Draft version March 18, 2013
Preprint typeset using LaTeX style emulateapj v. 11/10/09

ECLIPSE MAPPING CODE

Woody Austin

Draft version March 18, 2013

ABSTRACT

pass

1. SET-UP AND TERMINOLOGY

The eclipse mapping program described in this paper derives a model for the relative surface brightness of a transiting planet host star when given a single band, short cadence light curve of the target as an input. Each run of the program produces a static map of the stellar surface. By applying the code to a series of short segments of the light curve, we are able to extract information about the time evolution of the star's surface brightness.

In order for the program to create a brightness map, the physical properties of the star and the planet must be provided. Stellar rotation period, temperature of the star, orbital period, orbital separation, impact parameter, and transit duration are all required in order to produce an adequate model of the physical system. The star is assumed to be rotating as a uniform solid body. In addition, we require the spin-axis of the star to be aligned with the orbital axis of the planet and the planet to be on an approximately circular orbit. These criteria are both met for our test object, Kepler-17 .

Other inputs to the code determine how the model is able to fit the data. Choices include the binning cadence for both in and out-of-transit points of the light curve, the number of boxes and stripes there should be in the surface map, number of days per individual brightness solution or *window*, the number of days to increment each window, and which type of data to use (PDCSAP or SAP).

In order to efficiently describe the details of our program, the geometry of the problem must be defined and some basic terminology must be established. We divide the stellar surface into a series of large *regions*, as shown in Figure **??**. While describing our program in this paper, the regions along the line of transit will be called *boxes*; the total longitudinal regions will be called *longitudes*; and the longitudinal regions with the box areas subtracted will be referred to as *stripes*. When talked about as an ensemble or when the distinction is unimportant, these regions will be referred to as just that - *regions*.

Our program calculates the model flux at each time step, by summing over the surface of the visible sphere according to:

$$f_{mod,i} = \sum_j V_{i,j} b_j, \tag{1}$$

where b_j is the brightness per unit area for region j and $V_{i,j}$ is the *visibility* of that region at time, i.

This model flux has two parts: a visibility and a brightness value. The amoeba determines the brightness values while the visibility values are pre-defined as described in section 1.2. Both of these apply to a set of j regions. The planet will occlude only the boxes and will do so only during a transit. The longitudes' brightness values are defined as:

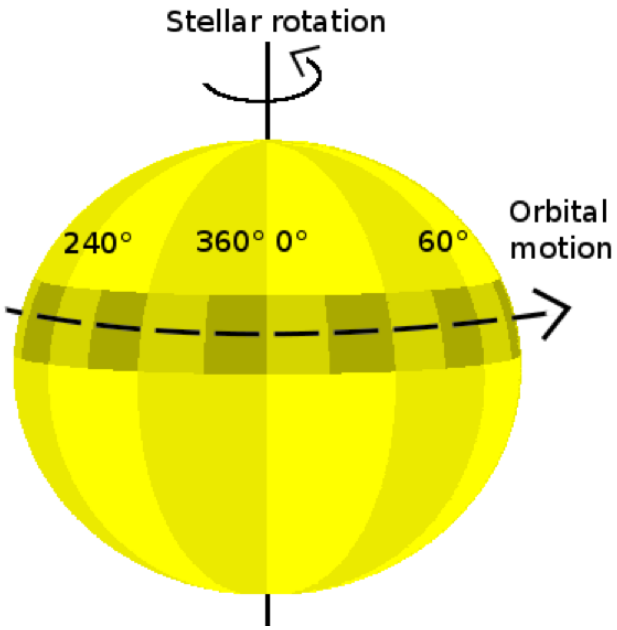

Figure 1. CoRoT brightness map.

$$Z_j = \frac{S_j - \frac{c}{q}\sum_{j=1}^{q} B_j}{1-c} \tag{2}$$

Where q is the ratio of n_{boxes} to $n_{stripes}$ and c is the ratio of the total eclipsed area to the non-eclipsed area. c can be calculated by the same set of integrals that will be used for the visibilities. If the Amoeba algorithm is given the set of boxes and stripes, they will be able to vary to offset each other creating a "zebra" pattern in the brightness map. The longitude values from equation 2 create parameter independence by warping the chi-squared space so that such a case is less likely to become a local minimum. The Amoeba algorithm takes the set of box brightness guesses and longitude brightness guesses as inputs. During each call of the chi-squared algorithm the stripe values are calculated from the longitudes and boxes. The boxes and the stripes are then used to calculate the model flux.

