Copyright Taylor & Francis 2018.

A

General Relativity

A.1 Chapter Overview

These general relativity (GR) problems have been added after requests from readers, and extend the problems in special relativity in Chapter 5. These problems deal with the visualization of wormholes, the deflection of starlight by the sun, the gravitational lensing by highly massive stars, and the motion of a particle in a Newtonian potential with a GR correction, and the computation of some GR quantities.

A.2 Visualizing Wormholes

During Christopher Nolan's direction of the science fiction movie *Interstellar*, Kip Thorne (now a Noble laureate) helped develop the visualizations of rocket flight based on solutions of the equations of Einstein's theory of general relativity. The key element of the movie was that interstellar travel was possible in a single human lifetime if a spaceship passed through a wormhole (an Einstein–Rosen bridge), a tunnel-like structure in spacetime that connects one location in spacetime to another, or possibly to another universe [James et al. (15)]. Fig.(A.1) is the visualization of such a wormhole.

Although wormholes have never been observed, they may occur as quantum fluctuations on the Planck scale, $\sqrt{G\hbar/c^3} \sim 10^{-35}$ m. Furthermore, it just might be possible to have some type of exotic matter with negative energy density at the throat of the wormhole that would enlarge the wormhole to a macroscopic size that might permit a rocket ship pass through it. However, if our 4-D universe resided in a higher-dimensional space (bulk), such as the 5-D one imagined in *Interstellar*, then there might not be the need for exotic matter to hold the wormhole open. In either case, interstellar travel can be imagined to be possible. While all of this is unlikely (it is called science fiction after all), it is not strictly forbidden.

Morris and Thorn [Morris, M. S., K. S. Thorne, *Wormholes in spacetime and their use for interstellar travel: A tool for teaching General Relativity*, Am. J. Phys., **56**, 395-412, (1988).] discuss the fundamentals of space travel using wormholes as an exercise in general relativity. Your **problem** is to reproduce some stills of the wormhole visualizations that were created

Copyright Taylor & Francis 2018.

Figure A.1. The Ellis wormhole connecting an upper and lower (flatter) spaces. Note that this visualization has the wormhole's 4-D bulk embedded within a 3-D space. The throat diameter is 2ρ and the proper distance traveled in a radial direction is ℓ.

for the movie. As an alternative, you can reproduce some of the (different) visualizations found in [T. A. Roman, *The inflated wormhole: A MATHEMATICA animation*, Comp. in Phys., 480-487, (1994).]. You will not be asked to actually solve Einstein's equations, although we would encourage you to do so. Another extension would be the creation of videos that visualize what travel through a wormhole would look like if recorded by a camera on the space ship, or from outside the wormhole. Some such visualizations from the *Interstellar* movie can be found on *Youtube* [https://www.youtube.com/watch?v=f3ptQ0CPMmU.].

The equations that Thorne used to create the visualizations were expressed in geometrized units in which $G = 1$, $c = 1$, time is measured in length $1s = c * 1 = 2.998 \times 10^8$ m, mass is also measured in length units, 1 kg$= G/c^2 \times$ 1 kg, so that 1 kg $= 0.742 \times 10^{-27}$ m, in which case the Sun's mass = 1.476 km. The wormhole consists of a 4-D cylinder with length $2a$ whose cross sections are spheres of radius ρ. In order to visualize the 4-D wormhole, it is embedded in a 3-D space so that the cross section are circles of radius ρ. The ends of the cylinder connect to flat 3-D spaces.

Thorne uses the Ellis extension of a spherical polar coordinates metric:

$$ds^2 = -dt^2 + d\ell^2 + r^2(d\theta^2 + \sin^2\theta d\phi^2). \tag{A.1}$$

Here the radius coordinate r is a function of ℓ, the physical distance (proper distance)

Copyright Taylor & Francis 2018.

traveled in a radial direction:

$$r(\ell) = \sqrt{\rho^2 + \ell^2}, \tag{A.2}$$

where ρ is the radius of the throat in a cylindrical-shaped wormhole. Note that the time coordinate t enters the metric (A.1) with a negative sign. This means that for fixed ℓ, θ, and ϕ, t increases in the timelike direction. Accordingly, t is the proper time as measured by a person at rest in the spatial (ℓ, θ, ϕ) coordinate system.

