Programa: difFin

por: Juan Carlos Jiménez Bedolla

A. Introducción

El código dado define una función llamada difer que calcula diferencias numéricas de un conjunto de datos dado. La función toma cuatro parámetros: x, y, tipo y orden. Los parámetros x e y representan los datos de entrada, mientras que tipo y orden son parámetros opcionales que controlan el tipo y orden de las diferencias a calcular.

La función calcula diferencias utilizando tres métodos diferentes: diferencias hacia atrás, diferencias centradas y diferencias hacia adelante. El método específico utilizado depende de los valores de los parámetros tipo y orden.

La función fue extraida de la siguiente bibliografía:

Métodos numéricos usando Python, *Jiménez J.*, Universidad Nacional Autónoma de México Facultad de Química, 2022 (1era edición)

B. difFin

Entrada:

- x: Representa los valores de la variable independiente.
- y: Representa los valores de la variable dependiente.
- tipo (opcional): Controla el tipo de diferencia a calcular. Puede tomar los siguientes valores:
    - -1: Diferencia hacia atrás.
    - 0: Diferencia centrada.
    - 1: Diferencia hacia adelante. El valor predeterminado es 0 (diferencia centrada).
- orden (opcional): Controla el orden de la diferencia a calcular. Puede tomar los siguientes valores:
    - 1: Primer orden.
    - 2: Segundo orden. El valor predeterminado es 1 (primer orden).

Salida:

- Lista que contiene las diferencias calculadas según los parámetros de entrada. El tamaño de la lista de salida depende del tamaño de los datos de entrada x y y.

C. Programa

- El programa comienza obteniendo el tamaño del arreglo "x" utilizando el método "size" y lo asigna a la variable "n". Luego, utiliza una serie de condicionales para determinar qué tipo de diferencia calcular y realiza los cálculos correspondientes.
- Si el valor de "tipo" es -1, se calculan diferencias hacia atrás. Si el valor de "orden" es 1, se utiliza la fórmula de diferencia hacia atrás de primer orden, que calcula la diferencia entre los elementos consecutivos de "y" dividida por la diferencia entre los elementos consecutivos de "x". Si el valor de "orden" es 2, se utiliza la fórmula de diferencia hacia atrás de segundo orden, que calcula la diferencia entre los elementos de "y" en tres posiciones consecutivas dividida por el cuadrado de la diferencia entre los elementos consecutivos de "x".
- Si el valor de "tipo" es 0, se calculan diferencias centradas. Si el valor de "orden" es 1, se utiliza la fórmula de diferencia centrada de primer orden, que calcula la diferencia entre los elementos de "y" en posiciones adyacentes dividida por el doble de la diferencia entre los elementos consecutivos de "x". Si el valor de "orden" es 2, se utiliza la fórmula de diferencia centrada de segundo orden, que calcula la diferencia entre los elementos de "y" en posiciones adyacentes y separadas por un elemento dividida por el cuadrado de la diferencia entre los elementos consecutivos de "x".
- Si el valor de "tipo" es 1, se calculan diferencias hacia adelante. Si el valor de "orden" es 1, se utiliza la fórmula de diferencia hacia adelante de primer orden, que calcula la diferencia entre los elementos de "y" en posiciones adyacentes dividida por la diferencia entre los elementos de "x" en posiciones adyacentes. Si el valor de "orden" es 2, se utiliza la fórmula de diferencia hacia adelante de segundo orden, que calcula la diferencia entre los elementos de "y" en posiciones separadas por dos elementos dividida por el cuadrado de la diferencia entre los elementos consecutivos de "x".
- Si ninguno de los valores de "tipo" coincide con los casos anteriores, se genera un error indicando que el parámetro "tipo" es incorrecto.
- En cada caso, se utiliza una comprensión de lista para calcular las diferencias correspondientes y se devuelve una lista que contiene las diferencias calculadas según los parámetros de entrada.
- Es importante tener en cuenta que el tamaño de la lista de salida depende del tamaño de los datos de entrada "x" y "y".

D. Código

```
def difer(x, y, tipo = 0, orden = 1):
    # tipo = (-1: Diferecia hacia atras , 0: Centrada , 1: Diferencia hacia adelante )
    # orden = (1: Primer orden , 2: Segundo orden )

    n=x. size #shape
    if tipo == -1: # Diferencias hacia atras
        if orden ==1:
            return [(y[i]-y[i -1]) /(x[i]-x[i -1]) for i in range (1,n)]
        elif orden ==2:
            return [(y[i] -2*y[i -1]+ y[i -2]) /(x[i]-x[i -1])**2 for i in range (2,n)]
        else :
            raise ValueError ( ' Parametro <Orden> incorrecto ')
    elif tipo == 0: # Diferencias centradas
        if orden ==1:
            return [(y[i+1] -y[i-1]) / (2*(x[i]-x[i -1])) for i in range (1,n -1) ]
        elif orden ==2:
            return [(y[i +1] -2* y[i]+y[i -1]) /(x[i]-x[i -1])**2 for i in range (1,n -1) ]
        else :
            raise ValueError ( ' Parametro <Orden> incorrecto ')
    elif tipo ==1: # Diferencias hacia adelante
        if orden ==1:
            return [(y[i+1] -y[i]) /(x[i+1] -x[i]) for i in range (n -1) ]
        elif orden ==2:
            return [(y[i +2] -2* y[i +1]+ y[i]) /(x[i]-x[i -1]) **2 for i in range (0,n -2)
        else :
            raise ValueError ( ' Parametro <Orden> incorrecto ')
    else :
        raise ValueError ( ' Parametro <Tipo > incorrecto ')
```