Magnetic Star Spots on the Red Giant Star KOI-1786

Thesis submitted in partial fulfillment of the requirement for Honors in Physics

Katelin Crabtree

Advisor, Associate Professor Leslie Hebb, Ph.D.
April 19, 2022

Abstract

KOI-1786 is an eclipsing binary system discovered by Kepler which consists of a red giant star primary and an M-dwarf companion in a 24.68 day eccentric orbit. We present the analysis of the star spot crossing features present in four years of transit light curves. Our results show a very active red giant primary despite its slow rotation period of approximately 66 days[20]. This level of activity in a binary with a highly active evolved companion is consistent with a long period RSCVn type system[22]. We find multiple spots features in every transit, as well as two distinct features which must be darker than typical sunspot umbras, indicating the presence of several large spots with a contrast of approximately 0.05 on the surface of the giant primary.

Acknowledgements

This work was supported by NSF Grant AST1909682.

Special thanks to Professor Leslie Hebb, my research advisor for both the summer and through this project.

This research made use of Lightkurve, a Python package for Kepler and TESS data analysis[18], as well as several other data analysis python packages such as numpy [11], BATMAN [16], matplotlib [14], and Pandas[19].

This research has made use of the NASA Exoplanet Archive, which is operated by the California Institute of Technology, under contract with the National Aeronautics and Space Administration under the Exoplanet Exploration Program, as well as the Vizier Database [24], and has made use of the SIMBAD database, operated at CDS, Strasbourg, France [33].

Contents

1 Introduction

1.1 Sun activity

The sun's magnetic field is approximately poloidal, or has 2 poles at times of low activity, and is toroidal during times of activity due to its dynamic shift over the course of the solar cycle[3]. One of the simplest analogies for the poloidal structure is to a bar magnet, which has simply a North and South pole with rounded field lines connecting them. The strength of this poloidal field is approximately 10 Gauss on the sun [17]. **Figure 1** shows the shape and direction of the field lines in this case. They curve out of the top of the North end and flow into the South end of the magnet. This is, however, the simplest case of the sun's magnetic field, and during times of maximum activity we know the field to be much more complicated than this.

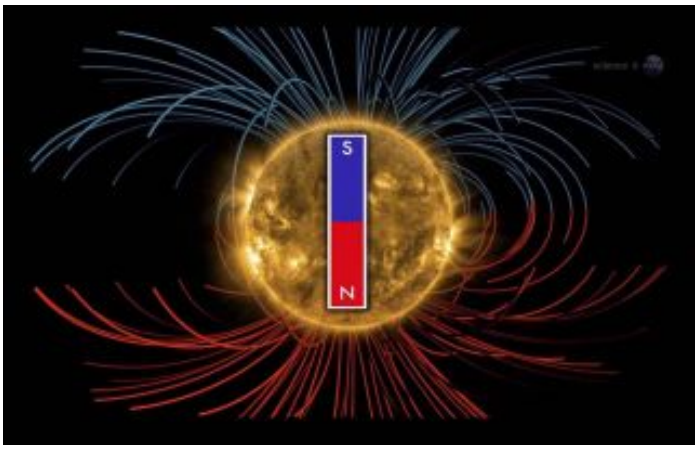


Figure 1: Image of the magnetic field of the sun during a low activity period.The bar magnet is superimposed to show the similarity of the field to that of a simple bar magnet. [29]

Ignoring small scale fields in spots and other activity related to the magnetic field, the simplest model of the shifting field of the sun over time is given by dynamo theory[3]. While there is some incredibly complex and dense theory of magneto-hydro-dynamics underlying this theory, a knowledge of the underlying theory in its conceptual form is all that is needed for this discussion, and any more is outside the scope and intent of this paper. For the general idea, it suffices to imagine that we can take a snapshot of the sun's magnetic field when it is as close to polar as possible. Now, imagine the field lines (like the ones in the bar magnet) are effectively stuck in the exact plasma on the surface they cross over [4][3]. The sun's rotation is differential, and it is faster at the equator than it is at the poles. Thus, over time, these frozen-in field lines get coiled up and tangled, until they eventually get so tangled there is a global pole flip induced in part by the migration of the sun spots

(an indicator of strong local fields). This reverts the sun back to a poloidal structure, but where the north pole was before the flip is not in the south pole and vice versa. This is known as Dynamo theory (**Figure 2**), and is the leading theory modeling the sun's magnetic field and activity over time though this cyclic tangling and pole flip [3][5].

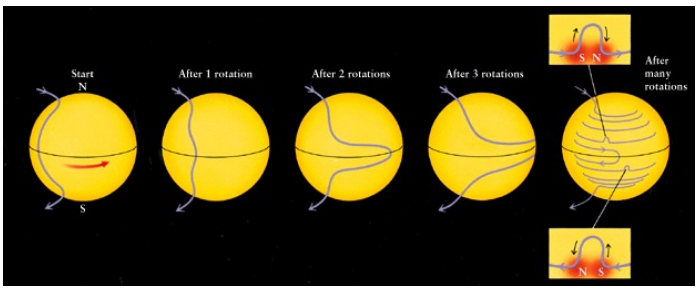


Figure 2: Image of the model of dynamo theory. The blue lines represent the magnetic field lines of the sun. Note the increasingly toroidal structure of the lines and the appearance of spots as local magnetic fields towards the end of the diagram. [10]

Now, this magnetic flip and its associated tangling of fields is not without consequence, and in fact it is believed that the activity of the sun is directly correlated with how "tangled" these field lines are. Dynamo theory, at heart, aims to model how the correlation of shifting magnetic fields relate to the activity of the sun[4]. At times of the sun's field's poloidal structure, we see a minimum of activity, and we see this activity increase with the tangling of the field line, including spots emerging around 30 degrees and slowly merging down towards the equator until the pole flip [4]. We see also a correlation in the amount of flares and coronal mass ejections (CMEs) during this time as well, suggesting the majority of the sun's activity is directly correlated to shifting magnetic fields[1][4].

The resulting activity markers for the sun include sun spots, flares, and CMEs. All of these are activity markers strongly associated with the turbulent magnetic fields, and all are affected by this dynamo action of the sun in different ways.

1.1.1 Sun Spots

Sun spots are parts of the sun that are cooler, and appear to be darker in color (linked to the cooler temperature) and appear exactly as they sound: as spots on the surface of the sun. They are one of the first recorded indicators of magnetic activity, as ancient civilizations have some accounts of sun spots. They can be large enough to be easily observed without a telescope at times, a method advised against unless you happen to posses a welding helmet, and there are numerous accounts of this phenomenon throughout history.[3].

These spots, along with their associated activity, also have a cooler temperature and their own unique structures. Spots not only emerge as clusters, and also themselves have a darker umbra towards the center and lighter penumbra around the outside perimeter of the spot itself [2], [3]. This structure can be seen in **Figure 3**. Spots will on average have a temperature of 4000K, lower compared to the 5800K average surface temperature of the sun[3]. They last on the order of months at a time, ranging from one to several months, and are physical manifestations of the turbulent magnetic field of the sun. Their sizes range from smaller than the radius of the earth to the radius of Jupiter[3].

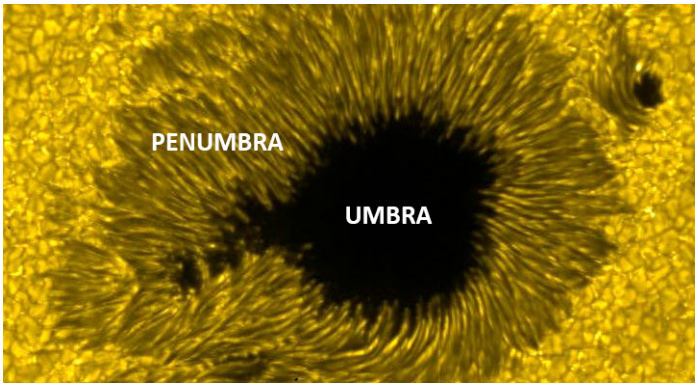


Figure 3: Image of the dark umbra and brighter penumbra of a spot. [28]

These dark portions of the sun are linked to strong, local magnetic fields in those locations [3] with their own associated polarities. Most sun spots will appear in clusters, with a dominant, larger spot with a polarity the same as that of the hemisphere it is in, and smaller spots with the opposite polarity[3]. This means that the clusters themselves act as microcosms of the sun's grater activity, and a visual representation of the fields of the spots can be seen in **Figure 4**[3].

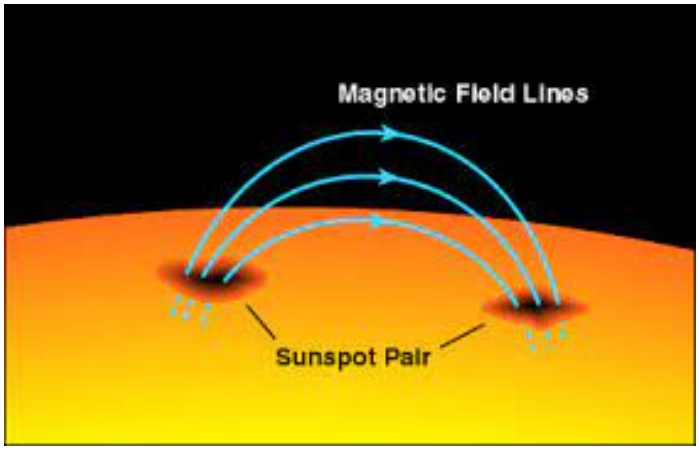


Figure 4: A basic diagram of the local magnetic fields characteristic of sun spots. The blue lines represent the field lines of the loca magnetic field that connects the two spots. This phenomenon is critical in the eventual pole flip as described by Dynamo Theory. [23]

This diagram not only shows the relation of spot clusters across the surface, but the direction of the field lines is important to note as well. These spots are almost like entrance and exit wounds of the magnetic field lines from the sun–the darker umbra is where the field lines exit approximately vertically and the field is very strong. The penumbra of the spot is where the lines begin to slope off and exit less steeply [3].

Interestingly, the spots measured using butterfly diagrams seem to show a shift of their appearance to be nearer and nearer the equator until their migration to the equator induces the flip of the poles and the lessened activity [4]. Note that the spots themselves do not move across the surface of the sun, they emerge at different latitudes over time and persist there. After the global field flip, they once again appear at latitudes further away from the equator and again, their appearance migrates until the pole flip. This is seen in **Figure 5** showing what is commonly referred to as a butterfly diagram of the appearance of the spots and their associated magnetic field orientation (north or south) and the orientation of the poles [3]. This cycle of emergence and disappearance directly correlates with the global shift of the sun and has an overall cycle period of 22 years [3].

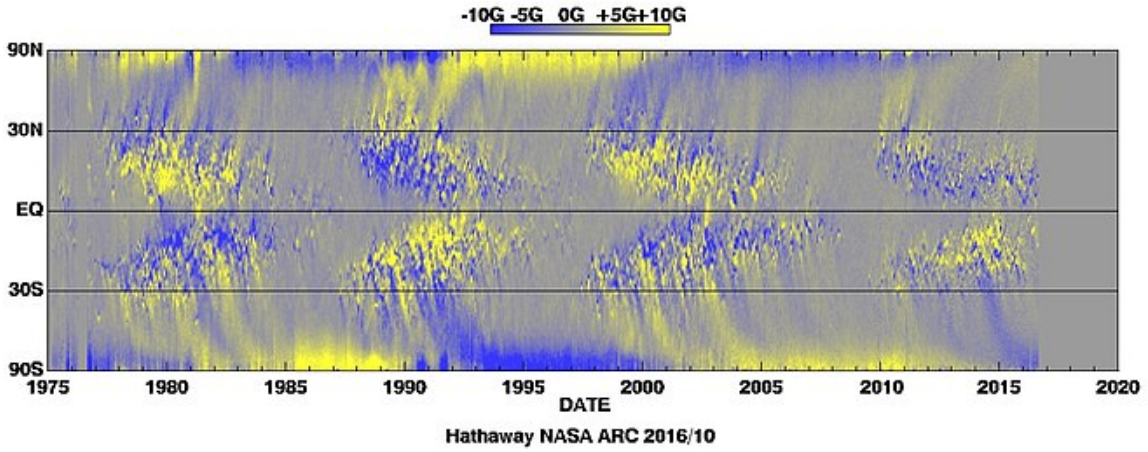


Figure 5: Image of the magnetic field of the sun and the polarity of emerging spots over time. The diagram shows the latitude spots emerge and their polarity over the solar cycle, as well as the global field of the sun at that time.[13]

The strength of the magnetic field associated with the spots on the sun is relatively easy to measure. They have been recorded at several tenths of a Tesla near the umbra, or several thousand Gauss [3] and this can be done through observation of Zeeman Splitting. These measurements, while straightforward for the sun, do not always work for other stars. The spots themselves on the sun seem to be an effect of the change in the pressure gradient of the chromosphere due to the presence of strong, exiting magnetic field lines[3]. These are not the only activity markers observed on the sun, however, and other activity markers also related to the magnetic field shift are readily observable and possibly devastating on earth.

1.1.2 CMEs and Flares

In general, the active regions of the sun host not only sun spots, but other types of activity associated with turbulent magnetic fields[31]. A much more aggressive form of activity from the sun comes in the form of Coronal Mass Ejections (CMEs) and flares. These are more aggressive expulsions from active regions on the sun, which typically contain spots, flares, prominences, and other activity markers.

CMEs, simply, are massive ejections of material from the corona of the sun[3]. They can be in a range of wavelengths, but most common to the sun are white-light CMEs[6]. These features are particularly interesting due to how they seem to follow the path of the magnetic field loops, like the image seen in **Figure 6**. This is, however, only one of the theories on the shapes of CMEs. This

shape and association with magnetic field lines is due to these features being a sort of release of magnetic energy from the sun. Though, they are released from the corona instead of lying on the photosphere like spots.

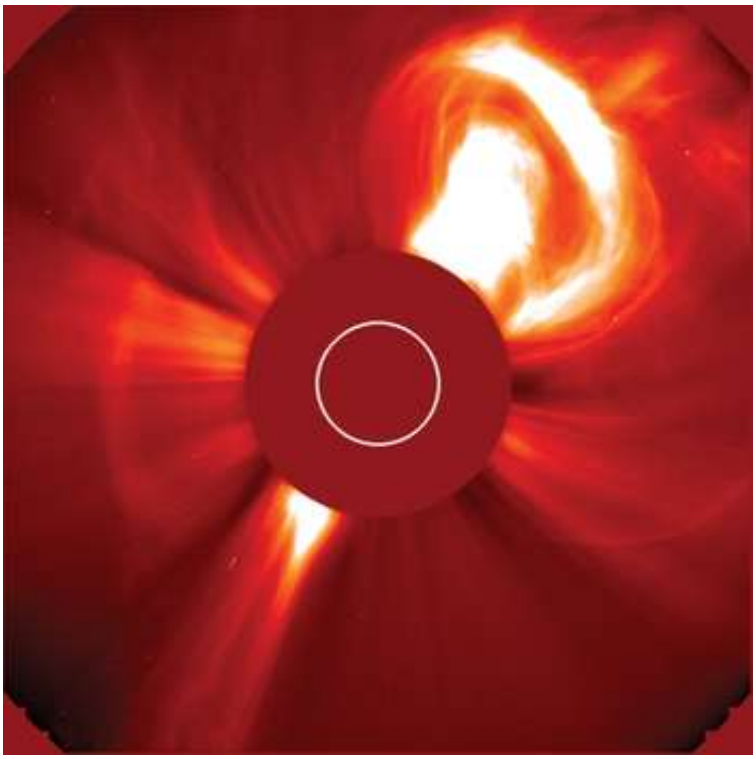


Figure 6: Image of a halo CME. These features have a characteristic, loop-like shape which follows the magnetic field shape in that area. The release of the energy trapped in this field is the CME eruption itself.[21]

These events are more common during the high activity period of the sun cycle, further solidifying their association with turbulent magnetic fields[6]. There are other, similar solar eruptions like flares which often accompany CMEs [6]. Flares, however, are somewhat less aggressive forms of a similar energy release due to turbulent magnetic fields[31]. CMEs are particularly troublesome due to the possibility of their interactions with the earth.

Due to the emission of mass from the sun, CMEs are able to interact with the earth in ways other activity are not. In particular, the charged particle cascade can cause widespread damage to any devices reliant on electricity, and certain events have been hypothesized to possibly destroy infrastructure due to the potential severity of the geomagnetic storms they can cause [28].

Flares, thankfully and unlike CMEs, do not actually send mass propelling through space towards the earth. They are instead electromagnetic bursts of energy across multiple wavelengths, likely due to the conversion of free magnetic energy in the chromosphere to thermal and kinetic energy[31].

Though this is similar to the theoretical mechanism by which CMEs are created, the two events differ in content and severity. However, both are another consequence of the overall magnetic turbulence of the sun during its toroidal cycle.

1.2 Star Spots and Activity

Unlike the sun, there are limited methods for observations and measurements of the magnetic fields of stars. Some of these methods are indirect measures of the activity of stars, such as the modeling of star spots and the tracking of flare and other activity from the star. Further, this analysis involves, where possible, an assessment of the rotation of the star to check any correlation with activity [1].

A few of the more popular methods of measuring stellar magnetic activity include photometric variability measurements, analysis of Hα emission lines, Calcium H and K line analysis, and Zeeman splitting[4].

The most direct analysis of the strength of stellar magnetic fields is through the analysis of Zeeman signatures. Strong magnetic fields can interfere with light exiting the star and characteristic broadening of the spectral lines can be used to directly analyze the strength of magnetic fields on stars [4], [3]. However, this is not possible for most stars due to the need for proximity.

An indirect measure of the activity is also seen in the Ca II H and K emission lines from stars, as well as Hα, as these generally correlate to chromospheric activity and heating and thus, magnetic activity on stars. These are relatively easy measurements to make as well, as the analysis of spectral lines from stars is a widespread tool for astronomy. These emission lines thus provide some insight into chromospheric activity associated with magnetic field turbulence.

Photometric variability is simply the analysis of the periodic brightening and dimming of stars over time, which can be due to star spots across the surface of the star facing us. Due to the increased precision of instrumentation, this is a viable method of analysis. However, it can be somewhat difficult if the star is brightening in some areas and spotted in others, making it sometimes difficult to execute if the star does not have a transiting companion to provide insight into more specifically where the spots are and how they are arranged. Still, it can be a useful tool for some stars and is relatively simple in execution as well[4]. Photometric variability is also an indirect measurement of magnetic field activity of the star, as we must assume that the increased activity is directly associated with the turbulent magnetic field[4].

Interestingly, it seems that the magnetic activity of the star is correlated to the presence of a

convective portion of the star [4]. Stars like the sun have a convective exterior which surrounds a radiative interior, whereas other larger stars have a convective interior with a radiative exterior. On solar-type stars, the division between the convective and radiative layer is the tachocline. The convective layer is rotating differentially, whereas the radiative layer is not. The tachocline thus marks a point where shearing from the convective layer amplifies magnetic activity.

On other stars, activity and rotation speeds seem to be correlated [4]. Thus, rapidly rotating stars are typically more active than slowly rotating stars, a characteristic also associated with age to a degree [4], as younger stars tend to be more rapidly rotating and "spin down" over time [1]. This activity is also associated with the presence and extent of the convective portions of the stars, as stars with exterior convective zones display activity more than those with radiative exteriors [1]. Finally, slow rotating giants (evolved stars) are typically not expected to have high activity levels due to both age and rotation speeds. However, it is evident that there are some stars of this type break the expected trends of activity through other interactions.

1.3 Red Giants

Red giant stars are a branch of star types which form off of the main sequence of stars. Stars on the main sequence fuse hydrogen to helium in their cores. This fusion is the source of energy production for main sequence stars, allowing them to emit light[3]. Brighter stars are thus usually shorter lived on the main sequence than dimmer stars, as their luminosity means a higher rate of fusion [3].

Red giant stars form from stars initially on the main sequence, undergoing the fusion of hydrogen to helium through the proton-proton chain. These types of main sequence stars have both a convective core and radiative exterior of the star, similar to the sun[26]. As the star begins to burn through hydrogen, it no longer produces the energy necessary to balance the pressure formed in the interior and the pressure of gravity. Thus, the star's core begins to contract significantly. However, with this contraction comes more heat in the center of the star, once again triggering the burning of hydrogen outside of the core and the rapid expansion of the outer layer of the star. For red giants, this outer layer expands over time to the now modified core of the star, causing an expansion in the radius of the star[3]. At this point, the star has made its way up the red giant branch.

Generally, main sequence stars spin more slowly as time goes on [4], and red giants do as well, though this is due to the increased size of the radius rather than energy loss over time. However, while this is a trend there are of course exceptions. Due to the slow rotation speed of red giants and

the link of rotation speed to activity [4], it is no surprise that most red giants aren't usually very active. Typically, young stars are more active[4] than slow rotators.

1.4 Eclipsing Binary Stars

Most stars do not stand alone, and often have companions in the form of other stars or exoplanets that are gravitationally bound to them. While there are many different ways to obtain data about the orbit of eclipsing binaries, the method I will be primarily discussing is photometry, as this is the method employed by the Kepler satellite and is thus the type of data I am analyzing.

Eclipsing binary stars (EBs, eclipsing binaries) are a gravitationally bound system that consists of two stars, a higher-temperature primary or host, and a cooler secondary or companion. These stars orbit each other in such a way that the plane of their orbit is approximately perpendicular to earth. This results in a view from earth of each of the stars passing in front of the other and eclipsing in periodic cycles[12]. These eclipses create a characteristic light curve of the brightness over time in which there are two apparent dips in the brightness of the system, such as the curve seen in **Figure 7**. The dips in the amount of light are due to the star in front blocking some of the light from the star behind. Due to the hotter primary being eclipsed by the cooler secondary, the primary eclipse is deeper than the secondary eclipse, providing insight into the relative temperatures of the stars as well as other parameters of the system. The eclipses themselves are also called the primary and secondary eclipses, corresponding to whichever star is being eclipsed.

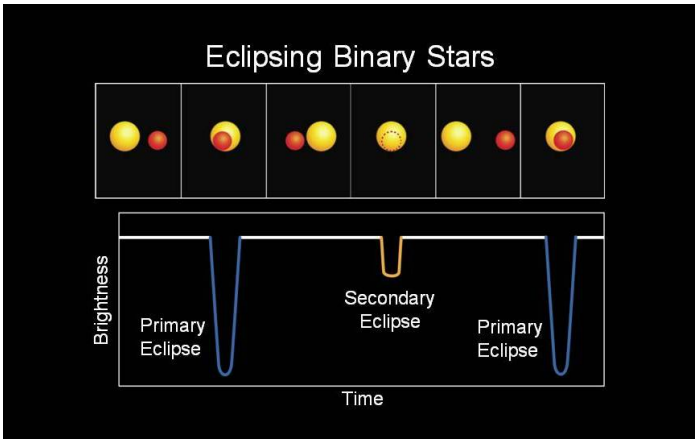


Figure 7: Image of the characteristic Primary (blue curve, yellow circle) and Secondary (orange curve, red circle) light curves of an example eclipsing binary system. The shape of the curves are further affected by various stellar and orbital parameters not explicitly shown here.[9]

They also have characteristic radial velocity curves, which show the speed at which the object is traveling towards or away from us. Measuring the spectra or distribution of wavelengths of light

coming in from the system over time gives us a relationship between the spectra when the system's components are not moving towards or away from us and when they are. When moving towards or away from us, we see a red or blue shift in the spectra, corresponding to the object moving away from or towards us respectively. For eclipsing binaries, the comparison of the change in wavelength from this baseline results in two peaks, one red shifted and/or one blue shifted, which gives us in turn the ratio of the masses of the bodies in the system.

These EB systems are similar in many ways to transiting exoplanet systems, where a planet outside of our solar system will pass in front of the star. The case of transiting exoplanets is essentially a special case of eclipsing binaries in which it can be assumed that one object is much more massive than the other. In some cases, such as with KOI-1786, the companion is so much cooler that there is no visible secondary eclipse. In these cases, the status as an eclipsing binary is usually determined through measurements of the radial velocity of the system, which provides the mass ratio of the two objects and thus allows us to know if the companion is a planet or a dim star. However, the light curves for these objects will still appear to be the same. Due to the similarities in the systems for stars with giant primaries and small companions, I will use "transit" and "eclipse" interchangeably when referring to KOI-1786.

The eclipses provide a wealth of information about the system as a whole. Generally speaking, eclipsing binary stars are the reason we are able to determine the masses and radii of stars individually [3], and we have a much firmer constraint on how the orbital plane is tilted towards or away from the observer, or the inclination angle. Through the analysis of the radial velocity curves and light curves, we are able to determine the properties outlined below[12].

Table 1: Observed Parameters of EB orbits and their descriptions

Name	**Symbol**	**Description**
Period	T	the amount of time between transits
Duration	T_{14}	The amount of time from the beginning (ingress) of the transit to the end (egress) of the transit
Transit Depth	δ	
Impact Parameter	b	distance from the center of the star vertically when the companion is mid transit
Epoch/Time of Conjunction	T_c	The time of middle of the first detected transit

The first of these parameters we can use comes directly from Newton's formulation of Kepler's laws, Equation 1. This is the general formulation which can still give the semi-major axis of any

two-body system generally.

$$\frac{a^3}{P^2} = \frac{G(M_* + M_p)}{4\pi^2} \tag{1}$$

The quantity here that is directly observed is the period of orbit, and in the case where the companion is significantly smaller than the primary, the semimajor axis can be obtained relatively easily through the approximation of the total mass of the system being equal to the mass of the primary, or it can be applied using the relative masses of the two bodies[12].

The next easily observed quantity is the transit depth, given by Equation 2.

$$\delta = (\frac{R_p}{R_*})^2 \tag{2}$$

The depth of the transit corresponds directly to the radius of the transiting companion, as the larger the radius of the companion, the deeper the transit is. This equation thus relies on measuring directly the depth of the lightcurve itself. The ratio of radii is all that is needed here and is provided by the light curve.

The next segment is the orbital separation.

$$b = acosi \tag{3}$$

The light curves also give information about where the companion passes in front of the host, or the inclination angle of the orbital plane. It is assumed to be approximately 90 degrees, and the points at which the eclipse begins and ends (ingress and egress respectively), as well as the points where the curve begins to flatten at the bottom, provide further insight into the inclination of the orbital plane[12].

Finally, the last easily observed quantity is the transit duration.

$$T_{dur} = \frac{P}{\pi} sin^{-1}(\frac{\sqrt{(R_* + R_p)^2 - a^2 cos^2 i)}}{a} \tag{4}$$

By using the period obtained from long term observations of the object, we are able to get information about the semimajor axis and the relationship of the radii of the star and companion. The duration itself is measured directly, as is the period, making this equation useful for the evaluation of the radii and semimajor axis of the system as well.

Multiple different parameters of the system can affect the same part of the light curve, which

makes disentangling the effect of each parameter difficult. Some of the direct effects of the system parameters on the shape of the lightcurve can be seen in **Figure 8**.