Because $r^2(d\theta^2 + \sin^2\theta d\phi^2)$ is the familiar metric describing the surface of a sphere of radius r, the wormhole is spherically symmetric. This means that when $l \to +\infty$, as well as when $l \to -\infty$, the radius of the sphere within the wormhole approaches proper distance ℓ. This also means that as $l \to \pm\infty$ we would have two separate flat spaces connected by the wormhole. The transition between the two flat spaces via the wormhole's throat is made to resemble the transition to an external space in which a nonspinning black hole resides. This is described by the Schwarzschild or hole metric [James et al. (15)]:

$$ds^2 = -(1 - 2\mathcal{M}/r)dr^2 + \frac{dr^2}{1 - 2\mathcal{M}/r} + r^2\,(d\theta^2 + \sin^2\theta\, d\phi^2), \tag{A.3}$$

where $\mathcal{M}$ is the black hole's mass. With this metric, the radius r becomes the outward coordinate rather than the proper distance ℓ. The visualizations in the movie required a solution for $r(\ell)$, that is, a solution or an expression for the outward coordinate as a function of proper distance. To reduce the effort involved, the visualizations used an analytic expression for $r(\ell)$ outside the wormhole's cylindrical interior that is similar to the Schwarzschild $r(\ell)$:

$$r(|\ell| > a) = \rho + \frac{2}{\pi}\int_0^{|\ell|-a} \arctan\Big(\frac{2\xi}{\pi\mathcal{M}}\Big)\, d\xi \tag{A.4}$$

$$= \rho + \mathcal{M}\left[x \arctan x - \frac{1}{2}\ln(1 + x^2)\right], \qquad x = \frac{2\,|\ell| - a}{\pi\mathcal{M}}. \tag{A.5}$$

For cylindrical coordinates, the z coordinate is the embedding space height above the wormhole's midplane, and so the embedding space metric becomes

$$ds^2 = dz^2 + dr^2 + r^2\, d\phi^2. \tag{A.6}$$

In this case, the spatial metric of the wormhole's two-dimensional equatorial surface becomes:

$$ds^2 = d\ell^2 + r^2(\ell)\, d\phi^2. \tag{A.7}$$

Combining these equations lets us solve for $z(\ell)$:

$$dz^2 + dr^2 = d\ell^2, \tag{A.8}$$

$$z(\ell) = \int_0^{\ell} \sqrt{1 - (dr/dL)^2}\, dL. \tag{A.9}$$

Copyright Taylor & Francis 2018.

You obtain the equations needed to program up the visualization of a wormhole by substituting (A.4) and (A.5) into (A.9).

1. In order to apply (A.9) we need to evaluate the derivative $dr/d\ell$. Use Python's symbolic algebra package `sympy` to show that

$$\frac{dr}{d\ell} = \frac{2}{\pi}\tan^{-1}\left(\frac{2\ell - a}{\pi\mathcal{M}}\right). \tag{A.10}$$

   Our program `WormHole.py` in Listing A.1 evaluates this derivative.

2. Insert this $dr/d\ell$ into (A.9) and evaluate the $z(l)$ integral numerically for

$$\rho = 1, \quad a = 1, \quad \mathcal{M} = 0.5. \tag{A.11}$$

3. The contour lines or rings shown in Fig. A.1 correspond to different values of ℓ. They were obtained by using Vpython in a Jupyter notebook with the program `VisualWorm.ipynb` given in Listing A.2[1].

4. Make your own plot of the wormhole for $\ell = 1, \cdots, 11$.

5. Create a cylindrical wormhole of length $2L$ with a spherical cross sections of radius ρ. Visualize the wormhole with a 3-D embedding diagram in which the missing dimension results in the cross sections appearing as circles rather than spheres. Follow the same steps as used for the Ellis wormhole, (A.1), but now with

$$r(\ell) = \begin{cases} \rho, & |\ell| \le L \quad \text{(Wormhole interior)}, \\ |\ell| - L + \rho, & |\ell| \ge L \quad \text{(Wormhole exterior)} \end{cases} \tag{A.12}$$

A.3 Gravitational Deflection of Light

A *geodesic* is the shortest path between two points on a curved surface. General relativity assumes that light travels on geodesics, which are curved paths in a 4-D spacetime. To determine a geodesic, one starts with the infinitesimal 4-D path length (interval)

$$ds^2 = c^2dt^2 - dx^2 - dy^2 - dz^2. \tag{A.13}$$

Since light travel a distance ct in time t, the interval vanishes for light, and its path is therefore called a *null* geodesic. Since material particles move slower than light, their interval is positive (time-like). The path that light takes in spacetime is the solution of the geodesic equation

$$\frac{d^2x^\beta}{d\lambda^2} + \Gamma^\beta_{\mu\nu}\frac{dx^\mu}{d\lambda}\frac{dx^\nu}{d\lambda} = 0. \tag{A.14}$$

[1] While the previous version of Vpython explicitly called the Visual package and was run via an editor, we have been able to run the latest Vpython only within a Jupyter notebook.

Copyright Taylor & Francis 2018.

Here $\Gamma^{\beta}_{\mu\nu}$ is the Christoffel symbol and would be obtained by solving the curvature equation $R_{\mu\nu} = 0$ for a given metric tensor $g_{\mu\nu}$. The curvature equation turns out to be a rather formidable set of ten nonlinear PDE's, which we are happy to leave for another time.

Figure A.2. A light ray being bent by an angle ϕ due to the gravitational effect of the sun.

