1. Introducción

En esta práctica representaremos gráficamente el difeomorfismo de una proyección estereográfica de la esfera $S_1^2 \subset \mathbb{R}^3$. Para que la esfera sea topológicamente equivalente al plano necesitamos extraer un punto que será el que mandaremos al infinito, en este caso será el polo norte o sur de la esfera. Como representaremos la esfera como una esfera de radio 1 centrada en el origen, el polo norte corresponde al punto $(0, 0, 1)$ y el sur al $(0, 0, -1)$.

2. Material empleado

Dividimos esta representación en dos apartados: el primero corresponderá en representar la esfera como un mallado de puntos y aplicaremos la proyección estereográfica a este mallado. También diseñaremos una curva que forma parte de la esfera para ver como es deformada por la proyección. El segundo apartado será una animación de la proyección de varios fotogramas para poder apreciar como se va deformando la esfera hasta estar embebida en el plano.

Como datos de entrada para representar la esfera contamos con 25 valores de latitud ($\phi \in [0, \pi)$) y 50 valores de longitud ($\varphi \in [0, 2\pi)$) representados como dos vectores de elementos equiespaciados. Con estos vectores podríamos representar el mallado de la esfera pero en coordenadas esféricas, para crear el mallado aplicamos las siguientes fórmulas y obtendremos tres matrices de dimensión 25×50:

$$\begin{aligned} X &:= \sin\phi \times \sin\varphi \\ Y &:= \sin\phi \times \cos\varphi \\ Z &:= \cos\phi \times I \end{aligned}$$

dónde I es un vector de "1" de la misma dimensión que φ. Después definimos una curva $s(t)$ cuya parametrización es la siguiente:

$$\begin{aligned} x(t) &:= |t| \sin\frac{Kt}{2} \\ y(t) &:= |t| \sin\frac{Kt}{2} \\ z(t) &:= \sqrt{1 - x(t)^2 - y(t)^2} \end{aligned}$$

donde $t \in (0, 1]$ y $K > 0$ es una constante. Después lo único que haría falta sería aplicar la función de la proyección a las tres matrices, la función que aplicamos es la siguiente:

$$\begin{aligned} x &\to \frac{x}{|1 - z|} \\ y &\to \frac{y}{|1 - z|} \\ z &\to 1 \end{aligned}$$

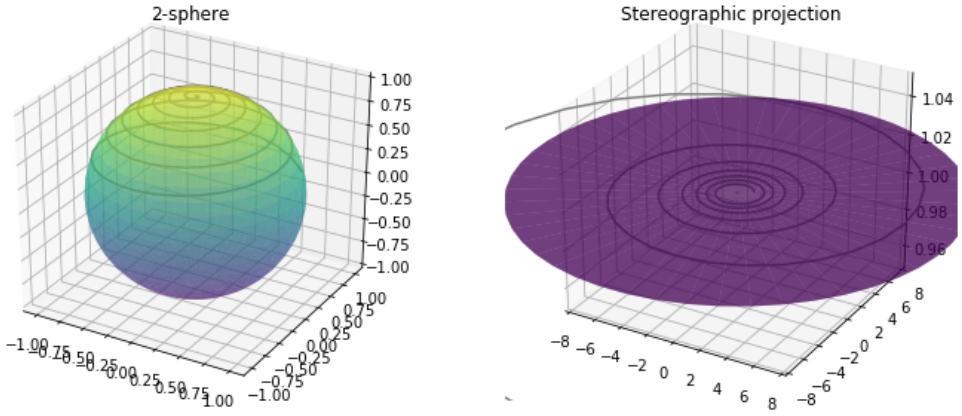
Y con esto ya obtendríamos la proyección estereográfica de la esfera S^2_1 en el plano $\{z = 1\}$ que manda el polo norte al infinito. Para simplificar los cálculos hemos decidido dejar la tasa de deformación $\alpha = 1$.

Para el segundo apartado necesitamos una proyección que sea una transformación continua, para ello definimos la siguiente transformación $f(t)$:

$$\begin{aligned} x &\to \frac{x}{(1-t) + |-1-z|t} \\ y &\to \frac{y}{(1-t) + |-1-z|t} \\ z &\to (-1)t + z(1-t) \end{aligned}$$

donde $t = [0, 1]$, $f(0)$ es la función identidad y deja cada punto de la esfera igual y $\lim_{t\to 1} f(t)$ es el plano $\{z = -1\}$. Discretizamos el intervalo [0,1] tomando puntos equiespaciados y utilizamos los diferentes valores de t para obtener los diferentes fotogramas de la animación.

