Mirage: A New Package for the Simulation of Gravitationally Microlensed Quasars

by

Jordan Koeller

An honors thesis submitted to the Department of Physics & Astronomy at

TRINITY UNIVERSITY

in partial fulfillment of the requirements for the
Bachelor of Science in Physics with Honors
May 2019

Accepted by .....
Prof. Kelvin Cheng

Accepted by .....
Prof. Nirav Mehta

Accepted by .....
Prof. David Pooley

Accepted by .....
Prof. Orrin Shindell

Accepted by .....
Prof. Jennifer Steele

Accepted by .....
Prof. Niescja Turner

Accepted by .....
Prof. Dennis Ugolini, Chair

Mirage: A New Package for the Simulation of Gravitationally Microlensed Quasars

by

Jordan Koeller

Submitted to the Department of Physics & Astronomy
on April 23, 2019, in partial fulfillment of the
requirements for the Bachelor of Science in Physics with Honors

Abstract

We present `Mirage`, a new package for simulating gravitationally lensed quasars that allows simulation of arbitrarily sized emitting regions of the quasar's accretion disk. We develop a robust, large-scale simulator, wirtten in Python, to model gravitationally lensed quasars. Numerical simulation of gravitationally microlensed quasars provides a tool to determine the physical size and temperature profile of quasars accretion disks which is impossible through direct observation. The method consists of ray-tracing approximately 10^10 paths through a simulated starfield, taking advantage of the latest technologies in cluster computing, to calculate flux received by the observer from each lensed image from different regions of the accretion disk as the quasar moves relative to the lensing galaxy. We compare our simulations to observations of QSO2237+0305 in optical and X-ray wavebands to place constraints on the relative size of the x-ray and optical emitting regions of the quasar's accretion disk.

Thesis Supervisor: Prof. D. Pooley

Acknowledgments

I would like to thank Dr. Pooley for his continual guidance and help on this project for the past three years. Thank you for being patient with me as I not only learn the physics needed for this project but the coding skill as well.

I would also like to thank Dr. Lewis for his help in developing and optimizing the code to perform well. Most of the algorithmic intricacies came from multiple long conversations with him. Thank you as well for providing me with access to use the computer science department's servers and laboratory computers for running large simulations on.

Lastly, thank you to the Trinity University Department of Physics and Astronomy, The Trinity University Department of Computer Science, the Mach Fellowship, and the Murchison Fellowship for funding support.

Contents

Chapter 1

Introduction

1.1 History of Gravitational Lensing

The premise of lensing - gravity deflecting the path of light - was theorized as far back as Newton and Laplace. However, they did not know how to describe lensing as light was understood to only be a wave at that time. In 1801, Soldner suggested a derivation of the deflection angle, but he did not have the machinery of general relativity at his disposal [1]. Hence, his derivation produced a result that is off by a factor of two. In 1915, Einstein applied his field equations to the problem and calculated the correct deflection $\vec{\alpha}$ of a photon for a point mass to be

$$\vec{\alpha} = -\frac{4Gm}{c^2 b}\hat{b}, \tag{1.1}$$

where m is the mass of the object and $\vec{b}$ is the vector from the massive object to the photon at the photon's closest approach if we disregard lensing effects (typically called the *impact parameter*). Applying this equation to our own sun, Einstein calculated that a photon grazing the surface of the sun would be deflected by an angle of $1.7''$. In 1919, this result was confirmed when a star behind the sun was seen as deflected by an amount consistent with this value during a solar eclipse [2]. This observation is seen as one of the first definitive experiments suggesting the validity of general relativity.

1.2 The Lens Equation

An idealized geometric description of a lensed system may be found in Figure 1-1. Physically, the source S emits a beam of light that is deflected by the lens at L at the angle $\vec{\alpha}$, causing it to reach the observer O. From the observer's perspective, however, the object appear at the location S'.

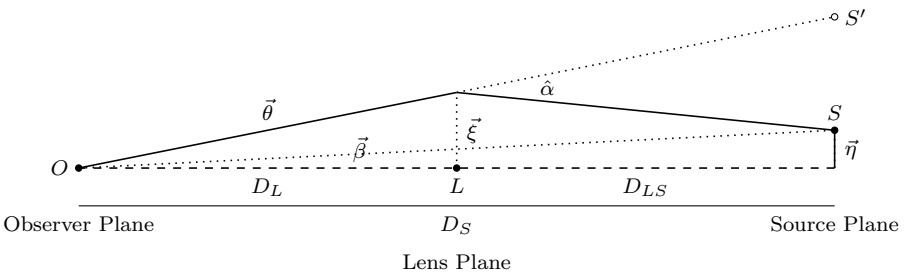


Figure 1-1: Typical geometry of a lensed system. The observer, looking along the direction of $\vec{\theta}$ sees an image of the source object form at S' due to the deflection of light caused by the lensing object L. Note that the distances involved are angular diameter distances. Thus $D_S \neq D_L + D_{LS}$.

Looking at Figure 1-1, we see that

$$D_S\vec{\theta} = D_S\vec{\beta} + D_{LS}\hat{\alpha}.$$

Note that the angles $\hat{\alpha},\ \vec{\beta},\ \vec{\theta}$ are two-dimensional vectors on the sky. Solving for $\vec{\theta}$,

$$\vec{\theta} = \vec{\beta} + \vec{\alpha}, \tag{1.2}$$

where $\vec{\alpha}$ is the reduced angle $\vec{\alpha} = \frac{D_{LS}}{D_S}\hat{\alpha}$. From Figure 1-1, we also see that $\vec{\eta} = D_S\vec{\beta}$. Thus, to put in dimensionless form, we let $\vec{x} = \frac{\vec{\xi}}{\xi_0}$, and $\vec{y} = \frac{\vec{\eta}}{\eta_0}$ and arrive at the dimensionless lens equation

$$\vec{y} = \vec{x} - \vec{\alpha}(\vec{x}). \tag{1.3}$$

The normalization constants ξ_0 and η_0 are given by

$$\xi_0 = \sqrt{\frac{4GM}{c^2}\frac{D_L D_{LS}}{D_S}} \tag{1.4}$$

$$\eta_0 = \frac{D_L}{D_S}\xi_0. \tag{1.5}$$

For radially symmetric lenses, if the source object is perfectly aligned behind the lens, the resulting image forms a perfect ring known as the *Einstein ring*. The radius of this ring θ_E for a point lens may be calculated by

$$\theta_E = \xi_0/D_L = \sqrt{\frac{4GM}{c^2}\frac{D_{LS}}{D_S D_L}}. \tag{1.6}$$

The Einstein radius of a lensed system is a useful unit for measuring angles for the system. For a galaxy of $10^{12}\ M_\odot$ at reasonable astrophysical distance ($z_{lens} \approx 0.5,\ z_{source} \approx 2.0$), the Einstein radius is approximately $2'$. For a $1\ M_\odot$ lens in the same system, it is on the order of *microarcseconds*. In reality, very few lensed systems have a perfectly symmetric lens. Most lenses are asymmetric with a large external shear component caused by the gravity of large scale structures like galaxy clusters (Figure 1-2).

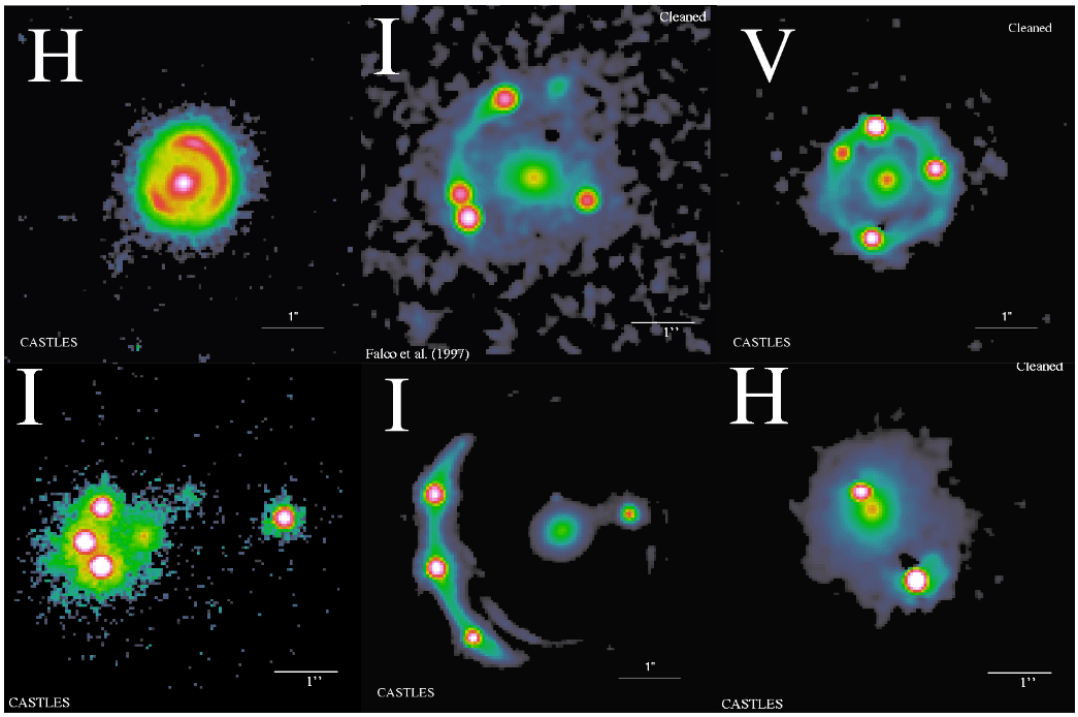


Figure 1-2: HST observations of B0631, MG0414, SDSS0924, HE2149, RXJ1131, and RXJ0991 in the H, I, and V bands (going clockwise starting with the top left image). Images courtesy of CASTLES gravitational lens database [3].

1.3 Lensing Potential

In order to explore lensed systems we need a better understanding of the deflection angle $\vec{\alpha}$. The common practice here is to scale gravitational potential as a potential describing the the strength of the lensing effect in a region of space called the lensing potential. To derive the lensing potential we start with a Newtonian understanding of gravitational potential Φ

$$\Phi = -\frac{GM}{(b^2 + z^2)^{1/2}},$$

where b is the impact parameter of the photon traveling through the lens and z is the distance away from the lens plane [4]. To simplify this expression, we invoke the thin lens approximation. We assert that for a gravitationally lensed system, the width of the lensing galaxy is negligible compared to the distances involved of D_L, D_{LS}, and D_S. Hence we treat the galaxy as a plane by integrating along the z-axis and projecting the mass into a thin sheet. After proper rescaling, we arrive at the dimensionless lensing potential for an arbitrary projected surface mass density $\Sigma(\vec{\xi})$ as

$$\Psi(\vec{x}) = \frac{4GD_L D_{LS}}{c^2 D_S} \int \Sigma(\vec{x}') \ln|\vec{x} - \vec{x}'| d^2x'. \tag{1.7}$$

The deflection of light may be computed from the lensing potential by

$$\vec{\alpha}(\vec{x}) = \vec{\nabla}\Psi(\vec{x}). \tag{1.8}$$

1.4 Image Locations & Fermat's Principle

An alternative treatment of gravitational lensing exists based on Fermat's Principle. Fermat's Priniple states that light takes paths that have relative extrema in travel time. Thus lensed images form at relative minima, relative maxima, and saddle points in the time delay surface of a gravitational lens given by

$$t(\vec{x}) = \frac{1+z_L}{c}\frac{D_S}{D_L D_{LS}}\big[\frac{1}{2}(\vec{x}-\vec{y}(\vec{x}))^2 - \Psi(\vec{x})\big]. \tag{1.9}$$

While images can form at all three types of extrema, relative maxima are highly demagnified, leading to images only being seen at saddle points and relative minima in light travel time. Images forming at saddle points are of negative parity and have a negative magnification associated with them. Negative parity images are mirror images of the object getting lensed. Images at relative minima are of positive parity and are not inverted. Two types of time delay exist in gravitational lenses; geometric time delay and gravitational time delay. Geometric delay comes from the geometry seen in Figure 1-1. The path length from source to observer depends on the angle $\vec{x}$, leading to different times required to traverse space from source to observer. Gravitational time delay is a completely relativistic effect caused by the speed of light decreasing in regions of highly curved spacetime.

1.5 Lens Characterization via Convergence & Shear

To describe how gravitational lensing distorts observation of distance source objects, we invoke the Jacobian matrix of the lensing potential

$$A = \delta_{ij} - \Psi_{ij} \tag{1.10}$$

where δ_{ij} is the Kronicker Delta. Note the shorthand notation of

$$\Psi_{ij} = \frac{\partial^2 \Psi}{\partial x_i \partial x_j}. \tag{1.11}$$

From A, there are three linear combinations that are important for describing gravitational lenses:

$$\kappa = \frac{1}{2}(\Psi_{11} + \Psi_{22}) = \frac{1}{2} tr(\Psi_{ij}) \tag{1.12}$$

$$\gamma_1 = \frac{1}{2}(\Psi_{11} - \Psi_{22}) \tag{1.13}$$

$$\gamma_2 = \Psi_{12} = \Psi_{21}. \tag{1.14}$$

The quantity κ describes how much the image expands or contracts after lensing, while the shear vector $\vec{\gamma} = \gamma_1 \hat{x_1} + \gamma_2 \hat{x_2}$ describes how the image "stretches" along a major axis (Figure 1-3). The convergence κ is also useful as it is a unitless quantity describing the surface mass density of the lens. When being used to describe surface mass density the symbol σ is often used rather than κ. The surface mass density in solar masses can be calculated by

$$\frac{\Sigma(\vec{x})}{\Sigma_{CR}} = \sigma, \tag{1.15}$$

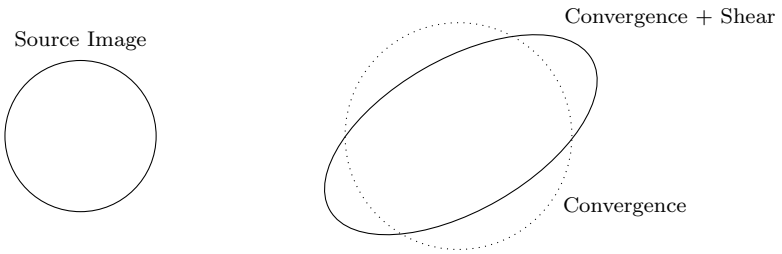


Figure 1-3: The effects of convergence and shear on the image of a circular source object. The magnitude of $\vec{\gamma}$ describes the degree to which the image is elongated while the direction of $\vec{\gamma}$ gives the major axis of the elongation.

where

$$\Sigma_{CR} = \frac{c^2}{4\pi G}\frac{D_S}{D_L D_{LS}}. \tag{1.16}$$

The critical density Σ_{CR} is the surface mass density necessary such that $\vec{\theta} = -\vec{\alpha}$.

1.6 Magnification Properties & Caustics

As surface brightness is conserved via Liouville's theorem, any increase or decrease in the apparent size of a source object by lensing affords a magnification or demagnification of the source object. The magnification coefficient is calculated by the inverse of the determinant of the Jacobian,

$$\mu = \det(A)^{-1} = \frac{1}{(1-\kappa)^2 - \gamma^2} \tag{1.17}$$

where $\gamma = |\vec{\gamma}|$. Note that μ may be positive or negative. If μ is positive, the quasar's image has *positive parity*. Similarly, if μ is negative, the image has *negative parity*. Negative parity images invert the image of the source. Additionally, if $\det(A) = 0$, this produces areas of formally infinite magnification. These areas form curves called critical curves on the lens plane, which when mapped to the source plane are called caustics. Depending on γ, these caustics have different shapes. See Table 1.1 which draws caustics and critical curves for a point mass lens. When a source object enters a region enclosed by a caustic, two new images appear on the critical curve and a spike in magnification is experienced. These two images are always of opposite parity, with an equivalent absolute value of μ between the two images. When the source object crosses a caustic again to leave the enclosed region, the images merge back together and disappear. Figure 1-4 provides a graphical representation of this process.

