Analytical and Numerical Solutions to the Inverse Problem of Solving for Exoplanet Albedo Maps

Numa Karolinski
Supervisor: Professor Nicolas Cowan
April 15, 2020

1 Introduction

1.1 Exoplanets

Exoplanets are planets that orbit a star outside of our solar system. The idea of an exotic far-off world has been very exciting to people since the discovery of other planets in our solar system, and stars other than our own. Exoplanets very similar to Earth have been mostly kept to science fiction, rather than science. Unfortunately, due to the vast distance between Earth and other solar systems, as well as the miniature size of habitable planets, Earth-sized planets are mostly undiscovered. Until now, the majority of exoplanets that have been discovered are gas giants similar to Jupiter. Powerful ground-based telescopes are currently still too weak to decipher small rocky planets within our galaxy. This fact alone, however, does not stop scientists from learning about exoplanets, and preparing for the future of advanced telescopes [1].

Some exoplanets have been directly imaged. The first exoplanets whose orbital motion was confirmed by direct imaging was the HR 8799 system. A video of four exoplanets orbiting their host star can be found online, but a snapshot can be seen in Figure 1 [2]:

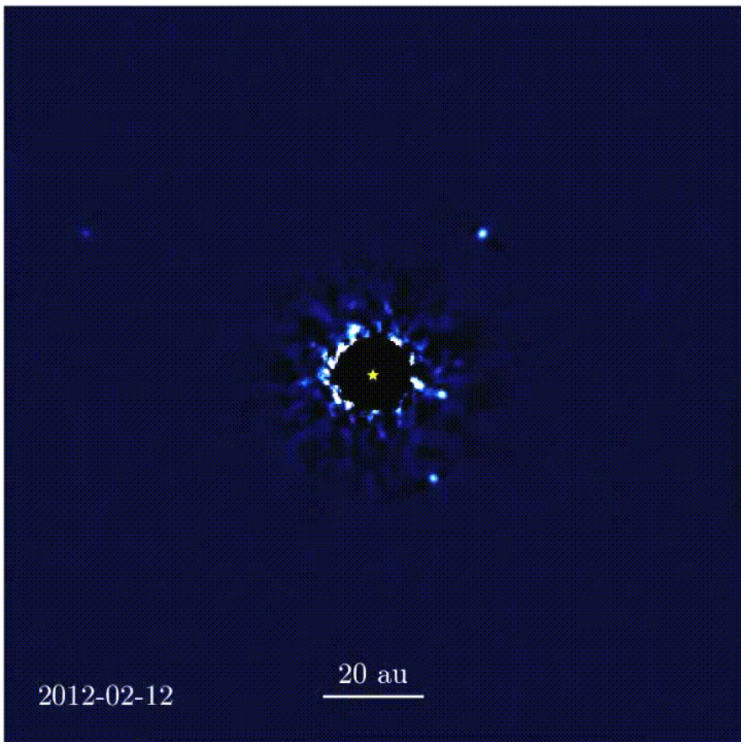

Figure 1: This is an image of the HR 8799 solar system. A black circle is placed at the center of the telescope to block out as much light from the host star as possible, this is indicated by the small yellow star. Four small circles of light, four super-Jupiters, are found orbiting around the host star. This image comes from a direct imaging video which shows 7 years of orbital motion. There is a bright mess of light at the center of the image that comes from the host star seeping past the central blockage.

There is a frontier of exoplanets ready to be discovered, and they just so happen to be exactly within the habitable zone that many astronomers are looking for. Hotter and larger planets tend to be easier to see, but neither of these qualities describe Earth. Telescopes may be only a couple of decades from discovering much smaller cooler planets like our own. Figure 2 depicts all planets that have been detected in some way by astronomers to this day. The large gap of planet discoveries with small radii and large orbital periods are noteworthy. Large orbital periods tend to imply a colder planet. This gap in discoveries is where many Earth-like planets would be found in the image [3]:

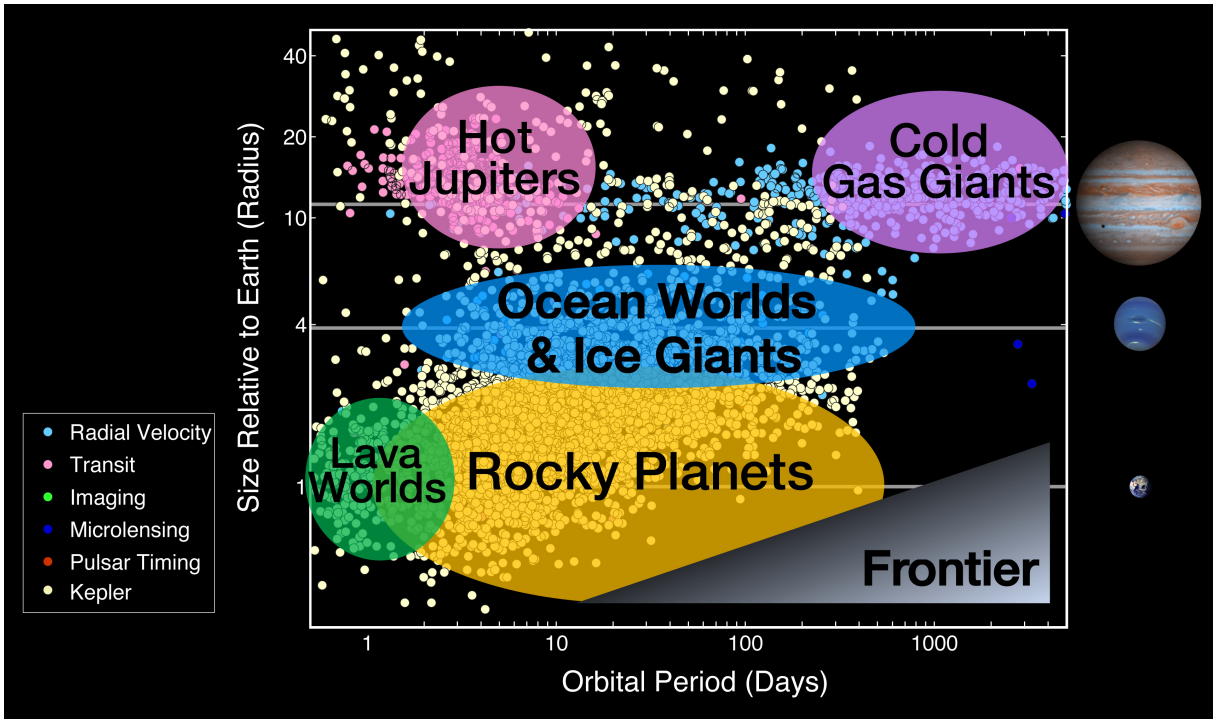

Figure 2: This is a NASA image depicting the additional exoplanets that the Kepler mission discovered. The legend depicts how planets were found, which describes the color of each dot. Most important in this image is the frontier, the bottom right of the image, which is a region of exoplanets characterized by their radius and orbital period that have never been discovered, and often covers the habitable zone of host stars [3].

1.2 Photometry

Photometry is the measurement of flux or intensity of light from astronomical objects. There is plenty of intricacy that could be elaborated on about detectors, and techniques behind photometry, but it is not important for understanding this project [4]. We can consider Figure 1 as being photometry. This image is just one snapshot, or one measurement, of light from four exoplanets and their host star.

Time-resolved photometry is the concept of making multiple measurements of light over time. Such measurements produce what is called a lightcurve. A lightcurve can take on many shapes, and many different types of analyses can be made of them. Different lightcurves

have notable features that indicate an exoplanet, and these features are used as criteria for exoplanet discovery. These features are given names that describe the features. An example of a transit event and a microlensing event are found in Figure 3.

In this report, lightcurves refer to the light that is emitted and reflected off of an exoplanet that is directly imaged.