1.1. *Model Flux*

The flux of the star at any given time i is given by $f_{mod,i} = \sum V_{i,j} b_j$. Each region on the star has some visibility value $V_{i,j}$ that varies with time. These visibility values are all governed by the amount of projected area that is seen by the observer over time and can be calculated via an analytical integral. Finding the brightness values, b_j, is the goal of the program. For a given set of data the brightness values cannot change using our technique. To remedy this, each month (or quarter) of data is divided into smaller intervals called *windows*. By moving the start and end times of the windows by small amounts, information about the evolution of starspots can be inferred. Previous estimates on the timescale of starspot evolution r eport that it occurs on the order of 10-30 days. This works well with the model described here. The model does a better job of fitting smaller sections of data than larger ones.

The set of brightness values is optimized by minimizing the chi-squared:

$$\chi^2 = \sum \frac{(f_{mod,i} - f_{obs,i})^2}{\sigma_i^2} \tag{3}$$

The Amoeba Algorithm is used to do the actual minimization. The algorithm runs quickly and will always converge given any set of inputs. The largest downside (of most optimization algorithms) is that it cannot distinguish between local and absolute minima. This becomes especially apparent when choosing an initial set of brightness guesses for the simplex structure. Because the average brightness of the star is 1.0, the initial conditions for the simplex structure are $n_{sb}+1$ brightness sets of n_{sb} values where $n_{sb} = n_{stripes} + n_{boxes}$ and every element is in the simplex is 1.0. A different (orthogonal) basis vector e_i times a scale factor, s, is then added to each of the $n_{sb} + 1$ sets. When the scale factor is too large, the Amoeba converges on answers that are not likely real. When the scale factor is too small or non-existent, the Amoeba does not do any work as the relative difference between successive χ^2 calculations is small despite having not converged on an actual minimum.

1.2. *Visibility Calculations of Regions*

Recall that in order to calculate $f_{mod,i}$, the visibility of each region at a given time step is required. These visibilities are calculated by finding the projected surface area of a sphere for a given set of angles. The basic integral is the spherical surface integral dotted with the unit vector, $\hat{x}$, along the line-of-sight:

$$V_{i,j} = \int_{\phi_1}^{\phi_2} \int_{\theta_1}^{\theta_2} \sin^2\theta \cos\phi \, \mathrm{d}\theta \, \mathrm{d}\phi \tag{4}$$

Where ϕ_1, ϕ_2, θ_1, and θ_2 are shown in Figure 2. For all types of visibilities, ϕ_1 and ϕ_2 are set by the number of stripes or boxes defined at the start of the program. The θ limits, however, will be different depending on region type. The θ limits for the boxes are set by the impact parameter and the radius of the planet. The limits will be equal to $(b \pm R_p) + \frac{\pi}{2}$. The addition of $\frac{\pi}{2}$ is necessary

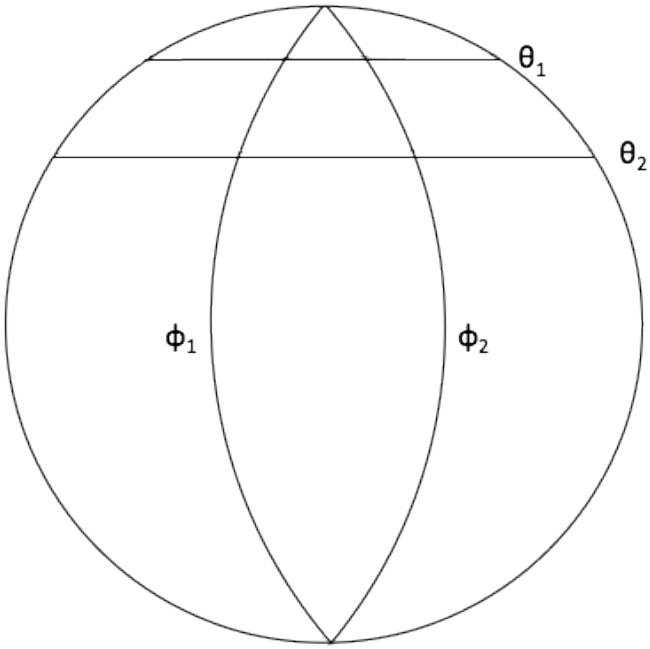

Figure 2. Angles.

when the bottom of the star is defined as latitude 0. For the visibility of a longitude, θ will range from 0 to π. The visibility of a stripe is calculated (in the chi-squared routine) by subtracting the sum of the box visibilities in a given longitude range from the longitude in the same range. A typical box visibility curve is shown in Figure 4. The sum of every region over time is shown in Figure 3.

