Simulación de Procesos Físicos

Actividad 3.2. Conducción de calor en un medio heterogéneo.

Alumno: Inserte aquí su nombre.

Suponga que tenemos una barra de longitud L construida con tres capas. Las fronteras entre las capas se denotan con $b_0 \cdots b_3$, donde $b_0 = 0$ y $b_3 = L$. Si las capas están hechas de materiales con propiedades diferentes y además estas se mantienen constantes dentro de la capa, podemos expresar el coeficiente de difusividad térmica como:

$$\alpha(x) = \begin{cases} \alpha_0, & b_0 \leq x < b_1 \\ \alpha_1, & b_1 \leq x < b_2 \\ \alpha_2, & b_2 \leq x \leq b_3 \end{cases}$$

donde $\alpha_0 = 1$, $\alpha_1 = \alpha_0/2$ y $\alpha_2 = \alpha_1/2$. Inicialmente se tiene una temperatura $T = 0$ °C a lo largo de toda la barra. En un tiempo $t = 0$, se calienta el extremo izquierdo ($x = 0$) manteniéndose a una temperatura de $T = 100$ °C y en el lado derecho se tiene un material aislante.

Instrucciones

- Escriba un código para resolver la ecuación de difusión de calor para la barra compuesta de tres capas. Utilice $b1 = 0.25$ y $b_2 = 0.5$ para la posición de las fronteras entre capas. Se recomienda utilizar un método implícito.
- Realice una gráfica para visualizar los resultados.
- Discuta los resultados.

Código.

```
import scipy.sparse.linalg
import numpy as np
import matplotlib.pyplot as plt


def solver(I, alpha, f, L, Nx, Nt,dx,dt, theta, u_L, u_R):
    """
    Theta = 0: Euler adelantado
    Theta = 1: Euler atrasado
    Theta = 1/2: Crank-Nicolson
    """
    x = np.linspace(0, L, Nx+1)
    T = Nt*dt
    t = np.linspace(0, T, Nt+1)

    print('dt=%g' % dt)
    print('dx=%g' % dx)

    if isinstance(alpha, (float , int )):
        alpha = np.zeros(Nx +1) + alpha
    elif callable(alpha):
        # calculamos alpha con la funcion
        a_0 = np.zeros(x.shape)
        for i in range(Nx +1):
            a_0[i] = alpha(x[i])
        alpha = a_0
    # calculamos mu con alpha
    mu = alpha*(dt/dx **2)

    # condiciones de frontera constantes
    if isinstance(u_L, (float,int)):
        u_L_ = float(u_L)
        u_L = lambda t: u_L_
    if isinstance(u_R, (float,int)):
        u_R_ = float(u_R)
        u_R = lambda t: u_R_

    # Funciones para las CI y el forzamiento
    if f is None or f == 0:
        f = lambda x, t: 0
    elif isinstance(f, (float , int )):
        f_ = float(f)
        f = lambda x, t: f_
    if I is None or I == 0:
        I = lambda x : 0



    u   = np.zeros(Nx+1)
    u_n = np.zeros(Nx+1)
```

```
    mul = 0.5*mu*theta
    mur = 0.5*mu*(1-theta)

    diagonal = np.zeros(Nx+1)
    lower    = np.zeros(Nx)
    upper    = np.zeros(Nx)
    b        = np.zeros(Nx+1)
    # Generamos las diagonales
    diagonal[1:-1] = 1 + mul[1: -1]*(alpha[2:] + 2*alpha[1:-1] + alpha[:-2])
    lower[:-1] = -mul[1: -1]*(alpha[1:-1] + alpha[:-2])
    upper[1:]  = -mul[1: -1]*(alpha[2:] + alpha[1:-1])
    # condiciones de frontera
    diagonal[0] = 1
    upper[0] = 0
    diagonal[Nx] = 1
    lower[-1] = 0
    # Armamos la matriz dispersa
    A = scipy.sparse.diags(
        diagonals=[diagonal, lower, upper],
        offsets=[0, -1, 1],
        shape=(Nx+1, Nx+1),
        format='csr')
    #Aplicamos condiciones iniciales
    for i in range(0,Nx+1):
        u_n[i] = I(x[i])
    # generamos un espacio vacio para la solucion
    sol = np.zeros((Nt,len(u)))
    # resolvemos para cada paso de tiempo
    for n in range(0, Nt):
        b[1:-1] = u_n[1:-1] + mur[1: -1]*(
            (alpha[2:] + alpha[1:-1])*(u_n[2:] - u_n[1:-1]) -
            (alpha[1:-1] + alpha[0:-2])*(u_n[1:-1] - u_n[:-2])) + \
            dt*theta*f(x[1:-1], t[n+1]) + \
            dt*(1-theta)*f(x[1:-1], t[n])
        # Condiciones de frontera
        b[0]  = u_L(t[n+1])
        b[-1] = u_R(t[n+1])
        u[:] = scipy.sparse.linalg.spsolve(A, b)
        u_n[:] = u[:]
        sol[n] = u[:]


    return u, x, t, sol, mu

#################################################################33

f = 0
L = 1
Nx = 100
Nt = 200
dt = 0.01
dx = L/Nx
u_L = 100 # lambda t: 0.5*np.cos(t)
u_R = 0 # lambda t: 0.5*np.cos(t)
```

```
I = 0
b = [L/4, L/2]
a0 = 1
a1 = a0/2
a2 = a1/2


alpha = lambda x : L if x < L/4 else L/2 if x < L/2 else L/4


u, x, t, sol, mu = solver(I = I, alpha = alpha, f = f, L = L, Nx = Nx, Nt = Nt, dx = d

plt.contourf(x, t[:-1], sol, cmap='viridis')
escalatemp = plt.colorbar()
escalatemp.set_label('Temperatura ( C )' , fontsize = 10)
plt.axvline( x = b [0] , color = 'k' , linestyle = '--')
plt.axvline( x = b [1] , color = 'k' , linestyle = '--')
plt.ylabel('Tiempo (s)')
plt.xlabel('Distancia (m)')
plt.savefig('Temperatura a traves de una barra.png')

plt.show()
```

