# Geodesic Trajectories Simulation in a Kerr Space-Time

Louis Touzalin

May 2024

# Contents

# 1 Introduction

General relativity, developed by Albert Einstein at the beginning of the 20th century, changed our knowledge about gravitation. This theory describes gravitation as a space-time curvature induced by the presence of matter-energy. One of the most important solutions to Einstein's field equations (and the one that interests us there) is Kerr metric (1963), which describes the geometry of space-time around a rotating black hole's (Based on Schwarzschild static solutions 1916).

Geodesics are the trajectories followed by particles on a manifold. this concept is essential to understand the structure of Kerr space-time. theses curved lines are related to partial differential equations, which can be derived from the principle of least action (Einstein Hilbert). Numerical simulation of these trajectories allows us to visualize and analyze how particles moves around massive objects, such as black hole's. It's one of the best way to study the effect of these extreme regions of space time.

In this project, I explore the study of these physical phenomena with a simulation code that combines stability and optimization without skimping on simulation. The results of these simulations are visualized using the software ParaView.

# 2 A Bit of Theory

Before discussing code, it is necessary to delve into the mathematical foundations of what black holes are and the equations that govern these mysterious regions of space-time.

## 2.1 Kerr Metric

Like the Schwarzschild metric, the Kerr metric describes a curved space-time, but the difference lies in the object being studied. It describes the geometry of space-time around a rotating black hole. It is a generalization of the Schwarzschild metric, which describes the geometry around a non-rotating BH. The Kerr metric is named after physicist Roy Kerr, who first described it in 1963.

It is given by:

$$\begin{aligned} ds^2 = &-\left(1 - \frac{2Mr}{\rho^2}\right) dt^2 - \frac{4aMr\sin^2\theta}{\rho^2} dt d\phi \\ &+ \frac{\rho^2}{\Delta} dr^2 + \rho^2 d\theta^2 \\ &+ \left(r^2 + a^2 + \frac{2a^2 Mr\sin^2\theta}{\rho^2}\right)\sin^2\theta d\phi^2 \end{aligned} \tag{2.1}$$

Where:

$$\begin{aligned} \Delta &= r^2 - 2Mr + a^2 \\ \rho^2 &= r^2 + a^2\cos^2\theta \end{aligned}$$

$M$ is the mass of the black hole, $a$ is the rotation parameter (angular momentum per unit mass), $r$ is the radial distance from the center of the black hole, $\theta$ is the polar angle, $\phi$ is the azimuthal angle, $c$ is the speed of light in a vacuum, and $G$ is the gravitational constant.

For the purposes of simulation, we have given the metric in this form since we will consider $G = c = 1$, which will reduce computational load.

Its metric components are:

$$\begin{aligned} g_{tt} &= -\left(1 - \frac{2Mr}{\rho^2}\right) \\ g_{t\phi} &= -\frac{4aMr\sin^2\theta}{\rho^2} \\ g_{rr} &= \frac{\rho^2}{\Delta} \\ g_{\theta\theta} &= \rho^2 \\ g_{\phi\phi} &= \left(r^2 + a^2 + \frac{2a^2 Mr\sin^2\theta}{\rho^2}\right)\sin^2\theta \end{aligned}$$

which in matrix form gives:

$$g_{\mu\nu} = \begin{pmatrix} -\left(1 - \frac{2Mr}{\rho^2}\right) & 0 & 0 & -\frac{2aMr\sin^2\theta}{\rho^2} \\ 0 & \frac{\rho^2}{\Delta} & 0 & 0 \\ 0 & 0 & \rho^2 & 0 \\ -\frac{2aMr\sin^2\theta}{\rho^2} & 0 & 0 & \left(r^2 + a^2 + \frac{2a^2Mr\sin^2\theta}{\rho^2}\right)\sin^2\theta \end{pmatrix}$$

The components of the Kerr metric, $g_{\mu\nu}$, are functions of the spatial and temporal coordinates that follow the same algebraic rules as the components of the Schwarzschild metric:

$$t \in \mathbb{R}, r \in \mathbb{R}^+, \theta \in [0, \pi], \phi \in [0, 2\pi], r \neq 2m, 0 \tag{2.2}$$

These metric components describe the geometric properties of the space-time around the rotating black hole, such as the curvature of space-time and the way particles move in this curved space-time.

The space-time is a manifold following the algebraic properties:

$$\mathcal{M} := \mathbb{R}^2 \times \mathbb{S}^2 / \mathcal{R} \tag{2.3}$$

where:

$$\mathcal{R} := \{p \in \mathbb{R}^2 \times \mathbb{S}^2 \mid r(p) = 0, \theta(p) = \frac{\pi}{2}\} \tag{2.4}$$

$(t, r)$ extends to $\mathbb{R}^2$ and $(\theta, \phi)$ extends to $\mathbb{S}^2$.

The Kerr metric is essential for understanding the properties of rotating black holes, including their ergosphere and event horizon. It is also used to study astrophysical phenomena associated with rotating black holes, such as matter jets and accretion of matter in the accretion disk.

# 3 Curvature Tensors

Before introducing the main notion of this trajectory simulation, it is relevant to present the most fundamental and elegant mathematical concept of general relativity: the Riemann tensor, which allows the calculation of the spacetime curvature from a given metric.

Indeed, metrics describe the space in which the studied physical system is placed. To correctly describe and represent the behavior of particles in this curved space-time, it is necessary to introduce the concept of curvature.