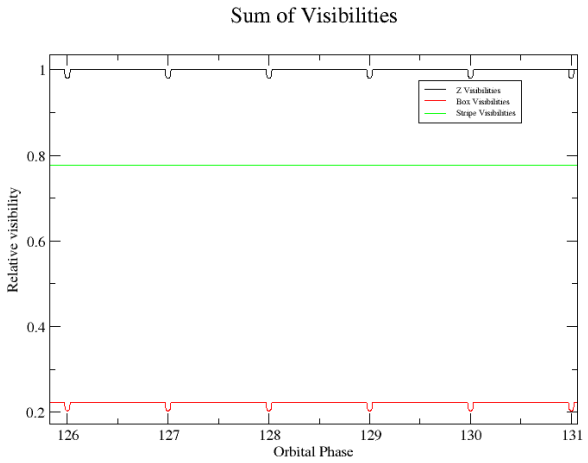

Figure 3. Sum of visibilities over time

1.3. *The transiting planet model*

To actually model the starspot features within the transit portions of the light curve, a reasonable understanding of the visibility profile of a transit for any given

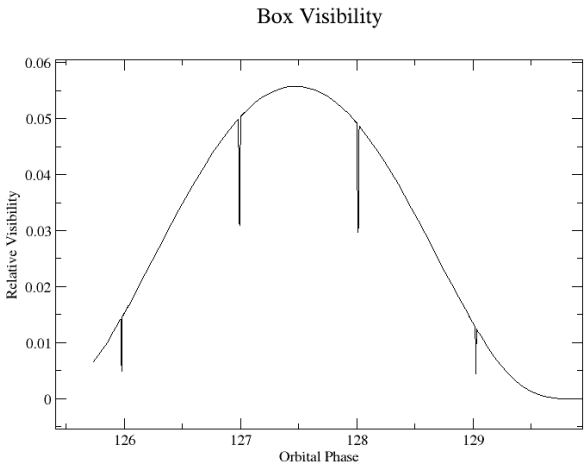

Figure 4. Box visibility curve

system is required. The visibility of the transits is determined by the portion of the star that the planet occludes during transit and some limb darkening effects. As discussed earlier, the planet will only transit along the boxes contained within the latitudes at $\pm r_p$ from the impact parameter, b.

The first step in determining the visibility occluded by a planet at a given time step involves simple geometry which will then be extended to the ten cases contained in Table 1. The simplifying approximation that each of the boxes is a cartesian rectangle when it is projected onto the surface of the star is made. Although this is not entirely accurate, it is a good approximation because when it overestimates a portion of the box, it must also underestimate a similar but not necessarily equal portion of the box.

The simplest case in which the planet is entirely contained within a box is:

$$A_{occluded} = \pi r_p^2 \tag{5}$$

The planet can also be partially contained in a box in many ways, shown in Table 1. The sliver of the planet that is not within the box will be calculated and then extended to take care of all of the various cases. To do this, the area of a sector of a circle is found and then the triangle that is formed within it is subtracted. The sector is the combination of the light blue and yellow regions in Figure 5.

$$A_{sector} = \frac{r_p \alpha^2}{2} \tag{6}$$

where:

$$\alpha = 2\cos^{-1}\left(\frac{r'}{r_p}\right) \tag{7}$$

The area of the triangle is cr'. c is one half the length of the chord in the figure. c can be found with the Pythagorean theorem. Knowing this, the area of the triangle becomes:

