University of Cambridge

Part II CATAM Project

14.1 Particle or Photon Orbits near a Black Hole

Academic Year:
2024–2025

Contents

Introduction

In this project, we investigate the motion of particles and photons in the Schwarzschild spacetime geometry. Starting from the effective Lagrangian for geodesic motion, we derive the equations governing the orbits and proceed to analyse them numerically using appropriate integration schemes.

1 Theory

1.1 Derivation of the Orbital Equation of Motion

We begin with the Schwarzschild metric expressed through the Lagrangian:

$$\mathcal{L} = F(r)\dot{t}^2 - \frac{\dot{r}^2}{F(r)} - r^2\dot{\phi}^2, \tag{1.1}$$

Where:

$$F(r) = 1 - \frac{1}{r} \tag{1.2}$$

with units such that $c = 1$ and $2GM = 1$. The affine parameter s parametrises the geodesics, and dots denote derivatives with respect to s. Note, the affine parameter, s, represents proper time, τ, when referring to a massive particle.

Upon computing the Euler-Lagrange equations for t and ϕ respectively, we obtain the following results:

$$F(r)\dot{t} = const = k, \quad r^2\dot{\phi} = const = h, \tag{1.3}$$

where k is a measure of the energy of the particle as measured by a stationary observer and h can be interpreted as the specific angular momentum.

The geodesic equation satisfies:

$$\mathcal{L} = \epsilon, \tag{1.4}$$

where

$$\epsilon = \begin{cases} 1, & \text{for a particle (timelike geodesic)} \\ \\ 0, & \text{for a photon (null geodesic)} \end{cases}.$$

Substituting the conserved quantities into the Lagrangian, in order to remove all dependence on t and ϕ, yields:

$$\dot{r}^2 + V_{\text{eff}}(r) = k^2. \tag{1.5}$$

Where $V_{\text{eff}}(r)$, shown in Figure 7, is the effective potential, making the problem analogous to the classical mechanics approach, with a relativistic correction to the gravitational potential:

$$V_{\text{eff}} = F(r)\left(\epsilon + \frac{h^2}{r^2}\right) \tag{1.6}$$

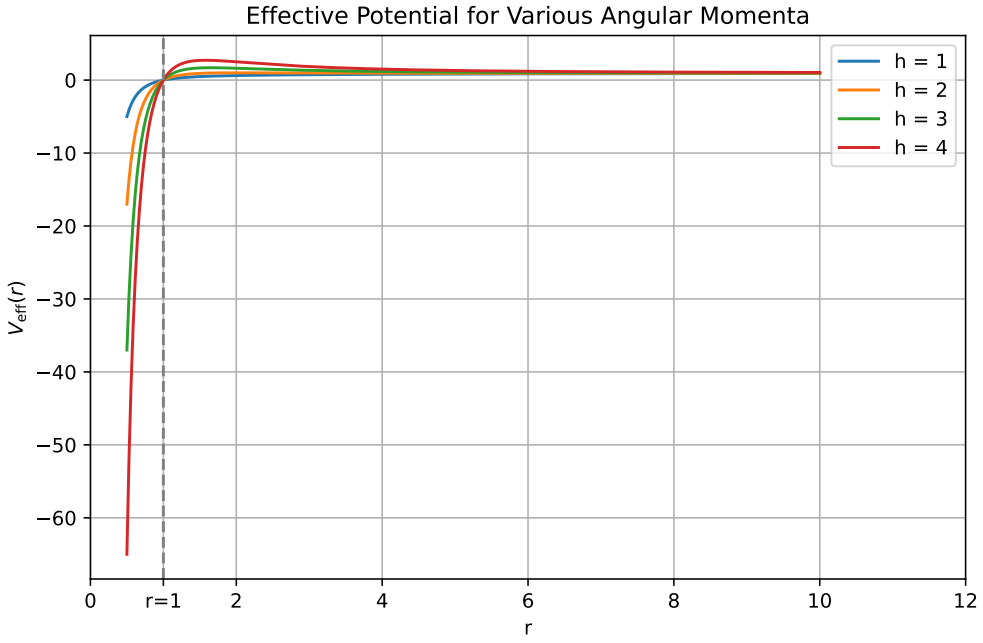

Figure 1: Effective potential as a function of radius for different angular momentum values.

1.2 Change of Variables: Introducing $u = 1/r$

Now we will manipulate (1.5) to replace all dependency on r with u. Additionally we will convert from a first derivative of r with respect to s into a second derivative in u with respect to s, in order to define an equation of motion in u.

Define:

$$u = \frac{1}{r}. \tag{1.7}$$

Then:

$$\dot{r} \equiv \frac{dr}{ds} = \frac{dr}{d\phi}\frac{d\phi}{ds} = \frac{dr}{d\phi}\dot{\phi} \tag{1.8}$$

Since:

$$\dot{\phi} = \frac{h}{r^2} = hu^2, \tag{1.9}$$

we find:

$$\dot{r} = hu^2\frac{dr}{d\phi}. \tag{1.10}$$

Differentiating u gives:

$$\frac{du}{d\phi} = \frac{du}{dr}\frac{dr}{d\phi} = -\frac{1}{r^2}\frac{dr}{d\phi} \quad \Rightarrow \quad -\frac{1}{u^2}\frac{du}{d\phi} = \frac{dr}{d\phi} \tag{1.11}$$

Then substituting (1.11) into (1.10):

$$\dot{r} = -h\frac{du}{d\phi}. \tag{1.12}$$

Substituting into the radial equation (1.5) for r and $\dot{r}$:

$$h^2\left(\frac{du}{d\phi}\right)^2 = k^2 - (1-u)\left(\epsilon + h^2u^2\right). \tag{1.13}$$

Expanding, simplifying and dividing by h^2:

$$(\frac{du}{d\phi})^2 = \frac{E^2}{h^2} - \frac{\epsilon(1-u)}{h^2} - u^2(1-u) \tag{1.14}$$

In order to convert from first to second derivatives, define:

$$f(u) = (\frac{du}{d\phi})^2 \tag{1.15}$$

And differentiating both sides wrt ϕ:

$$\frac{df}{du}\cdot\frac{du}{d\phi} = 2\frac{du}{d\phi}\cdot\frac{d^2u}{d\phi^2} \tag{1.16}$$

Therefore:

$$2\frac{d^2u}{d\phi^2} = \frac{df}{du} \tag{1.17}$$

Now, compute the derivative of $f(u)$:

$$(\frac{du}{d\phi})^2 = f(u) = \frac{k^2}{h^2} - \frac{\epsilon(1-u)}{h^2} - u^2(1-u) \tag{1.18}$$

$$\Rightarrow \frac{df}{du} = \frac{\epsilon}{h^2} - (2u - 3u^2) \tag{1.19}$$

Substituting $\frac{df}{du}$ into the RHS of (1.17):

$$\Rightarrow 2\frac{d^2u}{d\phi^2} = \frac{\epsilon}{h^2} - 2u + 3u^2 \tag{1.20}$$

Resulting in the equation for orbits in (u, ϕ) space:

$$\boxed{\frac{d^2u}{d\phi^2} + u = \frac{\epsilon}{2h^2} + \frac{3}{2}u^2} \tag{1.21}$$

which simplifies for a photon to not involve h at all when $\epsilon = 0$.

2 Particle Orbits

A model was simulated for a massive particle orbiting a black hole governed by (1.21). The particle starts from an initial radius of $r_0 = 6$ and zero initial radial velocity ($dr/d\phi = 0$). The simulation explores a range of angular momentum values, h, from $h = 0$ up to the value required to sustain a circular orbit at $r = 6$, h_{cir}.

2.1 Determining h_{cir}

We can gather the value of h_{cir} needed to plot a circular orbit, by setting the condition $\frac{dV_{\text{eff}}}{dr} = 0$ at $r = r_0$, forcing a minimum in the effective potential at this radius. Differentiating equation(1.6) gives:

$$V_{\text{eff}} = (1 - \frac{1}{r})(\epsilon + \frac{h^2}{r^2}) \tag{2.1}$$

Expanding, in order to take the derivative:

$$dV_{\text{eff}} = 1 + \frac{h^2}{r^2} - \frac{1}{r} - \frac{h^2}{r^3} \tag{2.2}$$

Leads to:

$$\frac{dV_{\text{eff}}}{dr} = \frac{-2h^2}{r^3} + \frac{1}{r^2} + \frac{3h^2}{r^4} \tag{2.3}$$

And evaluating at $r = 6$:

$$\left.\frac{dV_{\text{eff}}}{dr}\right|_{r=6} = 1 + \frac{h_{\text{cir}}^2}{6^2} - \frac{1}{6} - \frac{h_{\text{cir}}^2}{6^3} \tag{2.4}$$

Setting $\left.\frac{dV_{\text{eff}}}{dr}\right|_{r=6} = 0$ and rearranging leads to the solution:

$$h_{\text{cir}} = 2 \tag{2.5}$$

For all values of h lower than h_{cir}, the particle will be captured. For all values greater than h_{cir}, the particle will escape.

We now need to work out the corresponding value of k for this to be a circular orbit. The most direct method to do this coming from substituting $r = 6$ and $\dot{r} = 0$ into (1.5), giving:

$$0 = k^2 - (1 - \frac{1}{6})(\epsilon + \frac{h_{\text{cir}}^2}{6^2}). \tag{2.6}$$

Which when substituting for $\epsilon = 1$, $h_{\text{cir}} = 2$ gives

$$k^2 = \frac{25}{27} \Rightarrow k = \sqrt{\frac{25}{27}} = \frac{5}{3\sqrt{3}} \tag{2.7}$$

In the simulation, the value of h will be varied from 0 to h_{cir}, to show trajectories that represent the particle falling radially inwardly, up to the particle completing a circular orbit (note all other values of h will not satisfy $\left.\frac{dV_{\text{eff}}}{dr}\right|_{r=6} = 0$)

2.2 Graphical Output

A Runge–Kutta 4 technique was used for integration, with event detection to stop the integration once the particle crosses the event horizon at $r = 1$. The orbits in a plane are plotted in physical space in Figure 2. The orbiting distance from the centre of the black hole is also shown in Figure 3.

