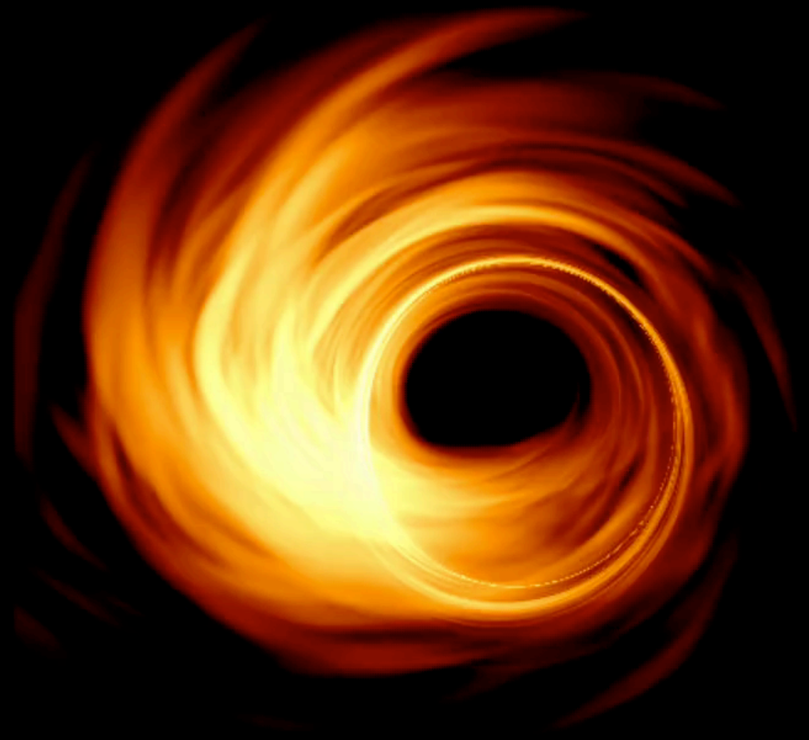


BLACK HOLES AND GRAVITATIONAL THEORIES BEYOND GENERAL RELATIVITY RESEARCH USING PYTHON

Eduard Larrañaga

Observatorio Astronómico Nacional
Facultad de Ciencias
Universidad Nacional de Colombia

General Relativity

Black Holes

Relativistic Celestial Mechanics

Relativistic Astrophysics

General Relativity

Black Holes

Relativistic Celestial Mechanics

Relativistic Astrophysics

- Theoretical relativity implies long (veeeery loooong) and tedious mathematical calculations.

General Relativity

Black Holes

Relativistic Celestial Mechanics

Relativistic Astrophysics

- Theoretical relativity implies long (veeeery loooong) and tedious mathematical calculations.
- Numerical relativity usually demands a heavy programming experience.

General Relativity

Black Holes

Relativistic Celestial Mechanics

Relativistic Astrophysics

- Theoretical relativity implies long (veeeery loooong) and tedious mathematical calculations.
- Numerical relativity usually demands a heavy programming experience.
- Many of the people working in gravitational physics have a good theoretical background but, usually, have little or no programming experience (and many are not interested in learning!).

Symbolic Tensor Calculus

$g_{\mu\nu}$: Metric Tensor

$g_{\mu\nu}$: Metric Tensor

$g^{\mu\nu}$: Inverse Metric Tensor $n \times n$ components

$g_{\mu\nu}$: Metric Tensor

$g^{\mu\nu}$: Inverse Metric Tensor $n \times n$ components

$$\Gamma^{\alpha}_{\mu\nu} = \frac{1}{2} g^{\alpha\sigma} \left[\partial_{\mu} g_{\sigma\nu} + \partial_{\nu} g_{\mu\sigma} - \partial_{\sigma} g_{\mu\nu} \right]$$: Connections

$g_{\mu\nu}$: Metric Tensor

$g^{\mu\nu}$: Inverse Metric Tensor $n \times n$ components

$$\Gamma^{\alpha}_{\mu\nu} = \frac{1}{2} g^{\alpha\sigma} \left[\partial_{\mu} g_{\sigma\nu} + \partial_{\nu} g_{\mu\sigma} - \partial_{\sigma} g_{\mu\nu} \right]$$: Connections

$\frac{n^2(n+1)}{2}$ components

$$R^{\alpha}_{\mu\beta\nu} = \partial_{\beta}\Gamma^{\alpha}_{\mu\nu} - \partial_{\nu}\Gamma^{\alpha}_{\mu\beta} + \Gamma^{\alpha}_{\sigma\beta}\Gamma^{\sigma}_{\mu\nu} - \Gamma^{\alpha}_{\sigma\nu}\Gamma^{\sigma}_{\mu\beta} \quad : \text{ Riemann Tensor}$$

$$R^{\alpha}_{\mu\beta\nu} = \partial_{\beta}\Gamma^{\alpha}_{\mu\nu} - \partial_{\nu}\Gamma^{\alpha}_{\mu\beta} + \Gamma^{\alpha}_{\sigma\beta}\Gamma^{\sigma}_{\mu\nu} - \Gamma^{\alpha}_{\sigma\nu}\Gamma^{\sigma}_{\mu\beta}$$: Riemann Tensor

$$\frac{n^2(n^2-1)}{12}$$ components

$$R^{\alpha}_{\mu\beta\nu} = \partial_{\beta}\Gamma^{\alpha}_{\mu\nu} - \partial_{\nu}\Gamma^{\alpha}_{\mu\beta} + \Gamma^{\alpha}_{\sigma\beta}\Gamma^{\sigma}_{\mu\nu} - \Gamma^{\alpha}_{\sigma\nu}\Gamma^{\sigma}_{\mu\beta}$$: Riemann Tensor

$$\frac{n^2(n^2-1)}{12}$$ components

$$R_{\mu\nu} = R^{\alpha}_{\mu\alpha\nu} = g^{\alpha\beta}R_{\alpha\mu\beta\nu}$$: Ricci Tensor

$$R^{\alpha}_{\mu\beta\nu} = \partial_{\beta}\Gamma^{\alpha}_{\mu\nu} - \partial_{\nu}\Gamma^{\alpha}_{\mu\beta} + \Gamma^{\alpha}_{\sigma\beta}\Gamma^{\sigma}_{\mu\nu} - \Gamma^{\alpha}_{\sigma\nu}\Gamma^{\sigma}_{\mu\beta}$$: Riemann Tensor

$$\frac{n^2(n^2-1)}{12}$$ components

$$R_{\mu\nu} = R^{\alpha}_{\mu\alpha\nu} = g^{\alpha\beta}R_{\alpha\mu\beta\nu}$$: Ricci Tensor

$$\frac{n(n+1)}{2}$$ components


WOLFRAM


EinsteinPy


Maple™

In [1]:
```
import sympy
from sympy import cos, sin, sinh
from einsteinpy.symbolic import *

sympy.init_printing()
```

In [7]:
```
coords = sympy.symbols("t r theta phi")
G, M, c, a = sympy.symbols("G M c a")
# using metric values of schwarschild space-time
# a is schwarzschild radius
list2d = [[0 for i in range(4)] for i in range(4)]
list2d[0][0] = - (1 - (2*G*M /(c**2 * coords[1])))
list2d[1][1] = 1 / ((1 - (2*G*M / (c**2 * coords[1]))))
list2d[2][2] = (coords[1] ** 2)
list2d[3][3] = (coords[1] ** 2) * (sympy.sin(coords[2]) ** 2)
sch = MetricTensor(list2d, coords)
sch.tensor()
```

Out[7]:

$$\begin{bmatrix} \frac{2GM}{c^2 r} - 1 & 0 & 0 & 0 \\ 0 & \frac{1}{-\frac{2GM}{c^2 r}+1} & 0 & 0 \\ 0 & 0 & r^2 & 0 \\ 0 & 0 & 0 & r^2 \sin^2(\theta) \end{bmatrix}$$

In [8]:
```
sch_ch = ChristoffelSymbols.from_metric(sch)
sch_ch.tensor()
```

```
CPU times: user 337 ms, sys: 8.72 ms, total: 346 ms
Wall time: 357 ms
```

Out[8]:

$$\left[\begin{bmatrix} 0 & -\frac{GM}{c^2r^2\left(\frac{2GM}{c^2r}-1\right)} & 0 & 0 \\ -\frac{GM}{c^2r^2\left(\frac{2GM}{c^2r}-1\right)} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\right.$$

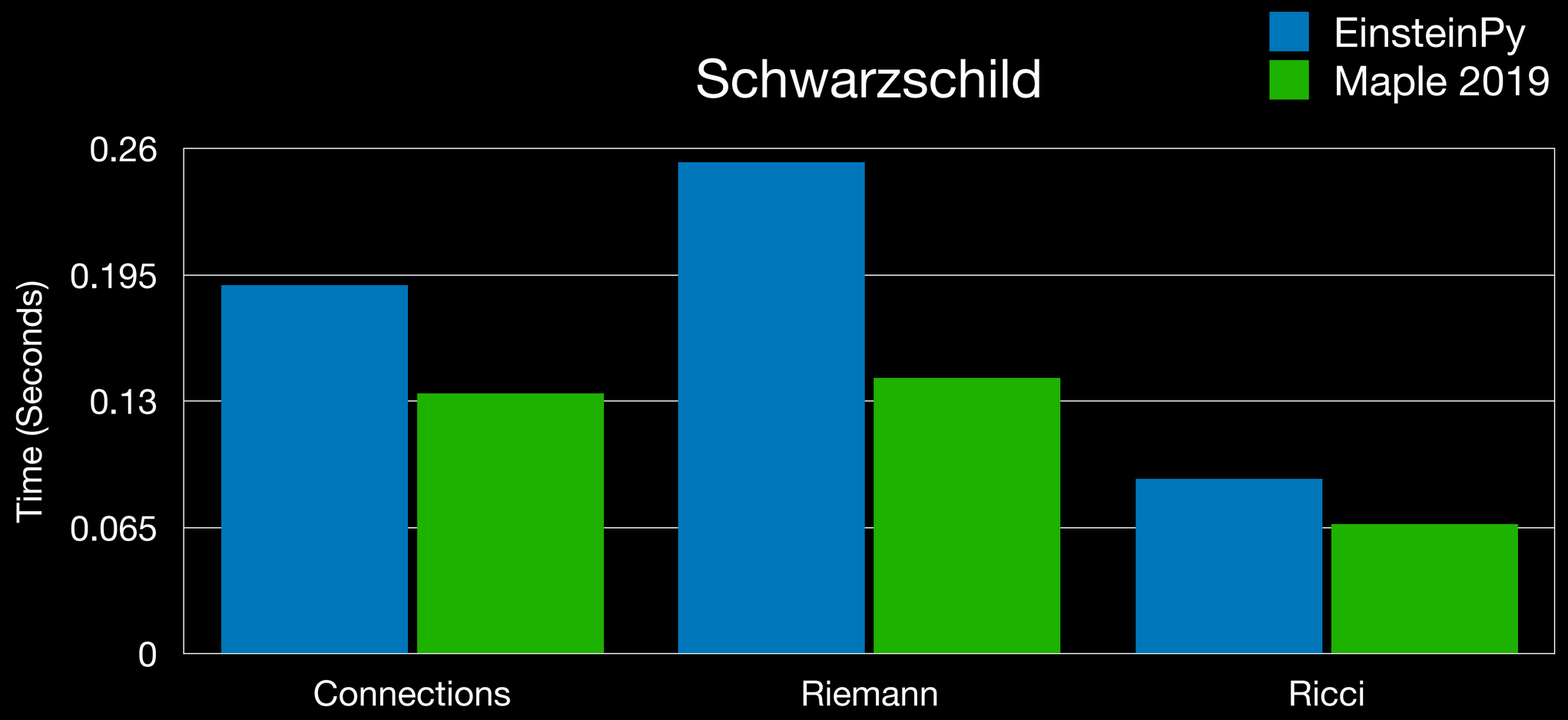
$$\left.\begin{bmatrix} \frac{2GM\left(-\frac{GM}{c^2r}+\frac{1}{2}\right)}{c^2r^2} & 0 & 0 & 0 \\ 0 & -\frac{2GM\left(-\frac{GM}{c^2r}+\frac{1}{2}\right)}{c^2r^2\left(-\frac{2GM}{c^2r}+1\right)^2} & 0 & 0 \\ 0 & 0 & -2r\left(-\frac{GM}{c^2r}+\frac{1}{2}\right) & 0 \\ 0 & 0 & 0 & -2r\left(-\frac{GM}{c^2r}+ \end{bmatrix}\right.$$

