Development of An Eclipse Mapping Routine Using Python for Analysis of *Kepler* Data

Nathan Oliver Smith

Supervisor: Dr. Richard Olenick

A thesis submitted in fulfilment of
the requirements for the degree of
Bachelor of Science

The Department of Physics
Constantin College
The University of Dallas
Irving Texas USA

14 August 2019

CHAPTER 2

Formalism

2.1 Assumptions

The eclipse mapping method studied and developed by Baptista and Steiner 1993, Horne 1985, and others, is a computational approach that utilizes the eclipse geometry of the secondary star orbiting around the primary at any point in time. Each instant during the orbit has an individual eclipse profile which can be studied to determine the brightness distribution of the accretion disk around the primary white dwarf. However, some fundamental assumptions must be made about the physical characteristics of the disk: First, the surface of the secondary star is defined by the Roche Lobe Equipotiental. Secondly, the matter contributing to the brightness distribution is constrained to the orbital plane. Finally, the emitted radiation is independent of orbital phase. The first two are basic assumptions about the qualities of dwarf novae and accretion disks. The third, however, may be problematic when performing analysis on anomalies such as outbursts and super outbursts. This is important to keep in mind during the future development of the Eclipse Mapping method.

2.2 General Characteristics

The problem of the Eclipse Mapping Method has two primary components: The eclipse map, and the maximum entropy equation, which are used in conjunction to reconstruct the accretion image. The eclipse map generates an eclipse geometry at every phase point, while the maximum entropy equation gives the most physically probable solution for the brightness distribution.

2.2.1 The Eclipse Map

The purpose of the eclipse map is to connect light curve data space with the physical space surrounding the white dwarf. This physical space is defined by a pixel grid that lies within the orbital plane of the orbit, where the area of each pixel corresponds to unique area in the accretion disk. Each pixel area is defined as

$$A_{pixel} = \frac{(\gamma R_{L1})^2}{N^2} \tag{2.1}$$

Where R_{L1} is the distance to the first Lagrange Point, *N* is the length of one side of the pixel gird and γ is a scaling factor that can be assigned arbitrarily. I left this parameter equal to one due to time constraints during testing, but experimentation with this parameter is greatly encouraged.

In essence, the eclipse map at some phase point ϕ denotes whether a particular pixel -and hence a particular area of the disk- is eclipsed. This gives valuable information about whether that area of the accretion disk is contributing to the light curve in that instance. The total flux of the disk, f_ϕ, can be calculated by taking the some of all visible pixels at phase ϕ:

$$f_\phi = \frac{\Theta^2}{4N^2} \sum_{j=1}^{N^2} I_v(j) V(j, \phi) \tag{2.2}$$

Where $I_v(j)$ is the is the intensity of pixel *j*, and $V(j, \phi)$ is the eclipse map visibility function at phase ϕ for pixel *j*. The output of this function is either 1 or 0, but a fraction can also be used to calculate a more detailed light curve. Increasing the resolution of the grid N^2 can achieve a similar effect at the expense of computational time. Θ is a constant defined as

$$\Theta^2 = \left(\frac{R_{L1}}{D}\right)^2 cos(i) \tag{2.3}$$

Where *i* is the orbital inclination, and *D* is the distance from the system to Earth. The distances to cataclysmic variables can be uncertain. Ideally, better values of Θ can be found through eclipse mapping analysis. Horne 1985 uses units of solar radii per kiloparsec for the value of Θ.

In order to determine whether a pixel is eclipsed at phase ϕ, consider a computational model of the orbit scaled by the the distance from the primary star to the R_{L1} point. Each pixel on the accretion disk grid, if visible, will direct line of sight from earth. Naturally, if it is eclipsed, this line of sight will be blocked. Therefore, a simple way to determine pixel visibility is to determine the orbital position at ϕ and check whether some pixel *j* is eclipsed by the secondary star with given radius R_2. If R_2 is uncertain, the Roche Lobe equation can be used determine the visibility of the pixel. Horne 1985 uses an estimate radius which encloses the companion star. If the pixel's line of sight intersects with this radius, then the Roche Lobe equation is solved at short intervals to determine whether the pixel is eclipsed. Ideally, this method will be supported in an updated version of the code described in Chapter 3. Currently the code only supports use of a given radius R_2. In Kepler data, many orbital parameters are already known, which makes the construction of the orbital model possible to determine visibility. Details regarding code structure and implementation will be discussed in Chapter 3. (Add more to this subsection later?)

