J. Korean Soc. Math. Educ. Ser. B: Pure Appl. Math.
https://doi.org/10.7468/jksmeb.2023.30.1.83
Volume 30, Number 1 (February 2023), Pages 83–108

ISSN(Print) 1226-0657
ISSN(Online) 2287-6081

NUMERICAL GEODESICS ON A SURFACE WITH PYTHON

Jong Ryul Kim

Abstract. We find geodesics on a surface numerically by using Runge-Kutta method with Python.

1. Introduction

In [2], we explained geodesics on the Poincaré disk and showed numerically a shortest path joining two points in the Poincaré disk by calculating length. Andrijana Burazin in a thesis titled 'Calculating Geodesics on Surfaces' calculated geodesics with Maple [1]. In this paper, solving the geodesic equations numerically by using Runge-Kutta method with Python, we show geodesics on the the Poincaré upper half plane, Poincaré disk and on a surface with a coordinate chart.

2. Geodesic Equations

We introduce basic notations in [4]. Let S be a regular surface in 3-dimensional Euclidean space $\mathbb{R}^3$. The first fundamental form

$$I_p : T_pS \times T_pS \longrightarrow \mathbb{R}, \quad I_p(v, w) = v \cdot w$$

is the inner product on the tangent space T_pS at $p \in S$ of a surface S induced by the dot product of $\mathbb{R}^3$. Let $\mathbf{X} : U \subseteq \mathbb{R}^2 \longrightarrow S \subset \mathbb{R}^3$ be a coordinate chart of a surface S, that is,

$$\mathbf{X}(u, v) = (x(u, v), y(u, v), z(u, v)).$$

Let $\alpha : [t_0, t_1] = I \longrightarrow S \subset \mathbb{R}^3$ be a curve in S such that

$$\alpha(t) = \mathbf{X}(u(t), v(t))$$

Received by the editors December 16, 2022. Accepted January 25, 2023.
2020 *Mathematics Subject Classification.* 53A35, 53C22, 65L06.
Key words and phrases. Poincaré disk, Runge-Kutta method, geodesic equations, line element.

for a curve $c(t) = (u(t), v(t)) \subset U \subseteq \mathbb{R}^2$. The length of a curve from $\alpha(t_0)$ to $\alpha(t)$ is

$$s(t) = \int_{t_0}^{t} |\alpha'(r)|\, dr = \int_{t_0}^{t} \sqrt{I_{\alpha(r)}\big(\alpha'(r), \alpha'(r)\big)}\; dr.$$

Since $\alpha'(r) = \mathbf{X}_u \frac{du}{dr} + \mathbf{X}_v \frac{dv}{dr}$ and

$$\begin{aligned} I_{\alpha(r)}\big(\alpha'(r), \alpha'(r)\big) &= I_{\alpha(r)}\Big(\mathbf{X}_u \frac{du}{dr} + \mathbf{X}_v \frac{dv}{dr},\ \mathbf{X}_u \frac{du}{dr} + \mathbf{X}_v \frac{dv}{dr}\Big) \\ &= \Big(\mathbf{X}_u \cdot \mathbf{X}_u\Big)\Big(\frac{du}{dr}\Big)^2 + 2\Big(\mathbf{X}_u \cdot \mathbf{X}_v\Big)\Big(\frac{du}{dr}\Big)\Big(\frac{dv}{dr}\Big) + \Big(\mathbf{X}_v \cdot \mathbf{X}_v\Big)\Big(\frac{dv}{dr}\Big)^2 \\ &= E\Big(\frac{du}{dr}\Big)^2 + 2F\Big(\frac{du}{dr}\Big)\Big(\frac{dv}{dr}\Big) + G\Big(\frac{dv}{dr}\Big)^2, \end{aligned}$$

where we put $E = \mathbf{X}_u \cdot \mathbf{X}_u$, $F = \mathbf{X}_u \cdot \mathbf{X}_v$ and $G = \mathbf{X}_v \cdot \mathbf{X}_v$, we get

$$\frac{ds}{dt} = \sqrt{E\Big(\frac{du}{dt}\Big)^2 + 2F\Big(\frac{du}{dt}\Big)\Big(\frac{dv}{dt}\Big) + G\Big(\frac{dv}{dt}\Big)^2}.$$

Therefore we have the so-called line element

$$ds^2 = Edu^2 + 2Fdudv + Gdv^2.$$

The line element ds^2 of the Poincaré upper half plane, the Poincaré disk is

$$ds^2 = \frac{dx^2 + dy^2}{y^2}, \quad ds^2 = \frac{4(dx^2 + dy^2)}{(1 - (x^2 + y^2))^2},$$

respectively ([2, 4, 5, 6]).

The Christoffel symbols of S in the parametrization $\mathbf{X}$ are defined by

$$\begin{aligned} \mathbf{X}_{uu} &= \Gamma_{11}^1 \mathbf{X}_u + \Gamma_{11}^2 \mathbf{X}_v + I_p(\mathbf{X}_{uu}, N)N\,, \\ \mathbf{X}_{uv} &= \Gamma_{12}^1 \mathbf{X}_u + \Gamma_{12}^2 \mathbf{X}_v + I_p(\mathbf{X}_{uv}, N)N\,, \\ \mathbf{X}_{vv} &= \Gamma_{22}^1 \mathbf{X}_u + \Gamma_{22}^2 \mathbf{X}_v + I_p(\mathbf{X}_{vv}, N)N\,, \end{aligned}$$

where $N = \frac{\mathbf{X}_u \times \mathbf{X}_v}{|\mathbf{X}_u \times \mathbf{X}_v|}$. So we get

$$\Gamma_{11}^1 = \frac{GE_u - 2\,FF_u + FE_v}{2(EG - F^2)},\ \Gamma_{12}^1 = \frac{GE_v - FG_u}{2(EG - F^2)},\ \Gamma_{22}^1 = \frac{2\,GF_v - GG_u - FG_v}{2(EG - F^2)}\,,$$

$$\Gamma_{11}^2 = \frac{-FE_u + 2\,EF_u - EE_v}{2(EG - F^2)},\ \Gamma_{12}^2 = \frac{-FE_v + EG_u}{2(EG - F^2)},\ \Gamma_{22}^2 = \frac{-2\,FF_v + FG_u + EG_v}{2(EG - F^2)}\,.$$

For a orthonormal basis $\{\alpha',\, N,\, \alpha' \times N\}$ along a unit speed $\alpha(t)$, the geodesic curvature of a curve on a surface is defined $k_g = I_{\alpha(t)}(\alpha'', \alpha' \times N)$. If k_g is zero, we have the following differential geodesic equations([4])

$$\begin{aligned} u'' + \Gamma_{11}^1 (u')^2 + 2\,\Gamma_{12}^1 (u')(v') + \Gamma_{22}^1 (v')^2 &= 0\,, \\ v'' + \Gamma_{11}^2 (u')^2 + 2\,\Gamma_{12}^2 (u')(v') + \Gamma_{22}^2 (v')^2 &= 0\,. \end{aligned}$$

Solving the geodesic equations numerically by using Runge-Kutta method with Python, we find out geodesics on a surface in the next sections.

