Ayudantía
SCT - Semana - 05

Repaso para Certamen 1

1. Se tiene la siguiente función igual a una sumatoria infinita:

$$S(x) = \sum_{k=1}^{\infty} \frac{\cos(kx)}{k}$$

Esta función no se puede evaluar en un múltiplo de 2π, pues cada uno de los $\cos(kx)$ internos se evaluaría a 1, obteniendo la famosa serie armónica que diverge a infinito: $S(2m\pi) = \sum_{k=1}^{\infty} \frac{1}{k} = +\infty$.

Nos interesa seguir analizando esta función, pero no podemos sumar infinitos términos en un computador. En su lugar, podemos definir una variante $S_n(x)$ que considera solo los primeros n términos:

$$S_n(x) = \sum_{k=1}^{n} \frac{\cos(kx)}{k}$$

a) Construya un algoritmo que, dado un entero j, realice j iteraciones del **método de Newton** para aproximar un **punto crítico** de la función $S_n(x)$, es decir, un valor r tal que $S_n'(r) = 0$.

b) Implemente el algoritmo anterior en Python, mediante una función que reciba dos parámetros: un entero n indicando la cantidad de términos en la sumatoria $S_n(x)$, y un entero j indicando la cantidad de iteraciones a realizar del método de Newton. La función que implemente no solo debe retornar el punto crítico r, sino también una estimación de la **tasa lineal de convergencia** S y la **tasa cuadrática de convergencia** M calculadas sobre las j iteraciones del método de Newton.

Use la librería NumPy y sus capacidades de vectorización donde sea adecuado. En particular, puede usar:

- `np.arange(start, stop)` para generar un arreglo `[start, start+1, start+2, ..., stop-2, stop-1]`;
- `np.sum` para calcular la suma de todos los valores de un arreglo: `np.sum([1, 2, 4]) = 7`; y
- `np.sin` y `np.cos` para calcular el seno/coseno de un valor o de un arreglo de valores. En este último caso, un ejemplo es `np.sin([0, 1, 2]) = [0.0, 0.84147098, 0.90929743]`.

La función debe llevar la siguiente firma:

```
'''
input:
n : (int64) Upper limit of the sum S_n(x).
j : (int64) Number of iterations to be used in Newton's method.
output:
r : (double) Root obtained by Newton's method.
S : (double) Estimated linear rate of convergence.
M : (double) Estimated quadratic rate of convergence.
'''
def find_critical_point_of_Sn(n, j):
    # Your code goes here
    return r, S, M
```

2. Revisa y resuelve, en la guía de ejercicios para C1, el problema 2.6: "Un triángulo y dos preguntas".