Práctica 5 - Geometría Computacional

Marcos Herrero Agustín

1. Introducción

En esta práctica se pretende representar gráficamente en 3D el difeomorfismo de una proyección estereográfica. En el apartado i) se representará en $\mathbb{R}^3$ la esfera unidad S^2 con una curva sobre ella y, a continuación, una proyección estereográfica suya con la curva anterior transformada. En el apartado ii) se construirá una animación que mostrará una familia paramétrica dada.

2. Datos y condiciones iniciales

Para la realización de la práctica se han utilizado los siguientes datos y condiciones iniciales:

- Para el apartado i):
  - Los rangos de valores para las coordenadas esféricas de la rejilla, así como el número de valores a generar. También la indicación de que se debe excluir el polo norte de $e_3 = (0, 0, 1)$ de la esfera.
  - El valor de $\alpha = 0.5$ que caracteriza a la proyección estereográfica.
- Para el apartado ii):
  - La familia paramétrica $\{f_t\}_{t\in[0,1]}$ a representar.
  - El número de fotogramas mínimo (20) de la animación pedida.

3. Metodología

En el apartado i), hemos seguido los siguientes pasos:

1. Generar las coordenadas esféricas de la rejilla según las restricciones pedidas.
2. Transformar las coordenadas esféricas anteriores a cartesianas.
3. Construir una curva sobre la esfera. Se ha elegido la curva:

$$\begin{cases} x(t) = \dfrac{\cos(2t)}{\sqrt{1+4t^2}} \\ y(t) = \dfrac{-\sin(2t)}{\sqrt{1+4t^2}} \\ z(t) = \sqrt{1-x^2(t)-y^2(t)} = \dfrac{2t}{\sqrt{1+4t^2}} \end{cases} , t \in [-1, 1]$$

4. Representar la esfera, con la curva sobre ella.
5. Proyectar la esfera y la curva sobre el plano $z = 0$ mediante una proyección estereográfica con $\alpha = 0.5$.

Para el apartado ii), hemos comenzado programando la función fAnimation (t, xs, ys, zs), que genera un fotograma representando un miembro f_t de la familia paramétrica. A continuación, simplemente hemos llamado a la función FuncAnimation de la biblioteca matplotlib para obtener la figura deseada, que hemos guardado como un gif.

4. Resultados y discusión

4.1. Apartado i)

En la figura 1 se muestra la esfera unidad con la curva elegida sobre ella. En la figura 2 se muestra la proyección estereográfica de ambas (con $\alpha = 0.5$) sobre el plano $z = 0$. Observamos que la curva escogida se deforma al proyectar la esfera sobre la que se apoya.

4.2. Apartado ii)

Junto a esta memoria se adjunta un archivo "animación.gif" que constituye la animación pedida, representando la familia paramétrica f_t. Las figuras 3 y 4 son algunos de los fotogramas que constituyen la animación. Se puede apreciar como la esfera se aplana conforme aumenta el valor de $t \in [0, 1]$

