GRAVITATIONAL LENSING

Bending of Light

Umang Barthwal
2018B5A70672G

Under Guidance of
Dr. Prasanta Kumar Das
Department of Physics
BITS Pilani, Goa Campus

April 30, 2020

April 30, 2020

1 Introduction

The relativistic theory of gravity paved its way to the ground in 1916, when Prof. Albert Einstein published his paper on the general theory of relativity (Einstein 1916). This elegant and rigorous framework was a generalized version of gravity-free theory special relativity, which he published earlier in 1905. Even after almost a hundred years after Einstein wrote down the equations of General Relativity, solutions of the Einstein field equations remain extremely difficult to find beyond those which exhibit significant symmetries.

We can study the behavior of solutions under a high degree of symmetry considerations and could even solve analytically for highly unrealistic systems. In case of problems relevant to astrophysical and gravitational physics research, it still remains a big daunting question on how to get around the problem of solving these field equations. This question is so profound that it has a separate profound field of research which goes with the name of numerical relativity which attempts to use computation science and programming to numerically obtain solutions of the equations for some turbulent region where most of the interesting dynamics are happening.

Numerical methods have been used to solve the Einstein equation for many decades, but the past decade has seen tremendous advances and adding to that, the interest of the community grew when the field found applications in areas of radio astronomy, cosmology, signal processing, data mining. The field of numerical relativity is growing rampantly with the vast literature on algorithms, numerical methods and theoretical formulations with the development of robust and involved frameworks that provide a complete programming ecosystem and have proved to be essential tools for any numerical relativity researcher.

In this project, we used python libraries to provide us set of tools which can make symbolical and numerical computations for solving Einstein's field equations. Those libraries are used to calculate Christoffel Symbols, Riemann Tensor, Ricci Tensor, Ricci Scalar, Geodesics Equations symbolically for a given metric tensor and the trajectory of particles for different space-time metrics through numerical calculation. We simulated the numerical values of positional vectors as well as velocity vectors of a test-particle in the distorted space-time due to the massive body. Finally, derived the formula for Angle of Deviation of light due to the distorted space-time.

2 Calculating Geodesics

The following are some useful quantities we have used in our project using Gravipy package :

2.1 Metric Tensor

Metric Tensor describes the differential length elements required to measure distance in a curved(also applies to flat) space-time. It is a second order tensor and is fundamental to describe any space-time geometry. The *MetricTensor* class, inherits all its functions and also have some functions and constraints of its own. For example, metric tensor is bound to be a second order tensor.

2.2 ChristoffelSymbols

The Christoffel class that represents Christoffel symbols of the first and second kind. (Note that the Christoffel symbols are not tensors) Components of the Christoffel objects are computed from the below formula :

$$\Gamma_{\rho\mu\nu} = g_{\rho\sigma}\Gamma^{\sigma}{}_{\mu\nu} = \frac{1}{2}(g_{\rho\mu,\nu} + g_{\rho\nu,\mu} - g_{\mu\nu,\rho})$$

2.3 RiemannCurvatureTensor

This quantity is the most extensive measure of curvature of a space-time. It supports all the functions of its parent class.

$$R_{\mu\nu\rho\sigma} = \frac{\partial\Gamma_{\mu\nu\sigma}}{\partial x^{\rho}} - \frac{\partial\Gamma_{\mu\nu\rho}}{\partial x^{\sigma}} + \Gamma^{\alpha}{}_{\nu\sigma}\Gamma_{\mu\rho\alpha} - \Gamma^{\alpha}{}_{\nu\rho}\Gamma_{\mu\sigma\alpha} - \frac{\partial g_{\mu\alpha}}{\partial x^{\rho}}\Gamma^{\alpha}{}_{\nu\sigma} + \frac{\partial g_{\mu\alpha}}{\partial x^{\sigma}}\Gamma^{\alpha}{}_{\nu\rho}$$

2.4 RicciTensor

This tensor is used in solving Eintein's equation and basically contraction of Riemann Curvature Tensor.

$$R_{\mu\nu} = \frac{\partial\Gamma^{\sigma}{}_{\mu\nu}}{\partial x^{\sigma}} - \frac{\partial\Gamma^{\sigma}{}_{\mu\sigma}}{\partial x^{\nu}} + \Gamma^{\sigma}{}_{\mu\nu}\Gamma^{\rho}{}_{\sigma\rho} - \Gamma^{\rho}{}_{\mu\sigma}\Gamma^{\sigma}{}_{\nu\rho}$$

2.5 RicciScalar

It is a scalar quantity obtained by contracting a given Ricci Tensor. Being a scalar quantity, it does not support change config unlike other tensors as discussed above.

