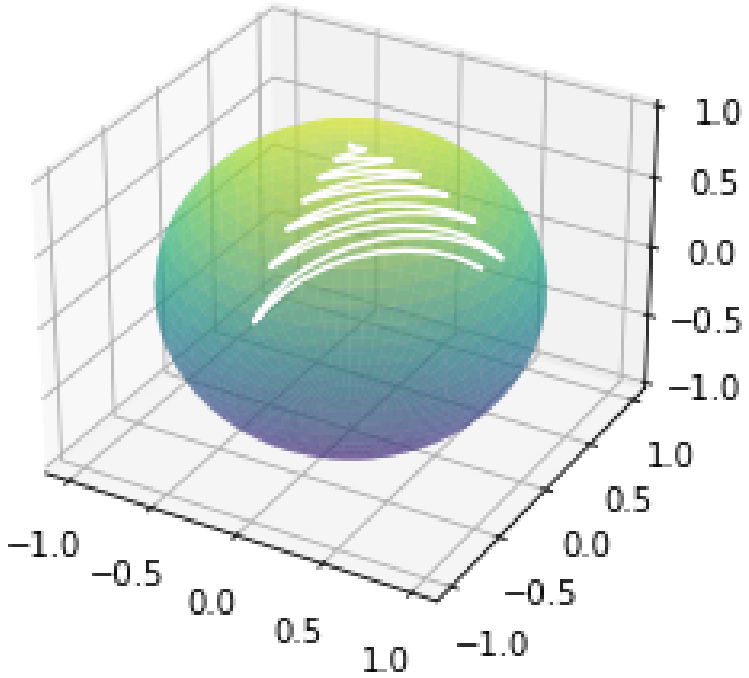
PRÁCTICA 5: Deformación de variedades diferenciables

Geometría Computacional

Belén Sánchez Centeno
belsan05@ucm.es

4 de mayo de 2022



Date Performed: 23 y 30 de marzo de 2022
Code Partner: Martín Fernández de Diego

1. Introducción

Considerar S_1^2 como la variedad diferencial dada por una 2-esfera de radio unitario, embebido en $\mathbb{R}^3$. En esta práctica se pretende representar gráficamente en 3D el difeomorfismo de una proyección estereográfica. Para ello, se extraerá el punto $e_3 := (0, 0, 1) \in S_1^2$, y se utilizarán las coordenadas cartesianas $(x, y, z) \in [-1, 1]^3$ para representar en $\mathbb{R}^3$.

Primer Objetivo

Estimar y representar una malla regular de puntos de S_1^2 con una resolución (paso de malla) que proporcione 30 valores de latitud ($\phi \in [0, \pi)$) y 60 valores de longitud ($\varphi \in [0, 2\pi)$). Estimar y representar la imagen de la proyección estereográfica $\Pi : S_1^2 \backslash e_3 \to \mathbb{R}^2$, con $\alpha = 1/2$ (ver teoría). Diseñar una curva* sobre S_1^2 para comprobar cómo se deforma cuando se proyecta en $\mathbb{R}^2$. *Nota: utilizar una curva claramente diferente a la del ejemplo de la plantilla.

Segundo Objetivo

Obtener una animación de al menos 20 fotogramas (imágenes) de la siguiente familia paramétrica:

$$f_t : S_1^2 \backslash e_3 \to \mathbb{R}^3 \tag{1}$$

$$p = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \frac{2}{2 - 2t + (1 - z)t} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -t + z(1 - t) \end{pmatrix} \tag{2}$$

donde $t = [0, 1]$ y $lim_{t \to 1} f_t = \Pi$, con $f_0(p) = p$ función identidad.

2. Material utilizado

Se resuelve el problema con un script de Python, en el que se usan funciones de implementación propia, de las plantillas proporcionadas y de las siguientes bibliotecas: `matplotlib.pyplot`, `matplotlib.animation` y `numpy`. Quedan detalladas a continuación:

- `proj`, que, dadas las coordenadas de x y z, devuelve la proyección de x sobre el plano $z_0 = 1$.
- `animate` e `init`, que encapsulan la transformación y la representación gráfica pedidas en el segundo apartado para un tiempo t y para un tiempo 0, respectivamente.

En primer lugar, se representa S_1^2 en $\mathbb{R}^3$. Para ello se toman 30 valores equidistantes (con `linspace` de `numpy`) entre 0.05 y π y otros 60 entre 0 y 2π, y se transforman las coordenadas de polares a paramétricas. Además se define, también en coordenadas paramétricas, la siguiente curva sobre S_1^2:

$$x(t) = |t| sen(40 * t/2)^2$$
$$y(t) = -|t| cos(40 * t/2)^2$$
$$z(t) = \sqrt{1 - x^2 - y^2}$$

con $t = [0{,}055, 1]$. Se muestra una gráfica de ambas.

A continuación, se proyectan tanto la esfera como la curva sobre el plano $z = 1$, mediante la función `proj`. El resultado también se representa gráficamente, quedando completada la *Figura 1.*

En segundo lugar, se proyectan la esfera y la curva mediante la siguiente transformación continua:

$$x(t) = \frac{2x}{2(1 - t) + (1 - z)t + eps}$$

$$y(t) = \frac{2y}{2(1 - t) + (1 - z)t + eps}$$

$$z(t) = (-1)t + z(1 - t)$$

siendo $t \in [0{,}055, 1]$ y con $eps = 10^{-16}$, para evitar la división por 0. Esto se representa gráficamente con la función `animate` para $t = [0, 1]$, en 20 pasos de 0.05 segundos, creando una animación con `FuncAnimation` de `matplotlib.animation` que se guarda como un archivo de tipo `.gif`.

