Física Numérica

Tarea #3

D. A. Vázquez Gutiérrez

Escuela Superior de Física y Matemáticas, Instituto Politécnico Nacional,Unidad Profesional .[A]dolfo López Mateos", Zacatenco, Edificio 9, Col. San Pedro Zacatenco, C.P. 07730 del. Gustavo A. Madero, Ciudad de México, México

email: dvazquezg1600@alumno.ipn.mx

30 de agosto de 2024

1. Estudiando la evolución de la temperatura en una barra.

Partamos del siguiente problema particular :

Considere una barra cilíndrica de aluminio de longitud $L = 1m$ y un grosor w colocada a lo largo del eje x. Dicha barra se encuentra aislada térmica mente a lo largo de su longitud, pero no en sus extremos. Inicialmente la barra se encuentra a una temperatura uniforme de $T_0 = 100C$, y los extremos se encuentran en contacto con una barra de hielo a $0C$. El calor fluye únicamente a través de los extremos no aislados.

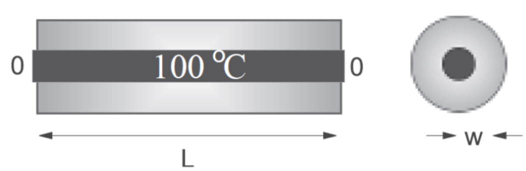

Figura 1: Diagrama del estado inicial de una barra a 100 ºC aislada por los alrededores con contacto por los extremos a hielo a 0 ºC

(a) **Solución Analítica** . Podemos recordar por nuestras clases de Métodos Matemáticos que un problema de este estilo es modelado por la llamada Ecuación Diferencial Parcial de Difusión , o también conocida solamente como *Ecuación de Calor* , su forma es la siguiente :

$$\frac{\partial \phi}{\partial t} = a^2 \nabla^2 \phi(\overrightarrow{r}, t) \tag{1}$$

Donde :

- ϕ es la concentración del material que se difunde
- t es el tiempo
- a^2 es el coeficiente de difusión
- $\overrightarrow{r}$ Coordenada espacial

Con:

$$a^2 = \frac{K}{C_p \rho} \tag{2}$$

Donde :

- ρ densidad
- K Conductividad térmica
- C_p Calor especifico a presión constante

Dado que la barra esta hecha de aluminio , veamos que entonces :

$$a_{Al}^2 = 9{,}786 \times 10^{-5} \frac{m^2}{s} \tag{3}$$

La Ecuación de Calor es posible resolver analíticamente , teniendo dos formas de hacerlo , siendo la primera el cambio de variable , pero por simplicidad , nos enfocaremos en la segunda :

Método de Separación de Variables Al ser el problema planteado en una sola dimensión , partimos suponiendo que la forma de ϕ es :

$$\phi(x,t) = X(x)T(t)$$

Evaluando esta en (1), esto solo es posible si los dos términos son iguales a una constate a elegir , sea entonces β

$$\frac{1}{T}\frac{dT}{dt} = \beta, \frac{1}{X}\frac{d^2X}{dx^2} = \frac{\beta}{a^2} \tag{4}$$

Notemos que esperamos que la temperatura disminuya , por lo tanto , consideramos que $\beta < 0$, por lo tanto , la solución tiene la siguiente forma :

$$\begin{aligned}\phi(x,t) =&(Acos(wx) + Bsen(wx))e^{-w^2a^2t} \\ &+ C_o'x + C_o\end{aligned} \tag{5}$$