$$A_{triangle} = r'\sqrt{r_p^2 - r'^2} \tag{8}$$

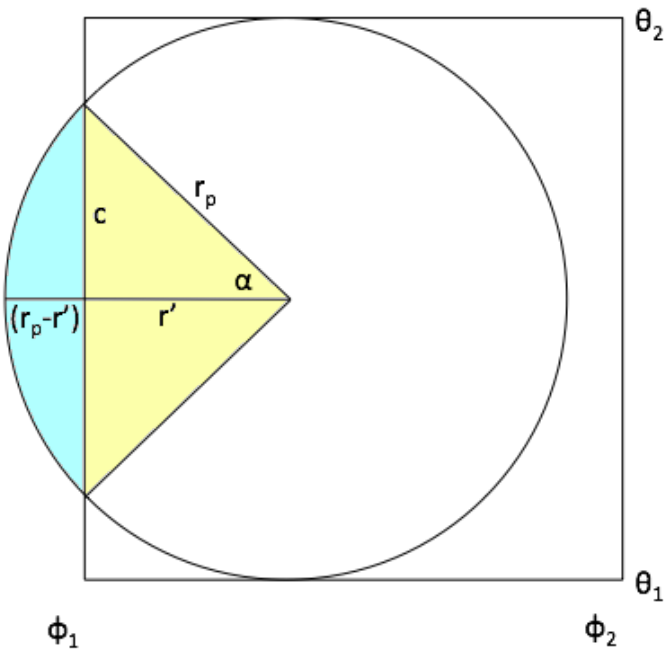

Figure 5. Geometry of eclipse path visibility.

Finally, the area of the segment that indicates the part of the planet just over the border of the box is:

$$A_{seg} = r_p^2 \cos^{-1}\left(\frac{r'}{r_p}\right) - r'\sqrt{r_p^2 - r'^2} \tag{9}$$

Using A_{seg} and Table 1, a believable box-car transit model is produced. However, limb darkening must also be included in order to give a better approximation to the actual shape of a real transit. Limb darkening is introduced in the visibility rather than in the model flux calculation. This is okay because $f_{mod,i} = V_{i,j} bj$. If limb-darkening were calculated during every flux calculation of every chi-squared call during the Amoeba algorithm run, it would require much more operational complexity in the code. However, limb-darkening can be calculated before the many chi-squared calls via the Amoeba algorithm and and avoid this. The geometric analysis of the transit and limb darkening is continued in the next section.

1.4. *Limb Darkening*

To actually calculate the limb-darkening, a quadratic limb-darkening law is used.

$$\frac{I(\psi)}{I(0)} = 1 - c_1(1-\mu) - c_2(1-\mu)^2 \tag{10}$$

Here, μ is the cosine of the angle between the line of sight and the normal at the point at which you are trying to calculate the limb darkening. Because stars are so distant, this can be approximated as $\hat{x} = R^* \sin\theta\cos\phi$, but R^* is defined as 1.0 throughout all calculations.

To complete the limb darkening model, the average value in one of the regions is calculated at every timestep. This is done analytically from the integral:

Table 1
Subcases for planetary eclipse visibility

Case	Description	Area covered by planet in box
I	Planet is completely out of the box to the right	0
II	Planet is completely out of the box to the left	0
III	Planet is completely contained in the box	πr_p^2
IV	Planet is partially off the right side of the box. Center is inside of the box	$\pi r_p^2 - A_{seg,l}$
V	Planet is partially off the right side of the box. Center is outside of the box	A_{seg}
VI	Planet is partially off the left side of the box. Center is inside of the box	$\pi r_p^2 - A_{seg,r}$
VII	Planet is partially off the left side of the box. Center is outside of the box	A_{seg}
VIII	Planet is partially off both sides of the box. Center is inside of the box	$\pi r_p^2 - A_{seg,r} - A_{seg,l}$
IX	Planet is partially off the right side of the box. Center is outside of the box to the left	$A_{seg,l1} - A_{seg,l2}$
X	Planet is partially off the left side of the box. Center is outside of the box to the right	$A_{seg,r1} - A_{seg,r2}$

$$\frac{1}{V_{i,j}} \int_{\phi_1}^{\phi_2} \int_{\theta_1}^{\theta_2} (1 - c_1(1 - \sin\theta\cos\phi) - c_2(1 - \sin\theta\cos\phi)^2) \sin\theta\cos\phi \, \mathrm{d}\theta \, \mathrm{d}\phi \tag{11}$$

This works well for limb-darkening determination in each region, but care must be taken when calculating limb darkening for the eclipse. Rather than calculating equations of limb darkening for the eclipse directly as in Mandol & Agol 2000, the ratio of the limb-darkened eclipsed region to the non-limb-darkened eclipsed region is used. If the number of regions in eclipse is not high enough, a boxy limb-darkening approximation to the transit path is produced using this scheme. This happens because the planet is moving too quickly relative to the surface of the star given the in-transit binning cadence. To fix this, the ratio of the limb-darkened to non-limb-darkened regions is calculated for multiple sub-regions. This gives a finer grain approximation to the real value.

