Raytracing a Black Hole Image

Ronan Hanley, Cassie Metzger, & Jeremy Robin

1 Methods

We began this project by first implementing the metric equations of general relativity. We did so in a header file titled `EoM.h`. Within this header file, we created a public class called `boyer_liquidist_metric`. This class allowed us to call our metric equations from any file within the repository. Then, we created a function titled computer_metric in order to calculate the metric equations given `a0`, `M0`, and two additional variables required by the method, In order to simplify our calculations, we defined the following:

$$\frac{d\rho^2}{dr} = 2r \quad \frac{d\Delta}{dr} = 2r - 2M$$

$$\frac{d\Sigma}{dr} = (4r)r^2a^2 - a^2\sin\theta^2\frac{d\Delta}{dr}$$

$$\frac{d\rho^2}{d\theta} = -a^2\sin 2\theta \quad \frac{d\Delta}{d\theta} = 0$$

$$\frac{d\Sigma}{d\theta} = -a^2\Delta\sin 2\theta$$

We also defined an intermediate step in order to avoid repeating the same calculation:

$$z = \rho^2\Delta \quad \frac{dz}{dr} = \Delta\frac{d\rho^2}{dr} + \rho^2\frac{d\Delta}{dr}$$

$$\frac{dz}{d\theta} = \Delta\frac{d\rho^2}{d\theta} + \rho^2\frac{d\Delta}{d\theta}$$

Using the values above, we then calculated the derivatives that appeared directly in the equations of motion for a photon.

$$\frac{d\alpha}{dr} = \frac{1}{2\alpha}\frac{\Sigma\frac{dz}{dr} - z\frac{d\Sigma}{dr}}{\Sigma^2}$$

$$\frac{d\alpha}{d\theta} = \frac{1}{2\alpha}\frac{\Sigma\frac{dz}{d\theta} - z\frac{d\Sigma}{d\theta}}{\Sigma^2}$$

$$\frac{d\beta^\phi}{dr} = -2Ma\frac{\Sigma - r\frac{d\Sigma}{dr}}{\Sigma^2}$$

$$\frac{d\beta^\phi}{d\theta} = 2Mar\frac{\frac{d\Sigma}{d\theta}}{\Sigma^2}$$

$$\frac{d\gamma_{rr}}{dr} = \frac{\rho^2\frac{d\Delta}{dr} - \Delta\frac{d\rho^2}{dr}}{\rho^2}$$

$$\frac{d\gamma_{rr}}{d\theta} = \frac{\rho^2\frac{d\Delta}{d\theta} - \Delta\frac{d\rho^2}{d\theta}}{\rho^2}$$

$$\frac{d\gamma_{\theta\theta}}{dr} = -\frac{\frac{d\rho^2}{dr}}{\rho^2}$$

$$\frac{d\gamma_{\theta\theta}}{d\theta} = -\frac{\frac{d\rho^2}{d\theta}}{\rho^2}$$

$$\frac{d\gamma_{\phi\phi}}{dr} = \frac{\Sigma\frac{d\rho^2}{dr} - \rho^2\frac{d\Sigma}{dr}}{\Sigma^2\sin\theta}$$

$$\frac{d\gamma_{\phi\phi}}{d\theta} = \frac{\Sigma(\sin\theta)^2\frac{d\rho^2}{d\theta} - \rho^2(\sin\theta)^2\frac{d\rho^2}{d\theta} + \Sigma\sin 2\theta}{\Sigma^2(\sin\theta)^4}$$

2 Implementing the Metric Equations

To begin a simulation, we defined any variables that were known numerically (ex. $a = 0$). Then, we called the `boyer_lindquist_metric`. We utilized the function `auto dydx = [&metric] (double x, const std::vector<double> &y` where y[0] is r, y[1] is θ, y[2] is ϕ and so on. Using the equations defined above, we defined the 6 equations of motion. We then used the adaptive version of 4th order runge kutta (RK4) in order to solve this system. We implemented adaptive RK4 under the header `rk4_adaptive.h` and solved the differential equations by calling the integrate function from this header.