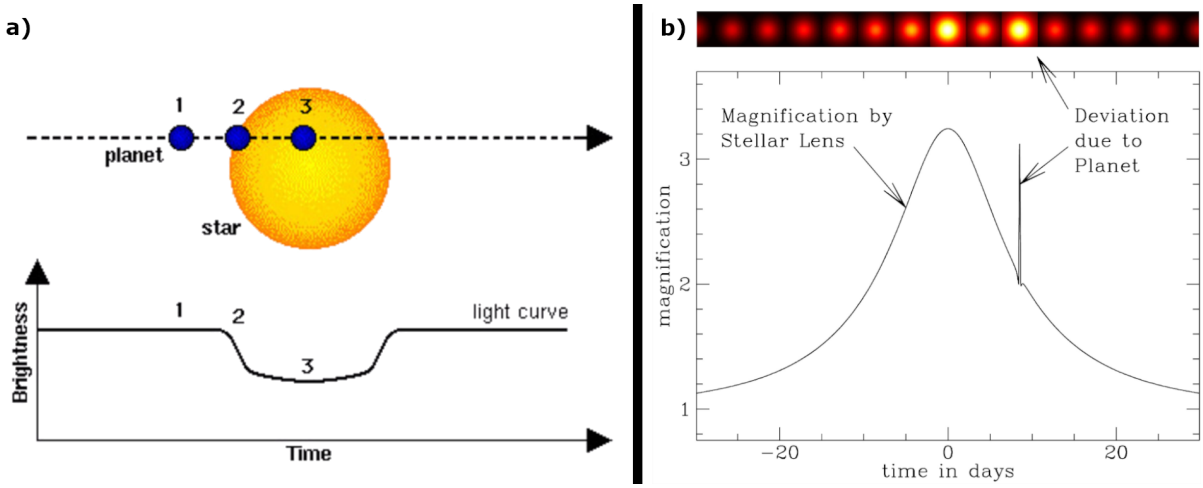

Figure 3: a), on the left, depicts a transit event. The brightness of the host star is being measured. When an exoplanet passes in front of the host star, the brightness decreases [5]. b), on the right, depicts a gravitational microlensing event. When a stellar object passes in front of a host star, the gravity of the stellar object bends the light, amplifying the light that gets to earth. If that stellar object has an exoplanet orbiting it, there is an additional spike [6]. Both of these events are signs of exoplanets.

1.3 Exoplanet Albedo Maps

Albedo is the proportion of incident light that is reflected. For exoplanets, this is the fraction of incident light from a host star that reflects off of an exoplanet. An exoplanet albedo map is thus a color diagram of the entire surface of an exoplanet, where the color represents the albedo at different points on the exoplanet. Figure 4 is an example of such a map. There are some important things to note about this map. The left to the right side of the map spans 360 degrees, hence the colors should be approximately the same on the left and right side. The top to the bottom of the map spans 180 degrees, and these are the north and south poles of the sphere, respectively.

1.4 Albedo Map Parameterization

1.4.1 HEALPixels

Albedo maps need a proper parameterization. Since exoplanets are spheres of roughly constant radii, we can think of an albedo map as $A(\theta, \phi)$, with a different albedo at different angles. Depicting a spherical map proves to be tricky, but a standard convention for discrete

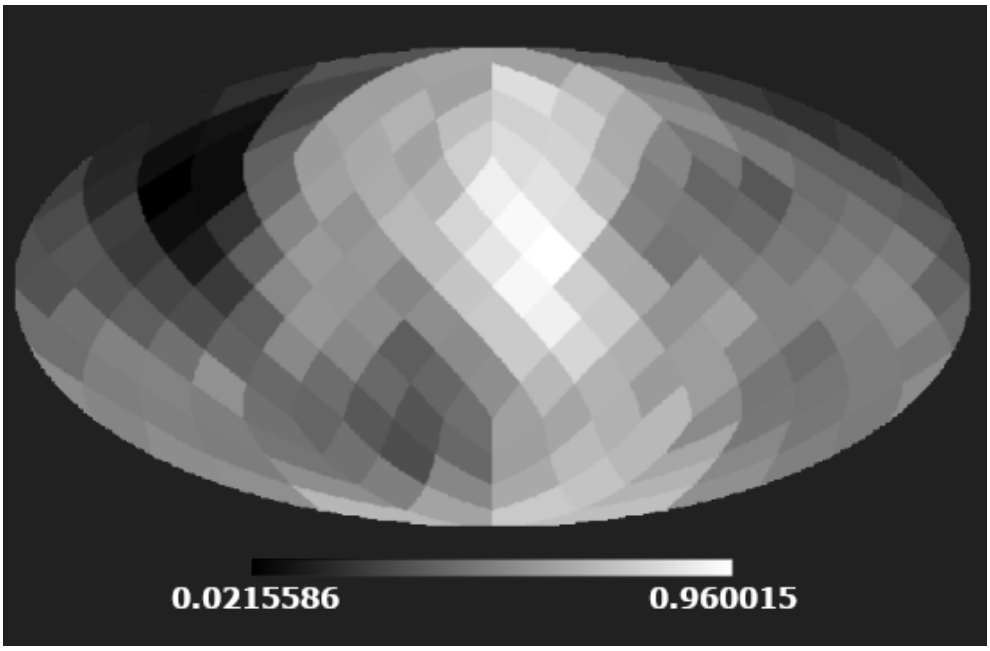

Figure 4: This is an exoplanet albedo map. It is created with a HEALPixel pixelization [7]. The maximum possible albedo is 1, which is depicted in white, and the minimum possible albedo is 0, which is depicted in black.

maps is the HEALPixel convention. HEALPixel stands for Hierarchical, Equal Area, and isoLatitude Pixelation. In order to add more pixels to such a map, the number of pixels must quadruple; this is the hierarchy since each pixel splits into four new pixels. Sets of pixels lie along equal latitude, and every pixel has the same area, but different sets are different shapes. This convention can be seen in Figure 5 [7].

1.4.2 Spherical Harmonics

Spherical harmonics are another way of parameterizing the exoplanet albedo maps. Spherical harmonics are much more useful because they are continuous maps. The spherical harmonic representation is a complete orthonormal basis that is defined on the surface of a sphere in a similar way that sine and cosine functions are used to represent functions on a circle (Fourier series). Spherical harmonics are often denoted in the form $Y_l^m(\theta, \phi)$, where $l = 0, 1, ...$ and $m = -l, ...0, ...l$ are integers that define the function [9]. The Y_l^m maps for $l = 0, 1, 2, 3, 4$ with positive m are depicted in Figure 6. Since spherical harmonics are continuous, they can be used to mathematically define exoplanet albedo maps, and then they can be converted into the HEALPix mapping so that they can be visualized.

Spherical harmonics, Y_l^m, form the basis for exoplanet albedo maps, and hence we can define the albedo map $A(\theta, \phi)$ as a linear combination of these spherical harmonics. The eigenvalues of this linear combination are referred to as c_l^m values, the relationship is:

$$A(\theta, \phi) = \sum_l \sum_m c_l^m Y_l^m(\theta, \phi) \quad [11] \tag{1}$$

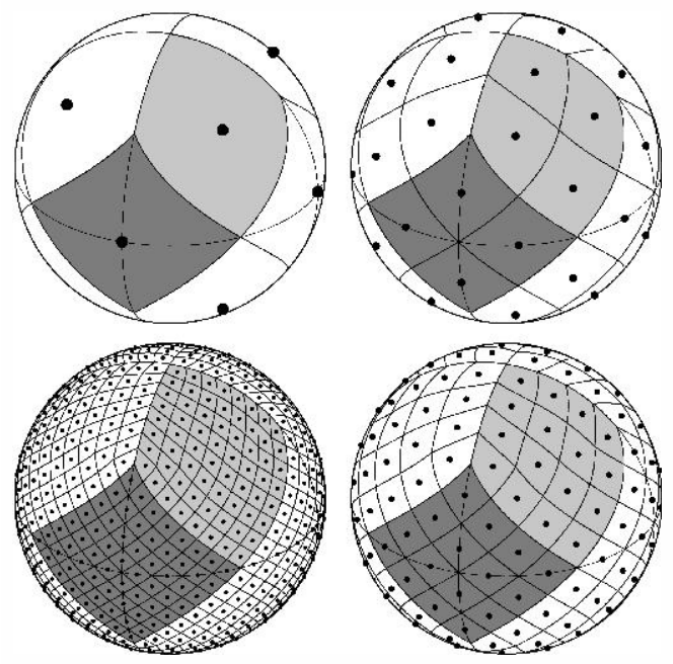

Figure 5: Four different HEALPix representations can be seen. In the top left is the smallest number pixel representation, to the right of that, each pixel is divided into four. Each pixel can be subdivided into four repeatedly to make more precise images. Starting from the top left, and going clockwise, there are 12, 48, 192, and 768 pixels on the spheres [7].