2.6 Geodesics

Calculating Geodesic equaitons for Schwarzschild Metric:
The Schwarzschilld line element is given by :

$$ds^2 = (\frac{2GM}{r} - 1)dt^2 + (\frac{1}{\frac{-2GM}{r} + 1})dr^2 + r^2 d\theta^2 + r^2 \sin^2\theta d\phi^2$$

(we have taken c=1)

Thus the metric tensor g_{ij} is given by:

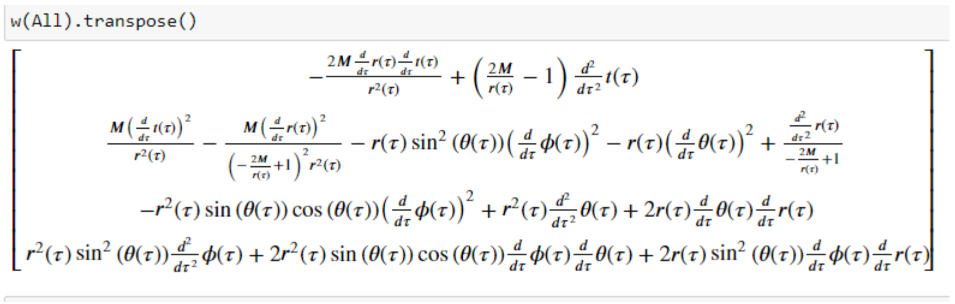
$$g_{ij} = \begin{pmatrix} (\frac{2MG}{r} - 1) & 0 & 0 & 0 \\ 0 & (\frac{1}{\frac{-2MG}{r}+1}) & 0 & 0 \\ 0 & 0 & r^2 & 0 \\ 0 & 0 & 0 & r^2 \sin^2\theta \end{pmatrix}$$

The geodesic equation is calculated from

$$\frac{\mathrm{d}^2 x^\mu}{\mathrm{d}s^2} + \Gamma^\mu_{\alpha\beta} \frac{\mathrm{d}x^\alpha}{\mathrm{d}s} \frac{\mathrm{d}x^\beta}{\mathrm{d}s} = 0$$

Where $\Gamma^\mu_{\alpha\beta}$ is the Christoffel Connection

The following are the geodesic equations for the t,r,θ,ϕ coordinates respectively.
(note : These quantities we calculated using our python program)



```
w(All).transpose()
```

$$\begin{bmatrix} -\frac{2M \frac{d}{d\tau} r(\tau) \frac{d}{d\tau} t(\tau)}{r^2(\tau)} + \left(\frac{2M}{r(\tau)} - 1\right) \frac{d^2}{d\tau^2} t(\tau) \\ \frac{M\left(\frac{d}{d\tau} t(\tau)\right)^2}{r^2(\tau)} - \frac{M\left(\frac{d}{d\tau} r(\tau)\right)^2}{\left(-\frac{2M}{r(\tau)}+1\right)^2 r^2(\tau)} - r(\tau)\sin^2(\theta(\tau))\left(\frac{d}{d\tau}\phi(\tau)\right)^2 - r(\tau)\left(\frac{d}{d\tau}\theta(\tau)\right)^2 + \frac{\frac{d^2}{d\tau^2} r(\tau)}{-\frac{2M}{r(\tau)}+1} \\ -r^2(\tau)\sin(\theta(\tau))\cos(\theta(\tau))\left(\frac{d}{d\tau}\phi(\tau)\right)^2 + r^2(\tau)\frac{d^2}{d\tau^2}\theta(\tau) + 2r(\tau)\frac{d}{d\tau}\theta(\tau)\frac{d}{d\tau}r(\tau) \\ r^2(\tau)\sin^2(\theta(\tau))\frac{d^2}{d\tau^2}\phi(\tau) + 2r^2(\tau)\sin(\theta(\tau))\cos(\theta(\tau))\frac{d}{d\tau}\phi(\tau)\frac{d}{d\tau}\theta(\tau) + 2r(\tau)\sin^2(\theta(\tau))\frac{d}{d\tau}\phi(\tau)\frac{d}{d\tau}r(\tau) \end{bmatrix}$$