One of the early tests of general relativity was its prediction of the angle of deflection ϕ for light starting at an impact parameter $b = R$ and just grazing the sun (Figure A.2). At first Newtonian mechanics solved this problem by calculating the orbit of a massive particle around the sun, and then taking the $m \to 0$ limit for the particle to obtain

$$\phi = 2\frac{GM}{Rc^2}. \tag{A.15}$$

Here G is the gravitational constant, M is the mass of the sun, and R is the radius of the sun. Later, Einsteinian mechanics was used to solve (A.14) approximately and obtained twice as large a value,

$$\phi = 4\frac{GM}{Rc^2}, \tag{A.16}$$

which agreed with the measurements.

Now let's try to calculate some numerical values for the deflection. In 1916 Schwarzschild found an exact solution of the Einsteinian equations using Schwarzschild metric [Moore(13)],

$$ds^2 = \left(1 - \frac{2GM}{c^2 r}\right) c^2 dt^2 - \left(1 - \frac{2GM}{c^2 r}\right)^{-1} dr^2 - r^2(d\theta^2 + \sin^2\theta d\phi^2). \tag{A.17}$$

For this metric and for light just grazing the sun ($b = R$), the orbit equation takes the simple form

$$\left(\frac{1}{r}\frac{dr}{d\phi}\right)^2 = \left(1 - \frac{2M}{R}\right)\frac{1}{R^2} - \left(1 - \frac{2M}{r}\right)\frac{1}{r^2}. \tag{A.18}$$

A change of variable to $u = R/r$ produces an easier equation to solve:

$$\left(\frac{du}{d\phi}\right)^2 = 1 - u^2 - \frac{2M}{R}(1 - u^3). \tag{A.19}$$

Copyright Taylor & Francis 2018.

Figure A.3. Three trajectories of light showing the bending arising from the sun's mass. Note that there are three images formed on the right. Actually, as indicated by the ellipse, an observer would see a circle (an Einstein ring) obtained by rotating this figure along the x axis.

1. Verify that an approximate solution to (A.19) is

$$\phi \simeq 4\frac{GM}{Rc^2}. \tag{A.20}$$

2. Evaluate this expression to determine a numerical value for the angle of deflection for light grazing the sun's surface (*hint:* It's small). Use parameters $M = 2\times10^{23}$ grams, $R = 7 \times 10^10$ cm, and $G/c^2 = 7.4 \times 10^{-29}$ cm/gram.

3. Although the ODE (A.19) is nonlinear, that is not an obstacle for a numerical solution. Solve (A.19) numerically and compare your result with the value from the approximate analytic expression.

A.4 Gravitational Lensing

In a different approach to the deflection of a light due to a very massive star, [Moore(13)] assumes a Schwarzschild spacetime to describe the curved space outside of a spherically symmetric gravitational source (star). In terms of the inverse variable $u = 1/r$, the geodesic equation is now

$$\frac{d^2u}{d\phi^2} = 3GM\,u^2 - u. \tag{A.21}$$

1. Modify your ODE solver appropriate to this equation. Employ units such that mass is measured in meters, GM=1477.1 m, and $M = 28M_\odot$ ($M_\odot$ is a solar mass).

Copyright Taylor & Francis 2018.

Figure A.4. Relativistic and Newtonian potential for $\ell/M = 4.3$. Different energies would correspond to differing values of the ordinate. One of the dots corresponds to the energy for a circular orbit.

2. Equation (A.21) is quite sensitive to the initial conditions. Assume that initially the light is very distant: $r = 10^6$, and $u(\phi = 0) = du(\phi)/d\phi = 10^{-6}$.
3. Convert your solution for $r(\phi)$ into one for (x, y), and plot up the photon paths for $0 \leq \phi \leq \pi$. Our plot is given in Figure A.3.
4. Employ the symmetry of this problem to rotate your solution about the x axis and thus obtain a circle. This is what an observer sees when viewing a distance light source lying behind a massive star that focuses the point source into a ring.

Our program `LensGravity.py` is given in Listing A.3.

A.5 Particle Orbits in GR Gravity

The classical solution of Newton's laws for the gravitational potential is just fine for most everything. However there are small corrections arising from relativity, and while small, these corrections are actually critical to the accuracy of modern gps devices. The usual approach is to determine an ODE with a GR correction to the familiar $1/r$ gravitational potential, and then solve the ODE approximately or numerically. We follow [Hartle(03)] and [Moore(13)] who derive an effective potential appropriate

Copyright Taylor & Francis 2018.