## 3.1 Riemann Tensor and Ricci Tensor

The Riemann tensor is defined as a tensorial quantity that appears in differential geometry and general relativity. It measures the intrinsic curvature of space-time. Defined with Levi-Civita connection operators as:

$$R(\mu, \nu)\sigma = \nabla_\mu \nabla_\nu \omega - \nabla_\nu \nabla_\mu \omega - \nabla_{[\mu,\nu]}\omega \tag{3.1}$$

Here, $R(\mu, \nu)$ is a linear transformation dependent on each point of the tangent space to the manifold, and $[\mu, \nu]$ is a Lie bracket. Conceptually, if $\mu = \frac{\partial}{\partial_{x_i}}$ and

$\nu = \frac{\partial}{\partial_{x_j}}$ are coordinate vector fields, then the bracket dependent on the tensor is zero. Thus, we can rewrite the expression as:

$$R(\mu, \nu)\sigma = \nabla_\mu \nabla_\nu \omega - \nabla_\nu \nabla_\mu \omega \tag{3.2}$$

In the context of a relativistic, hence spatial, application of the Riemann tensor, it is preferable to define it by its identities dependent on the coordinate system:

$$R^\alpha_{\beta\mu\nu} = \partial_\mu \Gamma^\alpha_{\beta\nu} - \partial_\nu \Gamma^\alpha_{\beta\mu} + \Gamma^\alpha_{\rho\mu}\Gamma^\rho_{\beta\nu} - \Gamma^\alpha_{\rho\nu}\Gamma^\rho_{\beta\mu} \tag{3.3}$$

With the Christoffel symbols $\Gamma^\alpha_{\beta\mu}$ defined in the previous section. It is possible to contract it into a compact Ricci tensor, which is an essential component of Einstein's equations that describe how matter and energy affect the curvature of space-time.

$$R^l_{ijk} = \frac{\partial \Gamma^l_{ij}}{\partial x^k} - \frac{\partial \Gamma^l_{ik}}{\partial x^j} + \Gamma^m_{ij}\Gamma^l_{mk} - \Gamma^m_{ik}\Gamma^l_{mj} \tag{3.4}$$

where $R^l_{ijk}$ represents the Riemann tensor, $\Gamma^k_{ij}$ are the Christoffel symbols, $x^k$ are the spatial coordinates, and $\partial$ represents the partial derivatives. This expression encapsulates how the second derivatives of the space-time metric components vary from one point to another, thus providing crucial information about the curvature of space-time.

To contract the Riemann tensor and obtain the Ricci tensor, we use Einstein summation notation, where repeated indices are summed. By contracting the first and third indices of the Riemann tensor, we get:

$$Ric_{\mu\nu} = R^\sigma_{\mu\sigma\nu}, \tag{3.5}$$

where $R^\sigma_{\mu\sigma\nu}$ is a component of the Riemann tensor $R^\alpha_{\beta\mu\nu}$. This contraction involves setting $\alpha = \sigma$ and summing over this common index $\sigma$.

Geometrically, the Ricci tensor provides a measure of the average curvature of space-time at a point, taking into account the contribution from all directions. It is a rank-two tensor that requires a good understanding of the Riemann tensor for easier calculations.

The Riemann tensor can be expressed as follows, according to the affine connections between the different vectors:

$$R(X, Y)Z = \nabla_Y \nabla_X Z - \nabla_X \nabla_Y Z - \nabla_{[X,Y]} Z \tag{3.6}$$

Where $\nabla$ is the affine connection symbol between the vector fields $X$ and $Y$ on the tensor field $Z$.

To further develop the concept, consider a Riemannian manifold $(M, g)$ with a Levi-Civita connection $\nabla$ and $M$ a field of smooth vectors $X, Y, Z$ such that:

$$R_p : T_pM \times T_pM \times T_pM \to T_pM, \quad p \in M \tag{3.7}$$

where $R$ is a tensor field in $\mathbb{R}$ such that:

$$\forall p \in \mathbb{R}, \quad Ric_p : T_pM \times T_pM \in \mathbb{R} \tag{3.8}$$

This allows defining Ricci as follows:

$$Ric_p(X,Z) := \mathrm{tr}(X \to R_p(X,Y)Z) \tag{3.9}$$

$$Ric_p(X,Z) := \sum_{i=1} \langle R_p(v_i,Y)Z_{v_i} \rangle \tag{3.10}$$

As an application of the Riemann tensor to $(v_i, Y)Z_{v_i}$, this can be algebraically verified by:

$$Ric_{\mu\nu} = R^{\sigma}_{\nu\sigma\mu} = R^{\sigma}_{\mu\sigma\nu} \tag{3.11}$$

Moreover, the Christoffel symbol $\Gamma^{\sigma}_{\mu\nu}$ can be expressed as a vector sum of the partial derivatives of the metric components:

$$\Gamma^{\sigma}_{\mu\nu} = \frac{1}{2} \sum_{\sigma=1}^{n} \left( \frac{\partial g_{\nu\sigma}}{\partial x^{\mu}} + \frac{\partial g_{\sigma\nu}}{\partial x^{\nu}} - \frac{\partial g_{\mu\nu}}{\partial x^{\sigma}} \right) g^{\lambda\sigma} \tag{3.12}$$

