14.1 - Particle or Photon Orbits near a Black Hole

Name: Luke Timmons

Student Number: 304757457

Date: 16 September 2019

Abstract

In this project, the motion of particles and photons around a black hole as studied in detail, via a number of methods. The shapes of particle orbits were determined through consideration of the geodesics of the Schwarzschild metric and numerical integration using the Runge Kutta fourth order method and the resulting orbits were subsequently analysed. From the analysis, the angular momentum of the particle for which a circular orbit was obtained, the critical angular momentum from which a particle would just plunge towards the centre of attraction, and the perturbation to the circular orbit angular momentum for this to occur were determined, for a given initial radius of the particle. By continuing this analysis for a range of initial radii, the behaviour of these noteworthy quantities could be successfully studied. By consideration of a second approach for obtaining the shape of the particle orbit, where the numerical integration was with respect to the co-ordinate time rather than the ϕ co-ordinate, the results produced were largely consistent but the second approach was more computationally intensive. Following this, the scattering of particles and photons by the black hole as they infell from very large initial radial positions was investigated. By investigating the scattering of particles for a range of particle speeds and impact parameters, the scattering cross-section of the black hole as a function of the particle speed was obtained and studied. Furthermore, the deflection angles of photons for large impact parameters were considered and compared to the analytical solutions expected from theory. The deflection angles were then determined for increasingly smaller impact parameters, until the approximate impact parameter for which a photon will radially plunge to the centre of the black hole was produced. The results produced from this project were largely consistent with the analytical solutions presented from existing theoretical analysis of the Schwarzschild metric, with the only exception being the results for the deflection angles of photons for large impact parameters which deviated from theoretical expectations.

Contents

1 Introduction

Space-time geometry is often represented as a line element denoting the distance between neighbouring points in that space-time. This line element represents a space-time geometry but due to changes in co-ordinate systems, i.e. between Cartesian, cylindrical, spherical, etc., many different line elements can refer to the same space-time [3]. As such, it can often be helpful to move between co-ordinate systems depending on what type of scenario is under investigation, as different co-ordinate systems may reveal symmetries within the space-time. An example of such a case is the line element of flat space-time which can be described in Cartesian and spherical polar co-ordinates, respectively, as:

$$ds^2 = -dt^2 + dx^2 + dy^2 + dz^2 \tag{1}$$

$$ds^2 = -dt^2 + dr^2 + r^2 d\theta^2 + r^2 \sin^2\theta d\phi^2 \tag{2}$$

The independence of the flat space-time line element from ϕ indicates that the flat space-time geometry is in fact spherically symmetric, a fact that may not be readily apparent when analysing the line element in Cartesian co-ordinates. [3, 4, 2]

The line element of any space-time can typically be described by the metric of that space-time, where a space-time metric is a 4×4 symmetric, position-dependent matrix which describes the geometric and structure of the space-time [3]. As such, the line element is described by the relation:

$$ds^2 = g_{\alpha\beta} dx^\alpha dx^\beta \tag{3}$$

where dx^α is a coordinate interval and $g_{\alpha\beta}$ is metric described by:

$$g_{\alpha\beta} = \begin{pmatrix} -g_{00} & g_{01} & g_{02} & g_{03} \\ g_{10} & g_{11} & g_{12} & g_{13} \\ g_{20} & g_{21} & g_{22} & g_{23} \\ g_{30} & g_{31} & g_{32} & g_{33} \end{pmatrix} \tag{4}$$

For example, the metric for flat space-time, in spherical co-ordinates, is given by [3, 4]:

$$g_{\alpha\beta} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & r^2 & 0 \\ 0 & 0 & 0 & r^2\sin^2\theta \end{pmatrix} \tag{5}$$

Or more simply as:

$$g_{\alpha\beta} = \mathrm{diag}(-1, 1, r^2, r^2\sin^2\theta) \tag{6}$$

The geodesics of a space-time are often used to describe the motion of a test particle or photon in a particular space-time geometry [3, 4]. These geodesics are the equations of motion of the test particles where the geodesics must conform to the variational principle for test particles [3]. This principle states that geodesics are worldlines, for particles, or null-points, for photons, that extremise the proper time between two timelike points [3]. The geodesics for a given space-time geometry can be obtained by solving the Euler-Lagrange equation for the Lagrangian of the space-time for each co-ordinate, where the Lagrangian of a space-time is given by:

$$L = -\frac{ds^2}{d\lambda^2} = -\left(\frac{ds}{d\lambda}\right)^2 \tag{7}$$

and the Euler-Lagrange equation is defined as:

$$-\frac{d}{d\lambda}\left(\frac{\partial L}{\partial \dot{x^\alpha}}\right) + \frac{\partial L}{\partial x^\alpha} = 0 \tag{8}$$

where x^α is a coordinate of the metric, i.e. $x^0 = t$, $x^1 = r$, etc, and λ is the Affine parameter [3].

The geodesics of a space-time geometry can also be determined by consideration of the Christoffel symbols of the metric, where the equations of motion are given by:

$$\frac{d^2x^\alpha}{d\alpha^2} = -\Gamma^\alpha_{\beta\gamma}\frac{dx^\beta}{d\lambda}\frac{dx^\gamma}{d\lambda} \tag{9}$$

and the Christoffel symbol $\Gamma^\alpha_{\beta\gamma}$ [4, 3] is defined by:

$$g_{\alpha\beta}\Gamma^\delta_{\beta\gamma} = \frac{1}{2}\left(\frac{\partial g_{\alpha\beta}}{\partial x^\gamma} + \frac{\partial g_{\alpha\gamma}}{\partial x^\beta} - \frac{\partial g_{\beta\gamma}}{\partial x^\alpha}\right) \tag{10}$$

A very famous and well-studied geometry, in both theoretical and experimental settings, is the Schwarzschild space-time geometry as it is one of the simplest curved space-time geometry, notable for its symmetry in several co-ordinates [3, 2, 4]. This space-time geometry describes the geometry of the empty space surrounding a spherically symmetric source of curvature such as a spherical star or a black hole. The Schwarzschild space-time is described by the line element:

$$ds^2 = -\left(1 - \frac{r_s}{r}\right)dt^2 + \left(1 - \frac{r_s}{r}\right)^{-1}dr^2 + r^2 d\Omega^2 \tag{11}$$

where:

$$d\Omega^2 = \sin^2\theta d\theta^2 + d\phi^2 \tag{12}$$

and r_s is the Schwarzschild radius [3, 4].

From analysis of this metric, there are several symmetries and notable properties that are readily apparent. Firstly, the metric of the Schwarzschild geometry does not have an explicit dependence on the co-ordinates t and ϕ, indicating that the metric is spherically symmetric and is symmetrical under displacements in co-ordinate time. The symmetry in time and spherical symmetry can be described respectively by the Killing vectors:

$$\xi = (1, 0, 0, 0) \tag{13}$$

$$\eta = (0, 0, 0, 1) \tag{14}$$

These Killing vectors imply conserved quantities in the metric [3, 4]. In this case, the Killing vector for the t co-ordinate shows that the total energy E is constant, while for the ϕ co-ordinate, the Killing vector implies constant angular momentum [3, 4].

The line element for Schwarzschild space-time also breaks down for two values of r: $r = 0$ and $r = r_s$. The first case represents a co-ordinate singularity as the time component of the metric blowing up to infinity. The second case represents the Schwarzschild radius, the characteristic length scale of curvature in the Schwarzschild geometry [3]. The Schwarzschild radius is often found to lie outside the 'surface' of a black hole while for spherical, static stars, the radius lies within the surface of the stars.

Schwarzschild space-time geometry is often analysed by studying the motion of particles and photons around the source of curvature. The orbits of particles and photons around a black hole or spherical stars can be loosely categorised into several types: circular, precessional, radial plunge and scattering orbits [3, 4]. The nature of these orbits are analysed much in the same way as in Newtonian mechanics, by the analysis of an equation of the form:

$$\left(\frac{dr}{d\tau}\right)^2 = E^2 - L - V_{eff}(r) \tag{15}$$

where E is the total energy of the particle and V_{eff} is the effective potential of the particle described by:

$$V_{eff} = \frac{l^2}{r^2} - \frac{Lr_s}{r} - \frac{l^2 r_s}{r^3} \tag{16}$$

where l is the angular momentum of the particle and L is the value of the Lagrangian, either one for particles, or zero for photon [3].

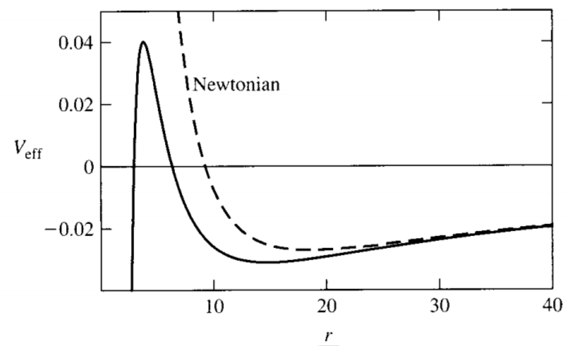


Figure 1: The effective potential of particles in a Schwarzschild (solid line) and Newtonian (dashed line) space-time geometries as a function of $\frac{r}{M}$. Note the agreement between the potentials at very large distances and the divergence of the Schwarzschild potential as $r \to 0$ [3].

By examining the form of the equation for the effective potential of the particle at very large distances, approaching infinity, and at small distances from the centre, i.e. at the Schwarzschild radius, some noteworthy observations can be made. Firstly, as the particle approaches infinity, the effective potential of the particle simplifies and begins to resemble that of a particle in Newtonian geometry with the r^{-2} and r^{-3} terms effectively vanishing [3, 4, 2]. However as the particle approaches the Schwarzschild radius, the r^{-3} begins to become more influential in the effective potential, resulting in a divergence from the r^{-1} potential observed in Newtonian mechanics (Fig. 1).

Ultimately, the behaviour of an orbit is dependent on two factors, the total energy of the particle, and the effective potential of the particle, which in turn have explicit dependencies on the angular momentum, speed, and position of the particle, and the mass of the black hole or star around which it is orbiting [3, 4, 2]. Circular orbits occur for radii at which the effective potential is either at a maximum or minimum, where a minimal effective potential represents a stable circular orbit and a maximal effective potential defines an unstable circular orbit, for which small perturbations in the angular momentum of the particle will result in the particle either escaping to infinity or undergoing a radial plunge orbit. Circular orbits can not be achieved at increasingly smaller radii due to instability of the effective potential at arbitrarily small distances. The radius of the Innermost Stable Circular Orbit (ISCO) [4, 3] can be determined from the effective potential and is found to be:

$$r_{ISCO} = 3r_s \tag{17}$$

If a circular orbit is achieved within this radius, small perturbations of the angular momentum of the particle will result in the decay of the circular orbit, likely leading to a radial plunge orbit [3, 4]. Circular orbits of particles are not possible at all within the photon sphere of a black hole, where:

$$r_{p-sphere} = \frac{3r_s}{2} \tag{18}$$

Further examination of Eq. 16, through differentiation with respect to the radial co-ordinate r, lead to radii at which the effective potential is maximal or minimal [3], given by the relation:

$$r_{\substack{min\\max}} = \frac{l^2}{r_s}\left(1 \pm \sqrt{1 - 3\frac{r_s^2}{l^2}}\right) \tag{19}$$

This equation can be re-arranged to obtain an expression for the angular momentum corresponding to those circular orbits:

$$l = \pm \frac{r_{\substack{min\\max}}}{\sqrt{\frac{2r_{\substack{min\\max}}}{r_s} - 3}} \tag{20}$$

Other bound orbits occur for instances in which the term (E^2 - L), in Eq. 15, is less than zero, where the resulting orbits resemble elliptical orbits. If the bound orbits in question do not close, the result is a precession of the perihelion of the orbit, where the resulting orbit appears to be elliptical in nature but slowly rotates about the centre of attraction [4, 3]. This phenomenon has been observed within the solar system, where the perihelion of Mercury has been observed to precess. This observation was an early experimental confirmation of the theories of general relativity [3, 4, 2].

If the orbit is such that the (E^2-1) term is positive and greater than the maximum of the effective potential, the orbit in question is termed a radial plunge orbit.

A radial plunge orbit is any orbit of a particle about a centre of attraction in which the particle moves partially around the mass before plunging towards the centre and passing the Schwarzschild radius. After passing the Schwarzschild radius, the particle or photon is unable to escape the gravitational body as doing so would require particle speeds exceeding that of the speed of light. The simplest case of radial plunge orbit is that of a stationary particle at infinity [3]. Several noteworthy phenomena can be observed from a case as simple as this. For example, upon investigation of the geodesics of this case, it can be seen that for an initial finite radial position, the particle will reach the Schwarzschild radius at a finite proper time, but an infinite co-ordinate time. The presence of infinite co-ordinate values corresponding to a change in a finite distance is indicative of flaws in the Schwarzschild geometry [3, 4].

When the (E^2-1) term is positive, but happens to be less than the maximum of the effective potential, the orbit of the particle is a scattering orbit where the particle moves in from infinity, orbits the centre of attraction, and then proceeds to move once again outwards to infinity [3].

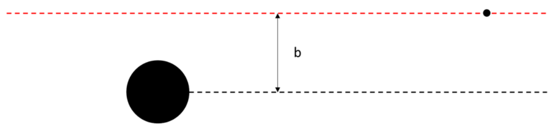


Figure 2: A particle approaching a black hole from infinity, where the red dashed line represents the undeflected path of the particle and b is the impact parameter.

Much like particles, photons can be scattered and deflected as they orbit around a black hole. The amount of deflection observed from a photon as it is scattered by the black hole is largely dependent on the impact parameter of the photon, where the impact parameter is defined as the perpendicular distance from the centre of attraction to the parallel path of the photon at very large distances, or infinity (Fig. 2). For scattering orbits, the equation describing the motion is of the form:

$$\frac{dr}{d\phi} = \pm r^2 \left(\frac{1}{b^2} - V_{eff}(r)\right)^{\frac{1}{2}} \tag{21}$$

where b is the impact parameter defined as $\frac{l}{E}$ [2, 3, 4].

In the instances in which the b^{-2} term is equal to the maximum of the effective potential, the orbit is, in fact, a circular orbit of the photon around the black hole [4, 3, 2]. For the b^{-2} term being less than the maximum effective potential, the photon undergoes a scattering orbit while for the case where the b^{-2} term is greater than the maximum effective potential, a radial plunge of the photon towards the centre of attraction is observed [3].

For these orbits, turning points in the orbit (r_t) occur for radial positions for which $b^{-2} = V_{eff}(r_t)$. The deflection angle can then be determined from the integral:

$$\Delta\phi = \int_{r_t}^{\infty} \frac{2}{r}\left(\frac{1}{b^2} - V_{eff}(r)\right)^{-\frac{1}{2}} dr \tag{22}$$

where the effective potential for a photon is:

$$V_{eff} = \frac{1}{r^2}\left(1 - \frac{r_s}{r}\right) \tag{23}$$

For the very large limit in the impact parameter b, the integral in Eq. 22 reduces to:

$$\Delta\phi \longrightarrow \pi + \frac{2r_s}{b} \tag{24}$$

where $\delta\phi$ is the change in the ϕ co-ordinate of the photon [3].

The deflection angle of the photon $(\delta\phi)$ can then be determined by finding the difference between the change in ϕ co-ordinates and that of an undeflected path, i.e. π:

$$\delta\phi = \Delta\phi - \pi = \frac{2r_s}{b} \tag{25}$$

2 Theory

2.1 Derivation of Equations of Motion:

In order to derive the equations of motion for a particle or photon in orbit about a black hole, the Schwarzschild metric, given in Eq. 11, is constrained to azimuthal plane such that $d\theta = 0$ and $\theta = \frac{\pi}{2}$, resulting in the following simplified metric:

$$ds^2 = -\left(1 - \frac{1}{r}\right)dt^2 + \left(1 - \frac{1}{r}\right)^{-1} dr^2 + r^2 d\phi^2 \tag{26}$$

It now remains to determine the Lagrangian of the system, determined by dividing the line element of the metric ds^2 by the squared differential of the affine parameter $d\lambda^2$:

$$L = -\frac{ds^2}{d\lambda^2} = \left(\frac{ds}{d\lambda}\right)^2 \tag{27}$$

As such, the resulting Lagrangian for this case of the Schwarzschild metric is given by:

$$L = \left(1 - \frac{1}{r}\right)\dot{t}^2 - \left(1 - \frac{1}{r}\right)^{-1} \dot{r}^2 - r^2\dot{\phi}^2 \tag{28}$$

where $\dot{t}$, $\dot{r}$, and $\dot{\phi}$ are the derivatives of the parameters with respect to the affine parameter λ.

It now remains to parameterise the metric as follows:

$$r = \frac{1}{u} \rightarrow \dot{r} = -\frac{1}{u^2}\dot{u} = -u^{-2}\dot{u} \quad (29)$$

Resulting in the following Lagrangian:

$$L = (1-u)\dot{t}^2 - (1-u)^{-1}u^{-4}\dot{u}^2 - \frac{\dot{\phi}^2}{u^2} \quad (30)$$

In order to determine the equations of motion for the particle or photon for each coordinate, the Euler-Lagrange equation must be solved in each instance, where the Euler-Lagrange equation is given by:

$$-\frac{d}{d\lambda}\left(\frac{\partial L}{\partial \dot{x}^\alpha}\right) + \frac{\partial L}{\partial x^\alpha} = 0 \quad (31)$$

where x^α is a coordinate of the metric, i.e. $x^0 = t$, $x^1 = r$, etc.

For the t and ϕ coordinates, the Euler-Lagrange equation is simplified further by considering the Schwarzschild metric (Eq. 11) which has two Killing vectors, one in the t direction and the other in the ϕ direction, noted by the fact that the metric is not explicitly dependent on t or ϕ. As such the derivative of the Lagrangian with respect to these parameters is zero. Therefore the Euler-Lagrange equation for the t coordinate can be solved as follows:

$$\frac{d}{d\lambda}\left(\frac{\partial L}{\partial \dot{t}}\right) = \frac{d}{d\lambda}\left(2(1-u)(\dot{t})\right) = 0 \quad (32)$$

$$(1-u)\dot{t} = E \quad (33)$$

$$\dot{t} = \frac{E}{1-u} \quad (34)$$

where E is a constant of integration denoting the energy of the particle or photon.

Similarly, the Euler-Lagrange equation for the ϕ coordinate can be solved as follows:

$$\frac{d}{d\lambda}\left(\frac{\partial L}{\partial \dot{\phi}}\right) = \frac{d}{d\lambda}\left(2u^{-2}\dot{\phi}\right) = 0 \quad (35)$$

$$\dot{\phi} = u^2 l \quad (36)$$

where l is a constant of integration denoting the angular momentum per unit mass of the particle or photon.