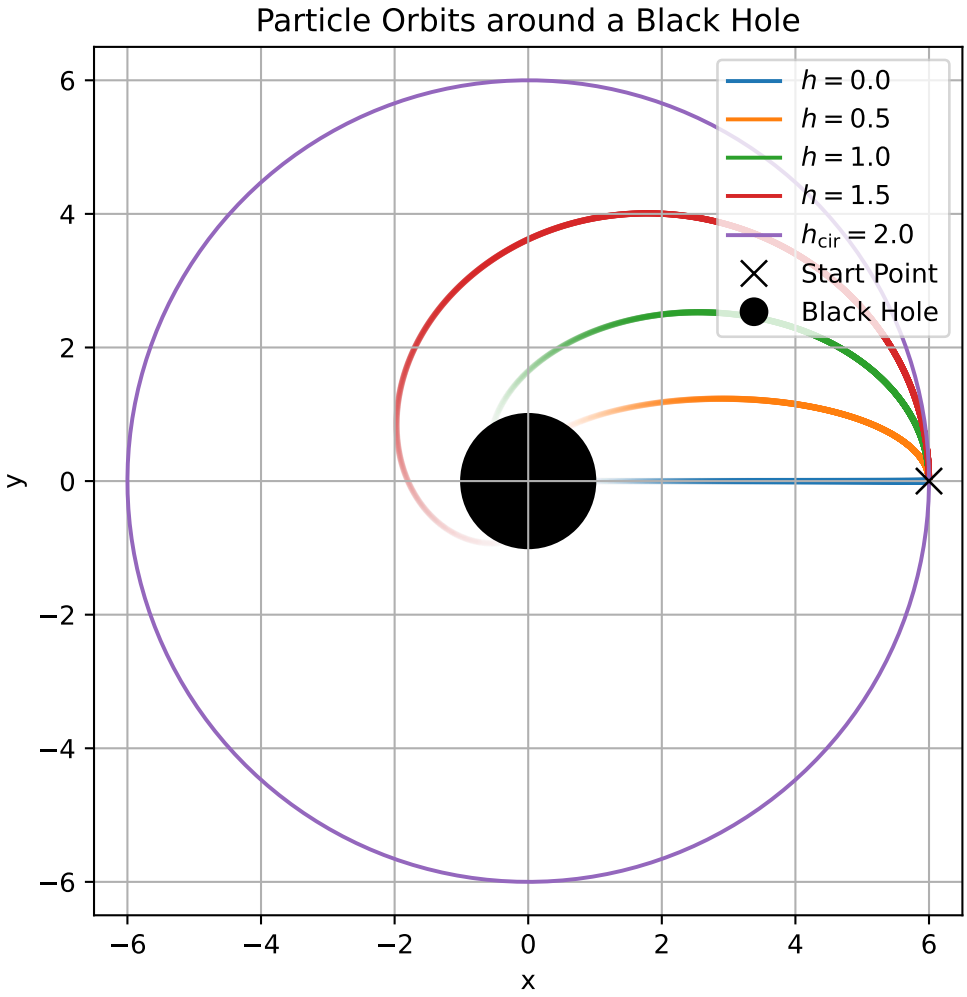

Figure 2: Orbits around a black hole, for multiple angular momentum values. as proper time passes, the colour of the orbit fades.

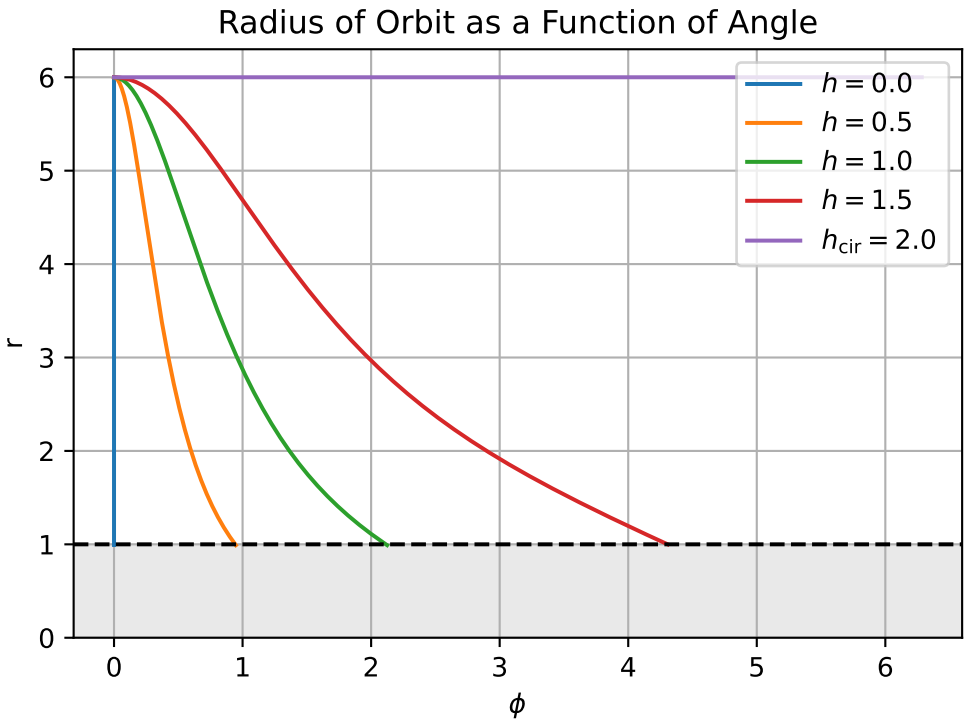

Figure 3: The distance of the particle from the centre of the black hole as a function of the angle from the initial starting position. The dashed horizontal line represents the event horizon of the black hole.

The orbits have been chosen, through selecting values of h, to show capture by the black hole and a circular orbit around the black hole.

2.3 Proper Time

The proper times taken to reach the Schwarzchild radius were calculated during the simulation and are shown in Table 1.

Table 1: Proper time for a particle to fall from $r = 6$ to $r = 1$ for various angular momenta h.

h	τ_{fall}
0.0	22.40
0.5	23.20
1.0	26.20
1.5	35.75
2.0	–

Table 1, shows a significant jump between the fall-time values between $h = 1.0$ and $h = 1.5$. whereas the earlier values were much close, this shows the exponential impact of the specific angular momentum.

3 Effect of Different Initial Radius on Particle Orbits

The analysis from Section 2 was repeated with a smaller initial radius of $r_0 = 2.5$, in order to investigate any diffrences. The same values for h were used.

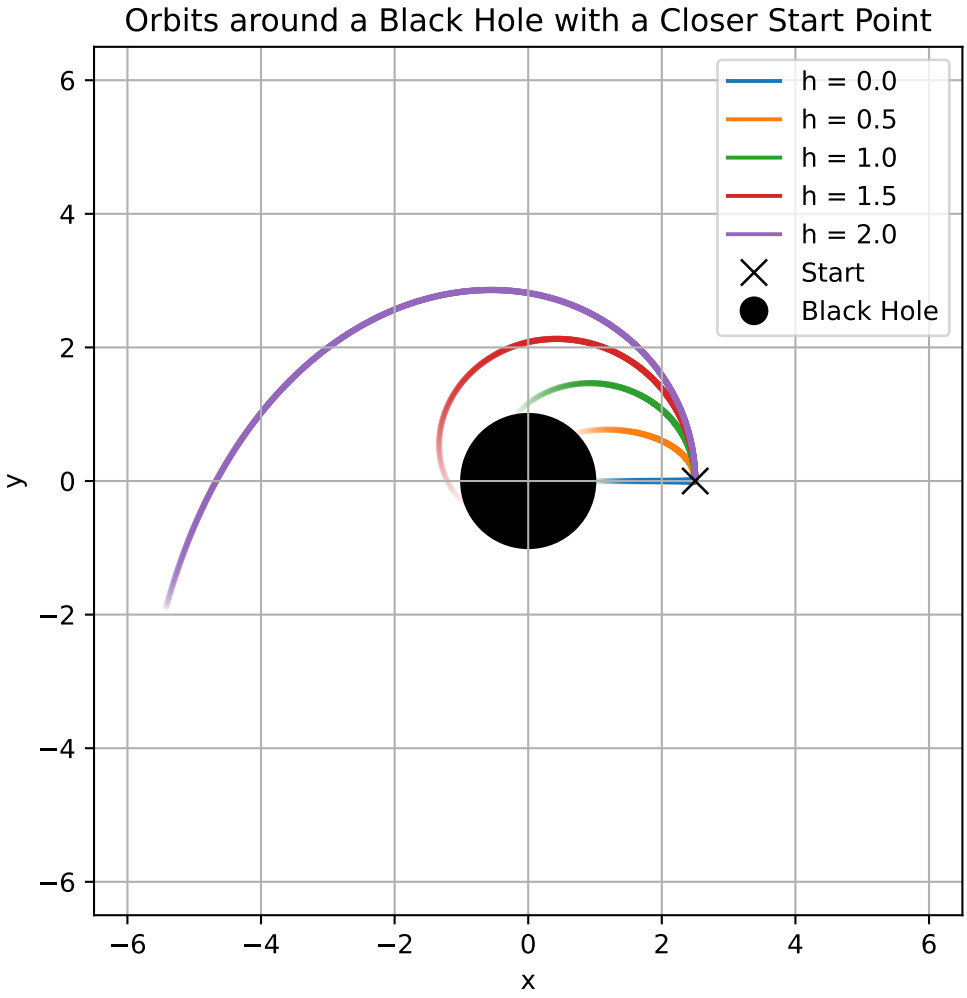

Figure 4: Updated trajectories for orbits around a black hole using the same values for specific angular momentum as in Figure 2, but with a smaller initial radius, $r_0 = 2.5$.

