Starspot crossing transits in long-cadence Kepler data: a search for correlations between starspots and stellar properties

Thesis submitted in partial fulfillment of the requirement for

Honors in

Physics

Michelle Gomez

Adviser Leslie Hebb November 10, 2015

Contents

1 Abstract

As visible manifestations of strong magnetic fields, starspots provide an opportunity for us to explore small-scale properties of magnetic fields. With the launch of NASA's Kepler satellite, we now have access to near-continuous, high-precision photometry of thousands of transiting planet host stars that we can use to study starspots. We have written a program that uses long cadence photometry of a sample transiting planet host stars to measure starspot variability caused as the planet traverses in front of starspots on the surface of the star. We analyzed a sample of 249 transiting planet host stars from the Kepler satellite and found 16 whose light curves are strongly affected by in-transit starspots. We are using our total sample of 249 planet host stars to investigate correlations between the presence of starspots and global stellar parameters such as effective temperature and rotation period. In addition, we are using the known position of the planet to explore the latitude of the starspots on the transiting planet host stars. In our analysis, we find that the presence of starspots correlates with rotation period such that there is a decrease in starspot variability in sampled light curves as rotation period increases. In addition, our data suggest that cool stars less than 5000 K are more likely to have starspot crossing features in their transit light curves than hotter stars greater than 5000 K.

2 Introduction

2.1 Stars With Magnetic Fields

Magnets are simple examples of natural magnetic fields. But guess what? The Sun has a huge magnetic field. To learn about the magnetic field on the Sun we must first understand how magnets work and what it actually means to have a magnetic field. A permanent magnet is an object that contains atoms that are arranged in a special type of way in to which exhibit special properties. Magnetic fields are an invisible field that creates repulsive and attractive forces with other objects containing cobalt, iron, and nickel. Although magnets come in difference shapes and sizes, all magnets contain a south pole and a north pole. Figure 1 illustrates sets of bar magnets that have a north pole on one end and a south pole at the other end. In the same way that "like" charges will repel one another and "unlike" charges will attract one another, we have "like" poles that will repel and "unlike" poles that will attract. If we take the north pole of one of the bar magnets and place it near the north pole of a second magnet, because these are like poles, they will both exert a repulsive force on one another. This force will be magnetic. At the same time, if we take the north pole of one magnet and place it near the south pole of another magnet, because these poles are unlike poles, they will create a magnetic force that will be attractive in nature. The term magnetic fields is used to describe the magnetic forces that magnets exert. Since magnets are capable of exerting a force over a distance, we can imagine that each magnet creates its own magnetic field which is composed of magnetic field lines. Magnetic field lines point out from the north poles and into the south poles. The magnetic field lines always form closed loops around a magnet and the number of magnetic field lines per unit area is always proportional to the strength of the magnetic field. In short, a permanent magnet has a magnetic field that is caused by moving electric charges and magnetic moments of particles associated with its spin.

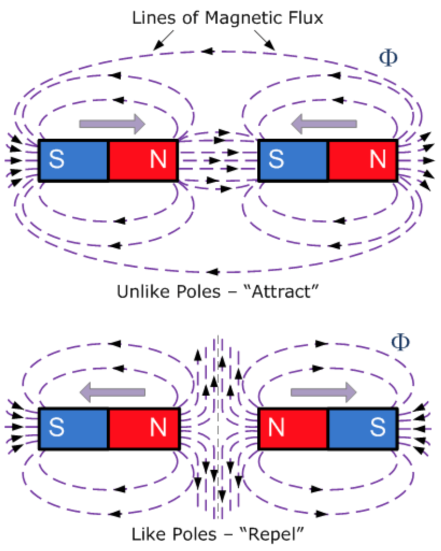


Figure 1: The top panel of the figure illustrates a set of bar magnets that exert an attractive force due to the unlike poles being near each other. The bottom panel of the figure shows a set of bar magnets that exert a repulsive force due to the like poles being near each other. The magnetic field lines are indicative of the repulsive or attractive forces by deflecting when like poles are near and merging when unlike poles are near. Image Credit www.electronics-tutorials.ws

2.1.1 The Magnetic Field on the Sun

The Sun is made up of positively charged ions and negatively charged electrons in a state called a plasma. The Sun is hot and is composed of $\sim 98\%$ Hydrogen and Helium atoms. The heat ionizes the gas and creates a plasma where ample energy is provided to free electrons from atoms, allowing them to move freely and accelerate. Essentially, the Sun is a huge ball of plasma in which hot gasses rotate and convect. Inside this inferno, a magnetic field is generated whose global structure is similar to that of a bar magnet. As shown in Figure 2, the global magnetic field of the Sun takes the form of a dipole whose magnetic field lines emerge out of the negative end of the magnet, at the South pole of the Sun, and into the positive end of the magnet, at North pole of the Sun. The Sun also has small scale magnetic fields that vary spatially and temporarily due to the movement of charged particles [1]. On average, the global magnetic field on the Sun is about 1 Gauss. The global magnetic field is stronger near the poles because the magnetic field lines are much closer together and weaker near the equator because the lines are more spread out. The various bright spots on the Sun come from small scale magnetic fields that play a vital role in the dynamics of the Sun and its atmosphere. To understand more about the Sun's overall magnetic field, we will be exploring small scale magnetic fields on the surface of the Sun.

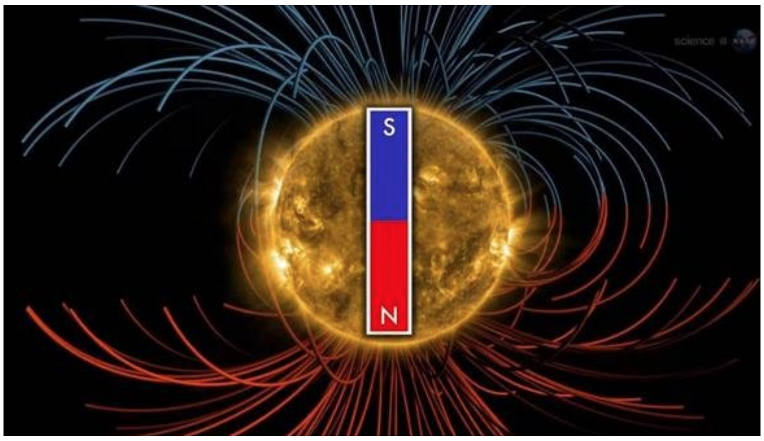


Figure 2: This is an X-ray image of the Sun taken at the Stanford's Wilcox Solar Observatory. The bright areas are regions of strong magnetic fields due to sunspots. Overlaid is a bar magnet indicating that the global magnetic field is a dipole similar to that of a permanent bar magnet. Image Credit Science@NASA

There are still many unanswered questions about magnetic fields on the Sun. For instance, how does the solar dynamo work to amplify magnetic fields and how does the Sun's Corona get heated through repeated sunspot radiation bursts? Stellar magnetic fields are an important area to study and need further exploration in order for astronomers to understand how they play a role in the various stages of stellar evolution. Today, there is a model for the creation of magnetic fields on our Sun. In order to understand the existing model for magnetic field creation, we must first learn more about our Sun and its interior composition as well as its exterior.

The interior of the Sun is segmented into four parts named the Core, the Radiative Zone, the Tachocline Layer or the "interface layer," and the Convective Zone.

The Sun's core can reach temperatures up to 15,000,000 K and is the region where nuclear reactions allow Hydrogen to be converted into Helium. This process releases energy that then leaves the surface of the Sun primarily as visible light. This three step process is called the proton-proton chain (pp-chain).

In the first step of the proton-proton chain, two protons which are also known as hydrogen nuclei collide to make deuterium, a heavy hydrogen atom. This process is the simplest form of nuclear fusion. Hydrogen atoms have a positive electric charge and if we bring two protons together, they will want to repel because they have like charges. The more we try to bring the hydrogen ions together the more energy will be needed to overcome this repulsive force. In high density and temperature environments, the protons will eventually stick, but in doing so the protons give up a little of their mass in the form of energy. This energy can be calculated using Albert Einstein's $E = mc^2$. Where E is the energy, m is the mass, and c is the speed of light. The second and third steps of the pp-chain also involve nuclear fusion of like charges in which energy is generated [2].

The following is an overview of the pp-chain,

1) Two protons collide to create deuterium, a positron, and a neutrino.

2) Deuterium collides with a proton to make helium-3 and a gamma ray.

3) Two helium-3 atoms collide to make a helium-4 atom and two protons.

The total energy of the three stages of the proton-proton chain is approximately $4 * 10^{-12}$ Joules (J). On the atomic scale $4 * 10^{-12}$ J is relatively large, but in everyday terms this is really small. But remember, this is $4 * 10^{-12}$ J for just one cycle of the pp-chain. In order to determine the total power output of the Sun, we need to take into account how large the Sun is and the rate at which the pp-chain occurs. Considering the size of the Sun and the frequency of cycles, the total energy output from the Sun is about $3.8 * 10^{26}$ Joules per second (J/s) [1].

Since the pp-chain does not produce photons (a particle representing a quantum of light) in the visible part of the spectrum, the photons created in the core do not flow through the rest of the Sun and get emitted into space. In the idealized case, the core releases many photons of a single energy and wavelength. A single high energy photon leaves the core and gets converted into many lower energy photons through the radiative processes in the radiative zone. The pp-chain is completely shut off at the outer edge of the core, which is about 175,000 km from the very center of the Sun [1].

The radiative zone extends (distance) from the core. It is named after the method of energy transport called radiation. The energy, in photons, created in the core is carried through the radiative zone by bouncing from particle to particle. The high energy photons interact with particles like electrons and protons as well as the nuclei of other atoms. Once they interact, the photons are scattered in random directions as shown by the zig-zag motion in Figure 3. When a photon interacts with another particle it can give some of its energy to other particles which results in a change in its wavelength. By the time the photons exit the radiative zone, the photons are converted to photons of visible light. It takes a very long time for photons to exit the radiative zone because the radiation zone is so dense that photons only travel a few millimeters before interacting with another particle. It is on the order of millions of years! This means that the sunlight that gave you your tan last year resulted from nuclear fusion and radiative fusion millions of years ago. The temperature at the base of the radiative zone is approximately 7,000,000 C while the outer most layer has a temperature of about 2,000,000 C [1].

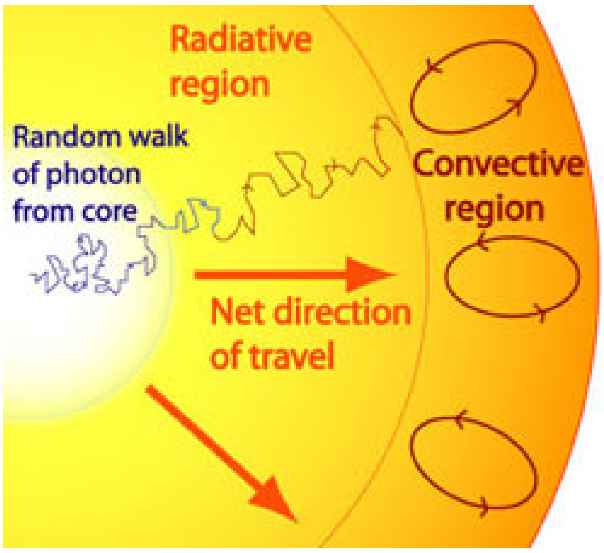


Figure 3: This is an image of the Sun's interior illustrating the net direction travel of photons being generated from the core through nuclear fusion. The core is at the center of the Sun and is surrounded by the layer called the radiative zone, which is surrounded by the convection zone. Image Credit http://stavretta-stavreta.blogspot.com

Extending from the radiative zone is the convective zone. The convective zone extends 496,000 km from the core [3]. The convective zone is made up of hot plasma which is made up of negatively charged particles. Since the outer most layer of the radiative zone is about 2,000,000 C, it is cool enough for heavy atoms, like oxygen and carbon, to hold on to its electrons in the convective zone. The heavy neutral atoms make it harder for energy to be transported through radiation and exit the Sun. The high opacity prevents heat from escaping and makes the plasma unstable. The plasma starts to convect and move in curricular motions different from the motion in the radiative zone as illustrated in Figure 3. The circular motion of the hot plasma quickly carries heat from the base of the convective zone to the surface of the Sun. The plasma cools as it raises and heats back up as it turns toward the base again. The plasma cools to about 5,700 K, otherwise known as the temperature of the Sun [1].

The tachocline is the name for the border between the radiative zone and the convective zone. Most leading theories suggest that this is the place where the solar dynamo mechanism takes place in which magnetic fields are formed and amplified. As a consequence, magnetic fields are present through out the convective zone [4, 5].

The solar exterior is called the photosphere for the surface of the Sun. It is a solid layer and is 100 km thick. The surface is notably known as the place where photons are able to escape. The motions in the convective zone are visible as "granules" on the photosphere. Granules look like small circles that cover the Sun entirely except for areas covered by sunspots. Granules are generated at the top of the convection cells where hot plasma raises from the base of the convection zone to the outer layer of the convective zone closest to the photosphere. Individual granules last for about 20 minutes. Old granules are pushed aside as new ones form, making the surface of the Sun dynamic and changing all the time due to fluid motions in the interior.

Sunspots are another element of the Suns photosphere. As shown in Figure 4, the central dark region of a sunspot is called the umbra and it is surrounded by a lighter region called the penumbra. Temperatures in the umbral zone are commonly about 2400 K while temperatures in the penumbral zone can be as high as 3700 K. Sunspots appear as dark patches on the Sun's photosphere because their brightness are scaled as the fourth-power of the temperature.

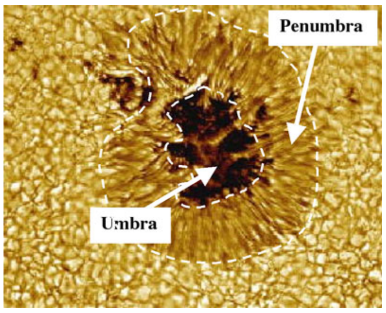


Figure 4: Blown up cartoon of a Sunspot. The umbra is the dark region of the Sunspot while the penumbra is the lighter region surrounding the umbra.

We will discuss Sunspots in more detail because they are an indication of the small-scale magnetic fields that add to the complexity of the Sun's overall magnetic field. In short, sunspots are the visible counterparts of the magnetic flux tubes emerging from the Sun's convective zone. This phenomena is explained through the Babcock Model [6].

The idea that differential rotation and convective motions drive the solar dynamo was first put fourth in 1961 by the astronomer Horace Babcock and is known as the Babcock Model [6]. It is thought that the change in rotation and velocities across the radiative, tachocline, and convection zones can manipulate magnetic field lines. The Babcock Model suggests that magnetic field lines get amplified in the tachocline layer due to its small diameter. The amplification of the field lines create magnetic flux tubes in the convective region that gradually get twisted into spirals by differential rotation in the Sun's interior (See Figure 3). The tubes reach a certain stress limit and burst through the surface of the Sun, creating what we know as sunspots. Convection is repressed at the puncture points which leads to a decrease in energy flux compared to the interior of the Sun, giving rise to the dark appearance of sunspots [6, 7, 8]. To put into perspective, sunspots are visible manifestations of local small scale magnetic fields that are one one hundred times stronger than magnetic field of Earth (0.25- 0.65 Gauss).

Sunspots are usually created in dipole pairs in which one spots will have a north magnetic field and the other will have a south magnetic field. The field is strongest in the umbral zone which is surrounded by a weaker field as part of the prenumbral zone. Groups of sunspots are usually surrounded by pores, or small sunspots, that do not have a penumbra and are just as dark as the umbra [9, 1].

Every 11 years, the direction of the Sun's magnetic field is flipped. This is called the solar cycle. According to the Babcock model, differential rotation is the main driver of the 11-year solar cycle. During the 11 year period, the number of sunspots quickly rises and slowly declines. From cycle to cycle, the global field of the Sun changes polarity from north/south to south/north and back. The point of high sunspot activity is during the solar maximum and the point of lowest activity is known as the solar minimum. Because sunspots are indicative of magnetic fields, this solar cycle is linked to the variation in the Sun's magnetic field [10].

Figure 5 is a two panel plot showing sunspot coverage and the number of sunspots over $\sim$ 12 solar cycles. The first panel shows time on the X-axis vs sunspot area in equal area latitude strips on the Y-axis. The panels shows us that sunspots are confined between +30 degrees north and - 30 degrees south. At the start of a solar cycle, sunspots appear at high latitudes and as the solar maximum approaches the sunspots move closer to the equator. At the end of the cycle, starspots start to migrate and re-appear at high latitudes. This cycle gives rise to a "butterfly" looking pattern. Because sunspots are indicative of magnetic fields, this solar cycle is linked to the variation in the Sun's magnetic field and can help us understand the solar dynamo. So far, this data show us that sunspot do not appear at random over the suns photosphere, but are confined over two latitude strips on either side of the equator.

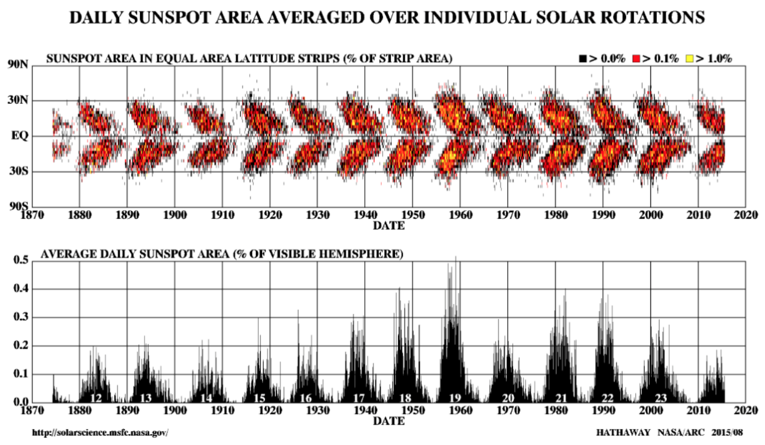


Figure 5: Charted data of sunspot migration and coverage over 12 solar cycles. Sunspots start at high latitudes and reappear toward the equator as the number of sunspots increases. This cycle gives rise to a butterfly looking pattern and indicates the the magnetic activity on the Sun is constantly fluctuating.

There are several ways to detect and measure the magnetic field strength of the Sun. Zeeman splitting and Helioseismic and Magnetic Imagers are a few ways to measure the magnetic field strength of the Sun which we will speak about.

2.1.2 Detection of Magnetic Fields on the Sun

Observed by physicist Pieter Zeeman, the Zeeman effect is characterized as the splitting of spectral lines into two or more components at different frequencies by an applied magnetic field.

When an element is heated, the resulting emission line is the result of an energized electron jumping to a higher energy state and falling back to a lower state. Upon falling to a lower state, it releases some energy as a specific wavelength of light. When in a magnetic field, the emission line is split into many lines as shown in Figure 6.

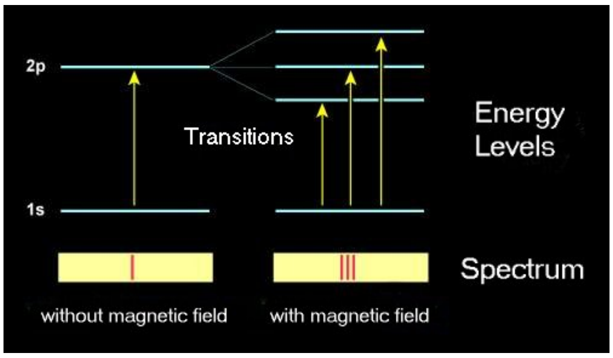


Figure 6: Example of energy state of an element when no magnetic field is applied and the splitting of the energy levels when a magnetic field is applied.

It is important to realize that electron spin is a factor in the interaction of atoms when a magnetic field is present because all spin one-half nuclei have a magnetic dipole. In other words, spin one-half nuclei, like electrons, have something like a compass needle that aligns itself with or against a magnetic field. If the dipole is aligned with the magnetic field, then it is at the alpha spin-state. If the dipole is against the magnetic field, then it is at the beta spin-state. The difference between the spin states is called Zeeman splitting. The amount of splitting is proportional to the strength of the magnetic field [2]. This can be better seen in an applied magnetic field vs energy plot in Figure 7. In Figure 7, ΔE is the change in energy, h is Planck's constant, ν is the light frequency, g is the gyromagnetic ratio, μ is the magnetic moment, and B_o is the initial magnetic field.

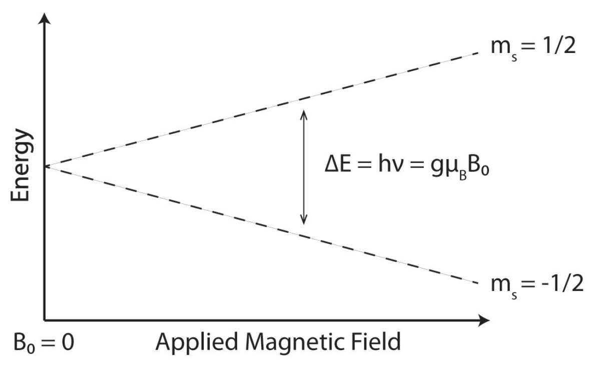


Figure 7: Plot illustrating how the energy of atoms change with an applied magnetic field. The difference in the alpha and beta spin states, otherwise known as the amount of splitting, is proportional to the strength of the magnetic field.