1.7 Microlensing of Quasars

My research focuses on gravitational lensing by large-scale objects such as galaxies or galaxy clusters, where the mass profile of the lens may be described by a singular isothermal ellipsoid in the presence of an external shear. The singular isothermal ellipsoid model is a parameterized description of the mass distribution of a galaxy, produced by treating stars as particles in an ellipsoidal cloud in hydrostatic equilibrium with its mutual gravity. Lensing caused by such objects is called *macrolensing*, and will often distort objects to

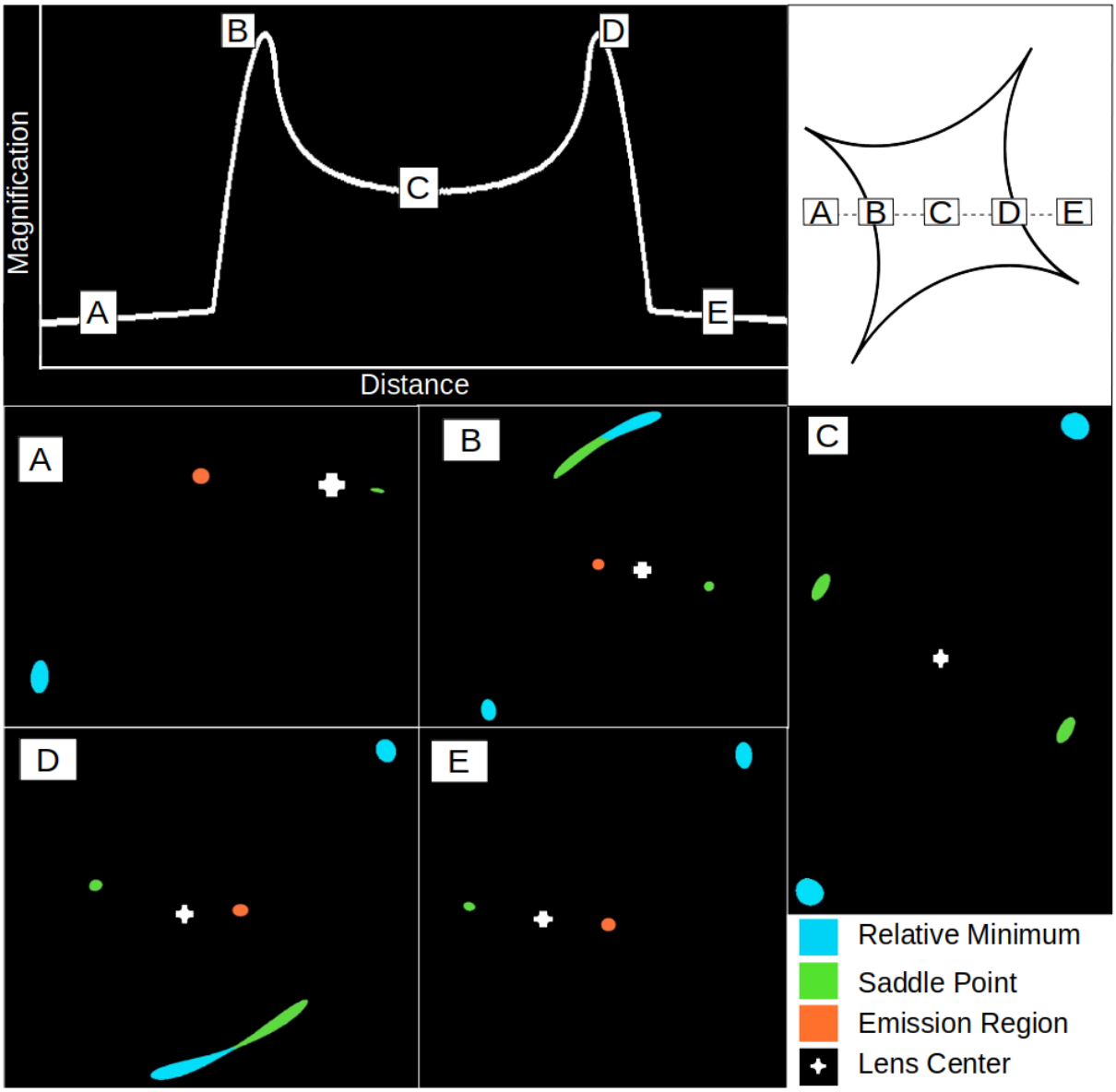


Figure 1-4: Light curve, lens plane, and source plane during a caustic crossing doublet event. In this simulation, a quasar drifts through a caustic region, depicted in the top right diagram. The quasar starts near A, forms two new images while crossing the first caustic line at point B, forms an Einstein cross at point C, destroys the two formed images while crossing another caustic to leave the region at point D, and ends its journey at point E. The plot shows the accompanying light curve for this simulation with points A-E highlighted. The observed lensed images at points A-E are shown below the plot. The center of the lensing galaxy is represented by a white cross, and the physical location of the quasar is shown with the orange circle. Images of positive parity are drawn in blue, and negative parity images are drawn in green. Note that in frame C the quasar is directly behind the galaxy, and thus the symbol for the quasar is eclipsed by the symbol for the lensing galaxy's center.

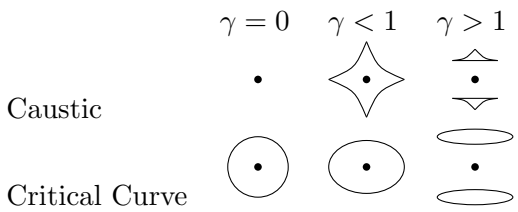


Table 1.1: Caustic and Critical Curve Shapes for a point mass with varying shears. For circularly symmetric systems ($\gamma = 0$) we see one caustic point directly behind the lensing object (represented by a black dot). For systems of low shear, we see an astroid-shaped caustic line, and for high-shear systems we see two triangular caustics form. Both examples with shear have the shear's primary axis along the horizontal axis.

appear as arcs in the sky, or split an image into two or more pieces. For most applications modeling galaxies as continuous distributions of matter is appropriate. However, if we think of a galaxy as a collection of stars modeled as point lenses in a sea of smooth dark matter, each lens produces one or two unique images of the source object. Recalling Equation 1.6, we see that the apparent size of the Einstein radius of a typical $1 M_{\odot}$ point mass is on the order of 10 *microarcseconds* (or μas). Thus, we call these images *microimages* caused by *microlensing*. In fact, the images we see of a lensed system are the sum-total of these microimages. Figure 1-5 depicts a macro-image as the aggregate of many micro-images. While they are far too small to be resolved, they can have a considerable impact on the total flux reaching our telescope's detectors. Additionally, associated with microlensing models are *micro-caustics*. If we calculate the caustic structure of a starfield within a lensing galaxy, we see a network of caustic lines form on the micro-arcsecond scale. Thus, if we consider a quasar source object which has sufficiently small apparent size ($\theta_{\text{obj}} \ll \theta_E = 1\mu as$), as it drifts through space it moves in and out of caustic regions. We can observe the magnitude of the macroimage fluctuate due to a microcaustic crossing. How the flux responds to a microcaustic crossing is highly dependent on the size and profile of the quasar. This enables us to study the size and profile of quasars by studying microcaustic phenomena [5].

1.8 Current Research in Microlensing

The study of quasar structure via microlensing is an idea that was first put forward in 1986 by B. Paczynski [5]. The standard simulation code for this purpose was originally written by Joachim Wambsganss in 1990 [6]. His code is used to generate *magnification maps* of lensed systems through a reverse ray-tracing computation (see Figure 1-6). Simply put, the code simulates a bundle of rays leaving the observer, deflecting through the lens, and then calculates where they intersect the source plane. Magnification maps are generated by sorting the rays into a grid and counting how many rays are found in each bin. These values correspond to each pixel of the magnificaton map, and represent the luminosity of a source placed at that location. While incredibly useful for visualizing caustic networks and calculating the expected frequency of a caustic crossing, magnification maps are limited in their ability to simulate a quasar and generate light curves. They can only accurately represent quasars that have a large enough radius to be adequately rasterized by the pixels in the magnification map. Thus, they fall short of accurately modeling the X-ray emitting region of a quasar, which is expected to only extend a few gravitational radii out from the central

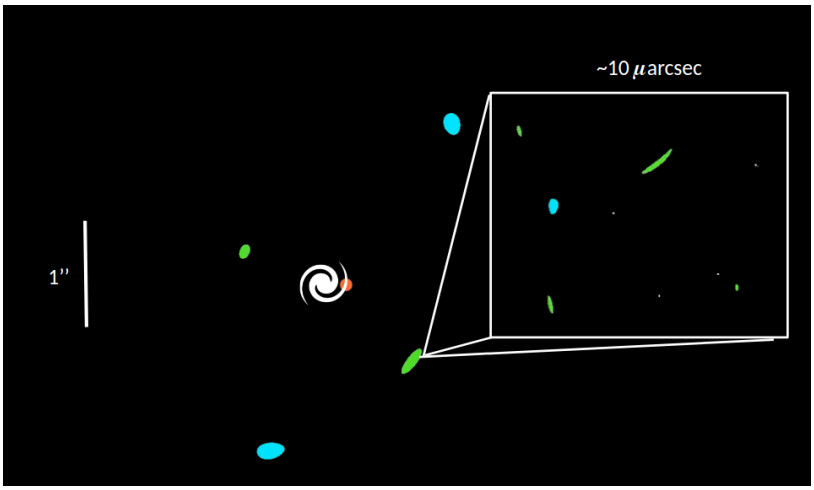


Figure 1-5: Simulation of a quadrupally lensed quasar with inset “zoom-in” on one of the four macro-images at the μasscale. The center of the lensing galaxy is represented as a white swirl, and the physical location of the quasar is shown with an orange dot. Positive parity images are rendered in blue, while negative parity images are rendered in green. White dots in the inset image represent individual stars modeled as point masses.

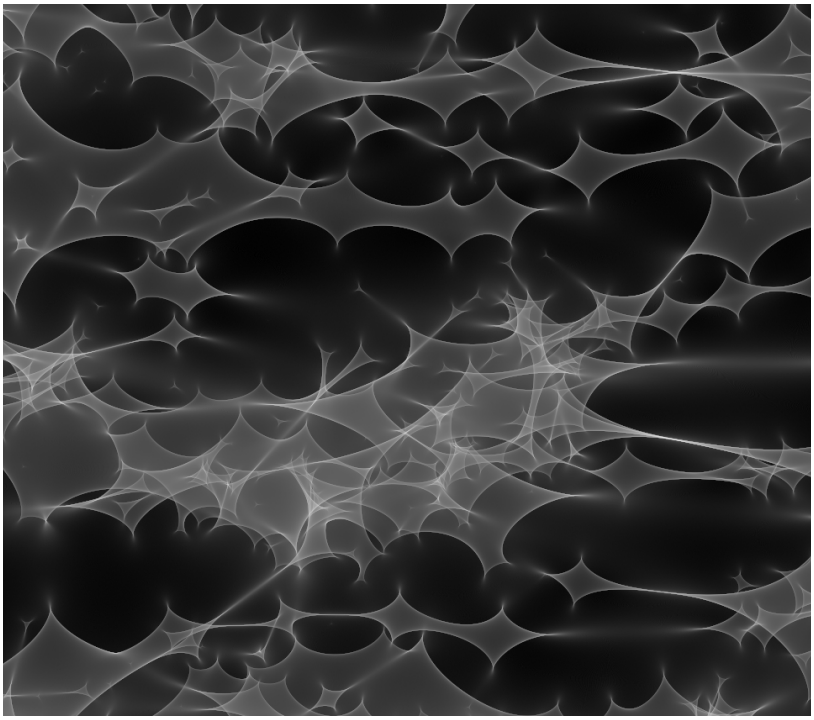


Figure 1-6: Sample magnification map produced with Wambsganss’ code. The dimensions of the image are 100×100 μas. Pixel intensity shows relative brightness, logarithmically scaled.

black hole, or *nanoarcsecond* scale.

Chapter 2

Astrophysical Methods

2.1 Large-Scale Mass Models

To simulate a lensed system we invoke the thin lens approximation and project all the gravitational potential along the path of the photon into a two dimensional lens plane. We do not consider complicated lensed systems involving more than one lensing galaxy at different distances. We model the lensing galaxy as a singular isothermal ellipsoid projected onto the source plane via the thin lens approximation in the presence of external shear. This model provides a simple parameterized model of a gravitational lens that is especially useful due to the existence of analytic solutions to the lens equation [7]. A full specification of the mass model may be found in Table 2.1. Computation of rays on the source plane is accomplished by solving Equation 1.2 for every ray. This mass model allows for simulation of quadruply lensed quasars and the analysis of *macro*-lensing phenomena.

2.2 Microlensing Mass Models

To model microlensing we consider a small "cut out" circular region around a lensed image and populate it with point masses, simulating stars and compact objects. The region must be sufficiently small that the local convergence and shear values caused by the lensing galaxy are effectively constant in the region (typically 100s of μas in diameter). The deflection caused by the lensing galaxy is modeled as a constant shear applied to the entire region. To model matter inside the region we break the surface mass density

$\alpha_1 = \frac{\theta_E}{\sqrt{1-q^2}} \tan^{-1}\left[\frac{\theta_1\sqrt{1-q^2}}{\theta_1^2 q^2+\theta_2^2}\right] + \gamma\theta\cos 2(\phi-\phi_\gamma)$	$\psi(\vec{\theta}) = \vec{\theta}\cdot\vec{\alpha_e} + \frac{\gamma}{2}\theta^2\cos(\phi-\phi_\gamma)$
$\alpha_2 = \frac{\theta_E}{\sqrt{1-q^2}} \tanh^{-1}\left[\frac{\theta_2\sqrt{1-q^2}}{\theta_1^2 q^2+\theta_2^2}\right] + \gamma\theta\cos 2(\phi-\phi_\gamma)$	$\theta_E = \frac{4\pi\sigma_v^2}{c^2}\frac{D_{LS}}{D_S}$

Table 2.1: Thin lens specification for a singular isothermal ellipsoid mass model where q represents the axis ratio for the ellipsoidal mass model and σ_v is the velocity dispersion of the lensing galaxy in the presence of external shear of strength γ at an angle ϕ_γ from the major axis of the ellipsoid [7]. Note how we assume that θ_1 is aligned with the major axis of the ellipse and θ_2 is aligned with the minor axis of the ellipse. Lastly, $\phi = \arctan\left(\frac{\theta_2}{\theta_1}\right)$.

in the region σ into a component contributed by discrete matter in the form of stars and compact objects (σ_*) and a component contributed by continuous matter in the form of dark matter (σ_c) such that

$$\sigma = \sigma_* + \sigma_c. \tag{2.1}$$

The amount of matter in the region to be modeled as stars may then be expressed as a fraction of the total matter (χ) in the region. Thus,

$$\sigma_* = \chi\sigma \tag{2.2}$$

$$\sigma_c = (1-\chi)\sigma. \tag{2.3}$$

Once a sufficiently sized region has been selected as well as an appropriate σ_c, the region must be populated with starry matter. We implement a similar scheme for populating the region with stars to the one used in Wambsganss's code [6]. Starry masses are generated from random sampling of the Kroupa (2001) initial mass function[1] [9]. The starfield is then "aged" by replacing heavy mass stars with masses of their respective compact objects after stellar death. Masses above $30M_\odot$ are replaced with a $10M_\odot$ black hole, masses above $8M_\odot$ are replaced with a $1.4M_\odot$ neutron star, and masses above $0.8M_\odot$ are replaced with a $0.6M_\odot$ white dwarf[2]. Masses are generated until the sum of their masses is approximately equal to the total mass of starry matter in the region, calculated via Equation 1.15. Each mass is given a randomly determined position within the circular region. Once the region has been populated with discrete matter, the lens position of each ray on the source plane is computed via

$$\vec{y} = \begin{pmatrix} 1-\gamma & 0 \\ 0 & 1+\gamma \end{pmatrix} \vec{x} - \sigma_c\vec{x} - \sum_{i=1}^{N} \frac{m_i(\vec{x}-\vec{x_i})}{|\vec{x}-\vec{x_i}|^2} \tag{2.4}$$

where the summation is over all the point mass lenses and σ and γ are computed from the *macro*lensing mass model using Equations 1.12 - 1.14.

In addition to deflecting all rays through the lensing galaxy, `Mirage` also computes the parity of the magnification for each ray. The magnification coefficient of each ray is computed via Equation 1.17 using the determinant of the Jacobian matrix. For our specific model, the elements of the Jacobian are given by

$$A_{11} = \gamma + \sigma_c + \sum_{i=1}^{N} m_i \frac{(x_2-x_{2i})^2 - (x_1-x_{1i})^2}{|\vec{x}-\vec{x_i}|^2} \tag{2.5}$$

$$A_{12} = A_{21} = -2\sum_{i=1}^{N} m_i \frac{(x_1-x_{1i})(x_2-x_{2i})}{|\vec{x}-\vec{x_i}|^2} \tag{2.6}$$

$$A_{22} = -\gamma + \sigma_c + \sum_{i=1}^{N} m_i \frac{(x_1-x_{1i})^2 - (x_2-x_{2i})^2}{|\vec{x}-\vec{x_i}|^2}. \tag{2.7}$$

[1] The Kroupa sampling algorithm implemented in `Mirage` is used with modification from the Python package PopStar [8]

[2] The star generator described here is only the default generator. Note that `Mirage`'s star generator is implemented as a general broken power law, with (optional) post-processing to age stars or pair up a fraction of stars into binary systems. As such `Mirage` is not constrained to only this "aged" Kroupa mass function.

The parity of each ray is determined by the sign of μ. The parity of each ray is then stored with the rays for use in computing parity maps (see Section 2.5.3).

2.2.1 Moving starfields

To model evolution of stars over time, each star may be given a random three-dimensional velocity vector projected into the two dimensions of the lens plane. Velocities are determined by random sampling from a Gaussian distribution, where the mean is the velocity dispersion of the galaxy and with a user-specified standard deviation[3].