The equation of motion for the r coordinate can be found by considering the Lagrangian and multiplying it by a factor of (1-u) as follows:

$$L(1-u) = (1-u)^2\dot{t}^2 - u^{-4}\dot{u}^2 - (1-u)\frac{\dot{\phi}^2}{u^2} \quad (37)$$

Substituting Eqs. 34 and 36 into Eq. 37, the final equation of motion for the u parameter can be found as follows

$$u^{-4}\dot{u}^2 = E^2 - (1-u)\left(L + u^2 l^2\right) \quad (38)$$

$$\dot{u}^2 = u^4(E^2 - (1-u)(L + u^2 l^2)) \quad (39)$$

The factor L is dependent on the nature of the particle and is equal to zero for photons and to one for particles, such that the equation of motion for particles is given by:

$$\dot{u}^2 = u^4(E^2 - (1-u)(1 + u^2 l^2)) \quad (40)$$

And the equation of motion for photons is:

$$\dot{u}^2 = u^4(E^2 - (1-u)u^2 l^2) \quad (41)$$

The equations of motion in the t and ϕ directions are independent of L such that the equations of motion of a particle orbiting a black hole are given by Eqs. 34, 36, and 40 and the equations of motion of a photon orbiting a black hole are given by Eqs. 34, 36, and 41.

2.2 Derivation of the Equation of Particle Orbit

The relation describing the orbit of a particle about a black hole can be determined by dividing the square of the equation of motion in the u direction by the square of the equation of motion in the ϕ direction as follows:

$$\frac{\dot{u}^2}{\dot{\phi}^2} = \left(\frac{du}{d\phi}\right)^2 = \frac{u^4(E^2 - (1-u)(1+u^2l^2))}{l^2u^4} \quad (42)$$

$$\left(\frac{du}{d\phi}\right)^2 = \frac{E^2}{l^2} - (1-u)\left(\frac{1}{l^2} + u^2\right) \quad (43)$$

It now remains to differentiate the resulting equation with respect to the parameter ϕ:

$$2\left(\frac{du}{d\phi}\right)\left(\frac{d^2u}{d\phi^2}\right) = \left(\frac{1}{l^2} - 2u + 3u^2\right)\left(\frac{du}{d\phi}\right) \quad (44)$$

$$\frac{d^2u}{d\phi^2} = \frac{1}{2l^2} - u + \frac{3u^2}{2} \quad (45)$$

After obtaining Eq. 45, the relation must be parameterised again with $u = \frac{1}{r}$:

$$\frac{du}{d\phi} = -\frac{1}{r^2}\frac{dr}{d\phi} \rightarrow \frac{d^2u}{d\phi^2} = \frac{2}{r^3}\left(\frac{dr}{d\phi}\right)^2 - \frac{1}{r^2}\frac{d^2r}{d\phi^2} \quad (46)$$

Inserting Eq. 46 into Eq. 45, the following relation is obtained:

$$\frac{2}{r^3}\left(\frac{dr}{d\phi}\right)^2 - \frac{1}{r^2}\frac{d^2r}{d\phi^2} = \frac{1}{2l^2} - \frac{1}{r} + \frac{3}{2r^2} \quad (47)$$

This can then be rearranged to obtain the equation describing the orbit of the particle around the black hole:

$$\frac{d^2r}{d\phi^2} = \frac{2}{r}\left(\frac{dr}{d\phi}\right)^2 - \frac{r^2}{2l^2} + r - \frac{3}{2} \tag{48}$$

2.3 Derivation of Shape of Orbit via Coordinate Time

Consider Eq. 39 and divide by the square of the equation of motion in the t co-ordinate (Eq. 34), such that the following relation is obtained:

$$\left(\frac{du}{dt}\right)^2 = \frac{u^4(1-u)^2}{E^2}(E^2 - (1-u)(1+u^2l^2)) \tag{49}$$

Differentiating Eq. 49 with respect to the co-ordinate time t, the following relating is obtained:

$$\frac{d^2u}{dt^2} = -\frac{u^3(1-u)}{2E^2}((6u-4)E^2 + (u-1)(9u^3l^2 - 6u^2l^2 + 7u - 4)) \tag{50}$$

Inserting Eq. 46 into Eq. 50, the second derivative of the r co-ordinate with respect to t can be found as follows:

$$\frac{2}{r^3}\left(\frac{dr}{dt}\right)^2 - \frac{1}{r^2}\frac{d^2r}{dt^2} = -\frac{1-\frac{1}{r}}{2E^2r^3}\left(\left(\frac{6}{r}-4\right)E^2 + \left(\frac{1}{r}-1\right)\left(\frac{9l^2}{r^3} - \frac{6l^2}{r^2} + \frac{7}{r} - 4\right)\right) \tag{51}$$

$$\frac{d^2r}{dt^2} = \frac{2}{r}\left(\frac{dr}{dt}\right)^2 + \frac{1-\frac{1}{r}}{2r}\left(\left(\frac{6}{r}-4\right) + \frac{\left(\frac{1}{r}-1\right)}{E^2}\left(\frac{9l^2}{r^3} - \frac{6l^2}{r^2} + \frac{7}{r} - 4\right)\right) \tag{52}$$

Now considering Eq. 36 and dividing by Eq. 34, the reciprocal of the total energy of the particle can be found as follows:

$$\frac{d\phi}{dt} = \frac{d\phi}{d\tau}\frac{d\tau}{dt} = \frac{l}{r^2}\frac{\left(1-\frac{1}{r}\right)}{E} \tag{53}$$

$$\frac{1}{E} = \frac{r^2}{l\left(1-\frac{1}{r}\right)}\frac{d\phi}{dt} \tag{54}$$

Define the derivative of the radial co-ordinate r with respect to the co-ordinate time as:

$$z = \frac{dr}{dt} \tag{55}$$

Inserting Eqs. 54 and 55 into Eq. 52, the following equation of motion was obtained:

$$\frac{dz}{dt} = \frac{2z^2}{r} + \frac{1-\frac{1}{r}}{2r}\left(\left(\frac{6}{r}-4\right) + \frac{r^4}{l\left(\frac{1}{r}-1\right)}\left(\frac{d\phi}{dt}\right)^2\left(\frac{9l^2}{r^3} - \frac{6l^2}{r^2} + \frac{7}{r} - 4\right)\right) \tag{56}$$

Returning to Eq. 39, and parameterising with $u = \frac{1}{r}$, the equation of motion in the ϕ co-ordinate with respect to the co-ordinate time can be determined, in terms of z and r, as follows:

$$\left(\frac{dr}{dt}\right)^2 = \left(1-\frac{1}{r}\right)^2 - \frac{1}{E^2}\left(1-\frac{1}{r}\right)^3\left(1+\frac{l^2}{r^2}\right) \tag{57}$$

$$z^2 = \left(1-\frac{1}{r}\right)^2 - \frac{r^4}{l^2}\left(1-\frac{1}{r}\right)\left(1+\frac{l^2}{r^2}\right)\left(\frac{d\phi}{dt}\right)^2 \tag{58}$$

$$\frac{d\phi}{dt} = \pm\frac{l}{r^2}\left(\frac{\left(1-\frac{1}{r}\right)^2 - z^2}{\left(1-\frac{1}{r}\right)\left(1+\frac{l^2}{r^2}\right)}\right)^{\frac{1}{2}} \tag{59}$$

As such, Eqs. 52, 55, and 59 are a set of coupled equations which can be used to describe the shapes of orbits of particles around a black hole or stationary star.

2.4 Derivation of Scattering Orbit Equation

In order to find the equation of motion of a particle or photon as they are scattered by the black hole, Eq. 39 must be considered once again:

$$\left(\frac{du}{d\lambda}\right)^2 = u^4\left(E^2 - (1-u)(L+u^2l^2)\right) \tag{60}$$

Reparameterising the relation with $u = \frac{1}{r}$, as in Eq. 46, and expanding the equation, the following equation of motion in the r co-ordinate is found:

$$\frac{1}{r^4}\left(\frac{dr}{d\lambda}\right)^2 = \frac{1}{r^4}\left(E^2 - \left(1-\frac{1}{r}\right)\left(L+\frac{l^2}{r^2}\right)\right) \tag{61}$$

$$\left(\frac{dr}{d\tau}\right)^2 = E^2 - L - \frac{l^2}{r^2} + \frac{L}{r} + \frac{l^2}{r^3} \tag{62}$$

It is now necessary to consider the equation of motion in terms of the first derivative of r with respect to ϕ, by dividing Eq. 62 by the square of Eq. 36, as follows:

$$\left(\frac{dr}{d\phi}\right)^2 = \frac{r^4}{l^2}\left(E^2 - L - \frac{l^2}{r^2} + \frac{L}{r} + \frac{l^2}{r^3}\right) \tag{63}$$

$$\left(\frac{dr}{d\phi}\right)^2 = r^4\left(\frac{E^2}{l^2} - \frac{L}{l^2} - \frac{1}{r^2} + \frac{L}{rl^2} + \frac{1}{r^3}\right) \tag{64}$$

$$\left(\frac{dr}{d\phi}\right)^2 = r^4\left(\frac{E^2-L}{l^2} - \frac{1}{r^2} + \frac{L}{rl^2} + \frac{1}{r^3}\right) \tag{65}$$

Define the impact parameter for the black hole (b) and the speed of the particle or photon as:

$$b = \frac{l}{p} \tag{66}$$

$$v = \frac{p}{E} \tag{67}$$

where p is the linear momentum of the particle or photon which through the equivalence principle is given by [3, 4, 2]:

$$p = (E^2 - L)^{\frac{1}{2}} \rightarrow p^2 = E^2 - L \tag{68}$$

As such, L can be found in terms of the speed as follows:

$$L = p^2 \left(\frac{E^2}{p^2} - 1 \right) = p^2 \left(\frac{1}{v^2} - 1 \right) \tag{69}$$

Inserting Eq. 67 into Eq. 69, the following relation is found:

$$L = \frac{l^2}{b^2} \left(\frac{1 - v^2}{v^2} \right) \tag{70}$$

$$\frac{L}{l^2} = \frac{1 - v^2}{b^2 v^2} \tag{71}$$

Inserting Eqs. 71 and 66 into Eq. 65, the derivative of r with respect to ϕ can be simplified to an equation consisting solely of the impact parameter b, the speed of the particle or photon, and the r co-ordinate:

$$\left(\frac{dr}{d\phi} \right)^2 = r^4 \left(\frac{p^2}{l^2} - \frac{1}{r^2} + \frac{L}{rl^2} + \frac{1}{r^3} \right) \tag{72}$$

$$\left(\frac{dr}{d\phi} \right)^2 = r^4 \left(\frac{1}{b^2} - \frac{1}{r^2} + \frac{1 - v^2}{rv^2 b^2} + \frac{1}{r^3} \right) \tag{73}$$

Thus, the final equation of motion for a particle scattered by the black hole is given by:

$$\frac{dr}{d\phi} = \pm r^2 \left(\frac{1}{b^2} - \frac{1}{r^2} + \frac{1 - v^2}{rv^2 b^2} + \frac{1}{r^3} \right)^{\frac{1}{2}} \tag{74}$$

For the case in which a photon is scattered by the black hole, $v = c = 1$ such that Eq. 74 simplifies to:

$$\frac{dr}{d\phi} = \pm r^2 \left(\frac{1}{b^2} - \frac{1}{r^2} + \frac{1}{r^3} \right)^{\frac{1}{2}} \tag{75}$$

The angle through which the photon is scattered, i.e. the deflection angle ($\delta\phi$), is given by the equation:

$$\delta\phi = \phi_f - \phi_i - \pi \tag{76}$$

where ϕ_f and ϕ_i are the final and initial ϕ co-ordinates of the particle.

2.5 Numerical Integration: Runge-Kutta Fourth Order Method

The method of numerical integration implemented in this code was the Runge-Kutta Fourth Order (RK4) method. The RK4 method is typically implemented in initial value problems for first-order ordinary differential equations [1]. In these problems, the result of the integral is unknown and must be approximated via numerical means by determining the rate of change of the function, say y, when given an initial value for the function and a step size over which to the derivative is incremented thus giving an approximate value for the function beyond the initial value [1]. Consider a first-order ODE:

$$\dot{y} = \frac{dy}{dx} = f(x, y) \tag{77}$$

with the initial condition:

$$y(x_0) = y_0 \tag{78}$$

Defining y_N as the value of the function to be obtained for a value of x_N, where N is the number of increments between the initial and final value of x [1], the step size can be defined by:

$$h = \frac{y_N - y_0}{N} \tag{79}$$

The i^{th} value of y and x can then be defined [1] by the following relation:

$$y_i = y_{i-1} + \frac{1}{6} (k_1 + 2k_2 + 2k_3 + k_4) \tag{80}$$

$$x_i = x_{i-1} + h \tag{81}$$

where k_1, k_2, k_3, and k_4 are defined by [1]:

$$k_1 = hf(x_{i-1}, y_{i-1}) \tag{82}$$

$$k_2 = hf \left(x_{i-1} + \frac{1}{2}h, y_{i-1} + \frac{1}{2}k_1 \right) \tag{83}$$

$$k_3 = hf \left(x_{i-1} + \frac{1}{2}h, y_{i-1} + \frac{1}{2}k_2 \right) \tag{84}$$

$$k_4 = hf \left(x_{i-1} + h, y_{i-1} + k_3 \right) \tag{85}$$

3 Implementation of Code

In order to implement the RK4 method, the equation for the orbit of the particle had to be reconsidered as it was a second-order ordinary differential equation where the RK4 method relies on the ODE to be solved being of the first-order [1]. However, this issue was resolved by parameterising the second-order equation so that it became a first order equation. In this case, the paramaterisation was as follows:

$$\frac{dr}{d\phi} = q = g(r, \phi) \tag{86}$$

As such, the equation of the orbit can be redefined as:

$$\frac{dq}{d\phi} = \frac{2q^2}{r} - \frac{r^2}{2l^2} + r - \frac{3}{2} = f(q, r, \phi) \tag{87}$$

which is now a method to which the RK4 method can be applied with:

$$q_i = q_{i-1} + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{88}$$

$$r_i = r_{i-1} + \frac{1}{6}(m_1 + 2m_2 + 2m_3 + m_4) \quad (89)$$

$$\phi_i = \phi_{i-1} + h \quad (90)$$

The factors k and m are defined as in the RK4 method outlined in Sect. 2.5, i.e.:

$$k_1 = hf(q_{i-1}, r_{i-1}, \phi_{i-1}) \quad (91)$$

$$m_1 = hg(r_{i-1}, \phi_{i-1}) \quad (92)$$

The proper time of the particle (τ) can also be determined via the RK4 method and by considering the reciprocal of Eq. 36:

$$\frac{d\lambda}{d\phi} = \frac{d\tau}{d\phi} = \frac{r^2}{l} = a(r, \phi) \quad (93)$$

As with r and q:

$$\tau_i = \tau_{i-1} + \frac{1}{6}\left(n_1 + 2n_2 + 2n_3 + n_4\right) \quad (94)$$

where, as before, the n parameters follow the scheme:

$$n_1 = ha(r, \phi) \quad (95)$$

3.1 Plotting of Particle Orbits

The python script used throughout this project implemented the RK4 method of numerical integration by setting an initial value for the parameters and a value for the angular momentum l, and incrementing through a loop within which the values of q, r, ϕ, and τ were calculated, via Eqs. 88, 89, 90, and 94, and placed into an array. At the end of each iteration of the loop, an if statement was used to determine whether the radial position of the particle had reached the Schwarzschild radius for the black hole or fallen below that value. In this event, the time taken for the radial plunge to occur was set as the value of an array and the loop was broken. Subsequently Matplotlib, specifically Pyplot, was used to produce a polar plot of the subsequent orbit of the particle based on the initial conditions and value for the angular momentum provided, where the initial radius and angular momentum were displayed in the legend of the plot as well as the plunge time, in the event of a radial plunge occurring.

3.2 Determination of Critical Angular Momenta

Following this, the code entered another loop where the value for the angular momentum was incremented through a loop. Within this loop, the RK4 method was employed, as in Sect.3.1, with the exception of the production of a polar plot. For each value for which a radial plunge orbit occurred, the value of the angular momentum was appended to an array and the proper time taken for the radial plunge to occur was appended to a separate array. As such, the first and final values in the angular momentum array provided the bounds of the angular momentum for which a radial plunge orbit, where the final value in the array was the critical angular momentum. The proper time for a radial plunge to occur was plotted as a function of angular momentum and the angular momentum for which a circular orbit will occur was calculated from Eq. 87 for the case of $\frac{dq}{d\phi} = 0$ and $q = 0$. The arrays for the radial plunge angular momenta and their corresponding proper times for the plunge to occur were then saved to a CSV file.

3.3 Analysis of Stability of Circular Orbits

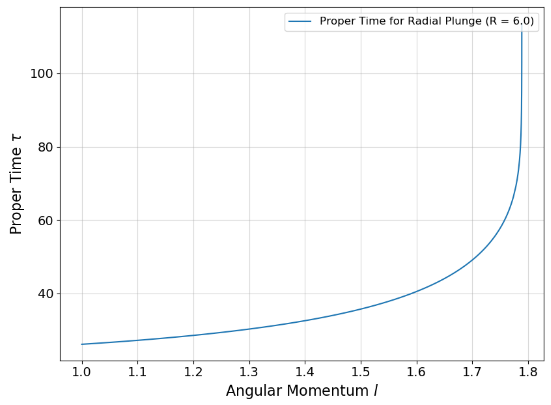
For the analysis of the orbits at increasingly smaller radii, the code followed the same procedure as in Sect. 3.2 where the code incremented the value of the initial radius of the orbit. For each iteration of the initial radial position, the array of angular momentum values for which a radial plunge occurred were saved to CSV file, along with the array for the proper time taken for the radial plunge to occur. The final values in the radial plunge angular momentum arrays were appended to a separate array for the critical angular momenta and the angular momenta values for a circular orbit, calculated from Eq. 87, were appended to an array for circular orbit angular momenta, for each initial radius value. The difference between the circular orbit angular momentum and the critical angular momentum, i.e. the perturbation required to cause a circularly orbiting particle to plunge to the centre, was also calculated by the code and appended to an array for the perturbation values, for each initial radius. The code then saved the initial radii, circular orbit angular momentum, critical angular momentum, and critical perturbation arrays were saved to a CSV. The critical angular momenta and circular orbit angular momenta were then plotted as a function of initial radius, on the same plot, and the critical perturbation was plotted as a function of initial radius, on a separate plot.

3.4 Particle Orbits via $r(t)$ and $\phi(t)$

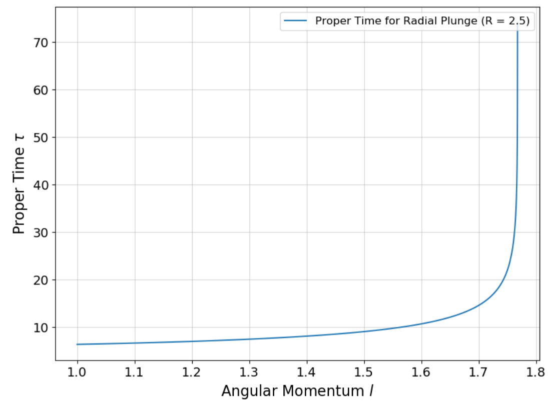
For the case of determining the particle orbits by numerically integrating the equations of motion (Eqs. 56, 55, and 59) to obtain $r(t)$ and $\phi(t)$, the code followed the same method as in Sects. 3.1 and 3.2, where the initial and final values over which Eqs. 52, 55, and 59 were to be integrated were in terms of the co-ordinate time t rather than the ϕ co-ordinate. The proper time for the radial plunge was determined by applying the RK4 method to the equation:

$$\frac{d\tau}{dt} = \frac{d\phi}{dt}\frac{r^2}{l} \quad (96)$$

A polar plots for the particle orbit was then produced by the code, for the given initial radius and the angular



(a)



(b)

Figure 3: Evolution of the proper time taken for a radial plunge to occur as a function of the angular momentum of the particle in orbit around the black hole, initially at a radius of (a): $R = 6$; and (b): $R = 2.5$.

momentum of the particle.