3. Numerical Solutions of Differential Equations

Example 1. Consider the following first order ordinary differential equation with initial value

$$\frac{dx(t)}{dt} = 3\,x(t) + 4\,t, \quad x(0) = 3, \quad 0 < t < 1.$$

The exact solution is $x(t) = \frac{31}{9}e^{3t} - \frac{4}{3}t - \frac{4}{9}$. Let us find the numerical solution by using Runge-Kutta method with Python ([3]).

```
import math
from math import *
import numpy as np
import matplotlib.pyplot as plt
def Runge_Kutta_1(f,t0,t,h,x0):
    # t0 time start, t time end, h increasing time value
    # x(0)=x_0 initial value
    xx,tt=[ ],[ ]
    xx.append(x0)
    tt.append(t0)
    while t0 < t:
        k1=f(t0,x0)
        k2=f(t0+h/2, x0+(k1*h)/2)
        k3=f(t0+h/2, x0+(k2*h)/2)
        k4=f(t0+h, x0+(k3*h))
        x1=x0+h*(k1+(2*k2)+(2*k3)+k4)/6
        xx.append(x1)
        t1=t0+h
        tt.append(t1)
        t0,x0=t1,x1
    return [np.array(tt),np.array(xx)]
f=lambda t, x:4*t+3*x
```

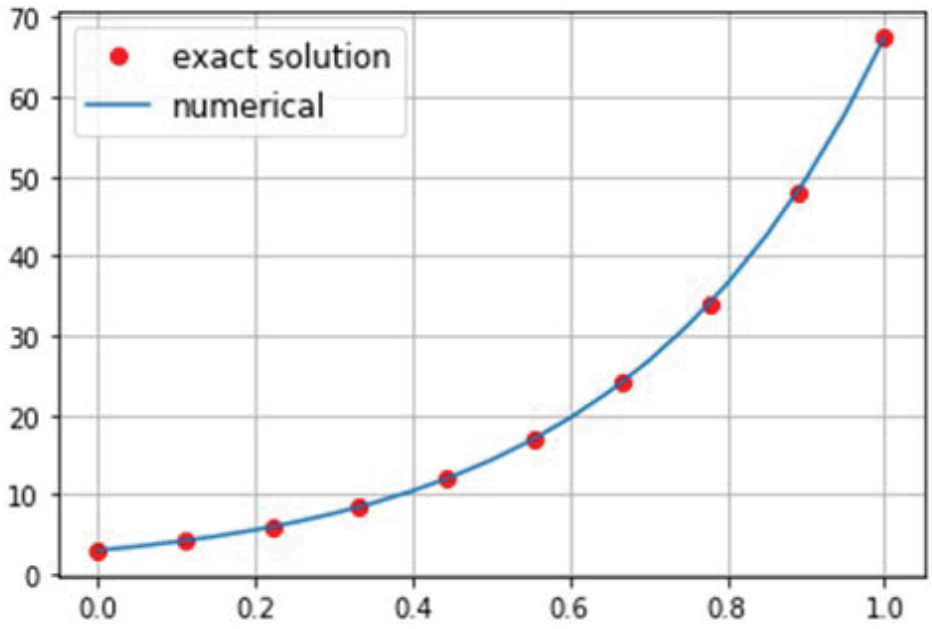

Figure 1. Runge_Kutta_1

```
rk=Runge_Kutta_1(f,0,1,0.05,3)
x=rk[0]
y=rk[1]
t=np.linspace(0,1,10)
g=lambda t:(31/9)*e**(3*t)-(4/3)*t-4/9
plt.plot(t,g(t),'ro',label='exact solution'),
plt.legend(loc='upper right',fontsize='large'),
plt.plot(x,y,label='numerical'),
plt.legend(loc='upper left',fontsize='large'),
plt.grid(True)
```

Figure 1

Example 2. Consider the following first order ordinary two differential equations with initial values

$$\frac{dx(t)}{dt} = -x + 2\,y, \quad \frac{dy(t)}{dt} = -2\,x - y, \quad x(0) = 1, \quad y(0) = 2, \quad 0 < t < 4.$$

The exact solution is

$$x(t) = e^{-t}(\cos(2t) + 2\sin(2t)), \quad y(t) = e^{-t}(2\cos(2t) - \sin(2t))\,.$$

Let us find the numerical solution by using Runge-Kutta method.

```
def Runge_Kutta_1_double(F,t0,t,h,x0,y0):
    # t0 time start, t time end, h increasing time value
    # x(0)=x_0, y(0)=y_0 initial values
    X0=np.array([x0,y0])
    xx,tt=[ ],[ ]
    xx.append(X0)
    tt.append(t0)
    while t0 < t:
        K1=F(t0,X0[0],X0[1])
        K2=F(t0+h/2, (X0+(K1*h)/2)[0],(X0+(K1*h)/2)[1])
        K3=F(t0+h/2, (X0+(K2*h)/2)[0],(X0+(K2*h)/2)[1])
        K4=F(t0+h, (X0+K3*h)[0],(X0+K3*h)[1])
        X1=X0+h*(K1+(2*K2)+(2*K3)+K4)/6
        xx.append(X1)
        t1=t0+h
        tt.append(t1)
        t0=t1
        X0=X1
    return [np.array(tt),np.array(xx)[:,0],np.array(xx)[:,1]]

f=lambda t,x,y:-x+2*y

g=lambda t,x,y:-2*x-y

F=lambda t,x,y:np.array([f(t,x,y),g(t,x,y)])

rkd=Runge_Kutta_1_double(F,0,4,0.05,1,2)

t=rkd[0]

x=rkd[1]
```

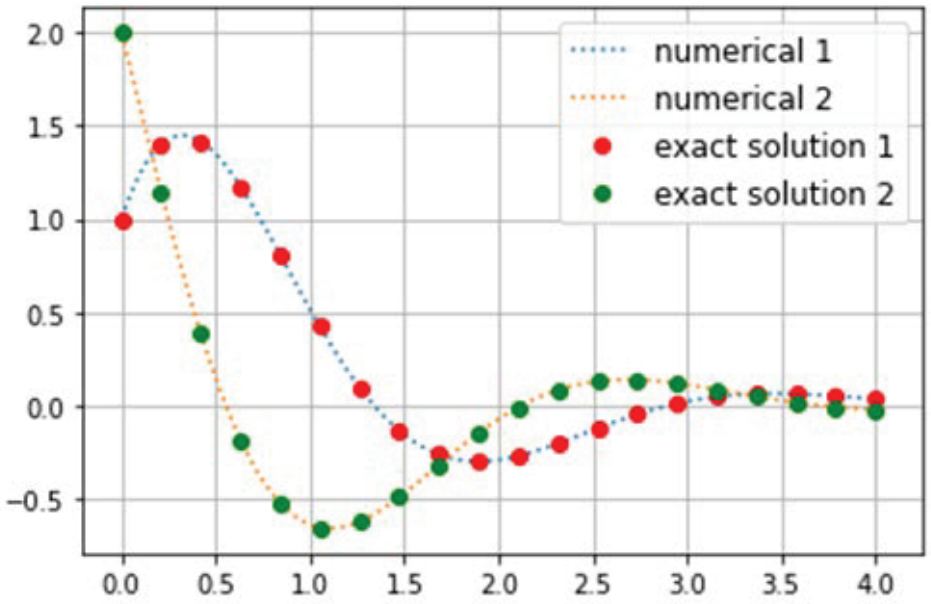

Figure 2. Runge_Kutta_1_double

```
y=rkd[2]
plt.plot(t,x,linestyle='dotted',label='numerical 1'),
plt.plot(t,y,linestyle='dotted',label='numerical 2'),
plt.legend(loc='upper right',fontsize='large'),
t1=np.linspace(0,4,20)
f1=lambda t:e**(-t)*(np.cos(2*t)+2*np.sin(2*t))
f2=lambda t:e**(-t)*(2*np.cos(2*t)-np.sin(2*t))
plt.plot(t1,f1(t1),'ro',label='exact solution 1'),
plt.plot(t1,f2(t1),'go',label='exact solution 2'),
plt.legend(loc='upper right',fontsize='large'),
plt.grid(True)
```