Con $w^2 = \frac{\beta}{a}$. Ahora , tenemos las siguientes condiciones de frontera

$$\phi(x,0) = 100C \tag{6}$$
$$\phi(0,t) = \phi(L,t) = 0C \tag{7}$$

Sin embargo , para facilitar el desarrollo de la solución analítica , veamos que podemos color la barra en el centro , de tal forma que podamos utilizar las propiedades simétricas del coseno , entonces, partamos del intervalo $(\frac{L}{2}, \frac{-L}{2})$ y tomando en cuenta que mas alla de los limites de la barra , la temperatura es cero , entonces $C_o'x + C_o$, luego:

$$\phi(x,0) = Acos(wx) + Bsen(wx) = 100C$$

$$\begin{aligned}\phi(\frac{L}{2},t) &= (Acos(w\frac{L}{2}) + Bsen(w\frac{L}{2}))e^{-w^2a^2t} \\ &= 0C\end{aligned}$$

$$\begin{aligned}\phi(\frac{-L}{2},t) &= (Acos(w\frac{-L}{2}) + Bsen(w\frac{-L}{2}))e^{-w^2a^2t} \\ &= 0C\end{aligned}$$

De las ultimas dos , encontramos que para un l impar, tenemos que :

$$w = \frac{l\pi}{L}$$

Por lo tanto ahora hay que tratar con :

$$\phi(x,0) = \sum_{l=0}^{\infty} a_l cos(\frac{l\pi}{L}x) = 100C$$

Utilizando series de Fourier , encontramos que :

$$a_l = 400\frac{(-1)^m}{(2m+1)\pi}$$

con m un numero natural . Por lo tanto , el valor de nuestra temperatura en cualquier momento t y posicion x , de forma analítica es:

$$\begin{aligned}\phi(x,t) = &\frac{400}{\pi}\sum_{m=0}^{\infty}\frac{(-1)^m}{(2m+1)}cos((2m+1)\frac{\pi x}{L}) \\ &\times e^{-t((2m+1)\pi\frac{a}{L})^2}\end{aligned} \tag{8}$$

Creamos entonces un programa que represente esta función, que es el codigo 1, con nombre `Funcion de calor 1.py` el cual se encuentra en A.1, obtenemos la grafica en la Figura 2.

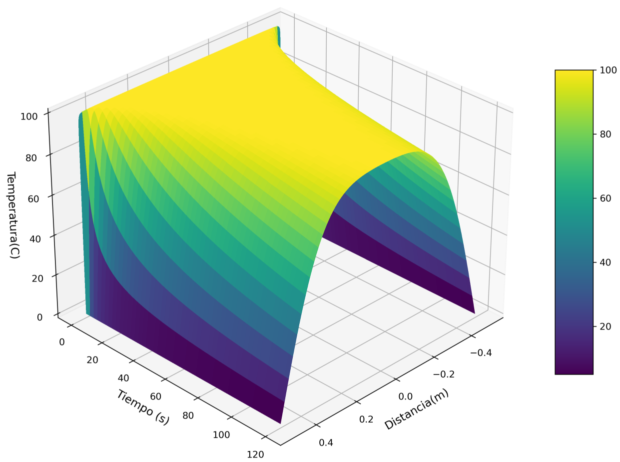

Figura 2: Grafica obtenida apartir de Codigo 1. "Funcion de calor 1". En esta podemos observar claramente como la temperatura va disminuyendo desde los extremos al centro.

(b) **Solución Numérica** .Podemos tener una solución numérica a este problema a través de una aproximación *central difference* , empleándola junto a la ecuación de calor , podemos llegar a la siguiente expresión :

$$\phi(x, t + \Delta t) =$$
$$\phi(x, t) + \frac{a^2 \Delta t}{(\Delta x)^2}[\phi(x + \Delta x, t)$$
$$+ \phi(x - \Delta x, t) - 2\phi(x, t)]$$

Donde , haciendo las siguientes sustituciones

$$\eta = \frac{a^2 \Delta t}{(\Delta x)^2}$$

$$x = i\Delta x, t = j\Delta t$$

$$\phi(x, t) = \phi(i\Delta x, j\Delta y) = \phi_{ij}$$

Obtenemos finalmente :

$$\phi_{ij+1} = \phi_{ij} + \eta[\phi_{i+1j} + \phi_{i-1j} - 2\phi_{ij}] \quad (9)$$

Creamos entonces un programa que represente esta ultima ecuación, el cual es el codigo 2 , con nombre `Funcion de calor 2.py`, que se encuentra en A.2,obteniendo con este la grafica en la Figura 3.