3.5 Scattering Cross-section of the Black Hole

In order to obtain the scattering cross-section of the black hole as a function of the particle speed, the code implemented the RK4 method to numerically integrate both the positive and negative instances of Eq. 74, while looping the values of the impact parameter, and the speed of the particle. The code applied the RK4 method to the negative equation initially, as the particle was infalling from a set initial value, calculating the resulting values of the r co-ordinate for each value of ϕ, while checking for each value of r whether the particle had fallen below the Schwarzschild radius or escaped to its initial position. If the position fell below the Schwarzschild radius, the loop was broken. If the particle reached its initial radial position, the value for b, v, and the scattering cross-section were appended to separate arrays for the critical values. A second test was implemented, checking if the term within the square root in Eq. 74 was equal to or less than zero. If this was the case, the code entered another for loop, incrementing through the remaining iterations of the original loop but with the initial radial position being that of the iteration before the test condition was satisfied. In this new loop, the RK4 method was applied to the positive equation, with the same testing conditions for the radial position of the particle as before. The code incremented through a range of values for the impact parameter for each value for the particle speed. After completing the method for each value of v, the scattering cross-section of the black hole was plotted as a function of particle speed. The code also implemented a power law model fit applied to the results produced via the RK4 method, where the model was plotted alongside the results of the RK4 method, and the parameters of the fit were printed by the code.

3.6 Deflection Angle of Photon

To determine the deflection angle of a photon as it was scattered by the black hole, the method outlined for analysing the particle scattering was applied, but for a fixed value of $v = 1$ and with some minor changes to the test conditions for the radial position of the photons. The RK4 method was implemented exactly as before but for the condition in which the photon escaped back to a radial position exceeding the initial value, the deflection angle was calculated via Eq. 76 and the result appended to an array for the deflection angles and the value for the impact parameters appended to a separate array. After iterating through each value for the impact parameter, the deflection angles of the photon were plotted as a function of the impact parameter. The code implemented a hyperbolic fit which was applied to the results for the deflection angle produced by the RK4 method and plotted alongside the computed deflection angles. The parameters of the model fit were then printed by the code.

The code also produced polar plots of the photons motion as it was scattered by the black hole, in the instance in which a fixed value for the impact parameter was given.

4 Results

4.1 Analysis of Orbits at R = 6

From Fig. 4a, it can be seen that for particle angular momenta between $l = 0.0$ and $l = 1.6$ that particles with an initial orbital radius of $R = 6$ underwent radial plunge orbits such that the radial position of the particle decayed to the Schwarzschild radius, where the particle was unable escape the attraction of the black hole. By iterating through values for the angular momenta $1 \leq l \leq 5$ in increments of $\delta l = 0.00001$, it was found

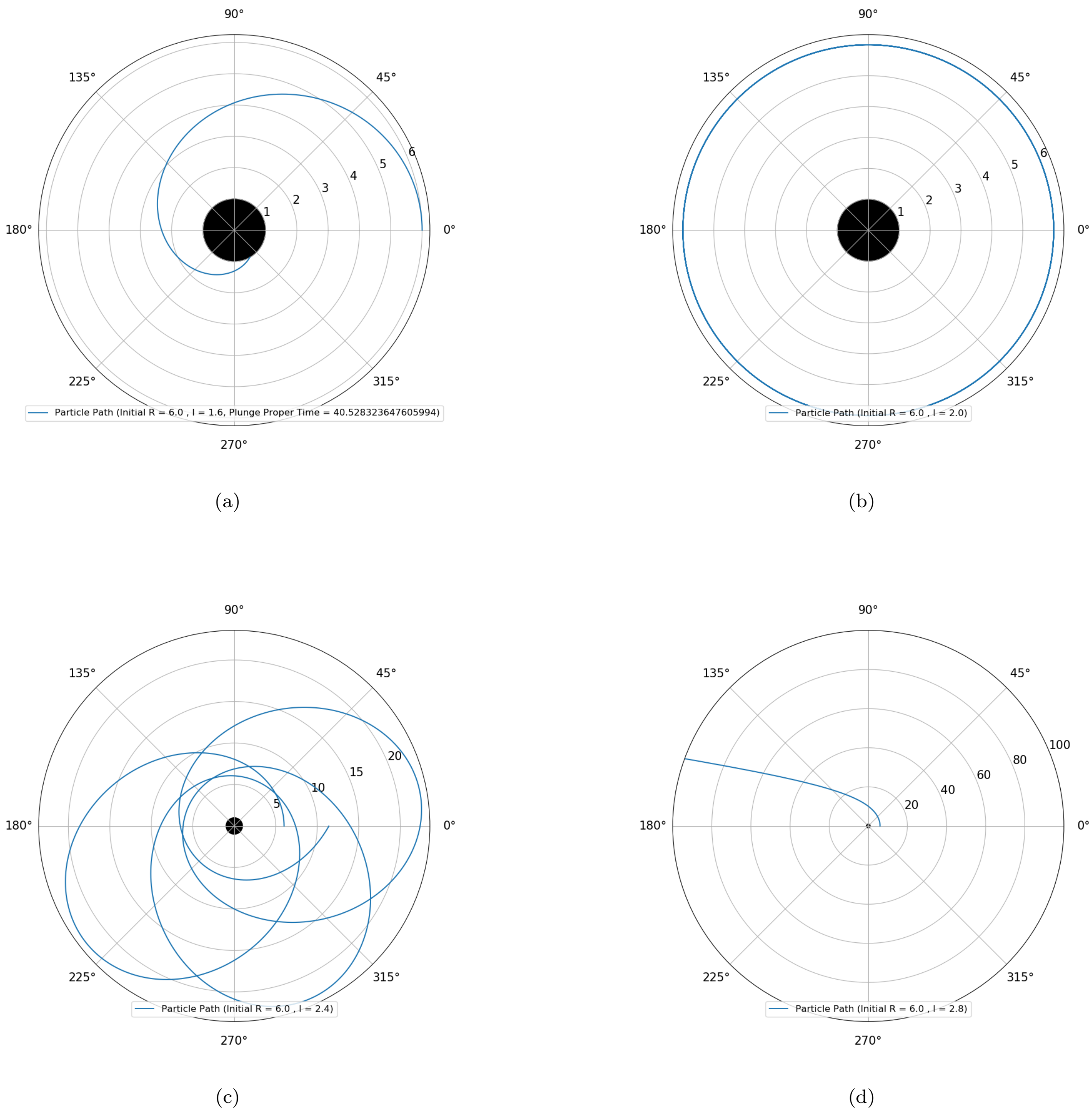


Figure 4: Polar plots of the motion of a particle around a black hole in (r, ϕ) space for several values of the angular momentum of the particle l, where the particle is initially at a radius of $R = 6$. The orbits shown have angular momenta values of (a): $l = 1.6$; (b): $l = 2.0$; (c): $l = 2.4$; (d): $l = 2.8$. The orbits of the particles were plotted for ϕ co-ordinates between $\phi = 0$ and $\phi = 8\pi$.

that radial plunge orbits occurred for angular momenta between $l = 0.00001$ and $l = 1.78884 \pm 0.00001$. It was noted that the proper time taken for the particle to plunge to the Schwarzschild radius increased as the angular momentum of the particle increased with a plunge time of $\tau = 26.16185 \pm (4 \times 10^{-7})$ for an angular momentum of $l = 1.00001$, while for an angular momentum of $l = 1.78884$, a plunge time of $\tau = 113.61728 \pm (4 \times 10^{-7})$ was determined.

Beyond these radial plunge orbits, precessional orbits were observed for angular momentum greater than $l = 1.78884 \pm 0.00001$, which more closely resembled a closed circular orbit as the angular momentum increased with a circular orbit achieved for an angular momentum of $l = 2.0$, as shown in Fig. 4b. As the angular momentum increased beyond $l = 2.0$, orbits where precession of the perihelion had occurred were observed (Fig. 4c) until the particle achieved an angular momentum allowing for the particle to escape the orbit of the black hole, where the approximate critical angular momentum for escape to infinity was taken to be $l = 2.8$ with each angular momentum value beyond this also resulting in the particle escaping the orbit of the black hole. An example of such an particle escape is shown in Fig. 4d.

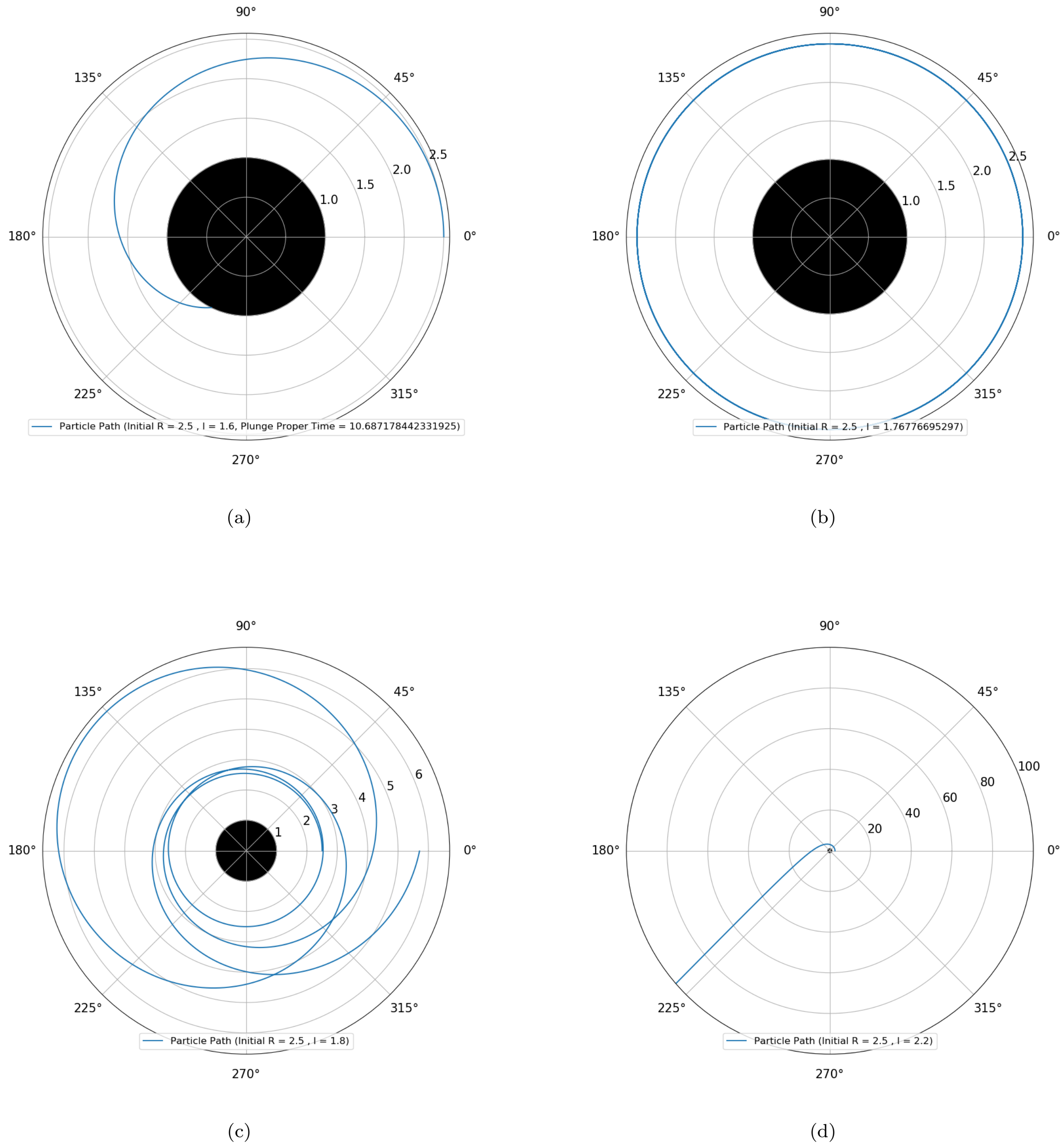


Figure 5: Polar plots of the motion of a particle around a black hole in (r, ϕ) space for several values of the angular momentum of the particle l, where the particle is initially at a radius of $R = 2.5$. The orbits shown have angular momenta values of (a): $l = 1.6$; (b): $l = \frac{5\sqrt{2}}{4}$; (c): $l = 1.8$; (d): $l = 2.2$. The orbits of the particles were plotted for ϕ co-ordinates between $\phi = 0$ and $\phi = 8\pi$.

4.2 Analysis of Orbits at R = 2.5

Analysis of orbits for an initial radius of $R = 2.5$ revealed that angular momenta values from $l = 0$ to $l = 1.6$ resulted in the particles plunging towards the centre of attraction, as can be seen from Fig. 5a. Upon further analysis by determining the instances in which radial plunge orbits occurred while incrementing the value of the angular momentum by $\delta l = 0.00001$, it was found that particles with angular momenta between $l = 0.00001$ and $l = 1.76775 \pm 0.00001$ plunged towards the Schwarzschild radius of the black hole. The proper time required for a radial plunge to occur increased as the angular momentum of the particle increased. This can be seen from Fig. 3b, where the particle plunge time was shown to increase with respect to the angular momentum, consistent with the $R = 6$ case.

As the particle's angular momentum increased, it was found that a circular orbit of the particle could be achieved for an angular momentum of $l = 1.767767 \approx \frac{5\sqrt{2}}{4}$, as determined by Eq. 48 for the instance of $\frac{dr}{d\phi}$ and $\frac{d^2r}{d\phi^2}$ being zero. As the angular momentum was increased further, the particle underwent precessional orbits around the black hole, as seen in Fig. 5c. For an angular momen-

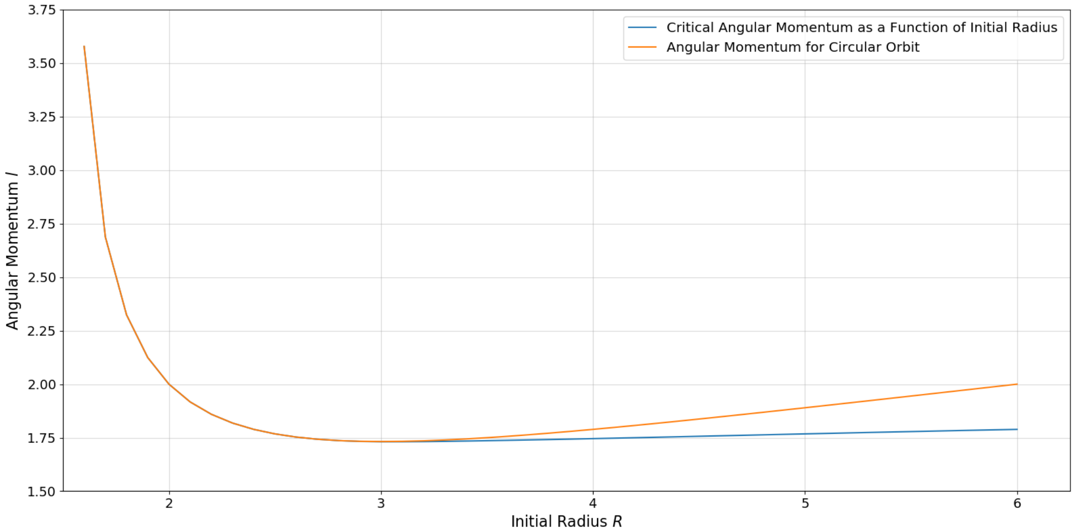


Figure 6: Plots of the critical angular momentum for radial plunge orbits (Blue) and angular momentum for circular orbits (Orange) as a function of the initial radius of the particle. At $R = 6$, the critical angular momentum and angular momentum for a circular orbit were observed to be $l = 1.78884 \pm 0.00001$ and $l = 2.0$, respectively, while for an initial radius $R = 2.5$, the angular momentum values were $l = 1.76775 \pm 0.00001$ and $l = 1.767767$, respectively.

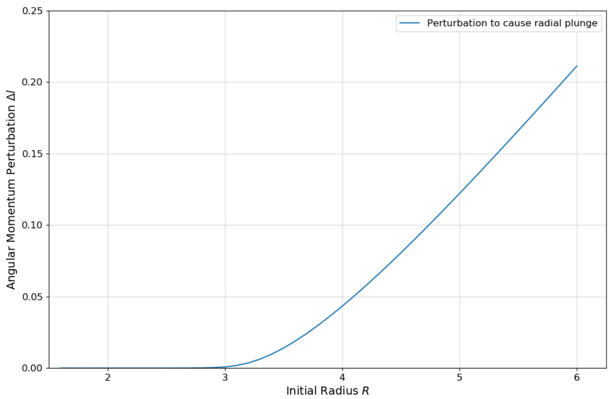


Figure 7: Plot of the perturbation to the angular momentum required to cause a circularly orbiting particle to plunge into the centre of attraction. For an initial radius of $R = 6$, the perturbation required to cause a particle to undergo a radial plunge from a circular orbit was calculated to be $\Delta l = 0.21116 \pm 0.00001$, while for an initial radius $R = 2.5$, the perturbation required was $\Delta l = (1.7 \pm 1) \times 10^{-5}$.

tum value of $l = 2.2$, it was observed that the particle, initially in orbit around the black hole at $R = 2.5$, was able to escape the orbit of the black hole and move outwards towards infinity (Fig. 5d). This was also observed for subsequently larger angular momenta, indicating that $l = 2.2$ can be considered to be an approximate critical angular momentum allowing for a particle's escape to infinity from an initial radial position of $R = 2.5$.

4.3 Stability of Circular Orbits at Various Radii

From the plot of angular momenta required to maintain a circular orbit as a function of the initial radius of a particle around a black hole, it was observed that as the initial radius of the particle approached $R = 1.5$, the angular momentum required for a particle to maintain a circular orbit approached infinity, as can be seen from Fig. 6. This was in agreement with the analytical solutions for angular momenta allowing for circular orbits, where the angular momentum values approach infinity for the limit of the radii approaching $R = 1.5$. For radii less than $R = 1.5$, the angular momenta required for circular orbits take imaginary values.

The critical angular momentum at which a particle will just plunge into the centre of attraction as a function of initial radius was shown to evolve in much the same way as that of the angular momentum required to maintain a circular orbit, as can be seen from Fig. 6. It can be seen that the angular momentum for each instance were largely the same for radii between $R = 1.6$ and $R = 3.0$, after which the angular momentum increased linearly with respect to the initial radius, albeit at different rates. This was reflected in the calculated perturbation to the angular momentum required to cause a particle in a circular orbit around the black hole to spiral into the centre of attraction. For initial radii of $R = 6$ and $R = 2.5$, this perturbation was determined to be $\Delta l = 0.21116 \pm 0.00001$ and $\Delta l = (1.7 \pm 1) \times 10^{-5}$, respectively.