Gráficas.

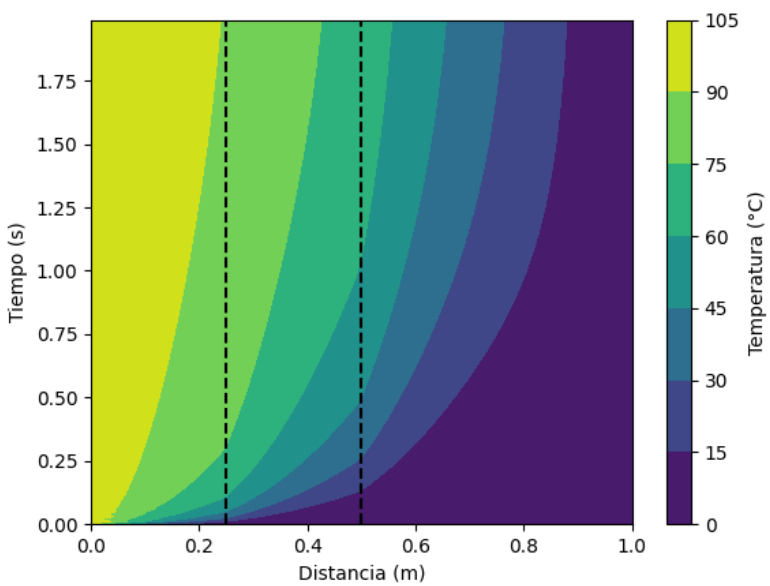

Figura 1: Comportamiento de la temperatura de una barra con materiales distintos

Discusión.

En figura 1 se puede ver como se comporta la temperatura de una barra de 3 distintos materiales con diferentes coeficientes de difusión, las barras verticales representan el punto de transición de un material a otro o también llamada termoclia y se notorio que se cambia la taza de temperatura que se trasmite por la barra. Dado que para cada transición de un material a otro tiene un menor coeficiente de difusión como es esperable la temperatura se trasmite mas lento.