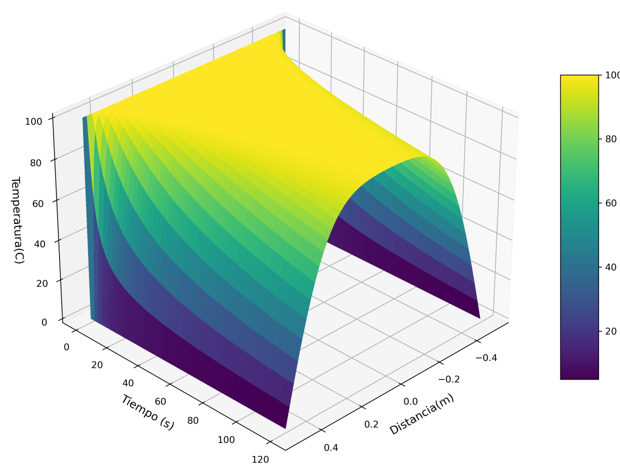

Figura 3: Grafica obtenida aparir de Código 2. Función de calor 2". Esta es prácticamente la misma que la de la Figura 1 , solo que obtenida mediante un método numérico. Con L= 1m, t_f= 120s , N_X= 1000, N_t= 25000

(c) **Condición de estabilidad de von Neumann-Courant.** Empleando la teoría aprendida en clase sobre el uso del *central difference*, sabemos que para que el algoritmo funcione apopiadamente debe haber una situacion especial. A esta le llamamos *Condicion de estabilidad de von Neumann-Courant* :

$$\eta = \frac{K\Delta t}{C_p \rho (\Delta x)^2} < \frac{1}{2} \tag{10}$$

Donde esta claro que :

$$\Delta x = \frac{L}{N_x}$$
$$\Delta t = \frac{t_f}{N_t}$$

Luego , veamos en la grafica generada en la Figura 3 se cumple que :

$$\eta_1 = 0{,}469728$$

Por lo tanto, cumple con la condicion de estabilidad de von Neumann-Courant.

Veamos que obtenemos no aplicamos las condiciones de estabilidad , diagmos entonces que ahora $N_t = 2300$, entonces :

$$\eta_2 = 0{,}510573$$

Esto no da como resultado la grafica de la figura 4.

No cumple las condiciones de frontera , ya que según la misma grafica , tiene temperatura 0ªC para toda la barra al iniciar la simulación. Por otro lado , en algún punto entre los segundos 80 y 100 hay un salto de divergencias tanto positivas como negativas, con lo que es claro que la solución no solo no es estable , si no que ademas no tiende a el equilibrio; por lo tanto no es una solución valida.

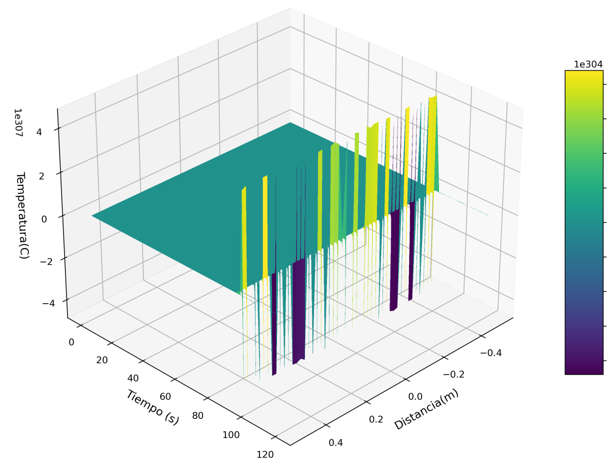

Figura 4: Grafica obtenida apartir de Código 2 sin cumplir condición de estabilidad Neumann-Courant. Función de calor 2". Con L= 1m, t_f= 120s , N_X= 1000, N_t= 23000 .