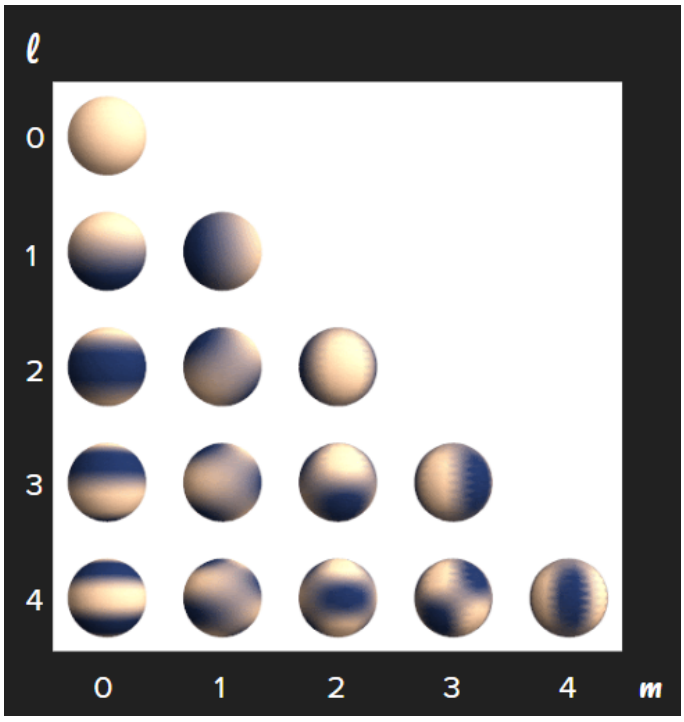

Figure 6: The Y_l^m maps for $l = 0, 1, 2, 3, 4$ with positive m are depicted here [8].

1.5 Motivation

Every exoplanet discovery teaches us something new about how the universe works. In the coming decades there will be direct imaging of many Earth-like planets. These direct imaging discoveries will produce mountains of data in the forms of lightcurves. Lightcurves have all the information required to produce exoplanet albedo maps. The goal of this project is to produce a software that will convert lightcurves into albedo maps. This will be done with the use of previously produced code called *QEARL* which implements analytical solutions for solving the forward problem. The inverse problem will be solved iteratively using a Markov Chain Monte Carlo (MCMC) method.

The motivation for this project stems partially from another software called *exocartographer* which solves the same problem, but numerically with a Gaussian process. The first half of this research project was spent diving into this software, partially to learn about the physics of the problem, as well as to learn how to use the software. On the way, issues were discovered in the code that were analyzed. Both of these motivations are explained further in the following sections.

2 Method and Instrumentation

2.1 Instrumentation

This is solely a coding project, and thus the instrumentation is limited to a computer. Ideally a computer with multiple cores or processors would be used. Multiprocessing will be very useful for running multiple exoplanet lightcurve generations simultaneously, but this was not necessary for the results of this project. An Intel(R) Core(TM) i5-8300H CPU @ 2.30 GHz processor and 8 GB of RAM was used to generate the lightcurves in Figures 14 and 15. The generated Markov chains required to create one lighcurve takes about 1.5 hours to create with this processor and RAM.

2.2 The Forward Problem

The problem in this project can be summarized by the equation:

$$R(t) = \oint A(\theta, \phi) K(\theta, \phi, S, t) d\Omega, \tag{2}$$

which relates the reflected lightcurve, represented by $R(t)$, to the albedo map $A(\theta, \phi)$. $K(\theta, \phi, S, t)$ is the convolutional kernel [10].

Solving this problem is as simple as having an albedo map $A(\theta, \phi)$, assuming properties about the kernel of the exoplanet $K(\theta, \phi, S, t)$, and integrating over all angles to get the lightcurve. Note, however, the huge issue, which is that we do not want to solve the forward problem. The goal of this project is to start with the lightcurve $R(t)$, and solve for an albedo map $A(\theta, \phi)$, the inverse problem.

2.2.1 The Kernel

In general convolutional kernels are used for image processing, for example blurring, sharpening, or edge detection of an image. For exoplanet image processing we assume the surface to reflect light diffusely, exhibiting Lambertian reflection, meaning that light reflecting from the surface has an equal probability of reflecting in every direction with respect to the cosine emission law [12]. This form of reflection can be observed in Figure 7.

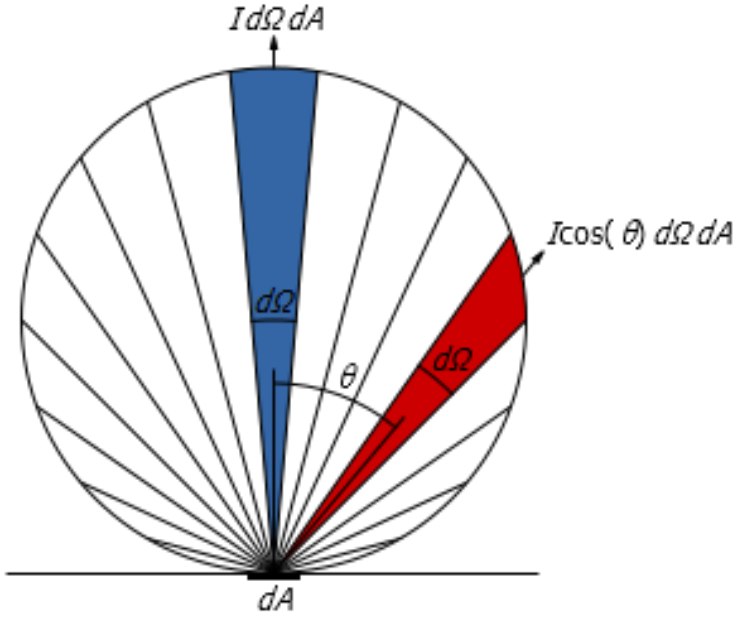

Figure 7: Depicted here are two different angular slices of a circle which photons pass through,being emitted from the area dA. Lambert's cosine law states that the number of photons going through such a solid angle will be proportional to the solid angle area [12].

For Lambertian scattering, the kernel is the product of visibility, $V(\theta, \phi, t)$ and illumination, $I(\theta, \phi, t)$:

$$K(\theta, \phi, t) = \frac{1}{\pi} V(\theta, \phi, t) I(\theta, \phi, t) \tag{3}$$

From the perspective of its host star, an exoplanet would appear as a bright disk. The center of the planetary disk is called the sub-stellar location, and the illumination is unity at this point. If we consider the center to be at angle 0 degrees, and the rim of the planetary disk to be at angle 90 degrees (from the planet core), the illumination falls from unity to zero as it approaches the rim, like a cosine. The illumination is zero behind the disk always. Visibility is defined with respect to the observer in the same way. The center of the exoplanet from our perspective is unity, and at the rim of the exoplanet from our perspective, the visibility drops to zero. We can use the rotation and orbit of an exoplanet as well as eclipses with its host star as information to produce albedo maps of the exoplanet; visibility and illumination depend significantly on these factors [11]. Figure 8 displays how visibility and illumination varies depending on these factors.

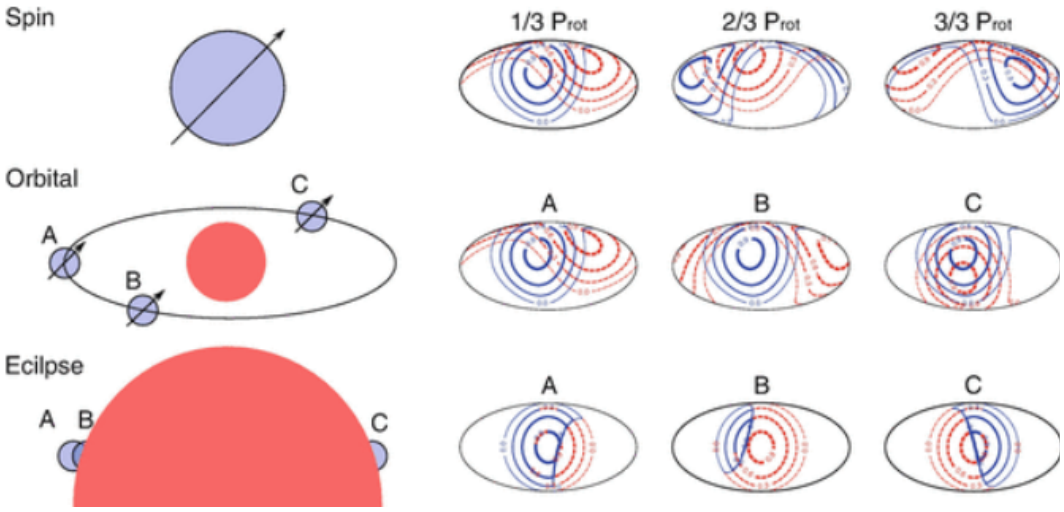

Figure 8: Visibility contours are shown in blue, while illumination contours are shown in red. The spin mapping technique is in the first row, the orbital mapping technique is in the second row, and the eclipse mapping technique is in the third row. Three different scenarios are given for each mapping technique [11].

2.2.2 The Inverse Problem

While the forward problem is to solve for the reflected lightcurve using an albedo map, the inverse problem is to solve for the albedo map given some reflected lightcurve. The forward problem is quite simple, but the inverse problem is much more difficult. This simple algorithm for solving the inverse problem is found in Figure 9. Essentially, generate an albedo map and solve the forward problem repeatedly until a lightcurve is found that is close enough to the given lightcurve.