In [10]:
```
# Calculating Riemann Tensor from Christoffel Symbols
rm1 = RiemannCurvatureTensor.from_christoffels(sch_ch)
rm1.tensor()
```

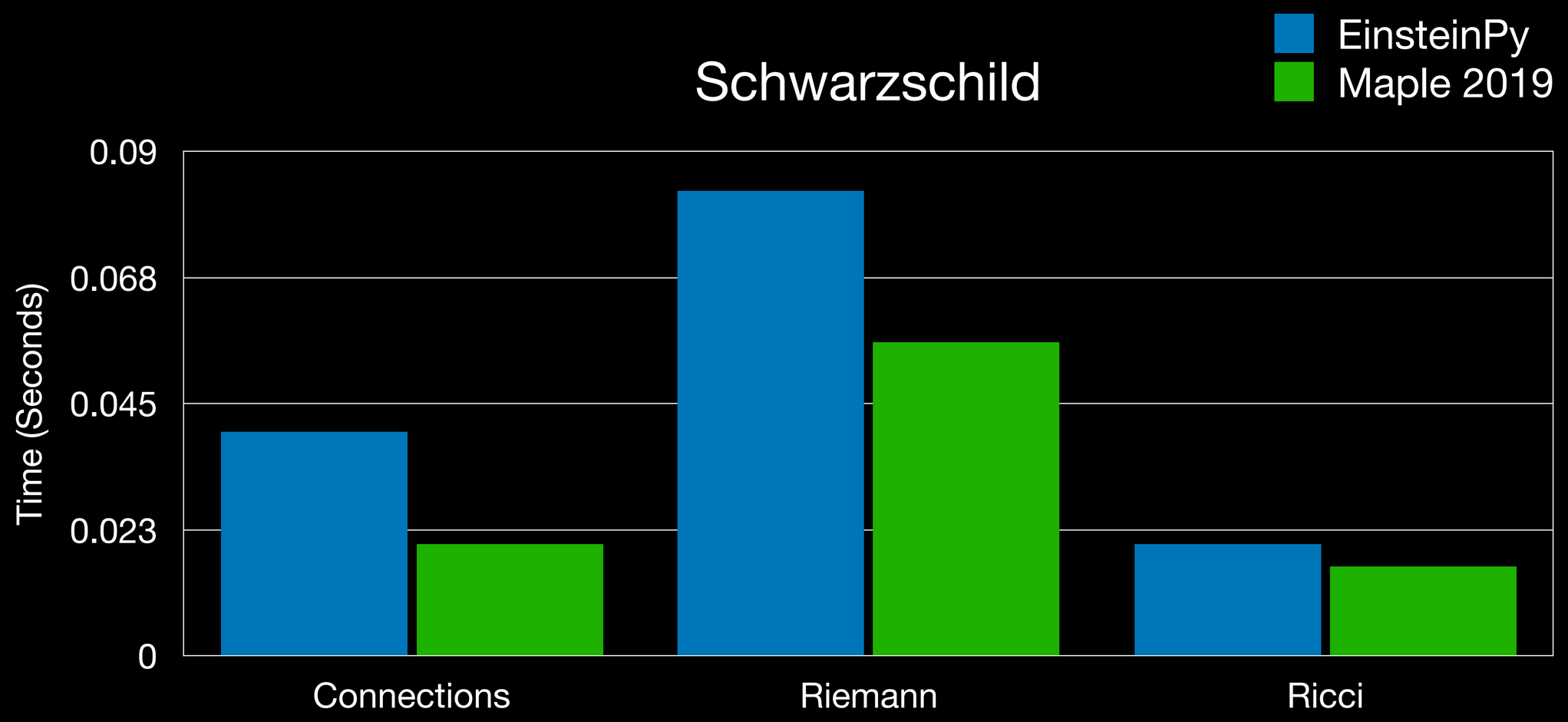
```
CPU times: user 1.04 s, sys: 8.07 ms, total: 1.04 s
Wall time: 1.05 s
```

Out[10]:

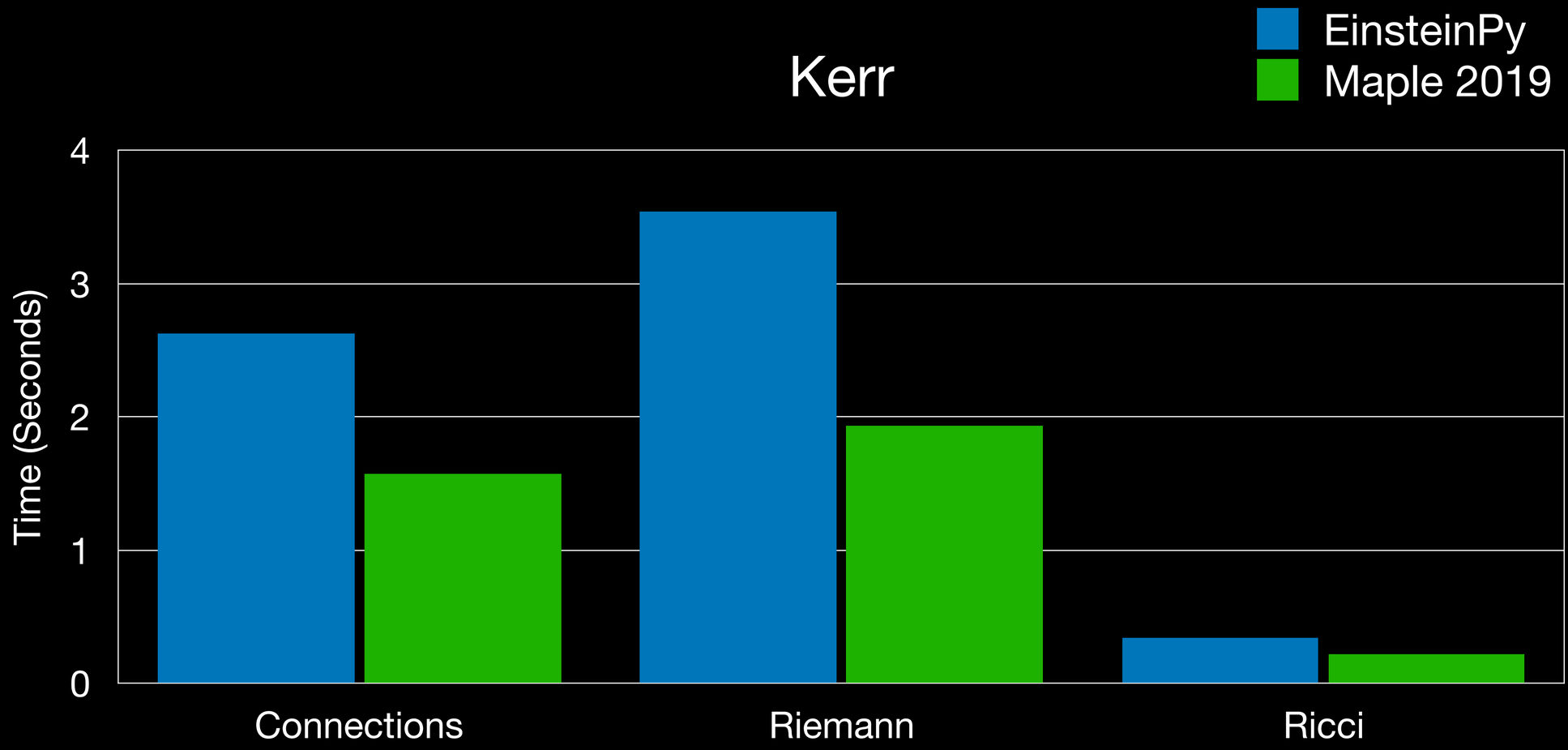
$$\begin{bmatrix}
\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{2GM}{r^2(2GM-c^2r)} & 0 & 0 \\ 0 & 0 & \frac{GM}{c^2r} & 0 \\ 0 & 0 & 0 & \frac{GM\sin^2(\theta)}{c^2r} \end{bmatrix} &
\begin{bmatrix} 0 & \frac{2GM}{r^2(-2GM+c^2r)} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} &
\begin{bmatrix} 0 & 0 & -\frac{GM}{c^2r} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
\begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{2GM(2GM-c^2r)}{c^4r^4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} &
\begin{bmatrix} \frac{2GM(-2GM+c^2r)}{c^4r^4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{GM}{c^2r} & 0 \\ 0 & 0 & 0 & \frac{GM\sin^2(\theta)}{c^2r} \end{bmatrix} &
\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{GM}{c^2r} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{GM(-2GM+c^2r)}{c^4r^4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} &
\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{GM}{r^2(2GM-c^2r)} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} &
\begin{bmatrix} \frac{GM(2GM-c^2r)}{c^4r^4} & 0 \\ 0 & \frac{GM}{r^2(-2GM+c^2r)} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}
\end{bmatrix}$$



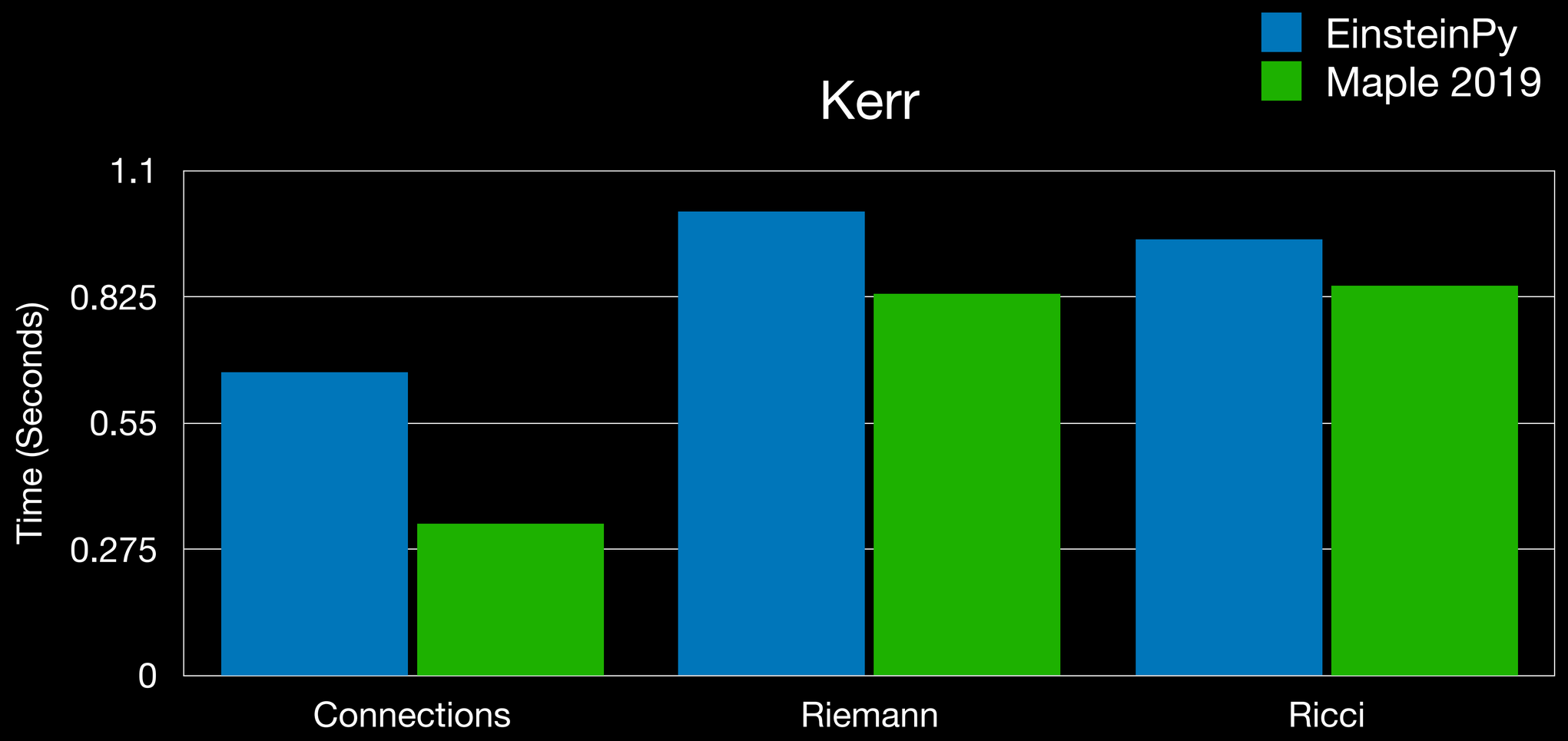
Macbook Air - Intel Core i5 @ 1.3 GHz - 4 GB RAM - Intel HD Graphics 5000 1536 MB



MacPro - Intel Xeon E5 @ 3.7 GHz - 12 GB RAM - 2 AMD FirePro D300 graphics processors 2GB each



Macbook Air - Intel Core i5 @ 1.3 GHz - 4 GB RAM - Intel HD Graphics 5000 1536 MB



MacPro - Intel Xeon E5 @ 3.7 GHz - 12 GB RAM - 2 AMD FirePro D300 graphics processors 2GB each

Particle Motion

$$\frac{d^2x^\alpha}{d\tau^2} + \Gamma^\alpha_{\mu\nu}\frac{dx^\mu}{d\tau}\frac{dx^\nu}{d\tau} = 0$$

Geodesics.
Set of 4 non-linear second-order differential equations.