The stability of the circular orbits of particles around the black hole can be seen from Fig. 7, where the per-

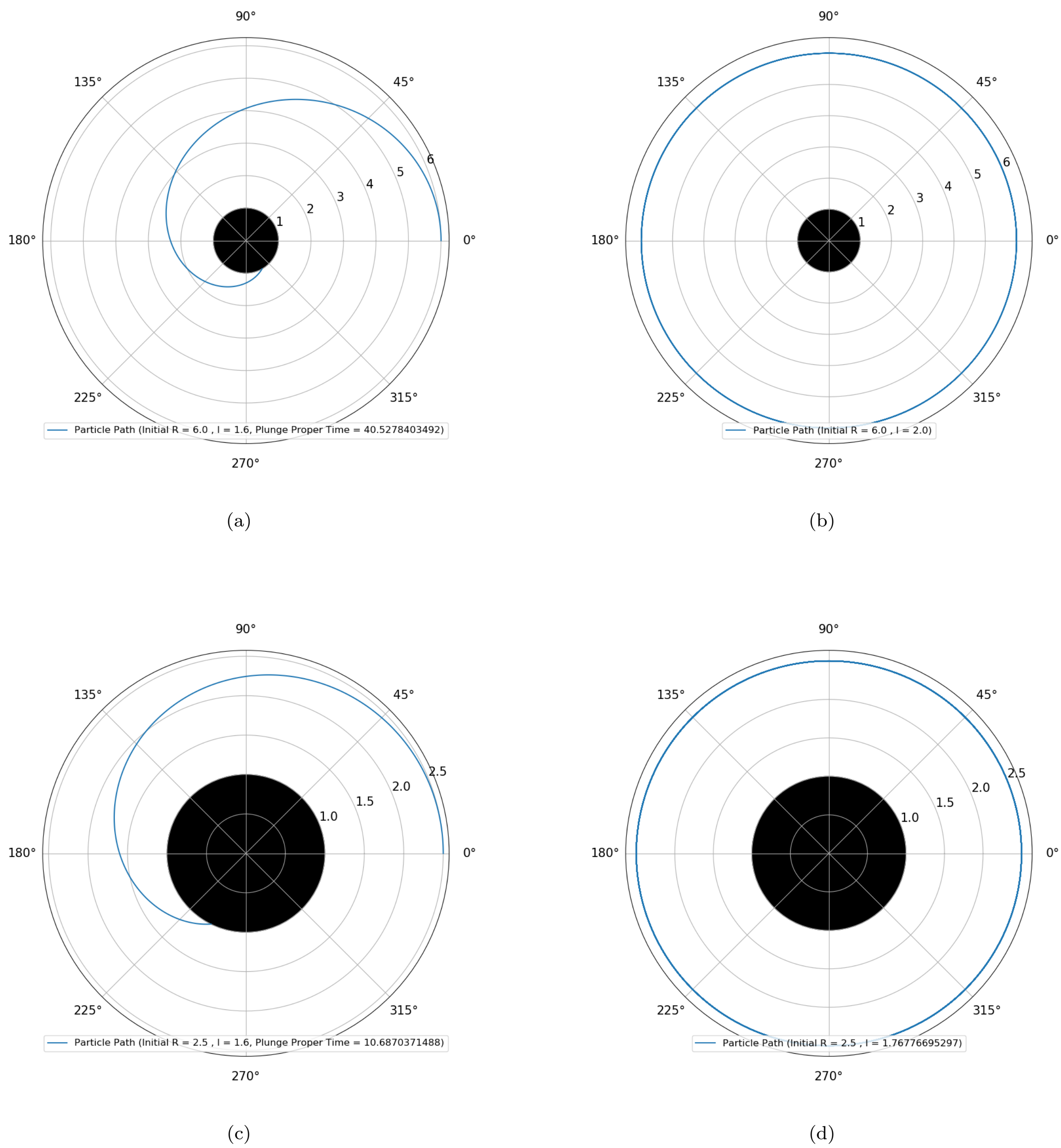


Figure 8: Polar plots of the motion of a particle around a black hole in (r, ϕ) space for several values of the angular momentum of the particle l, where the initial radius and angular momentum of the particle were (a): $R = 6$, $l = 1.6$; (b): $R = 6$, $l = 2.0$; (c): $R = 2.5$, $l = 1.6$; (d): $R = 2.5$, $l = \frac{5\sqrt{2}}{4}$. The orbits of the particles were plotted for co-ordinate times between $t = 0$ and $t = 1000$.

turbation to the angular momentum required to cause a radial plunge orbit from a circular orbit was plotted as a function of the initial radial position of the particle. For initial radii greater than $R = 3$, it was observed that the perturbation required increased linearly with respect to the initial radial position of the particle. However, as the initial radius decreased for $R = 3$, where $R = 3$ was a turning point in the plot, the perturbation required for a radial plunge decreased less sharply and asymptotically approached zero as the radius of the particle approached $R = 1.5$.

4.4 Analysis of Orbits via $r(t)$ and $\phi(t)$

When plotting the orbits of particles via Eqs. 52 and 59 for co-ordinate times between $t = 0$ and $t = 1000$, it was observed that radial plunge orbits occurred for particles with angular momentum $l = 1.6$ and initial radii at $R = 6$ and $R = 2.5$, where the proper time taken for the particles to cross the Schwarzschild radius was calculated as $\tau = 40.52784$ and $\tau = 10.68704$ respectively, as can be seen from Figs. 8a and 8c. For an angular momentum of $l = \frac{5\sqrt{2}}{4}$, it was observed that, for a particle with an initial radial position of $R = 2.5$, the particle

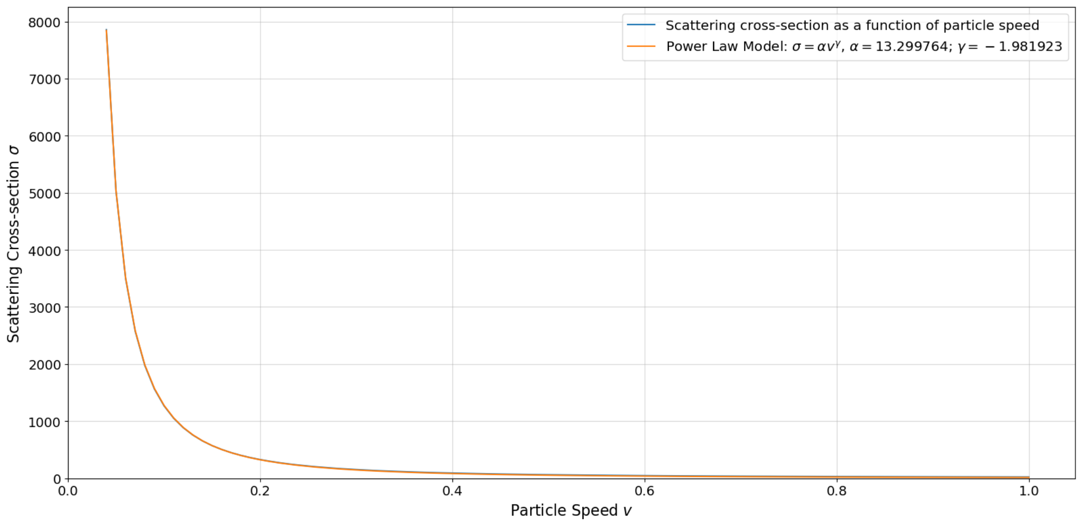


Figure 9: Plot of the scattering cross-section of particles around a black hole as a function of the speed of the particle (blue), where the cross-section was calculated via $\sigma = \pi b_{crit}^2$. The cross-section was plotted for particle speeds between 0.04 and 1.00. The orange plot shows a power law model fit, of the form $y = \alpha x^{\gamma}$, applied to the scattering cross-section values obtained via the RK4 method, where the parameters of the fit, where $y = \sigma$ and $x = v$, were $\alpha = 13.30 \pm 0.09$ and $\gamma = -1.982 \pm 0.002$.

maintained a circular orbit around the black hole, while for an increased angular momentum of $l = 2.0$, the particle maintained a circular orbit for an initial radius of $R = 6$ (Figs. 8b and 8d).

The critical angular momenta found for an initial radius of $R = 6$, via this method, was $l = 1.78885 \pm 0.00001$, while for an initial radius of $R = 2.5$, the critical angular momentum was found to be $l = 1.76776 \pm 0.00001$.

4.5 Particle Scattering

For the instances of a particle infalling towards a black hole from $R = 1000$, it was observed that for particle with initial speeds $v \ll 1$ that the scattering cross-section of the black hole increased with decreasing particle speed, where the cross-section increased from $\sigma = 151.1304 \pm 0.0001$ for a speed of $v = 0.3$ to a cross-section of $\sigma = 7857.124 \pm 0.0001$ for a particle speed of $v = 0.04$, as can be seen from Fig. 9. This behaviour was indicative of the scattering cross-section of the black hole approaching infinity as the particle speed approaches zero.

As the particle speed was increased, the scattering cross-section of the black hole declined at a much slower rate with a decrease in the cross-section of $\delta\sigma = 130.0732 \pm 0.0001$ as the speed of the particle increased from $v = 0.3$ to $v = 1$, i.e. where the particle was a photon. It was observed that the cross-section displayed asymptotic behaviour as the particle speed approached that of light, where a scattering cross-section of $\sigma = 21.2372 \pm 0.0001$ was calculated for a speed of $v = 1$.

4.6 Photon Scattering

From Fig. 10, it can be seen that as the impact parameter was increased from an initial value of $b = 10$ to a value of $b = 300$, in increments of $b = 1$, the deflection angles calculated via the Runge-Kutta fourth order method were initially divergent from those found from the large impact parameter limit for the deflection angle expected from theory. As the impact parameter was increased, the RK4 deflection angles were shown to become more consistent with the theoretical deflection angles, with the deflection angles being approximately equal for $b \approx 50$. As the impact parameter increased beyond this, the results for the deflection angle once again diverged from each other, with the RK4 method showing an approximate zero value for the deflection angle for an impact parameter of $b \approx 300$, while the theoretical large impact parameter limit will only approach zero as the impact parameter approaches infinity.

For an impact parameter of $b = 3.2$, the photon was determined to scatter though an angle of $\delta\phi = (1.40912 \pm 0.00001)$ rad from its original path as the photon orbited the black hole, as in Fig. 11a. The deflection angle of the photon was shown to increase as the photon was orbiting the black hole for the lesser impact parameter of $b = 2.8$ (Fig. 11b), where the increased deflection angle was calculated to be $\delta\phi = (2.29887 \pm 0.00001)$ rad. This increased angle of deflection indicated that further decreases in the impact parameter would result in a photon undergoing sufficient scattering such that it would be directed back towards its initial position. This was observed for an impact parameter of $b = 2.65$, where the photon incoming from $R = 100000$ was deflected by an angle of $\delta\phi = (3.55739 \pm 0.00001)$ rad as

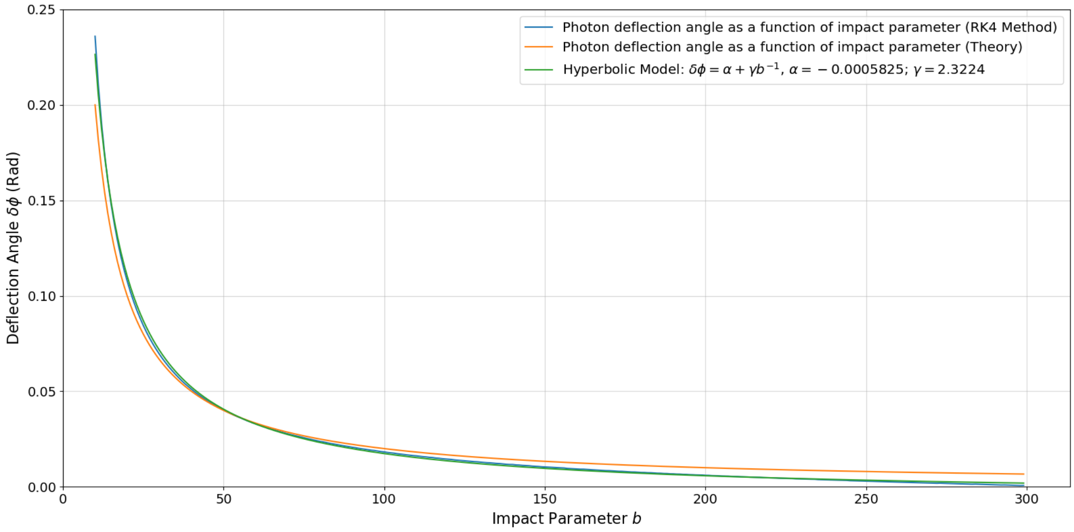


Figure 10: Plots of the deflection angle through which a photon is scattered by a black hole as a function of the impact parameter. The blue plot shows the deflection angle calculated via the Runge Kutta 4^{th} Order method and the orange curve shows the deflection angle obtained from the large impact parameter limit of the analytical solution for the deflection angle (Eq. 25). The green plot shows a hyperbolic model fit, of the form $y = \alpha + \frac{\gamma}{x}$, applied to the deflection angles obtained via the RK4 method. The parameters of the fit, where $y = \delta\phi$ and $x = b$, were found to be $\alpha = -0.000582 \pm 0.00008$ and $\gamma = 2.322 \pm 0.005$

it was scattered by the black hole.

However, as the impact parameter was decreased to 2.5, the photon plunged towards the Schwarzschild radius as it travelled around the black hole, where the photon underwent sufficient scattering such that the photon spiralled into the centre of attraction. This was expected from Sect. 4.5, where the critical impact parameter for particles with a speed of $v = 1$, i.e. for photons, was found to be $b = 2.6 \pm 0.01$.

5 Considerations for the Python Code

5.1 Conditions for Code:

The python codes utilised throughout this project worked for the majority of condition which were applied to them. However, there were instances in which the code was unable to carry out the numerical integration as required, largely due to singularities in the equations to be integrated. For the code for Questions 2, 3, and 4 (Sect. 3.1, 3.2, and 3.3), the code produced suitable results where a higher number of steps resulted in a higher accuracy of the results for initial radii greater than $R = 1$. However, not all angular momentum values were capable of being studied by the code, specifically the case of $l = 0$. It must be noted though that this case is heavily studied in theory [3, 4, 2], where a particle with a zero-value angular momentum will undergo a radial plunge orbit from any initial radial position such that the case does not be explicitly studied. In any case, the results expected for a zero valued angular momentum were extrapolated from the results obtained for angular momenta approaching zero. It must also be noted that the section of code that calculates the angular momentum required for a circular orbit to be maintained, for a given initial radius, was unable to do so for radii less than or equal $R \leq 1.5$, due to either an imaginary number being produced or a singularity in the value for the value of the angular momentum.

The code for Question 5, i.e. where the shape of particle orbits were determined from $r(t)$ and $\phi(t)$ (Sect. 3.4), did not have any dependencies on the angular momentum to plot the particle orbits. However, in order to proceed with the numerical integration to determine the proper time, the angular momentum was required to take a non-zero value. The code also has two other instances for which singularities occurred, the first being the case where the initial radius took a value of $R = 0$ and when the initial radius was equal to the Schwarzschild radius, i.e. for $R = r_s = 1$. However as these regions are not of interest, the initial radius should always be greater than the Schwarzschild radius.

For Question 6, where the scattering cross-section as a function of the particle speed was calculated (Sect. 3.5),the code largely worked for most values for the initial radius, impact parameter, and particle speeds. However, there were three cases for which singularities occurred such that the numerical integration could not take place: (i) $R = 0$; (ii) $b = 0$; (iii) $v = 0$. Similar cases were observed for Question 7, where the deflection angles of photons around a black hole were determined

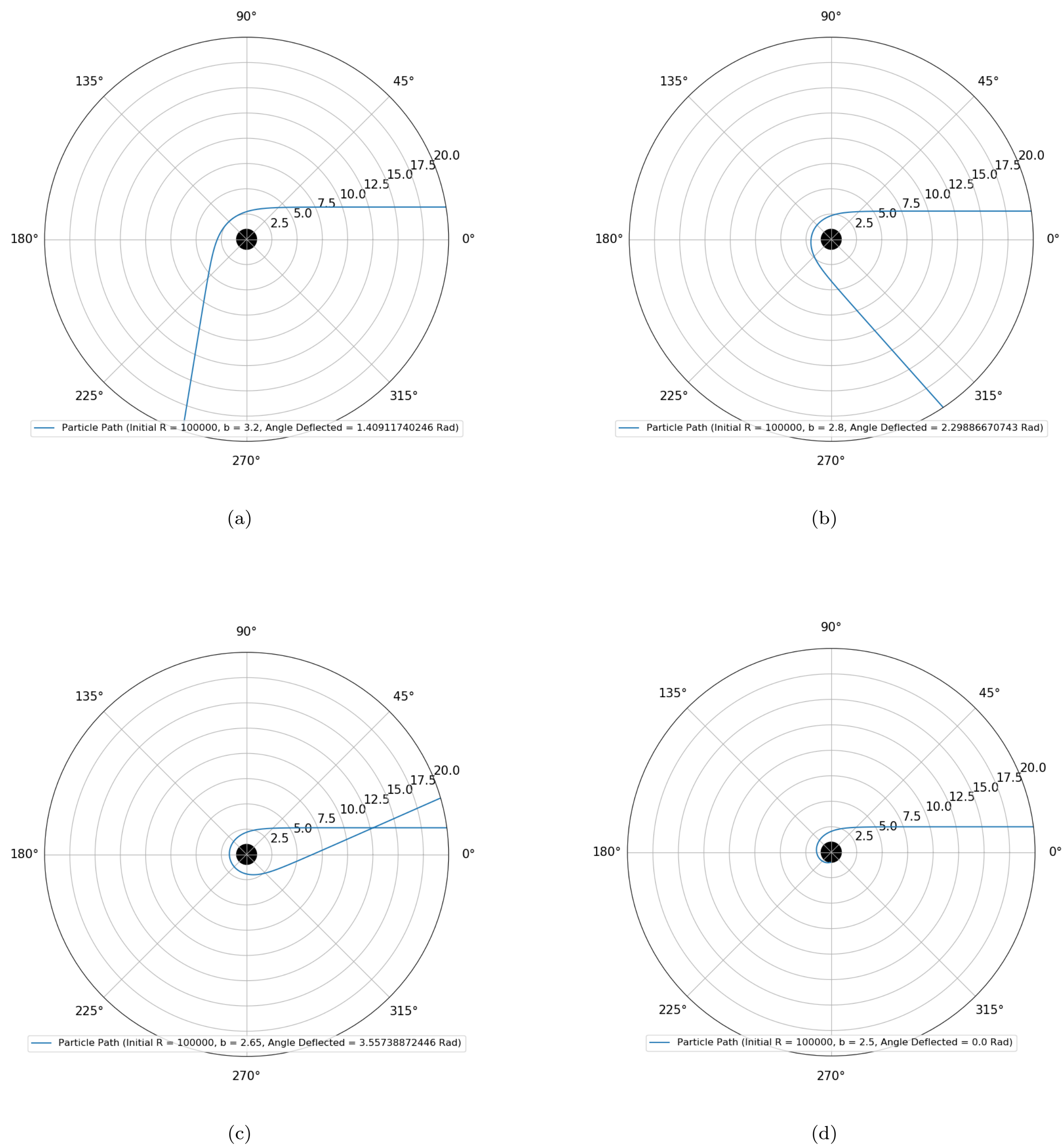


Figure 11: Polar plots of the motion of a photon around a black hole in (r,ϕ) space for several values of the impact parameter b, where the initial radial position of the photon was $R = 100000$ and the impact parameters were (a): $b = 3.2$; (b): $b = 2.8$; (c): $b = 2.65$; (d): $b = 2.5$.