Figure 4 shows that all of the orbits end in capture, as before. Apart from the greatest value of h, previously resulting in a circular orbit, it now escapes. This is because it has a higher effective potential as the new radius is less than the Inner Most Stable Orbit, known to be at:

$$r_{\text{ISCO}} = \frac{6GM}{c^2} \tag{3.1}$$

Which in our units corresponds to:

$$r_{\text{ISCO}} = 3 \tag{3.2}$$

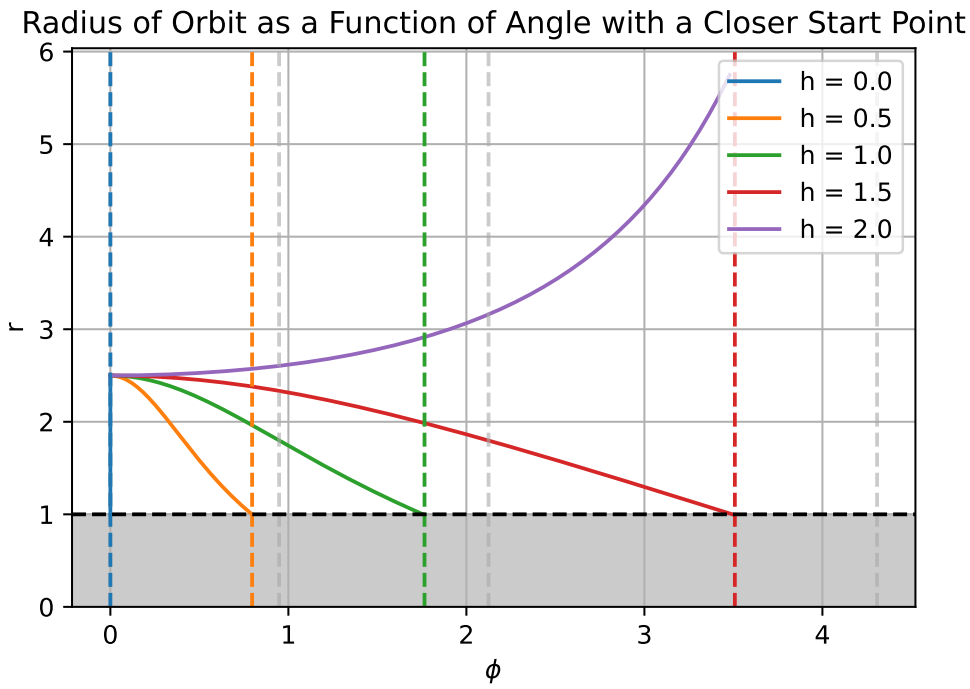

Figure 5: Radial distances from the centre of the black hole as a function of ϕ. The coloured vertical dashed lines show when each of the simulations reaches the Schwarzschild radius and the grey vertical line to the immediate right of these show the previous angle for the given value of h, for the simulation in Figure 3 $r_0 = 6$.

Table 2: Capture entrance angles and proper fall–in times for particles starting at $r_0 = 6$ and $r_0 = 2.5$. N/A values for $r_0 = 6$ are because the orbit is circular, whereas N/A values for $r_0 = 2.5$ are because the particle escapes.

h	$\phi_{\text{cap}}(r_0 = 6)$	$\tau(r_0 = 6)$	$\phi_{\text{cap}}(r_0 = 2.5)$	$\tau(r_0 = 2.5)$
0.0	0.00	22.39	0.00	5.44
0.5	0.95	23.18	0.80	5.63
1.0	2.12	26.17	1.76	6.38
1.5	4.31	35.75	3.51	9.07
2.0	N/A	N/A	N/A	N/A

The numerical results, shown in Table 2, clearly demonstrate that starting closer to the black hole at $r_0 = 2.5r_s$ leads to faster capture, both in angular distance covered and proper time passed.

4 Comparison with Theoretical Results

In this section we compare the numerical proper fall-in times calculated in Section 3, with the analytic stability criterion for circular particle orbits near a Schwarzschild black hole.

4.1 Analytic Predictions

For a given initial radius r_0, the analytic circular-orbit angular momentum, h_{cir}, and stability condition are derived from the effective potential in equation (1.6):

$$V_{\text{eff}}(r) = \left(1 - \frac{1}{r}\right)\left(1 + \frac{h^2}{r^2}\right) \tag{4.1}$$

by solving:

$$\left.\frac{dV_{\text{eff}}}{dr}\right|_{r_0, h_{\text{cir}}} = 0 \quad \Longrightarrow \quad h_{\text{cir}}(r_0) = \sqrt{\frac{r_0^2}{2r_0 - 3}}\,. \tag{4.2}$$

The sign of the second derivative,

$$\left.\frac{d^2V_{\text{eff}}}{dr^2}\right|_{r_0, h_{\text{cir}}} \quad \Longrightarrow \quad \begin{cases} \text{stable circular orbit}, & > 0, \\ \text{unstable circular orbit}, & < 0, \end{cases} \tag{4.3}$$

classifies the equilibrium.

Table 3: Analytic circular-orbit parameters and stability classification for $r_0 = 6.0$ and 2.5.

r_0	h_{cir}	$\left.\frac{d^2V_{\text{eff}}}{dr^2}\right\|_{(r_0, h_{\text{cir}})}$	Stability
6.0	2.000	$+3.09 \times 10^{-3}$	Stable
2.5	1.768	-3.14×10^{-2}	Unstable

The circular orbit value of specific angular momentum values are shown in Table 3. However, for an initial radius $r_0 = 2.5$, the particle cannot have a stable orbit at the value predicted, $h_{\text{cir}} = 1.768$, because it is less than $r_{\text{IMSO}} = 3$. This is shown as the orbit is classed as unstable in Table 3, as oppose to the $r_0 = 6$ case, where it is classed as Stable, and has been visually confirmed to be so in Figure 2.

4.2 Falltimes

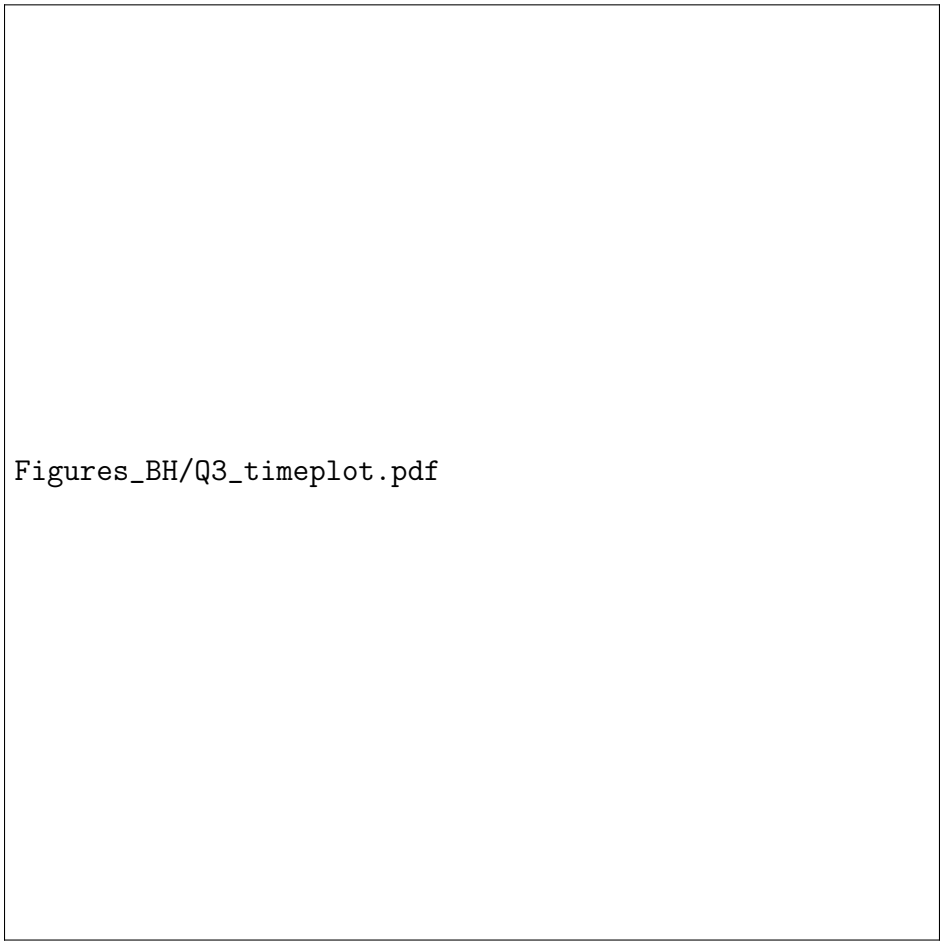

Figure 6: Proper fall-in time τ as a function of angular momentum h for $r_0 = 6.0$ (red) and $r_0 = 2.5$ (blue). The vertical dashed line marks h_{cir}.

Figure 6 plots the numerical proper fall-in time, τ ,versus angular momentum, h, for $r_0 = 6.0$ and $r_0 = 2.5$. As h approaches the analytic threshold h_{cir}, the curve for $r_0 = 6.0$ should diverge, indicating a stable circular orbit. Whereas for $r_0 = 2.5$ the value of $h_{\text{cir}} = 1.768$ should be a turning point, where past that point the particle is no longer captured and we should have no values of τ available to plot. This is not what we see, suggesting our simulation does not match perfectly with the theory. Both red and blue plots in figure 6 vary away from prediction near the $h = 2$ mark, again indicating that the simulation can break down near that mark.

4.3 Effective Potential Visualization

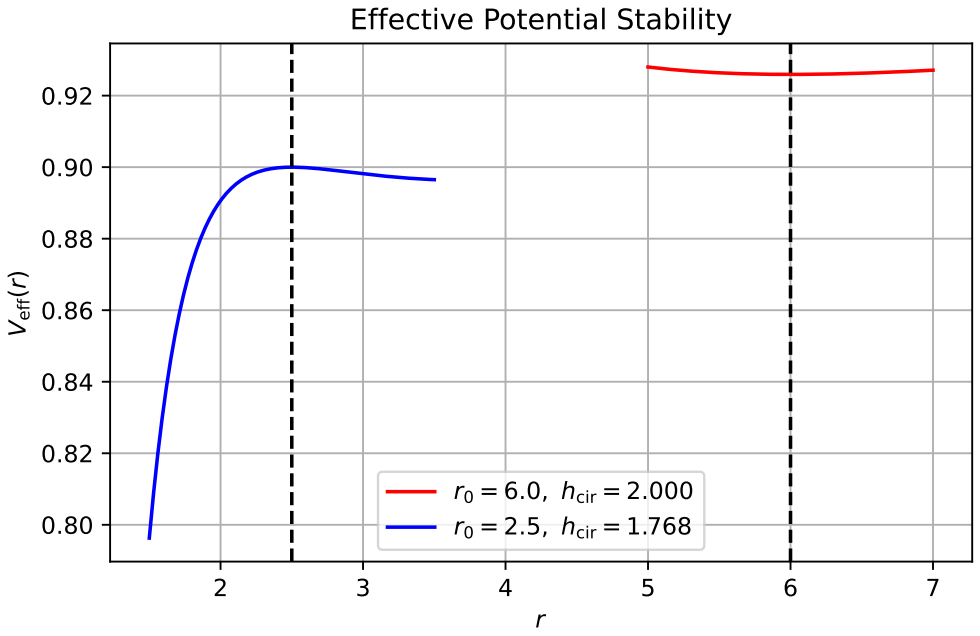

Figure 7: Effective potential $V_{\text{eff}}(r)$ around $r_0 = 6.0$ and 2.5 for their respective calculated theortical values of h_{cir}.