3 Einstein Ring in Schwarzschild Spacetime

In the file `implement.cpp` we wrote a code to portray the case where a photon passes close enough to a non-spinning black hole ($a = 0$) that the photon is pulled around the black hole, but is not sucked in. Therefore, the photon returns to the observer, creating an Einstein Ring around the black hole. This only occurs when the photon is shot at a specific angle, known as a critical angle. We tested two different scenarios. For both, we stopped RK4 integrated when $r < 2.01$. The black hole has a radius of 2, therefore, we choose $r < 2.01$ because if we'd chosen 2 as a stop condition the photon would approach the black hole forever. Choosing $r < 2.01$ allows the photon to fall into the black hole. RK4 is then stopped as the photon has entered the black hole and it is no longer relevant to track the photon's path. The stop condition for RK4 also depended on ϕ. However, this ϕ value changed depending on the total distance the photon was traveling and therefore changed for the two different cases.

3.1 Case 1

In this case, $r_{obs} = 10M$ while the critical angle is $\xi = 0.48857874$. This scenario can be plotted in 2D space with the photon traveling a total distance of $\phi = 2\pi$. As a result, the stop condition for RK4 was $\phi < 2\pi$ since after 2π the photon had completed its journey around the black hole and returned to the observer.

In a perfect case, we would not expect a difference between the initial and final differences. Therefore,

we conclude that this difference is an error due to rk4 adaptive integration. We can calculate this error using interpolation.
The photon's second to last coordinates are (9.92152, -0.0466267) and the final coordinates are (10.1119, 0.0665194). The slope between these coordinates can be calculated as 0.5943171552. Using ϕ, $y = 0$, we can calculate x as 9.999974238.
We can calculate the error using the equation $\frac{|r - r_o|}{|r_o|}100\%$. We can define r as our calculated final x and r_0 as 10, our desired final x. We find that the error is $2.57618403 \times 10^{-4}\%$. This is an error on the order of 10^{-6}. Therefore, this error is minimal for one orbit, but when the photon completes more than one orbit, this error increases.

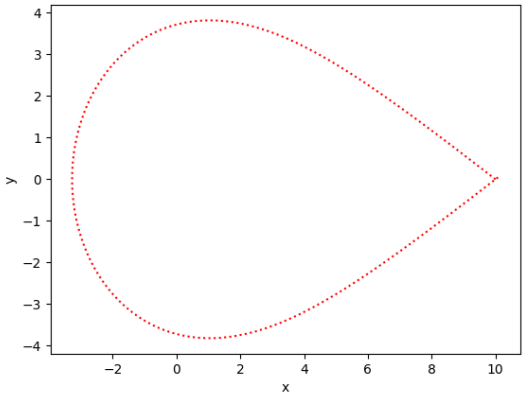

Figure 3.1: Case 1

3.2 Case 2

Now, $r_{obs} = 20M$ and $\xi = 0.24904964$. In this case, the photon travels a total distance of $\phi = 4\pi$. As a result, we change the rk4 adaptive stop condition so that rk4 adaptive will continue until the photon has traveled a distance of 4π and returned to the observer. In this situation, the photon completes two orbits around the black hole. Following the same procedure, we can calculate the error of rk4 adaptive integration by finding the difference between the initial and final positions. The initial coordinates are $(19.3737, -0.0980879)$ and the final coordinates are $(20.2442, 0.138635)$. This makes the slope 0.2685255186. We find that when $\phi = 2\pi$ and $y = 0$, $x = 19.99892146$. Using the same equation, we find that the error is 0.00539272365%. As a result, we can see that the more orbits the photon completes, the more the integration error increases.

4 Unstable Spherical Photon Orbits

Again in the file `implement.cpp`, we plot a variety of unstable orbits. This time, we plot these orbits in

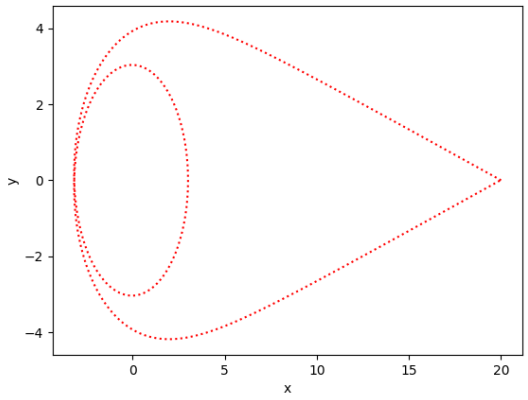

Figure 3.2: Case 2

3D. We stop our adaptive rk4 adaptive integration when $r_{photon} > R + \frac{R}{5.0}$ or when $r_{photon} < R - \frac{R}{5.0}$. Thus, we stop plotting the photon's path when it becomes significantly greater than its starting point (therefore going to infinity) or when it becomes much smaller and therefore falls into the black hole. From our file `test_2.ipynb`, we recover 3D plots for each unstable orbit. We note that some of the photon

