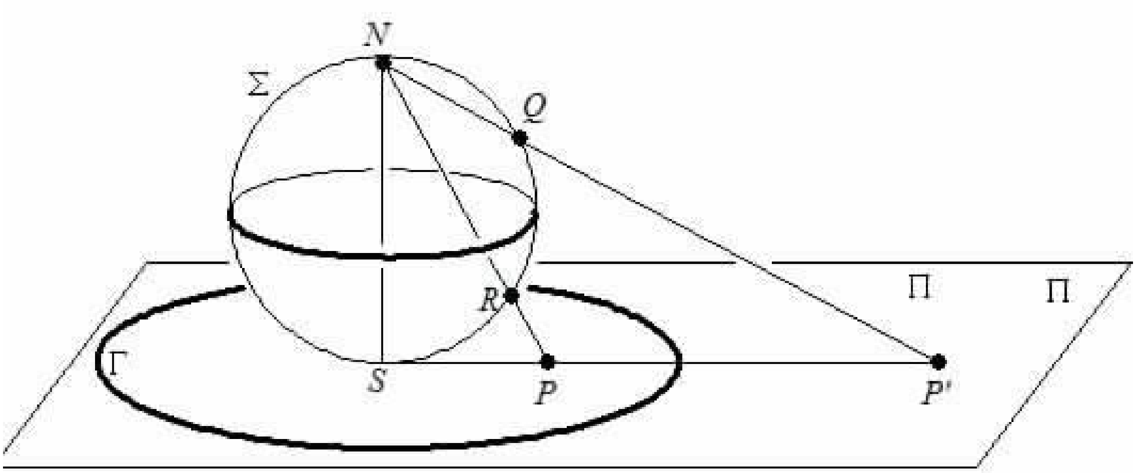
GEOMETRÍA COMPUTACIONAL

DEFORMACIÓN DE VARIEDADES DIFERENCIALES



LUCAS DE TORRE

Índice

1. Introducción

Buscamos una familia paramétrica $g_t : S_1^2 \setminus e_3 \to \mathbb{R}^3$ tal que exista un $t_0 \in \mathbb{R}$ que verifique que $\lim_{t \to t_0} g_t \simeq \Pi$ y $g_0(p) = p$. Para ello, utilizaremos las propiedades de las funciones tan y $\tan^{-1}$.

Después, obtendremos una animación de al menos 15 fotogramas de esa familia paramétrica.

2. Material

Para buscar la familia paramétrica, pretendemos que el parámetro t recorra el intervalo $[0,1]$. Por un lado, tenemos en cuenta que tan(0)=0. Con esto, podremos "mandar a infinito" el punto $e_3 = (0,0,-1)$. Así, podemos tener, por ejemplo, $\tan^{-1}(t|-1-z|)$, que se anula en el punto e_3 cuando $t = 1$.

Por otro lado, pretendemos que cuando $t = 0$, tengamos la identidad y sabemos que tan($\pi/4$)=1. Así, podemos tener, (1-t)$\pi/4$, que es igual a $\pi/4$ cuando $t = 0$.

De esta manera, terminando de ajustar coeficientes, tenemos la siguiente familia paramétrica:

$$g_t : S_1^2 \setminus e_3 \to \mathbb{R}^3$$

$$p = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \frac{1}{\tan(\tan^{-1}(t|-1-z|))\pi/4 + (1-t)\pi/4)} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ (-1)t + z(1-t) \end{pmatrix}$$

Veamos que es continua donde la hemos definido. La función $\tan^{-1}$ solo toma valores no negativos (porque $t|-1-z| \geq 0$) y es creciente. También (1-t)$\pi/4$ verifica que es no negativa para todo $t \in [0,1]$. Así, el menor valor que puede tomar el denominador es 0 (cuando t=1), que se corresponde con la proyección estereográfica. Ahora, como $\tan^{-1}$ es creciente, el mayor valor que puede tomar $\tan^{-1}(t|-1-z|)\pi/4$ es $\tan^{-1}(2)\pi/4$, mientras que el mayor valor de (1-t)$\pi/4$ es $\pi/4$, por lo que $\tan^{-1}(t|-1-z|))\pi/4 + (1-t)\pi/4 \leq \tan^{-1}(2)\pi/4 + \pi/4 < \pi/2$.

Por tanto, el denominador toma valores entre tan(0) y tan(p) con $p < \pi/2$, por lo que la familia paramétrica es continua y verifica que para $t = 0$ es la identidad y para $t = 1$ es una proyección estereográfica.

Por último, buscamos una animación de la familia paramétrica dada por g_t, la cual transforma la esfera unidad en el plano $z = -1$. Para ello, primero escribimos una función (que llamamos *proj2*) para, dados t y z, realizar la transformación de g_t para x

e y (la de z la hacemos directamente):

$$x(t) = \text{proj2}(x, z, t)$$
$$y(t) = \text{proj2}(y, z, t)$$
$$z(t) = -1 * t + z * (1 - t)$$

Después, utilizamos la función *animation.FuncAnimation* con 20 intervalos (más de 15), para, mediante el uso de la función proporcionada *animate*, hacer la representación gráfica de g_t sobre la esfera para 20 valores de t equiespaciados en [0,1] y, con esas representaciones gráficas, realizar la animación.

3. Resultados

La familia paramétrica g_t obtenida es

$$g_t : S_1^2 \setminus e_3 \to \mathbb{R}^3 \tag{1}$$
$$p = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \frac{1}{\tan(\tan^{-1}(t|-1-z|))\pi/4 + (1-t)\pi/4)} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ (-1)\,t + z(1-t) \end{pmatrix} \tag{2}$$

La animación dada por la familia paramétrica definida por g_t la podemos encontrar en el archivo *animaciontan.gif*

4. Conclusiones

Hemos visto que, aunque con la restricción de utilizar las propiedades de las funciones tan y $\tan^{-1}$ podría parecer imposible obtener la familia paramétrica, solo era necesario analizar con cuidado esas propiedades y ver cómo encajaban en los requisitos de la familia paramétrica.

5. Anexo A: Código

```
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 10 18:58:33 2020
@author: Jorge Sainero y Lucas de Torre
"""

#from mpl_toolkits import mplot3d

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

os.getcwd()


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(107 * t2/2)
y2 = abs(t2) * np.cos(107 * t2/2)
z2 = np.sqrt(1-x2**2-y2**2)

z0 = -1



def proj2(x,z,t,z0=-1,alpha=1):
    z0 = z*0+z0
    eps = 1e-16
    x_trans = x/(np.tan(np.arctan(t*abs(-z-1))*np.pi/4+(1-t)*np.pi/4)+eps)
    return(x_trans)
    #N tese que a adimos un  psilon  para evitar dividir 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 solucion():

    global x,y,z,t2,x2,y2,z2,z0

    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 ANIMACI N
    """



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


    fig = plt.figure(figsize=(12,12))
    ani = animation.FuncAnimation(fig, animate, np.arange(0,1,0.05)
    , init_func=init,
                                  interval=20)
    ani.save("animaciontan.gif", fps = 5)



solucion()
```