2.3 Image generation and magnification calculation

To simulate gravitational lenses we employ the reverse ray-tracing method pioneered by Paczynski in 1986 [5] and used in Wambsganss's microlensing code [6]. We simulate gravitational lensing by tracing a grid of rays from the observer, through the lens, and to the source plane by invoking the lens equation (Equation 1.2). Emission of a quasar is then simulated by searching the bundle of rays for the rays that intersect the emitting region. For a quasar with emitting region of radius r centered at the point $\vec{P}$, the rays that intersect the quasar are those that intersect the source plane at a location within the distance r of the point $\vec{P}$. Images of the lensed quasar can then be rendered by "lighting up" the rays that successfully intersect the quasar. Relative magnification M of the quasar is computed by counting the number of successful rays multiplied by the area between rays on the lens plane and dividing by the physical surface area of the quasar emitting region. Thus,

$$M = \frac{N_r \delta A}{\mu_G \pi r^2} \tag{2.8}$$

where N_r is the number of intersecting rays, μ_G is the magnification coefficient for the macro image caused by the entire lensing galaxy, and δA is the area subtended on the sky by one cell in the grid of rays on the lens plane. Note that if we are modeling macrolensing and thus multiple images with different magnifications we assume $\mu_G = 1$. Implicitly this calculation assumes constant surface brightness across the quasar's emitting region. Relative brightness in magnitudes may then be computed by taking the log of M,

$$m = -2.5 \log(M). \tag{2.9}$$

2.4 The starry region, ray region, and source region

In order for the reverse ray-tracing method to produce good quality simulations, a sufficiently large number of rays must be computed to adequately rasterize the quasar's images. For macrolensing simulations a square ray grid of 2000×2000 rays is typically sufficient. In order for all images of the quasar to be visualized, the ray region must have larger dimensions than the largest separation between images. A square ray region with dimensions of approximately $3\theta_E$ is typically suitable.

[3] To see a rendered animation of a star field evolving over time see `Mirage`'s GitHub page linked in Appendix B.

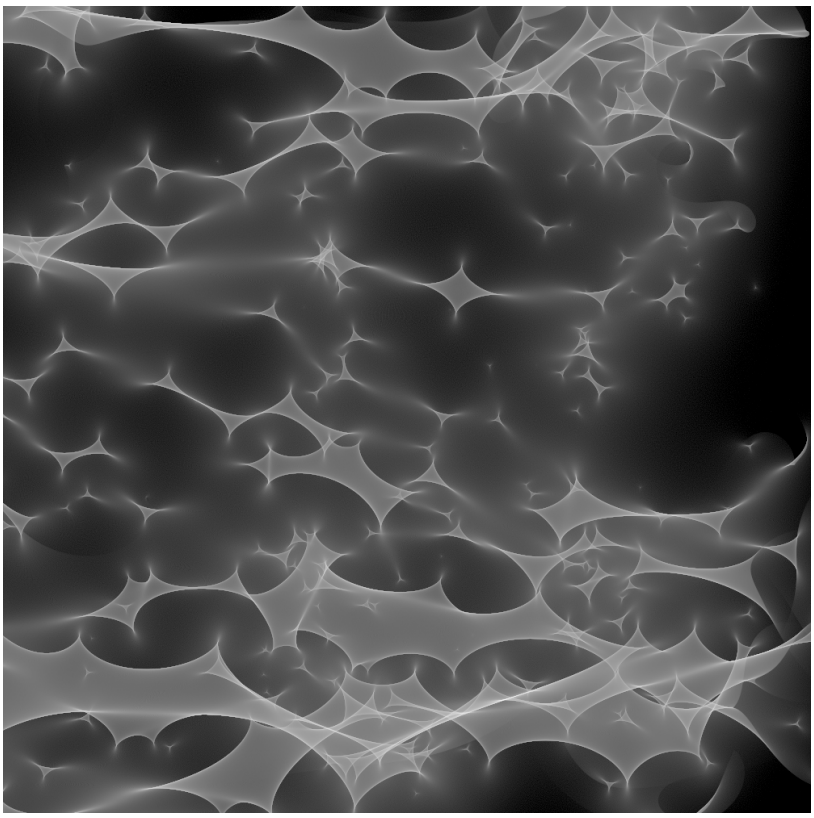


Figure 2-1: Sample magnification map with edge effects present. For this simulation, the mapped ray region and source region were set as the same size. Hence rays could not be scattered into the source region, causing the dark area on the right hand side of the image as well as loops between caustic events around the frame. Image generated using Wambsganss's microlensing code [6].

For microlensing simulations, more care must be taken to select the proper star, ray, and source regions to ensure the microlensing is properly modeled. Due to the convergence and shear of the galaxy, we send a rectangular grid of rays with axis ratio $q = (1 - \sigma + \gamma)/(1 - \sigma - \gamma)$. This axis ratio is used as it maps to a square region on the source plane [6]. Additionally, the starry region must be sufficiently large compared to the ray region to approximate a galaxy made of stars. Note that the starry region must always be circular to avoid the starry region causing an artificial shear. Failure to use a large enough starry region will cause edge effects in the computed relative magnification coefficients of the quasar (see Figure 2-1). Lastly, the dimensions of the ray region must be slightly larger than the dimensions of the source region being explored to allow some rays to be scattered into the source region. Too small of a ray region will also result in the formation of more edge effects [6]. The suitable sizes of these regions must be determined empirically by generating magnification maps and visually inspecting them for edge effects.

2.5 Result Types

`Mirage` is able to produce multiple results types for studying various astrophysical phenomena. Here I describe the various result types as well as the method by which we generate each result type.

2.5.1 Light Curves

The most straight-forward result `Mirage` can produce is a light curve. To simulate a lightcurve, `Mirage` simply linearly interpolates between two locations on the source plane and computes relative magnification along the line at a specified resolution. Lightcurves are extremely important results as they are the most directly comparable to observational data of physical quasars. They also offer the best route to take advantage of `Mirage`'s kD-Tree algorithm (see Section 3.3.1 for more information on the kD-Tree algorithm). Many light curves may be efficently generated from one ray-traced lensed system, and may simulate many differently sized quasars[4].

2.5.2 Magnification Maps

`Mirage` can produce magnification maps similar to those described in Section 1.8, though by a slightly different approach. When producing magnification maps, `Mirage` still simulates circular source objects. Thus magnification maps may be produced for differently sized quasars and how differently sized quasars impacts caustic structure may be directly perceived in the magnification map. Magnification maps are especially useful for performing studies on the frequency of expected microlensing events and for studying the evolution of caustic networks over time when combined with animated starfields[5].

[4] `Mirage` may compute specific light-curves between specified locations on the source plane, or `Mirage` may randomly generate batches of hundreds of lightcurves and compute them all at once. Lightcurve batches are especially useful for statistical analysis of light curve properties.

[5] `Mirage` also includes tools for generating lightcurves from magnification maps by sampling from the magnification map's pixel values. Producing light curves from the full simulation as described above, however, is recommended as the lightcurves produced from magnification maps are a much poorer approximation.

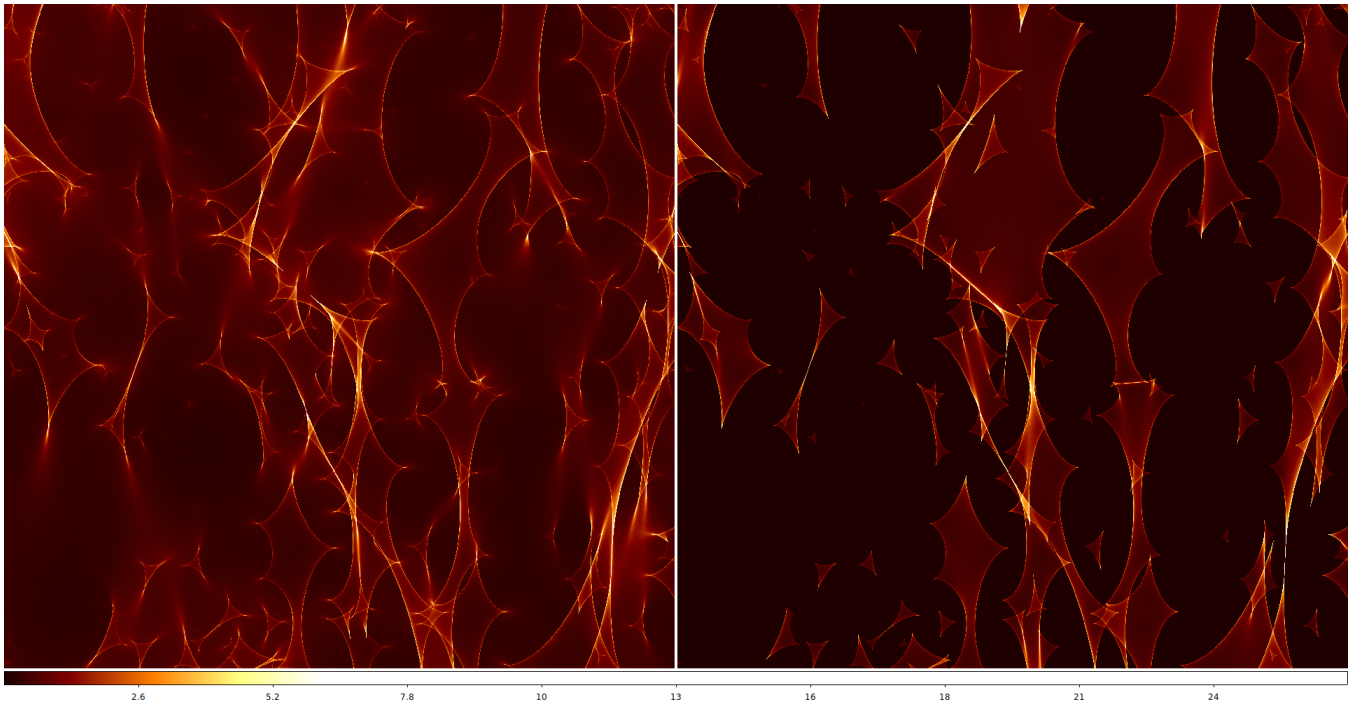


Figure 2-2: Sample parity maps of the same starfield, equally scaled. Left: Negative parity map. Right: Positive parity map. For these images, the macro-image is of negative parity. Thus the positive parity map exclusively shows microlensing phenomena, while the negative parity map has light contributed from micro-images as well as light contributed from the macro-image. Caustic crossings may be easily identified in the positive parity map from locations where adjacent pixels have zero and nonzero values.

2.5.3 Parity Maps

Figure 2-2 shows a sample Parity map result. Parity maps visualize the magnification contributions from a singular microimage parity. As such, there exist positive-parity maps and negative-parity maps[6]. Positive parity maps display the contribution of light from microimages at relative minima in the lens's time delay surface, while negative-parity maps display the contribution of light from saddle points in the lens's time delay surface. As lensed images must have positive or negative parity, macrolensed images must have a defined parity. If a macro-image has a positive parity, then macrolensing phenomena can only produce positive parity images. Hence the negative parity contribution outside of all caustic shapes must be exactly zero. Any location with any negative parity contribution must originate from microimages produced in a caustic crossing event. Similarly, if a macroimage has negative parity, any positive parity images must be the result of a caustic crossing event. Thus caustics are easily identified from locations where contribution from the opposite parity of the macroimage suddenly jumps from zero to any nonzero value.

[6] `Mirage` may produce lightcurves of specific parity as well.

Chapter 3

Computational Methods

To model gravitationally (micro)lensed quasars, we write the simulation software `Mirage`. We use Python as the main language for using and interfacing with the code. We also write a set of tools for analyzing simulation results in Python, taking advantage of scientific computing libraries such as Numpy, Scipy, Astropy, and Matplotlib [10, 11, 12, 13, 14, 15]. Most of the "heavy lifting" computation - the actual ray tracing and simulating of gravitational microlensing - is done in Scala using Apache Spark^{TM} as a framework. We use Apache Spark^{TM} as it enables easy and highly scalable parallelization of the code, affording faster compute times [16]. The code base consists of approximately 1,500 lines of Scala code and 5,000 lines of Python code.

The algorithm used consists of three primary phases. A) Ray-tracing from the observer to source plane, B) sorting the traced rays into a distributed kD-Tree for fast lookup, and C) Querying the distributed kD-Tree numerous times to generate light curves, magnification maps, parity maps, and/or caustic maps [1].

3.1 Apache Spark^{TM} as a compute framework

As specified on the homepage of their documentation, "Apache Spark is a fast and general-purpose cluster computing system," implemented in Scala, with high-level APIs in Java, Scala, R, and Python [17]. It employs a driver-executor model of distributed computation. One node acts as the driver while the rest are executors. Executors store data and are used to perform calculations, while the driver distributes computation to the executors, tabulates what is being computed on each executor, and collects the results of distributed computations after the executors are done. Spark distributes computation as well as data across a cluster of machines through its principle abstract data type, the *Resilient Distributed Dataset* or `RDD` [18]. The `RDD` is a distributed datastructure, meaning the data is partitioned into chunks and each executor stores and manages computation upon a partition of the data[2]. This distributed nature

[1] In addition to the Spark-based backend, `Mirage` also includes a backend written in C++ for local computation. The local backend uses the same algorithm as that described in this section, albeit with a single kD-Tree rather than the distributed kD-Tree described in Section 3.3. For the local backend, the same performance analysis applies as described in Table 3.1, but with $P = 1$. By default macrolensing models are computed locally and Apache Spark^{TM} is only used for microlensing computations.

[2] While Apache Spark^{TM} does a fantastic job at distributing work quickly and effectively, the performance of the system is highly dependent on some key parameters such as how many partitions should be used, how much memory to allocate on

allows for an `RDD` to encapsulate more data than can fit in the memory of any singular node. The `RDD` is also a resilient datastructure, meaning *it can recover from errors with minimal re-calculation of data.* As operations are being performed on an `RDD`, the driver node tabulates what data exists on what nodes, and what calculations have occurred to create that data. This allows for the cluster to recover in the case of an error, such as running out of memory or a node dropping its connection with the cluster. If a node disconnects, the driver node can consult its records for what data was lost and instruct an active node to recompute the lost data and resume the calculation. This aspect enables simulations on a truly massive scale, with thousands of computers working together to complete a simulation or long-running simulations to be performed where the likelihood of something going wrong such as a node momentarily loosing connection to the cluster is high [16].

3.2 Phase 1: Ray Tracing

As mentioned in prior sections we use the reverse ray-tracing method to model gravitational lensing. We send out an even grid of rays (as specified in Section 2.3) and deflect them through the lens to their intersection point with the source plane via equation Equation 2.4. For this computation a "brute force" calculation is used, where each ray is deflected by all the stars in the starfield[3]. All the rays exist in an `RDD` allowing the ray-tracing to be done in parallel across the computational cluster. Furthermore, as the computation of each ray is independent of all the other rays, the ray tracing may be done with almost no network traffic between executors. Each executor has all the information it needs for computing its collection of rays.

3.3 Phase 2: Constructing the Distributed kD-Tree

Technically speaking, all the data needed to simulate a microlensed quasar is produced in Phase 1 of the algorithm for all the different result types `Mirage` can produce. Computing results, however, would be extravagantly slow without further processing as a full search of all the rays becomes necessary for each pixel of a map or each step along a light curve. To enable fast computation of our result types we design and utilize a distributed kD-Tree algorithm.

3.3.1 The kD-Tree Algorithm

A kD-Tree is a D-dimensional datastructure that partitions data spatially into partitions of varying size such that the number of data points in each partition is approximately equal [19]. The structure is built by a binary tree algorithm. At the start of construction all the data exits within the root node of the tree. If the amount of data in the node exceeds a threshold (for our implementation, 64 data points), two children nodes are created and the data is sorted into the two children nodes. A split hyperplane is computed along one of the dimensions of the data and all data with a lower value for that dimension is sent to one child and all the data with a larger value than that split is sent to the other child. The split hyperplane is chosen such

each machine, etc. For a full analysis of optimal Spark settings, see Appendix B.

[3]For macrolensing simulations where no stars are present, the rays are only deflected by the isothermal ellipsoid. via the equations in Table 2.1. Otherwise the algorithm is the same

that approximately half of the data is sent to each child. This process is then repeated recursively on the children nodes until all nodes contain less data than the threshold amount. Note that with each recursive call the dimension along which a node is split changes, allowing for efficient partitioning of D-dimensional datasets [19]. For our purposes, our kD-Tree is optimized for two-dimensional data as the source plane is a two-dimensional space.

3.3.2 The Distributed kD-Tree

To construct a distributed kD-Tree, we sort each partition of our `RDD` of rays into a kD-Tree, so that a kD-Tree exists on each executor in our cluster. Due to the parallel nature of Apache Spark™, these kD-Trees are all made independently of all others and may be constructed in parallel. Searching the distributed kD-Tree is then quite easy. To simulate a quasar, each executor is provided with the location and radius of the emitting region. Each executor then queries that location via the kD-Tree algorithm described above, and sends the number of rays counted to the master node. The master node then sums up all the reported counts from each executor and produces a total count, representing the total number of rays that intersected that quasar from all the data across the cluster. These searches can also be done in parallel, with only the final summation that occurs on the driver having to be done sequentially. Note that the distributed kD-Tree requires a large amount of memory. While its time performance is quite good, the memory of storing the kD-Tree as well as billions of rays can be considerable[4]. Furthermore, building the kD-Tree is an $O(N \log N)$ operation in memory. For small simulations memory is rarely an issue, but depending on the size of the cluster used and how many rays are being traced, memory issues may occur.

