Sergio Abreu García y Diego Martínez Simarro

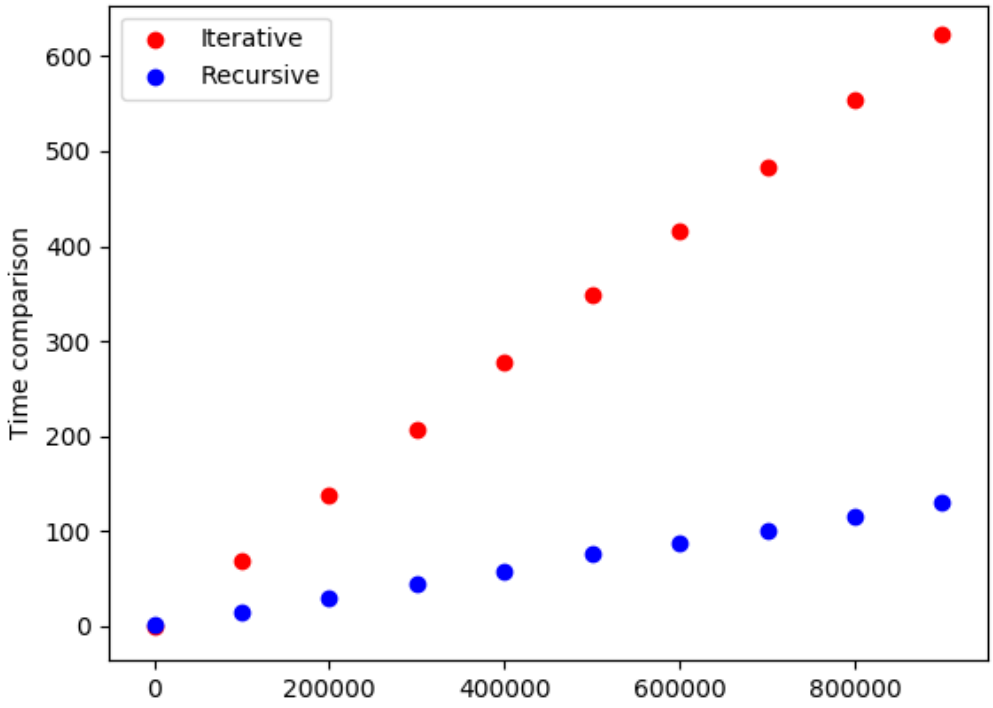
AAMD - Práctica 0

Resumen

Cálculo de la derivada de forma aproximada mediante el método de Monte Carlo.

Implementación de forma iterativa y vectorial.

Como se esperaba, la implementación vectorial es más rápida que la factorial, como podemos observar en esta gráfica.



Código

```
# Diego Martínez Simarro y Sergio Abreu García

import math
import random
import scipy.integrate as integrate
import numpy as np
import time
import matplotlib.pyplot as plt

def integra_mc_iterativa(fun, a, b, num_puntos=1000):

    #VARIABLE PARA CONTROLAR EL TIEMPO
    tic = time.time()

    #HALLAMOS M
    M = 0
    for i in np.arange(a, b, (b-a)/num_puntos):
        aux = fun(i)
        if aux > M :
            M = aux
    print("\n__Integral iterativa__")
    print("Máximo: ", M)

    #QUEDA DEFINIDO EL CUADRADO ENTRE A-B Y 0-M
    #LANZAMOS PUNTOS ALEATORIOS DENTRO DEL CUADRADO

    nDebajo = 0

    for i in range(num_puntos):
        x = random.uniform(a, b)
        y = random.uniform(0, M)

        if fun(x)>y :
            nDebajo+=1

    print("N Debajo: ", nDebajo)
    print("N Totales: ", num_puntos)

    I = (nDebajo/num_puntos)*(b-a)*M
    print ("I montecarlo: ", I)

    toc = time.time()
    print("Tiempo: ", 1000*(toc - tic))

    return 1000*(toc - tic)

def integra_mc_vectores(fun, a, b, num_puntos=1000):
```

```
    #VARIABLE PARA CONTROLAR EL TIEMPO
    tic = time.time()

    #CREAMOS VECTORES CON PUNTOS
    vectorX = np.random.uniform(low = a, high = b, size =(num_puntos))

    #HALLAMOS M
    vectorizedFun = np.vectorize(fun)
    funResults = vectorizedFun(vectorX)
    M = funResults.max()
    vectorY = np.random.uniform(low = 0, high = M, size =(num_puntos))

    print("\n__Integral vectorial__")
    print("Máximo: ", M)

    #CONTAMOS LOS PUNTOS DEBAJO
    nDebajo = np.greater(funResults, vectorY).sum()

    print("N Debajo: ", nDebajo)
    print("N Totales: ", num_puntos)

    #HALLAMOS I CON LOS DATOS OBTENIDOS
    I = (nDebajo/num_puntos)*(b-a)*M
    print ("I montecarlo: ", I)

    toc = time.time()
    print("Tiempo: ", 1000*(toc - tic))

    return 1000*(toc - tic)

#LLAMAMOS VARIAS VECES A AMBAS FUNCIONES Y RELLENAMOS UNA GRAFICA CON SUS
TIEMPOS
iterativeArray = []
vectorialArray = []
iValues = []

for i in range(1, 1000000, 100000):
    iterativeArray.append(integra_mc_iterativa(math.sin, 0, math.pi, i))
    vectorialArray.append(integra_mc_vectores(math.sin, 0, math.pi, i))
    iValues.append(i)

plt.figure()
plt.scatter(iValues, iterativeArray, color='red', label="Iterative")
plt.scatter(iValues, vectorialArray, color='blue', label="Recursive")
plt.ylabel('Time comparison')
plt.legend()
plt.savefig("Grafica.png")
plt.show()
print("\nI real: ", integrate.quad(math.sin, 0, math.pi)[0], "\n")
```