2.2.2 The Maximum Entropy Equation

The model used for the maximum entropy function is a variation of a metric known in information theory known as the Kullback–Leibler Divergence (cite) (KL Divergence), or relative entropy information gain. This function is written as

$$D_{KL}(P||Q) = -\sum_{X \in x} P(x) log \left(\frac{P(x)}{Q(x)} \right) \tag{2.4}$$

which is equivalent to

$$D_{KL}(P||Q) = \sum_{X \in x} P(x) log \left(\frac{Q(x)}{P(x)} \right) \tag{2.5}$$

This equation determines the amount of information lost when a known function *Q* is used to approximate *P*. In other words, how many more bits of information are needed to accurately represent *P*? In the eclipse mapping problem, the KL Divergence is denoted as the entropy function *S*, representing the information entropy of a solution set of pixels to some weighted default image.

$$S = -\sum_{j=1}^{N^2} p_j log\left(\frac{p_j}{q_j}\right) \tag{2.6}$$

Where

$$p_j = \frac{I_j}{\sum_k I_k} \qquad q_j = \frac{D_j}{\sum_k D_k} \tag{2.7}$$

In Eq. 2.7, I_j is the intensity of pixel *j*, and p_j normalizes this intensity over the sum of all pixel intensities. Similarly, q_j takes the weighted intensity from the default map D_j and normalizes over the sum of all weighted intensities. The default map is defined below as

$$D_j = \frac{\sum_k w_{jk} I_k}{\sum_k w_{jk}} \qquad w_{jk} = e^{-\frac{(R_j - R_k)^2}{2\Delta^2}} \tag{2.8}$$

Where R_j and R_k are the distances of pixels *j* and *k* from the center of the grid. There are a variety of weight functions for w_{jk} which emphasize different aspects of the disk image solution set *I*. The particular form of w_{jk} in Eq. 2.8 is, in essence, a convolution that emphasizes the radial structure of image for scales greater than Δ. Consequently, Δ acts as a smoothing value for the whole image. Other forms of w_{jk} can be found in the appendix. (it does do this right?)

With these definitions, Eq. 2.6 must be maximized with some constraint to observed data to find the most physically probable solution for *I*. Due to restrictions in the python library, Eq. 2.6 must be first made positive and then minimized. Hence, the solution for *I* corresponds to the minimum of the positive form of the entropy function *S*

$$I \widehat{=} min \left[\sum_{j=1}^{N^2} p_j log \left(\frac{p_j}{q_j} \right) \right] \tag{2.9}$$

When constrained by the chi squared function

$$\chi^2 = \frac{1}{M} \sum_{\phi=1}^{M} \frac{(f_\phi - d_\phi)^2}{\sigma_\phi^2} \tag{2.10}$$

Where f_ϕ is the calculated flux defined by Eq. 2.2, d_ϕ is the observed flux, σ_ϕ is the uncertainty of d_ϕ, and *M* is the number of data points in the observed light curve. A common method of constraint is to set up the problem with Lagrangian constraints by defining a χ_{aim} value, which will determine how much the maximum entropy algorithm will constrain the solution for *I* to the data. The constraint function will then be defined as

$$0 = \chi^2 - \chi_{aim}^2 \tag{2.11}$$

Where 0.6 as χ_{aim}^2 per degree of freedom usually gives the smoothest solutions and accounts for noise [Baptista and Steiner 1993]. A value of 1 will yield the highest constraint to the light curve, but can take significantly longer to solve and will not account for flickering. The algorithm may even fail to achieve this value in certain circumstances. In contrast, a smaller value will over-prioritize the entropy function and fail to produce and accurate image.