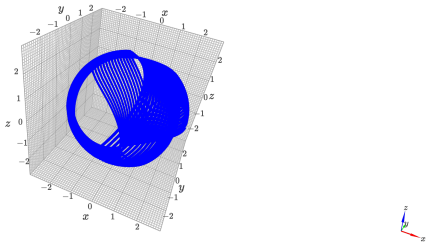

Figure 4.1: Test Case A

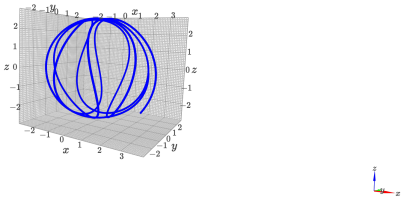

Figure 4.2: Test Case B

trajectories appear choppier than others. This may be due to the limitations of adaptive rk4 adaptive, or, due to an unfound rounding/type error in our algorithm. We plot the fractional error on the radius as a function of time. We express the fractional error as $\frac{r-r_0}{r_0}$. We can compare how the error increases depending on how quickly the order becomes unstable.

As seen in Figure [4.6], the fractional error of each orbit will remain minimal for a period of time before

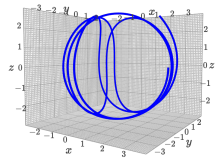

Figure 4.3: Test Case C

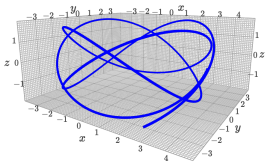

Figure 4.4: Test Case D

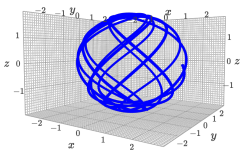

Figure 4.5: Test Case E

increasing rapidly. This rapid increase is an unfortunate drawback of using the rk4 adaptive algorithm. Once the photon diverges from the orbit, the fractional error increases rapidly and the algorithm is no longer valid. However, for all six unstable orbits we were able to trace more than one complete orbit before the algorithm diverged, which suggests that our methods are sufficient for raytracing.

5 Creating a Black Hole Image

After constructing several test cases, we move on to producing the image of a black hole, this is done in the file `raytracing.cpp`. In order to do this, we visualize a screen in space. The screen's length and width can be translated to the length and width of the image produced. The screen rests a certain distance away from the black hole. Based on the length and width of the screen, a number of pixels can be set, the greater the number of pixels, the higher the resolution. A photon is then shot from each pixel, similar to the two earlier test cases. However, in this case, only the photon's final location matters. The photon's final location is plotted in `Python` in

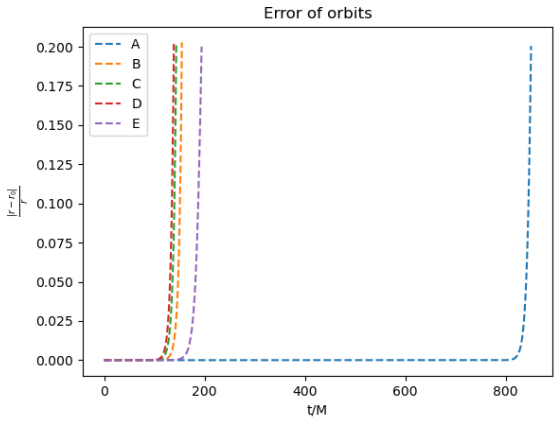

Figure 4.6: Comparison of the fractional errors of each unstable orbit

the file `raytracing.ipynb`. However, by just doing this, each pixel whose photon falls within the accretion disk remains the same color. In order to better understand the dynamics of the accretion disk, we can apply the redshift equations. By doing this, portions of the accretion disk that are moving toward the observer appear brighter while portions that are moving away from the observer appear darker.