(Sect. 3.6). As photons were being considered, the particle speed was set to $v = 1$, such that the only cases for which the code was unable to proceed with the numerical integration were an initial radius $R = 0$ and an impact parameter $b = 0$.

5.2 Accuracy of the Numerical Integration Method

The Runge-Kutta fourth order method derives its basis from the Euler method of numerical integration, for which one calculation is carried out per step of the numerical integration [1]. As such, the Euler method is said to have a first order truncation error, i.e. an error $\mathcal{O}(h)$, where h is the interval in the quantity over which the numerical integration is taking place [1]. By considering the case for the Euler method, a similar logic can be applied to determine the order of the error in the Runge-Kutta fourth order method. The fourth order Runge-Kutta method performs four calculations per step of the numerical integration such that the truncation error for the method is of the fourth order, i.e. the error is $\mathcal{O}(h^4)$ [1]. The intervals between steps for these methods are inversely proportional to the step size such that the error for the Runge Kutta Fourth can also be considered to be proportional to $\mathcal{O}(N^{-4})$, where N is the number of steps for the numerical integration [1].

Accordingly, the accuracy of the quantities found via the Runge-Kutta numerical integration method have an error $\mathcal{O}(h^4) \propto \mathcal{O}(N^{-4})$.

5.3 Accuracy of Critical Values

The accuracy of the critical values observed in the results, for example the critical angular momenta, was largely dependent on the nature of the arrays for the values being tested. As the arrays were created based on an initial value, a final value, and an interval between values, the accuracy of the critical values observed was dependent on the chosen interval size between each subsequent value in the array. For Questions 2 and 3 (Sects. 3.1 and 3.2), the intervals between subsequent angular momentum values, to test whether the particle had undergone a radial plunge orbit, were $\delta l = 0.001$ such that the critical angular momentum value observed were accurate to an order of $\mathcal{O}(\delta l)$.

For Question 4, where the perturbation to the angular momentum required such that a particle in a circular orbit underwent a radial plunge was investigated, the interval over which the angular momentum values were integrated was $\delta l = 0.00001$. As such, the accuracy of the critical perturbation and the critical angular momentum values were given by $\mathcal{O}(\delta l)$. For Question 5, the accuracy of the critical angular momenta was the same as in Question 2 and 3, where the accuracy of the critical momenta was $\mathcal{O}(\delta l)$. In Question 6, the critical values which were to be determined were in relation to the particle speed, impact parameter, and scattering cross-section. The accuracy of the particle speed and impact parameter were $\mathcal{O}(\delta v)$ and $\mathcal{O}(\delta b)$, where $\delta b = \delta v = 0.01$, while the accuracy of the scattering cross-section was $\mathcal{O}((\delta b)^2)$. For Question 7, the error in the deflection angle was of the first order, given that the ϕ co-ordinate was the parameter over which the bounds of the numerical were set. As such, the error for the deflection angle was $\mathcal{O}(h)$.

5.4 Theoretical Run-times and Accuracy of Python Scripts

For Questions 2 and 3, the theoretical run-time for an initial radius of $R = 6.0$ was determined to be $t_{run} \approx 283.5$s, for 1000 steps of the numerical integration and an interval between angular momentum values of $\delta l = 0.00001$ where the numerical integration took place for $0 \leq \phi \leq 8\pi$. In the case of the initial radius being $R = 2.5$, the run time was found to be $t_{run} \approx 229.6$s. For a greater step size of $N = 10000$ and smaller angular momentum interval, $\delta l = 0.0001$, the run time was noted to be $t_{run} \approx 274.9$s, while for an initial radius of $R = 2.5$, the run time was given by $t_{run} = 219.5$s. Given that the difference between run times for both cases, the former option was chosen to increase the accuracy of the critical angular momentum while also providing a suitable accuracy for the quantities determined by the Runge-Kutta method.

For Question 4, where the stability of the circular orbits was investigated, the run time of the script was found to be $t_{run} = 1280$s for a step size of $N = 1000$, and initial radii, angular momentum and ϕ co-ordinates of $1.6 \leq R \leq 6.0$, $1 \leq l \leq 5$, and $0 \leq \phi \leq 8\pi$, where the initial radius and angular momentum intervals were given by $\delta r = 0.1$ and $\delta l = 0.0001$. The number of steps and intervals chosen to produce this run time provided results of a suitable accuracy to study the stability of circular orbits around the black hole.

For analysing the particle orbits via a second approach (Sect. 3.4), a run time of $t_{run} = 317.4$s for an initial radius of $R = 6.0$, while for an initial radius of $R = 2.5$, the run time was $t_{run} = 122.2$s, where the number of steps for the numerical integration was $N = 1000$, the angular momenta values were $1 \leq l \leq 5$ which increased in increments of $\delta l = 0.00001$, and the numerical integration took place for co-ordinate times $0 \leq t \leq 1000$. For a step size of $N = 10000$ and an angular momentum interval of $deltal = 0.0001$, the run times were $t_{run} = 423$s and $t_{run} = 219.7$s for initial radii of $R = 6.0$ and $R = 2.5$ respectively. As such, the case with the lower step size proved to be more suitable with a lower run time and higher accuracy for critical angular momenta values while still maintaining a suitable accuracy for the Runge Kutta method.

In Question 6 (Sect. 3.5), the run time of the code was $t_{run} = 3807.7$s, for 10000 steps, where the numerical integration was performed over $0 \leq \phi \leq 4\pi$, and intervals for particle speed and impact parameter of $\delta v = 0.01$ and $\delta b = 0.01$, for $0.03 \leq v \leq 1$ and $0.01b \leq 60$. These parameters produced results of a suitable accuracy through which the scattering cross-section was accurately analysed and a model fit successfully applied.

For the analysis of photon deflection angles for large impact parameters, the run time of the python script was found to be $t_{run} = 2709$s for $N = 1000000$ steps taken between $0 \leq \phi \leq 4\pi$, where the impact parameters were $10 \leq b \leq 300$ for an interval of $\delta b = 1$ and the initial radius of the photon was $R = 100000$. The results produced for these parameters were found to suitably accurate such that a model fit was successfully applied. A smaller step size resulted in such a rapid decrease in the initial radius of the photon, initially, that the test condition to determine if the photon had reached the Schwarzschild radius was triggered. As such, the step size of $N = 1000000$ was used for the determination of the photon deflection angles and is recommended for future use of the code.

For the determination of deflection angle for a single impact parameter, the run time was found to be, on average, $t_{run} \approx 15$s, where the number of steps for the numerical integration was $N = 1000000$ for $0 \leq \phi \leq 4\pi$ and an initial radius of the photon of $R = 100000$. As the error in the ϕ co-ordinates was $\mathcal{O}(h)$, the step size for the numerical integration produced reliably accurate results for the deflection angle of the photon. For numbers of steps below $N = 1000000$, it was observed that the decrease in radial position with respect to the

ϕ co-ordinate was so rapid that the test condition for the photon passing Schwarzschild radius was immediately triggered. As such, the number of steps for the numerical integration was required to be $N \geq 1000000$.

It must be noted that the run times for the python scripts did not account for the time taken for the user to save the individual plots and make changes if required.

6 Discussion

For an initial radius of $R = 6$, it was calculated, via the fourth order Runge-Kutta method of numerical integration, that the critical angular momentum for which the particle in orbit around the black hole will just plunge into the centre of attraction was $l = 1.78884 \pm 0.00001$. As such, particles with an initial radius of $R = 6$ will undergo radial plunge orbits for angular momenta between $l = 0$ and $l = 1.78884 \pm 0.00001$. The proper time taken for these radial plunge orbits to occur were shown to increase with increasing angular momentum, where a proper time of $\tau = 113.61728 \pm (4 \times 10^{-7})$ was observed before the particle reached the Schwarzschild radius for the critical angular momentum, while a lesser plunge time of $\tau = 26.16185 \pm (4 \times 10^{-7})$ was calculated for an angular momentum of $l = 1.00001$. For a particle initially in orbit around a black hole at $R = 6$, the angular momentum required to maintain a circular orbit at this radial position was calculated to be $l = 2.0$, a value consistent with the analytical solution obtained from Eq. 20. Accordingly, the minimum perturbation to the angular momentum required to disrupt a circularly orbiting particle such that it plunges into the centre of attraction was $\Delta l = 0.21116 \pm 0.00001$, or a decrease from the angular momentum for a circular orbit of 10.558%. This indicated that the circular orbits of particles at $R = 6$ were stable circular orbits, consistent with both theory and the radii for circular orbits for a given angular momentum, determined from Eq. 19.

For particles with initial radii at $R = 2.5$, the critical angular momentum, for which the particle will undergo a radial plunge, was found to be $l = 1.76775 \pm 0.00001$, such that radial plunges of particles from $R = 2.5$ occurred for angular momentum values between $l = 0$ and $l = 1.76775 \pm 0.00001$. As with the $R = 6$ case, the proper time taken for a particle to plunge to the Schwarzschild radius was observed to increase with the angular momentum of the particle, with a proper time of $\tau = 6.37202 \pm (4 \times^1 0{-}7)$ for $l = 1.00001$, while the proper plunge time for $l = 1.76775$ was calculated as $\tau = 74.08312 \pm (4 \times 10^{-7})$. In order for a particle to maintain a circular orbit at $R = 2.5$, it was determined that an angular momentum of $l = \frac{5\sqrt{2}}{4}$ was required, such that the perturbation required to cause a radial plunge from a circular orbit was $\Delta l = (1.7 \pm 1) \times 10^{-5}$. This perturbation constituted a change in angular momentum from that of a circular orbit of 9.59×10^{-4}%, indicating that the circular orbit was an unstable circular orbit, confirmed both from theoretical results for Schwarzschild geometry and the radii for circular orbits produced by Eq. 19.

As can be seen from analysis of orbits of particles at $R = 6$ and $R = 2.5$, the perturbation of the angular momentum required to force a particle in a circular orbit to undergo a radial plunge was observed to decrease sharply. The evolution of the perturbation required for this occur as a function of the initial radius was more closely analysed, by calculating the critical angular momenta and angular momenta for circular orbits for initial radii between $R = 1.6$ and $R = 6.0$, in intervals of 0.1, and plotting the angular momenta values and resulting minimum perturbations as a function of the initial radius of the orbits. From Fig. 6, it was observed that the critical and circular orbit angular momenta values were observed to be approximately equal for initial radii between $R = 1.6$ and $R = 3$, after which the two angular momenta plots diverged from each other. This evolution was observed in Fig. 7, where the critical perturbation to cause a radial plunge from a circular orbit was observed to approach zero as the initial radius of the orbit approached $R = 1.5$, i.e. $\Delta l \to 0$ as $R \to 1.5$. As the initial radius increased beyond $R = 3$, the perturbation was shown to, at first, increase sharply before attaining a linearly proportional relationship with respect to the initial radius. The behaviour of the critical perturbation with relation to the initial radius of the particle indicated that $R = 3$ signified the radius where circular orbits of particles around the black hole changed from stable to unstable, where circular orbits with $R_{circ} < 3$ were unstable orbits and minor perturbations were required to cause radial plunge orbits while circular orbits for which $R_{circ} > 3$ were stable orbits with the stability of the orbit increasing with the initial radius of the particle. As such, the circular orbit of the particle $R = 2.5$ represented an unstable circular orbit, while the circular orbit at $R = 6$ was stable.

Further observations can be made from the plots of circular and critical angular momenta as a function of initial radius (Fig. 6), the first being that for a given angular momentum, say $l = 2.0$, there are two initial radii of a particle for which a circular orbit occurred: $R = 2$ and $R = 6$. By coupling this observation with those learned from examining the critical perturbation to cause a radial plunge, it can be determined that the $R = 2$ circular orbit was unstable and the $R = 6$ circular orbit was stable. The existence of two radii for which circular orbits can be achieved for a given angular momentum was consistent with what is expected from theory, specifically Eq. 19 where r_{min} is the radius for which a stable circular orbit occurs and r_{max} is the radius for which an unstable circular orbit is found. Upon further examination of Fig. 6, it can be seen that the only instance in which an angular momentum value resulted in a unique radius at which a circular orbit can occur was for $l = \sqrt{3}$ where the radius for a circular orbit was $R = 3$, i.e. the turning point of the plots in Fig. 6. This would appear to confirm that $R = 3$ signifies the radius of the innermost stable circular orbit, which was confirmed by examination of Eq. 19, where the term in the square root vanishes, and from what was expected from theory (Eq. 17) for a Schwarzschild radius of $r_s = 1$.

The approach of the angular momentum required to sustain a circular orbit to infinity as the initial radius of the particle approached $R = 1.5$ also confirmed the radius of the photon sphere, i.e. the radius at which only photons can obtain circular orbits and inside of which circular orbits of particles are not longer possible. The radius of the photon sphere determined from the results was in agreement with what is expected from theory (Eqs. 20 and 18), for the case in which the Schwarzschild radius of the attractive body is $r_s = 1$.

By examining the case in which the shape of orbits of particles around a black were determined by obtaining $r(t)$ and $\phi(t)$, it was determined that the $r(\phi)$ approach to examining the particle orbits was a more simple, elegant, and efficient method of analysis. The $r(t)$ and $\phi(t)$ approach was shown to be more complex, largely, as a result of the fact that in order to obtain the shape of the particle orbits, a pair of coupled equation must be numerically integrated, such that $\phi(t)$ could not be obtained without determining $r(t)$. This was not the case for the for the $r(\phi)$ approach, where a r co-ordinate was found for each ϕ co-ordinate to be considered rather than separately determining a r and a ϕ co-ordinate for every t co-ordinate over which the numerical integration. The $r(\phi)$ approach was also shown to be more efficient than the $r(t)$ and $\phi(t)$ method, an observation that was closely linked with the complexity of the two methods. It was observed that in order to obtain the shapes of the orbits, the $r(t)$ and $\phi(t)$ method had to numerically integrate Eqs. 55, 56 and 59 rather than Eqs. 87 and 86 for the $r(\phi)$ method, such that more computations took place per iteration of the numerical integration. As such, in order to obtain results of the same accuracy as the $r(\phi)$ method, the code for the $r(t)$ and $\phi(t)$ method required a longer total run time. However, it must be noted that the results produced via both methods were largely the same for the same number of steps in the integration, where any discrepancies were likely due to the fact that the integration was over different co-ordinates. The $r(t)$ and $\phi(t)$ method was considered inelegant when compared to the $r(\phi)$, due to the presence of the derivative of ϕ with respect to t in Eq. 56, such that in each iteration for the numerical integration, the ϕ derivative had to be calculated separately and inputted back into the equation to be integrated, thus making the numerical integration more computationally intensive. The presence of the ϕ derivative in Eq. 56 was due to the fact that the total energy did not conveniently factor out, as it did with the $r(\phi)$ method, such that a number of reparameterisations were required to obtain expressions that were independent of the total energy E which could then be numerically integrated.

There are some instances in which the $r(t)$ and $\phi(t)$ method would offer a more preferred approach to a specific scenario. For instance, the $r(\phi)$ approach is convenient in the sense that the motion of a particle around a black hole can be numerically integrated over a number of orbits, say four complete orbits (for $0 \leq \phi \leq 8\pi$). However, the $r(t)$ and $\phi(t)$ method would prove a more useful method in cases where the evolution of a particle over a specific co-ordinate time is to be examined.

The scattering cross-section of the black hole was determined, for a range of particle speeds between $v = 0$ and $v = 1$, by finding the impact parameter for which a radially infalling particle will just undergo a radial plunge towards the centre of attraction. By examining the evolution of the scattering cross-section of the black hole as a function of the particle speed (Fig. 9), several observations were made. Firstly, it was observed that in the $v \ll 1$ case that as the speed of the particle being scattered approached zero, the scattering cross-section asymptotically approached infinity, i.e. as $v \to 0$, $\sigma \to \infty$. This was a computational confirmation of the theoretical case in which a stationary particle initially at rest at a radial position at infinity will always undergo a radial plunge orbit, in Schwarzschild geometry. The second observation was in regard to the evolution as the particle speed increased towards the limit of $v = 1$. As the particle speed increased towards $v = 0.3$, it was shown that the scattering cross-section decreased sharply, after which the scattering cross-section appears to approach zero as the particle speed approaches one ($\sigma \to 0$ as $v \to 1$). However, this was not the case. The scattering cross-section attained a minimal value of $\sigma = 21.2372 \pm 0.0001$ at $v = 1$, where $b = 2.6 \pm 0.01$. As the speed of the particle can not approach infinity, there is, in fact, a critical impact parameter for which photons will spiral in towards the centre of attraction of a black hole. The behaviour shown by the plot of the scattering cross-section as a function of a particle speed was that of a power law, where the parameter of the power law model were given by $\alpha = 13.30 \pm 0.09$ and $\gamma = -1.982 \pm 0.002$.

The angle of deflection experienced by photons as they radially infall, from $R = 100000$, toward the centre of attraction of a black hole was also examined (Fig: 10). It appeared that as the impact parameter approached zero that the angle through which the photon was scattered approached infinity. However, this was not possible as for impact parameters less than $b = 1$, the photon would pass through the Schwarzschild radius such that escape from the gravitational attraction of the black hole would no longer be possible. In fact, for impact parameters less than $b = 2.6 \pm 0.01$, photons underwent radial plunge orbits. The relationship observed between the deflection angle of the photon and the impact parameter was shown to be consistent with a hyperbolic fit of the form:

$$y = \alpha + \frac{\gamma}{x} \tag{97}$$

where y and x were the deflection angle and impact parameter respectively.

By applying a model fit via the python code, it was determined that the parameters α and γ were given by $\alpha = -0.000582 \pm 0.00008$ and $\gamma = 2.322 \pm 0.004$.

The deflection angles obtained from the RK4 method or the hyperbolic model fit applied to the data were observed to have deviations from the theoretical values obtained from the large impact parameter limit for the deflection angle (Eq. 25), It was observed that as the impact parameter increased from $b = 10$ to $b = 50$ that the theoretical and computed values for the deflection angle more closely resembled each other. However, as the impact parameter parameter increased beyond $b = 50$, it was shown that the theoretical approximation for the deflection angles diverged from those calculated via the RK4 method.

For the specific cases for which $b = 3.2$, $b = 2.8$, $b = 2.65$, and $b = 2.5$, it was observed that for a decreasing impact parameter b, the angle through which the particle was deflected increased from $\delta\phi = (1.40912 \pm 0.00001)$ rad, for $b = 3.2$, to $\delta\phi = (3.55739 \pm 0.00001)$ rad as the impact parameter decreased to $b = 2.65$. For an impact parameter of $b = 2.5$, the photon underwent a radial plunge orbit towards attraction. This was likely due to the photon being scattered to such an extent that the photon was directed towards black hole. This behaviour was consistent with what was observed from the plot of critical impact parameters as a function of particle speed, where the critical impact parameter for $v = 1$, i.e. for a photon, was found to be $b = 2.6 \pm 0.01$. As such, it can be determined that for impact parameters less than $b = 2.6 \pm 0.01$, photons will be deflected from their original path to such an extent that they are captured by the black hole.