to the empty space external to a spherically symmetric star. For the Schwarzschild metric (A.3), they give the effective radial potential as

$$V_r(r) = -\frac{GM}{r} + \frac{\ell^2}{2r^2} - \frac{GM\ell^2}{r^3}, \tag{A.22}$$

where G is the gravitational constant, ℓ is the angular momentum per unit rest mass, M is the mass of the star, and the middle term is the usual angular momentum barrier. We see that (A.22) differs from the Newtonian potential by a $-GM\ell^2/r^3$ term that provides an strong attraction at very short distances, in addition to the usual $-GM/r$ attraction. We obtain a dimensionless, and simpler-to-compute, form of the potential by change of variables:

$$V_r(r') = -\frac{G}{r'} + \frac{\ell'^2}{2r'^2} - \frac{G\ell'^2}{r'^3}, \tag{A.23}$$

$$r' = \frac{r}{M}, \qquad \ell' = \frac{\ell}{M}. \tag{A.24}$$

1. Plot $V_r(r')$ versus r' for $\ell' = 4.3$ (FigureA.4).
2. Describe in words how the orbits within this potential change with energy. (*Hint:* one of the dots in Figure A.4 corresponds to the energy for a circular orbit.)
3. At what values of r' does the effective potential have a maximum and a minimum?
4. At what value of r' does a circular orbit exist?
5. Determine the range of r' values that occur for $\ell' = 4.3$.
6. Indicate the above range on your plot by a horizontal line, and describe the orbits.
7. Describe the orbit for energies corresponding to the maximum in the potential.

A.5.1 Orbit Computation

A fairly simple way to determine the orbits of massive particles in the effective potential (A.23) is based on energy conservation. It starts with the energy per unit mass expressed as the sum of kinetic and potential terms: [Moore(13)]:

$$E = \frac{1}{2}\left(\frac{dr}{d\phi}\right)^2 \frac{\ell^2}{r^4} - \frac{GM}{r} + \frac{\ell^2}{2r^2} - \frac{GM\ell^2}{r^3}, \tag{A.25}$$

Copyright Taylor & Francis 2018.

Figure A.5. *Left:* An orbit corresponding to an energy at the maximum of the potential. *Right:* A rapidly precessing orbit.

where ϕ is the polar angle. We obtain an ODE for the orbit by differentiating both sides of the equation with respect to ϕ:

$$\frac{d^2r}{d\phi^2} = -\frac{GM}{r^2} + \frac{\ell^2}{r^3} - \frac{3GM\ell^2}{r^4}, \tag{A.26}$$

where a common $dr/d\phi$ factor cancels out. The ODE is simplified by a change of variables:

$$\frac{d^2u'}{d\phi^2} = -u' + \frac{G}{\ell'^2} + 3Gu'^2, \tag{A.27}$$

$$u' = \frac{M}{r}, \quad \ell' = \frac{\ell}{M}. \tag{A.28}$$

As with Newtonian orbits, the energy of the system determines the orbit characteristics. For a numerical solution we use the energy integral to determine the initial conditions for the ODE. Specifically, the energy integral (A.25) can be solved for $du'/d\phi$:

$$\frac{du'}{d\phi} = \sqrt{\frac{2E}{\ell'^2} + 2\frac{Gu'}{\ell'^2} - u'^2 + 2Gu'^3}\,. \tag{A.29}$$

As you (should) have deduced qualitatively, the potential (A.23) produces qualitatively differing orbits depending upon the system's energy and angular momentum. The problems of this section ask you to use your ODE solver to explore numerically

Copyright Taylor & Francis 2018.

and graphically various orbits corresponding to differing initial conditions and energies. Our program `RelOrbits.py` is in Listing A.4 and runs in Spyder. Note that when you produce your graphs you should introduce some signal into your figures so that you can deduce in which direction the orbiting particle moves, something we have not done it in Figure A.5. Alternatively, you can produce animations or a time series of frames, in which case the direction of motion will be evident.

1. Set up your ODE solver appropriate for (A.29) using G =1. *Hint:*

$$y[1] = \sqrt{\frac{2E}{\ell'^2} + 2\frac{Gu'}{\ell'^2} - u'^2 + 2Gu'^3}, \tag{A.30}$$

$$y[0] = \sqrt{\frac{2(-0.028)}{4.3^2} + 2\frac{y[0]}{4.3^2 - y[0]^2} + 2y[0]^3}. \tag{A.31}$$

2. Choose an energy corresponding to the maximum of the effective potential compute your version of Figure A.4, and an initial r value at which the potential is a maximum. As you may have deduced, this should lead to an unstable orbit such as on the left of Figure A.5.

3. See if you can find initial conditions that lead to a circular orbit. Is it stable?

4. Investigate the effect of gradually decreasing the angular momentum.

5. Choose an energy that corresponds to the minimum in the effective potential and plot the orbits. Examine the sensitivity of these orbits to the choice of initial conditions.

6. Determine an energy and initial conditions that produce a precessing perihelion, such as seen on the right of Figure A.5. In this case the massive particle moves between two turning points, as shown by the horizontal line in the potential well in Figure A.4.

7. Examine the orbits that occur if a particle is bound by the inner strong attraction. Can such a particle start at infinity and be captured?

A.6 Riemann and Ricci Tensors

Figure A.6 shows two free particles moving along the infinitesimally close geodesics $x^a(\tau)$ and $x^b(\tau)$. We consider x^a as the reference particle with $u^\mu \equiv dx^\mu/d\tau$ its 4-velocity. The two trajectories start off parallel at time $\tau = 0$ and are connected by the vector $n(\tau)$:

$$x^a = x^b + n^\alpha(\tau). \tag{A.32}$$

Copyright Taylor & Francis 2018.