2. VALIDATING OUR MODEL

Theoretically, the model should reproduce the light curves with good precision, however degenracies exist in the solutions. Trying to extract a two dimensional brightness map from a one dimensional light curve is somewhat tricky. That the model light curve fits the data and that the brightness values are correct must be verified. This cannot be done with real data. Only half of the necessary information can be proved correct. There is no way of knowing *a priori* what the brightness values of a real system should be at any given time. To remedy this, we produce synthetic light curves using $f_{mod,i}$defined in Section 1.1. This technique will be described in greater detail below.

While trying to verify the models, the only things that can vary for the reproduction is the number of boxes and stripes and the binning cadences. The number of boxes and stripes are set to six and eighteen respectively. The in and out-of-transit binning cadences are chosen to mimic a real system. Because so many points exist out of transit and the overall trends tend to exist longer term a low cadence is desired for out-of-transit binning. For these example solves, the out-of-transit binning cadence is 95 Kepler short-cadence time steps per bin. The in-transit binning cadence is set at a higher frequency in order to remain sensitive to small scale variations in the light curves. For these example solves, the in-transit binning cadence is set to 3 Kepler short-cadence time steps per bin.

2.1. *Producing the model light curves*

To produce the model light curves, the model flux $f_{mod,i} = V_{i,j} b_j$ was used. The visibility calculations are the same as described in section 1.2. Rather than running the Amoeba algorithm to determine a best fit to an existing light curve, a predetermined set of brightness guesses $B = \{b_1, b_2, ..., b_j\}$ is used to produce a new light curve. The light curves that were used to test the code had stellar and planetary parameters similar to that of Kepler 17. P_{rot}, P_{orb}, R_p, orbital separation, limb darkening (temperature), and transit width are all identical to the Kepler 17 system. The light curves were produced using 6 stripes and 18 boxes. To simulate spots, certain regions are darkened more than the default brightness value of 1.0. When the light curves are produced, there is no noise. To remedy this, a simple python script generates Gaussian noise based on a Kepler-Magnitude input.

The noise for these models was based off of error counts similar to a given magnitude Kepler star. Our tests included models with no noise, simulated 12th magnitude noise, and simulated 14th magnitude noise. Figure 6 shows a *window* of our model light curve with the three levels of noise for comparison.

2.2. *Testing Complexity*

The ability of the program to solve for brightness values has only been tested in two parameter spaces, complexity of the spots and noise of the light curve. The noise of the curve is generated by a python script based on Kepler Magnitude noises. Models were tested for a noiseless, a 12th magnitude star, and a 14th magnitude star. There is no noticeable difference between these. The complexity of the spots is determined by the number of regions that vary from the default brightness value of 1.0. The model has been tested on 11 different spot configurations. These are one box, one box and one stripe, two boxes, two boxes and one stripe, two boxes and two

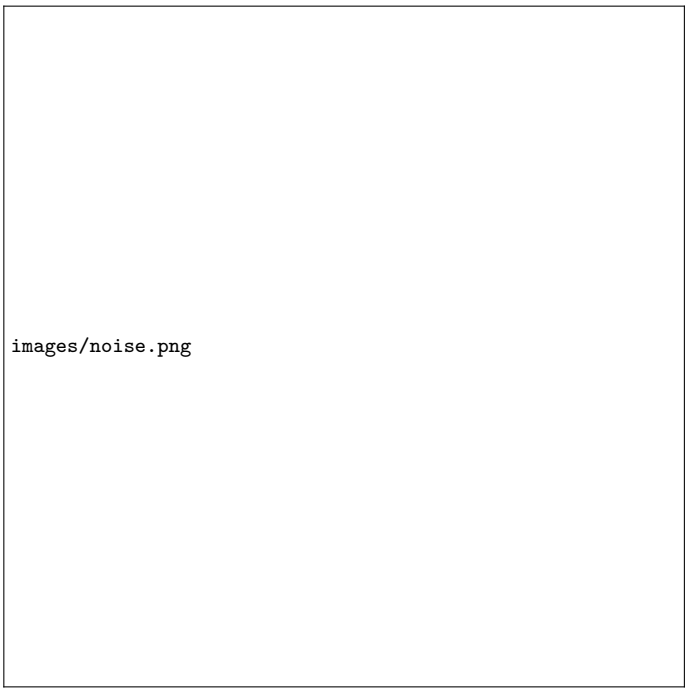

Figure 6. Comparison of noise of models.

stripes, three boxes and one stripe, three boxes and two stripes, three boxes and three stripes, four boxes and three stripes, all boxes randomly and two stripes, and all regions randomly darkened.