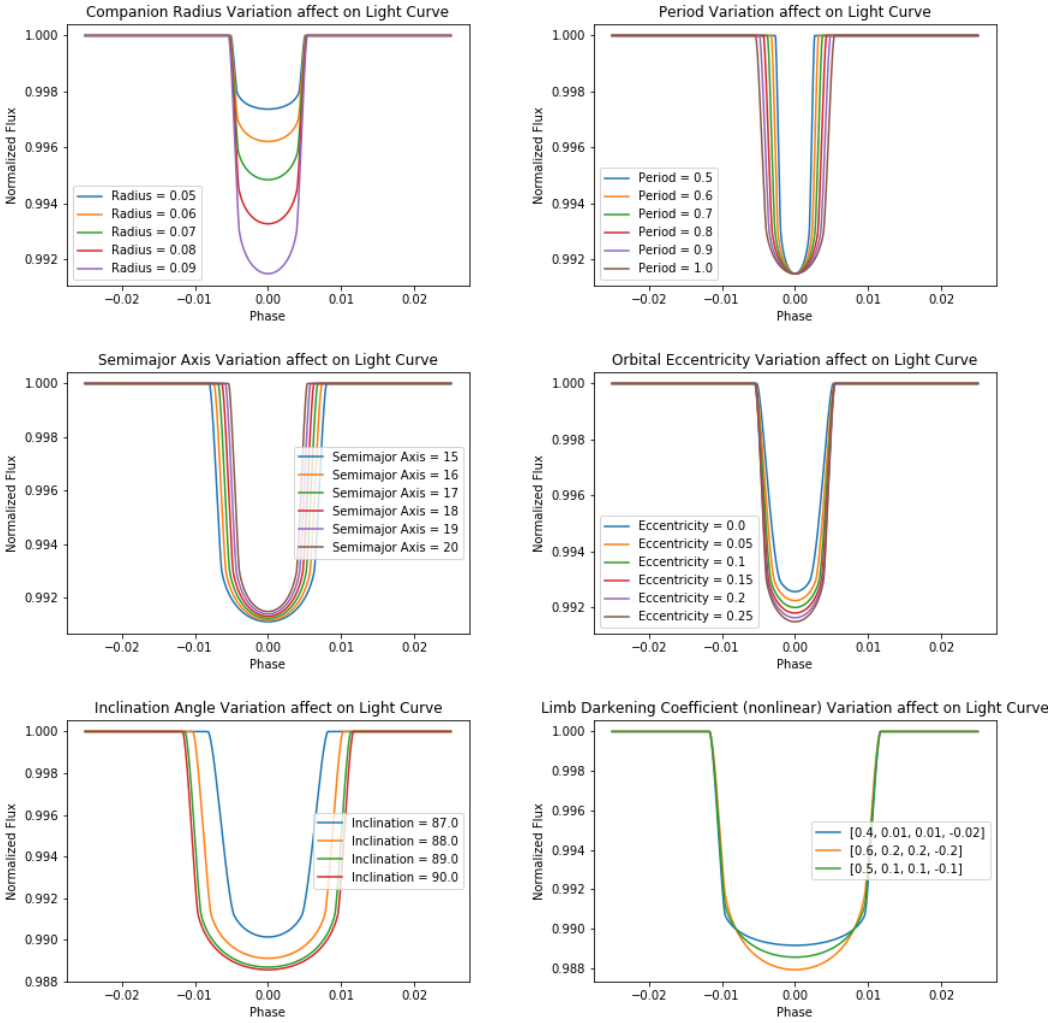


Figure 8: Demonstrations of the effect orbit parameters have on the shape of the light curve for each transit. There are some overlapping ares which are affected by changing different parameters.

Due to this multifaceted influence, an initial assumption for modeling is typically a circular orbit with no eccentricity. While this is not typically the best representation of the actual physical system, these initial assumptions make the process of determining the other parameters of the system more manageable, and a baseline transit shape can then be modified accordingly in order to make the model more accurate and better able to reflect the data that is taken of the system. These initial assumptions, basic properties, and directly observed quantities allow for the derivation of other aspects of the system such as those outlined in Table 3. These derived parameters provide a picture of the system itself and can be used to provide an accurate baseline model of the system and the

light curves they will produce. This is especially important when analyzing any deviations from this baseline light curve.

Table 2: Derived Parameters of EB orbits and their descriptions

Semi-Major Axis	a	The distance from the primary star to companion(s)
Eccentricity	e	The eccentricity of the orbit describes how much it deviates from a perfectly periodic or circular orbit. This affects the timing of the secondary
Inclination Angle	i	The tilt of the orbital plane from perpendicular to the observer
Planet or Companion Radius	R_p	Radius of secondary or exoplanet
Stellar Radius	R_*	Radius of Primary Star
Stellar Mass	M_*	Mass of Primary Star
Stellar Density	ρ_*	Density of the primary star, based on radius and mass of the star

Outside of these parameters, there are other aspects of the system which cannot be derived from the light curve which nonetheless are useful for analysis. These especially come into play when discussing magnetic activity of stellar systems.

Table 3: Other Significant Stellar Properties

Star Temperature	T_{eff}	Temperature of the Primary star in Kelvin
Stellar Surface Gravity	log g	Surface gravity of the star, tends to affect the shape of the light curve
Star Age	Age	Age of Primary Star
Stellar Rotation Period	P_{rot}	Rotation period of the star, describes how quickly its spinning

2 Methods

KOI-1786 is an object first discovered by the Kepler mission. The purpose of the Kepler mission was to find exoplanets and determine their size and if they are within the habitable zone of the star they orbit. It accomplished this through the observation of multiple objects, and eclipsing objects were designated as Kepler objects of interest (KOIs) and studied further. Kepler observed KOI-1786

during its operation.

The Kepler observations are done through time series photometry, which measures changes in the brightness of the star over time to find possible exoplanet transits. The telescope itself has a 1.4 m mirror and the field of view is approximately 110 square degrees. This allows it to observe different sections of the sky at once in an effort to find exoplanets. It observed over 150,000 stars for four years [15].

The Kepler Data was further segmented into observation quarters (3 months). Of the 17 Quarters, there was data available for KOI-1786 for 14 of them. There were 47 transits we were able to find for this object in the Kepler data. The radius of the host was determined by using the second Gaia Data Release [27]. One quarter of Kepler data is shown below in **Figure 9**.

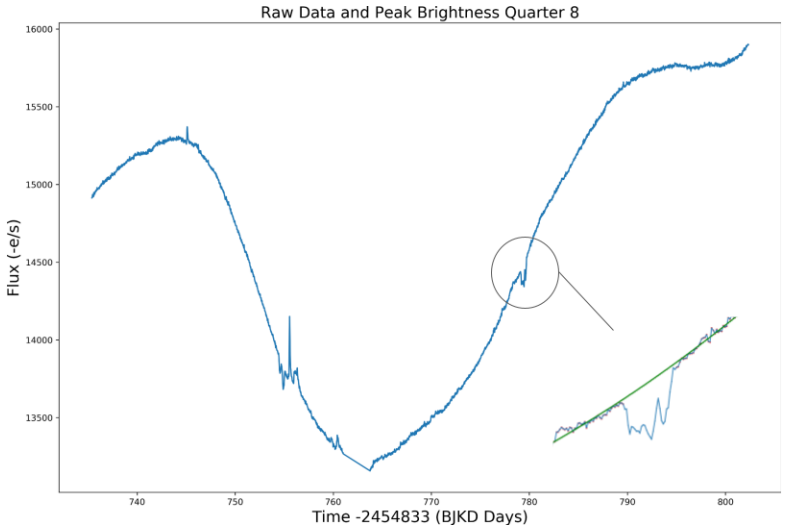


Figure 9: One quarter of Kepler data. A transit appears as a small bump below the sine-like global trend in brightness and is highlighted to the right. There are also flares which appear as sharp features above the same global trend. Each quarter is a 3-month time span.

The transits need to be pulled from the larger data pool for individual analysis so that variability in brightness over time can be distinguished from other spot crossing features and activity. This necessitates normalizing the transits and flattening them in order to perform a proper analysis, and then the resulting data needs to be modeled so that the system parameters can be derived from the light curve as discussed in Section 1.4. A model is generated by the Kepler team for the data taken by the satellite, and the associated parameters are derived. This model and the normalized data can be seen in **Figure 10**, and is generated from Data Release 25 (DR-25) parameters[30].

However, this method does not take into account the various spot crossing features in the transit or the larger scale out of transit variability, meaning the parameters derived do not accurately

describe the system. Thus, a re-normalizing and remodeling of the transit is necessary for the spot modeling process.

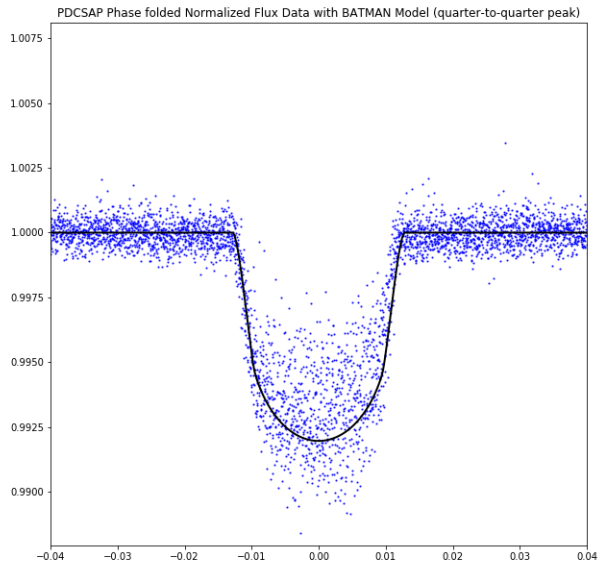


Figure 10: Normalized data with DR-25 parameter model. The model transects the data well, however, the data contains multiple star spot features which affect the fit of the model to the data.

2.1 Normalizing in the Presence of Variability

Data from the Kepler satellite is either Pre-data corrected simple aperture photometry (PDC-SAP) or it is not corrected (SAP). This correction is done through NASA's pipeline and corrects for things like instrument drift over time and other systematic errors due to instrumentation. For our purposes, PDCSAP data was used, and can be seen in **Figure 9**. However, the data itself is not in a workable form for our analysis, and so transits must be isolated and cut out from the overall data files from Kepler. Then, the portions outside of the eclipse need to be modeled for correcting and flattening the transits so they are able to be analyzed for spot crossing features.

First, we need an estimate of the unspotted brightness of the star. This gives us a method for normalizing as well as a way of accounting for variability in brightness due to spot presence on the primary. We take the 99.5th percentile average of the brightness per quarter (peak value) to be our value here. While usually there is small variability on a quarter to quarter basis, KOI-1786 is extremely active and fluctuates more than its similarly bright neighbors, as seen in **Figure 11**.

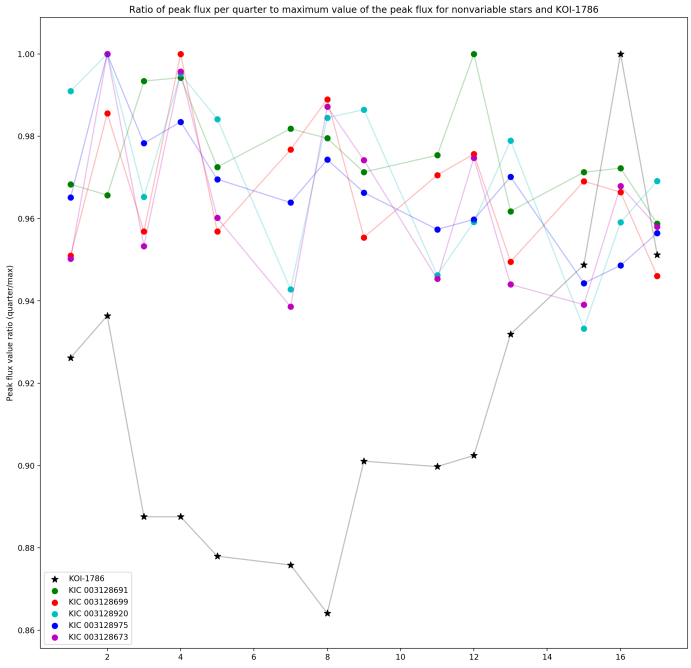


Figure 11: Ratio of the peak brightness in each quarter to the maximum peak value. Some variation is seen in nearby stars of similar brightness as KOI-1786 (Black) that is likely due to instrumentation or other systematic error, but KOI-1786 shows a much greater variability not attributable to systematic errors.

We thus decided to use the overall maximum peak value across all quarters for normalizing, as this was likely the best representation of the "unspotted" brightness of the star.

Next, we cut the transits out of the surrounding data as seen in **Figure 9**. This is done to isolate the transits from other forms of variability and to make more apparent the spot features in the transit itself. Each of these, however, was subject to the variability of the star outside of the transit, and so the points outside the eclipse were fit with a polynomial, as seen in **Figure 12**.

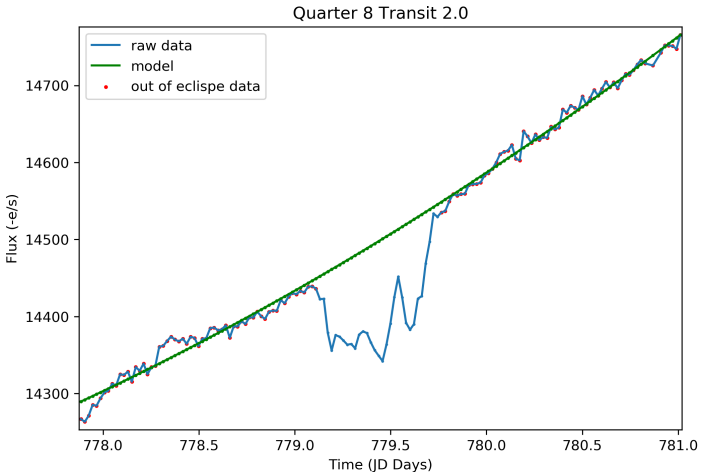


Figure 12: The normalizing method begins by first selecting a transit and some surrounding data for a 2nd degree polynomial fit. The red points show the raw data being fit by the model, and the polynomial fit model is displayed in green. This regression allows for the light curve to be flattened.

If the end goal was simply to flatten the transit, it would suffice to simply subtract this polynomial fit line from the data. However, our goal is to further put the transit in context of an unspotted star, meaning we must include the unspotted brightness in our normalization process. Thus, the normalization is done using the equation below.

$$NormalizedFlux = \frac{Data - Polyfit + Peak}{Peak} \tag{5}$$

Where as before, the peak is the maximum value of the 99.5th percentile average of the brightness of the star across all quarters. The data is simply the raw data taken from Kepler, and the polyfit line is the polynomial fit of the line formed by points outside of the transit itself. The resulting normalized transit is seen in **Figure 13**. One thing to note is that there were instances of rather large flares very close to or inside of the actual eclipse. These were dealt with either by removing the data associated with the flare for fitting (as it would throw off the polynomial fit) or, if they were inside of the actual eclipse, modeling as usual. This highly active giant did have a number of instances where this was necessary.

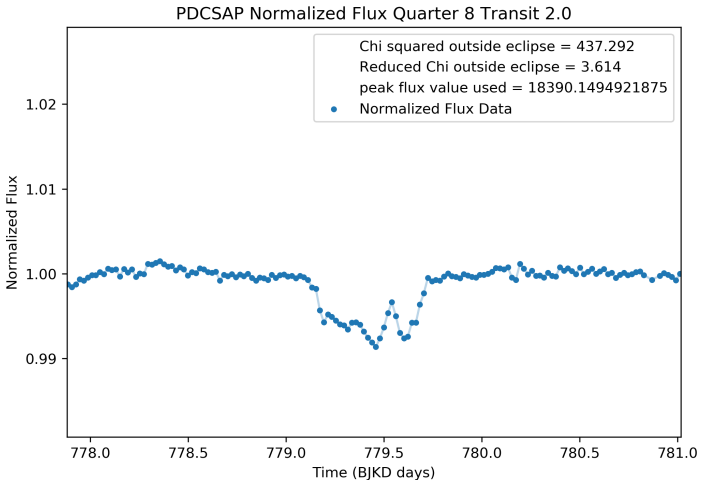


Figure 13: The resulting normalized transit for Quarter 8 Transit 2. This process makes clear the prominence of the spot within the transit itself and prepares the transits for eclipse modeling.

2.2 Transit Modeling

Modeling the transits, as noted earlier, requires deriving multiple parameters. For this project, in order to disentangle these properties for an adequate data model, we used a Markoff-chain Monte Carlo Transit Modeling program [7] specifically designed for the modeling of light curves. This program takes a light curve and radial velocity curve, as well as the temperature and limb darkening parameters of the star as inputs and tests multiple variations of the parameters. It then uses the χ^2 value of each set, which is a method of evaluating the fit of a model to the data, and navigates through the parameter variations to arrive at a minimum χ^2 value, corresponding to the best model fit to the data.

The typical approach for transit modeling using the Transit Modeling program involves looking by eye at the normalized light curves of individual transits and trying to find a transit without any spot crossing features. This "unspotted" transit, initial guesses for the parameters of the system, and radial velocity curves for the particular object are then input into the Transit Modeling program and used to generate a model of the unspotted transits according to the processes outlined in section 1.4. This model and the associated parameters can then be used for spot modeling. However, it turns out there is not a single full transit for this system that does not have a flare or spot crossing feature. This is obviously problematic for analysis, and required some work to overcome.

First, we used the modeling program to generate a model of all of the data. We expected this to approximately cut through the middle of the data, and we would use this as a way to cut data that likely corresponded to spot crossing features, as seen in **Figure 14**.

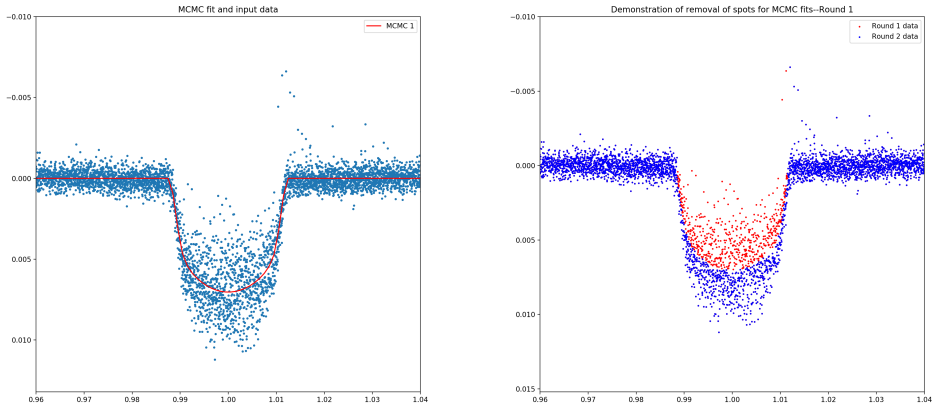


Figure 14: An image of the data processing in order to determine the system parameters of KOI-1786 using the Transit Modeling program. The data above the model given by Transit Modeling was cut, and the new data is shown in blue whereas the data that was removed is shown in red.

This first cut resulted in a new data pool for modeling. We then modeled the resulting data, and the resulting model is seen in **Figure 15**. A third trial was attempted, however, the scatter out of transit was significantly higher than the in transit scatter for the third cut. So, the second cut model was used.

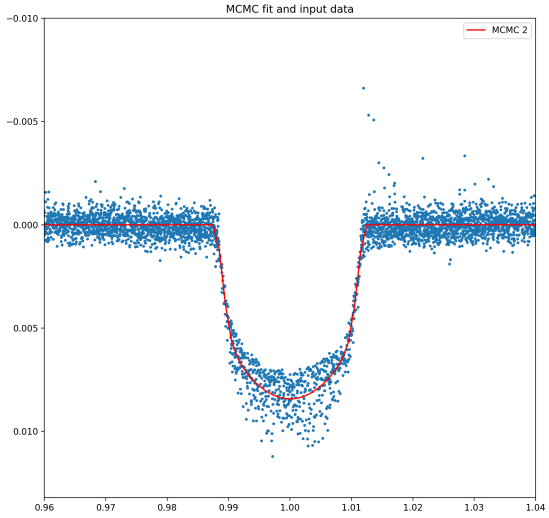


Figure 15: An image of the data processing in order to determine the system parameters of KOI-1786 using the Transit Modeling program. This is the model over plotted on the data after the first cut. While a second cut was made for this process, this is the cut with matching in and out of transit scatter.

Using the data from this cut, we were able to generate the system parameters of KOI-1786 from the transit modeling program.

Table 4: Parameters of KOI-1786

Parameter	**Model Value**
T_c	$5612.4254^{+0.000600}_{-0.000566}$
$(Rp/Rs)^2$	$.0064^{+0.000103}_{-0.000115}$
T	$24.67890^{+0.0000298}_{-0.0000304}$ days
T_{14}	$0.67^{+.0968}_{-0.10}$ days
b	$0.49^{+0.0321}_{-0.0386}$ r_*
e	$0.299^{+.00210}_{-.00203}$
i	$85.5^{+0.444}_{-0.387}$ degrees
T_{eff}	4504 K
R_*	$4.2^{+0.107}_{-0.117}$ $R_\odot$
M_*	$1.29^{+0.01522}_{-0.01677}$ $M_\odot$
log g	$3.28^{+0.0184}_{-0.0164}$
ρ_*	$1.656E-002^{+0.00120}_{-0.00100}$ $\rho_\odot$
P_{rot}	66.289 [20]
R_p	$0.34^{+0.0114}_{-0.0123}$ $R_\odot$
M_p	$0.496^{+0.00402}_{-0.00413}$ $M_\odot$

These parameters show that KOI-1786 has a red giant primary of spectral type K with an M-dwarf companion.

Using these parameters, the transit modeling program was able to generate a model for the transits. However, the model was generated in terms of the phase and magnitude of the transit. Due to this and the issues it causes with over plotting on flux from individual transits, the parameters seen in the table above and the associated model file were plotted using the Bad-Ass Transit Modeling cAluculatioN (BATMAN) package in python. The model and the associated data for Quarte 9 Transit 3 are seen in **Figure 16**.

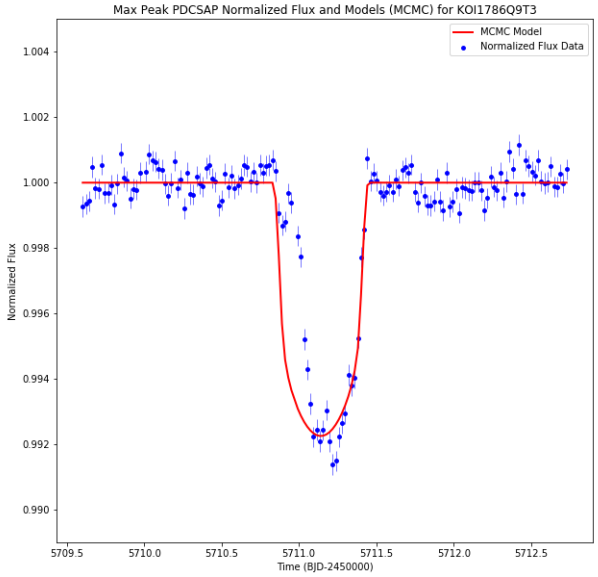


Figure 16: An image of the normalized flux data from Quarter 9, Transit 3. With the model, the size of the spot becomes much more clear and the dip in the curve afterwards seems to be a real feature and not a result of a faulty model, as previously assumed.

In this transit, there is a slight dip in the transit after the large spot crossing feature. With this model, we are able to see the dip is in fact a bright point, not a result of a faulty model. Such features are seen on other transits as well. Using the newfound parameters generated by MCMC, we are able to finally begin plotting the spots themselves on the surface of KOI-1786.

2.3 Spot Modeling

STarSpot (STSP) was the program used to generate the spot models and fits. For this object, we needed to determine the number, radii, and latitudes, and longitudes of the spots on the surface of the primary. Further, we would need to determine these parameters along with different values for the darkness of the spots (contrast). In the context of STSP, contrast is the ratio of the brightness of the spot and the brightness of the star. Thus, a perfectly dark spot would have a contrast of 0, and a typical spot on the sun would have a contrast of 0.7.

It became immediately apparent that the typical contrast used for the sun is simply not dark enough to adequately model some of the spots on KOI-1786. Due to this, the focus of the modeling became centered around determining which contrast ranges are best suited for modeling KOI-1786 transits and the size and location of the spots corresponding to each contrast.

For each transit modeled in STSP, we began by making a guess of the latitude, longitude, and size of the spot based on the light curves. Most basically, he longitude is restricted by the time at which the center of the feature occurs, the latitude is restricted by the latitude at which the companion crosses, the radius is restricted by the width of the spot crossing feature on the light curve, and the contrast corresponds to the height of the feature. Each of these parameters has an effect on the others, however, for the initial guesses the fit need only be fairly close, not exact. These guesses were each fed into STSP using an "Action l" which simply takes the parameters it is given and generates the shape of the light curve that would result from that exact configuration of spots. Through numerous trials, we can get an approximately good fit to the data for the next portion, which is refining the parameters and model using an "Action S" run in STSP.

The reason a good initial fit is so vital to the fitting process is due to the mechanism of the STSP Action S. the Action S runs through a Multi-chain Monte Carlo (MCMC), a similar process to the Transit Modeling program, with a given number of chains and trials. This program takes a light curve and the transit parameters as inputs, as well as the initial guesses for the spot size and location. It then tests multiple variations of the parameters in multiple chains (40 for our case) and finds the χ^2 value of each set, which is a method of evaluating the fit of a model to the data. It then varies these parameters within each chain for a given number of trials (500 in our case) and reports the lowest χ^2 score across each chain and the parameters for that value. Each of the initial parameters is used and modified and the best χ^2 value is kept and reported across all chains. Through this, the program effectively navigates through χ^2 space and finds minimums, which correspond to a technically "good" fit to the data.

The issue is that the χ^2 space is riddled with local minimums, meaning that STSP can be easily tripped up and can find a false minimum. Since it navigates by taking the lowest χ^2 value in sequence, it cannot usually remove itself from this false minimum, resulting in a less-than-optimal fit. Thus, a very good initial guess for the spot parameters is vital for preventing these false minimums from being used. The program provides the best fit parameters for the spots as well as the associated χ^2 value for those parameters, as is the model generated by the parameters.

Further, in the χ^2 space, there are equally good fits within a certain range of χ^2 values. All χ^2 values are reported out in a separate file from the parameters. Using this file, we can plot the range of equally good solutions and the ranges of allowed latitudes, longitudes, and radii for each contrast value. The formula which determines the 1σ range of the best fit can be seen below, and is a variation on the incomplete gamma function[32].

$$P(a,x) = \frac{1}{\Gamma(a)} \int_0^x t^{a-1} e^{-t} dt$$

The Gamma function itself governs the transformation used to find the range of values, which takes an input of the range of values (68% or 1σ in this case) and returns P. The equation below gives the desired range through:

$$\chi^2 \; 1\sigma \; Range = 2 * \frac{1}{(P(\frac{N}{2}, 0.683))}$$

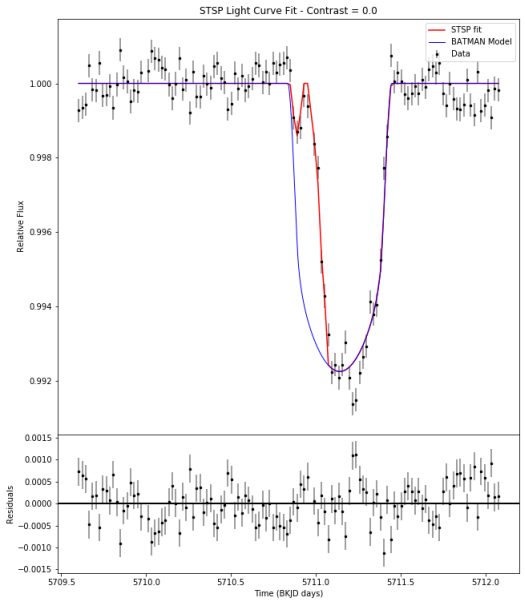
Where N is the number of free parameters (In this case, 3 x Number of spots).

We tested this process for several known values of the chi squared variations with a given number of free parameters. When this was verified to hold, we then applied it to unknown variations with more parameters. Using all of these, we are able to plot the location of the spots for the best parameters, the resulting light curve and model, and the 1σ range of parameters for each of the contrasts for each transit. All of these resulting graphs for the Quarter 9 Transit 3 data can be seen below, labeled according to the contrast used to generate the spots and model.

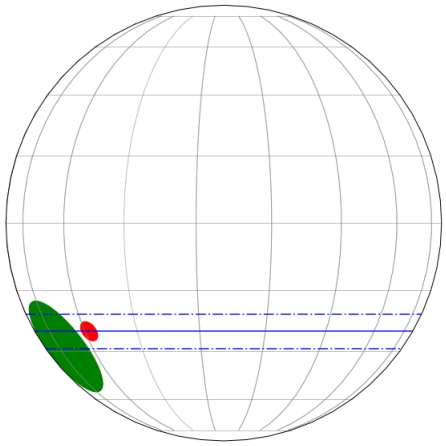
Qualitatively, the longitude of the spot is restricted by the point in the transit that the spot feature is seen, and the height of the spot feature in the light curve correlates to the contrast of the spot. Taller spot features are indicative of a darker spot. The Latitude and the Radius of the spot are closely linked, as the radius of the spot relates to the width of the transit and the latitude of the spot affects the transit width as well. Generally, a wider transit can be fit with a smaller spot directly in the path, or a larger spot out of the path. Due to this inter relatedness, it is sometimes difficult to disentangle degenerate solutions, meaning there is typically a wider range of radius and latitude values for equally good solutions than there are for longitudes.