7 Conclusion

To conclude, the analysis of the motion of particles in orbit around a black hole through numerical integration via the Runge-Kutta fourth order method yielded results that were largely consistent with the results observed from analytical solutions, for the Schwarzschild metric. The python code used allowed for the plotting of the particle orbits for a given initial radius and angular momentum, the determination of the range of angular momenta for which the orbit would be a radial plunge orbits, and the calculation of the proper time taken for a particle to cross the Schwarzschild radius for two separate approaches to the numerical integration. It was observed that the stability of circular orbits decreased with decreasing initial radius, where unstable orbits occurred for initial radii less than $R = 3$ and stable orbits for $R = 3$. It was also confirmed that, in accordance with analytical solutions, that for a given angular momentum value, there were two radii for which a circular orbit of a particle could be achieved, one for which the orbit was stable and one of which would result in an unstable orbit, where the only exception occurred for $R = 3$, where a circular orbit occurred with a unique angular momentum. The scattering cross-section of the black hole as a function of particle speed was also produced by the code, where the scattering cross-section with respect to the particle speed was shown to be consistent with a power law. The determination of the deflection angles of photons as they were scattered by the black hole was the only instance in which the results found via numerical integration deviated from the analytical results in the large impact parameter solution. Aside from this, it was shown that as the impact parameter was decreased, the deflection angle increased until the impact parameter reached $b =$, after which the photons were scattered to such an extent that they were captured by the black hole.

References

[1] R.L. Burden and J.D. Faires. *Numerical Analysis.* Brooks/Cole, Cenage Learning, 2011.

[2] K.S. Thorne C.W. Misner and J.A. Wheeler. *Gravitation.* W.H. Freeman and Company, 1973.

[3] J.B. Harlte. *Gravity: An Introduction to Einstein's General Relativity.* Pearson Education, Inc., 2003.

[4] B. Schutz. *A First Course in General Relativity.* Cambridge University Press, 2009.

8 Appendix

8.1 Python Code

```
"""

Code for Questions 2 and 3 of Project 14.1: Particle or Photon Orbits near a Black Hole

Name: Luke Timmons
Student Number: 304757457

"""


#import libraries to be used
import PIL
from PIL import Image
import pandas as pd
from numpy import exp,arange
from pylab import meshgrid,cm,imshow,contour,clabel,colorbar,axis,title,show
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import math
import pylab
from lmfit import Model
import os

#defines function for rate of change of 'r' as function of 'phi'
def func_f(z):
        return(z)

#defines function for rate of change of 'dr/dphi' as a function of 'phi'
def func_g(tau,r,z,l):
        return((2*(z**2)/r) + r - (r**2)/(2*l**2) -1.5)

#defines function for rate of change of 'tau' as a function of 'phi'
def func_a(tau,r,z,l):
        return (r**2/l)


def func_k_0(tau_i,r_i,z_i,h):
        return (h*func_f(z_i))


def func_l_0(tau_i,r_i,z_i,h,l):
        return (h*func_g(tau_i,r_i,z_i,l))

def func_m_0(tau_i,r_i,z_i,h,l):
        return (h*func_a(tau_i,r_i,z_i,l))

#defines function to determine the angular momentum for a circular orbi
def func_circ_orbit(r_i):
        return (r_i**2/(2*(r_i-1.5)))



pi = np.pi
print(pi)

```

```
#define initial and final values for phi co-ordinate and the number of steps for the
    numerical integration
a=0
b=8*pi
N=1000
h=(b-a)/N


#intialises arrays for numerical integation
r_vals=[]
z_vals=[]
tau_vals=[]
plunge_t=[]
phi_vals=[]


#initial values for co-ordinates for numerical integration
r_init = 6.0
z_init = 0.0
tau_init =0.0
phi_init=0.0

#appends initial values to arrays for numerical integration
r_vals.append(r_init)
z_vals.append(z_init)
tau_vals.append(tau_init)
phi_vals.append(phi_init)


#creates an arrays for values of angular momentum
l_vals= np.arange(1.00001, 5.00001, 0.00001)


#initialises array for radial plunge angular momentum values
rad_plunge_l=[]

#boolean used for test to determine if a radial plunge orbit had occured
plunge = False

#sets value for angular momentum
l=2.0

#for loop in which the runge-kutta method is implemented
for i in range(N-1):
        tau=tau_vals[i]
        r=r_vals[i]
        z=z_vals[i]
        phi = phi_vals[i]

        k_0 = func_k_0(tau,r,z,h)
        l_0 = func_l_0(tau,r,z,h,l)
        m_0 = func_m_0(tau,r,z,h,l)



        r1 = r+0.5*k_0
        z1 = z+0.5*l_0
        tau1 = tau+0.5*m_0

        k_1 = func_k_0(tau1,r1,z1,h)
        l_1 = func_l_0(tau1,r1,z1,h,l)
```

```
        m_1 = func_m_0(tau1,r1,z1,h,l)


        r2 = r+0.5*k_1
        z2 = z+0.5*l_1
        tau2 = tau+0.5*m_1

        k_2 = func_k_0(tau2,r2,z2,h)
        l_2 = func_l_0(tau2,r2,z2,h,l)
        m_2 = func_m_0(tau2,r2,z2,h,l)


        r3 = r+k_2
        z3 = z+l_2
        tau3 = tau+m_2

        k_3 = func_k_0(tau3,r3,z3,h)
        l_3 = func_l_0(tau3,r3,z3,h,l)
        m_3 = func_m_0(tau3,r3,z3,h,l)


        #calculates the next values in the runge-kutta method
        r_new = r_vals[i] + (1/6)*(k_0+2*k_1+2*k_2+k_3)
        z_new = z_vals[i] + (1/6)*(l_0+2*l_1+2*l_2+l_3)
        tau_new = tau_vals[i] + (1/6)*(m_0+2*m_1+2*m_2+m_3)
        phi_new = phi_vals[i] + h

        #appends next values in runge-kutta method into arrays
        r_test = r_new
        r_vals.append(r_new)
        z_vals.append(z_new)
        tau_vals.append(tau_new)
        phi_vals.append(phi_new)

        #if statement to test if particle has plunged to schwarzschild radius
        if r_test <=1:
                print(tau_new)
                plunge_time = tau_new
                plunge=True
                #breaks out of loop ending runge-kutta integration
                break
        else:
                plunge = False
        #if statement to test if particle has effectively escaped to infinity
        if r_test>=1e6:
                break


#creates polar plot
ax = plt.subplot(111,projection='polar')


#adds circle to polar plot to represent black hole
circle= plt.Circle((0,0), radius= 1,color='black',transform=ax.transData._b)
ax.add_artist(circle)

#if statement that will add the proper time taken for a radial plunge orbit to occur to the legend of the plot
if (plunge==False):
        ax.plot(phi_vals, r_vals, label='Particle Path (Initial R = ' +str(r_init)+' , l = ' + str(l) + ')')
```

```
else:
        ax.plot(phi_vals, r_vals, label='Particle Path (Initial R = ' +str(r_init)+' , l = ' + str(l) + ', Plunge Proper Time = '+str(plunge_time)+')')

#adds legend to the plot
ax.legend(loc='lower center', fontsize='large')
#sets aspect of the plot
ax.set_aspect('equal')
ax.tick_params(labelsize=15)
#shows the polar plot
plt.show()


#for loop to iterate through each value for the angular momentum
for j in range(len(l_vals)):
        #initialies array for the r,z,tau, and phi values for the numerical integration
        r_vals=[]
        z_vals=[]
        tau_vals=[]
        phi_vals=[]

        plunge=False

        #appends the initial values for the quantites to the appropriate arrays
        r_vals.append(r_init)
        z_vals.append(z_init)
        tau_vals.append(tau_init)
        phi_vals.append(phi_init)#sets the value for the angular momentum

        #sets the value for the angular momentum
        l = l_vals[j]

        #for loop to carry out the runge-kutta numerical integation
        for i in range(1,N):
                #sets the values for the proper time, r co-ordinate, phi co-ordinate, and the rate of change of r wrt phi
                tau=tau_vals[i-1]
                r=r_vals[i-1]
                z=z_vals[i-1]
                phi=phi_vals[i-1]




                k_0 = func_k_0(tau,r,z,h)
                l_0 = func_l_0(tau,r,z,h,l)
                m_0 = func_m_0(tau,r,z,h,l)

                r1 = r+0.5*k_0
                z1 = z+0.5*l_0
                tau1 = tau+0.5*m_0

                k_1 = func_k_0(tau1,r1,z1,h)
                l_1 = func_l_0(tau1,r1,z1,h,l)
                m_1 = func_m_0(tau1,r1,z1,h,l)

                r2 = r+0.5*k_1
                z2 = z+0.5*l_1
                tau2 = tau+0.5*m_1

                k_2 = func_k_0(tau2,r2,z2,h)
```

```
        l_2 = func_l_0(tau2,r2,z2,h,l)
        m_2 = func_m_0(tau2,r2,z2,h,l)

        r3 = r+k_2
        z3 = z+l_2
        tau3 = tau+m_2

        k_3 = func_k_0(tau3,r3,z3,h)
        l_3 = func_l_0(tau3,r3,z3,h,l)
        m_3 = func_m_0(tau3,r3,z3,h,l)


        #calculates values for the quantities for next step of the runge kutta integration
        r_new = r_vals[i-1] + (1/6)*(k_0+2*k_1+2*k_2+k_3)
        z_new = z_vals[i-1] + (1/6)*(l_0+2*l_1+2*l_2+l_3)
        tau_new = tau_vals[i-1] + (1/6)*(m_0+2*m_1+2*m_2+m_3)
        phi_new = phi_vals[i-1] + h


        #appends the values to the appropriate arrays to be used in next iteration of the loop
        r_vals.append(r_new)
        z_vals.append(z_new)
        tau_vals.append(tau_new)
        phi_vals.append(phi_new)

        r_test = r_new

        #else if statement to test whether the particle passes the Schwarzschild radius
        if ((r_test <=1)):
            #appends angular momentum value for radial plunge to appropriate array
            rad_plunge_l.append(l)
            #appends proper time for radial plunge to occur to appropriate array
            plunge_t.append(tau_vals[i])
            #breaks the loop
            plunge = True
            break

    if plunge==False:
        break

#prints the bounds of the angular momentum values for which a radial plunge orbit will occur
print('l values for radial plunge orbit are between 0 and ' + str(rad_plunge_l[-1]))
#calculates the squared function to detemine the angular momentum for a circular orbit
circ_l_sq = (func_circ_orbit(r_init))
#tests to see if the value is negative such that the square root of the value will produce an imaginary number. if so, prints that circular orbit not possible for given radius
if (circ_l_sq <0):
    print('A particle cannot achieve a circular orbit at this radial position.')
elif (circ_l_sq>0):
    #calculates value of circular orbit angular momentum
    circ_l = np.sqrt(circ_l_sq)
    #prints value of angular momentum for circular orbit to occur
    print('For a circular orbit and l value of l = ' + str(circ_l) + ' is required.')
elif (r_init==1.5):
    print('Only photons can achieve a circular orbit for an initial radius of R = 1.5')

#creates a plot
```

```
ax = plt.subplot(111)

#plots the proper time taken for a radial plunge to occur as a function of the angular momentum
ax.plot(rad_plunge_l,plunge_t, label='Proper Time for Radial Plunge (R =' + str(r_init) + ')')




ax.set_xlabel(r'Angular Momentum $l$', fontsize=16)
ax.set_ylabel(r'Proper Time $\tau$', fontsize=16)
ax.tick_params(labelsize=14)

#adds a grid to the plot
ax.grid(True,alpha=0.5)
#adds a legend to the plot
ax.legend(loc='upper right', fontsize='large')
#shows the plot
plt.show()




#saves the radial plunge angular momenta and proper time for plunge to occur to csv file
dict = {'Radial Plunge l': rad_plunge_l, 'Proper Time': plunge_t}

df=pd.DataFrame(dict)

df.to_csv('radial_plunge_values_r='+str(r_init)+'.csv')
```

```
"""

Code for Question 4 of Project 14.1: Particle or Photon Orbits near a Black Hole

Name: Luke Timmons
Student Number: 304757457

"""

#imports libraries to be used
import PIL
from PIL import Image
import pandas as pd
from numpy import exp,arange
from pylab import meshgrid,cm,imshow,contour,clabel,colorbar,axis,title,show
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import math
import pylab
from lmfit import Model
import os

#defines function for rate of change of 'r' as function of 'phi'
def func_f(z):
        return(z)

#defines function for rate of change of 'dr/dphi' as a function of 'phi'
def func_g(tau,r,z,l):
        return((2*(z**2)/r) + r - (r**2)/(2*l**2) -1.5)
```

```

#defines function for rate of change of 'tau' as a function of 'phi'
def func_a(tau,r,z,l):
        return (r**2/l)


def func_k_0(tau_i,r_i,z_i,h):
        return (h*func_f(z_i))


def func_l_0(tau_i,r_i,z_i,h,l):
        return (h*func_g(tau_i,r_i,z_i,l))

def func_m_0(tau_i,r_i,z_i,h,l):
        return (h*func_a(tau_i,r_i,z_i,l))


#defines function to determine the angular momentum for a circular orbi
def func_circ_orbit(r_i):
        return (r_i**2/(2*(r_i-1.5)))



pi = np.pi
print(pi)

#define initial and final values for phi co-ordinate and the number of steps for the numerical integration
a=0
b=8*pi
N=1000
h=(b-a)/N


#intialises arrays for numerical integation
r_vals=[]
z_vals=[]
tau_vals=[]
phi_vals=[]




#initial values for co-ordinates for numerical integration
z_init = 0.0
tau_init =0.0
phi_init=0.0

#appends initial values to arrays for numerical integration
z_vals.append(z_init)
tau_vals.append(tau_init)
phi_vals.append(phi_init)


#creates an arrays for values of angular momentum
l_vals= np.arange(1.0001, 5.0001, 0.0001)

#creates an arrays for values of the initial radius of the orbit
r_init_vals = np.arange(1.6,6.1,0.1)

#initialises arrays for radial plunge angular momenta, perturbation of angular momentum,
```

```
    critical angular momenta and corresponding radii, circular orbit angular momenta and
    correspondind radii
var_l = []
l_crit = []
r_crit_vals = []
circ_l_vals=[]
circ_r_vals=[]


#for loop to iterate through initial radii of the particle orbits
for m in range(len(r_init_vals)):
        #sets
        r_init = r_init_vals[m]
        print(m)
        #initialises arrays for radial plunge angular momenta and proper time taken for the
    radial plunge to occur
        rad_plunge_l=[]
        plunge_t=[]
        print(r_init)

        #for loop to iterate through each value for the angular momentum
        for j in range(len(l_vals)):
                #initialies array for the r,z,tau, and phi values for the numerical
    integration
                r_vals=[]
                z_vals=[]
                tau_vals=[]
                phi_vals=[]

                #appends the initial values for the quantites to the appropriate arrays
                r_vals.append(r_init)
                z_vals.append(z_init)
                tau_vals.append(tau_init)
                phi_vals.append(phi_init)


                #sets the value for the angular momentum
                l = l_vals[j]

                #for loop to carry out the runge-kutta numerical integration
                for i in range(1,N):
                        #sets the values for the proper time, r co-ordinate, phi co-ordinate,
     and the rate of change of r wrt phi
                        tau=tau_vals[i-1]
                        r=r_vals[i-1]
                        z=z_vals[i-1]
                        phi=phi_vals[i-1]
                        plunge=False

                        k_0 = func_k_0(tau,r,z,h)
                        l_0 = func_l_0(tau,r,z,h,l)
                        m_0 = func_m_0(tau,r,z,h,l)


                        r1 = r+0.5*k_0
                        z1 = z+0.5*l_0
                        tau1 = tau+0.5*m_0

                        k_1 = func_k_0(tau1,r1,z1,h)
                        l_1 = func_l_0(tau1,r1,z1,h,l)
                        m_1 = func_m_0(tau1,r1,z1,h,l)
```

```
            r2 = r+0.5*k_1
            z2 = z+0.5*l_1
            tau2 = tau+0.5*m_1

            k_2 = func_k_0(tau2,r2,z2,h)
            l_2 = func_l_0(tau2,r2,z2,h,l)
            m_2 = func_m_0(tau2,r2,z2,h,l)

            r3 = r+k_2
            z3 = z+l_2
            tau3 = tau+m_2

            k_3 = func_k_0(tau3,r3,z3,h)
            l_3 = func_l_0(tau3,r3,z3,h,l)
            m_3 = func_m_0(tau3,r3,z3,h,l)

            #calculates values for the quantities for next step of the runge kutta integration
            r_new = r_vals[i-1] + (1/6)*(k_0+2*k_1+2*k_2+k_3)
            z_new = z_vals[i-1] + (1/6)*(l_0+2*l_1+2*l_2+l_3)
            tau_new = tau_vals[i-1] + (1/6)*(m_0+2*m_1+2*m_2+m_3)
            phi_new = phi_vals[i-1] + h

            #appends the values to the appropriate arrays to be used in next iteration of the loop
            r_vals.append(r_new)
            z_vals.append(z_new)
            tau_vals.append(tau_new)
            phi_vals.append(phi_new)

            r_test = r_new

            #else if statement to test whether the particle passes the Schwarzschild radius or effectively escapes to infinity
            if ((r_test <=1)):
                #appends angular momentum value for radial plunge to appropriate array
                rad_plunge_l.append(l)
                #appends proper time for radial plunge to occur to appropriate array
                plunge_t.append(tau_vals[i])
                #sets boolean to state that a radial plunge has occurred
                plunge=True
                #breaks the loop
                break
            elif((r_test>= 1e6)):
                #breaks the loop
                break
        #if an orbit occurs for which a radial plunge does not occur, the loop is broken
        if(plunge==False):
            break

    #prints the bounds of the angular momentum values for which a radial plunge orbit will occur
    print('l values for radial plunge orbit are between 0 and ' + str(rad_plunge_l[-1]))
    #sets value for critical angular momentum
    final_l = rad_plunge_l[-1]
    #appends value to array for critical angular momenta
```

```
        l_crit.append(final_l)
        #appends initial radius value to array for instance in which the critical angular momentum is not within bounds set previously such that arrays are of two different sizes
        r_crit_vals.append(r_init)
        #calculates the squared function to detemine the angular momentum for a circular orbit
        circ_l_sq = (func_circ_orbit(r_init))
        #tests to see if the value is negative such that the square root of the value will produce an imaginary number. if so, prints that circular orbit not possible for given radius
        if (circ_l_sq <0):
                print('A particle cannot achieve a circular orbit at this radial position.')
        elif(circ_l_sq>0):
                #calculates value of circular orbit angular momentum
                circ_l = np.sqrt(circ_l_sq)
                #appends value to array for circular orbit angular momenta
                circ_l_vals.append(circ_l)
                #calculates value of critical perturbation, i.e. perturbation required to disturb from circular orbit to radial plunge
                l_diff = circ_l - rad_plunge_l[-1]
                #appends value to array for critical perturbation
                var_l.append(l_diff)
                #appends value of initial radius to array for consideration with the circular orbits in case a circular orbit is not possible for a given radius and arrays are of different size
                circ_r_vals.append(r_init)
                #prints value of angular momentum for circular orbit to occur
                print('For a circular orbit and l value of l = ' + str(circ_l) + ' is required.')
        elif(r_init==1.5):
                #prints that for an initial radius of 1.5, that the particle is at the radius of the photon sphere where only photons can achieve circular orbits
                print('Only a photon can achieve a circular orbit at this radial position. This radius is that of the photon sphere')


        #saves the radial plunge angular momenta and proper time for plunge to occur to csv file
        dict = {'Radial Plunge l': rad_plunge_l, 'Proper Time': plunge_t}

        df=pd.DataFrame(dict)

        df.to_csv('radial_plunge_values_r='+str(r_init)+'_new.csv')

        #prints iteration of loop for inital radius values to act as milestone marker
        print(m)

#saves circular orbit angular momenta, corresponding initial radii, and critical perturbation to cause radial plunge from circular orbit
dict = {'Circular Orbit l': circ_l_vals, 'Radius': circ_r_vals, 'Variation in l': var_l}
df2 = pd.DataFrame(dict)
df2.to_csv('circular_l_values_values_new.csv')

#creates plot
ax = plt.subplot(111)

#plots the critical angular momentum as a function of initial radius
ax.plot(r_crit_vals,l_crit, label='Critical Angular Momentum as a Function of Initial Radius')
ax.plot(circ_r_vals,circ_l_vals, label='Angular Momentum for Circular Orbit')