Figure 7 shows the effective potential $V_{\text{eff}}(r)$ near each r_0 for the analytic values of h_{cir}, with dashed lines marking the circular-orbit radii. The local minimum at $r_0 = 6.0$ confirms stability, while the local maximum at $r_0 = 2.5$ confirms instability.

5 Coordinate-Time Formulation

5.1 Changing time coordinates in equations

Thus far, all simulations and analyses have used τ, proper time, in place of the affine parameter, s, as we are simulating only massive particles for now. We now reformulate the problem using the coordinate time, t, as the evolution parameter. While this is not the standard procedure for calculating orbits, the results obtained using this, can be interpreted as viewing the system from the position of a stationary observer at infinity. Note, this is not what the particle itself experiences.

From the conserved quantities in (1.3), we have:

$$\dot{t} = \frac{k}{1 - \frac{1}{r}}, \qquad \dot{\phi} = \frac{h}{r^2}. \tag{5.1}$$

The geodesic condition for a massive particle ($\epsilon = 1$) yields the radial equation:

$$\left(\frac{dr}{ds}\right)^2 = k^2 - \left(1 - \frac{1}{r}\right)\left(\frac{h^2}{r^2} + 1\right). \tag{5.2}$$

To express this in terms of coordinate time, we apply the chain rule:

$$\frac{dr}{dt} = \frac{dr}{ds} \cdot \frac{ds}{dt} = \dot{r} \cdot \left(\frac{1 - \frac{1}{r}}{k} \right), \tag{5.3}$$

and squaring both sides gives:

$$\left(\frac{dr}{dt} \right)^2 = \left(\frac{1 - \frac{1}{r}}{k} \right)^2 \left[k^2 - \left(1 - \frac{1}{r} \right) \left(\frac{h^2}{r^2} + 1 \right) \right]. \tag{5.4}$$

This defines a first-order ODE for the radial coordinate $r(t)$. We now differentiate both sides of this equation with respect to t to obtain a second-order ODE. Using the chain rule:

$$\frac{d}{dt} \left(\left(\frac{dr}{dt} \right)^2 \right) = 2 \frac{dr}{dt} \cdot \frac{d^2 r}{dt^2} = \frac{d}{dt} \left[\left(\frac{1 - \frac{1}{r}}{k} \right)^2 \left(k^2 - \left(1 - \frac{1}{r} \right) \left(\frac{h^2}{r^2} + 1 \right) \right) \right]. \tag{5.5}$$

Since the right-hand side is a function of r, we apply the chain rule again:

$$2 \frac{dr}{dt} \cdot \frac{d^2 r}{dt^2} = \frac{d}{dr} \left[\left(\frac{1 - \frac{1}{r}}{k} \right)^2 \left(k^2 - \left(1 - \frac{1}{r} \right) \left(\frac{h^2}{r^2} + 1 \right) \right) \right] \cdot \frac{dr}{dt}. \tag{5.6}$$

Assuming $\frac{dr}{dt} \neq 0$ (noting this may break down near initial conditions), we divide both sides by $2\frac{dr}{dt}$ to isolate the second derivative:

$$\frac{d^2 r}{dt^2} = \frac{1}{2} \cdot \frac{d}{dr} \left[\left(\frac{1 - \frac{1}{r}}{k} \right)^2 \left(k^2 - \left(1 - \frac{1}{r} \right) \left(\frac{h^2}{r^2} + 1 \right) \right) \right]. \tag{5.7}$$

We now compute this derivative explicitly. First, we rewrite:

$$\left(\frac{1 - \frac{1}{r}}{k} \right)^2 = \frac{(r-1)^2}{k^2 r^2}, \qquad \left(1 - \frac{1}{r} \right) \left(\frac{h^2}{r^2} + 1 \right) = \left(1 - \frac{1}{r} \right) \cdot \frac{h^2 + r^2}{r^2}. \tag{5.8}$$

Define the full expression inside the derivative as:

$$f(r) = \frac{(r-1)^2}{k^2 r^2} \left[k^2 - \left(1 - \frac{1}{r} \right) \left(\frac{h^2}{r^2} + 1 \right) \right]. \tag{5.9}$$

Differentiating this product explicitly:

$$\begin{aligned} \frac{df}{dr} = \frac{d}{dr} \left(\frac{(r-1)^2}{k^2 r^2} \right) \left[k^2 - \left(1 - \frac{1}{r} \right) \left(\frac{h^2}{r^2} + 1 \right) \right] \\ + \frac{(r-1)^2}{k^2 r^2} \cdot \frac{d}{dr} \left[- \left(1 - \frac{1}{r} \right) \left(\frac{h^2}{r^2} + 1 \right) \right]. \end{aligned} \tag{5.10}$$

Substituting into (5.7), we obtain the final second-order coordinate-time radial equation:

$$\frac{d^2r}{dt^2} = \frac{1}{2}\left\{ \frac{d}{dr}\left(\frac{(r-1)^2}{k^2r^2}\right)\left[k^2 - \left(1-\frac{1}{r}\right)\left(\frac{h^2}{r^2}+1\right)\right] - \frac{(r-1)^2}{k^2r^2}\cdot\frac{d}{dr}\left[\left(1-\frac{1}{r}\right)\left(\frac{h^2}{r^2}+1\right)\right]\right\} \tag{5.11}$$

This equation defines the second-order ODE governing $r(t)$ in coordinate time. Despite its complexity, it is now suitable for numerical integration. We will run the simulation again, to look at spacial plots, in order to visualise a difference, and this time look at radial plots as a function of parameter to show how the orbits are effected.

5.2 Effect of Using Co-ordinate Time

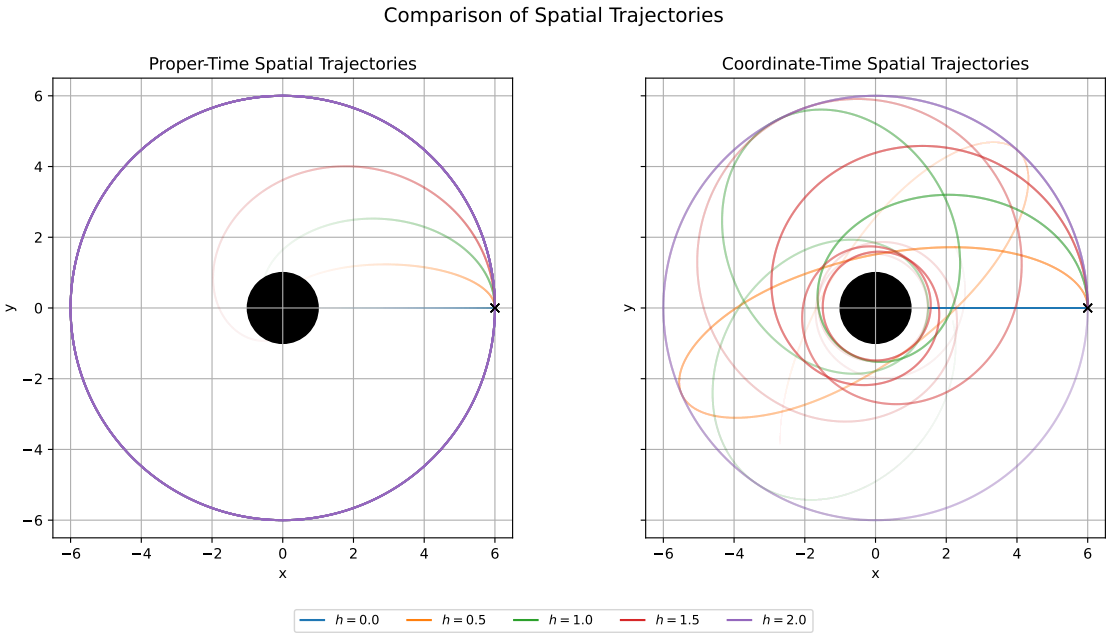

Figure 8: The orbits around a black hole when formulating the equations using t as well as τ.

As shown in the co-ordinate time plot in Figure 8, when formulating the simulation using t, particle trajectories never reach the event horizon. This is because the particle slows down exponentially as it gets closer to the black hole. Mathematically, $\frac{dr}{dt} \to 0$ as $r \to 1$. Instead, the particle will follow chaotic orbits. The difference between the two formulations becomes stark only near the black hole, where coordinate time "stretches" due to the Schwarzschild time dilation.