We can compare two different images both with resolution 400×160, both with the observer at angle of $85°$, one has a spin value of $a = 0.99$ and the other has a spin value of $a = 0$. In both images, we clearly see an accretion disk encircling a black hole. The black hole is the same intensity as our background showing that our algorithm does not plot intensity values for photons that are falling into the black hole as their light cannot escape. We also note that in Figure 9, the left side of the black hole appears brighter. We speculate that this is due to redshift. Higher pixel intensity indicated higher redshift from the Doppler effect, whereas lower pixel intensity indicates blueshift. We expect that when the black hole is spinning faster, the photons are moving towards the observer much faster. Therefore, they will appear brighter. On the other hand, the right side of the black hole appears darker, indicating that the photons on the right side are moving away from the viewer. The Doppler effect is, as expected, more significant in a black hole that is spinning faster, as a higher angular velocity would increase redshift and blueshift. A vertical black line appears in both images. We note that this is likely related to our equations of motion. To better understand the characteristics of our black hole, we can plot several different images for varying a values and varying radii of the accretion disk (`Rout`). All of these images can be generated at different angles.

In Figure [5.2], we notice a small red line trailing into the black hole. Traces of this same ring can be seen in Figures [5.1], [5.3], and [5.4]. Composed of

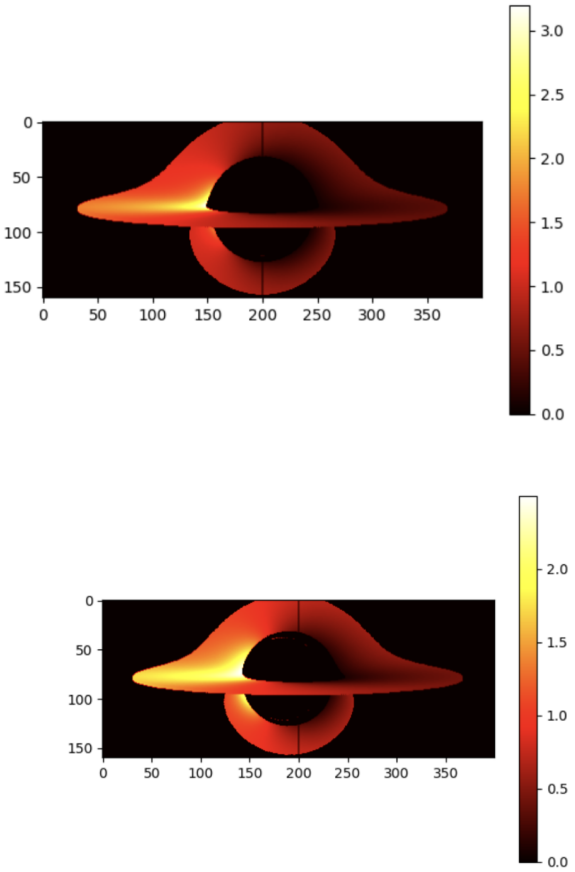

Figure 5.1: Both images display black holes that are viewed by an observer at 85°. The top image displays a black hole that is not spinning ($a = 0$) and the bottom image displays a black hole that is spinning rapidly ($a = 0.99$)

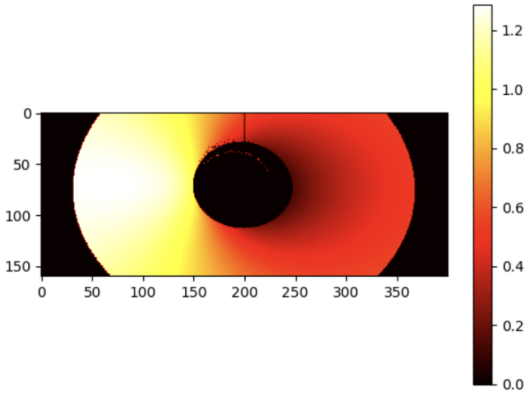

Figure 5.2: Viewed with an observer at 40°. This black hole has a spin parameter of 0.99

photons that orbit the black hole in the near-field region before escaping to infinity, the photon ring seems to be a universal feature among black holes and has recently become a topic of interest to theorists. We expect that this path may be clearer in a higher-resolution image, we could create this by increasing our number of pixels, however, this would increase our runtime.

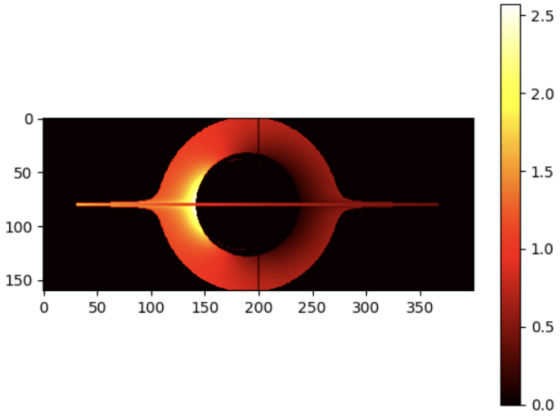

Figure 5.3: Viewed with an observer at 90°. This black hole has a spin parameter of 0.99