Thus,

$$R_{ij} = \sum_{\mu=1}^{n} \frac{\partial \Gamma^{\mu}_{\mu i}}{\partial x^{\mu}} - \sum_{\mu=1}^{n} \frac{\partial \Gamma^{\mu}_{\mu i}}{\partial x^{j}} + \sum_{\mu=1}^{n} \sum_{\nu=1}^{n} (\Gamma^{\mu}_{\mu\nu}\Gamma^{\nu}_{ij} - \Gamma^{\mu}_{i\nu}\Gamma^{\nu}_{\mu j}) \tag{3.13}$$

with:

$$\phi : U \to \mathbb{R} \tag{3.14}$$

Note that this section is an introduction to situate within the documentation of this article to understand the complex and elegant notion of tensors in general relativity. To give an idea, we could develop a complete article on these notions, which is not the goal here.

# 4 Usual Mathematics

In this section, we will remember some common mathematical tools that are required when simulating particle's paths in Kerr spacetime.

We will start with a simple concept of discretization, which has a crucial task of converting the differential equations of general relativity into the algebraic form suitable for computerized numerical solution.

First of, we will rely on the geodesic equations which govern the motion of particles in curved spacetime and then, we will expose how numerical resolution to such equations is best achieved through integration techniques.

## 4.1 Discretization of Concepts

Discretization is the process of mapping a numeric variable into a discrete set or approximation of that variable. This will often be required in numerical problem solving since discretizing a system in some way divides the system into finite elements that are suitable for numerical computation.

In numerical simulation, discretization is used to render the equations to a form that can be handled by a computer in relation to general relativity. This entails a discretization of spacetime into a three dimensional world and portray physical variables such as magnetic field, plasma density, etc. , algebraically on this world.

The choice of grid size (i.e., the values of $\delta\mu, \delta\beta, \delta\nu$) is a trade-off between precision and computational cost: a finer grid will provide more accurate results but will require more computing resources.

Let's take a simple example with a Christoffel symbol $\Gamma^{\sigma}_{\mu\nu}$ given by:

$$\Gamma^{\sigma}_{\mu\nu} = g^{\sigma\rho}\left(\frac{\partial g_{\rho\mu}}{\partial x^{\nu}} + \frac{\partial g_{\rho\nu}}{\partial x^{\mu}} - \frac{\partial g_{\mu\nu}}{\partial x^{\rho}}\right) \tag{4.1}$$

and discretize it using finite differences with the following equation:

$$\frac{1}{2}\sum_{\beta=1}^{n}\left(\frac{g_{\mu\beta}(\beta+\delta\beta) - g_{\mu\beta}(\beta)}{\delta\beta} + \frac{g_{\beta\nu}(\mu+\delta\mu) - g_{\beta\nu}(\mu)}{\delta\mu} - \frac{g_{\beta\mu}(\nu+\delta\nu) - g_{\beta\mu}(\nu)}{\delta\nu}\right) \tag{4.2}$$

In this equation:
- $\Gamma^{\sigma}_{\mu\nu}$ represents the Christoffel symbol.
- $n$ represents the dimension of spacetime.
- $\mu$, $\nu$, and $\sigma$ are spatial indices.
- $\beta$ is a summation index over all spacetime directions.
- $g_{\mu\beta}$, $g_{\beta\nu}$, and $g_{\beta\mu}$ are components of the metric tensor.
- $\delta\beta$, $\delta\mu$, and $\delta\nu$ are discretization intervals in the $\beta$, $\mu$, and $\nu$ directions, respectively.

Similarly, we can discretize the Riemann tensor as follows:

$$\begin{aligned} R_{ij} = &\sum_{\mu=1}^{n}\frac{\Gamma^{\mu}_{\mu i}(\mu+\delta\mu) - \Gamma^{\mu}_{\mu i}(\mu)}{\delta\mu} - \sum_{\mu=1}^{n}\frac{\Gamma^{\mu}_{\mu i}(j+\delta j) - \Gamma^{\mu}_{\mu i}(j)}{\delta j} \\ &+ \sum_{\mu=1}^{n}\sum_{\nu=1}^{n}\left(\frac{\Gamma^{\mu}_{\mu\nu}(\mu+\delta\mu) - \Gamma^{\mu}_{\mu\nu}(\mu)}{\delta\mu}\frac{\Gamma^{\nu}_{ij}(j+\delta j) - \Gamma^{\nu}_{ij}(j)}{\delta j}\right. \\ &\left. - \frac{\Gamma^{\mu}_{i\nu}(\mu+\delta\mu) - \Gamma^{\mu}_{i\nu}(\mu)}{\delta\mu}\frac{\Gamma^{\nu}_{\mu j}(j+\delta j) - \Gamma^{\nu}_{\mu j}(j)}{\delta j}\right) \end{aligned} \tag{4.3}$$

These equations show how spatial distributions can be estimated on a discrete grid using finite differences. In the context of this small project, I mainly used the Christoffel symbols, which I derived numerically from the metric tensor.

## 4.2 Geodesics and Numerical Derivatives

Geodesics are the trajectories followed by particles in a curved spacetime. They are defined as the curves that minimize the distance (or proper time) between two points in spacetime. The geodesic equations can be derived from the principle of least action:

$$\frac{d^2x^\mu}{d\tau^2} + \Gamma^\mu_{\nu\rho}\frac{dx^\nu}{d\tau}\frac{dx^\rho}{d\tau} = 0 \tag{4.4}$$

where $x^\mu(\tau)$ represents the coordinates of the particle as a function of proper time $\tau$.