Figure A.6. Two free particles move along the infinitesimally close geodesics $x^a(\tau)$ and $x^b(\tau)$. The particles start off parallel at time $\tau = 0$ and are connected by the vector $n(\tau)$.

If the relative acceleration of the particles is zero, then the geodesics remain parallel and so:

$$\frac{d^2 n}{d\tau^2} = 0.$$

This derivative acts on the basics vectors, which in turn requires knowledge of the Christoffel symbols:

$$\left(\frac{d^2 n}{d\tau^2}\right)^\alpha = \left(\partial_\sigma \Gamma^\alpha_{\mu\nu} - \partial_\nu \Gamma^\alpha_{\mu\sigma} + \Gamma^\alpha_{\sigma\gamma}\Gamma^\gamma_{\mu\nu} - \Gamma^\alpha_{\nu\gamma}\Gamma^\gamma_{\mu\sigma}\right) u^\sigma u^\mu u^\nu.$$

The quantity in parenthesis is called the *Riemann tensor*:

$$R^\alpha_{\mu\nu\sigma} = \partial_\sigma \Gamma^\alpha_{\mu\nu} - \partial_\nu \Gamma^\alpha_{\mu\sigma} + \Gamma^\alpha_{\sigma\gamma}\Gamma^\gamma_{\mu\nu} - \Gamma^\alpha_{\nu\gamma}\Gamma^\gamma_{\mu\sigma}. \tag{A.33}$$

A.6.1 Problems

The following three problems can all be solved with variations of the same code.

1. Use `sympy` to evaluate the Riemann tensor for the Schwarzchild metric.
2. Use `sympy` to extract the Ricci curvature tensor, defined as the contraction

   $$R_{\lambda\mu} \equiv R^\alpha_{\lambda\alpha\gamma} \tag{A.34}$$

   (*note* the implicit sum over α).
3. The Ricci scalar gives a single numerical measure of the curvature at each point in spacetime. It is defined as the contraction:

   $$R \equiv R^\lambda_\lambda = g^{\lambda\gamma} R_{\lambda\gamma}. \tag{A.35}$$

   If a spacetime is flat, then $R = 0$ and the initially parallel geodesics remain so in time. If a spacetime is curved, then $R \neq 0$. Use `sympy` to extract the Ricci scalar from Ricci curvature tensor.

Copyright Taylor & Francis 2018.

A.6.2 Help with Solution

1. Use the previously–developed code to create four matrices containing the Christoffel symbols, $\Gamma^0_{\mu\nu}$, $\Gamma^r_{\mu\nu}$, $\Gamma^\theta_{\mu\nu}$, and $\Gamma^\phi_{\mu\nu}$.
2. Define a 4-D array for the Riemann tensor $R^\alpha_{\lambda\alpha\gamma}$ with the indices corresponding to α, μ, ν, and σ. (There are four indices with each index having a range of 4.)
3. To deduce the Ricci curvature tensor $R_{\lambda\mu}$, define a 2-D array with each index having a range of 4.
4. Extract the Ricci scalar. Our version of said program is called `Ricci.py` and can be found in Listing A.5.

A.7 General Relativity Code Listings

```
# Wormhole.py: Symbolic evaluation of wormhole derivative

from sympy import *
L, x, M, rho, a, r,  lp= symbols('L x M rho a r lp')
x = (2*L-a)/(pi*M)
r = rho+M*(x*atan(x) -log(1+x*x)/2)
drdL = diff(r,L)
print ('drdL(raw) = ', drdL)
drdL = simplify(drdL)
print  (' And finally! dr/dL (simplified)=', drdL)
```

Listing A.1. **WormHole.py** evaluates symbolically a derivative needed in description of wormhole.

```
# VisualWorm.ipynb    Visualize wormhole with Vpython in notebook

from vpython import *
import numpy as np
import math

escene = canvas(width=400,height=400, range= 15)
a = 1 #2a is height inner cylinder
ring(pos=vector(0,0,0),radius=1,axis=vector(0,1,0),color=color.yellow)

def f(x):       # function to be integrated
    M = 0.5         # black hole mass
    a = 1           # 2a: cylinders height
    y = np.sqrt(1- (2*np.arctan(2*(x - a)/(np.pi*M))/np.pi)**2)
    return y

def trapezoid(Func,A,B,N):
    h = (B - A)/(N )          # step, A:initial, B:end
```

Copyright Taylor & Francis 2018.


```
    sum = (Func(A)+Func(B))/2  # initialize, (first + last)/2
    for i in range(1, N):      # inside
        sum += Func(A+i*h)     #
    return h*sum               # sum times h

def radiuss(L):   # radius as function of L
    ro = 1        # radius of cylinder (a/ro=1)
    a = 1         # 2a: height of inner cylinder
    M = 0.5       # black hole (mass M/ro)=1
    xx = (2*(L-a))/(np.pi*M)
    p = M*(xx*np.arctan(xx))
    q = -0.5*M*math.log(1+xx**2)
    r = ro+ p+q
    return r

for i in range(1,12): # to plot 2 rings (at z ant -z)
    A = 0                 #limits of integration
    B = i
    N = 300       # trapezoid rule points
    if i>6: N = 600   # more points
    z = trapezoid(f,A,B,N) # returns z
    L = i+1
    rr = radiuss(L)          # radius
    ring(pos=vector(0,z,0),radius=rr,axis=vector(0,1,0),
        color=color.yellow)
    ring(pos=vector(0,-z,0),radius=rr,axis=vector(0,1,0),
        color=color.yellow)
```