```

```

ax.set_xlabel('Initial Radius $R$', fontsize=14)
ax.set_ylabel('Critical Angular Momentum $l$', fontsize=14)
ax.tick_params(labelsize=12)

ax.grid(True,alpha=0.5)
#adds legend and shows the plot
ax.legend(loc='upper right', fontsize='large')
plt.show()

#saves the critical angular momenta and corresponding initial radii to a csv file
dict = {'Critical Angular momentum': l_crit, 'Radius': r_crit_vals}
df3 = pd.DataFrame(dict)
df3.to_csv('critical_l_values_new.csv')

#creates plot
ax = plt.subplot(111)

#plots critical angular momentum perturbation as a function of initial radius of the orbit
ax.plot(r_init_vals,var_l, label='Perturbation to cause radial plunge')


ax.set_xlabel('Initial Radius $R$', fontsize=14)
ax.set_ylabel('Angular Momentum Perturbation $\Delta l$', fontsize=14)
ax.tick_params(labelsize=12)
ax.grid(True,alpha=0.5)
#adds legend and shows plot
ax.legend(loc='upper right', fontsize='large')
plt.show()
```

```
"""

Code for Question 5 of Project 14.1: Particle or Photon Orbits near a Black Hole

Name: Luke Timmons
Student Number: 304757457

"""

#imports libraries to be used
import PIL
from PIL import Image
import pandas as pd
from numpy import exp,arange
from pylab import meshgrid,cm,imshow,contour,clabel,colorbar,axis,title,show
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import math
import pylab
from lmfit import Model
import os

#function for the rate of change of radial positiion wrt co-ordinate time
def func_f(z):
        return(z)

#function for snd derivative of r wrt t
def func_g(tau,r,z,l,q):
        return(2*(z**2)/r + ((1-1/r)/(2*r))*((6/r - 4) + (r**4/(l*l*(1/r -1)))*q*q*(9*l*l/(r*
    r*r) - 6*l*l/(r*r) + 7/r - 4)))
```

```
#function for derivative of phi co-ordinate wrt co-ordinate time
def func_a(tau,r,z,l):
        return ((l/(r*r))*np.sqrt(((1-1/r)**2 - z**2)/((1-1/r)*(1+l*l/(r*r)))))

#function for derivative of proper time wrt co-ordinate time
def func_b(tau,r,z,l,q):
        return(q*r*r/l)




#defines functions for numerical integration via runge kutta method
def func_k_0(tau_i,r_i,z_i,h):
        return (h*func_f(z_i))


def func_l_0(tau_i,r_i,z_i,h,l,q):
        return (h*func_g(tau_i,r_i,z_i,l,q))

def func_m_0(tau_i,r_i,z_i,h,l):
        return (h*func_a(tau_i,r_i,z_i,l))

def func_n_0(tau_i,r_i,z_i,h,l,q):
        return(h*func_b(tau_i,r_i,z_i,l,q))

#defines function to determine angular momentum required for a given angular momentum
def func_circ_orbit(r_i):
        return (r_i**2/(2*(r_i-1.5)))



pi = np.pi
print(pi)

#sets the bounds for the co-ordinate time over which the numerical integration will take place as well as the number of steps for the
a=0
b=1000
N=1000
h=(b-a)/N


#initialises arrays for numerical integration
r_vals=[]
z_vals=[]
t_vals=[]
plunge_t=[]
phi_vals=[]
tau_vals=[]

#sets the value for the angular momentum
l=2.0


#sets the initial values for the quantities
r_init = 6.0
z_init = 0.0
t_init =0.0
phi_init=0.0
```

```
tau_init=0.0

#appends the initial values to the appropriate arrays
r_vals.append(r_init)
z_vals.append(z_init)
t_vals.append(t_init)
phi_vals.append(phi_init)
tau_vals.append(tau_init)

#initialises array for angular momentum for which a radial plunge orbit will occur
rad_plunge_l=[]

#initialises array for proper time taken for a radial plunge to occur
plunge_tau=[]

#creates an array for angular momentum values
l_vals = np.arange(1.00001,5.00001,0.00001)


#for loop within which the numerical integration is carried out
for i in range(1,N):
        #sets the values for t,r, dr/dt, phi,dphi/dt, and tau
        t=t_vals[i-1]
        r=r_vals[i-1]
        z=z_vals[i-1]
        phi = phi_vals[i-1]
        #calculates the derivative of phi wrt t
        q=func_a(phi,r,z,l)
        tau=tau_vals[i-1]

        k_0 = func_k_0(phi,r,z,h)
        l_0 = func_l_0(phi,r,z,h,l,q)
        m_0 = func_m_0(phi,r,z,h,l)
        n_0 = func_n_0(phi,r,z,h,l,q)

        r1 = r+0.5*k_0
        z1 = z+0.5*l_0
        phi1 = phi+0.5*m_0
        q1 = func_a(phi1,r1,z1,l)
        tau1 = tau + 0.5*n_0


        k_1 = func_k_0(phi1,r1,z1,h)
        l_1 = func_l_0(phi1,r1,z1,h,l,q1)
        m_1 = func_m_0(phi1,r1,z1,h,l)
        n_1 = func_n_0(phi1,r1,z1,h,l,q1)

        r2 = r+0.5*k_1
        z2 = z+0.5*l_1
        phi2 = phi+0.5*m_1
        q2 = func_a(phi2,r2,z2,l)
        tau2 = tau + 0.5*n_1

        k_2 = func_k_0(phi2,r2,z2,h)
        l_2 = func_l_0(phi2,r2,z2,h,l,q2)
        m_2 = func_m_0(phi2,r2,z2,h,l)
        n_2 = func_n_0(phi2,r2,z2,h,l,q2)


        r3 = r+k_2
        z3 = z+l_2
```

```
        phi3 = phi+m_2
        q3 = func_a(phi3,r3,z3,l)
        tau3 = tau + 0.5*n_2

        k_3 = func_k_0(phi3,r3,z3,h)
        l_3 = func_l_0(phi3,r3,z3,h,l,q3)
        m_3 = func_m_0(phi3,r3,z3,h,l)
        n_3 = func_n_0(phi3,r3,z3,h,l,q3)

        #calculates the quantites for the next step of the numerical integration
        r_new = r_vals[i-1] + (1/6)*(k_0+2*k_1+2*k_2+k_3)
        z_new = z_vals[i-1] + (1/6)*(l_0+2*l_1+2*l_2+l_3)
        phi_new = phi_vals[i-1] + (1/6)*(m_0+2*m_1+2*m_2+m_3)
        t_new = t_vals[i-1] + h
        tau_new = tau_vals[i-1] + (1/6)*(n_0+2*n_1+2*n_2+n_3)


        #appends the values to the appropriate arrays
        r_test = r_new
        r_vals.append(r_new)
        z_vals.append(z_new)
        t_vals.append(t_new)
        phi_vals.append(phi_new)
        tau_vals.append(tau_new)

        #checks if a NaN value has been determined for the radial position of the particle
        nan_test = math.isnan(r_new)
        nan_tau_test = math.isnan(tau_new)

        #if statement to test if the radial positiion of the particle has passed the Schwarzschild radius, an NaN value was encountered
        if r_test <=1:
                plunge_time = tau_vals[i-1]
                break
        elif nan_test == True:
                plunge_time = tau_vals[i-1]
                break
        elif nan_tau_test == True:
                plunge_time = tau_vals[i-1]
                break
        else:
                plunge_time = 0.0

        #if statement to test if the particle effectively escaped to infinity
        if r_test>=1e6:
                break


#creates a polar plot
ax = plt.subplot(111,projection='polar')


#adds circle to centre of plot to represent the black hole
circle= plt.Circle((0,0), radius= 1,color='black',transform=ax.transData._b)
ax.add_artist(circle)

#plots the particle motion for two cases: for a radial plunge or otherwise
if (plunge_time==0.0):
        ax.plot(phi_vals, r_vals, label='Particle Path (Initial R = ' +str(r_init)+' , l = ' + str(l) + ')')
else:
```

```
        ax.plot(phi_vals, r_vals, label='Particle Path (Initial R = ' +str(r_init)+' , l = '
    + str(l) + ', Plunge Proper Time = '+str(plunge_time)+')')

#adds legend to the plot
ax.legend(loc='lower center', fontsize='large')
ax.set_aspect('equal')
ax.tick_params(labelsize=15)

#shows the plot
plt.show()



#for loop to iterate through each value for the angular momentum
for j in range(len(l_vals)):
        #initialises arrays for numerical integration
        r_vals=[]
        z_vals=[]
        t_vals=[]
        phi_vals=[]
        tau_vals=[]

        #appends the initial values to the appropriate arrays
        r_vals.append(r_init)
        z_vals.append(z_init)
        t_vals.append(t_init)
        phi_vals.append(phi_init)
        tau_vals.append(tau_init)


        #sets the value for the angular momentum
        l = l_vals[j]

        #boolean to test if a radial plunge has occured
        plunge=False

        #for loop within which the numerical integration is carried out
        for i in range(1,N):
                #sets the values for t,r, dr/dt, phi,dphi/dt, and tau
                t=t_vals[i-1]
                r=r_vals[i-1]
                z=z_vals[i-1]
                phi = phi_vals[i-1]
                #calculates the derivative of phi wrt t
                q=func_a(phi,r,z,l)
                tau=tau_vals[i-1]

                k_0 = func_k_0(phi,r,z,h)
                l_0 = func_l_0(phi,r,z,h,l,q)
                m_0 = func_m_0(phi,r,z,h,l)
                n_0 = func_n_0(phi,r,z,h,l,q)

                r1 = r+0.5*k_0
                z1 = z+0.5*l_0
                phi1 = phi+0.5*m_0
                q1 = func_a(phi1,r1,z1,l)
                tau1 = tau + 0.5*n_0


                k_1 = func_k_0(phi1,r1,z1,h)
                l_1 = func_l_0(phi1,r1,z1,h,l,q1)
```

```
            m_1 = func_m_0(phi1,r1,z1,h,l)
            n_1 = func_n_0(phi1,r1,z1,h,l,q1)

            r2 = r+0.5*k_1
            z2 = z+0.5*l_1
            phi2 = phi+0.5*m_1
            q2 = func_a(phi2,r2,z2,l)
            tau2 = tau + 0.5*n_1

            k_2 = func_k_0(phi2,r2,z2,h)
            l_2 = func_l_0(phi2,r2,z2,h,l,q2)
            m_2 = func_m_0(phi2,r2,z2,h,l)
            n_2 = func_n_0(phi2,r2,z2,h,l,q2)


            r3 = r+k_2
            z3 = z+l_2
            phi3 = phi+m_2
            q3 = func_a(phi3,r3,z3,l)
            tau3 = tau + 0.5*n_2

            k_3 = func_k_0(phi3,r3,z3,h)
            l_3 = func_l_0(phi3,r3,z3,h,l,q3)
            m_3 = func_m_0(phi3,r3,z3,h,l)
            n_3 = func_n_0(phi3,r3,z3,h,l,q3)

            #calculates the quantites for the next step of the numerical integration
            r_new = r_vals[i-1] + (1/6)*(k_0+2*k_1+2*k_2+k_3)
            z_new = z_vals[i-1] + (1/6)*(l_0+2*l_1+2*l_2+l_3)
            phi_new = phi_vals[i-1] + (1/6)*(m_0+2*m_1+2*m_2+m_3)
            t_new = t_vals[i-1] + h
            tau_new = tau_vals[i-1] + (1/6)*(n_0+2*n_1+2*n_2+n_3)


            #appends the values to the appropriate arrays
            r_test = r_new
            r_vals.append(r_new)
            z_vals.append(z_new)
            t_vals.append(t_new)
            phi_vals.append(phi_new)
            tau_vals.append(tau_new)

            #checks if a NaN value has been determined for the radial position of the particle or the proper time
            nan_test = math.isnan(r_new)
            nan_tau_test = math.isnan(tau_new)

            #if statement to test if the radial positiion of the particle has passed the Schwarzschild radius, an NaN value was encountered for the radial position or proper time. if so the angular momentum value, and proper time are appended to arrays and loop is broken
            if r_test <=1:
                    rad_plunge_l.append(l)
                    plunge_time = tau_vals[i-1]
                    plunge_tau.append(plunge_time)
                    plunge=True
                    break
            elif nan_test == True:
                    rad_plunge_l.append(l)
                    plunge_time = tau_vals[i-1]
                    plunge_tau.append(plunge_time)
```

```
                        plunge=True
                        break
                elif nan_tau_test==True:
                        rad_plunge_l.append(l)
                        plunge_time = tau_vals[i-1]
                        plunge_tau.append(plunge_time)
                        plunge=True
                        break

        #if statement to test if the particle effectively escaped to infinity
        if r_test>=1e6:
                break
        elif plunge==False:
                break

#prints the bounds of the angular momentum values for which a radial plunge orbit will occur
print('l values for radial plunge orbit are between 0 and ' + str(rad_plunge_l[-1]))
#calculates the squared function to detemine the angular momentum for a circular orbit
circ_l_sq = (func_circ_orbit(r_init))
#tests to see if the value is negative such that the square root of the value will produce an
     imaginary number. if so, prints that circular orbit not possible for given radius
if (circ_l_sq <0):
        print('A particle cannot achieve a circular orbit at this radial position.')
elif (circ_l_sq>0):
        #calculates value of circular orbit angular momentum
        circ_l = np.sqrt(circ_l_sq)
        #prints value of angular momentum for circular orbit to occur
        print('For a circular orbit and l value of l = ' + str(circ_l) + ' is required.')
elif (r_init==1.5):
        print('Only photons can achieve a circular orbit for an initial radius of R=1.5')

#creates a plot
ax = plt.subplot(111)

#plots the proper time for a radial plunge to occur as a function of angular momentum
ax.plot(rad_plunge_l,plunge_tau, label='Proper Time for Radial Plunge (R =' + str(r_init) + '
    )')




ax.set_xlabel('Angular Momentum $l$', fontsize=16)
ax.set_ylabel(r'Proper Time $\tau$', fontsize=16)
ax.tick_params(labelsize=14)

#adds a grid to the plot
ax.grid(True,alpha=0.5)

#adds a legend to the plot
ax.legend(loc='upper right', fontsize='large')
#shows the plot
plt.show()


#saves the radial plunge angular momenta and proper time for plunge to occur to csv file
dict = {'Radial Plunge l': rad_plunge_l, 'Proper Time': plunge_tau}

df=pd.DataFrame(dict)

df.to_csv('radial_plunge_values_r='+str(r_init)+'_second_approach.csv')
```

```
"""

Code for Question 6 of Project 14.1: Particle or Photon Orbits near a Black Hole

Name: Luke Timmons
Student Number: 304757457

"""

#import libraries to be used
import PIL
from PIL import Image
import pandas as pd
from numpy import exp,arange
from pylab import meshgrid,cm,imshow,contour,clabel,colorbar,axis,title,show
import numpy as np
import matplotlib.pyplot as plt
import math
import pylab
from lmfit import Model
import os


#defines a power law function
def powfit(v,a,b):
        return(a*v**b)

#defines a hyperbolic function
def hypbolfit(x,a,b):
        return(a + b/x)

#sets a power law and hyperbolic model fit
powmodel = Model(powfit)
hypmodel = Model(hypbolfit)

#defines function for rate of change of the radial co-ordinate with respect to the phi co-
    ordinate
def func_f(r_i,b,v,phi):
        return(-1*(r**2)*np.sqrt(1/(b**2) + 1/(r**3) - 1/(r**2) + (1-v**2)/(v*v*b*b*r)))

#defines function for term inside sqrt of func_f to test if turning point has been reached (i
    .e. if func_r_test <= 0)
def func_r_test(r,b,v,phi):
        return(1/(b**2) + 1/(r**3) - 1/(r**2) + (1-v**2)/(v*v*b*b*r))

#function for runge-kutta numerical integration of func_f
def func_k_0(r_i,b,v,phi,h):
        #print (func_f(r_i,b,v,phi))
        return (h*func_f(r_i,b,v,phi))

#function for conitinuation of numerical integration after turning point has been reached (i.
    e. as the photon moves away from the black hole)
def func_k_0_pos(r_i,b,v,phi,h):
        return (-1*h*func_f(r_i,b,v,phi))




pi = np.pi
print(pi)
```

```

#defines the bounds of the phi co-ordinate over which the numerical integration will take place as well as the number of steps of the numerical integration
a=0
b=4*pi
N=10000
h=(b-a)/N


#initialies arrays for the scattering cross-section and the particle speeds at which the critical impact parameter (accounts for instance in which critical impact parameter exceeds bounds of test value)
sigma_vals = []
v_crit_vals =[]



#creates arrays for particle speeds and impact parameters to be tested
v_vals= np.arange(0.03, 1.01, 0.01)
b_vals=np.arange(0.01,60.01,0.01)

rad_plunge_l=[]

#initial radial position of the particle
r_init =  1000

#initialises the array for the critical impact parameter
b_crit_val=[]


#for loop over which the particle speeds are iterated through
for i in range(len(v_vals)):
        #sets the value of the particle speed
        v=v_vals[i]

        #for loop to iterate through the impact parameters
        for j in range(len(b_vals)):
                #initialises arrays for r and phi co-ordinates and test values to determine if turning point has been reached
                r_vals = []
                phi_vals = []
                turn_pt_vals=[]

                #sets the impact parameter
                b = b_vals[j]
                #calculates the initial phi co-ordinate based on the geometry wrt the impact parameter and initial radius
                phi_init = np.arcsin(b/r_init)

                #calculates the test value to determine if the turning point, i.e. the term in the sqrt in func_f is less than zero
                turn_pt_init = func_r_test(r_init,b,v,phi_init)

                #appends value to the array for test values
                turn_pt_vals.append(turn_pt_init)

                #appends the initial r and phi co-ordinates to the appropriate arrays
                r_vals.append(r_init)
                phi_vals.append(phi_init)

                #sets boolean to say if a radial plunge to occur
```

```
        plunge=False


        #for loop to calculate the quantities for the runge-kutta method for the instance in which the photon is infalling
        for m in range(1,N):


            r=r_vals[m-1]
            phi = phi_vals[m-1]
            #print(r)


            k_0 = func_k_0(r,b,v,phi,h)

            r1 = r+0.5*k_0

            k_1 = func_k_0(r1,b,v,phi,h)


            r2 = r+0.5*k_1

            k_2 = func_k_0(r2,b,v,phi,h)

            r3 = r+k_2

            k_3 = func_k_0(r3,b,v,phi,h)

            #calculates the new values for the r and phi co-ordinate of the runge kutta method
            r_new = r_vals[m-1] + (1/6)*(k_0+2*k_1+2*k_2+k_3)
            phi_new = phi_vals[m-1] + h

            #elseif statement that breaks the loop if the photon crosses the schwarzschild radius or escapes to its original position
            if(r_new<=1):
                #sets boolean to say that a radial plunge has occured
                plunge=True
                #breaks the loop
                break
            elif(r_new>=r_init):
                plunge=False
                #breaks the loop
                break

            #calculates the value of term in sqrt in func_f