2.3. *Results*

Overall, the program does a good job of recovering the input brightness values and matching the light curves fairly independent of the level of noise. One important thing to note is that the program does not recover the correct brightness values every time, but does a good job when averaged over many windows. This means that the results are better interpreted as long term trends than as exactly correct in every window. For the synthetic light curves, starspot evolution was not introduced. However, in real systems starspot evolution would exist. This means that our program can give information about overall trends in starspot evolution, but it cannot be trusted to give the exact brightness values of a region for any given time step.

In all cases, the light curve fits look good for synthetic curves and real data alike. The fit of the light curve is good irrespective of noise level in the synthetic curve. An example light curve fit is shown in Figure 7.

To visualize the brightness values, we use Figures 9 and 8. The value on the far right is the input value for the synthetic light curves. The y-axis in these images represents longitude along the stellar surface.

This particular set of images shows the brightness recovery for a synthetic light curve produced by darkening two stripe regions and two box regions and simulated 14th magnitude Gaussian noise. The code does a good job of recovering the brightness values for both boxes and stripes. The stripes, in general, are recovered more of the time, more precisely, and more accurately than

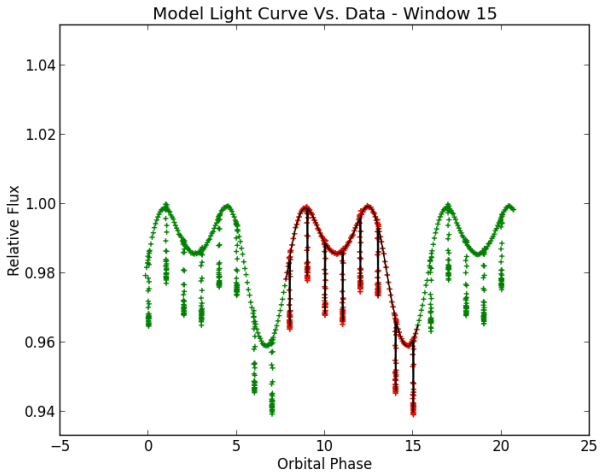

Figure 7. Typical light curve fit produced by the Eclipse Mapping code. The green points are the overall light curve data, the red points are the data for the current window, and the black line is the model fit for that window.

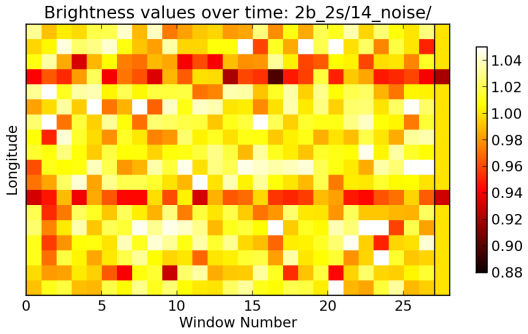

Figure 8. Recovered stripe brightness plot versus window number. The color bar (right) shows the relative brightness value relations to the color scale.

the boxes. At any given window, one can usually believe a particular stripe measurement. The resulting stellar brightness map from this same set of recovered brightness values is shown in Figure 10.

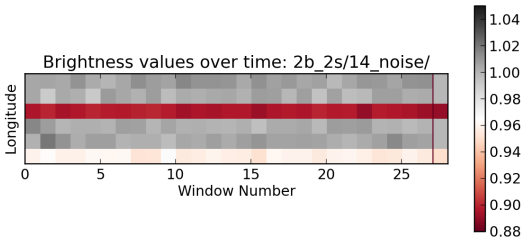

Figure 9. Recovered stripe brightness plot versus window number. The color bar (right) shows the relative brightness value relations to the color scale.

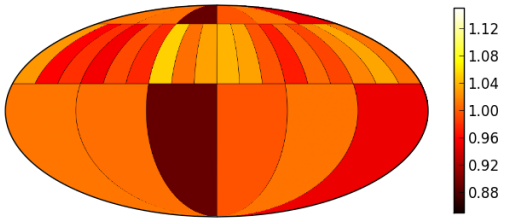

Figure 10. Typical stellar brightness map produced by the Eclipse Mapping code. This particular map is window 15 of the two dark stripes, two dark boxes synthetic light curve.