The Zeeman effect allows us to measure the strength and direction of the Sun's magnetic field by observing the difference in energy of the light emitted as electrons jump. In other words, the energy levels of atoms jump and split into more than one level when a magnetic field is applied. This causes spectral lines to also split into more than one line and the number of lines are proportional to the strength of the magnetic field. The change from more than one level of energy when a magnetic field is applied is called Zeeman Splitting. If we observe Zeeman splitting in the spectrum then we can infer the presence of magnetic fields. Now we can measure the strength of the magnetic field by measuring the amount of Zeeman splitting [11].

The Helioseismic and Magnetic Imager (HMI), also known as a vector magnetogram, is useful in mapping the magnetic field on the Sun's surface and how it changes over time. HMI interprets the way light in influenced and travels through the field in order to measure them from far away. Light is split into different wavelenths based on the magnetic field around and its strength. This is the Zeeman effect! The HMI observes the light and makes vector magnetograms at high resolution that will give plausible indications and detail about the Suns magnetic field on its surface [12].

We can use Zeeman splitting to measure the strength of magnetic fields on the Sun, including the magnetic fields emitted from sunspots. Although we can resolve sunspots on the Sun, we are not able to resolve starspots on other stars because they are too far and starspots are overlooked by its luminous surrounding. However, if stars are far away, we are not easily able to use these methods to measure the strength of the magnetic field on other stars. In order to study magnetic fields on stars other than our Sun, we will be using the light curves of stars with planets to resolve starspots.

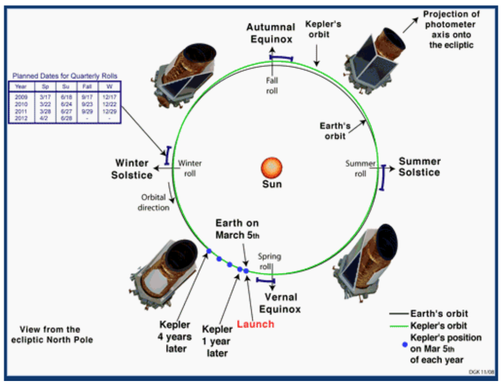
2.2 Transiting Planets as Tracers of Magnetic Fields on Other Stars

The Kepler satellite provides us with near-continuous high-precision photometry of thousands of transiting planet host stars. Kepler monitors the brightness of star's and detects orbiting planets through light curves. As planets cross in front of a star, they sometimes cross by starspots on the star.This has a unique effect on the light curve. As the star rotates, the sunspots appearance will come in and out of view causing changes in the star's brightness. To understand more about how magnetic fields are generated, I will use transiting planets as a probe to map starspots, a visible manifestation of magnetic fields.

Kepler is a satellite launched by NASA in March of 2009 from the Cape Canaveral Air Force Station in Florida and was expected to last up to 6 years, but only lasted 4.5 years. The scientific objective of Kepler is to analyze the diversity of planetary systems. Kepler's goal is to detect Earth-sized planets near the habitable zone in stars of different spectral type by examining the size and shape of the orbits of these planets.

Kepler is a 0.95 meter aperture Schmidt-type telescope with a 1.4 meter primary mirror. Its single purpose is to continuously stare at a large patch of the sky to observe the variation in brightness of thousands stars at the same time. Kepler is about 2.7 meters in diameter and 4.7 meters in height and orbits our Sun trailing behind the earths orbit as shown in Figure 8a. In order to maintain power, Kepler rotates its position every 3 months by 90°. The main instrument on Kepler is a photometer. A photometer is an instrument that measures the brightness of stars and their variations. The photometer consists of the telescope, the focal plane array and the local detector electronics. A telescope is an optical device designed to make distant objects appear nearer. The photometer features a detector with 95 million pixels. Starlight (photons) enters the telescope, reflects from the primary mirrors to the detector of 21 modules. Each module has two 50x25 mm 2200x1024 pixel CCDs. The pixels collect the photons and the array reads out the number of photons on each pixel every six seconds. The number of photons are stored on an on board computer. The computer adds the six-second data snapshots into 30 minute observations. The star's data is sent to earth once every month via NASA's Deep Space Network [13]. All of these components can be seen in Figure 8b.

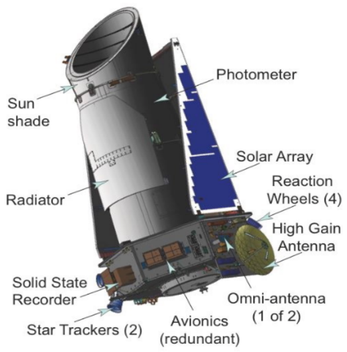
Data obtained from Kepler are available in two cadences, long cadence (LC) and short cadence (SC). A cadence is composed of several 6.02 second exposures with 0.52 readout times. The short cadence data integrate over nine exposures to give a 58.9 second observation. The long cadence data integrate over 270 exposures to give 29.9 minute observations [13].



Planned Dates for Quarterly Rolls

Year	Sp	Su	Fall	W
2009	3/17	6/18	9/17	12/17
2010	3/22	6/24	9/23	12/22
2011	3/28	6/27	9/29	12/29
2012	4/2	6/28	-	-

(a)



(b)

Figure 8: (a) Illustration of Kepler's orbit around the sun while trailing behind the Sun. Kepler reorients itself every three months by 90° to maintain power. (b) A replica of Kepler showing its different components and technologies

To fulfill the goals set by the Kepler team, the Kepler satellite looks at only one portion of our Milky Way Galaxy near the constellations of Cygnus and Lyra (see Figure 9). Cygnus and Lyra are the field of view of choice because it is rich in stars and it was a section in the sky that was far enough north from the plane of Earth that the Sun would not get in the way of Kepler's field of view. The stars that Kepler monitors are up to a thousand light years away where one light year is about 6 trillion miles [13].

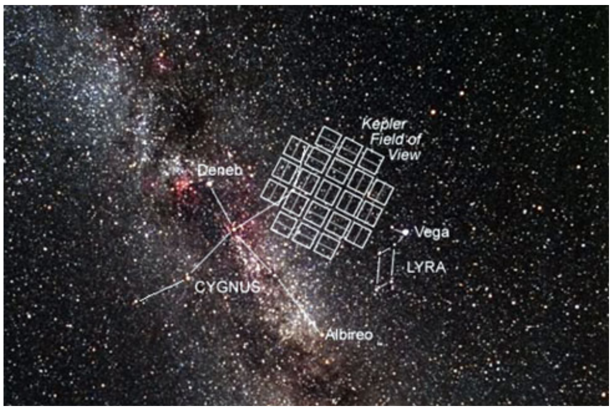


Figure 9: Kepler's field of view in the constellation of Cygnus and Lyra. The section is rich in stars which is perfect for Kepler's instrumental goals of detecting how the brightness of stars change over time.

To detect planets, Kepler looks for the slight dimming in stars as a planet crosses the star or 'transits the star.' Detecting changes in brightness caused by a transiting planet may be impossible to detect if the star is too dim or noisy. Important factors for selecting stars are its brightness and temperature.

Kepler magnitude (Kp), is defined as a measure of the stars brightness as seen by Kepler's photometer and is estimated using a broad filter. The Kp and the brightness of the star have an inverse relation. The brighter the star, the lower the Kepler Magnitude. The figure below is the distribution of Kp and temperature of stars observed by Kepler[13].

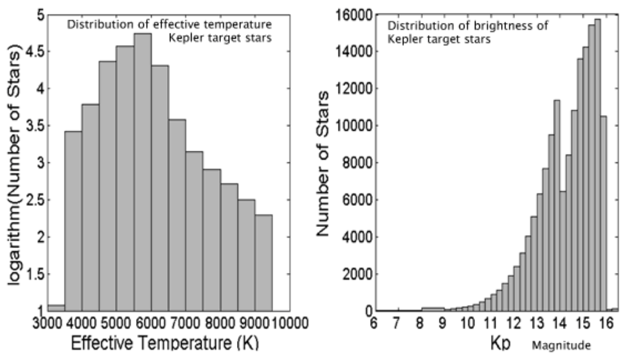


Figure 10: Temperature and Kepler Magnitude distributions of stars studied by Kepler. Most stars are Sun-like and have temperatures of about 5700 K. Kepler Magnitudes extend from 6 to 16 and are indicative of the stars brightness. Large Kepler magnitudes mean the star is faint while low values indicate that the star is brighter.

Not all stars in Kepler's field of view have transiting planets. In order to see the planet transiting the star, the planetary system needs to be nearly perfect with the satellite's line of sight. Although Kepler looks at thousands of stars, Figure 11 shows that only a fraction of observed stars will have transiting planets.

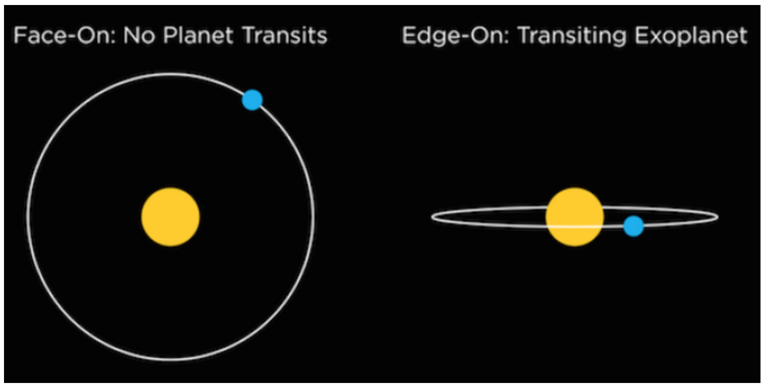


Figure 11: Not all stars monitored by Kepler have a transiting planet. If the planet is not in Kepler's line of sight, the planet will not change the brightness of the star.

During the past two decades thousands of planets have been discovered by this transiting method. To confirm a planet, the science mission team requires a minimum of three transits of the same period, depth and duration.

Just to make it clear, transits occur when a planet passes in front of its host star as viewed by Kepler and a portion of the star's overall light is blocked. How much of the stars light is blocked depends on the size of the planet and the size of the star.

2.2.1 Transiting Planets and Light Curves

A plot of brightness on the Y-axis vs time on the X-axis, as shown in Figure 12, is called a light curve. Brightness increases as you go up the graph and time advances as you move to the right. Light curves clearly depict the blocking effect of planets as they cross in front of a star.

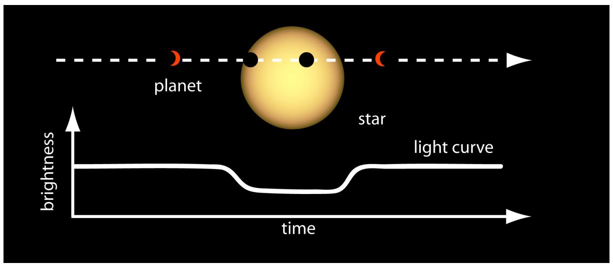


Figure 12: A transit is a characteristic drop in the stars brightness as a planet moves across its face.

Although the amount of blocked light only depends on the size of the planet and the size of the star, the shape and occurrence of the transit light curve is constrained by five parameters: The limb darkening of the star, the epoch, the depth of the transit, the impact parameter and the orbiting period of the star.

Limb darkening is an optical effect seen on all stars, where the center of the star appears brighter than the limb of the star. The photons emitted from the edge of the star reaches an optical depth at a different altitude. At this altitude the temperature is cooler and the radiation is less compared to the center of the star. This difference causes the apparent darkening at the edges of the star [14].

If the temperature on the star is constant, then the transit in the light curve will be boxy. If the same planet is now crossing in front of a star that is affected by limb darkening, then the light curve will have a symmetrical and smooth like transit due to the temperature gradient on the star as illustrated by Figure 13.

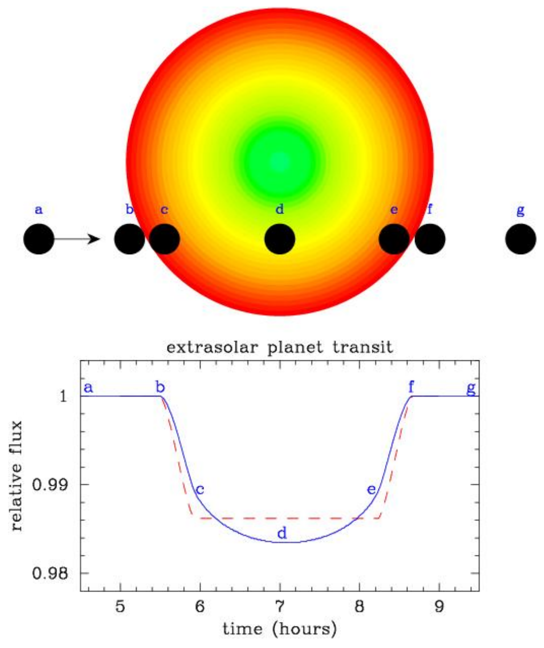


Figure 13: Illustration of a transit and the resultant light curve for both a stellar disk of uniform brightness (dashed line) and one that includes limb-darkening (solid line). Image Credit https://inspirehep.net

The epoch is a moment in time that is used as a reference point for a time-varying astronomical quantity. Transiting planets are time-varying astronomical quantities that create periodic transits in a light curve. Epoch times are needed for the periodic light curve in order to distinguish the times and how often transits occur. The number of transits in a light curve depends on the orbital period of the planet. The orbital period of the planet is the time it takes for the planet to make one complete revolution around its host star or how long a transit lasts in the light curve.

The impact parameter (b) is also an important parameter in determining the shape of the transit. The impact parameter gives us information about the latitude at which planets cross the stellar disc. At impact parameter $b = 0.0$ the planet is transiting directly at the center of the star while a value of b>1.0 implies that the planet is a non-transiting planet. This means that the maximum value of the impact parameter is b=1 (see Figure 14) [14].

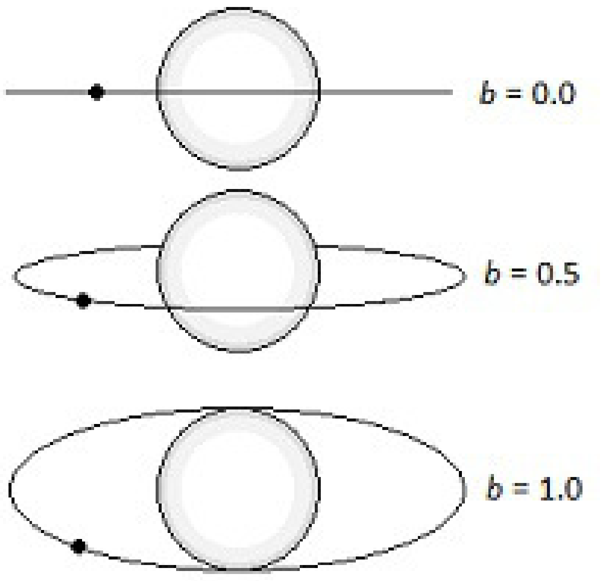


Figure 14: The impact parameter is how far the transit chord is from the center of the stellar disc. At an impact parameter of zero, the planet is passing directly in front of the star. As the impact parameter increases the planet transit chord moves toward the tip of the star. The maximal impact parameter value is one. Image Credit https://exoplanetmusings.wordpress.com

Changing the impact parameter will change the shape of the transit even if the planet and the star remain a constant size. Figure 15 shows two transit light curves. One light curve has an impact parameter of b = 0 and the other has a higher impact parameter. It is valuable to notice that changing the impact parameter changes both the shape of the transit as well as the duration of the transit. It takes more time for the planet to complete a full transit duration if the impact parameter is zero compared to when the impact parameter is higher.

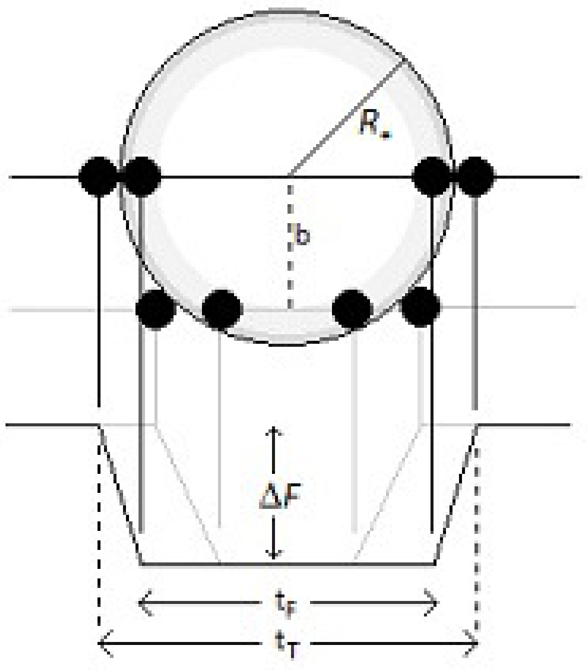


Figure 15: Illustration of two same size transiting planets with different impact parameters and their resultant light curves. A difference in impact parameter does not affect the depth of the transit because the amount of blocked light is dependant on the size of the planet and the size of the star. The amount that the planet blocks is denoted as ΔF. However, the impact parameter does manage to change the duration of the transit. Image Credit https://exoplanetmusings.wordpress.com

Figure 16 is a clear indication that the depth of the transit depends only on the size of the planet relative to the size of the star. There are planets in four instances of this cartoon. These four contact points are important to know in order to successfully have a discourse on light curves. Contact point 1 is the instance where the planet blocks the first photon of light from the star. Contact point 2 is the instance where the planet is fully in front of the star. Contact point 3 refers to the moment where the planet is lastly fully traversing in front of the star. The fourth contact point is when the planet covers the last photon of light from the star. The cycle repeats. Going from contact points 1 to 2 is considered the ingress because the planet is going into the star. Going from contact points 3 to 4 is considered the egress because the planet is exiting the star. Going from contact points 1-4 is considered a full transit [14].

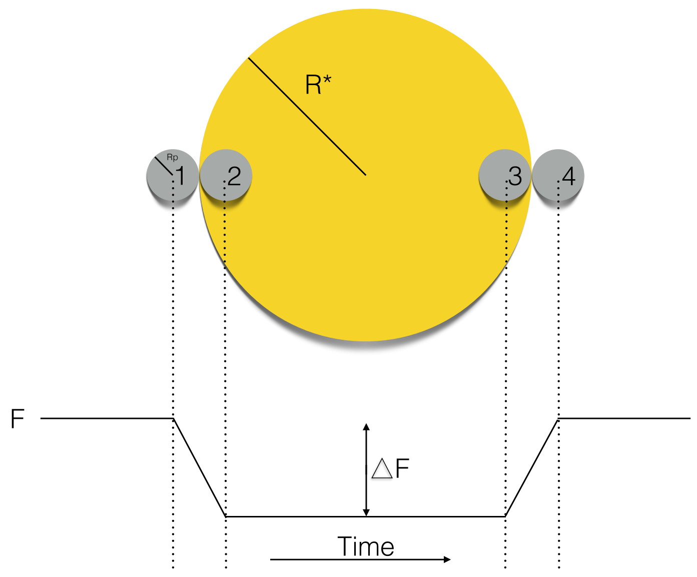


Figure 16: Cartoon showing the four contact points of a transit. Going from contact points 1 to 2 is considered the ingress and going from contact points 3-4 is considered the egress. In sum, going from contact points 1-4 is considered being in-transit. This plot shows us that the dransit depth is only dependant on the size of the planet and the size of the star.

The transit depth can be calculated using the following equation.

$$\frac{\Delta F}{F} = \frac{R_p^2}{R_*^2} \tag{1}$$

Where F is the flux measured from the star by Kepler, ΔF is the change in the flux as the planet fully transits in front of the star and $\frac{R_p^2}{R_*^2}$ is simply the planet to star radius ratio. This equation tells us that a large planet passing in front of a small star is going to produce a deeper transit and will block more of the stars light than a small planet passing in front of the same star. Or inversely, if a planet is transiting a large star it will produce a shallower transit depth. The same planet transiting a smaller star will produce a deeper transit depth (see Figure 17).

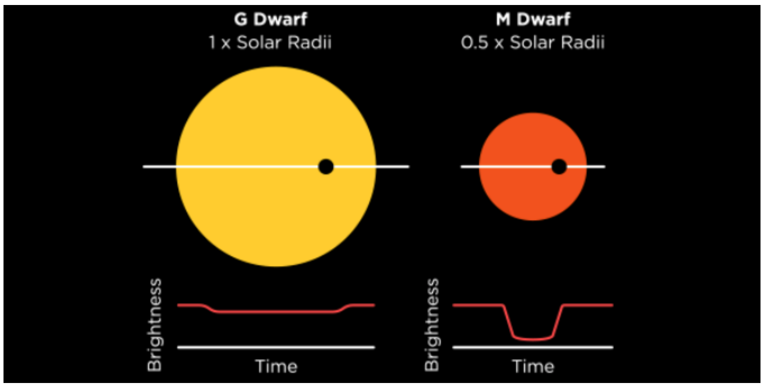


Figure 17: When the planet crosses in front of the large star it blocks less of the stars overall brightness. This results in a shallower transit depth in the light curve. If the same planet crosses in front of a smaller star, the planet blocks more of the stars overall brightness which will result in a deeper transit depth.

It is important to realize that the distance between the planet and the star does not matter when it comes to the depth of the transit light curve. This is because the distance between Kepler and the exoplanet is almost identical to the distance between the Kepler and the host star.Therefore, it does not matter how far away a planet is orbiting around the star, it will cause the same depth of the transit light curve.

