```
function geodesic(christoffel)
    """
    Returns the function f where \vec{x}'(tau) = f(tau, \vec{x})
    for a geodesic in the spacetime defined by the christoffel symbols.
    """
    function xddot(mu, xdot, affine)
        sum = 0
        for a=1:4
            for b=1:4
                sum -= affine[mu, a, b] * xdot[a] * xdot[b]
            end
        end
        sum
    end
    function func(tau, y)
        """
       tau is proper time
       y is [t, r, theta, phi, tdot, rdot, thetadot, phidot]
       """
        affine = christoffel(y[1:4])
        f = zeros(8)
        f[1], f[2], f[3], f[4] = y[5:8]
        for mu=1:4
            f[mu+4] = xddot(mu, y[5:8], affine)
        end
        f
    end
    func
end


function Schwarzschild(;G=1, M=1, c=1)
    """
    :return: the function f where \vec{x}'(tau) = f(tau, \vec{x(t)}).
    """
    function Christoffel(x)
        """
       Computes all the Christoffel symbols for the Schwartzschild
       metric at t, r, theta, phi.
       x = [t, r, theta, phi].
       """
        t, r, theta, phi = x
        Gamma = zeros(4, 4, 4)
        Gamma[1, 2, 1] = G*M / (r*(c^2*r-2*G*M))
        Gamma[2, 1, 1] = G*M*(1-(2*G*M)/(r*c^2)) / r^2
        Gamma[2, 2, 2] = G*M / (2*G*M*r-c^2*r^2)
        Gamma[2, 3, 3] = 2*G*M/c^2 - r
        Gamma[2, 4, 4] = (2*G*M-r*c^2)*sin(theta)^2 / c^2
        Gamma[3, 3, 2] = 1 / r
        Gamma[3, 4, 4] = -cos(theta)*sin(theta)
        Gamma[4, 4, 2] = 1 / r
        Gamma[4, 4, 3] = cot(theta)
        Gamma
    end
    geodesic(Christoffel)
end

gfunc = Schwarzschild()
tau, taumax = 0, 35.56
t0, r0, theta0, phi0 = 0, 10, pi/2.0, 0
#u^mu u_mu = -c^2
tdot0 = sqrt(1-2/r0)^-1
y0 = [t0, r0, theta0, phi0, tdot0, 0, 0, 0]

""" Used AdamsBash4 when running fraction_gwloss. Otherwise, used SimpleErrorStep. """
```

```
#taus, xs = AdaptiveStepper(gfunc, SimpleErrorStep, tau, y0, taumax)
h = 0.01
taus, xs = MultiStepper(gfunc, AdamsBash4, tau, y0, taumax, h)

ts = [a[1] for a=xs]
rs = [a[2] for a=xs]

using PyPlot
plot(ts, rs, "o-", label="coordinate time r(t)")
plot(taus, rs, "o-", label="proper time r(tau)")
xlabel("time")
ylabel("r")
title("Schwarzschild Metric")
legend()


function fraction_gwloss(rs, h)
    energy_radiated = 0
    for i=3:length(rs)-2
        r = rs[i]
        #estimate first, second, and third derivatives.
        rdot = (rs[i+1] - rs[i-1]) / (2*h)
        rddot = (rs[i+1] + rs[i-1] - 2*r) / (h^2)
        rdddot = (rs[i+2] - rs[i-2] - 2*rs[i+1] + 2*rs[i-1]) / (2*h^3)

        power = (8/15.0) * (3*rdot*rddot + r*rdddot)^2
        energy_radiated += power * h
    end

    #1 is the mass energy in natural units.
    energy_radiated / (1 + energy_radiated)
end

println(fraction_gwloss(rs, h))



function Kerr(;G=1, M=1, c=1, L=0)
    """
    :return: the function f where \vec{x}'(tau) = f(tau, \vec{x(t)}).
    """
    function metric(x)
        """
        The Kerr metric. Returns a 4x4 matrix.
        """
        t, r, theta, phi = x
        a = L / (M * c)
        rho = r^2 + a^2 * cos(theta)^2
        delta = r^2 - 2*G*M*r/c^2 + a^2

        g = zeros(4, 4)
        g[1, 1] = - (delta - a^2 * sin(theta)^2) / rho * c^2                    #dt
        g[2, 2] = rho / delta                                                    #dr
        g[3, 3] = rho                                                            #d(theta)
        g[4, 4] = (a^4+r^4+2*r^2*a^2-a^2*delta*sin(theta)^2)*sin(theta)^2 / rho #d(phi)
        g[1, 4] = (delta - r^2 - a^2) * (2 * a * c * sin(theta)^2) / rho        #dt d(phi)
        g[4, 1] = g[1, 4]                                                        #d(phi) dt

        g
    end

    ### Approximations to partial derivatives of metric ###
    #Map indices to the correct derivative metric. This particular
    #metric does not depend on t or phi so their derivatives are just 0.
```

```
    deriv = [
        x -> zeros(4, 4), x -> Derivative(x, metric, [0, 1, 0, 0]),
        x -> Derivative(x, metric, [0, 0, 1, 0]), x -> zeros(4, 4)
    ]

    ### Christoffel symbols ###
    function Christoffel(x)
        """
       Computes all the Christoffel symbols for the Kerr-Newman
       metric at t, r, theta, phi with approximations.
       x = [t, r, theta, phi].
       """
         #Evaluate partial derivative metric with repsect to each coordinate.
         D = [d(x) for d in deriv]
         invmetric, Gamma = inv(metric(x)), zeros(4, 4, 4)
         for i=1:4
             for j=1:4
                 for k=1:4
                     for a=1:4
                         Gamma[i, j, k] += .5*invmetric[a, i]*(D[j][a, k]+D[k][a, j]-D[a][j, k])
                     end
                 end
             end
         end
         Gamma
    end

    geodesic(Christoffel)
end

#Default arguments approximates the Schwarzschild metric.
gfunc = Kerr(L=0)
tau, taumax = 0, 32.78
t0, r0, theta0, phi0 = 0, 10, pi/2.0, 0
#u^mu u_mu = -c^2
tdot0 = sqrt(1-2/r0)^-1

y0 = [t0, r0, theta0, phi0, tdot0, 0, 0, 0]
taus, xs = AdaptiveStepper(gfunc, RKTrapStep, tau, y0, taumax, 10.0^-4)
#taus, xs = AdaptiveStepper(gfunc, CarpStep, tau, y0, taumax)

ts = [a[1] for a=xs]
rs = [a[2] for a=xs]
#thetas = [a[3] for a=xs]
phis = [a[4] for a=xs]

using PyPlot
figure(0)
plot(ts, rs, "o-", label="coordinate time r(t)")
plot(taus, rs, "o-", label="proper time r(tau)")
xlabel("time")
ylabel("r")
title("Kerr Metric with L = 0")
legend(loc=1)


figure(1)
plot(ts, phis, "o-", label="coordinate time phi(t)")
plot(taus, phis, "o-", label="proper time phi(tau)")
xlabel("time")
ylabel("radians")
title("Kerr Metric with L = 0")
legend()
```