The following are the Christoffel symbols.

$$\Gamma^r_{ij} = \begin{pmatrix} \frac{GM}{r^2}(1-\frac{2MG}{r}) & 0 & 0 & 0 \\ 0 & \frac{-GM}{r^2}(\frac{1}{\frac{-2MG}{r}+1}) & 0 & 0 \\ 0 & 0 & -r(1-\frac{2mG}{r}) & 0 \\ 0 & 0 & 0 & -r\sin^2\theta(1-\frac{2MG}{r}) \end{pmatrix}$$

$$\Gamma^t_{ij} = \begin{pmatrix} 0 & \frac{GM}{r^2}(\frac{1}{\frac{-2MG}{r}+1}) & 0 & 0 \\ \frac{GM}{r^2}(\frac{1}{\frac{-2MG}{r}+1}) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\Gamma^\theta_{ij} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{r} \\ 0 & 0 & 0 & \cot\theta \\ 0 & \frac{1}{r} & \cot\theta & 0 \end{pmatrix}$$

$$\Gamma^\phi_{ij} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{r} & 0 \\ 0 & \frac{1}{r} & 0 & 0 \\ 0 & 0 & 0 & -\sin\cos\theta \end{pmatrix}$$

3 Plotting the Geodesics

In our project, we used the python library EinsteinPy for numerical calculation of geodesics of a particle travelling in the distorted space-time. The EinsteinPy package provides some core data-types for calculating and operating on geodesics and black-holes. The most important data types for EinsteinPy are Body and Geodesic objects that support defining and creating new bodies, storing and calculating geodesic for bodies, and their handling. The main purpose of these core classes is to standardize the way calculations for various kinds of heavy cosmic bodies and black-holes. The classes always maintains the consistency in units of measurement wherever applicable.

3.1 Body

The Body data type helps define the central black-hole or the heavy massed object and the objects revolving around it, in our case i.e. source and test-particle respectively.

3.2 Metric

This module captures all the geometric of specific space-times, and being able to calculate trajectories in a given space-time. The module derives its name from Metric Tensor, a tensorial quantity used to describe differential lengths, especially in a curved space-time.

3.2.1 Schwarzschild Metric

During 1916, Karl Schwarzschild pioneered the analysis of the relation between the size of black hole and its mass and this work paved the way for the first exact solution of Einstein's Field Equation under the limited case of single spherical non- rotating, non-charged body.As r , the metric defined above starts to approximate flat Minkowski space-time, thus showing that Schwarzschild geometry is asymptotically flat. The metric equation corresponding to a space-time centred around a body in the above described conditions is given by,

$ds^2 = c^2(1 - \frac{2GM}{c^2 r})dt^2 - (1 - \frac{2GM}{c^2 r})^{-1}dr^2 - r^2 d\theta^2 - r^2 \sin^2\theta d\phi^2$

3.2.2 Kerr Metric

Kerr metric is a further generalization of Schwarzschild metric. "Kerr" space-time defined as a massive central body posses an angular momentum. The corresponding metric equation which defines the space-time is,

$$g_{\mu\nu} = \begin{bmatrix} -(1-\frac{r_s r}{\Sigma})c^2 & 0 & 0 & -\frac{r_s r a sin^2\theta}{\Sigma}c \\ 0 & \frac{\Sigma}{\Delta} & 0 & 0 \\ 0 & 0 & \Sigma & 0 \\ -\frac{r_s r a sin^2\theta}{\Sigma}c & 0 & 0 & (r^2+a^2+\frac{r_s r a^2}{\Sigma}sin^2\theta)sin^2\theta \end{bmatrix}$$

$$ds^2 = g_{\mu\nu}dx^\mu dx^\nu$$

$$a = \frac{J}{Mc},$$
$$\Sigma = r^2 + a^2 cos^2, \theta$$
$$\Delta = r^2 - r_s r + a^2$$

3.2.3 Kerr-Newman Metric

Kerr-Newman metric is a further generalization of Kerr metric. It is the single-body vacuum solutions of Einstein Field Equation. The space-time defined as a massive central body with rotation and charge.