2.2.2 Transiting Planets as Probes of Starspots

Lets not forget what is happening in real time as Kepler observes these stars. Not only is Kepler seeing planets orbiting around the stars, but the stars themselves are rotating! So Kepler is seeing a difference or variation in the stars overall brightness in time.

If we assume Kepler is looking at a star with no transiting planets and no starspots on its surface, then as the star rotates the overall brightness of the star will remain constant. If we were to plot this light curve we will only see a straight line. But what happens if the star has starspots?! In this next section we will show the effect starspots have on a light curve. Recall, starspots are visible manifestation of magnetic activity. Starspots affect the temperature of a star on its photosphere. In fact, the starspots are much cooler and darker then the rest of the star's photosphere.

If we assume Kepler is now looking at a star with starspots on its photosphere, then as the star rotates and the starspots go in and out of view, the light curve will have a long scale variation in brightness. If we plot this light curve we will see a sinusoidal like curve depicting the brightness of the star as it rotates.

In both cases, if we add transiting planets, then we will see transits or dips in the overall light curves. If we zoom into the actual transits we will see that if the planet is not transiting a starspot, then the transit dip will be symmetric and smooth like. However, the transit will look different if the planet is transiting in front of a starspot on the photosphere of the star as shown in Figure 18.

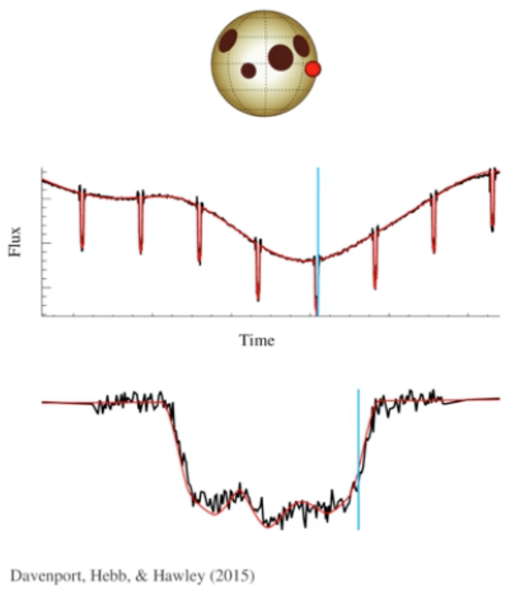


Figure 18

When a planet transverses in front of a star spot, there is a positive bump in the transit. We see this because the star spot is much cooler and darker than the rest of the stars photosphere. Therefore it is like the Star is blocking less light.

By the end of the project we will be able to single out every planetary system that Kepler studies that displays starspot crossing features in its light curve. We will be using the effect and shape that starspots bring to a transit light curve as a probe for mapping starspots. Starspots can be as small as the Earth and may be undetectable by eye, amateur telescopes, and even Kepler! By using transiting planets as a probe, we are able to use the well known position of the planet (the impact parameter) to explore latitudes of starpsots. Likewise we will look for correlations between the stellar properties and systems that display an indication of starspots in the light curve. In finding a pattern or a correlation we will better understand how and where starspots prefer to form and how these correlation may relate to the habitability of nearby planets as well as star evolution.

I have written a program in the computer programing language python designed to detect and characterize starspots in planet transit host stars observed by the Kepler satellite. The goal of my program is to create a method where I can detect starspot crossing events in-transits. I quantify the variation in brightness in-transit as the planet crosses the starspot. I will use this quantization as a measure of low scale magnetic activity.

3 Sample of Stars with Transiting Planets

3.1 Downloading Light Curves from Kepler

We downloaded all available long cadence photometry for 4,695 planet host stars through MAST (Mikulski Archive for Space Telescope). MAST is funded by the National Aeronautics and Space Administration (NASA) and provides astronomical data associated with an array of missions. Data are archived and available to the public free of charge. Since Kepler orients itself 90° every three months to maintain power by keeping solar panels pointed at the sun, Kepler light curve files are segmented in to files with three months worth of data. The Kepler mission lasted 4.5 years and accumulates up to 18 quarters.

Next, we downloaded properties pertaining to exoplanets and their hosts stars from the NASA Exoplanet Archive. The NASA Exoplanet Archive is a catalogue that collects information on exoplanets and their host stars and provides ways for astronomers to work conveniently with large quantities of data. Properties downloaded from the NASA Exoplanet Archive include: The planet name (KOI), the star name (Kepler ID), the planets orbital period, the epoch, the impact parameter, the inclination angle, the orbital semimajor axix to star radius ratio, the eccentricity, the longitude of periapsis, the planet to star radius ratio, the transit duration, the stellar temperature, the stars limb darkening coefficients, and kepler magnitudes. These properties are necessary for identifying and characterizing stars with starspot crossing events in their transit light curves.

I also downloaded all available stellar rotation periods for stars studied by Kepler. Because they were not include in the NASA Exoplanet Archive, I extracted these values from literature: Walkwicz 2013 [15]; Nielsen,2013 [16]; Reinhold, 2014 [17]; McQuillan, 2013 [18], 2014 [19]. These papers provided me with 87 unique stellar rotation periods.

3.2 Choosing Light Curves with Deep Transits

The ability to readily detect the signal of a starspot crossing feature in the light curve depends on both the properties of the star and the signal to noise of the data. The signal to noise ratio is a measure of photon counts relative to systematic read noise due to Kepler. We selected an initial subset of planet and host stars designed to maximize the probability of detecting the desired starspot feature in-transits.

Transits, which are caused by a dip in the stars light as the planet crosses in front of it, are difficult to detect because the decrease in brightness is microscopic compared to the overall brightness of the star. By constraining the size of the planet and its host star, our intention is to create a sample that maximizes the depth of transits in light curves. Transit depths gives us intuition about the size of the planet relative to its host star (see Figure 16). Light curves with deep transits inform us that planets are relatively large compared to the size of its host star. If planets are large, then they are able to block more of a stars overall flux and are more likely to cover a starspot when orbiting its host star. Light curves with shallow transits do not expand my probability of detecting a starspot signal in transits because the signal to noise of these light curves are fairly low which inhibits my ability to detect a starspot crossing event.

It is said that an Earth-size planet will obscure 1/100th of 1% of the stellar light. That being said, a Jupiter-size planet will cause a dip of $\sim 1\%$ or 0.01 of the stellar light. For this reason, the criteria for deep transits in my program was set so that the planet to star radius ratio ≥ 0.0025 which will obscure at least $\sim .25\%$ of the stellar light [13].

It is worth noting that the distance between the planet and the star does not affect the depth of the transit because the distances are enormous and essentially the same as seen by Kepler.

3.3 Choosing Light Curves with Flat-Bottom Transits

A grazing transit is defined such that not all of the planet transits the host star's disc. This means that the second and third contact points are missing from the transit light curves, which refer to when the planet is fully in front of the star (see Figure 16). When the planet is not fully in front of the star, we typically say that the planet is either in ingress or egress. Therefore, we can presume that the planet is always in an ingress/egress phase when its second and third contact points are missing. A star is not uniformly illuminated and its edges have a darker appearance compared to the center due to limb darkening (see Figure 13). Essentially, the planet is always crossing the edge of the star which causes a conformational change to the transit. When the second and third contact points are missing, the transit light curve has a continuously varying "V" shape.

At the end of my program we generate a model that takes the properties of the planet and star, such as the limb darkening coefficients. The model is designed to fit the data in the light curve as good as possible, including the rapid curvature of the ingress and egress sections. However, the rapid change lowers our signal to noise ratio and gives the model the potential to be less accurate due to the uncertain limb darkening parameters. Therefore, we decided not to include the ingress and egress portion of the light curves in our analysis as well as any planet that causes a grazing transit light curve.

If the planet to star radius ratio ($\frac{Rp}{R*}$) plus the impact parameter (b) is greater than one, then the planet is considered having a grazing transit.

$$\frac{Rp}{R*} + b \geq 1 \tag{2}$$

After applying these constraints to the 4696 hosts stars observed by Kepler, our final sample included 249 host stars.

4 Identifying Starspot Crossing Features in Light Curves

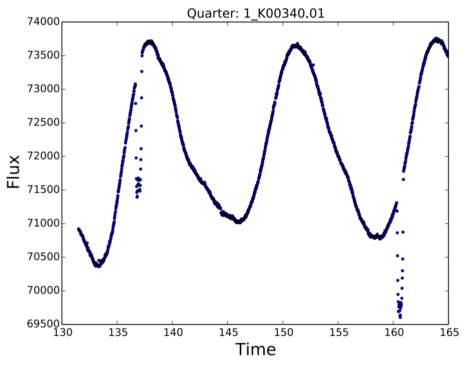
Before getting to understand the programming portion of the project it is key to remember that Kepler looks at a single patch of the sky and observes how the brightness of stars change over time. The change in brightness can be caused by several real phenomena. For example, Kepler can identify the fractional decrease in the stars brightness as a planet transverses the star. It can also detect how the brightness of a star changes over time as it rotates. Recall that starspots are dark patches on the surface of the star and are cooler than the rest of the photosphere. If a star has no star spots on its surface, then we will see no changes in the overall brightness of the star as it rotates over time. If starspots are uniformly distributed around the star, we will also see no changes in the overall brightness of the star. Furthermore, if the planet has more starspots on one side compared to the other, the overall brightness of the star will vary.

The program is designed to look at every unique planet host star system on a quarter by quarter basis.

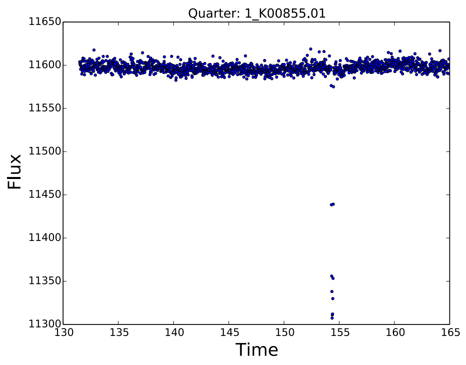
The underlying goal of the program will be to look at flux modulations due to starspots as the planet transverses in front of the star. We are not concerned with the long scale variation in brightness as the star rotates and starspots go in and out of view. We will want to modify the data for each transit in such a way that will allow us to compare the data to a perfect model transit light curve without modifying the intrinsic properties of the transit. The model light curve we are comparing the data to is normalized to have the overall brightness of the star set to unity, while the transit dip is represented as a fraction. Eventually, we will want to normalize our light curves to look like the model.

First, we open the first available long cadence file for the a planet host star system. The long cadence files provide information about the brightness of the star as a function of time so we extract three columns worth of data: Flux, Time, and Error. The flux is a measure of the brightness of the star and the higher the flux value, the brighter the star. The time column is the time at which Kepler creates one point for 30 minutes of integration and the error is the error associated with the flux measurement.

The following figure shows data for two different stars over 35 days.



(a) Full Kepler light curve for K00340.01



(b) Full Kepler light curve for K00855.01

Figure 19: Full Kepler light curves for quarter 1

Figure 19 displays information about two types of stars. Kepler object K00340.01 shows ample variation in the overall light curve. This allows us to assume that there are starspots on the surface of the star. We cannot say the same about Kepler object K00855.01 because the long scale variation for the overall light curve shows little to no variation. Since we want to modify each transits in a way that will allow us to compare the data to a model without modifying the properties in-transit, we will need to extract each transit in the light curve and segment them into three parts: The full transit (dip), one full transit duration to the left of the full transit, and one full transit duration to the right of full transit (see Figure 20).

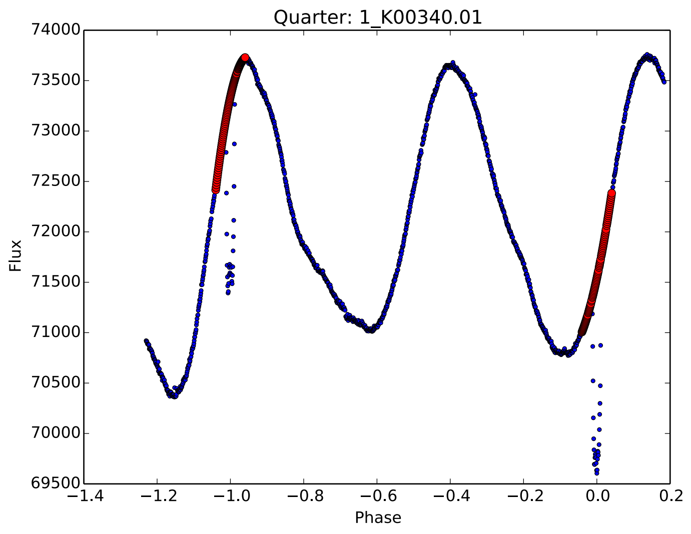


Figure 20: This is an overall light curve for Kepler object K00340.01. The blue points are the data obtained by Kepler while the red points mark the range of data we will extract for each transit. Range of data includes one transit duration to the left of transit, one transit, and one transit duration to the right of transit.

To remove the long scale variations out-of-transit without modifying the transit itself, we will only fit the out-of-transit region with a second order polynomial as shown in figure 21 , then we will take the difference between the polynomial fit and the individual transit.

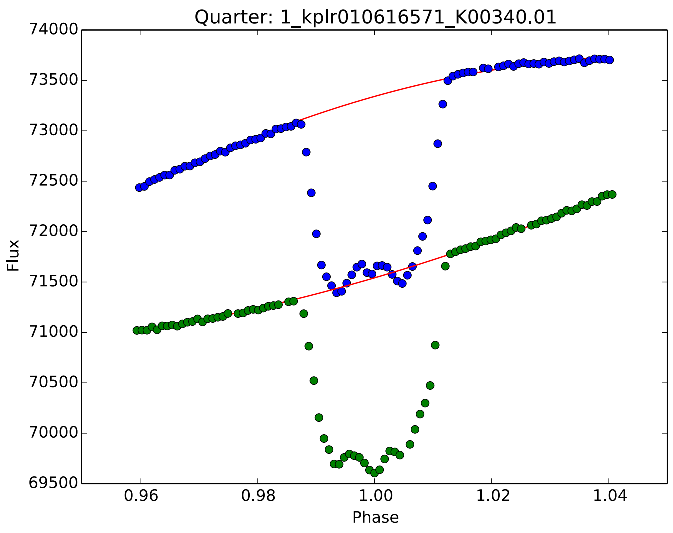


Figure 21: A second degree polynomial was fit to the out-of-transit region of Kepler object K00340.01 in order to remove long scare variation in the overall light curve.

Figure 21 possesses two transits which are denoted by the green and blue data points. The red solid curve is the second order polynomial fit which only runs through the out-of transit region. Fitting the out-of-transit region with a second degree polynomial fit, then subtracting the entire transit from the fit will flatten the out-of-transit region while preserving the shape in-transit as illustrated in Figure 22. Although not all transit light curves have a curvy-like out-of transit regions, we decided to apply this method to every transit light curve to keep the procedure consistent. When we subtract the polynomial fit from the entire transit the out-of-transit region will be centered at zero while the in-transit region will be in some arbitrary number of negative photon counts.

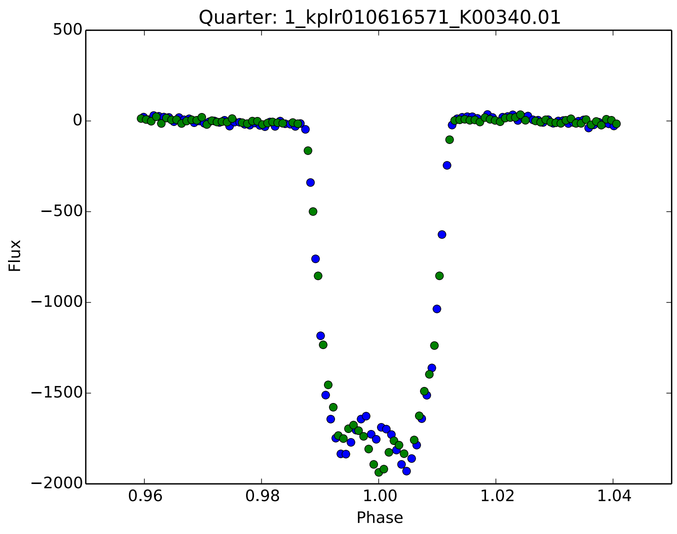
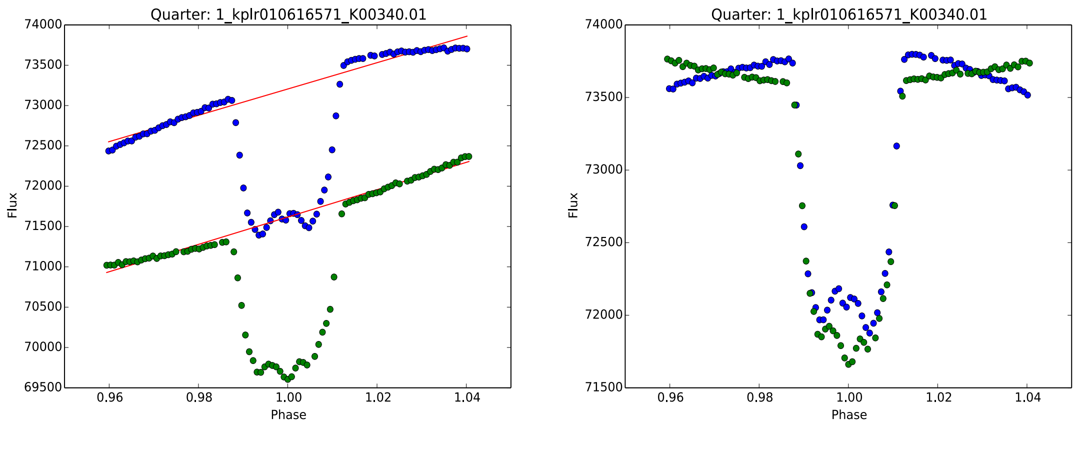


Figure 22: After taking the difference between the second order polynomial fit to the out-of-transit region and the entire transit, the new out out-of-transit region is flattened and the in-transit region is not modified.

A polynomial fit of the first order will not work out in our favor. A first degree polynomial will simply generate the line of best fit for the out-of-transit region. A line passing through a light curve that displays a varying out-of-transit region will not follow the curvature of the data. In fact, it will miss most of the data. Taking the difference between the line and the entire light curve will not result in a flat-like out-of- transit region and will modify the shape of the in-transit region. Figure 23 is an example of what the transits would look like if we applied a first degree polynomial fit. This method will work for quiet stars but not variable stars.



(a) (b)

Figure 23: Illustration (a) shows the line of best fit to the out-of-transit region for Kepler object K00340.01. This object has long scale variation in the overall light curve. When the difference between the line and the entire data is plotted, then we get a modified in-transit light curve with un-flattened out-of-transit regions as seen in part (b).

Similarly, we choose to take the difference between the polynomial fit and the data instead of dividing for a specific reason. Normally astronomers would divide. However, the two transit in quarter 1 for Kepler object K00340.01 occurs at two different flux baselines (about 15000 photon counts away). Taking the difference between the second order polynomial fit and the data will result in a change of units, but the flux points in the time series will look just as the same as before and the shape is still preserved. Controversy, dividing the second degree polynomial fit and the data will not preserve the shape of the transit because we will be dividing both transit by different numbers. Ideally, both transit are identical. If we divide both transit by a different value, then the shape and depth of each transit will change depending on value we divide each transit by. Dividing by different values will add unwanted shape and scatter to the data that will not allow a good fit to the theoretical model. We conclude that a second degree polynomial fit to the baseline then subtracting the polynomial from the data is the best method to remove the long scale variations out-of-transit.

Our model is normalized to unity and has a transit dip in relative counts. At this moment, our transit dips are in absolute photon counts. To make the data to look like the model, we added the peak flux value of the overall light curve (see Figure 19) to the differential light curve (see Figure 22), then divided the newly generated light curve by the same peak value.

To calculate the peak flux value of the overall light curve, we calculated the ninety-fifth percentile of the flux values in the light curve. We did not pick exactly the maximum value because the very spotty stars display solar like flares on its light curves. The flares are not representative of the natural atmosphere of the star because flares occur at random moments.

After applying a second degree polynomial fit to the out-of-transit region, taking the difference between the polynomial fit and the data, adding the peak flux value, and dividing by the same peak flux value we end up with a flat-like out-of-transit region. We also end up with a transit dip represented as a fraction as show in Figure 24.

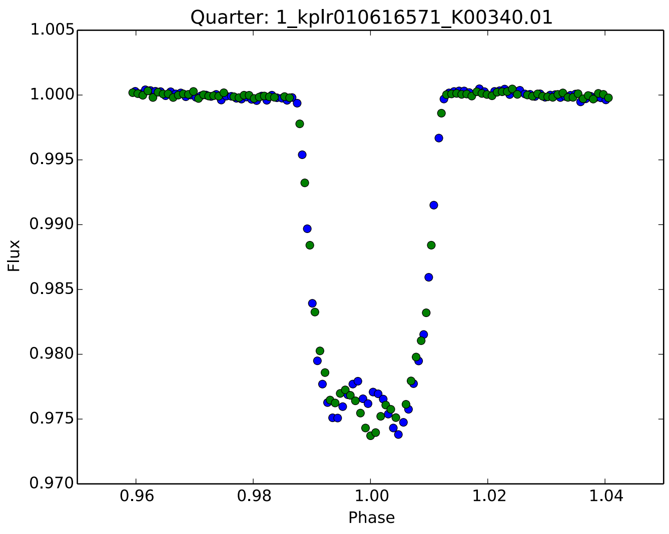


Figure 24: Normalized Light curve for Kepler object K00340.01 quarter 1. The light curve is normalized to one and the transit depth is represented as a fraction.

