Memoria Practica 5 – DEFORMACION DE VARIEDADES DIFERENCIABLES

Jorge del Valle Vázquez

1 Introducción

En esta práctica buscamos representar den 3D gráficamente el difeomorfismo de una proyección estereográfica, trabajando con la esfera unitaria S^2, embebida en $\mathbb{R}^3$. Para ello, extraemos el polo norte e3 := (0,0,1) de la esfera para dar lugar al homeomorfismo esperado entre S^2 y $\mathbb{R}^3$.

2 Material usado

Se hace uso de la plantilla en Python proporcionada por el docente. Las graficas se realizan por medio de la librería **matplotlib**, y en particular **animation** para las animaciones. Entre otras herramientas auxiliares estría **numpy** para tratamiento de vectores y cálculos asociados.

3 Resultados

En primer lugar, creamos una malla para representar con coordenadas esféricas la esfera. Definimos 30 valores de latitud con u ∈ (0.1, π] (para deshacernos del valor u =0 que da lugar a e3) y 60 valores de longitud con v ∈ [0, 2π). De esta forma obtenemos las coordenadas de S^2:

x = sin(u) × sin(u)

y = sin(u) × cos(v)

z= = cos(u) × (1…1)

Hemos obtenido matrices para las que cada columna indica una longitud concreta, y cada fila una latitud. Nota: (1…1) representa un vector de 60 unos, tantos como valores de longitud tiene v, pues para la misma latitud la coordenada z es invariante (corte por un plano z=k).

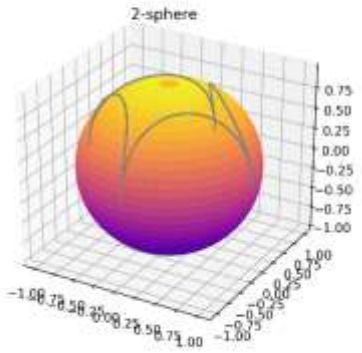
Después consideramos una curva sobre la esfera. Para ello, comenzamos con el astroide y posteriormente calculamos el valor de z

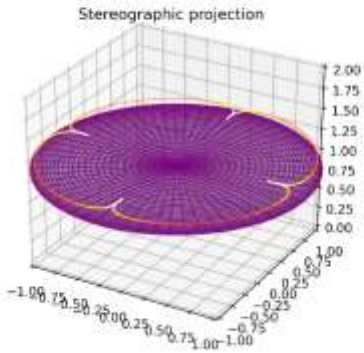
$$x = (\cos\theta)^3$$

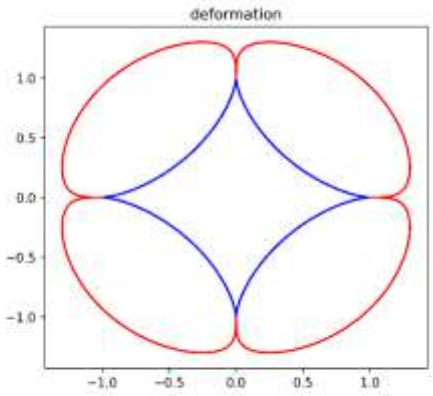
$$y = (\sin\theta)^3$$

$$z = \sqrt[2]{1 - x^2 - y^2}$$

Teniendo por proyección $x' = \frac{x}{(1-z)^\alpha}$ $y' = \frac{y}{(1-z)^\alpha}$ $z' = 1$ y considerando $\alpha = 0.5$ obtenemos la proyección sobre z = 1.







En el segundo apartado se considera una nueva familia paramétrica de t que tiene por limite la proyección estereográfica sobre z = -1.

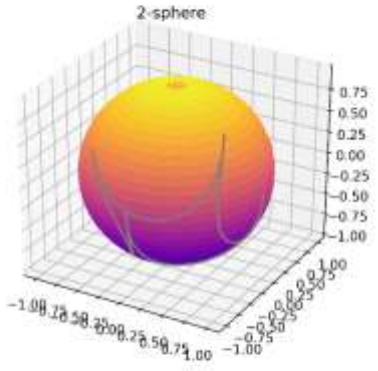
$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \frac{2}{2(1-t) + (1-z)t} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ (-1)t + z(1-t) \end{pmatrix}$$

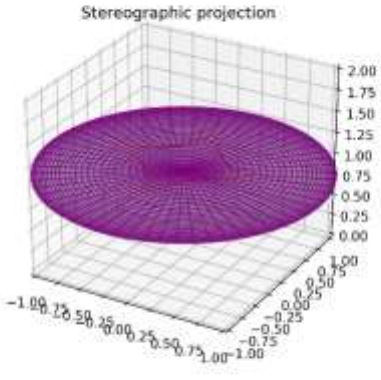
A partir de esta se construye una animación.