(d) **Comparación entre solución analítica y numérica**. Ya conocemos la forma de la función tanto analítica (Figura 2), como de la numérica (Figura 3) , sin embargo aun no sabemos que tanto se parecen realmente ambas soluciones.

Para esto , emplearemos una *Grafica por diferencias* , para así visualizar los puntos en los que ambas funciones mas se distancien.

Para lograr un tiempo factible de obtención de grafica , se redujo el numero de particiones de forma proporcional según la condición de estabilidad de Neumann-Courant a $N_x = 450$, y de N_t a 5000 , así como hicimos una partición similar en los códigos 1 y 2 para que las matrices tuviesen la misma dimensión.

Vemos en la figura 5 que las discrepancias mas grandes se dan tanto en la interfase entre los extremos y el interior de la barra al inicio de la simulación , así como , mientras mas evoluciona , en las zonas en las que mas la temperatura disminuye , sin embargo , no es lo suficientemente significativas las diferencias que se presentan entre la solución analítica con

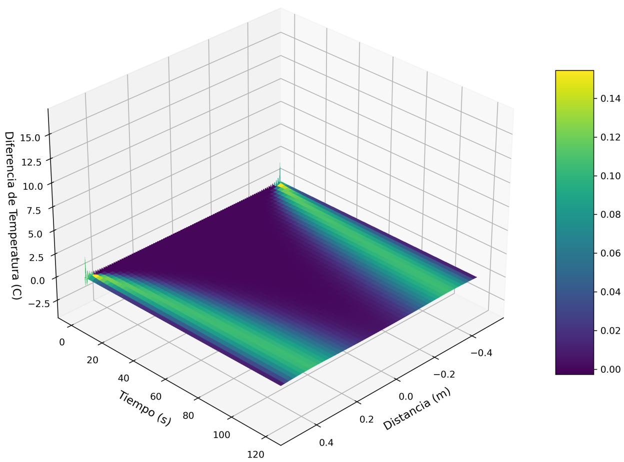

Figura 5: Grafica de diferencia entre la solución analítica y la numérica . Con L= 1m, t_f= 120s , N_X= 450, N_t= 500.

respecto a la numérica. Mientras mas iteraciones se dan a la numérica, mas cercana y parecida esta es a la analítica

(e) **Modificación en el material**. Ahora , veamos como se comporta nuestra función numérica obtenida si suponemos la barra esta compuesta de un mal conductor térmico como lo es la madera, cuya constante de difumino es :

$$a_{wood}^2 = 8{,}2 \times 10^{-8} m^2/s \tag{11}$$

Esto nos da como resultado la siguiente grafica :

Notamos en esta ultima grafica que , al ser el coeficiente de difusión tan bajo, , prácticamente durante toda la simulación se mantiene el interior de la barra a la temperatura inicial, solo con cambios mínimos a los costados .

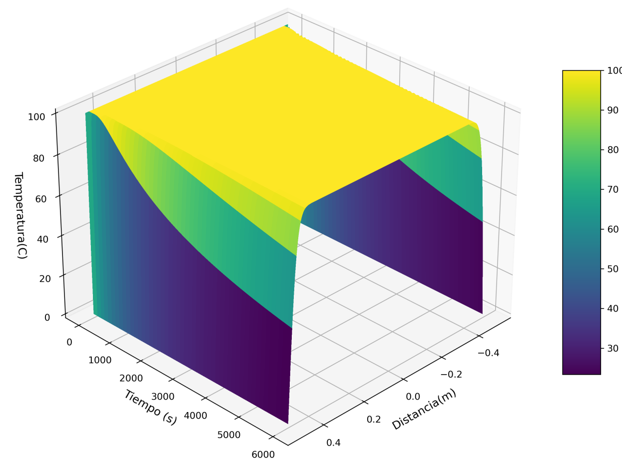

Figura 6: Grafica obtenida aparir de Código 2. bajo la hipotesis de que la barra esta hecha de madera . Función de calor 2". Esta es prácticamente la misma que la de la Figura 1 , solo que obtenida mediante un método numérico. Con L= 1m, t_f= 6000s , N_X= 1000, N_t= 1000