We generated a model or "theoretical transit" to compare the Kepler data to. The model uses the well known Mendel and Agol 2002 equations for light curve generation. Properties of the star and planet used to generate a model for each planet host star system are the epoch, time, period, inclination angle, the orbital semimajor axix to star radius ratio, the eccentricity, the longitude of periapsis, the planet to star radius ratio, and the limb darkening coefficients to make a theoretical light curve.

The model assumes instantaneous position of the star and the planet to get the total flux at each time. This is an acceptable assumption for SC data, but not for LC data where each data point includes the flux of the star and planet system over 30 minute intervals. To modify the model light curve, we convolve the model with a 30 minute box-hat function. Convolution is a mathematical function that blends two functions together. Our two functions are the 30-minute box-hat function and the model light curve. The blending of the two functions result in a new model with a smother line in the ingress and egress parts of the model light curve.

(Figure of non-convolved model compared to convolved model)

The model overlaid with the data is shown bellow:

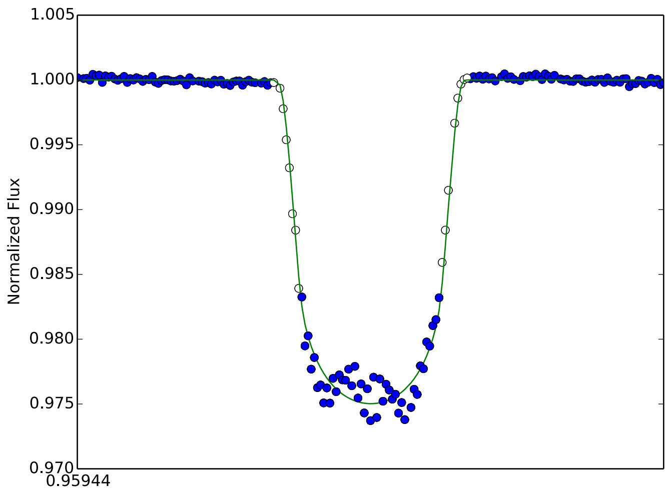


Figure 25: Normalized light curve for Kepler object K00340.01. The light curve is phase folded against a perfect transit model.

Now that we have modified the data and model in a way that will allow me compare the two, we will take the difference between the normalized transit and the convolved model to derive residuals as shown in Figure 26. Residuals are defined as the observed data minus the expected data. The observed data is the data taken form the long cadence Kepler files whereas the expected data is the model light curve. The residuals are segmented into two groups named the out-of-transit residuals and the in-transit residuals. We do not use the ingress/egress residuals because they are rapidly changing parts of the light curve that are prone to miss the model light curve. In addition, the ingress and egress portion of the light curve is high effected by limb darkening and a starspot signal during this time is very small. Thus, their is little benefit to including these data in the analysis.

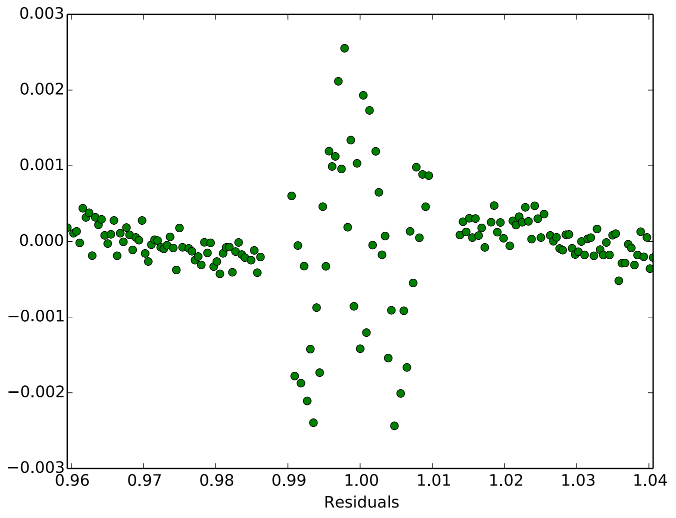


Figure 26: Residual points for Kepler object K00340.01 are calculated by taking the difference between the light curve and the model.

An overview of all the plots our program created is shown in Figure 27.

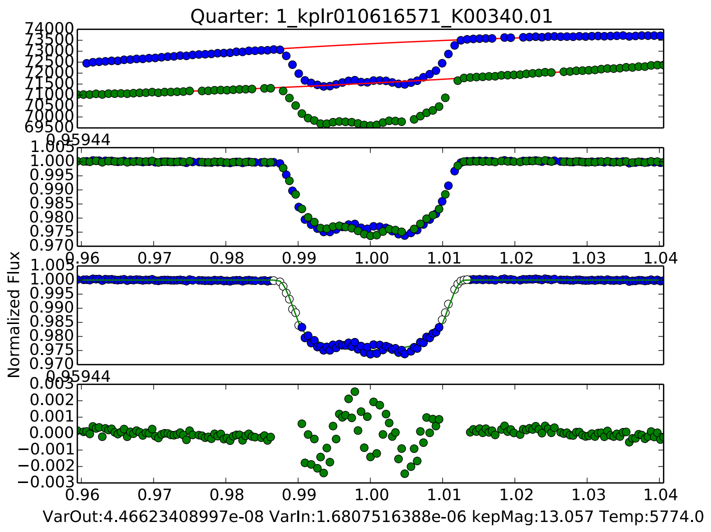


Figure 27: Written program takes the second degree polynomial fit to the out of transit region. The polynomial fit is subtracted from the overall light curve. The peak flux value is added then divided to the differential light curve. The model is generated and Residual points are calculated.

Flexible Image Transport System (FITS) files were created for all planet host stars that met our criteria. Data in the FITS files include residuals, properties obtained from the NASA Exoplanet archive catalogue, and flux, time and error values from the long cadence Kepler files. Each FITS file includes the following data for all available quarters of an planet host star system.

5 Results

The residuals generated at the end of our program were useful to us because we were able to use them in our statistical analysis. First we concatenated all residuals corresponding to its respective Kepler object. For example, Kepler object K00340.01 has x quarters so we concatenated x quarters worth of residuals and segmented the residuals into two samples: in-transit residuals and out-of-transit residuals. The residuals are a measure of how far away the raw data are from the theoretical model based on a uniformly bright star with no starspots. With this information we are able to look at flux modulations in-transit as well as out-of-transit and search for differences between them. We used several methods in order to compare the in-transit and out-of-transit residuals for all stars. We expect that stars that have starspots in the path of the planet will show higher varying in-transit residuals compared to the out-of transit residuals. Methods include generating a cumulative distribution function (CDF) to both samples, the Kolmogorov-Smirnov (KS) Test, the ratio of the variances, and the differences of the variances.

5.1 Statistics

5.1.1 Cumulative Distribution Function

The CDF is defined as:

$$F(x) = P(X \leq x) \tag{3}$$

Where F(x) gives the probability of being less than or equal to a given observed value x from the probability density function f(x). For instance, if I was looking for F(a), then it would yield the probability of being less than or equal to an observed value "a" from the probability density function f(x) [20].

$$F(a) = P(X \leq a) = \int_{x_0}^{a} f(x)dx \tag{4}$$

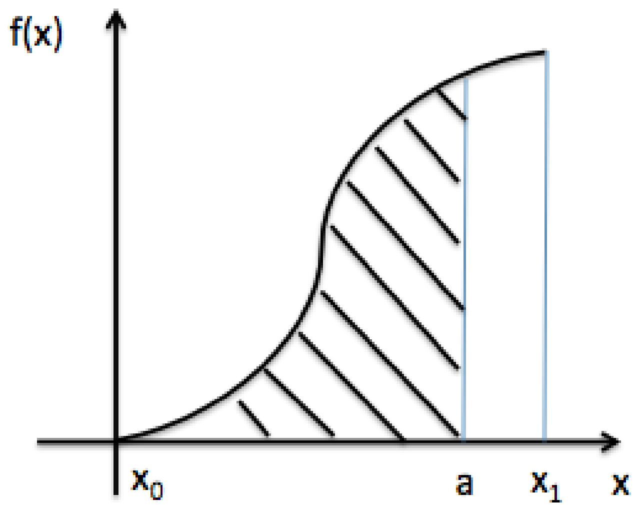
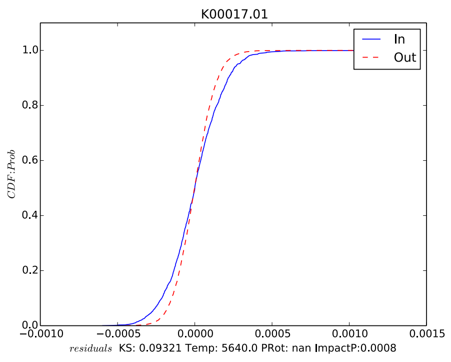
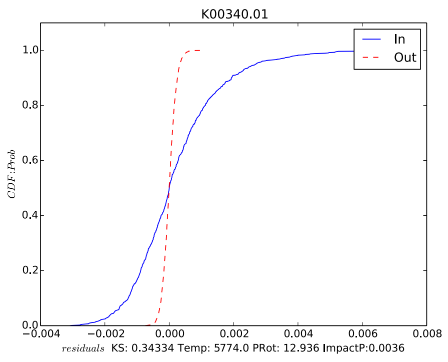


Figure 28: the CDF is the probability of being less than or equal to a given observed value "a" from the probability density function f(x)

We made graphs with residuals on the X axis and the probability of being less than or equal to an observed residual on the Y axis. Our generated CDF plots have two CDF curves over plotted against each other. The dashed curve corresponds to the out-of-transit residuals while the solid curve corresponds to the in-transit residuals. Figure 29 shows two sets of CDF plots. One plot is for K00017.01 and the other is for K00340.01.



(a) CDF curves for Kepler object K00017.01



(b) CDF curves for Kepler object K00340.01

Figure 29: Cumulative distribution curves for two Kepler objects.

The curves in Figure 29a are closer together compared to the curves in Figure 29b. This information reveals that for K00017.01, the deviation in the in-transit residuals are similar to that of out-of transit residuals. Figure 30 is a light curve plot for Kepler object for K00017.01 in quarter 3. We notice that the light curve does not show a large flux variation in-transit compared to out-of-transit.

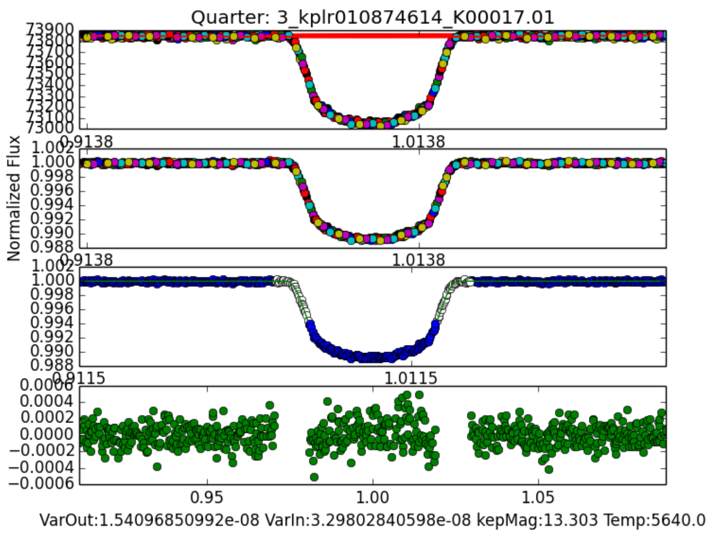


Figure 30: Four panel plots for Kepler object K00017.01. The residuals in-transit and out-of-transit are essentially the same which indicated that there is no starspot crossing feature in this light curve.

We cannot say the same for the CDF plot for K00340.1. In figure 29b, the CDF curves show a large separation. The large separation tells us that the deviation in the in-transit and out-of-transit residuals differ. Figure 27 shows the light curve for K00340.01 where we see that in-transit flux values and residuals have high variability and show large variation compared to the out-of-transit variation.

We realized that simply looking at CDF plots was not the ideal way to determine if a light curve displayed star spot crossing features. Sometimes Kepler objects displayed very little points in its light curve. Having very little points in the light curve biased the CDF plots. Sometimes, we would see separations in CDF plots but light curves did not display clear starspot crossing events. To quantify the difference between the in and out of transit CDF curves we use Kolmogorov-Smirnov (KS) Test.

5.1.2 Kolmogorov-Smirnov Test

The two sample Kolmogorov-Simirnov (KS) Test is a non-parametric test designed to test whether two samples of data come from same distribution. In our case our two samples are the in-transit residuals and the out-of-transit residuals. The KS test statistic is the maximum value between the CDF's of the two samples. Essentially, the KS test quantifies the maximum difference between the in-transit and out-of-transit CDF curves[21].

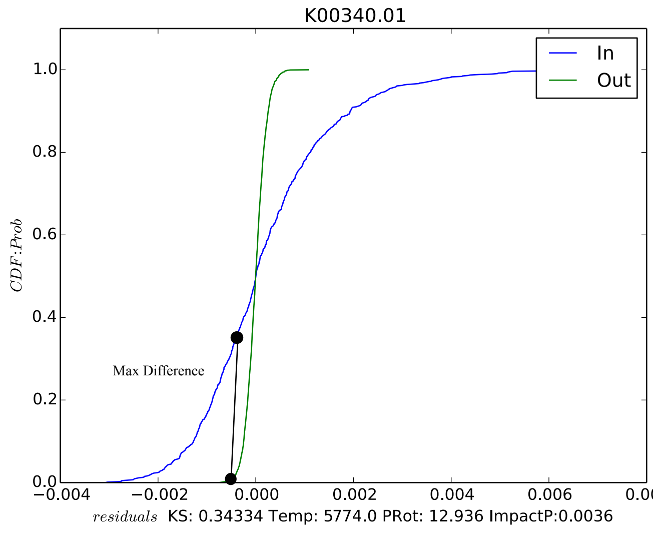


Figure 31: Example of CDF plot for Kepler object K00340.01. Black vertical line corresponds to the maximum difference between the green and blue curve, namely, the out-of transit and in-transit curves respectively.

The CDF curves demonstrated in Figure 31 correspond to the Kepler object K00340.01. The vertical line shows the maximum difference between the two CDF curves. The KS test calculates the actual number for this separation of the curves which is 0.3.

The KS test was applied to all Kepler objects that met our criteria. To do this we wrote a smaller program that takes all residuals for a specific Kepler object. We then fed the in-transit and out-of-transit residuals to a python function for a two sample KS test. We observed KS statistic values between 0 - 0.35.

Next, we looked at all light curves for every quarter for all Kepler objects by eye. We started from the Kepler objects that displayed high KS values and made our way down to objects that had low KS values. We noticed that most Kepler objects that displayed KS values greater than or equal to 0.12 appeared to have starspot crossing events multiple quarters. Table 1 shows all Kepler objects that have a KS values greater 0.12 and has clear starspot crossing features in multiple quarters.

Table 1: Objects With Starspot Crossing Features

Object	KS	Diff	Ratio	Prot (days)	Temp(K)
K01074.01	0.13407	4.5196e-07	2.7030	4.0925	6302.0
K00895.01	0.15299	5.2795e-07	3.5289	5.1448	5600.0
K01176.01	0.15733	1.7478e-06	4.0368	18.293	3806.0
K01353.01	0.16533	1.6187e-07	4.3919	8.6480	6279.0
K00806.02	0.16569	9.3152e-07	3.9685	16.004	5485.0
K00918.01	0.16852	4.1132e-07	4.6201	17.452	5552.0
K00676.01	0.18517	2.4080e-07	8.6638	12.290	3914.0
K00774.01	0.18716	9.4279e-07	2.5028	11.235	6108.0
K00217.01	0.19494	6.4174e-07	6.1088	20.461	5543.0
K00003.01	0.21721	2.3645e-08	23.446	29.472	5850.0
K00372.01	0.22047	8.9544e-08	8.4347	11.9	5838.0
K00883.01	0.22420	1.7588e-06	6.0798	9.1287	4809.0
K01784.01	0.23302	1.9027e-07	10.933	4.4940	5936.0
K01786.01	0.25997	1.9909e-06	2.2585	66.289	4461.0
K00203.01	0.28423	7.2625e-07	16.455	12.159	5624.0
K00063.01	0.28559	4.0582e-08	12.810	5.3985	5650.0
K00340.01	0.34334	1.8739e-06	47.107	12.936	5774.0

Although looking at the object by eye is not ideal, paired with the KS test statistic value, it gave us an idea for how to quantify objects with induced flux modulations in the light curve due to starspots. Therefore, the KS test was used as a major test to quantify the relationship between in-transit and out-of-transit flux modulation.

Why not the Anderson Darling Test?

Like the KS test, the Anderson Darling (AD) Test is a non-parametric test that is designed to test whether our two samples, in-transit and out-of-transit residuals, come from the same distribution. The AD test assumes that the data are normal and tests for any deviations that may suggest the data might not be normal. Compared to the KS test, the AD test is more powerful and is better able to detect differences in the two samples when sample sizes are small. Our sample sizes, however, are rather large. On average in-transit residuals have about 1,000- 5,000 points. When using large sample sizes the "sensitivity" of the AD test increases and the test returns extreme and implausible results. So when the sample size of our samples increase so does the sensitivity of the AD test. When the sensitivity increases any minute deviation from a normal bell curve will return and non-normal outcome. In other words, if in-transit and out-of-transit residuals have large sample sizes and show a normal bell curve , the AD test may return a non-normal result if it finds a significantly small deviation from the bell curve. If the sample sizes are large it is best to use your judgement instead of the Anderson Darling test for p value statistical significance [21].

5.1.3 Ratio of the Variances

The Ratio of the variances was also used to compare the residuals of our two samples. A ratio is an expression that compare quantities and compares the size of one sample to a second sample. The variance accounts for the dispersion or the variation of a data set relative to the mean of the same data set. The variance is defined as the following.

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1} \tag{5}$$

Where $\bar{x}$ is the mean of the sample x_i is the specified data point and N is the number of data points in the sample. It is key to keep in mind that the variance is always positive and a high variance suggests that the data are further away from each other and from the mean. When the variance is smaller it indicates that the data points are closer to one another and closer to the mean.

The ratio of the variances is exactly how it sounds. We calculated the variance of the in-transit and out-of-transit residuals for each star and divided the in-transit residuals by the out-of-transit residuals. The ratio of the variances of our two samples is defined as the following:

$$\sigma_{in}^2 = \frac{\sum(x_{in} - \bar{x}_{in})^2}{N_{in}-1} \tag{6}$$

$$\sigma_{out}^2 = \frac{\sum(x_{out} - \bar{x}_{out})^2}{N_{out}-1} \tag{7}$$

$$\frac{\sigma_{in}^2}{\sigma_{out}^2} = \frac{\sum(x_{in} - \bar{x}_{in})^2}{N_{in}-1} * \frac{N_{out}-1}{\sum(x_{out} - \bar{x}_{out})^2} \tag{8}$$

But what does of the ratio of the variance for these two samples mean? How do we make sense of this statistic? Suppose we have a light curve that is not affected by limb darkening as shown in the cartoon transit in Figure 32.

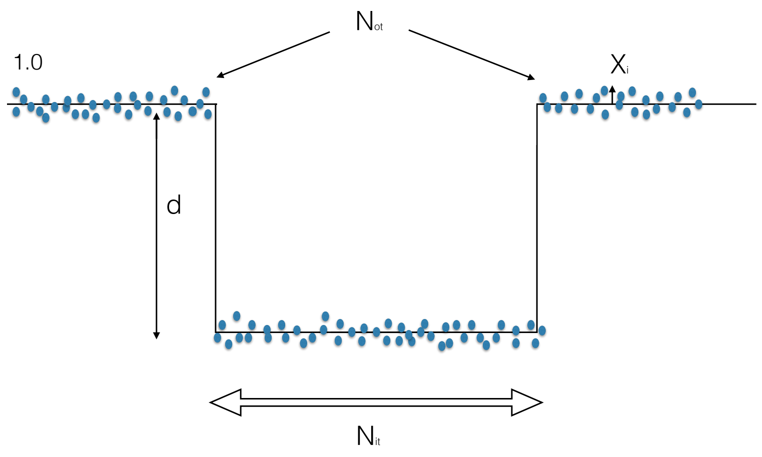


Figure 32: Transit cartoon with no starspot crossing event.

Where x_i is the offset from 1.0, d is the transit depth, and 1.0 is the mean of the out-of-transit light curve. Also, N_{it} corresponds to the number of points in the transit dip while N_{ot} corresponds to the number of out-of-transit data points. We have set the out-of-transit level to 1.0 to match out real normalized data. Using the cartoon in Figure 32, the variance of the out-of-transit region is defined as the following:

$$\sigma_{ot}^2 = \frac{\sum_{i=1}^{N_{ot}}((1.0 + x_i) - 1.0)^2}{N_{ot} - 1} = \frac{\sum_{i=1}^{N_{ot}} x_i^2}{N_{ot} - 1} \tag{9}$$

Since the mean of the out-of transit light curve is 1.0, the data point is defined at $1.0 + x_i$ where x_i is the offset from 1.0. The variance in-transit is then defined as the following:

$$\sigma_{it}^2 = \frac{\sum_{i=1}^{N_{it}}((t_i + x_i) - m_i)^2}{N_{it} - 1} = \frac{\sum_{i=1}^{N_{it}} x_i^2}{N_{it} - 1} \tag{10}$$

Where t_i is the underlying shape of the model, x_i is the offset from the underlying shape, m_i is the model in-transit and N_{it} is the number of points in transit. In the idealized case, the underlying shape of the dip and the model are equal. We can assume that the variance in-transit and out-of transit are equal because the scatter of points out of transit and in transit are not being affected by an extra factor and all variation is due to instrumental error which is constant for both samples.