$$-ds^2 = c^2 d\tau^2 = -(\frac{dr^2}{\Delta} + d\theta^2)\rho^2 + (cdt - a sin^2\theta d\phi)^2 \frac{\Delta}{\rho^2} - ((r^2+a^2)d\phi - acdt)^2 \frac{sin^2\theta}{\rho^2}$$

$$a = \frac{J}{Mc},$$
$$\Sigma = r^2 + a^2 cos^2\theta,$$
$$\Delta = r^2 - r_s r + a^2 + r_Q^2,$$
$$r_Q^2 = \frac{Q^2 G}{4\pi\epsilon_0 c^4}$$

3.3 Coordinates

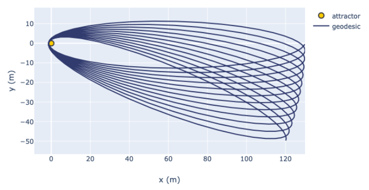
Coordinates sub-package provides support for representing and transforming different coordinate systems. Cartesian, Spherical and BoyerLindquist coordinate system are present under Coordinates.

3.4 Geodesic

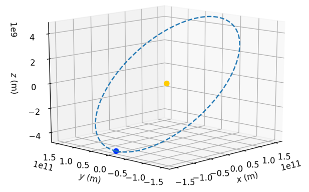
The Geodesic data type use the object of Body data type and defined type of metric tensor to calculate the position vectors and velocity vectors.

3.5 Plots

In the following figures, the massive object at the origin distorts the space-time around it. The blue-dotted line represents the geodesic path followed by a test-particle of negligible mass. By changing initial positional vectors, velocity vectors, mass of source, spin factor of source, charge of source and test-particle etc, we can simulate the values to get the trajectory.

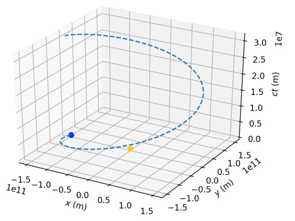


(a) 2-D Plot

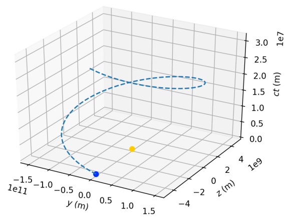


(b) 3-D Plot

Figure 1: Geodesic Trajectories for different initial conditions



(a) x-y-t Plot



(b) y-z-t Plot

Figure 2: Evolution of x-y and y-z with t

3.6 Unit Handling

EinsteinPy uses the astropy module to handle the units of different types of scientific quantities.All functions and objects must have their input constrained to the appropriate type of unit (e.g., length, mass or energy). Inputs can then be provided with any units that match the required type (e.g., millimeters and inches are both valid units for length) and conversions occur automatically without user intervention.

4 Bending of Light in Schwarzschild Space-Time

4.1 Schwarzschild Metric

As discussed in section 4, Schwarzschild Metric is the unique spherically symmetric vacuum solution of Einstein's Field Equation.

$$ds^2 = c^2(1 - \frac{2GM}{c^2 r})dt^2 - (1 - \frac{2GM}{c^2 r})^{-1}dr^2 - r^2 d\theta^2 - r^2 \sin^2\theta d\phi^2$$

4.2 Motion of any General Particle

The paths of particles with mass moving in the vicinity of a spherical massive object are given by the time-like geodesics of space-time.

$$\frac{d}{d\tau}\left(\frac{\partial L}{\partial \dot{x}^\mu}\right) - \frac{\partial L}{\partial x^\mu} = 0$$

where

$$\begin{aligned} L(x^\sigma, \dot{x}^\sigma) &= \frac{1}{2} g_{\mu\nu}\dot{x}^\mu \dot{x}^\nu \\ &= \frac{1}{2}\left(c^2(1 - \frac{2GM}{c^2 r})\dot{t}^2 - (1 - \frac{2GM}{c^2 r})^{-1}\dot{r}^2 - r^2\dot{\theta}^2 - r^2\sin^2\theta\dot{\phi}^2\right) \end{aligned}$$

Here dots denote derivatives with respect to τ *and* $x^\mu = (t, r, \theta, \phi)$.