The tricky part of the inverse problem is finding an efficient method for solving the forward problem so that each iteration does not take too long. The *exocartographer* code solves the inverse problem in a numerical way, while the code created for this report solves it in an analytical way with the use of *QEARL*.

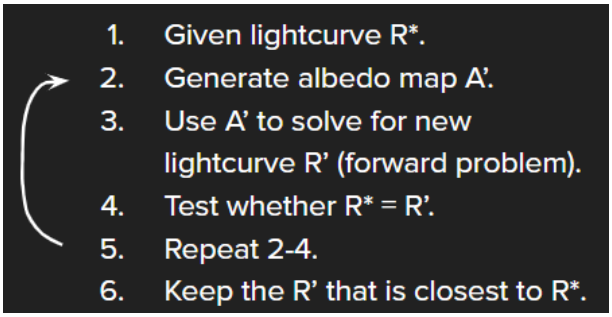

Figure 9: General steps for solving the inverse problem given the conditions of the forward problem in Equation (2).

2.3 Numerical Solution

A numerical solution is an approximation to the solution of a mathematical equation. It is like making guesses at a solution until the solution is close enough to the solution to stop. For the purposes of this project, the numerical solution in *exocarographer* uses a Gaussian process to solve the inverse problem [10].

2.3.1 Gaussian Process

Gaussian processes are quite complicated, hence for the purpose of this report the explanation will be kept brief. Gaussian processes are a powerful machine learning algorithm for both regression and classification tasks. Gaussian processes are very useful for tasks that have a lot of uncertainty. Gaussian processes combine Bayesian inference and machine learning. Our knowledge of some variable can be approximated by some probability distribution. For example, based on previous research on exoplanet observations, the mass of exoplanets would have some distribution; information based on prior knowledge is called a prior belief. When we observe new data, we can update our prior belief to create a posterior belief. These references are very informative for further understanding [13] [14]. A good example of how adding data closes in on an accurate posterior beliefs can be seen in Figure 10.

If we have a lot of information about an observed exoplanet (lightcurve data), we can use that information, alongside prior beliefs about exoplanets, to produce a posterior belief about the appearance (albedo map) of the observed exoplanet.

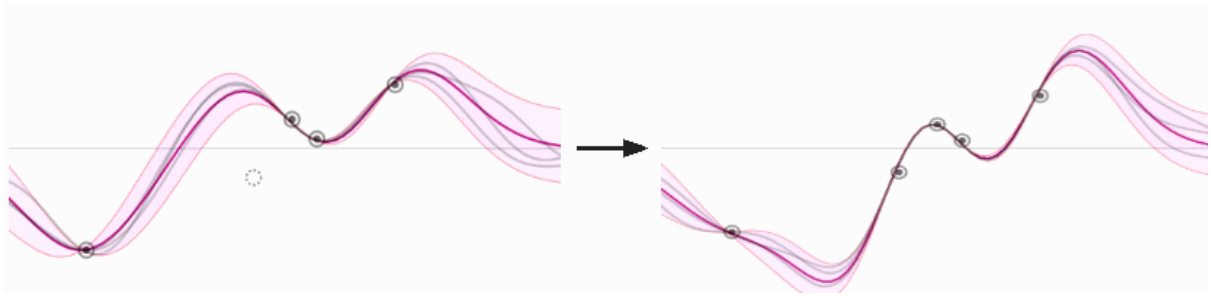

Figure 10: This image shows how adding data changes a prior belief to a posterior belief. The dark purple line is the most probable function. On the left is the prior belief. The light pink area is the most probable area that the function might be (within 1 standard deviation). On the right, a new data point is entered which changes the most likely area, and the most likely function [14].

2.3.2 *exocartographer*

Exocartographer is a Bayesian framework which creates albedo maps from photometric data (lightcurves). In its current state, *exocartographer* is fully based on numerical solutions of solving the inverse problem. The code for *exocartographer* can be found at *https://github.com/bfarr/exocartographer* [10].

2.4 Albedo Map Generation

The first task was to generate a random albedo map with both the HEALPixel and spherical harmonic parameterizations. This task proved to be more challenging than anticipated, but also rewarding for understanding the *exocartographer* code as well as the mathematical and physical semantics driving the code. To understand the difficulties in map generation, an intuition for tinkering with the albedo map constraints must be built.

2.4.1 Albedo Map Constraints

The first constraint set by the Gaussian process is the albedo mean. A good guess would be that if a random albedo map with 192 pixels were generated with the albedo mean constrained to 0.5, then if we averaged the albedo at the 192 pixels, it should be close to 0.5. This is not necessarily true, and can be quite far off in some extreme cases. What this constraint really means is that if an infinitely large number of albedo maps were produced, the mean of those albedo map means would be infinitely close to 0.5. The albedo cannot be set to any value outside the range of $(0, 1)$, and hence the albedo mean cannot be set outside of this range either.

The albedo standard deviation follows a similar definition. If an infinitely large number of albedo maps were produced with albedo standard deviation constraint 0.1, then the mean of these standard deviations would be infinitely close to 0.1, normally, but its actually not, which I will explain shortly. Theoretically the albedo standard deviation could be any value, but since albedos can only have a range of $(0, 1)$, a large standard deviation will cause pixel values outside this range to commonly be generated, and so the map creation will have to constantly restart.

Length scale can be defined as the characteristic angular scale with which two pixels are correlated with each other. In practice, the length scale constraint is used as the largest correlation scale that is observed, for example, on earth the Pacific ocean spans about 135^o of latitude, so this would be a decent constraint on the length scale for Earth. More generally, if the length scale of a map were to be 0 degrees, then every pixel would be drawn from its own Gaussian distribution, and hence the albedo map would look like whitenoise. If the length scale were to be an extremely large value, then things get a bit tricky. The furthest that two points on a sphere can be from each other is 180 degrees, but if a length scale of, for example, 900 degrees were used, then all pixels on the albedo map would be very correlated since they are much closer than 900 degrees. Now the definition I posed for the standard deviation and mean becomes a bit unclear because an albedo map has a correlation length that is much larger than its total size; if an albedo map could be created from a sphere with a 1800 degree arc circumference, then what was stated would be true, but since this is not the case, the albedo range for a generated map with a 900 degree length scale will only be a small subset ($\frac{1}{25}$ of all pixels) of the total range. This should decrease the albedo standard deviation of each map since the range of values will only be a small subset of the whole range, and it should make the albedo mean of each map vary much more for the same reason.

Relative white noise is the last constraint, and it signifies the fraction of the total albedo that is white noise. White noise is added to each pixel, and its value for each pixel is some Gaussian distribution with a mean of 0, and standard deviation dependent on the fraction

of white noise given.

2.5 Analytical Solution

The unfortunate part about Gaussian processes is that they are very computationally expensive. The hope for this project is that the inverse problem can be solved with an analytical solution that is faster than the numerical solution. An analytical solution means involves framing the problem in a well-understood form and calculating the exact solution; for this problem that means solving the forward problem analytically.

For an analytical solution to the inverse problem, the process for solving for an albedo map is still very similar to the algorithm given in Figure 9. In this case, a Markov Chain Monte Carlo simulation (MCMC) is run with analytical solutions. The problem has lightcurve data with errors given. Albedo maps are generated which are solved analytically into lightcurves, and then the χ^2 between the solved lightcurve and given lightcurve are solved. This χ^2 is minimized when the closest albedo map is generated.

2.5.1 *QEARL*

Quaternion exoplanet analytical reflected lightcurves (*QEARL* for short), is the software I use to implement analytical solutions for the forward problem. It is not published anywhere currently, but will be soon.

Figure 11 is a a clear representation of how an exoplanet albedo map is transformed into a lightcurve. Each spherical harmonic has some lightcurve signature called a harmonic lightcurve described in the following equation:

$$F_l^m = \oint K(\theta', \phi', G) Y_l^{m\prime} d\Omega' \quad [15] \tag{4}$$

In this equation, G represents the lune geometry (similar to viewing geometry), and the $Y_l^{m\prime}$ are further described by Legendre polynomials and Wigner D-matrices, all which are applied in the QEARL code, and described in the original paper on analytical reflect lightcurves [15].