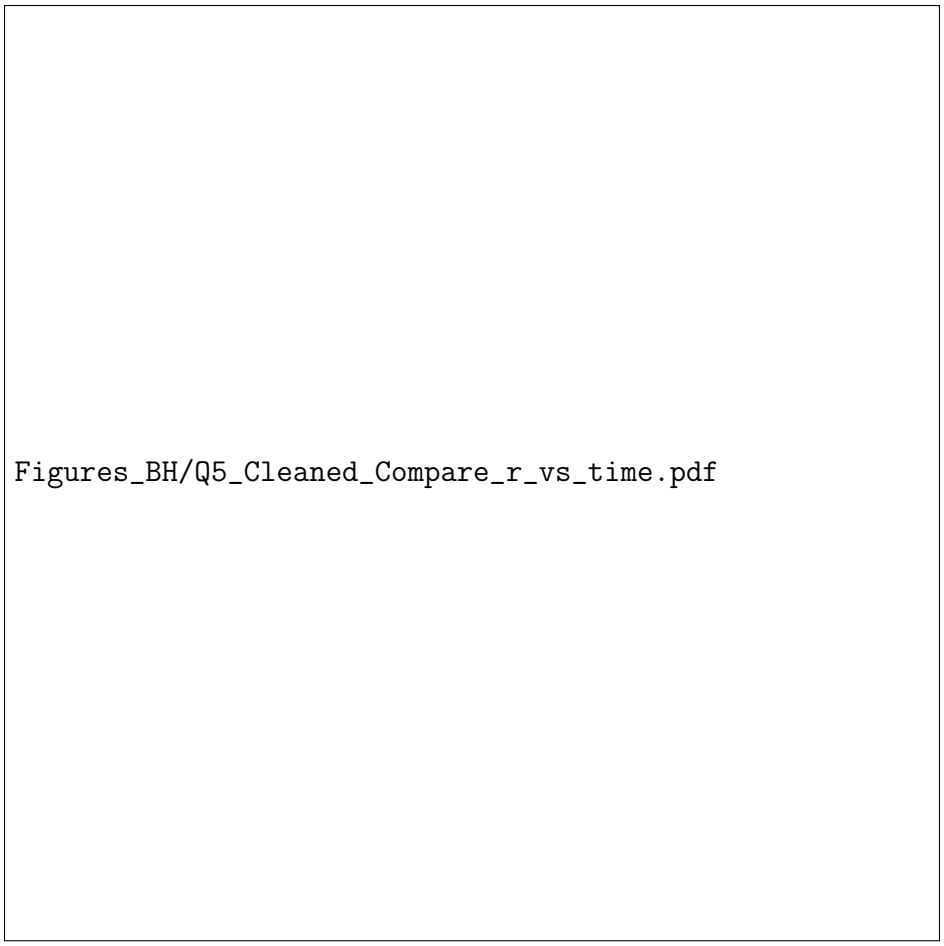

Figure 9: Radius of simulated trajectories, plotted as both a function of proper time, τ, and co-ordinate time, t, for comparison.

Figure 9 explicitly illustrates the way the massive particle does not fall into the black hole, and instead the trajectory means that the particle returns to it's starting radius, all be it at a different position. It appears here, in the spatial plot (Figure 8), more like a Newtonian system, with energy being conserved and particles coming back out to the same distance away as they started at.

6 Particle Scattering

6.1 Analytic equations for the Cross Section of Capture

A particle approaches from spatial infinity with asymptotic speed, v, and an impact parameter, b. The impact parameter, b, is defined as the perpendicular distance between the centre of the black hole and the particle's incoming trajectory assuming no gravitational deflection.

Due to the curvature of spacetime, particles with sufficiently small b are gravitationally captured by the black hole. For each v, there exists a *critical impact parameter* $b_{\text{crit}}(v)$ such that particles with $b < b_{\text{crit}}(v)$ are captured, and those with $b > b_{\text{crit}}(v)$ escape. The

corresponding *capture cross-section* is defined by:

$$\sigma(v) = \pi b_{\text{crit}}(v)^2. \tag{6.1}$$

To derive b_{crit}, we begin with the asymptotic relation between conserved angular momentum, h, energy, k, and the impact parameter, b. In asymptotically flat spacetime, a particle with speed v at infinity satisfies:

$$h = v\, b_{\text{crit}}\, k \tag{6.2}$$

Solving for b_{crit} gives:

$$b_{\text{crit}} = \frac{h}{vk} \tag{6.3}$$

For a massive particle, the conserved energy per unit mass is:

$$k = \frac{1}{\sqrt{1-v^2}} \tag{6.4}$$

Substituting this into the expression for b_{crit}:

$$b_{\text{crit}} = \frac{h\sqrt{1-v^2}}{v} \tag{6.5}$$

However, the critical value of the angular momentum, h_{crit}, is not constant — it depends on the orbital dynamics near the black hole. Specifically, the threshold for capture corresponds to the particle asymptotically approaching an unstable circular orbit at the photon sphere radius $r = \frac{3}{2}$. Imposing the condition that the orbit is at the maximum of the effective potential, we finds:

$$h_{\text{crit}}(v)^2 = \frac{27}{4}\left(\frac{1}{1-v^2} - \frac{1}{3}\right) = \frac{27}{4}\left(\frac{1+2v^2}{v^2}\right) \tag{6.6}$$

Substituting into the expression for b_{crit}, we obtain:

$$b_{\text{crit}}(v)^2 = \frac{h_{\text{crit}}^2(1-v^2)}{v^2} = \frac{27}{4}\cdot\frac{(1+2v^2)(1-v^2)}{v^4} \tag{6.7}$$

This expression simplifies cleanly if we first compute $b_{\text{crit}}(v)$ directly from:

$$b_{\text{crit}}(v) = \frac{3}{2v}\sqrt{1+2v^2} \tag{6.8}$$

Thus, the capture cross-section becomes:

$$\sigma(v) = \pi b_{\text{crit}}(v)^2 = \pi\left(\frac{3}{2v}\sqrt{1+2v^2}\right)^2 = \frac{9\pi}{4}\left(\frac{1+2v^2}{v^2}\right) \tag{6.9}$$

This expression correctly interpolates between the Newtonian limit and the ultra-relativistic regime:

- As $v \to 0$, $\sigma(v) \sim \frac{1}{v^2}$, diverging due to long-range gravitational deflection.

- As $v \to 1$, $\sigma(v) \to \frac{27\pi}{4}$, the known capture cross-section for photons — significantly larger than the black hole's horizon area due to strong spacetime curvature.

$$\boxed{\sigma(v) = \frac{9\pi}{4}\left(\frac{1+2v^2}{v^2}\right)} \tag{6.10}$$

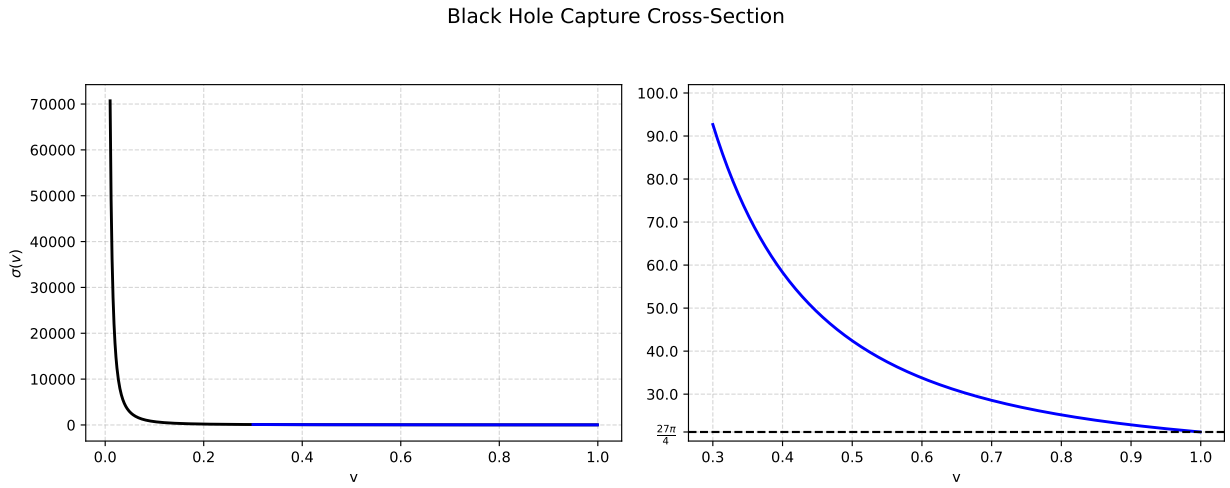

Figure 10: Cross section for capture as a function of initial velocity. The section in blue is shown again on the right. right side plot excludes the low velocity regime

As seen in Figure 10, in the limit $v \ll 1$, the cross section diverges to infinity. This behaviour is physically intuitive: a particle moving with velocity close to zero relative to the black hole can be captured from a great distance. Although it may take a significant amount of time to reach the event horizon, it can spiral inward and orbit the black hole multiple times before crossing it. As the particle becomes more relativistic, with greater velocity, the cross section decreases and tends towards a value of $\frac{27\pi}{a}$, calculated by finding the $\lim_{v\to\infty}$ of Equation (6.10).

7 Photon Scattering

In this section we consider photons approaching a Schwarzschild black hole as in Section 6. The motion is governed by the same geodesic equations as throughout, but with the parameter $\epsilon = 0$, reflecting the fact that photons follow null geodesics. The geodesic equation then becomes:

$$\frac{d^2u}{d\phi^2} + u = \frac{3}{2}u^2. \tag{7.1}$$

This non-linear, second-order differential equation governs the bending of light in the gravitational field of a Schwarzschild black hole.

The deflection angle, $\Delta\phi(b)$, is defined as the total deviation of the photon's path from a straight line as it passes near the black hole. In flat space, a photon approaching from infinity with impact parameter b would travel in a straight line and not be deflected. In curved spacetime, however, the gravitational field bends the photon's trajectory such that the asymptotic outgoing direction differs from the incoming one by a non-zero angle.

More precisely, if we denote the incoming and outgoing asymptotic angles as ϕ_{in} and ϕ_{out}, then the total deflection is:

$$\Delta\phi = \phi_{\text{out}} - \phi_{\text{in}} - \pi. \tag{7.2}$$

The subtraction of π accounts for the straight-line path that would occur in flat space. For trajectories that are only weakly perturbed (i.e. for $b \gg b_{\text{crit}}$), the photon path remains close to a straight line, and the deflection angle is small.

7.1 Small Deflection angles

When the impact parameter b is large — specifically when $b \gg b_{\text{crit}} = \frac{3\sqrt{3}}{2} \approx 2.598$ — the photon passes far from the black hole and its trajectory is only slightly deflected from a straight line. In this regime, the term $\frac{3}{2}u^2$ on the right-hand side of the geodesic equation (7.1), is small compared to the linear term u. To see this, note that for a nearly straight-line trajectory, the solution to the unperturbed equation $u'' + u = 0$ is:

$$u(\phi) = \frac{\sin\phi}{b}, \tag{7.3}$$

in this approximation, we have $u \sim \frac{1}{b}$, and thus the relativistic correction term scales as

$$\frac{3}{2}u^2 \sim \frac{3}{2b^2},$$

while the linear term remains $u \sim \frac{1}{b}$. Therefore, the correction is suppressed by a factor $\sim 1/b$, which is small when b is large.