To approximate the values taken for each grid point, I use a fourth-order Runge-Kutta method, which provides good accuracy while remaining computationally acceptable.

This method approximates the solution by calculating four intermediate increments $(k_1, k_2, k_3, k_4)$ at each integration step:

$$k_1 = f(x_n, v_n) \tag{4.5}$$

$$k_2 = f(x_n + \frac{h}{2}k_1, v_n + \frac{h}{2}k_1) \tag{4.6}$$

$$k_3 = f(x_n + \frac{h}{2}k_2, v_n + \frac{h}{2}k_2) \tag{4.7}$$

$$k_4 = f(x_n + hk_3, v_n + hk_3) \tag{4.8}$$

where $f(x, v)$ is the function that computes the second derivative of $x^\mu$ using the geodesic equation, and $h$ is the integration step size.

The position and four-velocity of the particle are then updated using a weighted average of the intermediate increments:

$$x_{n+1} = x_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{4.9}$$

$$v_{n+1} = v_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{4.10}$$

The points of the computed geodesic are stored using a function, which converts spherical coordinates $(r, \theta, \phi)$ into Cartesian coordinates $(x, y, z)$ and also stores the value of the affine parameter $\lambda$ corresponding to each point.

Using OpenMP `#pragma omp parallel for` directives, certain loops are parallelized to speed up computation by leveraging multi-processor systems.

Here, this function takes two arrays $x[4]$ and $v[4]$, which hold the initial positions and velocities of the particles that will be iterated over in the given directions.

```
1       void geodesic(double x[4], double v[4], double lambda_max,
            double christoffel[4][4][4], double step_size, void
            (*store_point)(double[], double))
2   {
3       double k1[4], k2[4], k3[4], k4[4], temp_x[4], temp_v[4];
4       double lambda = 0;
5       short step = 0;
6       (void)store_point;
7       for(; lambda < lambda_max;)
8       {
9           #pragma omp simd aligned(v, christoffel: ALIGNMENT)
10          for (int mu = 0; mu < 4; mu++) {
11              k1[mu] = v[mu];
12              for (int alpha = 0; alpha < 4; alpha++) {
13                  for (int beta = 0; beta < 4; beta++) {
14                      double temp = christoffel[mu][alpha][beta]
                            * v[alpha] * v[beta];
15                      k1[mu] -= temp;
16                  }
17              }
18              temp_x[mu] = x[mu] + 0.5 * step_size * k1[mu];
19              temp_v[mu] = v[mu] + 0.5 * step_size * k1[mu];
20          }
21          #pragma omp simd aligned(temp_v, christoffel: ALIGNMENT)
22          for (int mu = 0; mu < 4; mu++) {
23              k2[mu] = temp_v[mu];
24              for (int alpha = 0; alpha < 4; alpha++) {
25                  for (int beta = 0; beta < 4; beta++) {
26                      double temp = christoffel[mu][alpha][beta]
                            * temp_v[alpha] * temp_v[beta];
27                      k2[mu] -= temp;
28                  }
29              }
30              temp_x[mu] = x[mu] + 0.5 * step_size * k2[mu];
31              temp_v[mu] = v[mu] + 0.5 * step_size * k2[mu];
32          }
33          #pragma omp simd aligned(temp_v, christoffel: ALIGNMENT)
34          for (int mu = 0; mu < 4; mu++) {
35              k3[mu] = temp_v[mu];
36              for (int alpha = 0; alpha < 4; alpha++) {
37                  for (int beta = 0; beta < 4; beta++) {
38                      double temp = christoffel[mu][alpha][beta]
                            * temp_v[alpha] * temp_v[beta];
39                      k3[mu] -= temp;
40                  }
41              }
42              temp_x[mu] = x[mu] + step_size * k3[mu];
43              temp_v[mu] = v[mu] + step_size * k3[mu];
44          }
45                      #pragma omp simd aligned(temp_v,
                            christoffel: ALIGNMENT)
46              for (int mu = 0; mu < 4; mu++) {
47                  k4[mu] = temp_v[mu];
48                  for (int alpha = 0; alpha < 4; alpha++) {
49                      for (int beta = 0; beta < 4; beta++) {
50                          double temp =
                                christoffel[mu][alpha][beta] *
```

```
                                temp_v[alpha] * temp_v[beta];
                            k4[mu] -= temp;
                        }
                    }
                }
                #pragma omp simd aligned(x, v, k1, k2, k3, k4:
                    ALIGNMENT)
                for (int mu = 0; mu < 4; mu++) {
                    x[mu] += step_size * (k1[mu] + 2 * k2[mu] + 2 *
                        k3[mu] + k4[mu]) / 6.0;
                    v[mu] += step_size * (k1[mu] + 2 * k2[mu] + 2 *
                        k3[mu] + k4[mu]) / 6.0;
                }
            store_geodesic_point(x, lambda);
            step++;
            lambda += step_size;
        }
    }
```

