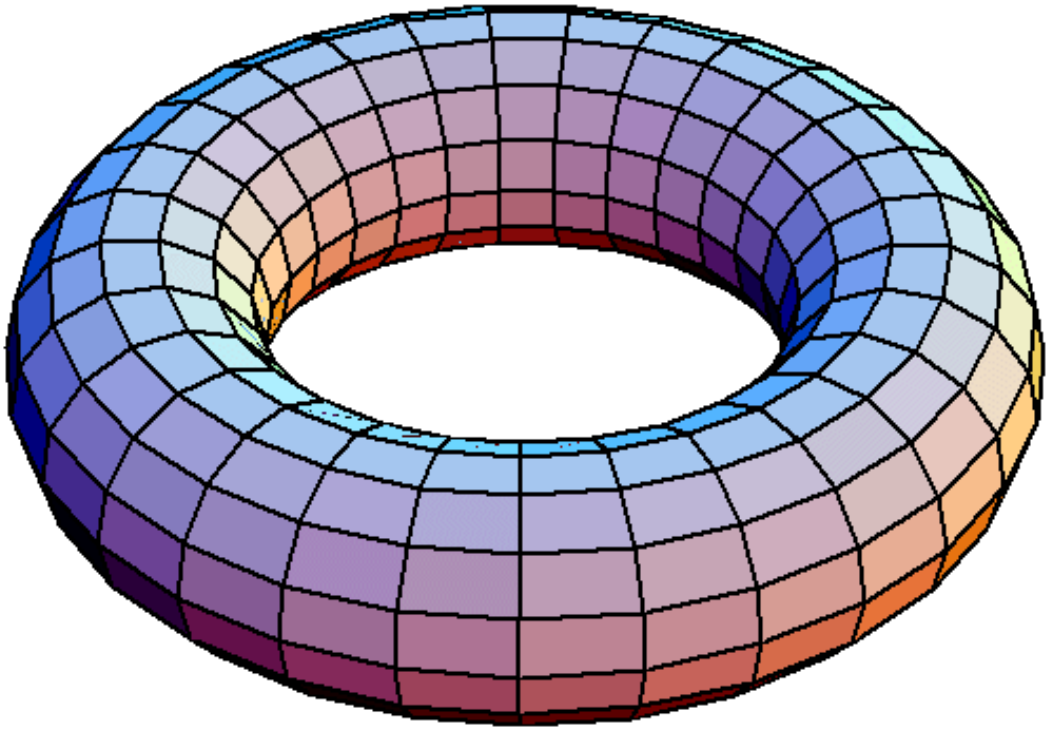
GEOMETRÍA COMPUTACIONAL

PRÁCTICA 5



DAVID SEIJAS PÉREZ

1. Introducción

En esta práctica queremos representar gráficamente en 3D el difeomorfismo de una proyección estereográfica, en concreto, queremos ver la proyección estereográfica de S_1^2 sobre $\mathbb{R}^2$ desde el punto $e^3 = (0, 0, 1)$

$$f : S_1^2\ e^3 \to \mathbb{R}^2$$
$$(x, y, z) \longmapsto (\frac{x}{(1-z)^\alpha},\ \frac{y}{(1-z)^\alpha})$$

Además, aprenderemos a representar animaciones, en este caso de la evolución de la proyección estereográfica con la familia de paramétrica indicada.

2. Material Usado

Para el desarrollo de esta práctica he utilizado diversas librerías dadas por Python. Primero, como siempre, hemos usado *matplotlib.pyplot* para la representación de gráficos y *np* para el manejo de arrays. Además de estas, hemos usado la librería *animation* de *matplotlib* para realizar la animación de la proyección.

Además, he usado la plantilla *GCOM2022-practica5_plantilla* para reutilizar diversas partes de código como la representación de la malla regular de puntos de S_1^2 de la manera indicada, la representación de las curvas del ecuador y meridiano de la esfera y algunas funciones implementadas en la plantilla.