The spot models are over-plotted on a star model (top left) and the light curve that would be generated from those spots is shown as the red line on the transit light curve (top right). The residual plot for that model is also shown directly below the model plot, as well as the blue model which shows the transit shape for an unspotted transit. On the bottom left the MCMC plot is shown with all points within a 1σ variation are plotted according to latitude and longitude. The size of

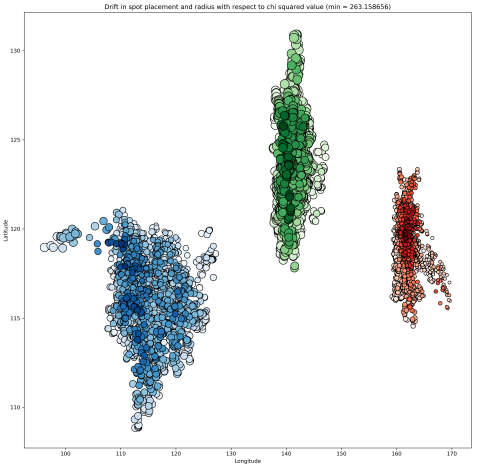
the points correspond to the size of the spots and the darker spots correspond to better chi squared values.



(a) model and light curve



(b) Spot map on star

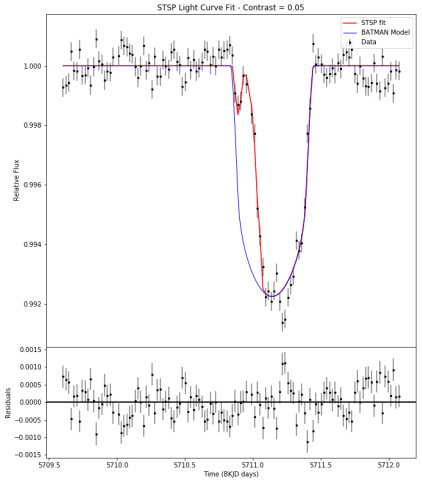


(c) MCMC Plot

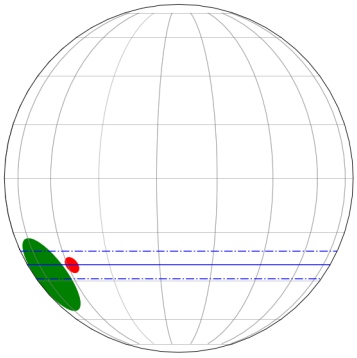
	Spot 1	Spot 2	Spot 3
Radius	$0.123^{+0.139}_{-0.056}$	$0.26^{+0.185}_{-0.115}$	$0.051^{+0.049}_{-0.115}$
Latitude	$117.9^{+3.156}_{-9.059}$	$125.776^{+5.128}_{-7.97}$	$119.719^{+5.128}_{-7.97}$
Longitude	$110.807^{+16.401}_{-14.623}$	$139.509^{+7.612}_{-1.838}$	$161.696^{+7.612}_{-1.838}$
Chi Squared	263.159		

(d) Table of best fit parameters

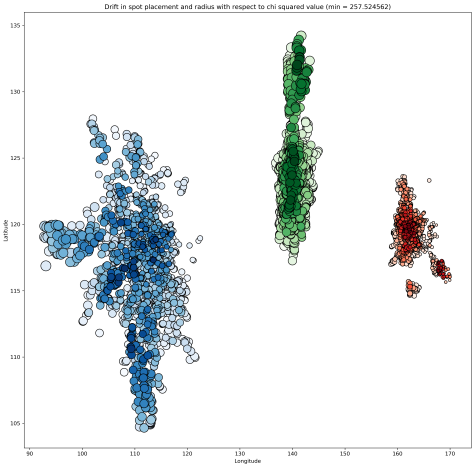
Figure 17: The different graphs and best fit parameters representing the spots on the surface of KOI-1786 with a contrast of 0.0. Both the MCMC plot and the circle plot show the first spot in blue, the second in green, and the third in red. The contrast of 0.0 was the limiting case of spot contrast for this object since the spot features were so large.



(a) model and light curve



(b) Spot map on star

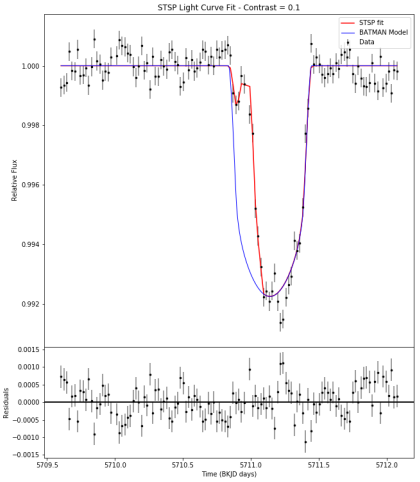


(c) MCMC Plot

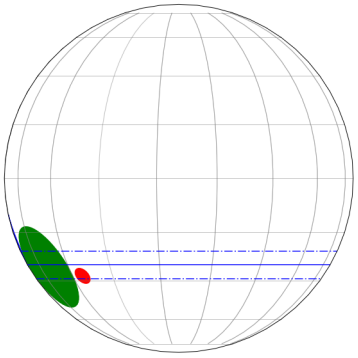
	Spot 1	Spot 2	Spot 3
Radius	$0.169^{+0.093}_{-0.102}$	$0.255^{+0.14}_{-0.11}$	$0.051^{+0.054}_{-0.11}$
Latitude	$116.68^{+4.376}_{-7.839}$	$124.751^{+6.153}_{-6.946}$	$119.814^{+6.153}_{-6.946}$
Longitude	$108.094^{+19.114}_{-11.91}$	$140.375^{+6.745}_{-2.704}$	$162.179^{+6.745}_{-2.704}$
Chi Squared	257.525		

(d) Table of best fit parameters

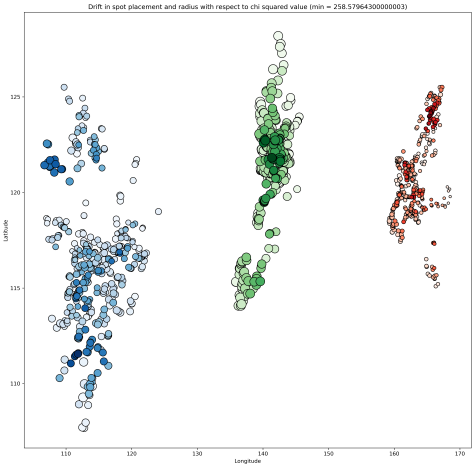
Figure 18: The different graphs and best fit parameters representing the spots on the surface of KOI-1786 with a contrast of 0.05. Both the MCMC plot and the circle plot show the first spot in blue, the second in green, and the third in red.



(a) model and light curve



(b) Spot map on star



(c) MCMC Plot

	Spot 1	Spot 2	Spot 3
Radius	$0.198^{+0.064}_{-0.13}$	$0.276^{+0.111}_{-0.131}$	$0.053^{+0.033}_{-0.131}$
Latitude	$111.559^{+9.497}_{-2.718}$	$121.821^{+9.083}_{-4.016}$	$124.006^{+9.083}_{-4.016}$
Longitude	$111.836^{+15.372}_{-15.652}$	$141.357^{+5.763}_{-3.686}$	$165.354^{+5.763}_{-3.686}$
Chi Squared	258.58		

(d) Table of best fit parameters

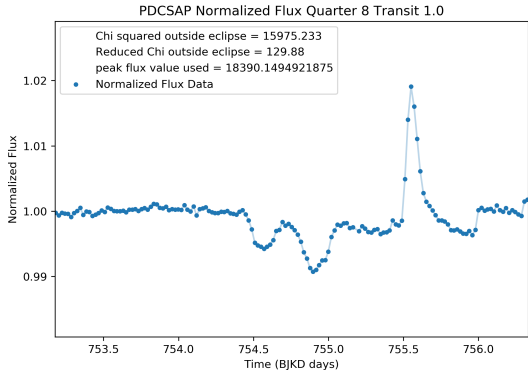
Figure 19: The different graphs and best fit parameters representing the spots on the surface of KOI-1786 with a contrast of 0.1. Both the MCMC plot and the circle plot show the first spot in blue, the second in green, and the third in red.

As noted earlier, it is clear that the feature in Q9T3 simply cannot be fit by a spot of contrast higher than 0.3, and in fact begins to fail even around contrasts of 0.1. This is seen in the inability of the spot model to reach the peak of the spot crossing feature, an issue associated with the contrast of the spot itself. This means the contrast for these spots is exceptionally dark, and even despite the darkness of the spots they must still be fairly large.

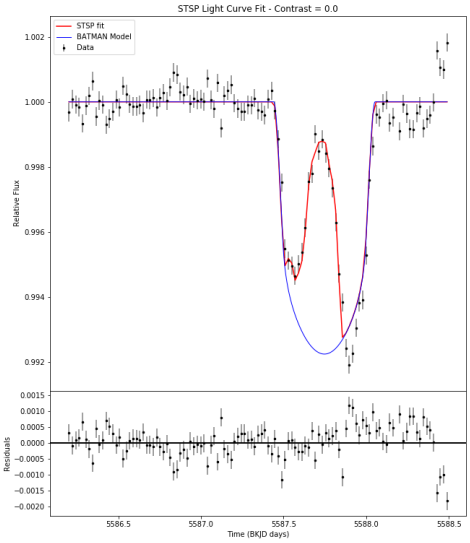
The associated errors with the parameters themselves comes directly from the analysis of the

MCMC output file with the one sigma variation in the chi squared value. The best fit for this particular spot is likely approximately 0.05 contrast. The distribution in possible radius, latitude, and longitude are also shown in the table and the MCMC plot.

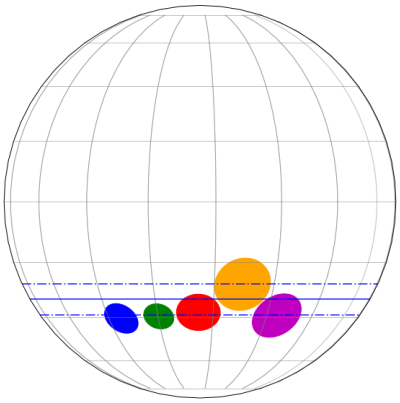
This is not the only transit with such features, and similarly large spots are seen in other transits. While the full analysis is yet to be completed, the same process for the limiting case of spot contrast (0.0–the darkest possible) was also completed for Quarter 8 Transit 1. This particular spot are is similarly massive, however, it is much wider, necessitating approximately 3-5 spots for a good fit to the light curve. The five spot model and associated parameters are displayed below.



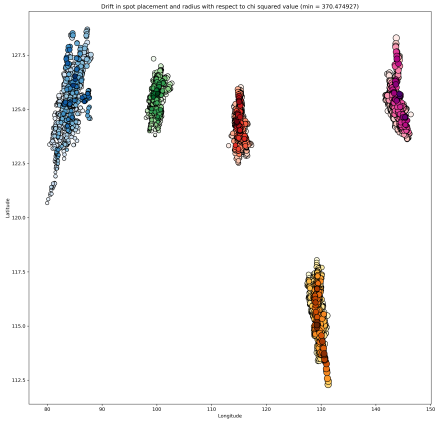
(a) Transit with Flare



(b) Spot Model



(c) Spot Diagram



(d) MCMC Plot

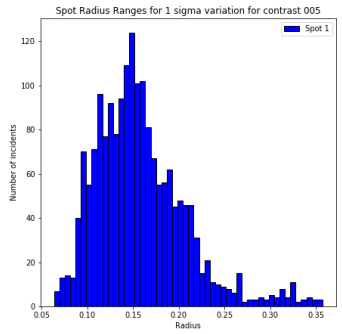
Figure 20: Preliminary Fits for Quarter 8 Transit 1. The light curve before adjustment for the flare is shown to demonstrate the size of the flare that accompanies this transit. The spot model and residuals are shown in figure b, and the spots on figure C correspond to the points of the same color on figure d.

As is evident by the height of the spot crossing feature, Quarter 8 Transit 1 will likely require similarly dark spots, and preliminary work with one spot feature suggests this is the case as well. Thus, there is more than one transit with large, dark spots and other associated activity (flares) can also be seen near the time of the spot crossing feature, suggesting a possible active region similar to those of the sun.

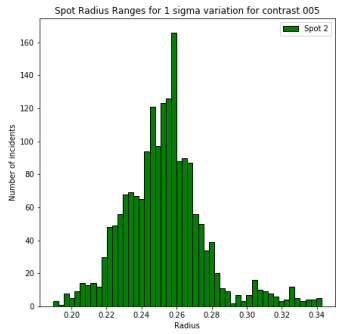
3 Results and Discussion

The best contrast for fitting Quarter 9 Transit 3 based on the chi squared value associated with that trial is 0.05. The associated parameters for this particular fit are displayed in the table below, as are the histograms which show the range and distributions of the solutions for the radius, latitude, and longitudes that fit the different spots are also shown below.

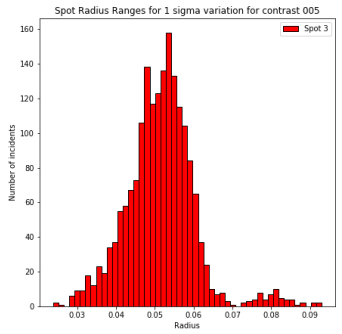
Contrast 0.05	Spot 1	Spot 2	Spot 3
Radius	$0.169^{+0.093}_{-0.102}$	$0.255^{+0.14}_{-0.11}$	$0.051^{+0.054}_{-0.11}$
Latitude	$116.68^{+4.376}_{-7.839}$	$124.751^{+6.153}_{-6.946}$	$119.814^{+6.153}_{-6.946}$
Longitude	$108.094^{+19.114}_{-11.91}$	$140.375^{+6.745}_{-2.704}$	$162.179^{+6.745}_{-2.704}$
Chi Squared	257.525		



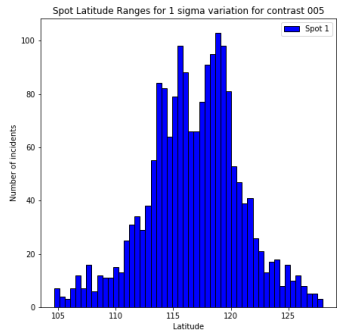
(a) Radius Ranges of Spot 1



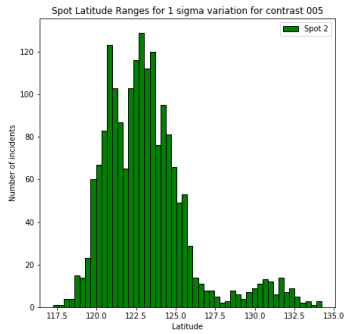
(b) Radius Ranges of Spot 2



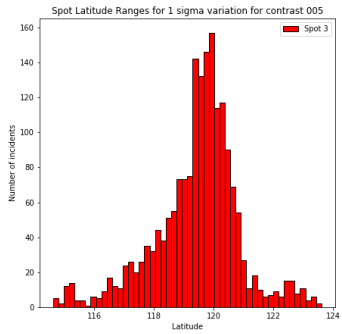
(c) Radius Ranges of Spot 3



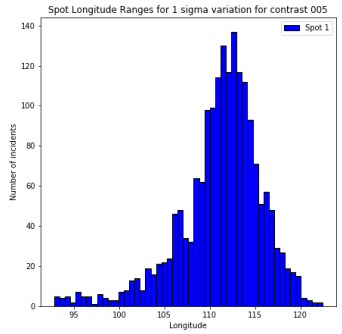
(d) Latitude Ranges of Spot 1



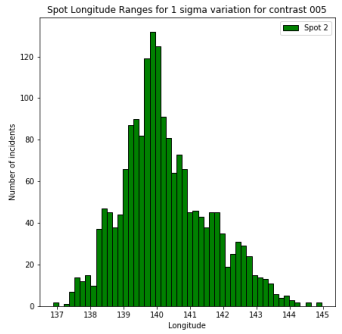
(e) Latitude Ranges of Spot 2



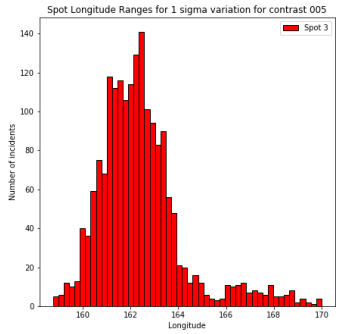
(f) Latitude Ranges of Spot 3



(g) Longitude Ranges of Spot 1



(h) Longitude Ranges of Spot 2



(i) Longitude Ranges of Spot 3

Figure 21: The distribution of values of the radius, latitude, and longitude of the spots for the best solution of Quarter 9 Transit 1.

There is also an associated relationship between the latitude of the spots and the radius of the spots, as the further out of the path of the planet the spot lies, the larger the spot must be to fit the width of the spot crossing feature. The graph showing this relationship for this transit, as well as the associated chi squared values is shown below.

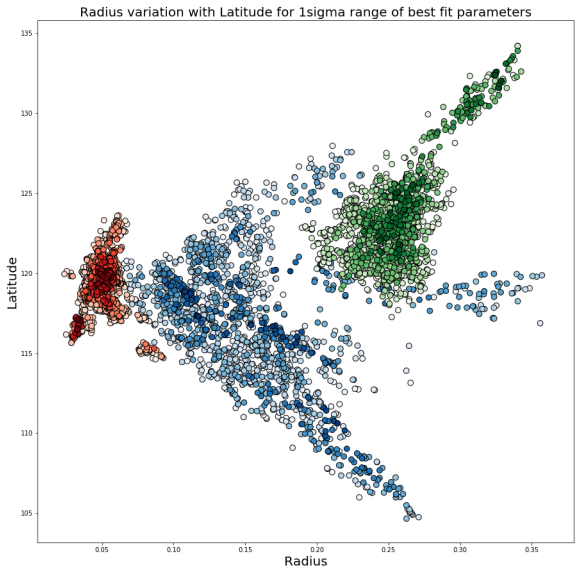


Figure 22: The trend of the radius of the spot and the latitude of the spot for the best fit solution for Quarter 9 Transit 3. The darker color of the points is associated with the chi squared value of the point, with a darker point representing a better chi squared value. Note that the latitude corresponds to the distance from the top of the star, meaning a larger latitude value corresponds to spot located lower on the surface of the star.

It is clear based on the STSP fits and models of the star spots the contrast of the spot must be at least darker than 0.1 (See Appendix, additional plots). This sort of contrast is much darker than the standard sun spot. Further, the size of the spots are larger than what would be expected. While there are transits with smaller spots (See Appendix), there are other transits with spots that are as large or larger, such as those of Quarter 8 Transit 1.

These results overall suggest that KOI-1786 is highly active with a highly turbulent and strong magnetic field, if dynamo theory is presumed to hold in this case. Further, the star boasts multiple flares, some of which are in the transit and shown in the Appendix section as well. These flares are further indicators of turbulent magnetic fields on KOI-1786, breaking the mold of what is to be expected on Red Giants. However, there are some clues as to the activity in the status of KOI-1786 as an active binary star.

A certain designation of binary stars is the RS Canus Venatici (RSCVn) binaries. [22]. These are binaries with one active star and one less active star, and include one main sequence star and one sub-giant or giant star [22]. In order to be considered a RSCVn binary, Montesinos, Giménez and Fernández-Figueroa (1988) note that it must include three key features. These are emission in CaII,

H, and K, activity markers in periodic variation in brightness, and an evolved active component with spectral type F, G, or K [22].

KOI-1786 meets these qualifications, as it consists of an active giant of spectral type K, star spot crossing features, and large flares (a proxy for CaII, H, and K emission lines). While we have yet to determine if the giant is in fact the active component quantitatively, qualitatively the size of the flares and the low luminosity of the M dwarf companion compared to the host suggests it is unlikely that the M dwarf is the culprit of the flaring or the spots themselves, meaning that it is safe to assume KOI-1786 fits in the classification of an RSCVn.

Though not much is known about stellar activity compared to the sun, certain markers and characteristic can give us an idea about the amount of stellar activity, and dynamo theory can be taken to be approximately true for the sake of predicting the expected amount of activity on different types of stars [1]. In general, activity seems to be linked to the rotation speed of a star and particularly the differential rotation of a star [4]. In the case of EBs, however, the primary star is not always rapidly rotating.

Notably, RSCVn type stars often have activity levels that would not correspond to the activity of single stars of the same spectral type, rotation period etc. [5]. This is where the classification of RSCVn stars becomes useful, as most systems of this classification have the larger, slower rotating primary as the active component[5]. This is due to interactions with the companion, not the properties of the primary itself [5].

While we do not have data on the companion due to its small size and lack of detectable secondary eclipse, the over-activity of the primary in this RSCVn system suggests that the secondary's effects are still notable and fit within the trend of stars of this type.

KOI-1786 is particularly interesting due to its status as a long-period CAB. There is a wealth of literature discussing the activity on short period CABs, but discussions on the activity of long-period CABs is much more sparse. Since the over-activity of these binaries with short rotation periods can be somewhat ascribed to the short periods and rapid rotation of the secondary [8]. There are more short and medium period CABs and thus, more information [8]. Thus, examining some of the activity and the parameters of KOI-1786 may provide additional insight into the activity of the long-period systems in this designation.

If we are to assume that the spots on the surface of KOI-1786 suggest the same properties as sun spots do for the sun, then it seems that KOI-1786 must be highly active and possess an extremely turbulent magnetic field. This is evident in the amount of flaring, the amount and size of the spots, and the contrast of the spots. Since the spots are very dark, it would suggest that there are

very strong local magnetic fields, and the size and amount of the spots suggest that the magnetic field is very turbulent for an extensive period of time (at least four years based on the amount of time Kepler observed this object). Finally, the presence of a massive flare near the time of the companion's transit across a massively spotted area on Quarter 8 Transit 1 suggests that similar to the sun, KOI-1786 possesses active regions associated with the spot clusters.

All of this activity is especially interesting in the context of KOI-1786's slow rotation. Since the slow rotation period of KOI-1786 would suggest limited differential rotation and thus, limited activity, this giant is breaking the expected activity level. This may be explained by the presence of the companion, however, the companion similarly has a long rotation period, unlike the majority of RSCVn binaries that have been studied in the context of stellar activity.

While there is somewhat limited discussion of long period CABs in general, there is at least one other instance of another red giant with a small companion. This giant, PZ Mon, is a similar binary to KOI-1786 which shows large, cool spots across the poles or else some other mechanism of a cool photosphere on the primary star[25]. These spots, however, are polar spots which are somewhat more common on large stars[25].

The interesting aspect of KOI-1786 is that the spots analyzed are not polar spots, based on our observations and our analysis of the rotation of the star. The latitude at which the spots are found is closer to the equator than the poles, meaning these large, dark spots are thus still remarkable in size and contrast. However, it seems that large cool spots across the surface of red giants in RSCVn binaries is not necessarily an isolated case[22].

In order to determine possible mechanisms of interactions between the giant and companion, we turn to analyzing the closest point of the orbit for KOI-1786. Using the equation below, we approximate the distance of closest contact using the derived orbit parameters and applying standard equations governing elliptical orbits.

$$Perihelion = a * (1 - e) = 6.366\ R_*$$

$$Aphelion = a * (1 + e) = 11.823\ R_*$$

This discrepancy in the perihelion and aphelion is directly associated with the eccentric orbit of KOI-1786's companion around the primary. The perihelion is the point of closest contact, and is $6.366 R_*$. This is 18950649.381 km or 0.12 AU, meaning the closest point of contact between the

companion and the primary is very close. This suggests that the companion could still be the culprit of the magnetic activity of the primary due to interactions between the two when the companion passes nearby the host.

Other future considerations for this project include confirming the flaring activity is associated with the primary and not the secondary quantitatively. As mentioned earlier, the size flares (Particularly in Quarter 8 Transit 1) suggests the primary is the active component of this system. However, a quantitative analysis of this parameter is necessary to concretely establish that KOI-1786 is an RSCVn binary.

Further, there are numerous other transits with large spot crossing features similar to those of Q9T3 and Q8T1. These, however, are less isolated than the spots on the transits we analyzed and so the process of spot modeling is considerably more involved. Every single transit we found in the Kepler data has multiple spot crossing features, meaning it may be possible to map the progression and emergence of spot features on this slow rotating giant. A full analysis of these spots may then allow us to determine what dynamo theory for long period CABs may look like and how the companion affects the emergence of spots and magnetic field strength.

References

[1] Svetlana V. Berdyugina. "Starspots: A Key to the Stellar Dynamo". In: *Living Reviews in Solar Physics* 2.1 (2005), p. 8. DOI: 10.12942/lrsp-2005-8. URL: https://doi.org/10.12942/lrsp-2005-8.

[2] Juan M. Borrero and Kiyoshi Ichimoto. "Magnetic Structure of Sunspots". In: *Living Reviews in Solar Physics* 8.1 (2011), p. 4. DOI: 10.12942/lrsp-2011-4. URL: https://doi.org/10.12942/lrsp-2011-4.

[3] Dale A. Ostlie Bradley W. Carroll. *An Introduction to Modern Astrophyscis, Second Edition*. Caimbridge University Press, 2017. ISBN: 9781108422161.

[4] Allan Sacha Brun and Matthew K. Browning. "Magnetism, dynamo action and the solar-stellar connection". In: *Living Reviews in Solar Physics* 14.1 (2017), p. 4. DOI: 10.1007/s41116-017-0007-8. URL: https://doi.org/10.1007/s41116-017-0007-8.

[5] Cornelius Zwaan Carolus J. Schrijver. *Solar and Stellar Magnetic Activity*. Cambridge Astrophysics Series. Cambridge University Press, 2000. ISBN: 9780521739863.

[6] P. F. Chen. "Coronal Mass Ejections: Models and Their Observational Basis". In: *Living Reviews in Solar Physics* 8.1 (2011), p. 1. DOI: 10.12942/lrsp-2011-1. URL: https://doi.org/10.12942/lrsp-2011-1.

[7] A. Collier Cameron et al. "Efficient identification of exoplanetary transit candidates from SuperWASP light curves". In: 380.3 (Sept. 2007), pp. 1230–1244. DOI: 10.1111/j.1365-2966.2007.12195.x. arXiv: 0707.0417 [astro-ph].

[8] Z. Eker et al. "A catalogue of chromospherically active binary stars (third edition)". In: *Monthly Notices of the Royal Astronomical Society* 389.4 (Sept. 2008), pp. 1722–1726. DOI: 10.1111/j.1365-2966.2008.13670.x. URL: https://doi.org/10.1111%2Fj.1365-2966.2008.13670.x.

[9] *File:light curve of binary star kepler-16 af.svg*. 2013. URL: https://commons.wikimedia.org/wiki/File:Light_curve_of_binary_star_Kepler-16_af.svg.

[10] Samir Hamouda. "Sunspots Production and Relation to Other Phenomena: A Review". In: (Aug. 2020).

[11] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[12] Carole A Haswell. *Transiting Exoplanets*. Cambridge University Press, 2010. ISBN: 9780521139380.

[13] David Hathaway. *NASA/Marshall Solar Physics*. URL: https://solarscience.msfc.nasa.gov/dynamo.shtml.

[14] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[15] Michele Johnson. *Mission overview*. Apr. 2015. URL: https://www.nasa.gov/mission_pages/kepler/overview/index.html.

[16] Laura Kreidberg. "batman: BAsic Transit Model cAlculatioN in Python". In: 127.957 (Nov. 2015), p. 1161. DOI: 10.1086/683602. arXiv: 1507.08285 [astro-ph.EP].

[17] Kenneth Lang. *A Magnetic Star: The photosphere and its magnetism*. 2010. URL: https://ase.tufts.edu/cosmos/view_chapter.asp?id=26&page=2.

[18] Lightkurve Collaboration et al. *Lightkurve: Kepler and TESS time series analysis in Python*. Astrophysics Source Code Library. Dec. 2018. ascl: 1812.013.

[19] Wes McKinney et al. "Data structures for statistical computing in python". In: *Proceedings of the 9th Python in Science Conference*. Vol. 445. Austin, TX. 2010, pp. 51–56.

[20] A. McQuillan, T. Mazeh, and S. Aigrain. "STELLAR ROTATION PERIODS OF THE iKEPLER/i OBJECTS OF INTEREST: A DEARTH OF CLOSE-IN PLANETS AROUND FAST ROTATORS". In: *The Astrophysical Journal* 775.1 (Sept. 2013), p. L11. DOI: 10.1088/2041-8205/775/1/l11. URL: https://doi.org/10.1088/2041-8205/775/1/l11.

[21] Mark Moldwin. *Coronal mass ejection*. 2019. URL: https://www.britannica.com/science/coronal-mass-ejection.