Listing A.2. **VisualWorm.ipynb** A Vpython visualization of a wormhole from within Jupyter notebook.

```
# LensGravity.py    Deflection of light by sun wi Matplotlib

import numpy as np
import matplotlib.pyplot as plt
y = np.zeros((2),float);      ph = np.zeros((181),float)
yy = np.zeros((181),float);   xx = np.zeros((181),float)
rx = np.zeros((181),float);   ry = np.zeros((181),float)
Gsun = 4477.1                     # Sum mass x G (m)
GM = 28.*Gsun                     # Sun mass
y[0] = 1.e-6; y[1] = 1e-6         # Initial condition for u=1/r

def f(t,y):                              # RHS, can modify
    rhs = np.zeros((2),float)
    rhs[0] = y[1]
    rhs[1] = 3*GM*(y[0]**2)-y[0]
    return rhs

def rk4Algor(t, h, N, y, f):    # Do not modify
    k1 = np.zeros(N); k2=np.zeros(N); k3=np.zeros(N); ↩
        k4=np.zeros(N);
    k1 = h*f(t,y)
    k2 = h*f(t+h/2.,y+k1/2.)
    k3 = h*f(t+h/2.,y+k2/2.)
    k4 = h*f(t+h,y+k3)
    y = y+(k1+2*(k2+k3)+k4)/6.
    return y
```

Copyright Taylor & Francis 2018.


```
f(0,y)                          # Initial conditions
dphi = np.pi/180.               # 180 values of angle phi
i = 0
for phi in np.arange(0,np.pi+dphi,dphi):
    ph[i] = phi
    y = rk4Algor(phi,dphi,2,y,f)            # Call rk4
    xx[i] = np.cos(phi)/y[0]/1000000        # Scale for graph
    yy[i] = np.sin(phi)/y[0]/1000000
    i = i+1
m = (yy[180] - yy[165])/(xx[180]-xx[165])   # Slope of straight ↩
    line
b = yy[180] - m*xx[180]                     # Intercept of line
j = 0

for phi  in np.arange(0,np.pi+dphi,dphi):
    ry[j] = m*xx[j] + b             # Eqn straight line
    j=j+1
plt.figure(figsize=(12,6))
plt.plot(xx,yy)                     # Straight line light tajectroy
plt.plot(xx,-yy)                    # Symmetric for negative y
plt.plot(0,0,'ro')                  # Mass at origin
plt.text(0.02,0.02, 'Sun')
plt.plot(0.98,0,'bo')               # Source
plt.plot(0.98,1.91,'go')            # Position source seen by O
plt.plot(0.98,-1.91,'go')
plt.text(1,0,'S observer')
plt.text(-1.00, 0.20,'Source')
plt.text(1.02, 1.91,"S' observer")
plt.text(1.02,-2,"S'' observer")
plt.plot([0],[3.])                  # Invisible point
plt.plot([0],[-3.])                 # Invisible point at -y
plt.plot(xx,ry)                     # Upper straight line
plt.plot(xx,-ry)                    # Lower straight line
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

Listing A.3. **LensGravity.py** solves for orbits of light around very massive star.

```
# RelOrbits.py: Reltiv orbits in gravitational pot (needs rk4)

import matplotlib.pyplot as plt
import numpy as np

dh = 0.04;  dt = dh;  ell = 4.3;  G = 1.0;  N = 2
E = 0.040139
phi = np.zeros((944),float)
rr =  np.zeros((944),float)
y =   np.zeros((2),float)
y[0] = 0.052
y[1] = np.sqrt(2*E/ell**2 + 2*G*y[0]/ell**2-G*y[0]**2+2*G*y[0]**3)

def f(t,y):
    rhs = np.zeros(2)
    rhs[0] = y[1]
    rhs[1] = -y[0]+G/ell**2 +3*G*y[0]**2
    return rhs
```

Copyright Taylor & Francis 2018.


```
f(0,y)
i = 0
for fi in np.arange(0,5.8*np.pi,dt):
    y = rk4(fi,dt,N,y,f)
    rr[i] = (1/y[0])*np.sin(fi) # Notice 1/r (=u)
    phi[i] = (1/y[0])*np.cos(fi)
    i = i+1
f1 = plt.figure()
plt.axes().set_aspect('equal') # Aspect ratio equal
plt.plot(phi[:455],rr[:455])
plt.xlabel("r/M")
plt.show()
```

Listing A.4. **RelOrbits.py** solves for orbits of a massive particle in a gravitational potential with a GR correction.