Figure 2

Example 3. Consider the following second order differential equation

$$\frac{d^2x(t)}{d^2t} + 3\frac{dx(t)}{dt} + 2\,x = 3t, \quad x(0) = 0, \quad x'(0) = 0, \quad 0 < t < 5\,,$$

The exact solution is

$$x(t) = -\frac{9}{4} + \frac{3}{2}t - \frac{3}{4}e^{-2t} + 3e^{-t}\,.$$

For

$$\frac{d^2x(t)}{d^2t} + a\frac{dx(t)}{dt} + b\,x = g(t), \quad x(0) = x0, \quad x'(0) = v0, \quad t_0 < t < t_1\,,$$

Runge-Kutta method with $f1(t, x, v) = v$ ($\frac{dx(t)}{dt} = v$) and $f(t, x, v) = -a\,v - b\,x + g(t)$ gives a numerical solution.

```
def Runge_Kutta_2(F,t0,t,h,x0,v0):
    # t0 time start, t time end, h increasing time value
    # x(0)=x_0, x´(0)=v_0 initial values
    X0=np.array([x0,v0])
    xx,tt=[ ],[ ]
    xx.append(X0)
    tt.append(t0)
    while t0 < t:
        K1=F(t0,X0[0],X0[1])
        K2=F(t0+h/2, (X0+(K1*h)/2)[0],(X0+(K1*h)/2)[1])
        K3=F(t0+h/2, (X0+(K2*h)/2)[0],(X0+(K2*h)/2)[1])
        K4=F(t0+h, (X0+K3*h)[0],(X0+K3*h)[1])
        X1=X0+h*(K1+(2*K2)+(2*K3)+K4)/6
        xx.append(X1)
        t1=t0+h
        tt.append(t1)
        t0=t1
        X0=X1
    return [np.array(tt),np.array(xx)[:,0]]

f1=lambda t,x,v:v

f=lambda t, x, v:-3*v-2*x+t*3

F=lambda t,x,v:np.array([f1(t,x,v),f(t,x,v)])

rk2=Runge_Kutta_2(F,0,5,0.1,0,0)
```

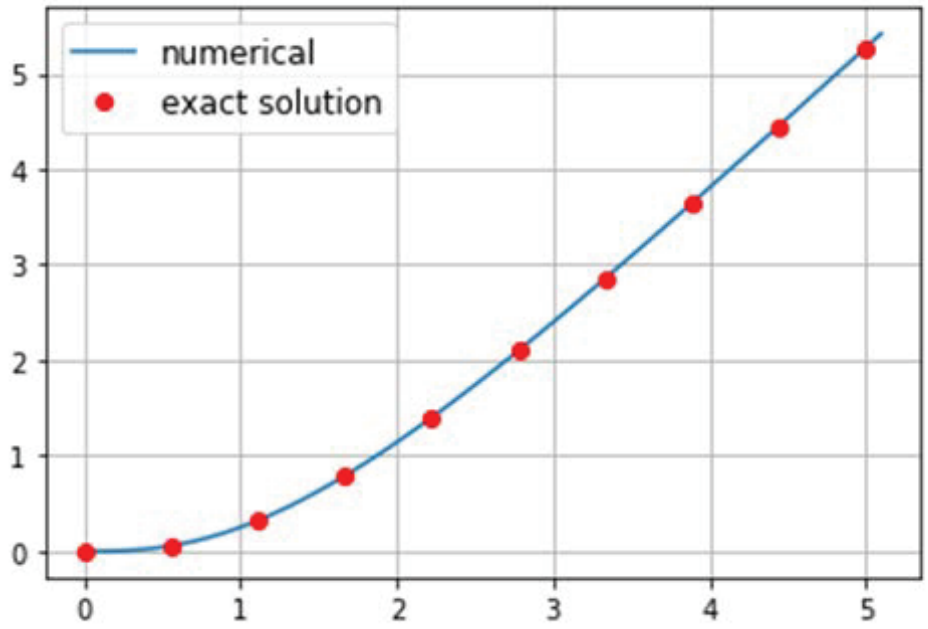

Figure 3. Runge_Kutta_2

```
t=rk2[0]
x=rk2[1]
plt.plot(t,x,label='numerical'),
plt.legend(loc='upper left',fontsize='large'),
t1=np.linspace(0,5,10)
g=lambda t:-9/4+(3/2)*t-(3/4)*e**(-2*t)+(3)*e**(-t)
plt.plot(t1,g(t1),'ro',label='exact solution'),
plt.legend(loc='upper left',fontsize='large'),
plt.grid(True)
```

Figure 3

Finally, for the second order two differential equations

$\frac{d^2u(t)}{d^2t} + a\frac{du(t)}{dt} + b\,u = g_1(t), \quad u(0) = u\_0, \quad u'(0) = u1\_0$
$\frac{d^2v(t)}{d^2t} + c\frac{dv(t)}{dt} + d\,v = g_2(t), \quad v(0) = v\_0, \quad v'(0) = v1\_0, \quad t_0 < t < t_1\,,$

Runge-Kutta method with

$$f1(t,u,u1,v,v1) = u1 \;\; (\frac{du(t)}{dt} = u1), \quad ff(t,u,u1,v,v1) = -a\,u1 - b\,u + g_1(t)$$

$$f2(t,u,u1,v,v1) = v1 \;\; (\frac{dv(t)}{dt} = v1), \quad gg(t,u,u1,v,v1) = -c\,v1 - d\,v + g_2(t)$$

gives a numerical solution.

Example 4.