QEARL saves some computation time by noting many symmetries such as that complex harmonic lightcurves satisfy:

$$F_l^{-m} = (-1)^m \left[F_l^m\right]^*, \tag{5}$$

and that Wigner D-matrices also have extensive symmetries and satisfy recursion relations [15]. A linear combination of these $F_l^m(t)$ values produce the lightcurve $F(t) = R(t)$. The eigenvalues of this linear combination are the same c_l^m values mentioned in Equation (1). These values values are also complex. By construction, the lightcurve $F(t)$ is real despite being the linear combination of imaginary components. The lightcurve is defined by:

$$F(t) = \sum_l \sum_m c_l^m F_l^m(t) \tag{6}$$

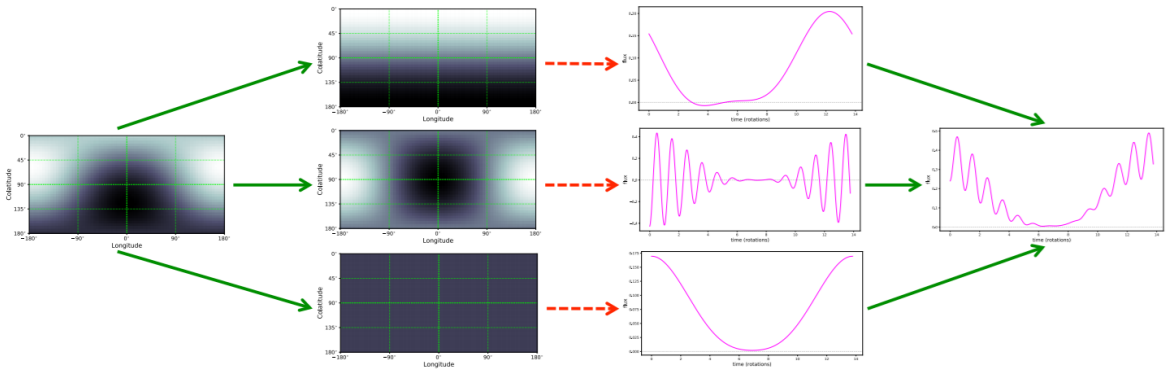

Figure 11: On the far left is some arbitrary generated albedo map. This albedo map can be decomposed into its constituent spherical harmonic basis. Each of these spherical harmonic bases can be transformed into their geometry dependent harmonic lightcurves which is on the right. These constituent lightcurves can be transformed into the exoplanet lightcurve on the far right by a summation. The green arrows represent easy tasks, while the red arrows represent the mathematically challenging task, solved by Hal M. Haggard et al. [15].

2.5.2 Implementation of MCMC

My code solves the inverse problem by applying the *Python* version of *QEARL*, aptly named *PEARL*, with *emcee* a *Python* software that implements a Markov Chain Monte Carlo simulation (MCMC).

The MCMC simulation is a two part system. The Markov chain portion works by iterating through different values of parameters to find the most optimal set of parameters. The most optimal set is defined by the set of parameters that provides a model with the smallest χ^2, where χ^2 is proportional to the sum of residuals of the data and the model values. The Monte Carlo portion of the name implies that there are random deviations from the search for the optimal parameters. Essentially there is a higher probability that the parameter will go towards the optimal value, but there is still a smaller chance that the parameter value will iterate away from the optimal parameter [16].

The parameters that need to be optimized in the MCMC simulation are the viewing geometry required to solve for F_l^m which includes the rotational and orbital angular frequencies of the exoplanet, the inclination and obliquity of the exoplanet, and the solstice phase of the exoplanet [15]. Besides these, all of the c_l^m values are parameters. Since *PEARL* implements imaginary F_l^m values, the c_l^m values are imaginary, meaning the real and imaginary component of each c_l^m is a parameter, this doubles the number of parameters. Fortunately, the number of c_l^m parameters are cut in half again because of the relation:

$$c_l^{-m} = (-1)^m \left[c_l^m\right]^* \tag{7}$$

The maximum value of m is dependent on the maximum value of l, and the maximum value of l, is proportional to the number of pixels on the HEALPixel map. The larger l_{max} is, the more precise the map will be, but also the more costly the simulation.

My code currently works by setting some pseudo-random initial values for the viewing

geometry and c_l^m parameters, which are consistent with normal values. These viewing geometry parameters, the l_{max}, and a list of times are plugged into *PEARL*, and it returns a list of $F_l^m(t)$ values. This is then used to solve for the initial $F^*(t)$ lightcurve by solving Equation (6). I then added error to each data point with a value $\frac{1}{100} * max\big[F(t)\big]$. I replicated this twice, so that there were two different initial lightcurves. I proceeded to use these two initial lightcurves in two different ways. I wanted to check how well the MCMC code could find the optimal set of parameters if the initial parameters were correct, and how well the MCMC code could find the optimal set of parameters if the initial parameters were completely random. I took the correct lightcurve values with their respective errors and the parameters, and passed this into a standard *emcee* formula. The MCMC simulation iterates about parameter values, solving the *PEARL* code, and then solving for a new $F'(t)$ at every iteration. The χ^2 (which is a function of the difference in $F'(t)$ and $F^*(t)$ as well as the errors on the lightcurve values) must be minimized. After 500 iterations, the code stopped, and the optimal set of parameters is returned. This new optimal set of parameters solves the optimal lightcurve $F''(t)$. The optimal set of c_l^m values are converted into an albedo map to see how accurate the simulations are to producing the correct albedo map.

3 Results and Discussion

3.1 Numerical Solution Map Generation Differences

There are clear differences between the corresponding albedo maps generated in Figure 12, this signifies that one of the map generation methods must be incorrect (or maybe both). All of the plots in Figure 12 have 0 whitenoise, which means that no white noise should be evident. The 5 degree length scale albedo map for the spherical harmonic implementation looks something like white noise, but this appearance is just due to the very small length scale which makes only very nearby points correlated. The 180 degree length scale albedo map for the pixel implementation also looks like it has white noise. Another thing to note is the median albedo for each map which is also consistent with what was previously introduced. At larger length scales, the mean/median of the albedo values finds itself further from 0.5 than smaller length scales, and the standard deviation is much smaller.

Most of, if not all issues that can be spotted relate to the length scale. First off, by contrasting the 5 degree length scale albedo maps, it is quite clear that patches of albedo with similar values are much larger for the pixel implementation. The same can certainly be said for the 30 degree length scale implementation. An argument could be made that the 180 degree length scale implementations look quite similar in terms of the size of the patches of albedo, but the smoothness of the spherical harmonic implementation contrasted with the whitenoise of the pixel implementation is very noticeable. After much thought, there isn't a good reason that white noise should be prevalent at large length scales, so this seems to be an issue with the pixel implementation. The difference in the number of patches is also likely an issue with the pixel implementation. By counting the number of patches of albedo from the top to bottom of the 30 degree length scale map, the pixel implementation has 2 or 3 patches, while the spherical harmonic implementation has 6 or 7 patches; since the

angle between the top and bottom of an albedo map is 180 degrees, the spherical harmonic implementation has ~ 30 degrees per patch, which is the length scale.

To see how the mean and standard deviation of albedo maps vary with the length scale for both implementations, 100 albedo maps were generated for a multitude of length scales varying from 20 to 1000 degrees. The average mean and average standard deviation of the 100 albedo maps was inspected for each of the length scales. There is a very interesting trend for the standard deviations. The average standard deviations as a function of length scale can be found in Figure 13. Visually it can be seen that the pixel standard deviations drop quickly as the length scale increases, but at about 500 degrees (between 2.5 and 3 on the x-axis), the spherical harmonic standard deviations catch up and pass beneath in value. After reading through the two code implementations, it seems that the values that go into the covariance matrices may have different dependencies on the length scale. The pixel implementation seems to have values scaling as a negative squared exponential, while the spherical harmonic function eigenvalues (sum over m values) c_l in the code, have values that form a softmax relation as a function of the length scale; a softmax function is similar in behavior to a logistic. These functions are consistent with the behaviour in Figure 13.

3.2 Analytical Solution Lightcurve & Albedo Map Generation

I have presented two different scenarios for solving the inverse problem. One scenario has correct initial values input into the MCMC simulation, while the second scenario has incorrect (pseudo-random) initial values input into the MCMC simulation. Note that the first scenario has one unique lightcurve, while the second scenario has two unique lightcurves. This is because in the first scenario, the parameters that form the lightcurve used to create the data are the same as the parameters which form the lightcurve that are input into the MCMC simulation. In the second scenario, these parameter sets are completely different. For both scenarios, the MCMC simulation over 500 iterations generates an optimal set of parameters which creates its own lightcurve. The two lightcurves of the first scenario are seen in Figure 14, and the three lightcurves of the second scenario are seen in Figure 15. The parameter sets of all 5 lightcurves were then combined with the viewing geometry, which forms the Y_l^ms, to create the albedo maps $A(\theta, \phi)$, as described in Equation (1). Figure 16 has the albedo maps for scenario 1, and Figure 17 has the albedo maps for scenario 2. I did not add the albedo map related to the initial parameters of the MCMC simulation onto Figure 17 as it is just an initialization and should not look similar to either of the other two maps (green line in Figure 15).

