TP de Diferenciación Numérica

Ejercicio especial Covid-19

Grupo 4:

- Costa Felipe
- Gutierrez Juan Santiago
- Indeau Federico
- Lucero Julieta

Código

Utilizando python 3.12

Se definieron las siguientes funciones,
El código se aloja en github

```
import numpy as np
import math
import pandas as pd

def generar_indices(n: int, traslacion:int = 0) -> np.array:
    """Genera un array simétrico de n/2 enteros alrededor de cero incluyendo el 0\n
    ,y si n es par prefiere a la derecha \n
    Traslada el array antes de devolverlo"""
    array = np.arange(-int(n/2), int(n/2)+1)
    if n%2 == 0:
    array = array[1:]
    return array + traslacion

def gen_matrix(derivada: int = 1, truncamiento: int = 2, traslacion: int = 0) -> np.ndarra
    """Crea una matrix cuadrada para el calculo una n-derivada usando expansiones de Taylo
    puntos = derivada + truncamiento
    indices = generar_indices(puntos, traslacion)

    # Definir una matrix cuadrada vacia
    A = np.ones((puntos,puntos))

    # Los elementos de la matrix se calculan así
    def coef(i:int ,j:int) -> float:
        return math.pow(j,i) / math.factorial(i)

    # Recorrer la matrix y generar cada elemento
    for i in range(puntos):
        for j in indices:
            k = j + sum(1 for i in indices if indices[i] < 0) # Ajusta el indice j, con el
            A[i][k] = round(coef(i=i,j=j), truncamiento + 2)
    # Retornar la matrix
    return A

    def coef_derivada(matrix: np.ndarray[float], derivada:int =1) -> np.ndarray[float]:

    """gen_matrix() -> A \n
    Matrix A para derivadas, aplica las condiciones para la n-derivada\n
    devolviendo los coeficientes X \n
    X -> evaluar_derivada()"""
```

```
    # Define un vector vacio y aplica las condiciones
    v = np.zeros((len(matrix)))
    v[derivada] = 1
    # Resuelve el sistema y lo retorna
    return np.linalg.solve(a=matrix, b=v)

def evaluar_derivada(X:np.ndarray,f:np.ufunc ,x:float=0.0, dx: float = 0.1, traslacion:in
    """coef_derivada() -> X \n
    Evalua la derivada en un punto en base a los coeficientes X y a cual derivada se diri
    -> f'(x)"""
    df = 0
    # Usa los coeficientes de X y los índices de los puntos para calcular df, recorriendo
    for i,j in zip(X, generar_indices(len(X), traslacion)):
        df += i * f(x + j * dx )
    return df / dx

def derivar(x:float,f:np.ufunc,dx:float,derivada: int =1, orden:int = 2, traslacion:int=0
    """Deriva una funcion númericamente, usando expansiones de Taylor y orden de Landau"""
    # Calcular la matrix, por defecto da la 1er derivada
    A = gen_matrix(derivada=derivada,truncamiento=orden,traslacion= t)

    # Calcular los coeficientes
    X = coef_derivada(matrix=A,derivada=derivada)

    # Evaluar la derivada
    df = evaluar_derivada(X=X, f=f, x=x, dx=dx,traslacion=traslacion)
    return df
```

Datos y guía

Con las funciones anteriormente definidas se realizan los ejercicios en la guía de ejercicios:

- *Ejercicio especial Derivacion 2021.pdf*

Con datos extraídos de https://www.argentina.gob.ar/coronavirus/informe-diario en uno de los reportes diarios.

- En partícular: "*Reporte diario COVID-19.csv*"

Resolución

- La resolución consta de el código realizado y la salida. Para los ejercicios 1, 2, 3, 5 se utiliza el siguiente código.
- Se presentan los 5 primeros datos.

```
# Variables {#variables }
traslacion = 0
derivada = 1
orden = 2
pasos = 1
DF = pd.read_csv("Data/Reporte diario COVID-19.csv")
def contagios(dia):
    return DF["acumulado"][dia-1]
L = []
for dia in DF["dia"]:
    traslacion = 0
    while True:
        try:
            df = derivar(x=dia, f=contagios, dx=pasos, derivada=derivada, orden=orden, tra
            break
        except:
            if dia < len(DF)//2: traslacion+=1
            else: traslacion-=1
    L += [df]
DF = DF.assign(df1_2_Orden_1_p = L)
print(DF.head())
```

1.

Código

- Traslacion = 0
- Derivada = 1
- Orden = 2
- Pasos = 1

Salida

```
   dia  nuevo  acumulado  df1_2_Orden_2_p
0    1      1          1             -3.5
1    2      1          2              5.5
2    3      3         12              7.5
3    4      5         17              3.5
4    5      2         19              2.0
```

2.

Código

pasos 1 -> 2

Salida

```
   dia  nuevo  acumulado  df1_2_Orden_2_p
0    1      1          1             6.50
1    2      1          2            10.25
2    3      3         12             4.50
3    4      5         17             4.75
4    5      2         19             4.75
```

3.

Código

El método de extrapolación de Richarson, utiliza un orden de 4. Entonces:

- orden 2 -> 4
- pasos 2 -> 1

Salida

```
   dia  nuevo  acumulado  df1_4_Orden_1_p
0    1      1          1       -12.166585
1    2      1          2         9.166679
2    3      3         12         8.500003
3    4      5         17         3.083332
4    5      2         19         1.083331
```

4.

Código

```
derivada = 1
traslacion = 0
orden = 4
A = gen_matrix(derivada=derivada, truncamiento=orden, traslacion=traslacion)
X = coef_derivada(A, derivada=derivada)
X = np.round(X,5)
print(f"({X[0]}*f-2 + {X[1]}*f-1+ {X[2]}*f0 + {X[3]}*f1+ {X[4]}*f2)/dx")
```

Salida

```
(0.08333*f-2 + -0.66667*f-1+ -0.0*f0 + 0.66667*f1+ -0.08333*f2)/dx
```