The main parameter is the array $christoffel[4][4][4]$, which holds precomputed symbol calculations based on the metric. Geodesics are calculated as follows, starting with a definition of the chosen metric in its matrix form:

```
double H = 10.0;
double r = sqrt_asm(powf(x[1], 2) + powf(a, 2) * \
                powf(cos(x[2]), 2));
double f = 2 * r * H / (powf(r, 2) + powf(a, 2) * \
                powf(cos(x[2]), 2));
double rho2 = powf(r, 2) + powf(a, 2) * \
                    powf(cos(M_PI / 2), 2);
double delta = powf(r, 2) - Rs * r + powf(a, 2);
double Sigma = powf(r, 2) + powf(a, 2);
double A = (powf(r, 2) + powf(a, 2)) * Sigma + Rs * \
                r * powf(a, 2) * powf(sin(M_PI / 2), 2);
double g00 = -(1-Rs*r/rho2);
double g03 = -Rs*r*a* powf(sinf(M_PI / 2), 2)/rho2;
double g11 = rho2 / delta;
double g22 = rho2;
double g30 = -Rs*r*a*powf(sinf(M_PI/2),2)/rho2;
double g33 = (Sigma*powf(sinf(M_PI/2),2))/rho2;
long double g_kerr[4][4] =
{
            {g00, 0, 0, g03},
            {0, g11, 0, 0},
            {0, 0, g22, 0},
            {g03, 0, 0 ,g33}
};
```

This first part defines the Kerr metric (another one can be chosen). Then, the parameter $gkerr[4][4]$ is passed to the function $christoffel$:

```
#pragma omp declare simd
void christoffel(double g[4][4], double christoffel[4][4][4])
{
    double (*g_aligned)[4] = aligned_alloc(ALIGNMENT, \
                                sizeof(double[4][4]));
    double (*christoffel_aligned)[4][4] =
        aligned_alloc(ALIGNMENT, \

        sizeof(double[4][4][4]));
    memcpy(g_aligned, g, sizeof(double[4][4]));
    memcpy(christoffel_aligned, christoffel, \
                                sizeof(double[4][4][4]));


    #pragma omp parallel for collapse(3)
    for (int mu = 0; mu < 4; mu++) {
        for (int beta = 0; beta < 4; beta++) {
            for (int nu = 0; nu < 4; nu++) {
                double s = 0;
                #pragma omp simd reduction(+:sum)aligned\
                                (g_aligned:ALIGNMENT)
                for (int sigma = 0; sigma < 4; sigma++) {
                    s += 0.5 * (g_aligned[mu][sigma] *
                        (g_aligned[sigma] \
                     [beta] + g_aligned[beta][sigma] -
                         g_aligned[beta][nu]));
                }
                christoffel_aligned[mu][beta][nu] = sum;
            }
        }
    }

    memcpy(christoffel, christoffel_aligned,
        sizeof(double[4][4][4]));
    free(g_aligned);
    free(christoffel_aligned);
}
```

The Christoffel function takes as input a $g[4][4]$ array representing the space-time metric, and returns a $christoffel[4][4][4]$ array containing the calculated Christoffel symbols.

where $g^{\mu\sigma}$ is the inverse of the metric $g_{\mu\sigma}$, and $\frac{\partial g_{\sigma\alpha}}{\partial x^\beta}$ represents the partial derivative of the metric component $g_{\sigma\alpha}$ with respect to the coordinate $x^\beta$.

In the christoffel function, this formula is approximated using finite differences. Specifically, the partial derivative is replaced by a centered finite difference:

$$\frac{\partial g_{\sigma\alpha}}{\partial x^\beta} \approx \frac{g_{\sigma\alpha}(x^\beta + h) - g_{\sigma\alpha}(x^\beta - h)}{2h} \tag{4.11}$$

where $h$ is a small discretization step.

Using this approximation, the formula for the Christoffel symbols becomes:

$$\begin{aligned}\Gamma^{\mu}\alpha\beta \approx & \frac{1}{2}g^{\mu\sigma}\left(\frac{g\sigma\alpha(x^{\beta}+h)-g_{\sigma\alpha}(x^{\beta}-h)}{2h}\right. \\ & +\frac{g_{\sigma\beta}(x^{\alpha}+h)-g_{\sigma\beta}(x^{\alpha}-h)}{2h} \\ & \left.-\frac{g_{\alpha\beta}(x^{\sigma}+h)-g_{\alpha\beta}(x^{\sigma}-h)}{2h}\right)\end{aligned} \tag{4.12}$$

The christoffel function implements this formula using nested loops over the indices $\mu$, $\alpha$, $\beta$, and $\sigma$. For each triplet of indices $(\mu, \alpha, \beta)$, it computes the sum over $\sigma$ using the finite difference approximation, and then stores the result in the $christoffel[\mu][\alpha][\beta]$ array.

It is important to note that this implementation uses finite differences approximations to compute the partial derivatives, which can introduce numerical errors. More accurate methods, such as higher-order finite differences or spectral methods, could be used to improve the accuracy of the Christoffel symbol calculation, at the expense of increased complexity.

# 5 Plot and Physical Interpretation

By visualizing the geodesics in the Kerr spacetime, we can directly observe the effects of spacetime curvature on the trajectories of particles. Geodesics can exhibit unique characteristics such as stable orbits, unstable orbits, trajectories captured by the black hole, or trajectories escaping gravitational attraction. Analyzing these trajectories helps us better understand the nature of black holes and the limits of general relativity.

Furthermore, studying the physical properties of geodesics, such as energy, angular momentum, and proper time, provides valuable information about the characteristics of particles moving in Kerr spacetime. This information can be used to study phenomena such as the Lense-Thirring effect, marginally stable orbits, and accretion processes around black holes.