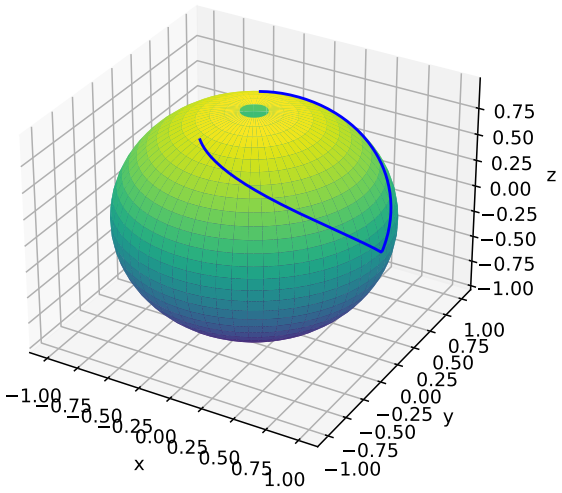

Figura 1

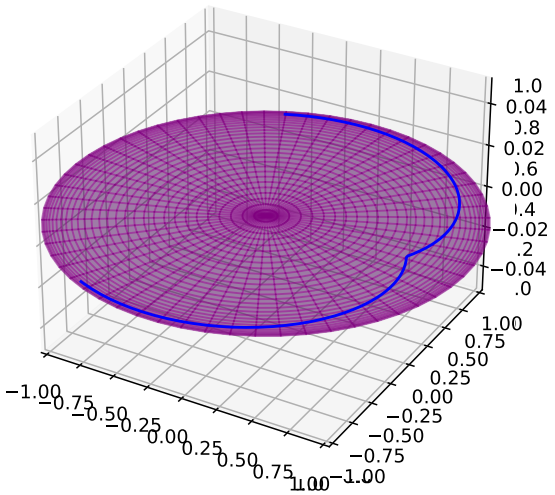

Figura 2

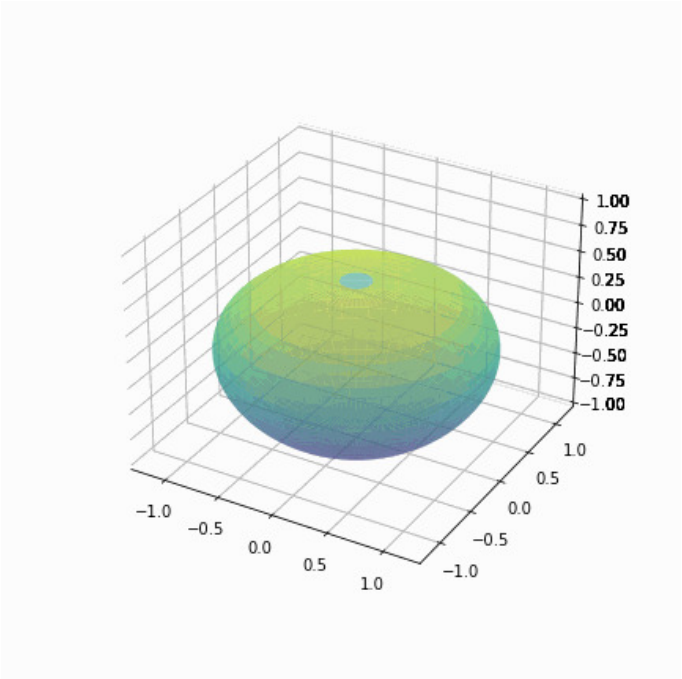

Figura 3: Fotograma 5

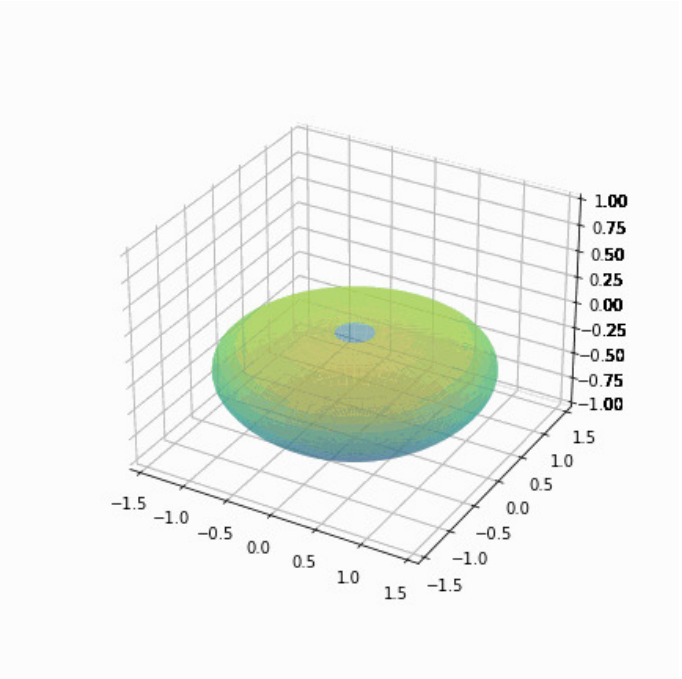

Figura 4: Fotograma 10

5. Conclusiones

La proyección estereográfica constituye una prueba de que la esfera unidad sin su polo norte es difeomorfa al plano $z = 0$. Se deduce que, en general, una esfera menos un punto cualquiera suyo es difeomorfa a cualquier plano, es decir, que vistos como variedades diferenciables tienen las mismas propiedades. La familia paramétrica $\{f_t\}_t$, por su parte, constituye otra transformación diferenciable de la esfera menos un punto en un plano, en este caso, el $z = -1$. Gracias a la animación, podemos apreciar visualmente la suavidad de la transformación.