Suppose we have a light curve that is not affected by limb darkening and has a positive bump in the light curve dip due to a star spot crossing event as shown in Figure 33.

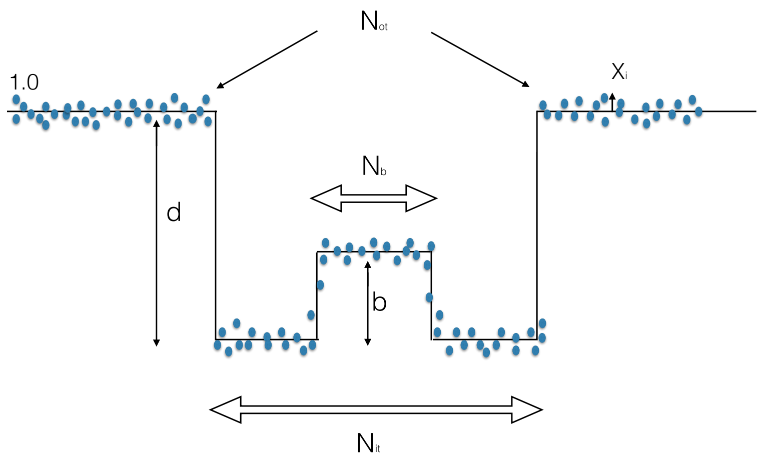


Figure 33: Transit cartoon with starspot crossing event.

Here, N_{it} corresponds to the points of the transit dip while N_b is the number of points in the bump located in the transit dip and N_{ot} still corresponds to the remaining number of data points or the out-of-transit data points. The variance out-of-transit will remain unchanged. Taking into account the starspot crossing event in-transit the variance in-transit is defined as the following:

$$\sigma_{it}^2 = \frac{\sum_{i=1}^{N_{it}-N_b}((t_i+x_i)-m_i)^2 + \sum_{i=1}^{N_b}((t_i+b+x_i)-m_i)^2}{N_{it}-1} \tag{11}$$

Where b is the height of the bump and every other variable is defined the same as in the generalized variance with no starspot crossing event. In this modified version we define the variance as the sum of the variance in the bump plus the variance of the transit dip excluding the bump.

Simplifying we get:

$$\sigma_{it}^2 = \frac{\sum_{i=1}^{N_{it}-N_b} x_i^2 + \sum_{i=1}^{N_b}(b+x_i)^2}{N_{it}-1} \tag{12}$$

$$\sigma_{it}^2 = \frac{\sum_{i=1}^{N_{it}-N_b} x_i^2 + \sum_{i=1}^{N_b}(b^2+2bx_i+x_i^2)}{N_{it}-1} \tag{13}$$

$$\sigma_{it}^2 = \frac{\sum_{i=1}^{N_{it}} x_i^2}{N_{it}-1} + \frac{\sum_{i=1}^{N_b}(b^2+2bx_i)}{N_{it}-1} \tag{14}$$

$$\sigma_{it}^2 = \sigma_{ot}^2 + P \tag{15}$$

Where $P = \frac{\sum_{i=1}^{N_b}(b^2+2bx_i)}{N_{it}-1}$ and P is a function of the bump height. Now that we have defined the variance for this example, we can say that the ratio of the variance is the following:

$$\frac{\sigma_{it}^2}{\sigma_{ot}^2} = \frac{\sigma_{ot}^2 + P}{\sigma_{ot}^2} \tag{16}$$

$$\frac{\sigma_{it}^2}{\sigma_{ot}^2} = 1 + \frac{P}{\sigma_{ot}^2} \tag{17}$$

How does the variation in-transit compare to the variation out-of-transit?

P/σ^2_{ot} is the added variation in-transit due to starspots compared to out-of transit variation. If no starspots then $P = 0.0$ and $\frac{\sigma^2_{it}}{\sigma^2_{ot}} = 1$. If there are starspots, then $P>0$ and $\frac{\sigma^2_{it}}{\sigma^2_{ot}}>1$.Remember P is a function of b or the bump height so the bigger the bump height gets the ratio becomes much greater than 1.0. We might even say that P/σ^2_{ot} acts like a signal to noise ratio. Practically, we found that ratio values ≥ 2.5 indicates starspot crossing events in the light curve.

5.1.4 Difference of the Variance

We also examined the difference of the variances. Assuming a light curve with flux modulations in transit due to a starspot crossing event, the difference of the variances is defined as the following:

$$\sigma^2_{it} - \sigma^2_{ot} = \left(\frac{\sum_{i=1}^{N_{it}} x_i^2}{N_{it}-1} + \frac{\sum_{i=1}^{N_b}(b^2 + 2bx_i)}{N_{it}-1}\right) - \frac{\sum_{i=1}^{N_{it}} x_i^2}{N_{it}-1} \tag{18}$$

$$\sigma^2_{it} - \sigma^2_{ot} = \frac{\sum_{i=1}^{N_b}(b^2 + 2bx_i)}{N_{it}-1} \tag{19}$$

$$\sigma^2_{it} - \sigma^2_{ot} = P \tag{20}$$

Remember, our assumption is that when there are no starspot crossing features in the light curve the dispersion of our two samples are equal. In this case, the difference of the variances is zero. If there is a starspot crossing feature in the light curve, we can say that the variance in-transit is equal to the variance out-of-transit plus an extra dispersion in the data due to the starspot crossing feature. When we subtract the out-of-transit variance from the in-transit variance we are left with a positive value that corresponds to the added scatter in-transit due tot he starspot.

So what is the difference between the ratio of the variances and the difference of the variances? The ratio of the variance is an indication of the amplitude of the starspot bump with respect to the typical scatter in the light curve. The ratio is a signal to noise measurement whereas the difference of the variance is more indicative of the absolute bump amplitude.

We are actually exploring what values of the difference of the variances presents a significant detection of a starspot crossing event. We have thought about this statistical formula for a while and decided to share even through we have not made any correlations between this statistic and properties of the star.

5.2 Global Plots

We plotted our statistics against the rotation period and stellar effective temperature of the stars in our sample. Figure 34 shows rotation period, in days,on the X-axis and the KS test statistic on the Y-axis. Recall, the KS test statistic quantifies the maximum difference between in- and out- of- transit CDF curves (see Figure 31). The larger the KS test statistic, the more in and out of transit regions differ in variation. By looking at all of our samples by eye, we realized that transit light curves that display starspot-crossing event generally have a KS value of 0.12 or greater.

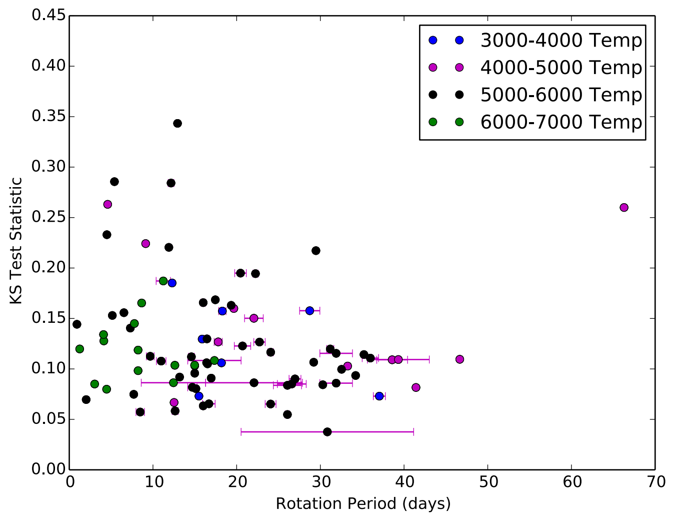
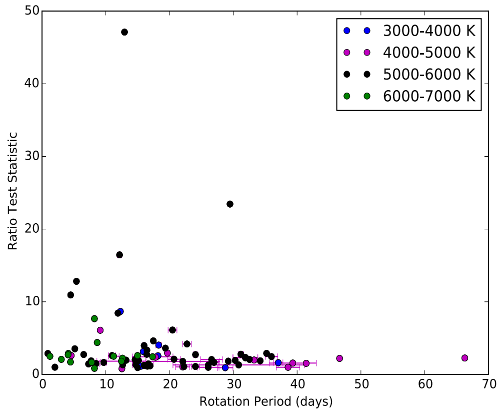


Figure 34: Plot illustrating rotation period on the X-axis and KS Value on the Y-axis. As the rotating period of stars increase, the less likely they are to have starspot crossing events.

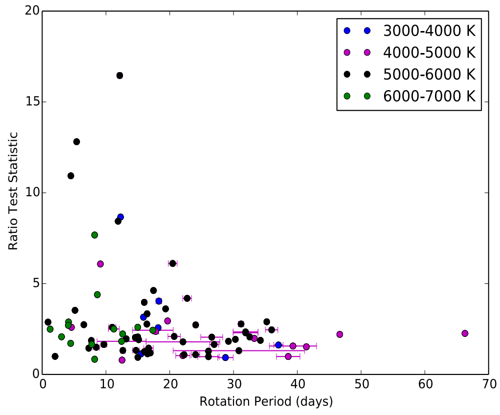
Figure 34 shows a general trend and that is that fast rotating stars have a range of KS values that are greater than and less than our significant value of 0.12 whereas slower rotating stars are confined to KS values that are merely less that 0.12. With the exception of one outlier, as the rotation period increases stars are less likely to exhibit starspots. This is consistent with studies of x-ray luminosity and hydrogen alpha which are tracers of magnetic activity. Results using these methods show that cooler and faster rotating stars have more magnetic activity and therefore more starspots.

Figure 34 also shows hot (6000-7000K) and fast rotating stars(0 -15 days) that have KS values less than 0.12. We can interpret what we are seeing in different ways. Perhaps starspots are not uniformly distributed on stars and planets are not crossing over the preferable latitudes of starspots for some stars. Alternatively, hot stars have smaller convective zones which hinders the motion of plasma and limits the amplification of the magnetic field. Therefore, starspots are less likely to appear on the photosphere of these stars. Both scenarios would produce a subset of rapidly rotating stars with low KS values as we see in Figure 34.

Figure 35 is a plot we made with rotation period, in days, on the X-axis and the ratio statistic on Y-axis. Recall, the ratio is the variation in-transit compared to out-of-transit. Stars with ratio values greater than 2.5 are indicative of starspot crossing features in light curves. At large we don't see any trends with this plot because of the two very large ratio values that do not allow us to see the majority of the other points. If we zoom in to stars that have ratio values of 0.0 - 20.0 we notice that this plot looks similar to the rotation period and KS statistic plot (see figure 34). We notice that as rotation periods increase, the ratio values decrease to approximately 1.0. These data are consistent with the notion that as rotation periods increases, stars are less likely to have starspots.



(a)



(b)

Figure 35: (a) 85 Kepler objects are compared with their rotation period and ratio statsitic. (b) Zoomed in version of (a) that will allow us to see that as the rotation of stars increase, they are less likely to have starspots.

Although we see a trend in KS value and rotation period in Figure 34, we do not see an immediate trend between KS value and stellar effective temperature. Figure 36 shows a plot of stellar effective temperature, in kelvin, on the X-axis and the KS value on the Y-axis.

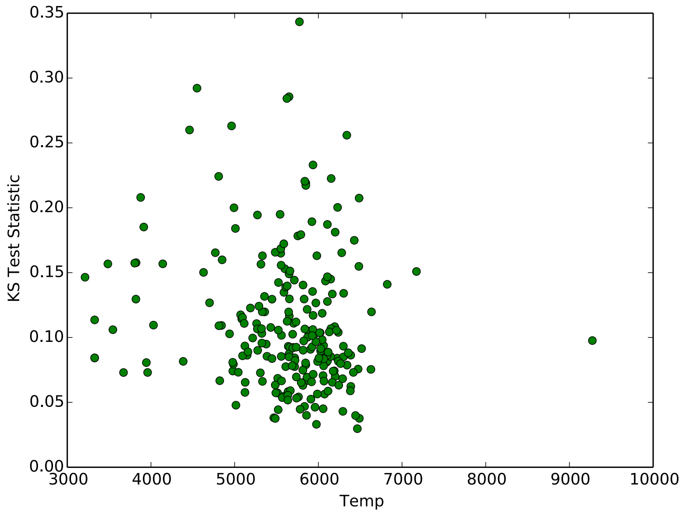
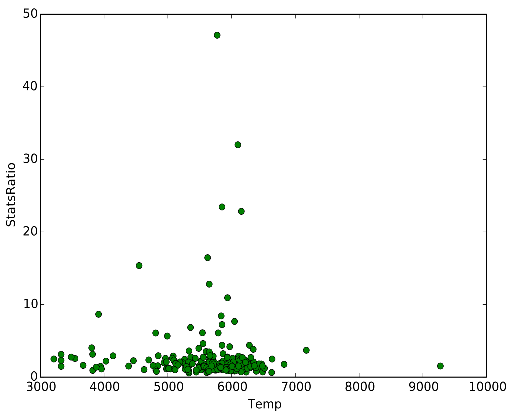


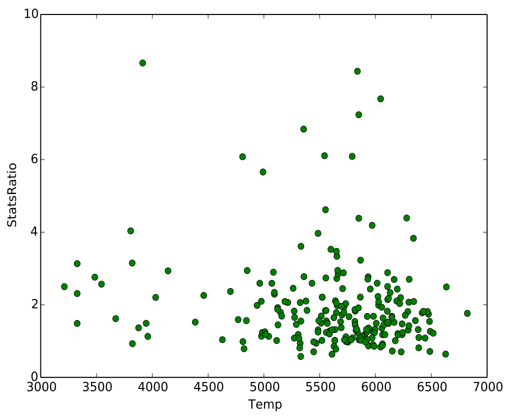
Figure 36: Displays the overall global plot for the temperature, in elvin, and the KS statistic for all 249 object in our sample.

We notice that there is a big clump of points between 5000- 7000 K. This is expected because most stars that are monitored by Kepler are Sun-like stars and the temperature of the Sun is approximately 5700 K. Strictly looking at Figure 36, we also do not see an immediate trend, but because eyes can be deceiving, we plotted an imaginary horizontal line on KS value of 0.12 and divided the stars as less than and greater than 5000K. When doing this, we calculated that the percentage of stars with temperatures less than 5000 K and a KS value greater than 0.12 is 51.5 % and the percentage of stars greater than 5000 K with a KS value greater than 0.12 is 27.7%. This result suggests that cool stars are more likely to have high KS values compared to hotter stars.

Figure 37 shows a plot of stellar effective temperature, in kelvin, on the X-axis and the ratio value on the Y-axis. Figure 37a shows the entire plot while Figure 37b illustrates a zoomed in version of Figure 37a.



(a)



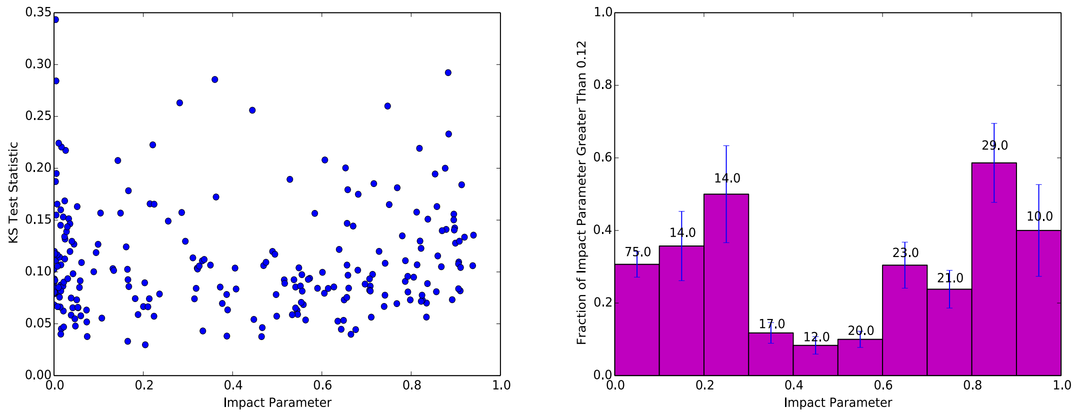
(b)

Figure 37: (a) Displays the overall global plot for the temperature, in kelvin, and ratio statistic for all 249 object in our sample. (b) Illustrates a zoomed in version of (a).

Similar to that of the temperature and KS result, we calculated that the percentage of stars with temperatures less than 5000 K and a ratio value greater than 2.5 is 39.4 % and the percentage of stars greater than 5000 K with a ratio value greater than 2.5 is 20.3%. This result also suggests that cool stars are more likely to have high ratio values compared to hotter stars.

We were also interested in comparing our statistics with the impact parameter. The impact parameter is a number ranging from 0 to 1 that tells us the latitude at which the planet is crossing in front of the star (see Figure 14). Scientists have made this quantization by assuming that both the angular momentum of the planet and the star are aligned. When we plotted the KS vs the impact parameter we did not really see a trend as shown by Figure 38a. We thought that we were sampling across every impact parameter equally, but to make sure of this we decided to bin the impact parameters into 10 equally sized bins.

Once we binned the impact parameter we calculated the fraction of KS values in the bins that are greater than 0.12. Next we binned these fraction, thus creating Figure 38b. Figure 38b shows the impact parameter on the X axis and the fraction of objects with KS value greater than 0.12 on the Y axis.



(a) (b)

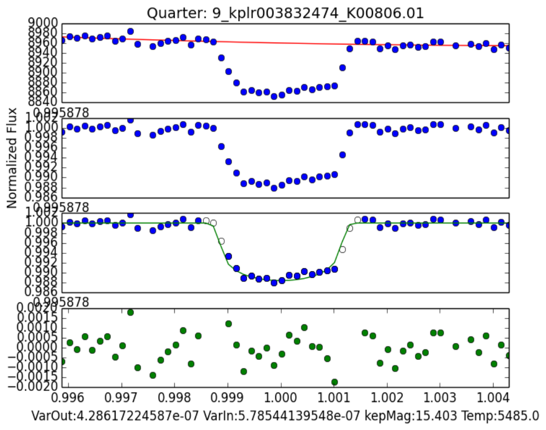
Figure 38

Figure 38b, suggests that starspots are mostly present near the equator and the poles of stars. This is interesting! The Butterfly diagram, which allows us to suggest the preferred latitudes of sunspots, says that sunspots are confined between 30 N and 30 S. We are not getting a consistent result here.

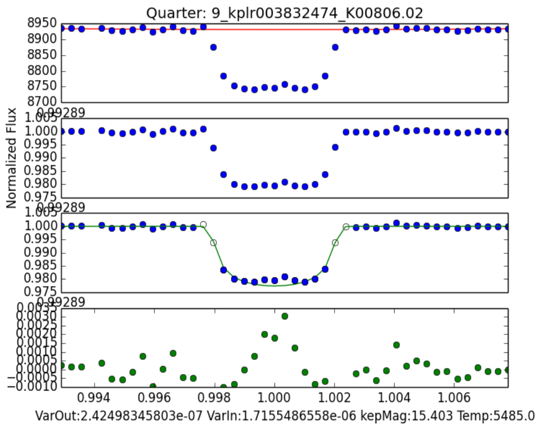
5.2.1 Interesting Individual Objects

Some of the transiting planet host star systems are multiplanet systems. We discovered a special multiplanet system in our sample. Kepler object K00806.01

has a KS value of 0.0634 and a ratio value of 1.2496 and we DO NOT see any star-spot crossing features in the light curves of this object. Kepler object K00806.02 has a KS value of 0.1657 and a ratio value of 3.9686 and we DO see some star-spot crossing features in the light curves of this object. light curves for both objects are shown for quarter 9 below in Figure 38:



(a)



(b)

Figure 39: Multiplanet system. (a) Kepler object K00806.01 shows no starspot crossing feature in-transit. (b) Kepler object K00806.02 shows starspot crossing features in transit. Both light curves correspond to quarter 9.

Using the well known position of the planet, we can use the impact parameter to distinguish where on the star these planets are crossing.It is exciting to know where on a star starspots are located because sunspots are confined between two latitudes on either side of the equator (30 N / 30 S). We hope to understand more about the latitudes at which starspots are found in order to compare to the Butterfly diagram (see Figure 5). Kepler object K00806.01 has an impact parameter of 0.3889 while Kepler object K00806.02 has an impact parameter of 0.2149. Now we know a bit more about this host star than ever before! One planet orbits at latitude 0.3889 and does not cross starspots while the other planet orbits at latitude 0.2149 and does crosses a starspots. This suggests that starspots are not uniformly distributed.

Another interesting object was object K01786.01. This object is probably orbiting the most magnetically active star in our sample. Figure 39 shows the light curve for K01786.01 in quarter 7 which shows a great indication of starstpot crossing features due to the increase in scatter in-transit. We also see these signature flares in 3 other quarters. Flares rise faster than they decay and are an indication of extremely strong magnetic fields.