3.4 Phase 3: Production of Result Types

To simulate a quasar the distributed kD-Tree must be searched for the data points that overlap our quasar. Given a central location and radius of the emitting region, each executor recursively descends its kD-Tree. With each recursion the kD-Trees only follow branches that overlap the circular region of the quasar. This effectively excludes approximately half of the remaining data to be searched with each recursion (see Figure 3-1). Each executor then reports back to the master how many rays intersect the emitting region and the master computes the total number of rays by taking the sum of all the executors' counts. This process is then repeated for each pixel in a magnification map or each step along a light curve[5]

3.5 Algorithm Performance and Complexity

With the algorithm implemented as described, we have the ability to simulate microlensing of quasars on a truly massive scale. The distributed nature of the computation via Apache Spark™ enables us to deflect billions of rays through a starfield in a reasonable timescale. The distributed kD-Tree is a highly parallel

[4] Apache Spark™ does include flags to store the distributed kD-Tree on disk rather than in memory, but doing so impacts the time required to compute results. While not ideal, the performance penalty for storing data on disk is managable. See Appendix B for more information on running simulations with `Mirage`.

[5] In order to generate a parity map the query algorithm is slightly modified in that the partitions are filtered down to the rays that only meet the parity requirements as outlined in Section 2.5.3.

Phase	Description	Complexity	Runtime
Phase 1	Ray Tracing	$\mathrm{O}(\frac{NS}{P})$	1.6 hours
Phase 2	kD-Tree Construction	$\mathrm{O}(\frac{N}{P}\log\frac{N}{P})$	0.2 hours
Phase 3	Querying	$\mathrm{O}(Q\log\frac{N}{P})$	6 minutes
Total	–	$\mathrm{O}\big[\big(\frac{NS}{P}+Q\big)\log\frac{N}{P}\big]$	2.0 hours

Table 3.1: Algorithmic Complexity and runtimes of the three phases of `Mirage`, where S is the number of stars in the star field, N is the number of rays deflected, P is the number of CPU cores in the cluster, and Q is the number of queries performed. The reported timescales are for a simulation where $S = 6988$, $N = 6.4 \times 10^9$, $P = 192$, and $Q = 4000^2$. The cluster consisted of nine identical machines; eight executor machines and one of them as the dedicated master machine. Each machine was equipped with an Intel(R) Xeon(R) E5-2683 processor, at 2.10 GHz and 32 cores, 32 GiB of memory, running Scientific Linux release 7.3 (Nitrogen). In total, `Mirage` utilized 240 cores and 217 Gb of RAM.

data structure that allows for fast computation of high-resolution magnification maps, light curves, and parity maps quickly [16]. Furthermore, any and all result types can be produced easily once the distributed kD-Tree has been built. As an example, `Mirage` can compute 1000s of high-resolution light curves for many differently sized quasars, magnification maps for many differently sized quasars, a caustic map, and parity maps after constructing the distributed kD-Tree only once. This allows for the study of a multitude of aspects of microlensing quickly and efficiently. For a complete analysis of algorithmic complexity as well as some sample runtimes, see Table 3.1.

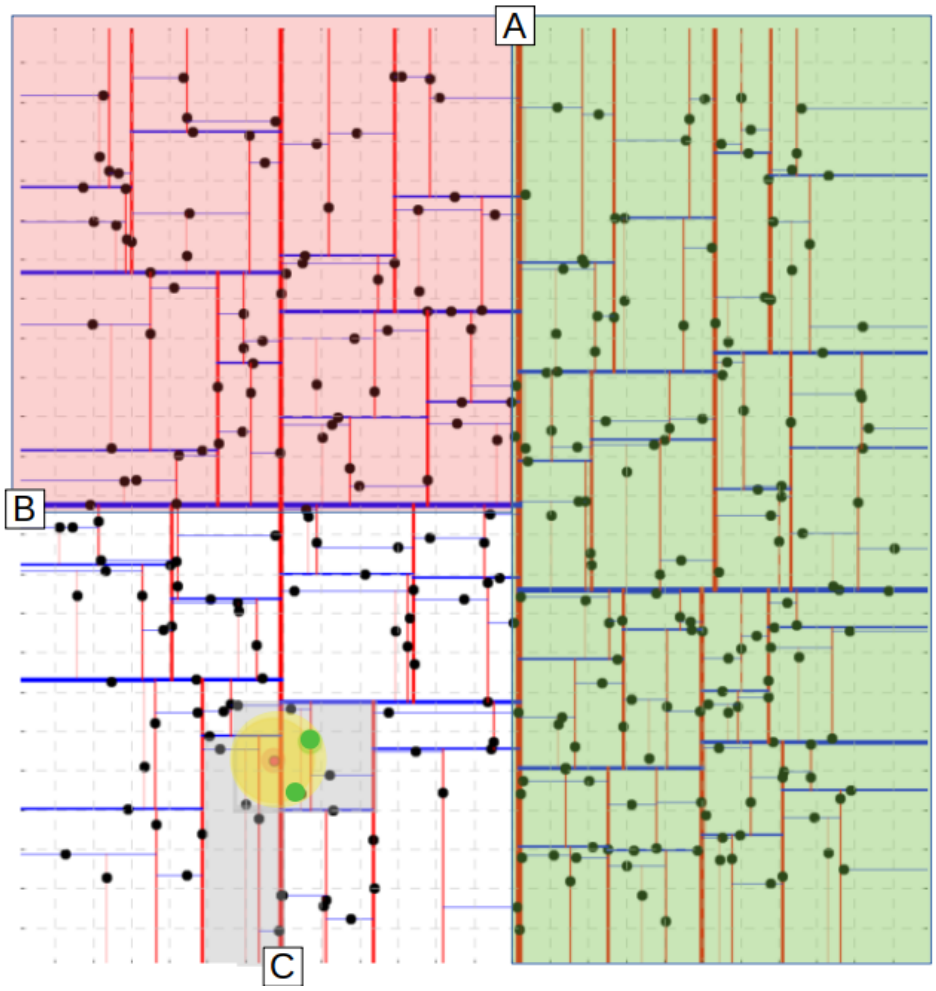


Figure 3-1: Visualization of a kD-Tree algorithm for two-dimensional data. Red lines represent vertical splits, while blue lines represent horizontal splits. When building the tree the data is recursively sorted along its X and Y coordinates, sorting the data into bins where points that are locally near each other are near each other in memory. Querying the tree then answers the question "How many data points fall within a circle of radius r around point P?" Here we use a query region specified by the red point and yellow circle as an example. To query the tree the algorithm starts at the highest level split line (A) and determines which side of the split the region of interest falls on, excluding all the data in the green region from having to be searched. The algorithm then recurses down that branch of the tree and performs the same analysis on the next split line (B), excluding all the data in the red area from having to be searched. If the region of interest falls on a split, the algorithm will recurse down both sides of the split (C). This process continues until no more splits exist. Once the tree has been fully descended, all the relevant data points are checked manually for falling in the region of interest by using the distance formula between the data points and the query location of interest. In this example, only the nine points in the gray region must be manually searched out of the 300 points in the entire space. Lastly, the green points fall in the region and the query returns a value of 2. Figure used with modification from Alexey Abramov [20].

Chapter 4

Mirage in Action: Analyzing QSO2237+0305

As a sample study we use `Mirage` to analyze the light curve of QSO2237+0305, depicted in Figure 4-1. QSO2237+0305 is an especially notable quadruply lensed quasar because of its distance. The lensing galaxy is close to us at a redshift of $z = 0.04$. Similarly, the quasar itself has a redshift (for a quasar) of $z = 1.69$. The small redshifts causes the einstein radius for the system to be small enough that images form near the galactic core where the majority of matter is baryonic in high-density starfields. This leads to a high likelihood of microlensing occurring, as the surface mass density of the system is essentially entirely starry [21]. In fact, recent observation of the system by the OGLE project suggest such an event occurred from 2013 to 2016 in image C of the system [22].

Microlensing offers a unique opportunity to analyze the structure of accretion disks around quasars to a higher level of precision than via other means. While models of accretion do exist, the most popular being the Shakura-Sunyaev model [23], these models are poorly constrained due to the difficulty of comparing their predicted properties with observations of quasars. The process of accretion is fundamental to many astronomical processes such as star formation, galaxy formation, and planet formation, yet how accretion works remains an open question.

Microlensing provides an avenue to study accretion disks via analysis of microlensing caustics. As mentioned in Chapter 1, the observed luminosity of a macroimage during a microcaustic event is highly sensitive to the size and profile of the emitting region of the quasar. By comparing observations made by OGLE in the visible band with those made by the *Chandra X-ray Observatory* allows for constraining of the relative scale of the X-ray and visible emitting regions of the quasar's accretion disk.

In this study, we simulate microlensing in Image C of QSO2237+0305 and generate light curves for many differently sized quasar emitting regions. We then isolate caustic crossing events and statistically characterize the light curve of a caustic crossing for each emitting region size. Lastly we compare our statistical model of a caustic crossing to the caustic crossing of QSO2237+0305 shown in Figure 4-2 in order to constrain the relative size of the optical and X-ray emitting regions of the quasar's accretion disk.

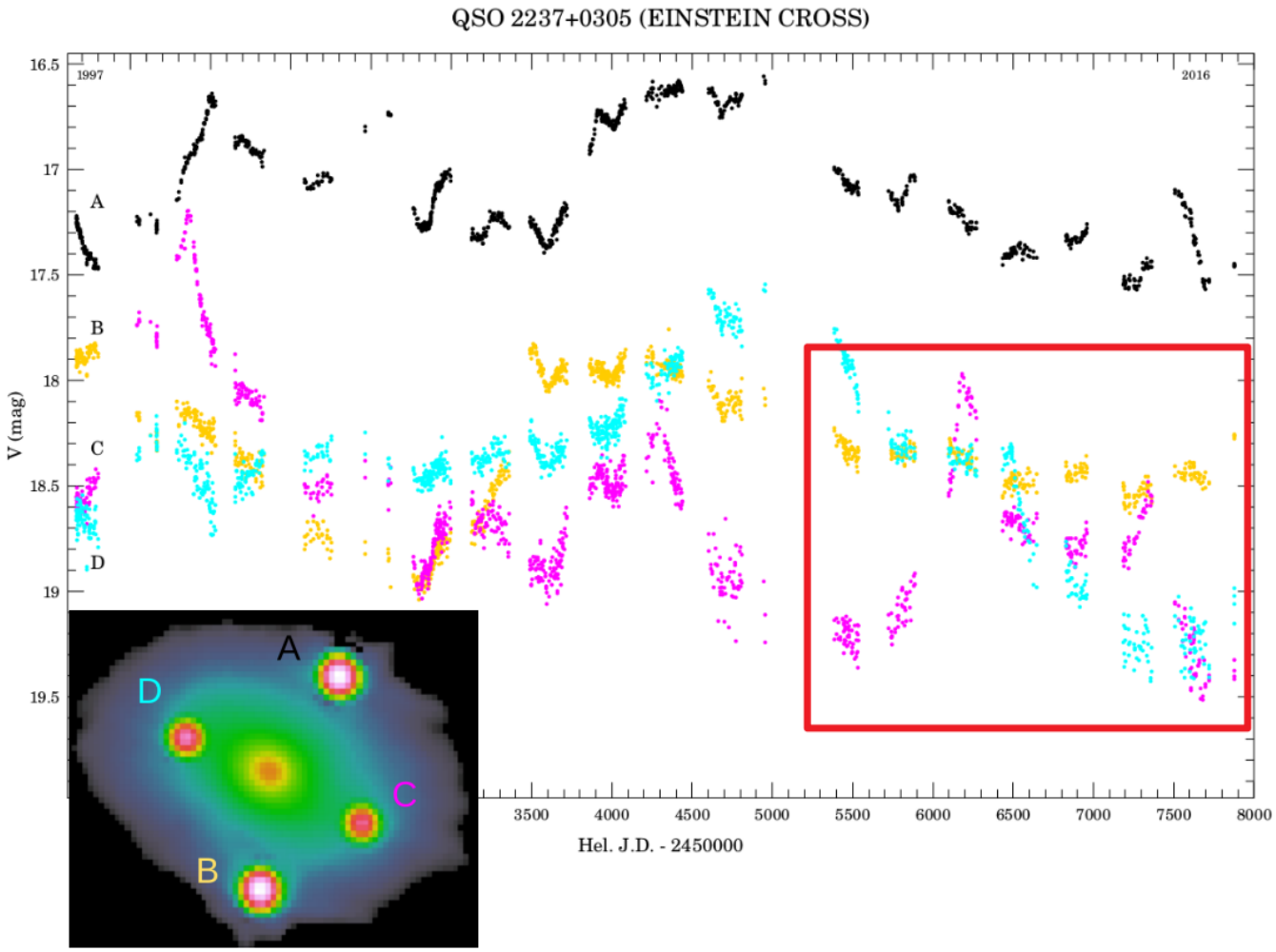


Figure 4-1: OGLE data observing Huchra's lens since 1997 with inset HST V band image of the system [22, 3]. The x-axis shows elapsed time, counted in days on the Julian calendar - 2450000. The four light curves correspond to the image of the quasar labeled in the same color as the light curve. The red box encloses the specific microlensing event observed in image C we are studying (see Figure 4-2).

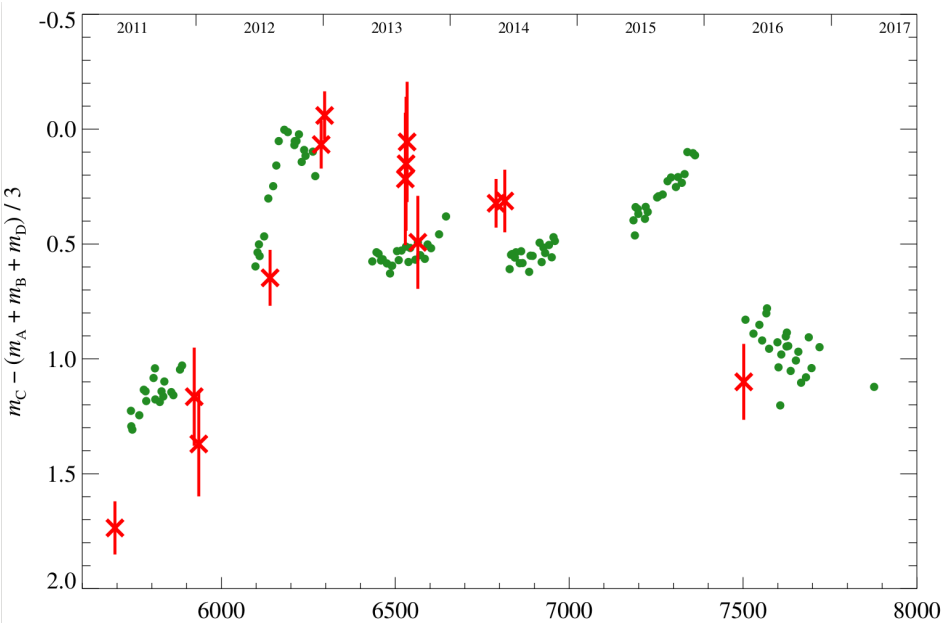


Figure 4-2: Zoom-in on the microlensing event in the red box of Figure 4-1 observed by OGLE and *Chandra* from 2012 until 2016 [22, 24]. The plot depicts the relative luminosity of Image C compared to the other three images of the quasar over time. The x-axis gives time in days in the Julian calendar - 2450000. OGLE observations are presented as the green data points while X-ray intensity measured by *Chandra* is shown with the red data points.

4.1 The Mass Model

To model QSO2237 we use a singular isothermal sphere model in the presence of external shear. Redshift values and image locations were obtained from the CASTLES gravitational lens database [3]. From those values a mass model was determined for the system using C. Keeton's `gravlens` software [25]. The parameters used to model QSO2237 may be found in Table 4.1. A visualization of the model compared to the HST images may also be found in Figure 4-3.

The only remaining parameter to set is the percentage of matter to model as stars. As the images form so close to the galactic core, we model 100% of the matter as starry [21].

4.2 Characterizing caustics via peak offset

In Figure 4-4 we show lightcurves for some sample caustic crossing events, simulated with many differently sized quasar emitting regions. In general we see that larger emitting region quasars (1) have rounder peaks, (2) do not peak as high, and (3) typically have an offset peak location relative to the peak for smaller emitting regions. Figure 4-5 provides a higher detail depiction of peak shift for many simulated quasar radii. Caustic events do exist that do not exhibit an offset peak location, but such events are often highly symmetric events (meaning have equivalent baselines to the left and right of the peak) and thus do not pertain to our study of the asymmetric event seen in QSO2237+0305. Additionally, we see that caustic crossings are highly variable, necessitating a statistical analysis of many simulated caustic crossings for comparing to the light curve in Figure 4-2.