Assume the particle is moving only in the "equatorial plane" given by $\theta = \pi/2$.Therefore, the geodesic equations are the following:

$$\left(1 - \frac{2m}{r}\right)^{-1}\ddot{r} + \frac{mc^2}{r^2}\dot{t}^2 - \left(1 - \frac{2m}{r}\right)^{-2}\frac{m}{r^2}\dot{r}^2 - r\dot{\phi}^2 = 0 \qquad (1)$$

$$\left(1 - \frac{2m}{r}\right)\dot{t} = k \qquad (2)$$

$$r^2\dot{\phi} = h \qquad (3)$$

where $m = \frac{GM}{c^2}$; k and h are arbitrary constants.
(Note: t and ϕ are cyclic coordinates)

We know Schwarzschild line element is

$$ds^2 = c^2 d\tau^2 = c^2(1 - \frac{2m}{r})dt^2 - (1 - \frac{2m}{r})^{-1}dr^2 - r^2 d\theta^2 - r^2 \sin^2\theta d\phi^2$$

$$\Rightarrow c^2 = c^2(1 - \frac{2m}{r})\dot{t^2} - (1 - \frac{2m}{r})^{-1}\dot{r^2} - r^2\dot{\phi}^2$$

$$\Rightarrow \frac{c^2}{\dot{\phi^2}} = c^2(1 - \frac{2m}{r})\frac{\dot{t^2}}{\dot{\phi^2}} - (1 - \frac{2m}{r})^{-1}\frac{\dot{r^2}}{\dot{\phi^2}} - r^2 \tag{4}$$

The equation (2) gives the relation between the coordinate time t and the proper time τ; equation (3) is clearly analogous to the equation of conservation of angular momentum. On substituting equation (2) and (3) in (4), we get

$$(\frac{dr}{d\phi}) + r^2(1 + \frac{c^2r^2}{h^2})(1 - \frac{2m}{r}) - \frac{c^2k^2r^4}{h^2} = 0 \tag{5}$$

Put u=1/r and $m = GM/c^2$ in the equation (5), we get :

$$\left(\frac{du}{d\phi}\right)^2 + u^2 = E + \frac{2GM}{h^2}u + \frac{2GM}{c^2}u^3 \tag{6}$$

where $E = \frac{c^2(k^2-1)}{h^2}$

4.3 Null Geodesics

Null Geodesics gives the paths of photons and any other particles having rest mass equal to zero. Photons moving in the equatorial plane of the massive body $\theta = \pi/2$. For Null Geodesic

$$ds^2 = 0$$

$$\Rightarrow c^2(1 - \frac{2m}{r})\dot{t^2} - (1 - \frac{2m}{r})^{-1}\dot{r^2} - r^2\dot{\phi}^2 = 0 \tag{7}$$

Substituting the values of $\dot{t}$ *and* $\dot{\phi}$ from equations (2) and (3).

4.4 Trajectory of Photon in absence of Massive Object

The path of a photon travelling in the equatorial plane is given by equation (7). Put M=0 i.e. m=0 and u=1/r in the equation (7), we get :

$$\left(\frac{du}{d\phi}\right)^2 + u^2 = F \tag{8}$$

where, $F = \frac{c^2k^2}{h^2}$

A particular solution of which is

$$u = u_0 sin\phi \; or \; r_0 = r sin\phi$$

where, $u_0^2 \equiv 1/r_0^2 = F$

This solution represents the straight-line path taken by the photon originating from infinity in the direction $\phi = 0$, and going off to infinity in the direction $\phi = \pi$. The point on the path nearest to the origin O is at a distance r_0 from it, and is given by $\phi = \pi/2$.

4.5 Trajectory of Photon in presence of Massive Object

Substituting the values of $\dot{t} \; and \; \dot{\phi}$ from equations (2) and (3) to equation (7), and replacing $\overline{u} \equiv u/u_0$.

$$\left(\frac{d\overline{u}}{d\phi}\right)^2 + \overline{u}^2 = \frac{F}{u_0^2} + \epsilon\overline{u}^3 \tag{9}$$

where, $\epsilon \equiv \frac{2GMu_0}{c^2}$

We know at the point of closest approach (i.e. $u = u_0$), $\frac{d\overline{u}}{d\phi} = 0$ and $\overline{u} = 1$, then $F/u_0^2 = 1 - \epsilon$. Putting this back to equation (9), we get

$$\left(\frac{d\overline{u}}{d\phi}\right)^2 + \overline{u}^2 = 1 - \epsilon + \epsilon\overline{u}^3 \tag{10}$$