Numerical solution is usually the best option!

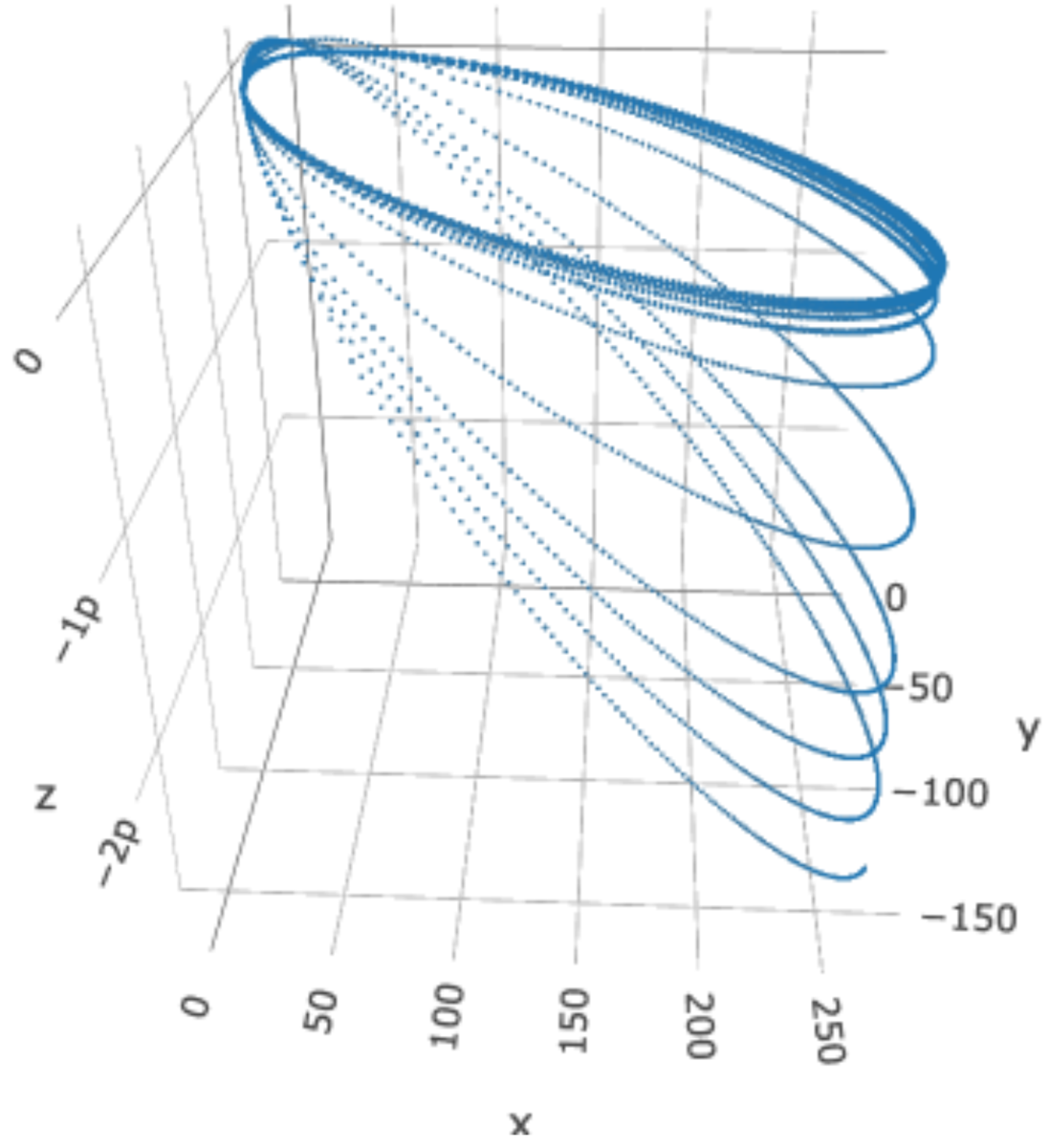
Massive Particles

In [1]:
```
import numpy as np
from astropy import units as u
from einsteinpy.coordinates import SphericalDifferential, CartesianDifferential
from einsteinpy.metric import Schwarzschild

import plotly
import plotly.graph_objs as go
```

In [2]:
```
M = 5.972e24 * u.kg
sph_coord = SphericalDifferential(306.0 * u.m, np.pi/2 * u.rad, -np.pi/6*u.rad,
                                  0*u.m/u.s, 0*u.rad/u.s, 1900*u.rad/u.s)
obj = Schwarzschild.from_coords(sph_coord, M , 0* u.s)
```

In [3]:
```
cartsn_coord = CartesianDifferential(.265003774 * u.km, -153.000000e-03 * u.km,  0 * u.km,
                     145.45557 * u.km/u.s, 251.93643748389 * u.km/u.s, 0 * u.km/u.s)
obj = Schwarzschild.from_coords(cartsn_coord, M , 0* u.s)
```

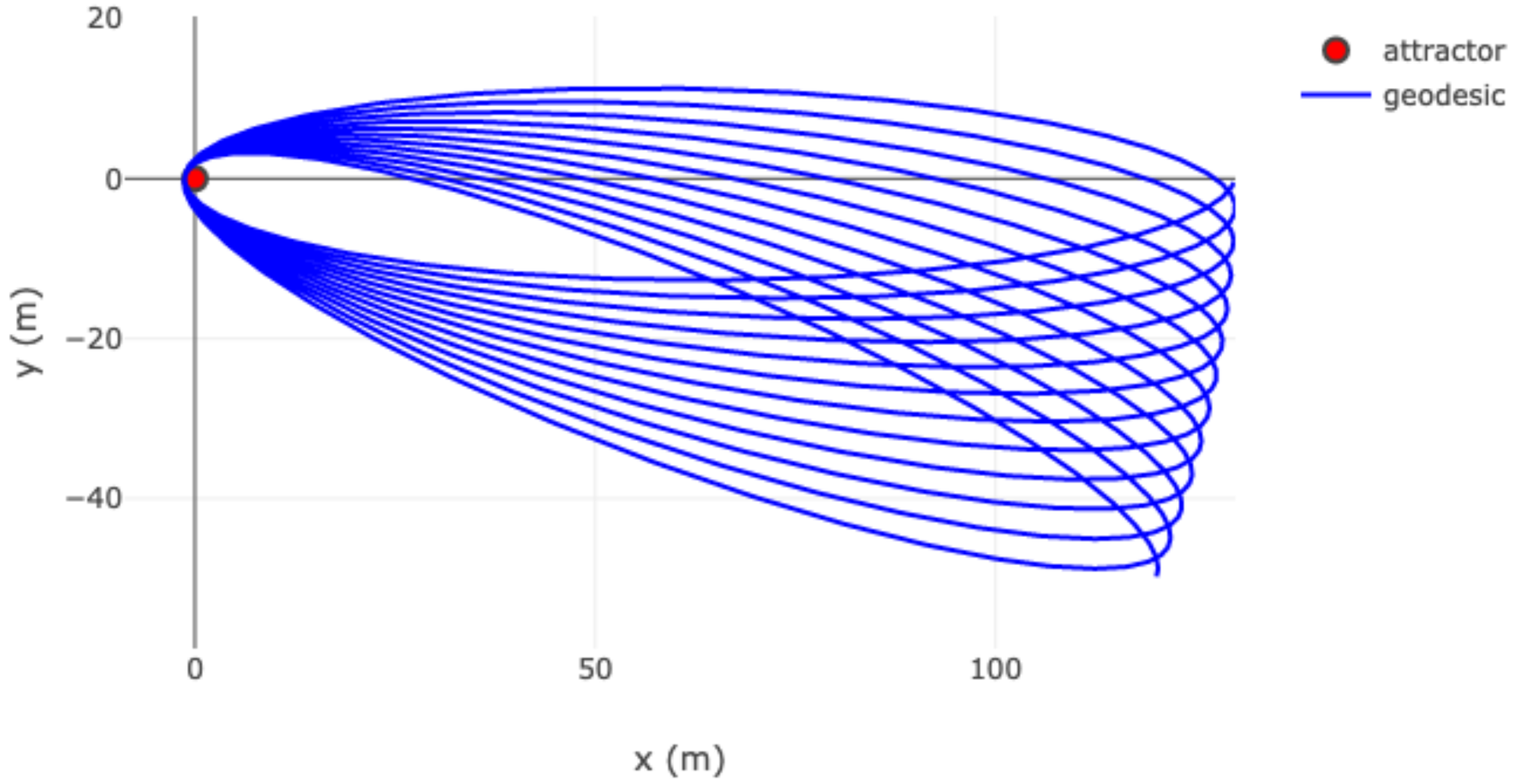
In [4]:
```
end_tau = 0.01 # approximately equal to coordinate time
stepsize = 0.3e-6
ans = obj.calculate_trajectory(end_lambda=end_tau, OdeMethodKwargs={"stepsize":stepsize})
print(ans)
```


0
−1p
−2p
z
0
50
100
150
200
250
x
0
−50
−100
−150
y

In [1]:
```
import numpy as np
import astropy.units as u

from plotly.offline import init_notebook_mode

from einsteinpy.plotting import GeodesicPlotter
from einsteinpy.plotting import StaticGeodesicPlotter

from einsteinpy.coordinates import SphericalDifferential
from einsteinpy.bodies import Body
from einsteinpy.geodesic import Geodesic
```

In [2]:
```
init_notebook_mode(connected=True)
# Essential when using Jupyter Notebook (May skip in Jupyter Lab)
```

In [3]:
```
Attractor = Body(name="BH", mass=6e24 * u.kg, parent=None)
sph_obj = SphericalDifferential(130*u.m, np.pi/2*u.rad, -np.pi/8*u.rad,
                                0*u.m/u.s, 0*u.rad/u.s, 1900*u.rad/u.s)
Object = Body(differential=sph_obj, parent=Attractor)
geodesic = Geodesic(body=Object, time=0 * u.s, end_lambda=0.002, step_size=5e-8)
```