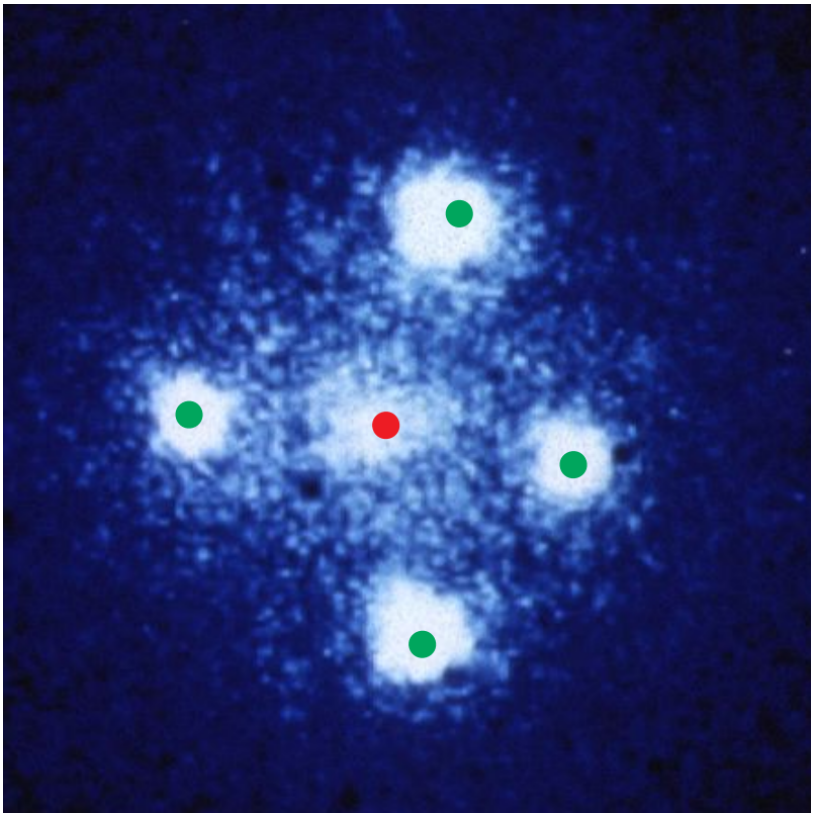


Figure 4-3: Hubble image of QSO2237+0305 with `Mirage` simulation overlaid. The green circles represent the four images computed by `Mirage`. The center of the isothermal sphere is represented by the red circle.

Comparing our sample events to that seen in Figure 4-2 we see that our event constitutes a "doublet" caustic event. In this context, we define a doublet caustic event as one in which the source (1) undergoes an extreme magnification during an initial caustic crossing, (2) stays somewhat magnified for a period of time, (3) undergoes another extreme magnification during the second caustic crossing, and (4) returns to its initial magnification. Thus we focus on analyzing doublet events. Additionally, we see that the light curve in Figure 4-2 is sparsely sampled, especially in the X-ray band. We conclude that attempting to characterize the curvature of caustics or the peak height is highly error-prone, and focus on characterizing our simulation by the peak offset between differently sized emitting regions, as this allows for the simplest comparison to be made between our characterization and the observational data we have from QSO2237+0305. For the first peak in Figure 4-2 the location of the peak in optical wavelengths is easily identified. The peak location is less obvious for the X-ray light curve, but we may assume it peaked somewhere in late 2012 to early 2013.

4.3 Analysis methodology

We generate three random starfields using the mass function described in Section 2.2 and generate magnification maps and light curves for each starfield. Note that for this analysis we do not give stars individual velocities. 300 light curves were randomly produced through each starfield at a resolution of 200 samples per μas. Light curves were generated from randomly selected quasar tracks through the source plane. To

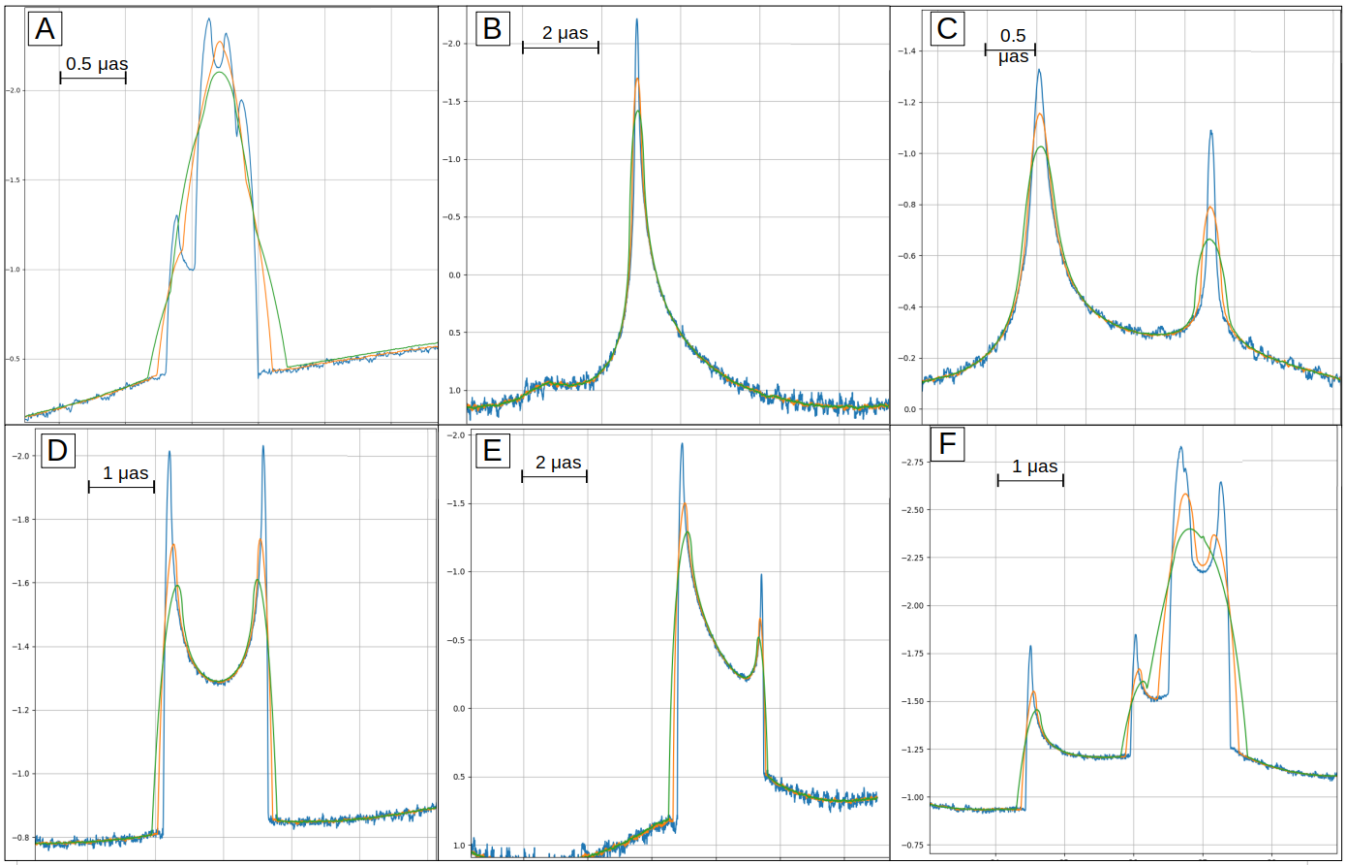


Figure 4-4: Zoom-in on specific caustics from the light curve shown in Figure 4-6 to μas scale. Curve A depicts a narrow multuplet where the large emission regions "blur" together the multiple distinct shoulders. Curve B depicts a symmetric singlet. Curve C depicts two neighboring symmetric singlets that could be misinterpreted as a doublet. Curve D depicts a doublet representative of the caustic crossing we focus on. Curve E shows an asymmetric with non-equal peak heights. These caustics are also of interest in our study of QSO2237+0305. Lastly, Curve F shows a broader multuplet where two unique caustic events blur together in large emission region simulations.

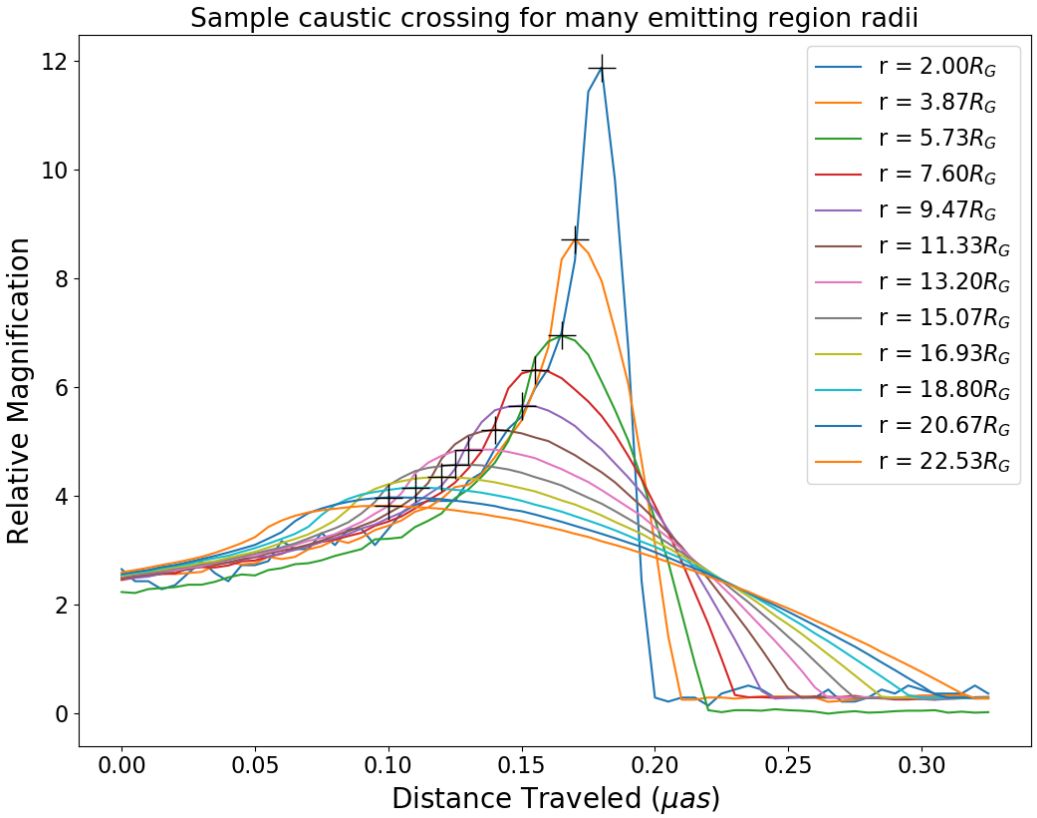


Figure 4-5: Sample caustic crossing event with all simulated emitting region sizes overlaid. The peak location of each light curve is highlighted with a black plus.

Lens Specification

Parameter	Value
Redshift (z_l)	0.04
Velocity Dispersion (σ_v)	$177.95 km/s$
Shear Magnitude (γ)	0.07
Shear Angle (γ_ϕ)	67.1^o
Starry Fraction (χ)	1.0
Starry Region radius (R_s)	$67.5\theta_E$

Quasar Specification

Parameter	Value
Quasar Mass (M)	$10^9 M_\odot$
Redshift (z_s)	1.69

Region Specifications

Parameter	Value
Rayfield Dimensions (R_D)	$112.5 \times 45\theta_E$
Number of Rays (N_R)	6.4×10^9
Source Plane Dimensions (S_D)	$30\ \theta_E \times 30\ \theta_E$

Result Type Specifications

Parameter	Value
Magnification Map Resolution (S_R)	2048×2048 pixels
Lightcurve sampling rate	$200/\mu as$

Table 4.1: Parameters used while simulating image C of QSO2237+0305. In this context θ_E refers to the einstein radius of a $1M_\odot$ point mass lens. To give a sense of scale, $1\mu as \approx 180R_g \approx 0.15\theta_E$.

ensure no edge effects exist in the light curves, we used the dimensions of the magnification as a bounding box of the source plane region explored. Light curves were generated for 12 differently sized circular quasar emitting regions of radii linearly spaced from $2R_g$ to $22.5R_g$. A sample light curve and magnification map may be found in Figure 4-6.

Caustic events were located heuristically by searching for locations in the magnification maps with sufficiently steep slope magnitude in the $2R_g$ magnification maps. Slope was computed by first smoothing the magnification maps with a Gaussian convolution of width 1.1 to reduce noise followed by applying a two-dimensional Sobel operator convolution to the magnification maps [26]. Any locations with a slope magnitude larger than 0.8 (relative magnification)/μas were considered to contain a caustic crossing. The exact parameters for the filters were determined by visually inspecting magnification maps for the algorithm's effectiveness at identifying the majority of caustics without false positives (see Figure 4-7). Caustics were then identified in our light curves by correlating query locations along each light curve with the locations in the source plane identified as caustics.

Once identified, the caustics were sliced out of the long light curves into 60 μas light curves. Slices were chosen such that the peak of the 2 R_g trial is centered in the slice. Slices with more than one caustic event present were filtered out of the batch as they were found to skew the measured peak offset due to the two caustics "blurring together" for quasars with large emission regions (see Figure 4-4, Curve F). Note that this introduces a selection bias for "wide" doublet events, where the second peak is more that 60 μas

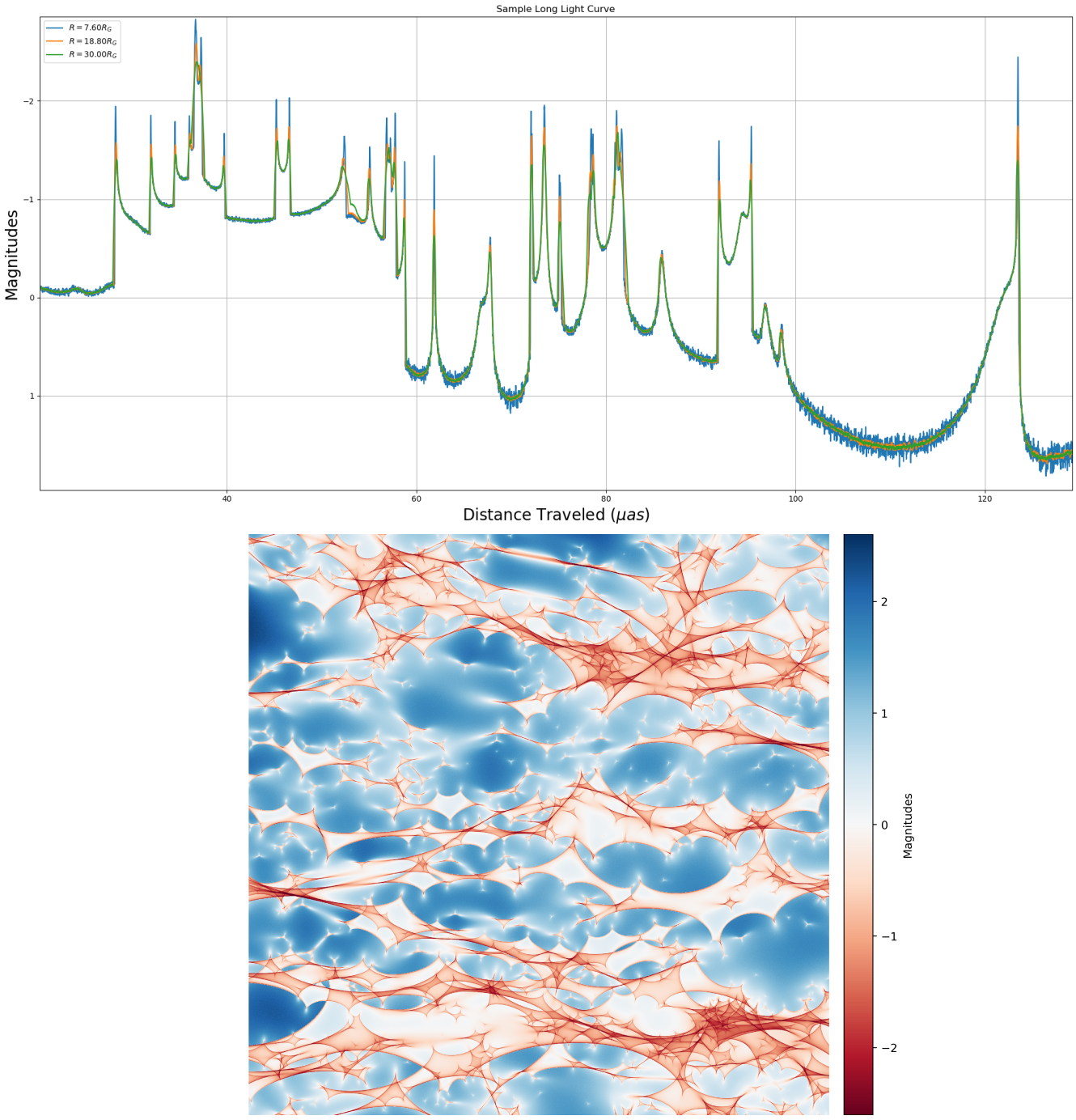


Figure 4-6: Sample light curve and magnification map used in QSO2237+0305 analysis. Above: The "long" light curve spanning the entire source region, as simulated with emitting regions of three different radii. Below: One of the three magnification maps generated in our analysis. Color displays relative brightness in magnitude. Orange areas are magnified, while blue areas area demagnified.