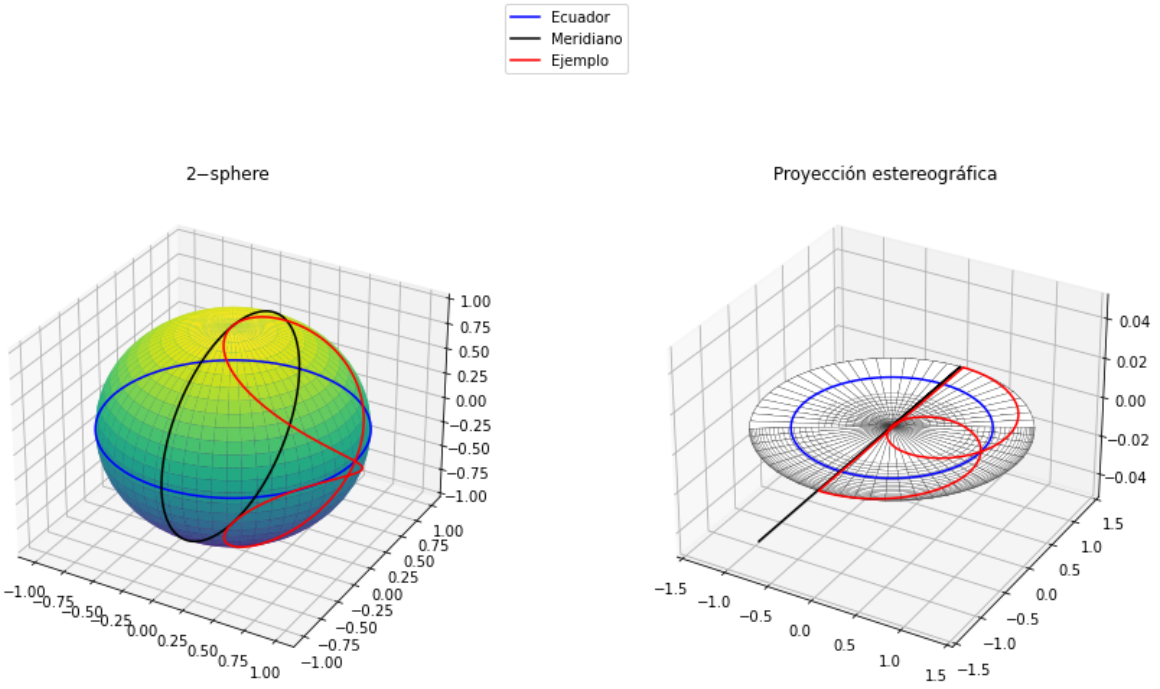
3. Metodología

Antes de nada, y gracias a la plantilla, definimos la malla de puntos de la esfera y las 3 curvas que vamos a usar, el ecuador, el meridiano y *la curva de Viviani*, representado exclusivamente por mí. Para el primer apartado, he hecho la función *apartado1*, reutilizando código de la plantilla para mostrar varios subgráficos en uno solo. De esta forma, en un gráfico mostramos la esfera y las 3 curvas usadas sobre ella dibujadas en distintos colores gracias al mapa de colores *viridis* usado. En un segundo gráfico mostramos la proyección estereográfica de la esfera y las 3 curvas gracias a la función *proj* reutilizada, también, de la plantilla.

En el segundo apartado, lo que he hecho es implementar las funciones *animation* e *init*, modificadas de la plantilla según la familia paramétrica que se especifica en el enunciado para ser usada. De esta manera, mostramos la animación de la proyección estereográfica de la esferea sobre el plano con 30 fotogramas a 5 fps.

4. Resultados

En el primer apartado he obtenido la siguiente representación de la esfera y sus curvas y las respectivas proyecciones estereográficas.



La curva que he usado es *la curva de Viviani*, dada por las ecuaciones

$$\begin{cases} x = r + r * cos(t),\ t \in [0, 4\pi] \\ y = r * sen(t) \\ z = 2 * r * sen(t/2) \end{cases}$$

Para el segundo apartado envío el archivo *animacion.gif* donde se va la animación de la proyección.