[22] B. Montesinos, A. Gimenez, and M. J. Fernandez-Figueroa. "General properties of RS CVn systems." In: 232 (May 1988), pp. 361–376. DOI: 10.1093/mnras/232.2.361.

[23] *Nature of the Universe, Chapter 11: The Sun*. URL: http://lifeng.lamost.org/courses/Hongkong/Hongkong_En/lecture/ch11/ch11.html.

[24] F. Ochsenbein, P. Bauer, and J. Marcout. "The VizieR database of astronomical catalogues". In: 143 (Apr. 2000), pp. 23–32. DOI: 10.1051/aas:2000169. arXiv: astro-ph/0002122 [astro-ph].

[25] Yu. V. Pakhomov et al. "Cool Spots on the Surface of the Active Giant PZ Mon". In: *Astronomy Letters* 45.3 (Mar. 2019), pp. 156–163. DOI: 10.1134/s1063773719030058. URL: https://doi.org/10.1134%2Fs1063773719030058.

[26] Sean G. Ryan and Andrew J. Norton. *Stellar evolution and nucleosynthesis*. Cambridge University Press, 2010.

[27] Keivan G. Stassun et al. "The Revised TESS Input Catalog and Candidate Target List". In: 158.4, 138 (Oct. 2019), p. 138. DOI: 10.3847/1538-3881/ab3467. arXiv: 1905.10694 [astro-ph.SR].

[28] Manuela Temmer. "Space weather: the solar perspective". In: *Living Reviews in Solar Physics* 18.1 (2021), p. 4. DOI: 10.1007/s41116-021-00030-3. URL: https://doi.org/10.1007/s41116-021-00030-3.

[29] *The Sun's magnetic field is about to flip: Science wire*. Aug. 2013. URL: https://earthsky.org/science-wire/the-suns-magnetic-field-is-about-to-flip/.

[30] Susan E. Thompson et al. "Planetary Candidates Observed by Kepler. VIII. A Fully Automated Catalog with Measured Completeness and Reliability Based on Data Release 25". In: 235.2, 38 (Apr. 2018), p. 38. DOI: 10.3847/1538-4365/aab4f9. arXiv: 1710.06758 [astro-ph.EP].

[31] Shin Toriumi and Haimin Wang. "Flare-productive active regions". In: *Living Reviews in Solar Physics* 16.1 (2019), p. 3. DOI: 10.1007/s41116-019-0019-7. URL: https://doi.org/10.1007/s41116-019-0019-7.

[32] Eric Weisstein. *Incomplete gamma function*. URL: https://mathworld.wolfram.com/IncompleteGammaFuncti
html.

[33] M. Wenger et al. "The SIMBAD astronomical database. The CDS reference database for astronomical objects". In: 143 (Apr. 2000), pp. 9–22. DOI: 10.1051/aas:2000332. arXiv: astro-ph/0002110 [astro-ph].

4 Appendix

4.1 Additional Plots

4.1.1 Quarter 9 Lower Contrast Spots

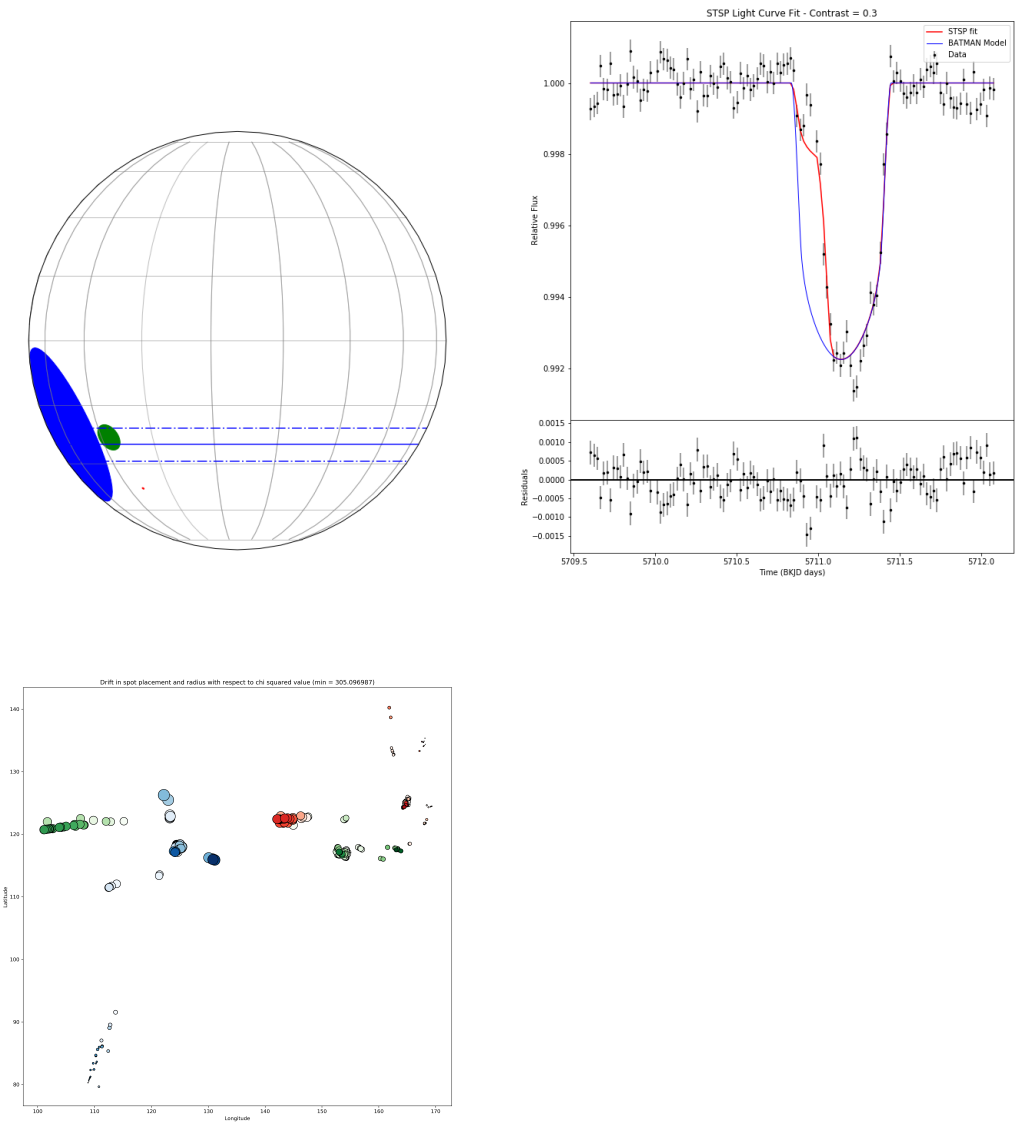


Figure 23: Spot parameters for Quarter 9 Transit 3, Contrast used was 0.3.

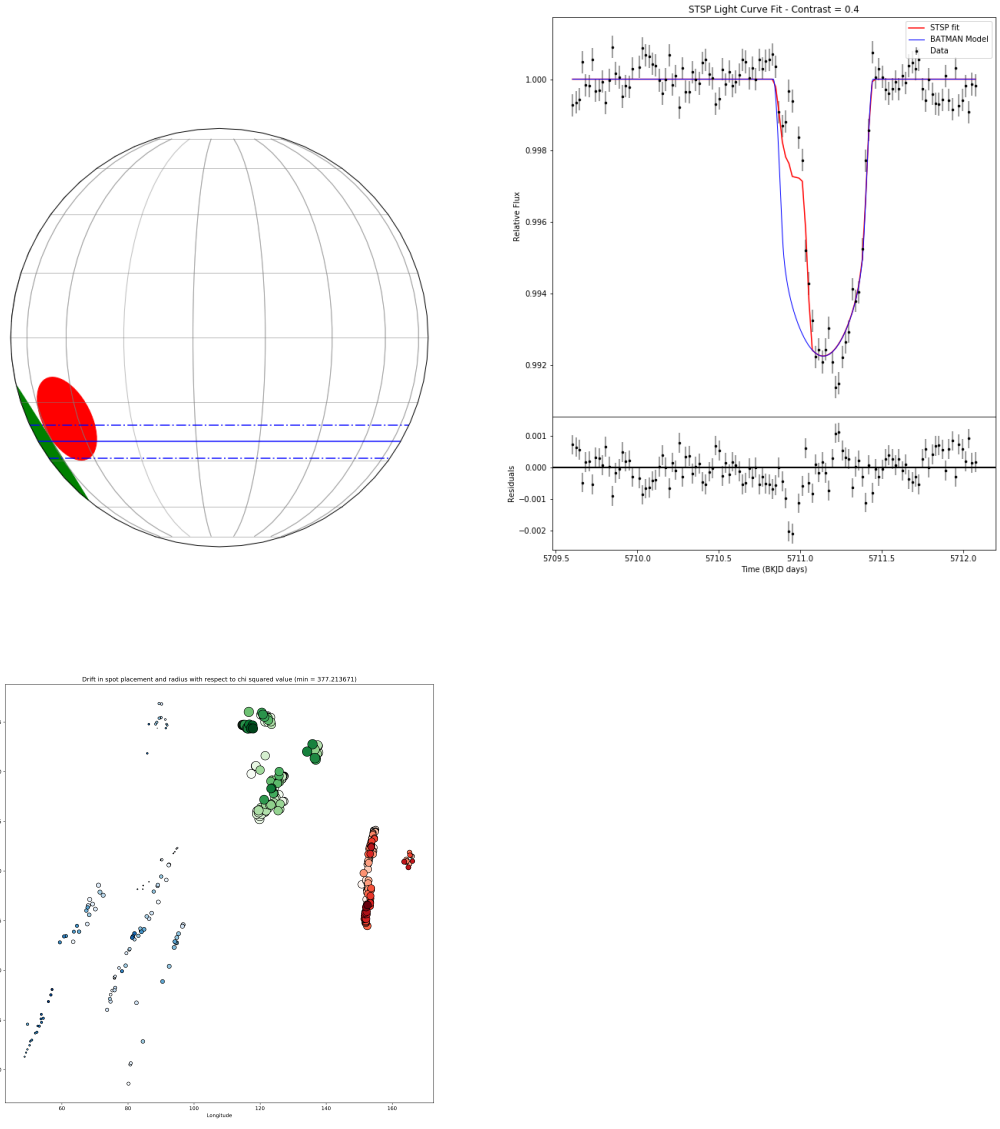


Figure 24: Spot parameters for Quarter 9 Transit 3, Contrast used was 0.4.

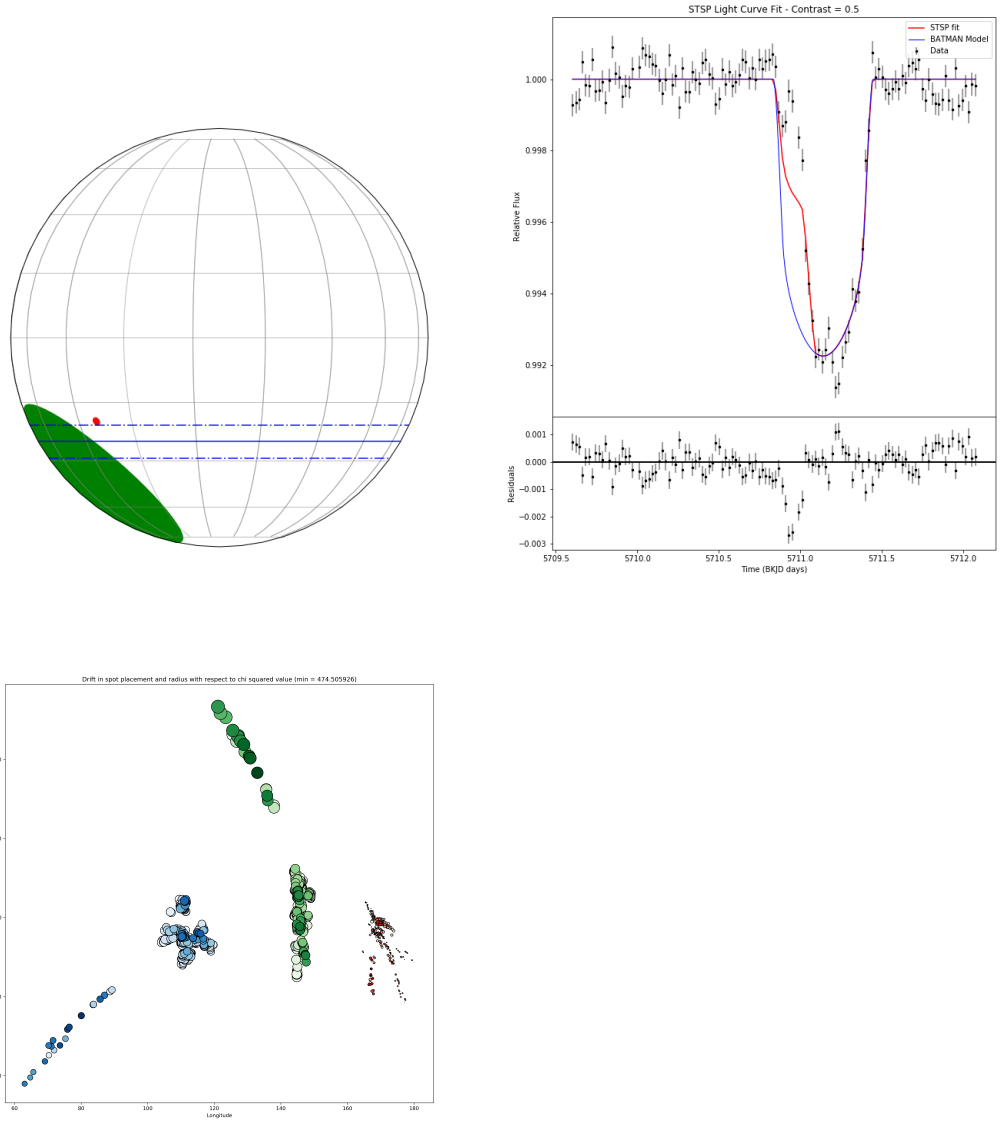
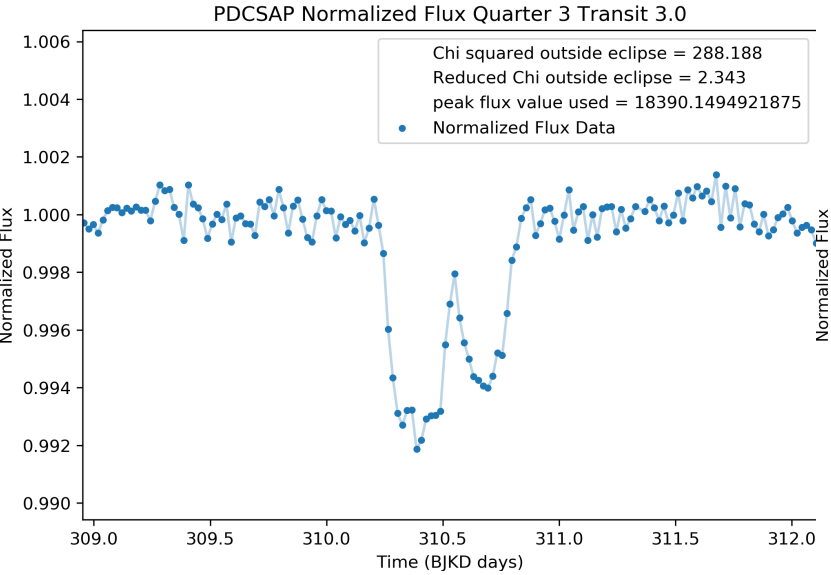
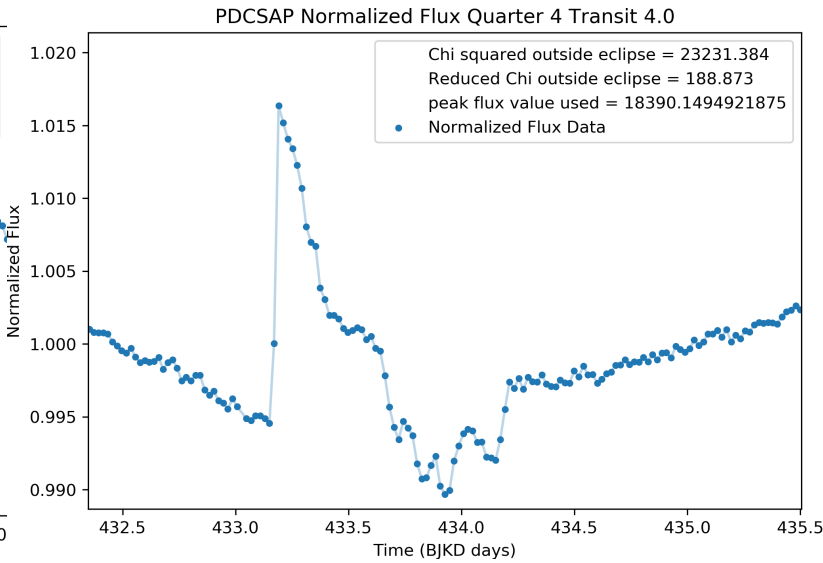
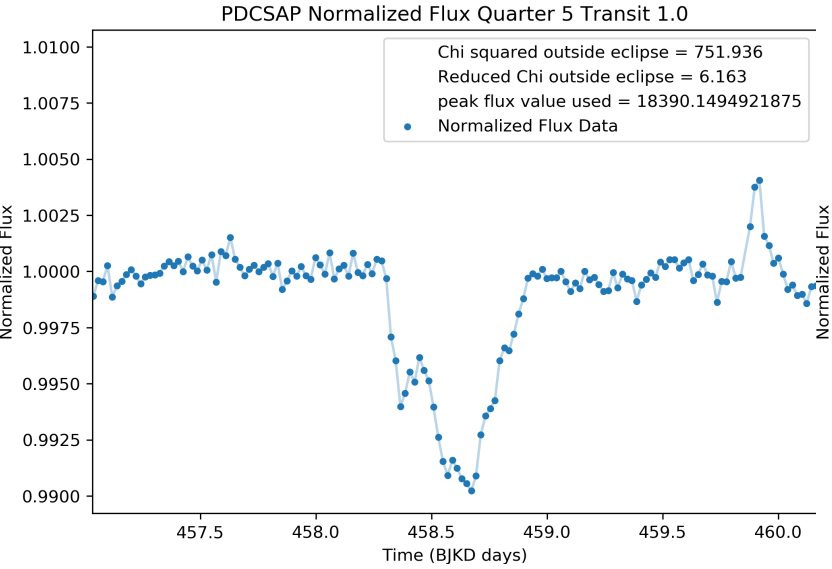


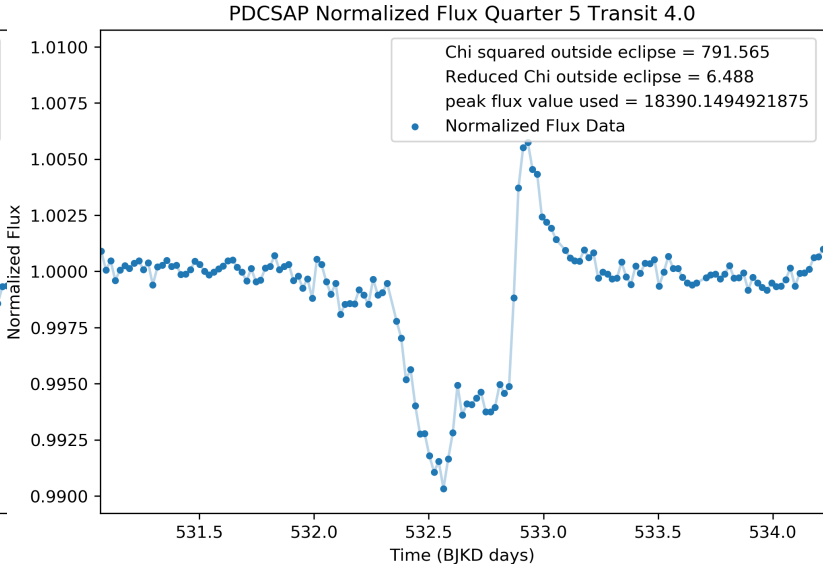
Figure 25: Spot parameters for Quarter 9 Transit 3, Contrast used was 0.5.

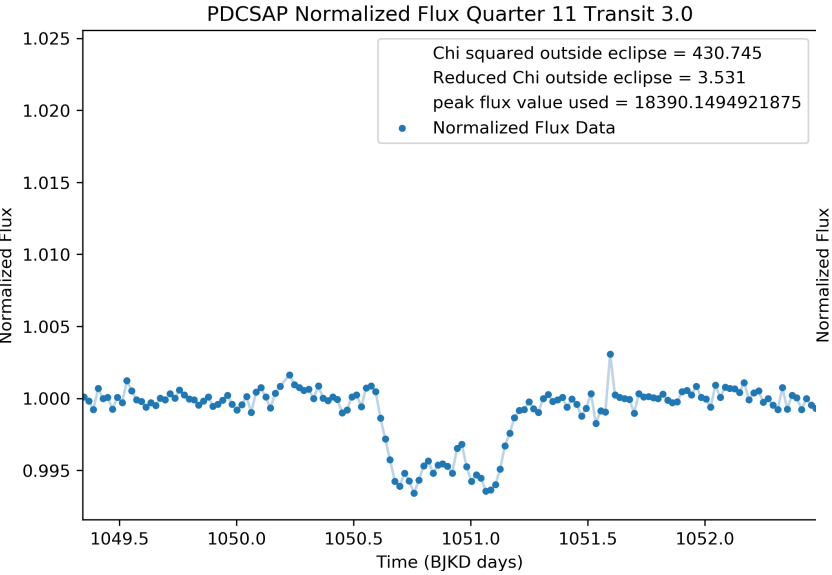
4.1.2 Select Transits with Flare Features

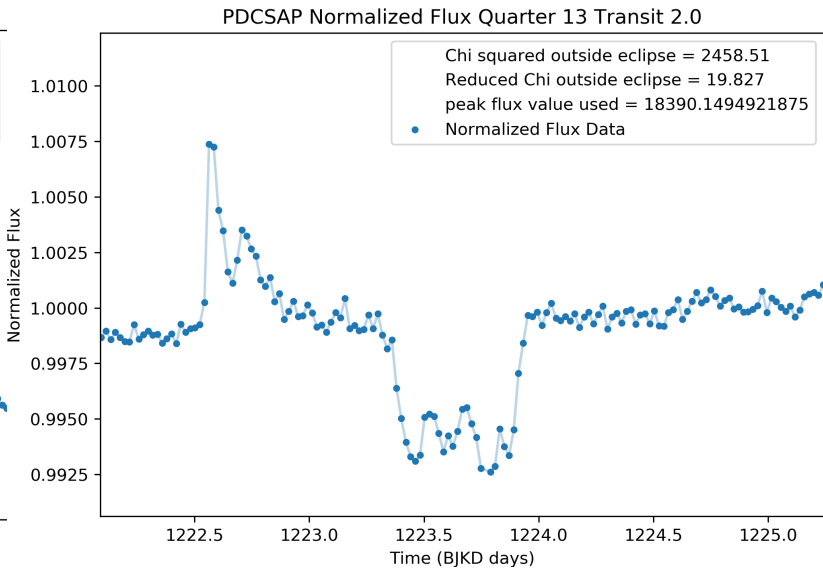

PDCSAP Normalized Flux Quarter 3 Transit 3.0
Chi squared outside eclipse = 288.188
Reduced Chi outside eclipse = 2.343
peak flux value used = 18390.1494921875
Normalized Flux Data
Normalized Flux
Time (BJKD days)

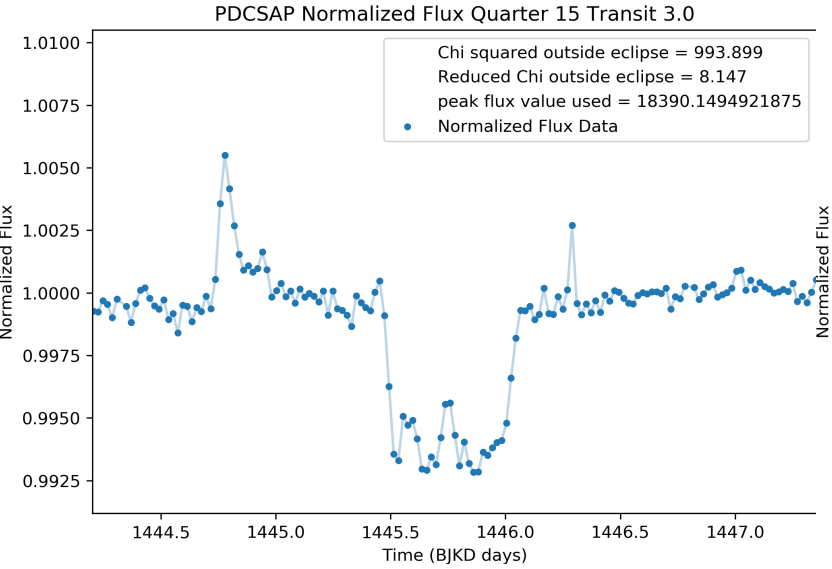

PDCSAP Normalized Flux Quarter 4 Transit 4.0
Chi squared outside eclipse = 23231.384
Reduced Chi outside eclipse = 188.873
peak flux value used = 18390.1494921875
Normalized Flux Data
Normalized Flux
Time (BJKD days)

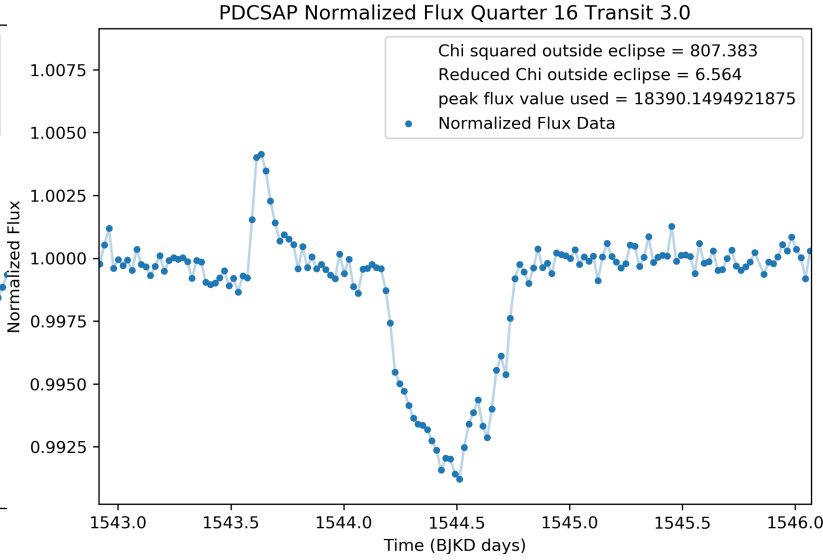

PDCSAP Normalized Flux Quarter 5 Transit 1.0
Chi squared outside eclipse = 751.936
Reduced Chi outside eclipse = 6.163
peak flux value used = 18390.1494921875
Normalized Flux Data
Normalized Flux
Time (BJKD days)


PDCSAP Normalized Flux Quarter 5 Transit 4.0
Chi squared outside eclipse = 791.565
Reduced Chi outside eclipse = 6.488
peak flux value used = 18390.1494921875
Normalized Flux Data
Normalized Flux
Time (BJKD days)

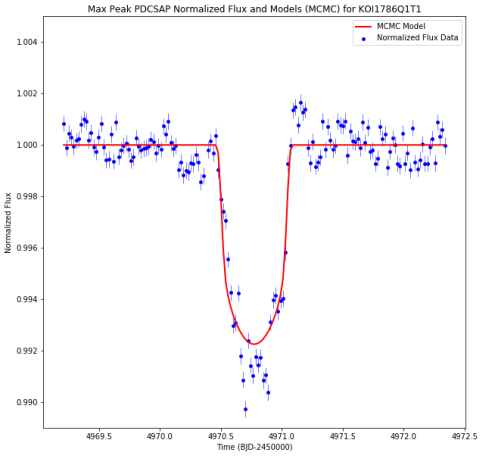

PDCSAP Normalized Flux Quarter 11 Transit 3.0
Chi squared outside eclipse = 430.745
Reduced Chi outside eclipse = 3.531
peak flux value used = 18390.1494921875
Normalized Flux Data
Normalized Flux
Time (BJKD days)