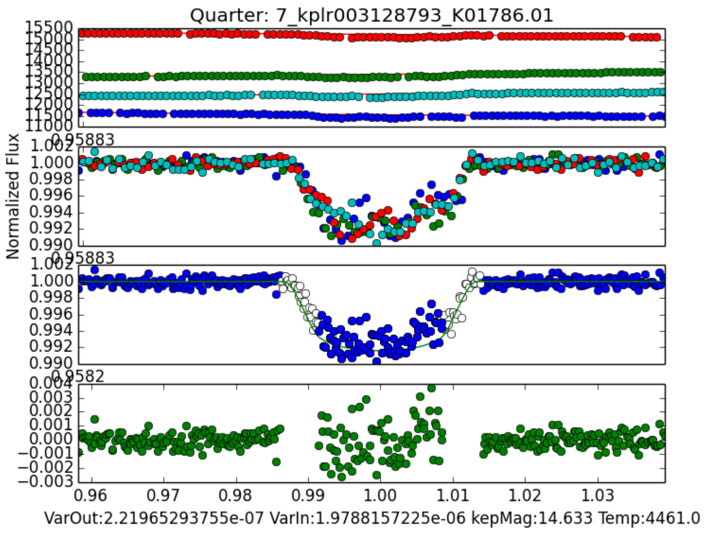


Figure 40: Four panel plot for Kepler object K01786.01 which displays various indication of starspot crossing features in every transit for quarter 7. This star has displayed various starspot flares in light curves and is considered a long period RS CVn star.

We further explored K01786.01 and found that the host star for K01886.01 is a Red Giant. Red giants have strong magnetic fields in their cores[22]. However, we are seeing starspot crossing features in the light curve meaning that we seeing manifestations of magnetic fields on the surface of the star. We investigated literature for any red giants with starspots and we think K01786.01 is like stars called long period RS Canum Venaticorum Variables (long period RS CVn). Other long period RS CVn stars are sigma Gem [23] and IM Pegasi [24]. Studies show that RS CVn stars have active longitudes as opposed to latitudes which we know usually have starspots. This is an interesting object because it can be be used to study active longitudes for future projects.

6 Conclusion and Future Directions

Our main goal is to study magnetic fields on stars other than our Sun to better understand the mechanisms for the solar dynamo. To do so, our novel technique is to use transiting planets as probes to map starspots on Kepler host stars. Starspots are visible manifestations of strong and local magnetic fields. As a planets crosses a starspot, there is a positive bump in the transit dip due to the dark appearance of the starspot on the stars photosphere. We have written a program that is useful in detecting starspot crossing events in a Kepler 246 object deep transit sample.

We were able to see trends in starspot variability and stellar properties. Residuals compared to the KS value and the ratio of the variances suggest that as the rotation period of stars increase, the less likely stars are to have starspots on their surface. Data also suggest that starspots prefer to form near the equator or poles of stars which is not what we expected considering that sunspots on the Sun are confined to 30N- 30S latitudes on the Sun. We hope to further explore our results. This projects has presented many surprising and interesting results whether it be globally or specific to an individual object.

In the future we wish to refine our sample by plotting the KS vs ratio values as illustrated by Figure 41 and removing the objects with high KS values and low ratio values. If we look at this figure carefully we see two directions as KS values increase. One subset of objects increase in ratio as KS increases. The other subset stay at low Ratio values as KS increases.

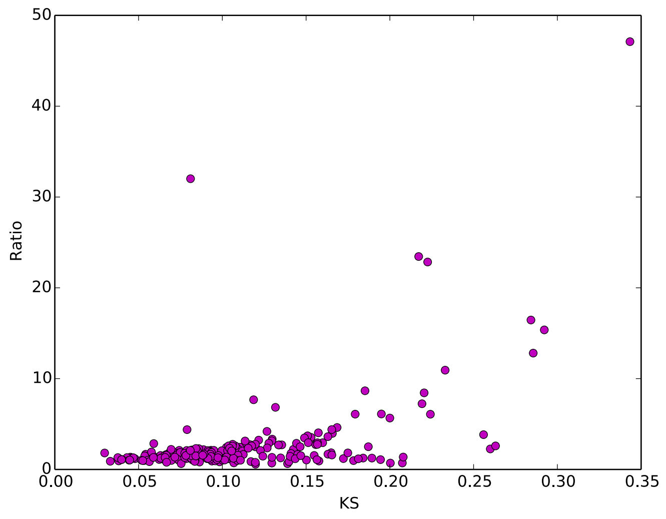


Figure 41: Plot Illustrating KS on the X-axis and the ratio of the variances on the Y-axis. As the KS value increases there is a subset of objects that increase in ratio value and a subset that decreases in ratio value. Preliminary thoughts suggest that objects with high KS values and low ratio values have too few points in light curves across all quarters.

We suspect that objects with low ratio and high KS values are those that have a small number point in each transit across all quarters. A small sample size affects the KS value greatly but we assume does not affect the Ratio statistic because the variation in- and out-of-transit, if no starspots are present, should be roughly the same no mater how many points we have in our comparative groups.

We also wish to compare our statistics to the rossby number instead of the rotation period of stars. The rossby number gives us the relation between activity, rotation, and spectral type [25]. The rossby number is defined as,

$$R_o = \frac{P_{rot}}{\tau_c} \tag{21}$$

where R_o is the rotation period of the star and τ_c is the convective turnover number $\tau_c = l/V$, where l is the mixing length and V is the convective velocity.

References

[1] M. Stix. *The sun : an introduction.* 2004.

[2] D. Griffiths. *Introduction to Elementary Particles.* 2008.

[3] J. Christensen-Dalsgaard, D. O. Gough, and M. J. Thompson. The depth of the solar convection zone. , 378:413–437, September 1991.

[4] T. V. Zaqarashvili, R. Oliver, A. Hanslmeier, M. Carbonell, J. L. Ballester, T. Gachechiladze, and I. G. Usoskin. Long-term Variation in the Suns Activity Caused by Magnetic Rossby Waves in the Tachocline. , 805:L14, June 2015.

[5] A. Strugarek, A. S. Brun, and J.-P. Zahn. Magnetic confinement of the solar tachocline: influence of turbulent convective motions. In N. H. Brummell, A. S. Brun, M. S. Miesch, and Y. Ponty, editors, *IAU Symposium*, volume 271 of *IAU Symposium*, pages 399–400, August 2011.

[6] H. W. Babcock. The Topology of the Sun's Magnetic Field and the 22-YEAR Cycle. , 133:572, March 1961.

[7] M. Kopecký. Babcock's Theory of the 22-year Solar Cycle and the Latitude Drift of the Sunspot Zone. *Advances in Astronomy and Astrophysics*, 7:57–81, 1970.

[8] K. S. Balasubramaniam. Physical Properties of Structures about Sunspots. In *American Astronomical Society Meeting Abstracts #204*, volume 36 of *Bulletin of the American Astronomical Society*, page 685, May 2004.

[9] C. Kiess, R. Rezaei, and W. Schmidt. Properties of sunspot umbrae observed in cycle 24. , 565:A52, May 2014.

[10] D. H. Hathaway. The Solar Cycle. *Living Reviews in Solar Physics*, 12:4, September 2015.

[11] D. J. Griffiths. *Introduction to quantum mechanics.* 1995.

[12] J. T. Hoeksema, Y. Liu, K. Hayashi, X. Sun, J. Schou, S. Couvidat, A. Norton, M. Bobra, R. Centeno, K. D. Leka, G. Barnes, and M. Turmon. The Helioseismic and Magnetic Imager (HMI) Vector Magnetic Field Pipeline: Overview and Performance. , 289:3483–3530, September 2014.

[13] D. Koch, W. Borucki, L. Webster, J. Jenkins, E. Dunham, S. Jordan, and Kepler Team. The Kepler Mission: A Search for Terrestrial Planets. In E. R. Schielicke, editor, *Astronomische Gesellschaft Meeting Abstracts*, volume 18 of *Astronomische Gesellschaft Meeting Abstracts*, page 406, 2001.

[14] C. A. Haswell. *Transiting Exoplanets.* July 2010.

[15] L. M. Walkowicz and G. S. Basri. Rotation periods, variability properties and ages for Kepler exoplanet candidate host stars. , 436:1883–1895, December 2013.

[16] M. B. Nielsen, L. Gizon, H. Schunker, and C. Karoff. Rotation periods of 12 000 main-sequence Kepler stars: Dependence on stellar spectral type and comparison with v sin i observations. , 557:L10, September 2013.

[17] T. Reinhold, A. Reiners, and G. Basri. Rotation differential rotation of the active Kepler stars. In P. Petit, M. Jardine, and H. C. Spruit, editors, *IAU Symposium*, volume 302 of *IAU Symposium*, pages 216–219, August 2014.

[18] A. McQuillan, T. Mazeh, and S. Aigrain. Stellar Rotation Periods of the Kepler Objects of Interest: A Dearth of Close-in Planets around Fast Rotators. , 775:L11, September 2013.

[19] A. McQuillan, T. Mazeh, and S. Aigrain. Rotation Periods of 34,030 Kepler Main-sequence Stars: The Full Autocorrelation Sample. , 211:24, April 2014.

[20] P. R. Bevington and D. K. Robinson. *Data reduction and error analysis for the physical sciences.* 2003.

[21] S. A. Teukolsky, W. H. Press, B. P. Flannery, C. Lloyd, and P. Rees. Book Review: Numerical recipes example book (C) / Cambridge U Press, 1993. *The Observatory*, 113:214, August 1993.

[22] J. Fuller, M. Cantiello, D. Stello, R. A. Garcia, and L. Bildsten. Asteroseismology can reveal strong internal magnetic fields in red giant stars. *Science*, 350:423–426, October 2015.

[23] P. Kajatkari, T. Hackman, L. Jetsu, J. Lehtinen, and G. W. Henry. Spot activity of the RS Canum Venaticorum star σ Geminorum. , 562:A107, February 2014.

[24] S. V. Berdyugina, I. Ilyin, and I. Tuominen. The long-period RS Canum Venaticorum binary IM Pegasi. I. Orbital and stellar parameters. , 347:932–936, July 1999.

[25] E. L. Schatzman, F. Praderie, and A. R. King. *The Stars.* 1993.