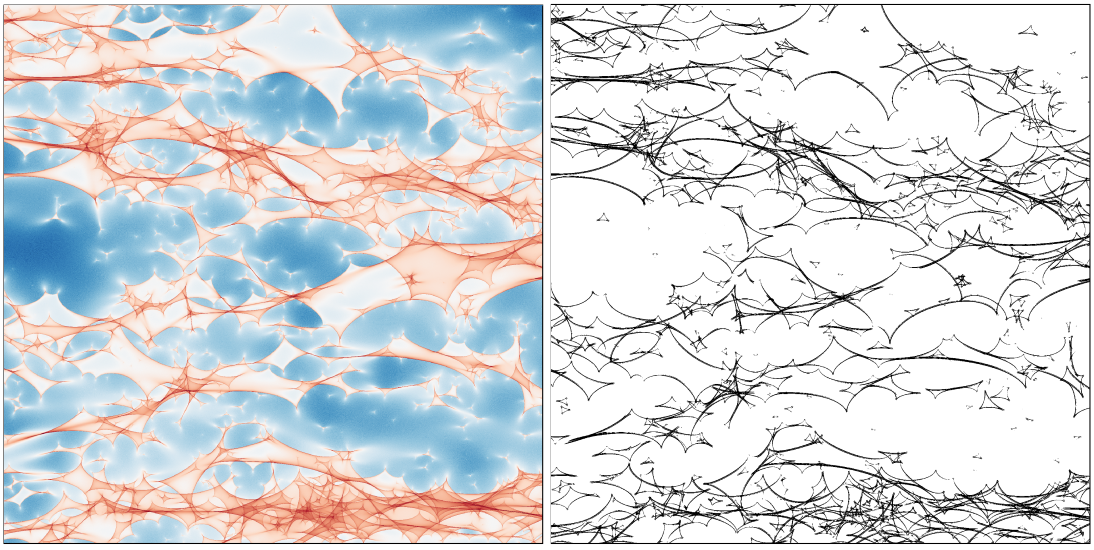


Figure 4-7: Left: Sample magnification map of QSO2237+0305 image C. Right: Magnification map after caustic lines are identified with the method described in Section 4.3. Comparing the two images, we see the algorithm has a very high success rate at locating caustics with few false positives.

away from the first peak. However, we deem this to be appropriate as no blurring of peaks is present in the data shown in Figure 4-2. In total, 23,379 suitable caustic events were found from across all the long light curves generated.

4.4 Selecting Asymmetric Events

To constrain the total set of caustics to only doublets we use peak asymmetry as a heuristic to select the relevant caustic events. Many different methods were attempted to algorithmically isolate doublets such as computing the number of prominent peaks, computing curvature and inflection points on peaks, and fitting to a model of a typical doublet event. Ultimately, peak asymmetry was found to be the best selector, as all peaks involved in a doublet are considerably asymmetric. To compute asymmetry of a peak, we first normalize the peak such that the tallest peak is of height 1 and the lowest data point is of height 0. We then reflect the data to the left of the peak over to the right of the peak and take their difference. Lastly, we compute the asymmetry by combining the differences in quadrature. Mathematically, this is accomplished by the following function. Assume we have a light curve with N many data points. Let x_i be the normalized magnification of data point i, and let $i = 0$ be the index of the central peak of the light curve. Then

$$\text{asymmetry} = \sqrt{\sum_{i=1}^{N/2} |x_i - x_{-i}|^2}. \tag{4.1}$$

We bin our caustic events into 20 bins, using each event's asymmetry (multiplied by 100) as a key to determine which bin each event is placed in. Asymmetry of each event was computed based on the light curve for the smallest emitting region radius. The smallest region was used as it is best represents the nature of the caustic. To determine which bins are appropriate for our analysis we visually inspect 20 random events from each bin and select the bins containing events characteristic of doublet caustic events.

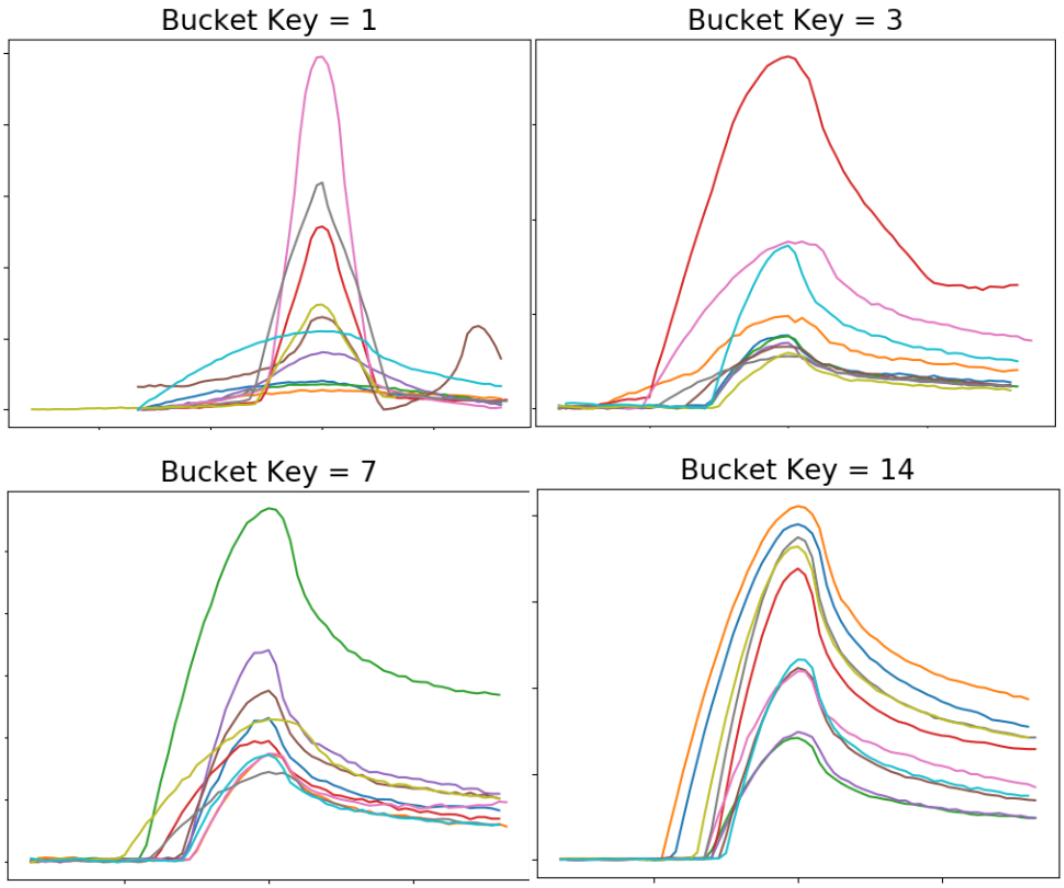


Figure 4-8: Four sample bins of light curve events grouped by asymmetry. Ten randomly selected light curves simulating $2R_g$ emission are shown from each bin. For this simulation light curves were binned into 20 bins, and bins of keys 8 to 18 were identified as bins containing asymmetric doublet caustic crossings.

Some sample bins are displayed in Figure 4-8. Lastly, we aggregate the relevant events into a list of identified doublet caustics. In total, $15,312$ doublet caustic events were identified.

4.5 Measuring Peak Shift

With doublets identified, we analyze how the peak location of the event shifts with increasing emitting region radius (see Figure 4-5). We measure the peak offset by computing the distance between each light curve with the peak location present in the light curve modeling an emitting region with a radius of 2 R_g. Note that we do not make assumptions about the direction the quasar is coming from as it corsses the caustic, so we report the absolute value of the peak shift. We then generate histograms of the measured peak shift across the entire set of identified caustic events for each emission region radius and present them in Figure 4-9. Plotting the most populous bin of each histogram as a function of emission region radius, we see a tight linear relationship between peak shift and emission region radius (Figure 4-10) with a line of best fit of

$$y = 0.702x - 1.346,\ r^2 = 0.998 \tag{4.2}$$

where y is the measured peak shift and x is the radius of the larger emitting region. To make this fit more usable as a way of comparing emitting region radii we rephrase this equation in terms of the difference in emitting region radius by letting $x = \Delta r + 2R_g$

$$D = 0.702(\Delta r + 2R_g) - 1.346 D = 0.702\Delta_r - 0.058 \tag{4.3}$$

where D is the peak shift.

4.6 Comparing to the Light Curve of QSO2237+0305

We now constrain the relative size of the X-ray and optical emitting regions for QSO2237+0305. In order to equate light curves to peak shift we must assume a velocity for the tangential velocity of the quasar relative to the lensing galaxy to compare the light curve (a time series) with the peak shift (a distance). Thus we assume the quasar is moving with a tangential velocity relative to the lensing galaxy of $600 km/s$, or $0.035 \frac{R_g}{\text{day}}$. We use this velocity as an approximate velocity of the quasar's host galaxy as it is the velocity of the Local Group relative to the CMB [27]. In order to measure peak shift we must estimate where we believe the sparsely sampled X-ray data to have peaked in actuality. As a conservative estimate, we assume the X-ray peaked somewhere in the time interval $[6130, 6530]$ days. Combined with the optical peak at day 6180, the magnitude of peak shift is on the interval $[0, 350]$ days, or $[0, 12.25]$ R_g. Combining with our model of peak shift expressed by Figure 4.3, we arrive at the conservative estimate for the difference in size of the X-ray and optical emitting regions of QSO2237+0305 as

$$\Delta R \leq 8.6\ R_g \tag{4.4}$$

where ΔR is the difference in radius of the optical and X-ray emitting regions of the quasar's accretion disk.

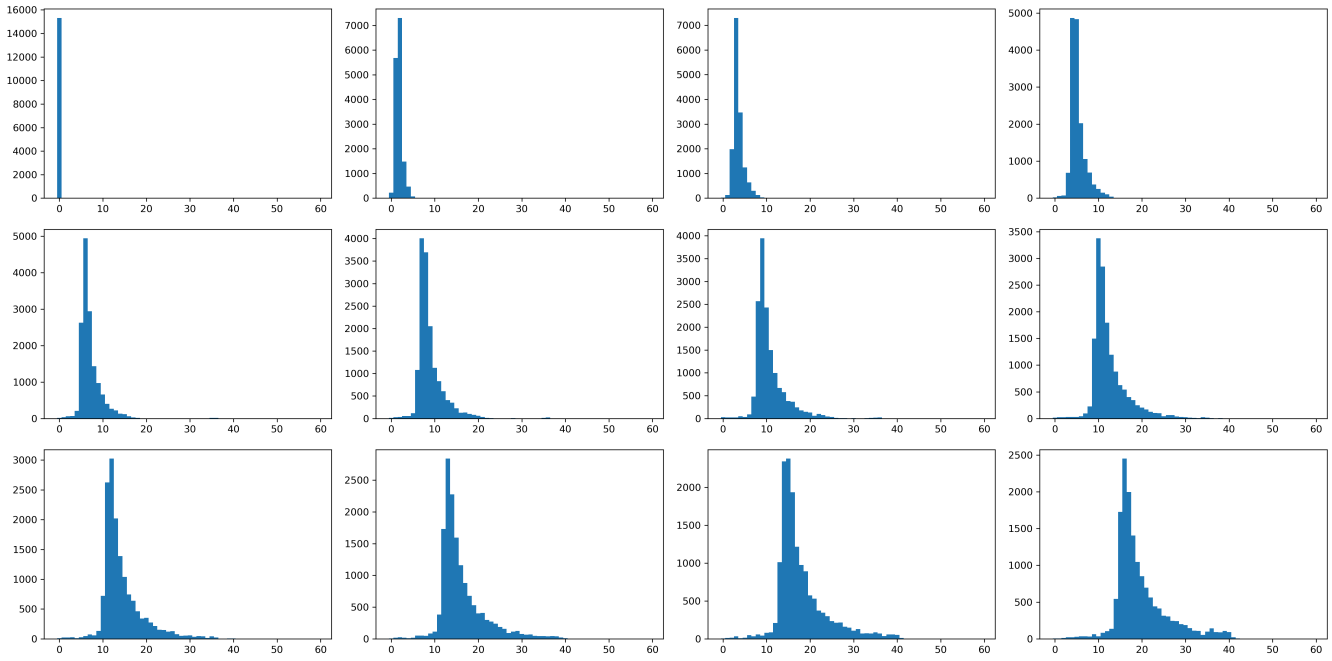


Figure 4-9: Histograms of measured peak shift in units of $1/200th$ μas for each simulated emission region radius. The 2 R_g sized emission region is in the top left corner with increasing radius moving left to right and descending.

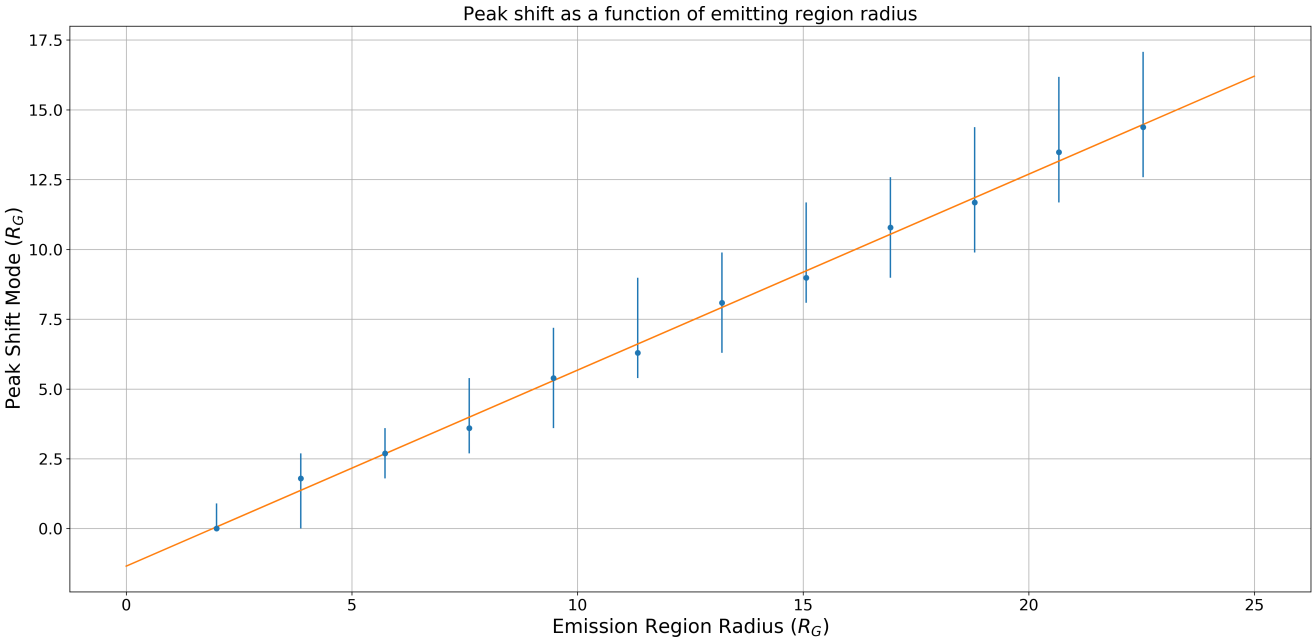


Figure 4-10: Peak shift of each histogram in Figure 4-9 as a function of emission region radius. Error bars display the half maximum measurements around the peak of each histogram. Linear regression yields a line of best fit of $P = 0.702x - 1.346,\ r^2 = 0.998$.

Chapter 5

Conclusion

In this thesis we present `Mirage`, a new package for simulating gravitational microlensing of quasars. We provide an overview of the theory and principles behind gravitational lensing and microlensing. We then explain how `Mirage` takes advantage of developments in cluster computing technology to enable simulation of quasars with arbitrarily sized emitting regions in their accretion disk. Lastly, we provide a sample study of QSO2237+0305 where we determine the radial distance between the x-ray and optical emitting regions of the disk to be at most 8.6 R_g apart. `Mirage` is a program initially developed for analyzing light curves from gravitationally lensed quasars. It is specifically designed for analyzing caustic phenomenon for comparison to QSO2237+0305, and as such includes tools for identifying and characterizing caustic events in light curves. That being said, it is a full-fledged, robust simulator of astrophysical gravitationally lensed objects. Built-in functionality includes:

- Generating images of gravitationally lensed systems at both the microlensing and macrolensing scale.
- Simulating quasar source objects of arbitrary radius.
- Rendering animations of macro- or micro-lensed images as a quasar moves.
- Creating high-resolution magnification maps.
- Generating random or pre-specified light curves, simulating relative motion between the quasar and galaxy.
- A highly customizable star population generator able to model star populations with different initial mass functions at different ages with different proportions of stars in binary systems.
- Simulating light curves through a random starfield where each star has its own unique velocity in the lensing galaxy.
- Bulk generation of magnification maps and light curves with specified variables changing for discovering correlation.
- Tools for statistical analysis of magnification maps and light curves.

- A Computational backend for running large simulations on a computational cluster via Apache Spark^{TM} as well as a backend written in C for local parallel computation.

With `Mirage` we may now explore other questions such as how random motions of stars in a galaxy impacts microlensing. The highly customizable star population generator enables much more nuanced studies of expected frequency of microlensing events and dark matter content of stars, as well as how those two properties change over time as a galaxy ages and stars die.

In addition to the functionality built into `Mirage`, `Mirage`'s code is written to be highly readable and extenable. We utilize an object-oriented approach in the software and multiple design patterns to ensure the code can be modified for specific use cases quickly and reliably. For more information about installing, using, and extending `Mirage` see Section B.

Appendix A

Model Validation

To confirm `Mirage` is computing magnification maps correctly, we compare magnification maps generated with `Mirage` to those generated using Wambsganss' microlensing code. For this purpose, we use `Mirage` to simulate all four images of QSO2237+0305 with the same parameters as used in [21]. Note that Pooley et al. use a different initial mass function from that used in our analysis described in Chapter 4 leading to qualitative differences in the magnification maps shown here and those in Chapter 4. For each image of the quasar we display a magnification map generated by `Mirage`, a magnification map generated by Wambsganss' microlensing code, and histograms of the two images. The comparisons may be found in Figures A-1, A-2, A-3, and A-4.