3. Resultados



4. Conclusión

De conclusiones, únicamente comentar que como se puede ver en los resultados, la curva se mantiene dentro de la esfera deformada transformándose en una curva plana. También cabe comentar un detalle en la implementación de la función proyección y es que añadimos un pequeño parámetro $\epsilon > 0$ al que elevamos el denominador de esta para evitar dividir entre cero.

5. Código

```
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 10 18:58:33 2020

@author: Jorge y Lucas con la plantilla de Robert
"""

#from mpl_toolkits import mplot3d

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

os.getcwd()

"""
Ejemplo1
"""
"""
fig = plt.figure()
ax = plt.axes(projection='3d')
X, Y, Z = axes3d.get_test_data(0.05)
cset = ax.contour(X, Y, Z, 16, extend3d=True)
ax.clabel(cset, fontsize=9, inline=1)
plt.show()

Ejemplo2


def g(x, y):
    return np.sqrt(1-x ** 2 - y ** 2)

x = np.linspace(-1, 1, 30)
y = np.linspace(-1, 1, 30)

X, Y = np.meshgrid(x, y)
Z = g(X, Y)


fig = plt.figure()
```

```
ax = plt.axes(projection='3d')
ax.contour3D(X, Y, Z, 10, cmap='binary')
ax.contour3D(X, Y, -1*Z, 10, cmap='binary')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')


Ejemplo3

fig = plt.figure()
ax = plt.axes(projection='3d')

t2 = np.linspace(1, 0, 100)
x2 = t2 * np.sin(20 * t2)
y2 = t2 * np.cos(20 * t2)
z2 = np.sqrt(1-x2**2-y2**2)

c2 = x2 + y2

ax.scatter(x2, y2, z2, c=c2)
ax.plot(x2, y2, z2, '-b')



2-sphere
"""

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

#Cambiamos de coordenadas polares a cartesianas
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))

#Definimos una curva en la superficie de la esfera
t2 = np.linspace(0.001, 1, 200)
x2 = abs(t2) * np.sin(90 * t2/2)
y2 = abs(t2) * np.cos(90 * t2/2)
z2 = np.sqrt(1-x2**2-y2**2)

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)
    #Nótese que añadimos un épsilon para evitar dividi entre 0!!



def apartado1():
    print("Apartado 1:\n")
    global x,y,z,t2,x2,y2,z2


    """
    2-esfera y su proyección estereográfica
    """

    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,alpha=0.5,
                    cmap='viridis', edgecolor='none')
    ax.plot(x2, y2, z2, '-b',c="gray")
    ax.set_title('2-sphere');
    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,
        alpha=0.5, cmap='viridis', edgecolor='none')
    ax.plot(proj(x2,z2), proj(y2,z2), 1, '-b',c="gray")
    ax.set_title('Stereographic projection');

    plt.show()
    plt.close(fig)

def proj2(x,z,t,z0=-1,alpha=1):
    z0 = z*0+z0
    eps = 1e-16
    x_trans = x/((1-t)+t*(abs(z0-z)**alpha+eps))
    return(x_trans)
    #Nótese que añadimos un épsilon para evitar dividi entre 0!!
```

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

def animate(t):
    xt = proj2(x,z,t)
    yt = proj2(y,z,t)
    zt = -1*t + z*(1-t)
    x2t = proj2(x2,z2,t)
    y2t = proj2(y2,z2,t)
    z2t = -1*t + z2*(1-t)

    ax = plt.axes(projection='3d')
    ax.set_zlim3d(-1,1)
    ax.plot_surface(xt, yt, zt, rstride=1, cstride=1,alpha=0.5,
                    cmap='viridis', edgecolor='none')
    ax.plot(x2t,y2t, z2t, '-b',c="gray")
    return ax,

def init():
    return animate(0),


def apartado2():

    global x,y,z,t2,x2,y2,z2,z0
    print("Apartado 2:\n")
    t = 0.1


    xt = proj2(x,z,t)
    yt = proj2(y,z,t)
    zt = -1*t + z*(1-t)
    x2t = proj2(x2,z2,t)
    y2t = proj2(y2,z2,t)
    z2t = -1*t + z2*(1-t)

    fig = plt.figure(figsize=(6,6))
    ax = plt.axes(projection='3d')
```

```
    ax.set_xlim3d(-1,1)
    ax.set_ylim3d(-1,1)
    ax.plot_surface(xt, yt, zt, rstride=1, cstride=1,alpha=0.5,
                    cmap='viridis', edgecolor='none')
    ax.plot(x2t,y2t, z2t, '-b',c="gray")

    plt.show()
    plt.close(fig)

    """
    HACEMOS LA ANIMACIÓN
    """



    animate(np.arange(0, 1,0.1)[1])
    plt.show()


    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)


apartado1()
apartado2()
```