$\frac{d^2u(t)}{d^2t} = -u, \quad u(0) = 0, \quad u'(0) = 1\,,$
$\frac{d^2v(t)}{d^2t} = -v, \quad v(0) = 1, \quad v'(0) = 0, \quad 0 < t < 2\pi\,.$

```
def Runge_Kutta_2_double(FF,t0,t,h,u_0,u1_0,v_0,v1_0):
    # u(0)=u_0, u´(0)=u1_0, and v(0)=v_0, v´(0)=v1_0 initial values
    X0=np.array([u_0,u1_0,v_0,v1_0])
    xx,tt=[ ],[ ]
    xx.append(X0)
    tt.append(t0)
    while t0 < t:
        K1=FF(t0,X0[0],X0[1],X0[2],X0[3])
        K2=FF(t0+h/2, (X0+(K1*h)/2)[0],(X0+(K1*h)/2)[1],\
        (X0+(K1*h)/2)[2],(X0+(K1*h)/2)[3])
        K3=FF(t0+h/2, (X0+(K2*h)/2)[0],(X0+(K2*h)/2)[1],\
        (X0+(K2*h)/2)[2],(X0+(K2*h)/2)[3])
        K4=FF(t0+h, (X0+K3*h)[0],(X0+K3*h)[1],\
        (X0+K3*h)[2],(X0+K3*h)[3])
        X1=X0+h*(K1+(2*K2)+(2*K3)+K4)/6
        xx.append(X1)
        t1=t0+h
        tt.append(t1)
        t0=t1
        X0=X1
```

```
    return [np.array(tt),np.array(xx)[:,0],np.array(xx)[:,2]]
f1=lambda t,u,u1,v,v1:u1
f=lambda t,u,u1,v,v1:-u
f2=lambda t,u,u1,v,v1:v1
g=lambda t,u,u1,v,v1:-v
FF=lambda t,u,u1,v,v1:np.array([f1(t,u,u1,v,v1),f(t,u,u1,v,v1),\
f2(t,u,u1,v,v1),g(t,u,u1,v,v1)])
rk2=Runge_Kutta_2_double(FF,0,2*pi,0.1,0,1,1,0)
t,x,y=rk2[0],rk2[1],rk2[2]
plt.axis([0,2*pi,-1,1])
plt.plot(t,x,label='numerical'),
plt.plot(t,y,label='numerical'),
plt.legend(loc='upper right',fontsize='large'),
t1=np.linspace(0,2*pi,18)
g1=lambda t:np.sin(t)
g2=lambda t:np.cos(t)
plt.plot(t1,g1(t1),'ro',label='exact solution'),
plt.plot(t1,g2(t1),'go',label='exact solution'),
plt.legend(loc='upper right',fontsize='large'),
plt.grid(True)
```

Figure 4

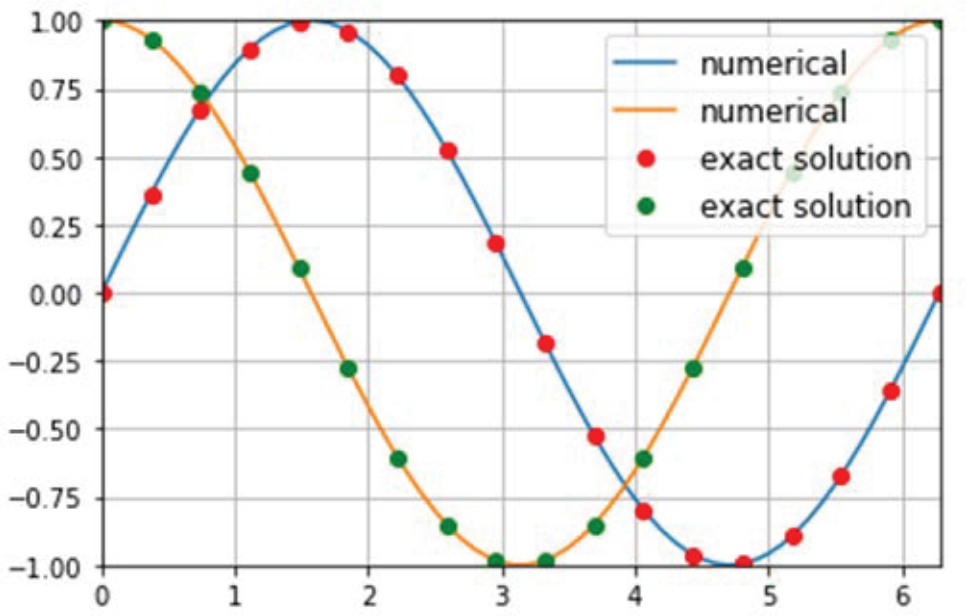

Figure 4. Runge_Kutta_2_double

4. Numerical Calculations of Geodesics with E, F, G

For the geodesic equations

$$u'' + \Gamma^1_{11}(u')^2 + 2\,\Gamma^1_{12}(u')(v') + \Gamma^1_{22}(v')^2 = 0, \quad u(0) = u\_0, \quad u'(0) = u1\_0\,,$$

$$v'' + \Gamma^2_{11}(u')^2 + 2\,\Gamma^2_{12}(u')(v') + \Gamma^2_{22}(v')^2 = 0, \quad v(0) = v\_0, \quad v'(0) = v1\_0\,,$$

Runge-Kutta method with

$$f1(t,u,u1,v,v1) = u1 \;\; (\frac{du(t)}{dt} = u1), \quad f2(t,u,u1,v,v1) = v1 \;\; (\frac{dv(t)}{dt} = v1)\,,$$

$$ff(t,u,u1,v,v1) = -\Gamma^1_{11}(f1)^2 - 2\,\Gamma^1_{12}(f1)(f2) - \Gamma^1_{22}(f2)^2\,,$$

$$gg(t,u,u1,v,v1) = -\Gamma^2_{11}(f1)^2 - 2\,\Gamma^2_{12}(f1)(f2) - \Gamma^2_{22}(f2)^2$$

gives a numerical solution.

```
import math
from math import *
```

```
import numpy as np

import matplotlib.pyplot as plt

def diff_1(f,x,y):
    h=1e-4
    d=(f(x+h,y)-f(x-h,y))/(2*h)
    return d

def diff_2(f,x,y):
    h=1e-4
    d=(f(x,y+h)-f(x,y-h))/(2*h)
    return d

def Runge_Kutta_2_double_EFG(E,F,G,t0,t,h,u_0,u1_0,v_0,v1_0):
    # t0 time start, t time end, h increasing time value
    # u(0)=u_0, u´(0)=u1_0, and v(0)=v_0, v´(0)=v1_0 initial values
    # E_u E_v partial derivative of E
    E_u=lambda u,v:diff_1(E,u,v)
    F_u=lambda u,v:diff_1(F,u,v)
    G_u=lambda u,v:diff_1(G,u,v)
    E_v=lambda u,v:diff_2(E,u,v)
    F_v=lambda u,v:diff_2(F,u,v)
    G_v=lambda u,v:diff_2(G,u,v)
    # C_ij_k Christoffel symbols
    C_11_1=lambda u,v:(G(u,v)*E_u(u,v) -2*F(u,v)*F_u(u,v)+F(u,v)*E_v(u,v))/\
    (2*(E(u,v)*G(u,v) -F(u,v)**2))
    C_12_1=lambda u,v:(G(u,v)*E_v(u,v) -F(u,v)*G_u(u,v))/\
    (2*(E(u,v)*G(u,v)-F(u,v)**2))
    C_22_1=lambda u,v:(2*G(u,v)*F_v(u,v) -G(u,v)*G_u(u,v)-F(u,v)*G_v(u,v))/\
    (2*(E(u,v)*G(u,v) -F(u,v)**2))
    C_11_2=lambda u,v:(-F(u,v)*E_u(u,v) +2*E(u,v)*F_u(u,v)-E(u,v)*E_v(u,v))/\
    (2*(E(u,v)*G(u,v) -F(u,v)**2))
    C_12_2=lambda u,v:(-F(u,v)*E_v(u,v) +E(u,v)*G_u(u,v))/\
    (2*(E(u,v)*G(u,v) -F(u,v)**2))
    C_22_2=lambda u,v:(-2*F(u,v)*F_v(u,v) +F(u,v)*G_u(u,v)+E(u,v)*G_v(u,v))/\
    (2*(E(u,v)*G(u,v) -F(u,v)**2))
```

```
f1=lambda t,u,u1,v,v1:u1
f2=lambda t,u,u1,v,v1:v1
# ff gg geodesic equations
ff=lambda t,u,u1,v,v1:-C_11_1(u,v)*(f1(t,u,u1,v,v1)**2)\
-2*C_12_1(u,v)*(f1(t,u,u1,v,v1)*f2(t,u,u1,v,v1))\
-C_22_1(u,v)*(f2(t,u,u1,v,v1)**2)
gg=lambda t,u,u1,v,v1:-C_11_2(u,v)*(f1(t,u,u1,v,v1)**2)\
-2*C_12_2(u,v)*(f1(t,u,u1,v,v1)*f2(t,u,u1,v,v1))\
-C_22_2(u,v)*(f2(t,u,u1,v,v1)**2)
# Runge Kutta method with FF
FF=lambda t,u,u1,v,v1:np.array([f1(t,u,u1,v,v1),ff(t,u,u1,v,v1),\
f2(t,u,u1,v,v1),gg(t,u,u1,v,v1)])
X0=np.array([u_0,u1_0,v_0,v1_0])
xx,tt=[ ],[ ]
xx.append(X0)
tt.append(t0)
while t0 < t:
    K1=FF(t0,X0[0],X0[1],X0[2],X0[3])
    K2=FF(t0+h/2, (X0+(K1*h)/2)[0],(X0+(K1*h)/2)[1],\
    (X0+(K1*h)/2)[2],(X0+(K1*h)/2)[3])
    K3=FF(t0+h/2, (X0+(K2*h)/2)[0],(X0+(K2*h)/2)[1],\
    (X0+(K2*h)/2)[2],(X0+(K2*h)/2)[3])
    K4=FF(t0+h, (X0+K3*h)[0],(X0+K3*h)[1],\
    (X0+K3*h)[2],(X0+K3*h)[3])
    X1=X0+h*(K1+(2*K2)+(2*K3)+K4)/6
    xx.append(X1)
    t1=t0+h
    tt.append(t1)
    t0=t1
    X0=X1
return [np.array(tt),np.array(xx)[:,0],np.array(xx)[:,2]]
```