```
# Ricci.py:  Riemann & Ricci tensors, Ricci scalar via sympy

from sympy import *
import numpy as np

t,r,th, fi, rg = symbols('t r th fi rg') # Schwarzchild metric
print("contravariant")                   # Upper indices

# Inverse matrix
gT = Matrix([[1/(-1 + rg/r), 0, 0, 0], [0, 1 - rg/r, 0, 0],
    [0, 0, r**(-2), 0], [0, 0, 0, 1/(r**2*sin(th)**2)]])

# 4-Dim array for alpha, beta, mu, nu
Ri = [[[[[] for n in range(4)] for a in range(4)] for b in range(4)]
  for c in range(4)]
RT = [[[] for m in range(4)]for p in range(4)]        # Ricci tensor

# Christoffel symbols upper index t,r,theta and phi
Cht = Matrix([[0, 0.5*rg/(r*(r-rg)), 0, 0],
  [0.5*rg/(r*(r-rg)),0,0,0], [0, 0, 0, 0], [0, 0, 0, 0]])
Chr = Matrix([[0.5*rg*(r-rg)/r**3,0,0,0], ↩
    [0,-0.5*rg/(r*(r-rg)),0,0],
  [0,0, -1.0*r + 1.0*rg, 0], [0,0,0, (-1.0*r + rg)*sin(th)**2]])
Chth = Matrix([[0, 0, 0, 0], [0, 0, 1.0/r, 0], [0, 1.0/r, 0, 0],
              [0, 0, 0, -0.5*sin(2*th)]])
Chfi = Matrix([[0,0,0,0], [0, 0, 0, 1.0/r], [0,0,0,1./tan(th)],
              [0, 1./r, 1.0/tan(th), 0]])
for alpha in range(0,4): # Upper index
   if alpha == 0:       Chalp = Cht
   elif alpha == 1:     Chalp = Chr
   elif alpha == 2:     Chalp = Chth
   else:                Chalp = Chfi
   for be in range(0,4):   # Beta
    for mu in range(0,4):
       if mu == 0:      der2 = t
       elif mu == 1:    der2 = r
       elif mu == 2:    der2 = th
       elif mu == 3:    der2 = fi
       for nu in range(0,4):
         if nu == 0:    der1 = t  # Derivative
         elif nu == 1:  der1 = r
```

Copyright Taylor & Francis 2018.