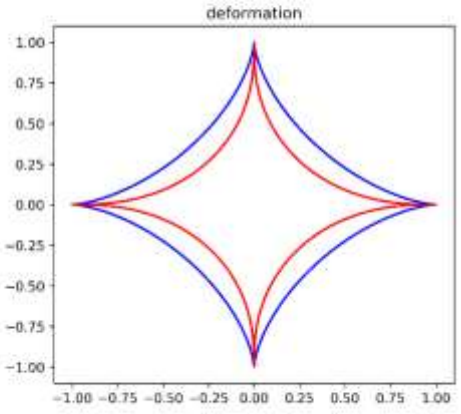
4 Conclusión

En la proyección del primer apartado, se aprecia como se deforma la proyección de la curva respecto de la proyección perpendicular, se aleja del centro de la circunferencia debido a que la curva está en el hemisferio norte, es decir, cercanos a e3, el punto extraído. Si hubiéramos

considerado $z = -\sqrt[2]{1 - x^2 - y^2}$ veríamos como se acerca al origen.







En la animación resalta la idea de eliminar el punto e3 para el correcto funcionamiento.

5 Anexo: Código

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

u = np.linspace(0.1, 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))
```

```
t2 = np.linspace(0.001, 1,5000)
x2 = np.sin(80 * t2/2)**3
y2 = np.cos(80 * t2/2)**3
z2 = np.sqrt(1-x2**2-y2**2)
c2 = x2 + y2

"""
2-esfera proyectada
"""

def proj(x,z,z0=1,alpha=0.5):
    z0 = z*0+z0
    eps = 1e-16
    x_trans = x/(abs(z0-z)**alpha+eps)
    return(x_trans)

z0 = 1

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

c2 = np.sqrt(x2**2+y2**2)
col = plt.get_cmap("hot")(c2/np.max(c2))

ax = fig.add_subplot(2, 2, 1, projection='3d')
ax.plot_surface(x, y, z, rstride=1, cstride=1, cmap='plasma',
edgecolor='none',alpha=0.9)
ax.plot(x2, y2, z2, '-b',c="gray",zorder=3)
ax.set_title('2-sphere');

ax = fig.add_subplot(2, 2, 2, projection='3d')
ax.set_xlim3d(-1,1)
ax.set_ylim3d(-1,1)
ax.set_zlim3d(0, 2)
ax.plot_surface(proj(x,z,z0=z0), proj(y,z,z0=z0), z*0+1, rstride=1,
cstride=1,cmap='viridis', alpha=0.5, edgecolor='purple')
ax.scatter(proj(x2, z2, z0=z0), proj(y2, z2, z0=z0),1+0.1, '-', c=col,
zorder=3, s=0.1)
ax.set_title('Stereographic projection');

ax = fig.add_subplot(2, 2, 3)
ax.plot(x2,y2, 0, '-b',c="blue",zorder=1)
ax.plot(proj(x2,z2,z0=z0), proj(y2,z2,z0=z0), 0, '-b',c="red",zorder=1)
ax.set_title('deformation');

plt.show()
fig.savefig('stereo1.png', dpi=250)
plt.close(fig)
```

```
z0=-1
from matplotlib import animation

def param(v,z,t,z0=-1):
    z0 = z*0+z0
    eps = 1e-16
    v_trans = (2*v) / (2*(1-t)+(1-z)*t + eps)
    return(v_trans)

def animate(t):
    xt = param(x, z, t)
    yt = param(y, z, t)
    zt = -t + z*(1-t)
    x2t = param(x2, z2, t)
    y2t = param(y2, z2, t)
    z2t = -t + z2*(1-t)

    ax = plt.axes(projection='3d')
    ax.set_xlim3d(-6, 6)
    ax.set_ylim3d(-6, 6)
    ax.set_zlim3d(-3, 3)
    ax.plot_surface(xt, yt, zt, rstride=1, cstride=1, alpha=0.5,
cmap='viridis', edgecolor='none')
    ax.scatter(x2t, y2t, z2t, '-', c=col, zorder=3, s=0.1)
    return ax,

def init():
    return animate(0),

fig = plt.figure(figsize=(6, 6))
ani = animation.FuncAnimation(fig, animate, np.arange(0, 1.001, 0.025),
init_func=init, interval=20)
ani.save("anim1.mp4", fps=5)
```