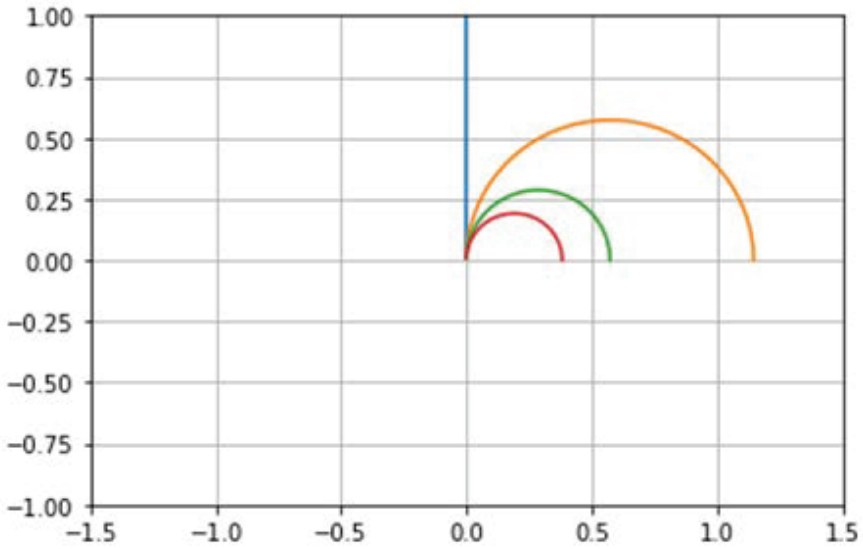

Figure 5. Geodesics on the Poincaré upper half plane

Geodesics 1. The line element ds^2 of the Poincaré upper half plane is

$$ds^2 = \frac{dx^2 + dy^2}{y^2}\,.$$

```
E=lambda u,v:1/v**2

F=lambda u,v:0

G=lambda u,v:1/v**2

plt.axis([-1.5,1.5,-1,1])

for k in range(4):
    th=np.radians(90-1*k)
    u1_0=np.cos(th)
    v1_0=np.sin(th)
    rk=Runge_Kutta_2_double_EFG(E,F,G,0,10,0.001,0,u1_0,0.01,v1_0)
    plt.plot(rk[1],rk[2])
    plt.grid(True)
```

Figure 5

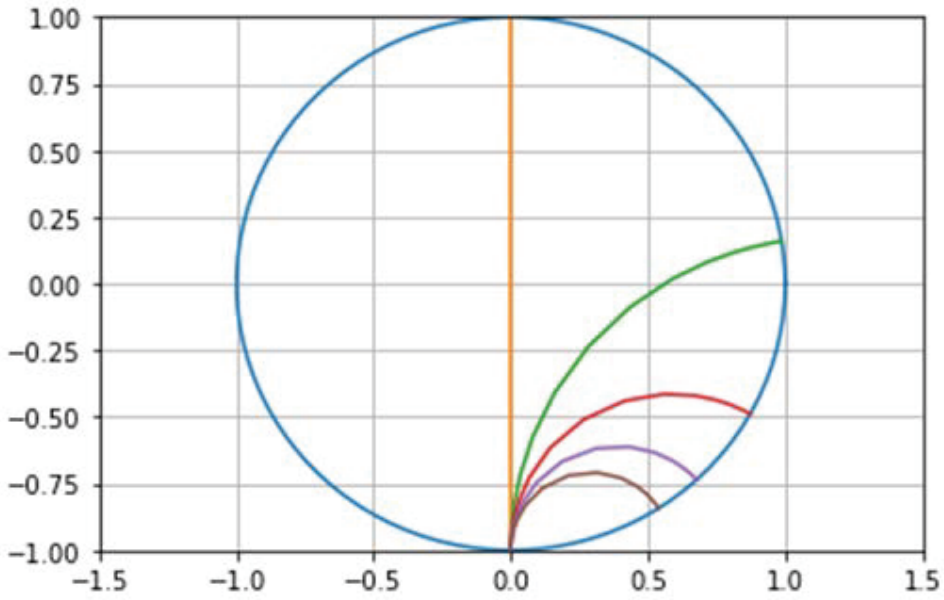

Figure 6. Geodesics on the Poincaré disk

Geodesics 2. The line element ds^2 of the Poincaré disk is

$$ds^2 = \frac{4(dx^2 + dy^2)}{(1 - (x^2 + y^2))^2}\,.$$

```
E=lambda u,v:4/(1-(u**2+v**2))**2
F=lambda u,v:0
G=lambda u,v:4/(1-(u**2+v**2))**2
t1=np.linspace(0,2*pi,100)
x1=lambda t:np.cos(t)
y1=lambda t:np.sin(t)
plt.axis([-1.5,1.5,-1,1])
plt.plot(x1(t1),y1(t1))
```

```
for k in range(5):
    th=np.radians(90-0.5*k)
    u1_0=np.cos(th)
    v1_0=np.sin(th)
    u_0=np.cos(-pi/2)
    v_0=np.sin(-pi/2)+0.01
    rk=Runge_Kutta_2_double_EFG(E,F,G,0,15,0.005,u_0,u1_0,v_0,v1_0)
    x,y=rk[1],rk[2]
    plt.plot(x,y)
    plt.grid(True)
```