To display the results, I used the visualization software Paraview, which simplifies the processing of such data with files in the appropriate format (VTK in my case).

Once the calculations are completed and stored in an array, I write it to a file and simply open it in Paraview.

For the example, I simulated a basic orbit using my code with a reasonable time step and a relatively small number of iterations to have a relatively good visibility for interpreting the results: In this simulation, we can clearly see the

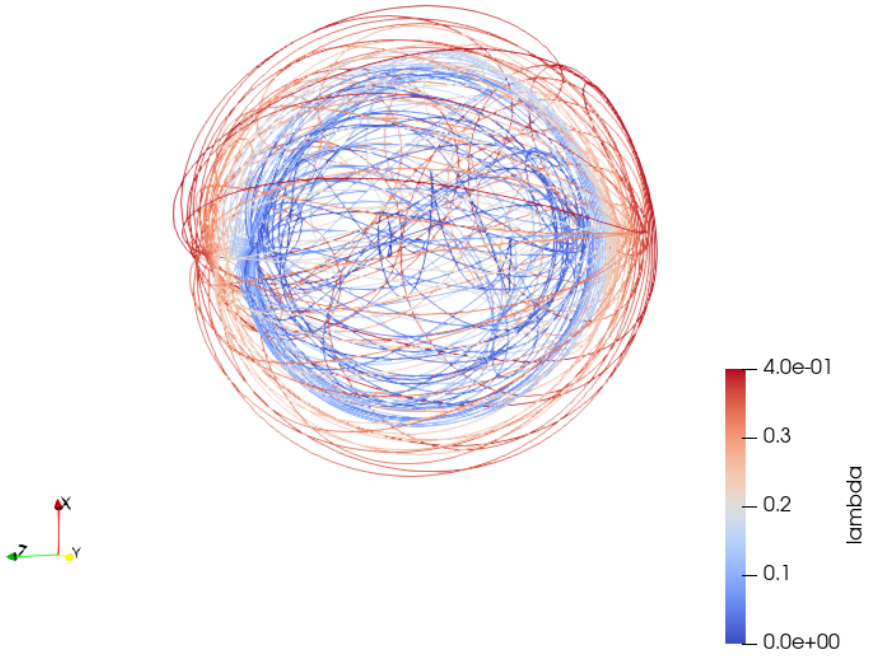


Figure 1: *Parameters:* $a = 0.4, M = 1.0 \times 10^{21}$ *and* $\lambda = 2.1$

precession of orbits due to the Lense-Thirring effect, which is evident in the spiral trajectories of particles, revealing the impact of the black hole's rotation on the curvature of the surrounding spacetime.

Furthermore, the phenomenon of particle captured by the event horizon is highlighted by geodesics that irreversibly spiral towards the center, disappearing beyond a certain critical limit.

The striking asymmetry and irregularity in the distribution of geodesics testify to the chaotic and nonlinear nature of particle motion subjected to intense tidal forces near the Kerr black hole.

However, it is possible to go further in the number of iterations to highlight the geometry of Kerr spacetime: By zooming in slightly, we can partly see the

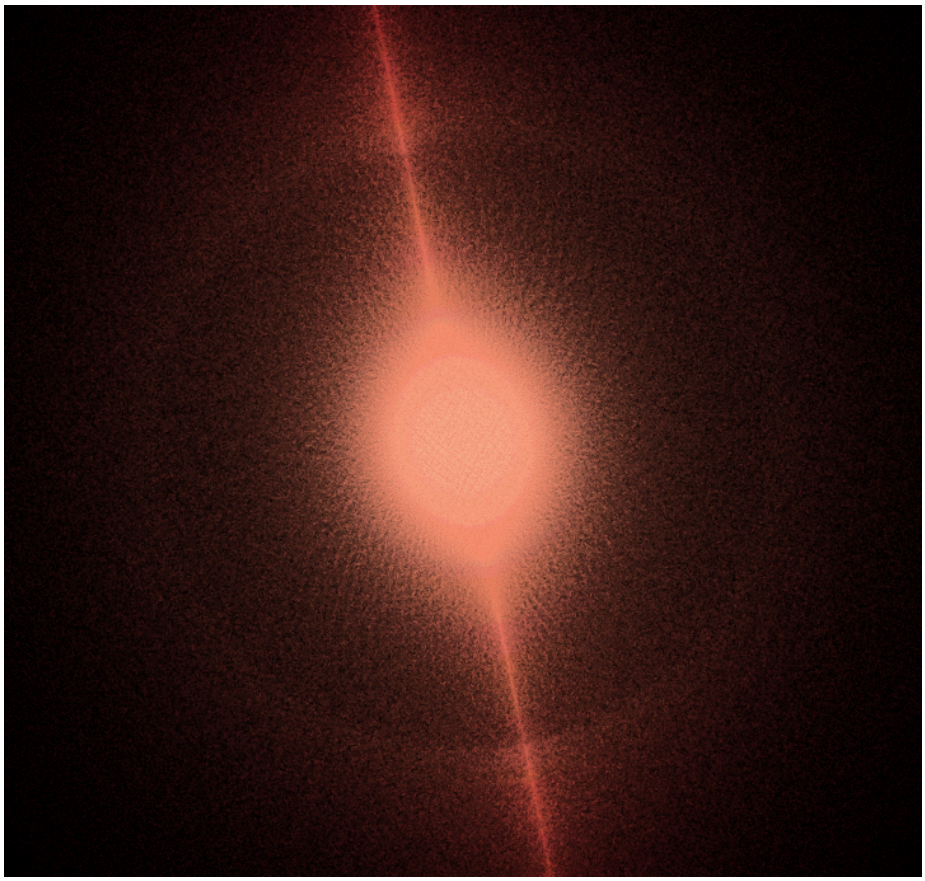

Figure 2: *Inner Horizon:* $a = 0.9, M = 1.989 \times 10^{21}$ *and* $\lambda = 40.0$