7 Appendix

7.1 Python Code

```
#import necessary modules
import csv
import fnmatch
import os
import sys
import scipy
from scipy import interpolate
import numpy as np
import pyfits
import matplotlib.pyplot as plt
import iancross_transit as transit
import iancross_analysis as an
from sys import argv
from scipy.interpolate import spline
import math
import pdb
from astropy.io import fits
import eastman_occultquad as ecode
import robust as ropoly
from scipy import stats

sys.argv                          #allows for command line arguments

def makeModel(epoch, period, inc_deg, ars, ecc,longperi,gamma, p0, tdiff,
    phaseSmooth, AllModPhase):
    tSmooth=(phaseSmooth)*period #generates new times from made up phases
    modPhaseSmooth=phaseSmooth-np.floor(phaseSmooth) #generates modphases from
        made up phases

    for index in range(len(modPhaseSmooth)):
        if modPhaseSmooth[index] < 0.5:
            modPhaseSmooth[index]=modPhaseSmooth[index]+1

    zSmooth=transit.t2z(epoch, period, inc_deg, tSmooth, ars, ecc, longperi)
        #generates z values from made up phases
    #modelFluxSmooth=transit.occultquad(zSmooth,p0,gamma,retall=False,
        verbose=False) #generates model flux values from made up phases

    modelFluxSmooth = ecode.occultquad(zSmooth, gamma[0], gamma[1], p0)[0] #
        jason eastman occultquad
        #gets rid of dip when planet is behind star
    for index in range(len(tSmooth)):
        n=math.cos(2*np.pi*(tSmooth[index])/period)
        if n<0:
            modelFluxSmooth[index]=1

    dtSmooth=tSmooth[1]-tSmooth[0]      #diference between two consecutive
        elemnts in Tsmooth
    kernelSize=np.round(tdiff/dtSmooth)      #calculates how many elements make
        up 29.4 minutes

    #creates a conv. kernel the length of kenel size(divide by the kernel size bc
        of scale)
    Kernel=np.ones(kernelSize)/float(kernelSize)

    #interpolation between made up phases and modelfluxes  to create an
```

```
        interpolated array of original phase size ( size from NASA file)
    InterpModel=interpolate.interp1d(modPhaseSmooth,modelFluxSmooth,kind='linear
        ',bounds_error= False)
    newModelFlux=InterpModel(AllModPhase)


    #limP=np.polyfit(newModelFlux, AllFlux,1) # linear fit between the model and
        the flux
    newModelFlux=np.array(newModelFlux)
    #modelFluxOriginal=modelFluxSmooth

    #convolves the conv. kernel with the model data
    ConvolvedModel=np.convolve(Kernel,modelFluxSmooth, mode='same')

    #gets rid of bad convolved points at both ends
    ConvolvedModel[0:10]=1
    ConvolvedModel[len(ConvolvedModel)-10:len(ConvolvedModel)]=1


    #"fitting for the transit depth"
    InterpModel=interpolate.interp1d(modPhaseSmooth,ConvolvedModel,kind='linear',
        bounds_error= False)

    modelFlux_=InterpModel(AllModPhase)
    return modelFlux_


def doParts(phase, durPhaseHalf, durPhase, zplusp0,firstTransit):
    flag=False
    #parts=[-1,-1,-1,-1,len(flux)]    #parts of dips
    parts=[-1,-1,-1,-1,-1]
    StartSection=0
    for index in range(len(phase)):
        Start=firstTransit-(durPhaseHalf)-durPhase       #left of transit
        transitStart=firstTransit-durPhaseHalf #transit starts in phase
        transitEnd=firstTransit+durPhaseHalf    #transit ends in phase
        End=firstTransit+(durPhaseHalf)+durPhase         #right of transit
        if phase[index]<=Start:
            StartSection=index

        if phase[index]>Start and phase[index]<=transitStart:
            #left of transit
            parts[0]=index
        if phase[index]> transitStart and phase[index]<=firstTransit:
            if zplusp0[index]>1.:
                #ingress
                parts[1]=index

        if phase[index]> transitStart and phase[index]< transitEnd :
            if zplusp0[index]<=1.:
                #fully in transit
                parts[2]=index
        if phase[index]>=firstTransit and phase[index]< transitEnd :
            #if zplusp0[index]>=1:
                #egress
            parts[3]=index
        if phase[index]>=transitEnd and phase[index]<=End:
            #right of transit
            parts[4]=index


    if parts[0]==-1:
        flag=True
    if parts[2]==-1:
        flag=True
    if parts[4]==-1:
        flag=True
    return (parts,StartSection,flag)


def delete_by_indices(lst, indices):
    indices_as_set = set(indices)
    return [ lst[i] for i in xrange(len(lst)) if i not in indices_as_set ]


#function that finds the first specific character in a string
def find(str, char):
    for ltr in str:
        if ltr!= char:
            return str[str.index(ltr):]

#function that removes "nan" values in an array
def removeNan(array):
    for index in range(len(array)):
        if str(flux[index]) =='nan':
            array=np.delete(array,index)
    return array

#function that calculates the chi2 value
def chi2(obs, expected, error):
    data=[]
    for index in range(len(obs)):
        data.append((obs[index]-expected[index])**2/error[index]**2)
    return sum(data)

#calculates the mmedian absolute deviation

def mad(array):
    n = float(len(array))
    med = np.median(array)
    diff = [ abs(x - med) for x in array ]
    trueMed= np.median(diff)
    Val=1.4826*trueMed
    return Val

def flagArray(int): #expandinf quality values in array of ones and zeros.
    num=int
    flag=[0]*17
    for index in range(0,17):
        remainder= num % 2
        flag[index]=remainder
        num=(num-remainder)/2
    return flag

def findOne(array): #for index with values of one sum 2^index
    num=0
    for index in range(len(array)):
        if array[index]==1:
            numVal= 2**index
            num= num+numVal
        if num == 128 or num== 8192:
            return True
    return False

def Pop(array1, array2):
    array2= array2.tolist()
    arrayTrue=[]
    for index in range(len( array1)):
        if array1[index]== False:
            arrayTrue.append(array2[index])
    return arrayTrue

def remove_duplicates(l):
    return list(set(l))


if __name__=='__main__':    # seperates page between function half and main
    program
    out=open("confirmedPlanetsTable_15_6_26.csv", "rU")     #opening file from
        NASA exoplanet archive site
    data=csv.reader(out)                #reading file
    data=[row for row in data]
    data.pop(0)                         #removing first row of file (these are
        strings)

    #arrays of all different parameters
    kepid=[]
    periodstr=[]
    epochstr=[]
    impactParameterstr=[]
    incstr=[]
    arsstr=[]
    eccstr=[]
    longperistr=[]
    p0str=[]
    transitDurationstr=[]
    LimDark=[]
    coef1str=[]
    coef2str=[]
    coef3str=[]
    coef4str=[]
    temp=[]
    planetName=[]
    rMagstr=[]
    iMagstr=[]
    kepMagstr=[]

    #insert elements of NASA exoplanet archive table into correct array (all are
        strings)
    for col in data:
        kepid.append(col[0]) #keplerID
        periodstr.append(col[1]) #period
        epochstr.append(col[4]) #epoch
        impactParameterstr.append(col[10]) #impact parameter
        incstr.append(col[13])  #inclination angle (radians)
        arsstr.append(col[25]) #orbital semimajor axix/star radius
        eccstr.append(col[19]) #orbital eccentricity
        longperistr.append(col[22]) #longitude of periapse (degrees)
        p0str.append(col[28]) ##planet/star radius ratio
        transitDurationstr.append(col[7]) #transit duration
        # 4 limbdarkening coeff.
        coef1str.append(col[34])
        coef2str.append(col[35])
        #coef3str.append(col[36])
        #coef4str.append(col[37])
        temp.append(col[31]) #stellar temp
        planetName.append(col[36])
        kepMagstr.append(col[37])
        rMagstr.append(col[38])
        iMagstr.append(col[39])

    #loop that generates lists(each with four elements) of limbdark coeff. within
        a bigger list
    for index in range(len(kepid)):
        allcoef=[]
        allcoef.append((coef1str[index]))
        allcoef.append((coef2str[index]))
        #allcoef.append(float(coef3str[index]))
        #allcoef.append(float(coef4str[index]))
        LimDark.append(allcoef)

    period_=[ float(x) for x  in periodstr if x !='']
    epoch_=[ float(x) for x  in epochstr if x !='']
    impactParameter_=[ float(x) for x  in impactParameterstr if x !='']
    inc_=[ float(x) for x  in incstr if x !='']
    ars_=[ float(x) for x  in arsstr if x !='']
    ecc_=[ float(x) for x  in eccstr if x !='']
    longperi_=[ float(x) for x  in longperistr if x !='']
    p0_=[ float(x) for x  in p0str if x !='']
    TransDuration_=[ float(x) for x  in transitDurationstr if x !='']
    Temp_=[ float(x) for x  in temp if x !='']
    rMag_=[ float(x) for x  in rMagstr if x !='']
    iMag_=[ float(x) for x  in iMagstr if x !='']
    kepMag_=[ float(x) for x  in kepMagstr if x !='']

    gamma_=[]
    for index in range(len(LimDark)):
        sep=[ float(x) for x  in LimDark[index] if x !='']
        gamma_.append(sep)

    paperN=pyfits.open('rotation_per_Nielsen2013.fit')   #open file in totallist
    print paperN
    dataFileN=paperN[1].data
    dataCubeN=dataFileN.columns
    rotation_per_Nielsen2013=dataFileN.Prot
    KIC_Nielsen2013=dataFileN.KIC

    paperR=pyfits.open('rotation_per_Reinhold2013.fit')   #open file in totallist
    print paperR
    dataFileR=paperR[1].data
    dataCubeR=dataFileR.columns
    rotation_per_Reinhold2013=dataFileR.P1
    KIC_Reinhold2013=dataFileR.KIC

    paperMQ=pyfits.open('rotation_per_McQuillan2013.fit')   #open file in
        totallist
    print paperMQ
    dataFileMQ=paperMQ[1].data
    dataCubeMQ=dataFileMQ.columns
    rotation_per_McQuillan2013=dataFileMQ.Prot
    KIC_McQuillan2013=dataFileMQ.KIC

    paperMQ2=pyfits.open('rotation_per_McQuillan2014.fit')   #open file in
        totallist
    print paperMQ2
    dataFileMQ2=paperMQ2[1].data
    dataCubeMQ2=dataFileMQ.columns
    rotation_per_McQuillan2014=dataFileMQ2.Prot
    KIC_McQuillan2014=dataFileMQ2.KIC

    paperW=pyfits.open('rotation_per_Walkowicz2014.fit')   #open file in
        totallist
    print paperW
    dataFileW=paperW[1].data
    dataCubeW=dataFileW.columns
    rotation_per_Walkowicz2014=dataFileW.Per
    KIC_Walkowicz2014=dataFileW.KIC

    listOfList = []
    for i in kepid:
        list=[]
        i=int(i)
        if i  in KIC_McQuillan2013:
            inde = [y for y, x in enumerate(KIC_McQuillan2013) if x == i]
            if len(inde)>0:
                inde=inde[0]
                #print i, inde, rotation_per_McQuillan2013[inde]
                list.append(rotation_per_McQuillan2013[inde])
        if i  in KIC_McQuillan2014:
            inde = [y for y, x in enumerate(KIC_McQuillan2014) if x == i]
            if len(inde)>0:
                inde=inde[0]
                #print i, inde, rotation_per_McQuillan2013[inde]
                list.append(rotation_per_McQuillan2014[inde])
        if i  in KIC_Nielsen2013:
            inde = [y for y, x in enumerate(KIC_Nielsen2013) if x == i]
            if len(inde)>0:
                inde=inde[0]
                #print i, inde, rotation_per_Nielsen2013[inde]
                list.append(rotation_per_Nielsen2013[inde])
        if i  in KIC_Walkowicz2014:
            inde = [y for y, x in enumerate(KIC_Walkowicz2014) if x == i]
            if len(inde)>0:
                inde=inde[0]
                #print i, inde, rotation_per_Walkowicz2014[inde]
                list.append(rotation_per_Walkowicz2014[inde])
        if i  in KIC_Reinhold2013:
            inde = [y for y, x in enumerate(KIC_Reinhold2013) if x == i]
            if len(inde)>0:
                inde=inde[0]
                #print i, inde, rotation_per_Reinhold2013[inde]
                list.append(rotation_per_Reinhold2013[inde])
        listOfList.append(list)

    medianArray=[]
    rangeError=[]
    protErrorMinVal=[]
    for index in range(len(listOfList)):
        if len(listOfList[index]) == 0:
            median1= np.median(listOfList[index])
            medianArray.append(median1)
            rangeVal= float('nan')
            rangeError.append(rangeVal)
            protErrorMinVal.append(float('nan'))
        else:
            median1= np.median(listOfList[index])
            medianArray.append(median1)
            minVal= np.min(listOfList[index])
            maxVal= np.max(listOfList[index])
            rangeVal= maxVal- minVal
            rangeError.append(rangeVal)
            medianMin= median1-minVal
            medianMax= maxVal-median1
            if medianMin <= medianMax:
                protErrorMinVal.append(medianMin)
            else:
                protErrorMinVal.append(medianMax)

    Allcount=0 #all planet candidates shared between kepID list and downloaded
        files
    grazzing =0 # All planets that are skipped bc grazzing
    transitZero=0 # All planets skipped because there are no transits

    KEP_ID=[]
    KOI_ID=[]
    P0=[]
    EPOCH=[]
    PERIOD=[]
    IMPACTP=[]
    INCLINATION=[]
    ARS=[]
    ECC=[]
    LONGPERI=[]
    TEMP=[]
    LIMBDARK=[]
    Var_IN=[]
```

```
        Var_OUT=[]
        MAD_IN=[]
        MAD_OUT=[]
        NUMPTS_IN=[]
        NUMPTS_OUT=[]
        KEPMAG=[]
        RMAG=[]
        IMAG=[]
        ROTATIONPER=[]
        ROTATIONPER_ERROR=[]
        ROTATIONPER_ERROR2=[]
        VRANGE_OUT=[]
        VRANGE_IN=[]
        VRANGE_ALL=[]
        SHIFT=[]
        Q=[]
        KS=[]
        COMMENTS=[]
        ERRMEAN=[]

        for object in planetName:
            TIME=[]
            PHASE=[]
            FLUX=[]
            ERROR=[]
            FLAT_FLUX=[]
            FLAT_PHASE=[]
            FLAT_QUARTER=[]
            RESIDUALS=[]
            FLAT_ERROR=[]
            QUARTER=[]
            TRANSIT_FLAG=[]
            VRANGE=[]
            Name='blah'
            print object
            for i in range(len(planetName)):
                if object == planetName[i]:
                    print planetName[i], object
                    planetname=planetName[i]
                    planetNameNoK=find(planetName[i],'K')
                    p0=p0_[i]#planet/star radius ratio
                    epoch=epoch_[i]#system epoch
                    period=period_[i] #the period (days)
                    impactParameter=impactParameter_[i]  #impact parameter
                    inc=inc_[i]  #inclination angle
                    ars=ars_[i]  #Orbital semimajor axix/star radius
                    inc1=math.acos(impactParameter/ars) #inclination angle in degrees
                    inc_deg=math.degrees(inc1)
                    ecc=ecc_[i] #orbital eccentricity
                    longperi=longperi_[i] #longitude of periapse (radians)
                    TransDuration=TransDuration_[i] #transit duration (hrs)
                    Temp=Temp_[i]   #Stellar temp
                    print type(Temp), Temp, 'Temp'
                    gamma=gamma_[i] #LimbDarkening coeff
                    kepMag=kepMag_[i]
                    rMag=rMag_[i]
                    iMag=iMag_[i]
                    Kepid=kepid[i]
                    print Kepid
                    listOfListProt=listOfList[i]
                    medianArrayProt=medianArray[i]
                    rangeErrorProt=rangeError[i]
                    protErrorMinValue= protErrorMinVal[i]
                    retall= False
                    verbose= False
            #for 18 quarters plot the flux vs phase and calculate residuals to
                calculate standard deviation
            for quarter in range(18):
                totalList= os.listdir('KICQuarters/KOI_Q'+str(quarter)+'/') #
                    totallist of files in directory
                print 'Quarter:'+ str(quarter)
                fileList=fnmatch.filter(totalList,  '*'+Kepid+'*llc.fits') #first
                    object
                for file in range(len(fileList)):
                    plt.cla()
                    plt.clf()
                    filename=fileList[file]
                    result = filename[:13]  #first thirteen characters of filename
                    KIC_= result[4:]        #first four characters of filename
                    KIC=find(KIC_,'0')      # characters between the 5th char and
                        first 0

                    if p0 < 0.05:

                        KEP_ID.append(KIC)
                        KOI_ID.append(planetNameNoK)
                        P0.append(p0)
                        EPOCH.append(epoch)
                        PERIOD.append(period)
                        IMPACTP.append(impactParameter)
                        INCLINATION.append(inc_deg)
                        ARS.append(ars)
                        ECC.append(ecc)
                        LONGPERI.append(longperi)
                        TEMP.append(Temp)
                        LIMBDARK.append(gamma)
                        RMAG.append(rMag)
                        IMAG.append(iMag)
                        KEPMAG.append(kepMag)
                        Var_IN.append(float('nan'))
                        Var_OUT.append(float('nan'))
                        MAD_IN.append(float('nan'))
                        MAD_OUT.append(float('nan'))
                        NUMPTS_IN.append(float('nan'))
                        NUMPTS_OUT.append(float('nan'))
                        ROTATIONPER.append(medianArrayProt)
                        ROTATIONPER_ERROR.append(rangeErrorProt)
                        ROTATIONPER_ERROR2.append(protErrorMinValue)
                        VRANGE_OUT.append(float('nan'))
                        KS.append(float('nan'))
                        Q.append('Q'+str(quarter))
                        ERRMEAN.append(np.mean(error))
                        COMMENTS.append('Planet star radius ratio < 0.05') #Planet
                            star radius ratio < 0.05 or KepMag >13.0
                        continue

                    if p0 >=0.05: #  originally and p0 >=0.05, kepMag < = 13.0
                        print 'File:'+ filename
                        print KIC, 'KIC'
                        print planetNameNoK, 'planetnamenoK'
                        print KIC
                        print 'kepMag', kepMag
                        print 'period', period
                        #print 'median', medianArrayProt
                        #print 'rangeError', rangeErrorProt

                        keplerFile=pyfits.open('KICQuarters/KOI_Q'+str(quarter)+'/'+
                            filename)   #open file in totallist
                        datafile=keplerFile[1].data
                        flux=datafile.PDCSAP_FLUX   # kepler flux
                        error=datafile.PDCSAP_FLUX_ERR #kepler errors
                        time=datafile.TIME     #TIME data from BinTable
                        quality=datafile.SAP_QUALITY
                        quality= quality[~np.isnan(flux)]
                        error= error[~np.isnan(flux)] #removing nan values from error
                        time= time[~np.isnan(flux)] #removing nan values from time
                        flux= flux[~np.isnan(flux)] #removing nan values from flux


                        removal=[] #remove all true values (cosmic ray correction)
                        for index in range(len(quality)):
                            flag=flagArray(quality[index])
                            qualityflag= findOne(flag)
                            removal.append(qualityflag)

                        #pop all true values of removal from coresponding array
                        flux=np.array(Pop(removal,flux))
                        time=np.array(Pop(removal,time))
                        error=np.array(Pop(removal, error))
                        phase=(time- epoch)/period #phase
                        modPhase=phase-np.floor(phase) #phase with value from 0-1
                        transNum=np.floor(max(phase))-np.floor(min(phase)) #Number of
                            transits

                        #t2z-converts dates to transiting z-parameter for circular
                            orbits
                        z=transit.t2z(epoch, period, inc_deg, time, ars, ecc,
                            longperi)

                        #creates a model light curve
                        #modelFlux=transit.occultquad(z,p0,gamma,retall=False,
                            verbose=False) #iancross occultquad
                        modelFlux= ecode.occultquad(z, gamma[0], gamma[1], p0)[0] #
                            jason eastman occultquad
                        #corrects for dip as planet is behind star
                        for index in range(len(time)):
                            n=math.cos(2*np.pi*(time[index]-epoch)/period)
                            if n<0:
```

```
                modelFlux[index]=1

        delTime=29.4/60./24.     #converting 29.4 min to days
        dur_d=TransDuration/24      #transit duration in days
        durPhaseBlurr=(delTime/period)             #increased phase
            one point
        durPhase=dur_d/period+(2*durPhaseBlurr)     #transdur in
            phase (epoch not need to be subtracted)and increased
            phase by 2 points
        durPhaseHalf=(durPhase/2)    # half of the transit duration
            in phase

        #useful in determining if planet is fully in transit
        zminp0=z-p0
        zplusp0=z+p0

        #creates a single array that includes data before 1.0 and
            after 1.0 in phase
        for index in range(len(modPhase)):
            if modPhase[index] < 0.5:
                modPhase[index]=modPhase[index]+1
        firstTransitE=np.floor(min(phase))+1  #intval of middle of
            first transit
        elim=[]
        inTransitArray=[]
        for numTrans in range(int(transNum)):
            parts= doParts(phase, durPhaseHalf, durPhase, zplusp0,
                firstTransitE)

            if parts[2]== True: #if flag is true then we will skip
                this ONE transit and move on to the next
                #skip transit
                firstTransitE=firstTransitE+1
                continue
            IngTransEg= range(parts[0][0]+1,parts[0][3]+1)
            trans=flux[parts[0][1]+1:parts[0][2]+1]
            for index in range(len(IngTransEg)):
                elim.append(IngTransEg[index])
            for index in range(len(trans)):
                inTransitArray.append(trans[index])
            firstTransitE=firstTransitE+1 # moves on to the next
                transit

        FOutAll=delete_by_indices(flux,elim)
        POutAll=delete_by_indices(phase,elim)
        divOut=FOutAll/(np.median(FOutAll))
        sub= abs(divOut-1)
        vRangeOut=np.percentile(sub,95)- np.percentile(sub,5)
        peak= np.percentile(flux, 95)
        print peak

        Allcount=Allcount+1 # all planet candidates shared between
            kepID list and downloaded files

        #if no transits, then next file
        if transNum ==0:
            print 'skip : no trans'
            print 'period: ', period
            print 'numoftransits: ', transNum
            transitZero=transitZero+1
            KEP_ID.append(KIC)
            KOI_ID.append(planetNameNoK)
            P0.append(p0)
            EPOCH.append(epoch)
            PERIOD.append(period)
            IMPACTP.append(impactParameter)
            INCLINATION.append(inc_deg)
            ARS.append(ars)
            ECC.append(ecc)
            LONGPERI.append(longperi)
            TEMP.append(Temp)
            LIMBDARK.append(gamma)
            RMAG.append(rMag)
            IMAG.append(iMag)
            KEPMAG.append(kepMag)
            Var_IN.append(float('nan'))
            Var_OUT.append(float('nan'))
            MAD_IN.append(float('nan'))
            MAD_OUT.append(float('nan'))
            NUMPTS_IN.append(float('nan'))
            NUMPTS_OUT.append(float('nan'))
            ROTATIONPER.append(medianArrayProt)
            ROTATIONPER_ERROR.append(rangeErrorProt)
            ROTATIONPER_ERROR2.append(protErrorMinValue)
            VRANGE_OUT.append(vRangeOut)
            Q.append('Q'+str(quarter))
            KS.append(float('nan'))
            ERRMEAN.append(np.mean(error))
            COMMENTS.append('No Transit')
            continue
        #if grazzing transit, then next file
        if impactParameter + p0> 1:
            print 'skip: grazing'
            print 'ImpactP +p0: ', impactParameter +p0
            grazzing= grazzing+1
            KEP_ID.append(KIC)
            KOI_ID.append(planetNameNoK)
            P0.append(p0)
            EPOCH.append(epoch)
            PERIOD.append(period)
            IMPACTP.append(impactParameter)
            INCLINATION.append(inc_deg)
            ARS.append(ars)
            ECC.append(ecc)
            LONGPERI.append(longperi)
            TEMP.append(Temp)
            LIMBDARK.append(gamma)
            RMAG.append(rMag)
            IMAG.append(iMag)
            KEPMAG.append(kepMag)
            Var_IN.append(float('nan'))
            Var_OUT.append(float('nan'))
            MAD_IN.append(float('nan'))
            MAD_OUT.append(float('nan'))
            NUMPTS_IN.append(float('nan'))
            NUMPTS_OUT.append(float('nan'))
            ROTATIONPER.append(medianArrayProt)
            ROTATIONPER_ERROR.append(rangeErrorProt)
            ROTATIONPER_ERROR2.append(protErrorMinValue)
            VRANGE_OUT.append(vRangeOut)
            Q.append('Q'+str(quarter))
            KS.append(float('nan'))
            ERRMEAN.append(np.mean(error))