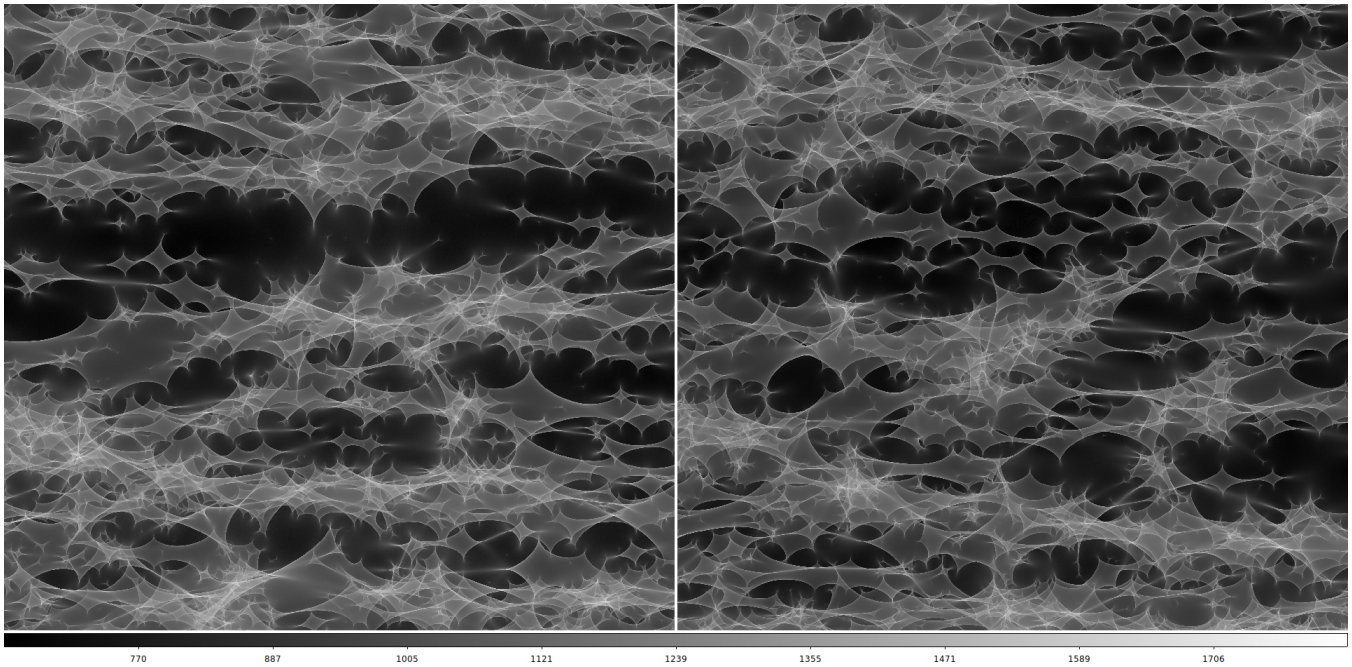
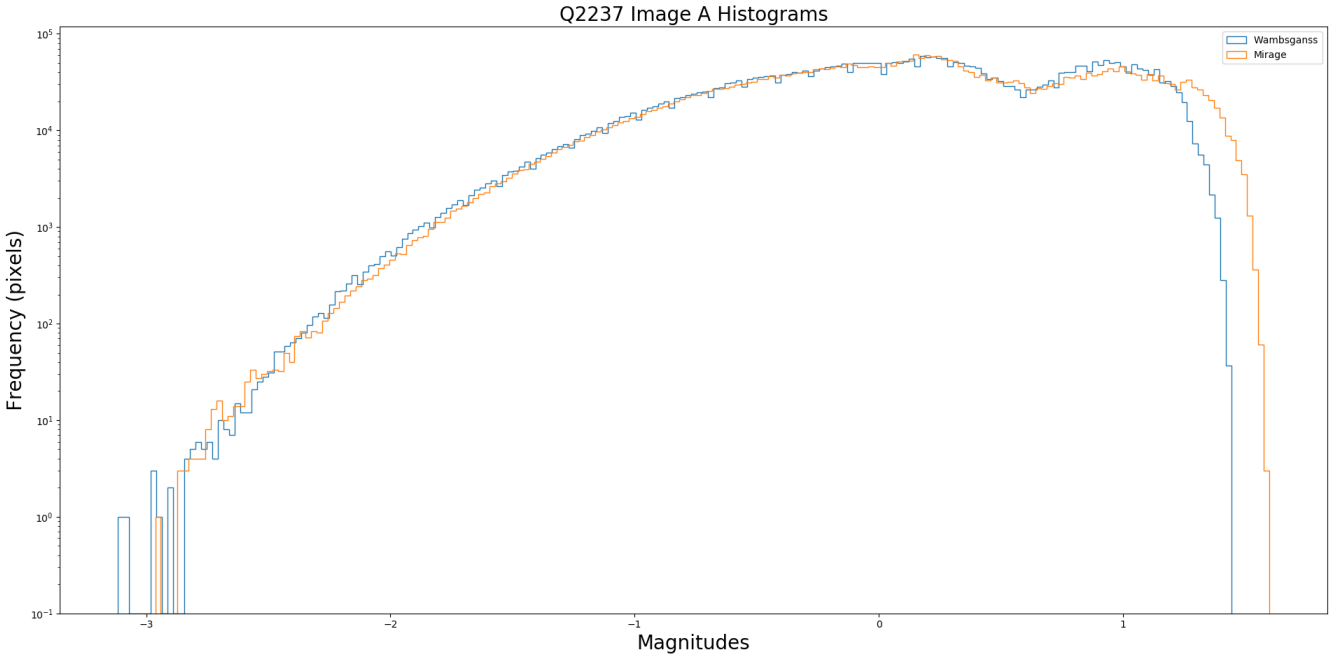


Figure A-1: Comparison of simulated microlensing in image A of QSO2237+0305 by `Mirage` and Wambsganss' microlensing code. Above: Side-by-side comparison of two magnification maps - one generated by Wambsganss' code (right) and the other by `Mirage` (left). The two simulations were set up with identical parameters, although each has its own unique star field. Thus they should not appear identical but exhibit similar patterns and textures. Below: Histograms of the two above images. Histograms provide a more quantitative comparison between the two computational models.



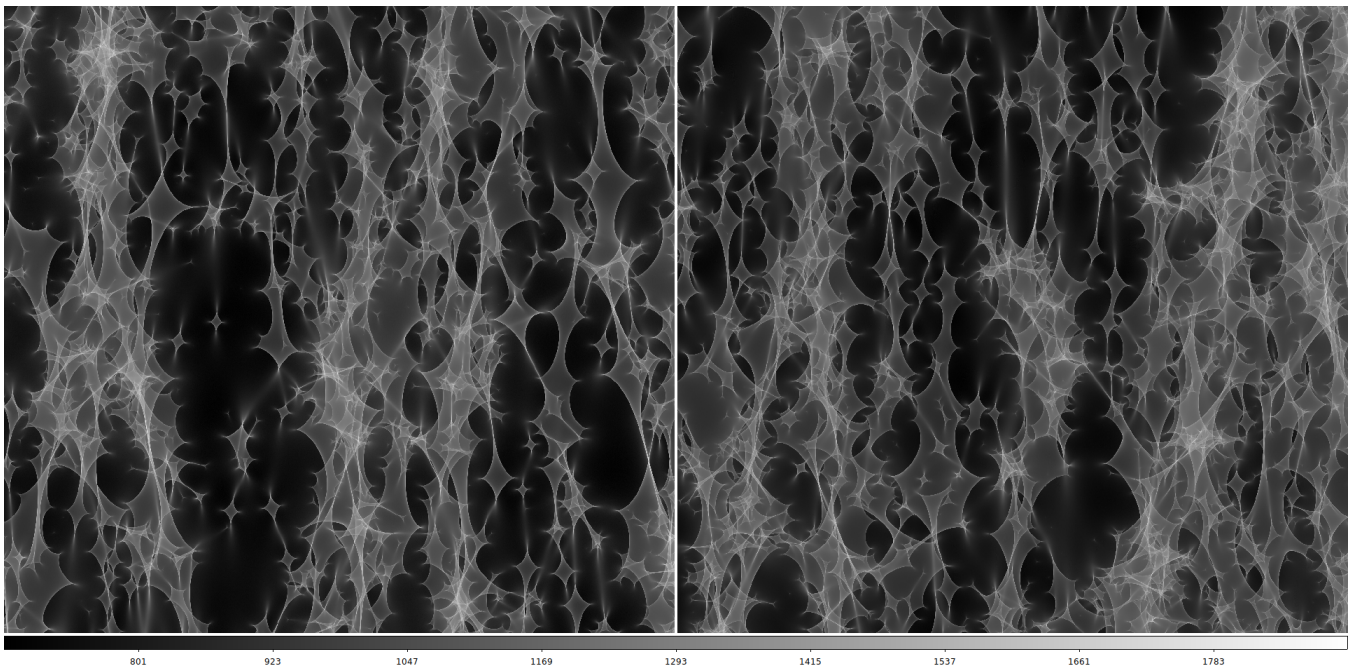
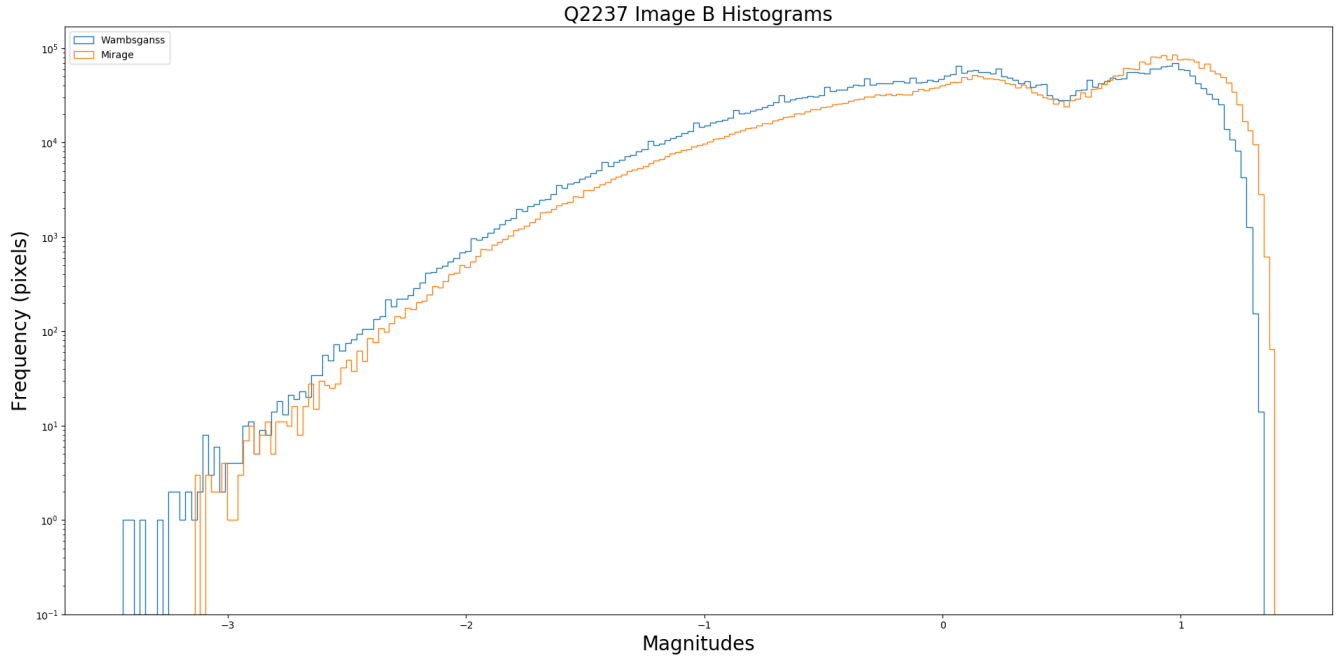


Figure A-2: Comparison of simulated microlensing in image B of QSO2237+0305 by `Mirage` and Wambsganss' microlensing code. Above: Side-by-side comparison of two magnification maps - one generated by Wambsganss' code (right) and the other by `Mirage` (left).



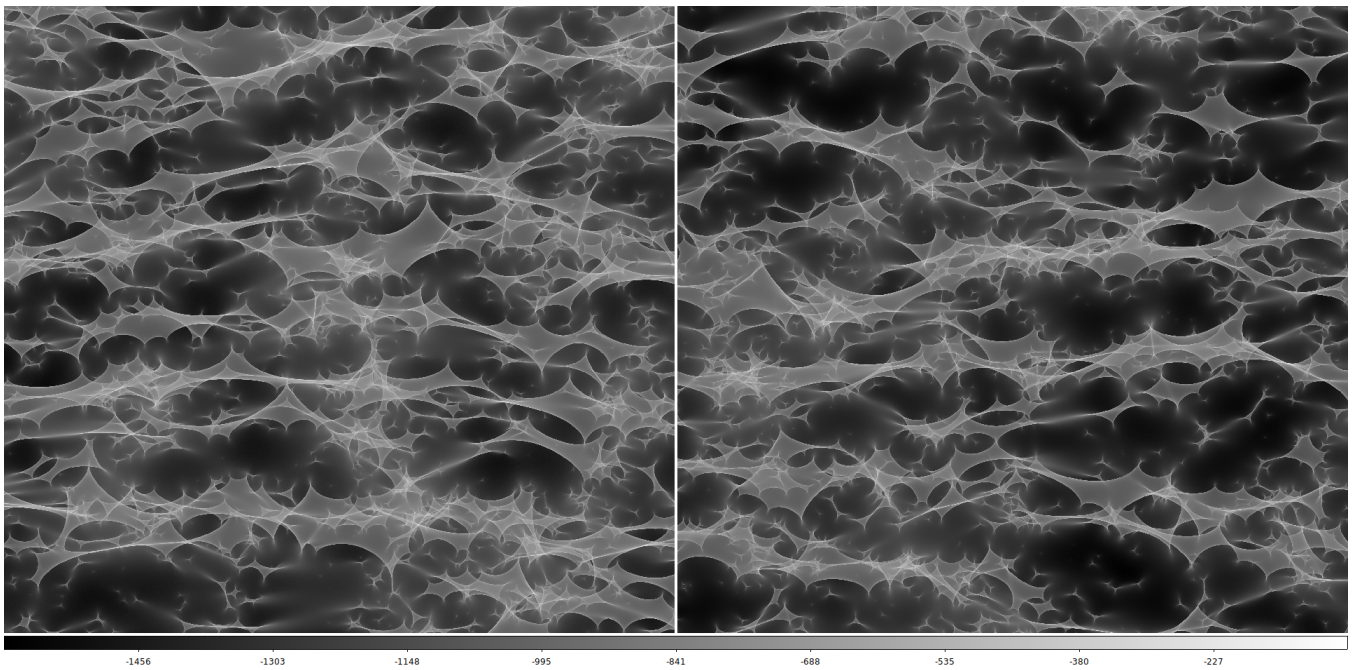
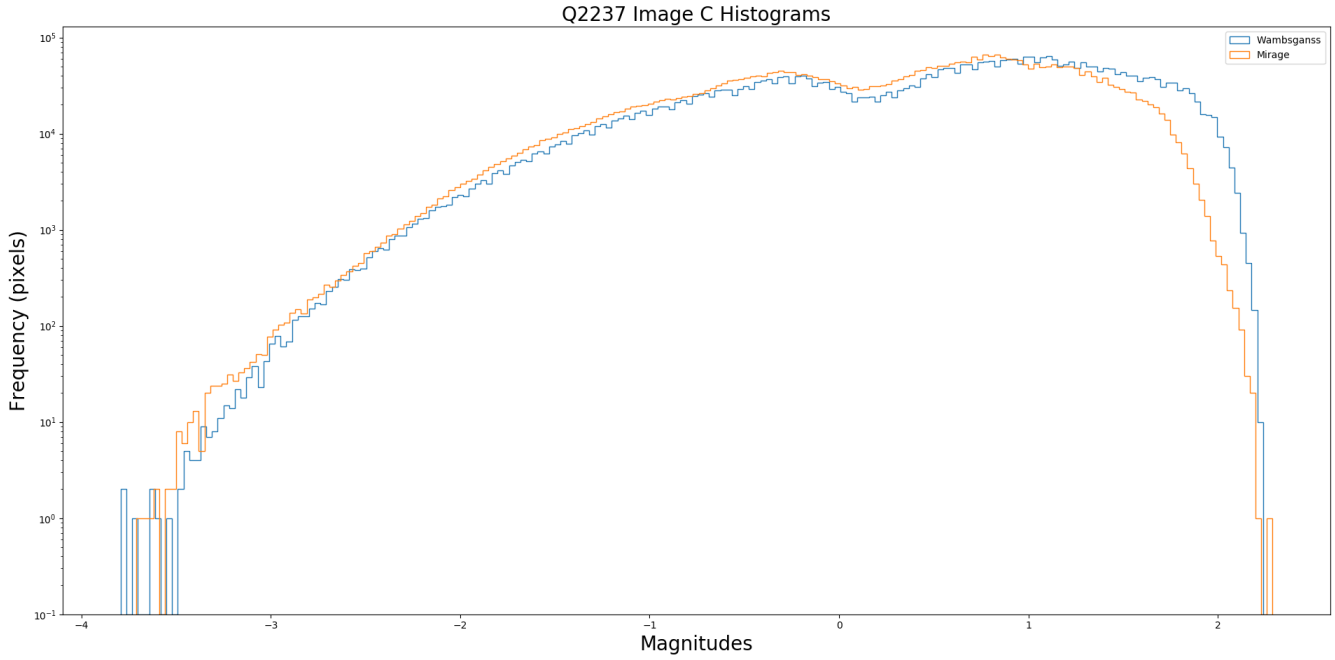


Figure A-3: Comparison of simulated microlensing in image C of QSO2237+0305 by `Mirage` and Wambsganss' microlensing code. Above: Side-by-side comparison of two magnification maps - one generated by Wambsganss' code (right) and the other by `Mirage` (left).