            turn_pt = func_r_test(r_new,b,v,phi_new)
            #appends value to array
            turn_pt_vals.append(turn_pt)

            #elseif statement to determine if the turning point of the orbit has been reached
            if(m==1):
                pass
            elif(turn_pt_vals[m]<=0):
                #for loop that calculates the r and phi co-ordinates of the photon as it moves away from the black hole for the remaining steps of te numerical integration
                for q in range(m-1,N):

                    #sets values for r and phi co-ordinates
```

```
                    r=r_vals[q]
                    phi = phi_vals[q]

                    k_0 = func_k_0_pos(r,b,v,phi,h)

                    r1 = r+0.5*k_0

                    k_1 = func_k_0_pos(r1,b,v,phi,h)


                    r2 = r+0.5*k_1

                    k_2 = func_k_0_pos(r2,b,v,phi,h)

                    r3 = r+k_2

                    k_3 = func_k_0_pos(r3,b,v,phi,h)

                    #values for r and phi for next step of the numerical integration
                    r_new = r_vals[q] + (1/6)*(k_0+2*k_1+2*k_2+k_3)
                    phi_new = phi_vals[q] + h


                    #appends values to appropriate array
                    r_vals.append(r_new)
                    phi_vals.append(phi_new)


                    #elseif statement that breaks the loop if the photon crosses the schwarzschild radius or escapes to its original position
                    if(r_new<=1):
                        plunge=True
                        break
                    elif(r_new>=r_init):
                        plunge=False
                        break


                #breaks loop if the photon does not escape to original position or plunge to centre of attraction but the numerical integration has been completed for all steps
                break

            #appends values to the appropriate arrays
            r_vals.append(r_new)
            phi_vals.append(phi_new)

        #if statement for case of radial plunge not occuring
        if (plunge==False):

            print(b)
            print(v)
            #appends critical impact parameter and corresponding particle speeds to appropriate arrays
            b_crit_val.append(b)
            v_crit_vals.append(v)
            #calculates scattering cross section and appends to array
            sigma = pi*b*b
            sigma_vals.append(sigma)
            #breaks the loop
```

```
                        break




#saves the critical impact parameters, corresponding particle speeds, and scattering cross-section to csv file
dict = {'Speed v': v_crit_vals, 'Critical Impact Parameter': b_crit_val, 'Cross-section': sigma_vals}

df=pd.DataFrame(dict)

df.to_csv('cross_section_r='+str(r_init)+'_new.csv')


#creates a plot
ax = plt.subplot(111)

result = powmodel.fit(sigma_vals,v=v_crit_vals,a=13.2,b=-1.98)

#plots the scattering cross-section as a function of critical impact parameter
ax.plot(v_crit_vals,sigma_vals, label='Scattering cross-section as a function of particle speed')
ax.plot(v_crit_vals,result.best_fit, label=r'Power Law Model: $\sigma \propto v^{\alpha}$, $\alpha = -1.981923$')


ax.set_xlabel('Particle Speed $v$', fontsize=14)
ax.set_ylabel('Cross-section $\sigma$', fontsize=14)
ax.tick_params(labelsize=12)

print(result.params)

#adds a grid to the plot
ax.grid(True,alpha=0.5)
#adds the legend to the plot
ax.legend(loc='upper right', fontsize='large')
#shows the plot
plt.show()
```

```
"""

Code for Question 7: Part 1 of Project 14.1: Particle or Photon Orbits near a Black Hole

Name: Luke Timmons
Student Number: 304757457

"""


#import libraries to be used
import PIL
from PIL import Image
import pandas as pd
from numpy import exp,arange
from pylab import meshgrid,cm,imshow,contour,clabel,colorbar,axis,title,show
import numpy as np
import matplotlib.pyplot as plt
import math
import pylab
from lmfit import Model
```

```
import os

def powfit(v,a,b):
        return(a*v**b)

def hypbolfit(x,a,b):
        return(a + b/x)


powmodel = Model(powfit)
hypmodel = Model(hypbolfit)


#defines function for rate of change of the radial co-ordinate with respect to the phi co-ordinate
def func_f(r_i,b,v,phi):
        return(-1*(r**2)*np.sqrt(1/(b**2) + 1/(r**3) - 1/(r**2) + (1-v**2)/(v*v*b*b*r)))

#defines function for term inside sqrt of func_f to test if turning point has been reached (i.e. if func_r_test <= 0)
def func_r_test(r,b,v,phi):
        return(1/(b**2) + 1/(r**3) - 1/(r**2) + (1-v**2)/(v*v*b*b*r))

#function for runge-kutta numerical integration of func_f
def func_k_0(r_i,b,v,phi,h):
        #print (func_f(r_i,b,v,phi))
        return (h*func_f(r_i,b,v,phi))

#function for conitinuation of numerical integration after turning point has been reached (i.e. as the photon moves away from the black hole)
def func_k_0_pos(r_i,b,v,phi,h):
        return (-1*h*func_f(r_i,b,v,phi))





pi = np.pi
print(pi)

#defines the bounds of the phi co-ordinate over which the numerical integration will take place as well as the number of steps of the numerical integration
a=0
b=4*pi
N=1000000
h=(b-a)/N


#creates array for values for impact parameter
b_vals=np.arange(10,300,1)

#initialises array for angular momentum values for which a radial plunge orbit will occur
rad_plunge_l=[]


#sets the initial radial position of the photon
r_init =  100000


#intiialises array for the critical impact parameters values
b_crit_val=[]
```

```

#sets the speed of the photon
v = 1
b = 3.2
phi_diff=0.0

#initialises the arrays for the deflection angles calculated via the RK4 method, and from the large impact parameter analytical solution
phi_diff_vals=[]
theory_vals=[]

#for loop through which the deflection angle of the photon is determined as the impact parameter is incremented
for j in range(len(b_vals)):
        #initialises the array for the quantities to be calculated via the runge kutta method
        r_vals = []
        phi_vals = []
        turn_pt_vals=[]

        #sets the values for impact parameter
        b = b_vals[j]
        #sets the initial value for the phi co-ordinate calculated from the geometry wrt to the impact parameter and initial radial position
        phi_init = np.arcsin(b/r_init)

        #calculates the initial test value to determine whether the turning point of the orbit has been reached
        turn_pt_init = func_r_test(r_init,b,v,phi_init)

        #appends the initial test value to an array
        turn_pt_vals.append(turn_pt_init)

        #appends initial values of the r and phi co-ordinates to their respective arrays
        r_vals.append(r_init)
        phi_vals.append(phi_init)




        #for loop to calculate the quantities for the runge-kutta method for the instance in which the photon is infalling
        for m in range(1,N):


                r=r_vals[m-1]
                phi = phi_vals[m-1]



                k_0 = func_k_0(r,b,v,phi,h)

                r1 = r+0.5*k_0

                k_1 = func_k_0(r1,b,v,phi,h)


                r2 = r+0.5*k_1

                k_2 = func_k_0(r2,b,v,phi,h)

                r3 = r+k_2
```

```

        k_3 = func_k_0(r3,b,v,phi,h)

        #calculates the new values for the r and phi co-ordinate of the runge kutta method
        r_new = r_vals[m-1] + (1/6)*(k_0+2*k_1+2*k_2+k_3)
        phi_new = phi_vals[m-1] + h

        #tests if photon has passed the Schwarzschild radius
        if(r_new<=1):
            #breaks the loop
            break

        #calculates the value of term in sqrt in func_f
        turn_pt = func_r_test(r_new,b,v,phi_new)
        #appends value to array
        turn_pt_vals.append(turn_pt)




        #elseif statement to determine if the turning point of the orbit has been reached
        if(m==1):
            pass
        elif(turn_pt_vals[m]<=0):
            #for loop that calculates the r and phi co-ordinates of the photon as it moves away from the black hole for the remaining steps of te numerical integration
            for i in range(m-1,N):

                #sets values for r and phi co-ordinates
                r=r_vals[i]
                phi = phi_vals[i]
                r_i = r_vals[m-1]


                k_0 = func_k_0_pos(r,b,v,phi,h)

                r1 = r+0.5*k_0

                k_1 = func_k_0_pos(r1,b,v,phi,h)


                r2 = r+0.5*k_1

                k_2 = func_k_0_pos(r2,b,v,phi,h)

                r3 = r+k_2

                k_3 = func_k_0_pos(r3,b,v,phi,h)

                #values for r and phi for next step of the numerical integration
                r_new = r_vals[i] + (1/6)*(k_0+2*k_1+2*k_2+k_3)
                phi_new = phi_vals[i] + h

                #appends values to appropriate array
                r_vals.append(r_new)
                phi_vals.append(phi_new)

                #elseif statement that breaks the loop if the photon crosses
```

```
the schwarzschild radius or escapes to its original position
                if(r_new<=1):
                    break
                elif(r_new>=r_init):
                    break


            #breaks loop if the photon does not escape to original position or plunge to centre of attraction but the numerical integration has been completed for all steps
            break


        #appends values to the appropriate arrays
        r_vals.append(r_new)
        phi_vals.append(phi_new)

    #calculates angle of deflection from RK4 method
    phi_diff = phi_vals[-1] - phi_init - pi
    #calculates angle of deflection expected from large impact parameter approximation
    phi_diff_theory = 2/b
    #appends RK4 value to array
    phi_diff_vals.append(phi_diff)
    #prints iteration of impact parameter loop (acts as mile marker)
    print(j)
    #appends theory value to array
    theory_vals.append(phi_diff_theory)

    print(phi_diff)


#creates plot
ax = plt.subplot(111)

result = hypmodel.fit(phi_diff_vals,x=b_vals,a=-0.0005825,b=2.3224)

#plots rk4 deflection angles as a function of impact parameter
ax.plot(b_vals,phi_diff_vals, label='Runge-Kutta 4th Order Method')
#plots theoretical deflection angles as a function of impact parameter
ax.plot(b_vals,theory_vals, label = 'Analytical Solution')
#plots the hyperbolic model fit
ax.plot(b_vals,result.best_fit, label=r'Hyperbolic Model: $\delta \phi = \alpha + \gamma b^{-1}$, $\alpha = -0.0005825$; $\gamma=2.3224$')


#sets y-axis label,x-axis label, x-axis bounds, and y-axis bounds
ax.set_xlabel('Impact Parameter $b$', fontsize=16)
ax.set_ylabel('Deflection Angle $\delta \phi$ (Rad)', fontsize=16)
ax.tick_params(labelsize=14)
ax.set_xlim(0,300)
ax.set_ylim(0, 0.5)

print(result.params)

#applies a grid to the plot
ax.grid(True,alpha=0.5)

#adds a legends to the plot
ax.legend(loc='upper right', fontsize='x-large')

#shows the resulting plot
```

```
plt.show()



#saves the arrays for the impact parameter, the rk4 deflection angle, and theoretical
    deflection angles to a csv file
dict = {'Impact Parameter': b_vals, 'Deflection Angle': phi_diff_vals, 'Deflection Angle (
    Theory)': theory_vals}

df=pd.DataFrame(dict)

df.to_csv('deflection_angles_r='+str(r_init)+'_new.csv')
```

```
"""

Code for Question 7: Part 2 of Project 14.1: Particle or Photon Orbits near a Black Hole

Name: Luke Timmons
Student Number: 304757457

"""

#import libraries to be used
import PIL
from PIL import Image
import pandas as pd
from numpy import exp,arange
from pylab import meshgrid,cm,imshow,contour,clabel,colorbar,axis,title,show
import numpy as np
import matplotlib.pyplot as plt
import math
import pylab
from lmfit import Model
import os

#defines function for rate of change of the radial co-ordinate with respect to the phi co-
    ordinate
def func_f(r_i,b,v,phi):
        return(-1*(r**2)*np.sqrt(1/(b**2) + 1/(r**3) - 1/(r**2) + (1-v**2)/(v*v*b*b*r)))

#defines function for term inside sqrt of func_f to test if turning point has been reached (i
    .e. if func_r_test <= 0)
def func_r_test(r,b,v,phi):
        return(1/(b**2) + 1/(r**3) - 1/(r**2) + (1-v**2)/(v*v*b*b*r))

#function for runge-kutta numerical integration of func_f
def func_k_0(r_i,b,v,phi,h):
        return (h*func_f(r_i,b,v,phi))

#function for conitinuation of numerical integration after turning point has been reached (i.
    e. as the photon moves away from the black hole)
def func_k_0_pos(r_i,b,v,phi,h):
        return (-1*h*func_f(r_i,b,v,phi))




pi = np.pi
print(pi)

#sets bounds for phi co-ordinate for numerical integration and the number of the steps of the
```

```
    numerical integration
a=0
b=4*pi
N=1000000
h=(b-a)/N



#sets an initial radial position of the photon
r_init =  100000


#sets a value for the particle speed, the impact parameter, and the initial deflection angle
v = 1
b = 3.2
phi_diff=0.0

#initialises arrays for the r and phi co-ordinates and the test values to determine if the
    turning point of the orbit has been achieved
r_vals = []
phi_vals = []
turn_pt_vals=[]

#calculates the initial phi co-ordinate based on the geometry wrt the impact parameter and
    the initial radial position  of the photon
phi_init = np.arcsin(b/r_init)


#calculates test value to determine if the turning point of the orbit has been reached, i.e.
    if the term in the sqrt in func_f <=0
turn_pt_init = func_r_test(r_init,b,v,phi_init)

#appends the value to the array for the test values
turn_pt_vals.append(turn_pt_init)

#appends the initial r and phi co-ordinates to the appropriate arrays
r_vals.append(r_init)
phi_vals.append(phi_init)

#for loop to calculate the quantities for the runge-kutta method for the instance in which
    the photon is infalling
for m in range(1,N):


        r=r_vals[m-1]
        phi = phi_vals[m-1]



        k_0 = func_k_0(r,b,v,phi,h)

        r1 = r+0.5*k_0

        k_1 = func_k_0(r1,b,v,phi,h)


        r2 = r+0.5*k_1

        k_2 = func_k_0(r2,b,v,phi,h)

        r3 = r+k_2
```

```

    k_3 = func_k_0(r3,b,v,phi,h)

    #calculates the new values for the r and phi co-ordinate of the runge kutta method
    r_new = r_vals[m-1] + (1/6)*(k_0+2*k_1+2*k_2+k_3)
    phi_new = phi_vals[m-1] + h


    #tests if photon has passed the Schwarzschild radius
    if(r_new<=1):
        #calculates the deflection angle in the case of a radial plunge orbit
        phi_diff=phi_new - phi_vals[0] - pi
        break

    #calculates the value for testing whether the turning point of the orbit has been reached
    turn_pt = func_r_test(r_new,b,v,phi_new)
    #appends the value to array for the test values
    turn_pt_vals.append(turn_pt)


    #elseif statement to determine if the turning point of the orbit has been reached
    if(m==1):
        pass
    elif(turn_pt_vals[m]<=0):
        #for loop that calculates the r and phi co-ordinates of the photon as it moves away from the black hole for the remaining steps of te numerical integration
        for i in range(m-1,N):

            #sets values for r and phi co-ordinates
            r=r_vals[i]
            phi = phi_vals[i]
            r_i = r_vals[m-1]


            k_0 = func_k_0_pos(r,b,v,phi,h)

            r1 = r+0.5*k_0

            k_1 = func_k_0_pos(r1,b,v,phi,h)


            r2 = r+0.5*k_1

            k_2 = func_k_0_pos(r2,b,v,phi,h)

            r3 = r+k_2

            k_3 = func_k_0_pos(r3,b,v,phi,h)

            #values for r and phi for next step of the numerical integration
            r_new = r_vals[i] + (1/6)*(k_0+2*k_1+2*k_2+k_3)
            phi_new = phi_vals[i] + h

            #appends values to appropriate array
            r_vals.append(r_new)
            phi_vals.append(phi_new)

            #elseif statement that breaks the loop if the photon crosses the schwarzschild radius or escapes to its original position
            if(r_new<=1):
```

```
                                break
                        elif(r_new>=r_init):
                                break


                #breaks loop if the photon does not escape to original position or plunge to centre of attraction but the numerical integration has been completed for all steps
                break


        #appends values to the appropriate arrays
        r_vals.append(r_new)
        phi_vals.append(phi_new)



#calculates angle of deflection from RK4 method
phi_diff = phi_vals[-1] - phi_init - pi


#creates polar plot
ax = plt.subplot(111,projection='polar')


#plots circle at centre of polar plot to represent the black hole
circle= plt.Circle((0,0), radius= 1,color='black',transform=ax.transData._b)
ax.add_artist(circle)
#plots the motion of the particle in (r,phi) space using the values determined from the runge kutta method
ax.plot(phi_vals, r_vals, label='Particle Path (Initial R = ' +str(r_init)+', b = '+ str(b) + ', Angle Deflected = '+str(phi_diff) + ' Rad)')
#adds legend to the plot
ax.legend(loc='lower center', fontsize='large')
#sets aspect ratio of the plot
ax.set_aspect('equal')
ax.tick_params(labelsize=15)
#shows the plot
plt.show()


#creates polar plot
ax = plt.subplot(111,projection='polar')


#plots circle at centre of polar plot to represent the black hole
circle= plt.Circle((0,0), radius= 1,color='black',transform=ax.transData._b)
ax.add_artist(circle)
#plots the motion of the particle in (r,phi) space using the values determined from the runge kutta method
ax.plot(phi_vals, r_vals, label='Particle Path (Initial R = ' +str(r_init)+', b = '+ str(b) + ', Angle Deflected = '+str(phi_diff) + ' Rad)')
#adds legend to the plot
ax.legend(loc='lower center', fontsize='large')
#sets aspect ratio of the plot
ax.set_aspect('equal')
ax.tick_params(labelsize=15)
ax.set_rmax(10.0)
#shows the plot
plt.show()
```