Lets assume the solution of the equation (10) of form $\overline{u} = sin\phi + \epsilon v$; where v is some function of ϕ. Substituting it back to equation (10) and neglecting higher order terms of ϵ, we get

$$\frac{d(vsec\phi)}{d\phi} = \frac{1}{2}\left(sec\phi tan\phi - sin\phi - sec^2\phi\right) \tag{11}$$

On further simplification, and integrating with respect to ϕ,

$$v = \frac{1}{2}\left(1 + cos^2\phi - sin\phi\right) + Acos\phi \tag{12}$$

where, A=Integration Constant

Assuming initial conditions as, photon originating from infinity in the direction $\phi = 0$. Then v=0 and A=-1, and we get

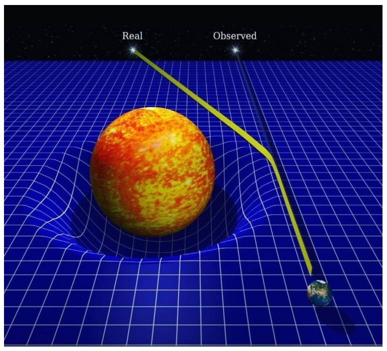
$$\overline{u} = \left(1 - \frac{\epsilon}{2}\right) sin\phi + \frac{1}{2}\epsilon(1 - cos\phi)^2 \tag{13}$$

This is the equation of the path of the photon, to the first order in ϵ.

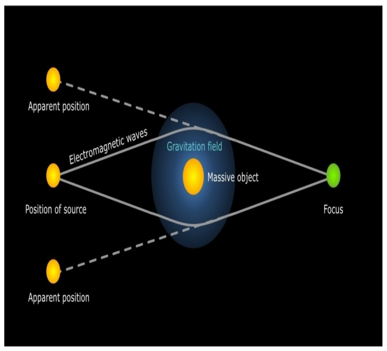
4.6 Angle of Deflection of Light

We no longer expect the photon to go off to infinity in the direction π, but in a direction $\pi + \alpha$, where α is small. Putting $\overline{u} = 0$ and $\phi = \pi + \alpha$ in the eqaution (13), and ignoring the higher power terms of *alpha* and $\epsilon\alpha$, we get

$$0 = \alpha + 2\epsilon \Rightarrow \alpha = 2\epsilon \Rightarrow \alpha = \frac{4GM}{r_0 c^2} \tag{14}$$



(a) Object distorting space-time



(b) Actual vs Apparent Position

Figure 3: Bending of Light

As we can see from the equation, the angle of deflection increases as the impact parameter r_0 decreases.

A APPENDIX

A.1 Code for Calculating Geodesics

```python
import sympy
import gravipy
from sympy import *
from gravipy.tensorial import *
from sympy import init_printing
import inspect
init_printing()

x,y,z = symbols('x,y,z')
t, r, theta, phi, M = symbols('t, r, \\theta, \\phi, M')
# M = mass of matter

x = Coordinates('\chi', [t, r, theta, phi])

# input the line elements Row-wise

Metric = Matrix([[-(1-2*M/r),0,0,0],[0,1/(1-2*M/r),0,0], [0,0,r
    **2,0], [0,0,0,r**2*sin(theta)**2]])


g = MetricTensor('g', x, Metric)

#print("# Christoffel\n")
Ga = Christoffel('Ga', g)

#print("# Riemann\n")
Rm = Riemann('Rm', g)

#print("# Ricci tensor\n")
Ri= Ricci('Ri',g)

#print("# Einstein Tensor\n")
G = Einstein('G', Ri)

tau = Symbol('\\tau')
w = Geodesic('w', g, tau)

x(-All)

g(All,All)

Ga(All,All,All)

Rm(All,All,All,All)

Ri(All,All)

Ri.scalar()

w(All).transpose()
```

