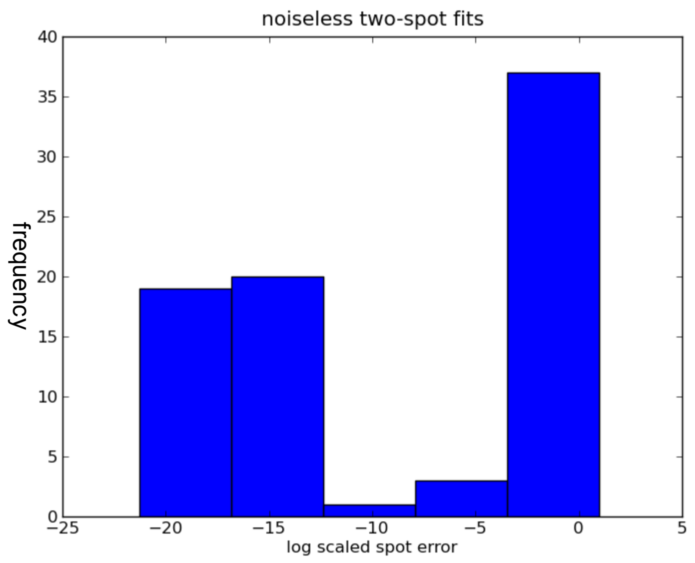

Figure 4

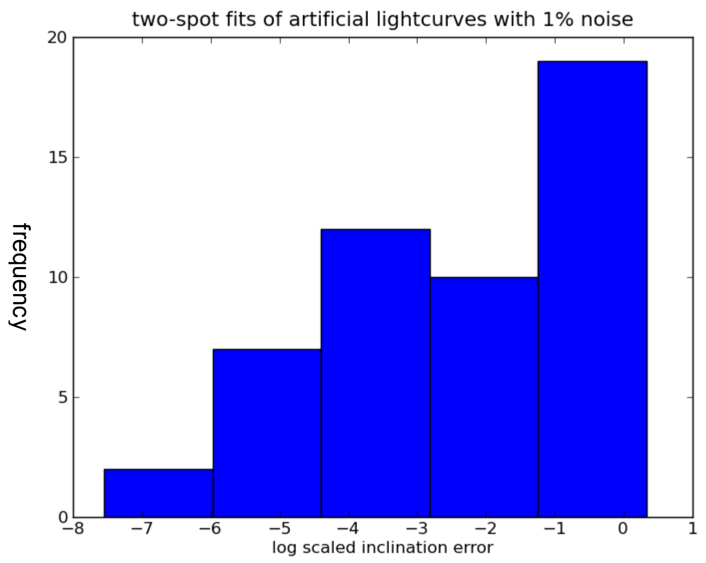

Figure 5

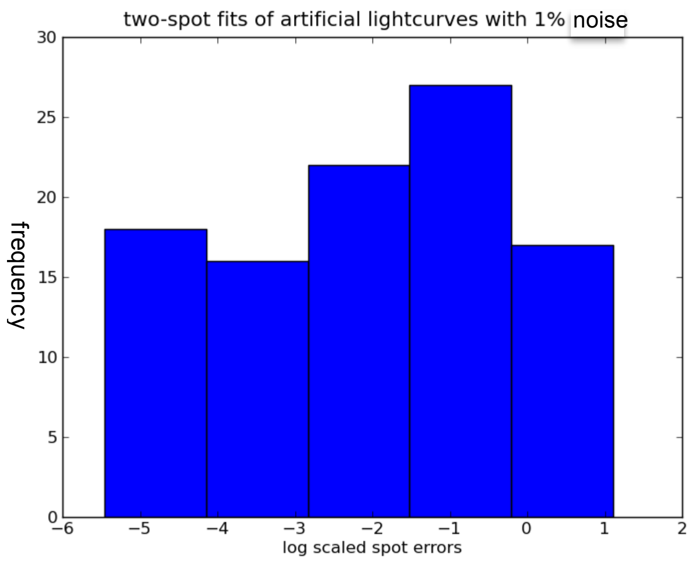

Figure 6