3.2.1 Discussion of Lightcurves & Albedo Maps

The green line in Figure 15 is simply to help visualize how different the lightcurve initial parameters were for the second simulation. The other lightcurves in both Figures 14 and 15 are never separated by much more than one standard deviation. Overall, this appears as a success for the MCMC simulation, producing a relatively consistent lightcurve no matter whether the initial parameters are the correct values, or other randomly generated ones. It should be noted, however, that the pseudo-random initial lightcurve (green) in Figure 15 is not that different from the others; from other simulations I noticed that the curve could have

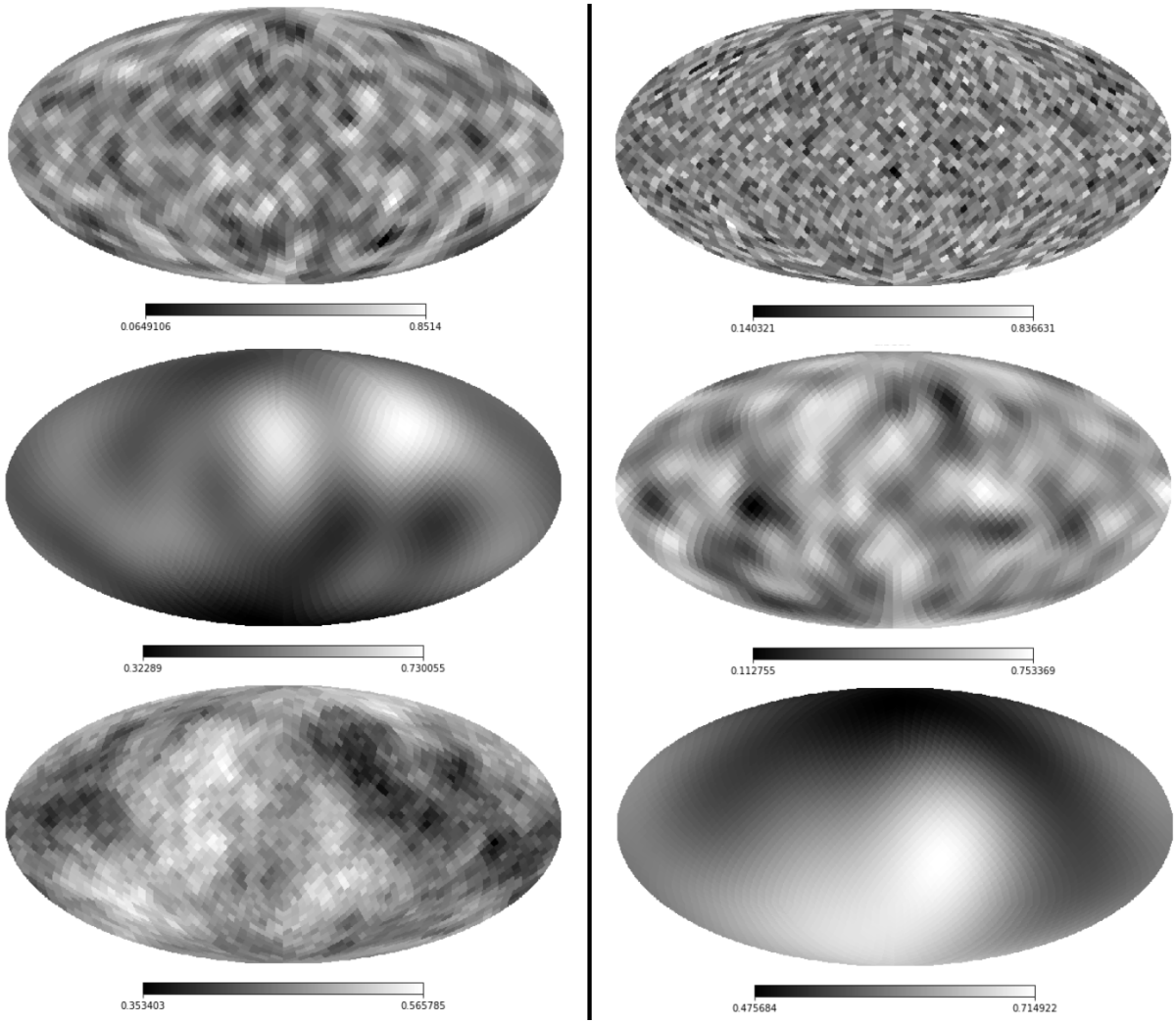

Figure 12: 6 albedo maps appear in this figure. Every map has an albedo mean of 0.5, and an albedo standard deviation of 0.2. In the left column are maps generated with the pixel implementation, and in the right column are maps generated with the spherical harmonic implementation. In the first row are maps generated with a length scale of 5 degrees. In the middle row are maps generated with a length scale of 30 degrees. In the last row are maps generated with a length scale of 180 degrees.

taken on a variety of different shapes, i.e. the green line could have been at a minimum flux at a time of 0 days and a maximum flux at a time of 180 days. For MCMC simulations with initial parameter sets that produce much different lightcurves, I found that it took upwards of 1000 iterations to generate a very similar lightcurve like the orange one found in Figure 15 (as opposed to 500 iterations for this simulation).

At first glance there are a few aspects within the results from the albedo maps that appear to be non-ideal, but this is simply deceiving because of the color scale. The color scale difference isn't as great for Figure 16 as Figure 17 where the maximum albedo has a disparity of 0.13. The range of albedos in the "given" (top) albedo map of Figure 17 is 0.38,

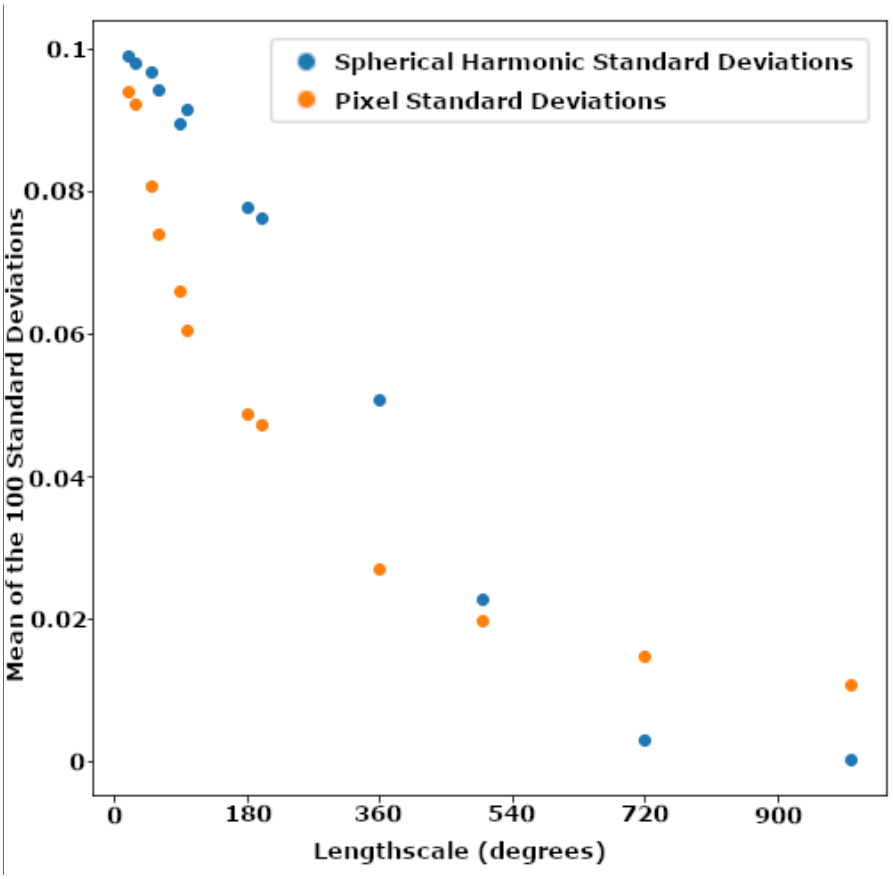

Figure 13: On the x-axis is the length scale Gaussian process constraint; this value is labeled in degrees. On the y-axis is the mean of 100 albedo map standard deviations. The HEALPixel implementation is in orange while the spherical harmonic implementation is in blue.

so a difference of 0.13 does skew the observers opinion, but the result is still quite different. With a detailed inspection, I have determined that in Figure 16 there are 7 out of 12 pixels which are close in value (<0.06 albedo difference), while in Figure 17 there are about 5 out of 12 pixels which are close in value. The generated albedo maps in Figure 16, thus, are mostly inconsistent with the true albedo map values. The generated albedo maps in Figure 17 do not match the true albedo map values well at all. The difference in quality between the first and second simulation make sense due to the way the MCMC simulation was initialized, although this does not agree with the similarity in the lightcurves previously mentioned.