Figure 6

```
E=lambda u,v:4/(1-(u**2+v**2))**2
F=lambda u,v:0
G=lambda u,v:4/(1-(u**2+v**2))**2
t1=np.linspace(0,2*pi,100)
x1=lambda t:np.cos(t)
y1=lambda t:np.sin(t)
plt.axis([-1.5,1.5,-1,1])
plt.plot(x1(t1),y1(t1))
for k in range(5):
    th=np.radians(90)
    u1_0=np.cos(th)
    v1_0=np.sin(th)
    u_0=k*0.2
    v_0=0
    rk=Runge_Kutta_2_double_EFG(E,F,G,0,15,0.005,u_0,u1_0,v_0,v1_0)
    x,y=rk[1],rk[2]
    plt.plot(x,y)
    plt.grid(True)
for k in range(5):
```

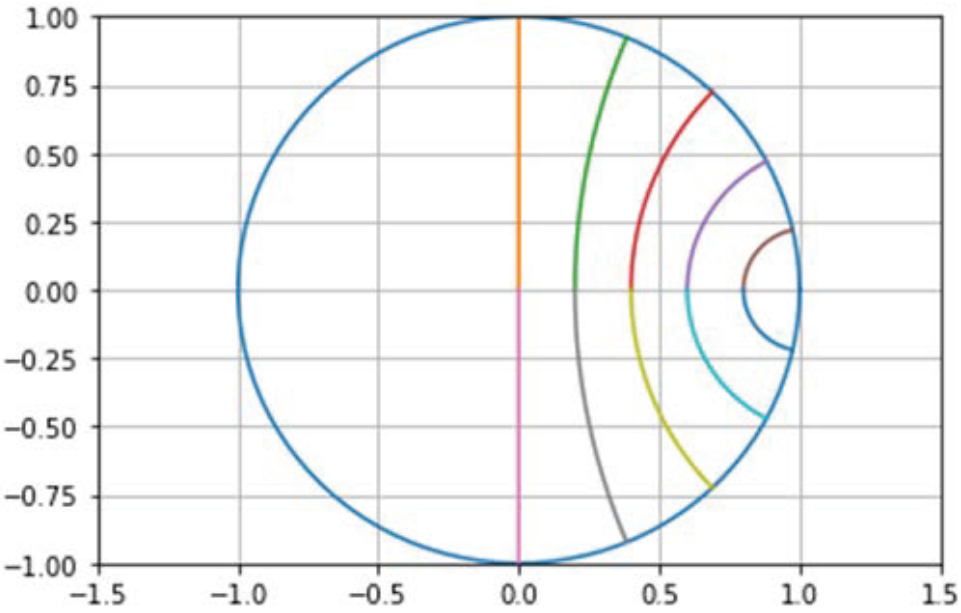

Figure 7. Geodesics on the Poincaré disk

```
th=np.radians(90)
u1_0=-np.cos(th)
v1_0=-np.sin(th)
u_0=k*0.2
v_0=0
rk=Runge_Kutta_2_double_EFG(E,F,G,0,15,0.005,u_0,u1_0,v_0,v1_0)
x,y=rk[1],rk[2]
plt.plot(x,y)
plt.grid(True)
```

Figure 7

5. Numerical Calculations of Geodesics with a Coordinate Chart $x(u,v)$, $y(u,v)$, $z(u,v)$

We show geodesics on a sphere, torus and a surface $z = e^{\frac{-x^2}{2}-y^2}$. A coordinate chart is given by

$X(u,v) = (r\sin u\cos v,\ r\sin u\sin v,\ r\cos u), \quad 0 < r\,,\ 0 \le v \le 2\pi,\ 0 \le u \le \pi\,,$

$X(u,v) = ((R + r\cos u)\cos v,\ (R + r\cos u)\sin v,\ r\sin u),$

$0 \le v \le 2\pi,\ 0 \le u \le 2\pi,\ r < R,$

$X(u,v) = (u,\ v,\ e^{\frac{-u^2}{2}-v^2})$, respectively.

```
import math
from math import *
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


def diff_1(f,x,y):
    h=1e-4
    d=(f(x+h,y)-f(x-h,y))/(2*h)
    return d
def diff_2(f,x,y):
    h=1e-4
    d=(f(x,y+h)-f(x,y-h))/(2*h)
    return d
def dot_product(a,b):
    n=len(a)
    sum=0
    for k in range(n):
        sum=sum+a[k]*b[k]
    return sum
def Runge_Kutta_2_double_coordinate_chart(x,y,z,t0,t,h,u_0,u1_0,v_0,v1_0):
```

```
# x,y,z a coordinate chart x(u,v), y(u,v), z(u,v) of a surface
# t0 time start, t time end, h increasing time value
# u(0)=u_0, u´(0)=u1_0, and v(0)=v_0, v´(0)=v1_0 initial values
x_u=lambda u,v:diff_1(x,u,v)
y_u=lambda u,v:diff_1(y,u,v)
z_u=lambda u,v:diff_1(z,u,v)
x_v=lambda u,v:diff_2(x,u,v)
y_v=lambda u,v:diff_2(y,u,v)
z_v=lambda u,v:diff_2(z,u,v)
X_u=lambda u,v:[x_u(u,v),y_u(u,v),z_u(u,v)]
X_v=lambda u,v:[x_v(u,v),y_v(u,v),z_v(u,v)]
E=lambda u,v:dot_product(X_u(u,v),X_u(u,v))
F=lambda u,v:dot_product(X_u(u,v),X_v(u,v))
G=lambda u,v:dot_product(X_v(u,v),X_v(u,v))
# E_u E_v partial derivative of E
E_u=lambda  u,v:diff_1(E,u,v)
F_u=lambda  u,v:diff_1(F,u,v)
G_u=lambda  u,v:diff_1(G,u,v)
E_v=lambda  u,v:diff_2(E,u,v)
F_v=lambda  u,v:diff_2(F,u,v)
G_v=lambda  u,v:diff_2(G,u,v)
# C_ij_k Christoffel symbols
C_11_1=lambda u,v:(G(u,v)*E_u(u,v) -2*F(u,v)*F_u(u,v)+F(u,v)*E_v(u,v))/\
(2*(E(u,v)*G(u,v) -F(u,v)**2))
C_12_1=lambda u,v:(G(u,v)*E_v(u,v) -F(u,v)*G_u(u,v))/\
(2*(E(u,v)*G(u,v)-F(u,v)**2))
C_22_1=lambda u,v:(2*G(u,v)*F_v(u,v) -G(u,v)*G_u(u,v)-F(u,v)*G_v(u,v))/\
(2*(E(u,v)*G(u,v) -F(u,v)**2))
C_11_2=lambda u,v:(-F(u,v)*E_u(u,v) +2*E(u,v)*F_u(u,v)-E(u,v)*E_v(u,v))/\
(2*(E(u,v)*G(u,v) -F(u,v)**2))
C_12_2=lambda u,v:(-F(u,v)*E_v(u,v) +E(u,v)*G_u(u,v))/\
(2*(E(u,v)*G(u,v) -F(u,v)**2))
C_22_2=lambda u,v:(-2*F(u,v)*F_v(u,v) +F(u,v)*G_u(u,v)+E(u,v)*G_v(u,v))/\
(2*(E(u,v)*G(u,v) -F(u,v)**2))
f1=lambda t,u,u1,v,v1:u1
```

```
f2=lambda t,u,u1,v,v1:v1
# ff gg geodesic equations
ff=lambda t,u,u1,v,v1:-C_11_1(u,v)*(f1(t,u,u1,v,v1)**2)\
-2*C_12_1(u,v)*(f1(t,u,u1,v,v1)*f2(t,u,u1,v,v1))\
-C_22_1(u,v)*(f2(t,u,u1,v,v1)**2)
gg=lambda t,u,u1,v,v1:-C_11_2(u,v)*(f1(t,u,u1,v,v1)**2)\
-2*C_12_2(u,v)*(f1(t,u,u1,v,v1)*f2(t,u,u1,v,v1))\
-C_22_2(u,v)*(f2(t,u,u1,v,v1)**2)
# Runge Kutta method with FF
FF=lambda t,u,u1,v,v1:np.array([f1(t,u,u1,v,v1),ff(t,u,u1,v,v1),\
f2(t,u,u1,v,v1),gg(t,u,u1,v,v1)])
X0=np.array([u_0,u1_0,v_0,v1_0])
xx,tt=[ ],[ ]
xx.append(X0)
tt.append(t0)
while t0 < t:
    K1=FF(t0,X0[0],X0[1],X0[2],X0[3])
    K2=FF(t0+h/2, (X0+(K1*h)/2)[0],(X0+(K1*h)/2)[1],\
    (X0+(K1*h)/2)[2],(X0+(K1*h)/2)[3])
    K3=FF(t0+h/2, (X0+(K2*h)/2)[0],(X0+(K2*h)/2)[1],\
    (X0+(K2*h)/2)[2],(X0+(K2*h)/2)[3])
    K4=FF(t0+h, (X0+K3*h)[0],(X0+K3*h)[1],\
    (X0+K3*h)[2],(X0+K3*h)[3])
    X1=X0+h*(K1+(2*K2)+(2*K3)+K4)/6
    xx.append(X1)
    t1=t0+h
    tt.append(t1)
    t0=t1
    X0=X1
return [np.array(tt),np.array(xx)[:,0],np.array(xx)[:,2]]
```