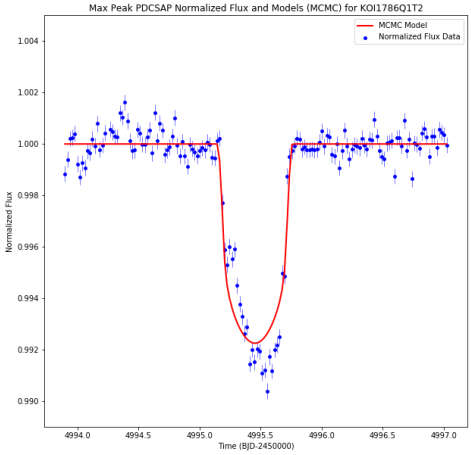

PDCSAP Normalized Flux Quarter 13 Transit 2.0
Chi squared outside eclipse = 2458.51
Reduced Chi outside eclipse = 19.827
peak flux value used = 18390.1494921875
Normalized Flux Data
Normalized Flux
Time (BJKD days)

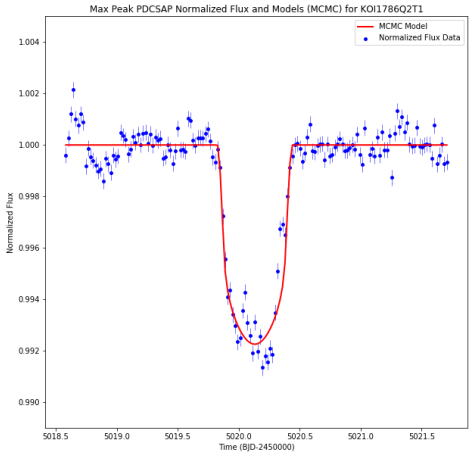

PDCSAP Normalized Flux Quarter 15 Transit 3.0
Chi squared outside eclipse = 993.899
Reduced Chi outside eclipse = 8.147
peak flux value used = 18390.1494921875
Normalized Flux Data
Normalized Flux
Time (BJKD days)

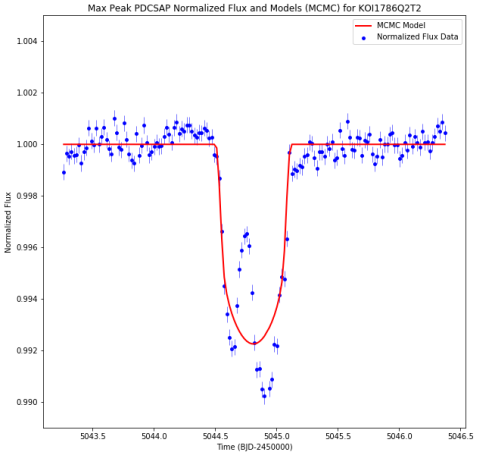

PDCSAP Normalized Flux Quarter 16 Transit 3.0
Chi squared outside eclipse = 807.383
Reduced Chi outside eclipse = 6.564
peak flux value used = 18390.1494921875
Normalized Flux Data
Normalized Flux
Time (BJKD days)

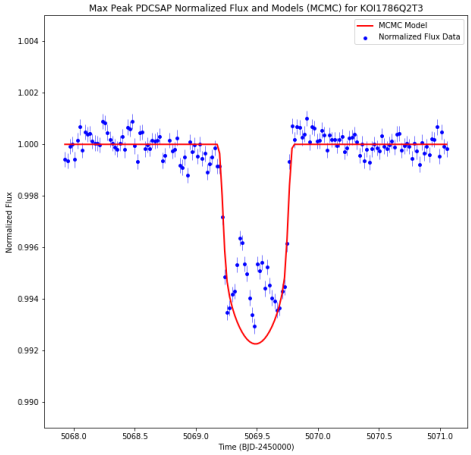
4.2 All Normalized Transits and Models

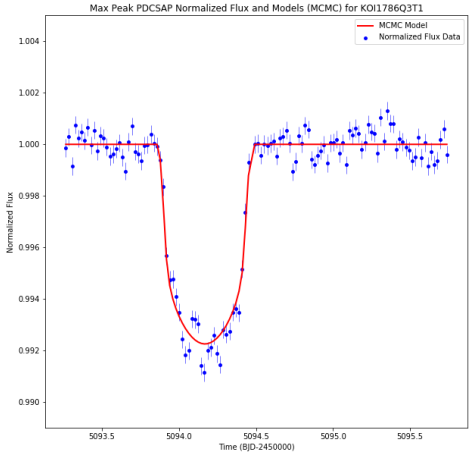

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q1T1
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

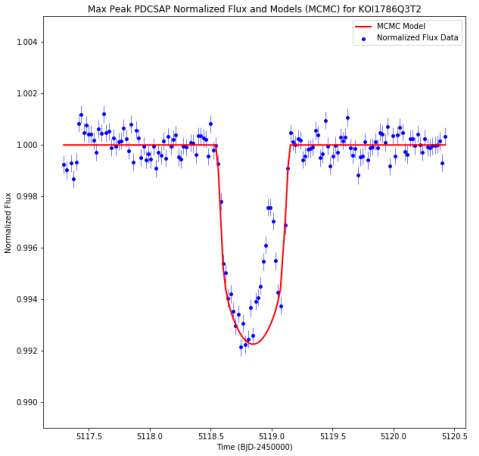

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q1T2
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

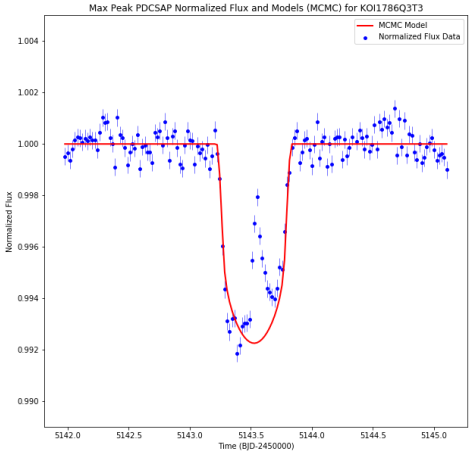

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q2T1
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

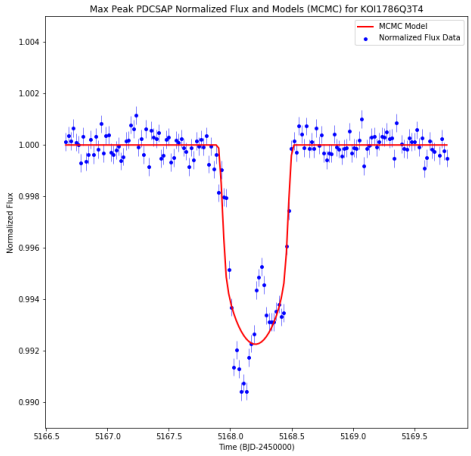

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q2T2
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

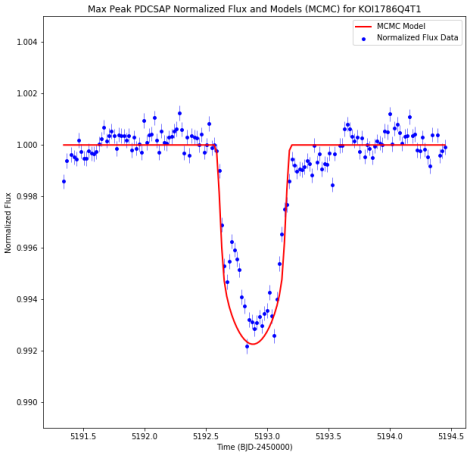

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q2T3
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

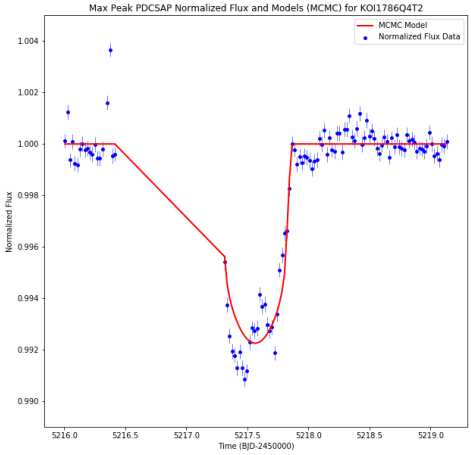

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q3T1
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

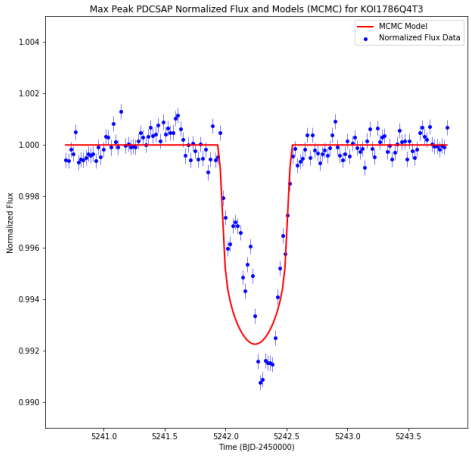

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q3T2
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

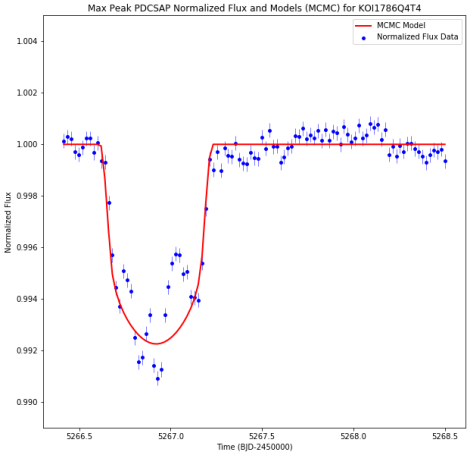

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q3T3
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

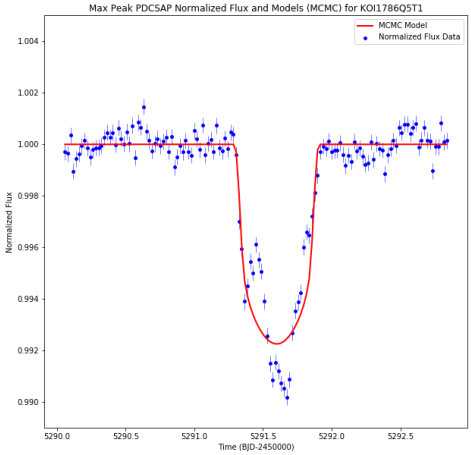

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q3T4
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

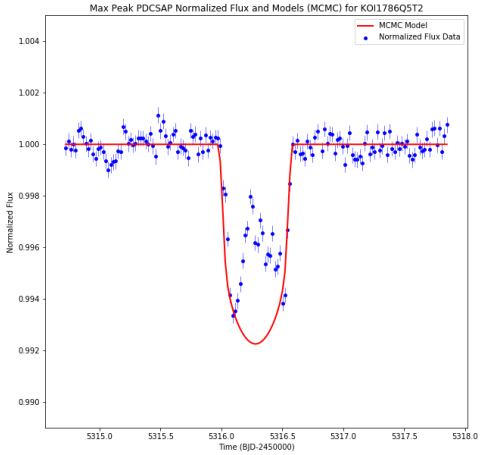

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q4T1
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

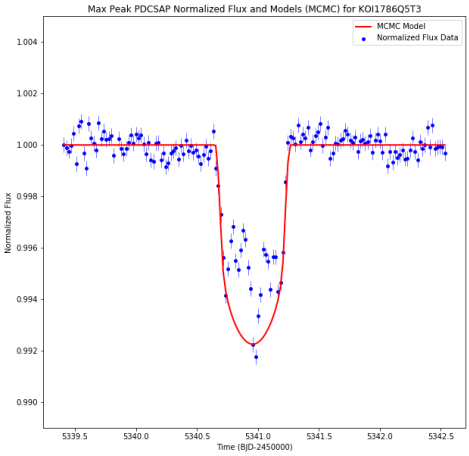

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q4T2
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

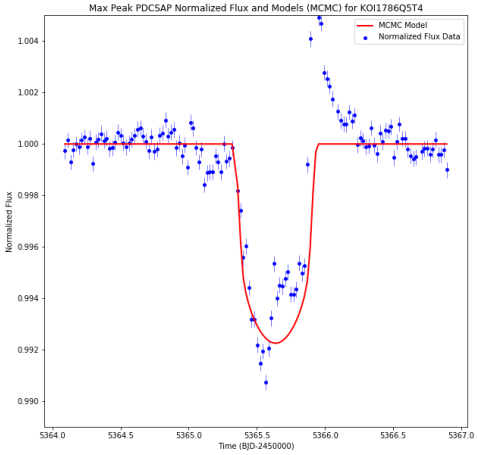

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q4T3
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

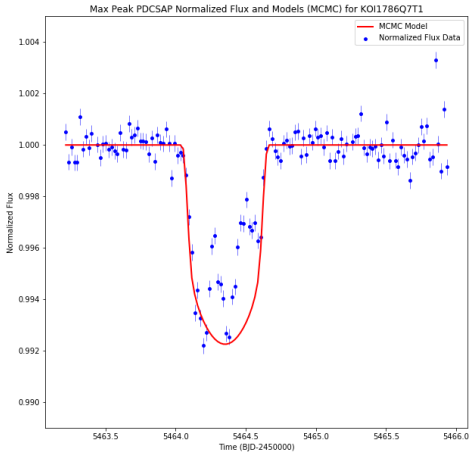

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q4T4
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

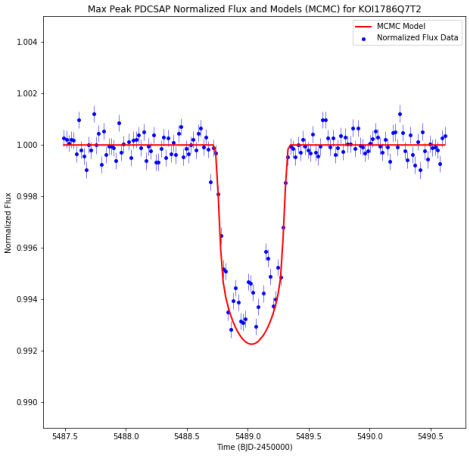

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q5T1
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

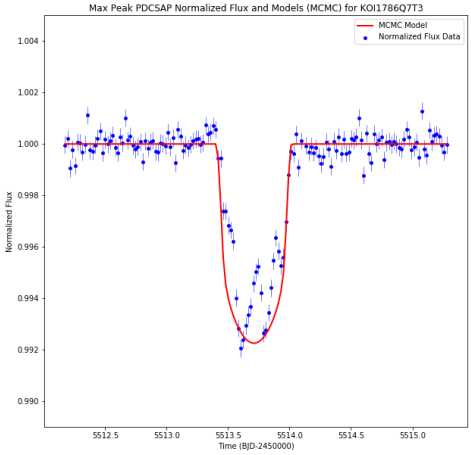

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q5T2
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

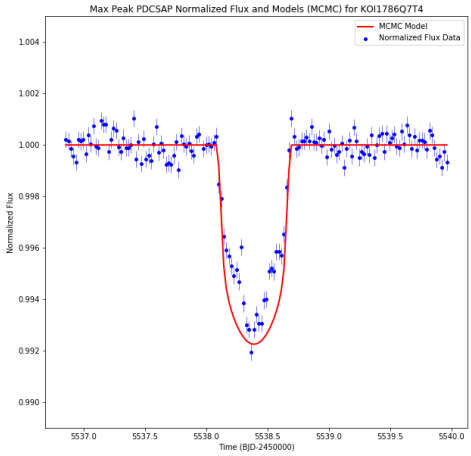

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q5T3
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

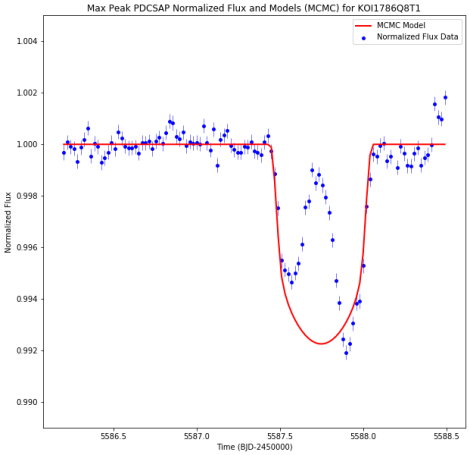

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q5T4
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

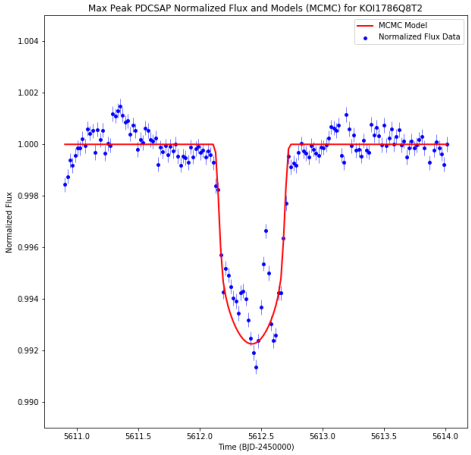

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q7T1
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

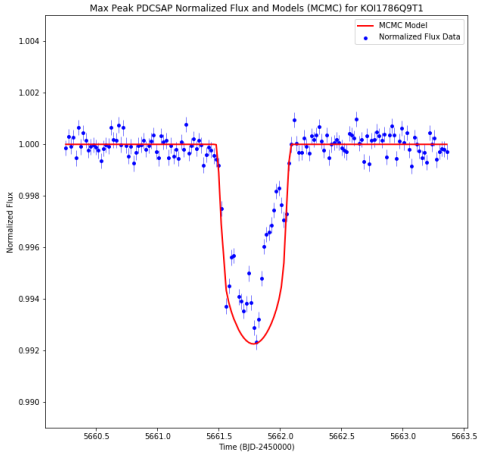

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q7T2
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

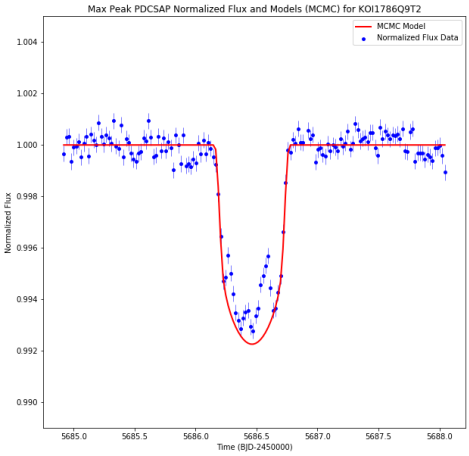

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q7T3
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

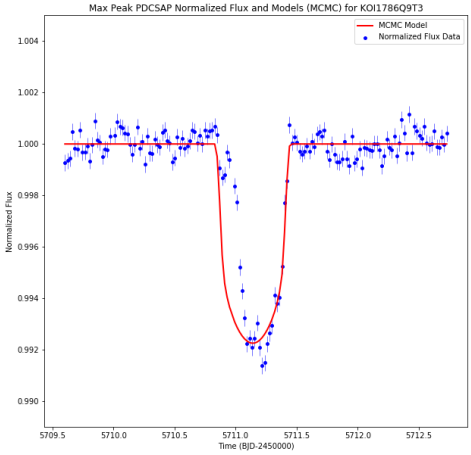

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q7T4
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

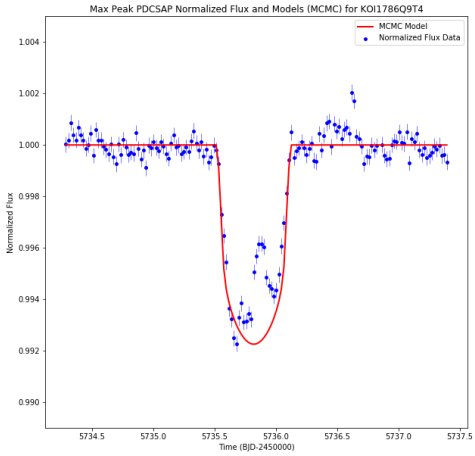

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q8T1
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

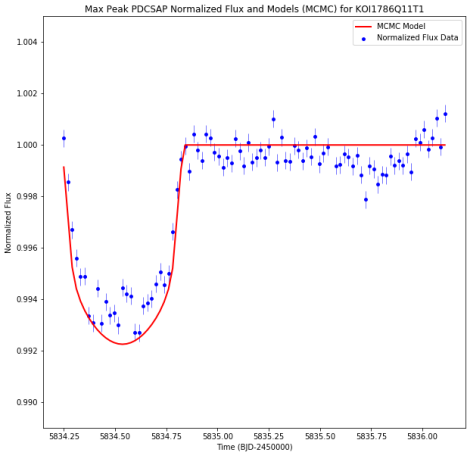

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q8T2
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

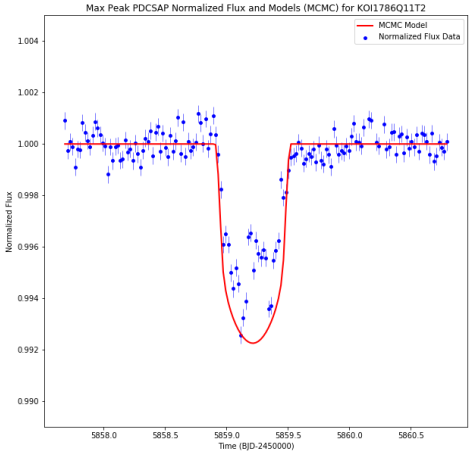

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q9T1
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

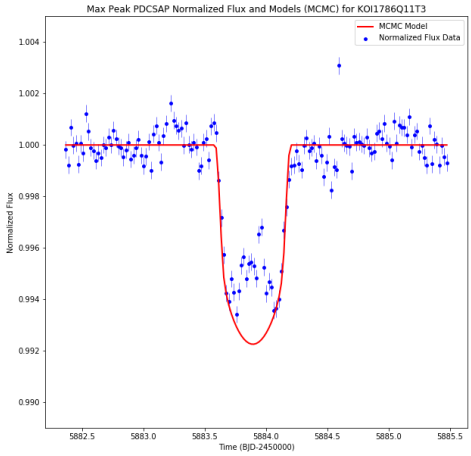

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q9T2
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

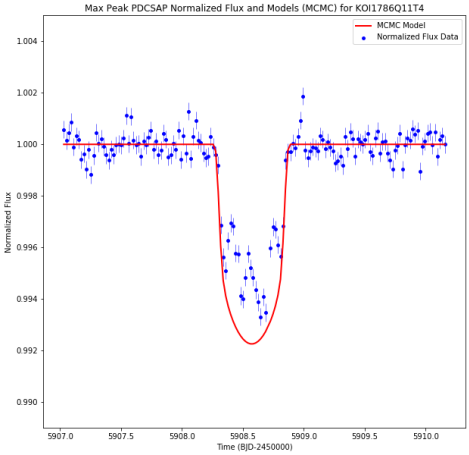

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q9T3
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

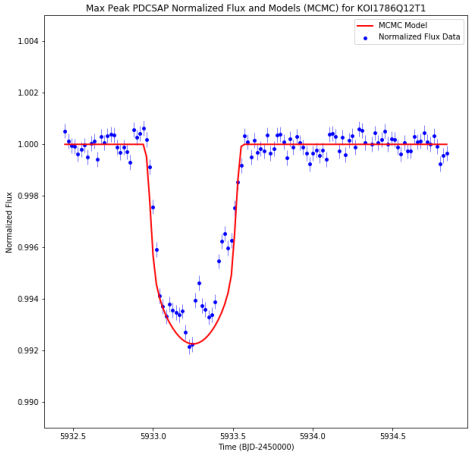

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q9T4
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

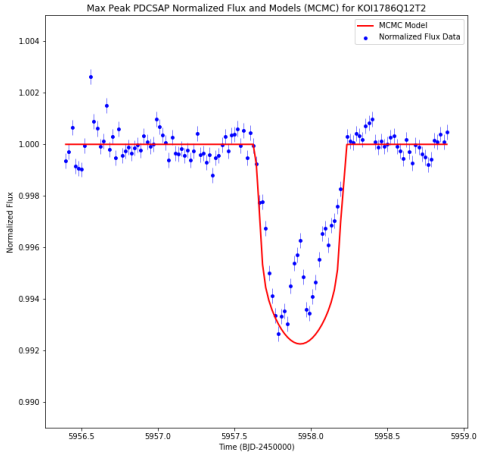

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q11T1
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

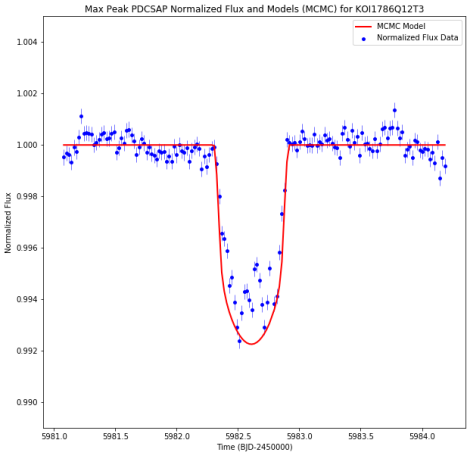

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q11T2
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

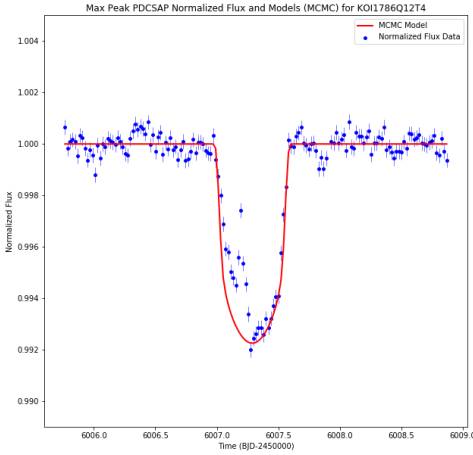

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q11T3
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

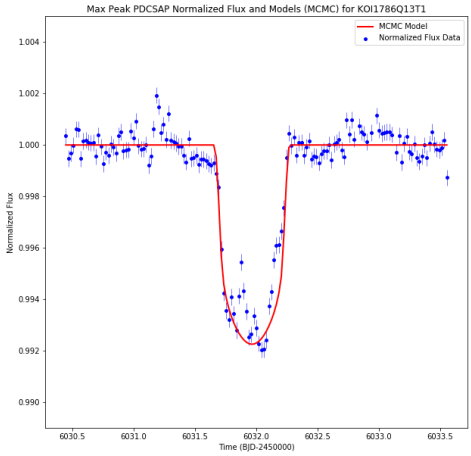

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q11T4
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

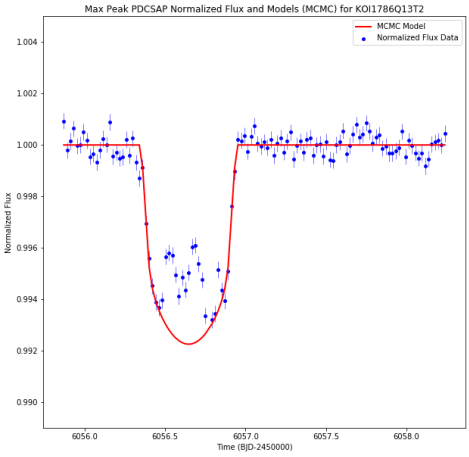

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q12T1
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

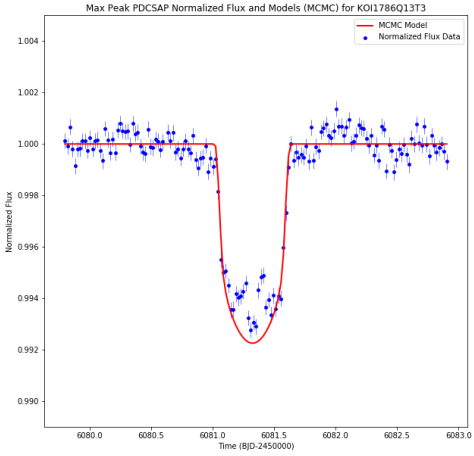

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q12T2
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

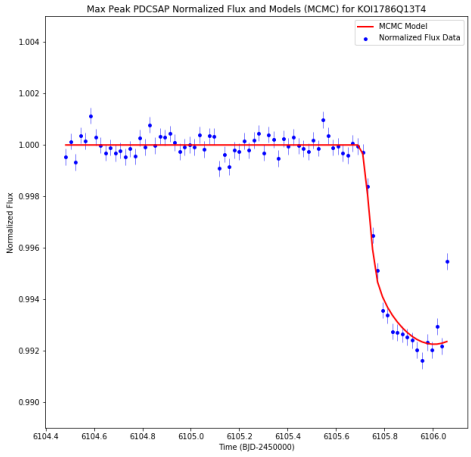

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q12T3
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