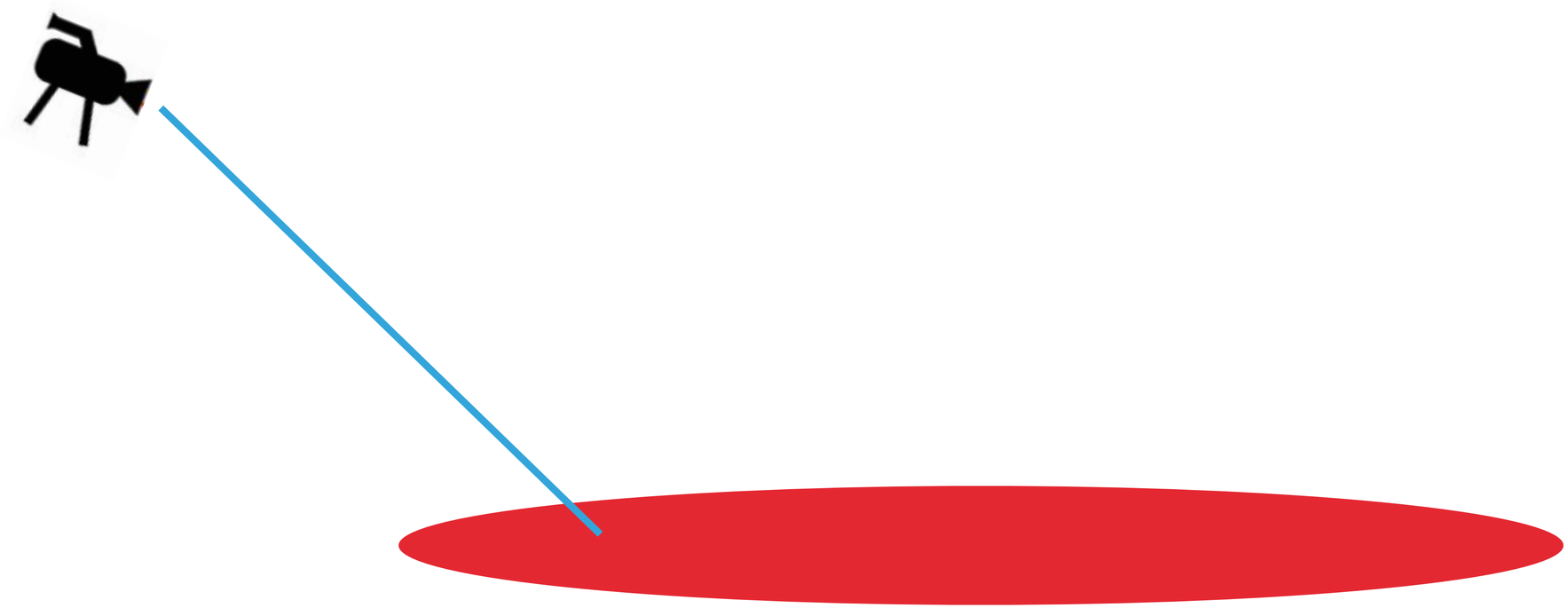

20
0
−20
−40
y (m)
0
50
100
x (m)
attractor
geodesic

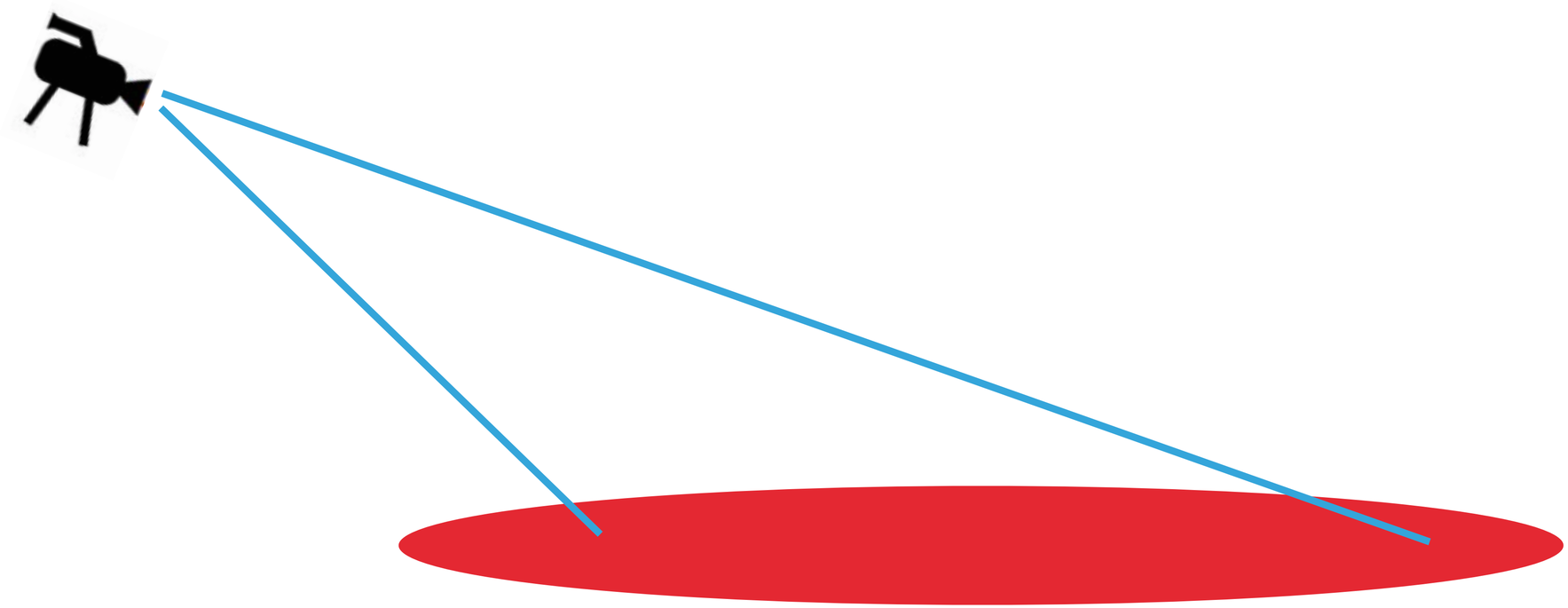
$$\frac{d^2x^\alpha}{d\tau^2} + \Gamma^\alpha_{\mu\nu}\frac{dx^\mu}{d\tau}\frac{dx^\nu}{d\tau} = 0$$

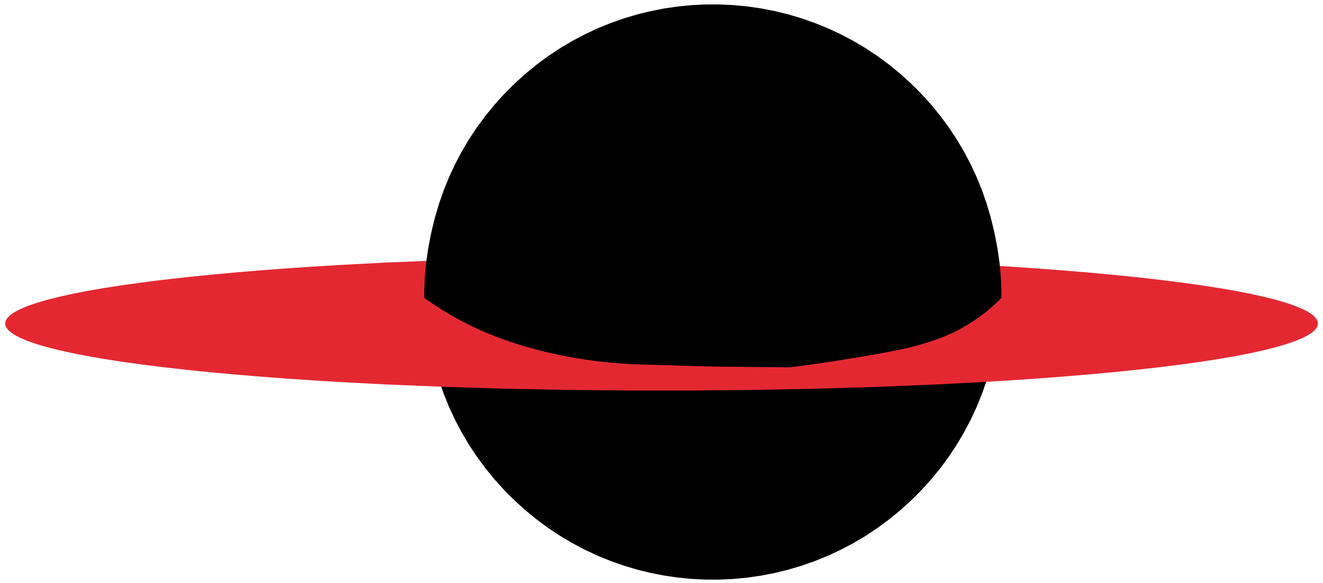
Geodesics.
Set of 4 non-linear second-order differential equations.

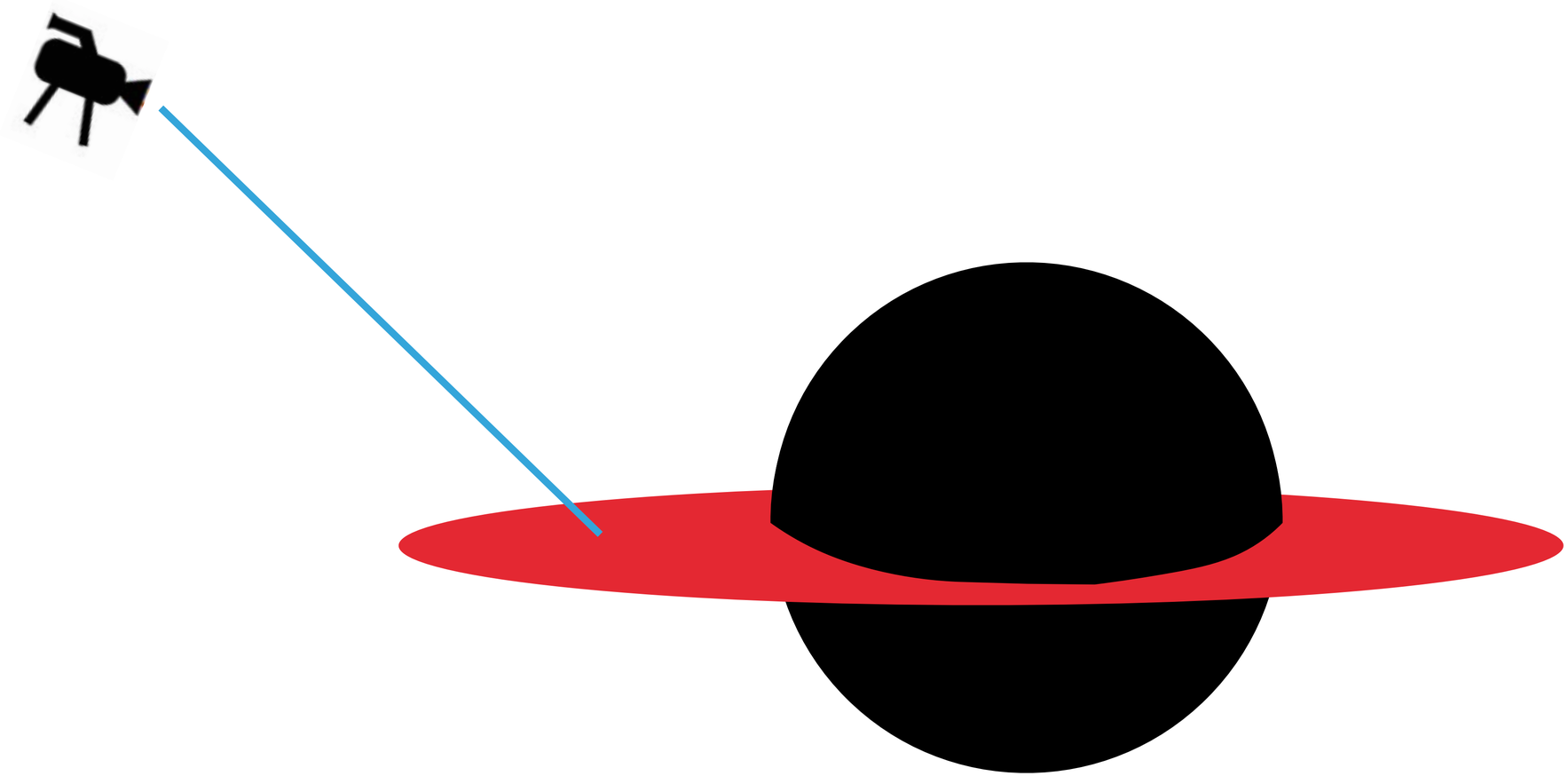
Numerical solution is usually the best option!