5. Conclusión

Gracias a esta práctica observamos y confirmamos diversas propiedas geométricas que ya conocíamos. La primera es la forma en que se transforman rectas y círculos a través de un proyección estereográfica respecto un punto **P**.

- Rectas y círculos que no pasan por el punto se transforma en rectas y círculos, respectivamente.
- Círculos que pasan por **P** se transforman en rectas.

Además, vemos como la esfera puede transformarse en el plano a través de un difeomorfismo (extrayendo el punto $e^3 = (0, 0, 1)$) por lo que podemos decir que S^2_1 y $\mathbb{R}^2$ son topológicamente equivalentes.

6. Anexo: Código

```
"""
DAVID SEIJAS PEREZ
PRACTICA 5
"""

import numpy as np
import matplotlib.pyplot as plt
from matplotlib import animation

u = np.linspace(0, np.pi, 30)
v = np.linspace(0, 2*np.pi, 60)
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))

#Curvas en esfera de radio 1
#Ecuador
t = np.linspace(0, 2*np.pi, 100)
x0 = np.sin(t)
y0 = np.cos(t)
z0 = np.zeros(len(t))

#Meridiano
x1 = np.zeros(len(t))
y1 = np.sin(t)
z1 = np.cos(t)

#Mi curva
t2 = np.linspace(0, 4*np.pi, 200)
x2 = 0.5*(1 + np.cos(t2))
y2 = 0.5*np.sin(t2)
z2 = np.sin(t2/2)


def proj(x, z, z0=1, alpha=1/2):
    z0 = z*0+z0
    eps = 1e-16
    return(x/(abs(z0-z)**alpha+eps))


def animate(t, eps = 1e-16):
    aux = 2/(2*(1-t) + (1-z)*t + eps)  #eps para no dividir por 0
    xt = aux*x
    yt = aux*y
    zt = z*(1-t)-t
```

```
46
47      ax = plt.axes(projection='3d')
48      ax.set_zlim3d(-1,1)
49      ax.plot_surface(xt, yt, zt, rstride=1, cstride=1, alpha=0.5, cmap=
            'viridis', edgecolor='none')
50
51      return ax,
52
53  def init():
54      return animate(0),
55
56
57  def apartado1():
58      Xp = proj(x,z)
59      Yp = proj(y,z)
60
61      #Mostramos la esfera y las curvas con sus proyeccion
            estereogr ficas
62      fig = plt.figure(figsize=(15,15))
63      fig.subplots_adjust(hspace=0.4, wspace=0.2)
64
65      ax = fig.add_subplot(2, 2, 1, projection='3d')
66      ax.plot_surface(x, y, z, rstride=1, cstride=1, cmap='viridis',
            edgecolor='none')
67      ax.plot(x0, y0, z0, '-b', zorder=3, label="Ecuador")
68      ax.plot(x1, y1, z1, 'black', zorder=3, label="Meridiano")
69      ax.plot(x2, y2, z2, '-r', zorder=3, label="Ejemplo")
70      ax.set_title('2  sphere  ')
71
72      ax = fig.add_subplot(2, 2, 2, projection='3d')
73      ax.set_xlim3d(-1.5, 1.5)
74      ax.set_ylim3d(-1.5, 1.5)
75      ax.plot_surface(Xp, Yp, z*0, rstride=1, cstride=1, cmap='binary',
            edgecolor='black', linewidth=0.2)
76      ax.plot(proj(x0, z0), proj(y0, z0), z0*0, '-b', zorder=3)
77      ax.plot(proj(x1, z1), proj(y1, z1), z1*0, 'black', zorder=3)
78      ax.plot(proj(x2, z2), proj(y2, z2), z2*0, '-r', zorder=3)
79      ax.set_title('Proyecci n estereogr fica')
80      fig.legend(loc="upper center")
81
82      plt.show()
83      plt.close()
84
85  def apartado2():
86      fig = plt.figure(figsize=(6,6))
87      ani = animation.FuncAnimation(fig, animate, np.arange(0, 1, 0.05)
            , init_func=init, interval=30)
88      ani.save("animacion.gif", fps=5)
89
90
91  apartado1()
92  apartado2()
```