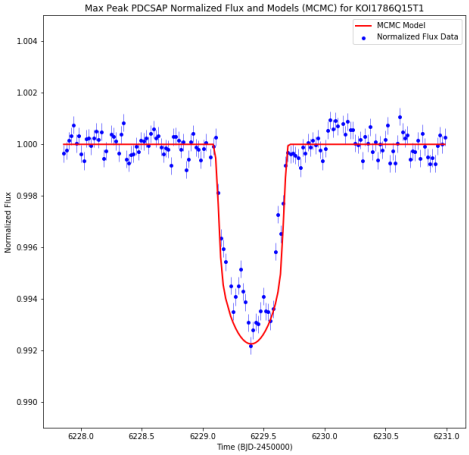

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q12T4
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

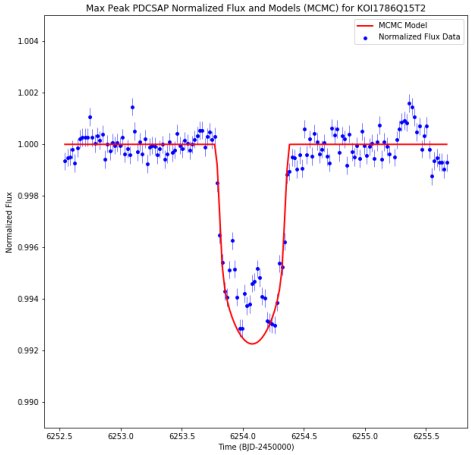

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q13T1
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

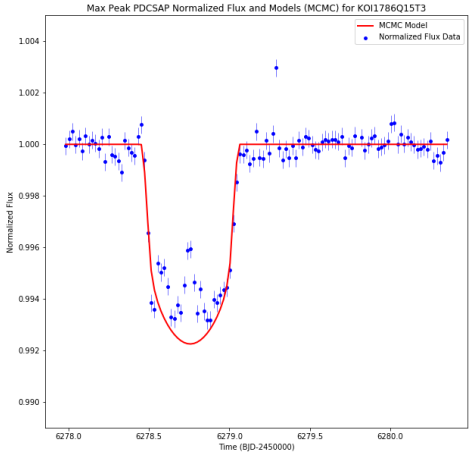

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q13T2
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

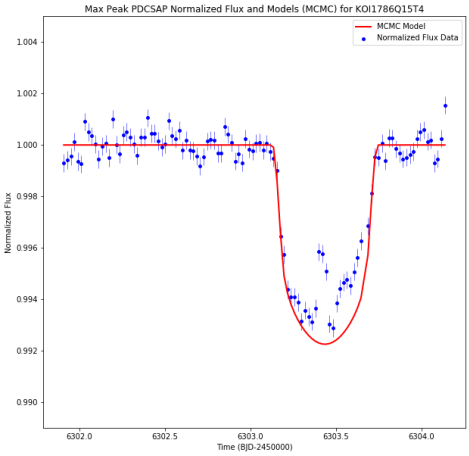

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q13T3
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

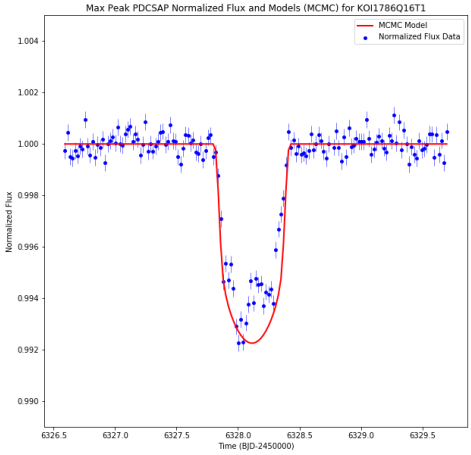

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q13T4
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

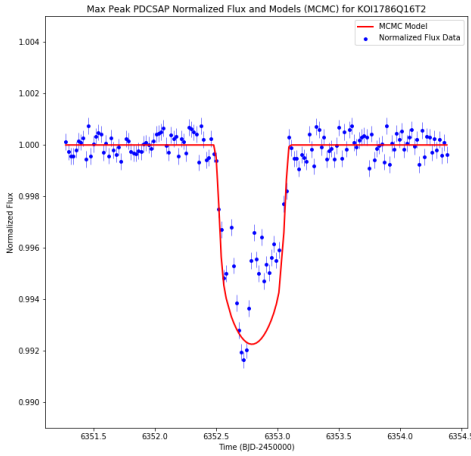

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q15T1
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

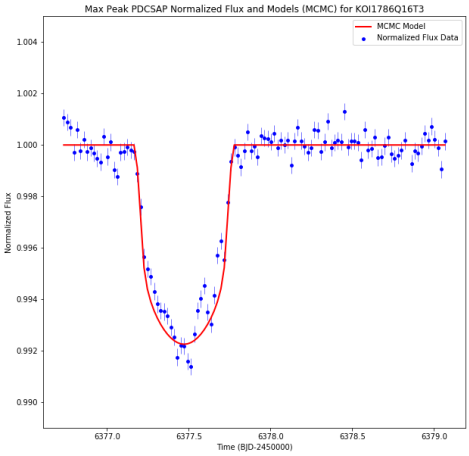

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q15T2
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

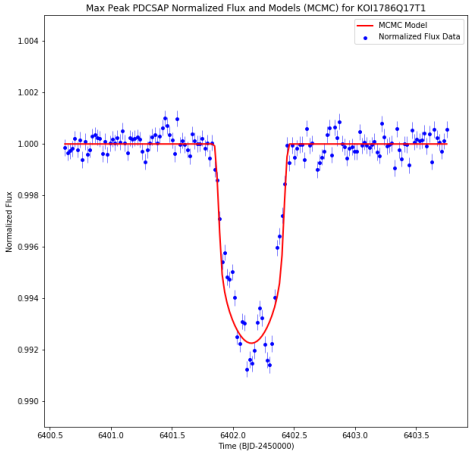

Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q15T3
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)


Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q15T4
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)


Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q16T1
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)


Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q16T2
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)


Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q16T3
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)


Max Peak PDCSAP Normalized Flux and Models (MCMC) for KOI1786Q17T1
MCMC Model
Normalized Flux Data
Normalized Flux
Time (BJD-2450000)

4.3 Code

4.3.1 Normalizing Transits

```
#!/usr/bin/env python
# coding: utf-8

# In[17]:


#####Importing all necessary programs
import numpy as np
import pandas as pd
from pandas import DataFrame as df
from matplotlib import pyplot as plt
from lightkurve import search_lightcurvefile
import seaborn as sns
import scipy as sp
import statistics
import batman
import juliet
#removes warnings for polyfit if using a dataframe, currently a bug that
    ↪ needs fixing in general
pd.options.mode.chained_assignment = None

from IPython.core.display import display, HTML
display(HTML("<style>.container␣{␣width:95%␣!important;␣}</style>"))
display(HTML("<style>.rendered_html␣{␣font-size:␣18px;␣}</style>"))
#display


# In[18]:


#####this is for all of the values that need to be input manually
#KIC ID number or TESS ID
#period, epoch, and duration for this EB
#the epoch, duration, and period were all retrieved from the kepler
    ↪ villanova database

#Kepler 63:
# IDnum = 'KIC 11554435'
# epoch = (55010.843+2400000-2454833) #(54970.54 +2400000-2454833)##bjd0
    ↪ +2400000-2454833
# period = (9.43414171)
# duration = 2.8112/24

#Kepler 17:
#IDnum = 'KIC 10619192'
#epoch = (54965.794+2400000-2454833) #(54970.54 +2400000-2454833)##bjd0
    ↪ +2400000-2454833
#period = (1.485710994)
#duration = 2.2874/24 #pwidth

#HAT-P-11:
# IDnum = 'KIC 10748390'
# epoch = (54957.813+2400000-2454833) #(54970.54 +2400000-2454833)##bjd0
    ↪ +2400000-2454833
```

```
# period = (4.887803076)
# duration = 2.36255/24

# #KOI-1786

#DR-25
IDnum = 'KIC 3128793'
duration = 15.168/24
epoch = (54970.770924+2400000-2454833) #villanova epoch
period = (24.6793813) #villanova period


#Desired Transit Parameters
#renorms: Q4T4, Q5T1, Q5T4, Q8T1, Q13T2, Q15T3, Q16T3 for KOI-1786
quarter_num = 8
#Transit_number = 1

#If short Cadence
month_number = 2

#Give a file path for finding and saving graphs and txts
save_path = '/Users/katelincrabtree/Desktop/Final_KOI1786_Transits/
    ↪ maxpeak_PDCSAP_Flux/'

#####this number will be multiplied by the duration to find the amount that
    ↪ needs to be included around each transit
#For most objects ~2.5 works well
#this can be adjusted to account for flares or other oddities
#amount before
bcut = 2.5
#amount after
ecut = 2.5

secondary_epoch = epoch+(period/2)
primary_epoch = epoch
###replace primary epoch with epoch--> change epoch for secondary if needed
#print(epoch)


# In[20]:


#renorms: Q4T4, Q5T1, Q5T4, Q8T1, Q13T2, Q15T3, Q16T3 for KOI-1786
#############importing the desired file, correcting it with pdcsap--->
    ↪ optional if file import program was already run
#############importing info from lightcurve
print('Have you run the file import program? (y/n)')
file_exist = input();

if file_exist == 'n':

    print('Would You like short cadence data or long cadence data? (s/l)')
    cadence = input();

    #if cadence == 's':
        #print('Okay, getting the file now. This may take a moment...')

        #file1 = search_lightcurvefile(IDnum, quarter = quarter_num, month =
            ↪ month_number, cadence = 'short').download() #sc data
```

```
        #file0 = search_lightcurvefile(IDnum, quarter = quarter_num, cadence
            ↪  = 'long').download() #lc data for peak
        #lcsap = file1.PDCSAP_FLUX
    if cadence == 'l':
        print('Okay, getting the file now. This may take a moment...')

        file1 = search_lightcurvefile(IDnum, quarter = quarter_num, cadence
            ↪ = 'long').download()
        file0 = search_lightcurvefile(IDnum, quarter = quarter_num, cadence
            ↪ = 'long').download() #lc data for peak
        lcsap = file1.PDCSAP_FLUX
    ########getting the sap files only
    #scsap= file1.SAP_FLUX

    #########turning them into dataframes
    #scframe = scsap.to_pandas()
    lcframe = lcsap.to_pandas()

    ####concat if needed for multiple frames and quarters, unneccesary here
    lightcurve_frame = lcframe #pd.concat([lcframe]) #lcframe1, lcframe2,
        ↪ lcframe3, lcframe4, lcframe5, lcframe7, lcframe8, lcframe9,
        ↪ lcframe11, lcframe12, lcframe13, lcframe15, lcframe16, lcframe17],
        ↪  axis=0)

    ######renaming if necessary
    lightcurve_frame.index.name = 'index'

    #####checking the frame
    #print(lightcurve_frame)

    #####dropping NaNs
    #print(lightcurve_frame.isna().sum())
    lightcurve_frame.dropna(inplace=True, how='any')


elif file_exist == 'y':
    print('okay, getting the file now...')
    #####If the importing program has already been run, then use this
        ↪ section instead
    lightcurve_frame = pd.read_csv(save_path+'lightcurve_frame' + 'Q' + str(
        ↪ quarter_num) + '.txt', sep = '\t', header = 0, names = ['time', '
        ↪ flux', 'flux_err', 'centroid_col', 'centroid_row'])

else:
    print('Im sorry, I dont understand. Please try again.')

#####adding new columns to the dataframe
orbit_number = np.floor(lightcurve_frame.time.sub(epoch).div(period)) #
    ↪ truncates with astype.int()
lightcurve_frame['orbit_number'] = orbit_number
phase = pd.DataFrame()
phase['phase_raw'] = lightcurve_frame.time.sub(epoch).div(period) -
    ↪ orbit_number
phase.loc[phase.phase_raw>0.4, 'phase_raw'] -= 1.0
lightcurve_frame['phase'] = phase.phase_raw

#for checking the code
#print(lightcurve_frame)
#print(lightcurve_frame)
```

```
# In[21]:


plt.scatter(lightcurve_frame.orbit_number, lightcurve_frame.flux)


# In[24]:


#####finding the number of eclipses in the dataframe
number_eclipses_min = lightcurve_frame['time'].min()
number_eclipses_max = lightcurve_frame['time'].max()
#print(number_eclipses_max)
#print(number_eclipses_min)

#####finding the primary and secondary epochs, as well as the first epoch
    ↪ for the primary and secondary
#####finding an integer number of eclipses for the later for loop
first_Pepoch = lightcurve_frame.orbit_number.iloc[1]*period+epoch  #first
    ↪ epoch isnt necessarily the first eclipse, find it with this
number_eclipses= np.floor(((number_eclipses_max-number_eclipses_min)/period)
    ↪ )+2

#print(number_eclipses)
#print(first_Pepoch)
#print("epoch = " + str(epoch))

#####peak is the point we use later for making sure the flux is normalized
    ↪ properly relative to the lightcurve of the star "without spots"
#In this particular case, we use maximum peak value across all quarters
peak = 18390.1494921875
###peak = np.percentile(lightcurve_frame.flux, 99.5)


#####Plottign the lightcurve and the peak as a point
plt.figure(figsize=(15, 10))
plt.plot(lightcurve_frame.time, lightcurve_frame.flux, label= 'quarter ' +
    ↪ str(quarter_num) + ' long cadence data')
#plt.plot(lightcurve_frame_lc.time, lightcurve_frame_lc.flux, color = 'r',
    ↪ label = 'long cadence')
plt.plot([np.min(lightcurve_frame.time), np.max(lightcurve_frame.time)], [
    ↪ peak, peak], c = 'm', label = 'Peak Flux Value (99.5th percentile avg)
    ↪ ')
plt.legend()#loc='center left')# bbox_to_anchor=(1, 0.5))
plt.xlabel('Time -2454833 (BJKD Days)', fontsize = 18)
plt.ylabel('Flux (-e/s)', fontsize = 18)
plt.title('Raw Data and Peak Brightness Quarter ' + str(quarter_num),
    ↪ fontsize = 18)


#plt.savefig('/Users/katelincrabtree/Desktop/
    ↪ Peak_and_SAPflux_Quarter_peak_quarter' + str(quarter_num) + '.png',
    ↪ orientation = 'portrait', bbox_inches = 'tight', pad_inches = 0, dpi =
    ↪  300)
plt.show()
#####
```

```
#print(rawpeak)


# In[6]:


print(number_eclipses)


# In[7]:


#####NORMALIZING FLUX AND ADDING MAGNITUDES TO FRAME

#making the eclipse_num array for the for loop
eclipse_num = np.arange(0, number_eclipses, 1) #(1, number_eclipses, 1)

#making an empty dataframe to add values that only deal with the eclipse
    ↪ with the normalized values
pframe = pd.DataFrame()

#for loop making the subplots for the primary eclipses to make a model for
    ↪ normalization
for eclipse1 in eclipse_num:

    #reset the axis values--cuts out unneccesary points
    x1 = first_Pepoch-(bcut*duration) +period*eclipse1 #start time of subset
    x2 = first_Pepoch+(ecut*duration) +period*eclipse1 #end time of subset
        ↪ of data

    #following variables plot flat points outside of the eclipse itself
    bx1 = first_Pepoch + period*eclipse1 - (bcut*duration) #section before
        ↪ eclipse points
    bx2 = first_Pepoch + period*eclipse1 - (0.5*duration)
    ax1 = first_Pepoch + period*eclipse1 + (0.5*duration) #section after
        ↪ eclipse points
    ax2 = first_Pepoch + period*eclipse1 + (ecut*duration)

    ie1 = first_Pepoch + period*eclipse1 - (0.5*duration)
    ie2 = first_Pepoch + period*eclipse1 + (0.5*duration)

    #building the frame for out of eclipse values and inner eclispe values
        ↪ PRIMARY
    #for renormalizing transits: & (lightcurve_frame.time <= bx2) added in
        ↪ wherever needed
    bpe_frame = lightcurve_frame[(lightcurve_frame.time >= bx1) & (
        ↪ lightcurve_frame.time <= bx2)] #before eclipse
    ape_frame = lightcurve_frame[(lightcurve_frame.time >= ax1) & (
        ↪ lightcurve_frame.time <= ax2)] #after eclipse
    ie_frame = lightcurve_frame [(lightcurve_frame.time >= ie1) & (
        ↪ lightcurve_frame.time <= ie2)] #in eclipse
    oope_frame = pd.concat([bpe_frame, ape_frame], axis=0) #builds total out
        ↪  of eclispe frame
    oope_frame.dropna(inplace = True, how = 'any')
    oope_frame = oope_frame.sort_values('time')

    if oope_frame.empty:
        continue
```

```
#checking values
#print(bx1)
#print(bx2)
#print(ax1)
#print(ax2)

#frame of just the transit portions of the eclipses
ptransit_frame = lightcurve_frame.loc[x1:x2]

#setting the arrangement of the subplots
plt.subplot(number_eclipses,1,eclipse1+1)  #(number of rows, number of
    ↪ columns, iteration)

#polyfit gives the coefficients of the eqn, 3rd or lower
flux = oope_frame['flux']
time = oope_frame['time']
pcoefficients = np.polyfit(time,flux, 2)
#print(pcoefficients)
ppolyfunc = np.poly1d(pcoefficients)
#print(ppolyfunc)

#makes a maodel for the data out of the eclipse
pmodel = ppolyfunc(ptransit_frame.time)
ptransit_frame['norm_model'] = pmodel
ptransit_frame['norm_flux'] = ((ptransit_frame.flux.subtract(
    ↪ ptransit_frame.norm_model))+ peak)/peak
ptransit_frame['mag']= -2.5*np.log10(ptransit_frame.norm_flux)
#print(ptransit_frame)
#print(ptransit_frame.mag)


#PRIMARY plots
plt.plot(ptransit_frame.time, ptransit_frame.flux, label = 'raw␣data')
plt.scatter(ptransit_frame.time, ptransit_frame.flux, s=1)
plt.scatter(oope_frame.time, oope_frame.flux, c='r', s=3, label = 'out␣
    ↪ of␣eclispe␣data')

###plotting the graphs and adding titles and labels
plt.plot(ptransit_frame.time, ptransit_frame.norm_model, c = 'green',
    ↪ label = 'model')
plt.scatter(ptransit_frame.time, ptransit_frame.norm_model, c= 'green',
    ↪ s = 2)
plt.title('Quarter␣' + str(quarter_num) + '␣Transit␣' + str(eclipse1))
plt.xlabel('Time␣(JD␣Days)')
plt.ylabel('Flux␣(-e/s)')
plt.legend()

#building pframe for furture use
pframe = pd.concat([pframe, ptransit_frame])

######finds the propegated error for the flux after normalization
pframe['norm_flux_error'] = (pframe.flux_err/peak)
pframe['mag_err'] = (2.5*pframe.norm_flux_error)/(np.log(10)*pframe.
    ↪ norm_flux)

#changing dimensions of each subplot and saving them
plt.tight_layout()
plt.subplots_adjust(top = number_eclipses, bottom = 0, right = 1, left =
```

```
        ↪  0, hspace = 0, wspace = 0)
    plt.margins(0,0)

    #plt.scatter(ptransit_frame.orbit_number, ptransit_frame.norm_flux)
    #plt.savefig(save_path + "Q"+ str(quarter_num)+ 'T' + str(eclipse1) + '
        ↪ outside eclipse normalizing.png', orientation = 'portrait',
        ↪ bbox_inches = 'tight', pad_inches = 0, dpi = 300)
    plt.savefig('/Users/katelincrabtree/Desktop/Q' + str(quarter_num) + 'T'
        ↪ +str(eclipse1)+'outside␣eclipse␣models.png', orientation = '
        ↪ portrait', bbox_inches = 'tight', pad_inches = 0, dpi = 300)
    plt.show()
    #print(pframe)

# print(eclipse_num)

#plt.savefig('/Users/katelincrabtree/Desktop/' + str(quarter_num)+'outside
    ↪ eclipse models.png')#, orientation = 'landscape', bbox_extra_artists =
    ↪  (lgd), bbox='tight')


# In[10]:


#####FLUX PLOTS
#plotting the normalized curve of the PRIMARY eclipse
#creating empty frames
model_frame = pd.DataFrame()
df = pd.DataFrame()



for eclipse1 in eclipse_num:
    #####sets timeframe for the eclipse and cuts out unneccesary areas
    x1 = first_Pepoch-(bcut*duration) +period*eclipse1
    x2 = first_Pepoch+(ecut*duration) +period*eclipse1

    model_transits = lightcurve_frame.loc[x1:x2]
    #####sets times for flat points before and after the actual eclipse
    bx1 = first_Pepoch + period*eclipse1 - (bcut*duration) #section before
        ↪ eclipse points
    bx2 = first_Pepoch + period*eclipse1 - (0.5*duration)
    ax1 = first_Pepoch + period*eclipse1 + (0.5*duration) #section after
        ↪ eclipse points
    ax2 = first_Pepoch + period*eclipse1 + (ecut*duration)
    ie1 = first_Pepoch + period*eclipse1 - (0.5*duration)
    ie2 = first_Pepoch + period*eclipse1 + (0.5*duration)

    #####building the frame for out of eclipse values and inner eclipse
        ↪ values PRIMARY
    bpe_frame = pframe[(pframe.time >= bx1) & (pframe.time <= bx2)] #before
        ↪ eclipse
    ape_frame = pframe[(pframe.time >= ax1) & (pframe.time <= ax2)] #after
        ↪ eclipse
    ipe_frame = pframe[(pframe.time >= ie1) & (pframe.time <= ie2)] #in
        ↪ eclipse
    oope_frame = pd.concat([bpe_frame, ape_frame], axis=0) #outside eclipse

    if oope_frame.empty:
        continue
```

```
def chi(data, error):
    return sum(((data-1)**2)/(error**2))

#####standard deviations of points in and out of transit
rawsd_intransit = chi(ipe_frame.norm_flux, ipe_frame.norm_flux_error) #
    ↪ standard deviation of points in transit
rawsd_outtransit = chi(oope_frame.norm_flux, oope_frame.norm_flux_error)
    ↪  #standard deviation of points out of transit

#rounding standard deviation so it doesnt look awful
sd_intransit = round(rawsd_intransit, 3)
sd_outtransit = round(rawsd_outtransit, 3)

reduced_chi_out = round(rawsd_outtransit/(len(oope_frame)), 3)

if reduced_chi_out >=5:
    print("Reduced␣Chi␣for␣Quarter␣" + str(quarter_num) + "␣Transit␣" +
        ↪ str(eclipse1) + "␣is␣more␣than␣5.")


#####plotting the idividual eclipses as fcn of normalized flux and time,
    ↪  as well as the comparative peak value in magenta
plt.subplot(number_eclipses, 1, eclipse1)
#plt.figure(figsize=(10,10))
plt.scatter(pframe.time[x1:x2], pframe.norm_flux[x1:x2], s=10, alpha=1,
    ↪ label = 'Normalized␣Flux␣Data')
plt.plot(pframe.time[x1:x2], pframe.norm_flux[x1:x2], alpha = 0.3)#flux
    ↪ and time
#plt.plot(pframe.time[x1:x2], pframe.norm_flux[x1:x2], c='g', alpha=
    ↪ 0.5)
#plt.errorbar(pframe.time[x1:x2], pframe.flux[x1:x2], yerr = pframe.
    ↪ norm_flux_error[x1:x2], ecolor = 'm')

#####plots peak flux value (optional)
#plt.plot([np.min(pframe.time[x1:x2]), np.max(pframe.time[x1:x2])], [
    ↪ peak, peak], c = 'm', label = 'Peak Flux Value (98th percentile
    ↪ avg) = ' + str(peak)) #peak line

#####adds standard deviations to legend by plotting nothing but with a
    ↪ label
#plt.plot([], [], label = "SDV in eclipse = " + str(sd_intransit), alpha
    ↪  = 0)
plt.plot([], [], label = "Chi␣squared␣outside␣eclipse␣=␣" + str(
    ↪ sd_outtransit), alpha = 0)
plt.plot([], [], label = "Reduced␣Chi␣outside␣eclipse␣=␣" + str(
    ↪ reduced_chi_out), alpha = 0)
plt.plot([], [], label = "peak␣flux␣value␣used␣=␣" + str(peak), alpha =
    ↪ 0)

#####adds various titles and labels and the legend location
plt.title('PDCSAP␣Normalized␣Flux␣Quarter␣' + str(quarter_num) + '␣
    ↪ Transit␣' + str(eclipse1))
plt.xlabel('Time␣(BJKD␣days)')
plt.ylabel('Normalized␣Flux')
#loc='center left', bbox_to_anchor=(1, 0.5))

#####I honestly don't know how this is working but the images look good
```

```
    ↪ and that's what I care about right now
plt.tight_layout()
plt.subplots_adjust(top = number_eclipses, bottom = 0, right = 1, left =
    ↪  0, hspace = 0, wspace = 0)
plt.margins(0,0)
plt.legend()
#plt.xlims(0.990, 1.005)
plt.ylim((pframe.norm_flux.min()-0.01), (pframe.norm_flux.max() + 0.01))


#changing the model dimensions
begin = pframe.time[x1:x2].min()
end = pframe.time[x1:x2].max()
length = (len(pframe[x1:x2]))

#adding time column for exporting files in non-BJKD time
pframe['time_(BJD)'] = pframe.time+4833.0 #7 decimal places also

#####Exporting the graph and the dataframe
#plt.savefig(save_path + '/flux_pngs/
    ↪ flux_pngsmaxpeak_PDCSAPKOI1786_lc_flux_Q' + str(quarter_num) + 'T'
    ↪  + str(eclipse1) + '.png', orientation = 'portrait', bbox_inches =
    ↪  'tight', pad_inches = 0, dpi = 300)
#pframe[x1:x2].to_csv(save_path + 'flux_txts/
    ↪ maxpeak_PDCSAPKOI1786_lc_flux_txts' + 'Q' + str(quarter_num) + 'T'
    ↪  + str(eclipse1) + '.txt', index = None, sep = '\t', mode = 'a',
    ↪ columns = ['time_(BJD)', 'norm_flux', 'norm_flux_error'], header =
    ↪  False)
#pframe[x1:x2].to_csv(save_path + 'KOI1786_lc_mag_txts' + 'Q' + str(
    ↪ quarter_num) + 'T' + str(eclipse1) + 'renorm.txt', index = None,
    ↪ sep = '\t', mode = 'a', columns = ['time_(BJD)', 'mag', 'mag_err
    ↪ '], header = False)

###this shows the center of each eclipse and prints it for use in STSP
    ↪ later
#print(params.t0)

plt.show()


# In[ ]:
```

