Práctica 5. Extra: Otra familia paramétrica

Guillermo Martín Sánchez

12/04/20

1 Introducción

En esta práctica vamos a hacer otra animación de las imágenes de una familia paramétrica al aplicarse sobre la 2-esfera unidad sin el polo sur $S_1^2 \setminus e_3 (:= (0,0,-1))$ y sobre una curva en su superficie γ (ver la memoria de la Práctica 5 para más detalles). La familia paramétrica de funciones $g_t : S_1^2 \setminus e_3 \to \mathbb{R}^2$ tiene que cumplir que $\lim_{t \to t_0} g_t = \Pi$ y $g_0 = Id$ (donde Π es la proyección estereográfica con $\alpha = 1$) y usar las propiedades de dominio e imagen de tan y tan^{-1}.

2 Material usado

2.1 Ejercicio 1

Utiliza las propiedades del dominio e imagen de las funciones tan y tan^{-1} para diseñar otra familia paramétrica $g_t : S_1^2 \setminus e_3 \to \mathbb{R}^2$ tal que existe un $t_0 \in \mathbb{R}$ que $\lim_{t \to t_0} g_t = \Pi$ y $g_0(p) = p$. Obtén una animación de al menos 15 fotogramas de dicha familia paramétrica.

La familia paramétrica que hemos decidido usar son las funciones $g_t : S_1^2 \setminus e_3 \to \mathbb{R}^2$ definidas por

$$g_t \begin{pmatrix} x \\ y \\ z \end{pmatrix} = tan(arctan(1-t) + \frac{\pi(1-z)}{4}t) \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ (-1)t + z(1-t) \end{pmatrix}$$

con $t \in [0,1]$. Para 20 valores de t entre 0 y 1, representamos $g_t(S_1^2 \setminus e_3)$ y $g_t(\gamma)$ haciendo así una animación. Cuando $t = 0$ tenemos

$$g_0 \begin{pmatrix} x \\ y \\ z \end{pmatrix} = tan(arctan(1-0) + \frac{\pi(1-z)}{4}0) \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ (-1)0 + z(1-0) \end{pmatrix} = 1 \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ z \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Y cuando $t = 1$ tenemos

$$g_1 \begin{pmatrix} x \\ y \\ z \end{pmatrix} = tan(arctan(1-1)+\frac{\pi(1-z)}{4}1) \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ (-1)1 + z(1-1) \end{pmatrix} = tan(\frac{\pi(1-z)}{4}) \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$$

donde podemos ver que $\lim_{z\to -1} g_1 = (\infty, \infty, -1)$ ya que $\lim_{z\to -1} tan(\frac{\pi(1-z)}{4}) = \lim_{a\to 2} tan(\frac{\pi a}{4}) = \lim_{b\to \frac{\pi}{2}} tan(b) = \infty$

3 Resultados y discusión

3.1 Ejercicio 1

En la Figura anexa *exercise3.gif* vemos el resultado de animar las imágenes de la familia paramétrica. Como era de esperar en $t = 0$ obtenemos la identidad y, conforme $t \to 1$ nos vamos acercando poco a poco a la imagen resultado de aplicar $\Pi(S_1^2 \setminus e_3)$ y $\Pi(\gamma)$. En general, si la comparamos con la animación obtenida en la parte obligatoria vemos que son casi iguales.

4 Conclusión

Hemos visto otra posible familia paramétrica que cumple las propiedades de $\lim_{t\to t_0} g_t = \Pi$ y $g_0 = Id$ pero en este caso usando las características del dominio e imagen de las funciones tan y tan^{-1}.