2. Ecuación de Poisson.

Resolveremos numéricamente la siguiente ecuación de Poisson :

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = f(x, y) \tag{12}$$

donde :

$$f(x, y) = cos(3x + 4y) - cos(5x - 2y)$$

Sujeta a las siguientes condiciones de frontera:

$$\phi(x, 0) = \phi(x, 2\pi)$$
$$\phi(0, y) = \phi(2\pi, y)$$

Para esto , haremos uso del método de *Gauss-Seidel*, implementandolo en el siguiente programa:

Entonces, notemos que partimos de una matriz cuyos coeficientes son números 'de relleno ' también referidos como 'basura', en este caso elegimos que los coeficientes sean unos.

También , por conveniencia y facilidad al observar lo que sucede en la grafica , elegimos que los limites del dominio de la función sean múltiplos enteros de π.

Lo que sigue transformar la matriz constante a valores en función de lo visto con el método de Gauss Sidel :

$$\phi_{i,j} = \frac{\rho_{i,j}}{4\epsilon_0} + \frac{1}{4}[\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1}] \tag{13}$$

Añadiendo a esto las restricciones de la función mediante condicionales.

Esto se repite y se pule tantas veces como se desee mediante la variable Z . El resultado del codigo 3, con nombre `Gauss Seidel4 Poisson.py` y que se encuentra en A.3 ,podemos verlo en la siguiente grafica :

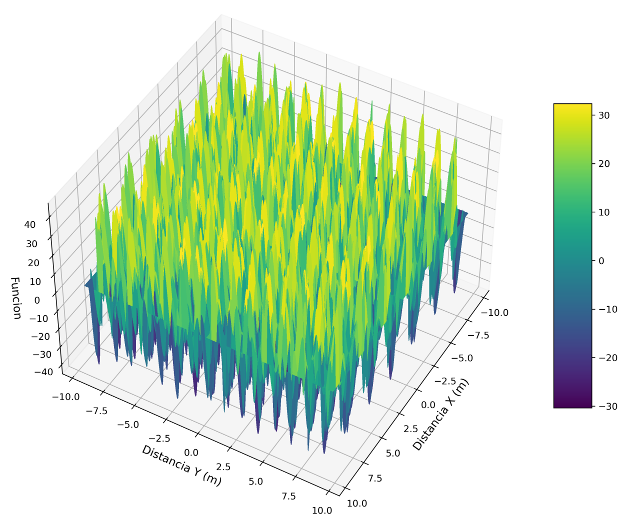

Figura 7: Grafica de la solución a la ecuación de Poisson de la ecuación 12 mediante el método de Gauss Seidel