To compute $\Delta\phi$ in this regime, evaluate the integral:

$$\Delta\phi(b) = 2\int_{r_{\min}}^{\infty} \frac{dr}{r^2\sqrt{1 - \frac{b^2}{r^2}\left(1 - \frac{1}{r}\right)}} - \pi, \tag{7.4}$$

where $r_{\min}$ is the point of closest approach, determined implicitly by the turning point condition of the trajectory.

In the large-b limit, this integral can be expanded in powers of $1/b$. The leading-order term yields the classical result for light bending by a Schwarzschild black hole:

$$\Delta\phi(b) \approx \frac{2}{b}. \tag{7.5}$$

This approximation is valid for $b \gg b_{\text{crit}}$, where the curvature effects are small and the photon remains far from the strong-field region near the photon sphere.

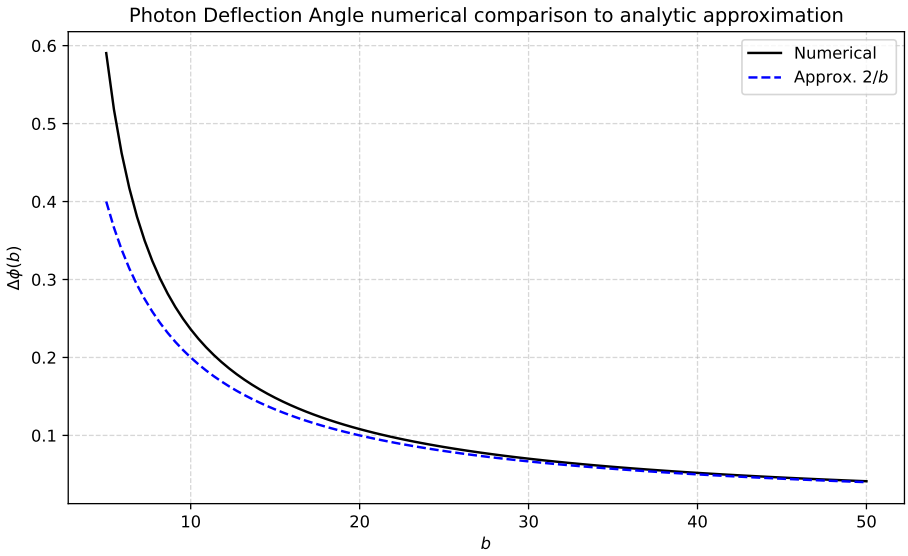

Figure 11: The numerical deflection angle from the simulation is shown in black. The blue dashed line represents the approximate analytic solution. Both curves are plotted only for $b \geq 5$.

We have simulated using the full equations in order to plot the computed values against the large b limit analytical solution to Equation (7.5) for values of b greater than 5 as shown in Figure 11. For roughly $b = 20$ onwards, the large b expansion is visually very accurate. it breaks down for smaller vales of b in Figure 11.

7.2 Specific Values of the Impact Parameter

We computed the deflection angle $\Delta\phi$ for several values of the impact parameter b, focusing on values near the *critical impact parameter*

$$b_{\text{crit}} = \frac{3\sqrt{3}}{2} \approx 2.598.$$

These results explore the *strong-field regime*, where the photon trajectory is significantly bent by the black hole:

Table 4: Photon deflection angles for selected impact parameters.

b	$\Delta\phi$ (rad)
3.20	1.409352
2.80	2.299150
2.65	3.557938
2.50	captured

As $b \to b_{\text{crit}}^{+}$, the deflection angle grows rapidly and formally diverges at the critical value, where the photon spirals into an unstable circular orbit. For $b = 2.5 < b_{\text{crit}}$, the photon is

captured by the black hole. These values are well outside the range where the approximation $\Delta\phi \approx 2/b$ is valid, and demonstrate the transition from weak to strong gravitational lensing.

Appendix A: Code Listing

Q1.py

```
import numpy as np
import matplotlib.pyplot as plt
import os

# Directory for saving figures
output_dir = r"C:\\CATAM_BlackHoleOrbits\\Figures_BH"
os.makedirs(output_dir, exist_ok=True)

epsilon = 1
h_values = [1, 2, 3, 4]

r = np.linspace(0.5, 10, 1000)

def V_eff(r, h):
    return (1 - 1/r) * (epsilon + h**2 / r**2)

plt.figure(figsize=(8, 5))
for h in h_values:
    plt.plot(r, V_eff(r, h), label=f'h = {h}')

plt.axvline(1, linestyle='--', color='gray')  # vertical dashed line at r=1
current_ticks = plt.xticks()[0].tolist()
if 1 not in current_ticks:
    current_ticks.append(1)

# Sort ticks
current_ticks = sorted(current_ticks)

# Create labels, marking 1 as 'r=1'
labels = [ 'r=1' if tick==1 else str(int(tick) if tick.is_integer() else f'{tick:.1f}') for tick in cur
plt.xticks(current_ticks, labels)

plt.xlabel('r')
plt.ylabel(r'$V_{\mathrm{eff}}(r)$')
plt.title('Effective Potential for Various Angular Momenta')
plt.legend()
plt.grid()

output_path = os.path.join(output_dir, 'Q1_V_eff.pdf')
plt.savefig(output_path, dpi=300, bbox_inches='tight')
plt.show()
```

Q2.py

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
import matplotlib.colors as mcolors
import os

# Directory for saving figures
output_dir = r"C:\CATAM_BlackHoleOrbits\Figures_BH"
os.makedirs(output_dir, exist_ok=True)

# Parameters
Hs = [0.0, 0.5, 1.0, 1.5, 2.0]  # angular momentum values
epsilon = 1
r0 = 6
k = np.sqrt(25 / 27)  # same energy constant for all

ds = 0.01  # step size
max_steps = int(200 * np.pi / ds)  # maximum steps for plunging orbits

cycle_colors = plt.rcParams['axes.prop_cycle'].by_key()['color']

# Schwarzschild metric function
def F(r):
    return 1 - 1/r

# Effective potential derivative for a given h
def dVeff_dr(r, h):
    return -2*h**2 / r**3 + 1 / r**2 + 3*h**2 / r**4

# ODE system: y = [r, r_dot, phi, s]
def derivatives(y, h):
    r, r_dot, phi, s = y
    dr_ds = r_dot
    d2r_ds = -0.5 * dVeff_dr(r, h)
    dphi_ds = h / r**2
    ds_ds = 1.0
    return np.array([dr_ds, d2r_ds, dphi_ds, ds_ds])

# Integrator
def integrate_orbit(h):
    y = np.array([r0, 0.0, 0.0, 0.0])
    traj = []
    if h == 2.0:
        period = 2 * np.pi / (h / r0**2)
        steps = int(np.ceil(period / ds)) + 1
    else:
        steps = max_steps
    for _ in range(steps):
        r = y[0]
        traj.append(y.copy())
        if h != 2.0 and r <= 1.0:
            break
```

```
        k1 = derivatives(y, h)
        k2 = derivatives(y + 0.5 * ds * k1, h)
        k3 = derivatives(y + 0.5 * ds * k2, h)
        k4 = derivatives(y + ds * k3, h)
        y = y + (ds / 6) * (k1 + 2*k2 + 2*k3 + k4)
    return np.array(traj)

# Storage for capture analysis
captured = []
capture_times = {}

# Cartesian plot with fading trajectories
fig1, ax1 = plt.subplots(figsize=(6,6))
handles = []

for idx, h in enumerate(Hs):
    traj = integrate_orbit(h)
    if traj[-1,0] <= 1.0 and h != 2.0:
        captured.append(h)
        capture_times[h] = traj[-1,3]
    r_vals = traj[:,0]
    phi_vals = traj[:,2]
    x_vals = r_vals * np.cos(phi_vals)
    y_vals = r_vals * np.sin(phi_vals)
    base_color = cycle_colors[idx % len(cycle_colors)]
    if h == 2.0:
        label = r"$h_{\mathrm{cir}} = %.1f$" % h
        line, = ax1.plot(x_vals, y_vals, color=base_color, label=label)
        handles.append(line)
    else:
        rgba_base = mcolors.to_rgba(base_color)
        alphas = np.linspace(1.0, 0.01, len(x_vals))
        colors_rgba = [(rgba_base[0], rgba_base[1], rgba_base[2], a) for a in alphas]
        ax1.scatter(x_vals, y_vals, color=colors_rgba, s=2)
        label = r"$h = %.1f$" % h
        handles.append(Line2D([0],[0], color=base_color, label=label))

# Start point marker
start_handle, = ax1.plot(r0, 0, marker='x', color='black', markersize=10, linestyle='None', label=r"Sta
handles.append(start_handle)

# Black hole marker
black_hole_patch = plt.Circle((0,0),1,color='black')
ax1.add_patch(black_hole_patch)
black_hole_handle = Line2D([0],[0],marker='o',color='black',linestyle='None',markersize=10,label=r"Blac
handles.append(black_hole_handle)

ax1.legend(handles=handles, loc='upper right')
ax1.set_aspect('equal')
ax1.set_xlim(-6.5,6.5)
ax1.set_ylim(-6.5,6.5)
ax1.set_xlabel('x')
ax1.set_ylabel('y')
ax1.set_title('Particle Orbits around a Black Hole')
```

```
ax1.grid(True)
plt.savefig(os.path.join(output_dir, 'Q2_Orbits_xy.pdf'), dpi=300, bbox_inches='tight')

# r-phi plot
fig2, ax2 = plt.subplots(figsize=(6,4))
# Shade region
ax2.axhspan(0, 1, color='lightgrey', alpha=0.5)
for h in Hs:
    traj = integrate_orbit(h)
    r_vals, phi_vals = traj[:,0], traj[:,2]
    if h == 2.0:
        label = r"$h_{\mathrm{cir}} = %.1f$" % h
    else:
        label = r"$h = %.1f$" % h
    ax2.plot(phi_vals, r_vals, label=label)
ax2.axhline(1, color='black', linestyle='--')
ax2.set_ylim(bottom=0)
ax2.set_xlabel(r'$\phi$')
ax2.set_ylabel('r')
ax2.set_title('Radius of Orbit as a Function of Angle')
ax2.legend(loc='upper right')
ax2.grid(True)
plt.savefig(os.path.join(output_dir, 'Q2_Orbits_rphi.pdf'), dpi=300, bbox_inches='tight')