3. Resultados

3.1. Primer objetivo

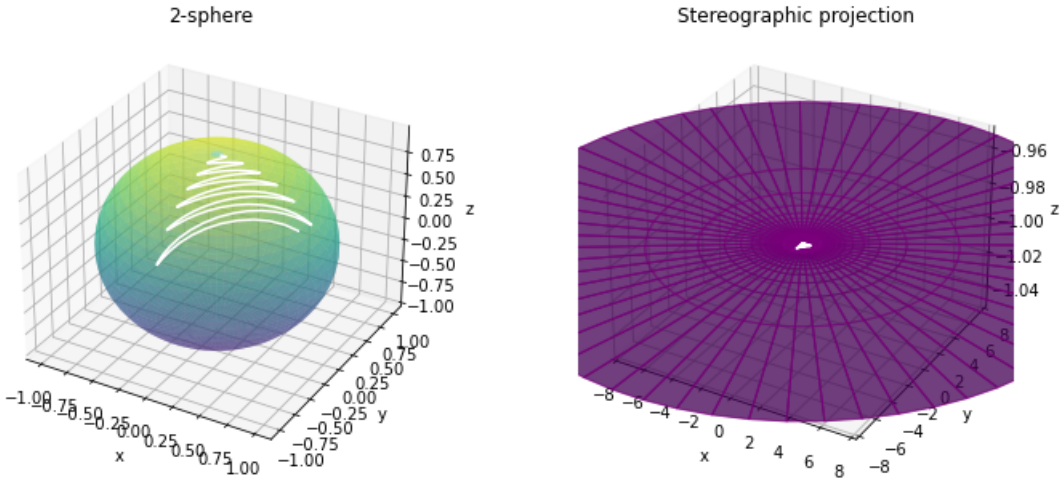


Figura 1: Proyección estereográfica

3.2. Segundo objetivo

Animación "ejemplo.gif" adjuntada aparte.

4. Conclusión

Ha sido muy ilustrativo realizar este ejercicio para visualizar las deformaciones. Se puede observar como la curva que hemos definido se mantiene dentro de la proyección estereográfica de la esfera pero se va alejando del polo norte de forma cada vez más espaciada y, en cambio, se superpone en la segunda proyección cuando t es cercano a 1.

5. Anexo: Código utilizado

```
"""
PRÁCTICA 5: DEFORMACIÓN DE VARIEDADES DIFERENCIABLES
Belén Sánchez Centeno
Martín Fernández de Diego
"""

#import os
import numpy as np
import matplotlib.pyplot as plt
#from mpl_toolkits.mplot3d import axes3d

from matplotlib import animation
#from mpl_toolkits.mplot3d.axes3d import Axes3D

"""
Dadas las coordenadas de x y z
devuelve la proyección de x sobre el eje z
z0 = 1 porque el polo extraído es el (0,0,1)
"""
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)
    # Nótese que añadimos un épsilon para evitar dividir entre 0

"""
Animación del APARTADO ii)
"""
def animate(t):
    z0 = -1
    xt = 2/(2*(1-t) + (1-z)*t + eps)*x
    yt = 2/(2*(1-t) + (1-z)*t + eps)*y
    zt = (-1)*t + z*(1-t)
    x2t = 2/(2*(1-t) + (1-z2)*t + eps)*x2
    y2t = 2/(2*(1-t) + (1-z2)*t + eps)*y2
    z2t = (-1)*t + z2*(1-t)

    ax = plt.axes(projection='3d')
    ax.set_zlim3d(-1,1)
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_zlabel('z')
    ax.plot_surface(xt, yt, zt, rstride=1, cstride=1, alpha=0.5, cmap='
        viridis', edgecolor='none')
    ax.plot(x2t, y2t, z2t, '-b', c="white", zorder=3)
    return ax

def init():
    return animate(0)



# FORMATO
class Formato:
    BOLD = "\033[1m"
```

```
    RESET = "\033[0m"



# APARTADO i)
print("\n" + Formato.BOLD + "Apartado i)" + Formato.RESET)

"""
2-esfera
definición en polares
"""
u = np.linspace(0.05, np.pi, 30)
v = np.linspace(0, 2 * np.pi, 60)

"""
2-esfera
transformación en paramétricas
"""
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))

"""
gamma-curva
definición en paramétricas
"""
t2 = np.linspace(0.055, 1, 200)

x2 = abs(t2) * np.sin(40 * t2/2)**2
y2 = abs(t2) * -np.cos(40 * t2/2)**2
z2 = np.sqrt(1-x2**2-y2**2)

# Representación: 2-esfera + 2-esfera proyectada
z0 = -1

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

ax = fig.add_subplot(2, 2, 1, projection='3d')
ax.plot_surface(x, y, z, rstride=1, cstride=1, cmap='viridis', alpha
    =0.5, edgecolor='none')
ax.plot(x2, y2, z2, '-b', c="white", zorder=3)

ax.set_title('2-sphere');
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')

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, z0), proj(y, z, z0), z*0+z0, rstride=1,
    cstride=1, cmap='viridis', alpha=0.5, edgecolor='purple')
ax.plot(proj(x2, z2, z0), proj(y2, z2, z0), z0, '-b', c="white", zorder
    =3)

ax.set_title('Stereographic projection');
```

```
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')

plt.show()
fig.savefig('stereo2.png', dpi=250)
plt.close(fig)



# APARTADO ii)
print("\n" + Formato.BOLD + "Apartado ii)" + Formato.RESET)

"""
2-esfera proyectada - familia paramétrica
"""
t = 0.1
eps = 1e-16

# Representación: animación
fig = plt.figure(figsize=(6, 6))
ani = animation.FuncAnimation(fig, animate, np.arange(0, 1, 0.05),
    init_func=init, interval=20)
ani.save("ejemplo.gif", fps = 5)
plt.close(fig)
```