```
            elif nu == 2: der1 = th
            elif nu == 3: der1 = fi
            a1 = diff(Chalp[be,nu],der2)  # Christoffel symbol
            a2 = diff(Chalp[be,mu],der1)  # Symbol and derivative
            sump = 0 #  Einstein convention
            sumn = 0 #  Einstein convention
            for gam in [t,r,th,fi]:
                if gam == t:
                    Chgam = Cht
                    gama = 0
                elif gam == r:
                    Chgam = Chr
                    gama = 1
                elif gam == th:
                  Chgam = Chth
                  gama = 2
                elif gam == fi:
                  Chgam = Chfi
                  gama = 3
                sump = sump + Chalp[mu,gama]*Chgam[be,nu]
                sumn = sumn + Chalp[nu,gama]*Chgam[be,mu]
            R = simplify(a1-a2+sump-sumn) # Riemann tensor
            if R == 0:  Ri[alpha][be][mu][nu] = 0
            else :
              Ri[alpha][be][mu][nu] = R
              print("Ri[",alpha,"][",be,"][",mu,"][",nu,"]=", ↩
                  Ri[alpha][be][mu][nu])
print("\n")
print("Ricci Tensor\n")
for ro in range(0,4):      # Find Ricci tensor
     for de in range (0,4):
            sum = 0
            for alp in range (0,4):   sum = sum+Ri[alp][ro][alp][de]
            RT[ro][de] = simplify(sum)
            print("RT[",ro,"][",de,"]",RT[ro][de]) # Ricci's tensor
sumR = 0  # Ricci Scalar
for be in range(0,4):
    for nu in range(0,4):     sumR = sumR+gT[be,nu]*RT[be][nu]
    print(sumR)
RS = (sumR)
print("RS",RS)     # Ricci Scalar R
```

Listing A.5. **Ricci.py** uses `sympy` to evaluate the Riemann tensor, the Ricci curvature tensor, and the Ricci scalar for the Schwarzchild metric. .

Copyright Taylor & Francis 2018.

Copyright Taylor & Francis 2018.

B

Web Materials

Python Code Directory

Name	Listing	Description	Name	Listing	Description
AdvecLax.py	4.5	Advection Eq	Beam.py	4.9	Navier-Stokes
Bisection.py	2.6	Bisection Algor	BoundCall.py	6.5	P Space Bound
Bugs.py	8.1	Logistic Map	CatFriction.py	4.2	Catenary Waves
CavityFlow.py	4.11	Cavity Flow	CentralValue.py	1.11	Central Value
CircPolarztn.py	5.11	Circular Polarized	CoulWF.py	6.16	Coulomb WF
CWT.py	2.4	Wavelet TF	DecaySound.py	7.3	Spontan Decay
DFTcomplex.py	2.1	DTF	DielectVis.py	5.10	FDTD Dielectric
DielectVis.py	5.10	FDTD Animate	DLA.py	7.6	Aggregation
DWT.py	2.5	Discrete Wavelets	EasyMatPlot.py	1.3	MatPlotLib e.g.
EasyVisualVP.py	1.1	VPthonl Plot	Entangle.py	6.21	Entangled QM
EqHeat.py	7.1	Heat Eq	EqHeatMov.py	7.2	Heat Eq Animate
EqStringMovMat.py	4.1	String Animated	FDTD.py	5.9	FDTD
Fern3D.py	8.4	Fern Fractal	FFTmod.py	2.3	FFT
Fit.py	2.8	Least Sq Fit	ForcedOscillate.py	3.1	ODE eg
GaussPoints.py	1.17	Gauss Points	GlauberState.py	6.19	Glauber States
GradesMatplot.py	1.4	MatPlotLib eg	Hdensity.py	6.8	H Density
HOmovSlow.py	6.9	HO Packet Slow	HOmov.py	6.10	HO Animate
HOanal.py	6.2	HO Analytic	HOchargeMat.py	6.13	Charged HO
HOchargeMat.py	6.13	Charged HO	HOnumeric.py	6.1	HO WF Numer
HOpacket.py	6.11	HO Packet Viz	HOpacketMat.py	6.12	HO Packet Mat
Hyperfine.py	6.20	H Hyperfine	ImagePlaneMat.py	5.5	Images Plane
ImagePlaneVP.py	5.6	Images Plane	ImageSphereVP.py	5.7	Images Sphere
IntegGaussCall.py	1.16	Gauss Integrate	IsingVP.py	7.9	Ising Model
LaplaceCyl.py	5.4	Lapace's Eq	LaplaceDisk.py	5.3	Laplace Eq Disk
LaplaceLine.py	5.2	Laplace's Eq Line	LaplaceTri.py	5.1	Laplace Triangle
LensGravity.py	A.3	Gravitational Lens	LorentzFieldVP.py	5.18	Lorentz TF Fields
MatPlot2figs.py	1.5	MatPlotLib eg	NewtonCall.py	2.7	Newton Search
MD1D.py	7.10	1-D MD	MD2D.py	7.11	2-D MD
MDpBC.py	7.12	Periodic BC MD	Permute.py	7.8	State Counting
Plm.py	6.7	Legendre Polys	PondMatPlot.py	1.7	Scatter Plot
PredatorPrey.py	8.2	Predator Prey	ProjectileAir.py	3.4	Projectile ODE
ProteinFold.py	8.3	Protein Folding	QMC.py	6.23	Path Integration
QuantumEigenCall.py	6.3	QM Eigenvalue	QuantumNumerov.py	6.4	Numerov Mth
QuarterPlate.py	5.12	1/4 Wave Plate	RelOrbits.py	A.4	Relatvist Orbits
Ricci.py	A.5	Ricci Tensor	rk4Algor.py	1.14	rk4 Algorithm
rk4Call.py	1.12	Calls rk4	rk4Duffing.py	1.13	rk4 e.g.
SincFilter.py	2.2	Noise Filtering	ScattSqWell.py	6.14	Sq Well Scatt
Scatter3dPlot.py	1.8	Scatter Plot	Simple3Dplot.py	1.6	Surface Plot
SlidingBox.py	1.10	Animate	Soliton.py	4.6	KdeV Soliton
SolitonAnimate.py	4.7	Soliton Movie	SqBilliardCM.py	3.3	Square Billiards
SU3.py	6.22	SU(3) Quarks	–		

Copyright Taylor & Francis 2018.

Name	Listing	Description	Name	Listing	Description
`3GraphVP.py`	1.2	VPython Plot	`3QMdisks.py`	6.17	3 QM Disks
`TeleMat.py`	5.13	Telegraph Eq	`TeleVis.py`	5.14	Telegraph Viz
`ThinFilm.py`	5.15	Thin Film	`Torricelli.py`	4.10	Torricelli Flow
`TrapMethods.py`	1.15	Trapezoid Integ	`TwoCharges.py`	5.16	Lorentz TF Q's
`TwoFields.py`	5.17	Lorentz Field	`TwoDsol.py`	4.8	2-D Soliton
`TwoForces.py`	1.9	Animate e.g.	`TwoWells.py`	6.18	T-Dep 2 Wells
`UranusNeptune.py`	3.5	Uranus Orbit	`VisualWorm.ipynb`	A.2	Worm Hole Viz
`Walk.py`	7.4	Random Walk	`Walk3D.py`	7.5	3D Random Walk
`Waves2D.py`	4.4	2-D Waves Num	`Waves2Danal.py`	4.3	2-D Waves Anal
`WormHole.py`	A.1	Worm Hole Derv			