5 Derivadas numéricas: ecuación de calor 1D.

Objetivo. - Aplicar diferencias finitas centradas en la solución numérica de la transferencia de calor en 1D.

Trabajo realizado con el apoyo del Programa UNAM-DGAPA-PAPIME PE101922

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import ipywidgets as widgets
import macti.visual as mvis
from macti.evaluation import *
```

```
quizz = Quizz('1', 'notebooks', 'local')
```

El modelo matemático para la conducción de calor en 1D con condiciones de frontera de tipo Dirichlet, con $\kappa =$ constante se escribe como sigue:

$$
\begin{aligned}
-\kappa \frac{d^2T}{dx^2} &= S \quad \text{para } x \in [0, L] \\
T(x = 0) &= T_A \\
T(x = L) &= T_B
\end{aligned}
$$

La solución analítica de este modelo matemático se escribe como sigue:

$$
T(x) = \left(\frac{T_B - T_A}{L} + \frac{S}{2\kappa}(L - x) \right) x + T_A \tag{1}
$$

5.1 Ejercicio 1.

En la siguiente celda complete el código para implementar la fórmula (1). Posteriormente, define los siguientes valores para calcular la solución exacta:

```
x=np.linspace(0,1,10)
TA = 1.0
TB = 0.0
S = 1.0
L = 1.0
k = 1.0
```

```
# Solucion exacta
def sol_exacta(x, TA, TB, S, L, k):
    """
    Calcula la temperatura usando la fórmula obtenida con Series de Taylor.

    Parameters
    ----------
    x: np.array
    Coordenadas donde se calcula la temperatura.

    TA: float
    Es la condición de frontera a la izquierda.

    TB: float
    Es la condición de frontera a la derecha.

    S: float
    es la fuente.

    L: float
    L es la longitud del dominio.

    k: float
    es la conductividad del material.

    Return
    ------
    al final esta función dibuja la solución.
    """
    ### BEGIN SOLUTION
    return ((TB - TA)/L + S /(2*k) * (L - x) ) * x + TA
    ### END SOLUTION
```

```
x=np.linspace(0,1,10)
TA = 1.0
TB = 0.0
S = 1.0
L = 1.0
k = 1.0

# Cálculo de la solución exacta.
# Te = ...
### BEGIN SOLUTION
Te = sol_exacta(x, TA, TB, S, L, k)

file_answer = FileAnswer()
file_answer.write("1", Te, 'Checa el arreglo secciones')
### END SOLUTION
```

```
print('T exacta = {}',Te)
```

```
T exacta = {} [1.         0.9382716  0.86419753 0.77777778 0.67901235 0.56790123
 0.44444444 0.30864198 0.16049383 0.        ]
```

```
quizz.eval_numeric('1', Te)
```

```
----------------------------------------
1 | Tu resultado es correcto.
----------------------------------------
```

5.2 Ejercicio 2. Error absoluto y error relativo.

El error absoluto y el error relativo se definen como sigue.

$$Error_{absoluto} = ||v_e - v_a||$$

$$Error_{relativo} = \frac{||v_e - v_a||}{||v_e||}$$

donde v_e es el valor exacto y v_a es el valor aproximado.

Implementa las fórmulas del $Error_{absoluto}$ y del $Error_{relativo}$ en la funciones `error_absoluto()` y `error_relativo()`, respectivamente.

```
def error_absoluto(ve, va):
    """
    Calcula el error absoluto entre el valor exacto (ve) y el valor aproximado (v
    """
    ### BEGIN SOLUTION
    return np.linalg.norm(ve - va)
    ### END SOLUTION
```

```
def error_relativo(ve, va):
    """
    Calcula el error relativo entre el valor exacto (ve) y el valor aproximado (v
    """
    # BEGIN SOLUTION
    return np.linalg.norm(ve - va) / np.linalg.norm(ve)
    # END SOLUTION
```

5.3 Ejercicio 3. Solución numérica (interactivo).

Si todo lo realizaste correctamente, ejecuta la siguiente celda para generar un interativo. Mueve los valores de k, S y N y observa lo que sucede.

```
def conduccion_1d(k, S, L, TA, TB, N):
    """
    Calcula la temperatura en 1D mediante diferencias finitas.

    Parameters
    ----------
    L: float
    L es la longitud del dominio.

    k: float
    es la conductividad del material.

    S: float
    es la fuente.

    TA: float
    Es la condición de frontera a la izquierda.

    TB: float
    Es la condición de frontera a la derecha.

    N: int
    Es el número de nodos internos (grados de libertad).

    Return
    ------
    al final esta función dibuja la solución.
    """

    # Cálculo de algunos parámetros numéricos
    h = L / (N+1)
    r = k / h**2

    # Definición de arreglos
    T = np.zeros(N+2)
    b = np.zeros(N)
    A = np.zeros((N,N))

    # Se inicializa todo el arreglo b con S/r
    b[:] = S / r
```

```
    # Condiciones de frontera en el arreglo de la Temperatura.
    T[0] = TA
    T[-1] = TB

    # Se ajusta el vector del lado derecho (RHS) con las condiciones de frontera.
    b[0] += TA
    b[-1] += TB

    # Se calculan las entradas de la matriz del sistema de ecuaciones lineales.
    A[0,0] = 2
    A[0,1] = -1
    for i in range(1,N-1):
        A[i,i] = 2
        A[i,i+1] = -1
        A[i,i-1] = -1
    A[-1,-2] = -1
    A[-1,-1] = 2

    # Se resuelve el sistema lineal.
    T[1:N+1] = np.linalg.solve(A,b)

    # Coordenadas para la solución exacta.
    xe = np.linspace(0,L,100)

    # Coordenadas para la solución numérica.
    xa = np.linspace(0,L,N+2)

    # Se calcula la solución exacta en las coordenadas xe.
    Te = sol_exacta(xe, TA, TB, S, L, k)

    # Se calcula el error absoluto.
    ea = error_absoluto(T, sol_exacta(xa,TA,TB,S,L,k))

    # Se calcula el error relativo
    er = error_relativo(T, sol_exacta(xa,TA,TB,S,L,k))

    # Se imprime el error absoluto y el relativo.
    print('Error absoluto = {:6.5e}, Error relativo = {:6.5e}'.format(ea, er))

    # Se realiza la gráfica de la solución.
    plt.plot(xa, T, 'o-', lw = 0.5, c='k', label = 'Numérica', zorder=5)
    plt.plot(xe, Te, lw=5, c='limegreen', label = 'Exacta')
    plt.xlabel('$x$')
    plt.ylabel('$T$')
    plt.legend()
    plt.grid()
    plt.show()

# Construcción del interactivo.
widgets.interactive(conduccion_1d,
                    k = widgets.FloatSlider(max=1.0, min=0.02, value=0.02, step=0
```

```
                         S = widgets.FloatSlider(max=10.0, min=0.0, value=0, step=1.0)
                         L = widgets.fixed(5.0),
                         TA = widgets.fixed(200),
                         TB = widgets.fixed(1000),
                         N = widgets.IntSlider(max=10, min=4, value=4))
```