Apéndice

A. Código en Python

A.1. Solución analítica difusión de calor

```python
import numpy as np
import math as math
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

L = 1  # longitud de la barra
alfa = 9.786e-5
To=100.0 #temperatura inicial
tf=120 #tiempo final
a = math.sqrt(alfa)
m = 1000  # Numero de iteraciones
N = 1000 # numero de pareticiones
Nx = 205
Nt = 1000

# Definir la funcion
def phi1(x, t, l, c):
    X = np.zeros_like(x)
    for i in range(m):
        sM = ((-1)**i)/(2*i+1) * np.cos((2*i+1)*np.pi*x/l) * np.exp(-t*((2*i+1)*np.pi*c/l)**2)
        X += sM
    return ((4*To)/math.pi)*X

# Crear matrices de coordenadas (x, t)
x = np.linspace(-L/2, L/2, Nx)
t = np.linspace(0, tf, Nt)
x, t = np.meshgrid(x, t)

# Calcular z utilizando la funcion definida
z = phi1(x, t, L, a)



# Crear la figura tridimensional
fig = plt.figure(figsize=(12, 12),dpi=300)
ax = fig.add_subplot(111, projection='3d')

# Graficar la superficie
surf = ax.plot_surface(x, t, z, cmap='viridis', linewidth=0, antialiased=False)



# Anadir etiquetas y titulo
```

```python
ax.set_xlabel('Distancia(m)', fontsize=12)
ax.set_ylabel('Tiempo (s) ', fontsize=12)
ax.set_zlabel('Temperatura(C)', fontsize=12)
ax.set_title('                    ', fontsize=50)


# Ajustes para mejorar la visualizacion
ax.view_init(elev=30, azim=45)  # Cambiar la elevacion y el angulo de vision
fig.colorbar(surf, shrink=0.5, aspect=8)  # Anadir barra de color


# Mostrar la grafica interactiva
plt.show()
```

Código 1: Programa que obtiene la grafica de la funcion analitica de la difusion de calor . NOMBRE : "Funcion de calor 1.py"

A.2. Solución numérica a la difusión de calor

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import math

L = 1  # longitud de la barra
#alfa = 9.786e-5 #Coeiciente de difusividad termica  aluminio
alfa = 8.091e-8
To = 100  # temperatura inicial , 100 grados celius
tf=6000#tiempo final , en este caso suponemos que transcurren 2 minutos
a = np.sqrt(alfa)
Nx = 1000 # Numero de particiones del eje x
Nt = 1000 # numero de particiones  del eje y

# Parametros
eta = alfa*(tf/Nt)/((L/Nx)**2) # Parametro eta , en funcion de la ecuacion 2
print('Eta es igual a :',eta)

# Inicializacion de la malla phi
phi = np.zeros((Nx, Nt))

# Condiciones iniciales
phi[:, 0] = To  # Todas las posiciones tienen la temperatura inicial

# el : en el phi[:,0] indica que sin importar el numero  en la coordenada x , para cualquiera que este relacionada con y=0
#tomara el valor que se quiera , en este caso To (temperatura incial)

# Condiciones de contorno
phi[0, :] = 0
phi[-1, :] = 0

#el -1 en el phi[-1, :] , es una forma en la que sin conocer la longitud de la dimencion en x de la malla , podemos indicar
#el ULTIMO elmento , de igual manera se puede indicar el PENULTIMO elemento empleando el -2 y asi sucesivamente.
#en este caso decimos que para todo ultimo valor de x en la malla, este simepre valdra 0


# Iteracion de la EDP
for j in range(1, Nt): #empezamos en 1 por que la ilera de t=0 ya esta inicializada
    for i in range(1, Nx-1): #vamos de 1 a Nx-1 por que tanto el primero (0) como el ultimo ya estan incializados en x
        phi[i, j] = phi[i, j-1] + eta * (phi[i+1, j-1] + phi[i-1, j-1] - 2 * phi[i, j-1]) #Por como esta definido phi en la ecuacion 9 , hacemos un cambio de j+1 a j

print()


# Crear una cuadricula 2D
x = np.linspace(-L/2, L/2, Nx)
t = np.linspace(0, tf, Nt)
t, x = np.meshgrid(t, x)  # Aqui t se transpone con x

# Grafico 3D de la superficie
fig = plt.figure(figsize=(12, 12),dpi=300) #dpi=300 aumenta la calidad de  la grafica , y la proporcion de figsize hace que los labels sean de un tamano apropiado
ax = fig.add_subplot(111, projection='3d') #el subpot 111, se refiere a que se proyecta la  grafica 1, de un total de graficas 1x1 , hay mas configuraciones 212, 223, etc
```