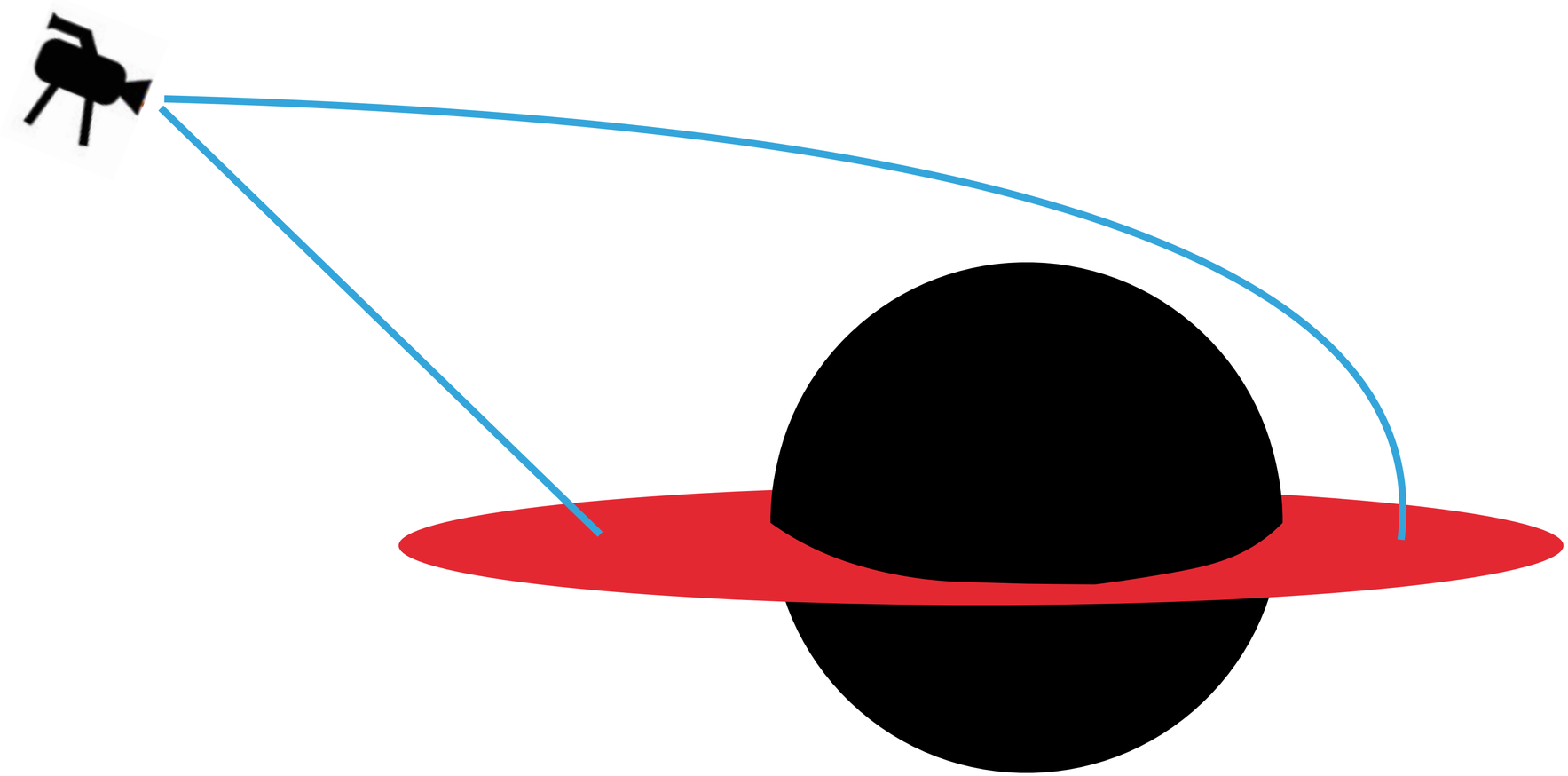
Light Rays

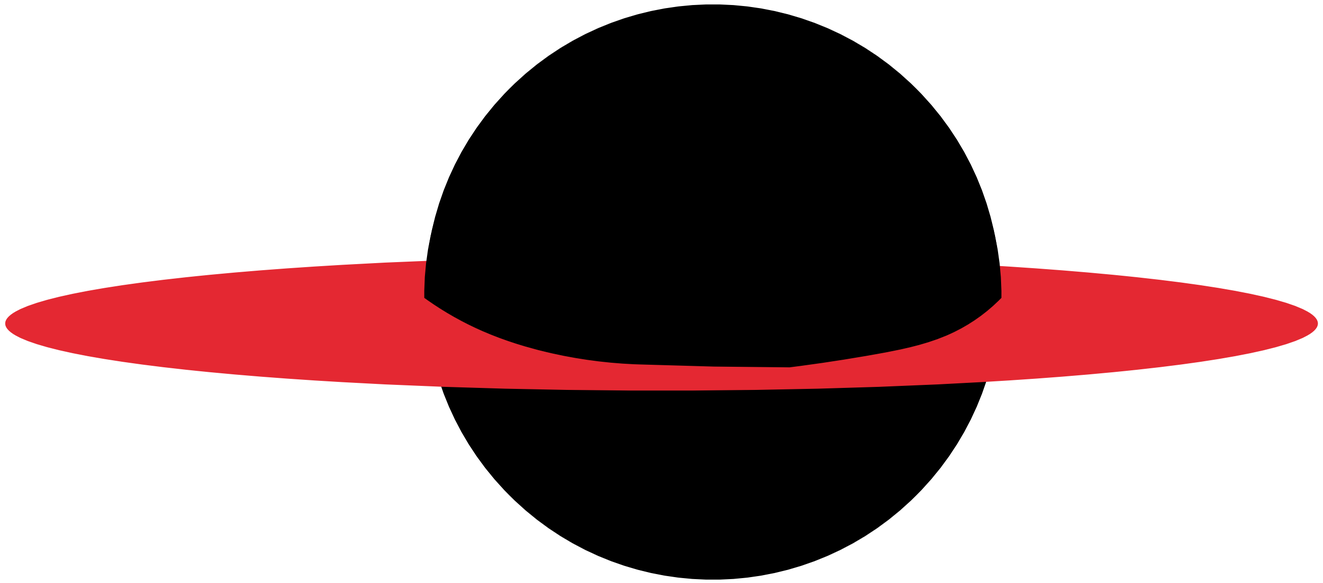


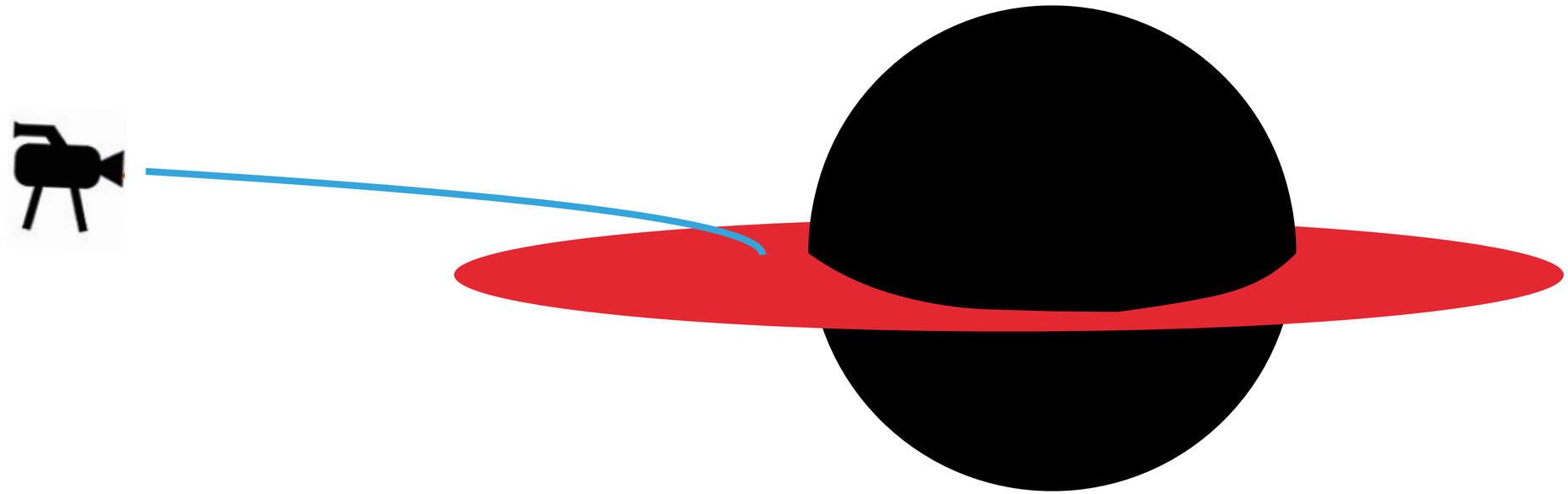


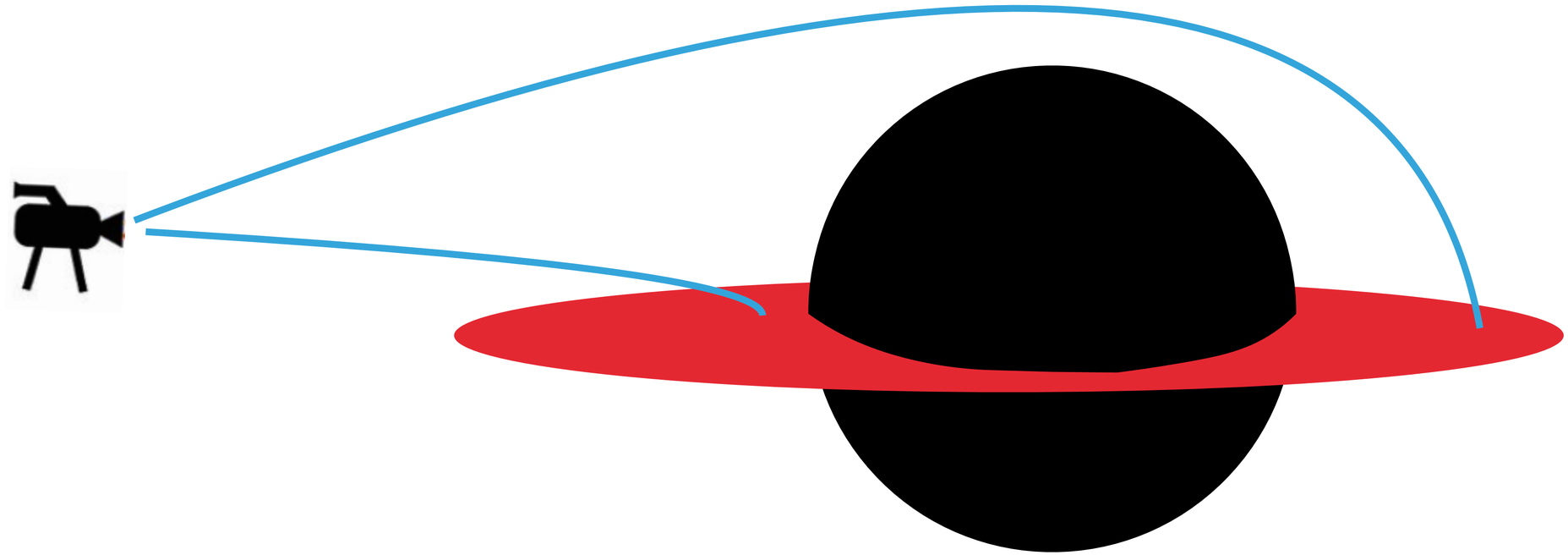


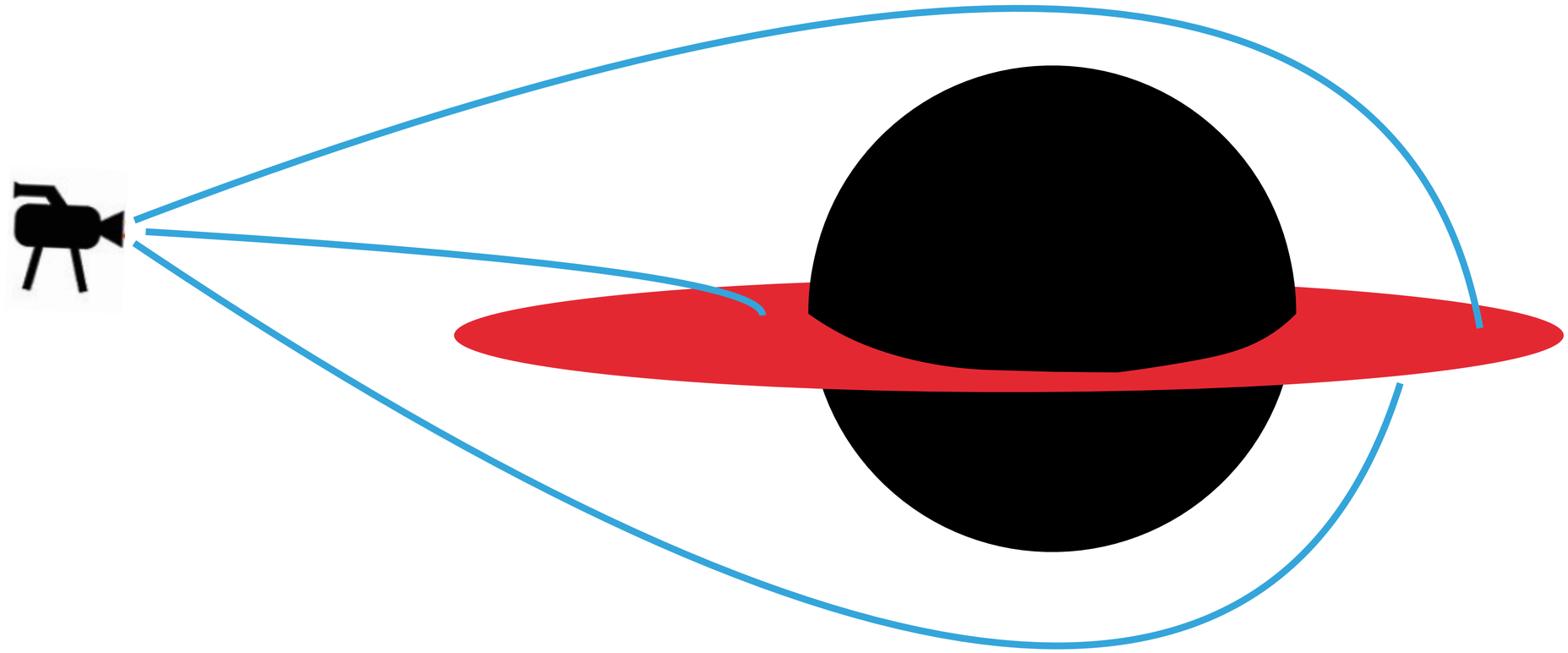


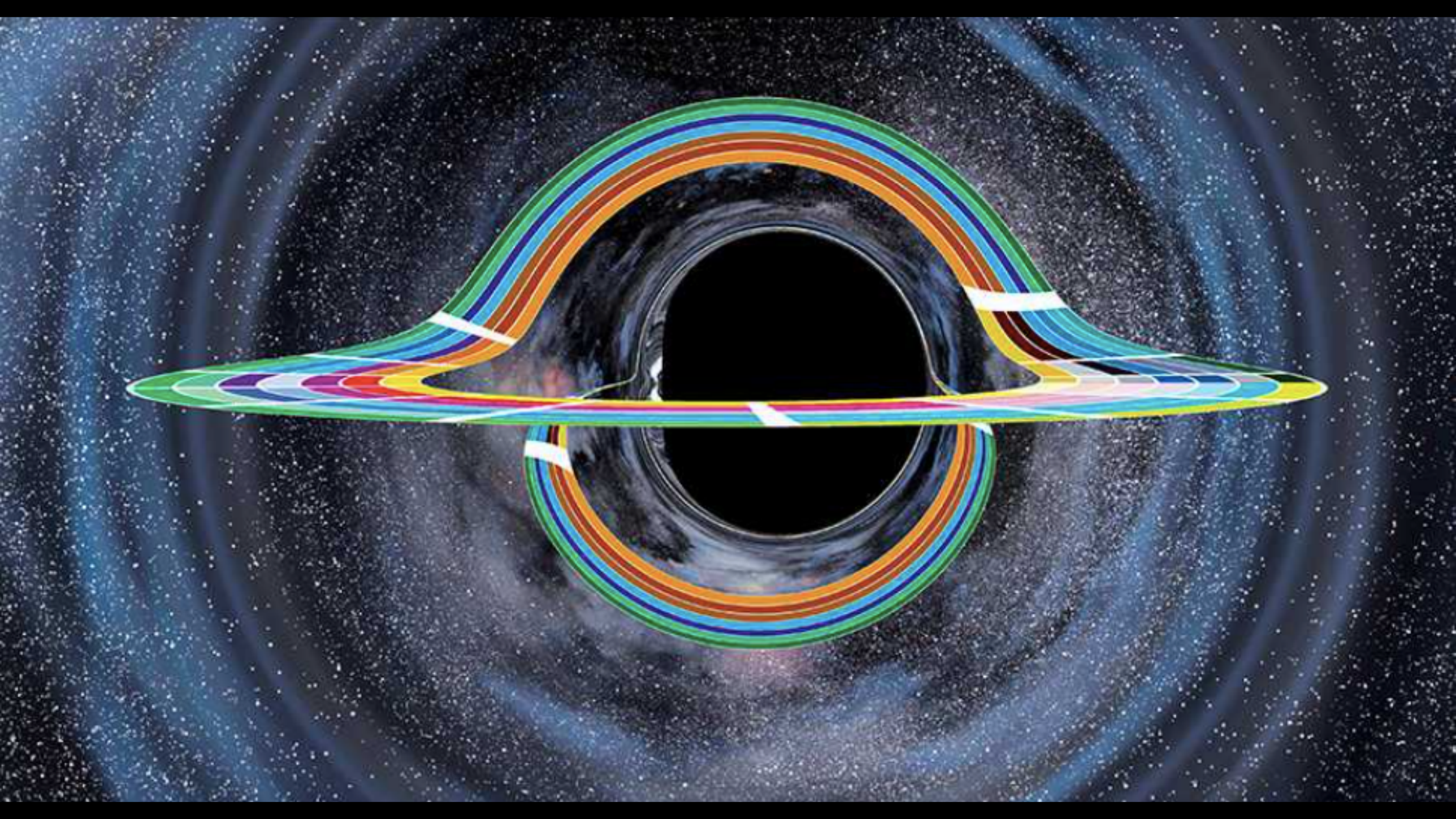


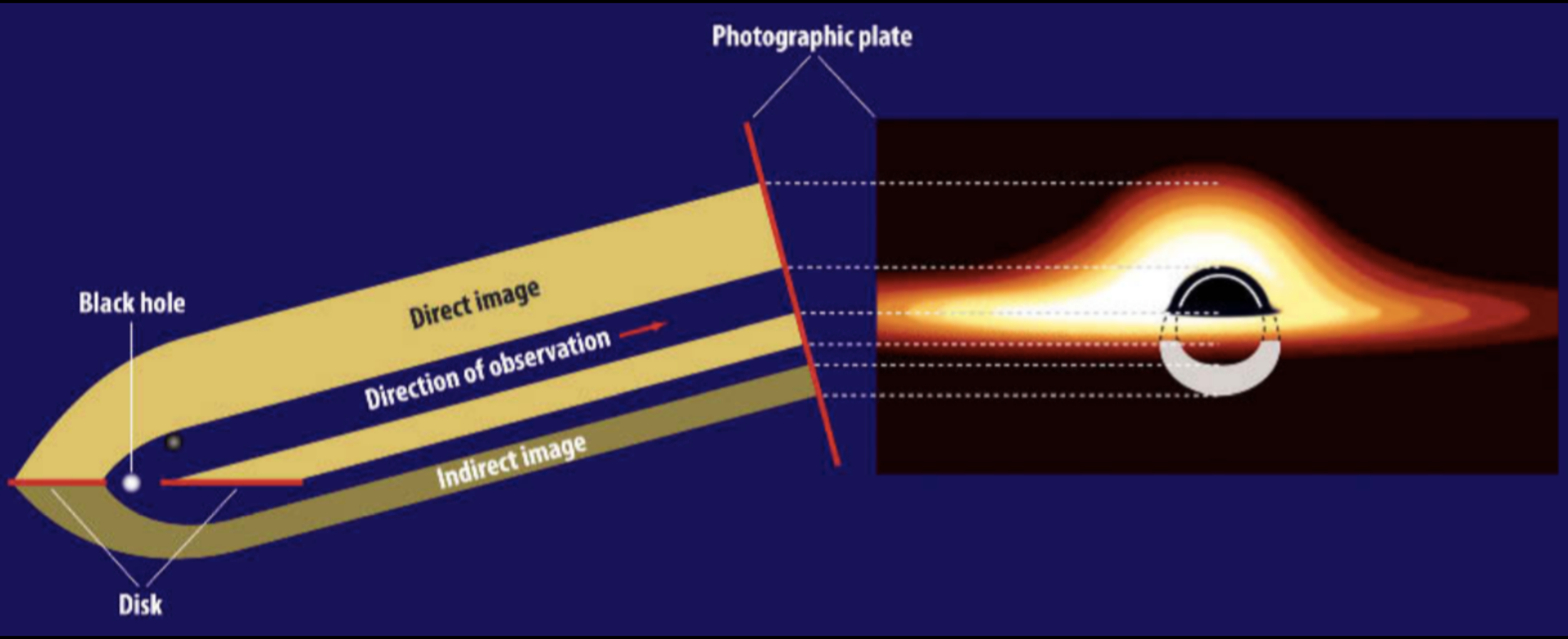




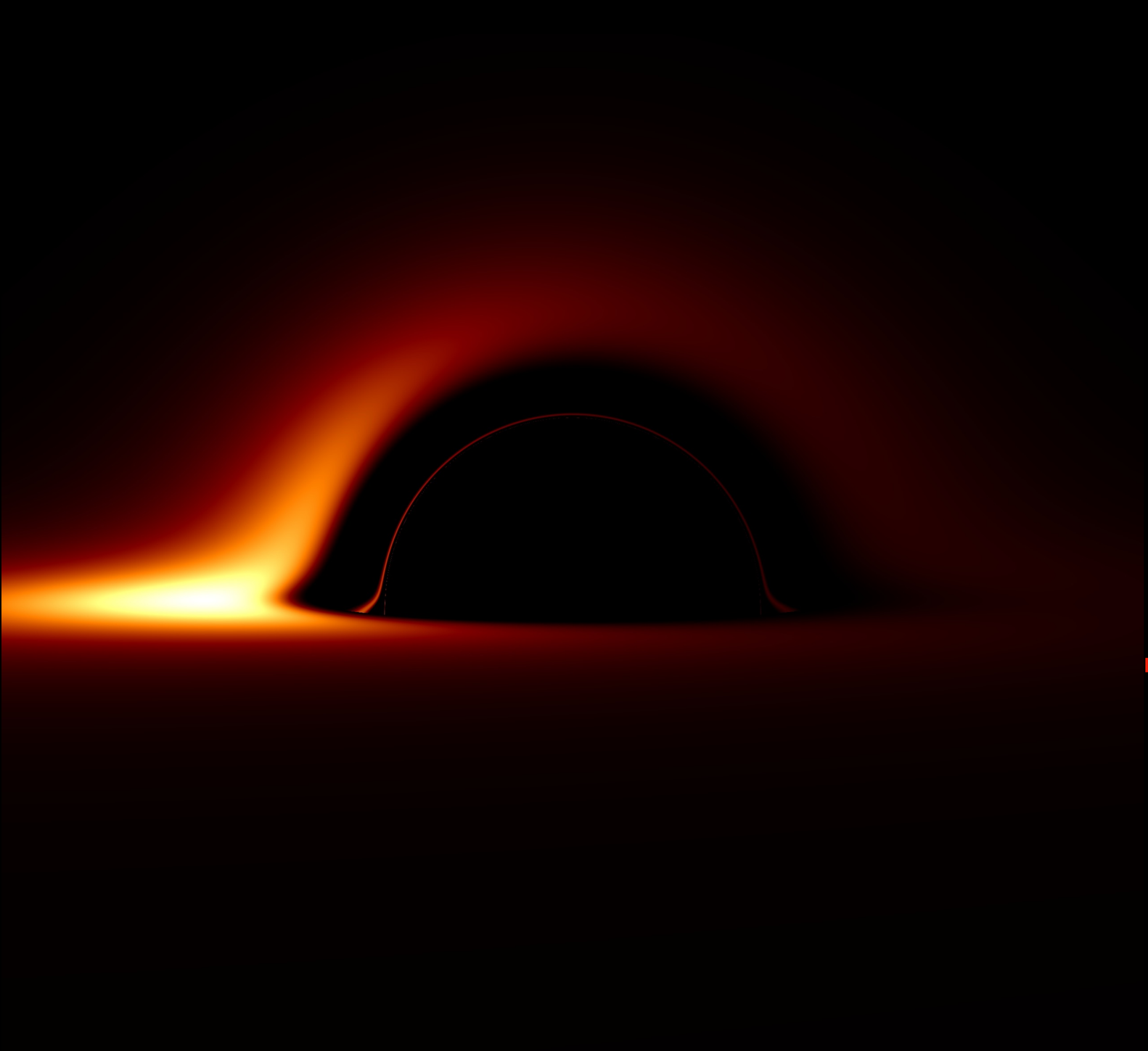






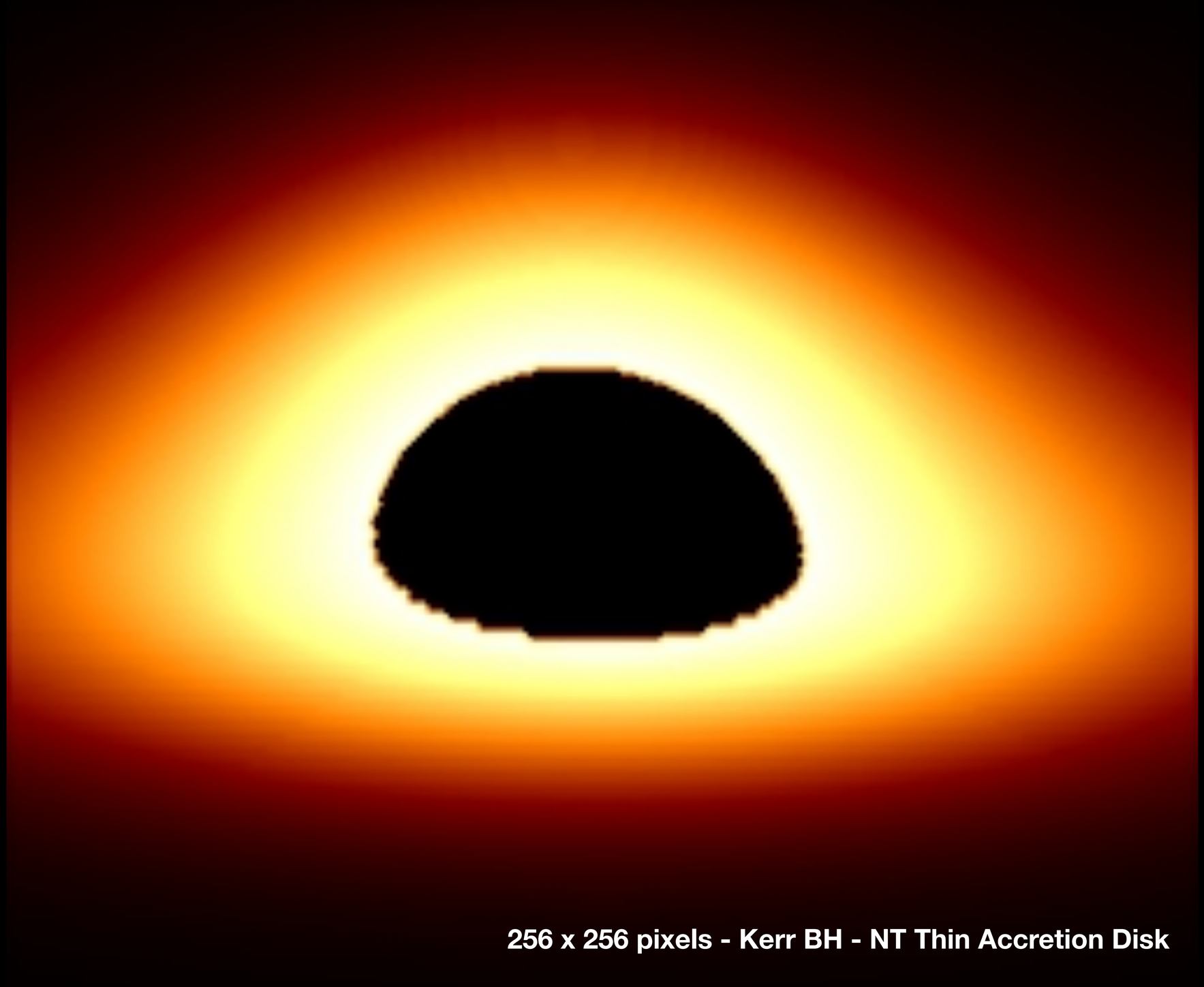

Photographic plate
Black hole
Direct image
Direction of observation
Indirect image
Disk

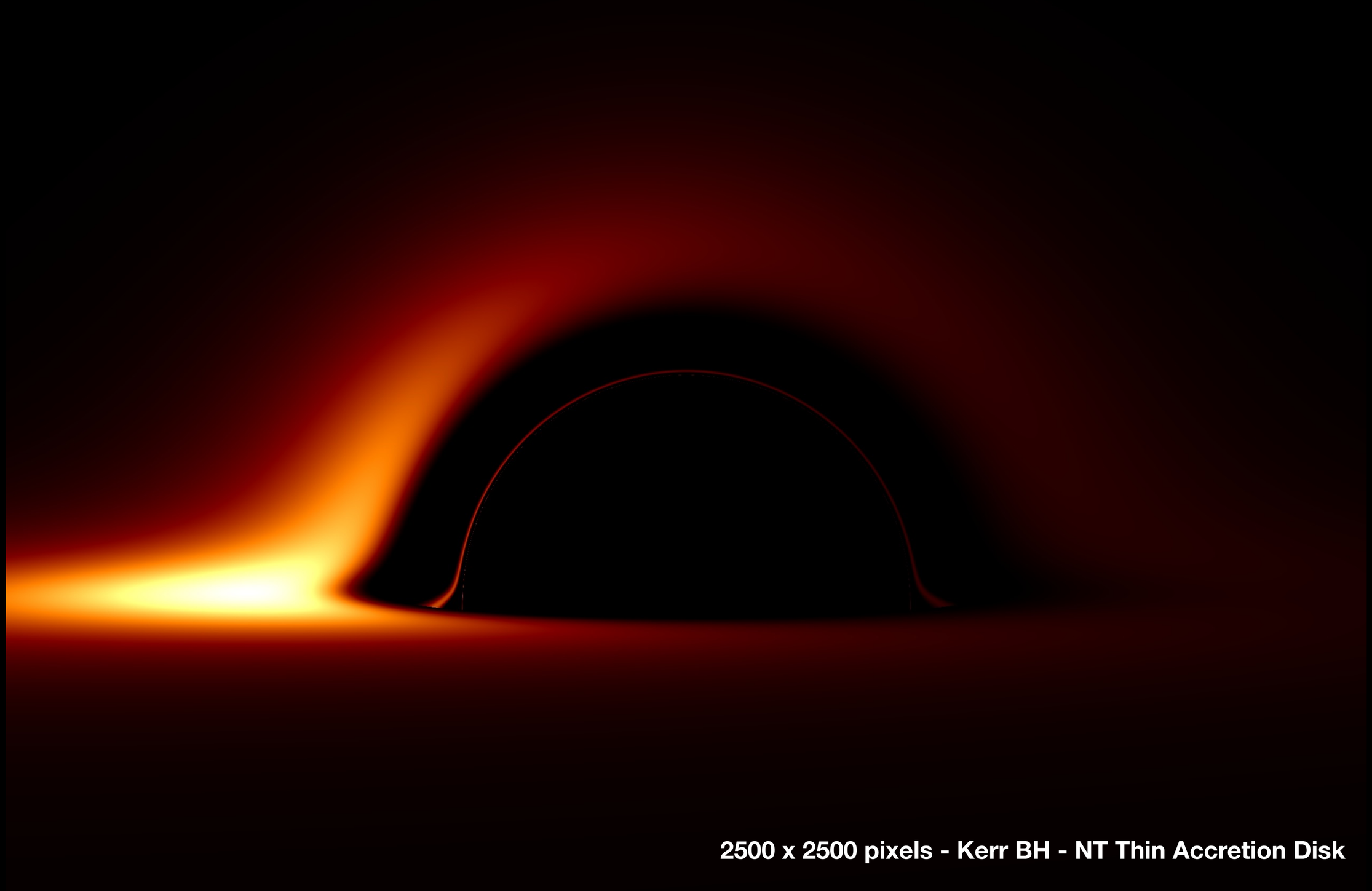

python
TM



Black Holes Imaging Tool