inner horizon of the black hole. This inner horizon, called the Cauchy horizon, is a null surface, just like the outer event horizon located at a radius $r_+$:

$$r_+ = M + \sqrt{M^2 - a^2} \tag{5.1}$$

Unlike the outer horizon, which acts as a one-way membrane through which particles and light can enter but not exit, the inner horizon possesses more exotic properties. It is considered a curvature singularity, beyond which causality is violated and determinism is broken.

It is important to note that for a physically realistic Kerr black hole, the rotation parameter is constrained by the relation $|a| \leq M$. In the limiting case where $|a| = M$, called an extremal black hole, the inner and outer horizons coincide at $r = M$. If $|a| > M$, the Kerr metric describes a hypothetical object called a naked singularity, which is not believed to exist in the real universe.

However, by zooming in further, we can highlight the inner surface of infinite redshift:

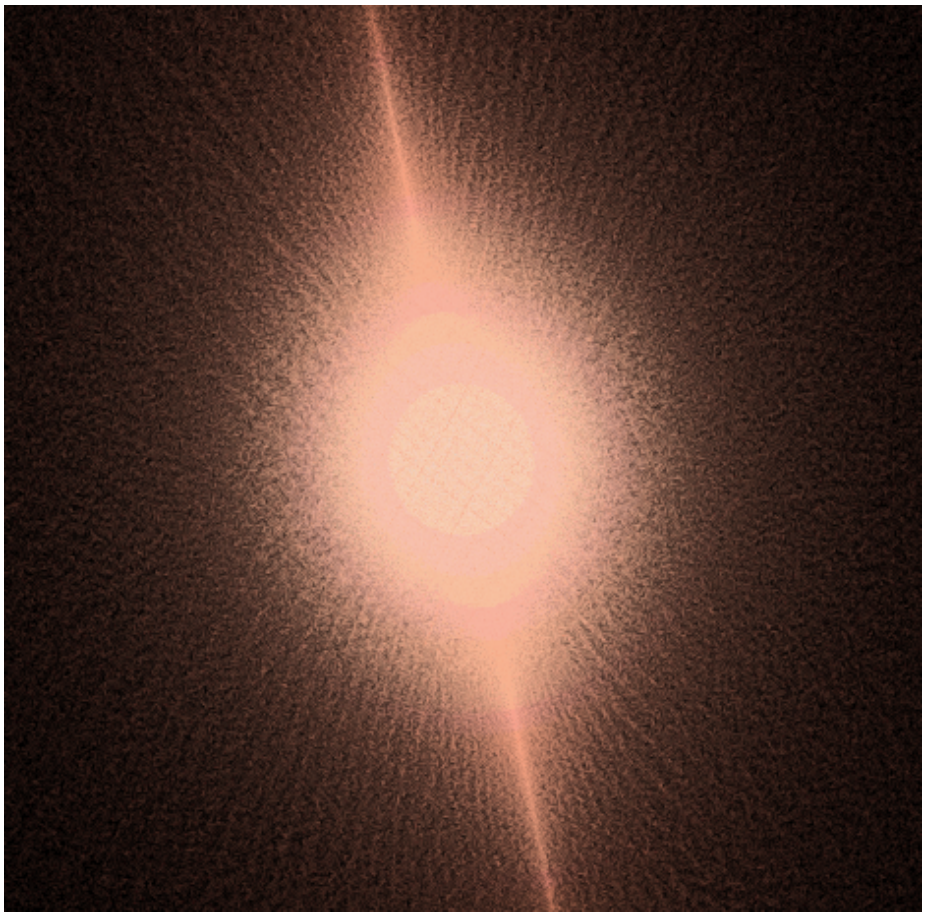

Figure 3: *Infinite Redshift Surface:* $a = 0.9, M = 1.0$ *and* $\lambda = 40.0$

The surface of infinite redshift (also known as the critical shift surface or inner ergospheric surface) is a boundary located inside the ergosphere, where the redshift factor becomes infinite. This surface is defined by the following equation:

$$r_{rs} = M - \sqrt{M^2 - a^2 \cos^2 \theta} \tag{5.2}$$

Inside this surface, the curvature of spacetime becomes so extreme that the redshift of objects emitting light tends toward infinity, making them invisible to an external observer. Moreover, in this region, the rotation of spacetime becomes superluminal, meaning that even light is dragged in the direction of the black hole's rotation.

The equation $r_{rs} = M - \sqrt{M^2 - a^2 \cos^2 \theta}$ thus captures a fundamental aspect of Kerr geometry and highlights the complexity of general relativity under

these extreme conditions.