4.3.2 Modeling Normalized Transits

```
#!/usr/bin/env python
# coding: utf-8

# In[8]:


#####Importing all necessary programs
import numpy as np
import pandas as pd
from pandas import DataFrame as df
from matplotlib import pyplot as plt
from lightkurve import search_lightcurvefile
import seaborn as sns
import scipy as sp
import statistics
```

```
import batman
import juliet
#removes warnings for polyfit if using a dataframe, currently a bug that
    ↪ needs fixing in general
pd.options.mode.chained_assignment = None

from IPython.core.display import display, HTML
display(HTML("<style>.container␣{␣width:95%␣!important;␣}</style>"))
display(HTML("<style>.rendered_html␣{␣font-size:␣18px;␣}</style>"))
#display


# In[9]:


#Give a file path for finding and saving graphs and txts
save_path = '/Users/katelincrabtree/Desktop/Final_KOI1786_Transits/'
#pd.read_csv(save_path + 'KOI1786_lc_flux_txts' + 'Q' + str(quarter_num) + '
    ↪ T' + str(eclipse1) + 'renorm.txt', index = None, sep = '\t', mode = 'a
    ↪ ', columns = ['time_(BJD)', 'norm_flux', 'norm_flux_error'], header =
    ↪ False


# In[10]:


#KOI-1786
#Villanova for koi-1786 and some additional parameters if necessary
IDnum = 'KIC␣3128793'
epoch = 5612.4254513878150#(54970.770924-50000) #villanova epoch
period = 24.678906375258098 #(24.6793813) #villanova period
duration = 15.168/24
limb_dark = "nonlinear"         #limb darkening model
limb_dark_u = [0.4065, 4.4820, -7.0634, 3.4164]
star_rot_period = 66.289

#Desired Transit Parameters
primary_epoch = epoch


# In[11]:


#renorms: Q4T4, Q5T1, Q5T4, Q8T1, Q13T2, Q15T3, Q16T3 for KOI-1786
###loops through the different transits for a given quarter
quarter = 9
PDCSAPtransit_array = [1, 2, 3, 4]


# In[13]:


for i in PDCSAPtransit_array:

#Reading In Data
    PDCSAPtransits = pd.read_csv(save_path + 'Final/flux_txts/
        ↪ maxpeak_PDCSAPKOI1786_lc_flux_txtsQ' + str(quarter) + 'T' + str(i)
        ↪  + '.0.txt', sep = '\t', header = 0, names = ['time', 'norm_flux',
        ↪  'norm_flux_err'])
```

```
#Adjusting columns as needed
    PDCSAPorbit_number = np.floor(PDCSAPtransits.time.sub(epoch).div(period)
        ↪ ) #truncates with astype.int()
    PDCSAPtransits['orbit_number'] = PDCSAPorbit_number
    PDCSAPphase = pd.DataFrame()
    PDCSAPphase['phase_raw'] = PDCSAPtransits.time.sub(epoch).div(period) -
        ↪ PDCSAPorbit_number
    PDCSAPphase.loc[PDCSAPphase.phase_raw>0.4, 'phase_raw'] -= 1.0
    PDCSAPtransits['phase'] = PDCSAPphase.phase_raw
    PDCSAPtransits = PDCSAPtransits.sort_values('time')

#uses input parameters to make a model from STSP
    newparams = batman.TransitParams()       #object to store transit
        ↪ parameters
    newparams.t0 = 5612.4254513878150                              #time of
        ↪ inferior conjunction
    newparams.per = 24.678906375258098                             #orbital period
    newparams.rp = np.sqrt(6.4916396428061109E-003 )                        #
        ↪ planet radius (in units of stellar radii)
    newparams.a = (1.6569523540787622E-002*newparams.per**2/0.01341561)
        ↪ **(1/3)#(density/(period^2))^1/3                          #semi-major
        ↪ axis (in units of stellar radii)
    newparams.inc = 85.537486826468268                          #orbital
        ↪ inclination (in degrees)
    newparams.ecc = 0.29997641038195499                           #eccentricity
    newparams.w = 102.17860441105046                             #longitude of
        ↪ periastron (in degrees)
    newparams.limb_dark = limb_dark        #limb darkening model
    newparams.u = limb_dark_u
    print(newparams.a)#limb darkening coefficients [u1, u2, u3, u4]
###generating BATMAN Model
    t = np.array(PDCSAPtransits.time[:]) #times at which to calculate light
        ↪ curve
    newm = batman.TransitModel(newparams, t)#initializes model
    newflux = newm.light_curve(newparams)

    #print(newparams.a)
#Making Model Dataframe
    PDCSAPmodel_transits = pd.DataFrame()
    PDCSAPmodel_transits['phase'] = PDCSAPtransits.phase
    PDCSAPmodel_transits['MCMC'] = newflux
    PDCSAPmodel_transits['time'] = t
    PDCSAPmodel_transits.set_index('time')

#adding model column to original transitsdataframe
    listofzeros = [0] * (len(PDCSAPtransits.time))
    PDCSAPtransits['MCMC'] = listofzeros
    PDCSAPtransits['MCMC'] += PDCSAPmodel_transits.MCMC
    #print(PDCSAPtransits.orbit_number[1])
#plotting the transits and the model
    plt.figure(figsize=(10,10))
    plt.errorbar(PDCSAPtransits.time, PDCSAPtransits.norm_flux, yerr=
        ↪ PDCSAPtransits.norm_flux_err, fmt = 'none', c='b', alpha = 0.75,
        ↪ lw = 0.7)
    plt.plot(PDCSAPtransits.time, PDCSAPtransits.MCMC, label = 'MCMC␣Model',
        ↪  c='r', lw = 2)
    plt.scatter(PDCSAPtransits.time, PDCSAPtransits.norm_flux, c='b', s=15,
        ↪ label = 'Normalized␣Flux␣Data')
```

```
    plt.ylim(0.989, 1.005)

#all of the labels and stuff
    plt.legend()
    plt.xlabel('Time (BJD-2450000)')
    plt.ylabel('Normalized Flux')
    plt.title('Max Peak PDCSAP Normalized Flux and Models (MCMC) for
        ↪ KOI1786Q'+ str(quarter) + 'T' + str(i))

#saving the transit and model to both txt and pngs
    #plt.savefig(save_path + 'Final/model_pngs/model_KOI1786Q'+ str(quarter)
        ↪  + 'T' + str(i) + 'norm_flux_model_MCMC.png')
    #PDCSAPtransits.to_csv(save_path + 'Final/model_txts/model_KOI1786Q'+
        ↪ str(quarter) + 'T' + str(i) + 'norm_flux_model_MCMC.txt', index =
        ↪ None, sep = '\t', mode = 'a', columns = ['time', 'norm_flux', '
        ↪ norm_flux_err', 'MCMC'], header = False)

    plt.show()


# In[16]:


print(9.0946 *(1-0.29998))
print(9.0946 *(1+0.29998))


# In[ ]:
```

4.3.3 Adjusting Transit Models

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


#####This program runs through and checks moseld generated by mcmc for their
    ↪  fit
###it also remves data points as needed in the transit that correspond to
    ↪ spot
###crossing features


# In[2]:


#####Importing all necessary programs
import numpy as np
import pandas as pd
from pandas import DataFrame as df
from matplotlib import pyplot as plt
from lightkurve import search_lightcurvefile
import seaborn as sns
import scipy as sp
import statistics
import batman
import juliet
#removes warnings for polyfit if using a dataframe, currently a bug that
```

```
    ↪ needs fixing in general
pd.options.mode.chained_assignment = None

from IPython.core.display import display, HTML
display(HTML("<style>.container␣{␣width:95%␣!important;␣}</style>"))
display(HTML("<style>.rendered_html␣{␣font-size:␣18px;␣}</style>"))
#display


# In[3]:


#Give a file path for finding and saving graphs and txts
save_path = '/Users/katelincrabtree/Desktop/Final_KOI1786_Transits/'
#pd.read_csv(save_path + 'KOI1786_lc_flux_txts' + 'Q' + str(quarter_num) + '
    ↪ T' + str(eclipse1) + 'renorm.txt', index = None, sep = '\t', mode = 'a
    ↪ ', columns = ['time_(BJD)', 'norm_flux', 'norm_flux_error'], header =
    ↪ False


# In[4]:


###KIC ID and limb darkening for red giants ~4500K
IDnum = 'KIC␣3128793'
limb_dark = "nonlinear"         #limb darkening model
limb_dark_u = [0.4065, 4.4820, -7.0634, 3.4164]


# In[6]:


####Importing MCMC files for the magnitudes, residuls, and the model
model_2 = pd.read_csv(save_path + 'MCMC/Round3/pltphot02.dat', sep = '\s+',
    ↪ header = 0, names = ['phase', 'mag', 'err', 'residual', 'time'])
model_data = pd.read_csv(save_path + 'MCMC/Round3/pltmod02.dat', sep = '\s+'
    ↪ , header = 0, names = ['phase', 'model', 'modrad'])
#print(model_2)


# In[7]:


###generate BATMAN model for overplotting and checking both work
#generating model--should be the same as before but in this case it
    ↪ overplots it all at once
params = batman.TransitParams()       #object to store transit parameters
params.t0 = 5612.4254513878150                          #time of inferior
    ↪ conjunction
params.per = 24.678906375258098                         #orbital period
params.rp = np.sqrt(6.4916396428061109E-003 )                       #planet
    ↪ radius (in units of stellar radii)
params.a = (1.6569523540787622E-002*params.per**2/0.01341561)**(1/3)#(
    ↪ density/(period^2))^1/3                          #semi-major axis (in
    ↪ units of stellar radii)
params.inc = 85.537486826468268                         #orbital inclination (in
    ↪  degrees)
params.ecc = 0.29997641038195499                          #eccentricity
params.w = 102.17860441105046                           #longitude of
```

```
    ↪ periastron (in degrees)
params.limb_dark = limb_dark        #limb darkening model
params.u = limb_dark_u      #limb darkening coefficients [u1, u2, u3, u4]

#generating Batman model
t = np.array(model_2.time[:]) #times at which to calculate light curve
m = batman.TransitModel(params, t)    #initializes model
flux = m.light_curve(params)

#making model DF
BATMAN_transits = pd.DataFrame()
BATMAN_transits['phase'] = model_2.phase
BATMAN_transits['batman'] = flux
BATMAN_transits['time'] = t
BATMAN_transits.set_index('time')
BATMAN_transits['BATMAN_mag']= -2.5*np.log10(BATMAN_transits.batman)
BATMAN_transits.sort_values('phase', inplace=True)

plt.scatter(BATMAN_transits.phase, BATMAN_transits.batman)
plt.show()

#print((1.9451134969878447E-002/params.per**2)**(1/3))


# In[8]:


plt.figure(figsize=(10,10), dpi=200)
plt.scatter(model_2.phase, model_2.mag, s=5)
plt.plot(model_data.phase, model_data.model, c='r', label = 'MCMC 3')
#plt.plot(BATMAN_transits.phase, BATMAN_transits.BATMAN_mag, c='g', label='
    ↪ BATMAN')
plt.xlim(0.96, 1.04)
plt.ylim(-0.010, model_2.mag.max()+0.002)
plt.title('MCMC fit and input data')
plt.legend()
plt.gca().invert_yaxis()
#plt.savefig('/Users/katelincrabtree/Desktop/
    ↪ Round3_MCMC_BATMAN_model_difference.png')
plt.show()


# In[9]:


###only cut in transit
model_2.index.names = ['index']
intransit_3_frame = pd.DataFrame()
outransit_3_frame = pd.DataFrame()

###Frame of only in transit
intransit_3_frame = model_2[(model_2.phase < 1.01151) & (model_2.phase >
    ↪ 0.988488)]
intransit_3_frame = intransit_3_frame[intransit_3_frame.residual > 0]

###Frame of out of transit
outransit_3_frame = model_2[(model_2.phase>1.01151) | (model_2.phase
    ↪ <0.988488)]
```

```
###Combined frames
transit_input_3 = intransit_3_frame.append([outransit_3_frame])
#print(model_1.phase.max())
#print(intransit_3_frame)

plt.figure(figsize=(10,10), dpi=200)
#plt.scatter(intransit_2_frame.phase, intransit_2_frame.mag, c='r', s=2,
    ↪ label = 'intransit')
plt.scatter(model_2.phase, model_2.mag, c='r', s=2, label = 'Round 3 data')
plt.scatter(transit_input_3.phase, transit_input_3.mag, c='b', s=2, label =
    ↪ 'Round 4 data')
plt.xlim(0.96, 1.04)
plt.ylim(-0.010, model_2.mag.max()+0.004)
plt.gca().invert_yaxis()
plt.title('Demonstration of removal of spots for MCMC fits--Round 3')
plt.legend()
#plt.savefig('/Users/katelincrabtree/Desktop/MCMC_round3_data_removal.png')
plt.show()
```

4.3.4 Spot Parameters Tests

```
#!/usr/bin/env python
# coding: utf-8

# In[2]:


#####Importing all necessary programs
import matplotlib.pyplot as plt
import matplotlib as mpl
import matplotlib.ticker as mticker
from astropy.io import fits
import glob
import numpy as np
import math
import csv
import os
import re
import cartopy.crs as ccrs
from cartopy.mpl.ticker import LongitudeFormatter, LatitudeFormatter
import time
from copy import deepcopy
import corner
from astropy.stats import sigma_clip
from matplotlib import colors
import batman
import juliet
import pandas as pd
#removes warnings for polyfit if using a dataframe, currently a bug that
    ↪ needs fixing in general
pd.options.mode.chained_assignment = None

import warnings
warnings.filterwarnings("ignore")


from IPython.core.display import display, HTML
display(HTML("<style>.container { width:95% !important; }</style>"))
display(HTML("<style>.rendered_html { font-size: 18px; }</style>"))
```

```
#display


# In[3]:


params = batman.TransitParams()
params.t0 = 5612.4254513878150                          #time of inferior
    ↪ conjunction
params.per = 24.678906375258098                         #orbital period
params.rp = np.sqrt(6.4916396428061109E-003 )                      #planet
    ↪ radius (in units of stellar radii)
params.a = (1.6569523540787622E-002*params.per**2/0.01341561)**(1/3)#(
    ↪ density/(period^2))^1/3                       #semi-major axis (in
    ↪ units of stellar radii)
params.inc = 85.537486826468268                      #orbital inclination (in
    ↪  degrees)
params.ecc = 0.29997641038195499                       #eccentricity
params.w = 102.17860441105046                         #longitude of
    ↪ periastron (in degrees)
params.limb_dark = "nonlinear"        #limb darkening model
params.u = [0.4065, 4.4820, -7.0634, 3.4164]    #limb darkening coefficients
    ↪  [u1, u2, u3, u4]
srot = 66.289 #stellar rotation
####note that this is one more than printed from my program
b = 0.49775
T0 = 5612.42545
p = 24.678906375258098 #period
srot = 66.289 #stellar rotation
rprstar = np.sqrt(6.4916396428061109E-003)


# In[1]:


##### For Testing STSP Spot Combinations #####

stsplat = 122 #115#120 ####always between 0 and 180
stsplon = 115#110#140
stsprad = 0.09

stsplat2 = 124 #120 ####always between 0 and 180
stsplon2 = 142 #140
stsprad2 = 0.10

stsplat3 = 116 #120 ####always between 0 and 180
stsplon3 = 129 #140
stsprad3 = 0.12

stsplat4 = 125 #120 ####always between 0 and 180
stsplon4 = 83 #140
stsprad4 = 0.08

stsplat5 = 125 #120 ####always between 0 and 180
stsplon5 = 100 #140
stsprad5 = 0.08

####DO NOT TOUCH
plt.figure(figsize=(10,10))
```

```
rad = stsprad * 6371
lat = 90 - stsplat
lon = stsplon

rad2 = stsprad2 * 6371
lat2 = 90 - stsplat2
lon2 = stsplon2

rad3 = stsprad3 * 6371
lat3 = 90 - stsplat3
lon3 = stsplon3

rad4 = stsprad4 * 6371
lat4 = 90 - stsplat4
lon4 = stsplon4

rad5 = stsprad5 * 6371
lat5 = 90 - stsplat5
lon5 = stsplon5

lon_0 = -1*((orbit_num*params.per/srot)-int((orbit_num*params.per/srot)))
    ↪ *360####longitude in front of me
center = -1*(np.arcsin(b))*180/np.pi
lower = -1*(np.arcsin((b+params.rp))*180/np.pi)
upper = -1*(np.arcsin((b-params.rp))*180/np.pi)
print(lon_0)



ax = plt.axes(projection=ccrs.Orthographic(central_longitude=lon_0,
    ↪ central_latitude=0))
ax.tissot(rad_km = rad, lons = lon, lats = lat, n_samples = 50, color = 'b',
    ↪  alpha = 1)

ax = plt.axes(projection=ccrs.Orthographic(central_longitude=lon_0,
    ↪ central_latitude=0))
ax.tissot(rad_km = rad2, lons = lon2, lats = lat2, n_samples = 50, color = '
    ↪ g', alpha = 1)

ax = plt.axes(projection=ccrs.Orthographic(central_longitude=lon_0,
    ↪ central_latitude=0))
ax.tissot(rad_km = rad3, lons = lon3, lats = lat3, n_samples = 50, color = '
    ↪ r', alpha = 1)

ax = plt.axes(projection=ccrs.Orthographic(central_longitude=lon_0,
    ↪ central_latitude=0))
ax.tissot(rad_km = rad4, lons = lon4, lats = lat4, n_samples = 50, color = '
    ↪ orange', alpha = 1)

ax = plt.axes(projection=ccrs.Orthographic(central_longitude=lon_0,
    ↪ central_latitude=0))
ax.tissot(rad_km = rad5, lons = lon5, lats = lat5, n_samples = 50, color = '
    ↪ m', alpha = 1)


gly = ax.gridlines(crs=ccrs.PlateCarree(),
                   linewidth=1, color='gray', alpha=0.5, linestyle='-')
```

```
gly.xformatter = LongitudeFormatter()
gly.xlocator = mticker.FixedLocator
    ↪ ([-340,-320,-300,-280,-260,-240,-220,-200,-180,-160,-140,-120,-100,-80,-60,-40,-20,
    ↪
                                            20,40,60,80,100,120,140,160,200,220,240,260,280,
                                                ↪

gly = ax.gridlines(crs=ccrs.PlateCarree(),
                   linewidth=1, color='r', alpha=1, linestyle='-')
gly.xlocator = mticker.FixedLocator([0])


###planet line (center)
glx = ax.gridlines(crs=ccrs.PlateCarree(),
                   linewidth=2, color='blue', alpha=0.7, linestyle='-')
glx.xlines = False
glx.ylines = True
glx.yformatter = LatitudeFormatter()
glx.ylocator = mticker.FixedLocator([center])

###planet lines (upper, lower)
glx = ax.gridlines(crs=ccrs.PlateCarree(),
                   linewidth=2, color='blue', alpha=0.7, linestyle='-.')
glx.xlines = False
glx.ylines = True
glx.yformatter = LatitudeFormatter()
glx.ylocator = mticker.FixedLocator([upper,lower])


plt.show()
print('Latitude␣of␣Spot␣1␣is:' + str((stsplat*np.pi)/180))
print('Longitude␣of␣Spot␣1␣is:' + str((stsplon*np.pi)/180))
print('')
print('Latitude␣of␣Spot␣2␣is:' + str((stsplat2*np.pi)/180))
print('Longitude␣of␣Spot␣2␣is:' + str((stsplon2*np.pi)/180))
print('')
print('Latitude␣of␣Spot␣3␣is:' + str((stsplat3*np.pi)/180))
print('Longitude␣of␣Spot␣3␣is:' + str((stsplon3*np.pi)/180))
print('')
print('Latitude␣of␣Spot␣4␣is:' + str((stsplat4*np.pi)/180))
print('Longitude␣of␣Spot␣4␣is:' + str((stsplon4*np.pi)/180))
print('')
print('Latitude␣of␣Spot␣5␣is:' + str((stsplat5*np.pi)/180))
print('Longitude␣of␣Spot␣5␣is:' + str((stsplon5*np.pi)/180))



print('')
print('')
print('')


# In[ ]:


#####ACTION L OUTPUT GRAPHING#####
```

```
sc_array = np.asarray(['0'])#,'05', '1', '2', '3', '4', '5'])

for c in sc_array:
    save_path = '/Users/katelincrabtree/Desktop/al/Q8T1/T1/al_Q8T1_c'
    #nospot = pd.read_csv(save_path + '0' + c + '_lcout.txt', sep = ' ',
        ↪ header = 0, names = ['time', 'norm_flux', 'norm_flux_err', 'model
        ↪ ', 'spot_bool'] )
    print('contrast is 0.' + str(c))
    #paramfile = np.genfromtxt(save_path + '0' + c + '_lcout.txt', comments
        ↪ ='#', delimiter='\t', usecols=0, dtype='f8')
    lcfile = pd.read_csv(save_path + '0' + c + '_lcout.txt', sep = ' ',
        ↪ header = 0, names = ['time', 'norm_flux', 'norm_flux_err', 'model'
        ↪ , 'spot_bool'] )
    print('contrast is 0.' + str(c))
    plt.figure(figsize=(10,10))
    plt.plot(lcfile.time, lcfile.model,'r-', label='STSP fit')
    plt.errorbar(lcfile.time, lcfile.norm_flux, xerr=None, yerr=lcfile.
        ↪ norm_flux_err, fmt='k.', ecolor='gray', label='Data')
    #print(lcfile)

    #generating Batman model
    t = np.array(lcfile.time[:]) #times at which to calculate light curve
    m = batman.TransitModel(params, t)    #initializes model
    flux = m.light_curve(params)

    #making model DF
    BATMAN_transits = pd.DataFrame()
    BATMAN_transits['batman'] = flux
    BATMAN_transits['time'] = t
    BATMAN_transits.set_index('time')
    plt.plot(BATMAN_transits.time, BATMAN_transits.batman, linewidth = 1)
    plt.show()

    ##### For Testing STSP Spot Combinations #####

    ####DO NOT TOUCH
    plt.figure(figsize=(10, 10))


    rad = stsprad * 6371
    lat = 90 - stsplat
    lon = stsplon

    rad2 = stsprad2 * 6371
    lat2 = 90 - stsplat2
    lon2 = stsplon2

    rad3 = stsprad3 * 6371
    lat3 = 90 - stsplat3
    lon3 = stsplon3

    rad4 = stsprad4 * 6371
    lat4 = 90 - stsplat4
    lon4 = stsplon4

    rad5 = stsprad5 * 6371
    lat5 = 90 - stsplat5
    lon5 = stsplon5
```

```
lon_0 = -1*((orbit_num*params.per/srot)-int((orbit_num*params.per/srot))
    ↪ )*360####longitude in front of me
center = -1*(np.arcsin(b))*180/np.pi
lower = -1*(np.arcsin((b+params.rp))*180/np.pi)
upper = -1*(np.arcsin((b-params.rp))*180/np.pi)
print(center)

ax = plt.axes(projection=ccrs.Orthographic(central_longitude=lon_0,
    ↪ central_latitude=0))
ax.tissot(rad_km = rad, lons = lon, lats = lat, n_samples = 50, color =
    ↪ 'b', alpha = 1)

ax = plt.axes(projection=ccrs.Orthographic(central_longitude=lon_0,
    ↪ central_latitude=0))
ax.tissot(rad_km = rad2, lons = lon2, lats = lat2, n_samples = 50, color
    ↪  = 'g', alpha = 1)

ax = plt.axes(projection=ccrs.Orthographic(central_longitude=lon_0,
    ↪ central_latitude=0))
ax.tissot(rad_km = rad3, lons = lon3, lats = lat3, n_samples = 50, color
    ↪  = 'r', alpha = 1)

ax = plt.axes(projection=ccrs.Orthographic(central_longitude=lon_0,
    ↪ central_latitude=0))
ax.tissot(rad_km = rad4, lons = lon4, lats = lat4, n_samples = 50, color
    ↪  = 'orange', alpha = 1)

ax = plt.axes(projection=ccrs.Orthographic(central_longitude=lon_0,
    ↪ central_latitude=0))
ax.tissot(rad_km = rad5, lons = lon5, lats = lat5, n_samples = 50, color
    ↪  = 'm', alpha = 1)


gly = ax.gridlines(crs=ccrs.PlateCarree(),
              linewidth=1, color='gray', alpha=0.5, linestyle='-')


gly.xformatter = LongitudeFormatter()
gly.xlocator = mticker.FixedLocator
    ↪ ([-340,-320,-300,-280,-260,-240,-220,-200,-180,-160,-140,-120,-100,-80,-60,-40,
    ↪
                                        20,40,60,80,100,120,140,160,200,220,240,260,280,
                                            ↪


gly = ax.gridlines(crs=ccrs.PlateCarree(),
              linewidth=1, color='r', alpha=1, linestyle='-')
gly.xlocator = mticker.FixedLocator([0])


###planet line (center)
glx = ax.gridlines(crs=ccrs.PlateCarree(),
              linewidth=2, color='blue', alpha=0.7, linestyle='-')
glx.xlines = False
glx.ylines = True
glx.yformatter = LatitudeFormatter()
glx.ylocator = mticker.FixedLocator([center])

###planet lines (upper, lower)
glx = ax.gridlines(crs=ccrs.PlateCarree(),
```

```
                linewidth=2, color='blue', alpha=0.7, linestyle='-.')
    glx.xlines = False
    glx.ylines = True
    glx.yformatter = LatitudeFormatter()
    glx.ylocator = mticker.FixedLocator([upper,lower])


    plt.show()
    print('Latitude of Spot 1 is:' + str((stsplat*np.pi)/180))
    print('Longitude of Spot 1 is:' + str((stsplon*np.pi)/180))
    print('')
    print('Latitude of Spot 2 is:' + str((stsplat2*np.pi)/180))
    print('Longitude of Spot 2 is:' + str((stsplon2*np.pi)/180))
    print('')
    print('Latitude of Spot 3 is:' + str((stsplat3*np.pi)/180))
    print('Longitude of Spot 3 is:' + str((stsplon3*np.pi)/180))
    print('')
    print('Latitude of Spot 4 is:' + str((stsplat4*np.pi)/180))
    print('Longitude of Spot 4 is:' + str((stsplon4*np.pi)/180))
    print('')
    print('Latitude of Spot 5 is:' + str((stsplat5*np.pi)/180))
    print('Longitude of Spot 5 is:' + str((stsplon5*np.pi)/180))



    print('')
    print('')
    print('')
```

4.3.5 Spot Plotting with Model and Residuals

```
#!/usr/bin/env python
# coding: utf-8

# In[2]:


#####Importing all necessary programs
import matplotlib.pyplot as plt
import matplotlib as mpl
import matplotlib.ticker as mticker
from astropy.io import fits
import glob
import numpy as np
import math
import csv
import os
import re
import cartopy.crs as ccrs
from cartopy.mpl.ticker import LongitudeFormatter, LatitudeFormatter
import time
from copy import deepcopy
import corner
from astropy.stats import sigma_clip
from matplotlib import colors
import batman
import juliet
import pandas as pd
#removes warnings for polyfit if using a dataframe, currently a bug that
```

```
    ↪ needs fixing in general
pd.options.mode.chained_assignment = None

import warnings
warnings.filterwarnings("ignore")


from IPython.core.display import display, HTML
display(HTML("<style>.container␣{␣width:95%␣!important;␣}</style>"))
display(HTML("<style>.rendered_html␣{␣font-size:␣18px;␣}</style>"))
#display


# In[3]:


params = batman.TransitParams()
params.t0 = 5612.4254513878150                          #time of inferior
    ↪ conjunction
params.per = 24.678906375258098                         #orbital period
params.rp = np.sqrt(6.4916396428061109E-003 )                      #planet
    ↪ radius (in units of stellar radii)
params.a = (1.6569523540787622E-002*params.per**2/0.01341561)**(1/3)#(
    ↪ density/(period^2))^1/3                       #semi-major axis (in
    ↪ units of stellar radii)
params.inc = 85.537486826468268                        #orbital inclination (in
    ↪  degrees)
params.ecc = 0.29997641038195499                        #eccentricity
params.w = 102.17860441105046                          #longitude of
    ↪ periastron (in degrees)
params.limb_dark = "nonlinear"        #limb darkening model
params.u = [0.4065, 4.4820, -7.0634, 3.4164]    #limb darkening coefficients
    ↪  [u1, u2, u3, u4]
srot = 66.289 #stellar rotation
####note that this is one more than printed from my program
b = 0.49775
T0 = 5612.42545
p = 24.678906375258098 #period
srot = 66.289 #stellar rotation
rprstar = np.sqrt(6.4916396428061109E-003)


# In[8]:


############ACTION S OUTPUT GRAPHING######
sc_array = np.asarray(['0','05', '1', '2', '3', '4', '5'])


for c in sc_array:
    orbit_num = 26 ####note that this is one more than printed from my
        ↪ program

    ####Reading in the different necessary files
    #save_path = '/Users/katelincrabtree/Desktop/aS3/Q9T3/T5/aS2_Q9T3_c0'
    save_path = '/Users/katelincrabtree/Desktop/aS3/aS2_Q8T3_c0'
    paramfile = np.genfromtxt(save_path  + c + '_parambest.txt', comments='#
        ↪ ', delimiter='\t', dtype='f8')
    lcfile = pd.read_csv(save_path  + c + '_lcbest.txt', sep = '␣', header =
```

```
    ↪  0, names = ['time', 'norm_flux', 'norm_flux_err', 'model', '
    ↪ spot_bool'] )
#nospot = pd.read_csv('/Users/katelincrabtree/Desktop/aS3/
    ↪ nospot_Q9T3_lcout.txt', sep = ' ', header = 0, names = ['time', '
    ↪ norm_flux', 'norm_flux_err', 'model', 'spot_bool'] )
print('contrast is 0.' + str(c))
print(len(lcfile))
#print(paramfile)
###grabbing the best parameters from STSP parambest file
stsprad = paramfile[0]
stsplat = paramfile[1]*(180/np.pi) ####always between 0 and 180
stsplon = paramfile[2]*(180/np.pi)

stsprad2 = paramfile[3]
stsplat2 = paramfile[4]*(180/np.pi) ####always between 0 and 180
stsplon2 = paramfile[5]*(180/np.pi)

stsprad3 = paramfile[6]
stsplat3 = paramfile[7]*(180/np.pi) ####always between 0 and 180
stsplon3 = paramfile[8]*(180/np.pi)

stsprad4 = paramfile[9]
stsplat4 = paramfile[10]*(180/np.pi) ####always between 0 and 180
stsplon4 = paramfile[11]*(180/np.pi)

stsprad5 = paramfile[12]
stsplat5 = paramfile[13]*(180/np.pi) ####always between 0 and 180
stsplon5 = paramfile[14]*(180/np.pi)