```python
surf=ax.plot_surface(x, t, phi, cmap='viridis', linewidth=0, antialiased=False)
# Anadir etiquetas y titulo
ax.set_xlabel('Distancia(m)', fontsize=12)
ax.set_ylabel('Tiempo (s) ', fontsize=12)
ax.set_zlabel('Temperatura(C)', fontsize=12)
ax.set_title('                    ', fontsize=50)

# Ajustes para mejorar la visualizacion
ax.view_init(elev=30, azim=45)  # Cambiar la elevacion y el angulo de vision
fig.colorbar(surf, shrink=0.5, aspect=8)  # Anadir barra de color


plt.show()
```

Código 2: Programa que obtiene la grafica de la funcion analitica de la difusion de calor . NOMBRE : "Funcion de calor 2.py"

A.3. Solución numérica a la ecuación de Poisson mediante método Gauss-Seidel

```python
# -*- coding: utf-8 -*-
"""
Created on Fri May  3 15:45:37 2024

@author: dangv
"""
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import math
h=3
Lx = h*np.pi
Ly=h*np.pi
Nx = 1000 # Numero de particiones del eje x
Ny = 1000 # numero de particiones  del eje y
err=0.01
Z=60 #iteraciones de Gauss Seidel
p=0
def rho(x,y):
    k=np.cos(3*x+4*y)-np.cos(5*x-2*y)
    return k

x=np.linspace(-Lx,Lx,Nx)
y=np.linspace(-Ly,Ly,Ny)

phi = np.ones((Nx, Ny))  #Por el metodo de Gauss Seidel , hay que dar un valor "basura" a cada uno de los valores en la matriz al iniciar el proceso


# Iteracion de la EDP
for u in range(Z):

    for j in range(1, Ny-1): #empezamos en 1 por que la ilera de t=0 ya esta inicializada
        for i in range(1, Nx-1): #vamos de 1 a Nx-1 por que tanto el primero (0) como el ultimo ya estan incializados en x

            if (y[j]%(2*np.pi))<err:
                phi[i, j] = phi[i,0]
            elif (x[i]%(2*np.pi))<err:
                phi[i, j] = phi[0,j]
            else:
                phi[i, j] = (rho(x[i],y[j])/4)+(1/4)*(phi[i+1,j]+phi[i-1,j]+phi[i,j+1]+phi[i,j-1])#Por como esta definido phi en la ecuacion 9 , hacemos un cambio de j+1 a j



    p=p+1
    print((p/Z)*100,'%')

y, x = np.meshgrid(y, x)  # Aqui t se transpone con x , nesesario para graficar correctamente la funcion hecha
# Grafico 3D de la superficie
fig = plt.figure(figsize=(12, 12),dpi=300) #dpi=300 aumenta la calidad de  la grafica , y la proporcion de figsize hace que los labels sean de un tamano apropiado
ax = fig.add_subplot(111, projection='3d') #el subpot 111, se refiere a que se proyecta la  grafica 1, de un total de graficas 1x1 , hay mas configuraciones 212, 223, etc
surf=ax.plot_surface(y, x, phi, cmap='viridis', linewidth=0, antialiased=False)
# Anadir etiquetas y titulo
ax.set_xlabel('Distancia X (m)', fontsize=12)
ax.set_ylabel('Distancia Y (m) ', fontsize=12)
ax.set_zlabel('Funcion', fontsize=12)
ax.set_title('
```

```
                  ', fontsize=50)

# Ajustes para mejorar la visualizacion
ax.view_init(elev=45, azim=30)  # Cambiar
    la elevacion y el angulo de vision
fig.colorbar(surf, shrink=0.5, aspect=8)
    # Anadir barra de color


plt.show()
```

Código 3: Programa que obtiene la grafica de la funcion de Poisson por el metodo de Gauss Seidel . NOMBRE : "Gauss Seidel4 Poisson.py"