A.2 Code for Plotting Geodesic Trajectory

```

import numpy as np
import astropy.units as u

from plotly.offline import init_notebook_mode

from einsteinpy.plotting import GeodesicPlotter
from einsteinpy.coordinates import BoyerLindquistDifferential
from einsteinpy.bodies import Body
from einsteinpy.geodesic import Geodesic
from einsteinpy.metric.kerrnewman import KerrNewman

init_notebook_mode(connected=True)

# Example: Simulating Earth

## Source ##

m = 2e30 # kg

Q = 0   # Charge on the massive body


spin_factor = 0 # spin_factor = J/(Mc) (metres)

## Test Particle ##

q0 = 0   # Charge per unit mass of test particle (check this part)

# initial position cordinates

r = 1.47e11                      # (metres)

theta = np.pi * 0.51             # (radians)

phi = np.pi                      # (radians)

# initial velocity vectors at initial position

Vr = 0                           # (m/s)

Vtheta = 0                       # (rad/s)

Vphi = 30.29e3/1.47e11           # (rad/s)


# Source

Attractor = Body(name="attractor", mass = m * u.kg, R=0 * u.m,
    differential = None, a = spin_factor * u.m, q = Q * u.C, parent
    =None)

# Test Object initial position (in spherical coordinates) & initial
     velocity
sph_obj = BoyerLindquistDifferential(r * u.m, theta * u.rad, phi *
    u.rad, Vr * u.m/u.s, Vtheta * u.rad/u.s, Vphi * u.rad/u.s,
    spin_factor * u.m)

####sph_obj = SphericalDifferential(r*u.m, theta*u.rad, phi*u.rad,
    Vr*u.m/u.s, Vtheta*u.rad/u.s, Vphi*u.rad/u.s)
Object = Body(name="testparticle", mass = 0 * u.kg, R = 0 * u.m,
    differential = sph_obj, a = 0 * u.m, q = q0 * (u.C/u.kg),
    parent=Attractor)

# geodesic simulation
# body = Test particle for which geodesics is to be calculated
# time = time of start
# start_lambda = 0.0 s (Default)
# end_lambda = Lambda(proper time in seconds) where iterations will
     stop
# step_size = Size of each increment in proper time

geodesic = Geodesic(body = Object, time=0 * u.s, end_lambda= ((1 *
    u.year).to(u.s)).value, step_size=((50 * u.min).to(u.s)).value,
     metric=KerrNewman)


import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib notebook
vals = geodesic.trajectory

## Each element here is w.r.t. proper-time (lambda) ##

t = np.array(vals[:, 0])
x = np.array(vals[:, 1])
y = np.array(vals[:, 2])
z = np.array(vals[:, 3])

vt = np.array(vals[:, 4])
vx = np.array(vals[:, 5])
vy = np.array(vals[:, 6])
vz = np.array(vals[:, 7])

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.set_zlabel("$z$ (m)")
ax.set_xlabel("$x$ (m)")
ax.set_ylabel("$y$ (m)")
ax.plot3D([x[0]], [y[0]], [z[0]], "o", color="#0033ff")
ax.plot3D([0], [0], [0], "o", color="#ffcc00")
ax.plot3D(x, y, [0], "--")

plt.show()

from einsteinpy.plotting import StaticGeodesicPlotter
%matplotlib notebook
obj = StaticGeodesicPlotter()
obj.animate(geodesic, interval=1)
obj.show()

fig = plt.figure()
```

```
ax = fig.add_subplot(111, projection='3d')
ax.set_zlabel("$ct$ (m)")
ax.set_xlabel("$x$ (m)")
ax.set_ylabel("$y$ (m)")
ax.plot3D([x[0]], [y[0]], [t[0]], "o", color="#0033ff")
ax.plot3D([0], [0], [0], "o", color="#ffcc00")
ax.plot3D(x, y, t, "--")


plt.show()


fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.set_zlabel("$ct$ (m)")
ax.set_xlabel("$y$ (m)")
ax.set_ylabel("$z$ (m)")
ax.plot3D([y[0]], [z[0]], [t[0]], "o", color="#0033ff")
ax.plot3D([0], [0], [0], "o", color="#ffcc00")
ax.plot3D(y, z, t, "--")


plt.show()


fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.set_zlabel("$ct$ (m)")
ax.set_xlabel("$z$ (m)")
ax.set_ylabel("$x$ (m)")
ax.plot3D([z[0]], [x[0]], [t[0]], "o", color="#0033ff")
ax.plot3D([0], [0], [0], "o", color="#ffcc00")
ax.plot3D(z, x, t, "--")


plt.show()
```