There are a few reasons that I believe that the albedo maps do not match well in either Figure 16 or 17. To save time on running simulation experiments, I limited the number of data points to only 30 for the whole lightcurve. This amount of data is quite small, and is similar to the number of parameters, thus an interesting aspect to explore in further work would be to increase the number of data points, it is possible that the lightcurves would match the expected curve much better, and thus the set of parameters (which doesn't change with the number of data points) would match better. It's also possible that running the simulation for a long enough of a time could provide even better results, but this is not very likely as the optimal set of parameters stopped changing far before the last iteration (about

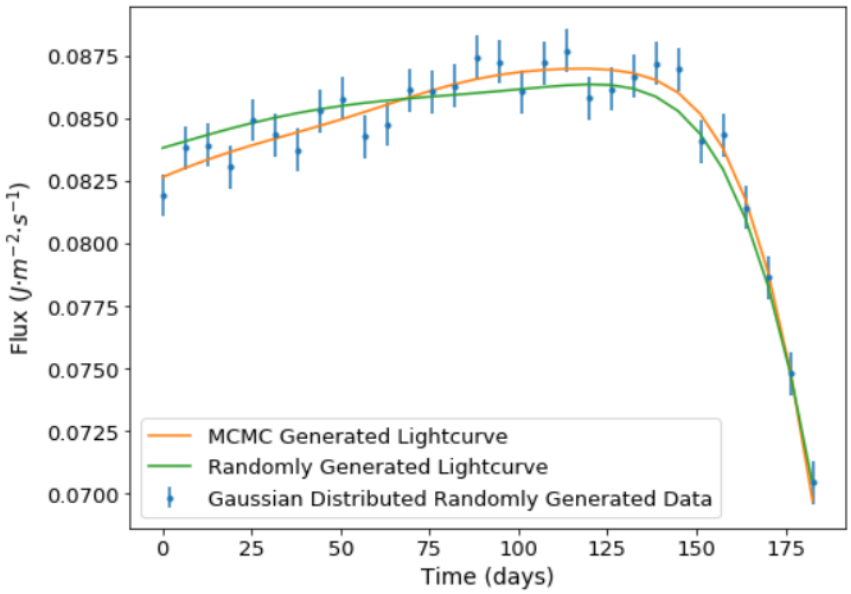

Figure 14: The green line is the pseudo-randomly generated lightcurve that is the "given" lightcurve $R^*(t) = F^*(t)$. The blue data is Gaussian distributed above and below this lightcurve according to their errorbars. The orange line is the lightcurve generated by the MCMC simulation. This direct imaging event lasts 180 days.

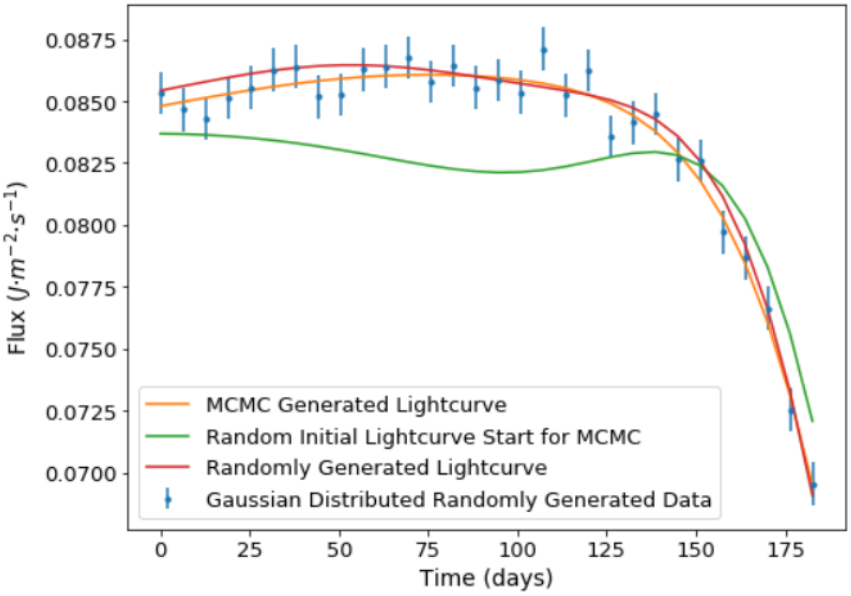

Figure 15: The red line is the pseudo-randomly generated lightcurve that is the "given" lightcurve $R^*(t) = F^*(t)$. The blue data is Gaussian distributed above and below this lightcurve according to their errorbars. The orange line is the lightcurve generated by the MCMC simulation. The green line is the pseudo-randomly generated lightcurve that represents the parameters that were input into the MCMC simulation to start off its iterations. This direct imaging event lasts 180 days.

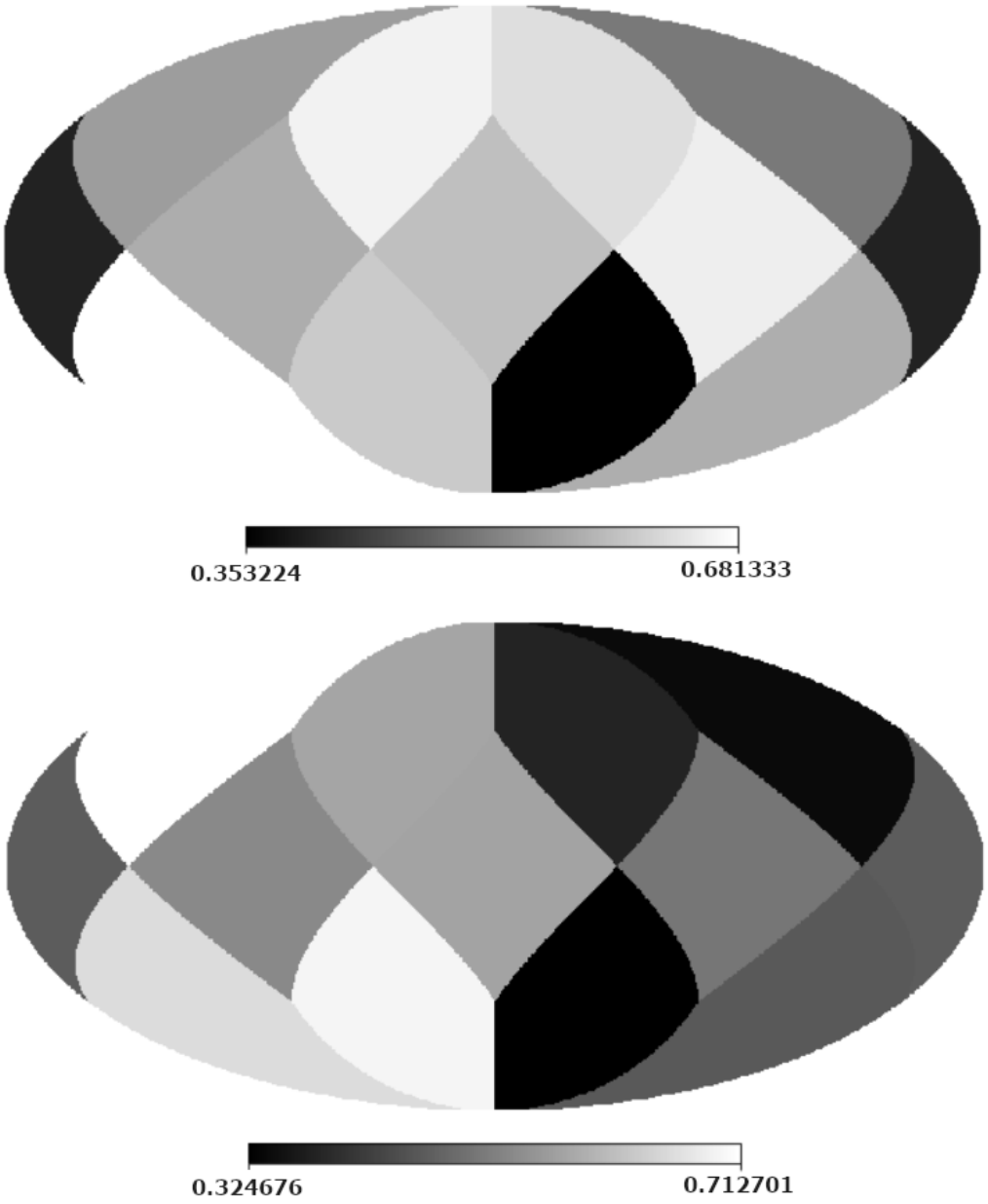

Figure 16: On the top is the albedo map generated from the pseudo-randomly generated initial lightcurve that produced the data. On the bottom is the albedo map generated from the lightcurve that was generated by the MCMC simulation. Note the minimum albedo on the color scale is different by approximately 0.03, while the maximum albedo is different by approximately 0.03.

steps before the end of the chain). Finally, one last aspect that could be improved within further work is to use Markov chains differently. I pulled out the set of parameters that had the minimum χ^2 value, but I think it could be worthwhile to look into the use of the median (or mean) value of each parameter; the optimal (minimum χ^2) set of parameters could have a parameter with an outlier that severly visually changes the albedo map.

