Javier Cordero Calvo y Juan Ruiz Jiménez

Práctica 0 – Comparación de tiempos al hacer integrales entre Numpy y Python

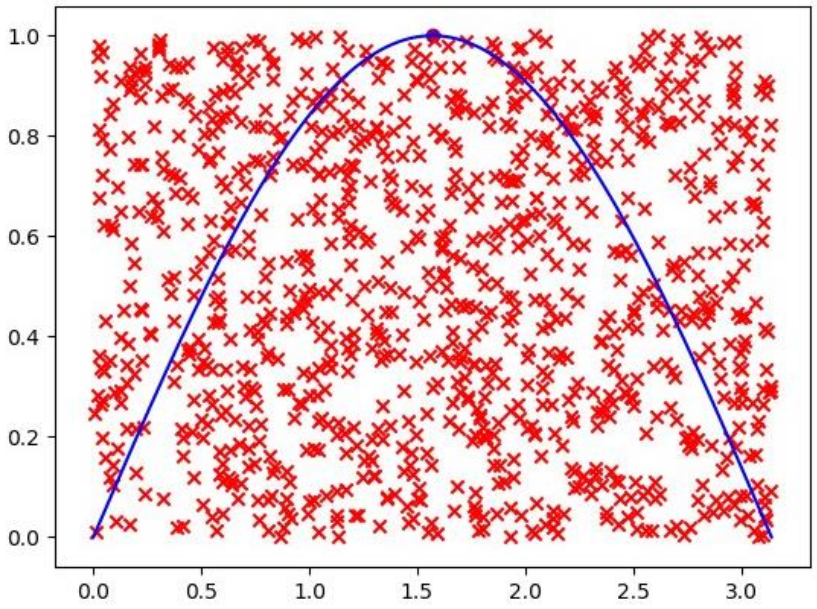
Código:

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

def comparaTiempos(fun, a, b, num_puntos = 1000):
    tiempo_lento = []
    tiempo_rapido = []
    xVector = []

    for i in range (1, num_puntos, 10000):
        tiempo_lento += [ integra_mc(fun, a, b, i + 1, puntosLento)]
        tiempo_rapido += [ integra_mc(fun, a, b, i + 1, puntosRapido)]
        xVector += [i]

    plt.figure()
    plt.plot(xVector, tiempo_lento, c='red', label='tiempoLento')
    plt.plot(xVector, tiempo_rapido, c='blue', label='tiempoRapido')
    save('compara_tiempos.png')

    show()


def save(name):
    plt.savefig(name)

def integra_mc(fun, a, b, num_puntos, f):

    X = np.linspace(a, b, num_puntos)
    Y = fun(X)
    indexM = np.argmax(Y)
    Mx = X[indexM]
    My = max(Y)

    return f(a, b, My, num_puntos, fun)




def puntosLento(a, b, My, num_puntos, fun):
    tic = time.process_time()
```

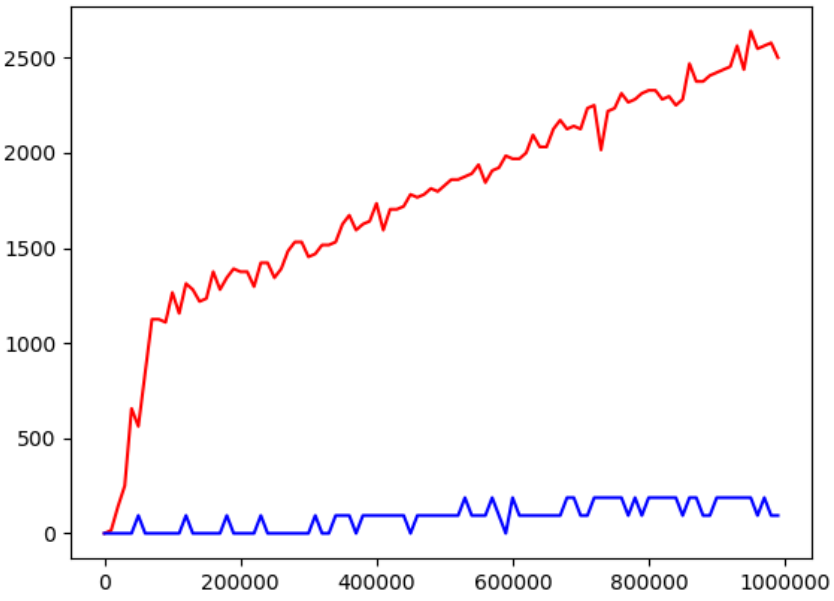
```
    count = 0
    for i in range(num_puntos):
        Xr = np.random.uniform(a, b)
        Yr = np.random.uniform(0, My)

        Y = fun(Xr)

        if Yr < Y:
            count += 1

    integral = count / num_puntos * (b - a)  * My

    toc = time.process_time()
    return 1000 * (toc - tic)

def puntosRapido(a, b, My, num_puntos, fun):
    tic = time.process_time()
    Xr = np.random.uniform(a,b,num_puntos)
    Yr = np.random.uniform(0,My,num_puntos)

    aux = Yr[Yr<fun(Xr)]

    integral = aux.size / num_puntos * (b - a) * My

    #print(integral)
    #print(integrate.quad(fun, a, b)[0])
    #print('---------------------')

    toc = time.process_time()
    return 1000 * (toc - tic)

def pintaPuntosAleatorios(X, Y, name):
     plt.scatter(X, Y, c='red', marker='x')
     plt.legend()
     plt.savefig(name)

def pintaFuncion(X, Y):
    plt.figure()
    plt.plot(X, Y, c='blue', label='Funcion')

def show():
    plt.legend()
    plt.show()

def pintaPuntoMaximo(Mx, My):
    plt.scatter(Mx, My, c='purple', label='Maximo')

comparaTiempos(np.sin, 0, np.pi, 1000000)
```

Imágenes:

Función y puntos generados aleatoriamente en el intervalo x, y de la función:



Comparación de velocidades entre Numpy (azul) y Python (rojo)



Conclusiones:

Numpy es más fácil de utilizar a la hora de realizar trabajos con arrays y la velocidad que proporciona es mucho mayor a la propia de los arrays de Python.