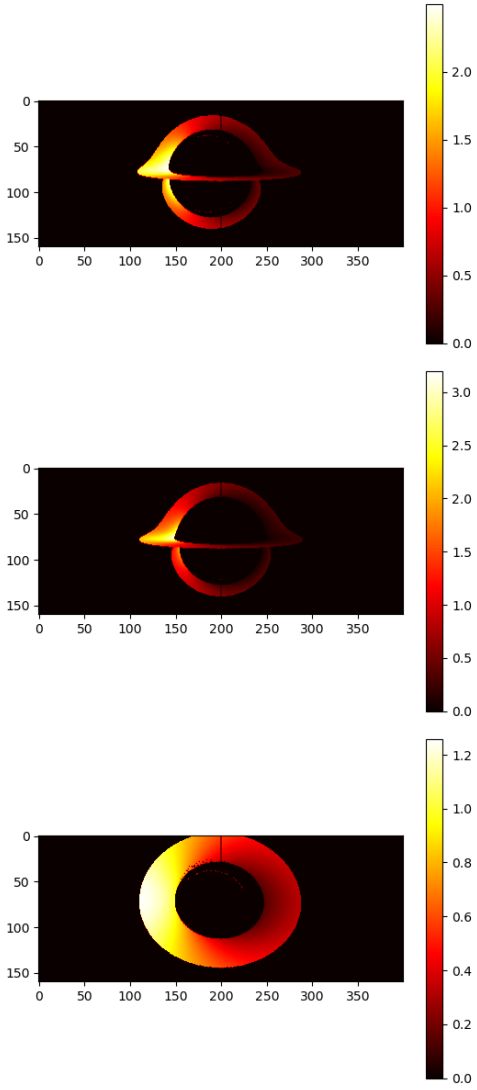

Figure 5.4: The black hole where `Rout` is equal to 10.0 while `Rin` remains as 5.0, therefore the difference between `Rout` and `Rin` is decreased. The top and bottom images have $a = 0.99$ while the middle image has $a = 0.0$

It is also important to note that in Figure [5.2], there is some amount of black speckling in the accretion

disk near to the center. This is not a genuine result, and rather is an artifact of RK4 adaptive. The speckling was more pronounced when we ran our code with a higher tolerance in RK4 adaptive and was reduced when we increased tolerance. Increasing the tolerance further would likely completely eliminate these speckles.

Overall, we find that our algorithm works adequately. We produce intensity values for photons whose final values fall within the accretion disk of the black hole. When our black hole is spinning, we receive higher intensity values for photons that are moving toward the observer, showing that our redshift equations are functioning well. We may receive a better image and display features of the black hole better with a higher resolution image. We find that by decreasing the size our black hole, our redshift does not appear to change at any angle. No matter the size of the black hole, the equations hold true.

However, our equations of motion could be improved to account for the vertical black line that runs through the center of our image. In addition, using a higher-order integration method would make our code more accurate. As seen in Section 4, adaptive RK4 is functional, but its error value blows up when the photon diverges from a stable orbit. We also see that the more distance the photon travels, the greater the error is (Section3). We could improve this by implementing RK4 adaptive5 or another method.

Overall, we can describe notable characteristics of our black hole with the following image. This shows

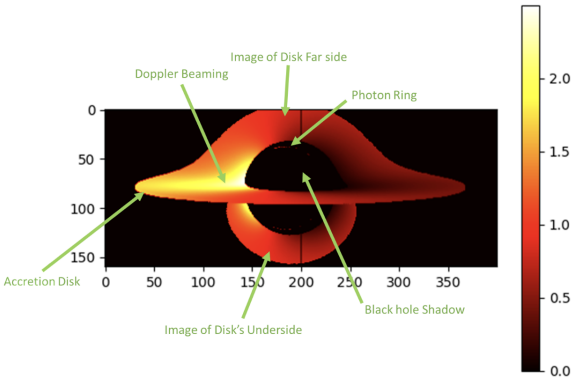

Figure 5.5: Figure [5.1] with arrows pointing towards important characteristics of a black hole that our image accurately displays

that our code accurately uses the metric equations of general relativity to depict the situation where photons are aimed at a black hole. Our image accurately captures the accretion disk surrounding the central black hole and the Doppler beaming that occurs as a result of the motion of photons around the black hole.