Geodesics 3. Geodesics on a sphere with a coordinate chart

$$X(u,v) = (0.7\sin u\cos v,\ 0.7\sin u\sin v,\ 0.7\cos u)\,.$$

```
x=lambda u,v:0.7*np.sin(u)*np.cos(v)
y=lambda u,v:0.7*np.sin(u)*np.sin(v)
z=lambda u,v:0.7*np.cos(u)
t=[ ]
for k in range(2):
    th=np.radians(90-60*k)
    u1_0=np.cos(th)
    v1_0=np.sin(th)
    rk=Runge_Kutta_2_double_coordinate_chart(x,y,z,0,8,0.1,1,u1_0,1,v1_0)
    t.append([rk[1],rk[2]])
t=np.array(t)
x1=0.7*np.sin(t[0,0])*np.cos(t[0,1])
y1=0.7*np.sin(t[0,0])*np.sin(t[0,1])
z1=0.7*np.cos(t[0,0])
x2=0.7*np.sin(t[1,0])*np.cos(t[1,1])
y2=0.7*np.sin(t[1,0])*np.sin(t[1,1])
z2=0.7*np.cos(t[1,0])
u = np.linspace(0+0.01, pi-0.01, 20)
v = np.linspace(0, 2*pi, 20)
u,v = np.meshgrid(u,v)
ff=lambda u,v:0.7*np.sin(u)*np.cos(v)
g=lambda u,v:0.7*np.sin(u)*np.sin(v)
h=lambda u,v:0.7*np.cos(u)
fig = plt.figure(figsize=(9, 6))
ax = fig.add_subplot(111, projection='3d')
ax.view_init(65, 35)
ax.plot_wireframe(ff(u,v),g(u,v),h(u,v), color='blue')
ax.plot(x1, y1, z1, 'ro')
```

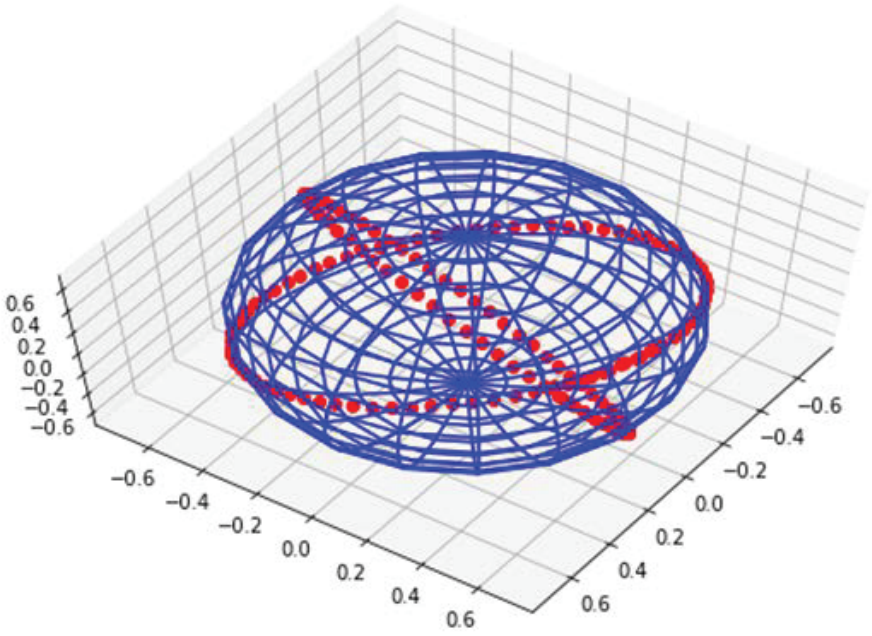

Figure 8. Geodesics on a sphere

```
ax.plot(x2, y2, z2, 'ro')
plt.show()
```

Figure 8

Geodesics 4. Geodesics on a torus with a coordinate chart

$$X(u,v) = ((6+2\cos u)\cos v,\ (6+2\cos u)\sin v,\ 2\sin u)\,.$$

```
x=lambda u,v :(6+2*np.cos(u))*np.cos(v)
y=lambda u,v :(6+2*np.cos(u))*np.sin(v)
z=lambda u,v :2*np.sin(u)
t=[ ]
for k in range(3):
    th=np.radians(45*k)
```

```
    u1_0=np.cos(th)
    v1_0=np.sin(th)
    rk=Runge_Kutta_2_double_coordinate_chart(x,y,z,0,10,0.1,0,u1_0,0,v1_0)
    t.append([rk[1],rk[2]])
t=np.array(t)
x1=(6+2*np.cos(t[0,0]))*np.cos(t[0,1])
y1=(6+2*np.cos(t[0,0]))*np.sin(t[0,1])
z1=2*np.sin(t[0,0])
x2=(6+2*np.cos(t[1,0]))*np.cos(t[1,1])
y2=(6+2*np.cos(t[1,0]))*np.sin(t[1,1])
z2=2*np.sin(t[1,0])
x3=(6+2*np.cos(t[2,0]))*np.cos(t[2,1])
y3=(6+2*np.cos(t[2,0]))*np.sin(t[2,1])
z3=2*np.sin(t[2,0])
u = np.linspace(0, 2*pi,20)
v = np.linspace(0, 2*pi,20)
u,v = np.meshgrid(u,v)
ff=lambda u,v:(6+2*np.cos(u))*np.cos(v)
g=lambda u,v:(6+2*np.cos(u))*np.sin(v)
h=lambda u,v:2*np.sin(u)
fig = plt.figure(figsize=(9, 6))
ax = fig.add_subplot(111, projection='3d')
ax.view_init(65, 35)
ax.plot_wireframe(ff(u,v),g(u,v),h(u,v), color='blue')
ax.plot(x1, y1, z1, 'ro')
ax.plot(x2, y2, z2, 'go')
ax.plot(x3, y3, z3, 'ro')
plt.show()
```