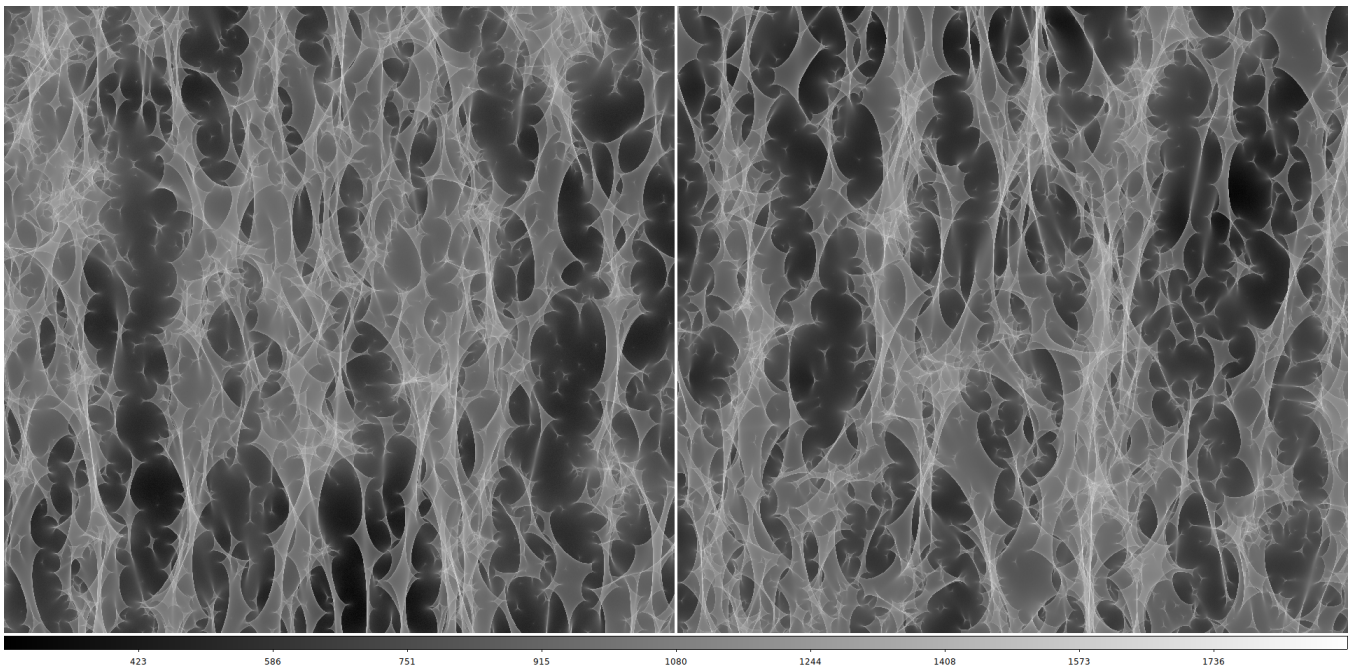
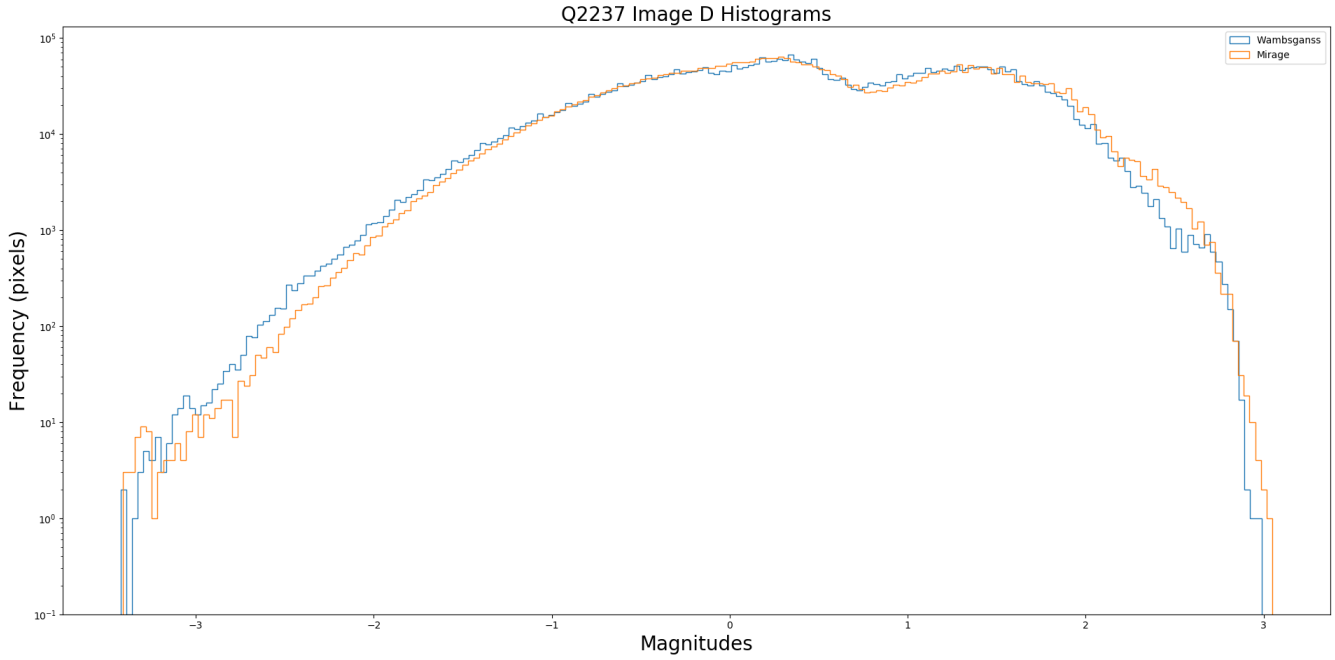


Figure A-4: Comparison of simulated microlensing in image D of QSO2237+0305 by `Mirage` and Wambsganss' microlensing code. Above: Side-by-side comparison of two magnification maps - one generated by Wambsganss' code (right) and the other by `Mirage` (left).



Appendix B

Using `Mirage`

B.1 Accessing the Code

`Mirage` is distributed as an open-source project available on GitHub at
https://github.com/JordanKoeller/Mirage2.

In order to install and use `Mirage` see the most updated installation instructions in the README of the GitHub repository.

B.2 Documentation

Documentation for `Mirage` may be found at http://cs.trinity.edu/~jkoeller/Mirage, as well as a tutorial for performing some basic computations.

B.3 Optimizing the Apache Spark™ Engine

To fully optimize `Mirage` for large computations there are many preferences that need to be set within `Mirage`. As such I list out some of the important parameters to specify for `Mirage` to perform optimally. See `Mirage`'s documentation for information on the relevant flags and functions to specify these parameters. For instructions for setting up a cluster, see the article in Spark's documentation about starting a Spark standalone cluster (https://spark.apache.org/docs/latest/spark-standalone.html). The flags and parameters described here are for clusters configured as Spark standalone clusters. The optimal configurations for Apache Spark™ running on a Mesos, YARN, or Hadoop cluster may be different.

B.3.1 Workers, Executors, and Partitions

We begin by defining some terms used by Spark to describe a cluster.

- **Worker**: A physical computer. A worker is the actual hardware being used to perform computation on.
- **Executors**: The executor is the compute unit of Spark. Each executor is a unique Java Virtual Machine instantiated to encapsulate computation. By default when a Spark cluster receives a simu-

lation to compute an executor will be made for each core on each worker connected to the cluster. As an example, for a cluster containing 4 workers with a 4-core CPU in each machine 16 executors will be initialized (one for each CPU core) and begin computing. Each executor has its own memory space. Executors can be configured to use more than one core of the CPU.

- **Partitions** A partition is a segment of an `RDD` containing a part of the total data in the `RDD`. When an `RDD` is computed all the executors begin computing on partitions of the `RDD`. If an executor completes the computation on its partition it begins computing another partition that was waiting for resources to become available. As such, partitions are dynamic, and can be shuffled around the executors in a cluster as resources become available. Once all partitions have been computed the `RDD`'s computation is complete and Spark can move to the next computation. If an executor is configured to use more than one CPU core it will accept multiple partitions at a time and compute them concurrently.

In order to get the most performance out of Spark, the optimal cluster is configured with approximately 4 to 8 CPU cores per executor. Additionally, optimal cluster configurations do not use all cores available on each worker. It is advised that at least one core remain unused per worker for the worker to perform operating system related tasks without interrupting `Mirage`. As an example, for our cluster consisting of workers with 32 CPU cores each, we configure Spark to create 5 executors with 6 cores per executor on each worker. Thus we utilize 30 cores and leave two cores untouched. When we configured Spark to use all 32 cores by having 4 executors with 8 cores per executor per worker we saw notably worse performance.

In addition to the number of executors and cores per executor being important, the number of partitions `RDD`s are broken up into can have a considerable impact on performance. Having many partitions enables Spark to load-balance more precisely. However, it also necessitates more network traffic which is intrinsically slow as the workers communicate with each other in the cluster. In practice, the optimal number of partitions was found to be approximately 3 to 4 times as many cores utilized by the cluster. As an example, for our cluster utilizing 546 cores (78 executors, 7 cores each) we found the best performance when `RDD`s were configured to have approximately 2000 partitions.

B.3.2 Memory and Data Management

The largest bottleneck on performance of `Mirage` is the amount of memory available to the cluster [16]. For high-quality simulations we often produced 100s of GB of data while producing the distributed kD-Tree. We often ran into `OutOfMemoryError` exceptions while developing `Mirage`. As a result, we configure Spark to store `RDD` partitions on disk, rather than in memory. While saving data to disk is typically catastrophic for performance, because of Spark's `RDD` model the ramifications on performance of saving to disk are actually quite small. After an executor finishes computing a partition, it caches the partition to disk before starting to compute the next partition. Thus the entirety of the distributed kD-Tree never exists in memory at a time. On average, approximately 1/4 of the distributed kD-Tree will exist in memory at the same time. Furthermore, Spark includes an option to replicate partitions across the cluster. Whenever a partition is computed the executor saves the data to disk and also copies the data to another worker that also saves the disk. Thus if the original worker disconnects from the cluster all lost segments of the distributed kD-Tree are backed up somewhere else in the cluster and Spark can recover within a matter

of seconds. While redundancy is not typically necessary for most clusters, we found it quite useful as it enabled us to add classroom computers as workers in the cluster. If a student wanted to use a classroom computer that we were using as a worker the worker could disconnect and reconnect after the student was finished with minimal penalty to the simulation's runtime.

With all these considerations in mind, we recommend using a smaller ray number of rays while testing the methodology of a proposed study, such that the distributed kD-Tree may be stored in memory. This allows for fast testing and refining of the methodology. Once a methodology has been developed we recommend increasing the number of rays and switching to storing the distributed kD-Tree in disk space (with or without backups) to produce publication-quality results.

Bibliography

[1] Stanley L. Jaki. "Johann Georg von Soldner and the gravitational bending of light, with an English translation of his essay on it published in 1801". In: *Foundations of Physics* 8 (Dec. 1978), pp. 927–950. DOI: 10.1007/BF00715064.

[2] Frank Watson Dyson, Arthur Stanley Eddington, and C. Davidson. "IX. A determination of the deflection of light by the sun's gravitational field, from observations made at the total eclipse of May 29, 1919". In: *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 220.571-581 (1920), pp. 291–333. DOI: 10.1098/rsta.1920.0009. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.1920.0009. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.1920.0009.

[3] C.S. Kochanek, E.E. Falco, C. Impey, J. Lehar, B. McLeod, H.-W. Rix. *CASTLES Survey*. URL: https://www.cfa.harvard.edu/castles/.

[4] Ramesh Narayan and Matthias Bartelmann. "Lectures on Gravitational Lensing". In: (1996). arXiv: 9606001 [astro-ph]. URL: http://arxiv.org/abs/astro-ph/9606001.

[5] Bohdan Paczynski. "Gravitational microlensing at large optical depth". In: *ApJ* 301 (1986), p. 503. ISSN: 0004-637X. DOI: 10.1086/163919. URL: http://adsabs.harvard.edu/abs/1986ApJ...301..503P.

[6] Joachim Wambsganss. *Gravitational microlensing*. 1990. arXiv: 9709212 [astro-ph].

[7] C. S. Kochanek. "The Saas Fee Lectures on Strong Gravitational Lensing". In: (2004), pp. 1–183. DOI: 2006glsw.conf.....M. arXiv: 0407232 [astro-ph]. URL: http://arxiv.org/abs/astro-ph/0407232.

[8] J. Pflamm-Altenburg. *Libimf*. Astrophysics Source Code Library. June 2012. ascl: 1206.009.

[9] P. Kroupa. "On the variation of the initial mass function". In: *Monthly Notices of the Royal Astronomical Society* 322 (Apr. 2001), pp. 231–246. DOI: 10.1046/j.1365-8711.2001.04022.x.

[10] A. M. Price-Whelan et al. "The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package". In: *Astrophyscial Journal* 156, 123 (Sept. 2018), p. 123. DOI: 10.3847/1538-3881/aabc4f.

[11] S. van der Walt, S. C. Colbert, and G. Varoquaux. "The NumPy Array: A Structure for Efficient Numerical Computation". In: *Computing in Science Engineering* 13.2 (Mar. 2011), pp. 22–30. ISSN: 1521-9615. DOI: 10.1109/MCSE.2011.37.

[12] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001–. URL: http://www.scipy.org/.

[13] J. D. Hunter. "Matplotlib: A 2D Graphics Environment". In: *Computing in Science Engineering* 9.3 (May 2007), pp. 90–95. ISSN: 1521-9615. DOI: 10.1109/MCSE.2007.55.

[14] F. Perez and B. E. Granger. "IPython: A System for Interactive Scientific Computing". In: *Computing in Science Engineering* 9.3 (May 2007), pp. 21–29. ISSN: 1521-9615. DOI: 10.1109/MCSE.2007.53.

[15] S. Behnel et al. "Cython: The Best of Both Worlds". In: *Computing in Science Engineering* 13.2 (Mar. 2011), pp. 31–39. ISSN: 1521-9615. DOI: 10.1109/MCSE.2010.118.

[16] Jordan Koeller, Mark Lewis, and David Pooley. "Applications of Apache Spark(TM) for Numerical Simulation". In: *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA 2018, Las Vegas, Nevada, USA, July 30-August 2, 2018, Volume 2*. 20018, pp. 139–144.

[17] *The Apache Spark^{TM} website*. URL: https://spark.apache.org.

[18] Matei Zaharia et al. "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing". In: *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. San Jose, CA: USENIX, 2012, pp. 15–28. ISBN: 978-931971-92-8. URL: https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/zaharia.

[19] William H. Press et al. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 3rd ed. New York, NY, USA: Cambridge University Press, 2007. ISBN: 0521880688.

[20] Alexey Abramov. *Kd-tree and Nearest neighbor (NN) search (2D case)*. 2014. URL: https://salzis.wordpress.com/2014/06/28/kd-tree-and-nearest-neighbor-nn-search-2d-case/ (visited on 04/18/2019).

[21] David Pooley et al. "X-ray and optical flux ratio anomalies in quadruply lensed quasars. II. Mapping the dark matter content in elliptical galaxies". In: *Astrophysical Journal* 744.2 (2012). ISSN: 15384357. DOI: 10.1088/0004-637X/744/2/111. arXiv: 1108.2725v1.

[22] Paul Schechter. private communication. Apr. 18, 2019.

[23] N I. Shakura and R A. Sunyaev. "A Theory of the Instability of disk Accretion on to Black Holes and the Variability of Binary X-ray Sources, Galactic Nuclei and Quasars*". In: *Monthly Notices of the Royal Astronomical Society* 175 (May 1976), pp. 613–632. DOI: 10.1093/mnras/175.3.613.

[24] David Pooley. private communication. Apr. 18, 2019.

[25] Chuck Keeton. "Software for Gravitational Lensing". In: (2004).

[26] Irwin Sobel. "An Isotropic 3x3 Image Gradient Operator". In: *Presentation at Stanford A.I. Project 1968* (Feb. 2014).

[27] A. Kogut et al. "Dipole Anisotropy in the COBE Differential Microwave Radiometers First-Year Sky Maps". In: *apj* 419 (Dec. 1993), p. 1. DOI: 10.1086/173453. eprint: astro-ph/9312056.