Apéndice A Código utilizado

```
"""
Práctica 5 de Geometría Computacional
Autor: Marcos Herrero
"""

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

'''
Funciones auxiliares
'''

#Realiza la proyección estereográfica de la esfera con el alpha dado.
#Devuelve las coordenadas cartesianas proyectadas de la esfera (xs,ys,zs) sobre el plan
def proyEstereo(xs,ys,zs,alpha):
    xsproy = np.divide(xs,abs(1-zs)**alpha, out = np.zeros(xs.shape),where = zs != 1)
    ysproy =  np.divide(ys,abs(1-zs)**alpha, out = np.zeros(ys.shape), where = zs != 1)
    zsproy = np.full(zs.shape,0)

    return xsproy,ysproy,zsproy

def fAnimation(t,xs,ys,zs):
    xstcurva = 2*xs/(2*(1-t)+(1-zs)*t)
    ystcurva = 2*ys/(2*(1-t)+(1-zs)*t)
    zstcurva = -t+zs*(1-t)

    ax = plt.axes(projection='3d')
    ax.set_zlim3d(-1,1)
    ax.plot_surface(xstcurva, ystcurva, zstcurva, rstride=1, cstride=1, alpha=0.5,
                    cmap='viridis', edgecolor='none')


'''
Apartado i): Estimar y representar una malla regular de puntos de S1.
Estimar y representar la imagen de la proyección estereográfica
Diseñar una curva sobre S1 para comprobar cómo se deforma.
'''

#Coordenadas esfericas segun las restricciones pedidas
lats = np.linspace(0.1,np.pi,30)
lons = np.linspace(0,2*np.pi,60)

#Las pasamos a coordenadas cartesianas
xs = np.outer(np.sin(lats), np.sin(lons))
ys = np.outer(np.sin(lats), np.cos(lons))
zs = np.outer(np.cos(lats), np.ones_like(lons))

#Diseñamos una curva sobre la esfera
ts = np.linspace(-1.0, 1.0, 200)
xscurva = np.cos(2*ts)/np.sqrt(1+4*ts**2)
yscurva = -np.sin(2*ts)/np.sqrt(1+4*ts**2)
zscurva = np.sqrt(1-xscurva**2-yscurva**2)
```

```
#Representamos la esfera con una curva sobre ella
fig = plt.figure(figsize=(5,5))
ax = plt.axes(projection='3d')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
ax.plot_surface(xs, ys, zs, rstride=1, cstride=1,
                cmap='viridis', edgecolor='none')
ax.plot(xscurva, yscurva, zscurva, '-b',zorder=3)
fig.savefig("Esfera.pdf",format='pdf')
plt.show()

#Proyectamos sobre el plano z= 0 mediante proy. estereo. con alpha = 0.5
fig = plt.figure(figsize=(5,5))
ax = plt.axes(projection ='3d')
alpha = 0.5
xsproy, ysproy, zsproy = proyEstereo(xs,ys,zs,alpha)
xscurvproy, yscurvproy, zscurvproy = proyEstereo(xscurva,yscurva,zscurva,alpha)
ax = plt.axes(projection='3d')
ax.set_xlim3d(-1,1)
ax.set_ylim3d(-1,1)
ax.plot_surface(xsproy, ysproy,zsproy,rstride=1, cstride=1,
                cmap='viridis',alpha =0.5, edgecolor='purple')
ax.plot(xscurvproy,yscurvproy,zscurvproy, '-b',zorder=3)
fig.savefig('EsferaProyectada.pdf',format='pdf')
plt.show()

'''
Apartado ii): Obtener una animación de al menos 20 fotogramas de la familia paramétrica
'''

fig = plt.figure(figsize=(6,6))
ani = animation.FuncAnimation(fig, fAnimation,np.arange(0,1.05,0.05),
                              fargs = (xs,ys,zs),interval = 20)
ani.save("animación.gif", fps = 5)
```