www.github.com/ashcat2005/BHIT


256 x 256 pixels - Kerr BH - NT Thin Accretion Disk


2500 x 2500 pixels - Kerr BH - NT Thin Accretion Disk

RAY TRACING

- Solve the geodesic equations in a specified curved spacetime beginning at the observer and ending at the accretion structure.
- The metric is defined in a specific module.
  :: Schwarzschild - Kerr - etc.
- The accretion structure is implemented in another module
  :: Novikov-Thorne thin disks - Polish donut - etc.
- On average, our code integrates the path of 3 photons per second.

FUTURE DEVELOPMENT

- Implement more metrics.
  :: Kiselev - Kerr-Sen - ...
- Implement more accretion structures
  :: Accretion torus - ADAF
- Optimize the integration of the geodesics.

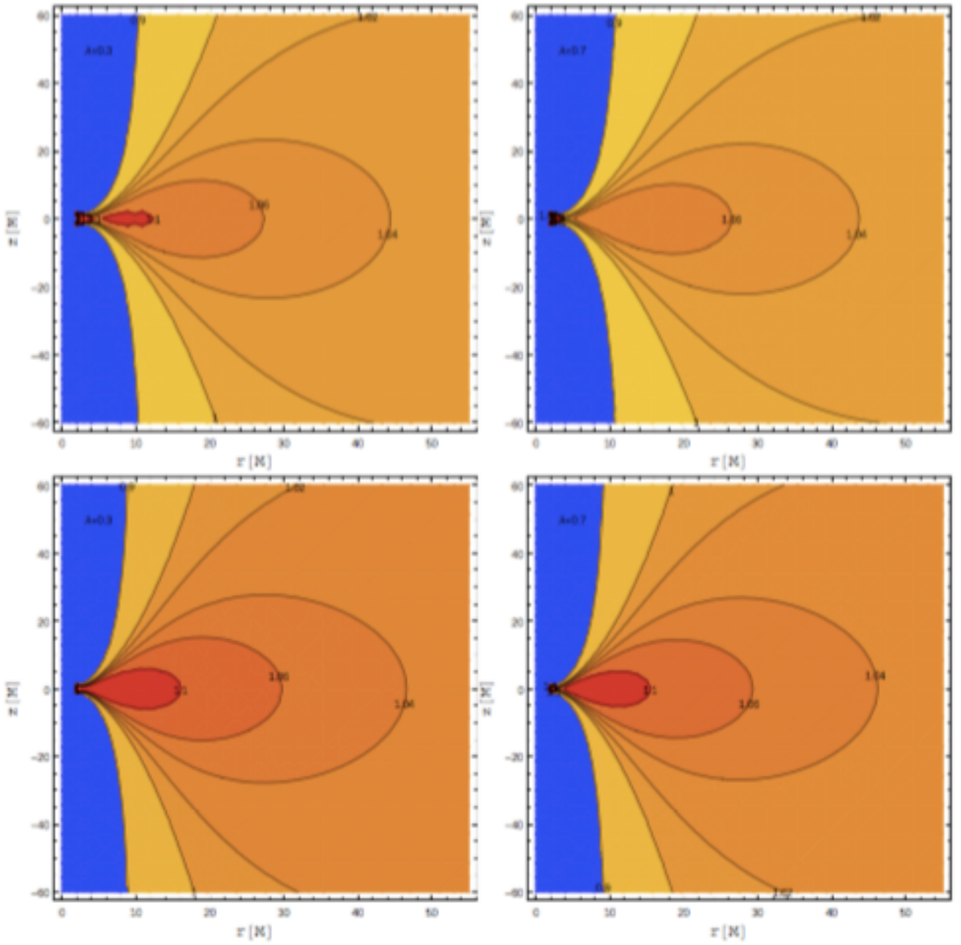


Figure 5.1: Meridional cuts through a $\lambda = 0.3$ and $\lambda = 0.3$ around a black hole. (*Top panels*) Schwarzschild case. (*Bottom panels*) Kerr case with a $a = 0.5\,M$. Note that the surfaces of constant pressure, which represent the possible boundaries of the fluid configuration. The numbers on the curves refer to different values of W. This figures can be compared with Fig. 1 of Refs. [1, 85].