            COMMENTS.append('No Flat Bottom')
            continue

        if len(inTransitArray)==0:
            print 'skip'
            KEP_ID.append(KIC)
            KOI_ID.append(planetNameNoK)
            P0.append(p0)
            EPOCH.append(epoch)
            PERIOD.append(period)
            IMPACTP.append(impactParameter)
            INCLINATION.append(inc_deg)
            ARS.append(ars)
            ECC.append(ecc)
            LONGPERI.append(longperi)
            TEMP.append(Temp)
            LIMBDARK.append(gamma)
            RMAG.append(rMag)
            IMAG.append(iMag)
            KEPMAG.append(kepMag)
            Var_IN.append(float('nan'))
            Var_OUT.append(float('nan'))
            MAD_IN.append(float('nan'))
            MAD_OUT.append(float('nan'))
            NUMPTS_IN.append(float('nan'))
            NUMPTS_OUT.append(float('nan'))
            ROTATIONPER.append(medianArrayProt)
            ROTATIONPER_ERROR.append(rangeErrorProt)
            ROTATIONPER_ERROR2.append(protErrorMinValue)
            VRANGE_OUT.append(vRangeOut)
            Q.append('Q'+str(quarter))
            KS.append(float('nan'))
            ERRMEAN.append(np.mean(error))
            COMMENTS.append('All Transits are Skipped: No Flat Bottom
                ')
            continue

        #plots time-flux and time-phase
        figure1=plt.figure(1)
        plt.subplot(211)
        plt.title(r'Quarter: '+str(quarter)+'_'+result+'_'+planetname
            )
        plt.plot(time, flux,'o')
        plt.xlabel('Time')
        plt.ylabel('Flux')
```

```
plt.subplot(212)
plt.plot(phase,flux,'o',)
plt.xlabel('Phase')
plt.ylabel('Flux')
plt.savefig('AllQ_15_08_14/TimeOrPhaseFlux/'+planetname+'_'+
    result+'Q'+str(quarter)+'_TimeOrPhase_Flux'+'.pdf')

firstTransit=np.floor(min(phase))+1  #intval of middle of
    first transit

fluxNew=[]
timeNew=[]
errorNew=[]
phaseNew=[]
modPhaseNew=[]
modelFluxNew=[]
zFlatNew=[]
fluxOriginalChopped=[]

#only works if not sorted
'''
loops through every flux value in a file and seperates them
    into categories (ingress, egress,left of transit, right
    of transit and fully in transit).
The last index number of the correspnding category is
    inserted in an array named parts.
1)Removed one transit duration around the transit on both
    ends for each transit.
2)Fit a second degree polynomial to the out of transit region
    .
3)Took the difference between the polynomial fit and the out
    of transit region.

'''

for numTrans in range(int(transNum)):

    parts= doParts(phase, durPhaseHalf, durPhase, zplusp0,
        firstTransit)

    if parts[2]== True: #if flag is true then we will skip
        this ONE transit and move on to the next
        #skip transit
        firstTransit=firstTransit+1
        continue

    '''
    Out of transit flux and phase values
    Removed one transit duration around the transit on both
        ends for each transit.
    '''
    fluxOut=np.concatenate((flux[parts[1]+1:parts[0][0]+1],
        flux[parts[0][3]+1:parts[0][4]+1]))
    phaseOut=np.concatenate((phase[parts[1]+1:parts[0][0]+1],
        phase[parts[0][3]+1:parts[0][4]+1]))
```

```
#takes all flux, phase, times, errors, modphases and
    models from parts[0]-parts[4]
fluxAll=flux[parts[1]+1:parts[0][4]+1]
phaseAll=phase[parts[1]+1:parts[0][4]+1]
timeAll=time[parts[1]+1:parts[0][4]+1]
errorAll=error[parts[1]+1:parts[0][4]+1]
modPhaseAll=modPhase[parts[1]+1:parts[0][4]+1]
modelAll=modelFlux[parts[1]+1:parts[0][4]+1]

#ypoly=np.polyfit(phaseOut, fluxOut,2) # Fit a second
    degree polynomial to the out of transit region.
ypoly=ropoly.polyfit(phaseOut, fluxOut,2) #generates
    coeffiecients
p=np.poly1d(ypoly) # generates flux as a funtion of phase
    equation
newyAll=p(phaseAll) # generates points for flux values
    using equation above

#Took the difference between the polynomial fit and the
    out of transit region. to get a "flat flux" with less
    variation
flatFlux=(fluxAll-newyAll+peak)/peak

#print 'Transit number: ', numTrans+1
chiarray=[]
EpochChi=[]

num=int(np.floor(len(fluxAll)/2))*2

for index in range(-num,num):
    epochSteps=epoch+(index*(1.7*dur_d/(2.*num))) # we
        used 1.5 meaning the min. val needs to be withing
        one duration from the epoch of zero
    zSteps=transit.t2z(epochSteps, period, inc_deg,
        timeAll, ars, ecc, longperi)
    modelFlux__2= ecode.occultquad(zSteps, gamma[0],
        gamma[1], p0)[0] # jason eastman occultquad
    chisquared_= chi2(flatFlux, modelFlux__2, errorAll)
    chiarray.append(chisquared_)
    EpochChi.append(epochSteps)

minval= min(chiarray)
minIndex=chiarray.index(minval) #EpochhChi[minIndex] ====
    min val of epoch
space=(num/4)
if space < 8:
    space= 8

lenghTrue=len(EpochChi)
minIndexPlusSpace=minIndex+space
minIndexMinusSpace=minIndex-space
minIndexMinusSpace= minIndex-space
if minIndexPlusSpace >= len(EpochChi):
    minIndexPlusSpace= lenghTrue-1
```

```
                if minIndexMinusSpace <= 0:
                    minIndexMinusSpace=0


                figure10=plt.figure(10)
                plt.title(r'Quarter: '+str(quarter)+'_'+result+'_'+
                    planetname)
                plt.plot(EpochChi, chiarray, 'o', markersize=3)
                plt.plot(EpochChi[minIndexMinusSpace:minIndexPlusSpace],
                    chiarray[minIndexMinusSpace:minIndexPlusSpace], 'o')

                polyChi=ropoly.polyfit(np.array(EpochChi
                    [minIndexMinusSpace:minIndexPlusSpace]), np.array
                    (chiarray[minIndexMinusSpace:minIndexPlusSpace]),2)
                chiequation=np.poly1d(polyChi)
                EpochChiSmooth=np.linspace(EpochChi[minIndexMinusSpace],
                    EpochChi[minIndexPlusSpace] ,1000)
                chi=chiequation(EpochChiSmooth)

                plt.plot(EpochChiSmooth, chi)
                plt.xlabel('Epoch')
                plt.ylabel('Chi2')
                plt.savefig('AllQ_15_08_14/EpochChi/'+planetname+'_'+
                    result+'Q'+str(quarter)+'EpochChi'+'.pdf')


                chi=chi.tolist()
                minIndexPolyChi=chi.index(min(chi))
                epochWithShift=EpochChiSmooth[minIndexPolyChi]

                SHIFT.append(np.abs(epoch-epochWithShift))
                print epoch-epochWithShift
                if np.abs(epoch-epochWithShift)<=0.008:
                    epochWithShift=epoch
                print epoch-epochWithShift

                phaseIndiv=(timeAll-epochWithShift)/period
                modPhaseIndiv=phaseIndiv-np.floor(phaseIndiv) #phase with
                    value from 0-1
                zFlatIndiv=transit.t2z(epochWithShift, period, inc_deg,
                    timeAll, ars, ecc, longperi) #<-------changed this

                #creates a single array that includes data before 1.0 and
                    after 1.0 in phase
                for index in range(len(modPhaseIndiv)):
                    if modPhaseIndiv[index] < 0.5:
                        modPhaseIndiv[index]=modPhaseIndiv[index]+1

                figure12=plt.figure(12)
                plt.subplot(211)
                #plt.xticks(np.arange(min(modPhaseAllB),
                    max(modPhaseAllB), .1))
                plt.title(r'Quarter: '+str(quarter)+'_'+result+'_'+
                    planetname)
                plt.plot(modPhaseIndiv, newyAll, color='r')
                plt.plot(modPhaseIndiv, fluxAll, 'o')
```

```
        plt.ylabel('Normalized Flux')
        plt.subplot(212)
        plt.plot(modPhaseIndiv, flatFlux, 'o')
        plt.xlabel('ModPhase')


        #appending all parts of the transit to an outside array
        for index in range(len(flatFlux)):
            fluxNew.append(flatFlux[index])
            timeNew.append(timeAll[index])
            errorNew.append(errorAll[index])
            phaseNew.append(phaseIndiv[index])
            modPhaseNew.append(modPhaseIndiv[index])
            modelFluxNew.append(modelAll[index])
            zFlatNew.append(zFlatIndiv[index])
            fluxOriginalChopped.append(fluxAll[index])

        firstTransit=firstTransit+1 # moves on to the next
            transit


    #converting lists into numpy arrays
    AllTime=np.array(timeNew)
    AllFlux=np.array(fluxNew)
    AllError=np.array(errorNew)
    AllModPhase=np.array(modPhaseNew)
    AllPhase=np.array(phaseNew)
    AllZFlat=np.array(zFlatNew)
    AllZFlatPlusP0=AllZFlat+p0
    AllfluxOriginalChopped=np.array(fluxOriginalChopped)


    #-------
    firstTransit2=np.floor(min(AllPhase))+1
    fluxNewB=[]
    timeNewB=[]
    errorNewB=[]
    phaseNewB=[]
    modPhaseNewB=[]
    zFlatNewB=[]

    for numTrans in range(int(transNum)):


        parts2= doParts(AllPhase, durPhaseHalf, durPhase,
            AllZFlatPlusP0, firstTransit2)

        if parts2[2] == True: #if flag is true then we will skip
            this ONE transit and move on to the next
            #skip transit
            firstTransit2=firstTransit2+1
            print 'cont'
            continue

        fluxOutB=np.concatenate((AllfluxOriginalChopped[parts2[1]
            +1:parts2[0][0]+1],AllfluxOriginalChopped[parts2[0][3
            ]+1:parts2[0][4]+1]))
        phaseOutB=np.concatenate((AllPhase[parts2[1]+1:parts2[0]
            [0]+1],AllPhase[parts2[0][3]+1:parts2[0][4]+1]))

        #takes all flux, phase, times, errors, modphases and
            models from parts[0]-parts[4]
        fluxAllB=AllfluxOriginalChopped[parts2[1]+1:parts2[0][4]
            +1]
        phaseAllB=AllPhase[parts2[1]+1:parts2[0][4]+1]
        timeAllB=AllTime[parts2[1]+1:parts2[0][4]+1]
        errorAllB=AllError[parts2[1]+1:parts2[0][4]+1]
        modPhaseAllB=AllModPhase[parts2[1]+1:parts2[0][4]+1]
        zFlatAllB=AllZFlat[parts2[1]+1:parts2[0][4]+1]

        #for individual fits files
        for index in range(len(timeAllB)):
            TIME.append(timeAllB[index])
            PHASE.append(phaseAllB[index])
            FLUX.append(fluxAllB[index])
            ERROR.append(errorAllB[index])
            QUARTER.append('Q'+str(quarter))

        #ypoly=np.polyfit(phaseOut, fluxOut,2) # Fit a second
            degree polynomial to the out of transit region.

        ypolyB=ropoly.polyfit(phaseOutB, fluxOutB,2) #generates
            coeffiecients
        pB=np.poly1d(ypolyB) # generates flux as a funtion of
            phase equation
        newyAllB=pB(phaseAllB) # generates points for flux values
            using equation above

        #Took the difference between the polynomial fit and the
            out of transit region. to get a "flat flux" with less
            variation
        flatFluxB=(fluxAllB-newyAllB+peak)/peak
        errorAllB= errorAllB/peak

        figure7=plt.figure(7)
        plt.subplot(211)
        plt.xticks(np.arange(min(modPhaseAllB), max(modPhaseAllB)
            , .1))
        plt.title(r'Quarter: '+str(quarter)+'_'+result+'_'+
            planetname)
        plt.plot(modPhaseAllB, newyAllB, color='r')
        plt.plot(modPhaseAllB, fluxAllB, 'o')
        plt.subplot(212)
        plt.xlim(min(modPhaseAllB), max(modPhaseAllB))
        plt.plot(modPhaseAllB, flatFluxB, 'o')

        for index in range(len(timeAllB)):
            FLAT_FLUX.append(flatFluxB[index])
            FLAT_ERROR.append(errorAllB[index])
            FLAT_QUARTER.append('Q'+str(quarter))

        #appending all parts of the transit to an outside array
```

```
        for index in range(len(flatFluxB)):
            fluxNewB.append(flatFluxB[index])
            timeNewB.append(timeAllB[index])
            errorNewB.append(errorAllB[index])
            phaseNewB.append(phaseAllB[index])
            modPhaseNewB.append(modPhaseAllB[index])
            zFlatNewB.append(zFlatAllB[index])
        firstTransit2=firstTransit2+1 # moves on to the next
            transit


    #Added this last minute, bc all tranists had been skiped so
        all of parts was -1.

    if len(timeNewB)==0:
        print 'skip'
        KEP_ID.append(KIC)
        KOI_ID.append(planetNameNoK)
        P0.append(p0)
        EPOCH.append(epoch)
        PERIOD.append(period)
        IMPACTP.append(impactParameter)
        INCLINATION.append(inc_deg)
        ARS.append(ars)
        ECC.append(ecc)
        LONGPERI.append(longperi)
        TEMP.append(Temp)
        LIMBDARK.append(gamma)
        RMAG.append(rMag)
        IMAG.append(iMag)
        KEPMAG.append(kepMag)
        Var_IN.append(float('nan'))
        Var_OUT.append(float('nan'))
        MAD_IN.append(float('nan'))
        MAD_OUT.append(float('nan'))
        NUMPTS_IN.append(float('nan'))
        NUMPTS_OUT.append(float('nan'))
        KS.append(float('nan'))
        ROTATIONPER.append(medianArrayProt)
        ROTATIONPER_ERROR.append(rangeErrorProt)
        ROTATIONPER_ERROR2.append(protErrorMinValue)
        VRANGE_OUT.append(vRangeOut)
        Q.append('Q'+str(quarter))
        COMMENTS.append('All Transits are Skipped: No Flat Bottom
            ')
        continue

    #__________________________________*****

    timeNewB=np.array(timeNewB)
    fluxNewB=np.array(fluxNewB)
    errorNewB=np.array(errorNewB)
    modPhaseNewB=np.array(modPhaseNewB)
    phaseNewB=np.array(phaseNewB)
    zFlatNewB=np.array(zFlatNewB)
```

```
        # sorting phase, flux, error etc. by modphase
        modPhaseNewB, timeNewB, fluxNewB,errorNewB, phaseNewB,
            zFlatNewB = (np.array(x) for x in zip(*sorted(zip
            (modPhaseNewB, timeNewB, fluxNewB,errorNewB,phaseNewB,
            zFlatNewB))))

        timeNewB=np.array(timeNewB)
        fluxNewB=np.array(fluxNewB)
        errorNewB=np.array(errorNewB)
        modPhaseNewB=np.array(modPhaseNewB)
        phaseNewB=np.array(phaseNewB)
        zFlatNewB=np.array(zFlatNewB)


        #generates a phase array of 1000 equally spaced elements
            (made up phases)

        phaseSmooth=np.linspace(durPhaseHalf*-4, durPhaseHalf*4,1000)
        startEpoch=0.0

        modelFlux_= makeModel(startEpoch, period, inc_deg, ars, ecc,
            longperi,gamma, p0, delTime, phaseSmooth, modPhaseNewB)

        epochShift=0

        #-----------------------------------------------------
        #calculating residuals (sorted arrays)
        shift= (epochShift/period) # shift in phase
        Residuals=modelFlux_-fluxNewB
        timeNewB=np.array(timeNewB)

        #Z values using flat flux
        zFlatNewBPlusP0=zFlatNewB+p0 #useful in determining if
            flatflux values corresponds to planet fully in transit

        parts3= doParts(modPhaseNewB, durPhaseHalf, durPhase,
            zFlatNewBPlusP0, 1) #firsttransit= 1

        fluxOut=np.concatenate((fluxNewB[parts3[1]:parts3[0][0]+1],
            fluxNewB[parts3[0][3]+1:parts3[0][4]+2]))
        modPhaseOut=np.concatenate((modPhaseNewB[parts3[1]:parts3[0]
            [0]+1],modPhaseNewB[parts3[0][3]+1:parts3[0][4]+2]))
        modelOut=np.concatenate((modelFlux_[parts3[1]:parts3[0][0]+1]
            ,modelFlux_[parts3[0][3]+1:parts3[0][4]+2]))
        errorOut=np.concatenate((errorNewB[parts3[1]:parts3[0][0]+1],
            errorNewB[parts3[0][3]+1:parts3[0][4]+2]))


        f=fluxNewB[parts3[1]-1:parts3[0][4]+2]
        mp=modPhaseNewB[parts3[1]-1:parts3[0][4]+2]


        fluxIn=fluxNewB[parts3[0][1]+1:parts3[0][2]+1]
        modPhaseIn=modPhaseNewB[parts3[0][1]+1:parts3[0][2]+1]
        modelIn=modelFlux_[parts3[0][1]+1:parts3[0][2]+1]
        ErrorIn=errorNewB[parts3[0][1]+1:parts3[0][2]+1]
        ingress=fluxNewB[parts3[0][0]+1:parts3[0][1]+1]
        egress=fluxNewB[parts3[0][2]+1:parts3[0][3]+1]
        ingressModPhase=modPhaseNewB[parts3[0][0]+1:parts3[0][1]+1]
        egressModPhase=modPhaseNewB[parts3[0][2]+1:parts3[0][3]+1]

        residualsOut=fluxOut-modelOut
        residualsI=fluxIn-modelIn
        residualsIn= residualsI-np.median(residualsI)

        for index in range(len(residualsOut)):
            RESIDUALS.append(residualsOut[index])
            TRANSIT_FLAG.append('OUT_OF_TRANSIT')
        for index in range(len(residualsIn)):
            RESIDUALS.append(residualsIn[index])
            TRANSIT_FLAG.append('IN_TRANSIT')

        VRANGE.append(vRangeOut)
        VarIn= np.nanstd(residualsIn)**2
        VarOut=np.nanstd(residualsOut)**2

        figure8=plt.figure(8)
        plt.subplot(211)
        plt.xticks(np.arange(min(modPhaseOut), max(modPhaseOut), .1))
        plt.plot(modPhaseOut, fluxOut,'o', color='b')
        plt.plot(modPhaseIn, fluxIn, 'o', color='b')
        plt.plot(ingressModPhase, ingress, 'o', color='w')
        plt.plot(egressModPhase, egress, 'o', color='w')
        plt.plot(modPhaseNewB,modelFlux_, color='g')
        plt.plot(modPhaseIn, fluxIn, 'o', color='b')
        plt.ylabel('Normalized Flux')

        plt.subplot(212)
        plt.xlim(min(modPhaseOut), max(modPhaseOut))
        plt.plot(modPhaseOut, residualsOut, 'o',  color='g')
        plt.plot(modPhaseIn, residualsIn, 'o', color='g')
        plt.xlabel('ModPhase, Residuals')
        plt.savefig('AllQ_15_08_14/ModPhaseFlux/'+planetname+'_'+
            result+'Q'+str(quarter)+'_modPhaseFlux'+'.pdf')

        figure7=plt.figure(7)
        plt.xlabel('VarOut:'+str(VarOut)+' VarIn:'+str(VarIn)+ '
            kepMag:'+str(kepMag)+' Temp:'+str(Temp))
        plt.savefig('AllQ_15_08_14/Baseline/'+planetname+'_'+result+
            'Q'+str(quarter)+'_Baseline'+'.pdf')

        madIn=mad(residualsIn)
        madOut=mad(residualsOut)

        KEP_ID.append(KIC)
        KOI_ID.append(planetNameNoK)
        P0.append(p0)
        EPOCH.append(epoch)
        PERIOD.append(period)
        IMPACTP.append(impactParameter)
        INCLINATION.append(inc_deg)
```

```
                ARS.append(ars)
                ECC.append(ecc)
                LONGPERI.append(longperi)
                TEMP.append(Temp)
                LIMBDARK.append(gamma)
                RMAG.append(rMag)
                IMAG.append(iMag)
                KEPMAG.append(kepMag)
                Var_IN.append(VarIn)
                Var_OUT.append(VarOut)
                MAD_IN.append(madIn)
                MAD_OUT.append(madOut)
                NUMPTS_IN.append(len(fluxIn))
                NUMPTS_OUT.append(len(fluxOut))
                ROTATIONPER.append(medianArrayProt)
                ROTATIONPER_ERROR.append(rangeErrorProt)
                ROTATIONPER_ERROR2.append(protErrorMinValue)
                VRANGE_OUT.append(vRangeOut)
                Q.append('Q'+str(quarter))
                COMMENTS.append('None')
                ERRMEAN.append(np.mean(error))
                KS.append(scipy.stats.ks_2samp(residualsIn,residualsOut)[0])
                Name=planetname
                for arg in sys.argv:
                    if 'showplot' in sys.argv:
                        plt.show()
                plt.close('all')



        if Name == 'blah':
            continue
        b1=np.array(QUARTER)
        b2=np.array(TIME)
        b3=np.array(PHASE)
        b4=np.array(FLUX)
        b5=np.array(ERROR)
        b6=np.array(FLAT_QUARTER)
        b7=np.array(TRANSIT_FLAG)
        b8=np.array(FLAT_FLUX)
        b9=np.array(FLAT_PHASE)
        b10=np.array(FLAT_ERROR)
        b11=np.array(RESIDUALS)



        outResid=[]
        inResid=[]
        for index in range(len(b7)):
            if b7[index]== 'OUT_OF_TRANSIT':
                outResid.append(b11[index])
            if b7[index] == 'IN_TRANSIT':
                inResid.append(b11[index])


        c1 = pyfits.Column(name='QUARTER',format='31A', array=b1)
        c2 = pyfits.Column(name='TIME',format='1E', array=b2)
        c3 = pyfits.Column(name='PHASE',format='1E', array=b3)
        c4 = pyfits.Column(name='FLUX', format='1E', array=b4)
        c5 = pyfits.Column(name='ERROR', format='1E', array=b5)
        c6= pyfits.Column(name='FLAT_QUARTER', format='31A', array= b6)
        c7= pyfits.Column(name='TRANSIT_FLAG', format='31A', array= b7)
        c8 = pyfits.Column(name='FLAT_FLUX', format='1E', array=b8)
        c9 = pyfits.Column(name='FLAT_PHASE', format='1E', array=b9)
        c10 = pyfits.Column(name='FLAT_ERROR', format='1E', array=b10)
        c11= pyfits.Column(name='RESIDUALS', format='1E', array= b11)
        cAll = pyfits.ColDefs([c1, c2,c3,c4,c5,c6,c7,c8,c9,c10,c11])
        tbhdu2 = pyfits.BinTableHDU.from_columns(cAll)
        tbhdu2.writeto('AllQ_15_08_14/QuartersFitsFile/'+object+'.fits', clobber=
            True)
        paperN=pyfits.open('AllQ_15_08_14/QuartersFitsFile/'+object+'.fits')
            #open file in totallist
        dataFileN=paperN[1].data
        dataCubeN=dataFileN.columns
        paperN[0].header['name']= object
        paperN[0].header['KEPMAG']= str(kepMag)
        print 'Temp', Temp
        paperN[0].header['TEMP']= str(Temp)
        paperN[0].header['IMPACT']= str(impactParameter)
        paperN[0].header['PERIOD'] = str(period)
        paperN[0].header['EPOCH'] = str(epoch)
        paperN[0].header['IncDeg'] =str(inc_deg)
        paperN[0].header['SmaORs'] = str(ars)
        paperN[0].header['Ecc'] = str(ecc)
        paperN[0].header['LongPeri'] = str(longperi)
        paperN[0].header['LDcoeff1'] = str(gamma[0])
        paperN[0].header['LDcoeff2'] = str(gamma[1])
        paperN[0].header['RpORs'] = str(p0)
        paperN[0].header['KS']= str(scipy.stats.ks_2samp(outResid,inResid)[0])
        paperN[0].header['ErrMean']= str(np.mean(ERROR))        # see if this is
            correct
        paperN[0].header['VRMean']= str(np.mean(VRANGE))
        paperN[0].header['VRError']= str(max(VRANGE)- min(VRANGE))
        paperN[0].header['PRot']=str(medianArrayProt)
        paperN[0].header['ErrProt']=str(rangeErrorProt)
        paperN[0].header['ErrProt2']=str(protErrorMinValue)
        paperN.writeto('AllQ_15_08_14/QuartersFitsFile/'+object+'.fits', clobber=
            True)

    #converting to a numpy array
    a1=np.array(KEP_ID)
    a2=np.array(KOI_ID)
    a3=np.array(P0)
    a4=np.array(EPOCH)
    a5=np.array(PERIOD)
    a6=np.array(IMPACTP)
    a7=np.array(INCLINATION)
    a8=np.array(ARS)
    a9=np.array(ECC)
    a10=np.array(LONGPERI)
    a11=np.array(TEMP)
    a12=np.array(LIMBDARK)
```

```
        a13=np.array(KEPMAG)
        a14=np.array(RMAG)
        a15=np.array(IMAG)
        a16=np.array(Var_IN)
        a17=np.array(Var_OUT)
        a18=np.array(MAD_IN)
        a19=np.array(MAD_OUT)
        a20=np.array(NUMPTS_IN)
        a21=np.array(NUMPTS_OUT)
        a22=np.array(ROTATIONPER)
        a23=np.array(ROTATIONPER_ERROR)
        a24=np.array(ROTATIONPER_ERROR2)
        a25=np.array(VRANGE_OUT)
        a26=np.array(ERRMEAN)
        a27=np.array(KS)
        a28=np.array(Q)
        a29=np.array(COMMENTS)
        a30=np.array(SHIFT)


        #creating a fits table and writing to a fits table
        col1 = pyfits.Column(name='KEP_ID', format='1E', array=a1)
        col2 = pyfits.Column(name='KOI_ID', format='1E', array=a2)
        col3 = pyfits.Column(name='PLANET_STAR_RADIUSRATIO', format= '1E', array=a3)
        col4 = pyfits.Column(name='EPOCH', format = '1E', array=a4)
        col5 = pyfits.Column(name='PERIOD', format = '1E', array=a5)
        col6 =pyfits.Column(name='IMPACTPARAMETER',format='1E', array=a6)
        col7 = pyfits.Column(name= 'INCLINATION_DEG', format= '1E', array=a7)
        col8 = pyfits.Column(name= 'SEMIMAJORAXIS_STARRADIUS_RATIO', format= '1E',
            array =a8)
        col9 = pyfits.Column(name='ECCENTRICITY', format = '1E', array= a9)
        col10 = pyfits.Column(name= 'LONGITUDE_ PERIAPSIS', format= '1E', array =a10)
        col11 = pyfits.Column(name='TEMPERATURE',format='1E', array=a11)
        col12 = pyfits.Column(name='LIMBDARK', format= 'PJ()', array=a12)
        col13 = pyfits.Column(name='KEPMAG', format= '1E', array= a13)
        col14 = pyfits.Column(name='IMAG', format= '1E', array= a14)
        col15 = pyfits.Column(name='RMag', format= '1E', array= a15)
        col16 = pyfits.Column(name='Variance_INTRANSIT',format='1E', array=a16)
        col17 = pyfits.Column(name='Variance_OUTOFTRANSIT',format='1E', array=a17)
        col18 = pyfits.Column(name='MEANABSDEV_INTRANSIT',format='1E', array=a18)
        col19 = pyfits.Column(name='MEANABSDEV_OUTOFTRANSIT', format='1E', array=a19)
        col20 = pyfits.Column(name='NUMPTS_INTRANIST', format='1E', array=a20)
        col21 = pyfits.Column(name='NUMPTS_OUTOFTRANSIT',format='1E', array=a21)
        col22 = pyfits.Column(name='ROTATION_PERIOD',format='1E', array=a22)
        col23 = pyfits.Column(name='ROTATION_PERIOD_ERROR',format='1E', array=a23)
        col24 = pyfits.Column(name='ROTATION_PERIOD_ERROR2', format='1E', array=a24)
        col25 = pyfits.Column(name='VRANGE_OUT', format='1E', array=a25)
        col26 = pyfits.Column(name= 'ERROR_MEAN', format='1E', array=a26)
        col27 = pyfits.Column(name= 'KS', format='1E', array=a27)
        col28= pyfits.Column(name='QUARTER', format='31A', array= a28)
        col29= pyfits.Column(name='COMMENTS', format='31A', array= a29)
        col30 = pyfits.Column(name='SHIFT', format='1E', array=a30)

        cols = pyfits.ColDefs([col1, col2,col3,col4,col5,col6,col7,col8,col9,col10,
            col11,col12,col13,col14,col15,col16,col17,col18,col19,col20,col21,col22,
            col23,col24,col25,col26,col27,col28, col29, col30])
        tbhdu = pyfits.BinTableHDU.from_columns(cols)

        tbhdu.writeto('AllQ_15_08_14/QuartersFitsFile/StarSpotStatsAll.fits',clobber=
            True)

    print 'AllCount: ', Allcount
    print 'grazzing: ', grazzing
    print 'transitZero: ', transitZero
```