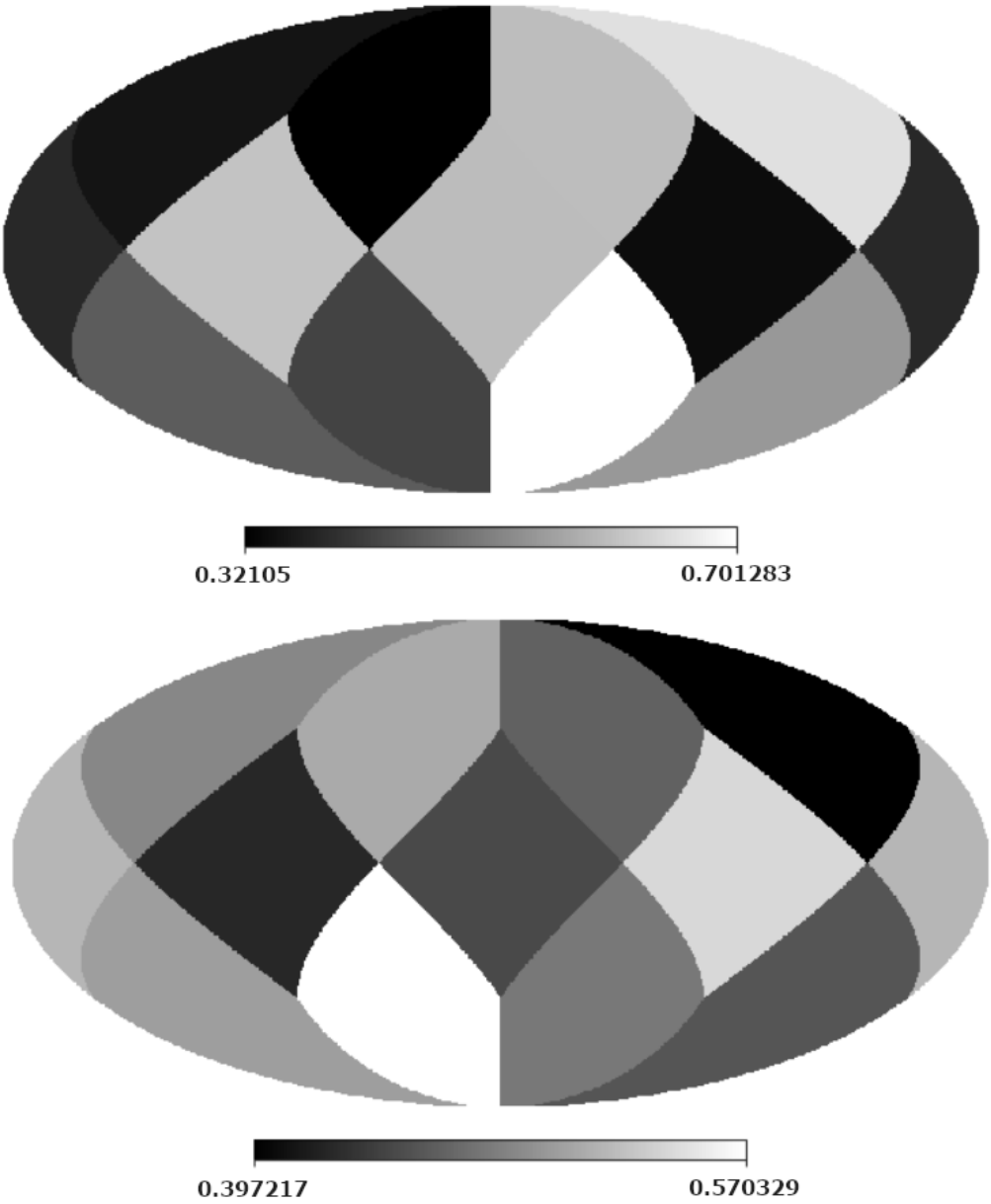

Figure 17: On the top is the albedo map generated from the pseudo-randomly generated initial lightcurve that produced the data. On the bottom is the albedo map generated from the lightcurve that was generated by the MCMC simulation. Note the minimum albedo on the color scale is different by approximately 0.08, while the maximum albedo is different by approximately 0.13. The differences are much more different in this plot.

4 Conclusion

For the numerical solutions to the inverse problem, there seems to be an issue with the pixel implementation of the albedo map creation. This is suspected to be due to the spherical harmonic implementation having albedo patch widths consistent with the length scale, while

the pixel implementation does not. Additionally, there will be a challenge with the length scale at larger values for the pixel implementation in the sense that white noise seems to be introduced. Additionally, the behaviour of the average standard deviations are not consistent between the two implementations, leading me to believe that the HEALPixel implementation has issues.

For the analytical solutions to the inverse problem, the code was created successfully, and the plots were generated successfully as well. The MCMC simulation produced lightcurves that are consistently within one standard deviation from the true lightcurves, and it did not matter whether the initial set of parameters passed into the MCMC simulation was far from the true values. The albedo maps produced from the lightcurves generated by the MCMC simulation were not very consitent with the true albedo maps. This could be due to the optimal set of parameters being used rather than the median set of parameters from the Markov chain or that the number of data points used in the simulation was too low to account for random error in the parameter values.

References

[1] "Your Guide to Exoplanets." The Planetary Society Blog, www.planetary.org/explore/space-topics/exoplanets/.

[2] HR 8799 Orbiting Exoplanets. In Wikipedia. Retrieved March 06, 2020, from https://commons.wikimedia.org/wiki/File:HR_8799_Orbiting_Exoplanets.webm.

[3] Chen, Rick. "Exoplanet Populations." NASA, NASA, 16 June 2017, www.nasa.gov/image-feature/ames/kepler/exoplanet-populations.

[4] "Principles of Photometry." Handbook of Practical Astronomy, by Roth Gunter D., Springer, 2009, pp. 205–238.

[5] Deeg, Hans. "Planet Transit." ESA, European Space Agency, 13 June 2003, www.esa.int/ESA_Multimedia/Images/2003/06/Planet_transit.

[6] Gaudi , S., and D. Bennett. "Exploring Exoplanets with Microlensing." NASA Exoplanet Science Institute, 25 July 2011, nexsci.caltech.edu/workshop/2011/.

[7] Gorski, et al. "The HEALPix Primer." ArXiv.org, 23 May 1999, arxiv.org/abs/astro-ph/9905275.

[8] Spherical Harmonics. In Wikipedia. Retrieved March 06, 2020, from https://en.wikipedia.org/wiki/Spherical_harmonics/media/File:Rotating_spherical_harmonics.gif

[9] Spherical Harmonics. In Wikipedia. Retrieved March 06, 2020, from https://en.wikipedia.org/wiki/Spherical_harmonics

[10] Farr, Ben, et al. "Exocartographer: A Bayesian Framework for Mapping Exoplanets in Reflected Light." The Astronomical Journal, vol. 156, no. 4, 2018, p. 146., doi:10.3847/1538-3881/aad775.

[11] Cowan, et al. "Mapping Exoplanets." ArXiv.org, 25 July 2017, arxiv.org/abs/1704.07832.

[12] Pedrotti, Frank L., et al. Introduction to Optics. Cambridge University Press, 2018.

[13] Knagg, Oscar. "An Intuitive Guide to Gaussian Processes." Medium, Towards Data Science, 15 Jan. 2019, towardsdatascience.com/an-intuitive-guide-to-gaussian-processes-ec2f0b45c71d.

[14] Görtler, Jochen, et al. "A Visual Exploration of Gaussian Processes." Distill, 13 Nov. 2019, distill.pub/2019/visual-exploration-gaussian-processes/.

[15] Haggard, Hal M, and Nicolas B Cowan. "Analytic Reflected Light Curves for Exoplanets." Monthly Notices of the Royal Astronomical Society, vol. 478, no. 1, 2018, pp. 371–385., doi:10.1093/mnras/sty1019.

[16] Karolinski, N. Zhu, W., Isolated black holes from the absence of parallax effect in microlensing, in preparation.