# Print capture results
print("Captured angular momenta:", captured)
for h in captured:
    print(f"h = {h} captured in proper time s = {capture_times[h]:.2f}")

plt.show()
```

Q3.py

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
import matplotlib.colors as mcolors
import os

# Directory for saving figures
output_dir = r"C:\CATAM_BlackHoleOrbits\Figures_BH"
os.makedirs(output_dir, exist_ok=True)

# Parameters
Hs = [0.0, 0.5, 1.0, 1.5, 2.0]
epsilon = 1
k = np.sqrt(25/27)
ds = 0.01
max_steps = int(200*np.pi/ds)
cycle_colors = plt.rcParams['axes.prop_cycle'].by_key()['color']

# Radii: old (r0=6 for capture points), new (r0=2.5)
r0_old = 6.0
```

```
r0 = 2.5

def F(r):
    return 1 - 1/r

def dVeff_dr(r,h):
    return -2*h**2/r**3 + 1/r**2 + 3*h**2/r**4

def derivatives(y,h):
    r, r_dot, phi, s = y
    return np.array([r_dot, -0.5*dVeff_dr(r,h), h/r**2, 1.0])

# Integrator
def integrate_orbit(r0_val, h):
    y = np.array([r0_val, 0.0, 0.0, 0.0])
    traj = []
    steps = int(np.ceil((2*np.pi)/(h/r0_val**2)/ds)) if h==2.0 else max_steps
    for _ in range(steps):
        traj.append(y.copy())
        r = y[0]
        if h!=2.0 and r<=1.0:
            break
        k1 = derivatives(y,h)
        k2 = derivatives(y+0.5*ds*k1, h)
        k3 = derivatives(y+0.5*ds*k2, h)
        k4 = derivatives(y+ds*k3, h)
        y += (ds/6)*(k1 + 2*k2 + 2*k3 + k4)
    return np.array(traj)

# Compute capture angles at r0_old
capture_phis_old = {}
captured_old = []
for h in Hs:
    traj_old = integrate_orbit(r0_old, h)
    if traj_old[-1,0] <= 1.0 and h!=2.0:
        captured_old.append(h)
        capture_phis_old[h] = traj_old[-1,2]

# Simulate for r0=2.5
captured = []
capture_times = {}

fig1, ax1 = plt.subplots(figsize=(6,6))
handles = []
for idx,h in enumerate(Hs):
    traj = integrate_orbit(r0, h)
    if traj[-1,0] <= 1.0:
        captured.append(h)
        capture_times[h] = traj[-1,3]
    r_vals,phi_vals = traj[:,0],traj[:,2]
    x = r_vals*np.cos(phi_vals)
    y = r_vals*np.sin(phi_vals)
    rgba = mcolors.to_rgba(cycle_colors[idx])
    alphas = np.linspace(1,0.01,len(x))
```

```
    colors = [(rgba[0],rgba[1],rgba[2],a) for a in alphas]
    ax1.scatter(x,y,color=colors,s=2)
    handles.append(Line2D([0],[0],color=cycle_colors[idx],label=f"h = {h}"))
# Start & BH markers
h0,=ax1.plot(r0,0,'x',color='black',markersize=10,linestyle='None',label='Start')
handles.append(h0)
bh=plt.Circle((0,0),1,color='black'); ax1.add_patch(bh)
h1=Line2D([0],[0],marker='o',color='black',linestyle='None',markersize=10,label='Black Hole')
handles.append(h1)
ax1.legend(handles=handles,loc='upper right')
ax1.set(aspect='equal',xlim=[-6.5,6.5],ylim=[-6.5,6.5],xlabel='x',ylabel='y',
        title=r'Orbits around a Black Hole with a Closer Start Point')
ax1.grid(True)
plt.savefig(os.path.join(output_dir,'Q3_Orbits_xy_r25.pdf'),dpi=300,bbox_inches='tight')

# r-phi plot with both old and new capture lines
fig2, ax2 = plt.subplots(figsize=(6,4))
ax2.axhspan(0,1,color='darkgrey',alpha=0.6)
for idx,h in enumerate(Hs):
    traj = integrate_orbit(r0,h)
    ax2.plot(traj[:,2],traj[:,0],color=cycle_colors[idx],label=f"h = {h}")
# old capture lines (dashed, light)
for h in captured_old:
    phi_old = capture_phis_old[h]
    ax2.axvline(phi_old,color='darkgrey',linestyle='--',alpha=0.6)
# new capture lines (dashed, color)
for h in captured:
    phi_new = capture_times[h] and integrate_orbit(r0,h)[-1,2]
    ax2.axvline(phi_new,color=cycle_colors[Hs.index(h)],linestyle='--')
ax2.axhline(1,color='black',linestyle='--')
ax2.set_ylim(bottom=0)
ax2.set(xlabel=r'$\phi$',ylabel='r',title='Radius of Orbit as a Function of Angle with a Closer Start P
ax2.legend(loc='upper right')
ax2.grid(True)
plt.savefig(os.path.join(output_dir,'Q3_Orbits_rphi_r25.pdf'),dpi=300,bbox_inches='tight')
plt.show()
```

Q4.py

```
import numpy as np
import mpmath as mp
import matplotlib.pyplot as plt
import os

# Directory for saving figures
output_dir = r'C:\CATAM_BlackHoleOrbits\Figures_BH'
os.makedirs(output_dir, exist_ok=True)

# Parameters
Hs = np.linspace(0.0, 2.0, 200)
r0_list = [6.0, 2.5]               # starting radii

# Constant energy
```

```
k_const = np.sqrt(25/27)

# Functions
def Veff(r, h):
    return (1 - 1/r) * (1 + h**2 / r**2)

def fall_time(h, r0):
    integrand = lambda rr: 1/mp.sqrt(k_const**2 - Veff(rr, h))
    try:
         = mp.quad(integrand, [1, r0])
        return float(mp.re())
    except ValueError:
        return np.nan

# Compute proper fall-in times
tau_data = {r0: [] for r0 in r0_list}
for r0 in r0_list:
    h_cir = np.sqrt(r0**2 / (2*r0 - 3))
    cutoff = h_cir if r0 == 6.0 else 2.0
    for h in Hs:
        tau_data[r0].append(np.nan if h >= cutoff else fall_time(h, r0))

# Figure 1: Proper fall-in time
fig, ax = plt.subplots(figsize=(6,4))
colors = ['red', 'blue']
for r0, color in zip(r0_list, colors):
    ax.plot(Hs, tau_data[r0], color=color, label=f'$r_0={r0}$')

# Vertical lines at both circular thresholds
h_cir_60 = np.sqrt(6.0**2 / (2*6.0 - 3))      # = 2.0
h_cir_25 = np.sqrt(2.5**2 / (2*2.5 - 3))      # 1.7678
ax.axvline(h_cir_60, linestyle='--', color='black')
ax.axvline(h_cir_25, linestyle='--', color='black')

# Customize x-ticks
ticks = [t for t in ax.get_xticks() if abs(t - 1.75) > 1e-2]
ticks.extend([h_cir_25, h_cir_60])
ticks = sorted(set(ticks))
ax.set_xticks(ticks)

for label in ax.get_xticklabels():
    try:
        val = float(label.get_text())
        if abs(val - h_cir_25) < 1e-3 or abs(val - h_cir_60) < 1e-3:
            label.set_fontweight('bold')
    except ValueError:
        pass

ax.set_xlabel('$h$')
ax.set_ylabel('$\\tau$')
ax.set_title('Proper Fall-In Time')
ax.legend(loc='best')
ax.grid(True)
plt.tight_layout()
```

```
plt.savefig(os.path.join(output_dir, 'Q4_Timelot_extended.pdf'), dpi=300, bbox_inches='tight')

# Figure 2: Effective potential stability
fig2, ax2 = plt.subplots(figsize=(6,4))
for r0, color in zip(r0_list, colors):
    h_cir = np.sqrt(r0**2 / (2*r0 - 3))
    r_vals = np.linspace(max(1.01, r0 - 1), r0 + 1, 400)
    ax2.plot(r_vals, Veff(r_vals, h_cir), color=color,
             label=fr'$r_0={r0},\ h_{{\rm cir}}={h_cir:.3f}$')
    ax2.axvline(r0, linestyle='--', color='black')

ax2.set_xlabel('$r$')
ax2.set_ylabel('$V_{\\mathrm{eff}}(r)$')
ax2.set_title('Effective Potential Stability')
ax2.legend(loc='best')
ax2.grid(True)
plt.tight_layout()
plt.savefig(os.path.join(output_dir, 'Q4_Veff_stability.pdf'), dpi=300, bbox_inches='tight')

plt.show()
```

Q5.py

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
from matplotlib.collections import LineCollection
import os

# Output directory
output_dir = r"C:\CATAM_BlackHoleOrbits\Figures_BH"
os.makedirs(output_dir, exist_ok=True)

# Parameters
Hs = [0.0, 0.5, 1.0, 1.5, 2.0]
r0 = 6
k = np.sqrt(25 / 27)
ds = 0.01
t_max = 200
dt = 0.01
t_eval = np.arange(0, t_max, dt)
max_steps = int(300 * np.pi / ds)

def dVeff_dr(r, h):
    return -2*h**2 / r**3 + 1 / r**2 + 3*h**2 / r**4

def derivatives_affine(y, h):
    r, r_dot, phi, tau = y
    dr_ds = r_dot
    d2r_ds = -0.5 * dVeff_dr(r, h)
    dphi_ds = h / r**2
    return np.array([dr_ds, d2r_ds, dphi_ds, 1.0])

```