####DO NOT TOUCH
plt.figure(figsize=(10,10))

rad = stsprad * 6371
lat = 90 - stsplat
lon = stsplon

rad2 = stsprad2 * 6371
lat2 = 90 - stsplat2
lon2 = stsplon2

rad3 = stsprad3 * 6371
lat3 = 90 - stsplat3
lon3 = stsplon3

rad4 = stsprad4 * 6371
lat4 = 90 - stsplat4
lon4 = stsplon4

rad5 = stsprad5 * 6371
lat5 = 90 - stsplat5
lon5 = stsplon5

lon_0 = -1*((orbit_num*p/srot)-int((orbit_num*p/srot)))*360####longitude
    ↪  in front of me
center = -1*(np.arcsin(b))*180/np.pi
lower = -1*(np.arcsin((b+rprstar))*180/np.pi)
upper = -1*(np.arcsin((b-rprstar))*180/np.pi)
#print(center)
```

```
#generating Batman model
t = np.linspace(lcfile.time.min(), lcfile.time.max(), 1000) #np.array(
    ↪ lcfile.time[:]) lcfile.time.max(), 1000) #times at which to
    ↪ calculate light curve
m = batman.TransitModel(params, t)    #initializes model
flux = m.light_curve(params)

#making model DF
BATMAN_transits = pd.DataFrame()
BATMAN_transits['batman'] = flux
BATMAN_transits['time'] = t
BATMAN_transits.set_index('time')
#BATMAN_transits['BATMAN_mag']= -2.5*np.log10(BATMAN_transits.batman)

####Beginning spot plotting
ax = plt.axes(projection=ccrs.Orthographic(central_longitude=lon_0,
    ↪ central_latitude=0))
ax.tissot(rad_km = rad, lons = lon, lats = lat, n_samples = 50, color =
    ↪ 'b', alpha = 1)

ax = plt.axes(projection=ccrs.Orthographic(central_longitude=lon_0,
    ↪ central_latitude=0))
ax.tissot(rad_km = rad2, lons = lon2, lats = lat2, n_samples = 50, color
    ↪  = 'g', alpha = 1)

ax = plt.axes(projection=ccrs.Orthographic(central_longitude=lon_0,
    ↪ central_latitude=0))
ax.tissot(rad_km = rad3, lons = lon3, lats = lat3, n_samples = 50, color
    ↪  = 'r', alpha = 1)

ax = plt.axes(projection=ccrs.Orthographic(central_longitude=lon_0,
    ↪ central_latitude=0))
ax.tissot(rad_km = rad4, lons = lon4, lats = lat4, n_samples = 50, color
    ↪  = 'orange', alpha = 1)

ax = plt.axes(projection=ccrs.Orthographic(central_longitude=lon_0,
    ↪ central_latitude=0))
ax.tissot(rad_km = rad5, lons = lon5, lats = lat5, n_samples = 50, color
    ↪  = 'm', alpha = 1)


gly = ax.gridlines(crs=ccrs.PlateCarree(),
                   linewidth=1, color='gray', alpha=0.5, linestyle='-')


gly.xformatter = LongitudeFormatter()
gly.xlocator = mticker.FixedLocator
    ↪ ([-340,-320,-300,-280,-260,-240,-220,-200,-180,-160,-140,-120,-100,-80,-60,-40,
    ↪
                                        20,40,60,80,100,120,140,160,200,220,240,260,280,
                                            ↪


gly = ax.gridlines(crs=ccrs.PlateCarree(),
                   linewidth=1, color='r', alpha=1, linestyle='-')
gly.xlocator = mticker.FixedLocator([0])


#planet line (center)
glx = ax.gridlines(crs=ccrs.PlateCarree(),
```

```
                linewidth=2, color='blue', alpha=0.7, linestyle='-')
glx.xlines = False
glx.ylines = True
glx.yformatter = LatitudeFormatter()
glx.ylocator = mticker.FixedLocator([center])

###planet lines (upper, lower)
glx = ax.gridlines(crs=ccrs.PlateCarree(),
                linewidth=2, color='blue', alpha=0.7, linestyle='-.')
glx.xlines = False
glx.ylines = True
glx.yformatter = LatitudeFormatter()
glx.ylocator = mticker.FixedLocator([upper,lower])
plt.savefig(save_path + str(c) + 'spot_circle_plot.png')
plt.show()

#####Light Curve
fig1 = plt.figure(1, figsize = (10, 12))
frame1 = fig1.add_axes((.1,.3,.8,.6))
plt.plot(lcfile.time, lcfile.model,'r-', label='STSP fit')
plt.errorbar(lcfile.time, lcfile.norm_flux, xerr=None, yerr=lcfile.
    ↪ norm_flux_err, fmt='k.', ecolor='gray', label='Data')
plt.plot(BATMAN_transits.time, BATMAN_transits.batman, linewidth = 1,
    ↪ label = 'BATMAN Model', c='b')
#plt.plot(nospot.time, nospot.model, label = 'No spot STSP', c='g',
    ↪ linewidth = 1)
#plt.text(lcfile.time.min(), lcfile.norm_flux.min(), 'Final Chi is ' +
    ↪ chi)
plt.legend()
#plt.xlim(5710.5, 5711.5)
frame1.set(title='STSP Light Curve Fit - Contrast = 0.' + str(c), ylabel
    ↪ =' Relative Flux')
frame1.set_xticklabels([])

diff = lcfile.model - lcfile.norm_flux
frame2 = fig1.add_axes((.1,.1,.8,.2))
plt.errorbar(lcfile.time, diff, yerr=lcfile.norm_flux_err, fmt='k.',
    ↪ ecolor='gray')

frame2.axhline(ls='-', lw=2, color='k')
plt.xlabel('Time (BKJD days)')
plt.ylabel('Residuals')
fig1 = plt.gcf()
plt.savefig(save_path + str(c) + 'stsp_model_and_residuals.png')
plt.show()

#####Prints the central longitude since this program htes plotting lines
    ↪  I guess
print('Central longitude (line in front) is ' + str(lon_0))
print('')

#####Prints Latitude and Longitude for Each Spot
print('Latitude of Spot 1 is:' + str((stsplat*np.pi)/180))
print('Longitude of Spot 1 is:' + str((stsplon*np.pi)/180))
print('')
print('Latitude of Spot 2 is:' + str((stsplat2*np.pi)/180))
print('Longitude of Spot 2 is:' + str((stsplon2*np.pi)/180))
print('')
print('Latitude of Spot 3 is:' + str((stsplat3*np.pi)/180))
```

```
    print('Longitude of Spot 3 is:' + str((stsplon3*np.pi)/180))
    print('')
#     print('Latitude of Spot 4 is:' + str((stsplat4*np.pi)/180))
#     print('Longitude of Spot 4 is:' + str((stsplon4*np.pi)/180))
#     print('')
#     print('Latitude of Spot 5 is:' + str((stsplat5*np.pi)/180))
#     print('Longitude of Spot 5 is:' + str((stsplon5*np.pi)/180))
#     print('')
#     print('')
#     print('')


# In[7]:


# ###generating false frame type thing for use in STSP
# save_path = '/Users/katelincrabtree/Desktop/aS3/aS2_Q8T3_c0'
# lcfile = pd.read_csv(save_path + 'c00_lcbest.txt', sep = ' ', header = 0,
    ↪ names = ['time', 'norm_flux', 'norm_flux_err', 'model', 'spot_bool'] )


# t = np.linspace(lcfile.time.min(), lcfile.time.max(), 1000) #np.array(
    ↪ lcfile.time[:]) lcfile.time.max(), 1000) #times at which to calculate
    ↪ light curve
# #m = batman.TransitModel(params, t)     #initializes model
# #flux = m.light_curve(params)

# spotless = pd.DataFrame()
# spotless['dummy_flux'] = 100
# spotless['time'] = t
# spotless['dummy_err'] = spotless.dummy_flux*0.001

# spotless.to_csv('/Users/katelincrabtree/Desktop/aS3/spotless_Q9T3.txt',
    ↪ index = None, sep = '\t', mode = 'a', columns = ['time', 'dummy_flux',
    ↪  'dummy_err'], header = False)
```

4.3.6 MCMC Parameter Plotting

```
#!/usr/bin/env python
# coding: utf-8

# In[8]:


#####Importing all necessary programs
import matplotlib.pyplot as plt
import matplotlib as mpl
import matplotlib.ticker as mticker
from astropy.io import fits
import glob
import scipy
import numpy as np
import math
import csv
import os
import re
import statistics as stats
import cartopy.crs as ccrs
from cartopy.mpl.ticker import LongitudeFormatter, LatitudeFormatter
```

```
import time
from copy import deepcopy
import corner
from astropy.stats import sigma_clip
from matplotlib import colors
import batman
import juliet
import pandas as pd
from mpl_toolkits.axes_grid1 import make_axes_locatable
#removes warnings for polyfit if using a dataframe, currently a bug that
    ↪ needs fixing in general
pd.options.mode.chained_assignment = None

import warnings
warnings.filterwarnings("ignore")


from IPython.core.display import display, HTML
display(HTML("<style>.container␣{␣width:95%␣!important;␣}</style>"))
display(HTML("<style>.rendered_html␣{␣font-size:␣18px;␣}</style>"))


####Give some aspects of the system the MCMC is for
##contrast used
con = '00'
###number of spots
num_spots = 3

###save paths and files
#save_path = '/Users/katelincrabtree/Desktop/aS3/Q9T3/T5/aS2_Q9T3_c'
save_path = '/Users/katelincrabtree/Desktop/aS3/aS2_Q8T3_c'
#mcmc_file = pd.read_csv(save_path + con + '_mcmc.txt', sep = ' ', header =
    ↪ 0, names = ['x', 'y', 'z', 'chi', 'rad1', 'lat1', 'lon1', 'rad2', '
    ↪ lat2', 'lon2', 'rad3', 'lat3', 'lon3', 'bright'])
mcmc_file = pd.read_csv(save_path + '00'+ '_mcmc.txt', sep = '␣', header =
    ↪ 0, names = ['x', 'y', 'z', 'chi', 'rad1', 'lat1', 'lon1', 'rad2', '
    ↪ lat2', 'lon2', 'rad3', 'lat3', 'lon3', 'rad4', 'lat4', 'lon4','rad5',
    ↪ 'lat5', 'lon5','bright'] )


# In[9]:


###creates cutoff for chisquared values within 1 sigma

nw2 = num_spots*3
Ci68=(2*scipy.special.gammaincinv(nw2/2, 0.683))#/nw2
print(Ci68)


# In[10]:


####imports MCMC file and cuts based on chisquared values needed

mcmc_2 = pd.DataFrame()
mcmc_2 = mcmc_file[mcmc_file['chi'] < (mcmc_file.chi.min()+Ci68)]
mcmc_2['lat1'] = mcmc_2.lat1*(180/np.pi)
mcmc_2['lat2'] = mcmc_2.lat2*(180/np.pi)
```

```
mcmc_2['lat3'] = mcmc_2.lat3*(180/np.pi)
mcmc_2['lat4'] = mcmc_2.lat4*(180/np.pi)
mcmc_2['lat5'] = mcmc_2.lat5*(180/np.pi)


mcmc_2['lon1'] = mcmc_2.lon1*(180/np.pi)
mcmc_2['lon2'] = mcmc_2.lon2*(180/np.pi)
mcmc_2['lon3'] = mcmc_2.lon3*(180/np.pi)
mcmc_2['lon4'] = mcmc_2.lon4*(180/np.pi)
mcmc_2['lon5'] = mcmc_2.lon5*(180/np.pi)

mcmc_2 = mcmc_2.sort_values('chi', ascending = False)


# In[13]:


####creating the plot based on lat, lon, spot size, and chisquared value
plt.figure(figsize=(15,15))
#spot 1
plt.scatter(mcmc_2.lon1, mcmc_2.lat1, c =mcmc_2.chi, cmap = 'Blues_r', s =
    ↪ mcmc_2.rad1*1000, edgecolors='k')
#spot 2
plt.scatter(mcmc_2.lon2, mcmc_2.lat2, c =mcmc_2.chi, cmap = 'Greens_r', s =
    ↪ mcmc_2.rad2*1000, edgecolors='k')
#spot 3
plt.scatter(mcmc_2.lon3, mcmc_2.lat3, c =mcmc_2.chi, cmap = 'Reds_r', s =
    ↪ mcmc_2.rad3*1000, edgecolors='k')

plt.scatter(mcmc_2.lon4, mcmc_2.lat4, c =mcmc_2.chi, cmap = 'YlOrBr_r', s =
    ↪ mcmc_2.rad4*1000, edgecolors='k')

plt.scatter(mcmc_2.lon5, mcmc_2.lat5, c =mcmc_2.chi, cmap = 'RdPu_r', s =
    ↪ mcmc_2.rad5*1000, edgecolors='k')

#plt.gca().invert_yaxis()
plt.ylabel('Latitude')
plt.xlabel('Longitude')
plt.title('Drift␣in␣spot␣placement␣and␣radius␣with␣respect␣to␣chi␣squared␣
    ↪ value␣for␣contrast␣' + con + '␣(min␣=␣'+ str(mcmc_2.chi.min())+ ')')
#plt.text(80, 140, 'chi square minimum is ' + str(mcmc_2.chi.min()),
    ↪ fontsize = 18)
#plt.legend()
#plt.savefig(save_path + 'mcmc_graph_chi_' + con + '.png', dpi=300)
plt.show()


# In[12]:


####creating histograms of various spot parameters

#####sizes of radii for best fit files per spot
plt.figure(figsize=(7,7))
plt.hist(mcmc_2.rad1, bins = 50, lw = 1, ec = 'k', color = 'b', label = '
    ↪ Spot␣1')
plt.ylabel('Number␣of␣incidents')
plt.xlabel('Radius')
plt.legend()
```

```
plt.title('Spot␣Radius␣Ranges␣for␣1␣sigma␣variation␣for␣contrast␣' + con)
#plt.savefig(save_path + con + '_mcmc_spot_1_rad_hist.png')
plt.show()

#spot 2
plt.figure(figsize=(7,7))
plt.hist(mcmc_2.rad2, bins = 50, lw = 1, ec = 'k', color = 'g', label = '
    ↪ Spot␣2')
plt.ylabel('Number␣of␣incidents')
plt.xlabel('Radius')
plt.legend()
plt.title('Spot␣Radius␣Ranges␣for␣1␣sigma␣variation␣for␣contrast␣' + con)
#plt.savefig(save_path + con + '_mcmc_spot_2_rad_hist.png')
plt.show()

#spot 3
plt.figure(figsize=(7,7))
plt.hist(mcmc_2.rad3, bins = 50, lw = 1, ec = 'k', color = 'r', label = '
    ↪ Spot␣3')
plt.ylabel('Number␣of␣incidents')
plt.xlabel('Radius')
plt.legend()
plt.title('Spot␣Radius␣Ranges␣for␣1␣sigma␣variation␣for␣contrast␣' + con)
#plt.savefig(save_path + con + '_mcmc_spot_3_rad_hist.png')
plt.show()

# #spot 4
# axs[3].hist(mcmc_2.rad4, bins = 50, lw = 1, ec = 'k', color = 'orange',
    ↪ label = 'Spot 4')
# axs[3].set_ylabel('Number of incidents')
# axs[3].set_xlabel('Radius Size range')
# axs[3].legend()

# #spot 5
# axs[4].hist(mcmc_2.rad5, bins = 50, lw = 1, ec = 'k', color = 'm', label =
    ↪  'Spot 5')
# axs[4].set_ylabel('Number of incidents')
# axs[4].set_xlabel('Radius Size range')
# axs[4].legend()
#plt.savefig
plt.show()


# In[86]:


####creating histograms of various spot parameters

#####latitude ranges for best fit files per spot

#fig, axs = plt.subplots(num_spots, 1, figsize=(num_spots*4, num_spots*7))#,
    ↪  sharey=True)
##spot 1
plt.figure(figsize=(7,7))
plt.hist(mcmc_2.lat1, bins = 50, lw = 1, ec = 'k', color = 'b', label = '
    ↪ Spot␣1')
plt.ylabel('Number␣of␣incidents')
plt.xlabel('Latitude')
plt.legend()
```

```
plt.title('Spot Latitude Ranges for 1 sigma variation for contrast ' + con)
#plt.savefig(save_path + con + '_mcmc_spot_1_lat_hist.png')
plt.show()


#spot 2
plt.figure(figsize=(7,7))
plt.hist(mcmc_2.lat2, bins = 50, lw = 1, ec = 'k', color = 'g', label = '
    ↪ Spot 2')
plt.ylabel('Number of incidents')
plt.xlabel('Latitude')
plt.title('Spot Latitude Ranges for 1 sigma variation for contrast ' + con)
plt.legend()
#plt.savefig(save_path + con + '_mcmc_spot_2_lat_hist.png')
plt.show()


#spot 3
plt.figure(figsize=(7,7))
plt.hist(mcmc_2.lat3, bins = 50, lw = 1, ec = 'k', color = 'r', label = '
    ↪ Spot 3')
plt.ylabel('Number of incidents')
plt.xlabel('Latitude')
plt.title('Spot Latitude Ranges for 1 sigma variation for contrast ' + con)
plt.legend()
#plt.savefig(save_path + con + '_mcmc_spot_3_lat_hist.png')
plt.show()


# # #spot 4
# axs[3].hist(mcmc_2.lat4, bins = 50, lw = 1, ec = 'k', color = 'orange',
    ↪ label = 'Spot 4')
# axs[3].set_ylabel('Number of incidents')
# axs[3].set_xlabel('Radius Size range')
# axs[3].legend()

# #spot 5
# axs[4].hist(mcmc_2.lat5, bins = 50, lw = 1, ec = 'k', color = 'm', label =
    ↪  'Spot 5')
# axs[4].set_ylabel('Number of incidents')
# axs[4].set_xlabel('Radius Size range')
# axs[4].legend()
#plt.savefig(save_path + con + '_mcmc_spot_1_lat.png')
plt.show()


# In[87]:


####creating histograms of various spot parameters

#####longitude ranges for best fit files per spot

##spot 1
plt.figure(figsize=(7,7))
plt.hist(mcmc_2.lon1, bins = 50, lw = 1, ec = 'k', color = 'b', label = '
    ↪ Spot 1')
plt.ylabel('Number of incidents')
plt.xlabel('Longitude')
```

```
plt.legend()
plt.title('Spot Longitude Ranges for 1 sigma variation for contrast ' + con)
#plt.savefig(save_path + con + '_mcmc_spot_1_lon_hist.png')
plt.show()

#spot 2
plt.figure(figsize=(7,7))
plt.hist(mcmc_2.lon2, bins = 50, lw = 1, ec = 'k', color = 'g', label = '
    ↪ Spot 2')
plt.ylabel('Number of incidents')
plt.xlabel('Longitude')
plt.title('Spot Longitude Ranges for 1 sigma variation for contrast ' + con)
plt.legend()
#plt.savefig(save_path + con + '_mcmc_spot_2_lon_hist.png')
plt.show()

#spot 3
plt.figure(figsize=(7,7))
plt.hist(mcmc_2.lon3, bins = 50, lw = 1, ec = 'k', color = 'r', label = '
    ↪ Spot 3')
plt.ylabel('Number of incidents')
plt.xlabel('Longitude')
plt.title('Spot Longitude Ranges for 1 sigma variation for contrast ' + con)
plt.legend()
#plt.savefig(save_path + con + '_mcmc_spot_3_lon_hist.png')
plt.show()



# #spot 4
# axs[3].hist(mcmc_2.lon4, bins = 50, lw = 1, ec = 'k', color = 'orange',
    ↪ label = 'Spot 4')
# axs[3].set_ylabel('Number of incidents')
# axs[3].set_xlabel('Radius Size range')
# axs[3].legend()

# #spot 5
# axs[4].hist(mcmc_2.lon5, bins = 50, lw = 1, ec = 'k', color = 'm', label =
    ↪  'Spot 5')
# axs[4].set_ylabel('Number of incidents')
# axs[4].set_xlabel('Radius Size range')
# axs[4].legend()
#plt.savefig
plt.show()


# In[88]:


plt.figure(figsize=(15,15))
plt.scatter(mcmc_2.rad1, mcmc_2.lat1, c =mcmc_2.chi, cmap = 'Blues_r', s=75,
    ↪  edgecolors='k')
plt.scatter(mcmc_2.rad2, mcmc_2.lat2, c =mcmc_2.chi, cmap = 'Greens_r', s
    ↪ =75, edgecolors='k')
plt.scatter(mcmc_2.rad3, mcmc_2.lat3, c =mcmc_2.chi, cmap = 'Reds_r', s=75,
    ↪ edgecolors='k')
#plt.legend()
plt.title('Radius variation with Latitude for 1sigma range of best fit
    ↪ parameters', fontsize=20)
```

```
plt.xlabel('Radius', fontsize=20)
plt.ylabel('Latitude', fontsize=20)
plt.savefig('/Users/katelincrabtree/Desktop/Latitude_variation' + con + '.
    ↪ png')
plt.show()


# In[16]:


c = '00'
paramfile = np.genfromtxt(save_path  + c + '_parambest.txt', comments='#',
    ↪ delimiter='\t', dtype='f8')

p = pd.DataFrame(index = ['Radius', 'Latitude', 'Longitude', 'Chi␣Squared']
    ↪ )
chisquared = paramfile[16]

stsprad = paramfile[0]

stsplat = paramfile[1]*(180/np.pi) ####always between 0 and 180

stsplon = paramfile[2]*(180/np.pi)

stsprad_plus = str(round(mcmc_2.rad1.max()-stsprad, 3))
stsprad_minus = str(round(stsprad-mcmc_2.rad1.min(), 3))
stsplat_plus = str(round(mcmc_2.lat1.max()-stsplat, 3))
stsplat_minus = str(round(stsplat-mcmc_2.lat1.min(), 3))
stsplon_plus = str(round(mcmc_2.lon1.max()-stsplon, 3))
stsplon_minus = str(round(stsplon-mcmc_2.lon1.min(), 3))


stsprad2 = paramfile[3]
stsplat2 = paramfile[4]*(180/np.pi) ####always between 0 and 180
stsplon2 = paramfile[5]*(180/np.pi)

stsprad2_plus = str(round(mcmc_2.rad2.max()-stsprad, 3))
stsprad2_minus = str(round(stsprad2-mcmc_2.rad2.min(), 3))
stsplat2_plus = str(round(mcmc_2.lat2.max()-stsplat2, 3))
stsplat2_minus = str(round(stsplat2-mcmc_2.lat2.min(), 3))
stsplon2_plus = str(round(mcmc_2.lon2.max()-stsplon2, 3))
stsplon2_minus = str(round(stsplon2-mcmc_2.lon2.min(), 3))

stsprad3 = paramfile[6]
stsplat3 = paramfile[7]*(180/np.pi) ####always between 0 and 180
stsplon3 = paramfile[8]*(180/np.pi)

stsprad3_plus = str(round(mcmc_2.rad2.max()-stsprad2, 3))
stsprad3_minus = str(round(stsprad2-mcmc_2.rad2.min(), 3))
stsplat3_plus = str(round(mcmc_2.lat2.max()-stsplat2, 3))
stsplat3_minus = str(round(stsplat2-mcmc_2.lat2.min(), 3))
stsplon3_plus = str(round(mcmc_2.lon2.max()-stsplon2, 3))
stsplon3_minus = str(round(stsplon2-mcmc_2.lon2.min(), 3))

stsprad4 = paramfile[9]
stsplat4 = paramfile[10]*(180/np.pi) ####always between 0 and 180
stsplon4 = paramfile[11]*(180/np.pi)

stsprad4_plus = str(round(mcmc_2.rad4.max()-stsprad4, 3))
```

```
stsprad4_minus = str(round(stsprad4-mcmc_2.rad4.min(), 3))
stsplat4_plus = str(round(mcmc_2.lat4.max()-stsplat4, 3))
stsplat4_minus = str(round(stsplat4-mcmc_2.lat4.min(), 3))
stsplon4_plus = str(round(mcmc_2.lon4.max()-stsplon4, 3))
stsplon4_minus = str(round(stsplon4-mcmc_2.lon4.min(), 3))

stsprad5 = paramfile[12]
stsplat5 = paramfile[13]*(180/np.pi) ####always between 0 and 180
stsplon5 = paramfile[14]*(180/np.pi)

stsprad5_plus = str(round(mcmc_2.rad5.max()-stsprad5, 3))
stsprad5_minus = str(round(stsprad5-mcmc_2.rad5.min(), 3))
stsplat5_plus = str(round(mcmc_2.lat5.max()-stsplat5, 3))
stsplat5_minus = str(round(stsplat5-mcmc_2.lat5.min(), 3))
stsplon5_plus = str(round(mcmc_2.lon5.max()-stsplon5, 3))
stsplon5_minus = str(round(stsplon5-mcmc_2.lon5.min(), 3))

#p[]=[chisquared, '', '']
#p['']= ['Radius', 'Latitude', 'Longitude', 'Chi Squared']
p['Spot␣1'] = ['$' + str(stsprad.round(3)) + '^{+' + stsprad_plus+ '}_{-'+
    ↪ stsprad_minus +'}$', '$' + str(stsplat.round(3)) + '^{+' +
    ↪ stsplat_plus+ '}_{-'+ stsplat_minus +'}$', '$' + str(stsplon.round(3))
    ↪  + '^{+' + stsplon_plus+ '}_{-'+ stsplon_minus +'}$', chisquared]
p['Spot␣2'] = ['$' + str(stsprad2.round(3)) + '^{+' + stsprad2_plus+ '}_{-'+
    ↪  stsprad2_minus +'}$', '$' + str(stsplat2.round(3)) + '^{+' +
    ↪ stsplat2_plus+ '}_{-'+ stsplat2_minus +'}$', '$' + str(stsplon2.round
    ↪ (3)) + '^{+' + stsplon2_plus+ '}_{-'+ stsplon2_minus +'}$', '']
p['Spot␣3'] = ['$' + str(stsprad3.round(3)) + '^{+' + stsprad3_plus+ '}_{-'+
    ↪  stsprad3_minus +'}$', '$' + str(stsplat3.round(3)) + '^{+' +
    ↪ stsplat3_plus+ '}_{-'+ stsplat3_minus +'}$', '$' + str(stsplon3.round
    ↪ (3)) + '^{+' + stsplon3_plus+ '}_{-'+ stsplon3_minus +'}$', '']

p['Spot␣4'] = ['$' + str(stsprad4.round(3)) + '^{+' + stsprad4_plus+ '}_{-'+
    ↪  stsprad4_minus +'}$', '$' + str(stsplat4.round(3)) + '^{+' +
    ↪ stsplat4_plus+ '}_{-'+ stsplat4_minus +'}$', '$' + str(stsplon4.round
    ↪ (3)) + '^{+' + stsplon4_plus+ '}_{-'+ stsplon4_minus +'}$', '']

p['Spot␣5'] = ['$' + str(stsprad5.round(3)) + '^{+' + stsprad5_plus+ '}_{-'+
    ↪  stsprad5_minus +'}$', '$' + str(stsplat5.round(3)) + '^{+' +
    ↪ stsplat5_plus+ '}_{-'+ stsplat5_minus +'}$', '$' + str(stsplon5.round
    ↪ (3)) + '^{+' + stsplon5_plus+ '}_{-'+ stsplon5_minus +'}$', '']


#p['Chi Squared Value'] = [chisquared, '', '']

#p.set_index('')
#p['Chi Squared Value'] = [chisquared, '','']
#['Minimum Chi Squared', 'Radius 1', 'Latitude 1', 'Longitude 1', 'Radius
    ↪ 2', 'Latitude 2', 'Longitude 2', 'Radius 3', 'Latitude 3', 'Longitude
    ↪ 3']

#print(p)
print(p.to_latex())


# In[ ]:
```

```
# In[ ]:
```