A. Cárdenas. *Probing the Regular Naturee of the Spacetime by Direct Measurement of Black Hole Properties.* MSc Thesis. Observatorio Astronómico Nacional (2015)

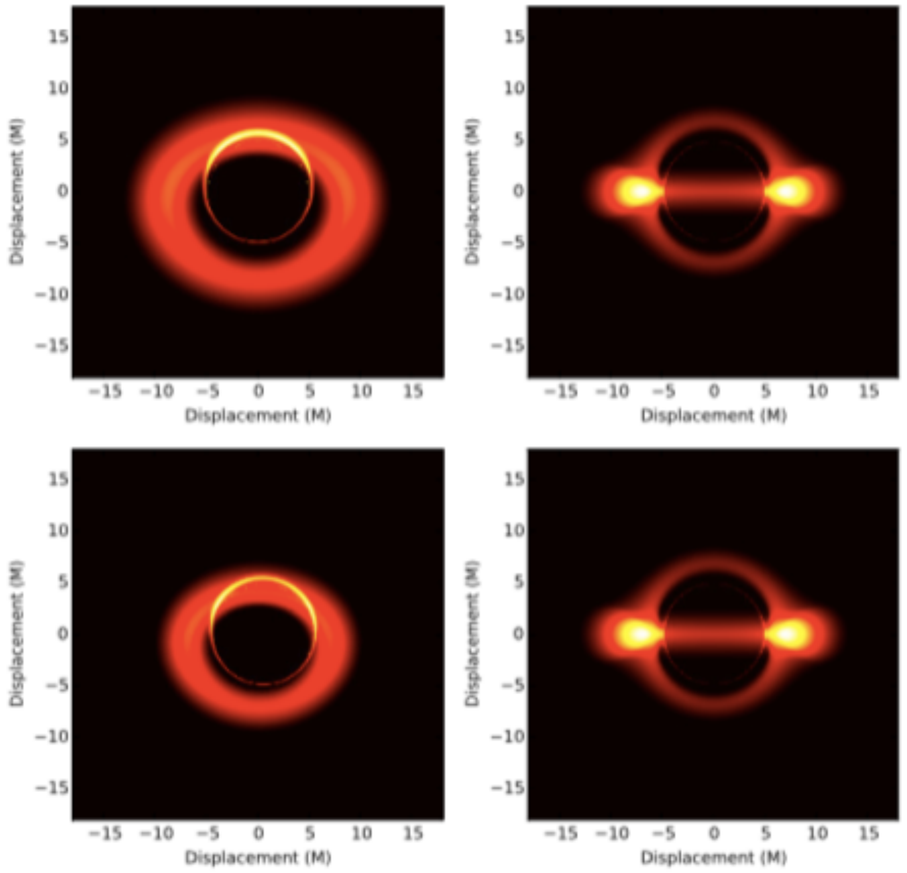


Figure 5.4: Images of the reference ion torus corresponding to the parameters listed on Table 5.1, as observed by an observer on Earth. *Top panels:* Schwarzschild case. (*Upper left*) Inclination angle $i = \pi/4$. (*Upper right*) Inclination angle $i = \pi/2$. *Bottom panels:* Kerr case with $a = 0.4\,M$. (*Lower left*) Inclination angle $i = \pi/4$. (*Lower right*) Inclination angle $i = \pi/2$.

A. Cárdenas. *Probing the Regular Naturee of the Spacetime by Direct Measurement of Black Hole Properties.* MSc Thesis. Observatorio Astronómico Nacional (2015)

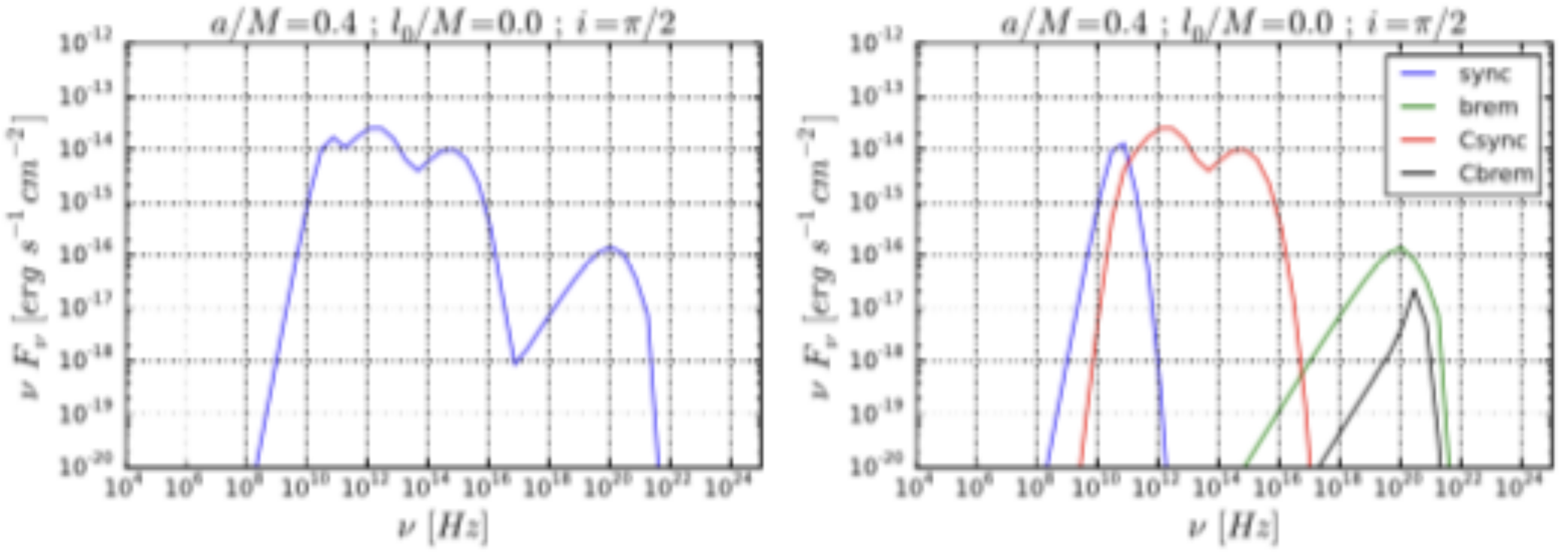
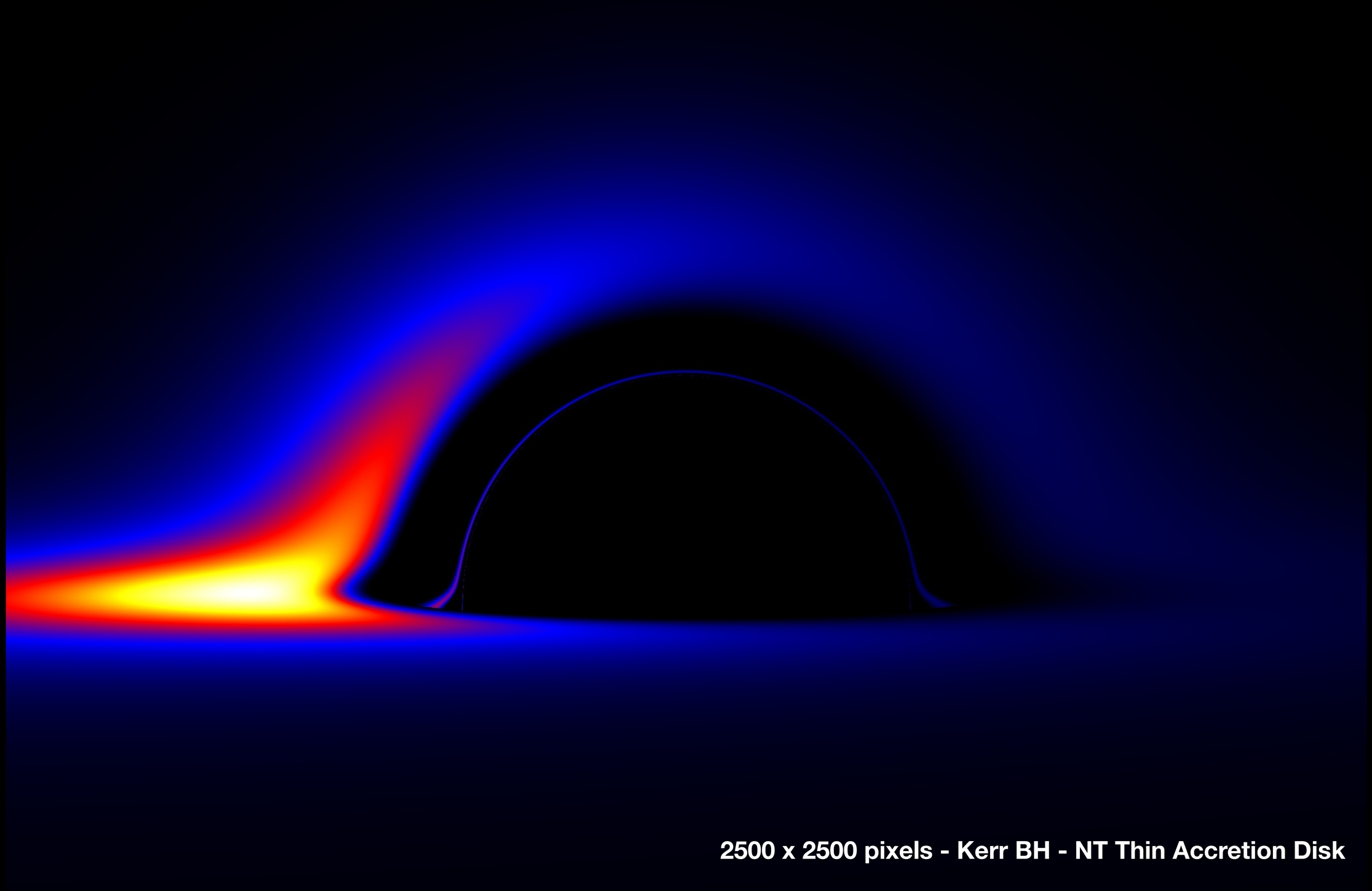


Figure 5.6: (*Left*) Spectrum of the ion torus, showing the quantity νF_ν for different values of frequencies. All the parameters not shown in the figures are set to their reference values listed on Table 5.1. (*Right*) Contribution from every radiative mechanism, namely bremsstrahlung emission, Comptonisation of bremsstrahlung emission, synchrotron radiation, and Comptonisation of synchrotron radiation.

A. Cárdenas. *Probing the Regular Naturee of the Spacetime by Direct Measurement of Black Hole Properties.* MSc Thesis. Observatorio Astronómico Nacional (2015)


2500 x 2500 pixels - Kerr BH - NT Thin Accretion Disk

@astronomiaOAN

AstronomiaOAN

@astronomiaOAN