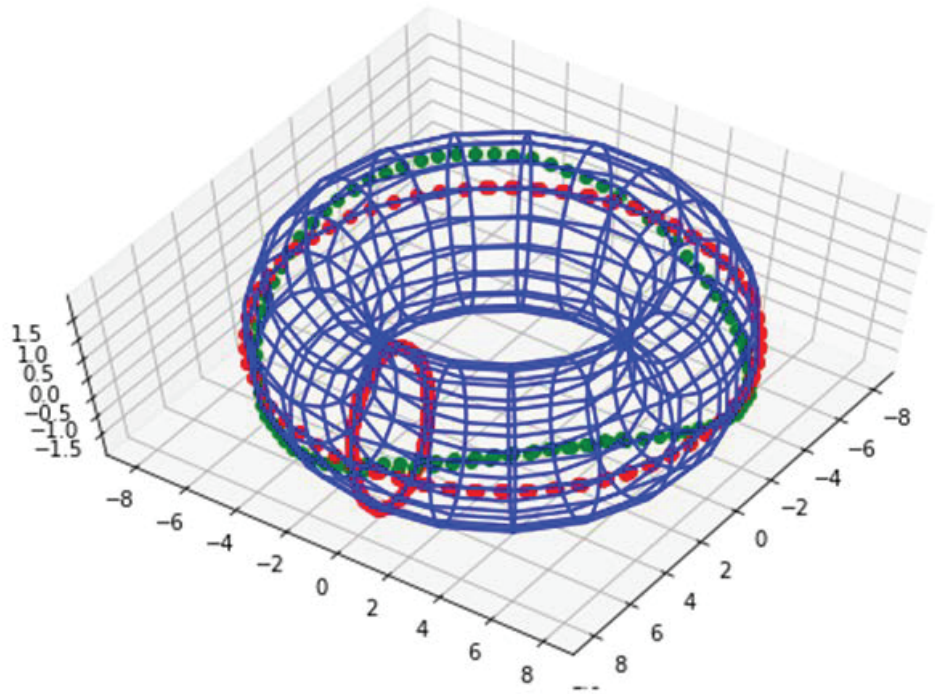

Figure 9. Geodesics on a torus

Figure 9

Geodesics 5. Geodesics on a surface with a coordinate chart

$$X(u,v) = (u,\ v,\ e^{\frac{-u^2}{2}-v^2}).$$

```
x=lambda u,v :u
y=lambda u,v :v
z=lambda u,v :np.e**((-1/2)*u**2-v**2)
t=[ ]
u1_0=1
v1_0=0
```

```
rk=Runge_Kutta_2_double_coordinate_chart(x,y,z,0,9,0.1,-4.5,u1_0,0.6,v1_0)
t.append([rk[1],rk[2]])
rk=Runge_Kutta_2_double_coordinate_chart(x,y,z,0,9,0.1,-4.5,u1_0,0,v1_0)
t.append([rk[1],rk[2]])
rk=Runge_Kutta_2_double_coordinate_chart(x,y,z,0,9,0.1,-4.5,u1_0,-0.6,v1_0)
t.append([rk[1],rk[2]])
t=np.array(t)
x1,y1,z1=t[0,0],t[0,1],np.e**((-1/2)*t[0,0]**2-t[0,1]**2)
x2,y2,z2=t[1,0],t[1,1],np.e**((-1/2)*t[1,0]**2-t[1,1]**2)
x3,y3,z3=t[2,0],t[2,1],np.e**((-1/2)*t[2,0]**2-t[2,1]**2)
u = np.linspace(-5, 5,30)
v = np.linspace(-5, 5,30)
u,v = np.meshgrid(u,v)
ff=lambda u,v:u
g=lambda u,v:v
h=lambda u,v:np.e**((-1/2)*u**2-v**2)
fig = plt.figure(figsize=(9, 6))
ax = fig.add_subplot(111, projection='3d')
ax.view_init(82, 30)
ax.plot_wireframe(ff(u,v),g(u,v),h(u,v), color='blue')
ax.plot(x1, y1, z1, 'ro')
ax.plot(x2, y2, z2, 'go')
ax.plot(x3, y3, z3, 'ro')
plt.show()
```

Figure 10

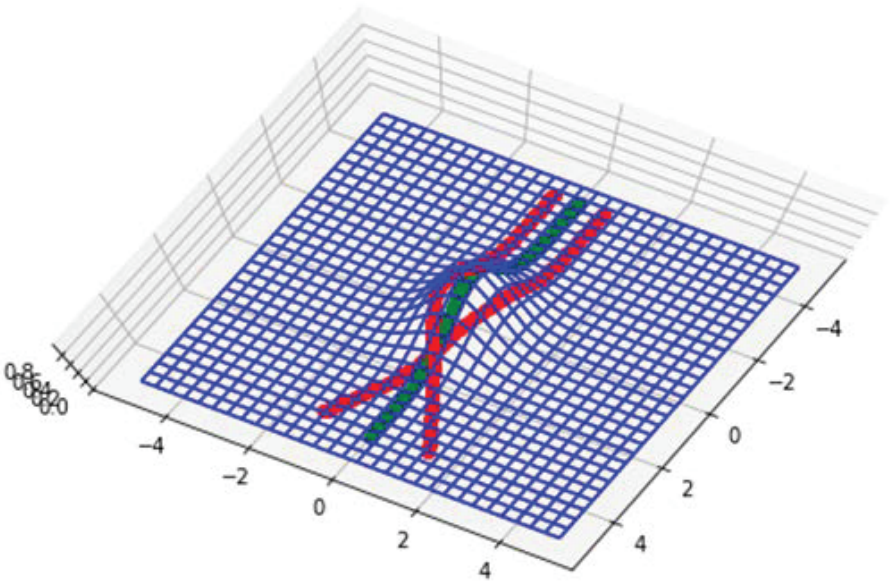

Figure 10. Geodesics on a surface $z = e^{\frac{-x^2}{2}-y^2}$

REFERENCES

1. Andrijana Burazin: Calculating Geodesics on Surfaces. McMaster University, Thesis for Master of Science, (2008), 1-72
2. J.R. Kim: The line element approach for the geometry of Poincaré disk. Honam Mathematical J. **43** (2021), no. 3, 385-402
3. J.R. Kim: Python data analysis matrix mathematics. Kyungmoon Press, 2020
4. John McCleary: Geometry from a Differentiable Viewpoint. Cambridge university pres, 1994
5. Barrett O' Neill: Elementary Differential Geometry 2nd edition. Academic press, 1997
6. James R. Smart: Modern geometries 5th edition. Brooks/Cole Publishing Company, 1998

PROFESSOR: DEPARTMENT OF MATHEMATICS, KUNSAN NATIONAL UNIVERSITY, KUNSAN 573-701, REPUBLIC OF KOREA
Email address: kimjr0@kunsan.ac.kr