To delve deeper, I had fun zooming in near the ring singularity to see what the geodesics might do in the presence of this singularity: The complex and

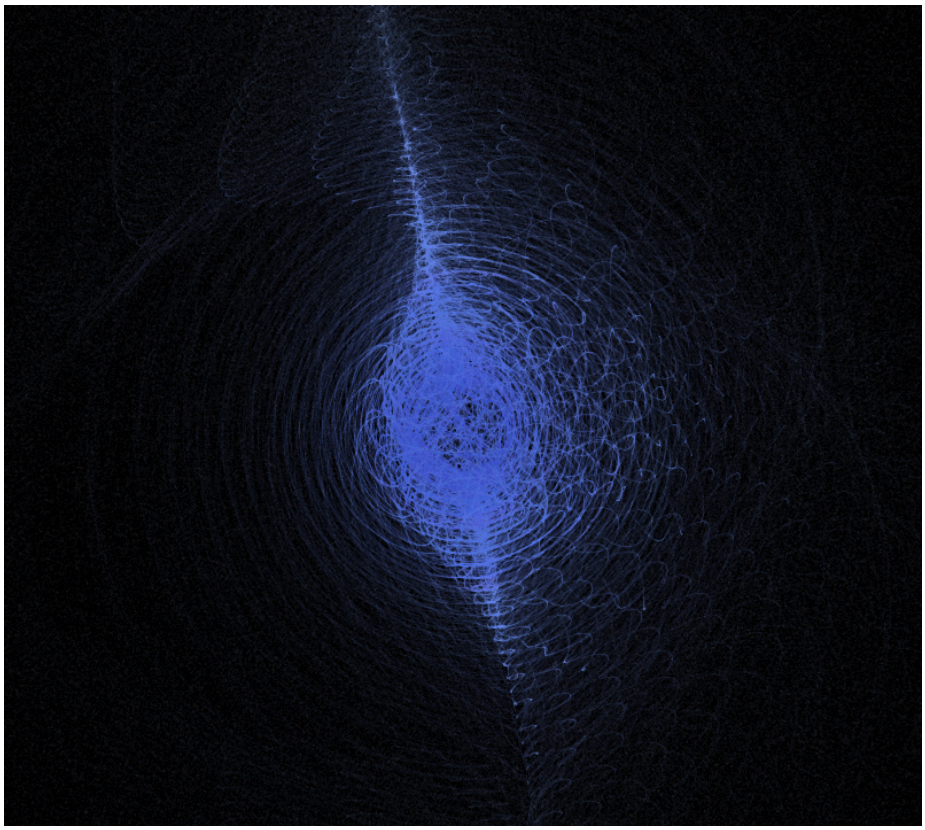

Figure 4: *Kerr Singularity:* $a = 0.9, M = 1.989 \times 10^{21}$ *and* $\lambda = 40.0$

swirling structure of the geodesics reveals the extreme gravitational effects and the distortion of spacetime in the vicinity of the singularity.

At the center of the image, a region of deep and intense blue can be distinguished, which seems to represent the singularity itself. The concentration of gravitational field lines around this region suggests an extreme curvature of spacetime, where density and tidal forces tend toward infinity.

Emanating from the singularity, we observe a vortex-like structure composed of interwoven luminous filaments. These filaments likely represent the geodesics of particles and light rays that are strongly deflected and distorted by the intense gravitational field of the black hole. Their tangled and chaotic appearance testifies to the complexity of trajectories in this extreme region of spacetime.

This ring singularity is located in the equatorial plane of the black hole, at a radial coordinate r given by:

$$r_{sing} = 0 \tag{5.3}$$

However, this singularity is not a true point in spacetime, as it cannot be reached by any timelike or lightlike geodesic originating from outside the black hole. Indeed, in Boyer-Lindquist coordinates, the true nature of the singularity is somewhat obscured.

In my simulation, the geodesics reach it although the visual representation of geodesics near the singularity can be misleading. The singularity itself is not a true point in spacetime, but rather a theoretical limit. The way geodesics are traced near this limit may give the impression that they pass through it, although mathematically speaking, this is not the case.

# 6 References

Xiaoling Yang, 2009. YNOGK : *A new public code for calculating null geodesics in the kerr spacetime.* https://arxiv.org/abs/1305.1250

Piotr T. Chrusciel, 2018 revised. *Geometry of black holes.* University of Vienna. https://homepage.univie.ac.at/

Susan Larsen, 2023, *Chapter 4: Christoffel Symbols, Geodesic Equations and Killing Vectors*, https://physicssusan.mono.net/

Tristan OZUCH-MEERSSEMAN, *Introduction au domaine de recherche Singularités en analyse géométrique : Flot de Ricci et Équations d'Einstein*, ens

A. DIMAKIS and F. MULLER-HOISSEN, *Discrete Riemannian Geometry* Max-Planck-Institut fur Stromungsforschung, Bunsenstrasse 10, D-37073 Gottingen, Germany, https://arxiv.org/pdf/gr-qc/9808023

T.Stillfjord *SRKCD: a stabilized Runge-Kutta method for stochastic optimization*, https://arxiv.org/abs/2201.12782

D.Doehring *Many-Stage Optimal Stabilized Runge-Kutta Methods for Hyperbolic Partial Differential Equations*, https://arxiv.org/abs/2402.12140

G.Panou *Geodesic equations and their numerical solutions in geodetic and Cartesian coordinates on an oblate spheroid* , https://arxiv.org/pdf/1612.01357