```
def integrate_affine_orbit(h):
    y = np.array([r0, 0.0, 0.0, 0.0])
    traj = []
    for _ in range(max_steps):
        r = y[0]
        traj.append(y.copy())
        if h != 2.0 and r <= 1.01:
            break
        k1 = derivatives_affine(y, h)
        k2 = derivatives_affine(y + 0.5 * ds * k1, h)
        k3 = derivatives_affine(y + 0.5 * ds * k2, h)
        k4 = derivatives_affine(y + ds * k3, h)
        y = y + (ds / 6) * (k1 + 2*k2 + 2*k3 + k4)
    return np.array(traj)

def f_prime(r, k, h):
    if r <= 1.01:
        return 1e-8  # small fallback to avoid early stop
    A = (r - 1)**2 / (k**2 * r**2)
    B = k**2 - (1 - 1/r) * (h**2 / r**2 + 1)
    dA = (2*(r - 1)*r**2 - (r - 1)**2*2*r) / (k**2 * r**4)
    term1 = (1 - 1/r)
    term2 = (h**2 + r**2) / r**2
    dterm2 = -2 * (h**2 + r**2) / r**3 + 2 / r
    dB = -dterm2 * term1 - term2 / r**2
    return dA * B + A * dB

def coordinate_time_rhs(t, y, k, h):
    r, r_dot = y
    return [r_dot, 0.5 * f_prime(r, k, h)]

def compute_phi_t(sol, h):
    r_vals = sol.y[0]
    phi = np.zeros_like(r_vals)
    for i in range(1, len(r_vals)):
        r2 = r_vals[i]**2
        if r2 < 1e-6:
            break
        dphi = h / r2 * (sol.t[i] - sol.t[i-1])
        phi[i] = phi[i-1] + dphi
    return phi

def fading_line(ax, x, y, color, label=None):
    points = np.array([x, y]).T.reshape(-1, 1, 2)
    segments = np.concatenate([points[:-1], points[1:]], axis=1)
    alpha_vals = np.linspace(1.0, 0.05, len(segments))
    lc = LineCollection(segments, colors=[(*color[:3], a) for a in alpha_vals], linewidths=1.5, label=la
    ax.add_collection(lc)

# Run simulations
trajectories_affine = {}
trajectories_coord = {}
for h in Hs:
    trajectories_affine[h] = integrate_affine_orbit(h)
```

```
    sol = solve_ivp(coordinate_time_rhs, [0, t_max], [r0, 0.0],
                    args=(k, h), t_eval=t_eval, rtol=1e-8, atol=1e-10)
    trajectories_coord[h] = sol

# Radial Coordinate Plot
fig, (ax_tau, ax_t) = plt.subplots(2, 1, figsize=(7, 6), sharex=True)
color_list = plt.get_cmap('tab10').colors

for idx, h in enumerate(Hs):
    color = color_list[idx % len(color_list)]
    traj = trajectories_affine[h]
    sol = trajectories_coord[h]

    # Clip tau
    mask_tau = traj[:, 3] <= t_max
    ax_tau.plot(traj[mask_tau, 3], traj[mask_tau, 0], '-', color=color, label=fr"$h = {h:.1f}$")

    # Clip t
    mask_t = sol.t <= t_max
    ax_t.plot(sol.t[mask_t], sol.y[0][mask_t], '-', color=color)

for ax in [ax_tau, ax_t]:
    ax.axhline(1, color='black', linestyle='--')
    ax.set_ylabel(r"$r$")
    ax.grid(True)

ax_tau.set_title("Radial Coordinate vs Time")
ax_tau.set_ylabel(r"$r(\tau)$")
ax_tau.set_xlabel(r"$\tau$")
ax_t.set_ylabel(r"$r(t)$")
ax_t.set_xlabel(r"$t$")

handles, labels = ax_tau.get_legend_handles_labels()
fig.legend(handles, labels, loc='upper center', ncol=len(Hs), fontsize=9)
plt.tight_layout(rect=[0, 0, 1, 0.95])
fig.savefig(os.path.join(output_dir, 'Q5_Compare_r_vs_time.pdf'), dpi=300, bbox_inches='tight')
plt.show()

# Spatial Plot
fig_spatial, (ax_left, ax_right) = plt.subplots(1, 2, figsize=(12, 6), sharex=True, sharey=True)

for idx, h in enumerate(Hs):
    color = color_list[idx % len(color_list)]

    # Proper-time
    traj = trajectories_affine[h]
    r_aff, phi_aff = traj[:, 0], traj[:, 2]
    x_aff = r_aff * np.cos(phi_aff)
    y_aff = r_aff * np.sin(phi_aff)
    fading_line(ax_left, x_aff, y_aff, color, label=fr"$h = {h:.1f}$")
    ax_left.plot(x_aff[0], y_aff[0], 'x', color='black', markersize=6)

    # Coordinate-time
    sol = trajectories_coord[h]
```

```
    r_coord = sol.y[0]
    phi_coord = compute_phi_t(sol, h)
    x_coord = r_coord * np.cos(phi_coord)
    y_coord = r_coord * np.sin(phi_coord)
    fading_line(ax_right, x_coord, y_coord, color)
    ax_right.plot(x_coord[0], y_coord[0], 'x', color='black', markersize=6)

for ax in [ax_left, ax_right]:
    ax.add_patch(plt.Circle((0, 0), 1, color='black'))
    ax.set_aspect('equal')
    ax.set_xlim(-6.5, 6.5)
    ax.set_ylim(-6.5, 6.5)
    ax.grid(True)

ax_left.set_title("Proper-Time Spatial Trajectories")
ax_right.set_title("Coordinate-Time Spatial Trajectories")
ax_left.set_xlabel("x")
ax_right.set_xlabel("x")
ax_left.set_ylabel("y")
ax_right.set_ylabel("y")

handles, labels = ax_left.get_legend_handles_labels()
fig_spatial.suptitle("Comparison of Spatial Trajectories", fontsize=14, y=0.98)
fig_spatial.legend(handles, labels, loc='lower center', ncol=len(Hs), fontsize=9, bbox_to_anchor=(0.5,
plt.tight_layout(rect=[0, 0.05, 1, 0.95])
fig_spatial.savefig(os.path.join(output_dir, 'Q5_Spatial_Comparison_SideBySide_Faded.pdf'), dpi=300, bb
plt.show()
```

Q6.py

```
import numpy as np
import matplotlib.pyplot as plt
import os

# Output directory
output_dir = r"C:\CATAM_BlackHoleOrbits\Figures_BH"
os.makedirs(output_dir, exist_ok=True)

def capture_cross_section(v: np.ndarray) -> np.ndarray:
    return (9 * np.pi / 4) * (1 + 2 * v**2) / v**2

def plot_sigma_subplots(v_min=0.01, v_max=0.999,
                        zoom_min=0.3, zoom_max=0.999, n_points=1000):

    # full range
    v_full     = np.linspace(v_min, v_max, n_points)
    sigma_full = capture_cross_section(v_full)

    # zoom range
    v_zoom     = np.linspace(zoom_min, zoom_max, n_points)
    sigma_zoom = capture_cross_section(v_zoom)

    sigma_photon_limit = 27 * np.pi / 4
```

```
24
25      fig, axes = plt.subplots(1, 2, figsize=(12, 5))
26
27      # Full-range plot (black) with zoom segment in blue
28      axes[0].plot(v_full, sigma_full, color='k', linewidth=2)
29      mask = v_full >= zoom_min
30      axes[0].plot(v_full[mask], sigma_full[mask], color='b', linewidth=2)
31      axes[0].set_xlabel('v')
32      axes[0].set_ylabel(r'$\sigma(v)$')
33      axes[0].grid(True, linestyle='--', alpha=0.5)
34
35      # Zoomed-in plot in blue
36      axes[1].plot(v_zoom, sigma_zoom, color='b', linewidth=2)
37      axes[1].set_xlabel('v')
38      axes[1].grid(True, linestyle='--', alpha=0.5)
39
40      # Horizontal dashed line at the photon limit
41      axes[1].axhline(y=sigma_photon_limit, color='k', linestyle='--')
42
43      # Build yticks
44      yticks = list(axes[1].get_yticks())
45      yticks = [t for t in yticks if not np.isclose(t, 20.0)]
46      yticks.append(sigma_photon_limit)
47      axes[1].set_yticks(yticks)
48
49      # Build matching tick labels
50      yticklabels = [f"{y:.1f}" for y in yticks]
51      yticklabels[-1] = r"$\frac{27\pi}{4}$"
52      axes[1].set_yticklabels(yticklabels)
53
54      # Set custom y-limits for the zoomed plot
55      min_zoom = np.min(sigma_zoom)
56      max_zoom = np.max(sigma_zoom)
57      axes[1].set_ylim(min_zoom * 0.9, max_zoom * 1.1)
58
59      # Main title|positioned
60      fig.suptitle("Black Hole Capture Cross-Section", y=0.95, fontsize=14)
61      plt.tight_layout(rect=[0, 0, 1, 0.9])
62
63      plt.savefig(os.path.join(output_dir, 'Q6_sigma_subplots.pdf'), dpi=300)
64      plt.show()
65
66  if __name__ == '__main__':
67      plot_sigma_subplots()
```

Q7.py

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   from scipy.optimize import brentq
4   from scipy.integrate import quad
5   import os
6
```

```
# Output directory
output_dir = r"C:\CATAM_BlackHoleOrbits\Figures_BH"
os.makedirs(output_dir, exist_ok=True)

def f_turning(r, b):
    return 1 - (b**2 * (1 - 1/r) / r**2)

def deflection_angle(b):
    b_crit = 3 * np.sqrt(3) / 2
    if b <= b_crit:
        return np.nan  # captured, no deflection

    # Solve for r_min
    r_min = brentq(f_turning, 1.5001, b, args=(b,))

    # Integrand includes the factor b
    integrand = lambda r: b / (r**2 * np.sqrt(1 - (b**2 * (1 - 1/r) / r**2)))

    # Perform the integral
    I, _ = quad(integrand, r_min, np.inf, limit=200)

    # Return the net deflection
    return 2 * I - np.pi

# Impact parameters
b_values = np.linspace(5, 50, 100)
delta_phi_numeric = np.array([deflection_angle(b) for b in b_values])
delta_phi_approx = 2 / b_values  # Approximation

# Plotting
plt.figure(figsize=(8, 5))
plt.plot(b_values, delta_phi_numeric, 'k-', label='Numerical')
plt.plot(b_values, delta_phi_approx, 'b--', label=r'Approx. $2/b$')
plt.xlabel('$b$')
plt.ylabel(r'$\Delta\phi(b)$')
plt.title('Photon Deflection Angle numerical comparison to analytic approximation')
plt.legend()
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.savefig(os.path.join(output_dir, 'Q7_deflection_angle_large_b.pdf'), dpi=300)
plt.show()
```