5 Código

```
"""
Coursework 5: Diffeomorphisms
"""

import os
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import animation
from mpl_toolkits.mplot3d import Axes3D

vuestra_ruta = "C:/Users/guill/Documents/Carrera/GEOComp/5-Diffeomorphism"

os.getcwd()
os.chdir(vuestra_ruta)

#%%
"""
```

```python
Exercise 1: 2-sphere projected
"""

# Stereographic projection to plane z0 (=-1)
def proj(x,z,z0=-1,alpha=1):
    z0 = z*0+z0
    eps = 1e-16
    x_trans = x/(abs(z0-z)**alpha+eps)
    return(x_trans)


u = np.linspace(0, np.pi, 25)
v = np.linspace(0, 2 * np.pi, 50)

# Parametric equations of 2-sphere
x = np.outer(np.sin(u), np.sin(v))
y = np.outer(np.sin(u), np.cos(v))
z = np.outer(np.cos(u), np.ones_like(v))

# Parametric equations of curve
t2 = np.linspace(0, 2*np.pi, 200)
x2 = 0.32*(2.1*np.cos(10*t2)-np.cos(21*t2))
y2 = 0.32*(2.1*np.sin(10*t2)-np.sin(21*t2))
z2 = -np.sqrt(1-(x2**2+y2**2))


fig = plt.figure(figsize=(12,12))
fig.subplots_adjust(hspace=0.4, wspace=0.2)

# Plot the 2-sphere and the curve
ax = fig.add_subplot(2, 2, 1, projection='3d')
ax.plot_surface(x, y, z, rstride=1, cstride=1,
                cmap='viridis', edgecolor='none')
ax.plot(x2, y2, z2, '-b',c="white",zorder=3)
ax.set_title('2-sphere');

# Plot the projection of the 2-sphere and the curve
ax = fig.add_subplot(2, 2, 2, projection='3d')
ax.set_xlim3d(-8,8)
ax.set_ylim3d(-8,8)
ax.plot_surface(proj(x,z), proj(y,z), z*0-1, rstride=1, cstride=1,
                cmap='viridis', edgecolor='none')
ax.plot(proj(x2,z2), proj(y2,z2), -1, '-b',c="white",zorder=3)
ax.set_title('Stereographic projection');

plt.show()
```

```python
plt.close(fig)

#%%
"""
Exercise 2: Parametric transformation
"""

# Plot a frame of the parametric transformation at time t
def animate(t):
    frac = 1/((1-t) + np.abs(-1-z)*t)
    xt = frac*x
    yt = frac*y
    zt = -t + z*(1-t)

    frac2 = 1/((1-t) + np.abs(-1-z2)*t)
    x2t = frac2*x2
    y2t = frac2*y2
    z2t = -t + z2*(1-t)

    ax = plt.axes(projection='3d')
    ax.set_xlim3d(-2,2)
    ax.set_ylim3d(-2,2)
    ax.set_zlim3d(-2,2)
    ax.plot_surface(xt, yt, zt, rstride=1, cstride=1,
                    cmap='viridis', edgecolor='none')
    ax.plot(x2t,y2t, z2t, '-b',c="white",zorder=3)
    return ax,

# Plot the first frame
def init():
    return animate(0),

# Save a .gif with the animation of the parametric transformation
fig = plt.figure(figsize=(6,6))
ani = animation.FuncAnimation(fig, animate, np.arange(0,1,0.05), init_func=init,
                              interval=20)
ani.save("exercise2.gif", fps = 5)

#%%
"""
Exercise 3: Another parametric transformation
"""

# Plot a frame of the parametric transformation at time t
def animate(t):
    frac = np.tan(np.arctan(1-t) + t*(np.pi*(-z+1)/4))
```

```
    xt = frac*x
    yt = frac*y
    zt = -t + z*(1-t)

    frac2 = np.tan(np.arctan(1-t) + t*(np.pi*(-z2+1)/4))
    x2t = frac2*x2
    y2t = frac2*y2
    z2t = -t + z2*(1-t)

    ax = plt.axes(projection='3d')
    ax.set_xlim3d(-2,2)
    ax.set_ylim3d(-2,2)
    ax.set_zlim3d(-2,2)
    ax.plot_surface(xt, yt, zt, rstride=1, cstride=1,
                    cmap='viridis', edgecolor='none')
    ax.plot(x2t,y2t, z2t, '-b',c="white",zorder=3)
    return ax,

# Plot the first frame
def init():
    return animate(0),

# Save a .gif with the animation of the parametric transformation
fig = plt.figure(figsize=(6,6))
ani = animation.FuncAnimation(fig, animate, np.arange(0,1,0.05), init_func=init,
                              interval=20)
ani.save("exercise3.gif", fps = 5)
```